ON THE ROLE OF STRUCTURE IN HIERARCHICAL GRAPH NEURAL NETWORKS

Luca Sbicego, Sevda Öğüt, Manuel Madeira, Yiming Qin, Dorina Thanou & Pascal Frossard
LTS4, EPFL, Lausanne, Switzerland

ABSTRACT

Hierarchical Graph Neural Networks (GNNs) integrate pooling layers to generate graph representations by progressively coarsening graphs. These GNNs are provably more expressive than traditional GNNs that solely rely on message passing. While prior work shows that hierarchical architectures do not exhibit empirical performance gains, these findings are based on small datasets where structure-unaware baselines often perform well, limiting their generalizability. In this work, we comprehensively investigate the role of graph structure in pooling-based GNNs. Our analysis includes: (1) reproducing previous studies on larger, more diverse datasets, (2) assessing the robustness of different architectures to structural perturbations of the graphs at varying depths of the network layers, and (3) comparing against structure-agnostic baselines. Our results confirm previous findings and demonstrate that they hold across newly tested datasets, even when graph structure is meaningful for the task. Interestingly, we observe that hierarchical GNNs exhibit improved performance recovery to structural perturbations compared to their flat counterparts. These findings highlight both the potential and limitations of pooling-based GNNs, motivating the need for more structure-sensitive benchmarks and evaluation frameworks.

1 INTRODUCTION

Modern pooling-based GNN architectures often interleave graph convolutions with graph pooling layers to successively reduce the graph size by exploiting structural connectivity and node features. These hierarchical GNNs produce increasingly coarser representations at each layer, enabling the network to capture both localized and global patterns in order to create graph-level representations (Liu et al., 2023).

While the representations learned by hierarchical GNNs are provably more expressive than their flat (non-pooling-based) counterparts (Bianchi & Lachi, 2023; Lachi et al., 2023), empirical evidence indicates that pooling layers often fail to enhance predictive performance. In particular, Mesquita et al. (2020) show that hierarchical GNNs tend to learn homogeneous node representations in the initial convolutions, even before the first pooling operator is applied, which they identify as the reason why these models perform on par with both their randomized variants and flat GNNs on graph-level tasks. However, this observation and the derived conclusions focus solely on the pooling methods themselves, overlooking potential inherent dataset limitations. For example, most of these datasets consist of relatively small graphs, while pooling would intuitively be expected to work better for large graphs, where its fine-to-coarse processing approach could potentially mitigate well-known issues in flat GNNs, such as oversquashing. Additionally, Errica et al. (2022) have recently shown that the graph structure is of limited relevance in most of these commonly benchmarked datasets since structure-agnostic baselines already outperform GNNs. Thus, the existing analyses lack the granularity needed to disentangle whether the observed underperformance of pooling stems from the method itself or from dataset characteristics, such as limited structural complexity or scale.

In order to thoroughly investigate the role of data structure in the learning dynamics of pooling-based GNNs, we conduct an extensive analysis designed to overcome potential limitations in previous studies. Specifically, we (1) evaluate the work by Mesquita et al. (2020) on larger and more diverse datasets; (2) assess the robustness of different architectures to structural perturbations by permuting graph edges at varying network depths and analyzing their impact on predictive performance; and (3)

include comparisons with structure-agnostic baselines to test the structural dependency of common benchmarks.

Our results indicate that common benchmarks often require limited structural information, which can be largely captured by the initial convolutional layers. As a result, structure-agnostic baselines frequently outperform more complex GNNs, both with and without pooling. Moreover, we find that hierarchical GNNs do not consistently outperform flat GNNs, even when graph structure is informative. Nevertheless, they demonstrate a greater capacity to recover from structural perturbations across tested operators and datasets, ultimately enhancing GNN robustness.

2 Assessing the Relevance of Graph Structure

2.1 Extending Prior Analysis

Building on Mesquita et al. (2020), we expand the analysis to a broader range of datasets, including both overlapping and newly introduced ones, to assess the impact of dataset choice on previous conclusions. We illustrate this diversity in terms of graph size in Appendix D, where we also report our replication results. Additionally, to broaden the coverage of pooling schemes, we extend the original study, which focuses on dense pooling operators, by also evaluating sparse pooling methods (Knyazev et al., 2019; Grattarola et al., 2022). Overall, we confirm their findings on the original settings and show that similar trends hold across the newly tested datasets and pooling schemes.

2.2 Isolating Structural Information: Graph Randomization Experiment

While Mesquita et al. (2020) analyzed pooling by replacing hierarchical operators with randomized counterparts, we take a complementary approach by directly randomizing the input graph structure, i.e., randomize edge connectivity by replacing the groundtruth edge indices with uniformly random ones (more details in Appendix B). This allows us to disentangle the role of structural information from the effect of pooling operators themselves.

Experimental Design. In our setup, we follow the standard network architecture used by Mesquita et al. (2020), in which the input graph $\mathcal{G}$ is passed through an initial convolution layer before entering L pooling blocks. Each block consists of a convolution layer followed by a pooling operator. The pooled graph undergoes a final convolution, and the resulting representation is fed into a two-layer MLP for graph classification. As depicted in Figure 1, we randomly permute all the edges of each graph at different depths within the architecture, including a baseline with no randomization:

- **REGULAR**: The original edge connectivity is used, no randomization is applied.
- **RANDOM-0**: Edges in $\mathcal{G}$ are randomized before the initial convolution layer. This corresponds to predicting without any information on graph structure.
- **RANDOM-1**: Edges in $\mathcal{G}$ are randomized after the initial convolution. Only the first convolution uses the original graph structure.
- **RANDOM-2**: Edges in $\mathcal{G}$ are randomized after the second convolution, i.e., after the convolution of the first pooling block.
- **RANDOM-3**: Edges in the pooled graph $\mathcal{G}'$ are randomized after the third convolution, i.e., after the first convolution of the second pooling block.

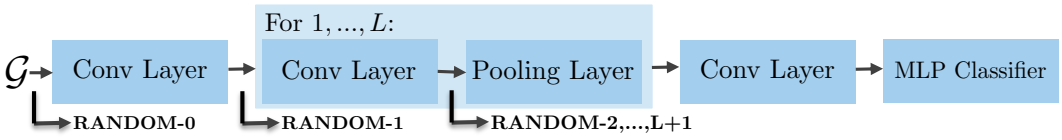


Figure 1: Schematic visualization of the experiment setup.

Model Choice. To ensure broad coverage of potential hierarchical GNN schemes, we again include both sparse and dense pooling operators. In particular, we choose Top-K pooling (Gao & Ji, 2019) as a sparse and simple scheme, and MinCutPool (Bianchi et al., 2020) as a dense and

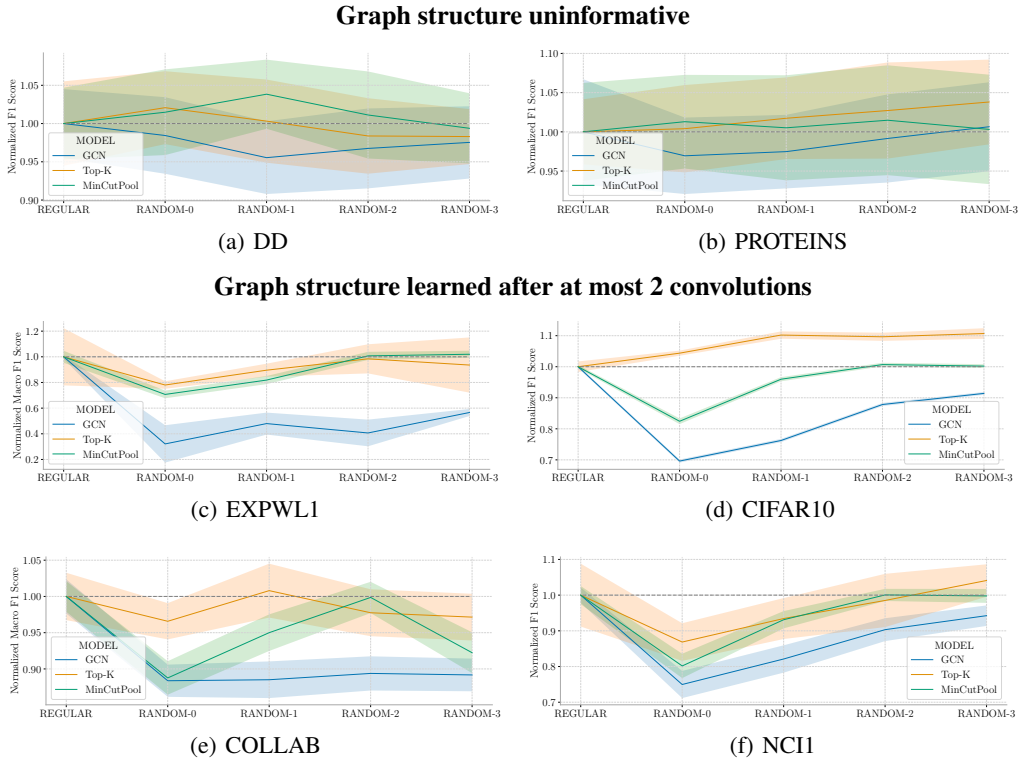


Figure 2: F1 score for binary tasks and macro F1 score for multi-class tasks $\pm$ std for different randomization schemes across six datasets. Scores are normalized such that the REGULAR model has a performance of 1.0.

more powerful method. As a non-pooling baseline, we employ a standard GCN (Kipf & Welling, 2017). Importantly, as in the study of Mesquita et al. (2020), our aim is not to evaluate the absolute performance of these models, but to compare their relative performance when the input graph is randomized at different stages in the network. Therefore, we do not perform hyperparameter tuning.

Comparison to Structure-Unaware Baselines. Following previous work by Errica et al. (2022), we isolate the effect of graph structure on model performance by evaluating structure-unaware baselines. Specifically, we use an MLP and a GNN with only self-loops for message passing. Note that for datasets without node features, the LocalDegreeProfile (LDP) (Cai & Wang, 2018) is added as a feature vector for these baselines. Additionally, we introduce NOD-EDG, which leverages the number of nodes and edges as features for Gradient Boosted Decision Trees (Friedman, 2001) to make non-linear predictions based on this simple 2D feature space.

Setup. We train the aforementioned models with 20 random seeds and test each of the randomization stages described above. We consider nine popular benchmarking datasets spanning different sizes and application domains: MalNet-Tiny, REDDIT-MULTI-12K, DD, CIFAR10, EXPWL1, COLLAB, PROTEINS, NCI1 and IMDB-BINARY (Dwivedi et al., 2023; Morris et al., 2020; Freitas et al., 2021; Bianchi & Lachi, 2023). Further details regarding used datasets and our implementation can be found in Appendix A and B.

3 Experimental Results

In this section, we present our main results. Figure 2 shows the line plots achieved by GCN, Top-K pooling, and MinCutPool for each dataset and for each randomization experiment. Note that the F1 score on the y-axis is *normalized* by the performance of the non-randomized variant (*REGULAR*), such that its corresponding entry is always 1. Table 1 demonstrates the performance of hierarchical and flat GNNs as well as the structure-agnostic baselines on the *REGULAR* experiment. We report the F1 score of the positive class for binary classification tasks and the macro F1 score in the multi-class classification setting. In what follows, we discuss our main experimental findings.

Table 1: F1 score for binary tasks and macro F1 score for multi-class tasks (avg ± std over 20 runs) of tested models with no randomization. Best performing model(s) by at least one std (two-way) is **bolded**. Missing entries indicate running times longer than 3 days. LDP is used for MLP and GNN-SL in datasets with no node features (MalNet-T, RDT-12K, EXPWL1, COLLAB, IMDB-B).

	MalNet-T	RDT-12K	DD	CIFAR10	EXPWL1	COLLAB	PROT.	NCI1	IMDB-B
Top-K	58.7 ± 11.0	27.5 ± 6.5	68.3 ± 5.5	32.4 ± 1.7	68.8 ± 22.3	64.0 ± 3.3	62.2 ± 4.2	69.3 ± 8.8	63.1 ± 7.2
MinCut	–	–	71.4 ± 4.7	39.6 ± 0.5	89.5 ± 4.7	**73.1 ± 2.1**	**65.3 ± 6.2**	**78.5 ± 2.5**	**71.2 ± 5.4**
GCN	53.5 ± 1.3	41.5 ± 2.4	70.3 ± 4.5	37.8 ± 0.4	**97.8 ± 1.5**	64.9 ± 2.4	**67.8 ± 6.7**	**77.4 ± 2.1**	63.0 ± 3.8
MLP	87.3 ± 1.3	**45.3 ± 1.8**	63.9 ± 8.5	42.5 ± 1.0	94.6 ± 3.6	**73.6 ± 2.2**	58.8 ± 6.1	62.3 ± 2.7	**71.3 ± 5.5**
GNN-SL	**89.3 ± 0.9**	**46.8 ± 1.7**	**69.2 ± 4.3**	**45.9 ± 0.7**	**96.8 ± 2.8**	**75.2 ± 2.2**	58.4 ± 6.0	63.1 ± 3.9	**71.8 ± 6.3**
NOD-EDG	74.6 ± 1.9	33.3 ± 1.4	**73.6 ± 4.4**	10.1 ± 0.6	65.9 ± 2.2	71.7 ± 1.6	**69.2 ± 4.2**	62.6 ± 2.3	68.0 ± 2.5

In most datasets, structure is either uninformative or can be learned after the initial convolutions. For DD and PROTEINS (Figures 2(a) and 2(b)), we find that randomizations at different depths do not significantly affect performance. This suggests that node features or basic graph statistics, such as the number of nodes and edges, suffice for strong performance, as indicated by the NOD-EDG baseline in Table 1. In contrast, for EXPWL1, CIFAR10, COLLAB, and NCI1 (Figures 2(c)-2(f)), the *RANDOM-0* version shows a notable drop in performance, highlighting the importance of graph structure. However, as randomization is applied at later stages in the network, the original performance is gradually restored, indicating that the graph structure is captured early in the convolutional layers, consistent with Mesquita et al. (2020). These findings are further supported by additional results in Appendix C.

Structure-agnostic baselines already achieve comparable, and in some cases superior, performance compared to GNNs. In Table 1, we observe that structure-unaware architectures perform exceptionally well, ranking among the best models in eight out of nine datasets. Notably, the NOD-EDG baseline achieves strong performance across several datasets, despite not leveraging explicit graph topology or node features (whereas MLP and GNN-SL either use the available node features or are enriched with LDP features).

Hierarchical GNNs are not superior to flat GNNs even when graph structure is informative. As shown in Table 1, hierarchical GNNs do not outperform flat GNNs or simpler baselines, even on datasets where structural information is relevant (e.g., NCI1).

Pooling may make GNNs more robust towards perturbations in graph structure. For datasets where the structure is informative, Figures 2(c)-2(f), pooling-based methods consistently show better relative performance compared to flat GNNs when the graph structure is randomized. The performance achieved for *RANDOM-0* is markedly better for Top-K and MinCut, and predictive accuracy is recovered in earlier randomization levels, whereas the GCN often fails to reach the performance of the *REGULAR* variant. This may suggest that pooling operators, by aggregating information in a fine-to-coarse fashion, are less sensitive to local perturbations, potentially enabling them to reconstruct the original graph representation. However, further investigation is needed to fully understand and confirm this effect, which we leave for future work.

4 Conclusion

In this work, we address the conflicting theoretical and empirical evidence regarding the utility of pooling schemes in graph-level tasks. We replicate previous results and extend them to larger and more diverse datasets. We observe that graph pooling has minimal impact on performance, supporting critiques of its practical relevance in common datasets. By isolating the effect of graph structure on the performance of pooling and non-pooling GNNs, we show that many popular benchmarks require limited structural information, with structure-agnostic models performing on par with, or even outperforming, GNNs. The widespread use of these datasets limits progress in understanding the true utility of pooling. Future work should focus on developing and validating large-scale benchmarks that better capture complex topologies. Notably, our analysis offers preliminary evidence that hierarchical GNNs may be more robust to graph structure perturbations, highlighting an exciting avenue for future research.

REFERENCES

Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI*, 2021.

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

Filippo Maria Bianchi and Veronica Lachi. The expressive power of pooling in graph neural networks. In *Advances in Neural Information Processing Systems*, 2023.

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral Clustering with Graph Neural Networks for Graph Pooling. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Chen Cai and Yusu Wang. A simple yet effective baseline for non-attributed graph classification. *arXiv preprint arXiv:1811.03508*, 2018.

Esther Danenberg, Helen Bardwell, Vito RT Zanotelli, Elena Provenzano, Suet-Feung Chin, Oscar M Rueda, Andrew Green, Emad Rakha, Samuel Aparicio, Ian O Ellis, et al. Breast tumor microenvironment structures are associated with genomic features and clinical outcome. *Nature genetics*, 54(5):660–669, 2022.

Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Weighted Graph Cuts without Eigenvectors A Multilevel Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11), 2007. doi: 10.1109/TPAMI.2007.1115.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A Fair Comparison of Graph Neural Networks for Graph Classification, 2022. arXiv:1912.09893.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Scott Freitas, Yuxiao Dong, Joshua Neil, and Duen Horng Chau. A large-scale database for graph representation learning. *arXiv preprint arXiv:2011.07682*, 2021.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In *International Conference on Machine Learning*, pp. 2083–2092. PMLR, 2019.

Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi. Understanding Pooling in Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Amir Hosein Khasahmadi, Kaveh Hassani, Parsa Moradi, Leo Lee, and Quaid Morris. Memory-based graph networks. In *International Conference on Learning Representations*, 2020.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. In *Advances in Neural Information Processing Systems*, 2019.

Veronica Lachi, Alice Moallemy-Oureh, Andreas Roth, and Pascal Welke. Graph pooling provably improves expressivity. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*, 2023.

Chuang Liu, Yibing Zhan, Jia Wu, Chang Li, Bo Du, Wenbin Hu, Tongliang Liu, and Dacheng Tao. Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities, 2023. arXiv:2204.07321.

Diego Mesquita, Amauri Souza, and Samuel Kaski. Rethinking pooling in graph neural networks. *Advances in Neural Information Processing Systems*, 33:2220–2231, 2020.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14:347–375, 2008.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Advances in Neural Information Processing Systems*, 2018.

Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954*, 2019.

A Dataset Details

Table 2 summarizes the characteristics of the datasets used in this study, including both datasets from Mesquita et al. (2020) and additional ones introduced in our experiments. Note that for datasets without node features, the LocalDegreeProfile (Cai & Wang, 2018) is added as a feature vector for the MLP and GNN-SL baselines. Further details on each dataset are provided in the remainder of this section.

Table 2: Summary of dataset statistics and application domains

Domain	Dataset	# Graphs	Avg Nodes	Avg Edges	# Features	# Classes	Class Imbalance[1]
Histopathology	METABRIC	611	1,410.61	8,309.50	32	2	Slight Imbalance
Computer Science	MalNet-Tiny	4,814	1,371.19	1,314.40	0	5	Balanced
	EXPWL1	3,000	77	93	1	2	Balanced
Social	REDDIT-12K	11,929	391.41	456.89	0	11	Slight Imbalance
	COLLAB	5,000	74.49	2,457.22	0	3	Slight Imbalance
	IMDB-Binary	1,000	19.77	96.53	0	2	Balanced
Computer Vision	CIFAR10	60,000	117.63	470.53	3	10	Balanced
Molecular	NCI1	4,110	29.87	32.30	37	2	Balanced
Bioinformatics	DD	1,178	284.32	715.66	89	2	Slight Imbalance
	PROTEINS	1,113	39.06	72.82	3	2	Slight Imbalance

METABRIC: This is a popular digital pathology dataset, consisting of cell graphs derived from stained tissue samples of breast cancer patients. Nodes in these graphs represent relevant cells within the tumor microenvironment, while edges capture spatial or functional relationships between them (Danenberg et al., 2022). The graphs are large and highly variable in size and the graph-level task is to predict the estrogen receptor status. Although METABRIC is not a standard benchmarking dataset, it is included here due to the relevance of hierarchical graph pooling in digital pathology settings.

MalNet-Tiny: The MalNet dataset is a collection of function call graphs of malicious software that has been created to serve as a truly large and diverse benchmarking dataset for graph representation learning (Freitas et al., 2021). While the original dataset contains 1.2 million graphs averaging 15,000 nodes and 35,000 edges per graph, the authors also introduced a more manageable version called *MalNet-Tiny*, where the largest graph has 5,000 nodes and each graph belongs to one out of five malware types. We use this smaller dataset due to computational constraints in this work.

EXPWL1: The EXPWL1 dataset is specifically designed to evaluate the expressive power of GNNs in relation to the Weisfeiler-Lehman (WL) test (Bianchi & Lachi, 2023). Each graph represents a propositional formula. Variables and literals are represented as nodes, whereas edges represent the logical relationships. Nodes contain a single feature indicating whether they correspond to a clause or literals (Abboud et al., 2021). The predictive task is to determine whether a given formula is satisfiable or unsatisfiable. Given the dataset's construction, GNNs as expressive as the WL1 test are expected to achieve near 100% classification accuracy.

REDDIT-MULTI-12K: Each graph represents a discussion thread on the popular social news website Reddit. Nodes correspond to users and edges are drawn between users that responded to one other's comment. The graph-level task consists of predicting the topic (subreddit) the discussion originated from (Morris et al., 2020).

COLLAB: The graphs represent ego-networks of scientists conducting research within the fields of high energy physics, condensed matter physics, or astrophysics. The task is to predict the research domain based on the graph's topology (Morris et al., 2020).

IMDB-Binary: Similar to COLLAB, this dataset is constructed as the ego-network of actors based on IMDB data (Morris et al., 2020). The task is to predict the genre of a movie (action or romance) in which the actor in question participated.

[1]For binary labels, slight imbalance is at most 20%-80%. For multi-class, the ratio between the smallest and the largest class is at most 1-to-5.

Superpixel CIFAR10: Graphs for the CIFAR10 dataset are constructed by segmenting each image into superpixels using the SLIC algorithm (Achanta et al., 2012), which produces compact and dense regions. Each superpixel is treated as a node, with edges connecting adjacent superpixels. The task is to classify each graph into one of ten categories (Dwivedi et al., 2023).

NCI1: The NCI1 dataset consists of molecules collected from anti-cancer screen tests. Each atom corresponds to a node in the graph and atom types as well as other chemical information are represented as node features. Edges naturally denote the bonds between atoms (Morris et al., 2020). The task is predicting whether a given molecule is active or inactive in a cancer-screening test (Wale et al., 2008).

DD: The graphs encode protein structures, where nodes represent amino acids and edges denote spatial proximity between them (Morris et al., 2020). The graph-level task is to predict whether a given protein is an enzyme.

PROTEINS: In this dataset, nodes represent structural elements of proteins, such as helices, and sheets, while their features encode chemical and physical properties. Edges connect neighbors in the amino acid sequence or physical space (Borgwardt et al., 2005; Morris et al., 2020). The task is to predict whether a given protein graph corresponds to an enzyme.

B Experiment Design Details

All neural networks followed the standardized architecture shown in Figure 1. For common benchmarking datasets, we adopted the layer and hidden dimension settings from Mesquita et al. (2020), with minor adaptations for new datasets (see Table 3). The Adam optimizer was employed with early stopping. The initial learning rate was set to 0.001 and was progressively halved after 10 epochs of validation loss stagnation.

Table 3: Number of layers, hidden dimension, batch size for each dataset and model combination. We largely adopt parameter settings from Mesquita et al. (2020), with minor adaptations for new datasets.

	Top-K	**MinCut**	**GCN**	**MLP**
METABRIC	6, 128, 8	6, 64, 8	6, 128, 8	6, 64, 8
MalNet-Tiny	5, 128, 32	-	5, 128, 32	5, 64, 32
REDDIT-12K	4, 64, 64	4, 64, 16	4, 64, 64	4, 32, 64
DD	3, 64, 32	3, 64, 16	3, 64, 32	3, 32, 32
CIFAR10	4, 64, 256	4, 64, 512	4, 64, 256	4, 32, 256
EXPWL1	3, 64, 32	3, 64, 32	3, 64, 32	3, 32, 32
COLLAB	3, 64, 64	3, 64, 16	3, 64, 64	3, 32, 64
PROTEINS	3, 64, 64	3, 64, 16	3, 64, 64	3, 32, 64
NCI1	3, 64, 64	3, 32, 16	3, 64, 64	3, 32, 64
IMDB-BINARY	2, 64, 8	3, 32, 16	2, 64, 8	2, 32, 8

We use PyTorch Geometric (Fey & Lenssen, 2019) for our experiments, representing the structural connectivity of a graph with a matrix $\boldsymbol{E} \in \mathbb{R}^{2\times|E|}$, where $|E|$ is the total number of edges. Each column in this matrix specifies an edge, with the two rows indicating the source and target nodes, respectively. To randomize the graph structure for the experiment presented in Section 2, we draw two rows of node indices independently and uniformly at random, keeping the number of edges in the randomized graph the same as in the original graph, i.e., $\boldsymbol{E}_i \sim \mathcal{U}[1, |E|]$ for $i \in \{0, 1\}$.

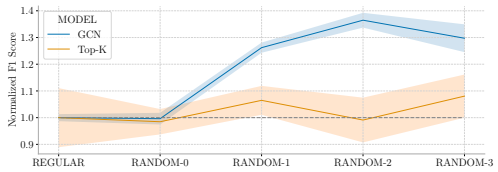
The randomization described above applies to experiments from *RANDOM-0* to *RANDOM-2*. However, in the *RANDOM-3* experiment, the graph structure is randomized after the initial pooling layer. Since all employed pooling schemes focus on clustering nodes, the number of edges in the pooled graph, $|E'|$, is unknown prior to training and can vary across graphs and models. Therefore, to prevent adding more edges than in the original graph, we estimate $|E'|$ based on the pooling scheme's behavior and the density of the unpooled graph, defined as $\textit{density} = \frac{|E|}{N(N-1)}$, where N is the number of nodes.

For MinCutPool, each graph is initially mapped to a standardized size with N_{max} nodes, where N_{max} is the number of nodes in the largest graph. The pooling layer reduces the number of nodes to $N' = N_{max} \times k$, where k is the pooling ratio. The number of edges in the pooled graph is then estimated as $|E'| = \frac{N'(N'-1)}{2} \times \textit{density}$, where *density* is calculated based on the original graph.

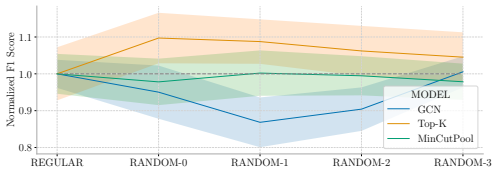
In the case of Top-K pooling, the number of nodes in the pooled graph, N', is determined individually for each graph, as the scheme is *adaptive*. The number of edges $|E'|$ is then computed in the same way as in MinCutPool using the estimated N' and *density*.

C Additional Results: Other Datasets

Graph structure largely irrelevant, but results vary by model type

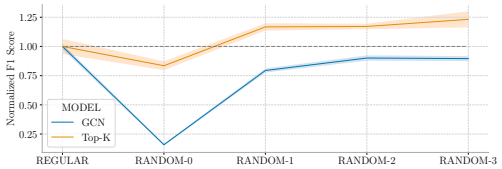


(a) MalNet-Tiny



(b) IMDB-B

Graph structure learned after at most two convolutions (additional example)



(c) REDDIT-MULTI-12K

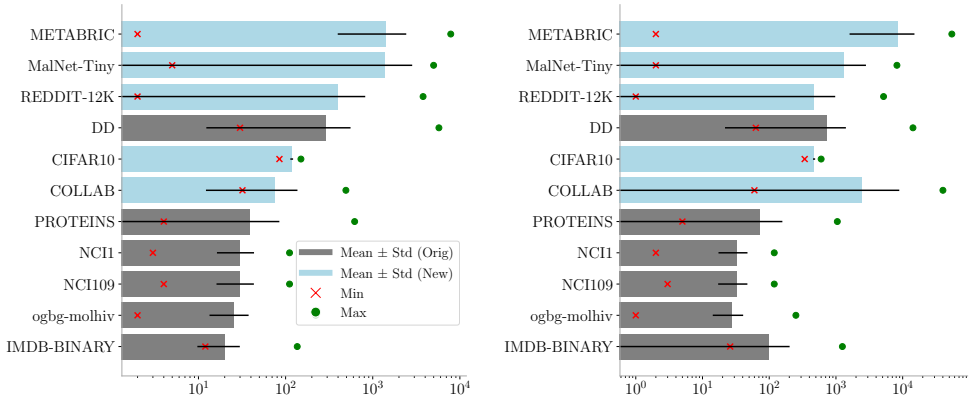
Figure 3: F1 score for binary tasks and macro F1 score for multi-class tasks $\pm$ std for different randomization schemes across datasets. Scores are normalized such that the REGULAR model has a performance of 1.0. Missing entries indicate running times longer than 3 days.

For the MalNet-Tiny and IMDB datasets (Figures 3(a) and 3(b)), graph structure has minimal relevance, as the *RANDOM-0* results are comparable to those of the *REGULAR* experiment. This indicates that models can achieve reasonable performance by relying solely on graph-level features such as the number of nodes and edges, given the absence of node attributes in both datasets. This interpretation is further supported by the performance of the NOD-EDG baseline in Table 1. Moreover, REDDIT-MULTI-12K (Figure 3(c)) serves as an additional example of a large dataset where the graph structure is learned in the early layers of the network. Note that we could not evaluate MinCutPool for all randomization levels due to computational constraints.

D Additional Results: Replication of Mesquita et al. (2020)

Mesquita et al. (2020) test the three dense graph pooling schemes of DiffPool (Ying et al., 2018), Graph Memory Networks (Khasahmadi et al., 2020), and MinCut (Bianchi et al., 2020) as well as a non-trainable scheme Graclus (Dhillon et al., 2007) on eight datasets. This experiment was inherently limited in scope, encompassing relatively small datasets from similar application domains and primarily focusing on dense pooling schemes (Bianchi & Lachi, 2023).

To validate and extend the findings of Mesquita et al. (2020), we replicate their experiments on a broader range of larger datasets from diverse application domains, where pooling layers may better exploit structural properties. Specifically, we include METABRIC, MalNet-Tiny, REDDIT-MULTI-12K, CIFAR10, and COLLAB, while retaining DD and NCI1 from the original study. Figure 4 compares the newly added datasets (blue) with those from the original study (gray) based on the average number of nodes and edges per graph.



(a) Number of Nodes (Log Scale) (b) Number of Edges (Log Scale)

Figure 4: Dataset overview visualizing the distribution of nodes and edges. Datasets colored in blue are newly added while datasets in gray are included in Mesquita et al. (2020).

Table 4: Accuracy (avg ± std) of pooling schemes compared to their randomized variants over 10 runs. Accuracies in **bold** are statistically significantly (p-value < 0.05) better than their counterpart. Missing numbers indicate OOM or running times longer than 3 days.

	METABRIC	MalNet-Tiny	REDDIT-12K	DD	CIFAR10	COLLAB	NCI1
Top-K	79.5 ± 4.4	56.4 ± 10.8	26.9 ± 6.1	73.5 ± 6.3	32.8 ± 1.3	69.4 ± 3.2	**77.4 ± 2.8**
Randomized	77.4 ± 0.0	**67.5 ± 5.9**	**36.8 ± 1.9**	74.7 ± 5.5	**37.9 ± 1.1**	72.2 ± 2.8	70.0 ± 2.1
HGP-SL	79.5 ± 5.1	-	-	74.1 ± 3.1	-	**74.8 ± 1.3**	70.9 ± 2.3
Randomized	80.2 ± 4.7	-	-	75.2 ± 4.5	-	72.5 ± 1.8	68.5 ± 3.3
MinCut	79.3 ± 4.0	-	-	75.5 ± 2.3	33.6 ± 12.5	75.9 ± 2.2	79.3 ± 2.2
Randomized	79.0 ± 5.6	-	-	76.1 ± 3.4	39.1 ± 1.3	73.9 ± 2.7	79.7 ± 2.5

Including larger datasets motivates the use of more scalable operators to efficiently replicate the experiment. Therefore, we consider two sparse pooling schemes: Top-K (Gao & Ji, 2019) and HGP-SL (Zhang et al., 2019). We also consider MinCut Pooling (Bianchi et al., 2020), a popular and relatively efficient dense pooling scheme from the original study.

Table 4 presents the accuracy of the three pooling schemes versus their randomized counterparts. For most dataset-operator combinations, the randomized variants perform on par or better than the original scheme. Thus, we largely replicate the results found by Mesquita et al. (2020). Top-K on NCI1 and HGP-SL on the COLLAB dataset present the only two exceptions for which one could argue that the original pooling operates perform better than their randomized counterparts.