When Are Graph Neural Networks Better Than Structure-Agnostic Methods?

Anonymous Author(s)
Affiliation
Address
email

Abstract

Graph neural networks (GNNs) are commonly applied to graph data, but their performance is often poorly understood. It is easy to find examples in which a GNN is unable to learn useful graph representations, but generally hard to explain why. In this work, we analyse the effectiveness of graph representations learned by shallow GNNs (2-layers) for input graphs with different structural properties and feature information. We expand on the failure cases by decoupling the impact of structural and feature information on the learning process. Our results indicate that GNNs' implicit architectural assumptions are tightly related to the structural properties of the input graph and may impair its learning ability. In case of mismatch, they can often be outperformed by structure-agnostic methods like multi-layer perceptrons.

1 Introduction

Graph neural networks (GNNs) have emerged as the default approach for graph representation learning in machine learning tasks. Despite this fact, GNN performance is often poorly understood and it is easy to find examples where a GNN is unable to learn a useful graph representation. In this work, we focus on analysing the underlying assumptions of GNN architectures that are key to determining their performance. We consider the problem of semi-supervised node classification. GNNs make class predictions using two sources of information: the individual node properties described by the feature vectors associated with each node, and structural information represented by the relationships (edges) between the different nodes. The main idea behind using graph representations is that relational information can provide additional information to solve the target task, e.g. friendship links in a social network can be predictive of a person's interests. Given that we are augmenting the node properties with additional information from node relations, one might assume that GNNs always outperform feature-only methods, such as multi-layer perceptrons (MLPs), that do not exploit any structural information. However, recent benchmark results [1] show state-of-the-art graph neural networks performing on-par or worse than basic MLPs on a variety of node classification tasks. These results hint at the fact that the use of structural information by GNNs is not always helpful and may even be detrimental to classification performance.

Our goal is to provide a methodical, empiric investigation of the use of structural vs. feature information by different GNN methods. We aim to show that, due to fundamental limitations, basic GNNs are unable to learn useful representations when their assumptions are not met, both on structure and feature levels, even when their depth is small. Furthermore, we provide evidence that more advanced GNN architectures can avoid this limitation, but may still fail to exploit structural information altogether and reduce to feature-only methods.

The main contributions of this work are two-fold: 1) the proposition of an empirical, model-agnostic method for decoupling feature and structure influence on GNN node classification performance; 2)

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

the disclosure of new empirical insights on how GNN performance can be hindered by input graph's structural properties and node features in cases when oversmoothing does not occur.

2 Methods

Our investigation of the cases when GNNs can and cannot learn useful node representations is conducted by constructing artificial graphs with certain engineered properties. In particular, we manipulate homophily and edge density (structural properties) and feature signal-to-noise ratio (SNR), due to their direct relation with the implicit assumptions of message-passing graph models.

We implement two operations designed to derive a set of mutations for each original graph. These operations are set to methodically destroy structure and/or feature information while preserving the remaining graph attributes. With this procedure, we aim to decouple the influence of feature and structure information in each node classification task and separately violate GNNs' assumptions of meaningful underlying structure and features. We finally compare the respective performances with those of node classification on the original graphs and those of a structure-agnostic baseline model (MLP). These experiments are further extended by extrapolating our insights to more challenging settings by means of several real-world datasets, commonly used as benchmarks for machine learning on graphs. The following subsections elaborate on the implementation details of our approach.

Artificial graphs generation Artificial graphs are generated using the stochastic block model (SBM) as implemented by Palowitch et al. [2], which enables control of certain graph properties, namely edge density, homophily, and feature SNR (a metric of feature homogeneity between classes, which equals to 1 in homogeneous scenarios and increases with heterogenity, i.e. as features become more separable). We generate graphs with 1000 nodes. Each node is assigned a label to create a class-balanced binary node classification problem, i.e., each graph comprises 500 nodes of each class. All artificially generated graphs are available as supplementary material[1] for repeatability.

Benchmarks Four real-world datasets with different properties are also selected to extend our experiments and for benchmarking purposes. Cora [3] and CiteSeer [3] are homophilic citation networks with 7 and 6 classes, respectively; Chameleon [4] and Texas [5] are heterophilic graphs, with 5 classes each, where nodes correspond to web pages and edges to hyperlinks between them. A summary of these datasets' properties can be found in the Appendix.

Graph mutations Each artificial and real-world graph is also submitted to transformations on the structural and feature levels: 1) *random connectivity* - shuffle columns of graph connectivity matrix; 2) *random feature assignment* - randomize the attribution of feature vectors across all nodes. With these operations, we conceive up to three mutations for each original graph whenever appropriate: a mutation with random connectivity but same feature distribution; another with same structural information but random feature assignment; and a final one with both transformations.

Models We consider the Graph Convolutional Networks (GCN) [6] as a base for all experiments. More complex layer and model types are also used to extend our analyses: reversible GCN (RevGCN) [7], due to its robustness to oversmoothing even for deep GNNs; APPNP [8] and FiLM convolution [9] which seem to perform adequately for input graphs that belong to different parts of the graph properties spectrum proposed in [2], where vanilla-GCNs do not.

All neural network architectures consist of a single linear layer for feature transformation into 8 channels, followed by 2 message-passing layers (or fully connected layers for the MLP baseline), and a node classification head (linear layer). We use the PyG library [10] and all experiments are run using GraphGym framework [11]. Reported results refer to performance (accuracy) on the test sets (best epoch on the validation set, averaged over 10 runs). No hyperparameter tuning is performed to facilitate the comparison of approaches. Configuration files are available as supplementary material for repeatability. Further implementation details can be found in the Appendix.

[1] URL blinded for review

Table 1: Node classification performance (accuracy) of GCN and MLP models on an artificial graph G with highly informative structure and features, and the mutations of G with random connectivity (uninformative structure), random feature assignment (uninformative features), and both (uninformative structure and features). Standard deviation values are shown in Table 6 of the Appendix.

		Structure			
		Informative		Uninformative	
Features	Informative	GCN	0.96	GCN	0.69
		MLP	0.92	MLP	0.90
	Uninformative	GCN	0.61	GCN	0.48
		MLP	0.50	MLP	0.50

3 Results

3.1 Does structure encoding always contribute to learning useful node representations?

Let us consider the simple case of binary node classification in a graph with both highly informative features (SNR = 1.5) and structure (homophily = 0.95; edge density = 0.06). Table 1 compares the performance of a 2-layer vanilla-GCN with that of a structure-agnostic model (MLP) on the original version of this graph and its mutations (uninformative versions).

While GCN exhibits adequate performance (superior to the MLP) when both structural and feature information are present, results show its evident drop when they lose either. Despite the fact that features are highly informative, GCN does not seem able to fully leverage them when the structure of the input graph was meaningless towards its inherent assumptions, leading to a significant loss of performance even relatively to its structure-agnostic counterpart. A similar drop is verified in the scenario where structure is preserved but feature information is lost, as GCNs are not able to aggregate neighborhood information in meaningful node representations, despite the highly informative structure of the input graph. We verify this behavior using shallow models of 2 message-passing layers, for which graph oversmoothing does not occur, as the appropriate performance in the informative scenario corroborates.

These results suggest that GCN models need both feature and structural information to be meaningful in order to learn useful node representations. When only one of these is present the model does not seem to be able to separate the useful from meaningless information. This result makes sense given the intuition of GCN as a smoothing operator [12]. Blindly aggregating either features of dissimilar nodes due to lack of structural information or combining non-informative features of similar nodes does not extract useful node descriptions.

3.1.1 Graphs with different properties

Given the empirical verification that GCN performance can be tightly related to feature information and structural properties of the input graph, we consider these attributes in separate methodical studies. Let us take a base graph with fixed characteristics (homophily = 0.8; edge density = 0.03; feature SNR = 1.2). Table 2 presents the node classification results on several versions of this graph that correspond to assigning it different connectivity matrices (and respective mutations). These matrices define structures of different homophily, while keeping density constant (and vice-versa). Analogously, Table 3 displays node classification results for versions of the base graph with different feature information, measured by its SNR; results for the respective random connectivity mutations are also shown.

Homophily The inspection of GCN's response to different homophily conditions reveals its adequate performance on the most and least homophilic original graphs. While adequate performance in the most heterophilic scenario might seem surprising at first glance, as GCN's limitations in dealing with such settings are well-know, it is not unexpected in our experiment. This behavior relates to the fact that our learning problem only considers two distinguishable types of nodes and has also been recently reported in other works [13]. Nodes are able to encode meaningful representations through neighborhood aggregation, despite most of their neighbors belonging to a different class,

Table 2: Node classification performance (accuracy) of GCN and MLP (baseline) models on artificial graphs with fixed features (SNR = 1.2) and different structural attributes: homophily (H), edge density (D_e). Results are shown for each original graph G and the respective mutations of G with random connectivity, random feature assignment, and both. A single MLP (feature-only) result is shown per feature transformation. Standard deviation values are shown in Table 7 of the Appendix.

	H	Structure		D_e	Structure		MLP (baseline)
		Original	Random		Original	Random	
Original Features	0.2	0.78	0.58	0.003	0.78	0.63	0.71
	0.5	0.60	0.61	0.03	0.81	0.59	
	0.8	0.81	0.59	0.15	0.79	0.57	
Random Feature Assignment	0.2	0.51	0.48	0.003	0.49	0.48	0.49
	0.5	0.53	0.50	0.03	0.49	0.49	
	0.8	0.49	0.49	0.15	0.52	0.49	

Table 3: Node classification performance (accuracy) of GCN and MLP (baseline) models on artificial graphs with fixed structure (homophily = 0.8; edge density = 0.03) and different feature signal-to-noise ratio (SNR). Results are shown for each original graph G and the respective mutations of G with random connectivity. Standard deviation values are shown in Table 8 of the Appendix.

SNR	Structure		MLP (baseline)
	Original	Random	
1.0	0.57	0.51	0.51
1.2	0.81	0.59	0.71
1.5	0.92	0.68	0.91
2.0	0.99	0.72	1.00

due to its consistency. While this outcome may not hold under different conditions (such as some multi-class problems), it also draws attention to the potential insufficiency of solely resorting to homophily-related assumptions to steer GNN architecture research endeavors, as discussed by recent works [13]. Similar to the previous highly informative artificial graph, we verify a significant loss of performance when structure and/or feature information of original graphs are destroyed, except for when homophily is close to 0.5 (mediocre performance on the original graph, on-par with the random structure mutation). This means that being as connected to nodes of a different class as to those of the same class produces an uninformative structure based on which GCN will perform local smoothing operations that will decrease feature expressivity and lead to poor node representations. These results verify that simply attributing a GCN's poor performance to graph heterophily may be insufficient, as some heterophilous graphs can encode relevant structure information while others do not.

Edge density The impact of edge density in node classification performance leads to useful insights on the feature-structure information co-dependence within GCN models. For the original version of the graphs, where both feature and structure information can be considered relevant for node distinction, the GCN model gets the most information out of a moderately dense structure. However, if we destroy its feature information, a more densely connected graph leads to the most useful node representations. On the contrary, if its structure information is meaningless, the most sparsely connected structure leads to the best results. This outcome is coherent with the architectural assumptions of message-passing approaches: if structural information is irrelevant, we should expect better node representations from the structures with fewer connections, as these lead to a minimal smoothing effect. Furthermore, it provides evidence that feature/structure trade-offs associated with graph-related tasks can pose particular learning challenges for GNNs.

Feature SNR Table 3 shows the impact of considering different levels of separability of node features when structure encodes useful information and when it does not. The results indicate a significant loss of performance for the random connectivity mutation in comparison with the original version, even in scenarios when base features are easily separable by a feature-only method. This

Table 4: Node classification performance (accuracy) of different GNN architectures and MLP (baseline) models on real-world benchmarks. Results are shown for each original graph G and the respective mutations of G with random connectivity, random feature assignment, and both. A single MLP (feature-only) result is shown per feature transformation. Standard deviation values are shown in Table 9 of the Appendix.

		Original Structure				Random Connectivity				
		GCN	RevGCN	APPNP	FiLM	GCN	RevGCN	APPNP	FiLM	MLP
Original Features	Informative*	0.96	0.96	0.95	0.96	0.69	0.92	0.70	0.92	0.92
	Cora	0.78	0.72	0.79	0.67	0.43	0.67	0.48	0.64	0.64
	CiteSeer	0.73	0.72	0.74	0.68	0.45	0.68	0.49	0.67	0.66
	Chameleon	0.39	0.39	0.41	0.40	0.41	0.42	0.46	0.44	0.43
	Texas	0.53	0.54	0.56	0.72	0.49	0.66	0.56	0.73	0.69
Random feature assignment	Informative*	0.61	0.68	0.54	0.65	0.48	0.49	0.49	0.50	0.50
	Cora	0.57	0.45	0.64	0.39	0.23	0.19	0.25	0.20	0.19
	CiteSeer	0.56	0.40	0.61	0.37	0.20	0.19	0.20	0.20	0.20
	Chameleon	0.23	0.22	0.21	0.21	0.23	0.22	0.22	0.22	0.22
	Texas	0.49	0.51	0.54	0.57	0.51	0.43	0.52	0.54	0.54

* Artificial graph with informative features (SNR=1.5) and structure (homophily=0.95; edge density=0.06)

outcome supports that GNNs should not be treated as a one-size-fits-all approach for machine learning on graphs, as they demand careful inspection of all levels of graph information prior to their use.

3.2 Can advanced GNN architectures cope with poor feature or structure information?

Table 4 aims to extend the insights of the previous section by bringing forward the results of applying the same methodology to more advanced GNN architectures and real-world benchmarks.

The trend in homophilic graphs (Informative, Cora, CiteSeer) supports that all models perform adequately on the original graph versions. In these versions, RevGCN and FiLM are however associated with lower performance; contrastively, these are the models that can better cope with structure information loss, being the only ones that can reach the performance of the feature-only model in the random connectivity scenario. These models are also the ones which recovered the least information from structure only for Cora and CiteSeer, as the results for the random feature assignment mutations corroborate. This outcome suggests that RevGCN and FiLM do not leverage these graphs' structure information as effectively as the vanilla GCN and the APPNP, ultimately refraining from fully exploiting it and resorting to a more feature-preserving encoding method.

Looking into Chameleon and Texas, one can find evidence of how different architectures can respond differently to input graph's structure. None of the studied models appears suitable to exploit Chameleon's structure to their benefit. However, FiLM exhibits feature-only level performance for the original Texas graph, while all other models appear to be hindered by such structure to the point of losing base feature expressiveness. In fact, for some models, random connectivity seems to be easier to handle than the original graph structures of Chameleon and Texas. This can indicate that there are several level of structure impact on GNN performance: there are structures that help, structures that hinder, and structures that are ultimately irrelevant for the classification outcome.

The coherence of results with respect to the ability of models to preserve feature information and/or exploit structure information in different application scenarios is also an indication of the effectiveness of our method. By randomizing feature and structural information, we are able to provide consistent insights on GNNs learning behavior by considering how much each model can learn from features vs. structure for each learning task. This can be an important outcome for GNN explainability. Moreover, it can provide a measure of how meaningful are the features and graph structure for a certain GNN architecture and explicitly assist the identification of learning bottlenecks.

4 Discussion and Related work

Many authors have investigated poor GNN performance due to apparent oversmoothing or loss of expressivity in deeper GNNs [12, 14, 15, 16, 17]. Others have argued that it is not oversmoothing, but rather the training difficulty of GNNs that leads to poor results [18, 19, 20]. Our experiments indicate that performance loss can also result from the smoothing operation itself being ill-suited to certain types of graphs. New GNN architectures that allow deeper models without oversmoothing have been proposed in [7, 8, 21]. Our evaluations suggest that some of these advanced methods are able to rely more on basic node features rather than on network structure. While they preserve feature information, they do not remedy the fact that GCN-like operations may not extract more useful features. This suggests a need for more expressive graph operations, as also noted in several recent papers [22, 23].

Decoupling feature and structure impact in GNNs has recently been investigated for transfer learning and graph generation purposes [24]. However, the authors assume node homophily and do not explore the cases where this condition is not met, which is where our work differs. Other works also explore homo-/heterophilic settings and create advanced architectures to handle challenging scenarios [25]. Nevertheless, it is not clear whether these architectures lead to more useful node representations or if they solely overcome its performance hindering impact, as we can see in recent benchmarks [1]. Our work intends to complement these efforts by providing insights on how structure can not only be harmful but also simply uninformative, leading to local smoothing operations that decrease feature expressivity, in which case one might simply resort to feature-only methods.

Limitations Our results suggest that we should not only explore feature/structure co-dependence but also how models respond to certain combinations of graph properties. This scenario was not explored, despite its potential influence in some of our analyses, thus posing a limitation of this work. Our method also relied on artificial graphs for which we occasionally make an assumption of how meaningful is their information based on our own, theoretically-based criteria. This procedure can limit the conclusions drawn upon those graphs. Furthermore, our method for losing structure information through random connectivity does not destroy all structural properties (e.g. graph density remains the same); however, this transformation does destroy the original encoding of real, intelligible links for the real-world benchmarks. As such, results for artificial graphs may not present direct correspondence with benchmark observations. These limitations are tied to the fact that we only experimented with extreme scenarios. Conducting more experiments could further validate our assumptions by considering more demanding and diverse conditions, for both artificial graphs and benchmarks. Finally, we did not perform hyperparameter tuning for any model to facilitate the comparison of approaches; this, however, makes it harder to compare our results with those of other works. We expect that by making the used hyperparameters available in our configuration files we can diminish the impact of this limitation.

Future work As future work, we must deepen our insights on how models respond to graph properties by extending our method to more complex scenarios, including combinations of structural properties (e.g. simultaneous variation of sets of properties) and feature information. We shall also explore the potential of the feature/structure decoupling method as an empirical indication of how informative graph structures are to a certain network architecture, as such methods are still in demand.

5 Conclusion

This work expanded on the cases when GNNs may not be better than feature-only methods for node classification on graphs. We propose a method that provides new insights on GNN learning behavior by decoupling how much we can learn from features vs. structure for each task. This can be an important outcome towards GNN explainability and effectively assist the identification of learning bottlenecks. Our results suggest that GNNs may lead to poor node representations when the input graph does not fit the inherent assumptions of their architectures, even without oversmoothing. While some advanced architectures can avoid this limitation, we verify that when they cannot leverage structural information, these mostly refrain from exploiting it and ultimately resort to a feature-preserving encoding, similar to feature-only methods. This conclusion supports that GNNs should not be considered a one-size-fits-all approach for machine learning on graphs, but rather demand careful inspection of all levels of graph information prior to their application.

References

[1] Francesco Di Giovanni, James Rowbottom, Benjamin P Chamberlain, Thomas Markovich, and Michael M Bronstein. Graph neural networks as gradient flows. *arXiv preprint arXiv:2206.10991*, 2022.

[2] John Palowitch, Anton Tsitsulin, Brandon Mayer, and Bryan Perozzi. Graphworld: Fake graphs bring real insights for gnns. *arXiv preprint arXiv:2203.00112*, 2022.

[3] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[4] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

[5] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

[6] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[7] Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pages 6437–6449. PMLR, 2021.

[8] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[9] Marc Brockschmidt. Gnn-film: Graph neural networks with feature-wise linear modulation. In *International Conference on Machine Learning*, pages 1144–1152. PMLR, 2020.

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

[11] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021, 2020.

[12] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[13] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134*, 2021.

[14] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

[15] Muhammet Balcilar, Renton Guillaume, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[16] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

[17] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. *arXiv preprint arXiv:2102.06462*, 2021.

[18] Sitao Luan, Mingde Zhao, Xiao-Wen Chang, and Doina Precup. Training matters: Unlocking potentials of deeper graph convolutional neural networks. *arXiv preprint arXiv:2008.08838*, 2020.

[19] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[20] Weilin Cong, Morteza Ramezani, and Mehrdad Mahdavi. On provable benefits of depth in training graph convolutional networks. *Advances in Neural Information Processing Systems*, 34:9936–9949, 2021.

[21] Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor Prasanna, Long Jin, and Ren Chen. Decoupling the depth and scope of graph neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 19665–19679. Curran Associates, Inc., 2021.

[22] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3950–3957, 2021.

[23] Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Liò, and Michael M Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *arXiv preprint arXiv:2202.04579*, 2022.

[24] Duong Chi Thang, Hoang Thanh Dat, Nguyen Thanh Tam, Jun Jo, Nguyen Quoc Viet Hung, and Karl Aberer. Nature vs. nurture: Feature vs. structure for graph neural networks. *Pattern Recognition Letters*, 159:46–53, 2022.

[25] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 4
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See URL available in page 2.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Training details are explained in Section 2, and further disclosed the Appendix and configuration files available at the provided URL.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We report standard deviation relative to the random seed after running experiments multiple times in Tables 6 to 9 of the Appendix, due to paper length constraints.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] We did not inspect this in the scope of our work, since all experiments can be replicated with modest compute power (shallow models, small-sized graphs).
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See URL available in page 2.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Benchmarks' properties

Table 5: Graph properties of benchmark datasets.

Dataset	# Nodes	# Edges	# Classes	Homophily	Edge density
Cora [3]	2485	10138	7	0.81	0.002
CiteSeer [3]	2120	7358	6	0.74	0.002
Chameleon [4]	2277	65019	5	0.23	0.013
Texas [5]	183	558	5	0.11	0.017

A.2 Training details

We used the PyG library [10] and all experiments were run using GraphGym framework [11], which we extended to include more advanced architectures (new layer types or models) when needed. We performed mini-batch training using neighbourhood sampling (batch size of 32) and considered a train-validation-test split of 80%-10%-10%. Further information can be found in the disclosed configuration files, which easily map to GraphGym documentation.

A.3 Detailed tables of results

Table 6: Node classification performance (mean accuracy ± standard deviation) of GCN and MLP models on an artificial graph G with highly informative structure and features, and the mutations of G with random connectivity (uninformative structure), random feature assignment (uninformative features), and both (uninformative structure and features).

		Structure			
		Informative		Uninformative	
Features	Informative	GCN	0.96 ± 0.02	GCN	0.69 ± 0.02
		MLP	0.92 ± 0.02	MLP	0.90 ± 0.04
	Uninformative	GCN	0.61 ± 0.04	GCN	0.48 ± 0.03
		MLP	0.50 ± 0.05	MLP	0.50 ± 0.03

Table 7: Node classification performance (mean accuracy ± standard deviation) of GCN and MLP (baseline) models on artificial graphs with fixed features (SNR = 1.2) and different structural attributes: homophily (H), edge density (D_e). Results are shown for each original graph G and the respective mutations of G with random connectivity, random feature assignment, and both. Given its feature-only nature, a single MLP result is shown per feature transformation.

	H	Structure		D_e	Structure		MLP (baseline)
		Original	Random		Original	Random	
Original Features	0.2	0.78 ± 0.08	0.58 ± 0.03	0.003	0.78 ± 0.03	0.63 ± 0.05	0.71 ± 0.03
	0.5	0.60 ± 0.02	0.61 ± 0.03	0.03	0.81 ± 0.03	0.59 ± 0.03	
	0.8	0.81 ± 0.03	0.59 ± 0.03	0.15	0.79 ± 0.02	0.57 ± 0.02	
Random Feature Assignment	0.2	0.51 ± 0.05	0.48 ± 0.04	0.003	0.49 ± 0.06	0.48 ± 0.05	0.49 ± 0.03
	0.5	0.53 ± 0.04	0.50 ± 0.03	0.03	0.49 ± 0.03	0.49 ± 0.02	
	0.8	0.49 ± 0.03	0.49 ± 0.02	0.15	0.52 ± 0.04	0.49 ± 0.04	

Table 8: Node classification performance (mean accuracy ± standard deviation) of GCN and MLP (baseline) models on artificial graphs with fixed structure (homophily = 0.8; edge density = 0.03) and different feature signal-to-noise ratio (SNR). Results are shown for each original graph G and the respective mutations of G with random connectivity.

SNR	Structure		MLP (baseline)
	Original	Random	
1.0	0.57 ± 0.05	0.51 ± 0.03	0.51 ± 0.05
1.2	0.81 ± 0.03	0.59 ± 0.03	0.71 ± 0.03
1.5	0.92 ± 0.02	0.68 ± 0.02	0.91 ± 0.02
2.0	0.99 ± 0.01	0.72 ± 0.01	1.00 ± 0.00

Table 9: Node classification performance (mean accuracy ± standard deviation) of different GNN architectures and MLP (baseline) models on real-world benchmarks. Results are shown for each original graph G and the respective mutations of G with random connectivity, random feature assignment, and both. Given its feature-only nature, a single MLP result is shown per feature transformation.

		Original Structure				Random Connectivity				
		GCN	RevGCN	APPNP	FiLM	GCN	RevGCN	APPNP	FiLM	MLP
Original Features	Informative*	0.96 ± 0.02	0.96 ± 0.01	0.95 ± 0.02	0.96 ± 0.01	0.69 ± 0.02	0.92 ± 0.03	0.70 ± 0.02	0.92 ± 0.03	0.92 ± 0.04
	Cora	0.78 ± 0.02	0.72 ± 0.02	0.79 ± 0.03	0.67 ± 0.02	0.43 ± 0.03	0.67 ± 0.04	0.48 ± 0.04	0.64 ± 0.03	0.64 ± 0.04
	CiteSeer	0.73 ± 0.02	0.72 ± 0.02	0.74 ± 0.03	0.68 ± 0.03	0.45 ± 0.02	0.68 ± 0.03	0.49 ± 0.03	0.67 ± 0.02	0.66 ± 0.03
	Chameleon	0.39 ± 0.05	0.39 ± 0.03	0.41 ± 0.02	0.40 ± 0.04	0.41 ± 0.05	0.42 ± 0.04	0.46 ± 0.04	0.44 ± 0.05	0.43 ± 0.04
	Texas	0.53 ± 0.09	0.54 ± 0.12	0.56 ± 0.12	0.72 ± 0.14	0.49 ± 0.12	0.66 ± 0.06	0.56 ± 0.13	0.73 ± 0.10	0.69 ± 0.12
Random feature assignment	Informative*	0.61 ± 0.04	0.68 ± 0.04	0.54 ± 0.05	0.65 ± 0.07	0.48 ± 0.03	0.49 ± 0.03	0.49 ± 0.04	0.50 ± 0.06	0.50 ± 0.05
	Cora	0.57 ± 0.02	0.45 ± 0.04	0.64 ± 0.03	0.39 ± 0.04	0.23 ± 0.02	0.19 ± 0.02	0.25 ± 0.02	0.20 ± 0.04	0.19 ± 0.04
	CiteSeer	0.56 ± 0.03	0.40 ± 0.04	0.61 ± 0.02	0.37 ± 0.06	0.20 ± 0.02	0.19 ± 0.02	0.20 ± 0.02	0.20 ± 0.03	0.20 ± 0.03
	Chameleon	0.23 ± 0.02	0.22 ± 0.03	0.21 ± 0.04	0.21 ± 0.03	0.23 ± 0.05	0.22 ± 0.04	0.22 ± 0.03	0.22 ± 0.03	0.22 ± 0.04
	Texas	0.49 ± 0.13	0.51 ± 0.10	0.54 ± 0.12	0.57 ± 0.10	0.51 ± 0.13	0.43 ± 0.12	0.52 ± 0.12	0.54 ± 0.11	0.54 ± 0.11

* Artificial graph with informative features (SNR=1.5) and structure (homophily=0.95; edge density=0.06)