Graph Neural Networks Use Graphs When They Shouldn't

Maya Bechler-Speicher [1] **Ido Amos** [2] **Ran Gilad-Bachrach** [3] **Amir Globerson** [1 4]

Abstract

Predictions over graphs play a crucial role in various domains, including social networks and medicine. Graph Neural Networks (GNNs) have emerged as the dominant approach for learning on graph data. Although a graph-structure is provided as input to the GNN, in some cases the best solution can be obtained by ignoring it. While GNNs have the ability to ignore the graph-structure in such cases, it is not clear that they will. In this work, we show that GNNs actually tend to overfit the given graph-structure. Namely, they use it even when a better solution can be obtained by ignoring it. We analyze the implicit bias of gradient-descent learning of GNNs and prove that when the ground truth function does not use the graphs, GNNs are not guaranteed to learn a solution that ignores the graph, even with infinite data. We examine this phenomenon with respect to different graph distributions and find that regular graphs are more robust to this overfitting. We also prove that within the family of regular graphs, GNNs are guaranteed to extrapolate when learning with gradient descent. Finally, based on our empirical and theoretical findings, we demonstrate on real-data how regular graphs can be leveraged to reduce graph overfitting and enhance performance.

1. Introduction

Graph labeling problems arise in many domains, from social networks to molecular biology. In these settings, the goal is to label a graph or its nodes given information about the graph. The information for each graph instance is typically provided in the form of the graph-structure (i.e., its adjacency matrix) as well as the features of its nodes.

Graph Neural Networks (GNNs) (Kipf & Welling, 2017b; Gilmer et al., 2017; Veličković et al., 2018; Hamilton et al., 2017) have emerged as the leading approach for such tasks. The fundamental idea behind GNNs is to use neural-networks that combine the node features with the graph-structure, in order to obtain useful graph representations. This combination is done in an iterative manner, which can capture complex properties of the graph and its node features.

Although graph-structures are provided as input to the GNN, in some cases the best solution can be obtained by ignoring them. This may be due to these graph-structures being non-informative for the predictive task at hand. For instance, some molecular properties such as the molar mass (i.e., weight) depend solely on the constituent atoms (node features), and not on the molecular structure. Another case is when the provided graph-structure does contain valuable information for the task, but the GNN cannot effectively exploit it. In such cases, better test accuracy may be achieved by ignoring the graph-structure. In other cases, the node features alone carry most of the information and the graph-structure conveys just a small added value. For example, assume that node features contain the user's zipcode. Then the user's income is highly predictable by that feature, and their social structure will add little accuracy of this prediction.

Motivated by this observation, we ask a core question in GNN learning: will GNNs work well in cases where it is better to ignore the graph-structure or will they overfit the graph-structure, resulting in reduced test accuracy? Answering this question has several far-reaching practical implications. To illustrate, if GNNs lack the ability to discern when to disregard the graph, then providing a graph can actually hurt the performance of GNNs, and thus one must carefully re-think which graphs to provide a GNN. On the other hand, if GNNs easily reject the structure when they fail to exploit it, then practitioners should attempt to provide a graph, even if their domain knowledge and expertise suggest that there is only a small chance it is informative.

We consider the common setting of over-parameterized GNNs. Namely, when the number of parameters the GNN

[1]Blavatnik School of Computer Science, Tel-Aviv University [2]School of Electrical Engineering, Tel-Aviv University [3]Department of Bio-Medical Engineering and Edmond J. Safra Center for Bioinformatics, Tel-Aviv University [4]Now also at Google Research. Correspondence to: Maya Bechler-Speicher <mayab4@mail.tau.ac.il>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

uses is larger than the size of the training data. This is a very common case in deep-learning, where the learned model can fit any training data. Previous studies showed that models learned using Gradient Descent (GD) often generalize well despite over-parameterization. Hence, it was suggested that the learning algorithm exhibits an implicit bias (e.g., low parameter norm) to avoid spurious models that happen to fit the training data (e.g., Zhang et al., 2017; Lyu & Li, 2020; Gunasekar et al., 2018; Soudry et al., 2017).

Our focus is thus on the implicit bias of GNN learning, and specifically, whether GNNs are biased towards using or not using the graph-structure. If the implicit bias is towards "simple models" that do not use the graph-structure when possible, then one would expect GNNs to be oblivious to the graph-structure when it is not informative. Our first empirical finding is that this is actually not the case. Namely, GNNs tend to *not* ignore the graph, and their performance is highly dependent on the provided graph-structure. Specifically, there are graph-structures that result in models with low test accuracy.

Next, we ask which properties of the learned graph distribution affect the GNN's ability to ignore the graph. We empirically show that regular graphs result in more resilient GNNs. We then analyze the implicit bias of learning GNNs with gradient descent and prove that despite the ground truth function being "simple" in the sense it does not use the graph, GNNs are not guaranteed to learn a solution that ignores the graph, even with infinite data. We prove that as a result of their implicit bias, GNNs may fail to extrapolate. We then prove that within the family of regular graphs, GNNs are guaranteed to extrapolate when learning with gradient descent and provide a sufficient condition for extrapolation when learning on regular graphs.

Finally, we empirically examine on real-world datasets if the properties of regular graphs are also beneficial in cases where the graph should not necessarily be ignored. We show that modifying the input graph to be "more regular" can indeed improve performance in practice.

We note that we focus on the implicit bias of GNNs, i.e., what GNNs *actually* do when the graph *should* be ignored. Understanding this bias can also shed light on the phenomenon of entanglement as, i.e., the intricate interplay between the graph structure and the node features (Liu et al., 2020; Seddik et al., 2022). It was previously shown that as a result of entanglement, GNNs sometimes do not work as well as set methods (Errica et al., 2022; Chen et al., 2020; Zhang et al., 2022). This work is motivated by those findings.

The main contributions of this work are (1) We show that GNNs tend to overfit the graph-structure, when it should be ignored. (2) We evaluate the graph-structure overfitting phenomenon with respect to different graph distributions and find that the best performance is obtained for regular graphs. (3) We theoretically analyze the implicit bias of learning GNNs, and show that when trained on regular graphs, they converge to unique solutions that are more robust to graph-structure overfitting. (4) We show empirically that transforming GNN input graphs into more regular ones can mitigate the GNN tendency to overfit and improve performance.

2. GNNs Overfit the Graph-Structure

In this section, we present an empirical evaluation showing that GNNs tend to overfit the graph-structure, thus hurting their generalization accuracy. Graph overfitting refers to any case where the GNN uses the graph when it is preferable to ignore it (e.g. because it is non-informative for the task).

2.1. Preliminaries

A graph example is a tuple $G = (A, X)$. A is an adjacency matrix representing the graph-structure. Each node i is assigned a feature vector $\mathbf{x}_i \in \mathbb{R}^d$, and all the feature vectors are stacked to a feature matrix $X \in \mathbb{R}^{n\times d}$, where n is the number of nodes in G. The set of neighbors of node i is denoted by $N(i)$. We denote the number of samples in a dataset by m. We focus on the common class of Message-Passing Neural Networks (Morris et al., 2021). In these networks, at each layer, each node updates its representation as follows:

$$h_i^{(k)} = \sigma\left(W_1^{(k)} h_i^{(k-1)} + W_2^{(k)} \sum_{j\in N(i)} h_j^{(k-1)} + b^{(k)}\right) \quad (1)$$

where $W_1^{(k)}, W_2^{(k)} \in \mathbb{R}^{d_k\times d_{k-1}}$. The initial representation of node i is its feature vector $h_i^{(0)} = \mathbf{x}_i$. The final node representations $\{h_i^{(L)}\}_{i=1}^n$ obtained in the last layer can then be used for downstream tasks such as node or graph labeling. We focus on graph labeling tasks, where a graph representation vector is obtained by combining all the node representations, e.g., by summation. This is then followed by a linear transformation matrix W_3 that provides the final classification/regression output (referred to as a *readout* layer). For the sake of presentation, we drop the superscript in cases of one-layer GNNs. For binary classification, we assume the label is the sign of the output of the network.

We refer to $W_1^{(k)}$ as the *root-weights* of layer k and to $W_2^{(k)}$ as the *topological-weights* of layer k. A natural way for GNNs to ignore the graph-structure is by zeroing the topological-weights $W_2^{(k)} = \bar{0}$ in every layer. We say that a function $f(X, A)$ is *graph-less* if $f(X, A) = f(X)$, i.e., the function does not use the graph-structure, and is practically a set function.

Table 1: The accuracy of a fixed GNN architecture, trained once on the given graphs in the data (GNN) and once on the same data where the graph-structure is omitted ($GNN_\emptyset$), i.e., on empty graphs. The solution of $GNN_\emptyset$ is realizable by GNN, and the only difference between the runs is the given graph-structures. This suggests that the decreased performance of GNN is due to graph-structure overfitting.

	Sum	Proteins	Enzymes
GNN	94.5 ± 0.9	67.4 ± 1.9	55.2 ± 3.1
$GNN_\emptyset$	97.5 ± 0.7	74.1 ± 2.5	64.1 ± 5.7

It is important to note that some GNNs, e.g., Kipf & Welling (2017a), do not possess the ability to ignore the graph-structure as the root and topological weights are the same. We therefore focus on the most general GNN type that does have the ability to ignore the graph (Gilmer et al., 2017).

In the Appendix, we extend our empirical evaluation to multiple GNN variations, including Graph Attention Network (Veličković et al., 2018; Brody et al., 2022), Graph Transformer (Shi et al., 2021) and Graph Isomorphism Network (Xu et al., 2019), and to node classification, which show similar trends.

2.2. Evidence for Graph Overfitting

Our goal is to examine what happens when GNNs learn over graphs that should be ignored, either because they are non-informative for the task, or because the GNN fails to exploit their information. To that end, we conducted experiments on three datasets.

Sum This is a binary classification synthetic task with a graph-less ground truth function. To generate the label, we use a teacher GNN that simply sums the node features and applies a linear readout to produce a scalar. The data contains non-informative graph-structures which are drawn from the GNP graph distribution (Erdös & Rényi, 1959), where the edges are sampled i.i.d with probability p (we used $p = 0.5$).

Proteins and Enzymes These are two classification tasks on real-world molecular data (Morris et al., 2020). In Errica et al. (2022) the authors reported on a thorough GNNs comparison, that the best accuracy on these datasets is achieved when the graph-structure is omitted. We note that with a fixed architecture, the solution learned by a GNN trained on empty graphs is always realizable by the same GNN trained on non-empty graphs. This is a straightforward argument, and it is explained in the Appendix for the sake of completeness. Therefore, with a fixed architecture, better performances that are achieved when learning over empty graphs indicates that it was better for the GNN to ignore the graph, and it could, but it didn't. Errica et al. (2022) used a different model for the empty graphs, which was not an instance of the other compared GNNs. Therefore, their results do not imply that the compared GNNs overfitted the graph-structure, as the superiority of the model trained on empty graphs may be due to its architecture. In our experiments, we use a fixed architecture to ensure that a discrepancy in performance implies graph overfitting.

Protocol and Results On each of the three datasets, we trained the same GNN twice: once on the given graph-structures in the data (GNN), and once when the graph-structure is replaced with an empty graph, and only the node features are given for training ($GNN_\emptyset$). This difference between these setups shows the effect of providing the graph-structure.

The GNNs architecture is fixed, and the learning hyper-parameters are tuned on a validation set for the Sum task and 10-fold cross-validation for Protein and Enzymes. We report test errors averaged over 10 runs with random seeds on a separate holdout test set. More information can be found in the Appendix.

Table 1 shows the results of the experiments. In the three tasks, $GNN_\emptyset$ achieves higher accuracy than GNN. This suggests that GNN made use of the graphs, although a better result, i.e., the one learned by $GNN_\emptyset$, could be obtained by ignoring them. This *graph overfitting* led to lower test accuracy.

2.3. How Graph-Structure Affects Overfitting

The previous section showed that in the Sum task, where the given graph-structures are non-informative and should be ignored, the GNN overfits them instead. Here we further study how this phenomenon is affected by the specific graph-structure provided to the GNN. Thus, we repeat the setup of the Sum task but with different graph distributions.

Data We used the Sum task described in Section 2.2. We created four different datasets from this baseline by sampling graph-structures from different graph distributions. The set of node feature vectors remains the same across all the datasets, and thus, the datasets differ only in their graph-structures. The graph distributions we used are: r-regular graphs (Regular) where all the nodes have the same degree r, star-graph (Star) where the only connections are between one specific node and all other nodes, the Erdös-Rényi graph distribution (GNP) (Erdös & Rényi, 1959), where the edges are sampled i.i.d with probability p, and the preferential attachment model (BA) (Barabasi & Albert, 1999), where the graph is built by incrementally adding new nodes and con-

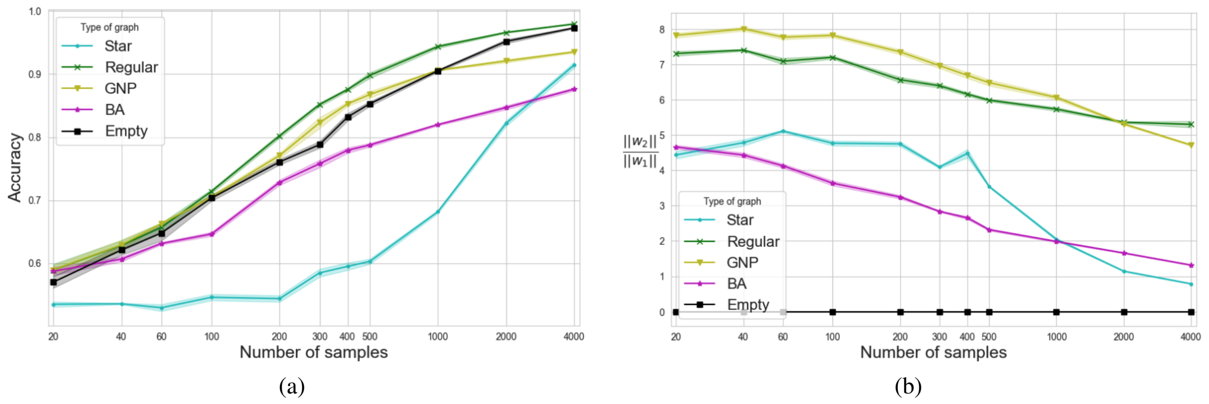

Figure 1: (a) The learning curves of the same GNN model trained on graphs that have the same node features and only differ in their graph-structure, which is sampled from different distributions. The label is computed from the node features without the use of any graph-structure. If GNNs were to ignore the non-informative graph-structure they were given, similar performance should have been observed for all graph distributions. Among the different distributions, regular graphs exhibit the best performance. (b) The norm ratio between the topological and the root weights along the same runs. Except for the empty graphs, the ratio is always greater than 1, which indicates that more norm is given to the topological weights.

necting them to existing nodes with probability proportional to the degrees of the existing nodes.

Protocol The GNN model is the same as in the Sum task in the previous section. On each dataset, we varied the training set size and evaluated test errors on 10 runs with random seeds. More information can be found in the Appendix.

Results For the sake of presentation, we present the results on one instance from each distribution: Regular with $r = 10$, GNP with $p = 0.6$, and BA with $m = 3$. Additional results with more distribution parameters are given in the Appendix, and similar trends are shown. Recall that the datasets differ only by the edges and share the same set of nodes and features. Therefore, had the GNN ignored the graph-structures, we would expect to see similar performance for all datasets. As shown in Figure 1(a), the performance largely differs between different graph distributions, which indicates the GNN overfits the graphs rather than ignores them.

To further understand what the GNN learns in these cases, we evaluate the ratio between the norms of the topological and root weights. Results are shown in Figure 1(b). It can be seen that for all the graphs except the empty graphs, the ratio is larger than 1, indicating that there is more norm on the topological weights than on the root weights. Specifically, the graph-structure is not ignored. In the case of empty graphs, the topological weights are not trained, and the ratio is 0 due to initialization. We also present the norms of the root and topological weights separately in the Appendix.

Figure 1 suggests that some graph distributions are more robust to graph-structure overfitting. The GNN trained on regular graphs performs best across all training set sizes.

The good performance on regular graphs would seem to suggest that it learns to use low topological weights. However, as Figure 1(b) shows, the opposite is actually true. This may seem counter-intuitive, but in the next section, we theoretically show how this comes about.

3. Theoretical Analysis

In the previous section, we saw that GNNs tend to overfit the graph-structure when it should be ignored. We now turn to a theoretical analysis that sheds light on what GNNs learn when the ground truth teacher is graph-less.

For the sake of clarity, we state all theorems for a one-layer GNN with sum-pooling, no readout, and output dimension 1. For simplicity, we also assume no bias term in our analysis. All the proofs and extensions can be found in the Appendix.

3.1. Implicit bias of Gradient Descent for GNNs

Let S denote a training set of labeled graphs. Each instance in S is a triplet (X, A, y), where X is a stacked feature matrix of the node feature vectors, A is the adjacency matrix, and $y \in \pm 1$ is the class label (we consider binary classification). To examine the solutions learned by GNNs, we utilize Theorem 4 from Gunasekar et al. (2018). This theorem states that homogeneous neural networks trained with GD on linearly separable data converge to a KKT point of a max-margin problem. Translating this theorem to the GNN in our formulation, we get that gradient-based training will converge to the solution of the following problem:

$$\begin{array}{ll} \min_{\mathbf{w}_1,\mathbf{w}_2} \|\mathbf{w}_1\|_2^2 + \|\mathbf{w}_2\|_2^2 & \\ s.t. \quad y[\mathbf{w}_1 \cdot \sum_i^n \mathbf{x}_i + \mathbf{w}_2 \sum_i^n deg(i)\mathbf{x}_i] \geq 1 & \\ \forall (X, A, y) \in S & \end{array} \tag{2}$$

Equation 2 can be viewed as a max-margin problem in $2d$ space, where the input vector is $[\sum_i^n \mathbf{x}_i, \sum_i^n deg(i)\mathbf{x}_i]$. Therefore, the graph input can be viewed as the sum of the node feature vectors concatenated with their weighted sum, according to the node degrees.

When trained on r-regular graphs, Equation 2 can be written as:

$$\begin{array}{ll} \min_{\mathbf{w}_1,\mathbf{w}_2} & \|\mathbf{w}_1\|_2^2 + \|\mathbf{w}_2\|_2^2 \\ s.t. & y[(\mathbf{w}_1 + r\mathbf{w}_2) \cdot \sum_i^n \mathbf{x}_i] \geq 1 \\ \forall (X, A, y) \in S & \end{array} \tag{3}$$

This can be viewed as a max-margin problem in $\mathbb{R}^d$ where the input vector is $\sum_i^n \mathbf{x}_i$. So the GNN is a linear classifier on the sum of the node features, but the regularizer is not the L_2 norm of the weights because of the r factor.

The next theorem shows that when a GNN is trained using GD on regular graphs, the learned root and topological weights are aligned.

Lemma 3.1 (Weight alignment)**.** *Let S be a set of linearly separable r-regular graph examples. A GNN trained with GD that fits S perfectly converges to a solution such that $\mathbf{w}_2 = r\mathbf{w}_1$. Specifically, the root weights $\mathbf{w}_1$ and topological weights $\mathbf{w}_2$ are aligned.*

We prove Lemma 3.1 in the Appendix by analyzing the KKT conditions for first-order stationary points of Equation 3.

The next section will use this result to explain why regular graphs are better for learning graph-less teachers.

3.2. Extrapolation with graph-less teachers

In this section, we analyze what happens when GNNs learn from training data generated by a graph-less model (we refer to this as a graph-less teacher). As we saw empirically in Section 2.2, these learned models will sometimes generalize badly on test data. We begin with a theorem that proves that such bad cases indeed exist. The theorem considers the extrapolation case, where the train and test distribution of graphs is not the same but is labeled by the same graph-less teacher. Had the GNN learned a graph-less model, it would have had the same train and test performance (for infinite data). However, we show this is not the case, indicating that GNNs can overfit the graph structure arbitrarily badly. In other words, they do not extrapolate.

Theorem 3.2 (Extrapolation may fail)**.** *Let f^* be a graph-less teacher. There exist graph distributions P_1 and P_2, with node features drawn from the same fixed distribution, such that when learning a linear GNN with GD over infinite data drawn from P_1 and labeled with f^*, the test error on P_2 labeled with f^* will be $\geq \frac{1}{4}$. Namely, the model will fail to extrapolate.*

The setting in the above result is that a graph-less ground truth teacher is learned using graphs from P_1. Ideally, we would have liked GD to "ignore" the graphs, so that the output of the learned model would not change when changing the support to P_2. However, our result shows that when the graph distribution is changed to P_2, performance is poor. This is in line with our empirical observations. The key idea in proving the result is to set P_2 such that it puts weights on isolated nodes and thus exposes the fact that the learned function does not simply sum all nodes, as the graph-less solution does.

Despite Theorem 3.2 showing GNNs may fail to extrapolate, the following result shows that GNNs are guaranteed to extrapolate within the family of regular distributions.

Theorem 3.3 (Extrapolation within regular distributions)**.** *Let D_G be a distribution over r-regular graphs and D_X be a distribution over node features. Assume a training set of infinite size sampled from D_G and D_X and labeled with a graph-less teacher. Denote the model learned with GD by f. Assume that test examples are sampled from D'_G, a distribution over r'-regular graphs, and D_X. Then f will have zero test error.*

To prove Theorem 3.3, we utilize Equation 3 and Lemma 3.1, and show that the direction of the weight vector used by the GNN does not change when the regularity degree is changed.

It was previously shown in Yehudai et al. (2020) that when there is a certain discrepancy between the train and test distributions, GNNs may fail to extrapolate. The argument extends to our case, and therefore learning without GD could fail to generalize.

3.2.1. Characterizing Extrapolation Accuracy

Theorems 3.2 and 3.3 show extreme cases of good and bad extrapolation. Next, we examine what determines the extrapolation accuracy.

First, we empirically observe that GNNs trained on regular graphs exhibit good extrapolation to other non-regular graph distributions as well, as presented in Table 2.

For example, a GNN trained on 5-regular graphs, generalizes perfectly to GNP graphs, and there is a decrease in performance when tested on star-graphs. The training setup and more information on the graphs can be found in the Appendix.

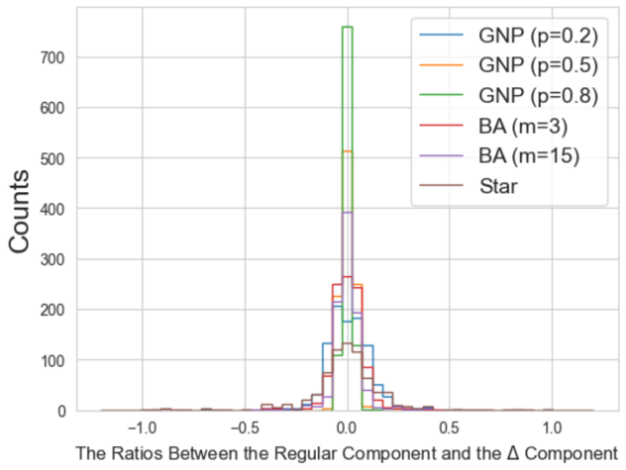

Figure 2: The ratios histogram for test examples that are correctly classified in the extrapolation evaluation presented in Table 2. The condition in Theorem 3.4 is met for all the correctly classified examples.

Next, we present a sufficient condition for extrapolation and empirically show on these test sets that indeed when the GNN successfully extrapolates, this sufficient condition holds.

We utilize Lemma 3.1 and write the GNN trained on r-regular graphs in a new form acting on a test graph G as:

$$f(G) = W_1 \sum_{i=1}^{n} \mathbf{x}_i + \underbrace{rW_1 \sum_{i=1}^{n} r'\mathbf{x}_i}_{Regular\ Component} + \underbrace{rW_1 \sum_{i=1}^{n} \Delta_{r',G}(i)\mathbf{x}_i}_{\Delta\ Component}$$

where $\Delta_{r',G}(i) = deg_G(i) - r'$, for any $0 \le r'$. This notation shows that applying f to a graph G is equivalent to applying it to an r'-regular graph plus applying it to another Δ-graph that depends on r'.

Using this notation, the following Theorem provides a sufficient condition for extrapolation. For simplicity, we state the results as extrapolation to the same training set, with modified graphs.

Theorem 3.4 (Sufficient condition for extrapolation). *Let S be a set of r-regular graphs examples, labeled with a graph-less teacher f^*. Let f denote a GNN trained with GD on S. Now assume an instance $G = (X, A) \in S$ has been modified to a different graph $\tilde{G} = (X, \tilde{A})$ such that there exists an $0 \le r' \le n-1$ where:*

$$\left| \frac{r\mathbf{w}_1 \sum_{i=1}^{n} \Delta_{r',\tilde{G}}(i)\mathbf{x}_i}{\mathbf{w}_1\tilde{x} + r'r\mathbf{w}_1\tilde{x}} \right| \le 1$$

Then $f(\tilde{G}) = f^*(\tilde{G})$.

Theorem 3.4 suggests that applying the GNN to graphs that are "closer" to regular graphs, i.e., have smaller Δ, results in better extrapolation. To prove it, we show that when these conditions hold, the extrapolation is guaranteed from Theorem 3.3.

Table 2: Accuracy of a GNN trained on 5-regular graphs and tested on different distribution shifts. The GNN extrapolates perfectly to regular graph distributions, as guaranteed by Theorem 3.3.

Test distribution	Accuracy
Regular (r=10)	100 ± 0.0
Regular (r=15)	100 ± 0.0
GNP (p=0.2)	100 ± 0.0
GNP (p=0.5)	100 ± 0.0
GNP (p=0.8)	100 ± 0.0
BA (m=3)	98.0 ± 1.7
BA (m=15)	93.2 ± 0.9
Star Graph	75.9 ± 1.1

Next, we empirically show that, indeed, all the samples that were classified correctly in Table 2 satisfy this condition of Theorem 3.4.

Figure 2 presents histograms of the values of the ratio in Theorem 3.4 for every example that is correctly classified over the test examples presented in Table 2. We do not include regular graphs in the histograms because extrapolation within regular graphs is guaranteed from Theorem 3.4. The ratio is computed for the r' that minimizes the denominator of the ratio. Indeed, all the ratios are less than 1, and therefore the sufficient condition holds. These results demonstrate that, indeed, "closeness to a regular" graph is an important determinant in extrapolation accuracy.

4. Are Regular Graphs Better when Graphs are Useful?

In the previous sections, we showed that regular graphs exhibit robustness to the tendency of GNNs to overfit non-informative graphs that should be completely ignored. In this section, we examine if regular graphs are also beneficial in scenarios when the graph may be informative. We perform an empirical evaluation on real-world data, where we do not know in advance if the graph is indeed informative or not. We compare the performance of the same method when trained on the original graph, and on the same graph when transformed to be "more regular".[1]

Setup Ideally, we would like to examine the performance change when a graph is transformed into a regular graph.

[1]The code is available on https://github.com/mayabechlerspeicher/Graph_Neural_Networks_Overfit_Graphs

Table 3: Performance of different GNNs when trained on the original graphs versus when the COV of the graphs is reduced. The best model is in bold and with an underline in cases where the p-value < 0.05 using the Wilcoxon signed-rank test.

Model	Graph	Proteins	NCI1	Enzymes	D&D	mol-hiv	mol-pcba
DeepSet	Empty Graph	74.1 ± 2.5	72.8 ± 2.1	64.2 ± 3.0	77.5 ± 2.0	69.5 ± 2.9	15.0 ± 0.6
GraphConv	Original Graph	73.1 ± 1.6	76.5 ± 1.2	58.2 ± 2.1	72.5 ± 1.7	78.2 ± 3.0	20.5 ± 0.5
	Original Graph + R-COV	$\underline{\mathbf{75.5 \pm 1.8}}$	$\underline{\mathbf{80.1 \pm 0.9}}$	$\underline{\mathbf{61.0 \pm 1.5}}$	**74.8 ± 2.9**	$\underline{\mathbf{80.9 \pm 1.8}}$	$\underline{\mathbf{22.8 \pm 0.5}}$
GIN	Original Graph	72.2 ± 2.9	79.2 ± 1.5	58.9 ± 1.8	74.5 ± 2.3	77.0 ± 1.9	21.1 ± 0.5
	Original Graph + R-COV	$\underline{\mathbf{74.8 \pm 2.1}}$	**80.0 ± 1.1**	**59.7 ± 1.4**	**75.7 ± 3.9**	**77.9 ± 1.3**	**21.5 ± 0.2**
GATv2	Original Graph	73.5 ± 2.8	80.4 ± 1.6	59.9 ± 2.8	70.6 ± 4.0	78.7 ± 2.5	23.5 ± 0.9
	Original Graph + R-COV	**76.5 ± 2.0**	$\underline{\mathbf{83.0 \pm 1.5}}$	$\underline{\mathbf{63.9 \pm 3.5}}$	**73.9 ± 1.2**	$\underline{\mathbf{80.9 \pm 2.0}}$	$\underline{\mathbf{24.3 \pm 0.7}}$
GraphTransformer	Original Graph	73.9 ± 1.5	80.5 ± 1.1	60.9 ± 2.1	74.1 ± 1.9	80.5 ± 2.9	29.1 ± 0.7
	Original Graph + R-COV	$\underline{\mathbf{76.7 \pm 1.4}}$	$\underline{\mathbf{83.1 \pm 1.9}}$	$\underline{\mathbf{64.0 \pm 1.9}}$	$\underline{\mathbf{77.1 \pm 1.8}}$	**82.4 ± 1.5**	$\underline{\mathbf{30.5 \pm 0.2}}$

Model	Graph	IMDB-B	IMDB-M	Collab	Reddit-B	Reddit-5k
DeepSet	Empty Graph	70.0 ± 3.0	48.2 ± 2.5	71.2 ± 1.3	80.9 ± 2.0	52.1 ± 1.7
GraphConv	Original Graph	69.6 ± 1.7	47.5 ± 1.0	73.5 ± 1.3	83.2 ± 1.5	50.0 ± 2.1
	Original Graph + R-COV	$\underline{\mathbf{72.9 \pm 0.5}}$	$\underline{\mathbf{50.0 \pm 1.5}}$	**74.2 ± 2.1**	$\underline{\mathbf{87.0 \pm 1.8}}$	$\underline{\mathbf{52.5 \pm 1.7}}$
GIN	Original Graph	70.1 ± 2.9	48.1 ± 2.5	75.3 ± 2.9	89.1 ± 2.7	56.1 ± 1.5
	Original Graph + R-COV	$\underline{\mathbf{71.3 \pm 1.5}}$	**48.5 ± 1.7**	$\underline{\mathbf{77.2 \pm 2.0}}$	**91.0 ± 1.1**	**56.7 ± 0.8**
GATv2	Original Graph	72.8 ± 0.9	48.4 ± 2.1	73.9 ± 1.7	90.0 ± 1.5	56.4 ± 1.5
	Original Graph + R-COV	$\underline{\mathbf{75.8 \pm 1.5}}$	$\underline{\mathbf{50.8 \pm 1.7}}$	**75.1 ± 1.9**	$\underline{\mathbf{92.1 \pm 0.9}}$	**57.0 ± 0.9**
GraphTransformer	Original Graph	73.1 ± 1.3	49.0 ± 1.9	73.8 ± 1.5	90.6 ± 1.3	51.4 ± 1.7
	Original Graph + R-COV	$\underline{\mathbf{76.1 \pm 2.0}}$	**51.1 ± 2.3**	$\underline{\mathbf{76.0 \pm 1.8}}$	$\underline{\mathbf{92.3 \pm 1.0}}$	$\underline{\mathbf{56.0 \pm 1.2}}$

While it is possible to make a graph regular by simply completing it into a full graph, this approach may be computationally expensive or even infeasible when learning with GNNs. Unfortunately, other approaches may require removing edges from the original graph, which may lead to information loss or even make the task non-realizable. Therefore, we evaluate a transformation of a given graph to a "more regular" graph, while keeping all its original information. We modify a given graph by adding edges between low-degree nodes in order to reduce its node degrees coefficient of variation (COV), i.e., the ratio between the standard deviation and mean of the node degrees. The COV of regular graphs is 0, because all the nodes have the same degree. In order to maintain the original information about the given graph we augment each edge with a new feature that has values 1 and 0.5 for the original and added edges, respectively.

We evaluate[2] the four GNN architectures for which Section 2.2 shows a tendency to overfit the graph. These are: GraphConv (Gilmer et al., 2017), GIN (Xu et al., 2019), GATv2 (Veličković et al., 2018; Brody et al., 2022) and GraphTransformer (Shi et al., 2021). We adopted the neighbors' aggregation component of each model to process the edge features in a non-linear way. The complete forms of each architecture we used with the addition of edge feature processing can be found in the Appendix.

[2]The code is provided in the Supplementary Material.

4.1. Datasets

We used 11 graph datasets, including two large-scale datasets, which greatly differ in their average graph size and density, the number of node features, and the number of classes.

Enzymes, D&D, Proteins, NCI1 (Shervashidze et al., 2011) are datasets of chemical compounds where the goal is to classify each compound into one of several classes.
IMDB-B, IMDB-M, Collab, Reddit-B, Reddit-5k (Yanardag & Vishwanathan, 2015) are social network datasets.
mol-hiv, mol-pcba (Hu et al., 2020) are large-scale datasets of molecular property prediction.

More information on the datasets and their statistics can be found in the Appendix.

Evaluation For each model and for each task, we evaluate the model twice: on the original graph provided in the dataset (Original Graph) and on the original graph with the COV reduced (R-COV). Because different graphs have different COVs, we set COV to a fixed percentage of the original average COV of each dataset separately. The percentage is a hyperparameter, and we tested the values $\{80\%, 50\%\}$. We also include as a baseline the performance when the graph-structure is omitted (Empty Graphs), which is equivalent to using DeepSets (Zaheer et al., 2018).

For all the datasets except mol-hiv and mol-pcba we used 10-fold nested cross-validation with the splits and protocol of Errica et al. (2022). The final reported result on these datasets is an average of 30 runs (10-folds and 3 random seeds). The mol-hiv and mol-pcba datasets have pre-defined train-validation-test splits and metrics Hu et al. (2020). The metric of mol-hiv is the test AUC averaged over 10 runs with random seeds. The metric of mol-pcba the metric is the averaged precision (AP) over its 128 tasks.
Additional details and the hyper-parameters are provided in the Appendix.

Results Across all datasets and all models, reducing the COV of the graphs improves generalization. Particularly intriguing outcomes are obtained in the PROTEINS and IMDB-M datasets. Within these two datasets, superior performance is attained when learning over empty graphs in comparison to the provided graphs. Nonetheless, reducing the COV improves performance also with respect to the empty graphs. This observation suggests that the structural information inherent in the data is indeed informative, yet the GNN fails to exploit it correctly as it is.

The tradeoff between COV and graph density As we see consistent improvement when the COV is reduced, we further examined if this improvement is monotone with respect to the COV reduction. We evaluated the Proteins dataset with an increasing percentage of COV reduction, up to the full graph. Indeed as shown in Figure 3, the performance keeps improving as the COV is reduced. This is in alignment with the results of Alon & Yahav (2021) where a full graph was used in the last layer of the network to allow better information flow between nodes of long distance. Note that in our case, we also distinguish the original edges with the added edges using edge features and allow the network to ignore the added edges. Clearly, using a full graph comes with a computational cost, a problem that also arises when using full-graph transformers. Our results suggested that improvement in generalization can be achieved also without the cost of using the full graph. Practically, one can limit the percentage of reduced COV according to their computation limit in advance.

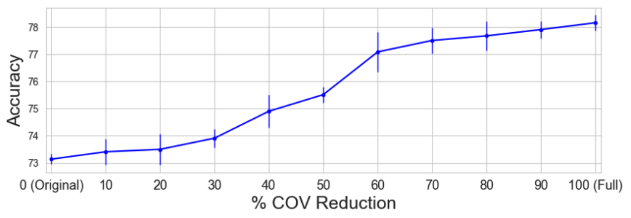

Figure 3: Accuracy and error bars of the Proteins datasets as the COV reduces. The performance is monotonically improving.

5. Practical Implications

In practice, possible graph structures are typically determined based on domain knowledge, and it is common to explore multiple possible structures. In some cases, a natural graph-structure inherently exists within the data, such as in social networks, where the network connections naturally define the graph layout. Nevertheless, it is usually not clear in advance if these graph layouts are informative for the task, or if the GNN will manage to exploit them. The fact that certain layouts may provide valuable information for the task while others might not, and this distinction isn't clear beforehand, was the driving question for our research. Indeed we found that the definition of the graph-structure, typically determined by users, emerges as a pivotal factor in performance outcomes due to the tendency of GNNs to overfit the provided graph. This revelation opens up a fascinating avenue for further research into the significance of topological information during the training of GNNs. Understanding how GNNs respond to different structural layouts and why certain graph-structures are more effective than others could significantly impact the way we design and train these models.

6. Future Work

We believe this work opens up many new avenues for exploration. One simple takeaway from our paper is to always try learning a model over empty graphs as well, i.e., using DeepSets (Zaheer et al., 2018). When the graph is known to have little contribution to the task, regularizing the topological weights may be useful. The main difficulty is finding ways to improve the GNN's ability to exploit useful information from the graph if it exists and ignore it otherwise, without prior knowledge. While we show in Section 4 that reducing the graph's COV can enhance performance, there may be other ways to mitigate the graph overfitting. In recent years many methods were introduced to mitigate different phenomena that limit GNNs performance (Rong et al., 2019; Alon & Yahav, 2021). It is interesting to examine whether these methods are useful in mitigating graph overfitting. Another interesting avenue for future research is analyzing the implicit bias of non-linear GNNs, including Graph Attention and Transformers.

7. Conclusion

In this study, we showed that although GNNs have the ability to disregard the provided graph when needed, they don't. Instead, GNNs tend to overfit the graph-structures, which results in reduced performance. We theoretically analyzed the implicit bias of gradient-descent learning of GNNs and proved that even with infinite data, GNNs are not guaranteed to learn a solution that ignores the graph when the graph should be ignored. We showed that regular graphs are more robust to graph overfitting, and provided a theoretical explanation and extrapolation results for this setting. Our study shows that in some cases, the graph structure hurts the performance of GNNs, and therefore graph selection is of great importance, as well as having a model that can ignore the graph when needed.

Acknowledgements

This work was supported by the Tel Aviv University Center for AI and Data Science (TAD), the Israeli Science Foundation grants 1186/18 and 1437/22, and a grant from Meta (Facebook) to Tel Aviv University.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications, 2021.

Barabasi, A.-L. and Albert, R. Emergence of scaling in random networks. *Science*, 286(5439): 509–512, 1999. doi: 10.1126/science.286.5439. 509. URL http://www.sciencemag.org/cgi/content/abstract/286/5439/509.

Brody, S., Alon, U., and Yahav, E. How attentive are graph attention networks?, 2022.

Chen, L., Chen, Z., and Bruna, J. On graph neural networks versus graph-augmented mlps, 2020.

Dobson, P. D. and Doig, A. J. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330 4:771–83, 2003.

Erdös, P. and Rényi, A. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290, 1959.

Errica, F., Podda, M., Bacciu, D., and Micheli, A. A fair comparison of graph neural networks for graph classification, 2022.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry, 2017.

Gunasekar, S., Lee, J. D., Soudry, D., and Srebro, N. Implicit bias of gradient descent on linear convolutional networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs, 2017. URL https://arxiv.org/abs/1706.02216.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs, 2020. URL https://arxiv.org/abs/2005.00687.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks, 2017a.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017b. URL https://openreview.net/forum?id=SJU4ayYgl.

Liu, M., Gao, H., and Ji, S. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pp. 338–348, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403076. URL https://doi.org/10.1145/3394486.3403076.

Lyu, K. and Li, J. Gradient descent maximizes the margin of homogeneous neural networks, 2020.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL www.graphlearning.io.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks, 2021.

Rong, Y., bing Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019. URL https://api.semanticscholar.org/CorpusID:212859361.

Seddik, M. E. A., Wu, C., Lutzeyer, J. F., and Vazirgiannis, M. Node feature kernels increase graph convolutional network robustness, 2022.

Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.*, 12:2539–2561, 2011. URL http://dblp.uni-trier.de/db/journals/jmlr/jmlr12.html#ShervashidzeSLMB11.

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. Masked label prediction: Unified message passing model for semi-supervised classification, 2021.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data, 2017. URL https://arxiv.org/abs/1710.10345.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 1365–1374, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336642. doi: 10.1145/2783258.2783417. URL https://doi.org/10.1145/2783258.2783417.

Yehudai, G., Fetaya, E., Meirom, E. A., Chechik, G., and Maron, H. On size generalization in graph neural networks. *CoRR*, abs/2010.08853, 2020. URL https://arxiv.org/abs/2010.08853.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., and Smola, A. Deep sets, 2018.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization, 2017.

Zhang, S., Liu, Y., Sun, Y., and Shah, N. Graph-less neural networks: Teaching old mlps new tricks via distillation, 2022.

A. Proofs and Extensions

All our analysis assumes that train and test data are labeled via some graph-less teacher. Namely, a function f^* that classifies a graph instance based only on its features and not the graph. We let f^* be defined via a weight vector $\mathbf{w}_1^*$ as follows:

$$f^*(X) = \mathbf{w}_1^* \cdot \sum_{i=1}^{n} \mathbf{x}_i \tag{4}$$

For the sake of simplicity, we assume that all the hidden states are of dimension d. We denote the number of vertices with n, the number of samples with m, and denote $\tilde{\mathbf{x}} = \sum_i^n \mathbf{x}_i$ for a set of node feature vectors $x_i \in G, 1 \leq i \leq n, 1 \leq l \leq m$. $\mathbf{x}_i^{(l)}$ is the feature vector of node i in the graph sample l.

A.1. Proof of Lemma 3.1

For the sake of simplicity, we begin by proving the simplest case of a GNN with one layer and no readout. Then we extend the proof to the case of readout and multiple layers. In r-regular graphs, $deg(i) = r$ for all nodes $i \in n$. Therefore Equation 2 can be written as:

$$\begin{array}{ll} \min_{\mathbf{w}_1,\mathbf{w}_2} & \|\mathbf{w}_1\|_2^2 + \|\mathbf{w}_2\|_2^2 \\ s.t. & y^{(l)}[(\mathbf{w}_1 + r\mathbf{w}_2) \cdot \tilde{\mathbf{x}}^{(l)}] \geq 1 \quad \forall (X^{(l)}, A^{(l)}, y^{(l)}) \in S \end{array}$$

Now, writing the KKT stationarity condition:

$$\begin{aligned} &\mathcal{L}(\mathbf{w}, \alpha) = \frac{1}{2}(\|\mathbf{w}_1\| + \|\mathbf{w}_2\|) - \sum_{l=1}^{m} \alpha_l [y^{(l)}(\mathbf{w}_1 + r\mathbf{w}_2)\tilde{\mathbf{x}}^{(l)} - 1] \\ &\nabla_{\mathbf{w}}\mathcal{L} = 0 \iff \\ &\frac{\partial \mathcal{L}}{\partial \mathbf{w}_1} = \mathbf{w}_1 - \sum_{l=1}^{m} \alpha_l y^{(l)} \tilde{\mathbf{x}}^{(l)} = 0 \implies \mathbf{w}_1 = \sum_{l=1}^{m} \alpha_l y^{(l)} \tilde{\mathbf{x}} \\ &\frac{\partial \mathcal{L}}{\partial \mathbf{w}_2} = \mathbf{w}_2 - r\sum_{l=1}^{m} \alpha_l y^{(l)} \tilde{\mathbf{x}}^{(l)} = 0 \implies \mathbf{w}_2 = r\sum_{l=1}^{m} \alpha_l y^{(l)} \tilde{\mathbf{x}}^{(l)} \end{aligned}$$

Therefore $\mathbf{w}_2 = r\mathbf{w}_1$. □

With Readout Assume a readout W_3 is applied after the sum-pooling, and denote $W = [W_1, W_2, W_3]$.

$$f(X, A, W) = W_3 \sum_{i}^{n} h_i^{(2)} = W_3 W_1 \cdot \sum_{i}^{n} \mathbf{x}_i + W_3 W_2 \cdot \sum_{i}^{n} \sum_{j \in N(i)} \mathbf{x}_j = (W_3 W_1 \cdot + r W_3 W_2)\, \tilde{\mathbf{x}}$$

Then similarly to Equation 3, the max-margin problem becomes:

$$\begin{array}{ll} \min_{W_1,W_2,W_3} & \|W_1\|_F^2 + \|W_2\|_F^2 + \|W_3\|_F^2 \\ s.t. & y^{(l)}\left[(W_3 W_1 \cdot + r W_3 W_2)\tilde{\mathbf{x}}^{(l)}\right] \geq 1 \quad \forall (X^{(l)}, A^{(l)}, y^{(l)}) \in S \end{array}$$

Then, the KKT stationarity condition:

$$\begin{aligned}
&\mathcal{L}(W,\alpha) = \frac{1}{2}(\|W_3\|_F^2 + \|W_1\|_F^2 + \|W_2\|_F^2) - \sum_{l=1}^{m} \alpha_l \left(y^{(l)} W_3 \left[\sum_{i=1}^{n} \left(W_1 \mathbf{x}_i^{(l)} + r W_2 \mathbf{x}_i^{(l)} \right) \right] - 1 \right) \\
&\frac{\partial \mathcal{L}}{\partial W_1} = W_1 - \sum_{l=1}^{m} \alpha_l y^{(l)} \sum_{i=1}^{n} W_3 \mathbf{x}_i^{(l)} \\
&\frac{\partial \mathcal{L}}{\partial W_2} = W_2 - r \sum_{l=1}^{m} \alpha_l y^{(l)} \sum_{i=1}^{n} W_3 \mathbf{x}_i^{(l)} \\
&\frac{\partial \mathcal{L}}{\partial W_3} = W_3 - \sum_{l=1}^{m} \alpha_l y^{(l)} \sum_{i=1}^{n} \left(W_1 \mathbf{x}_i^{(l)} + r W_2 \mathbf{x}_i^{(l)} \right) \\
&\nabla_W \mathcal{L} = 0 \iff \\
&W_1 = W_3 \sum_{l=1}^{m} \alpha_l y^{(l)} \sum_{i=1}^{n} \mathbf{x}_i^{(l)} \\
&W_2 = r W_3 \sum_{l=1}^{m} \alpha_l y^{(l)} \sum_{i=1}^{n} \mathbf{x}_i^{(l)} \\
&W_3 = \sum_{l=1}^{m} \alpha_l y^{(l)} \left(W_1 + r W_2 \right) \left(\sum_{i=1}^{n} \mathbf{x}_i^{(l)} \right)
\end{aligned}$$

Therefore W_1 and W_2 are aligned, as well as $W_3 W_1$ and $W_3 W_2$ $\square$

Two Layers The updates in a 2-layer GNN are:

$$\begin{aligned}
\mathbf{h}_i^{(1)} &= W_1^{(0)} \cdot \mathbf{x}_i + W_2^{(0)} \sum_{j \in N(i)} \mathbf{x}_j \\
\mathbf{h}_i^{(2)} &= W_1^{(1)} \cdot \mathbf{h}_i^{(1)} + W_2^{(1)} \sum_{j \in N(i)} \mathbf{h}_j^{(1)}
\end{aligned}$$

Then the final predictor is:

$$f(X,A,W) = W_3 \left(W_1^{(1)} W_1^{(0)} + \left(W_1^{(1)} W_2^{(0)} + W_2^{(1)} W_1^{(0)} \right) r + W_2^{(1)} W_2^{(0)} r^2 \right) \sum_{i}^{n} \mathbf{x}_i$$

Therefore, let $W = [W_1^{(0)}, W_2^{(0)}, W_1^{(1)}, W_2^{(1)}, W_3]$ and we define:

$$P(W) = W_3 \left(W_1^{(1)} W_1^{(0)} + \left(W_1^{(1)} W_2^{(0)} + W_2^{(1)} W_1^{(0)} \right) r + W_2^{(1)} W_2^{(0)} r^2 \right)$$

In this case Equation 2 becomes the following max-margin problem:

$$\begin{array}{lll}
\min_W & \|W_1^{(0)}\|_F^2 + \|W_2^{(0)}\|_F^2 + \|W_1^{(1)}\|_F^2 + \|W_2^{(1)}\|_F^2 + \|W_3\|_F^2 & \\
s.t. & y^{(l)} \left[P(W) \cdot \tilde{\mathbf{x}}^{(l)} \right] \geq 1 & \forall (X^{(l)}, A^{(l)}, y^{(l)}) \in S
\end{array} \tag{5}$$

Using the KKT stationarity condition:

$$\mathcal{L}(W,\alpha) = \frac{1}{2}\left(\sum_{\substack{t\in\{0,1\}\\k\in\{0,1\}}} \|W_t^{(k)}\|_F^2 + \|W_3\|_F^2\right) - \sum_{l=1}^{m} \alpha_l [yP(W)\cdot\tilde{\mathbf{x}}^{(l)} - 1]$$
$$\frac{\partial\mathcal{L}}{\partial W_j} = W_j - \sum_{l=1}^{m}\alpha_l y \frac{\partial}{\partial W_j}(P(W)\cdot\tilde{\mathbf{x}}_i^{(l)})$$
$$\frac{\partial}{\partial W_j}(P(W)\cdot\tilde{\mathbf{x}}^{(l)}) = \tilde{\mathbf{x}}^{(l)}\frac{\partial P(W)}{\partial W_j}$$
$$\frac{\partial\mathcal{L}}{\partial W_j} = 0 \implies W_j = \sum_{l=1}^{m}\alpha_l y^{(l)}\tilde{\mathbf{x}}_i^{(l)}\frac{\partial P(W)}{\partial W_j}$$

$$\frac{\partial P(W)}{\partial W_1^{(0)}} = \frac{\partial}{\partial W_1^{(0)}}[W_3(W_1^{(1)} + rW_2^{(1)})W_1^{(0)}]$$
$$\frac{\partial P(W)}{\partial W_2^{(0)}} = \frac{\partial}{\partial W_2^{(0)}}[W_3(rW_1^{(1)} + r^2W_2^{(1)})W_2^{(0)}] = r\frac{\partial P(W)}{\partial W_1^{(0)}}$$

$$\frac{\partial P(W)}{\partial W_1^{(1)}} = \frac{\partial}{\partial W_1^{(1)}}[W_3W_1^{(1)}(W_1^{(0)} + W_2^{(0)}r)]$$
$$\frac{\partial P(W)}{\partial W_2^{(1)}} = \frac{\partial}{\partial W_2^{(1)}}[W_3W_2^{(1)}(W_1^{(0)}r + W_2^{(0)}r^2)] = r\frac{\partial P(W)}{\partial W_1^{(1)}}$$

Therefore, the root and topological weights are aligned in every layer. □

L Layers - Overview In the case of L layers, the same holds, only $P(W)$ is different. We provide the sketch of the proof. The max-margin problem is

$$\begin{aligned} \min\nolimits_W &\quad \|W_3\|_F^2 + \textstyle\sum_{k=0}^{L-1}\|W_0^{(k)}\|_F^2 + \|W_1^{(k)}\|_F^2 \\ s.t. &\quad y^{(l)}[P(W)\tilde{\mathbf{x}}^{(l)}] \geq 1 \qquad\qquad \forall(X^{(l)},A^{(l)},y^{(l)})\in S \end{aligned} \tag{6}$$

Then, the KKT stationarity condition:

$$\mathcal{L}(W,\alpha) = \|W_3\|_F^2 + \sum_{k=0}^{L-1}\|W_0^{(k)}\|_F^2 + \|W_1^{(k)}\|_F^2 - \sum_{l=1}^{m}\alpha_l[y^{(l)}P(W)\tilde{\mathbf{x}}^{(l)} - 1]$$
$$\frac{\partial\mathcal{L}}{\partial W_j^{(k)}} = 2W_j^{(k)} - \sum_{l=1}^{m}\alpha_l y^{(l)}\frac{\partial}{\partial W_j^{(k)}}\left(P(W)\tilde{\mathbf{x}}^{(l)}\right)$$
$$\frac{\partial\mathcal{L}}{\partial W_j^{(k)}} = 0 \implies W_j^{(k)} = \frac{1}{2}\frac{\partial P(W)}{\partial W_j^{(k)}}\cdot\sum_{l=1}^{m}\alpha_l y^{(l)}\tilde{\mathbf{x}}^{(l)}$$

Therefore, to show alignment between the root and topological weights in layer k, it is enough to show that

$$r\frac{\partial P(W)}{\partial W_0^{(k)}} = \frac{\partial P(W)}{\partial W_1^{(k)}}$$

A.2. Proof of Theorem 3.2

Here, we prove Theorem 3.2 by providing distributions P1 and P2 such that GNNs trained on graphs from P1 will fail to extrapolate to graphs from P2. We consider the case where P1 is a distribution over r-regular graphs, and P2 is a distribution over star graphs with a random center node. The key intuition in our proof is that learning with P1 will learn a model that averages over nodes. However when testing it on P2, mostly the center node will have to determine the label, and this will typically result in an error. We will take the graph size to ∞ to simplify the analysis, but results for finite graphs with high probability can be obtained using standard concentration results.

Let f^* be a graph-less function, $f^*(X) = w_1^* \sum_{i=1}^n x_i$. We assume that f^* labels the training graphs, and graphs are drawn from $P1$, namely a distribution over r-regular graphs. Let f be the learned function when trained with infinite data on r-regular graphs, with node features with dimension 1 drawn from $\mathcal{N}(0,1)$. Then $f(G) = sign(w_1 \sum_{i=1}^n x_i + rw_1 \sum_{i=1}^n deg(i)x_i)$, where w_1 and $rw_1 = w_2$ (following Lemma 3.1) are the learned parameters.

We now proceed to show that extrapolation to $P2$ fails in this case. Let G be a star graph, with features of dimension 1 drawn from $\mathcal{N}(0,1)$, and assume w.l.o.g. that the center node of the star has index 1. Then applying the f (learned on $P1$) to this G can be written as

$$f(G) = sign(w_1 \sum_{i=1}^n \mathbf{x}_i + rw_1(n-1)x_1 + rw_1 \sum_{i=2}^n x_i)$$

We will first show that when the number of vertices grows to infinity, the sign is determined by the first (central) node. Denote

$$X = rw_1(n-1)x_1 \quad Y = w_1 \textstyle\sum_{i=1}^n x_i \quad Z = rw_1 \textstyle\sum_{i=2}^n x_i \quad W = X + Y + Z$$

We will show that the correlation coefficient between X and W goes to 1 as the number of vertices n approaches infinity. It holds that

$$X \sim \mathcal{N}(0, r^2 w_1^2 (n-1)^2) \quad Y \sim \mathcal{N}(0, w_1^2 n) \quad Z \sim \mathcal{N}(0, r^2 w_1^2 (n-1))$$

$$\begin{aligned}
&Var(W) = Var(X+Z+Y) = Var(X+Z) + Var(Y) + 2COV(X+Z,Y)\\
&COV(X+Z,Y) = COV(X,Y) + COV(Z,Y)\\
&COV(X,Y) = COV(rw_1(n-1)x_1, w_1 \sum_{i=1}^n x_i) = rw_1^2(n-1)COV(x_1,x_1) = rw_1^2(n-1)\\
&COV(Y,Z) = COV(w_1 \sum_{i=1}^n x_i, rw_1 \sum_{i=2}^n x_i) = rw_1^2) \sum_{i=2}^n COV(x_i,x_i) = rw_1^2(n-1)\\
&\implies COV(X+Z,Y) = 2rw_1^2(n-1)\\
&\implies Var(W) = r^2w_1^2(n-1)^2 + r^2w_1^2(n-1) + w_1^2 n + 2rw_1^2(n-1)\\
&COV(X,W) = COV(rw_1(n-1)x_1, w_1 \sum_{i=1}^n x_i + rw_1 \sum_{i=2}^n x_i + rw_1(n-1)x_1)\\
&= rw_1(n-1)[w_1 COV(x_1,x_1) + rw_1(n-1)COV(x_1,x_1)] = rw_1^2(n-1) + r^2w_1^2(n-1)^2\\
&\rho(X,W) = \frac{rw_1^2(n-1) + r^2w_1^2(n-1)^2}{(rw_1(n-1))(w_1\sqrt{r^2(n-1)^2 + r^2(n-1) + n + 2r(n-1)})}\\
&= \frac{rw_1^2(n-1)(1 + r(n-1))}{\sqrt{r^2(n-1)^2 + r^2(n-1) + n + 2r(n-1))}} = \frac{1 + r(n-1)}{\sqrt{r^2(n-1)^2 + r^2(n-1) + n + 2r(n-1)}}\\
&\frac{rn + 1 - r}{\sqrt{rn^2 + (1 + 2r - 3r^2)n - 2r}} \to_{n\to\infty} 1
\end{aligned}$$

As X and W are fully correlated, they have the same sign. We will now show that when n approaches infinity, the probability that X will have a different sign from $w_1^* \sum_{i=1}^n x_i$ is 0.5, and therefore conclude that the error on $P2$ is > 0.25 as specified in the theorem. We will do so by showing that the correlation coefficient between X and $w_1^* \sum_{i=1}^n x_i$ converges to 0.

$$COV(X, w_1^* \sum_{i=1}^{n} x_i) = r_1^w (n-1) w_1^*$$

$$\rho(X, w_1^* \sum_{i=1}^{n} x_i) = \frac{r w_1 (n-1) w_1^*}{r w_1 (n-1) w_1^* \sqrt{n}} \rightarrow_{n\rightarrow\infty} 0$$

We conclude that a model trained on $P1$ will fail to extrapolate to $P2$.

□

A.3. Proof of Theorem 3.3

We consider the case of a graph-less teacher as in (4) We wish to show that if the training data consists of infinitely many samples from a distribution over r-regular graphs, then the learned model will extrapolate perfectly to a distribution over r'-regular graphs. We assume the same feature distribution in all cases.

Let $f = [\mathbf{w}_1, \mathbf{w}_2]$ be a minimizer of Equation 3 on the training distribution. Then f has perfect accuracy on the support of the training distribution (ie it is equal to the graph-less teacher f^* there). Let $G = (X_G, A_r)$ be in the support of the training distribution. Then:

$$f(G) = sign\left((\mathbf{w}_1 + r\mathbf{w}_2)\tilde{x}_G\right) \overset{(*)}{=} sign\left((\mathbf{w}_1 + r^2\mathbf{w}_1)\tilde{x}_G\right) \overset{(**)}{=} sign\left(\mathbf{w}_1\tilde{x}_G\right) \overset{(***)}{=} sign\left(\mathbf{w}_1^*\tilde{x}_G\right) \tag{7}$$

(*) Follows from Theorem 3.1 by substituting $\mathbf{w}_2 = r\mathbf{w}_1$. (**) Follows from the fact that the direction of $(\mathbf{w}_1 + r^2\mathbf{w}_1)$ and $\mathbf{w}_1$ is the same. (***) Follows from the fact that f is equal to f^* on the training distribution.

Now let $G_{r'} = (X_{G_{r'}}, A_{r'})$ be an r'-regular graph example, with features drawn from D. Following Equation 2, we get that:

$$f(G_{r'}) = sign\left((\mathbf{w}_1 + r'\mathbf{w}_2)\tilde{x}_{G_{r'}}\right) = sign\left((\mathbf{w}_1 + r'r\mathbf{w}_1)\tilde{x}_{G_{r'}}\right) \overset{(***)}{=} sign\left(\mathbf{w}_1\tilde{x}_{G_{r'}}\right) = sign\left(\mathbf{w}_1^*\tilde{x}_{G_{r'}}\right)$$

(***) Follows from the assumption that the features are drawn from D. We thus have that all instances drawn from the test distribution of r'-regular graphs are classified correctly, and therefore we have perfect extrapolation in this case. □

A.4. Proof of Theorem 3.4

Let f^* be a graph-less function as in (4), and f be a GNN minimizing Equation 3, on a training set of r-regular graph examples. Assume we have modified an example in S from $G = (X, A)$ to $\tilde{G} = (X, \tilde{A})$. Let $\tilde{x} = \sum_{i=1}^{n} \mathbf{x}_i,\ \mathbf{x}_i \in G$. Let $0 \le r' \le n-1,\ r' \in \mathbb{N}$ and let $\Delta_{r',\tilde{G}}(i) = deg_{\tilde{G}}(i) - r'$.
Then using Equation 2:

$$\begin{aligned} f(\tilde{G}) &= \mathbf{w}_1\tilde{x} + \mathbf{w}_2 \sum_{i=1}^{n} deg(i)\mathbf{x}_i \overset{(*)}{=} \mathbf{w}_1\tilde{x} + r\mathbf{w}_1 \sum_{i=1}^{n} deg(i)\mathbf{x}_i \\ &= \mathbf{w}_1\tilde{x} + r\mathbf{w}_1 \sum_{i=1}^{n} (\Delta_{r',\tilde{G}}(i) + r')\mathbf{x}_i \\ &= \mathbf{w}_1\tilde{x} + r\mathbf{w}_1 \sum_{i=1}^{n} \Delta_{r',\tilde{G}}(i) + r'r\mathbf{w}_1 \sum_{i=1}^{n} \mathbf{x}_i \\ &= \mathbf{w}_1\tilde{x} + r'r\mathbf{w}_1\tilde{x} + r\mathbf{w}_1 \sum_{i=1}^{n} \Delta_{r',\tilde{G}}(i) \\ &= \underbrace{\mathbf{w}_1\tilde{x} + r'r\mathbf{w}_1\tilde{x}}_{Regular\ Component} \underbrace{+ r\mathbf{w}_1 \sum_{i=1}^{n} \Delta_{r',\tilde{G}}(i)\mathbf{x}_i}_{\Delta\ Component} \end{aligned} \tag{8}$$

Where (*) follows from Theorem 3.1.

Now assume there exists an r' such that:

$$\left| \frac{r\mathbf{w}_1 \sum_{i=1}^{n} \Delta_{r',\tilde{G}}(i)\mathbf{x}_i}{\mathbf{w}_1\tilde{x} + r'r\mathbf{w}_1\tilde{x}} \right| \leq 1$$

Therefore, the Δ-component is small with respect to the regular component and can be dropped below because it doesn't change the sign.

$$f(\tilde{G}) = sign(\mathbf{w}_1\tilde{x} + r'r\mathbf{w}_1\tilde{x}) \overset{(**)}{=} f^*(\tilde{G})$$

Where $(**)$ follows from Theorem 3.3.

B. Additional Experimental Results

B.1. Empirical Validation of Lemma 3.1

The validation of Theorem 3.1 is presented in Figure 4. We plot the ratio between the topological weights and root weights, during the training of linear GNNs with one or two layers, with readout. The GNNs are trained on regular graphs with different regularity degrees. In all cases, the ratio converges to the regularity degree, as guaranteed by Theorem 3.1.

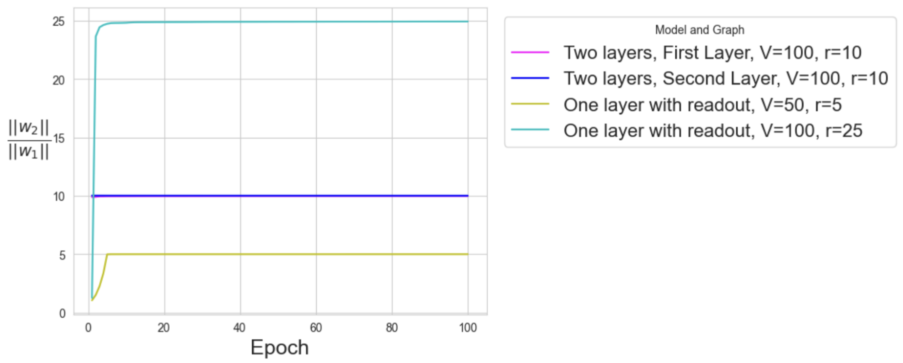

Figure 4: An empirical validation of Theorem 3.1. The ratio between the topological and root weights is equal to the regularity degree of the graphs. V is the number of nodes in each graph, and r is the regularity degree.

B.2. Evidences For Graph Overfitting With Additional Models (Section 2.2)

In Section 2.2, we presented an empirical evaluation of the model from Gilmer et al. (2017) as described in Equation 1. Here we provide the results of the same evaluation with more GNNs. All models show similar trends as presented in the main paper. The results are shown in Figures 5 (GIN (Xu et al., 2019)), 6 (GAT (Veličković et al., 2018)), 7 (Graph Transformer (Shi et al., 2021)) and 8 (GraphConv with Normalized Neighbor Aggregation (Gilmer et al., 2017)).

B.3. Evidence For Graph Overfitting With Node Task (Section 2.2)

We evaluated the learning curve in a teacher-student setup of a graph classification task, where the teacher is graph-less GNN. The teacher readout is sampled once from $\mathcal{N}(0, 1)$ to generate the train, validation, and test labels. The training graph is over 4000 nodes, and the validation and test graphs are over 500 nodes. Each node is assigned with a feature vector in $\mathbb{R}^{128}$ sampled i.i.d from $\mathcal{N}(0, 1)$. Figure 9 shows that also, in this case, although the teacher does not use the graph, giving the model different graphs affects generalization. Therefore, in this case, the GNN overfits the given graph, although it should be ignored.

B.4. Additional Results on How Graph Structure Affects Overfitting (Section 2.3)

In Section 2.3 for the sake of presentation, we presented only one curve from each distribution. Figure 10 presents the learning curve of all the distributions we tested, with multiple parameters for each distribution. Additionally, in Figure 11, we present the weights norms of the root and topological weights separately for the curves presented in the main paper.

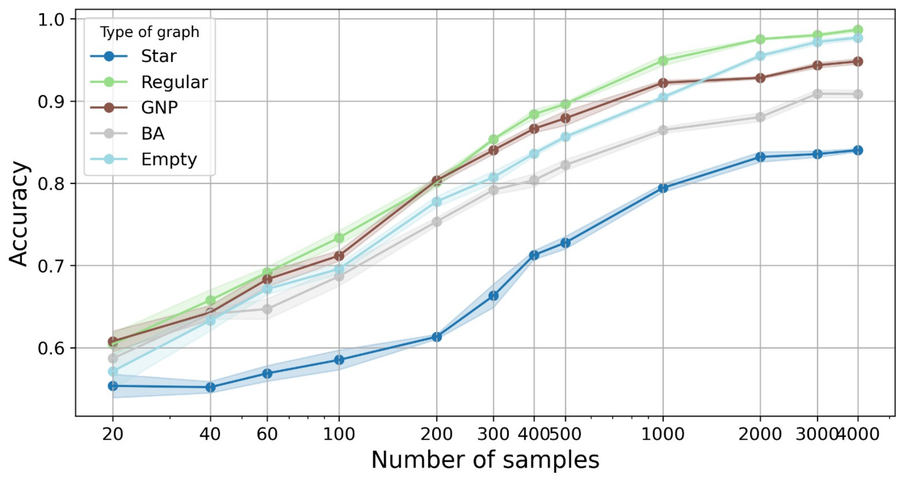

Figure 5: Evaluation of the GIN (Xu et al., 2019) model on the Sum task where the graph should be ignored, as described in Section 2.2 in the main paper.

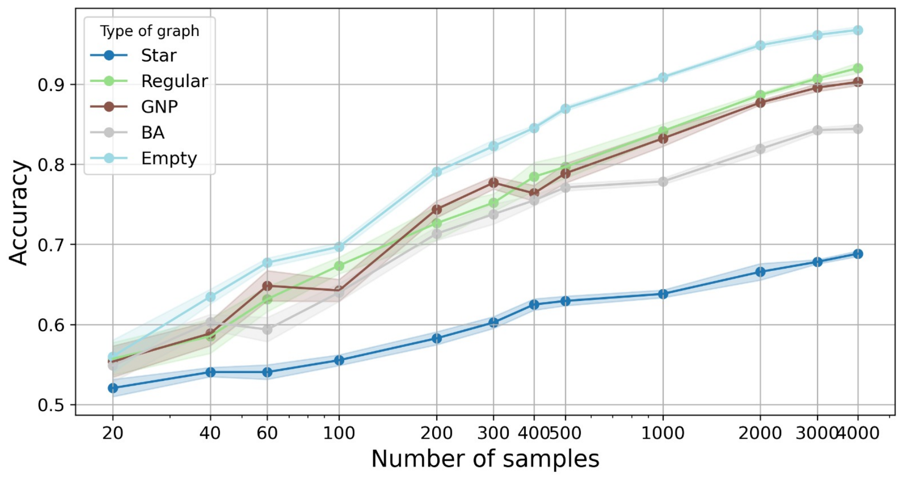

Figure 6: Evaluation of the GAT (Veličković et al., 2018) model on the Sum task where the graph should be ignored, as described in Section 2.2 in the main paper.

C. Additional Experimental Details

C.1. Additional Implementation Details For Section 2.2

The teacher readout is sampled once from $\mathcal{N}(0, 1)$ and used for all the graphs. All graphs have $n = 20$ nodes, and each node is assigned with a feature vector in $\mathbb{R}^{128}$ sampled i.i.d from $\mathcal{N}(0, 1)$.

For the Sum task, we used a 1-layer "student" GNN following the teacher model, with readout and ReLU activations. For the PROTEINS and ENZYMES tasks, we used 3-layers.

We evaluated the learning curve with an increasing amount of $[20, 40, 60, 100, 200, 300, 400, 500, 1000, 2000, 4000]$ samples. We note that the GNN has a total of ~16,000 parameters, and thus, it is overparameterized and can fit the training data with perfect accuracy.

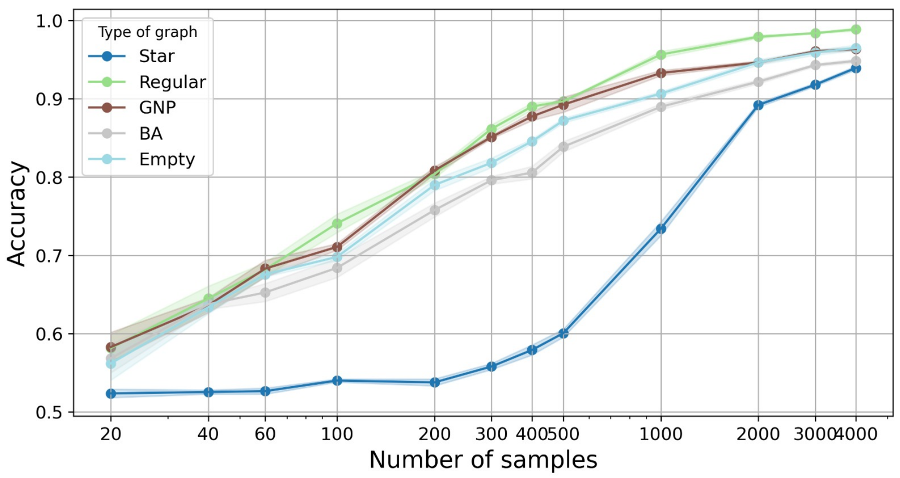

Figure 7: Evaluation of the Graph Transformer (Shi et al., 2021) model on the Sum task where the graph should be ignored, as described in Section 2.2 in the main paper.

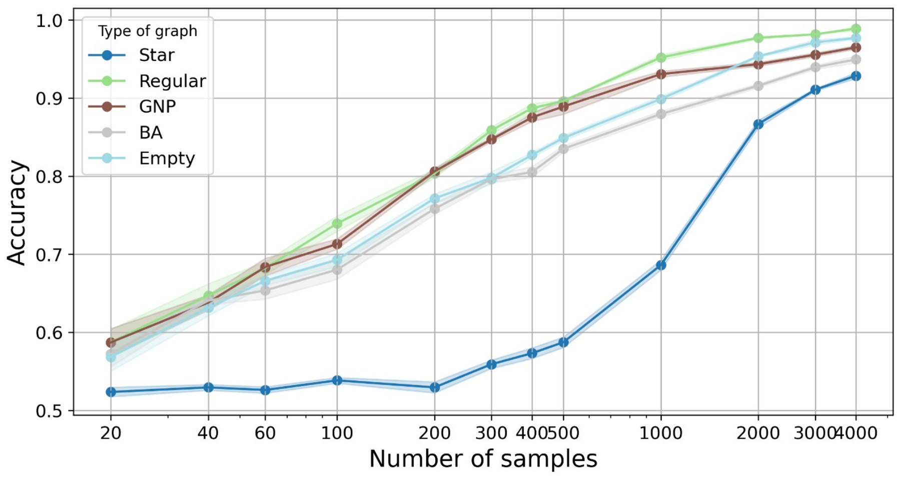

Figure 8: Evaluation of the same model presented in Equation 1 (Gilmer et al., 2017) with normalized neighbor aggregation on the Sum task where the graph should be ignored, as described in Section 2.2 in the main paper.

C.2. Experimental setup in Section 3.2.1

We trained a one-layer linear GNN with readout on 5-regular graphs over 20 nodes. We then applied it to the test sets presented in Table 2. Each test set contains 100 graph examples, and each graph has 20. All the test sets share the same node features and differ in the graph structure, which is drawn from different graph distributions.

C.3. Additional Dataset Information

The dataset statistics are summarized in Table 4.

IMDB-B & IMDB-M (Yanardag & Vishwanathan, 2015) are movie collaboration datasets. Each graph is derived from a genre, and the task is to predict this genre from the graph. Nodes represent actors/actresses, and edges connect them if they have appeared in the same movie.

Proteins, D&D &Enzymes (Shervashidze et al., 2011; Dobson & Doig, 2003) are datasets of chemical compounds. The goal in the first two datasets is to predict whether a compound is an enzyme or not, and the goal in the last datasets is to classify the type of an enzyme among 6 classes.

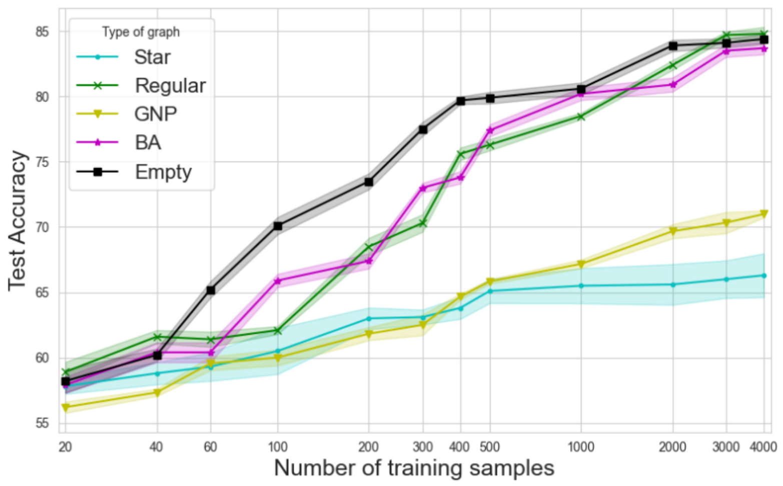

Figure 9: Evaluation of the same model presented in Equation 1 (Gilmer et al., 2017) on the Sum task for node classification where the graph should be ignored.

Table 4: Statistics of the real-world datasets used in our evaluation.

Dataset	# Graphs	Avg # Nodes	Avg # Edges	# Node Features	# Classes
Proteins	1113	39.06	72.82	3	2
NCI1	4110	29.87	32.3	37	2
Enzymes	600	32.63	62.14	3	6
D& D	1178	284.32	715.66	89	2
IMDB-B	1000	19	96	0	2
IMDB-M	1500	13	65	0	3
Collab	5000	74.49	2457.78	0	3
Reddit-B	2000	429.63	497.75	0	2
Reddit-5k	4999	508.52	594.87	0	5
mol-hiv	41,127	25.5	27.5	9	2
mol-pcba	437,929	26.0	28.1	9	2 (128 tasks)

NCI1 (Shervashidze et al., 2011) is a datasets of chemical compounds. Vertices and edges represent atoms and the chemical bonds between them. The graphs are divided into two classes according to their ability to suppress or inhibit tumor growth.

Collab (Morris et al., 2020) is a scientific collaboration dataset. A graph corresponds to a researcher's ego network, i.e., the researcher and their collaborators are nodes, and an edge indicates collaboration between two researchers. A researcher's ego network has three possible labels, which are the fields that the researcher belongs to.

Reddit-B, Reddit-5k (Morris et al., 2020) are datasets of Reddit posts from the month of September 2014, with binary and multiclass labels, respectively. The node label is the community, or "subreddit", that a post belongs to. 50 large communities have been sampled to build a post-to-post graph, connecting posts if the same user comments on both.

mol-hiv, mol-pcba (Hu et al., 2020) are large-scale datasets of molecular property prediction.

Following Errica et al. (2022), we added a feature of the node degrees for datasets that have no node features at all.

C.4. Hyper-Parameters

All GNNs use ReLU activations with $\{3, 5\}$ layers and 64 hidden channels. They were trained with Adam optimizer over 1000 epochs and early on the validation loss with a patient of 100 steps, eight Decay of $1e-4$, learning rate in

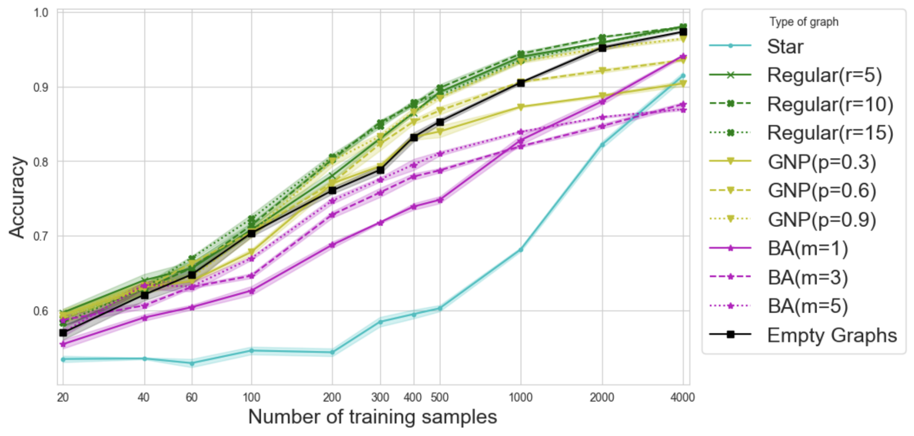

Figure 10: The learning curves of the same GNN model trained on graphs that have the same node features and only differ in their graph-structure. The label is computed via a graphless teacher. If GNNs were to ignore the non-informative graph-structure they were given, similar performance should have been observed for all graph distributions. Among the different distributions, regular graphs exhibit the best performance.

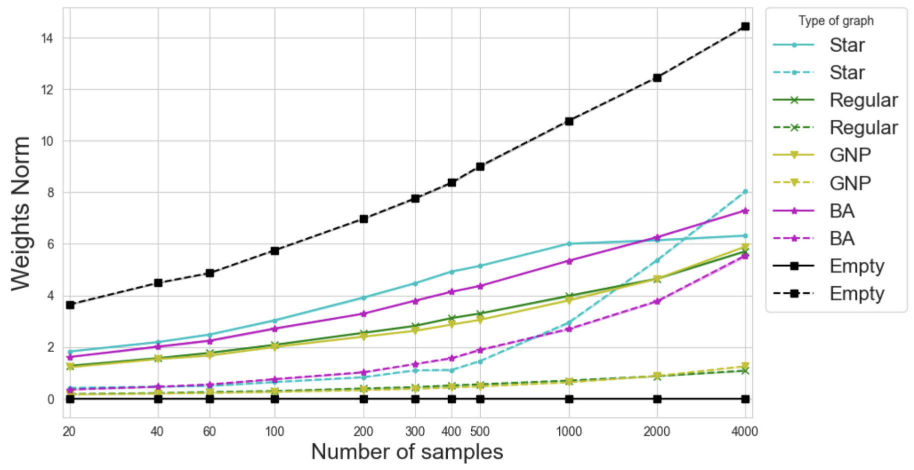

Figure 11: The weights norm of the topological (dashed) and the root (smooth) weights along the same runs. On the empty graphs, the topological weights are not trained and the ratio is 0 due to initialization.

$\{1e-3, 1e-4\}$, dropout rate in $\{0, 0.5\}$, and a train batch size of 32. The preserved COV is among $\{80\%, 50\%\}$.

C.5. Models

In Section 4, when evaluating graphs with reduced COV, we add edge features to differ between the original and added edges. We adapt each neighbor's aggregation component to process this edge information in a non-linear way.

GraphConv

$$\mathbf{x}' = W_1 \mathbf{x}_i + W_2 \sum_{j \in N(i)} x_i + \sigma(W_3 \mathbf{e}_{i,j})$$

GIN

$$\mathbf{x}' = W_4 \left((1+\epsilon)\mathbf{x}_i + W_2 \sum_{j \in N(i)} \sigma\left(\mathbf{x}_i + \sigma(W_3 \mathbf{e}_{i,j})\right) \right)$$

GATv2

$$x' = \alpha_{i,i} W_1 \mathbf{x}_i + \sum_{j \in N(i)} \alpha_{i,j} W_2 \mathbf{x}_j$$

$$\alpha_{i,j} = \frac{exp\left(a^T LeakyReLU\left(W_1\mathbf{x}_i + W_2\mathbf{x}_j + \sigma(W_3\mathbf{e}_{i,j})\right)\right)}{\sum_{k \in N(i) \cup \{i\}} exp\left(a^T LeakyReLU\left(W_1\mathbf{x}_i + W_2\mathbf{x}_k + \sigma(W_3\mathbf{e}_{i,k})\right)\right)}$$

GraphTransformer

$$x' = W_1 \mathbf{x}_i + \sum_{j \in N(i)} \alpha_{i,j} \left(W_2 \mathbf{x}_j + \sigma(W_5 \mathbf{e}_{i,j})\right)$$

$$\alpha_{i,j} = \frac{(W_3\mathbf{x}_i)^T (W_4\mathbf{x}_j + W_5\mathbf{e}_{i,j})}{\sqrt{d}}$$