Fine-Tuning Graph Neural Networks by Preserving Graph Generative Patterns

Yifei Sun[1], **Qi Zhu**[2], **Yang Yang**[1*], **Chunping Wang**[3], **Tianyu Fan**[1], **Jiajun Zhu**[1], **Lei Chen**[3]

[1]Zhejiang University, Hangzhou, China
[2]University of Illinois Urbana-Champaign, USA
[3]FinVolution Group, Shanghai, China
{yifeisun, yangya, fantianyu, junnian}@zju.edu.cn, qiz3@illinois.edu, {wangchunping02, chenlei04}@xinye.com

Abstract

Recently, the paradigm of pre-training and fine-tuning graph neural networks has been intensively studied and applied in a wide range of graph mining tasks. Its success is generally attributed to the structural consistency between pre-training and downstream datasets, which, however, does not hold in many real-world scenarios. Existing works have shown that the *structural divergence* between pre-training and downstream graphs significantly limits the transferability when using the vanilla fine-tuning strategy. This divergence leads to model overfitting on pre-training graphs and causes difficulties in capturing the structural properties of the downstream graphs. In this paper, we identify the fundamental cause of structural divergence as the discrepancy of *generative patterns* between the pre-training and downstream graphs. Furthermore, we propose **G-TUNING** to preserve the generative patterns of downstream graphs. Given a downstream graph G, the core idea is to tune the pre-trained GNN so that it can reconstruct the *generative patterns* of G, the graphon W. However, the exact reconstruction of a graphon is known to be computationally expensive. To overcome this challenge, we provide a theoretical analysis that establishes the existence of a set of alternative graphons called graphon bases for any given graphon. By utilizing a linear combination of these graphon bases, we can efficiently approximate W. This theoretical finding forms the basis of our model, as it enables effective learning of the graphon bases and their associated coefficients. Compared with existing algorithms, **G-TUNING** demonstrates consistent performance improvement in 7 in-domain and 7 out-of-domain transfer learning experiments.

1 Introduction

The development of graph neural networks (GNNs) has revolutionized many tasks of various domains in recent years. However, labeled data is extremely scarce due to the time-consuming and laborious labeling process. To address this obstacle, the "pre-train and fine-tune" paradigm has made substantial progress (Xia et al. 2022; Li, Zhao, and Zeng 2022; Jiao et al. 2023) and attracted considerable research interests. Specifically, this paradigm involves pre-training a model on a large-scale graph dataset, followed by fine-tuning its parameters on downstream graphs by specific tasks.

[*]Corresponding author.

The success of "pre-train and fine-tune" paradigm is generally attributed to the structural consistency between pre-training and downstream graphs (Hu et al. 2020b,a; Qiu et al. 2020; Sun et al. 2021; Jiarong et al. 2023). However, in real-world scenarios, structural patterns vary dramatically across different graphs, and patterns in the downstream graphs may not be readily available in the pre-training graph. In the context of molecular graphs, a well-known out-of-distribution problem arises when the training and testing graphs originate from different environments, characterized by variations in *size* or *scaffold*. As a consequence, the downstream molecular graph data often encompasses numerous novel substructures that has not been encountered during training. Hence, structural consistency does not always hold. Fig 1(a) shows that while structural consistency (shown in orange) between the pre-training and downstream dataset A ensures the promotion of performance, the *structural divergence* (shown in green) causes a degradation of performance when fine-tuned on downstream dataset B. In some cases, it can even lead to worse results than those obtained without pre-training.

In light of this, we are intrigued by the relationship between structural divergence and the extent of performance improvements on downstream graphs. Specifically, *graphon* is a well-known non-parametric function on graph that has been proved to effectively describe the generative mechanism of graphs (*i.e., generative patterns*) (Lovász 2012). In Fig 1(b), we calculate the Gromov-Wasserstain (GW) discrepancy (a distance metric between geometry objects) of graphons between different pre-training and one test graph and report the corresponding performance on the same downstream graph. Interestingly, as the difference between graphons of pre-training dataset and that of downstream dataset increases, the performance improvement diminishes. To further validate this, we compute the Pearson CC (Correlation Coefficient) between other representative graph measurements (e.g., density, transitivity, *etc.*) and the performance improvement. As Fig 1(c) suggests, most of them cannot reflect the degree of performance improvement, and only the graphon discrepancy is consistently negatively correlated with the degree of performance promotion. Hence, we attribute the subpar performance of fine-tuning to the disparity in the *generative patterns* between the pre-training and fine-tuning graphs. Nevertheless, fine-tuning with respect to *generative patterns* poses significant challenges: (1) the structural information

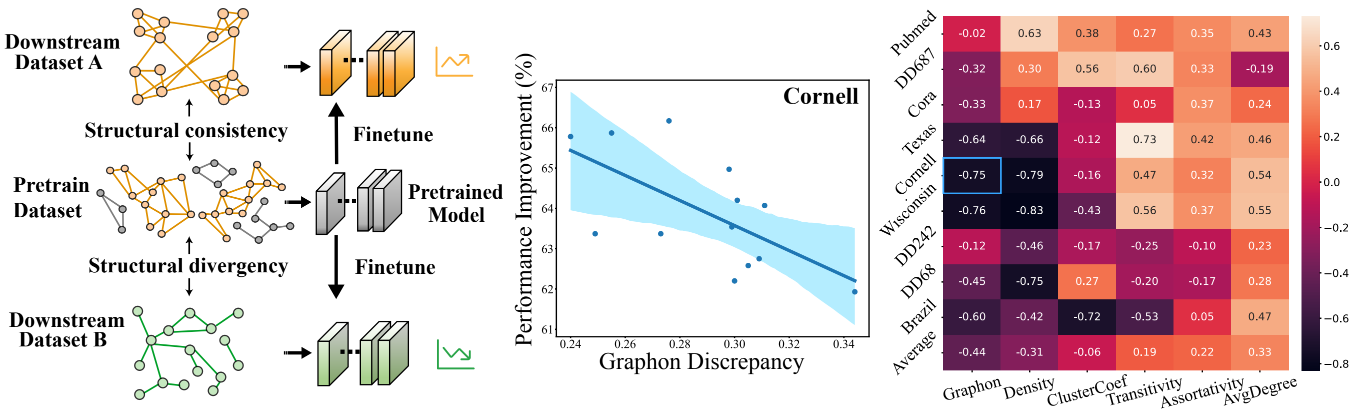

Figure 1: The sketch and the observations of the structural divergence. (a) shows the performance under different scenarios. (b) shows that the larger the graphon Gromov-Wasserstain (GW) discrepancy between different pre-train datasets (shown in App § A.4) and downstream dataset Cornell is, the less the performance (%) is promoted. (c) The pearson correlation between: (Horizontal) discrepancy of different graph measurements and graphon between pre-training and downstream datasets, (Vertical) the performance promotion for different downstream datasets when using different pre-train datasets.

of the pre-training may not be accessible during fine-tuning, and (2) effectively representing intricate semantics within these *generative patterns*, such as graphons, requires careful design considerations.

In this paper, we aim to address these challenges by proposing a fine-tuning strategy, **G-TUNING**, which is agnostic to pre-training data and algorithms. Specifically, it performs graphon reconstruction of the downstream graphs during fine-tuning. In order to enable efficient reconstruction, we provide a theoretical result (Theorem 1) that given a graphon W, it's possible to find a set of other graphons, called graphon bases, whose linear combination can closely approximate W. Then, we develop a graphon decoder that transforms the embeddings from the pre-trained model into a set of coefficients. These coefficients are combined with the structure-aware learnable bases to form the reconstructed graphon. To ensure the fidelity of the reconstructed graphon, we introduce a GW-discrepancy based loss, which minimizes the distance between the approximated graphon and an oracle graphon (Xu et al. 2021). Furthermore, by optimizing our proposed **G-TUNING**, we obtain provable results regarding the discriminative subgraphs relevant to the task (Theorem 2).

The main contributions of our work are as follows:[1]

- We identify the generative patterns of downstream graphs as a crucial step in bridging the gap between pre-training and fine-tuning.
- Building upon our theoretical results, we design the model architecture, **G-TUNING** to efficiently reconstruct graphon as generative patterns with rigorous generalization results.
- Empirically, our method shows consistent performance improvement in 7 in-domain and 7 out-of-domain transfer learning datasets over the best baseline.

[1]Supplement materials: https://github.com/zjunet/G-Tuning

2 Preliminaries

Notations. Let $\mathcal{G} = (V, A, X)$ denote a graph, where V is the node set, $A \in \{0,1\}^{|V|\times|V|}$ is the adjacency matrix and $X \in \mathbb{R}^{|V|\times d}$ is the node feature matrix where d is the dimension of feature. $\mathcal{G}_s$ and $\mathcal{G}_t$ denotes a pre-training graph and a downstream graph respectively. The classic and commonly used pre-training paradigm is to first pre-train the backbone Φ on abundant unlabeled graphs by a self-supervised task with the loss as $\mathcal{L}_{\text{SSL}}$. Then the pre-trained Φ is employed to fine-tuning on labeled downstream graphs. The embedding $H \in \mathbb{R}^{|V|\times d}$ with the hidden dimension d encoded by Φ is further input to a randomly initialized task-specific shallow model f_ϕ. The goal of fine-tune is to adapt both Φ and f_ϕ for the downstream task with the loss $\mathcal{L}_{\text{task}}$ and the label Y. The pre-training setup has two variations: one is the in-domain setting, where $\mathcal{G}_s$ and $\mathcal{G}_t$ come from the same domain, and the other is the out-of-domain setting, where $\mathcal{G}_s$ and $\mathcal{G}_t$ originate from different domains. In the latter case, the structural divergence between $\mathcal{G}_s$ and $\mathcal{G}_t$ is greater, posing greater challenges for fine-tuning.

Definition 1 (Graph generative patterns). *Graph generative patterns are data distributions parameterized by Θ where the observed graphs $\{G_1, ..., G_n\}$ are sampled from, $G_i \sim P(G; \Theta)$.*

According to our definition, Θ can be traditional graph generative model such as Erdos-Rényi (Erdős and Rényi 1959), stochastic block model (Airoldi et al. 2008), forest-fire graph (Leskovec, Kleinberg, and Faloutsos 2007) and *etc.* Besides, Θ can also be any deep generative model like GraphRNN (You et al. 2018). In this paper, we propose a theoretically sound fine-tuning framework by preserving the graph generative pattern of the downstream graphs.

Graphon. A graphon (Airoldi, Costa, and Chan 2013), short for "graph function", can be interpreted as a generalization of a graph with an uncountable number of nodes or a graph generative model or, more important for this work, a mathematical object representing Θ from graph generative patterns $P(G;\Theta)$. Formally, a graphon is a continuous and symmetric function $W:[0,1]^2 \rightarrow [0,1]$. Given two points $u_i, u_j \in [0,1]$ as "nodes", $W(i,j) \in [0,1]$ indicates the probability of them forming an edge. The main idea of graphon is that when we extract subgraphs from the observed graph, the structure of these subgraphs becomes increasingly similar to that of the observed one as we increase the size of subgraphs. The structure then converges in some sense to a limit object, graphon. The convergence is defined via the convergence of homomorphism densities. Homomorphism density $t(F,G)$ is used to measure the relative frequency that homomorphism of graph F appears in graph G: $t(F,G)=\frac{|\operatorname{hom}(F,G)|}{|V_G|^{|V_H|}}$, which can be seen as the probability that a random mapping of vertices from F to G is a homomorphism. Thus, the convergence can be formalized as $\lim_{n\rightarrow\infty} t(F,G_n)=t(F,W)$. When used as the graph generative patterns, the adjacency matrix A of graph $\mathcal{G}$ with N nodes are sampled from $P(G;W)$ as follows,

$$v \sim \mathbb{U}(0,1), v \in V; A_{ij} \sim \text{Ber}(W(v_i,v_j)), \forall i,j \in [N]. \tag{1}$$

where $\text{Ber}(\cdot)$ is the Bernoulli distribution. Since there is no available closed-form expression of graphon, existing works mainly employ a two-dimensional step function, which can be seen as a matrix, to represent a graphon (Xu et al. 2021; Han et al. 2022). In fact, the weak regularity lemma of graphon (Lovász 2012) indicates that an arbitrary graphon can be approximated well by a two-dimensional step function. Hence, we follow the above mentioned works to employ a step function $W \in [0,1]^{D\times D}$ to represent a graphon, where D is a hyper-parameter.

3 Related Work

Fine-tuning strategies. Designing fine-tuning strategies first attracts attention in computer vision (CV), which can be categorized into model parameter regularization and feature regularization. L2_SP(Xuhong, Grandvalet, and Davoine 2018) uses L^2 distance to constrain the parameters around pre-trained ones. StochNorm (Kou et al. 2020) replaces BN(batch normalization) layers in pre-trained model with their StochNorm layers. DELTA (Li et al. 2019) selects features with channel-wise attention to constrain. BSS (Chen et al. 2019) penalizes small eigenvalues of features to prevent negative transfer. However, there is only one work focusing on promoting performance of downstream task during fine-tuning phase specially for the graph structured data. GTOT-Tuning (Zhang et al. 2022) presents an optimal transport-based feature regularization, which achieves node-level transport through graph structure. Moreover, the gap between pre-train and fine-tune is also noted in L2P (Lu et al. 2021) and AUX-TS (Han et al. 2021). Specifically, L2P leverages meta-learning to adjust tasks during pre-training stage. AUX-TS adaptively selects and combines auxiliary tasks with the target task in fine-tuning stage, which means only one pre-train task is not enough. However, they both require to use the same dataset for both pre-train and fine-tune, and to insert auxiliary tasks in pre-training phase , which indicates they are not generally applicable. Unlike them, we focus on the fine-tuning phase and propose that preserving generative patterns of downstream graphs during fine-tuning is the key to mine knowledge from pre-trained GNNs without altering the pre-training process.

Graphon. Graphon has been studied intensively as a mathematical object (Borgs et al. 2008, 2012; Lovász and Szegedy 2006; Lovász 2012) and been applied broadly, like graph signal processing (Ruiz, Chamon, and Ribeiro 2020b, 2021), game theory (Parise and Ozdaglar 2019), network science (Avella-Medina et al. 2018; Vizuete, Garin, and Frasca 2021). Moreover, G-mixup (Han et al. 2022) is proposed to conduct data augmentation for graph classification since a graphon can serve as a graph generator. From the another perspective of being the graph limit, (Ruiz, Chamon, and Ribeiro 2020a) leverage graphon to analyse the transferability of GNNs. Graphon, as limit of graphs, forms a natural method to describe graphs and encapsulates the generative patterns (Borgs and Chayes 2017). Thus, we incorporate the graphon into the fine-tuning stage to preserve the generate patterns of downstream graphs. In App § A.1, we provide more detailed related work about graph pre-training and graphon.

4 G-Tuning

4.1 Framework Overview

Similar to our seminal results in Fig 1, recent research (Zhu et al. 2021) has started to analyze the transfer performance of a pre-trained GNN *w.r.t.* discrepancy between pre-train and fine-tune graphs. However, the pre-training graphs are typically not accessible during the fine-tuning phase. Thus, **G-Tuning** aims to adapt the pre-trained GNN to the fine-tuning graphs by preserving the generative patterns. During fine-tuning, a pre-trained GNN Φ obtains latent representations H for downstream graphs $\mathcal{G}_t=\{G_1,...,G_n\}$ and feeds them into task-specific layers f_ϕ to train with the fine-tuning labels Y. For a specific graph $G_i(A,X)$, the pre-trained node embeddings H are obtained by pre-trained model Φ:

$$\mathcal{L}_{\text{task}}=\mathcal{L}_{\text{CE}}(f_\phi(H),Y), \quad H=\Phi(A,X), \tag{2}$$

where Φ can be any pre-training backbone model, $\mathcal{L}_{\text{CE}}$ is the cross entropy classification loss and f_ϕ is a shallow neural network $f_\phi: H \rightarrow \hat{Y}$.

However, the vanilla strategy may fail to improve fine-tuning performance due to the large discrepancy between pre-training and fine-tuning graphs, namely *negative transfer*. To alleviate this, we propose to enable the pre-trained GNN Φ to preserve the generative patterns of the downstream graphs $\mathcal{G}_t$ by reconstructing their graphons W (overall workflow in Fig 2). However, the pre-trained model is inherently biased towards the generative patterns of the pre-training graphs. Consequently, at the begining of the fine-tuning, the embeddings H also contain the bias from the pre-training data. Thus, we need both the H and graph structure A of downstream graph to reconstruct the graphon. Specifically, we design a graphon reconstruction module Ω to reconstruct the graphon $\hat{W}$ of each downstream graph $G_i(A,X) \in \mathcal{G}_t$:

$$\hat{W}=\Omega(A,H), \tag{3}$$

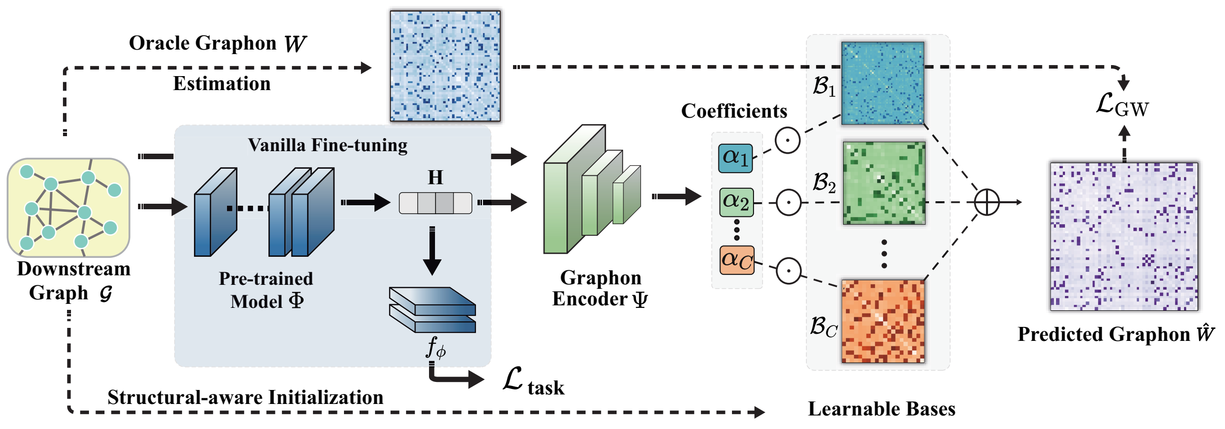

Figure 2: Overall workflow of **G-Tuning**. Each basis $\mathcal{B}_i$ and predicted graphon $\hat{W}$ is produced during actual training.

However, the computation of oracle graphons W of real-world graphs in continuous function form itself is intractable (Han et al. 2022). Thus, the graphon reconstruction module Ω approximates an estimated oracle graphon (*i.e.* $W \in [0,1]^{D\times D}$) for each downstream graph through $\mathcal{L}_{\text{aux}}$, D is the size of oracle graphon. Finally, in the framework of **G-Tuning** (Fig 2), we leverage both the downstream task loss and our reconstruction loss to optimize the parameters of pre-trained GNN encoder Φ, the layer f_ϕ and graphon reconstruction module Ω as follows,

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\textbf{G-Tuning}}(W, \hat{W}). \tag{4}$$

where λ is a hyper-parameter of **G-Tuning**.

4.2 Approximating Oracle Graphon

In this section, we first discuss the calculation of the underlying graphon, denoted as W, given downstream graphs $\mathcal{G}_t$. Then we introduced proposed algorithm to reconstruct this graphon as an auxiliary loss during the fine-tuning process.

Extensive research has been conducted on the methods for estimating graphons (Airoldi, Costa, and Chan 2013; Chatterjee 2015; Pensky 2019; Ruiz, Chamon, and Ribeiro 2020a) from observed graphs. In this paper, the oracle graphon W used in Eq 4 can be estimated via structured Gromov-Wasserstein barycenters (SGB) (Xu et al. 2021). Suppose there are N observed graphs $\{G_n\}_{n=1}^N$ and their adjacency matrices $\{A_n\}_{n=1}^N$, at each step t,

$$\min_{\mathbf{T}_n \in \Pi(\mu, \mu_W)} d^2_{gw,2}\left(W_t, A_n\right), \tag{5}$$

$$W_{t+1} = \frac{1}{\mu_W \mu_W^\top} \sum_{n=1}^{N} \mathbf{T_n}^\top A_n \mathbf{T_n}, \tag{6}$$

where $W_{t+1} \in [0,1]^{D\times D}$ is calculated barycenters with the optimal transportation plan $\{\mathbf{T_n}\}_{n=1}^N$ of 2-order Gromov-Wasserstein distance $d^2_{\text{gw},2}$. The probability measures μ_W is estimated by merging and sorting the normalized node degrees in $\{A_n\}_{n=1}^N$. Further details regarding the calculations of $d^2_{\text{gw},2}$ and the iterative procedure for estimating the oracle graphon can be found in App § A.3.

After obtaining the oracle graphon $W \in [0,1]^{D\times D}$ from the downstream graphs and the reconstructed graphon $\hat{W} \in [0,1]^{M\times M}$ from the reconstruction module Ω, we also adopt the Gromov-Wasserstein distance to measure the distance between the two unaligned graphons. Formally, our p-order GW distance is calculated as:

$$\mathcal{L}_{\textbf{G-Tuning}}(W, \hat{W}) = \min_{T\in\Gamma} \sum_{i,j,k,l} (W_{i,k} - \hat{W}_{j,l})^p T_{i,j} T_{k,l}, \tag{7}$$

where Γ is the set of transportation plans that satisfy $\Gamma = \{T \in \mathbb{R}_+^{D\times M} | T1_M = 1_D, T^\intercal 1_D = 1_M\}$. In each epoch, an optimal T minimizes the transportation cost between two graphons $(W, \hat{W})$. Fixing T, the graphon reconstruction module Ω is optimized to minimize the GW discrepancy. We will introduce the model design of Ψ in the next section for scalable training and inference.

4.3 Efficient Optimization of $\mathcal{L}_{\textbf{G-Tuning}}$

A straightforward way to approximate the graphon is to learn a mapping function graph structure A and node embedding H to the target W. For example, we can simply define the graphon reconstruction module Ω in Eq 3 as a GNN or a shallow MLP. Suppose we have M graphs and each graph has at most $|V|$ nodes, brute-force graphon reconstruction from the pre-trained node embeddings $H \in \mathbb{R}^{|V|\times d}$ requires a large number of parameters, e.g., $\Psi : \mathbb{R}^{|V|\times d} \to \mathbb{R}^{M\times M}$. Additionally, properties such as permutation-variance of graphons are not guaranteed without a carefully designed architecture.

To this end, we first establish a theorem for graphon decomposition and utilize it for efficient graphon approximation. Specifically, we propose that any graphon can be reconstructed by a linear combination of graphon bases $\mathcal{B}_k \in \boldsymbol{\mathcal{B}}$.

Theorem 1. *$\forall\ W(x,y) \in \mathcal{W}_{C+1}$, there exists C graphon bases $\mathcal{B}_k(x,y)$ that satisfies $W(x,y) = \sum_{k=1}^{C} \alpha_k \mathcal{B}_k(x,y) + R_{C+1}(x,y)$, where $\alpha_i \in \mathbb{R}$ and $R_{C+1}(x,y)$ is the remainder of order $C+1$.*

$$W(x,y) = \quad \textstyle\sum_{k=0}^{C} \alpha_k \mathcal{B}_k(x,y) + R_{C+1}. \tag{8}$$

For every $W \in \mathcal{W}_{C+1}$, it has continuous partial derivatives of order $0, ..., C+1$ at any point, so we call apply Taylor expansion at (x, y). See full proof at App § A.2. Then we set the graphon basis, denoted as $\mathcal{B}_k : [0, 1]^2 \rightarrow [0, 1]$, to keep the same size as the graphon $\hat{W}$, i.e., $\mathcal{B}_k \in [0, 1]^{M \times M}$. According to the above theorem, instead of direct graphon reconstruction, we can estimate the bases $\boldsymbol{\mathcal{B}}$ and coefficients $\boldsymbol{\alpha}$. Each coefficient α_k reflects how close the corresponding graphon basis $\mathcal{B}_k$ is to the graphon being approximated. To summarize, we devise our graphon reconstruction module Ω as another GNN to transform the encoded node representation H and graph structure A into coefficients $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_C\}$:

$$\boldsymbol{\alpha} = \Psi(A, H), \tag{9}$$

where $\Psi : (A, H) \rightarrow \mathbb{R}_+^C$ maps the embeddings H from Eq 2 into C coefficients of graphon bases with $\sum \alpha_i = 1$. **G-Tuning** also learns a set of graphon bases $\mathcal{B}_i \in \mathbb{R}^{K \times K}, \mathcal{B}_i \in \boldsymbol{\mathcal{B}}$ to get the optimal bases to reconstruct the oracle graphon W. Overall, the amount of the parameters can be greatly reduced from $\mathcal{O}(|V|hM^2)$ to $\mathcal{O}(C \cdot (M^2 + |V|d))$ since $C \ll \min\{M^2, |V|d\}$. It is worth noting that bases $\boldsymbol{\mathcal{B}}$ are graphons related with the structure $\{A_n\}_{n=1}^N$ of downstream graphs. Therefore, we initialize each basis $\mathcal{B}_k$ as follows: (1) for each run we randomly select a downstream graph $G_i \in \mathcal{G}_t$ and its adjacency matrix A_i; (2) sort the adjacency matrix in descending order of its node degrees; (3) randomly draw a graphon with the size M from the sorted A_i as the initialized b_k. Moreover, we constrain the learbable bases between $[0, 1]$ by setting $\mathcal{B}_k = \sigma(b_k)$ where σ is Sigmoid function. Finally, as the Theorem 1 implies, the approximated graphon $\hat{W}$ can be aggregated as,

$$\hat{W} = \sum_{k=0}^{C} \boldsymbol{\alpha}_k \boldsymbol{\mathcal{B}}_k. \tag{10}$$

Moreover, as the theory indicates, the number of bases corresponds to the order of Taylor expansion (*i.e.*, the more accurate the approximation). In **G-Tuning**, we have two major hyper-parameters: the number of learnable bases C and graphon size M. The overall learning process of **G-Tuning** can be found in Algorithm 1 from App § A.3.

Complexity Analysis. We now analyze the additional time complexity of **G-Tuning** besides vanilla tuning. Suppose $|V|$ and $|E|$ are the average number of nodes and edges, d is the hidden dimension and C is number of graphon bases. The total time complexity of **G-Tuning** includes two parts: (i) the graphon decoder costs $\mathcal{O}(CM^2 + |V|d)$; (ii) the oracle graphon estimation costs $\mathcal{O}(|E|D + |V|D^2)$ (Xu et al. 2021), where D is the size of oracle graphon. Thus, the overall additional time complexity is $\mathcal{O}(|E|D + |V|D^2 + CM^2 + |V|d)$, which is the same magnitudes with the vanilla tuning process of $\mathcal{O}(|E|d + |V|d)$ assuming $M, D \ll |V|$.

4.4 Theoretical Analysis

In this section, we further illustrate the ability of **G-Tuning** to capture the discriminative subgraphs present in the downstream graph $\mathcal{G}_t$. We first give the definition of the discriminative subgraph.

Definition 2 (Discriminative subgraph)**.** *A discriminative subgraph F_G of graph G with a label Y is the subgraph $F \subseteq G$ that minimize the information from G and maximize the information to the label Y formulated as the optimization:*

$$F_G = \underset{F \subseteq G}{\arg\max}\, [I(Y; F) - \beta I(G; F)], \tag{11}$$

where $I(\cdot; \cdot)$ denotes the mutual information and the positive number β operate as a tradeoff parameter.

According to the definition, discriminative subgraphs are the minimal subsets of subgraphs that determine the label of a graph. For example, in the case of a molecular graph, the benzene ring is a discriminative subgraph that distinguishes benzene. Therefore, downstream tasks can benefit from the ability to preserve such discriminative subgraphs. We provide a theoretical insight below that **G-Tuning** is capable of preserving discriminative subgraphs while reconstructing graphons of downstream graphs.

Theorem 2. *Given an arbitrary graph G, the oracle graphon W_G, the predicted graphon $\hat{W}_G$, and a discriminative subgraph F_G, the upper bound of the difference between the homomorphism density of F_G in the oracle graphon W_G and that of the predicted graphon $\hat{W}_G$ is decribed as*

$$|t(F_G, \hat{W}_G) - t(F_G, W_G)| \leq \frac{\mathrm{e}(F_G)}{C} ||R_{C+1}||_\infty, \tag{12}$$

where $\mathrm{e}(F)$ is the number of nodes in subgraph F and R_{C+1} is the remainder in Theorem 1.

See detailed proof at App § A.2. Assuming oracle graphon captures the discriminative subgraphs at $t(F_G, W_G)$, **G-Tuning** is optimized to preserve these discriminative subgraphs during fine-tuning. Moreover, we discuss the generalization bound of **G-Tuning** in App § A.2.

5 Experiments

In this section, we answer the following questions:

Q1. (Effectiveness) Does **G-Tuning** improve the performance of fine-tuning?

Q2. (Transferability) Can **G-Tuning** enable the better transferability than baselines?

Q3. (Integrity) How does each component of **G-Tuning** contribute to the performance?

Q4. (Efficiency) Can **G-Tuning** improve the performance of fine-tuning at an acceptable time consumption?

Baselines. We implement several representative baselines in computer vision that were originally designed for CNNs, including StochNorm (Kou et al. 2020), DELTA and the version with fixed attention coefficients (Feature-Map) (Li et al. 2019), L2_SP(Xuhong, Grandvalet, and Davoine 2018) and BSS(Chen et al. 2019). To the best of our knowledge, there is only one baseline dedicated to improving the fine-tuning of GNNs, which is agnostic to the pre-training strategy, namely GTOT-Tuning (Zhang et al. 2022). To validate the effectiveness of reconstructing graphons, we introduce VGAE-Tuning for comparison, which employs VGAE (Kipf and Welling 2016) as the auxiliary loss to reconstruct the adjacency matrices of downstream graphs. We reproduce the baselines based on the code released by the authors and set the hyperparameters according to their released code and settings depicted in their papers. More details can be found in App §A.4.

	BBBP	Tox21	Toxcast	ClinTox	MUV	HIV	BACE	Rank
Supervised	65.84±4.51	74.02±0.89	61.45±0.64	58.06±4.42	71.81±2.56	75.31±1.93	70.13±5.41	10.0
Vanilla Tuning	68.99±3.13	75.39±0.94	63.33±0.74	65.92±3.78	75.78±1.73	78.21±0.51	79.22±1.20	5.4
StochNorm	69.27±1.67	74.97±0.77	62.69±0.69	65.53±4.25	76.05±1.61	77.58±0.84	81.48±2.10	6.7
Feature Map	60.42±0.78	70.58±0.28	61.50±0.24	64.05±3.40	78.36±1.11	74.50±0.49	76.32±1.15	9.1
L2_SP	67.70±3.21	73.55±0.82	62.43±0.34	68.12±3.77	76.73±0.92	75.74±1.50	82.21±2.44	7.1
DELTA	67.79±0.76	75.22±0.54	63.34±0.58	72.11±3.06	**80.18±1.13**	77.49±0.88	81.83±1.17	4.1
BSS	68.02±2.76	75.01±0.71	63.11±0.49	70.75±4.98	77.92±2.01	77.63±0.83	82.48±2.10	4.6
GTOT-Tuning	70.04±1.48	75.20±0.94	62.89±0.46	71.77±5.38	79.82±1.78	78.13±1.13	82.48±2.18	3.6
VGAE-Tuning	71.69±0.51	75.79±0.41	63.93±0.38	68.63±1.30	77.70±1.43	77.50±0.13	77.83±0.64	4.6
Ours w/o Pre	65.91±2.54	70.09±0.73	59.79±0.62	61.04±2.55	73.39±1.88	75.55±2.06	81.63±0.71	9.4
Ours	**72.59±0.32**	**75.80±0.29**	**64.25±0.27**	**74.64±4.30**	75.84±1.97	**78.33±0.67**	**84.79±1.39**	**1.3**

Table 1: Mean and standard deviation of ROC-AUC(%) of fine-tuning on the same domain with pre-training dataset.

5.1 Fine-tuning GNNs

Setting. To answer **Q1**, we evaluate **G-TUNING** on the molecular property prediction task to show its effectiveness. Following the setting of (Hu et al. 2020a; Zhang et al. 2022), we use the model pre-trained by unsupervised context prediction task as the backbone model. Specifically, we pre-train GIN (Xu et al. 2019) by self-supervised Context Prediction task on the ZINC15 dataset with 2 million unlabeled molecules (Sterling and Irwin 2015). Next, we perform fine-tuning of the backbone model on 7 binary classification datasets obtained from MoleculeNet (Wu et al. 2018). We use the scaffold split at an 8:1:1 ratio. Since our framework is agnostic to the backbone GNNs, we focus on evaluating whether our model achieves better fine-tuning results. For each dataset, we run 5 times and report the average ROC-AUC with the corresponding standard deviation.

Results. Tab 1 shows that **G-TUNING** achieves 6 best performance among 7 datasets against the baselines, taking a top average rank. We notice that constraining the embedding from pre-trained model like Feature-Map or DELTA sometimes bring worse performance than vanilla tuning. From the comparison between supervised learning and Ours w/o Pre, although there might be occasional instances of slight performance drops when applying the **G-TUNING** loss to supervised learning, the majority of supervised training experiences benefits from **G-TUNING** loss. From the comparison between vanilla tuning and the last two rows of the table, the performance of **G-TUNING** without pre-training is lower than that with pre-training but sometime better than the vanilla tuning. The results generally prove that in the case that datasets come from the same domains, **G-TUNING** can compensate for structural divergence by preserving generative patterns and lead to better performance.

5.2 Fine-tuning GNNs Across Domains

Settings. To answer **Q2**, we propose to evaluate **G-TUNING** in the cross-domain setting where the pre-training and downstream datasets are not from the same domains. It is a more challenging yet more realistic setting (Qiu et al. 2020; You et al. 2020) due to the larger structural divergence which can in turn degrade the performance. Therefore, we adopt GCC (Qiu et al. 2020) as the backbone model and its subgraph discrimination as the pre-train task. Following the setting of GCC, we pre-train on 7 different datasets ranging from academia to social domains, and evaluate our approach on 7 downstream graph classification benchmarks: IMDB-M, IMDB-B, MUTAG, PROTEINS, ENZYMES, MSRC_21 and RDT-M12K from the TUDataset (Morris et al. 2020). These datasets cover a wide range of domains including movies, chemistry, bioinformatics, vision graphs and social network. We report the results under 10-fold cross-validation.

Results. From Tab 2, we find that our model outperforms all baselines on 6 out of 7 datasets and presents competitive results on MUTAG (1.92% lower than the best). **G-TUNING** improves performance on PROTEINS by 7.63% and 4.71% when compared to vanilla tuning and the second best baseline respectively. We can observe a more substantial improvement of **G-TUNING** compared with the previous experiment (Tab 1), because we explicitly preserve the generative patterns. Although GTOT also incorporates structural information, it sometimes even performs worse than vanilla tuning (*i.e.* PROTEINS and ENZYMES). Generally, **G-TUNING** clearly demonstrates its effectiveness when pre-training and fine-tuning graphs exhibit large structural divergence.

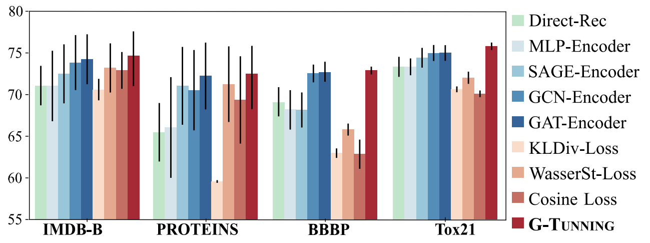

Figure 3: Ablation study on **G-TUNING**.

5.3 Model Ablation and Hyper-parameter Study

Model ablation study. To answer **Q3**, we choose 2 datasets from above 2 experiments to perform ablation study (Fig 3). Across four datasets, we observe **G-TUNING** always outperforms “Direct-Rec”. Next, we compare different GNN

	IMDB-M	IMDB-B	MUTAG	PROTEINS	ENZYMES	MSRC_21	RDT-M12K	Rank
Supervised	36.67±6.67	52.40±7.20	82.89±6.16	63.51±3.60	20.50±4.02	7.45±3.52	38.09±0.56	10.1
Vanilla Tuning	50.20±2.72	72.10±3.65	83.45±5.71	64.42±4.91	21.33±5.62	7.99± 2.39	40.53±1.21	6.6
StochNorm	49.87±3.11	72.20±2.99	82.40±7.95	64.08±3.61	23.33±4.25	9.08±3.21	41.02±0.98	6.0
Feature Map	50.87±2.89	72.90±2.70	82.95±7.80	63.06±4.80	22.67±4.78	9.77±4.30	40.74±1.26	5.6
L2_SP	51.07±2.11	71.90±2.59	82.98±3.91	65.95±4.57	21.50±4.50	10.29±3.91	38.36±0.88	6.0
DELTA	50.67±2.81	71.80±3.99	82.98±6.12	63.96±5.35	22.33±5.01	10.48±3.84	39.87±1.47	6.6
BSS	47.35±1.76	73.20±3.25	84.56±5.52	65.58±6.79	23.41±4.79	10.47±3.29	39.90±1.39	4.7
GTOT-Tuning	51.13±2.72	72.30±2.93	**87.50±6.94**	62.89±3.88	20.67±5.49	8.32±3.92	39.90±0.85	6.0
VGAE-Tuning	49.27±2.37	72.50±2.91	80.35±6.16	67.34±3.52	19.33±7.61	8.88±2.75	41.43±1.88	6.6
Ours w/o Pre	49.27±3.09	72.10±4.99	83.04±3.09	69.88±3.30	20.17±4.37	7.63±2.95	40.92±1.78	6.7
Ours	**51.80±2.31**	**74.30±3.29**	86.14±5.50	**72.05±3.80**	**26.70±4.28**	**11.01±2.08**	**42.80±1.62**	**1.1**

Table 2: Mean and standard deviation of Accuracy(%) of fine-tuning on different domains from pre-training datasets.

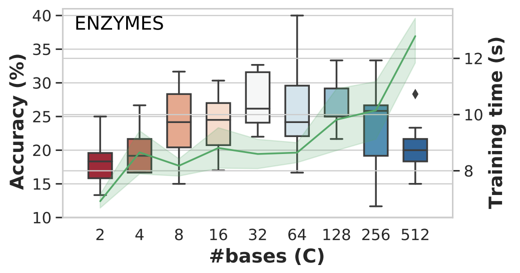

Figure 4: Performance & time varying # graphon bases.

architectures (two-layered MLP, GCN(Welling and Kipf 2016), GraphSAGE(Hamilton, Ying, and Leskovec 2017) and GAT (Veličković et al. 2018)) with the default backbone (*i.e.* GIN (Xu et al. 2019)). In Fig 3, we observe MLP-encoder performs the worst, which proves the effectiveness of incorporating structural information to reconstruct graphon. Lastly, we replace our loss with KL divergence, Wasserstein distance and cosine similarity. We can observe that our GW discrepancy loss significantly outperforms the others. Since KL divergence does not satisfy the commutative law, it is difficult to converge when reconstructing graphons. Though Wasserstein distance is also based on optimal transport, it fails to capture the geometry between two graphons.

Hyper-parameter Study. As analyzed in Theorem 1, more bases can represent more information and better approximate the oracle graphon. Fig 4 shows that the performance increases as the number of bases grows from 2 to 32. However, when the number keep increasing, the improvement becomes smaller.We attribute this phenomenon to the optimization difficulty brought by the increased number of parameters. Therefore, **G-TUNING** only requires a small amount of bases to improve the fine-tuning performance.

5.4 Running Time

To answer **Q4**, we conduct a running time comparison (Tab 3). The time complexity mainly consists of two parts: (i) the graphon approximation from the pre-trained model and (ii) the oracle graphon estimation. Please note that **G-TUNING** incurs additional time only during the training process of the fine-tuning stage, while the rest of the time consumption is the same as the vanilla tuning. As indicated by the number of graphs in the first row, it is evident that as the number of graphs increases, the time consumption of our method becomes more comparable to that of other baselines. It means our method exhibits excellent scalability. Thus, the time consumption of our method is kept in an acceptable range. Moreover, the last row shows the mean proportion of the oracle graphon estimation during the complete training process in the fine-tuning stage. It is evident that a significant portion of time is dedicated to the oracle graphon estimation. If the graphon estimation research advances and can run faster, our method will also benefit from corresponding speed improvements. Due to the space limit, please refer to other experiments in App § A.5.

	BBBP	MUV	HIV
# graphs	2039	93087	41127
Vanilla Tuning	1.95±0.12	81.56±3.21	15.62±0.47
L2_SP	2.22±0.08	118.64±11.41	86.06±4.23
DELTA	2.04±0.10	109.69±14.22	99.13±0.21
Feature Map	1.91±0.16	113.62±11.09	31.69±5.20
BSS	2.09±0.20	114.84±12.50	72.04±5.81
StochNorm	2.05±0.13	120.37±13.24	67.26±5.83
GTOT	1.97±0.11	94.62±3.90	59.32±0.97
VGAE-Tuning	5.67±0.07	238.98±15.78	95.59±3.68
G-TUNING	4.36±0.19	198.04±15.82	73.81±1.35
Oracle W Est	55.70%	56.65%	63.88%

Table 3: The Comparison of Running time.

6 Conclusion

In this paper, we attribute the unsatisfactory performance of the pre-training on graphs to the structural divergence between pre-training and downstream datasets. Moreover, we identify the cause of this divergence as the discrepancy of generative patterns between pre-training and downstream graphs. Building upon our theoretical analysis, we propose **G-TUNING**, a GNN fine-tuning strategy based on graphon, to adapt the pre-trained model to the downstream dataset. Finally, we empirically prove the effectiveness of **G-TUNING**.

Acknowledgments

This work is supported by NSFC (No.62322606) and the Fundamental Research Funds for the Central Universities.

References

Airoldi, E. M.; Blei, D.; Fienberg, S.; and Xing, E. 2008. Mixed membership stochastic blockmodels. *Advances in neural information processing systems*, 21.

Airoldi, E. M.; Costa, T. B.; and Chan, S. H. 2013. Stochastic blockmodel approximation of a graphon: Theory and consistent estimation. *Advances in Neural Information Processing Systems*, 26.

Avella-Medina, M.; Parise, F.; Schaub, M. T.; and Segarra, S. 2018. Centrality measures for graphons: Accounting for uncertainty in networks. *IEEE Transactions on Network Science and Engineering*, 7(1): 520–537.

Borgs, C.; and Chayes, J. 2017. Graphons: A nonparametric method to model, estimate, and design algorithms for massive networks. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, 665–672.

Borgs, C.; Chayes, J. T.; Lovász, L.; Sós, V. T.; and Vesztergombi, K. 2008. Convergent sequences of dense graphs I: Subgraph frequencies, metric properties and testing. *Advances in Mathematics*, 219(6): 1801–1851.

Borgs, C.; Chayes, J. T.; Lovász, L.; Sós, V. T.; and Vesztergombi, K. 2012. Convergent sequences of dense graphs II. Multiway cuts and statistical physics. *Annals of Mathematics*, 151–219.

Chatterjee, S. 2015. Matrix estimation by universal singular value thresholding. *The Annals of Statistics*, 43(1): 177–214.

Chen, X.; Wang, S.; Fu, B.; Long, M.; and Wang, J. 2019. Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning. *Advances in Neural Information Processing Systems*, 32.

Erdős, P.; and Rényi, A. 1959. On random graphs I. Publicationes Mathematicae (Debrecen).

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*, volume 30, 1025–1035.

Han, X.; Huang, Z.; An, B.; and Bai, J. 2021. Adaptive transfer learning on graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 565–574.

Han, X.; Jiang, Z.; Liu, N.; and Hu, X. 2022. G-Mixup: Graph Data Augmentation for Graph Classification. *arXiv preprint arXiv:2202.07179*.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; and Leskovec, J. 2020a. Strategies For Pre-training Graph Neural Networks. In *ICLR (ICLR)*.

Hu, Z.; Dong, Y.; Wang, K.; Chang, K.-W.; and Sun, Y. 2020b. Gpt-gnn: Generative pre-training of graph neural networks. In *KDD*, 1857–1867.

Jiao, R.; Han, J.; Huang, W.; Rong, Y.; and Liu, Y. 2023. Energy-Motivated Equivariant Pretraining for 3D Molecular Graphs. *AAAI*.

Jiarong, X.; Renhong, H.; Xin, J.; Yuxuan, C.; Carl, Y.; Chunping, W.; and Yang, Y. 2023. Better with Less: A Data-Centric Prespective on Pre-Training Graph Neural Networks. *Advances in Neural Information Processing Systems*.

Kipf, T. N.; and Welling, M. 2016. Variational Graph Auto-Encoders. *NIPS Workshop on Bayesian Deep Learning*.

Kou, Z.; You, K.; Long, M.; and Wang, J. 2020. Stochastic normalization. *Advances in Neural Information Processing Systems*, 33: 16304–16314.

Leskovec, J.; Kleinberg, J.; and Faloutsos, C. 2007. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1): 2–es.

Li, H.; Zhao, D.; and Zeng, J. 2022. KPGT: Knowledge-Guided Pre-training of Graph Transformer for Molecular Property Prediction. *KDD*.

Li, X.; Xiong, H.; Wang, H.; Rao, Y.; Liu, L.; Chen, Z.; and Huan, J. 2019. Delta: Deep learning transfer using feature map with attention for convolutional networks. *ICLR*.

Lovász, L. 2012. *Large networks and graph limits*, volume 60. American Mathematical Soc.

Lovász, L.; and Szegedy, B. 2006. Limits of dense graph sequences. *Journal of Combinatorial Theory, Series B*, 96(6): 933–957.

Lu, Y.; Jiang, X.; Fang, Y.; and Shi, C. 2021. Learning to pre-train graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4276–4284.

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

Parise, F.; and Ozdaglar, A. 2019. Graphon games. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, 457–458.

Pensky, M. 2019. Dynamic network models and graphon estimation.

Qiu, J.; Chen, Q.; Dong, Y.; Zhang, J.; Yang, H.; Ding, M.; Wang, K.; and Tang, J. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. *KDD*.

Ruiz, L.; Chamon, L.; and Ribeiro, A. 2020a. Graphon neural networks and the transferability of graph neural networks. *Advances in Neural Information Processing Systems*, 33: 1702–1712.

Ruiz, L.; Chamon, L. F.; and Ribeiro, A. 2020b. The graphon fourier transform. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5660–5664. IEEE.

Ruiz, L.; Chamon, L. F.; and Ribeiro, A. 2021. Graphon signal processing. *IEEE Transactions on Signal Processing*, 69: 4961–4976.

Sterling, T.; and Irwin, J. J. 2015. ZINC 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11): 2324–2337.

Sun, M.; Xing, J.; Wang, H.; Chen, B.; and Zhou, J. 2021. MoCL: data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 3585–3594.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Vizuete, R.; Garin, F.; and Frasca, P. 2021. The Laplacian spectrum of large graphs sampled from graphons. *IEEE Transactions on Network Science and Engineering*, 8(2): 1711–1721.

Welling, M.; and Kipf, T. N. 2016. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*.

Wu, Z.; Ramsundar, B.; Feinberg, E. N.; Gomes, J.; Geniesse, C.; Pappu, A. S.; Leswing, K.; and Pande, V. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical science*, 9(2): 513–530.

Xia, J.; Zhu, Y.; Du, Y.; and Li, S. Z. 2022. A survey of pretraining on graphs: Taxonomy, methods, and applications. *arXiv preprint arXiv:2202.07893*.

Xu, H.; Luo, D.; Carin, L.; and Zha, H. 2021. Learning graphons via structured gromov-wasserstein barycenters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 10505–10513.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

Xuhong, L.; Grandvalet, Y.; and Davoine, F. 2018. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning*, 2825–2834. PMLR.

You, J.; Ying, R.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, 5708–5717. PMLR.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph contrastive learning with augmentations. *NeurIPS*.

Zhang, J.; Xiao, X.; Huang, L.-K.; Rong, Y.; and Bian, Y. 2022. Fine-Tuning Graph Neural Networks via Graph Topology Induced Optimal Transport. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 3730–3736. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Zhu, Q.; Yang, C.; Xu, Y.; Wang, H.; Zhang, C.; and Han, J. 2021. Transfer learning of graph neural networks with ego-graph information maximization. *Advances in Neural Information Processing Systems*, 34: 1766–1779.