From Local Structures to Size Generalization in Graph Neural Networks

Gilad Yehudai [1] **Ethan Fetaya** [2] **Eli Meirom** [1] **Gal Chechik** [1 2] **Haggai Maron** [1]

Abstract

Graph neural networks (GNNs) can process graphs of different sizes, but their ability to generalize across sizes, specifically from small to large graphs, is still not well understood. In this paper, we identify an important type of data where generalization from small to large graphs is challenging: graph distributions for which the local structure depends on the graph size. This effect occurs in multiple important graph learning domains, including social and biological networks. We first prove that when there is a difference between the local structures, GNNs are not guaranteed to generalize across sizes: there are "bad" global minima that do well on small graphs but fail on large graphs. We then study the size-generalization problem empirically and demonstrate that when there is a discrepancy in local structure, GNNs tend to converge to non-generalizing solutions. Finally, we suggest two approaches for improving size generalization, motivated by our findings. Notably, we propose a novel Self-Supervised Learning (SSL) task aimed at learning meaningful representations of local structures that appear in large graphs. Our SSL task improves classification accuracy on several popular datasets.

1. Introduction

Graphs are a flexible representation, widely used for representing diverse data and phenomena. In recent years, graph neural networks (GNNs), deep models that operate over graphs, have become the leading approach for learning on graph-structured data (Bruna et al., 2013; Kipf & Welling, 2016a; Veličković et al., 2017; Gilmer et al., 2017).

In many domains, graphs vary significantly in size. This is the case in molecular biology, where molecules are represented as graphs and their sizes vary from few-atom compounds to proteins with thousands of nodes. Graph sizes are even more heterogeneous in social networks, ranging from dozens of nodes to billions of nodes. Since a key feature of GNNs is that they can operate on graphs regardless of their size, a fundamental question arises: **"When do GNNs generalize to graphs of sizes that were not seen during training?"**.

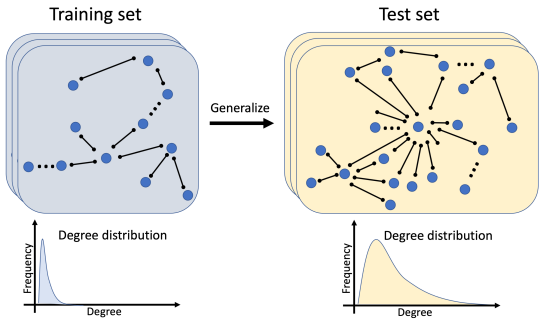


Figure 1. We study the ability of GNNs to generalize from small to large graphs, focusing on graphs in which the local structure depends on the graph size. The figure shows two graph distributions that differ in size and degree distribution. We show that when the local structures in the test set are different from the local structures in the training set, it is difficult for GNNs to generalize. Additionally, we suggest ways to improve generalization.

Aside from being an intriguing theoretical question, the size-generalization problem has important practical implications. In many domains, it is hard to collect ground-truth labels for large graphs. For instance, many combinatorial optimization problems can be represented as graph classification problems, but labeling large graphs for training may require solving large and hard optimization problems. In other domains, it is often very hard for human annotators to correctly label complex networks. It would therefore be highly valuable to develop techniques that can train on small graphs and generalize to larger graphs. This first requires that we develop an understanding of size generalization.

In some cases, GNNs can naturally generalize to graphs whose size is different from what they were trained on, but it is largely unknown when such generalization occurs. Empirically, several papers report good size-generalization performance (Li et al., 2018; Luz et al., 2020; Sanchez-Gonzalez et al., 2020). Other papers (Velickovic et al.,

[1]NVIDIA [2]Bar-Ilan University. Correspondence to: Gilad Yehudai <gilad.yehudai@weizmann.ac.il>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

2019; Khalil et al., 2017; Joshi et al., 2020) show that size generalization can be hard. Recently, Xu et al. (2020) provided theoretical evidence of size generalization capabilities in one-layer GNNs.

The current paper characterizes an important type of graph distributions where size generalization is challenging. Specifically, we analyze graphs for which the distribution of local structures (defined formally in Sec. 4) depends on the size of the graph. See Fig. 1 for an illustrative example. This dependency is prevalent in a variety of graphs, including for instance, the preferential attachment (PA) model (Barabási & Albert, 1999), which captures graph structure in social networks (Barabâsi et al., 2002), biological networks (Eisenberg & Levanon, 2003; Light et al., 2005) and internet link data (Capocci et al., 2006). In PA, the maximal node degree grows with the graph size. As a second example, in a graph representation of dense point clouds, the node degree grows with the cloud density, and hence with the graph size (Hermosilla et al., 2018).

To characterize generalization to new graph sizes, we first formalize a representation of local structures that we call d-patterns, inspired by (Weisfeiler & Lehman, 1968; Morris et al., 2019; Xu et al., 2018). d-patterns generalize the notion of node degrees to a d-step neighborhood of a given node, capturing the values of a node and its d-step neighbors, as seen by GNNs. We then prove that even a small discrepancy in the distribution of d-patterns between the test and train distributions may result in weight assignments that do not generalize well. Specifically, it implies that when training GNNs on small graphs there exist "bad" global minima that fail to generalize to large graphs.

We then study empirically the relation between size generalization and d-pattern discrepancy in synthetic graphs where we control the graph structure and size. We find that as d-pattern discrepancy grows, the generalization of GNNs to new graph sizes deteriorates.

Finally, we discuss approaches for improving size generalization. We take a self-supervised learning approach and propose a novel pretext task aimed at learning useful d-pattern representations from both small and large graphs. We show that when training on labeled small graphs and with our new self-supervised task on large graphs, classification accuracy increases on large graphs by 4% on average on real datasets.

This paper makes the following contributions: (1) We identify a family of important graph distributions where size generalization is difficult, using a combination of theoretical and empirical results. (2) We suggest approaches for improving size generalization when training on such distributions and show that they lead to a noticeable performance gain. The ideas presented in this paper can be readily extended to other graph learning setups where there is a discrepancy between the local structures of the train and test sets.

2. Preliminaries

Notation. We denote by $\{(a_1, m_{a_1}), \ldots, (a_n, m_{a_n})\}$ a **multiset**, that is, a set where we allow multiple instances of the same element. Here $a_1, \ldots, a_n$ are distinct elements, and m_{a_i} is the number of times a_i appears in the multiset. Bold-face letters represent vectors.

Graph neural networks. In our theoretical results, we focus on the message-passing architecture from (Morris et al., 2019). Let $G = (V, E)$ be a graph, and for each node $v \in V$ let $\mathbf{h}_v^{(0)} \in \mathbb{R}^{d_0}$ be a node feature vector and $\mathcal{N}(v)$ its set of neighbors. The t-th layer of first-order GNN is defined as follows for $t > 0$:

$$\mathbf{h}_v^{(t)} = \sigma\left(W_2^{(t)}\mathbf{h}_v^{(t-1)} + \sum_{u \in \mathcal{N}(v)} W_1^{(t)}\mathbf{h}_u^{(t-1)} + \mathbf{b}^{(t)}\right).$$

Here, $W_1^{(t)}, W_2^{(t)} \in \mathbb{R}^{d_t \times d_{t-1}}$, $\mathbf{b}^{(t)} \in \mathbb{R}^{d_t}$ denotes the parameters of the t-th layer of the GNN, and σ is some non-linear activation (e.g ReLU). It was shown in (Morris et al., 2019) that GNNs composed from these layers have maximal expressive power with respect to all message-passing neural networks. For node prediction, the output of a T-layer GNN for node v is $\mathbf{h}_v^{(T)}$. For graph prediction tasks an additional readout layer is used: $g^{(T)} = \sum_{v \in V} \mathbf{h}_v^{(T)}$, possibly followed by a fully connected network.

Graph distributions and local structures. In this paper we focus on graph distributions for which the local structure of the graph (formally defined in Sec. 4) depends on the graph size. A well-known distribution family with this property is $G(n, p)$ graphs, also known as Erdős-Rényi. A graph sampled from $G(n, p)$ has n nodes, and edges are drawn i.i.d. with probability p. The mean degree of each node is $n \cdot p$; hence fixing p and increasing n changes the local structure of the graph, specifically the node degrees.

As a second example, we consider the preferential attachment model (Barabási & Albert, 1999). Here, n nodes are drawn sequentially, and each new node is connected to exactly m other nodes, where the probability to connect to other nodes is proportional to their degree. As a result, high degree nodes have a high probability that new nodes will connect to them. Increasing the graph size, causes the maximum degree in the graph to increase, and thus changes its local structure. We also show that, in real datasets, the local structures of small and large graphs differ. This is further discussed in Sec. 7 and Appendix G.5.

3. Overview

3.1. The size-generalization problem

We are given two distributions over graphs P_1, P_2 that contain small and large graphs accordingly, and a task that can be solved for all graph sizes using a GNN. We train a GNN on a training set $\mathcal{S}$ sampled i.i.d. from P_1 and study its performance on P_2. In this paper, we focus on distributions that have a high discrepancy between the local structure of the graphs sampled from P_1 and P_2.

Size generalization is not trivial. Before we proceed with our main results, we argue that even for the simple regression task of counting the number of edges in a graph, which is solvable for all graph sizes by a 1-layer GNN, GNNs do not naturally generalize to new sizes. Specifically, we show that training a 1-layer GNN on a non-diverse dataset reaches a non-generalizing solution with probability 1 over the random initialization. In addition, we show that, in general, the generalizing solution is not the least L1 or L2 norm solution, hence cannot be reached using standard regularization methods. See full derivation in Appendix A.

3.2. Summary of the main argument

This subsection describes the main flow of the next sections in the paper. We explore the following arguments:

(i) d-patterns are a correct notion for studying the expressivity of GNNs. To study size generalization, we introduce a concept named d-patterns, which captures the local structure of a node and its d-step neighbors, as captured by GNNs. This notion is formally defined in Section 4. For example, for graphs without node features, a 1-pattern of a node represents its degree, and its 2-pattern represents its degree and the set of degrees of its immediate neighbors. We argue that d-patterns are a natural abstract concept for studying the expressive power of GNNs: first, we extend a result by (Morris et al., 2019) and prove that d-layer GNNs (with an additional node-level network) can be programmed to output any value on any d-pattern independently. Conversely, as shown in (Morris et al., 2019), GNNs output a constant value when given nodes with the same d-pattern, meaning that the expressive power of GNNs is limited by their values on d-patterns.

(ii) d-pattern discrepancy implies the existence of bad global minima. In Section 5, we focus on the case where graphs in the test distribution contain d-patterns that are not present in the train distribution. In that case, we prove that for any graph task solvable by a GNN, there is a weight assignment that succeeds on training distribution and fails on the test data. In particular, when the training data contains small graphs and the test data contains large graphs, if there is a d-pattern discrepancy between large and small graphs, then there are "bad" global minima that fail to generalize to larger graphs.

(iii) GNNs converge to non-generalizing solutions. In Section 6 we complement these theoretical results with a controlled empirical study that investigates the generalization capabilities of the solutions that GNNs converge to. We show, for several synthetic graph distributions in which we have control over the graph structure, that the generalization of GNNs in practice is correlated with the discrepancy between the local distributions of large and small graphs. Specifically, when the d-patterns in large graphs are not found in small graphs, GNNs tend to converge to a global minimum that succeeds on small graphs and fail on large graphs. This happens even if there is a "good" global minimum that solves the task for all graph sizes. This phenomenon is also prevalent in real datasets as we show in Section 7.

(iv) Size generalization can be improved. Lastly, In Section 7, we discuss two approaches for improving size generalization, motivated by our findings. We first formulate the learning problem as a domain adaptation (DA) problem where the source domain consists of small graphs and the target domain consists of large graphs. We then suggest two learning setups: (1) Training GNNs on a novel self-supervised task aimed at learning meaningful representations for d-patterns from both the target and source domains. (2) A semi-supervised learning setup with a limited number of labeled examples from the target domain. We show that both setups are useful in a series of experiments on synthetic and real data. Notably, training with our new SSL task increases classification accuracy on large graphs in real datasets.

4. GNNs and local graph patterns

We wish to understand theoretically the conditions where a GNN trained on graphs with a small number of nodes can generalize to graphs with a large number of nodes. To answer this question, we first analyze what information is available to each node after a graph is processed by a d-layer GNN. It is easy to see that every node can receive information from its neighbors which are at most d hops away. We note, however, that nodes do not have full information about their d-hop neighborhood. For example, GNNs cannot determine if a triangle is present in a neighborhood of a given node (Chen et al., 2020).

To characterize the information that can be found in each node after a d-layer GNN, we introduce the notion of d-patterns, motivated by the structure of the node descriptors used in the Weisfeiler-Lehman test (Weisfeiler & Lehman, 1968): a graph isomorphism test which was recently shown to have the same representational power as GNNs ((Xu et al., 2018; Morris et al., 2019)).

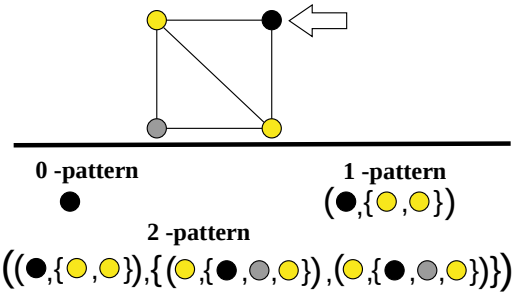


Figure 2. **Top:** A graph with 4 nodes. Each color represent a different feature. **Bottom:** The 0,1 and 2-patterns of the black node.

Definition 4.1 (d-patterns)**.** *Let C be a finite set of node features, and let $G = (V, E)$ be a graph with node feature $c_v \in C$ for every node $v \in V$. We define the **d-pattern** of a node $v \in V$ for $d \geq 0$ recursively: For $d = 0$, the 0-pattern is c_v. For $d > 0$, the d-pattern of v is $p = (p_v, \{(p_{i_1}, m_{p_{i_1}}), \ldots, (p_{i_\ell}, m_{p_{i_\ell}})\})$ iff node v has $(d-1)$-pattern p_v and for every $j \in \{1, \ldots, \ell\}$ the number of neighbors of v with $(d-1)$-pattern p_{i_j} is exactly $m_{p_{i_j}}$. Here, ℓ is the number of distinct neighboring $d-1$ patterns of v.*

In other words, the d-pattern of a node is an encoding of the $(d-1)$-patterns of itself and its neighbors. For example, assume all the nodes in the graphs start with the same node feature. The 1-pattern of each node is its degree. The 2-pattern of each node is for each possible degree $i \in \mathbb{N}$ the number of neighbors with degree i, concatenated with the degree of the current node. In the same manner, the 3-pattern of a node is for each possible 2-pattern, the number of its neighbors with this exact 2-pattern.

Fig. 2. illustrates $0, 1$ and 2-patterns for a graph with three categorical node features, represented by three colors (yellow, grey, and black). For this case, which generalizes the uniform node feature case discussed above, the 0-pattern is the node's categorical feature; 1-patterns count the number of neighbors with a particular feature. The same definition applies to high-order patterns.

We claim that the definition of d-patterns gives an exact characterization to the potential knowledge that a d-layer GNN has on each node. First, Theorem 4.2 is a restatement of Theorem 1 in (Morris et al., 2019) in terms of d-patterns:

Theorem 4.2. *Any function that can be represented by a d-layer GNN is constant on nodes with the same d-patterns.*

The theorem states that any d-layer GNN will output the same result for nodes with the same d-pattern. Thus, we can refer to the output of a GNN on the d-patterns themselves. We stress that these d-patterns only contain a part of the information regarding the d-neighborhood (d hops away from the node), and different neighborhoods could have the same d-patterns. The full proof can be found in Appendix B and follows directly from the analogy between the iterations of the WL algorithm and d-patterns.

Next, the following theorem shows that given a set of d-patterns and the desired output for each such pattern, there is an assignment of weights to a GNN with $d+2$ layers that perfectly fits the output for each pattern.

Theorem 4.3. *Let C be a finite set of node features, P be a finite set of d-patterns on graphs with maximal degree $N \in \mathbb{N}$, and for each pattern $p \in P$ let $y_p \in \mathbb{R}$ be some target label. Then there exists a GNN with $d+2$ layers, width bounded by $\max\left\{(N+1)^d \cdot |C|, 2\sqrt{|P|}\right\}$ and ReLU activation such that for every graph G with nodes $v_1, \ldots, v_n$ and corresponding d-patterns $p_1, \ldots, p_n \subseteq P$, the output of this GNN on v_i is exactly y_{p_i}.*

The full proof is in Appendix B. This theorem strengthens Theorem 2 from (Morris et al., 2019) in two ways: (1) We prove that one can specify the output for every d-pattern while (Morris et al., 2019) show that there is a d-layer GNN that can distinguish all d-patterns; (2) Our network construction is more efficient in terms of width and dependence on the number of d-patterns ($2\sqrt{|P|}$ instead of $|P|$).

We note that the width of the required GNN from the theorem is not very large if d is small, where d represents the depth of the GNN. In practice, shallow GNNs are very commonly used and are empirically successful. The $d+2$ layers in the theorem can be split into d message-passing layers plus 2 fully connected layers that are applied to each node independently. Thm. 4.3 can be readily extended to a vector output for each d-pattern, at the cost of increasing the width of the layers.

Combining Thm. 4.2 and Thm. 4.3 shows that we can independently control the values of d-layer GNNs on the set of d-patterns (possibly with an additional node-wise function) and these values completely determine the GNN's output.

5. "Bad" global minima exist

We now consider *any* graph-prediction task solvable by a d-layer GNN. Assume we have a training distribution of (say, small) graphs and a possibly different test distribution of (say, large) graphs. We show that if the graphs in the test distribution introduce unseen d-patterns, then there exists a $(d+3)$-layer GNN that solves the task on the train distribution and fails on the test distribution. We will consider both graph-level tasks (i.e. predicting a single value for the entire graph, e.g., graph classification) and node-level tasks (i.e. predicting a single value for each node, e.g., node classification).

Theorem 5.1. *Let P_1 and P_2 be finitely supported distributions of graphs. Let P_1^d be the distribution of d-patterns over P_1 and similarly P_2^d for P_2. Assume that any graph in P_2 contains a node with a d-pattern in $P_2^d \setminus P_1^d$. Then, for any graph regression task solvable by a GNN with depth d there exists a GNN with depth at most $d+3$ that perfectly solves the task on P_1 and predicts an answer with arbitrarily large error on all graphs from P_2.*

The proof directly uses the construction from Thm. 4.3, and can be found in Appendix C. The main idea is to leverage the unseen d-patterns from P_2^d to change the output on graphs from P_2.

As an example, consider the task of counting the number of edges in the graph. In this case, there is a simple GNN that generalizes to all graph sizes: the GNN first calculates the node degree for each node using the first message-passing layer and then uses the readout function to sum the node outputs. This results in the output $2|E|$, which can be scaled appropriately. To define a network that outputs wrong answers on large graphs under our assumptions, we can use Thm. 4.3 and make sure that the network outputs the node degree on patterns in P_1^d and some other value on patterns in $P_2^d \setminus P_1^d$.

Note that although we only showed in Thm. 4.3 that the output of GNNs can be chosen for nodes, the value of GNNs on the nodes has a direct effect on graph-level tasks. This happens because of the global readout function used in GNNs, which aggregates the GNN output over all the nodes.

Next, we prove a similar theorem for node tasks. Here, we show a relation between the discrepancy of d-pattern distributions and the error on the large graphs.

Theorem 5.2. *Let P_1 and P_2 be finitely supported distributions on graphs, and let P_1^d be the distribution of d-patterns over P_1 and similarly P_2^d for P_2. For any node prediction task which is solvable by a GNN with depth d and $\epsilon > 0$ there exists a GNN with depth at most $d+2$ that has 0-1 loss (averaged over the nodes) smaller then ϵ on P_1 and 0-1 loss $\Delta(\epsilon)$ on P_2, where $\Delta(\epsilon) = \max_{A:P_1^d(A)<\epsilon} P_2^d(A)$. Here, A is a set of d-patterns, and $P(A)$ is the total probability mass for that set under P.*

This theorem shows that for node prediction tasks, if there is a large discrepancy between the graph distributions (a set of d-patterns with small probability in P_1^d and large probability in P_2^d), then there is a solution that solves the task on P_1, and generalizes badly for P_2. The full proof can be found in Appendix C.

Examples. The above results show that even for simple tasks, GNNs may fail to generalize to unseen sizes, here are two examples. (i) Consider the task of counting the number of edges in a graph. From Thm. 5.1 there is a GNN that successfully outputs the number of edges in graphs with max degree up to N, and fails on graphs with larger max degrees. (ii) Consider some node regression task, when the training set consists of graphs sampled i.i.d from an Erdős-Rényi graph $G(n,p)$, and the test set contains graphs sampled i.i.d from $G(2n,p)$. In this case, a GNN trained on the training set will be trained on graphs with an average degree np, while the test set contains graphs with an average degree $2np$. When n is large, with a very high probability, the train and test set will not have any common d-patterns, for any $d > 0$. Hence, by Thm. 5.2 there is a GNN that solves the task for small graphs and fails on large graphs.

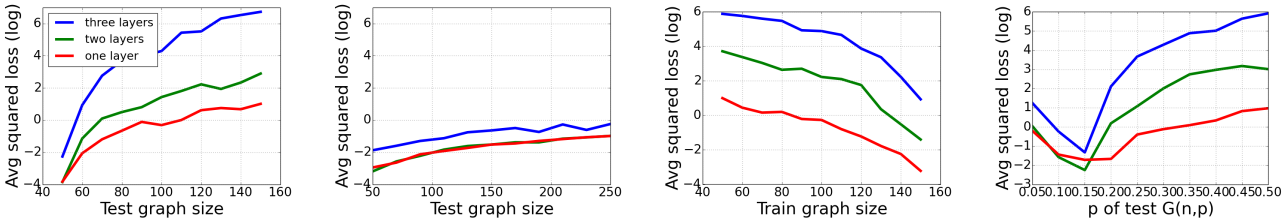
The next section studies the relation between size generalization and local graph structure in controlled experimental settings on synthetic data.

6. A controlled empirical study

The previous section showed that there exist bad global minima that fail to generalize to larger graphs. In this section, we study empirically whether common training procedures lead to bad global minima in practice. Specifically, we demonstrate on several synthetic graph distributions, that reaching bad global minima is tightly connected to the discrepancy of d-pattern distributions between large and small graphs. We identify two main phenomena: **(A)** When there is a large discrepancy between the d-pattern distributions of large and small graphs, GNNs fail to generalize; **(B)** As the discrepancy between these distributions gets smaller, GNNs get better at generalizing to larger graphs.

Tasks. In the following experiments, we use a controlled regression task in a student-teacher setting. In this setting, we sample a "teacher" GNN with random weights (drawn i.i.d from $U([-0.1, 0.1])$), freeze the network, and label each graph in the dataset using the output of the "teacher" network. Our goal is to train a "student" network, which has the same architecture as the "teacher" network, to fit the labels of the teacher network. The advantages of this setting are twofold: (1) *A solution is guaranteed to exist*: We know that there is a weight assignment of the student network which perfectly solves the task for graphs of any size. (2) *Generality*: It covers a diverse set of tasks solvable by GNNs. As the evaluation criterion, we use the squared loss.

Graph distribution. Graphs were drawn from a $G(n,p)$ distribution. This distribution is useful for testing our hypothesis since we can modify the distribution of d-patterns simply by changing either p or n. For example, 1-patterns represent node degrees, and in this model, the average degree of graphs generated from $G(n,p)$ is np. We provide experiments on additional graph distributions like PA in Appendix D.



(a) (b) (c) (d)

Figure 3. The effect of graph size and d-pattern distribution on generalization in $G(n, p)$ graphs in a student-teacher graph regression task. The y-axis represents the squared loss in $\log_{10}$ scale. (a) Bounded training size $n \in [40, 50]$ and varying test size with constant $p = 0.3$ (b) Bounded training training size $n \in [40, 50]$ and varying test size while keeping node degrees constant by changing $p \in [0.15, 0.3]$. (c) Varying train size with constant test size. We train on graphs with n nodes and constant $p = 0.3$. Here, n is drawn uniformly from $[40, x]$ and x varies; test on $n = 150$, $p = 0.3$. (d) Train on n drawn uniformly from $[40, 50]$ and $p = 0.3$ test on $n = 100$ and varying p. See discussion in the text.

Architecture and training protocol. We use a GNN as defined in (Morris et al., 2019) with ReLU activations. The number of GNN layers in the network we use is either $1, 2$ or 3; the width of the teacher network is 32 and of the student network 64, providing more expressive power to the student network. We obtained similar results when testing with a width of 32, the same as the teacher network. We use a summation readout function followed by a two-layer fully connected suffix. We use ADAM with a learning rate of 10^{-3}. We added weight decay (L_2 regularization) with $\lambda = 0.1$. We performed a hyper-parameters search on the learning rate and weight decay and use validation-based early stopping on the source domain (small graphs). The results are averaged over 10 random seeds. We used Pytorch Geometric (Fey & Lenssen, 2019) on NVIDIA DGX-1.

Experiments We conducted four experiments, shown in Figure 3 (a-d). We note that in all the experiments, the loss on the validation set was effectively zero. First, we study the generalization of GNNs by training on a bounded size range $n \in [40, 50]$ and varying the test size in $[50, 150]$.

Figure 3 (a) shows that when p is kept constant while increasing the test graph sizes, size generalization degrades. Indeed, in this case, the underlying d-pattern distribution diverges from the training distribution. In Appendix D we demonstrate that this problem persists to larger graphs with up to 500 nodes.

On the flip side, Figure 3 (b) shows that when p is properly normalized to keep the degree np constant while varying the graph size then we have significantly better generalization to large graphs. In this case, the d-pattern distribution remains similar.

In the next experiment, shown in Figure 3 (c) we keep the test size constant $n = 150$ and vary the training size $n \in [40, x]$ where x varies in $[50, 150]$ and $p = 0.3$ remains constant. In this case we can see that as we train on graph sizes that approach the test graph sizes, the d-pattern discrepancy reduces and generalization improves.

In our last experiment shown in Figure 3 (d), we train on $n \in [40, 50]$ and $p = 0.3$ and test on $G(n, p)$ graphs with $n = 100$ and p varying from 0.05 to 0.5. As mentioned before, the expected node degree of the graphs is np, hence the distribution of d-patterns is most similar to the one observed in the training set when $p = 0.15$. Indeed, this is the value of p where the test loss is minimized.

Conclusions. First, our experiments confirm phenomena **(A-B)**. Another conclusion is that size generalization is more difficult when using deeper networks. This is consistent with our theory since in these cases the pattern discrepancy becomes more severe: for example, 2-patterns divide nodes into significantly more d-pattern classes than 1-patterns. Further results on real datasets appear in Sec. 7.

Additional experiments. in Appendix D, We show that the conclusions above are consistent along different tasks (max clique, edge count, node regression), distributions (PA and point cloud graphs), and architectures (GIN (Xu et al., 2018)). We also tried other activation functions (tanh and sigmoid). Additionally, we experimented with generalization from large to small graphs. Our previous understanding is confirmed by the findings of the present experiment: generalization is better when the training and test sets have similar graph sizes (and similar d-pattern distribution).

7. Towards improving size generalization

The results from the previous sections imply that the problem of size generalization is not only related to the size of the graph in terms of the number of nodes or edges but to the distribution of d-patterns. Based on this observation, we now formulate the size-generalization problem as a domain adaptation (DA) problem. We consider a setting where we are given two distributions over graphs: a source distribution $\mathcal{D}_S$ (say, for small graphs) and a target distribution $\mathcal{D}_T$ (say,

for large graphs). The main idea is to adapt the network to unseen d-patterns appearing in large graphs.

We first consider the *unsupervised* DA setting, where we have access to labeled samples from the source $\mathcal{D}_S$ but the target data from $\mathcal{D}_T$ is unlabeled. Our goal is to infer labels on a test dataset sampled from the target $\mathcal{D}_T$. To this end, we devise a novel SSL task that promotes learning informative representations of unseen d-patterns. We show that this approach improves the size-generalization ability of GNNs.

Second, we consider a *semi-supervised* setup, where we also have access to a small number (e.g., 1-10) of labeled examples from the target $\mathcal{D}_T$. We show that such a setup, when feasible, can lead to equivalent improvement, and benefits from our SSL task as well.

7.1. SSL for DA on graphs

In SSL for DA, a model is trained on unlabeled data to learn a *pretext* task, which is different from the main task at hand. If the pretext task is chosen wisely, the model learns useful representations (Doersch et al., 2015; Gidaris et al., 2018) that can help with the main task. Here, we train the pretext task on both the source and target domains, as was done for images and point clouds (Sun et al., 2019; Achituve et al., 2020). The idea is that the pretext task aligns the representations of the source and target domains leading to better predictions of the main task for target graphs.

Pattern-tree pretext task. We propose a novel pretext task which is motivated by sections 5-6: one of the main causes for bad generalization is unseen d-patterns in the test set. Therefore, we design a pretext task to encourage the network to learn useful representations for these d-patterns.

Our pretext task is a node prediction task in which the output node label is specifically designed to hold important information about the node's d-pattern. For an illustration of a label see Figure 4. The construction of those labels is split into two procedures.

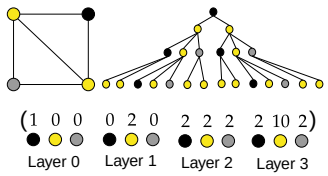


Figure 4. **Top left:** a graph with node features represented by colors. **Top right:** A tree that represents the d-patterns for the black node. **Bottom:** The tree descriptor is a vector with each coordinate containing the number of nodes from each class in each layer of the tree.

First, we construct a tree that fully represents each node's d-pattern. The tree is constructed for a node v in the following way: we start by creating a root node that represents v. We then create nodes for all v's neighbors and connect them to the root. All these nodes hold the features of the nodes they represent in the original graph. We continue to grow the tree recursively up to depth d by adding new nodes that represent the neighbors (in the original graph) of the current leaves in our tree.

This is a standard construction, see e.g., (Xu et al., 2018). For more details about the construction of the pattern tree see Appendix E.

We then calculate a descriptor of the tree that will be used as the SSL output label for each node. The descriptor is a concatenation of histograms of the different node features in each layer of the tree. The network is then trained in a node regression setup with a dedicated SSL head to predict this descriptor.

7.2. Experiments

Baselines. We compare our new pretext task to the following baselines: (1) **Vanilla**: standard training on the source domain; (2) **HomoGNN** (Tang et al., 2020) a homogeneous GNN without the bias term trained on the source domain; (3) **Graph autoencoder (GAE)** pretext task (Kipf & Welling, 2016b); (4) **Node masking (NM)** pretext task from (Hu et al., 2019) where at each training iteration we mask 10% of the node features and the goal is to reconstruct them. In case the graph does not have node features then the task was to predict the degree of the masked nodes. (5) **Node metric learning (NML)**: we use metric learning to learn useful node representations. We use a corruption function that given a graph and corruption parameter $p \in [0, 1]$, replaces $p|E|$ of the edges with random edges, and thus can generate positive ($p = 0.1$) and negative ($p = 0.3$) examples for all nodes of the graph. We train with the triplet loss (Weinberger & Saul, 2009). (6) **Contrastive learning (CL)**: In each iteration, we obtain two similar versions of each graph, which are used to compute a contrastive loss (Qiu et al., 2020; You et al., 2020a) against other graphs. We follow the protocol of (You et al., 2020a), using a corruption function of edge perturbation that randomly adds and removes 5% of the edges in the graph.

Datasets. We use datasets from (Morris et al., 2020) and (Rozemberczki et al., 2020) (Twitch egos and Deezer egos). We selected datasets that have a sufficient number of graphs (more than 1,000) and with a non-trivial split to small and large graphs as detailed in Appendix G.4. In total, we used 7 datasets, 4 from molecular biology (NCI1, NCI109, D&D, Proteins), and 3 from social networks (Twitch ego nets, Deezer ego nets, IMDB-Binary). In all datasets, 50% smallest graphs were assigned to the training set, and the largest 10% of graphs were assigned to the test set. We further split a random 10% of the small graphs as a validation set.

Architecture and training protocol. The setup is the same as in Sec. 6 with a three-layer GNN in all experiments.

DATASETS	DEEZER	IMDB - B	NCI1	NCI109	PROTEINS	TWITCH	DD	AVERAGE
TOTAL-VAR. DISTANCE	1	0.99	0.16	0.16	0.48	1	0.15	-
SMALL GRAPHS	56.5 ± 0.8	63.2 ± 3.3	75.5 ± 1.6	78.4 ± 1.4	75.4 ± 3.1	69.7 ± 0.2	71.1 ± 4.4	70.0%
VANILLA	41.1 ± 6.8	55.9 ± 7.8	65.9 ± 4.3	68.9 ± 3.8	76.0 ± 8.5	60.5 ± 3.6	76.3 ± 3.2	63.5%
HOMO-GNN	40.5 ± 6.6	56.3 ± 7.0	66.0 ± 3.7	68.8 ± 3.2	77.1 ± 10.0	60.8 ± 2.3	**76.8 ± 3.0**	63.8%
NM MTL	**51.6 ± 8.5**	55.6 ± 6.8	49.9 ± 7.8	61.7 ± 5.7	78.8 ± 8.4	49.5 ± 2.8	67.4 ± 5.4	59.2%
NM PT	50.1 ± 7.5	54.9 ± 6.7	51.7 ± 6.6	55.8 ± 5.0	78.2 ± 8.2	48.4 ± 4.0	60.3 ± 15.9	57.1%
GAE MTL	49.4 ± 11.0	55.5 ± 6.0	51.2 ± 9.9	57.6 ± 9.4	79.5 ± 11.7	62.5 ± 5.1	67.8 ± 10.0	60.5%
GAE PT	47.1 ± 10.0	54.1 ± 6.8	58.9 ± 7.6	67.2 ± 5.6	70.5 ± 9.4	53.6 ± 4.7	69 ± 7.1	60.1%
NML MTL	46.4 ± 9.5	54.4 ± 7.0	52.3 ± 6.3	56.2 ± 6.5	78.7 ± 6.8	57.4 ± 4.1	64.7 ± 11.9	58.6%
NML PT	48.4 ± 10.7	53.8 ± 6.1	54.6 ± 6.2	56.1 ± 8.1	76.3 ± 8.0	54.9 ± 4.7	61.4 ± 15.1	57.9%
CL MTL	48.2 ± 10.9	54.6 ± 6.6	52.2 ± 6.8	55.7 ± 5.8	76.6 ± 7.7	59.4 ± 3.5	63.6 ± 15.0	58.6%
CL PT	47.6 ± 9.7	53.6 ± 7.5	57.4 ± 8.1	57.3 ± 6.1	77.6 ± 4.7	53.9 ± 7.1	69.2 ± 5.5	59.5%
PATTERN MTL (OURS)	45.6 ± 8.8	56.8 ± 9.2	60.5 ± 7.5	67.9 ± 7.2	75.8 ± 11.1	61.6 ± 3.5	**76.8 ± 3.0**	63.6%
PATTERN PT (OURS)	44.0 ± 7.7	**61.9 ± 3.2**	**67.8 ± 11.7**	**74.8 ± 5.7**	**84.7 ± 5.1**	**64.5 ± 3.3**	74.9 ± 5.2	**67.5%**

Table 1. Test accuracy of compared methods in 7 binary classification tasks. The Pattern tree method with pretraining achieves the highest accuracy in most tasks and increases the average accuracy from 63% to 67% compared with the second-best method. High variance is due to the domain shift between the source and target domain.

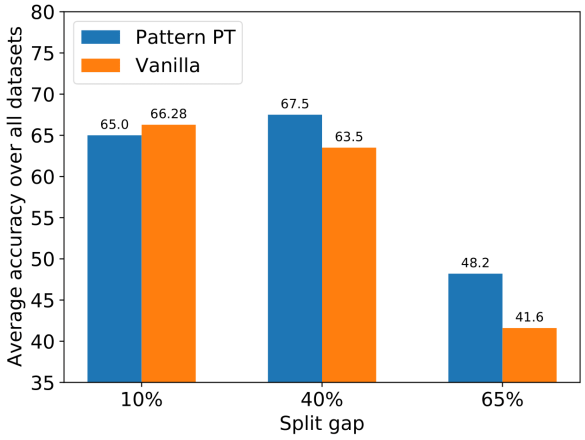


Figure 5. Average accuracy on different size splits in the unsupervised setup for (i) d-pattern pretraining and (ii) no SSL (Vanilla). Accuracy is averaged over all the datasets in table 1.

Given a pretext task we consider two different training procedures: (1) **Multi-task learning (MTL)** (You et al., 2020b); (2) **Pretraining (PT)** (Hu et al., 2019). For MTL we use equal weights for the main and SSL tasks. In the semi-supervised setup, we used equal weights for the source and target data. More details on the training procedures and the losses can be found in Appendix F.

d-pattern distribution in real datasets. In Appendix G.5 we study the discrepancy between the local patterns between small and large graphs on all the datasets mentioned above. The second row of Table 1 summarizes our findings with the total variation (TV) distances between d-pattern distributions of small and large graphs. The difference between these distributions is severe for all social network datasets ($TV \approx 1$), and milder for biological datasets ($TV \in [0.15, 0.48]$).

Next, we will see that a discrepancy between the d-patterns leads to bad generalization and that correctly representing the patterns of the test set improves performance.

Results for unsupervised DA setup. Table 1 compares the effect of using the Pattern-tree pretext task to the baselines described above. The *small graphs* row presents vanilla results on a validation set with small graphs for comparison. The small graph accuracy on 5 out of 7 datasets is larger by 7.3%-15.5% than on large graphs, indicating that the size-generalization problem is indeed prevalent in real datasets.

Pretraining with the d-patterns pretext task outperforms other baselines in 5 out 7 datasets, with an average 4% improved accuracy on all datasets. HOMO-GNN slightly improves over the vanilla while other pretext tasks do not improve average accuracy. Specifically, for the datasets with high discrepancy of local patterns (namely, IMDB, Deezer, Proteins, and Twitch), pretraining with our SSL task improves nicely over vanilla training (by 5.4% on average). Naturally, the accuracy here is lower than SOTA on these datasets because the domain shift makes the problem harder.

Fig. 5 shows two additional experiments, conducted on all datasets using different size splits. First, using a gap of 65% (training on the 30% smallest graphs and testing on the 5% largest graphs), and second, using a gap of 10% (training on the 50% smallest graphs and testing on graphs in the 60-70-percentile). The results are as expected: (1) When training without SSL, larger size gaps hurt more (2) SSL improves over Vanilla training with larger gaps.

Results for semi-supervised DA setup. Figure 6 compares the performance of vanilla training versus pretraining with the pattern-tree pretext task in the semi-supervised setup. As expected, the accuracy monotonically increases with respect to the number of labeled examples in both cases. Still, we would like to highlight the improvement we get by training on only a handful of extra examples. Pretraining with the pretext task yields better results in the case of 0,1,5 labeled examples and comparable results with 10 labeled examples.

Additional experiments We provide additional experi-

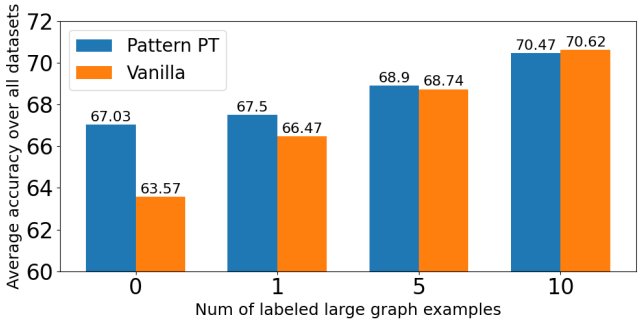


Figure 6. Average classification results in the semi-supervised setup for (i) d-pattern pretraining and (ii) no SSL (Vanilla). Results were averaged over all the datasets in table 1.

ments on the synthetic tasks discussed in Sec. 6 in Appendix G. We show that the pattern-tree pretext task improves generalization in the student-teacher setting (while not solving the edge count or degree prediction tasks). In addition, adding even a single labeled sample from the target distribution significantly improves performance. We additionally tested our SSL task on a combinatorial optimization problem of finding the max clique size in the graph, our SSL improves over vanilla training by a factor of 2, although not completely solving the problem. Also, we tested on several tasks from the "ogbg-molpcba" dataset (see (Hu et al., 2020)), although the results are inconclusive. This is further discussed in Sec. 9.

8. Related work

Size generalization. Several papers observed successful generalization across graph sizes, but the underlying reasons were not investigated (Li et al., 2018; Maron et al., 2018; Luz et al., 2020). More recently, (Veličković et al., 2019) showed that when training GNNs to perform simple graph algorithms step by step they generalize better to graphs of different sizes. Unfortunately, such training procedures cannot be easily applied to general tasks. (Knyazev et al., 2019) studied the relationship between generalization and attention mechanisms. (Bevilacqua et al., 2021) study graph extrapolation using causal modeling. On the more practical side, (Joshi et al., 2019; 2020; Khalil et al., 2017), study the Traveling Salesman Problem (TSP), and show empirically that size generalization on this problem is hard. (Corso et al., 2020) study several multitask learning problems on graphs and evaluate how the performance changes as the size of the graphs change. In another line of work, Tang et al. (2020); Nachmani & Wolf (2020) considered adaptive depth GNNs. In our paper, we focus on the predominant GNN architecture with a fixed number of message-passing layers. Several works also studied size generalization and expressivity when learning set-structured inputs (Zweig & Bruna, 2020; Bueno & Hylton, 2020). In (Santoro et al., 2018) the authors study generalization in abstract reasoning.

Generalization in graph neural networks. Several works studied generalization bounds for certain classes of GNNs (Garg et al., 2020; Puny et al., 2020; Verma & Zhang, 2019; Liao et al., 2020; Du et al., 2019), but did not discuss size generalization. (Sinha et al., 2020) proposed a benchmark for assessing the logical generalization abilities of GNNs.

self-supervised and unsupervised learning on graphs. One of the first papers to propose an unsupervised learning approach for graphs is (Kipf & Welling, 2016b), which resulted in several subsequent works (Park et al., 2019; Salha et al., 2019). (Velickovic et al., 2019) suggested an unsupervised learning approach based on predicting global graph properties from local node descriptors. (Hu et al., 2019) suggested several unsupervised learning tasks that can be used for pretraining. More recently, (Jin et al., 2020; You et al., 2020b) proposed several self-supervised tasks on graphs, such as node masking. These works mainly focused on a single graph learning setup. (You et al., 2020a; Qiu et al., 2020) applied contrastive learning techniques for unsupervised representation learning on graphs. The main difference between our SSL task and contrastive learning is that following our theoretical observation, our SSL task focuses on representing the local structure of each node, rather than a representation that takes into account the entire graph.

9. Conclusion and Discussion

This work is a step towards gaining an understanding of the size-generalization problem in graph neural networks. We showed that for important graph distributions, GNNs do not naturally generalize to larger graphs even on simple tasks. We started by defining d-patterns, a concept that captures the expressivity of GNNs. We then characterized how the failure to generalize depends on d-patterns. Lastly, we suggested two approaches that can improve generalization. Although these approaches are shown to be useful for multiple tasks, there are still some tasks where generalization could not be improved.

A limitation of our approach is that it assumes categorical node features and bidirectional edges with no features. We plan to expand our approach in the future to address these important use cases. As a final note, our characterization of d-patterns, as well as the methods we proposed, can be applied to other cases where generalization is hindered by distribution shifts, and may also be able to improve results in these situations.

References

Achituve, I., Maron, H., and Chechik, G. Self-supervised learning for domain adaptation on point-clouds. *arXiv preprint arXiv:2003.12641*, 2020.

Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

Barabâsi, A.-L., Jeong, H., Néda, Z., Ravasz, E., Schubert, A., and Vicsek, T. Evolution of the social network of scientific collaborations. *Physica A: Statistical mechanics and its applications*, 311(3-4):590–614, 2002.

Bevilacqua, B., Zhou, Y., Murphy, R. L., and Ribeiro, B. On single-environment extrapolations in graph classification and regression tasks, 2021. URL https://openreview.net/forum?id=wXBt-7VM2JE.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Bueno, C. and Hylton, A. G. Limitations for learning from point clouds, 2020. URL https://openreview.net/forum?id=r1x63grFvH.

Capocci, A., Servedio, V. D., Colaiori, F., Buriol, L. S., Donato, D., Leonardi, S., and Caldarelli, G. Preferential attachment in the growth of social networks: The internet encyclopedia wikipedia. *Physical review E*, 74(3):036116, 2006.

Chen, Z., Chen, L., Villar, S., and Bruna, J. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025*, 2020.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets. *arXiv preprint arXiv:2004.05718*, 2020.

Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pp. 1422–1430, 2015.

Du, S. S., Hou, K., Póczos, B., Salakhutdinov, R., Wang, R., and Xu, K. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *arXiv preprint arXiv:1905.13192*, 2019.

Eisenberg, E. and Levanon, E. Y. Preferential attachment in the protein network evolution. *Physical review letters*, 91 (13):138701, 2003.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Garg, V. K., Jegelka, S., and Jaakkola, T. Generalization and representational limits of graph neural networks. *arXiv preprint arXiv:2002.06157*, 2020.

Gidaris, S., Singh, P., and Komodakis, N. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Hermosilla, P., Ritschel, T., Vázquez, P.-P., Vinacua, À., and Ropinski, T. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Transactions on Graphics (TOG)*, 37(6):1–12, 2018.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

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. *arXiv preprint arXiv:2005.00687*, 2020.

Jin, W., Derr, T., Liu, H., Wang, Y., Wang, S., Liu, Z., and Tang, J. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141*, 2020.

Joshi, C. K., Laurent, T., and Bresson, X. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

Joshi, C. K., Cappart, Q., Rousseau, L.-M., Laurent, T., and Bresson, X. Learning tsp requires rethinking generalization. *arXiv preprint arXiv:2006.07054*, 2020.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs. In *Advances in neural information processing systems*, pp. 6348–6358, 2017.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Knyazev, B., Taylor, G. W., and Amer, M. Understanding attention and generalization in graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 4202–4212, 2019.

Li, Z., Chen, Q., and Koltun, V. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems*, pp. 539–548, 2018.

Liao, R., Urtasun, R., and Zemel, R. A pac-bayesian approach to generalization bounds for graph neural networks. *arXiv preprint arXiv:2012.07690*, 2020.

Light, S., Kraulis, P., and Elofsson, A. Preferential attachment in the evolution of metabolic networks. *Bmc Genomics*, 6(1):1–11, 2005.

Luz, I., Galun, M., Maron, H., Basri, R., and Yavneh, I. Learning algebraic multigrid using graph neural networks. *arXiv preprint arXiv:2003.05744*, 2020.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Morris, C. and Mutzel, P. Towards a practical k-dimensional weisfeiler-leman algorithm. *arXiv preprint arXiv:1904.01543*, 2019.

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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

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.

Nachmani, E. and Wolf, L. Molecule property prediction and classification with graph hypernetworks. *arXiv preprint arXiv:2002.00240*, 2020.

Park, J., Lee, M., Chang, H. J., Lee, K., and Choi, J. Y. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6519–6528, 2019.

Puny, O., Ben-Hamu, H., and Lipman, Y. From graph low-rank global attention to 2-fwl approximation. *arXiv preprint arXiv:2006.07846*, 2020.

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., and Tang, J. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1150–1160, 2020.

Rozemberczki, B., Kiss, O., and Sarkar, R. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, pp. 3125–3132. ACM, 2020.

Salha, G., Hennequin, R., and Vazirgiannis, M. Keep it simple: Graph autoencoders without graph convolutional networks. *arXiv preprint arXiv:1910.00942*, 2019.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. W. Learning to simulate complex physics with graph networks. *arXiv preprint arXiv:2002.09405*, 2020.

Santoro, A., Hill, F., Barrett, D., Morcos, A., and Lillicrap, T. Measuring abstract reasoning in neural networks. In *International Conference on Machine Learning*, pp. 4477–4486, 2018.

Sinha, K., Sodhani, S., Pineau, J., and Hamilton, W. L. Evaluating logical generalization in graph neural networks. *arXiv preprint arXiv:2003.06560*, 2020.

Sun, Y., Tzeng, E., Darrell, T., and Efros, A. A. Unsupervised domain adaptation through self-supervision. *arXiv preprint arXiv:1909.11825*, 2019.

Tang, H., Huang, Z., Gu, J., Lu, B.-L., and Su, H. Towards scale-invariant graph-related problem solving by iterative homogeneous graph neural networks. *ICML 2020 Workshop on Graph Neural Networks & Beyond*, 2020.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. In *ICLR (Poster)*, 2019.

Veličković, P., Ying, R., Padovano, M., Hadsell, R., and Blundell, C. Neural execution of graph algorithms. *arXiv preprint arXiv:1910.10593*, 2019.

Verma, S. and Zhang, Z.-L. Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1539–1548, 2019.

Weinberger, K. Q. and Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(2), 2009.

Weisfeiler, B. and Lehman, A. A. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9): 12–16, 1968.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Xu, K., Li, J., Zhang, M., Du, S. S., Kawarabayashi, K.-i., and Jegelka, S. How neural networks extrapolate: From feedforward to graph neural networks. *arXiv preprint arXiv:2009.11848*, 2020.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33, 2020a.

You, Y., Chen, T., Wang, Z., and Shen, Y. When does self-supervision help graph convolutional networks? *arXiv preprint arXiv:2006.09136*, 2020b.

Yun, C., Sra, S., and Jadbabaie, A. Small relu networks are powerful memorizers: a tight analysis of memorization capacity. In *Advances in Neural Information Processing Systems*, pp. 15558–15569, 2019.

Zweig, A. and Bruna, J. A functional perspective on learning symmetric functions with neural networks. *arXiv preprint arXiv:2008.06952*, 2020.