The Effectiveness of Curvature-Based Rewiring and the Role of Hyperparameters in GNNs Revisited

Floriano Tori
Data Analytics Laboratory
Vrije Universiteit Brussel
Floriano.Tori@vub.be

Vincent Holst
Data Analytics Laboratory
Vrije Universiteit Brussel
Vincent.Thorge.Holst@vub.be

Vincent Ginis
Data Analytics Laboratory
Vrije Universiteit Brussel
SEAS, Harvard University
Vincent.Ginis@vub.be

Abstract

Message passing is the dominant paradigm in Graph Neural Networks. Its efficiency, however, can be limited by the graph's topology, as information is lost during propagation due to being *oversquashed* when travelling through bottlenecks. Recent efforts have therefore focused on rewiring techniques, which disconnect the input graph originating from the data and the computational graph. A prominent approach for this is to use discrete graph curvature measures to identify and rewire around bottlenecks. In this work, we reevaluate the performance gains that curvature-based rewiring brings to non-synthetic datasets. We show that edges selected during rewiring do not satisfy theoretical criteria identifying bottlenecks, implying that they do not necessarily oversquash information. Subsequently, we demonstrate that reported accuracies after rewiring on these datasets are outliers originating from sweeps of training and rewiring hyperparameters, instead of consistent performance gains. In conclusion, our analysis nuances the effectiveness of curvature-based rewiring in real-world datasets and brings a new perspective on the methods to evaluate GNN accuracy improvements.

1 Introduction

The field of Graph Neural Networks (GNNs) has undergone rapid development over the past few years. Both in terms of architecture variations [1–4] as theoretical understanding [5–8]. Due to their large flexibility GNNs have been applied in a variety of domains, from physical sciences to knowledge graphs or social sciences [9, 10]. The basis of the *message passing* paradigm of GNNs [11] rests on the idea of information diffusion where messages, namely the node's feature vector, are propagated along the edges of the graph to their neighbours.

This approach has been shown to be very successful, as it combines both topological information (the graph) and specific node information (feature vectors) for predictions. However, this paradigm can also suffer from drawbacks. Recently, a lot of attention has been drawn to the problem of *oversquashing* [12] where structural properties of the graph, called bottlenecks, cause a loss of information as the messages passing through them get too compressed. Research efforts have therefore focused on understanding this phenomenon [13, 14] as well as on ways to alleviate it. The most pragmatic approach consists of rewiring, i.e., a targeted addition or removal of edges.

Contributions. In this work, we investigate the role of rewiring to improve the performance of GNNs. Discrete notions of curvatures on graphs can be used to detect the location of bottlenecks, allowing for the development of algorithmic rewiring methods such as Stochastic Discrete Ricci Flow [15]. While work on synthetic datasets does indicate the occurrence of oversquashing [13, 14], we here reevaluate performance gains of curvature-based rewiring, specifically Stochastic Discrete Ricci Flow (SDRF) [15], on non-synthetic, benchmark datasets. First, we show experimentally that theoretical conditions from Topping *et al.* [15] to identify edges that cause bottlenecks are not met in

F. Tori et al., The Effectiveness of Curvature-Based Rewiring and the Role of Hyperparameters in GNNs Revisited (Extended Abstract). Presented at the Third Learning on Graphs Conference (LoG 2024), Virtual Event, November 26–29, 2024.

most cases when rewiring. Second, with this nuanced perspective on the theory underlying rewiring algorithms, we reevaluate the performance gains they bring. We compare accuracies obtained when training on the original graphs as well as graphs rewired according to different curvature measures. Our results show limited measurable improvements when considering results from hyperparameter sweeps globally. We find that previously reported results correspond to hyperparameter tuning outliers and that rewiring does not bring a systematic performance improvement on all datasets. Consequently, our work highlights the importance of bridging theory and experiment beyond synthetic datasets while reevaluting the current effectiveness of rewiring.

2 Preliminaries

Graph Neural Networks. Given a graph $G = (V, E)$ with a set of nodes V, which are described by a feature vector $\mathbf{x}_i \in \mathbb{R}^{n_0}$, and a set of edges $E \subset V \times V$ we write the adjacency matrix as A. The representation of node i at layer l is $\mathbf{h}_i^{(l)}$ ($\mathbf{h}_i^0 = \mathbf{x_i}$). Given layer dependent, differentiable functions $\phi^l : \mathbb{R}^{n_l} \times \mathbb{R}^{n_l'} \to \mathbb{R}^{n_{l+1}}$ and $\psi^l : \mathbb{R}^{n_l} \times \mathbb{R}^{n_l} \to \mathbb{R}^{n_l'}$, we write the message passing function as

$$\mathbf{h}_i^{(l+1)} = \phi^l\left(\mathbf{h}_i^{(l)}, \sum_{j=1}^{n} \hat{A}_{ij}\psi^l\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right). \tag{1}$$

Here, $\hat{A}$ denotes A augmented by self-loops and then normalised by $\tilde{D} = D + I$, where D denotes the diagonal degree matrix. More precisely, we have $\hat{A} = \tilde{D}^{-\frac{1}{2}} \cdot \tilde{A} \cdot \tilde{D}^{-\frac{1}{2}}$.

Curvature and Oversquashing. Applying discrete curvature notions to detect local bottlenecks in graphs originally stems from differential geometry. Ricci curvature describes whether two geodesics which start close to each other, either diverge (negative curvature), stay parallel (zero curvature) or converge (positive curvature). Prominent examples are respectively the hyperbolic space, Euclidean space and the sphere. The graph analogues for these spaces are trees, four-cycles and triangles whose occurrence around a given edge is captured by the discrete curvature notions. The most fundamental result connecting curvature with oversquashing is given in Topping *et al.* [15] where theorem 4 identifies edges with a very negative *Balanced Forman curvature (BFc)* [15] as sources of distorted information for a large set of nodes in the local neighbourhood of a given edge $i \sim j$.

Theorem 4 [15]. Consider a MPNN as in Equation 1. Let $i \sim j$ with $d_i \leq d_j$ and assume that:

1. $|\nabla\phi_l| \leq \alpha$ and $|\nabla\psi_l| \leq \beta$ with $L \geq 2$ the depth of the MPNN.
2. There exists $\delta > 0$ such that[1] $\delta < \frac{1}{\sqrt{(d_i \vee d_j)}}$; $\delta < \frac{1}{\gamma_{max}}$ for which $\mathrm{BFc}(i, j) \leq -2 + \delta$.

Then there exists $Q_j \subset S_2(i)$ satisfying $|Q_j| > \frac{1}{\delta}$ and for $0 \leq l_0 \leq L - 2$ we have

$$\frac{1}{|Q_j|}\sum_{k \in Q_j}\left|\frac{\partial h_k^{(l_0+2)}}{\partial h_i^{l_0}}\right| < (\alpha\beta)^2\delta^{\frac{1}{4}}. \tag{2}$$

Theorem 4 above gives a reason to rewire the graph around negatively curved edges as it can increase the δ and therefore soften the bound in Eq. (2). However, we will now look at how well Theorem 4 can be applied to benchmark datasets by seeing if edges selected during rewiring are contributing to the bottleneck as indicated by condition 2 (and a less stringent but sufficient version we call condition 2b). In the second phase, we take a closer look at the performances that different curvature measures deliver. We then take a look at the distribution of accuracies obtained when sweeping parameters.

3 Results

3.1 Benchmark datasets have a lack of sufficiently negatively curved edges

In our experiments, we make exclusive use of the Stochastic Discrete Ricci Flow (SDRF) [15] algorithm for rewiring, which works in two steps. First, it selects the most negatively curved edge

[1] We write d_i for the degree of node i. The common neighbours of node i and j are $\sharp_\triangle(i, j)$. γ_{max} is the maximum number of 4-cycles containing edge $i \sim j$ without diagonals inside. We denote by $x \vee y \doteq \max(x, y)$.

based on the curvature measure. Around this edge, all edges that can lead to three-cycles and four-cycles are considered, and for each candidate edge, the potential improvement of the curvature is computed. The edge to be added is then selected in a stochastic way, regulated by the temperature parameter τ, where the probability is determined by the improvement the edge brings to the curvature of the original edge.

Finally, SDRF also allows, at each iteration, the removal of (very) positively curved edges, determined by the threshold curvature value C^+. Although different algorithms have been proposed (for example BORF where edges are added in batches), we only work with SDRF due to the simplicity of its approach, allowing us to look at the impact of the added edges more clearly.

Table 1: For each edge selected by SDRF, we verify if this edge satisfies condition 2 and its softer variant, condition 2b (Eq. (3)), of Theorem 4.

Dataset	Edges rewired	Cond. 2 (%)	Cond. 2b (%)
Texas	89	0 (0%)	6 (6.7%)
Cornell	126	0 (0%)	15 (11.90%)
Wisconsin	136	0 (0%)	11 (8.09%)
Chameleon	2441	4 (0.16%)	141 (5.78%)
Actor	1000	11 (1.1%)	237 (23.70%)
Squirrel	787	0 (0%)	34 (4.32%)
Cora	100	0 (0%)	68 (68.0%)
Citeseer	84	0 (0%)	24 (28.57%)
Pubmed	166	25 (15.06%)	116 (69.88%)
MUTAG	3497	0 (0%)	1228 (35.16%)
PROTEINS	50936	0 (0%)	5944(11.67%)

Looking at the condition from Theorem 4, we experimentally check if the edges in benchmark datasets satisfy the necessary condition 2, which identifies the edge as an 'oversquashing' edge. For each edge selected to be rewired we compute the upper threshold $\delta_{max}(i,j) = BFc(i,j) + 2$. For these edges, we then verify if δ_{max} determined by the curvature is compatible with the condition. From our results in Table 1, we see that these conditions on δ are seldom satisfied by the graphs in the datasets. However, upon examining the derivation of the theorem, we find that the degree-based condition is too stringent. The inequality $0 < \delta < 1/\sqrt{(d_i \vee d_j)}$ is solely used to guarantee that $\delta \leq 1/\sharp_{\triangle}$, and it is this condition that is subsequently used in the proof. In the second column of Table 1, we display the number of edges that satisfy the actual modified condition required, namely

$$\delta \leq \frac{1}{\sharp_{\triangle}} \;\; \& \;\; \delta < \frac{1}{\gamma_{max}} \;\; \text{(Condition 2b)}\,. \tag{3}$$

As this bound is looser, we see that more selected edges satisfy condition 2b, especially when looking at the citation graphs. However, these numbers still imply that most edges selected do not satisfy the conditions for Theorem 4, limiting their interpretation as bottlenecks during message passing.

Finally, we can analyse the temporal aspect of edges being selected. Figure 1 shows in which step of SDRF (in % of maximum number of iterations) the edges that do and do not satisfy condition 2b are selected to be rewired. It also shows that edges that do satisfy the condition are sometimes close to the upper bound of $1/\sharp_{\triangle}$ again reducing their interpretation as bottlenecks, as the δ bound on the Jacobian of the features is therefore looser. This temporal information for both types of edges tells us that this phenomenon occurs at any time, indicating that this is not a saturation-type effect.

3.2 Hyperparameter dependency of rewiring

Graph rewiring depends on several hyperparameters. We have both training and model hyperparameters as well as the rewiring hyperparameters from the SDRF algorithm. While hyperparameter tuning is an important aspect of optimising models, we argue here that top results in accuracy should not be the main judge for a new technique's performance, but one should consider the overall improvement over a wide array of hyperparameter choices. By looking at the distribution obtained when performing a hyperparameter sweep on various graph benchmark datasets using different curvature measures[2], instead of solely top performers, we aim to explain the gap between previously reported accuracies due to rewiring and the experimental results we find on the lack of edges being identified as bottlenecks. Complete details of this experiment can be found in Appendix C.

Results from hyperparameter sweeps. In Appendix C we display the full distributions originating from the hyperparameter sweep in Figure 3 together with the top 10% of the reported accuracies given by hyperparameters, which is presented in Table 2. Overall we find that no curvature measure consistently shifts the distribution of the mean test accuracy away from the None distribution (meaning no rewiring) over all datasets.

[2]In this work we analyse three main branches of discrete curvatures, with variations within the *Balanced Forman curvature (BFc)* [15], the *Jost and Liu curvature (JLc)* [16] and the *Augmented Forman Curvature (AFc)* [17]. Further details on the curvature notions can be found in appendix A.

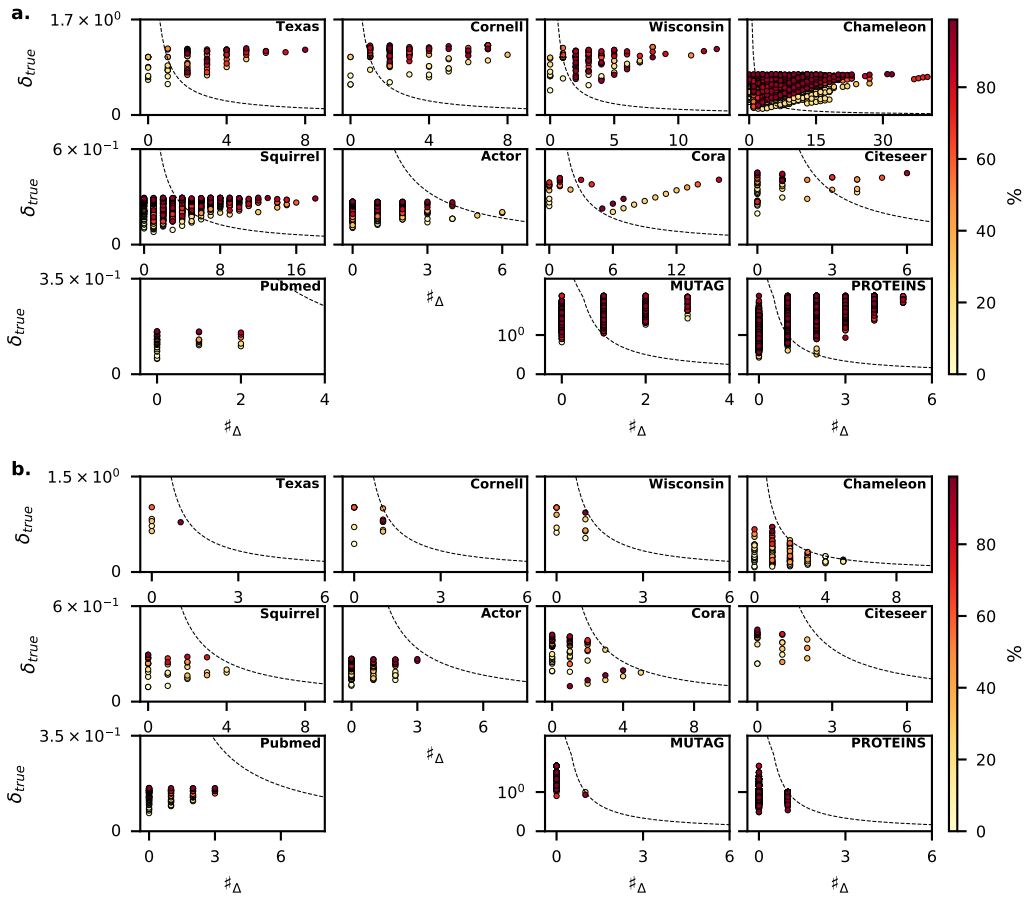

Figure 1: A visualisation of the edges selected during the SDRF rewiring algorithm. Dotted line shows $y = 1/\sharp_\triangle$. **a.** The panels display edges that do **not** satisfy condition 2b, both due to $\delta > 1/\sharp_\triangle$ (if the edge is situated above the dotted line) or $\delta > 1/\gamma_{max}$ (if the edge is situated below the dotted line). **b.** The panels show the opposite, namely the edges that satisfy condition 2b. This means that the edge is situated below the dotted line and $\delta < 1/\gamma_{max}$. The color code of the edges indicates at which step of the rewiring process (in %) the edge is selected.

4 Conclusion

In our work, we have taken a closer look at the effectiveness of graph-rewiring on benchmark graph datasets. Our results show that the conditions for oversquashing based on theorems proposed from the literature are not always met when considering these datasets. This could imply that the edges selected during rewire do not necessarily cause oversquashing during message-passing and that severe bottlenecks are in fact not present in these datasets. Although one might interpret oversquashing as a continuous phenomenon, these results suggest that it is limited to specific graph topologies.

Our analysis is further substantiated by examining the role of hyperparameter sweeping when benchmarking curvature-based graph rewiring methods. We found that most performance gain is due to finding an optimal hyperparameter configuration rather than a structural shift in the performance.

The message of our work is twofold. First, it serves as a re-evaluation of curvature-based rewiring methods, and can importantly influence further development in GNN as we argue that theoretical results and experiments should be more closely checked. Future work could explore the possibility of theorem-aware rewiring. Secondly, we argue that future rewiring methods should take into consideration the dependency of their method on hyperparameters, both GNN and rewiring when evaluating their performance.

Acknowledgment

The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government. Work at VUB was partially supported by the Research Foundation Flanders under Grants No. G032822N and No. G0K9322N.

References

1. Wu, F. *et al. Simplifying graph convolutional networks* in *International conference on machine learning* (2019), 6861–6871. 1
2. Kipf, T. N. & Welling, M. *Semi-Supervised Classification with Graph Convolutional Networks* in *International Conference on Learning Representations* (2016). 1, 9
3. Veličković, P. *et al. Graph Attention Networks* in *International Conference on Learning Representations* (2018). 1
4. Hamilton, W., Ying, Z. & Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems* **30** (2017). 1
5. Xu, K., Hu, W., Leskovec, J. & Jegelka, S. *How Powerful are Graph Neural Networks?* in *International Conference on Learning Representations* (2018). 1
6. Bronstein, M. M., Bruna, J., Cohen, T. & Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478* (2021). 1
7. Bodnar, C. *et al.* Weisfeiler and lehman go cellular: Cw networks. *Advances in neural information processing systems* **34,** 2625–2640 (2021). 1
8. Bodnar, C. *et al. Weisfeiler and lehman go topological: Message passing simplicial networks* in *International Conference on Machine Learning* (2021), 1026–1037. 1
9. Zhou, J. *et al.* Graph neural networks: A review of methods and applications. *AI open* **1,** 57–81 (2020). 1
10. Wu, Z. *et al.* A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* **32,** 4–24 (2021). 1
11. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. *Neural message passing for quantum chemistry* in *International conference on machine learning* (2017), 1263–1272. 1
12. Alon, U. & Yahav, E. *On the Bottleneck of Graph Neural Networks and its Practical Implications* in *International Conference on Learning Representations* (2021). https://openreview.net/forum?id=i80OPhOCVH2. 1
13. Di Giovanni, F. *et al. On over-squashing in message passing neural networks: The impact of width, depth, and topology* in *International Conference on Machine Learning* (2023), 7865–7885. 1
14. Black, M., Wan, Z., Nayyeri, A. & Wang, Y. *Understanding oversquashing in gnns through the lens of effective resistance* in *International Conference on Machine Learning* (2023), 2528–2547. 1
15. Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X. & Bronstein, M. M. *Understanding over-squashing and bottlenecks on graphs via curvature* in *International Conference on Learning Representations* (2021). 1–3, 7–9
16. Jost, J. & Liu, S. Ollivier's Ricci curvature, local clustering and curvature-dimension inequalities on graphs. *Discrete & Computational Geometry* **51,** 300–322 (2014). 3, 7
17. Fesser, L. & Weber, M. *Mitigating Over-smoothing and Over-squashing using Augmentations of Forman-Ricci Curvature* in *The Second Learning on Graphs Conference* (2023). 3, 7
18. Forman. Bochner's method for cell complexes and combinatorial Ricci curvature. *Discrete & Computational Geometry* **29,** 323–374 (2003). 7
19. Sreejith, R., Mohanraj, K., Jost, J., Saucan, E. & Samal, A. Forman curvature for complex networks. *Journal of Statistical Mechanics: Theory and Experiment* **2016,** 063206 (2016). 7
20. Weber, M., Saucan, E. & Jost, J. Coarse geometry of evolving networks. *Journal of complex networks* **6,** 706–732 (2018). 7
21. Samal, A. *et al.* Comparative analysis of two discretizations of Ricci curvature for complex networks. *Scientific reports* **8,** 8650 (2018). 7

22. University, C. M. *WebKB* http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwk. 8
23. Rozemberczki, B., Allen, C. & Sarkar, R. Multi-scale attributed node embedding. *Journal of Complex Networks* **9,** cnab014 (2021). 8
24. Tang, J., Sun, J., Wang, C. & Yang, Z. *Social influence analysis in large-scale networks* in *Knowledge Discovery and Data Mining* (2009). https://api.semanticscholar.org/CorpusID:4931148. 8
25. McCallum, A. K., Nigam, K., Rennie, J. & Seymore, K. Automating the construction of internet portals with machine learning. *Information Retrieval* **3,** 127–163 (2000). 8
26. Sen, P. *et al.* Collective classification in network data. *AI magazine* **29,** 93–93 (2008). 8
27. Namata, G., London, B., Getoor, L., Huang, B. & Edu, U. *Query-driven active surveying for collective classification* in *10th international workshop on mining and learning with graphs* **8** (2012), 1. 8
28. Morris, C. *et al.* Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663* (2020). 8
29. Pei, H., Wei, B., Chang, K. C.-C., Lei, Y. & Yang, B. *Geom-GCN: Geometric Graph Convolutional Networks* in *International Conference on Learning Representations* (2019). 9
30. Gasteiger, J., Weißenberger, S. & Günnemann, S. Diffusion improves graph learning. *Advances in neural information processing systems* **32** (2019). 9
31. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 9

A Curvature definitions

Curvature notions on graphs depend on topological aspects of the graph with the needed ingredients being the following. For a simple, undirected graph we consider an edge $i \sim j$. We denote by d_i the degree of node i and by d_j the degree of node j. The common neighbours of node i and node j are denoted by $\sharp_{\triangle}(i,j)$. They correspond to the triangles located at edge $i \sim j$. The neighbours of i (resp. j) that form a four-cycle based at $i \sim j$ without diagonals inside are denoted by $\sharp^i_{\square}$ (resp. $\sharp^j_{\square}$) and the maximum number of four-cycles without diagonals inside that share a common node is denoted by γ_{max}. We denote by $x \vee y \doteq \max(x,\, y)$ (resp. $x \wedge y \doteq \min(x,\, y)$) the maximum (resp. minimum) of two real numbers.

In this paper we analyse three main branches of discrete curvatures, with variations within.

First, we consider *Balanced Forman Curvature (BFc)* [15] and variations thereof:

- *Balanced Forman Curvature*: For an edge $i \sim j$ we have $BFc(i,j) = 0$ if $\min(d_i, d_j) = 1$ and otherwise

$$BFc(i,j) = \frac{2}{d_i} + \frac{2}{d_j} - 2 + 2\frac{|\sharp_{\triangle}(i,j)|}{d_i \vee d_j} + \frac{|\sharp_{\triangle}(i,j)|}{d_i \wedge d_j} + \frac{\left(|\sharp^i_{\square}| + |\sharp^j_{\square}|\right)}{\gamma_{max}(d_i \vee d_j)}\,. \tag{4}$$

- *Balanced Forman Curvature (without four-cycles)*: Determining the number of four-cycles without diagonals inside is a costly computational effort, especially for dense graphs. We therefore analyse the rewiring performance of *BFc* without these four-cycles to evaluate the need of more intensive computations. For an edge $i \sim j$ we have $BFc_3(i,j) = 0$ if $\min(d_i, d_j) = 1$ and otherwise

$$BFc_3(i,j) = \frac{2}{d_i} + \frac{2}{d_j} - 2 + 2\frac{|\sharp_{\triangle}(i,j)|}{d_i \vee d_j} + \frac{|\sharp_{\triangle}(i,j)|}{d_i \wedge d_j}\,. \tag{5}$$

- *Modified Balanced Forman Curvature*: The original code implementation provided in [15] contained an error in the counting of four-cycles (See Appendix B for more details). We therefore implement this version as well for comparison. For an edge $i \sim j$ we have $BFc_{mod}(i,j) = 0$ if $\min(d_i, d_j) = 1$ and otherwise

$$BFc_{mod} = \frac{2}{d_i} + \frac{2}{d_j} - 2 + 2\frac{|\sharp_{\triangle}(i,j)|}{d_i \vee d_j} + \frac{|\sharp_{\triangle}(i,j)|}{d_i \wedge d_j} + \mathcal{O}\left(|\sharp^i_{\square}| + |\sharp^j_{\square}|\right)\,. \tag{6}$$

Secondly, we consider Jost and Liu Curvature (*JLc*) [16]. For an edge $i \sim j$ we have, with $s_+ \doteq \max(s, 0)$,

$$JLc(i,j) = -\left(1 - \frac{1}{d_i} - \frac{1}{d_j} - \frac{|\sharp_{\triangle}(i,j)|}{d_i \wedge d_j}\right)_+ - \left(1 - \frac{1}{d_i} - \frac{1}{d_j} - \frac{|\sharp_{\triangle}(i,j)|}{d_i \vee d_j}\right)_+ + \frac{|\sharp_{\triangle}(i,j)|}{d_i \vee d_j}\,. \tag{7}$$

Finally, we consider the *Augmented Forman Curvature* used for rewiring in [17]. Originally, the *Forman curvature* was introduced in [18] as a discrete analogue of the Ricci curvature that aims to mimic the Bochner–Weitzenböck decomposition of the Riemannian Laplace operator for quasiconvex cell complexes (compact regular CW complexes which are quasiconvex, i.e. the boundary of two cells can at most consist of one lower-dimensional cell). It was then adapted to graphs [19] and augmented to also include two dimensional cells such as triangles [20, 21]. The *Augmented Forman curvature* comes in two variants, similar to *BFc*.

- A variant where we consider only three-cycle contributions to the curvature. For an edge $i \sim j$ we have

$$\mathcal{AF}_3(i,j) = 4 - d_i - d_j + 3|\sharp_{\triangle}(i,j)|\,. \tag{8}$$

- A variant where we also consider the four-cycle contributions to this curvature. It is important to note that, unlike the *Balanced Forman curvature*, the term $\square(i,j)$ in $\mathcal{AF}_4$—as uniquely used in [17]—counts all four-cycles located at a given edge $i \sim j$ without obstructions on diagonals inside. For an edge $i \sim j$ we have

$$\mathcal{AF}_4(i,j) = 4 - d_i - d_j + 3|\sharp_{\triangle}(i,j)| + 2\square(i,j)\,. \tag{9}$$

B On the implementation of the Balanced Forman curvature

During the setup of our experiment, we noticed that the implementation of the Balanced Forman curvature provided by the authors of [15] under https://github.com/jctops/understanding-oversquashing does not match the theoretical definition presented in their paper (Definition 1 in [15]). More precisely, the issue for a given edge $i \sim j$ evolves around the terms corresponding to the four-cycle contributions, i.e. $|\sharp^i_\square|$, $|\sharp^j_\square|$ and γ_{max}. The degree d_i and d_j of the involved nodes and the number of triangles $|\sharp_\triangle(i,j)|$ are calculated correctly. However, even for the sample graph provided in Figure. 3 in [15] (Figure 2 here) the publicly available implementation produces demonstrably wrong results. The four-cycle contribution is computed as illustrated in the code below with $\mathrm{sharp}_{ij} = |\sharp^i_\square| + |\sharp^j_\square|$, $\mathrm{lambda}_{ij} = \gamma_{\max}$.

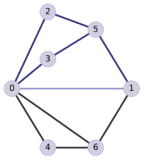

Figure 2: Sample graph provided in Figure. 3 in [15]

If we consider the edge $0 \sim 1$ of the sample graph in Figure 2 and using the definition of the Balanced Forman curvature, Eq. (4) we find $BFc(0,1) = 0.10$. In contrast, when using the publicly available code we find $BFc(0,1) = 0.08$.

To control that our computation of BFc was correct we implemented with the *NetworkX* library the set-theoretical definition (evolving around the 1-hop neighbourhoods $S_1(i)$ and $S_1(j)$ of the involved nodes) provided in Definition 1 in [15]. These are

1. $\sharp\Delta(i,j) := S_1(i) \cap S_1(j)$ are the triangles based at $i \sim j$.
2. $\sharp^i_\square(i,j) := \{k \in S_1(i) \backslash S_1(j), k \neq j : \exists w \in (S_1(k) \cap S_1(j)) \backslash S_1(i)\}$ are the neighbours of i forming a 4-cycle based at the edge $i \sim j$ without diagonals inside.
3. $\gamma_{\max}(i,j)$ is the maximal number of four-cycles based at $i \sim j$ traversing a common node

Additionally, we used Remark 10 from [15] to calculate the four-cycle contribution and in particular γ_{max}. Our CUDA implementation of *BFc* was then controlled with the *Networkx* implementation. Both our implementations are available in our code.

Listing 1: Code snippet from [15] to compute the 4-cycle contribution of the *BFc*

```
// 4-cycle contribution
for k in range(N):
                TMP = A[k, j] * (A2[i, k]-A[i, k]) * A[i, j]
                if TMP > 0:
                    sharp_ij += 1
                    if TMP > lambda_ij:
                        lambda_ij = TMP
                TMP = A[i, k] * (A2[k, j]-A[k, j]) * A[i, j]

                if TMP > 0:
                    sharp_ij += 1
                    if TMP > lambda_ij:
                        lambda_ij = TMP

            C[i, j] = ((2 / d_max) + (2 / d_min) - 2 + (2 / d_max + 1 / d_min) *
                A2[i, j] * A[i, j] )
            if lambda_ij > 0:
                C[i, j] += sharp_ij / (d_max * lambda_ij)
```

C Hyperparameter experiment

C.1 Experimental setup

For the node classification tasks, we used 9 datasets to evaluate the different curvature notions: Texas, Cornell and Wisconsin from WebKB [22], Chameleon and Squirrel [23], Actor [24], Cora [25], Citeseer [26] and Pubmed [27]. For graph classification, we used MUTAG and PROTEINS [28]. For each dataset we only use the largest connected component, extracted with the algorithm provided in https://github.com/jctops/understanding-oversquashing. Directed graphs were subsequently transformed to undirected.

If the dataset contains disconnected components we report the statistics for the largest connected component selected. The homophily index $\mathcal{H}(G)$ defined by [29] is defined as

$$\mathcal{H}(G) = \frac{1}{|V|} \sum_{v \in V} \frac{\text{Number of } v \text{ 's neighbors who have the same label as } v}{\text{Number of } v \text{ 's neighbors}} . \tag{10}$$

	Texas	Cornell	Wisconsin	Chameleon	Squirrel	Actor	Cora	Citeseer	Pubmed
$\mathcal{H}(G)$	0.06	0.11	0.16	0.25	0.24	0.22	0.83	0.72	0.79
Nodes	135	140	184	832	2186	4388	2485	2120	19717
Edges	251	219	1703	12355	65224	21907	5069	3679	44324
Features	1703	1703	1703	2323	2089	931	1433	3703	500
Classes	5	5	5	5	5	5	7	6	3
Directed?	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No

Our hyperparameter grid is defined in line with hyperparameters reported from [15]: learning rate: $[0.0001, 0.5555]$, number of layers: $\{1, 2, 3\}$, layer width: $\{16, 32, 64, 128\}$, dropout: $[0.0001, 0.5555]$, weight decay: $[0.0001, 0.9999]$, C^+: $[0.2, 21.2]$, τ: $[1, 500]$ and the max number of iterations is dataset dependent, where we take the lower and upper boundary to be 20% above and below the reported best hyperparameters from [15]. We perform a random-grid search with 1000 iterations for all datasets, except for Pubmed and Chameleon where we performed 500 iterations. Our evaluations follows [30]. Each dataset is split into a development set and a test set with a fixed seed. The development set is then split 100 times into a validation and training set using the same seeds as in [15]. Each hyperparameter configuration is then trained and evaluated on each of the 100 training-validation sets. The reported accuracy is then the mean test accuracy over the 100 validation sets. We train the networks with early stopping, where the patience is 100 epochs of no improvement on the validation set. We use a GCN [2] for all datasets together with the Adam optimizer [31].

C.2 Hyperparameter accuracy distributions

Figure 3 shows the smoothed distributions (kernel density estimates), together with boxenplots, of the average accuracy obtained per dataset for each rewiring measure. Next to looking at the distributions, we can look at the top 10% of the reported accuracies given by hyperparameters, which is presented in Table 2.

Table 2: For each dataset we take the top 10% results from the hyperparameter sweeps and compute the average mean test accuracy obtained together with the standard deviation. For some datasets, the top 10% showed almost no variability which resulted in a standard deviation of (almost) 0.

	Texas	Cornell	Wisconsin	Chameleon	Cora	Citeseer
None	59.95 ± 1.15	53.66 ± 0.14	54.92 ± 0.51	40.76 ± 3.52	58.83 ± 16.36	58.14 ± 7.33
BFc	59.26 ± 0.00	53.61 ± 0.28	54.06 ± 0.01	34.58 ± 3.19	28.39 ± 17.24	35.99 ± 13.82
BFc_3	59.26 ± 0.00	53.59 ± 0.12	54.06 ± 0.01	30.93 ± 0.10	21.86 ± 08.75	30.35 ± 12.03
BFc_{mod}	59.26 ± 0.00	53.68 ± 0.43	54.91 ± 1.72	31.60 ± 2.04	27.73 ± 13.08	44.16 ± 12.46
JLc	59.26 ± 0.00	53.57 ± 0.01	54.06 ± 0.02	30.91 ± 0.02	26.83 ± 13.82	42.48 ± 7.83
AFc_3	59.58 ± 0.52	54.20 ± 1.57	56.37 ± 1.60	36.93 ± 5.14	59.25 ± 14.83	60.11 ± 6.30
AFc_4	59.79 ± 0.54	53.63 ± 0.10	54.60 ± 0.80	31.20 ± 0.62	58.68 ± 16.10	61.67 ± 5.43

	Pubmed	Actor	Squirrel	MUTAG	PROTEINS
None	41.99 ± 12.58	27.73 ± 0.02	36.73 ± 1.96	55.34 ± 3.67	61.45 ± 1.49
BFc	39.67 ± 8.30	27.73 ± 0.01	35.35 ± 1.00	54.38 ± 1.71	61.36 ± 1.23
BFc_3	40.97 ± 12.01	27.73 ± 0.02	34.66 ± 0.54	54.45 ± 2.25	61.16 ± 0.00
BFc_{mod}	41.23 ± 11.30	27.73 ± 0.01	34.89 ± 0.85	54.56 ± 2.32	61.20 ± 0.18
JLc	39.47 ± 8.97	27.73 ± 0.02	34.53 ± 0.26	54.53 ± 2.86	61.20 ± 0.37
AFc_3	42.74 ± 12.91	28.00 ± 0.93	35.78 ± 1.83	54.63 ± 2.91	61.22 ± 0.45
AFc_4	41.40 ± 12.65	28.14 ± 1.02	35.64 ± 1.40	54.54 ± 2.40	61.27 ± 1.02

A first observation is that no curvature measure consistently shifts the distribution of the mean test accuracy away from the None distribution (meaning no rewiring) over all datasets. We can notice interestingly that the curvature measure AFc_3 does provide better performance on datasets such as Cornell and Wisconsin, but it does not do this consistently for the other low homophily datasets Texas and Chameleon. When looking at the average of the top 10% results we see that AFc based rewiring does perform better than other variants while having a smaller standard deviation, but does

not consistently advance the performance with respect to no rewiring, indicating a specific dataset dependency. It is interesting to note that rewiring can also negatively impact the performance of a dataset, as seen by the larger spread for the citation networks. Secondly, when comparing the results from Table 1 with the distribution of BFc in Figure 3, we can see that the datasets with more edges that satisfy condition 2b do not perform better than others (e.g. Pubmed). Finally, we can note the similar performance of less-computational intensive curvature measures (BFc_3 and AFc_3) in comparison with their more intensive variant (BFc and AFc_4).

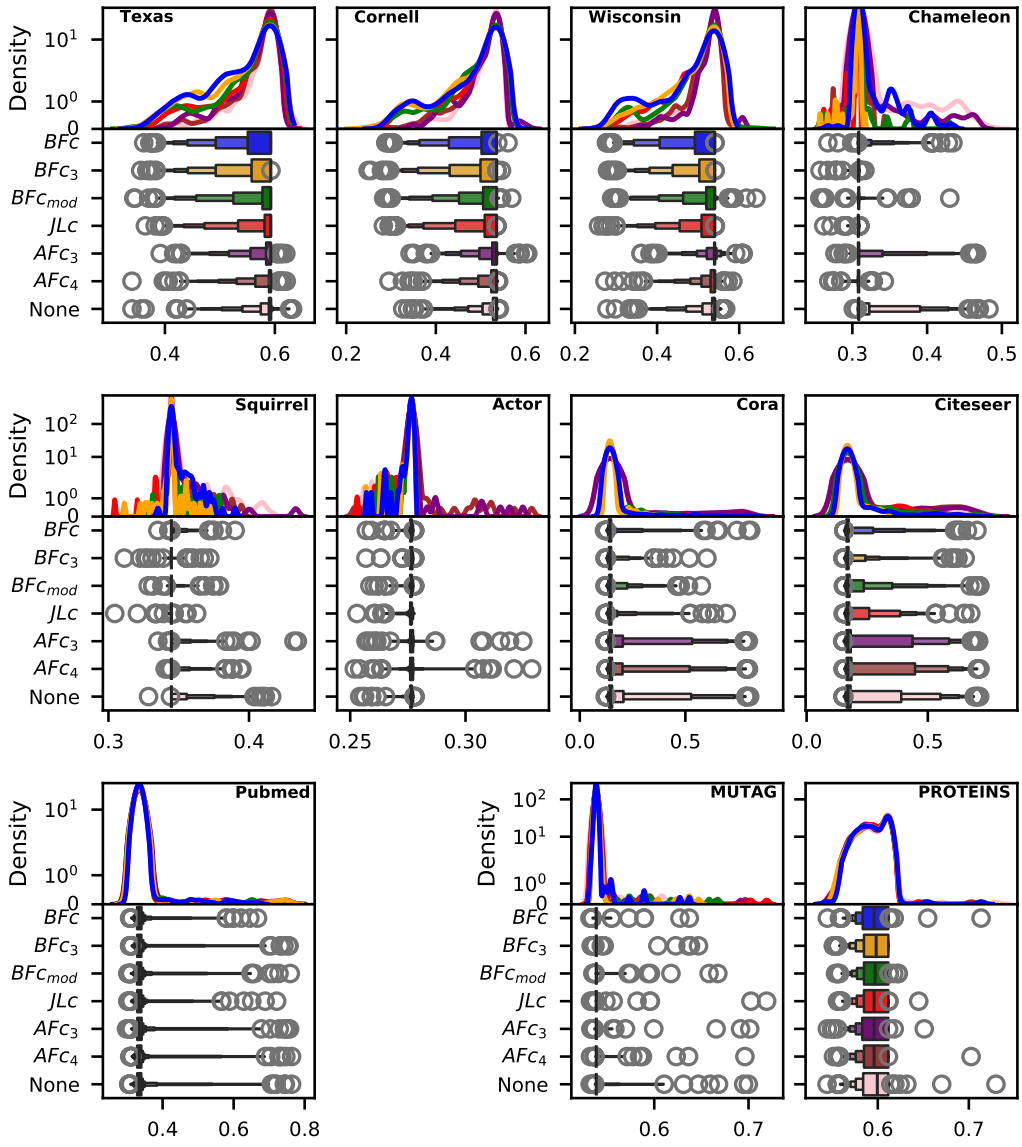

Figure 3: Distribution of mean test accuracy over the sweep of hyperparameters for the different curvature measures and node-classification datasets used. We show boxenplots which first identify the median, then extend boxes outward, each covering half of the remaining data on which outliers (circles) are identifiable. For each dataset we also show the smoothed distribution using kernel density estimates from the seaborn package.