Understanding Sparsification in Graph Neural Networks

Ilias Laoukili

October 20, 2025

Abstract

This article introduces sparsification as a geometric restructuring technique for Graph Neural Networks, motivated by the role of graph curvature and message-passing bottlenecks. Rather than treating sparsification as a compression method, we frame it as a topology-level intervention that modifies information flow and inductive bias. This work establishes the theoretical foundations for a broader research program on curvature-aware graph reparameterization.

1 Introduction

Graphs provide a natural and expressive way to model structured relationships in a wide range of domains, from molecules and biological networks to transportation systems, knowledge graphs, and social interactions. The recent surge of interest in Graph Neural Networks (GNNs) stems from their ability to learn meaningful representations directly from graph-structured data without relying on handcrafted features or explicit feature engineering. By operating on nodes, edges, and neighborhoods, GNNs generalize the success of deep learning to non-Euclidean domains.

However, the very property that makes GNNs powerful—their ability to propagate and aggregate information across graph neighborhoods—also introduces fundamental challenges. As message passing layers deepen, the number of involved edges grows combinatorially, increasing computational cost, memory overhead, and training instability. This results in several well-documented bottlenecks such as *over-smoothing*, where node features become indistinguishable, and *over-squashing*, where exponential amounts of distant information are forced through a limited number of edges. These issues are not merely implementation artefacts; they arise from intrinsic geometric and topological properties of the underlying graph.

The problem becomes particularly acute in real-world graphs that are large, heterogeneous, and highly irregular. In such settings, dense message passing does not necessarily correspond to useful learning. Many edges are redundant for representation learning, while a much smaller subset carries the informative structure required for prediction. This observation motivates a fundamental shift in how we design graph learning architectures: rather than focusing solely on "more expressivity" through deeper or more complex aggregation mechanisms, one may instead *restructure* the graph so that the signal of interest flows more efficiently.

This is where sparsification enters the picture. Unlike pruning techniques in classical deep learning, sparsification is not merely about reducing model size or computation. It is about reshaping the graph's geometry to improve information flow, mitigate curvature-induced bottlenecks, and highlight structural dependencies that matter for downstream inference. A well-sparsified graph can preserve—or even enhance—learning performance while significantly reducing topological and computational complexity.

The purpose of this article is to provide a principled introduction to sparsification in the context of Graph Neural Networks, bridging the geometric insights developed in the Distill publications on message passing and GNN expressivity with emerging work on efficient graph reparameterization. Rather than approaching sparsification as a heuristic or post-processing technique, we position it as a first-class architectural tool that addresses the topological origins of GNN bottlenecks.

This introductory post lays the foundation for the research direction explored in the associated project, which focuses on understanding *when*, *where*, and *how* sparsification improves GNN expressivity. Future posts will build on this foundation, covering theoretical perspectives (graph curvature, bottlenecks, and contraction), algorithmic strategies for structural and topological sparsification, and empirical validation on representative benchmarks.

2 Background

2.1 Graphs and Notation

We consider a graph $G = (V, E)$ where V denotes the set of nodes with $|V| = n$ and $E \subseteq V \times V$ the set of edges with $|E| = m$. Each node $v \in V$ may be associated with a feature vector $x_v \in \mathbb{R}^d$, and the structure of the graph is represented through the adjacency relation A. For each node v, the neighborhood is defined by

$$\mathcal{N}(v) = \{u \in V : (u, v) \in E\}.$$

Learning on graphs exploits this relational structure as an *inductive bias*: the topology of G constrains which information can flow where and at what "speed" across successive GNN layers.

In practical applications, graphs may be large, irregular, and heterogeneous. Some nodes have dozens or hundreds of neighbors, whereas others are nearly isolated. This heterogeneity means that $\mathcal{N}(v)$ is not only a local descriptor of structure but also determines the fidelity with which information from the graph can be represented, propagated, and aggregated.

2.2 Message Passing Neural Networks

Graph Neural Networks (GNNs) can be understood through the message passing paradigm. Each layer consists of two conceptual steps: (1) the computation of messages from neighbors, and (2) the aggregation and update of node embeddings. The general formulation of a message passing layer is:

$$h_v^{(k+1)} = \mathrm{UPDATE}^{(k)}\Big(h_v^{(k)},\ \mathrm{AGGREGATE}^{(k)}(\{\mathrm{MESSAGE}^{(k)}(h_v^{(k)}, h_u^{(k)}, e_{uv}) : u \in \mathcal{N}(v)\})\Big),$$

where $h_v^{(k)}$ is the representation of node v at layer k, and e_{uv} may encode edge features.

This framework captures a broad family of architectures. Intuitively, each node "asks" its neighbors for some representation of their state, combines those responses, and updates its embedding accordingly. Repeating this process over multiple layers allows information to propagate further in the graph, giving rise to a growing receptive field. However, this also implies growing dependency on the connectivity pattern of the graph itself.

2.3 Representative GNN Architectures

Several widely used architectures instantiate the general message passing scheme using different aggregation mechanisms:

- **GCN (Graph Convolutional Network):** uses a normalized averaging operator over neighbors, which smooths node features along the graph structure. Computation can be written as
$$H^{(k+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(k)}W^{(k)}).$$
This highlights a diffusion-like interpretation.

- **GraphSAGE:** aggregates a *sampled* neighborhood using mean, max-pooling, or LSTM-based operators, enabling scalability to large graphs by controlling computational cost.

- **GAT (Graph Attention Network):** assigns learned weights to edges through an attention mechanism, allowing the model to differentiate between neighbors based on feature relevance.

Although their aggregation differs, these models share a reliance on the underlying graph topology: the expressive power and stability of the learned representation depend on the structure of $\mathcal{N}(v)$ and how information flows through E.

2.4 Expressivity and Connectivity

The expressivity of message passing increases with the ability of a node to absorb informative signals from distant parts of the graph. However, information must travel along edges, and this imposes a structural constraint: distant information is compressed into a limited number of communication paths. This leads to two distinct regimes of degradation:

- *over-smoothing*, where repeated aggregation causes node states to converge toward a low-rank subspace, reducing discriminative power;

- *over-squashing*, where many distant signals are forced through too few structural "channels", causing information bottlenecks.

The key insight is that these limitations do not arise only from model design choices or insufficient training—they are rooted in the topology of the graph itself. Highly clustered or "negatively curved" regions create bottlenecks in which exponentially many node interactions are funneled through few edges. This geometric perspective foreshadows the role of graph curvature, which we will revisit in the next section as a structural explanation for why sparsification can improve GNN performance by modifying the graph rather than the model itself.

3 Bottlenecks in Graph Neural Networks

3.1 Oversmoothing

Oversmoothing refers to the phenomenon where node embeddings become progressively indistinguishable as the number of GNN layers increases. In architectures based on neighborhood averaging, such as GCNs, each propagation step acts as a low-pass filter with respect to the graph Laplacian. After repeated applications, signals converge toward the principal eigenspaces of the Laplacian, which have limited discriminative capacity.

Formally, for a normalized adjacency operator $\tilde{A}$,

$$H^{(k)} \approx \tilde{A}^k H^{(0)}.$$

As $k \to \infty$, $\tilde{A}^k$ tends toward a projection that contracts variation across connected components, driving $h_v^{(k)} \approx h_u^{(k)}$ for many nodes u, v. In this regime, structural information becomes homogenized: deep GNNs do not fail because "too much" context is incorporated, but because information is *washed out* into a low-rank subspace. This is an inherently spectral limitation of diffusion-based aggregation.

3.2 Oversquashing

Oversquashing is a distinct bottleneck arising when a node must compress exponentially many distant signals into a fixed-size representation through a small number of connecting edges. Unlike oversmoothing, which concerns loss of discriminative variance, oversquashing concerns the *capacity of information flow* through the graph.

To illustrate this effect, consider a rooted tree-like expansion around a target node v. Suppose the branching factor is b; then the number of nodes at distance k from v is $O(b^k)$. Yet the "surface" through which this information must enter the 1-hop neighborhood of v grows only linearly with b. Thus, exponentially many signals must be funneled through the handful of edges $\partial B_k(v)$ that separate v from its k-hop neighborhood. When k grows, the mismatch between the volume b^k and the boundary size $O(b)$ creates congestion: the representation at v cannot encode all relevant distant information.

More generally, oversquashing occurs whenever the graph exhibits a rapidly expanding neighborhood volume but a narrow interface through which messages must propagate. This is not due to underparameterization or missing attention coefficients—rather, it is a topological limitation imposed by graph connectivity itself. Increasing the number of GNN layers does not fix the problem; it exacerbates it, because distant nodes contribute increasingly noisy and compressed signals.

3.3 Topological Origin of Bottlenecks

Both oversmoothing and oversquashing originate in structural constraints of the graph topology. Oversmoothing reflects the contraction induced by repeated diffusion over a graph, while oversquashing reflects bottlenecks in the combinatorial structure of neighborhoods. These effects persist even when modern architectural enhancements are added:

- attention mechanisms cannot create new communication channels;
- residual connections do not change structural boundary growth;
- deeper receptive fields worsen congestion rather than alleviating it.

Thus, the limitations stem not from the design of the GNN, but from the geometry of the space in which message passing unfolds.

3.4 Curvature as a Lens on Bottlenecks

A more principled way to understand oversquashing is through graph curvature, and in particular through discrete Ricci curvature. Regions of a graph with negative Ricci curvature (loosely corresponding to "hyperbolic" expansion) amplify the mismatch between neighborhood volume and boundary flow, making message-passing bottlenecks inevitable. Intuitively, negatively curved regions create structural choke points that force many distant signals through few edges, directly causing oversquashing.

This geometric interpretation is central: it reveals that alleviating GNN bottlenecks is not merely a matter of improving architectures, but may instead require *modifying the graph itself*. This motivates sparsification not as a compression technique, but as a *geometric intervention* aimed at reshaping connectivity to improve information flow—an idea developed in the next section.

4 Sparsification as a Geometric Solution

4.1 Why Modify the Graph Rather than the Model

The bottlenecks discussed in the previous section arise not from insufficient model capacity, but from structural constraints inherent to the graph itself. Oversmoothing reflects repeated diffusion through a fixed topology, and oversquashing reflects insufficient boundary growth relative to neighborhood volume. In both cases, message passing is limited not by how embeddings are processed, but by how information is able to move across the graph.

This perspective implies that architectural improvements alone cannot resolve these issues. Attention layers can rescale signals, and residual connections can preserve intermediate states, but neither changes the underlying connectivity of the graph. Information must still flow through the same structural choke points. Thus, a more fundamental intervention targets the topology itself: if the geometry of the graph is responsible for restricting information, then modifying that geometry becomes a natural path to restoring expressive power.

4.2 Sparsification as Geometric Restructuring

In this context, sparsification does not mean "deleting edges" for the sake of efficiency. Instead, sparsification refers to the intentional reshaping of graph connectivity to improve information flow. The goal is not to make the graph smaller, but *better*: to restructure it so that message passing reflects meaningful signal pathways rather than raw adjacency.

This distinction is crucial. Classical pruning removes edges based on heuristics such as degree, weight, or redundancy. Geometric sparsification, by contrast, is guided by an understanding of how curvature, congestion, or expansion properties influence the propagation of information. The objective is not simplification but *reparameterization*: constructing a functionally equivalent graph with reduced bottlenecks.

Rather than asking *"Which edges can we safely remove?"*, geometric sparsification asks:

"Which edges contribute to efficient information flow, and how should the graph be reorganized to emphas

This reframing turns sparsification from a compression tool into a representational transformation.

4.3 Structural vs. Topological Sparsification

There are two complementary ways to interpret what sparsification modifies in a graph:

- **Structural sparsification** preserves local patterns while reducing redundant or low-utility edges. It treats sparsification as a refinement of adjacency with respect to useful signal structure.
- **Topological sparsification** reconfigures connectivity at a more global level by altering the graph's large-scale organization, potentially modifying shortcuts, detours, and bottlenecks in the communication structure.

Structural sparsification can be understood as improving the *local* inductive bias of the graph, whereas topological sparsification improves its *global* geometric capacity to propagate information. The latter is directly relevant for oversquashing, which is caused by a mismatch between expansion and interface size at large radii. In practice, both perspectives interact: reshaping local neighborhoods often improves global geometry, and conversely, controlling global bottlenecks clarifies local structure.

4.4 Curvature-Guided Sparsification

A geometric interpretation clarifies why sparsification can alleviate bottlenecks: regions of negative discrete Ricci curvature correspond to areas where volume grows faster than edge boundary size, creating information congestion. In such regions, oversquashing is not accidental but inevitable, because the topology forces exponentially many upstream signals through a small number of structural interfaces.

Ricci curvature provides a quantitative way to diagnose where such bottlenecks occur. Edges with highly negative curvature typically connect regions that expand rapidly, indicating overloaded communication channels. Conversely, edges with positive curvature often correspond to redundant or low-information pathways, where multiple parallel edges convey similar signals. From this perspective, sparsification is a curvature-balancing transformation: it suppresses redundant positive-curvature edges and restructures negatively curved regions to widen or redistribute flow.

The goal is therefore not to reduce connectivity, but to *optimize* it—to sculpt the graph into a geometry better aligned with the actual flow of information required by downstream tasks.

4.5 Algorithmic Preview: From Curvature to Reparameterization

Because curvature links structural topology with information flow, it also provides a guiding principle for algorithm design. Early sparsification methods stabilized training by reducing edge count, but modern approaches leverage curvature to selectively reorganize connectivity:

- some methods *remove or attenuate* overly redundant edges (positive curvature),
- others *rewire* the graph to relieve bottlenecks in negatively curved regions,
- and more advanced approaches apply *Ricci flow*, iteratively modifying edge weights or structure to balance curvature across the graph.

This places sparsification inside a broader family of *geometric reparameterization techniques*: rather than optimizing the GNN alone, one optimizes the space in which the GNN learns. Sparsification thus becomes a form of architectural preconditioning: by modifying curvature, it creates a more learnable graph geometry before message passing begins.

In the context of this project, sparsification will be studied not as a heuristic reduction procedure but as a geometric intervention motivated by graph curvature. The central research question is therefore not simply *"how to sparsify"* but rather:

"how should a graph be restructured so that its geometry improves information flow for learning?"

This shifts sparsification from a model-side technique to a topology-side transformation, establishing the foundation for curvature-aware graph refinement explored in the next stages of this work.

5 Research Framework

5.1 Problem Scope and Objectives

The objective of this project is to study sparsification not as a compression heuristic, but as a geometric intervention that modifies the topology of a graph to improve the flow of information in Graph Neural Networks. Rather than asking whether sparsification reduces computational cost, the central question is *when* and *under which geometric conditions* sparsification improves expressivity, stability, and learning performance.

This perspective shifts the focus from architecture-driven improvements to geometry-driven restructuring. The goal is not to design a new GNN, but to understand how changing the graph itself affects the inductive bias and communication structure experienced by message passing networks.

5.2 What is Evaluated When Sparsifying

Studying sparsification as a geometric transformation requires evaluating more than the final predictive performance of a model. The relevant quantities include:

- **topological quality**, such as the presence or reduction of bottlenecks;

- **geometric alignment**, i.e., whether the modified graph better reflects the task-relevant signal structure;
- **message passing efficiency**, meaning the ability of information to propagate without collapse or congestion;
- **stability of representations**, particularly with respect to oversmoothing or over-squashing.

These criteria treat sparsification as a change in representational geometry rather than as an architectural or algorithmic trick.

5.3 Methodological Roadmap

Because this work concerns graph geometry rather than model engineering, the methodological approach is conceptually comparative rather than architecturally incremental. The investigation proceeds along three axes:

1. **Characterization:** identify regions of a graph where curvature indicates communication bottlenecks or redundancy;
2. **Transformation:** apply sparsification strategies that restructure connectivity to relieve these bottlenecks;
3. **Evaluation:** measure changes in information flow, curvature distribution, and downstream learning behavior.

This roadmap frames sparsification as a reparameterization problem: we do not attempt to "fix" the model, but to place the model in a more learnable geometric space. As such, the methodology is iterative at the graph level rather than the architectural level.

5.4 Toward Future Work

This introductory post establishes the theoretical motivation and research direction for the year-long investigation conducted under the Tremplin Recherche program. Subsequent stages will deepen this framework by:

- introducing curvature-based diagnostic tools for identifying bottlenecks,
- comparing different families of sparsification techniques,
- and eventually integrating curvature-guided rewiring into empirical evaluation.

The next article in this series will focus on how discrete Ricci curvature is computed in practice, and why it provides a meaningful proxy for information flow in message-passing networks. This will bridge the gap between geometric theory and implementable sparsification strategies.

6 Conclusion

This article introduced the motivation for studying graph sparsification from a geometric perspective rather than as a compression technique. We reviewed how message passing in Graph Neural Networks is tightly constrained by the topology of the graph, and how these constraints produce oversmoothing and oversquashing phenomena that cannot be resolved by architectural modifications alone. The key observation is that these bottlenecks originate from the graph structure itself: improving learning therefore requires improving the geometry through which information flows.

Framing sparsification as geometric restructuring rather than edge removal opens a broader research direction. Instead of asking how to reduce graph size, the relevant question becomes how to reshape the graph to alleviate structural bottlenecks and improve the inductive bias available to GNNs. This establishes the conceptual foundation for the research program developed in the following work, where sparsification will be analyzed as a topology-aware and curvature-informed transformation of graph structure.

7 Future Work

This introduction establishes the motivation for treating sparsification as a geometric restructuring problem rather than a compression step. The next stages of this research will build on this foundation by making the link between curvature, graph topology, and learnability increasingly explicit. The year-long investigation will proceed along three complementary axes.

7.1 Theoretical Development

The next step is to formalize the connection between discrete Ricci curvature and information flow in message passing. This involves examining how curvature quantifies congestion, and how curvature imbalances identify regions of the graph where bottlenecks arise. Multiple curvature notions (e.g., Ollivier-type transport curvature and Forman-type combinatorial curvature) will be compared to determine which most accurately predicts GNN degradation in practice.

7.2 Methodological Development

Once a curvature-based diagnostic perspective is established, sparsification strategies can be analyzed as geometric transformations. Future work will investigate families of approaches ranging from structural filtering to rewiring and curvature-balancing procedures. The emphasis will be on understanding how different sparsification regimes alter the effective graph geometry rather than on optimizing any particular algorithm.

7.3 Empirical Evaluation

The final stage consists of validating these geometric effects in learning settings. This will involve controlled experiments on representative graph families where curvature and expan-

sion properties can be measured before training, as well as on benchmark datasets used in graph representation learning. The objective is not to achieve state-of-the-art performance, but to evaluate whether improvements in graph geometry correlate with improvements in information flow and downstream learning behavior.

References

[1] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2017.

[3] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2018.

[5] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "A gentle introduction to graph neural networks," *arXiv preprint arXiv:2104.13478*, 2021. Available at Distill: https://distill.pub/2021/gnn-intro/.

[6] U. Alon and E. Yahav, "Understanding oversquashing and bottlenecks in gnns," *arXiv preprint arXiv:2106.05237*, 2021. Available at Distill: https://distill.pub/2021/understanding-gnns/.

[7] Y. Ollivier, "Ricci curvature of markov chains on metric spaces," *Journal of Functional Analysis*, vol. 256, no. 3, pp. 810–864, 2009.

[8] R. Forman, "Bochner's method for cell complexes and combinatorial ricci curvature," *Discrete & Computational Geometry*, vol. 29, pp. 323–374, 2003.