Delaunay Graph: Addressing Over-Squashing and Over-Smoothing Using Delaunay Triangulation

Hugo Attali [1] **Davide Buscaldi** [1] **Nathalie Pernelle** [1]

Abstract

GNNs rely on the exchange of messages to distribute information along the edges of the graph. This approach makes the efficiency of architectures highly dependent on the specific structure of the input graph. Certain graph topologies lead to inefficient information propagation, resulting in a phenomenon known as over-squashing. While the majority of existing methods address over-squashing by rewiring the input graph, our novel approach involves constructing a graph directly from features using Delaunay Triangulation. We posit that the topological properties of the resulting graph prove advantageous for mitigating over-smoothing and over-squashing. Our extensive experimentation demonstrates that our method consistently outperforms established graph rewiring methods.

1. Introduction

The proliferation of graph-structured data has fueled the creation of specialized machine learning algorithms finely tuned to handle this distinctive data format. Among these, graph neural networks (GNNs) have emerged as the standard approach for effective learning with such data. GNNs applied to graph structures play a crucial role in addressing various tasks associated with graphs (Goller & Kuchler, 1996; Gori et al., 2005; Scarselli et al., 2008; Bruna et al., 2014). These architectures find applications in diverse fields such as chemistry, information retrieval, social network analysis, knowledge graphs (Zhou et al., 2020; Wu et al., 2020).

GNNs employ an iterative approach, updating node representations through the local aggregation of information from neighboring nodes, known as the message-passing paradigm (Gilmer et al., 2017).

[1]LIPN, Universite Sorbonne Nord. Correspondence to: Hugo Attali <attali@lipn.univ-paris13.fr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

However, a significant limitation arises where adjacent nodes exhibit dissimilar labels (heterophilic case), classical MPNNs may experience diminished performance (Zheng et al., 2022). This limitation, rooted in one-hop message propagation, necessitates the stacking of additional layers to capture non-local interactions. Unfortunately, this excessive message passing tends to make node features more and more similar, leading to oversmoothing.

Another well-explored issue is the phenomenon of over-squashing (Alon & Yahav, 2021), which occurs when an exponentially expanding quantity of information is compressed into a fixed-size vector. These phenomena can be identified through local structural properties of the graph, such as discrete Ricci curvature (Topping et al., 2022).

Previous studies have demonstrated a correlation between the presence of bottlenecks in graphs and over-squashing, causing a significant loss of information that can markedly decrease the effectiveness of message passing (Alon & Yahav, 2021) (Topping et al., 2022) .

Many existing methods mitigate over-squashing by rewiring the input graph to minimize structural bottlenecks. To achieve this graph rewiring, most methods rely on structural graph features, which can be either local, such as edge curvature (Topping et al., 2022) (Nguyen et al., 2023), or more global, such as resistance (Black et al., 2023). While these methods yield promising results, their primary limitation lies in the necessity to analyse the input graph structure using methods that may not readily scale with the increasing number of nodes. Besides, the choice of the hyperparameters often relies heavily on the characteristics of the dataset. Additionally, the existing rewiring methods modify the original graph, but in numerous applications such as document classification (Yao et al., 2019; Guille & Attali, 2022), physics simulation (Sanchez-Gonzalez et al., 2020), and entity representation for natural language processing (Luan et al., 2019) , only the features are available, while the graph structure is not accessible. This constraint hampers the effectiveness and the applicability of all structural rewiring methods.

Table 1: Comparison of various reviewing methods

Models	Input Graphs	Type of Rewiring	Complexity	Hyperparameters
DIGL	Required	Structural Rewiring	$\mathcal{O}(N)$	Grid-search
SDRF	Required	Structural Rewiring	$\mathcal{O}(\lvert\mathcal{E}\rvert d_{\max}^2)$	Grid-search
FOSR	Required	Structural Rewiring	$\mathcal{O}(N^2)$	Grid-search
BORF	Required	Structural Rewiring	$\mathcal{O}(\lvert\mathcal{E}\rvert d_{\max}^3)$	Grid-search
GTR	Required	Structural Rewiring	$\mathcal{O}(\lvert\mathcal{E}\rvert \text{ poly } \log N + N^2 \text{ poly } \log N)$	Grid-search
DR (Ours)	Not necessary	Feature Rewiring	$\mathcal{O}(N \log(N))$	Heuristic

Reproducibility. Our code to reproduce the experiments of the paper is available. [1]

Main Contributions In this paper, we present Delaunay Rewiring (DR), a novel rewiring method that incorporates node features with reasonable complexity to alleviate both over-squashing and over-smoothing issues :

- Instead, of altering the existing graph structure, our approach involves reconstructing the graph by leveraging features extracted through Delaunay triangulation.
- We demonstrate that the structural properties of the graph contribute significantly to the efficient propagation of information to mitigate oversmoothing and over-squashing.
- We carry out an extensive evaluation illustrating the effectiveness of our method not only on heterophilic graphs but also on homophilic ones.

2. Preliminaires

We start by introducing notations used throughout this paper. A graph is written as a tuple $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ and $\mathcal{E}$ denote the set of nodes and edges, respectively. We note the numbers of nodes $N = |\mathcal{V}|$, and an edge connecting nodes i (from) and j (to) is denoted by $e_{ij} \in \mathcal{E}$. In this work, we focus on undirected graphs, i.e., if $e_{ij} \in \mathcal{E}$, then $e_{ji} \in \mathcal{E}$. We define $N \times N$ adjacency matrix $\mathbf{A}$ as $\mathbf{A}_{i,j} = 1$ if $(i, j) \in \mathcal{E}$ and zero otherwise. In addition, we let $\mathbf{D}$ be the diagonal matrix with $D_{ii} = d_i$, the degree of node i, and denote $d_{\max}$ and $d_{\min}$ as the maximal and minimal degrees.

3. MPPNs

Graph Neural Networks operate on the Message Passing Neural Networks (MPNNs) paradigm. This paradigm involves iteratively applying AGGREGATE and UPDATE functions to enhance node representations. The process updates the representations by utilizing the information contained in the neighbors.

$$h_i^{(\ell+1)} = \phi^\ell \left(h_i^{(\ell)}, \oplus_{j=1}^{|V|} \hat{\mathbf{A}}_{ij} \psi_\ell \left(h_i^{(\ell)}, h_j^{(\ell)} \right) \right) \quad (1)$$

Where ψ is the message function , depends on h_i, h_j and possibly the edge weight e_{ji}, $\oplus$ represents a permutation invariant aggregation operation and ϕ the update function, final transformation to obtain new embedding after aggregating messages.

Multiple approaches exist for Message Passing Neural Networks (MPPNs), each employing distinct strategies:

- Fixed Graph Approach: This method utilizes a static graph structure for message passing, exemplified by Graph Convolutional Network (GCN) (Kipf & Welling, 2017) or Graph Isomorphism Network (Xu et al., 2018). Here, the connections between nodes remain constant throughout the network's operation.
- Feature Reweighting Edges Approach : In this approach, exemplified by Graph Attention Networks (GAT) (Veličković et al., 2018) and its variants (Brody et al., 2021), edges are dynamically reweighted during message passing based on the relevance of features. This enables the network to adaptively focus on specific edges, enhancing its ability to capture intricate relationships within the graph.

Despite their widespread use, Graph Neural Networks (GNNs) have been shown to encounter various challenges, particularly in heterophilic environments where neighboring nodes tend to have different labels (Zhu et al., 2020; Platonov et al., 2023). Additionally, studies have highlighted the difficulty GNNs face in effectively modeling long-range interactions (Alon & Yahav, 2021). The primary limitation of MPPNs lies in their local operation. To exchange information between two nodes at distance k, at least k layers must be stacked.

However, increasing the number of layers gives rise to two primary flaws in MPPNs: oversmoothing and over-squashing.

[1] Code available from: https://github.com/Hugo-Attali/Delaunay-Rewiring

3.1. Oversmoothing

As the number of layers increases, the message passing becomes excessively intensive. In such instances, with each layer, the features of the nodes gradually become more similar. This phenomenon results in a decline in performance, as reported in (Oono & Suzuki, 2020) and (Cai & Wang, 2020a). A proficient strategy to mitigate oversmoothing involves incorporating residual connections in deep networks. Other approaches lie in the idea of sparsifying the graph (Cai & Wang, 2020b; Chen et al., 2020; Rong et al., 2019). More specifically,some subgraph structures such as large cliques may increase the effect of oversmoothing. So, some approaches aim to specifically sparsify such structures (Nguyen et al., 2023).

3.2. Over-squashing

Long-range tasks necessitate information propagation across multiple layers. Node representations are aggregated with others at each stage before being forwarded to the next node. However, the fixed size of node feature vectors leads to a rapid depletion of their representational capacity, especially when accommodating previously integrated information. This results in over-squashing when an exponentially expanding amount of information is compressed into a fixed-size vector, as discussed by (Alon & Yahav, 2021)

In such scenarios, local information spreading alone proves insufficient. To address this challenge, GNNs must incorporate additional global graph features during the representation learning process (Gilmer et al., 2017). Another effective strategy involves rewiring the input graph to enhance connectivity and to alleviate structural bottlenecks. This adjustment allows for a more effective and balanced information flow within the network.

Recent studies emphasize the significance of local structural properties, such as edge curvature (Topping et al., 2022; Nguyen et al., 2023), in facilitating knowledge dissemination throughout the graph.

3.3. Homophily

The homophily of a graph plays a crucial role in determining the efficiency of architectures for node classification tasks. Various homophily measures have been discussed in the literature, as highlighted by (Pei et al., 2020; Zhu et al., 2020; Lim et al., 2021), and (Platonov et al., 2022). Among these measures, two commonly employed ones are node homophily, as described by (Pei et al., 2020), which calculates the average proportion of neighbors sharing the same class label for each node, and edge homophily (Zhu et al., 2020), representing the fraction of edges connecting nodes of the same class. For graphs exhibiting low homophily, i.e., where neighboring nodes tend to have different labels, it becomes crucial to seek information over longer distances. In this case graph structures with bottlenecks pose the greatest challenges to MPPNS in such scenarios.

3.4. Curvature on graphs

By representing a manifold as a graph, leveraging the curvature of graph edges becomes a valuable method for capturing local graph information. Positive curvature edges establish connections between nodes belonging to the same community, while negative curvature edges connect nodes from different communities. More broadly, discrete graph curvature characterizes the structural connectivity between the neighbors of two nodes. The pioneering works of (Forman, 2003) and (Ollivier, 2007) introduced the concept of measuring discrete graph curvature. Numerous studies have highlighted the efficacy of edge curvature in various graph-related tasks. Notably, (Jost & Liu, 2014; Ni et al., 2019; Sia et al., 2019) leverage Ollivier curvature for community detection. Furthermore, in a different approach, (Ye et al., 2019) proposed the Curvature Graph Neural architecture, which calculates an attention mechanism based on Ollivier curvature. They illustrate the advantages of such an architecture, particularly in the context of node classification tasks. Many edge curvature measurements have been studied:

Augmented Forman Curvature The curvature measure proposed by (Samal et al., 2018) proposes to extend Forman's curvature taking into account the triangles in the graph to make it more expressive. For an undirected graph the curvature c_{ij} of an edge e_{ij} is computed as follows:

$$c_{ij} = 4 - d_i - d_j + 3m \tag{2}$$

where m is the number of triangles that contain e_{ij}.

Balanced Forman Curvature The curvature measure proposed by (Topping et al., 2022) allows to consider cycles of size 4:

$$c_{ij} = \frac{2}{d_i} + \frac{2}{d_j} - 2 + 2\frac{m}{\max\{d_i, d_j\}} + \frac{m}{\min\{d_i, d_j\}} + \frac{(\Gamma_{\max})^{-1}}{\max\{d_i, d_j\}}(\gamma_i + \gamma_j) \tag{3}$$

where $\Gamma_{\max}(i,j)$ is the maximal number of 4-cycles based at e_{ij} and γ_i is the number of 4-cycles based at e_{ij} without diagonals inside.

Recent research has unveiled a connection between edge curvature and phenomena like oversmoothing and over-squashing. These phenomena are intricately tied to the underlying graph structure.

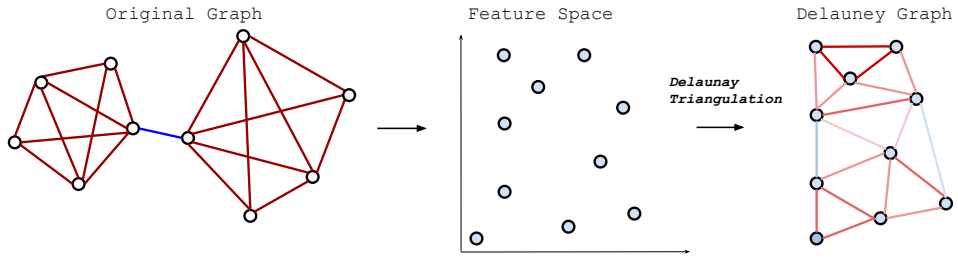


Figure 1: Delaunay graph construction process, in red the edges with positive curvature, in blue with negative curvature

The study conducted by (Nguyen et al., 2023) sheds light on the origin of oversmoothing, attributing it to regions of the graph with highly positively curved edges. On the other hand, (Topping et al., 2022) demonstrates that strongly negatively curved edges play a pivotal role in creating bottlenecks, which in turn lead to the over-squashing phenomenon.

3.5. Rewiring Methods

As the majority of GNNs are based on the message passing paradigm, the structural properties of graphs play a crucial role. Consequently, numerous methods have focused on understanding how the quality of message passing is influenced by the topology of the graph and how to quantify it. To mitigate this phenomenon, modifying the input graph is a commonly used method.

(Alon & Yahav, 2021), pioneers in addressing the issue of GNN over-squashing, advocate modifying the last layer of the GNN to establish connections among all nodes.

In a complementary study, (Topping et al., 2022) identifies highly negatively curved edges as indicative of the bottleneck phenomenon, disrupting effective message passing. They introduce a stochastic discrete Ricci Flow (SDRF) rewiring approach, aimed at enhancing the balanced Forman curvature of negatively curved edges by strategically adding and removing edges.

(Karhadkar et al., 2023) presents the FOSR algorithm, which systematically adds edges at each step to maximize the spectral gap. To mitigate the computational cost associated with calculating the spectral gap for every edge addition, FOSR employs a first-order spectral gap approximation grounded in matrix perturbation theory.

Examining the connection between positive edge curvature and oversmoothing, (Nguyen et al., 2023) underscore the need for rewiring methods. Their proposed approach involves removing overly positively curved edges to reduce oversmoothing while simultaneously eliminating excessively negatively curved edges to alleviate over-squashing.

In a distinct study, (Black et al., 2023) explores the link between Resistance and over-squashing. They demonstrate that Resistance considers a more global aspect of the graph structure compared to curvature. Their findings highlight that edges with high Effective Resistance contribute to over-squashing. To address this, they propose a rewiring method that strategically adds edges to the graph to minimize total resistance.

Without explicitly attenuating against over-squashing (Klicpera et al., 2019) aims to enhance connectivity between nodes with short diffusion by adding edges based on the PageRank algorithm.

4. Delaunay rewiring

While the majority of the rewiring methods modify the original graph structure, we propose a complete rebuild of the graph, based only on the features of the nodes, ignoring the edges of the original graph. We choose to introduce the new edges of the rewired graph by applying a Delaunay triangulation on the node features, as shown in Figure 1. This kind of triangulation ensures some positive and desirable structural properties: through the analysis of the curvature of the new graph, we demonstrate that this rewiring exhibits good topological properties and effectively mitigates the phenomena of over-squashing and bottleneck.

4.1. Delaunay Triangulation

Definition 1 A Delaunay triangulation, denoted as $DT(P)$, for a set P of points in the d-dimensional Euclidean space, is a triangulation where no point in P resides within the circum-hypersphere of any d-simplex in $DT(P)$.

In two dimensions, Delaunay triangulations maximize the angles of triangles formed by a set of points, striving to create triangles that are as close as possible to being equilateral. Simultaneously, it ensures that the circumscribed circle of each triangle contains no other points from the set. In this context, the proximity of two points enhances the probability of them being included in the same triangle.

Type	Dataset	# Nodes	# C	# O-Edges	# O-Homo	# D-Edges	# D-homo	# Homo gain
Heterophilic	Squirrel	5021	5	217073	0.22	31170	0.59	168%
	Chameleon	2277	5	36101	0.25	13630	0.69	176%
	Texas	181	5	309	0.06	1072	0.63	950%
	Wisconsin	251	5	499	0.06	1470	0.55	817%
	Cornell	181	5	295	0.11	1064	0.67	509%
	Roman-empire	22662	18	32927	0.06	135922	0.58	1060%
	Actor	7600	5	33544	0.24	45520	0.40	33%
Homophilic	Citeseer	3 312	6	4 715	0.71	19923	0.78	10%
	Cora	2 708	7	5 429	0.83	16214	0.88	6%
	Pubmed	19 717	3	44348	0.77	118192	0.86	9%

Table 2: Comparison of the statistics of the original dataset (O) and the graph obtained after Delaunay triangulation (D)

To alleviate the problem of over-squashing, we want to reduce the number of negatively curved edges. According to many definitions of curvature, the number of triangles incident on edges plays a role in increasing edge curvature. Applying a triangulation allows to maximize the value of m in equations 2 and 3 while ensuring a maximum clique size of 3.

We perform the triangulation on the set of nodes, whose position in a d-dimensional space is represented by their d features. However, as d depending on the problem can vary significantly, and considering that for large d there are known problems with distance estimation (Aggarwal et al., 2001; Liberti, 2020), we opt to reduce the dimensions to 2 applying UMAP (McInnes et al., 2018). UMAP effectively retains the spatial relationships among features and the overall data topology. A discussion about triangulation in higher dimensions is available in Appendix A.

Initially, we triangulated the graphs considering the original features. However, these starting features often lack expressiveness or suffer from poor quality, as it can be observed from the results of the experiments conducted on heterophilic datasets using this strategy in Table 3.

	GCN (O)	GCN (DG)
Cham.	**65.35**±0.54	27.82 ±0.48
Squir.	**51.30**±0.38	22.73 ±0.29
Actor	30.02 ±0.22	**30.73** ±0.26
Texas	56.19 ±1.61	**70.68** ±1.60
Wisc.	55.12±1.51	**70.98** ±1.50
Corn.	44.78 ±1.45	**67.22** ±1.48
R-emp.	51.66 ±0.17	**61.99** ±0.14

Table 3: Experimental results (accuracy) on **heterophilic** datasets. O denotes GCN on the original input graph and DG denotes the GCN on the Delaunay Graph. Best score in bold.

The observed results exhibit a notable irregularity, primarily stemming from the inherent dependence of graph quality on the initial features (Platonov et al., 2023) used for triangulation. To improve the expressiveness of the obtained graph, we can perform the triangulation on the node embeddings obtained using a GNN. This led us to another strategy, illustrated in Figure 2, in which the original features are passed to a first GNN (in our case, a Graph Convolutional Network following the model proposed by (Kipf & Welling, 2017)) to obtain the embeddings that are used to build the Delaunay graph (after UMAP dimensionality reduction). Subsequently, we carry out the prediction task by taking into account the Delaunay triangulation graph structure and the original features (H_0) for the nodes.

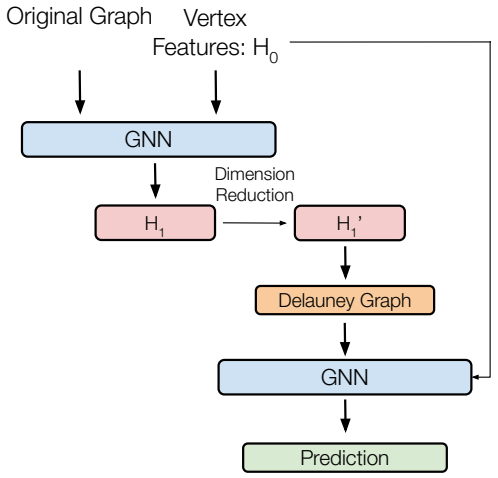
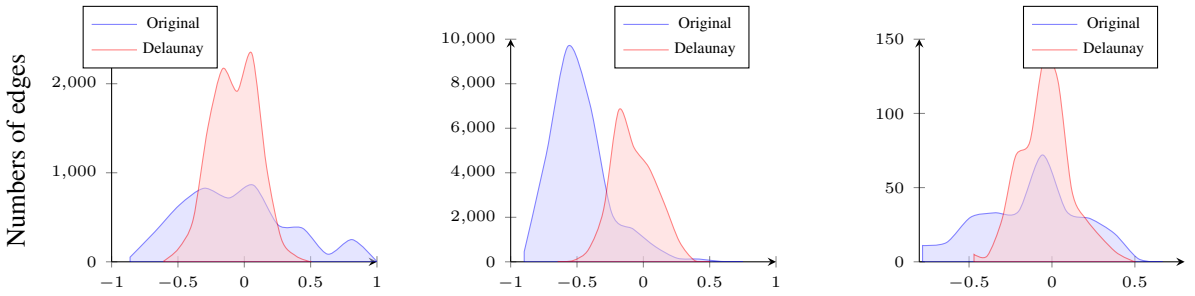


Figure 2: Illustration of the rewiring method using the features obtained by a GNN.

4.2. Delaunay Graph properties

Thanks to the generation of a novel graph via the Delaunay triangulation, we optimize the number of triangles in the graphs which strategically prevents bottlenecks and excessive compression. Additionally, we can emphasize that the largest cliques within the newly formed graph are limited to



(a): Citeseer: :
D_1= -0.25 D_9 =0.20
D_1= -0.46 D_9 =0.50

(b): Actor:
D_1= -0.21 D_9 =0.18
D_1= -0.63 D_9 =0.11

(c): Cornell :
D_1= -0.18 D_9 =0.20
D_1= -0.49 D_9 =0.33

Figure 3: Comparison of edge curvature in the graph. The x- axis denotes Ollivier curvature. The captions display the curvature deciles, including the first D_1 and the remaining nine D_9, for both the original and Delaunay graphs.

a maximum size of 3. This constraint allows to steer clear of oversmoothing (Nguyen et al., 2023), contributing to a better dissemination of information. By operating directly on the individual features of the nodes rather than the graph structure itself, we provide a way to create graphs for non-graph data structures. This flexibility allows us to apply our methodology to various domains and other non-graph-based scenarios.

Statistics on Delaunay Graphs Table 2 shows the statistics of the original graphs and the Delaunay graphs. The number of edges of the Delaunay graphs is about 6 times lower than the number of edges in the original graph (3 times if the graph is directed). This strategy makes it possible to drastically sparsify graphs, in particular those that have a high average degree. High-degree nodes are particularly prone to over-squashing. In addition, this strategy also avoids working with nodes with too low a degree, which also encourages excessive over-squashing. Indeed, a low edge-to-node ratio is characteristic of a large-diameter graph (Deac et al., 2022), making it more difficult to exchange information over long distances. The graph's sparsification not only enhances its overall efficiency but also yields notable computational performance benefits. This is particularly evident for the benchmarks where the average degree significantly exceeds 6, as reported in studies such as (Rozemberczki et al., 2021) and (Platonov et al., 2023). More discussion about the comparaison of original graph and Delaunay graph is provided in Appendix B.

We also present in Table 2 the edge homophily measure on the graph (Zhu et al., 2020) obtained after triangulation and the gain compared with the original graph. While on the homophilic graphs the gain is rather low, the homophily value is much higher for the graphs obtained after Delaunay triangulation of the heterophilic graphs (i.e. from 67% to 1067%).

Curvature edges on Delaunay Graphs Figure 3 illustrates the distribution of edge count based on edge curvature in the graph. We use Olliver's curvature (Ollivier, 2007), as it is bounded and more easily interpretable. Since the structure of the Delaunay graph exhibits minimal variation across different datasets, the curvature of its edges remains consistently similar in the obtained graphs. It is noteworthy that our graph, constructed after triangulation, removes naturally highly negatively curved edges, which are responsible for bottlenecks (Topping et al., 2022). Additionally, the new graph obtained after triangulation does not possess strongly positively curved edges, mitigating the phenomenon of over-smoothing (Nguyen et al., 2023). These local properties help alleviate the two main shortcomings of MPPNS architecture.

Resistance on Delaunay Graphs As highlighted by (Black et al., 2023), resistance serves as an effective metric for quantifying over-squashing in a somewhat broader manner than curvature. Table 4 displays the average resistance between each pair of nodes across small datasets such as Cornell, Wisconsin, and Texas. These datasets exhibit a substantial reduction in resistance, indicating a corresponding decrease in bottlenecks and therefore a better diffusion of information in the Delaunay graphs.

	Original graph	Delaunay graph
Wisconsin.	1.50	**0.73**
Texas.	1.76	**0.70**
Cornell	1.82	**0.72**

Table 4: Average resistance between each pair of nodes

	Base (GCN)	DIGL	FA	SRDF	FOSR	BORF	GTR	**DR**
Cham.	65.35±0.54	54.82 ±0.48	26.34 ±0.61	63.08 ±0.37	67.98 ±0.40	65.35 ±0.51	68.03 ±0.61	**74.28** ±0.48
Squir.	51.30±0.38	40.53 ±0.29	22.88 ±0.42	49.11±0.28	52.63 ±0.30	$\geq$ 24h	53.32 ±0.44	**65.25** ±0.26
Actor	30.02±0.22	26.75 ±0.23	26.03±0.30	31.85 ±0.22	29.26±0.23	31.36 ±0.27	31.08 ±0.28	**41.36** ±0.20
Texas	56.19 ±1.61	45.95 ±1.58	55.93 ±1.76	59.79 ±1.71	61.35 ±1.25	56.30±1.61	57.18 ±1.64	**70.46** ±1.61
Wisc.	55.12±1.51	46.90 ±1.28	46.77±1.48	58.49 ±1.23	55.60 ±1.25	55.37 ±1.47	57.22 ±1.50	**70.98**±1.50
Corn.	44.78 ±1.45	44.46 ±1.37	45.33±1.55	47.73 ±1.51	45.11 ±1.47	46.81 ±1.56	47.57 ±1.52	**67.22** ±1.48
R-emp.	51.66 ±0.17	53.93 ±0.14	OOM	52.53 ±0.13	52.38 ±0.21	58.58 ±0.14	53.31 ±0.23	**61.99** ±0.14
Cora	87.73 ±0.25	88.31 ±0.29	29.86 ±0.28	87.73 ±0.31	87.94 ±0.26	87.72±0.27	87.86 ±0.28	**91.39** ±0.24
Citeseer	76.01 ±0.25	76.22 ±0.34	22.31 ±0.34	76.43 ±0.32	76.34 ±0.27	76.49 ±0.28	76.12 ±0.28	**81.14** ±0.34
Pubmed	88.20 ±0.10	88.51 ±0.10	OOM	88.16 ±0.11	88.42 ±0.10	88.34 ±0.10	88.44 ±0.10	**88.69** ±0.10

Table 5: Experimental results (accuracy) on **heterophilic** and **homophilic** datasets with **GCN** as backbone. Best score in bold and second-best score underlined.

	Base (GAT)	DIGL	FA	SRDF	FOSR	BORF	GTR	**DR**
Cham.	65.07 ±0.41	56.34 ±0.43	27.11 ±0.56	63.15±0.44	66.61 ±0.45	66.92 ±0.51	65.97 ±0.54	**72.04** ±0.37
Squi.	50.87 ±0.56	41.65 ±0.68	21.49 ±0.71	50.36 ± 0.38	52.02 ±0.43	$\geq$ 24h	52.72 ±0.48	**61.47** ±0.29
Actor	29.92 ±0.23	31.22 ±0.47	28.20 ±0.51	31.47 ±0.25	29.73 ±0.24	29.64 ± 0.33	30.13 ±0.31	**40.25** ±0.23
Texas	56.84 ±1.61	46.49 ±1.63	56.17 ±1.71	57.45 ±1.62	61.85 ±1.41	56.68 ± 1.49	57.88 ±1.65	**74.30** ±1.38
Wisc.	53.58 ±1.39	46.29 ±1.47	46.95 ±1.52	56.80 ±1.29	54.06±1.27	55.39 ± 1.23	56.53±1.64	**74.33** ±1.24
Cornell	46.05 ±1.49	44.05 ±1.44	44.60 ±1.74	48.03 ±1.66	48.30±1.61	48.57 ± 1.56	48.70 ±1.63	**68.03** ±1.62
R-Emp.	49.23 ±0.33	53.89 ±0.16	OOM	50.75 ±0.17	49.54 ±0.31	51.03 ± 0.26	50.60 ±0.24	**61.80** ±0.16
Cora	87.65 ±0.24	88.31 ±0.29	30.44 ±0.26	88.11 ±0.28	88.13 ±0.27	87.72±0.27	87.94±0.23	**91.37** ±0.23
Citeseer	76.20 ±0.27	76.22 ±0.34	23.11 ±0.32	76.26 ±0.31	75.94±0.32	76.44 ±0.44	76.35 ±0.28	**81.61** ±0.25
Pubmed	87.39 ±0.11	87.96 ±0.10	OOM	87.44 ±0.12	87.56 ±0.11	87.61 ±0.12	87.31 ±0.12	**89.14** ±0.09

Table 6: Experimental results (accuracy) on **heterophilic** and **homophilic** datasets with **GAT** as backbone. Best score in bold and second-best score underlined.

4.3. Complexity and hyperparameters

The complexity of our rewiring method is $\mathcal{O}(N\log(N))$, rendering it more efficient than the majority of structural rewiring methods, particularly advantageous for handling large datasets (Dwivedi et al., 2022). In contrast to numerous approaches relying on grid search for rewiring adjustments (such as determining the number of edges added or removed), which often renders results sensitive to the intricacies of grid search, our method stands out. Notably, our rewiring method is free from hyperparameters, relying exclusively on the node features, ensuring heightened robustness and easier to use compared to alternative techniques.

Running Time We simulate data in two dimensions and analyze the time needed to construct the graph resulting from triangulation. Table 7 shows the efficient use of Delauney triangulation in scenarios with a large number of observations.

Number of data	**Time (in sec)**
100 000.	≤ 1
1 000 000	15
5 000 000	200

Table 7: Time for triangulation on simulated data

5. Experiments

5.1. Datasets

We conducted experiments on ten different datasets for the node classification task, comprising seven heterophilic datasets (Tang et al., 2009; Rozemberczki et al., 2021; Platonov et al., 2023) and three homophilic datasets (Sen et al., 2008). The dataset statistics are presented in Table 2.

5.2. Baseline

We conducted a comparative analysis of our method against six other techniques based on rewiring strategies. The methods include FA (Alon & Yahav, 2021), DIGL (Klicpera et al., 2019)[2], SDRF,(Topping et al., 2022)[3], FOSR (Karhadkar et al., 2023)[4],BORF [5](Nguyen et al., 2023) and GTR[6](Black et al., 2023).

Hyperparameters We performed hyperparameter finetuning for various rewiring methods. For DIGL, we refined the top k values from the set $\{8, 16, 32, 64, 128\}$ and personalized PageRank parameters from the set $\{0.05, 0.1, 0.15\}$. Concerning SDRF, we followed the hyperparameters out-

[2]https://github.com/gasteigerjo/gdc

[3]https://github.com/jctops/understanding-over-squashing/tree/main

[4]https://github.com/kedar2/FoSR/tree/main

[5]https://github.com/hieubkvn123/revisiting-gnn-curvature

[6]https://github.com/blackmit/gtr_rewiring

lined in the original publications and fine-tuned the number of iterations. For FOSR, we fine-tuned the number of iterations. In the case of BORF, we fine-tuned the top values from the sets of the number of batches n $\{1, 2, 3\}$, the number of edges added per batch h $\{10, 20, 30, 40\}$, and the number of edges removed per batch k $\{10, 20, 30, 40\}$. For GTR, our tuning involved the top values for the number of edges added, selecting from $\{8, 16, 32, 64, 96\}$.

Delaunay Rewiring configuration In the case of the Roman Empire, Cornell, Texas, and Wisconsin datasets, we employ triangulation on the original features. Meanwhile, for the remaining datasets, we adopt the configuration outlined in the following Figure 2. Results on both strategies are available Table 18.

5.3. Setup

For our experiments, we employ the same framework as outlined in (Pei et al., 2020) to assess the robustness of each method. We hypothesize that refining certain hyperparameters can lead to improved results. Hence, we set the number of layers to 2, dropout to 0.5, learning rate to 0.005, patience to 100 epochs, and weight decay to $5E^{-6}$ (Texas, Wisconsin and Cornell) or $5E^{-5}$ (other datasets). The number of hidden states is set to 32 (Texas/Wisconsin/Cornell), 48 (Squirrel, Chameleon and Roman-Empire), 32 (Actor), and 16 (Cora, Citeseer and Pubmed).

We utilize the two most popular GNNs, GCN (Kipf & Welling, 2017) and GAT (Veličković et al., 2018), as a basis and compare various methods for rewiring the input graph.

For all graph datasets, we randomly sample 60% of nodes for training, allocate 20% for validation, and reserve another 20% for testing. We also add self-loops to each node. We report the average accuracy of each method across 100 random samples.

5.4. Results

Table 5 and 6 shows that the Delaunay Rewiring method (DR), has obtained the best accuracy for the 7 heterophilic datasets and the 3 homophilic datasets. On average, this method demonstrated a substantial increase of 21.8% in performance when compared to the basic GCN and GAT. Baseline methods like SDRF, FOSR, BORF, and GTR have shown promising advancements on heterophilic datasets but demonstrate limitations when handling homophilic graphs. The discrepancy in results primarily stems from existing approaches focusing on mitigating oversmoothing and oversquashing through adjustments to the graph's original structure, neglecting to enhance homophily effectively.

5.5. Delaunay graph and k-NN graph

When structural information is not available, a common strategy for solving classical graph learning problems is to use nearest-neighbour graphs. This approach effectively connects nodes with similar representations, promoting homophily within the network. However, this structure is sub-optimal and does not offer significant advantages in fully supervised scenarios (Errica, 2023), and its usefulness is limited for large graphs due to its complexity $\mathcal{O}(N^2)$ (Dwivedi et al., 2022). In addition, the selection of the parameter k is not straightforward, introducing challenges such as non-connectivity in the resulting graph, making it less suitable for some applications. Exploring alternatives to the KNN graph with a manageable complexity is a interesting area of research, currently under scrutiny (Errica, 2023; Wu et al., 2022).

To illustrate the advantages of using a Delaunay graph over a simple k-NN graph for node classification tasks constructed with identical features, we perform experiments with k=3 for fair comparison.

	3-NN	DR
Chameleon	72.69 ±0.44	**74.28** ±0.48
Squirrel	64.91±0.25	**65.25** ±0.26
Actor	39.64 ±0.23	**41.36** ±0.20
Texas	67.93 ±1.71	**70.46** ±1.69
Wisconsin.	67.81 ±1.51	**70.98** ±1.50
Cornell	63.88 ±1.49	**67.22**±1.49
R-empire	**63.01** ±0.17	61.99 ±0.14

Table 8: Experimental results on **heterophilic** datasets with **GCN** as backbone. Best score in bold and second-best score underlined.

DR has obtained the best accuracy for 6 of the 7 heterophilic datasets. k-NN graph can obtain higher scores than the baselines on such datasets but this method is not designed to mitigate bottleneck structures.

6. Conclusion

In this paper, we introduce an innovative rewiring method characterized by a reasonable complexity and an absence of hyperparameters, leveraging Delaunay triangulation. Our approach involves utilizing node features to restructure the connectivity of the graph, showcasing the advantageous effects of Delaunay triangulation in enhancing graph homophily and ensuring structural properties favorable to information diffusion.

Through the avoidance of strongly negatively and positively curved edges, the newly constructed graph exhibits reduced sensitivity to oversmoothing and oversquashing, thereby

yielding a substantial performance boost across various datasets in the context of node classification tasks. Our empirical evaluations underscore the efficiency of our graph construction method in comparison to the conventional k-NN graph built exploiting the same node features. This proposal paves the way for an alternative graph construction solution, particularly beneficial for non-graph-based scenarios.

Limitations and future works Given the dependency of our method on features, it is imperative to ensure the quality of these features. In cases where the original features are suboptimal or the learning method struggles to acquire effective representations, the resultant graph from the triangulation process may lack quality. For prospective research, it would be valuable to explore the applicability of our method to long-range graph benchmarks. An intriguing avenue for further investigation involves evaluating our method in scenarios where only features are accessible. This exploration could shed light on the method's effectiveness in applications where structural information is limited or unavailable.

Acknowledgements

We are grateful for anonymous reviewer feedback.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. On the surprising behavior of distance metrics in high dimensional space. In *Database Theory—ICDT 2001: 8th International Conference London, UK, January 4–6, 2001 Proceedings 8*, pp. 420–434. Springer, 2001.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

Black, M., Wan, Z., Nayyeri, A., and Wang, Y. Understanding oversquashing in gnns through the lens of effective resistance. In *International Conference on Machine Learning*, pp. 2528–2547. PMLR, 2023.

Brody, S., Alon, U., and Yahav, E. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs, 2014.

Cai, C. and Wang, Y. A note on over-smoothing for graph neural networks. *Graph Representation Learning*, 2020a.

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

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3438–3445, 2020.

Deac, A., Lackenby, M., and Veličković, P. Expander graph propagation. In *Learning on Graphs Conference*, pp. 38–1. PMLR, 2022.

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022.

Errica, F. On class distributions induced by nearest neighbor graphs for node classification of tabular data. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Forman, R. Bochner's method for cell complexes and combinatorial ricci curvature. 2003.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Goller, C. and Kuchler, A. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pp. 347–352. IEEE, 1996.

Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Guille, A. and Attali, H. Document classification with hierarchical graph neural networks. In *18th International Workshop on Mining and Learning with Graphs*, 2022.

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

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

Kipf, T. N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations*, ICLR, 2017.

Klicpera, J., Weißenberger, S., and Günnemann, S. Diffusion improves graph learning. In *Advances in neural information processing systems*, NeurIPS, 2019.

Liberti, L. Distance geometry and data science. *Top*, 28(2): 271–339, 2020.

Lim, D., Hohne, F., Li, X., Huang, S. L., Gupta, V., Bhalerao, O., and Lim, S. N. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Advances in neural information processing systems*, NeurIPS, pp. 20887–20902, 2021.

Luan, Y., Wadden, D., He, L., Shah, A., Ostendorf, M., and Hajishirzi, H. A general framework for information extraction using dynamic span graphs. *arXiv preprint arXiv:1904.03296*, 2019.

McInnes, L., Healy, J., and Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Nguyen, K., Hieu, N. M., Nguyen, V. D., Ho, N., Osher, S., and Nguyen, T. M. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *International Conference on Machine Learning*, pp. 25956–25979. PMLR, 2023.

Ni, C.-C., Lin, Y.-Y., Luo, F., and Gao, J. Community detection on networks with ricci flow. *Scientific reports*, 9(1):9984, 2019.

Ollivier, Y. Ricci curvature of metric spaces. *Comptes Rendus Mathematique*, 345(11):643–646, 2007.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. *Proceedings of the International Conference on Learning Representations*, 2020.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In *Advances in neural information processing systems*, ICLR, 2020.

Platonov, O., Kuznedelev, D., Babenko, A., and Prokhorenkova, L. Characterizing graph datasets for node classification: Beyond homophily-heterophily dichotomy. *arXiv preprint arXiv:2209.06177*, 2022.

Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., and Prokhorenkova, L. A critical look at the evaluation of gnns under heterophily: are we really making progress? In *International Conference on Learning Representations*, 2023.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *International conference on learning representations*, ICLR, 2019.

Rozemberczki, B., Allen, C., and Sarkar, R. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Samal, A., Sreejith, R., Gu, J., Liu, S., Saucan, E., and Jost, J. Comparative analysis of two discretizations of ricci curvature for complex networks. *Scientific reports*, 8(1): 8650, 2018.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pp. 8459–8468. PMLR, 2020.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model, 2008.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Sia, J., Jonckheere, E., and Bogdan, P. Ollivier-ricci curvature-based method to community detection in complex networks. *Scientific reports*, 9(1):9800, 2019.

Tang, J., Sun, J., Wang, C., and Yang, Z. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 807–816, 2009.

Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. *Proceedings of the International Conference on Learning Representations*, 2022.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. ICLR, 2018.

Wu, Q., Zhao, W., Li, Z., Wipf, D. P., and Yan, J. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems*, 35:27387–27401, 2022.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. In *IEEE transactions on neural networks and learning systems*, volume 32, pp. 4–24. IEEE, 2020.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Yao, L., Mao, C., and Luo, Y. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 7370–7377, 2019.

Ye, Z., Liu, K. S., Ma, T., Gao, J., and Chen, C. Curvature graph network. In *International conference on learning representations*, 2019.

Zheng, X., Liu, Y., Pan, S., Zhang, M., Jin, D., and Yu, P. S. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

A. Notes about triangulation dimension

In the paper, we present a triangulation in 2 dimensions (resulting from the dimensionality reduction of node embeddings). In high dimensions, a Delaunay triangulation is defined as a triangulation where no point lies within the hypersphere circumscribed by a simplex of the triangulation. In this section we discuss the effects of choosing a different dimension for the triangulation. First of all, it should be noted that, as the dimensionality increases, the number of vertices in the graph also increases. For instance, in dim=3 the generalization of triangles is tetrahedra, which have 6 edges, while in dim=4 the generalization of triangles has 10 edges. Thus, performing triangulation in higher dimensions yields denser resulting graphs. We present the results of GCN and GAT for higher dimensions. In the vast majority of cases, reducing the dimension to 2 yields better results. Moreover Triangulation in two dimensions typically takes less than a second for the datasets used in our experiments. However, triangulation in higher dimensions may require a longer time.

A.1. Experiments on larger dimension

	Original Graph	**DR**	DR (dim =3)	DR (dim =4)	DR (dim =5)	DR (dim =7)
Homophily	0.06	0.58	0.54	0.48	0.42	
Number of edges	22662	135922	238366	422404	803846	
Max degree	14	21	49	124	257	
Mean degree	3	6	10	20	38	
Accuracy GCN	51.66 ±0.2	**61.99** ±0.1	61.84 ±0.2	60.87 ±0.2	58.40 ±0.2	
Accuracy GAT	49.23 ±0.3	**61.80 ±0.2**	61.59 ±0.2	60.31 ±0.2	59.49 ±0.2	
Time for triangulation (in sec)	-	≤ 1	2	12	105	

Table 9: Experimental results with Delaunay Rewiring in differents dimensions reduction Roman-Empire dataset.

	Original Graph	**DR**	DR (dim =3)	DR (dim =4)	DR (dim =5)	DR (dim =7)
Homophily	0.06	0.65	0.63	0.60	0.54	0.44
Number of edges	499	1470	2534	4064	6266	16148
Max degree	24	14	24	42	57	167
Mean degree	12	6	12	18	28	66
Accuracy GCN	55.12±1.51	**70.98±1.5**	69.45 ±1.5	68.59 ± 1.5	68.55 ±1.5	66.42 ± 1.7
Accuracy GAT	46.05 ±1.49	**74.33 ±1.24**	74.23 ± 1.4	70.75 ± 1.4	72.43±1.5	67.16 ±1.7
Time for triangulation (in sec)	-	≤ 1	≤ 1	≤ 1	2	50

Table 10: Experimental results with Delaunay Rewiring in different dimensions reduction for Wisconsin dataset.

	Original Graph	**DR**	DR (dim =3)	DR (dim =4)	DR (dim =5)	DR (dim =7)
Homophily	0.11	0.67	0.58	0.51	0.47	0.38
Number of edges	295	1064	1780	2888	4342	10406
Max degree	94	21	18	34	52	117
Mean degree	3	6	10	17	26	66
Accuracy GCN	44.78 ±1.5	**67.22 ±1.5**	64.97 ±1.6	64.27 ±1.6	62.75 ±1.6	49.19 ±1.6
Accuracy GAT	46.05 ±1.5	**68.03 ±1.6**	67.62 ±1.6	64.16 ±1.6	63.81 ±1.6	59.35 ±1.6
Time for triangulation (in sec)	-	≤ 1	≤ 1	≤ 1	≤ 1	19

Table 11: Experimental results with Delaunay Rewiring in different dimensions reduction for Cornell dataset.

	Original Graph	**DR**	DR (dim =3)	DR (dim =4)	DR (dim =5)	DR (dim =7)
Homophily	0.06	0.63	0.64	0.54	0.53	
Number of edges	181	1072	1718	2780	4172	10948
Max degree	104	10	16	33	53	123
Mean degree	3	6	10	16	25	70
Accuracy GCN	56.19 ±1.61	**70.46 ±1.61**	69.24 ±1.7	65.78 ±1.7	65.56 ±1.7	56.05 ±2.0
Accuracy GAT	56.84 ±1.61	**74.30 ±1.38**	0.7378 ±1.6	69.29 ±1.7	70.54 ±1.7	66.81 ±1.8
Time for triangulation (in sec)	-	≤ 1	≤ 1	≤ 1	≤ 1	18

Table 12: Experimental results with Delaunay Rewiring in different dimensions reduction for Texas dataset.

	Original Graph	**DR**	DR (dim =3)	DR (dim =4)	DR (dim =5)	DR (dim =7)
Homophily	0.24	0.40	0.36	0.34	0.32	0.31
Number of edges	33544	45520	81876	156058	295208	662438
Max degree	1303	16	36	123	344	
Mean degree	7	6	11	22	40	
Accuracy GCN	30.02±0.2	41.36 ±0.2	**42.59±0.3**	41.08 ±0.3	40.15 ±0.3	42.15
Accuracy GAT	29.92 ±0.2	40.25 ±0.2	**41.12 ±0.3**	40.79 ±0.3	40.43 ±0.3	
Time for triangulation (in sec)	-	≤ 1	≤ 1	4	20	856

Table 13: Experimental results with Delaunay Rewiring in different dimensions reduction for Actor dataset.

	Original Graph	**DR**	DR (dim =3)	DR (dim =4)	DR (dim =5)	DR (dim =7)
Homophily	0.25	0.69	0.67	0.60	0.52	0.38
Number of edges	36101	13630	23544	42454	78516	318376
Max degree	732	14	49	123	273	1216
Mean degree	28	6	12	21	36	142
Accuracy GCN	65.35±0.5	**74.28 ±0.5**	66.11±0.6	67.21 ±0.6	67.43 ±0.6	66.87 ±0.6
Accuracy GAT	65.07 ±0.4	**72.04 ±0.4**	74.03 ±0.5	73.21 ±0.5	70.82 ±0.5	68.71 ±0.6
Time (in sec)	≤ 1	≤ 1	≤ 1	≤ 1	5	296

Table 14: Experimental results with Delaunay Rewiring in different dimensions reduction for Chameleon dataset.

	Original Graph	**DR**	DR (dim =3)	DR (dim =4)	DR (dim =5)	DR (dim =7)
Homophily	0.22	0.59	0.57	0.54	0.49	0.43
Number of edges	217073	31170	54780	103652	199156	467702
Max degree	1905	13	55	155	637	2268
Mean degree	76	6	11	21	40	94
Accuracy GCN	51.30±0.4	**65.25 ±0.3**	65.19 ±0.5	64.42 ±0.6	63.76 ±0.6	63.49 ±0.6
Accuracy GAT	50.87 ±0.6	64.47 ±0.3	**66.26** ±0.5	64.08 ±0.5	63.28 ±0.7	63.08 ±0.6
Time (in sec)	-	≤ 1	≤ 1	2	12	399

Table 15: Experimental results with Delaunay Rewiring in different dimensions reduction for Squi dataset.

	Original Graph	**DR**	DR (dim =3)	DR (dim =4)	DR (dim =5)	DR (dim =7)
Homophily	0.83	0.88	0.85	0.78	0.71	0.51
Number of edges	5429	16214	28578	49990	87738	237164
Max degree	168	12	65	222	493	1173
Mean degree	4	6	11	20	34	96
Accuracy GCN	87.73 ±0.3	**91.39 ±0.2**	90.05 ± 0.3	89.39 ± 0.3	87.59± 0.3	83.47 ± 0.4
Accuracy GAT	87.65 ±0.2	**91.37 ±0.2**	90.21 ± 0.3	89.58 ± 0.3	87.76± 0.3	85.30 ± 0.3
Time (in sec)	-	≤ 1	≤ 1	≤ 1	5	324

Table 16: Experimental results with Delaunay Rewiring in different dimensions reduction for Cora dataset.

	Original Graph	**DR**	DR (dim =3)	DR (dim =4)	DR (dim =5)	DR (dim =7)
Homophily	0.71	0.78	0.73	0.70	0.62	0.52
Number of edges	4715	19923	32075	62828	108710	283162
Max degree	99	14	54	167	399	1138
Mean degree	3	6	11	20	35	92
Accuracy GCN	76.01 ±0.3	**81.14 ±0.3**	80.82 ±0.3	80.45 ±0.3	80.01 ±0.3	77.57 ±0.4
Accuracy GAT	76.20 ±0.3	**81.61 ±0.3**	80.86 ±0.3	80.18 ±0.3	80.66 ±0.3	78.09 ±0.4
Time (in sec)	-	≤ 1	≤ 1	≤ 1	6	306

Table 17: Experimental results with Delaunay Rewiring in different dimensions reduction for Citeseer dataset.

B. Effects of Delaunay triangulation on the degree of graphs.

In this section, we show the degree distribution for the datasets used in the experiments. For Delaunay graphs, the degrees generally range between 3 and 15. Conversely, for the original graphs, degrees vary between very low (1) and very high values, such as 100 for a master node. As indicated by (Deac et al., 2022) , nodes with very high degree, like master nodes, create a bottleneck, exacerbating the phenomenon of oversquashing.

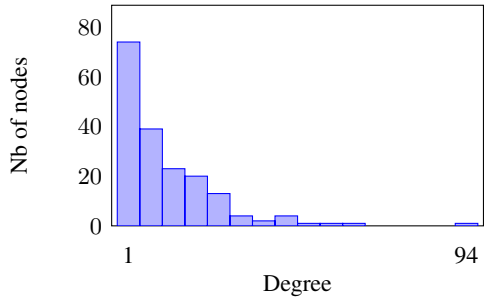


Figure 4: Histogram of the degree distribution for the original Cornell graph

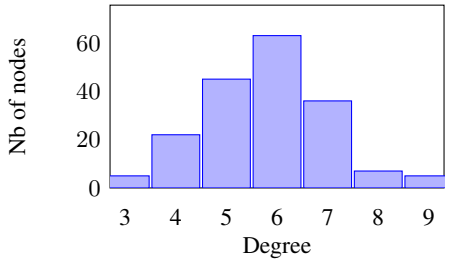


Figure 5: Histogram of the degree distribution for the Delaunay Cornell graph

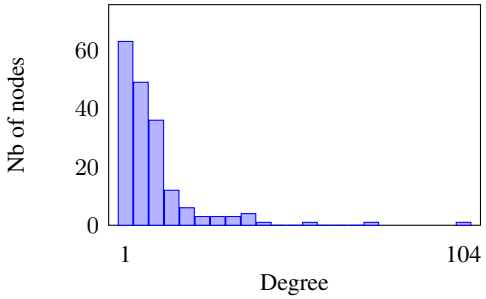


Figure 6: Histogram of the degree distribution for the original Texas graph

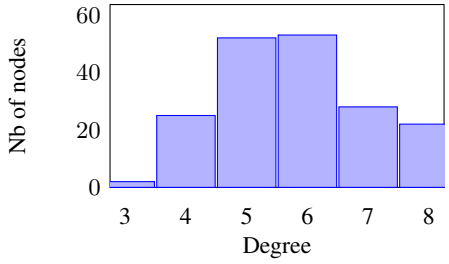


Figure 7: Histogram of the degree distribution for the Deauney Texas graph

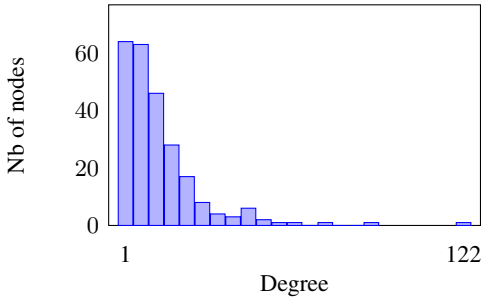


Figure 8: Histogram of the degree distribution for the original Wisconsin graph

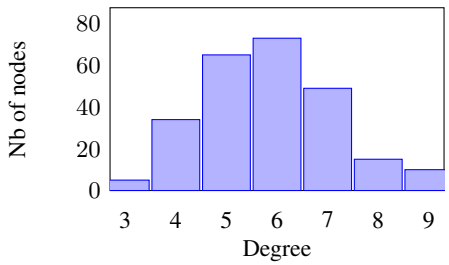


Figure 9: Histogram of the degree distribution for the Delaunay Wisconsin graph

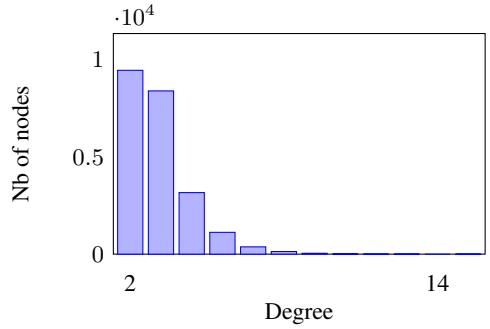


Figure 10: Histogram of the degree distribution for the original Roman-Empire graph

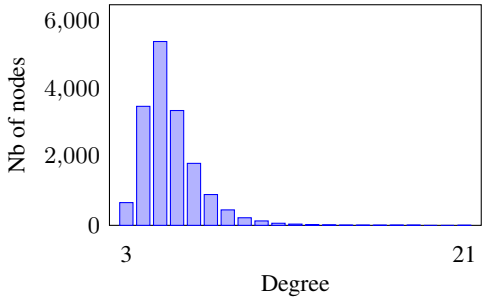


Figure 11: Histogram of the degree distribution for the Delaunay Roman-Empire

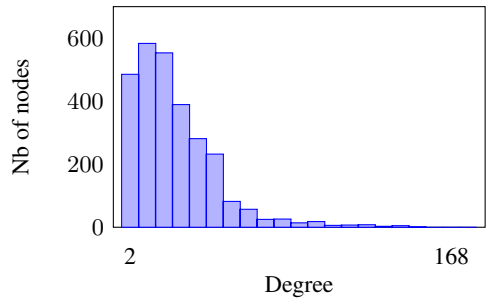


Figure 12: Histogram of the degree distribution for the original Cora graph

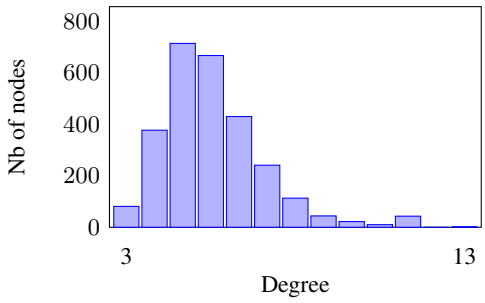


Figure 13: Histogram of the degree distribution for the Delaunay Cora graph

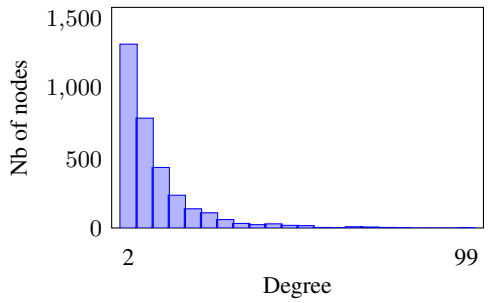


Figure 14: Histogram of the degree distribution for the original Citeseer graph

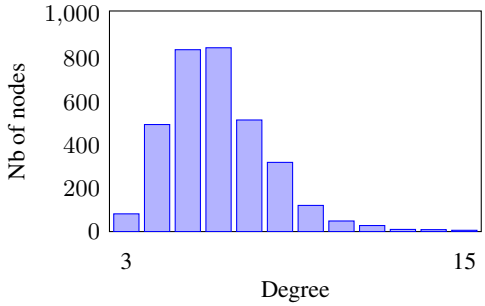


Figure 15: Histogram of the degree distribution for the original Citeseer graph

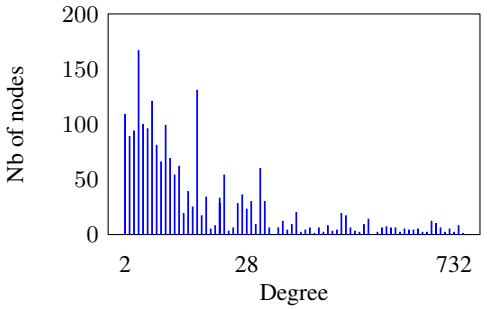


Figure 16: Histogram of the degree distribution for the original chameleon graph

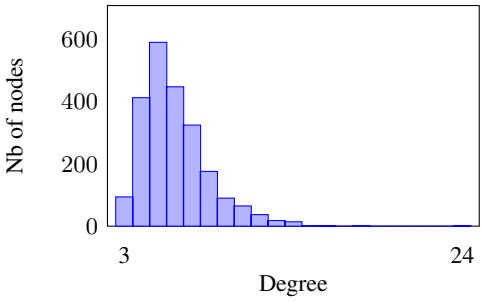


Figure 17: Histogram of the degree distribution for the Delaunay chameleon graph

C. Details of datasets

We detail the content of the 7 heterophilic datasets and 3 homophilic datasets:

- **WebKB**: The WebKB dataset encompasses data from Cornell, Texas, and Wisconsin, where nodes symbolize web pages connected by hyperlinks. Node features are represented by bag-of-words representations of the web pages, while labels include student, project, course, staff, and faculty.
- **Actor**: Actor nodes signify individual actors, with edges denoting co-occurrences on Wikipedia pages (Tang et al., 2009). Node features correspond to keywords found within Wikipedia pages, with a requirement to classify nodes into five distinct classes.
- **Wikipedia network**: Within the Squirrel and Chameleon datasets, nodes depict web pages interconnected by mutual links (Rozemberczki et al., 2021). Node features are determined by informative nouns extracted from Wikipedia pages. Nodes are to be categorized into five distinct classes based on their popularity, gauged by the average monthly traffic to the web page.
- **Roman-empire**: Based on the Roman Empire article from English Wikipedia, this dataset is notable for being one of the longest articles on Wikipedia (Platonov et al., 2023). Each node represents a word in the text, with edges connecting words that follow each other in the text or are linked in the sentence's dependency tree. Node classification is determined by the syntactic role of each word.
- **Scientific publication networks**: Cora, Citeseer, Pubmed datasets (Sen et al., 2008) document citations to scientific publications. Each publication is characterized by a bag-of-words representation indicating the presence or absence of words in the publication abstract. Classes represent categories of publications.

D. Experiments details

We show here all the results on the datasets using a triangulation on the original features (OF) and on the learned features (LF) on all datasets;

	GCN (O)	GCN + DG$_{OF}$	GCN + DG$_{LF}$
Chameleon	65.35±0.54	27.82 ±0.48	**74.28** ±0.48
Squirrel	51.30±0.38	22.73 ±0.29	**65.25** ±0.26
Actor	30.02 ±0.22	30.73 ±0.26	**41.36** ±0.20
Texas	56.19 ±1.61	**70.68** ±1.60	67.38 ±1.50
Wisconsin	55.12±1.51	**70.98** ±1.50	63.63 ±1.37
Cornell	44.78 ±1.45	**67.22** ±1.48	61.01 ±1.48
Romain-Empire	51.66 ±0.17	**61.99** ±0.14	50.47 ±0.12
Cora	87.73 ±0.25	66.78 ±0.37	**91.39** ±0.24
Citeseer	76.20 ±0.27	32.56 ±0.41	**81.61** ±0.25
Pubmed	87.39 ±0.11	75.27 ±0.11	**89.14** ±0.09

Table 18: Experimental results. O denotes GCN on the original input graph, DG_{OF} denotes the GCN on the Delaunay Graph based on the Original Feature and DG_{LF} denotes Delaunay Graph based on the Learned Feature. Best score in bold.

On some datasets we note that triangulation on original features gives better results than features learned by a GCN. This may be due to the poor performance of the GCN on these datasets and the variability of the results.