RelWire: Metric Based Rewiring

Editors: List of editors' names

Abstract

Oversquashing is a major hurdle to the application of geometric deep learning and graph neural networks to real world applications. Recent work has found connections between oversquashing and commute times, effective resistance, and the eigengap (or spectral gap) of the underlying graph. Graph rewiring is the most promising technique to alleviate this issue. Some prior work adds edges locally to highly negatively curved subgraphs. These local changes, however, have a small effect on global statistics such as commute times and the eigengap. Other prior work uses the spectrum of the graph Laplacian to target rewiring to increase the eigengap. These approaches, however, make large structural and topological changes to the underlying graph. We use ideas from geometric group theory to present RelWire, a rewiring technique based on the geometry of the graph. We explore topological properties of different rewiring techniques and show that RelWire is Pareto optimal: it has the best balance between improvement in eigengap and commute times and minimizing changes in the topology of the underlying graph, while performing comparably well on downstream tasks.

Keywords: Persistent Homology, Hierarchical Hyperbolic Spaces, Oversquashing

1. Introduction

Graph neural networks (GNNs) are a promising generalization of fully connected neural networks based upon the premise that many real world data sets exhibit graphical structure. That is, data points are related to one another in complex ways that are best captured by relations on a graph, with vertices being the data points and edges the relations between those points. Hence, GNNs are methods for aggregating information across the relation graph and then propagating those updates. All of these methods, however, seem to suffer from a structural problem: oversquashing.

Oversquashing is a challenging problem to define. The problem was initially observed in [2], where the authors create the neighbors match problem in which the task is to match a target subgraph with a template subgraph. They noticed that the feature vectors could not store enough information, a problem that they called oversquashing. However, since then, oversquashing has been defined in a different manner. Following [63], who defined the *influence* of a node on another as $\left\| \frac{\partial x_i^{(\ell)}}{\partial x_j^{(0)}} \right\|$, where $x_i^{(\ell)}$ is the feature at the i^{th} node after ℓ layers of message passing, the perspective on oversquashing has shifted. Since then a variety of papers [58, 45, 10, 21] have defined oversquashing to be the problem of having nodes with small influence on each other (i.e., small Jacobian), including theoretical connections between oversquashing and structural properties like commute time [21] and eigengaps. As a result, many recent works have introduced the notion of "rewiring" a graph, adding edges or relations between the data points so as to improve the performance of GNNs. These methods leverage notions from spectral graph theory (e.g., effective resistance, spectral or eigengap),

discrete graph geometry (e.g., graph curvature), and random walks (via commute time). All of these methods make large structural changes to the underlying graph, some more effective than others, which are aimed at affecting one or more of the above quantities.

In this paper, we present a novel graph rewiring regime, RelWire, which imports techniques from geometric group theory. We use the geometry of the underlying graph to define two relations between the graph's edges, one which roughly encodes negative curvature and the other flat curvature. RelWire is effective at improving GNN performance not only in terms of commute time and eigengap, but also on downstream tasks. Moreover, we develop a novel framework for structure analysis. As with all rewiring techniques, our regime intentionally changes a graph's geometry. Nonetheless, it is a long-standing assumption in the area that graph structure matters. This leads us to ask: *What kind of non-geometric structural information matters?* We look to topology for an answer. By utilizing techniques from topological data analysis, we develop two quantitative measures of topological distortion and show that RelWire achieves a novel balance as compared to existing rewiring methods: it effectively alleviates oversquashing while preserving graph structure.

Contributions The main contributions are as follows:

- We define two new topological distance called the *rank* and distance that can be used to measure the structural changes to a graph after rewiring. (Section 3)
- We present a new method for rewiring graphs, RelWire. Our method uses new ideas and concepts that have not been applied to the field of geometric deep learning before. Specifically, it introduces a new global notion of curvature. (Section 4)
- We present topological differences between RelWire and prior rewiring techniques. (Section 5). We extensively test on real world data to show that RelWire is Pareto optimal for the graph statistics. That is, it performs the best at improving eigengap and commute times while simultaneously preserving the graph topology.

Other Related Work: In this paper, we will compare against transductive methods, but there are also inductive methods for graph rewiring such as [3, 27, 18] and other methods such as [44, 11, 5]. The use of other geometries, especially hyperbolic geometries, has been widely considered; embeddings [55, 46, 54, 47, 36], and geometric graph neural networks [16, 15, 65, 37]. Finally, there are mixed curvature geometries [17, 56, 66, 64, 39, 38, 26].

2. Background and Problem Setup

Prior work has shown that the norm of Jacobian J (which controls oversquashing) can be bounded by a variety of graph properties, deriving bounds of the following form: $J := \left\| \frac{\partial x_i^{(\ell)}}{\partial x_j^{(0)}} \right\| \leq c_{act}^{\ell} T_{ij}^{\ell}$. Here $x_i^{(\ell)}$ is the feature at the i^{th} node after ℓ layers of message passing, c_{act} is a constant from the architecture of the neural network, and T_{ij} is a topological statistic of the graph. Specifically, T can be dependent on commute times [21], the curvature [58, 45], or the effective resistance of the graph [10], which is closely related to commute time, see Appendix A.7. Following this, in recent work, [20] looked at the Hessian instead of the Jacobian and again showed that its norm can be bounded using the commute times. Importantly, prior work shows that these quantities can be improved via graph rewiring. *In this paper, we are interested in the problem of graph rewiring. That is, given a graph G, we*

want to add k edges to improve graph statistics mentioned above while preserving structural information.

Preserving structural information. One of the foundational principles of GNNs was that the structure of the graph had important information, and there are many tasks that illustrate this. One example of this is the NeighborsMatch problem from [2], which was the first paper to identify oversquashing. For this, the task is to identify the labeled node whose neighbor subgraph is exactly the same as the given query node. Hence, changing the graph structure changes the answer, making structural integrity critical. Another example comes from the (real world) ZINC dataset of molecules, where the task is to predict a value that depends on the number of cycles with at least 6 atoms. Once again, changing the graph structure would change the answer.

On the other hand, if we are trying to rewire a graph to alleviate oversquashing while not caring about the graph structure, then a natural extreme conclusion might be to use the complete graph. This was explored in [58, 32] among other papers, and they saw that this did not have the best performance. At the very least, this indicates that the graph structure is not always irrelevant and, moreover, that more fine-tuned approaches can get better results. However, the process of graph rewiring changes this structure. Hence, we are interested in quantifying this change and keeping it to a minimum. To do this, we introduce two notions of distance that measure the change in relevant topological features of rewired graphs relative to the base graph; see Section 3. The use of topology to measure distances between graphs appears in [49, 52].

3. Capturing topological distortion: Distances from persistent homology

In this paper, we consider two notions of distances between graphs using topological information. The first is based on comparing 1-dimensional information, which is already quite powerful in the context of graphs. The second is based on techniques from topological data analysis, which takes into account higher dimensional features of the graphs. This latter machinery is called *persistence homology*, as it attempts to capture "persistent" homological features as one takes larger samples of the space. The following is a minimal treatment of persistent homology, see [53, 1] for more details.

These topological calculations involve integral *homology groups*. The integral d^{th}*-homology group* of a topological space X, denoted $H_d(X;\mathbb{Z})$, is an abelian group which encodes certain d-dimensional topological features up to a natural topological equivalence. The *rank* of $H_d(X;\mathbb{Z})$—namely the number of its $\mathbb{Z}$-factors—encodes the number of d-dimensional "holes", and is called the d^{th} Betti number β_d. Notably, in dimensions 0 and 1, these numbers have concrete meanings: β_0 encodes the number of connected components of X, and β_1 encodes the number of loops on X (up to homotopy). In what follows, we will want to consider the homology of simplicial complexes obtained by iteratively adding higher dimensional simplices, with our starting point being a graph. This sequence of simplicial complexes, called a *filtration*, as well as a notion of how topological features can appear and vanish along the filtration, which is called *persistence*; see [35, 34, 6] and Appendix A.6

In this paper, we consider two different filtrations where the base complex is a graph. The first is a standard filtration known as the Vietoris-Rips filtration. The idea behind the Vietoris-Rips filtration is that it transforms a metric space into a filtration of simplicial complexes,

which, in the context of a graph, involves introducing higher dimensional topological features that are derived from the geometry of the graph. The second is the filtration defined by the subsequent addition of edges by a graph rewiring procedure, in which every level G_k is a graph. We will use these filtrations to define distances, with the first type of distance being similar to those used in prior work such as [53, 28]. Using a persistence diagram, we can generalize Betti numbers to a more expressive quantity known as the Betti curve [31, 30]. We then use this to define the Betti distance between persistence diagrams. In practice, we will use the Betti distance to compute the higher dimensional "topological distortion" from a base graph G and some other graph G' built from G by adding edges via a rewiring process.

Definition 1 (Betti Distance) *Given two graphs G, G', the* betti distance *is the L_2 norm of the difference between their respective Betti curves for their respective Vietoris-Rips filtrations $\{VR_r(G)\}_{r\in\mathbb{R}_+}$ and $\{VR_r(G')\}_{r\in\mathbb{R}_+}$.*

Our second notion of distance measures 1-dimensional topological distortion. A *graph filtration* $G_0 \subset G_1 \subset \cdots$ has G_i a simplicial graph for each i. Graph filtrations naturally arise in the iterative graph rewiring procedures considered in this paper. Since simplicial graphs have no homology beyond dimension 1 and all edges have length 1, the only relevant features of a graph filtration are loops, and each birth and death happens at integer time values. Hence, their Betti curves are step functions, and we obtain:

Lemma 2 (Rank distance) *If $G_0 \subset G_1 \subset \cdots$ and $G'_0 \subset G'_1 \subset \cdots$ are two graph filtrations, then their persistence distance equals* $\text{average}_i|\text{rank}\, H_1(G_i;\mathbb{Z}) - \text{rank}\, H_1(G'_i;\mathbb{Z})|$. *Hence we call the persistence distance between a pair of graph filtrations the* rank distance.

4. RelWire: relations on graphs

Hierarchical hyperbolicity [8] is an axiomatic framework for studying hybrid spaces that exhibit aspects of coarse negative, flat, and positive curvature. This hierarchical approach builds on work in several areas of low dimensional topology, including mapping class groups ([40], Teichmüller spaces ([13, 51, 22]), and hyperbolic 3-manifolds ([41, 14]). These *hierarchically hyperbolic spaces* (HHSes) are coarsely built out of hyperbolic spaces, which are combined in both negative and flat curvature ways based on various *relations* between the spaces. We will apply a simplified version of this hierarchical framework to study the curvature properties of graphs. In particular, we will use the geometry of a fixed graph to induce two (mutually exclusive) types of relations among its edges.

In our setting, the ambient space $X = G$ is a simplicial graph, and this philosophy becomes quite simple: the spaces in the hierarchy are the edges of the graph, and a projection of a vertex of G to an edge E is a collection of its endpoints. Specifically, given a vertex $v \in G^{(0)}$ and an edge E of G, the *projection* $\pi_E(v) \subset E^{(0)}$ of v to E is the endpoint of E which is closest in G to v. When both endpoints are equidistant to v, then we set $\pi_E(v) = E^{(0)}$ to be both endpoints. With these projections defined, we define our (simplified) relations.

The first relation, called *orthogonal*, encodes flat curvature. In the setting of an HHS X, when two hyperbolic spaces U, V in the hierarchy are orthogonal, the product map $\pi_U \times \pi_V : X \to U \times V$ is surjective, and there is a coarsely isometrically embedded flat

(*a*) 4 cycle (*b*) Cross

Figure 1: Here are two simple graphs in which: (a) all adjacent edges are independent and (b) all edges are transverse.

subspace of X (see e.g. Subsection 5B of [7]). For instance, $\mathbb{R}^2$ is an HHS where the hyperbolic spaces are the coordinate axes (i.e., copies of $\mathbb{R}$), and the flat subspace corresponding to their product is the whole ambient space $\mathbb{R} \times \mathbb{R} = \mathbb{R}^2$. We will say two edges E_1, E_2 are *independent* when $\pi_{E_1} \times \pi_{E_2} : G \to E_1^{(0)} \times E_2^{(0)}$ is surjective. Otherwise, we will say that E_1, E_2 are *transverse*. Roughly, this notion of transversality encodes negative curvature (see, e.g., [9]). While the connection between independence/transversality and flat/negative curvature is not exact, the connection is more than a vague analogy:

Lemma 3 *Let E_1, E_2 be edges of a simplicial graph G. If for E_1, E_2 we have that they*
1. *are contained in a clique subgraph of G, then E_1, E_2 are independent;*
2. *are separated by vertex v with $\pi_{E_1}(v)$ and $\pi_{E_2}(v)$ both singletons, then E_1, E_2 are transverse.*

Much more is true in practice. For instance, most edges in a given loop will be pairwise independent. More refined geometric relations are capable of exactly encoding the equivalence between independence and being in a loop, but these conditions are difficult to state and even slower to implement algorithmically.

Remark 4 (Global vs. local curvature) *As every pair of edges in a graph satisfies one of our two relations, they are both capable of capturing local and global properties of the graph. While item (2) of Lemma 3 says that independence is frequently more locally focused, transversality captures negative curvature in a fundamentally different way than existing notions of graph curvature.*

RelWire We present our new rewiring technique RelWire(Algorithm 1). The basic idea is to eliminate negative curvature by adding edges, so the main task is to identify pairs of vertices which belong to the most transverse edge pairs, weighted by their distance in the graph. We begin by determining for each pair of edges if they are independent or transverse. Then for each pair of nodes u, a, we consider all neighboring edges of the form (u, v) and (a, b) for all $v \in \mathcal{N}(u)$ and $b \in \mathcal{N}(a)$. Then we define $r(u, a) := d(u, a) \sum_{\substack{v \in \mathcal{N}(u) \\ b \in \mathcal{N}(a)}} \mathbb{1}\{(u, v) \not\perp (a, b)\}$.
We then connect the k node pairs that are not adjacent in the graph that have the highest r values, where k is our rewiring parameter. That is, for a pair of nodes, we count the number

of transverse edge pairs that the two nodes are in. We then weight this count by the distance between the two nodes and connect the pairs of nodes with the highest weighted r value. As discussed above, transversality captures some notion of negative curvature at both the local and global scale of the graph. Hence, connecting a highly transverse distant pair morally helps remove negative curvature at a global level. Time time complexity and empirical run times can be found in Appendix A.8.

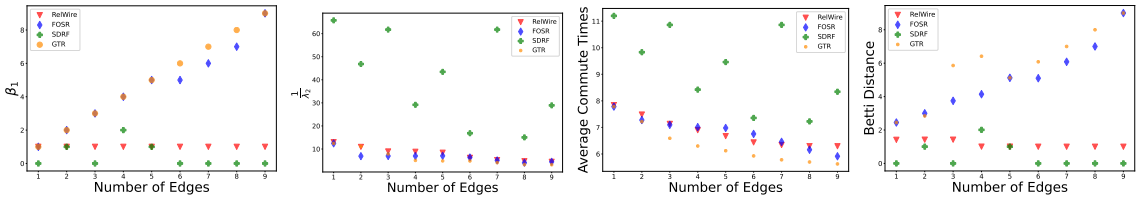
Algorithm 1 RelWire

1: **Input:** G - Graph, k - number of edges added
2: Compute shortest distance $d(u, v)$ between all pairs of nodes u, v.
3: Compute $T : E \times E \to \{0, 1\}$ such that $T(e_1, e_2) = 1$ if and only if $e_1 \not\perp e_2$.
4: Compute $r(u, a) = d(u, a) \sum_{v \in \mathcal{N}(u), b \in \mathcal{N}(a)} T\big((u, v), (a, b)\big)$.
5: Connect the k non-adjacent node pairs with largest r value.
6: **Return:** Rewired Graph

Connecting RelWire to topology. By Lemma 3, one should expect that RelWire, applied to a very large graph, will identify a "maximally transverse" pair of vertices p, q which are very far apart in the graph. Many of the vertices v occurring along any geodesic between p, q will likely be separators, in the sense of item 2 of that lemma. Adding an edge between p, q then creates a number of loops containing these separators. On the other hand, one should expect any such separator v to participate in a comparable number of transverse pairs with both p, q, making it likely that RelWire will fill in the loops it creates. This philosophy is most clearly illustrated in Figure 4.

5. Preserving the topology

In this section, we explore the topological differences between RelWire, FOSR, GTR, and SDRF. We will do this using the standard example of a barbell graph G (two K_5 connect by a path with 6 edges) that has been used in prior work such as [32, 20, 21, 58]. We shall use all four methods to add between one and nine edges, and we shall see that the results are quite different. Some of the rewired graphs can be seen Figure 4 in the appendix. To understand the topological distortion caused by rewiring, we compute both the rank and Betti distances relative to the underlying graph.



(a) Ranks (b) $\frac{1}{\lambda_2}$ (c) Commute Times (d) Betti Distance

Figure 2: Figure showing the ranks of the first homology group of the rewired graphs, $\frac{1}{\lambda_2}$, the average commute times, as well as the Betti distance between the rewired graph and the original graph for RelWire, SDRF, GTR, and FOSR.

Figure 2 in the appendix shows the ranks for adding up to 9 edges. Here, we see that RelWire initially introduces a loop, and the rank increases to 1, while successive edges fill in that loop. On the other hand, both FOSR and GTR create a loop with each successive edge addition. Finally, it is not clear what SDRF does to the topology. We also analyze how rewiring affects the statistics related to oversquashing, i.e., average commute times and eigengaps of the graph. These quantities, along with the Betti distance to the original graph, are plotted in Figure 2. As we can see, RelWire, GTR, and FOSR have the best average commute times and eigengaps. On the other hand, we see that FOSR and GTR result in large topological changes, whereas RelWire has relatively little effect on the topology.

Dataset	Eigengap				Betti Distance				Commute Times			
	Relwire	FOSR	GTR	SDRF	Relwire	FOSR	GTR	SDRF	Relwire	FOSR	GTR	SDRF
Zinc	0.09	0.07	0.13	0.03	2.7	3.5	5.2	0.9	10	11	9.2	13
ESOL	0.4	0.37	0.40	0.21	1.7	3.0	3.5	0.7	6.1	6.3	5.7	7.0
BACE	0.05	0.03	0.08	0.02	3.5	3.8	5.9	1.0	14	15	13	18
Lipo	0.08	0.06	0.12	0.03	3.1	3.7	5.3	1.0	-	-	-	-
Tox21	0.24	0.22	0.26	0.12	2.2	3.2	4.3	0.7	-	-	-	-
Mutag	0.18	0.15	0.23	0.10	2.2	3.2	4.2	1.3	7.7	8	7.2	8.8
Enzymes	1.10	1.10	1.30	0.06	2.5	3.5	4.6	1.2	-	-	-	-
AIDS	0.46	0.44	0.49	0.31	1.0	2.9	3.0	0.6	4.5	4.4	4.1	4.8
Alkane	0.46	0.47	0.43	0.24	0.4	2.9	3.0	0.4	4.8	4.6	4.2	5.2
Linux	0.60	0.62	0.57	0.3	0.6	2.9	2.7	0.4	3.9	3.8	3.6	4.3

Table 1: Eigengap λ_2, average commute times, and Betti distance for the rewired graphs.

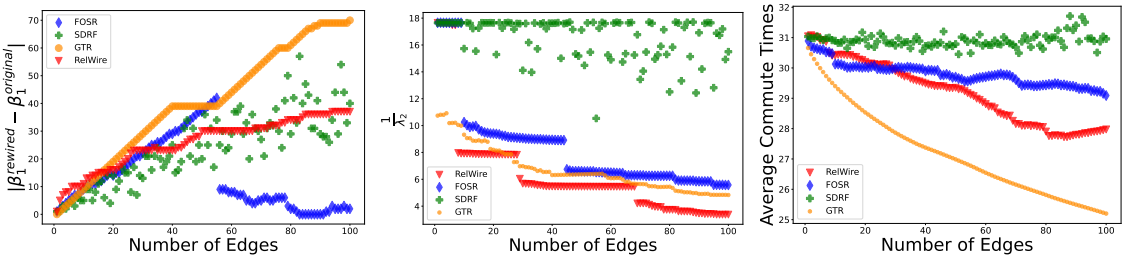
To further validate our method on real data[1], we took ten different datasets with roughly $\sim 50,000$ graphs for rewiring (see the Appendix A). We rewired these datasets by adding three edges using RelWire, FOSR, GTR, and SDRF. We then computed the spectral gap for each of the graphs in the dataset and took the average spectral gap for each dataset. Similarly, we computed the average commute times for each of the graphs and then averaged that as well. Note that for three of the datasets (Lipo, Tox21, and Enzymes) all rewiring strategies produced disconnected graphs, hence we did not compute the commute times for these datasets. Finally, we computed the Betti distance. Table 1 has the various statistics. The values in green are the best observed values, while those in blue are the second best.

Here, we can see there is a tradeoff between the eigengap and commute times with the Betti distance. In particular, GTR greatly decreases the eigengap and commute times at the expense of transforming the topology, as measured by large Betti distance. On the other hand, SDRF relatively preserves the topology as well as the eigengap and commute times. Hence, if we are to reduce oversquashing while preserving the graph structure, we must find a balance. In this regard, we see that RelWire is Pareto optimal in that we have the second best eigengap, commute times, and Betti distance.

Betti Curve for Rank Persistence In the previous experiments, we only added a fixed number of edges to the graphs. As with the barbell graph example, it is interesting to see

1. All code can be found anonymized at Github

how the statistics change as we vary the number of edges added. Hence, we took Texas and Cornell from the WebKb dataset [50] and added up to 100 edges. Figure 3 shows the results for Texas. The one for Cornell can be seen in the Appendix. Here we see that RelWire has the best eigengap, FOSR has the best rank distance, and GTR has the best commute times. There are many other interesting aspects to the curves. The first is the jump discontinuity in the rank of the first homology group from FOSR. This implies that FOSR reaches a critical number of edges, after which adding loops is no longer beneficial and starts eliminating loops. On the other, GTR and SDRF seem to always add loops. RelWire on the other hand, seems to always want to eliminate topological loops. The jump discontinuity in the rank of the first homology group for FOSR seems to correlate with the jump discontinuity in the eigengap curve. However, interestingly, we see no discontinuity in the commute times curve.



(*a*) Rank distance (*b*) $1/\lambda_2$ (*c*) Commute Time

Figure 3: Comparing the four rewiring methods: in (a), rank distance; (b) eigengap (λ_2^{-1}), and (c) the average commute times, each as a function of the number of edges added.

6. Graph Regression and Classification: When is topology important

We show that preserving the topology helps with downstream tasks. To do so, we take 11 datasets - ZINC, Peptides from the LRGB dataset, Lipo, BACE, and ESOL from the MoleculeNet dataset, and the six datasets from the TUDataset. The six datasets from the TUDataset are for graph classification, and the other datasets are for graph regression. For each dataset, we considered twelve different models. Specifically, we use 4 different architectures - GCN, GraphConv, GAT, and GIN, and depths of 3, 4, and 5 layers. For each of the twelve different models, we trained the model five times for seven different rewiring techniques - RelWire, FOSR, SDRF, GTR, no rewiring, replacing the graph with the fully connected graph, and replacing the graph with the empty graph. We then picked the trial with the best validation accuracy and looked at the corresponding test accuracy. Next, we ranked the different rewiring techniques with ranks from 0 (best) to 6 (worst). Finally, we computed the average rank over the twelve models. This is reported in Table 2.

As we can see, RelWire has the best mean rank for Graph Classification but doesn't do as well for Graph Regression. Further, since we tried four different architectures, it is interesting to see how well the rewiring technique depends on the architecture used. Table 3 presents the mean rank averaged over all three depths and eleven datasets. Here, we see that despite the fact that RelWire does not do well for Graph Regression, we see that RelWire has the best overall performance when using the GCN architecture.

It is also interesting to note what happens when RelWire does badly. Specifically, if we look at Lipo dataset, we see that RelWire does very badly. Hence, this suggests that

preserving the topological structure is not relevant to this task. This is further verified by seeing that the method with the best performance is replacing the graph with the complete graph. *Our experiment further highlights that the connection between rewiring and the performance on the downstream task is not straightforward and can depend on many factors, including the architecture of the model, the type of data, and the type of rewiring.*

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
IMDB	3.25 ± 0.57	3.25 ± 0.59	3.58 ± 0.58	1.67 ± 0.50	4.25 ± 0.46	2.75 ± 0.69	2.25 ± 0.45
Reddit	2.00 ± 0.46	N/A	5.00 ± 0.00	2.08 ± 0.45	1.92 ± 0.38	1.75 ± 0.35	2.25 ± 0.46
Collab	1.75 ± 0.45	N/A	5.00 ± 0.00	1.42 ± 0.40	2.33 ± 0.45	2.17 ± 0.32	2.33 ± 0.43
Mutag	1.17 ± 0.58	3.33 ± 0.70	1.83 ± 0.42	3.42 ± 0.53	4.08 ± 0.36	2.92 ± 0.51	4.25 ± 0.41
Proteins	3.25 ± 0.39	0.92 ± 0.56	6.00 ± 0.00	2.00 ± 0.35	3.08 ± 0.42	2.33 ± 0.40	3.42 ± 0.51
Enzymes	1.33 ± 0.45	5.50 ± 0.19	4.67 ± 0.36	2.50 ± 0.58	2.58 ± 0.47	2.58 ± 0.45	1.83 ± 0.41
Peptides	4.00 ± 0.12	N/A	4.92 ± 0.08	1.58 ± 0.15	1.42 ± 0.15	0.00 ± 0.00	3.08 ± 0.08
Lipo	4.92 ± 0.45	0.67 ± 0.50	2.08 ± 0.62	4.00 ± 0.43	2.58 ± 0.45	3.67 ± 0.31	3.08 ± 0.42
BACE	0.92 ± 0.29	5.42 ± 0.19	5.50 ± 0.15	2.83 ± 0.44	2.17 ± 0.32	1.58 ± 0.45	2.58 ± 0.38
ESOL	2.58 ± 0.63	3.33 ± 0.64	5.17 ± 0.21	2.58 ± 0.50	1.50 ± 0.50	1.83 ± 0.37	4.00 ± 0.41
Zinc	1.25 ± 0.39	5.67 ± 0.14	5.25 ± 0.18	3.25 ± 0.39	2.08 ± 0.29	2.25 ± 0.30	1.25 ± 0.46

Table 2: Table with the mean rank and standard error for each dataset and rewiring method averaged over the twelve different models considered for Graph Classification and Regression. The cells in green are the best, while the cells in blue are the second best.

	None	Full	Empty	RelWire	FOSR	GTR	SDRF
GCN	3.58 ± 0.35	4.79 ± 0.44	5.73 ± 0.24	3.12 ± 0.28	3.97 ± 0.28	3.21 ± 0.28	3.61 ± 0.31
GC	3.42 ± 0.35	5.91 ± 0.34	5.42 ± 0.34	3.36 ± 0.27	3.36 ± 0.26	2.91 ± 0.27	3.61 ± 0.25
GAT	3.24 ± 0.33	4.88 ± 0.42	5.09 ± 0.33	4.03 ± 0.35	3.45 ± 0.30	3.48 ± 0.27	3.82 ± 0.31
GIN	3.36 ± 0.36	5.18 ± 0.43	5.58 ± 0.26	3.42 ± 0.24	3.39 ± 0.28	3.06 ± 0.31	4.00 ± 0.27

Table 3: Table with the mean rank and standard error for each architecture and rewiring method averaged over the three different and eleven datasets.

7. Conclusion

We use ideas from geometric group theory to develop a new rewiring technique known as RelWire using a new curvature-like relation on edges. We introduce a new topological distance, which measures how rewiring changes the structure of a graph. We also show that different rewiring techniques have different topological properties and that whether we should preserve the topological information is dependent on the data and the task. We show that compared to other methods, RelWire is Pareto optimal in that it makes small topological changes to the graph and makes big changes to statistics connected to oversquashing, such as eigengap and commute times. We test RelWire on the downstream task of graph classification and report positive results.

References

[1] Mehmet E Aktas, Esra Akbas, and Ahmed El Fatmaoui. "Persistence homology of networks: methods and applications". In: *Applied Network Science* 4.1 (2019), pp. 1–28 (cit. on p. 3).

[2] Uri Alon and Eran Yahav. "On the Bottleneck of Graph Neural Networks and its Practical Implications". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=i80OPhOCVH2 (cit. on pp. 1, 3).

[3] Adrian Arnaiz-Rodriguez et al. "DiffWire: Inductive Graph Rewiring via the Lovasz Bound". In: *The First Learning on Graphs Conference*. 2022. URL: https://openreview.net/pdf?id=IXvfIex0mX6f (cit. on pp. 2, 16).

[4] Yunsheng Bai et al. "Simgnn: A neural network approach to fast graph similarity computation". In: *Proceedings of the twelfth ACM international conference on web search and data mining*. 2019, pp. 384–392 (cit. on p. 15).

[5] Pradeep Kr Banerjee et al. "Oversquashing in GNNs through the lens of information contraction and graph expansion". In: *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2022, pp. 1–8 (cit. on p. 2).

[6] Ulrich Bauer and Michael Lesnick. "Induced Matchings of Barcodes and the Algebraic Stability of Persistence". In: *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*. SOCG'14. Kyoto, Japan: Association for Computing Machinery, 2014, pp. 355–364. ISBN: 9781450325943. DOI: 10.1145/2582112.2582168. URL: https://doi.org/10.1145/2582112.2582168 (cit. on p. 3).

[7] Jason Behrstock, Mark Hagen, and Alessandro Sisto. "Hierarchically hyperbolic spaces II: Combination theorems and the distance formula". In: *Pacific Journal of Mathematics* 299.2 (2019), pp. 257–338 (cit. on p. 5).

[8] Jason Behrstock, Mark Hagen, and Alessandro Sisto. "Hierarchically hyperbolic spaces, I: Curve complexes for cubical groups". In: *Geometry & Topology* 21.3 (2017), pp. 1731–1804 (cit. on p. 4).

[9] Mladen Bestvina, Ken Bromberg, and Koji Fujiwara. "Constructing group actions on quasi-trees and applications to mapping class groups". In: *Publications mathématiques de l'IHÉS* 122.1 (2015), pp. 1–64 (cit. on p. 5).

[10] Mitchell Black et al. "Understanding oversquashing in gnns through the lens of effective resistance". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 2528–2547 (cit. on pp. 1, 2, 18, 19).

[11] Jakub Bober et al. "Rewiring Networks for Graph Neural Network Training Using Discrete Geometry". In: *arXiv preprint arXiv:2207.08026* (2022) (cit. on p. 2).

[12] Karsten M. Borgwardt et al. "Protein Function Prediction via Graph Kernels". In: 21.1 (2005), pp. 47–56. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bti1007. URL: https://doi.org/10.1093/bioinformatics/bti1007 (cit. on p. 14).

[13] Jeffrey Brock. "The Weil-Petersson metric and volumes of 3-dimensional hyperbolic convex cores". In: *Journal of the American Mathematical Society* 16.3 (2003), pp. 495–535 (cit. on p. 4).

[14] Jeffrey F Brock, Richard D Canary, and Yair N Minsky. "The classification of Kleinian surface groups, II: The ending lamination conjecture". In: *Annals of Mathematics* (2012), pp. 1–149 (cit. on p. 4).

[15] Ines Chami et al. "Hyperbolic Graph Convolutional Neural Networks". In: *Advances in neural information processing systems* 32 (2019), pp. 4869–4880 (cit. on p. 2).

[16] Weize Chen et al. "Fully Hyperbolic Neural Networks". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 5672–5686. DOI: 10.18653/v1/2022.acl-long.389. URL: https://aclanthology.org/2022.acl-long.389 (cit. on p. 2).

[17] Xinyue Cui and Rishi Sonthalia. "Hyperbolic and Mixed Geometry Graph Neural Networks". In: *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*. 2022 (cit. on p. 2).

[18] Andreea Deac, Marc Lackenby, and Petar Veličković. "Expander Graph Propagation". In: *Proceedings of the First Learning on Graphs Conference*. Ed. by Bastian Rieck and Razvan Pascanu. Vol. 198. Proceedings of Machine Learning Research. PMLR, 2022, 38:1–38:18. URL: https://proceedings.mlr.press/v198/deac22a.html (cit. on p. 2).

[19] Asim Kumar Debnath et al. "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity." In: *Journal of medicinal chemistry* 34 2 (1991), pp. 786–97. URL: https://api.semanticscholar.org/CorpusID:19990980 (cit. on p. 14).

[20] Francesco Di Giovanni et al. "How does over-squashing affect the power of GNNs?" In: *arXiv preprint arXiv:2306.03589* (2023) (cit. on pp. 2, 6).

[21] Francesco Di Giovanni et al. "On Over-Squashing in Message Passing Neural Networks: The Impact of Width, Depth, and Topology". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 7865–7885 (cit. on pp. 1, 2, 6).

[22] Matthew Gentry Durham. "The augmented marking complex of a surface". In: *Journal of the London Mathematical Society* 94.3 (2016), pp. 933–969 (cit. on p. 4).

[23] Lukas Fesser and Melanie Weber. "Mitigating over-smoothing and over-squashing using augmentations of Forman-Ricci curvature". In: *Learning on Graphs Conference*. PMLR. 2024, pp. 19–1 (cit. on p. 19).

[24] Matthias Fey and Jan E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019 (cit. on p. 16).

[25] Rafael Gómez-Bombarelli et al. "Automatic chemical design using a data-driven continuous representation of molecules". In: *ACS central science* 4.2 (2018), pp. 268–276 (cit. on p. 14).

[26] Albert Gu et al. "Learning Mixed-Curvature Representations in Product Spaces". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HJxeWnCcF7 (cit. on p. 2).

[27] Benjamin Gutteridge et al. "DRew: Dynamically Rewired Message Passing with Delay". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 12252–12267. URL: https://proceedings.mlr.press/v202/gutteridge23a.html (cit. on p. 2).

[28] Max Horn et al. "Topological Graph Neural Networks". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=oxxUMeFwEHd (cit. on p. 4).

[29] Sergei Ivanov, Sergei Sviridov, and Evgeny Burnaev. "Understanding isomorphism bias in graph data sets". In: *arXiv preprint arXiv:1910.12091* (2019) (cit. on p. 14).

[30] Megan Johnson and Jae-Hun Jung. "Instability of the betti sequence for persistent homology and a stabilized version of the betti sequence". In: *arXiv preprint arXiv:2109.09218* (2021) (cit. on p. 4).

[31] Alperen Karan and Atabey Kaygun. "Time series classification via topological data analysis". In: *Expert Systems with Applications* 183 (2021), p. 115326 (cit. on p. 4).

[32] Kedar Karhadkar, Pradeep Kr. Banerjee, and Guido Montufar. "FoSR: First-order spectral rewiring for addressing oversquashing in GNNs". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=3YjQfCLdrzz (cit. on pp. 3, 6, 18, 19).

[33] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *ICLR*. 2017 (cit. on p. 16).

[34] Michael Lesnick. "The theory of the interleaving distance on multidimensional persistence modules". In: *Foundations of Computational Mathematics* 15.3 (2015), pp. 613–650 (cit. on p. 3).

[35] Sunhyuk Lim, Facundo Memoli, and Osman Berat Okutan. "Vietoris-rips persistent homology, injective metric spaces, and the filling radius". In: *arXiv preprint arXiv:2001.07588* (2020) (cit. on p. 3).

[36] Ya-Wei Eileen Lin et al. "Hyperbolic Diffusion Embedding and Distance for Hierarchical Representation Learning". In: *ArXiv* abs/2305.18962 (2023) (cit. on p. 2).

[37] Qi Liu, Maximilian Nickel, and Douwe Kiela. "Hyperbolic Graph Neural Networks". In: *NeurIPS*. 2019 (cit. on p. 2).

[38] Federico Lopez et al. "Symmetric Spaces for Graph Embeddings: A Finsler-Riemannian Approach". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 7090–7101. URL: https://proceedings.mlr.press/v139/lopez21a.html (cit. on p. 2).

[39] Federico López et al. "Vector-valued distance and gyrocalculus on the space of symmetric positive definite matrices". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18350–18366 (cit. on p. 2).

[40] Howard A Masur and Yair N Minsky. "Geometry of the complex of curves II: Hierarchical structure". In: *Geometric and Functional Analysis* 10.4 (2000), pp. 902–974 (cit. on p. 4).

[41] Yair Minsky. "The classification of Kleinian surface groups, I: Models and bounds". In: *Annals of Mathematics* (2010), pp. 1–107 (cit. on p. 4).

[42] Christopher Morris et al. "Tudataset: A collection of benchmark datasets for learning with graphs". In: *arXiv preprint arXiv:2007.08663* (2020) (cit. on p. 14).

[43] Christopher Morris et al. "Weisfeiler and leman go neural: Higher-order graph neural networks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 4602–4609 (cit. on p. 16).

[44] Huda Nassar, Kyle Kloster, and David F. Gleich. "Strong Localization in Personalized PageRank Vectors". In: WAW 2015. Eindhoven, The Netherlands: Springer-Verlag, 2015, pp. 190–202. ISBN: 9783319267838. DOI: 10.1007/978-3-319-26784-5_15. URL: https://doi.org/10.1007/978-3-319-26784-5_15 (cit. on p. 2).

[45] Khang Nguyen et al. "Revisiting Over-smoothing and Over-squashing Using Ollivier-Ricci Curvature". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 25956–25979 (cit. on pp. 1, 2, 18).

[46] Maximilian Nickel and Douwe Kiela. "Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry". In: *ArXiv* abs/1806.03417 (2018) (cit. on p. 2).

[47] Maximilian Nickel and Douwe Kiela. "Poincaré Embeddings for Learning Hierarchical Representations". In: *ArXiv* abs/1705.08039 (2017) (cit. on p. 2).

[48] Yann Ollivier. "Ricci curvature of Markov chains on metric spaces". In: *Journal of Functional Analysis* 256.3 (2009), pp. 810–864 (cit. on p. 18).

[49] Sun Woo Park et al. "The PWLR graph representation: A Persistent Weisfeiler-Lehman scheme with Random Walks for graph classification". In: *Topological, Algebraic and Geometric Learning Workshops 2022*. PMLR. 2022, pp. 287–297 (cit. on p. 3).

[50] Hongbin Pei et al. "Geom-GCN: Geometric Graph Convolutional Networks". In: *ArXiv* abs/2002.05287 (2020). URL: https://api.semanticscholar.org/CorpusID:210843644 (cit. on p. 8).

[51] Kasra Rafi. "Hyperbolicity in Teichmüller space". In: *Geometry & Topology* 18.5 (2014), pp. 3025–3053 (cit. on p. 4).

[52] Bastian Rieck, Christian Bock, and Karsten Borgwardt. "A persistent weisfeiler-lehman procedure for graph classification". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5448–5458 (cit. on p. 3).

[53] Bastian Rieck et al. "Neural Persistence: A Complexity Measure for Deep Neural Networks Using Algebraic Topology". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=ByxkijC5FQ (cit. on pp. 3, 4).

[54] Frederic Sala et al. "Representation Tradeoffs for Hyperbolic Embeddings". In: *Proceedings of machine learning research* 80 (2018), pp. 4460–4469 (cit. on p. 2).

[55] Rishi Sonthalia and Anna Gilbert. "Tree! i am no tree! i am a low dimensional hyperbolic embedding". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 845–856 (cit. on p. 2).

[56] Rishi Sonthalia, Anna C Gilbert, and Matthew Durham. "CubeRep: Learning Relations Between Different Views of Data". In: *Topological, Algebraic and Geometric Learning Workshops 2022*. PMLR. 2022, pp. 298–303 (cit. on p. 2).

[57] T. Sterling and John J. Irwin. "ZINC 15 – Ligand Discovery for Everyone". In: *Journal of Chemical Information and Modeling* 55 (2015), pp. 2324–2337. URL: https://api.semanticscholar.org/CorpusID:327319 (cit. on p. 14).

[58] Jake Topping et al. "Understanding over-squashing and bottlenecks on graphs via curvature". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=7UmjRGzp-A (cit. on pp. 1–3, 6, 18, 19).

[59] Domenico Tortorella and Alessio Micheli. "Is Rewiring Actually Helpful in Graph Neural Networks?" In: *arXiv preprint arXiv:2305.19717* (2023) (cit. on p. 16).

[60] Petar Veličković et al. "Graph Attention Networks". In: *ICLR*. 2018 (cit. on p. 16).

[61] Zhenqin Wu et al. "MoleculeNet: a benchmark for molecular machine learning". In: *Chemical science* 9.2 (2018), pp. 513–530 (cit. on p. 14).

[62] Keyulu Xu et al. "How powerful are graph neural networks?" In: *arXiv preprint arXiv:1810.00826* (2018) (cit. on p. 16).

[63] Keyulu Xu et al. "Representation Learning on Graphs with Jumping Knowledge Networks". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 5453–5462 (cit. on p. 1).

[64] Xiaoyu Xu et al. "Joint hyperbolic and Euclidean geometry contrastive graph neural networks". In: *Inf. Sci.* 609 (2022), pp. 799–815 (cit. on p. 2).

[65] Yiding Zhang et al. "Hyperbolic Graph Attention Network". In: *ArXiv* abs/1912.03046 (2021) (cit. on p. 2).

[66] Weichen Zhao et al. "Modeling Graphs Beyond Hyperbolic: Graph Neural Networks in Symmetric Positive Definite Matrices". In: *ArXiv* abs/2306.14064 (2023) (cit. on p. 2).

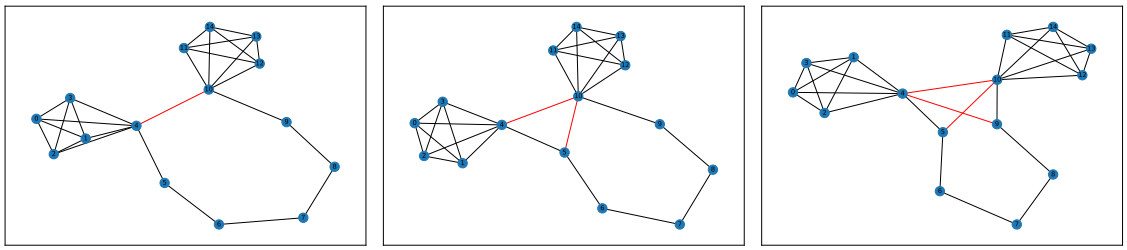
Appendix A. Appendix

A.1. Barbell Experiment

Figure 4 shows the rewired graphs for $k = 1, 2, 3$.

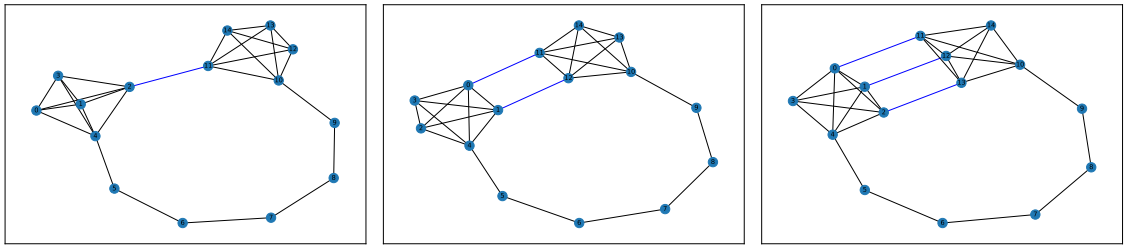
A.2. Datasets

Specifically, we look at The ZINC dataset [57, 25], which consists of 10,000 molecular graphs. From [61], we look at ESOL, which is the water solubility of 1,128 compounds, BACE, which 1,522 compounds representing the inhibitors or human β-secretase 1, Lipophilicity, which is 4,200 drug compounds, and Tox21 measure the toxicity of 7831 compounds. From the TUDataset [42, 29], we use MUTAG [19], which consists of 188 nitroaromatic compounds and ENZYMES [12] which is a dataset of 600 protein tertiary structures obtained from the BRENDA enzyme database. We also use Proteins [12], which is a collection 1113 to determine if a protein is an enzyme. Additionally, Reddit, Collab, and IMDB are social



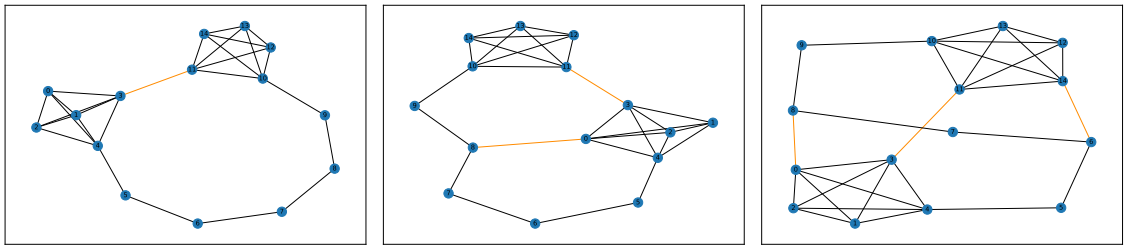
(*b*) $k = 1$ (*c*) $k = 2$ (*d*) $k = 3$

(*a*) RelWire rewiring of the barbell graph



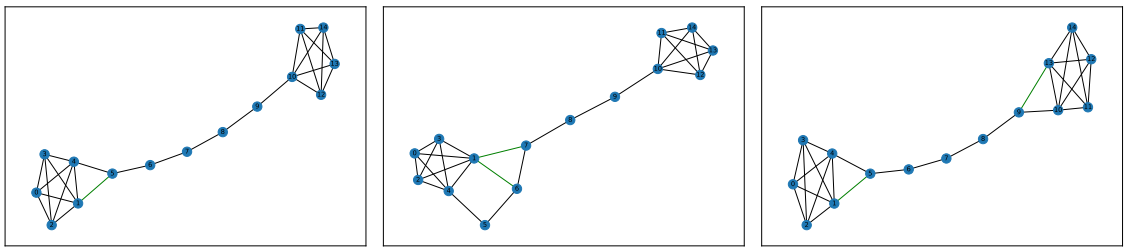
(*f*) $k = 1$ (*g*) $k = 2$ (*h*) $k = 3$

(*e*) FOSR rewiring of the barbell graph



(*j*) $k = 1$ (*k*) $k = 2$ (*l*) $k = 3$

(*i*) GTR rewiring of the barbell graph



(*n*) $k = 1$ (*o*) $k = 2$ (*p*) $k = 3$

(*m*) SDRF rewiring of the barbell graph

Figure 4: Rewired barbell graph.

networks. Finally, we use three datasets from [4] consisting of 1520 graphs in total. The statistics for the datasets can be seen in Table 4.

	AIDS	Alkane	Linux	Proteins	Collab	IMDB	Reddit
Nodes	8.9	8.9	7.6	39.1	74.5	19.8	429.6
Edges	17.6	15.8	13.9	145.6	4914.4	193.1	995.5
# Graphs	700	150	1000	1113	5000	1000	2000
	Zinc	ESOL	BACE	Lipo	Tox21	Mutag	Enzymes
Nodes	23.2	13.3	34.1	27	18.6	17.9	32.6
Edges	49.8	27.4	73.7	59	38.6	39.6	124.3
# Graphs	10000	1128	1513	4200	7831	188	600

Table 4: Table showing the average sizes of the graphs in each dataset.

A.3. Graph Tasks

As we see from the Table, we don't have any consistent trends. However, we do see that RelWire does perform well on average. This lack of trends is further supported by [59], where they do node classification tests on different data sets. This suggests that more work needs to be done to understand when graph rewiring is helpful.

Data Split For each dataset, we used an 80/10/10 split random split.

Models All of the models had the following structure we had ℓ convolutional layers for $\ell \in \{2, 3, 4\}$. This is followed by global mean pooling and then a linear layer. The hidden dim for each model is 64 dimensional.

The architectures we used are GCN [33], GraphConv [43], GAT [60], and GIN [62]. For GIn we used a 2-layer ReLU network with hidden dim 64.

Optimization For all methods, we used Adam optimizer with the default parameters. We also used cosine annealing as the learning rate decay.

For the smaller datasets, we used batch sizes of 10 to 25 and trained them for 100 epochs, and for the bigger datasets, we used a batch size of 100 and trained it for 100 epochs.

For the graph regression tasks, we used the mean squared. For graph classification, we used the cross entropy loss to train the modes.

For testing, we used the MSE loss and accuracy to rank the models.

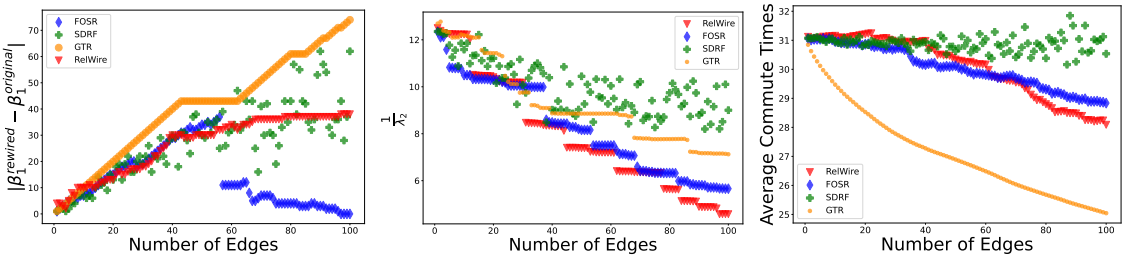
Computing Test Error We trained each model five times. We then picked the trial with the smallest validation accuracy and then reported the corresponding test accuracy.

Computer Resource All datasets were accessed using Pytorch Geometric [24]. The models were all trained on Google Colab using a V100 GPU and pytorch geometric.

For rewiring, for used the official implementations of FOSR and GTR. For SDRF, we used the implementation from the LOG conference tutorial on graph rewiring [3].

A.4. WebKd Experiments

In Figure 5, we show the graphs for Cornell.



(a) Rank Betti Curve (b) $1/\lambda_2$ (c) Commute Time

Figure 5: Graph properties for Cornell for the different rewiring methods.

A.5. Proof

Lemma 5 *Let E_1, E_2 be edges of a simplicial graph G. If for E_1, E_2 we have that they*
1. are contained in a clique subgraph of G, then E_1, E_2 are independent;
2. are separated by vertex v with $\pi_{E_1}(v)$ and $\pi_{E_2}(v)$ both singletons, then E_1, E_2 are transverse.

Proof We start by proving (1). For this let $E_i = (u, v)$ and $E_2 = (a, b)$. Then since this is a clique with all edge weights equal to 1. We have that

$$\pi_{E_1}(a) = \{u, v\} \text{ and } \pi_{E_2}(u) = \{a, b\}.$$

Thus, we have independence.

For (2), we note that since v separates tthe graph into at least two components G_1, G_2 such that $E_1 \in G_1$ and $E_2 \in G_2$. Then we see that for all $x \in G_1$, we have that

$$\pi_{E_2}(x) = \pi_{E_2}(v).$$

Similarly for all $x \in G_2$, we have that

$$\pi_{E_1}(x) = \pi_{E_1}(v).$$

Then since $\pi_{E_1}(v), \pi_{E_2}(v)$ are singletons, we see that we cannot get all four projection pairs. Thus, the edges are transverse. ■

A.6. Topology Definitions

Definition 6 (Simplicial Complexes) *A k-*simplex *C is the convex hull of $k+1$ affinely independent vectors. A* simplicial complex *$\mathcal{K}$ is a collection of simplices such that for every $C \in \mathcal{C}$ every face of C is in $\mathcal{K}$ and for every $C_1, C_2 \in \mathcal{K}$, if $C_1 \cap C_2$ is not empty then $C_1 \cap C_2$ is a face of both. The d-*skeleton *of $\mathcal{K}$, denoted $\mathcal{K}^{(d)}$, is the simplicial subcomplex of $\mathcal{K}$ consisting of simplices of dimension at most d.*

Definition 7 (Filtration) *A filtration of simplicial complexes is a collection of nested simplicial complexes $G_0 \subset G_1 \subset \cdots$. The complex G_k is called the k^{th} level of the filtration.*

Definition 8 (Persistence) *Given a filtration $G_0 \subset G_1 \subset \cdots \subset G_k$, we can compute the homology groups for each G_i. Then, for any* feature *(homology class), we can compute the first level k at which the feature appears, called the birth of the feature, and the level at which the feature disappears, called the death. This collection of birth and death tuples is known as the* persistence diagram.

Definition 9 (Vietoris-Rips filtration) *Let $X = \{x_1, \ldots, x_n\}$ be a collection of data points and d a metric on X. Then for any $r \in \mathbb{R}_+$, the* Vietoris-Rips simplicial complex *$VR_r(X)$ is defined by*

$$VR_r(X) = \{[x_{i_1}, \ldots x_{i_k}] : \forall j, \ell,\, d(x_{i_j}, x_{i_\ell}) \leq r\}.$$

We call $\{VR_r(X)\}_{r \in \mathbb{R}_+}$ the Vietoris-Rips Filtration.

Definition 10 (Betti Curve) *Let P be a persistence diagram. The* Betti curve *$\beta : \mathbb{R} \to \mathbb{N}$ is a function where $\beta(r)$ is the number of features (counted with multiplicity) whose birth b and death d satisfy $b \leq r < d$.*

A.7. Prior Rewiring Works: SDRF, FOSR, and GTR

raphEigengap, Commute time, and Curvature.

In this section, we detail connections between oversquashing and different statistics. See Appendix A.7 for a description of prior work that we compare against. We start by setting up notation for the paper. Throughout the paper, $G = (V, E)$ will refer to a graph on the vertex set V with edges E. We shall have that G has n nodes and m edges. Let A denote the adjacency matrix of the graph, and let D denote the degree matrix of the graph. Then the *Combinatorial Laplacian* is $L(G) := D - A$. The *eigengap* or spectral gap $\lambda_2(G)$ is the second largest eigenvalue of the Combinatorial Laplacian $L(G)$. Finally, the *Cheeger constant* h_G is defined as $\min_{(C_1,C_2)} \frac{|cut(C_1,C_2)|}{|C_1||C_2|}$. Here *cut* $cut(C_1, C_2)$ is a disjoint partition of the nodes V and the *size* of a cut is $|cut(C_1, C_2)| := |\{(u, v) \in E, u \in C_1, v \in C_2\}|$. We can see that the Cheeger constant tells how connected the graph is, giving it a clear relation to oversquashing. However, the Cheeger constant is difficult to compute but can be well approximated by the eigengap.

$$\frac{\lambda_2(G)}{2} \leq h_G \leq \sqrt{2\lambda_2(G)} \text{ and } 2h_G \geq \lambda_2(G) \geq \frac{h_G^2}{2}.$$

Having a large $\lambda_2(G)$ results in a large Cheeger constant and a better connected graph. Methods such as [32] optimize for the eigengap.

Another measure that is believed to be related to oversquashing is commute time. Consider a random walk on G with transition probabilities given by $P = D^{-1}A$. The *hitting time* $H(i, j)$ is the expected time for a random walk starting at node i to hit node j. The *commute time* is $CT(i, j) = H(i, j) + H(j, i)$. These notions are very related to effective resistance of a graph and are directly optimized by some rewiring techniques [10].

The final property related to oversquashing is graph curvature. As discussed in [58], the Ricci curvature is a natural method for measuring information dispersion on a manifold. Similar to the Ricci curvature on manifolds, Ricci curvature has also been defined for graphs [48] and has been used for rewiring [58]. This has been formalized by recent work such as [45],

where the authors show that negative curvature results in the sharply decaying importance of distant nodes. Thus, increasing the curvature of the graphs helps alleviate oversquashing. Methods such as [58, 23] optimize for this quantity.

A.7.1. Prior Methods

We review the existing rewiring methods against which we compare our own method RelWire.

SDRF As we have seen, the curvature of a graph is related to oversquashing. Thus, [58] design a method to increase the curvature of negatively curved areas. However, the Ollivier Ricci curvature is computationally expensive to calculate, hence they approximate it using a notion called Balanced Forman Curvature $Ric(i,j)$. In [58] show that if $Ric(i,j) > k$ for all edges then we have that $\frac{k}{2} \leq h_G \leq \frac{\lambda_2}{2}$. Thus, showing the connection between the curvature and other quantities such as the eigengap and the Cheeger constant. They then create a method that finds the most negatively curved edge and then add the edge that increases the curvature of this edge the most.

FOSR In [32], they showed that if f is the second eigenvector for the normalized Laplacian $(I - D^{-1/2}AD^{-1/2})$) then adding an edge i, increases the second eigenvalue by

$$\frac{2f_i f_j}{\sqrt{1+d_i}\sqrt{1+d_j}} + 2\lambda_2 \left[f_i^2 \left(\frac{\sqrt{d_i}}{\sqrt{1+d_i}} - 1 \right) + f_j^2 \left(\frac{\sqrt{d_j}}{\sqrt{1+d_j}} - 1 \right) \right].$$

They use this method to design an algorithm FOSR, that maximizes the first order term.

GTR Another notion of relevance is the total resistance of a graph, G. Let L be the Combinatorial Laplacian of a graph. Then the resistance $R(i,j)$ between nodes i and j is given by $R(i,j) = (1_i - 1_j)^T L^\dagger (1_i - 1_j)$. Here 1_i is the indicator vector for the ith node and $L^\dagger$ is the pseudoinverse of the Combinatorial Laplacian. The total resistance R_{tot} is $R_{tot} = \sum_{i,j} R(i,j)$. Then the biharmonic distance $B(i,j)$ between nodes i and j is given by $B(i,j) = \sqrt{(1_i - 1_j)^T (L^\dagger)^2 (1_i - 1_j)}$.

[10] show that the increase in the total resistance of adding an edge (i,j) is given by $\frac{B(i,j)^2}{1+R(i,j)}$. Hence they design a method GTR that maximizes this quantity. This quantity is related to the eigengap as well [10], where the maximum resistance between any two pairs of nodes R_{max} is bounded by

$$\frac{1}{n\lambda_2} \leq R_{max} \leq \frac{1}{\lambda_2}.$$

A.8. Run Time

Time complexity. Let n be the number of nodes in the graph, m be the number of edges we start with, and k the number of edges we want to add. For RelWire, there are 4 steps. The first step is the All Pair Shortest Path, which takes $O(n^2 \log(n) + nm)$. Next is to determine independence, this can be done naively in $O(m^2 n)$ time. Specifically for each pair of edges E_1, E_2, we look at all the projections and see if the map $\pi_{E_1} \times \pi_{E_2} : G \to E_1^{(0)} \times E_2^{(0)}$

is surjective. Finally, determining which edges to add can be done in $O(k\log(m))$ time. Thus the total time complexity is $O(n^2\log(n) + m^2n + k\log(m))$.

In addition to time complexity, we see the amount of time taken in practice. We note that different methods require different forms of computation. For SDRF, FOSR, and GTR we use pytorch on a machine with V100 GPU and 50GB RAM on Google Cloud. For RelWire, python is not the best language due to the combinatorial nature of the method, hence we use Julia. For this we use a personal laptop with an i5 processor, no GPU, and 8gb of RAM.

For practical use cases, we would need to do a hyperparameter search for the number of edges added. Hence we see how much time it takes to get rewired graphs for each number of edges from 1 to k.

Dataset	**Added Edges**	**GTR**	**FoSR**	**SDRF**	**RelWire**
Wisconsin	300	$<$ 1 second	$\sim$ 20 seconds	$\sim$ 12 minutes	$<$ 1 second
Wisconsin	3000	$<$ 1 second	$\sim$ 8.5 minutes	$>$ 1 hour	$<$ 1 second
Cora	1000	$\sim$ 6 seconds	$>$ 1 hour	$>$ 1 hour	$\sim$ 7 minutes
CiteSeer	1000	$\sim$ 4 seconds	$>$ 1 hour	$>$ 1 hour	$\sim$ 6 minutes

Table 5: Runtime Comparison of Different Algorithms on Various Datasets for different number of added edges.

From the above we can see that GTR is fastest. However, the point of this experiment is not to determine which method is fastest but as to whether RelWire can scale well. Here we can clearly see that the method does scale well.

Implementation notes: We use the official implementation go GTR and FOSR, and the implementation of SDRF from the LOG 2022 rewiring tutorial.

A.9. Raw Tables

As mentioned, we do extensive experimentation. Specifically, we consider 6 comparison methods (7 methods, including ours), 11 datasets, 4 different GNN architectures, and 3 different depths for each network. Hence, we have 924 data points for comparison. For each of these 924 experimental settings, we did five trials. To succinctly present these and interpret the results, we presented the results in Tables 2, 3, and 4 in the paper. Here, we provide the raw tables.

Here, we present one table for each of the 12 architectures (model type and depth). We present the test mse/accuracy for the trial with the best validation mse/accuracy. Note that we are doing graph regression for Peptides, Lipo, BACE, ESOL, and ZINC. So we present MSE, and smaller is better. We are doing graph classification for IMDB, MUTAG, REDDIT, PROTEIN, COLLAB, and ENZYMES. So, we present classification accuracy, and larger is better.

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
PeptidesStruct	0.635	NA	0.634	0.528	0.556	0.500	0.566
Lipo	1.571	1.525	1.567	1.534	1.539	1.563	1.532
BACE	0.224	0.231	0.235	0.225	0.222	0.221	0.219
ESOL	1.639	1.547	1.650	1.291	1.282	1.482	1.748
Zinc	4.951	5.937	5.620	5.268	5.096	4.695	4.380
IMDB	0.500	0.550	0.530	0.570	0.420	0.510	0.480
Reddit	0.650	NA	0.555	0.630	0.625	0.645	0.680
Collab	0.622	NA	0.296	0.614	0.596	0.596	0.606
Mutag	0.842	0.895	0.842	0.895	0.842	0.895	0.842
Proteins	0.736	0.745	0.566	0.736	0.736	0.745	0.728
Enzymes	0.267	0.183	0.233	0.267	0.233	0.283	0.233

Table 6: GCN Depth 3

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
PeptidesStruct	0.503	NA	0.548	0.440	0.456	0.391	0.502
Lipo	1.710	1.529	1.715	1.635	1.601	1.612	1.605
BACE	0.212	0.228	0.228	0.217	0.212	0.199	0.206
ESOL	1.210	1.369	1.532	1.374	1.223	1.210	1.322
Zinc	4.874	5.776	5.753	4.439	4.709	4.948	4.823
IMDB	0.470	0.430	0.450	0.590	0.410	0.450	0.430
Reddit	0.615	NA	0.560	0.650	0.570	0.595	0.655
Collab	0.628	NA	0.258	0.606	0.608	0.614	0.654
Mutag	0.842	0.842	0.842	0.842	0.842	0.842	0.842
Proteins	0.705	0.753	0.566	0.736	0.745	0.720	0.705
Enzymes	0.250	0.200	0.233	0.233	0.233	0.217	0.233

Table 7: GCN Depth 4

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
PeptidesStruct	0.494	NA	0.535	0.416	0.434	0.358	0.445
Lipo	1.756	1.525	1.738	1.648	1.630	1.653	1.695
BACE	0.192	0.205	0.226	0.208	0.198	0.188	0.199
ESOL	0.943	1.408	1.410	1.202	1.635	1.254	1.232
Zinc	4.749	5.025	5.262	4.622	4.439	4.561	4.422
IMDB	0.440	0.460	0.480	0.480	0.470	0.510	0.520
Reddit	0.580	NA	0.560	0.635	0.645	0.690	0.615
Collab	0.634	NA	0.220	0.636	0.618	0.626	0.616
Mutag	0.842	0.842	0.842	0.789	0.842	0.842	0.789
Proteins	0.711	0.753	0.566	0.720	0.696	0.688	0.705
Enzymes	0.250	0.150	0.233	0.250	0.183	0.200	0.217

Table 8: GCN Depth 5

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
PeptidesStruct	0.496	NA	0.655	0.389	0.374	0.308	0.451
Lipo	1.774	1.598	1.550	1.748	1.705	1.765	1.693
BACE	0.173	0.227	0.233	0.176	0.168	0.158	0.174
ESOL	1.177	0.879	1.772	0.980	0.905	1.032	0.983
Zinc	3.901	6.461	5.380	4.322	3.986	4.246	3.747
IMDB	0.540	0.400	0.480	0.490	0.520	0.570	0.540
Reddit	0.675	NA	0.570	0.570	0.640	0.670	0.655
Collab	0.578	NA	0.504	0.564	0.598	0.568	0.570
Mutag	0.789	0.842	0.895	0.842	0.789	0.842	0.842
Proteins	0.680	0.793	0.566	0.703	0.695	0.711	0.720
Enzymes	0.350	0.217	0.200	0.383	0.367	0.317	0.317

Table 9: GraphConv Depth 3

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
PeptidesStruct	0.471	NA	0.570	0.369	0.346	0.285	0.417
Lipo	2.074	1.596	1.646	2.023	1.892	1.950	1.971
BACE	0.134	0.279	0.229	0.164	0.165	0.157	0.191
ESOL	0.638	1.282	1.599	0.896	0.799	0.863	1.128
Zinc	3.142	9.793	5.756	3.548	3.837	3.559	3.435
IMDB	0.510	0.420	0.520	0.560	0.430	0.530	0.560
Reddit	0.685	NA	0.540	0.730	0.725	0.710	0.685
Collab	0.614	NA	0.514	0.622	0.620	0.626	0.602
Mutag	0.842	0.684	0.842	0.842	0.789	0.842	0.789
Proteins	0.758	0.668	0.566	0.742	0.710	0.711	0.693
Enzymes	0.367	0.183	0.117	0.333	0.333	0.333	0.400

Table 10: GraphConv Depth 4

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
PeptidesStruct	0.446	NA	0.543	0.362	0.331	0.275	0.404
Lipo	2.147	8.155	1.718	2.104	1.985	2.193	2.130
BACE	0.125	0.238	0.232	0.160	0.183	0.183	0.182
ESOL	1.415	3.061	1.427	0.728	0.852	0.682	0.978
Zinc	3.279	125.248	5.584	3.436	3.422	3.265	3.085
IMDB	0.480	0.520	0.460	0.540	0.480	0.600	0.510
Reddit	0.705	NA	0.560	0.705	0.725	0.710	0.750
Collab	0.620	NA	0.504	0.592	0.630	0.608	0.578
Mutag	0.895	0.737	0.842	0.737	0.789	0.842	0.789
Proteins	0.725	0.580	0.574	0.767	0.727	0.727	0.727
Enzymes	0.533	0.200	0.117	0.367	0.367	0.317	0.350

Table 11: GraphConv Depth 5

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
PeptidesStruct	0.519	NA	0.629	0.463	0.507	0.434	0.539
Lipo	1.673	1.616	1.573	1.663	1.682	1.639	1.667
BACE	0.206	0.229	0.233	0.218	0.204	0.215	0.214
ESOL	1.334	1.464	1.675	1.500	1.037	1.324	1.867
Zinc	4.898	5.968	5.066	5.567	4.424	4.989	4.345
IMDB	0.400	0.410	0.530	0.480	0.440	0.520	0.450
Reddit	0.605	NA	0.560	0.650	0.650	0.605	0.585
Collab	0.496	NA	0.362	0.568	0.540	0.516	0.548
Mutag	0.842	0.842	0.842	0.737	0.842	0.842	0.789
Proteins	0.720	0.745	0.566	0.728	0.736	0.736	0.711
Enzymes	0.250	0.167	0.200	0.233	0.200	0.200	0.233

Table 12: GAT Depth 3

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
PeptidesStruct	0.518	NA	0.549	0.406	0.407	0.375	0.460
Lipo	1.788	1.610	1.695	1.863	1.863	1.776	1.796
BACE	0.163	0.238	0.232	0.200	0.193	0.192	0.205
ESOL	0.805	1.087	1.472	1.443	1.115	0.925	1.332
Zinc	4.563	5.937	5.177	5.023	4.619	4.955	4.554
IMDB	0.480	0.610	0.510	0.510	0.480	0.510	0.580
Reddit	0.645	NA	0.560	0.675	0.610	0.570	0.600
Collab	0.598	NA	0.306	0.578	0.554	0.522	0.564
Mutag	0.842	0.842	0.842	0.842	0.842	0.842	0.842
Proteins	0.695	0.753	0.566	0.711	0.688	0.696	0.671
Enzymes	0.217	0.183	0.233	0.233	0.267	0.250	0.267

Table 13: GAT Depth 4

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
PeptidesStruct	0.502	NA	0.539	0.396	0.393	0.355	0.424
Lipo	1.907	1.676	1.736	1.935	1.756	1.869	1.831
BACE	0.186	0.224	0.231	0.182	0.204	0.204	0.196
ESOL	1.042	1.407	1.378	1.383	1.001	0.894	1.293
Zinc	3.851	5.131	5.681	4.459	4.379	4.435	4.182
IMDB	0.490	0.510	0.460	0.490	0.480	0.570	0.650
Reddit	0.715	NA	0.535	0.630	0.630	0.640	0.605
Collab	0.548	NA	0.346	0.558	0.592	0.550	0.546
Mutag	0.842	0.789	0.842	0.737	0.842	0.789	0.842
Proteins	0.688	0.738	0.566	0.705	0.688	0.688	0.703
Enzymes	0.217	0.217	0.250	0.217	0.267	0.267	0.267

Table 14: GAT Depth 5

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
PeptidesStruct	0.480	NA	0.540	0.383	0.362	0.305	0.434
Lipo	1.823	1.552	1.708	1.723	1.670	1.723	1.669
BACE	0.148	0.236	0.237	0.166	0.163	0.144	0.173
ESOL	1.585	0.886	1.520	1.035	1.180	1.354	1.529
Zinc	4.080	4.966	5.598	4.511	4.179	4.234	4.838
IMDB	0.490	0.500	0.500	0.520	0.520	0.440	0.510
Reddit	0.635	NA	0.590	0.650	0.655	0.705	0.660
Collab	0.514	NA	0.474	0.582	0.564	0.566	0.564
Mutag	0.842	0.684	0.842	0.842	0.737	0.842	0.789
Proteins	0.680	0.793	0.566	0.693	0.710	0.718	0.680
Enzymes	0.300	0.167	0.167	0.217	0.233	0.317	0.267

Table 15: GIN Depth 3

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
PeptidesStruct	0.457	NA	0.564	0.371	0.346	0.283	0.413
Lipo	1.704	1.541	1.742	1.851	1.845	1.846	1.959
BACE	0.138	0.279	0.242	0.165	0.177	0.176	0.183
ESOL	1.469	0.954	1.494	1.033	0.934	1.133	1.620
Zinc	3.630	5.117	5.341	4.019	4.031	3.967	4.235
IMDB	0.480	0.510	0.450	0.520	0.490	0.480	0.500
Reddit	0.710	NA	0.625	0.625	0.640	0.660	0.670
Collab	0.650	NA	0.496	0.636	0.618	0.642	0.622
Mutag	0.842	0.684	0.842	0.842	0.789	0.895	0.789
Proteins	0.703	0.777	0.566	0.710	0.727	0.727	0.693
Enzymes	0.300	0.150	0.167	0.300	0.283	0.317	0.400

Table 16: GIN Depth 4

Dataset	None	Full	Empty	RelWire	FOSR	GTR	SDRF
PeptidesStruct	0.462	NA	0.566	0.366	0.340	0.284	0.408
Lipo	1.973	1.600	1.731	1.934	1.807	1.951	1.861
BACE	0.147	0.247	0.239	0.195	0.177	0.198	0.183
ESOL	1.284	1.848	1.443	0.797	0.691	0.976	1.114
Zinc	3.639	6.103	5.022	4.019	3.707	3.668	3.776
IMDB	0.590	0.530	0.510	0.540	0.540	0.500	0.570
Reddit	0.650	NA	0.550	0.695	0.735	0.710	0.680
Collab	0.650	NA	0.500	0.656	0.648	0.656	0.674
Mutag	0.895	0.737	0.842	0.789	0.789	0.842	0.895
Proteins	0.718	0.739	0.574	0.711	0.710	0.735	0.742
Enzymes	0.400	0.167	0.183	0.267	0.333	0.350	0.333

Table 17: GIN Depth 5

Proceedings Track