Mitigation of Over-Squashing in Graph Neural Networks

Nasib Huseynzade

· *huseynzade@uni-hildesheim.de*
· *University of Hildesheim, 31141 Hildesheim, Germany*

Publication date: 22/03/2025

Abstract— In a variety of graph-based learning tasks, Graph Neural Networks (GNNs) have demonstrated impressive performance. They do, however, have the over-squashing issue, which results in poor model performance and inefficient message delivery as long-range dependencies are compressed through low-degree nodes. Numerous rewiring strategies, such as spectral approaches, curvature-based methods, and stochastic rewiring, have been put forth to address this problem. The state-of-the-art approach used in this work is the Stochastic Discrete Ricci Flow (SDRF) method, which we improve upon by implementing commute time metric. The commute time metric is used as a termination condition for the rewiring process as well as a decision mechanism for choosing the best edges to rewire. Experimental studies look into how other methods, such as increasing depth, increasing width, and using fully connected graph, affect over-squashing.Our results demonstrate that commute time based SDRF significantly alleviates over-squashing, improving GNN performance in long-range tasks while providing insights into the effectiveness of alternative mitigation approaches. Code is available at https://github.com/nasibhuseynzade/Over-squashing-in-GNNs

Keywords— Graph Neural Networks, Over-squashing, Commute time, Curvature

I. Introduction

Graph Neural Networks (GNNs) have arisen as prosperous deep learning architectures to learn graph-structured data, where traditional shallow machine learning methods struggle with capturing richer relational dependencies. In comparison to your typical neural net models that operate on Euclidean data, such as images or text, GNNs can utilize message passing to learn representations of the nodes, edges, and graph itself by aggregating information from neighboring nodes. This unique ability provides great power for performing node classification, link prediction, and graph classification tasks in various application domains, including social networks, molecular chemistry, and recommendation systems. Since that initial work, numerous related GNN variants have been proposed to improve expressiveness and account for problems faced by classical message passing architectures, like over-smoothing and over-squashing. [15]

Graph Neural Networks (GNNs) are a relevant category of models focusing on learning from graph-based information. Of the various GNN architectures, Graph Convolutional Networks (GCNs), Graph Isomorphism Networks (GINs), and Graph Attention Networks (GATs) are popular due to their unique methods of information aggregation and representation learning within graphs. This comparison looks at the architecture, advantages and disadvantages of each model, and the challenges of over-smoothing and over-squashing over layers. Graph Convolutional Networks (GCNs) also use a neighborhood aggregation scheme, where a node's representation is updated through averaging its immediate neighbor's representation. This layer-wise propagation allows GNNs to learn local graph structures well. However, the primary disadvantage to this approach is over-smoothing, where a node's distinguishability decreases as layers increase [16]. In other words, as more layers are added, it is possible for the node's representation to be similar to all of its neighbors leading to distinguishability due to the embeddings becoming similar. In order to address this, GNNs can be equipped with techniques, such as skip connections or layer normalization, which maintain discriminative feature representations, through the layers of architecture [17] [15].

a. The Challenge of Over-Squashing

A key challenge of graph neural networks(GNNs) is that they lose important information from compressing or "squashing" the data as it passes through several layers before arriving at the target node. This issue is most painful for deep GNNs, where nodes must aggregate and process signals from distant areas of the graph, resulting in lost crucial information and diminished model performance. [14] This experience is strongly felt in domains such as knowledge graph reasoning, molecular property prediction, and recommendation systems that rely on long-range dependencies. In these applications, accurately predicting the future is often contingent on capturing relationships across distant nodes. However, severed relationships due to oversquashing means we cannot maintain meaningful relationships and instead serve suboptimal

performance. [14]

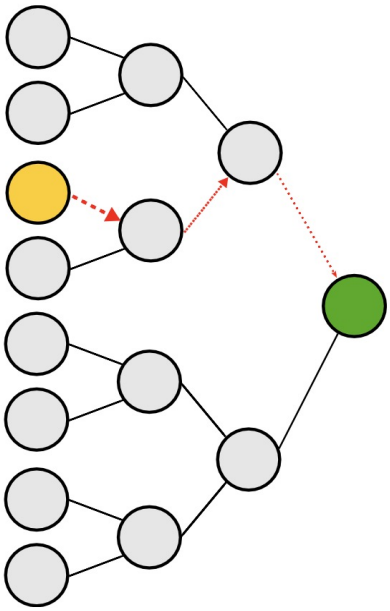

Fig. 1: Over-Squashing

It is vital to address oversquashing for the improved applicability of GNNs in real-world applications. Several methods have been introduced to address this issue, such as adjusting the architecture of the network, changing model hyperparameters, and applying graph rewiring methods. GNNs will be ineffective without these changes to address this issue and will continue to be limited in generalization across complex large-scale graphs, which dramatically limits the impact of these systems in practical applications. Therefore, understanding and addressing oversquashing is a major focus of research into the future of machine learning. [7] [14]

II. Related Work

The problem of over-squashing in GNNs has led to the creation of several approaches, all of which try to mitigate this issue. Akansha [6] broadly divided them into three kinds, each of which provides a different perspective on how to solve the over-squashing problem.

1. **Graph Rewiring and Curvature-Based Methods**
   Graph rewiring, which adds or suppresses edges in the graph to create a new graph with the same nodes and node features but a different collection of edges, could be used to mitigate over-squashing. In order to increase information flow between components and lower the possibility of features becoming over-squashed, the rewiring aims to better support the bottleneck and provide alternate routes of access between components. To regulate the graph's size, edges that don't significantly affect the information flow can be removed. Figure 1 displays an illustration of a over-squashing in graphs [4]

   In this area, highly efficient rewiring techniques have been created and are still developing, pushing the limits of graph optimization. Curvature-based methods are important among them due to their mathematical foundations and impact. The most significant of these algorithms can be outlined as follows:

   - **+FA (Fully Adjacent Last Layer)**
     This graph rewiring method was presented by Alon and Yahav [2] incorporating a Fully-Adjacent (FA) layer in the final GNN layer to help alleviate over-squashing. It is the simplest and based methods in the literature which improves long-range information propagation by introducing dense connectivity in the final layer. Also, it retains the original sparse structure in earlier layers and preserving locality. The drawback if this method was that, excessive edge additions may lead to over-smoothing. [10]
   - **SDRF**
     Further, Topping et al. [11] and Di Giovanni et al. [5] considered over-squashing from a topological standpoints and the implications it has for GNN design. Topping et al. developed a curvature-based rewiring procedure using Ricci flow curvature, which included reinforcing negatively-curved edges while removing positively-curved edges to allow for better multi-hop information aggregation.
   - **BORF**
     BORF shares some similarities to SDRF, but with notable differences. Since SDRF is based on BFC, it can only accurately enforce a lower bound on the Ollivier-Ricci curvature. Furthermore, its design lacks the capability to be used as a net edge minus rewiring algorithm. As such, it is ill-equipped to deal with the over-smoothing issue or to be used on denser graphs. Another difference is BORF calculates the graph curvature very infrequently, while SDRF has to constantly recalculate for each possible new edge. Finally, by rewiring edges in batch, BORF affects a uniform change across the graph. This helps to preserve the graph topology and prevent the possibility that a small outlier subgraph gets continually rewired, while other parts of the graph do not see any geometrical improvement. [13]

2. **Graph Spectral-Based Methods**

   By iteratively adding edges that maximize the increase in the spectral gap, this method aims to enhance the spectral properties of the graph, improve connectivity, and reduce over-squashing in GNNs. [**?**] The spectral gap of the graph is defined as the difference between the first and second eigenvalues of the graph Laplacian matrix. Specifically, let L be the Laplacian matrix of the graph G, and λ_1 and λ_2 be the first and second eigenvalues of L before edge addition. For each potential edge

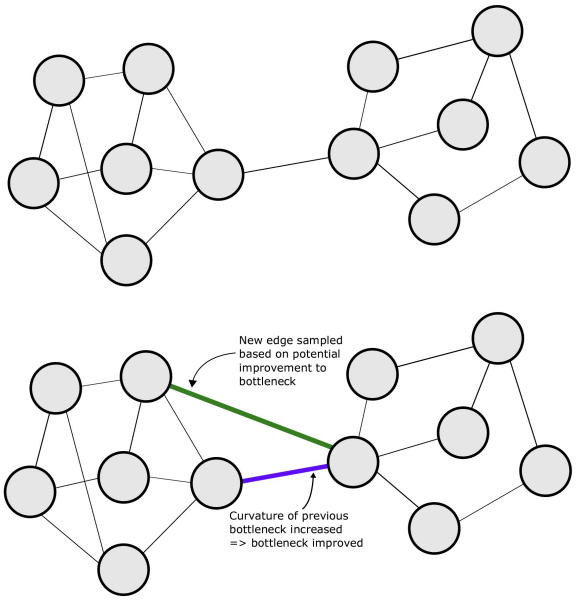

Fig. 2: Rewiring process

$e = (u, v)$ that can be added to the graph, compute the change in the spectral gap $\Delta\lambda_e$ as:

$$\Delta\lambda_e = \lambda_{2e} - \lambda_{1e} - (\lambda_2 - \lambda_1) \tag{1}$$

Then, by selecting the edge e^* that maximizes the increase in the spectral gap it is possible to reduce over-squashing:

$$e^* = \arg\max_e \Delta\lambda_e \tag{2}$$

After adding the selected edge e^* to the graph G, a new graph $G' = (V, E \cup \{e^*\})$ is created and the Laplacian matrix L' of the new graph is updated. This process is repeated for a specified number of iterations or until a convergence criterion is met.

However, one of the limitations of the Spectral expansion optimization techniques such as FOSR is that they may be tailored to specific graph structures or datasets, limiting their generalizability across diverse graph types. A possible research area would be to explore multi-objective optimization frameworks that consider spectral expansion alongside other graph properties, such as robustness, interpretability, and fairness, to achieve a more balanced and comprehensive graph representation. [12]

3. **Graph Transformers**
Graph transformers present a novel approach to tackle the issues of over-smoothing and over-squashing by leveraging the inherent strengths of transformer architectures. Graph Transformers extend the Transformer architecture, widely used for processing sequential data, to graph-structured data. Transformers are based on self-attention mechanisms that enable them to weigh the importance of different parts of the input data differently. Given a node v, the attention mechanism considers only the neighbors N(v) of v as defined by the graph's edges. The attention scores are calculated based on the features of the node and its neighbors. Mathematically, the attention from node i to node j in a graph can be expressed as:

$$A_{ij} = softmax(\frac{e_{ij}}{\sqrt{d_k}}) \tag{3}$$

where e_{ij} is a function of the features of nodes i and j, and of the edge between them, while d_k is the dimensionality of the key vectors, ensuring the dot products don't grow too large. After computing the attention scores, the Graph Transformer aggregates information from a node's neighbors (including possibly itself) to update its representation. This can be formulated as:

$$H'_v = \sum_{j \in N(v) \cup (v)} A_{ij} \cdot W \cdot H_j \tag{4}$$

where H_j is the feature vector of node j, W is a learnable weight matrix, and H'_v is the updated feature vector of node v.

Even though graph transformers solve the problem of over-squashing, computation cost is a main limitation to using this approach. Computational complexity is $O(N^3)$ [1] considering all eigenvectors. Reducing computational complexity to linear or logarithmic is a growing research area, as graph transformers present an alternative to message-passing, achieving higher performance with some costs. [1]

Our Contribution: We summarize our main contributions:

- We integrate a well-established metric from the literature into the state-of-the-art SDRF method, elevating it to serve as both a robust decision mechanism and precise threshold value.
- We conduct extensive experimentation on the QM9 and ZINC datasets—substantially larger and more complex than conventional benchmark datasets—providing more rigorous validation of our approach.
- We perform comprehensive comparative analysis of various architectural enhancements, including increased network depth and width, as well as fully-connected graph topologies, demonstrating the relative efficacy of each modification.

III. Problem Formulation

Let $G = (V, E)$ be a graph with node set V and edge set E. A GNN updates node embeddings through L layers of message passing:

$$h_v^{(l)} = \sigma\left(W^{(l)} \sum_{u \in \mathcal{N}(v)} f(h_u^{(l-1)}, h_v^{(l-1)}, e_{uv})\right) \tag{5}$$

where:

- $h_v^{(l)}$ is the embedding of node v at layer l.
- $W^{(l)}$ is the weight matrix at layer l.
- $\mathcal{N}(v)$ is the set of neighbors of node v.
- $f(\cdot)$ is the message aggregation function.
- e_{uv} represents edge features.
- $\sigma(\cdot)$ is a non-linearity (e.g., ReLU).

As the number of layers increases, nodes accumulate messages from exponentially growing neighborhoods. However, due to the fixed-size representation $h_v^{(l)}$, the information from distant nodes is compressed (squashed) into a small vector, leading to a bottleneck in learning long-range dependencies.

Given a graph G, let v be a node that requires information from a distant node u through L message-passing steps. Over-squashing can be mathematically characterized by analyzing the Jacobian of the node representation with respect to input features:

$$J_v = \frac{\partial h_v^{(L)}}{\partial x_u} \tag{6}$$

where x_u is the feature of node u. Over-squashing occurs when: $\|J_v\| \to 0$.[4]

IV. METHODOLOGY

a. State-of-art method

During each round, the preprocessing algorithm will modify the graph by adding an edge to preferentially support the most (negatively) curved edge present, while removing the maximum (positively) curved edge. The newly - added edge is selected specifically to retain the local nature of the modification by either adding an additional 3-cycle or an additional 4-cycle around the (negatively) curved edge. The overall structural difference between the original graph and the global preprocess graph is restrained by the maximum number of iterations (therefore bounded by a factor of two). The temperature parameter controls the degree of randomness with respect to the edge addition, where a higher temperature leads to selecting the next edge more deterministically. In each step, the method removes the edge with the maximum positive curvature in order to further balance the degree distribution with respect to curvature, and a threshold can be introduced to prevent excessive skewing of the curvature distribution. When the user does not want to retain a remove edge, we can set the threshold to be very high. Relying on the continuous backward Ricci flow idea of uniform curvature distributions, the alternative preprocessed options create a more uniform curvature distribution over the graph. Unlike a straight-forward adaptation of Ricci flow on graphs where (negatively) curved edges impede message propagation, this is a more efficient and structured re-wired option based on curvature over the graph. A comparison with its continuous counterpart highlights the effectiveness of this technique in improving graph connectivity while maintaining computational efficiency.

For any edge $i \sim j$ in a simple, unweighted graph G, $\mathrm{Ric}(i,j)$ is stated as below:

$$\begin{aligned}\mathrm{Ric}(i,j) :=& \frac{2}{d_i} + \frac{2}{d_j} - 2 + 2\frac{|\#_{\triangle}(i,j)|}{\max\{d_i,d_j\}} + \frac{|\#_{\triangle}(i,j)|}{\min\{d_i,d_j\}} \\ &+ \frac{(\gamma_{\max})^{-1}}{\max\{d_i,d_j\}}(|\#^i_{\square}| + |\#^j_{\square}|)\end{aligned} \tag{7}$$

where

(i) $|\#_{\triangle}(i,j)| := S_1(i) \cap S_1(j)$ are the triangles based at $i \sim j$.

(ii) $|\#^i_{\square}(i,j)| := \{k \in S_1(i) \setminus S_1(j), k \neq j : \exists w \in (S_1(k) \cap S_1(j)) \setminus S_1(i)\}$ are the neighbors of i forming a 4-cycle based at the edge $i \sim j$ without diagonals inside.

(iii) $\gamma_{\max}(i,j)$ is the maximal number of 4-cycles based at $i \sim j$ traversing a common node.

Algorithm 1 Stochastic Discrete Ricci Flow (SDRF)

Require: Graph G, temperature $T > 0$, max number of iterations, optional Ricci upper-bound C^+

repeat

For edge $i \sim j$ with minimal Ricci curvature $\mathrm{Ric}(i,j)$:

- Calculate vector $\mathbf{x}$ where

$$x_{kl} = \mathrm{Ric}_{kl}(i,j) - \mathrm{Ric}(i,j)$$

The improvement to $\mathrm{Ric}(i,j)$ from adding edge $k \sim l$ where $k \in B_1(i)$, $l \in B_1(j)$; Sample index k,l with probability $\mathrm{softmax}(\tau\mathbf{x})_{kl}$ and add edge $k \sim l$ to G.

- Remove edge $i \sim j$ with maximal Ricci curvature $\mathrm{Ric}(i,j)$ if $\mathrm{Ric}(i,j) > C^+$.

until convergence, or max iterations reached

b. Proposed Method

1. Commute time metric

In graph theory, the **commute time** between two nodes u and v in a graph is defined as the expected number of steps a random walker takes to travel from u to v and back to u. Commute time plays a crucial role in Graph Neural Networks (GNNs), particularly in spectral-based methods that leverage graph structure to encode information.[4]

Mathematically, the commute time $CT(u,v)$ can be expressed using the pseudoinverse of the graph Laplacian matrix:

$$CT(u,v) = 2\left(L^{\dagger}_{uu} + L^{\dagger}_{vv} - 2L^{\dagger}_{uv}\right), \tag{8}$$

where $L^{\dagger}$ is the Moore-Penrose pseudoinverse of the Laplacian matrix L. The Laplacian matrix is defined as:

$$L = D - A, \tag{9}$$

where D is the degree matrix and A is the adjacency matrix of the graph.

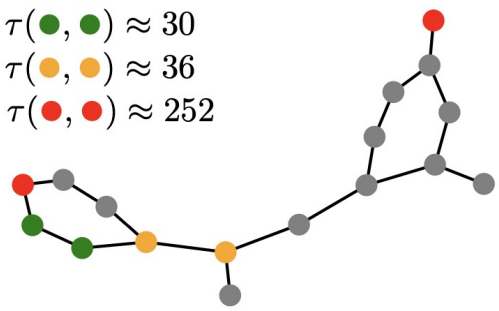

Fig. 3: Commute time

Commute time is a valuable metric in graph analysis because it provides both aggregate and pairwise measurements of connectivity. By incorporating commute time into GNN architectures, models can improve their ability to preserve graph topology and learn meaningful node representations, making them particularly useful for tasks such as node classification and regression. [4]

Building upon the foundational approach, we introduced several modifications to enhance the effectiveness and adaptability of the rewiring process.

Firstly, we discovered that the τ randomness variable, which was calculated using the softmax function, was of little meaningful value in the selection process and created unnecessary computation weighting. Therefore, we removed this component and put a more useful metric back in its place. The integrated metric allows a global and pairwise measurement so that the algorithm can add edges in each iteration in a more nuanced manner instead of randomly adding edges. In doing so, we improve the overall quality of edge rewiring in our method—this ensures every rewiring counts and is relevant to the overall structure.

Secondly, we observed that the original iteration number used in the method was a static input parameter, and therefore limited to datasets with different graph sizes. A fixed iteration number does not account for the different structures of different graphs, and will either modify too much or too little. To develop a solution, we replaced this static iteration parameter with a dynamic threshold. This new system allows the algorithm to identify its stopping condition based on the particular structure of the graph, and does not require consideration of a fixed iteration parameter. A higher commute time threshold means that we make fewer modifications to the graph, thereby preserving more of the original dataset's structure and minimizing disruption to its inherent properties. Conversely, a lower threshold allows for more aggressive rewiring, potentially improving connectivity but at the cost of altering the graph's original topology. This allows our method to be more scalable and generalizable to datasets with different graph sizes, and also be more efficient without tuning parameters.

These observations, which do not seem to have been pointed out in earlier works, are the main motivation for the approach that we develop in this work.

Algorithm 2 CT-based SDRF Method

Require: Graph G, commute time threshold τ, optional Ricci upper-bound C^+

repeat

For edge $i \sim j$ with minimal Ricci curvature $\mathrm{Ric}(i, j)$:

- Sample index $k \sim l$ that reduces the commute time most

$$k \sim l = \arg\max \left(\mathrm{CT}(G) - \mathrm{CT}_{kl}(G)\right)$$

  and add edge $k \sim l$ to G.
- Remove edge $i \sim j$ with maximal Ricci curvature $\mathrm{Ric}(i, j)$ if $\mathrm{Ric}(i, j) > C^+$.

until convergence, or commute time threshold reached

V. Experiments

During our regression tests, we examined the effectiveness of our proposed SDRF method, which adjusted for commute times, in improving GNN performance in estimating continuous target variable. We conducted our experiments on the QM9 dataset where the accurate predictions of molecular properties requires learning long-range dependencies between atoms or groups of atoms. We tested the potential of our rewiring process to improve message passing in depth-based networks since over-squashing can obstruct improvements in predictive performance.

We used R^2, the coefficient of determination, as our main accuracy metric because it is normalized and hence permits the comparison of prediction accuracy across models or datasets. R^2 is more advantageous than mean squared error (MSE), which is dependent on the scale, because it reflects the models ability to explain variance in the target (R^2 closer to 1 indicates a better model's predictive performance and loss of patterns in the estimated relationship). R^2 was a model-based compatibility approach aimed at improving long-range message passing in GNNs and specifically, we were able to leverage our approach to determine if predictive performance on regression tasks were improved (of the model as whole) with respect to R^2.

a. Datasets

We use the QM9 and ZINC benchmark datasets, which are among the largest and most widely used benchmarks for evaluating graph-based learning models, particularly in the domains of molecular property prediction. The QM9 and ZINC datasets serve as popular yet trusted benchmark datasets for assessing Graph Neural Networks (GNNs) with regards to molecular graphs. QM9 is composed of quantum mechanical calculations of 134,000 organic molecules, allowing GNNs to be evaluated in predicting molecular properties including heat of formation and electronic energy. Korolev et al. (2019) showcased message-passing GNNs being

successful in learning local and global molecular structures, demonstrating their state-of-the-art prediction performance. Likewise, ZINC — a collection of over 250,000 drug-like molecules — was again used to study drug generation and optimization.

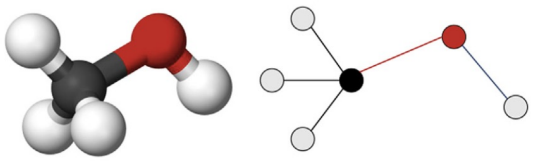

Fig. 4: Graph as a molecule

Bojchevski et al. (2020) pointed to GNNs' ability to learn meaningful graph representations for drug discovery. However, challenges such as over-squashing and over-smoothing in performance of GNN on QM9 and ZINC datasets have presented obstacles. Bober et al. (2022) studied curvature-based geometric tweaks to facilitate improved information flow through the GNN architectures, similar to Giraldo et al. (2023), who studied the trade-offs of likelihood bottlenecks while depth increases. Further studies by Din and Qureshi (2024) assessed how HNN depth variations would succeed or fail neglecting the language of architecture for certain types of molecular representation learning. Both QM9 and ZINC continue to play a role in the development of new neural network schemes, in an ongoing effort to improve methodologies pervious to these datasets, specifically performance and fundamental issues of learning via graphs. [17] [4]

	Number of graphs	Number of features	Number of edges per graph	Number of nodes per graph
QM9	129433	11	37.3	18.0
ZINC	249456	10	49.8	23.8

TABLE 1: DATASET CHARACTERISTICS

b. Results

The experimental findings from both the QM9 and ZINC-datasets illustrate that graph rewiring is an effective means for improving GNN performance. In both datasets, SDRF and the proposed CT-SDRF outperform the Original dataset as expected, confirming that graph structure changes enhance message-passing operations in GNNs. The level of improvement varies across models and datasets, revealing that different architectures can handle rewiring strategies differently, and the realm of molecular property prediction allows for more model performance variability.

In particular for QM9, CT-SDRF usually achieves the largest performance gains, specifically for GIN and GAT, where CT-SDRF outperforms SDRF. This suggests that for QM9, curved re-wiring using CT-SDRF is well suited for the molecular property prediction task. In comparison, for GCN, SDRF achieves a slightly better performance level than CT-SDRF, further implying that the architecture can take advantages from more simple rewiring. Even still, both methods provide significant increases over the Original dataset performance, further adding to the notion that structure alterations in graphs advance the field of graph learning, especially pertaining to GNN implementations.

	GAT	GIN	GCN
Original	0.615 ± 0.011	0.779 ± 0.023	0.704 ± 0.014
SDRF	0.636 ± 0.014	$\mathbf{0.823 \pm 0.022}$	$\mathbf{0.7573 \pm 0.016}$
CT-SDRF	$\mathbf{0.657 \pm 0.013}$	$\mathbf{0.826 \pm 0.014}$	0.749 ± 0.010

TABLE 2: FINAL RESULTS OF QM9 DATASET

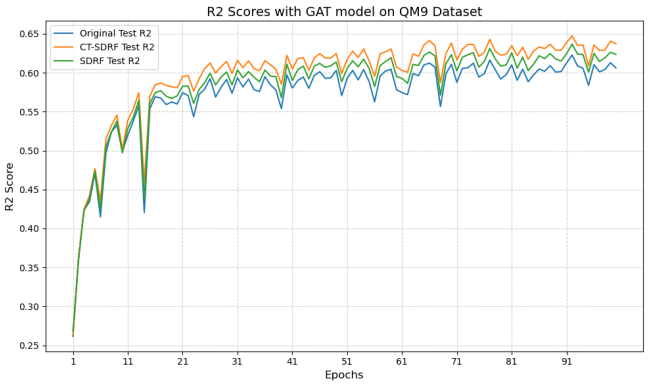

Fig. 5: GAT Model on QM9 dataset

We see a similar pattern in the ZINC dataset where GIN gets the biggest boosts, supporting the claim of its sensitivity to graph rewiring. However, in this dataset, fine-tuning the rewired graph, SDRF method outperforms the CT-SDRF method in some GAT scenarios. This indicates that dataset characteristics affect performance. Nevertheless, reconciling the two methods, CT-SDRF performed solidly and offered meaningful improvements over the baseline which indicates its performance remains robust across different graph labeler tasks.

Overall, the results indicate CT-SDRF improves GNN touched performance, particularly in complicated creatures where passing message structures are considered in GIN and GAT. While SDRF performs solidly as well, especially in cases of GCN, our work provided a method that is much more structured and adaptive. As such, we think this a promising way forward for examining improved influence use in deep graph learning.

	GAT	GIN	GCN
Original	0.631 ± 0.014	0.739 ± 0.013	0.724 ± 0.014
SDRF	$\mathbf{0.688 \pm 0.012}$	$\mathbf{0.783 \pm 0.012}$	0.7373 ± 0.016
CT-SDRF	0.677 ± 0.011	0.766 ± 0.014	$\mathbf{0.749 \pm 0.016}$

TABLE 3: FINAL RESULTS OF ZINC DATASET

The similarity in the trends of the three curves in Figure 5. suggests that while SDRF and CT-SDRF modify the graph structure, they preserve the overall topology and key features. Since these rewiring methods enhance connectivity without disrupting fundamental graph properties, the model learns from a similar feature space, leading to comparable performance across all versions.

VI. Ablation Study

a. Impact of depth

Di Giovanni et al. claim, increasing depth does not alleviate oversquashing but rather leads to a transition into vanishing gradients. [5] To investigate this, we tested various depths of GNNs, and we summarize our results below.

Our results show that increasing depth improves performance at first. Both GIN and GAT improve with depth but begin to plateau at depth 5. GCN remains fairly consistent with marginal improvement until eventually plateauing at depth 6. These results suggest that while increasing depth is helpful at first, oversquashing will re-emerge and possibly revert into vanishing gradients, reinforcing Di Giovanni et al.'s theoretical claims. Additionally, as seen in Figure 5, increasing the depth of the model reduces the average frequency of performance spikes which helps stabilize accuracy. This suggests deeper learning architectures provide a more stable learning process that likely improves the robustness of the model against fluctuations in accuracy.

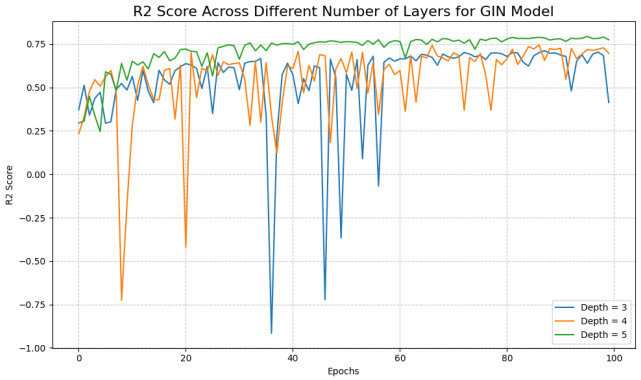

Fig. 6: Depth Comparison

	depth = 3	depth = 4	depth = 5	depth = 6
GAT	0.625 ±0.006	0.697 ±0.008	0.724 ±0.015	**0.729 ± 0.032**
GIN	0.709 ±0.011	0.758 ±0.014	0.791 ±0.013	**0.796 ± 0.012**
GCN	0.701 ±0.004	0.715 ±0.008	0.721 ±0.017	**0.720 ± 0.010**

Table 4: Impact of depth

b. Impact of width

Di Giovanni et al. argue that while increasing the hidden dimension or varying widths of a GNN might mitigate oversquashing, these techniques may detrimentally affect generalizability. Increasing model contribution does not affect over-squashing, which is inherently caused by the topology of the graph. This fully increases model complexity, which may induce overfitting (Bartlett et al., 2017).

Our case study suggests that the above conclusion is valid. The table below shows the performance of different graph neural networks (GNNs) with various width values. It can be observed that increasing the width of each model had no meaningful performance improvement across models. For example, GCN saw a modest benefit from larger widths, increasing performance from 0.707 to 0.779, before dropping, while GAT presented a drop in performance across larger widths. GIN performance was unstable, dropping significantly at width 128, then recovering again with width 256.

This supports Di Giovanni et al. that increasing width alone, or adjusting the model itself, seems suspect for oversquashing. Sensitivity of particular node pairs induced by topology is a challenge for future work that requires a targeted solution like graph rewiring or curvature-based solutions, to mitigate impacts of over-squashing without deterring from generalization.

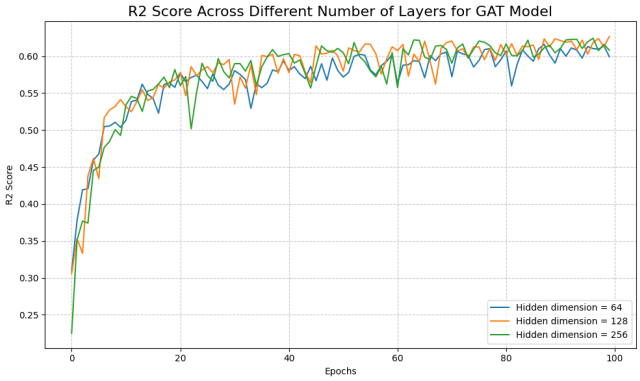

Fig. 7: Width Comparison

	width = 64	width = 128	width = 256
GAT	**0.621 ± 0.011**	0.609 ± 0.023	0.604 ± 0.014
GIN	0.718 ± 0.014	0.683 ± 0.072	**0.7373 ± 0.016**
GCN	0.707 ± 0.013	0.746 ± 0.014	**0.779 ± 0.010**

Table 5: Impact of width

c. Fully Connected Graph

In graph neural networks (GNNs), it is vital to find the right amount of connectivity to optimize effective message propagation with structural adoption. While greater connectivity densities can help address the over-squashing problem by providing alternative paths for information transfer, being fully connected also comes with some clear disadvantages. For example, while more edges enable messages to propagate along more paths, the redundant edges can inhibit differential feature discrimination because correlated (or uninformative) nodes transmit noise instead of signal. In addition, the noise diffusion throughout the network becomes paramount, acting as additional "noise carriers," so that non-informative nodes transmit irrelevant information and will add further fluctuations to GNN performance—creating reputational noise. In order to avoid such performance issues, it is necessary to design an optimal graph topology that maintains local structural integrity while still assuring efficient message passing. Design-based graph structures are better suited to avoiding unnecessary noise with lighter computational burden than as compared with fully connected structures; while size and sparsity are two separate concepts, a sparse and informative connectivity pattern will allow a more pronounced degree of learning of the connectedness of the graph structure without extraneous noise and computational

burden. To have the right kind of information flow and avoid overloaded information transmission, rather than attempting to achieve a fully connected graph by adding edges, connectivity should be increased by randomly adding selective strategic edges and defining optimal graph topologies to achieve a differentiable degree of learning while avoiding performance issues.

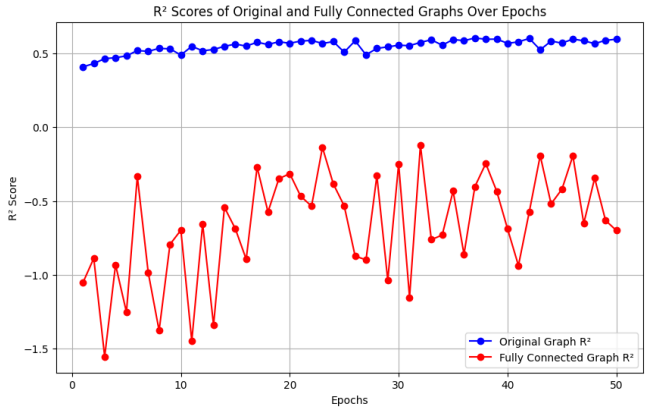

Fig. 8: Fully Connected Graphs' accuracy

VII. Conclusion

While this SDRF and CT-SDRF rewiring methods improve the connectivity of the graph, there are drawbacks to making too many modifications to the input. The most obvious problem is that we are losing out on topological information about the original graph. If the structure of the original graph is indeed relevant, adding and removing edges diminishes that benefit to the task. Another issue arises from the smoothing effects of adding edges: If we add too many edges to the input graph, an ordinary GNN will suffer from oversmoothing (Li et al., 2018). In other words, if we use this natural approach to rewiring, we experience a trade-off between over-squashing and over-smoothing.

SDRF and CT-SDRF rewiring methods' application to certain types of datasets requires careful consideration. In our experiments, we applied these techniques to chemical datasets, where the underlying molecular structure plays a crucial role in determining properties and interactions. Unlike general graph datasets, chemical compounds have strict structural constraints, and excessive modifications may lead to the loss of critical domain-specific information. For instance, adding or removing edges in molecular graphs could disrupt the validity of functional groups, alter molecular stability, or misrepresent bonding patterns, ultimately affecting the predictive performance of GNN models in tasks such as property prediction or drug discovery.

Thus, while rewiring effectively mitigates issues like over-squashing, its impact must be carefully balanced to ensure that the essential characteristics of the molecular graphs remain intact. Future work could focus on developing more chemistry-aware rewiring strategies, incorporating domain-specific constraints to preserve molecular validity while still improving message passing efficiency.

Common Hyperparameters	
Number of Layers	4
Hidden Dimension	64
Learning rate	0.0005
Batch Size	32

Table 6: Common Hyperparameters

References

[1] D. Kreuzer, D. Beaini, W. Hamilton, V. L´etourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," Advances in Neural Information Processing Systems, vol. 34, pp. 21 618–21 629, 2021.

[2] P.K. Banerjee, K. Karhadkar, Y. G. Wang, U. Alon and G. Montufar. Oversquashing in GNNs through the lens of information contraction and graph expansion. arXiv:2208.03471v1

[3] M. Black, Z. Wan, A. Nayyeri and Y. Wang, Understanding Oversquashing in GNNs through the Lens of Effective Resistance, arXiv:2302.06835v2

[4] J. Bober, A. Monod, E. Saucan and K.N. Webster, Rewiring Networks for Graph Neural Network Training Using Discrete Geometry, arXiv:2207.08026v1

[5] F.D.Giovanni, L. Gioti, F.Barbero, G.Luise, P.Lio, M. Bronstein, On Over-Squashing in Message Passing Neural Networks: The Impact of Width, Depth, and Topology, arXiv:2302.02941

[6] F.D.Giovanni, T.K. Rusch, M. M. Bronstein, A. Deac, M. Lackenby, S. Mishra, How does over-squashing affect the power of GNNs?, arXiv:2306.03589v3

[7] L. Fesser and M. Weber, Mitigating Over-Smoothing and Over-Squashing using Augmentations of Forman-Ricci Curvature, arXiv:2309.09384v3

[8] H. Shirzad, A. Velingker, B. Venkatachalam, D. J. Sutherland, and A. K. Sinop, 'Exphormer: Sparse Transformers for Graphs', arXiv [cs.LG]. 2023.

[9] U. Alon and E. Yahav. On the bottleneck of graph neural networks and its practical implications. In International Conference on Learning Representations. arXiv preprint arXiv:2006.05205, 2020.

[10] Akansha. Over-Squashing in Graph Neural Networks: A Comprehensive survey. arXiv:2308.15568v1 [cs.AI] 29 Aug 2023

[11] J. Topping, F. D. Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. arXiv preprint arXiv:2111.14522, 2021.

[12] K. Karhadkar. FOSR: First-order spectral rewiring for addressing over-squashing in GNNs, arXiv:2210.11790v2

[13] K. Nguyen, H. Nong, V. Nguyen, N. Ho, S. Osher, T. Nguyen, Revisiting Over-smoothing and Over-squashing Using Ollivier-Ricci Curvature, arXiv:2211.15779v3

[14] M. Yang, R. Wang, Y. Shen, H. Qi and B. Yin, Breaking the Expression Bottleneck of Graph Neural Networks, arXiv:2012.07219

[15] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, arXiv:1812.08434

[16] Din, A., & Qureshi, M. (2024). Limits of Depth: Over-Smoothing and Over-Squashing in GNNs. Big data mining and analytics. DOI: 10.26599/bdma.2023.9020019

[17] Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. DOI: 10.48550/arxiv.1609.02907