Measuring Task Similarity and Its Implication in Fine-Tuning Graph Neural Networks

Renhong Huang[1,2*†], **Jiarong Xu**[2*‡], **Xin Jiang**[3], **Chenglu Pan**[1], **Zhiming Yang**[2], **Chunping Wang**[4], **Yang Yang**[1]

[1]Zhejiang University, [2]Fudan University, [3]Lehigh University, [4]FinVolution Group
{renh2,chenglupan,yangya}@zju.edu.cn, {jiarongxu,zmyang20}@fudan.edu.cn,
xjiang@lehigh.edu, wangchunping02@xinye.com

Abstract

The paradigm of pre-training and fine-tuning graph neural networks has attracted wide research attention. In previous studies, the pre-trained models are viewed as universally versatile, and applied to a diverse range of downstream tasks. In many situations, however, this practice results in limited or even negative transfer. This paper, for the first time, studies the specific application scope of graph pre-trained models, *i.e.*, the extent to which downstream tasks can benefit from specific pre-training tasks. We find that not all downstream tasks can effectively benefit from a graph pre-trained model. In light of this, we introduce the measure *task consistency* to quantify the similarity between graph pre-training and downstream tasks. This measure assesses the extent to which downstream tasks can benefit from specific pre-training tasks. Moreover, a novel fine-tuning strategy, Bridge-Tune, is proposed to further diminish the impact of the difference between pre-training and downstream tasks. The key innovation in Bridge-Tune is an intermediate step that bridges pre-training and downstream tasks. This step takes into account the task differences and further refines the pre-trained model. The superiority of the presented fine-tuning strategy is validated via numerous experiments with different pre-trained models and downstream tasks.

1 Introduction

The paradigm of pre-training and fine-tuning graph neural networks (GNNs) has recently become an active research area and is able to learn transferable knowledge from graph data without costly labels (Hu et al. 2020b; Liu et al. 2022; Rong et al. 2020; Qiu et al. 2020; Xu et al. 2023; Ma et al. 2023; Xu et al. 2022). This paradigm typically involves two steps: (1) pre-train a GNN encoder on unlabeled graph data via a pre-training task; (2) fine-tune the pre-trained GNN on unseen data so as to benefit different downstream tasks. Such a design hopes to build a one-fits-all model that always benefits the downstream.

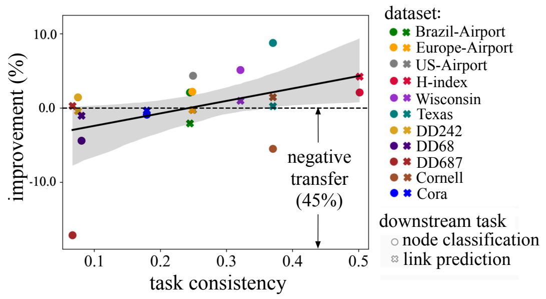


Figure 1: Plot of performance improvement versus the proposed task consistency measure. It shows a clear positive correlation: a larger task consistency implies higher improvement, which in turn suggests that the downstream task can benefit more from the pre-training task. Different points represent improvement on different downstream tasks and different datasets. The black solid line is fitted via linear regression, and the gray shaded area indicates the 95% confidence interval.

However, this ideal expectation is far from the truth in real-world scenarios. As demonstrated in Figure 1, the fine-tuned GCC model (Qiu et al. 2020) suffers from negative transfer in 45.5% of downstream tasks tested, and the given pre-trained model excels in some downstream tasks while underperforms in others. (The improvement result is computed as the relative difference in downstream performance between fine-tuned GCC and GCC learned from scratch.) This undesirable phenomenon is largely attributed to the difference between the pre-training task and the downstream task; as also observed in (Hu et al. 2020b; Lu et al. 2021; Ju et al. 2023).

In view of this, it is then crucial to examine in which case the downstream benefits from the pre-trained model, which in turn asks for a measure to quantify the similarity between a pre-training task and a downstream task. Task similarity has been studied in the literature, but is typically defined based on label distributions (Ganin et al. 2016; Geng 2016; Chen et al. 2020). These approaches are not applicable in our case because graph pre-training is conducted on unlabeled data. What's worse, the pre-training and downstream tasks are often defined on different spaces or have distinct objectives, which makes the comparison more difficult.

Considering the above practical needs and challenges, this paper proposes a novel measure of *task consistency* to

[*]These authors contributed equally.

[†]This work was done when the author was a visiting student at Fudan University.

[‡]Corresponding author.

quantify the similarity between various graph pre-training and downstream tasks within a unified space. Specifically, we introduce a pair-wise label space that is able to encompass different pre-training and downstream tasks even if they are originally defined in different ways. With this novel pair-wise label space, the *task consistency* measure is then proposed to identify those downstream tasks that might benefit from a given graph pre-trained model, as demonstrated in Figure 1.

For those downstream tasks that can potentially make good use of a given pre-trained model, the next question is *how to diminish the impact of task inconsistency* so as to better leverage the knowledge in the pre-trained model. In this case, the proposed task consistency measure is not helpful: The difference between two tasks is intrinsic and is not altered in any learning process. To resolve this difficulty, we introduce the concept of *representation consistency* and a novel fine-tuning strategy *Bridge-Tune*.

The first step is to modify the task consistency measure to take into account the representations learned by the pre-trained model. Such a measure is called *representation consistency*, and can be viewed as a soft version of the task consistency measure. It is able to quantify the contribution of the learned representations to downstream tasks, and is used to guide the refinement of the pre-trained model.

Second, with the proposed representation consistency, we develop a novel fine-tuning strategy, *Bridge-Tune*. The key innovation in Bridge-Tune is an intermediate step between pre-training and downstream. This process is called *pre-trained model refinement*, and aims to maximize the proposed representation consistency. The effectiveness of Bridge-Tune is illustrated in Figure 2. The traditional fine-tuning easily falls into a suboptimal point in the downstream task. In comparison, the pre-trained model refinement step helps find a better starting point for fine-tuning and so Bridge-Tune potentially builds a better model for the downstream task.

Our contributions are summarized as follows.

- **New measure.** We propose a *task consistency* measure to quantify the potential benefits gained by the downstream task from a graph pre-trained model.
- **New method.** We introduce *Bridge-Tune*, a novel fine-tuning strategy. Instead of directly fine-tuning a pre-trained model, Bridge-Tune takes an intermediate step that bridges the pre-training and downstream tasks and refines the model representations.
- **Theoretical guarantees and numerical results.** The effectiveness of Bridge-Tune is verified via theoretical analysis and demonstrated by numerical experiments. In particular, the superiority of Bridge-Tune is justified with different choices of pre-trained models and downstream tasks.

The rest of the paper is organized as follows. The classical paradigm of GNN pre-training and fine-tuning is reviewed in §2. In §3 we introduce the measure of task consistency to quantify the similarity between pre-training and downstream tasks. Then §4 unveils the proposed novel fine-tuning strategy, Bridge-Tune. Numerical experiments in §5 demonstrate the superiority of our approach across various settings.

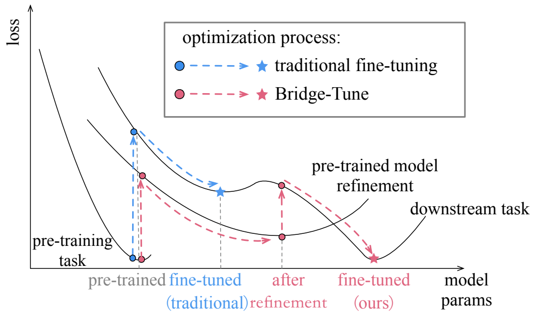


Figure 2: Illustration of the optimization process in traditional fine-tuning (blue) and our fine-tuning strategy (red). The arrow from one (solid) curve to another indicates a change in tasks, and the arrow along one curve represents the optimization process.

2 Preliminary and Related Works

In this section, we introduce the basic paradigm of graph pre-training. It typically consists of two steps: pre-training and fine-tuning. First, given a collection of unlabeled graphs G_{pre}, we *pre-train* a generic GNN encoder f_θ by optimizing the self-supervised learning objective $\mathcal{L}_{\text{pre}}$:

$$\theta_{\text{pre-train}} = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}_{\text{pre}}\left(f_\theta; G_{\text{pre}}\right).$$

The learned parameter $\theta_{\text{pre-train}}$ is expected to capture unified and transferable structural patterns in the training graphs. The choice of $\mathcal{L}_{\text{pre}}$ relies on the pre-training task, and this paper focuses on the following three: graph contrastive learning, graph reconstruction and graph context prediction.

Graph Contrastive Learning. The goal of contrastive pre-training task is to capture the similarities (and dissimilarities) between subgraph instances (You et al. 2020; Qiu et al. 2020; Zheng et al. 2022). Specifically, given a subgraph instance ξ_i from an ego network Γ_i centered at the node v_i, we could get its representation $x_i = f(\xi_i)$ via the graph encoder f. The encoder f aims to encourage high similarity between x_i and the representations of another subgraph instance ξ_i^+ which is sampled from the same ego network. This work could be done through optimizing the InfoNCE loss (Oord, Li, and Vinyals 2018): $\mathcal{L}_{\text{pre}} = -\log \frac{e^{x_i^\top f(\xi_i^+)/\tau}}{e^{x_i^\top f(\xi_i^+)/\tau} + \sum_{\xi_i' \in \Omega_i^-} e^{x_i^\top f(\xi_i')/\tau}}$, where Ω_i^- denotes the collection of subgraph instances that sampled from different ego networks $\Gamma_j\,(j \neq i)$ and τ denotes a pre-defined hyper-parameter. The inner product here denotes the similarity measure between the two subgraph instances.

Graph Reconstruction. Graph autoencoder is another popular approach for GNN pre-training, and utilizes graph reconstruction as self-supervised tasks (Hamilton, Ying, and Leskovec 2017). The main objective of the graph encoder f in graph reconstruction is to encourage high similarity between connected node pairs and low similarity between unconnected node pairs: $\mathcal{L}_{\text{pre}} = -\log \sigma\left(h_u^\top h_v\right) - \log\left(1 - \sigma(h_u^\top h_{v'})\right)$, where v is connected to u but disconnected to v', h_u denotes the representation of node u, and $\sigma(\cdot)$ is the sigmoid function.

Graph Context Prediction. Graph context prediction aims to leverage subgraphs to make predictions about the surrounding graph structures (namely, context graph) (Hu et al. 2020b), by classifying whether a particular neighborhood and a context graph belong to the same node within a K-hop neighborhood. The objective can be formulated as $\mathcal{L}_{\text{pre}} = -\log \sigma(h_v^\top c_v) - \sum_{v' \sim \Omega_v^-} \log\left(1 - \sigma(h_v^\top c_{v'})\right)$, where Ω_v^- is the set of nodes excluding node v, and $h_v^{(K)}$ and c_v are representations of K-hop neighborhood and context graph of node v.

Second, in the fine-tuning stage, the GNN model (initialized with the pre-trained parameters $\theta_{\text{pre-train}}$) is trained on the loss of downstream task $\mathcal{L}_{\text{down}}$ end-to-end together with the classifier on the downstream task. Recent works focus on how to make the most use of the knowledge in pre-trained models during the fine-tuning phase, they can be categorized into parameter regularization (Xuhong, Grandvalet, and Davoine 2018) and representation regularization (Li et al. 2019; Chen et al. 2019; Kou et al. 2020; Flamary et al. 2016; Xu et al. 2020). In the graph domain, some efforts have been also made to develop better fine-tuning strategies. (Zhang et al. 2022) adapts the optimal transport to constrain the fine-tuned model behaviors, which is a kind of representation regularization. (Xia et al. 2022) uses a regularization built on dropout to control the complexity of pre-trained models. Although there are various forms of fine-tuning, it is evident that a gap exists between the learning objectives of the pre-training task and the downstream task.

3 Measure Task Similarity

This section presents a measure to quantify the similarity between pre-training and downstream tasks. We begin to introduce a pair-wise label space to relocate these two tasks in a common space in §3.1, and then our proposed measure of task consistency is presented in §3.2.

3.1 Pair-Wise Label Space

Graph pre-trained models cannot retain competitive performance across all downstream tasks, as significant difference exists between pre-training and downstream tasks. Specifically, the pre-training task typically works on the representation space: By mapping the input data to representations, it attempts to optimize the (dis)agreement of node representations. In comparison, the downstream task is often defined on the label space, aiming to classify the downstream data.

To facilitate the comparison of these tasks, we introduce a new label in both pre-training and downstream tasks. The presented new label is applied to a pair of nodes, and with this definition, the tasks in graph pre-training and downstream can be converted into the same *pair-wise label space*.

Definition 1 (Pair-wise label space)**.** *Given two samples v_i, v_j and their respective labels $y(v_i), y(v_j)$, the label of the node pair (v_i, v_j) is defined as*

$$y^*(v_i, v_j) = \mathbf{1}(y(v_i) = y(v_j)), \tag{1}$$

where $\mathbf{1}(\cdot)$ is the indicator function. The pair-wise label space $\mathcal{Y}^*$ consists of all the labels $y^*(v_i, v_j)$.

With Definition 1, a variety of pre-training and downstream tasks can be represented in the pair-wise label space. As an example, node classification in the pair-wise label space sets $y^*(v_i, v_j) = 1$ if v_i and v_j have the same label. The conversion of link prediction into the pair-wise label space is also straightforward: $y^*(v_i, v_j) = 1$ if a link exists between node v_i and v_j, and $y^*(v_i, v_j) = 0$ otherwise.

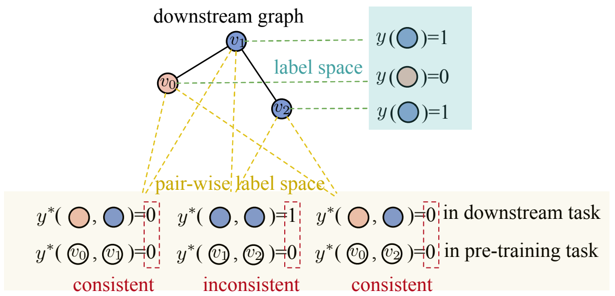


Figure 3: Illustrating example of the pair-wise label space and the measure of task consistency. Consider node classification as a downstream task, originally defined in the label space, a node pair comprising nodes with different labels are labeled as 0 in the pair-wise label space. Take contrastive learning as the pre-training task, a node pair comprising nodes distinct nodes are labeled as 0 in the pair-wise label space. Based on the shared pair-wise label space, the task consistency can be defined to measure the probability of the labels of node pairs being the same in both tasks.

For pre-training tasks, we study three self-supervised learning approaches and convert them to the pair-wise label space.

- *Graph Contrastive Learning.* Contrastive learning can be viewed as an instance discrimination task, where each instance is treated as a distinct class of its own (Wu et al. 2018). Accordingly, the label of a node pair in contrastive learning can be defined as $y^*(v_i, v_j) = 0$ if $v_i \neq v_j$.
- *Graph Reconstruction.* Graph reconstruction aims to reconstruct the existence of links between node pairs. Hence, the labels of node pairs in this task can be naturally defined as $y^*(v_i, v_j) = 1$ if a link exists between v_i and v_j, and 0 otherwise.
- *Graph Context Prediction.* Graph context prediction is originally a graph-level task, and aims to determine whether a specific neighborhood and a context graph correspond to the same node (Hu et al. 2020b). Roughly speaking, if two nodes are located close enough within K-hops of each other, their neighborhoods and context graphs are considered similar enough. Based on this relaxation, graph context prediction can be converted to a node-level task: $y^*(v_i, v_j) = 1$ if (v_i, v_j) are within K-hops of each other, and $y^*(v_i, v_j) = 0$ otherwise.

Figure 3 presents an illustrating example of converting contrastive learning and node classification into the pair-wise label space.

3.2 Task Consistency

After converting pre-training and downstream tasks to the same pair-wise label space, we introduce the measure *task consistency* to quantify the similarity between these tasks.

Definition 2 (Task consistency)**.** *Given a pre-training task $\mathcal{P}$ and a downstream task $\mathcal{D}$, denote by $\mathcal{Y}^*_{\mathcal{D}}$ its pair-wise label space on the downstream task and by $\mathcal{Y}^*_{\mathcal{P}}$ its pair-wise label space on the pre-training task. The task consistency of the downstream task $\mathcal{D}$ with the pre-training task $\mathcal{P}$ is defined by*

$$C_{\mathrm{T}}(\mathcal{D}, \mathcal{P}) = \mathbb{P}[y^*_{\mathcal{D}}(\boldsymbol{n}) = y^*_{\mathcal{P}}(\boldsymbol{n})], \tag{2}$$

where $\mathcal{V}_{\mathcal{D}}$ is the space of nodes in the downstream graph, and $\boldsymbol{n} \in \mathcal{V}_{\mathcal{D}} \times \mathcal{V}_{\mathcal{D}}$ is a node pair taken from the downstream graph.

Figure 3 presents an illustrating example of how task consistency is computed when the pre-training task is contrastive learning and the downstream task is node classification. Towards an empirical verification of task consistency, Figure 1 clearly presents a positive correlation between the task consistency and the performance improvement on downstream tasks brought by pre-trained model (see experiment details in Appendix A.2). The larger the task consistency of a downstream task, the more benefit the task can benefit from the pre-trained model. It is also significant that those downstream tasks with low task consistency suffer from negative transfer. This provides us with the rationality to leverage task consistency to determine the extent to which downstream tasks can benefit from specific graph pre-trained models.

The following theorem builds a theoretical connection between the proposed task consistency and the generalization ability from a pre-training task to a downstream task. Its proof is postponed to Appendix A.5.

Theorem 1 (Connection between generalization error and task consistency)**.** *Let $\mathcal{P}$ and $\mathcal{D}$ be the pre-training task and the downstream task, defined on a shared pair-wise label space. Let $C_{\mathrm{T}}(\mathcal{D}, \mathcal{P})$ be the task consistency between $\mathcal{P}$ and $\mathcal{D}$, let $\mathcal{S}$ be an infinite hypothesis set, and let $R(s)$ be the generalization error of a hypothesis $s \in \mathcal{S}$ on $\mathcal{D}$. Then, for any $\delta > 0$, the following inequality holds with probability at least $1 - \delta$:*

$$R(s) \leq \frac{\log(|\mathcal{S}|/\delta)}{mC_{\mathrm{T}}(\mathcal{D}, \mathcal{P})}.$$

4 Bridge-Tune Enhances Downstream

The proposed task consistency measure reveals which downstream tasks can benefit from specific graph pre-trained models. Building upon this finding, this section introduces a novel fine-tuning strategy that aims at maximizing the utilization of pre-trained models to enhance downstream task performance. This is achieved by mitigating the impact of the difference between pre-training and downstream tasks.

Towards this purpose, in §4.1, drawing inspiration from task consistency, we introduce a measure called *representation consistency* to monitor the impacts of model representations on the downstream tasks during the fine-tuning process. Subsequently, we unveil our novel wisdom of fine-tuning strategy, termed *Bridge-Tune*.

4.1 Representation Consistency

To diminish the impact of task inconsistency during fine-tuning, further improvement on the graph pre-trained model should be made. However, the proposed task consistency is an intrinsic property of the two tasks, so cannot help to further the pre-trained model. In view of this, *representation consistency* is proposed to guide the refinement of the pre-train model. It is a soft version of the task consistency measure, and quantifies the contribution of model representations to the downstream.

Definition 3 (Representation consistency)**.** *Given the output representation space $\mathcal{H}$ of the graph pre-training model, the downstream task $\mathcal{D}$, and the pre-training task $\mathcal{P}$, the representation consistency is defined as*

$$C_{\mathrm{R}}(\mathcal{H}, \mathcal{D}, \mathcal{P}) = \mathbb{E}_{\boldsymbol{n}}\left[\rho \mathrm{Sim}(h(\boldsymbol{n})) | y^*_{\mathcal{D}}(\boldsymbol{n}) = y^*_{\mathcal{P}}(\boldsymbol{n})\right], \quad (3)$$

where $\boldsymbol{n}$ is a random node pair from $\mathcal{V}_{\mathcal{D}} \times \mathcal{V}_{\mathcal{D}}$, ρ is a binary indicator (with $\rho = 1$ if $y^*_{\mathcal{P}}(\boldsymbol{n}) = y^*_{\mathcal{D}}(\boldsymbol{n}) = 1$ and -1 if $y^*_{\mathcal{P}}(\boldsymbol{n}) = y^*_{\mathcal{D}}(\boldsymbol{n}) = 0$), $h\colon \mathcal{V}_{\mathcal{D}} \times \mathcal{V}_{\mathcal{D}} \to \mathcal{H} \times \mathcal{H}$ is a mapping from node pair to representation pair, and $\mathrm{Sim}(\cdot)$ *is the cosine similarity.*

The introduction of $\mathrm{Sim}(h(\boldsymbol{n}))$ in (3) is inspired by the observation that for a downstream graph, if nodes with similar (dissimilar) representations also exhibit the same (different) labels, fine-tuning is more likely to maximize the potential of the pre-trained model. Moreover, the representation consistency C_{R} can be viewed as a soft version of the task consistency C_{T}: If $y^*_{\mathcal{D}}(\boldsymbol{n}) = y^*_{\mathcal{P}}(\boldsymbol{n})$, then $C_{\mathrm{R}} = \mathbb{E}[\rho \mathrm{Sim}(h(\boldsymbol{n}))] \leq 1 = C_{\mathrm{T}}$. Further discussion on representation consistency, especially its theoretical connection with some other existing measures (*e.g.*, inter-class distance), can be found in Appendix A.5.

4.2 Our Fine-Tuning Strategy

Motivated by the intuition that a larger representation consistency is more desirable for the downstream, we present in this section a novel fine-tuning strategy, Bridge-Tune, which introduces an intermediate task between pre-training and traditional fine-tuning and further improve downstream performance.

Given a graph pre-trained model, Bridge-Tune consists of two stages: (1) *Pre-trained model refinement*: We maximize the empirically computed representation consistency. This stage acts as an intermediate task between pre-training and traditional fine-tuning. (2) *Downstream fine-tuning*: We conduct traditional fine-tuning, in which the graph model is initialized with the learned parameters of the refined pre-trained model.

Two challenges exist during the refinement process. Computation of the empirical representation consistency needs node labels from the downstream task. However, in many cases, part of the downstream labels are inaccessible, making refinement loss computation difficult. Besides the lack of downstream labels, computing refinement loss is expensive as it involves all pairs of nodes. Thus, the computation efficiency is another concern for our Bridge-Tune model. We tackle these two problems below.

Better estimation of refinement loss. During fine-tuning, only part of the label set is accessible (call it $\boldsymbol{Y}_{\mathrm{L}}$); the remaining part $\boldsymbol{Y}_{\mathrm{U}}$ is inaccessible and needs to be predicted. With only the labeled part of the downstream data, the estimation of representation consistency is far from accurate. We now propose an improved approach for estimating refinement loss. The key insight is that if an unlabeled node is predicted by the downstream classifier with high confidence during downstream fine-tuning, its prediction can serve as an addition to enhancing the estimation of refinement loss during pre-trained model refinement. In view of this, the two stages are suggested to be performed in a progressive and iterative way as below.

Step 1 (Pre-trained model refinement). The graph encoder model is further trained to maximize the refinement loss on the downstream graph. The update at t-th iteration is

$$\theta_{\text{refine}}^{(t)} = \underset{\theta}{\text{argmax}}\, \mathcal{L}_{\text{refine}} \left(f_\theta; \boldsymbol{Y}_{\text{L}} \cup \boldsymbol{Y}_{\text{P}}^{(t)} \right),$$

where $\mathcal{L}_{\text{refine}}$ is the refinement loss (*i.e.*, the computed representation consistency), f is the graph encoder, $\boldsymbol{Y}_P^{(t)}$ is the predictions of unlabeled nodes given by the downstream classifier, and we set $\boldsymbol{Y}_P^{(0)} = \emptyset$. This optimization process is initialized at $\theta = \theta_{\text{down}}^{(t)}$ at each iteration (see step 2), and we take $\theta_{\text{down}}^{(0)} = \theta_{\text{pre-train}}$.

Step 2 (Downstream fine-tuning). In this step, the graph encoder f is initialized with $\theta_{\text{refine}}^{(t)}$, and trained end-to-end together with the downstream classifier g (parameterized by $\phi_{\text{down}}^{(t)}$) on a downstream task:

$$(\theta_{\text{down}}^{(t+1)}, \phi_{\text{down}}^{(t+1)}) = \underset{\theta,\phi}{\text{argmin}}\, \mathcal{L}_{\text{down}} \left(f_\theta, g_\phi; \boldsymbol{Y}_{\text{L}} \right),$$

$$\boldsymbol{Y}_{\text{P}}^{(t+1)} = g_{\phi_{\text{down}}^{(t)}} \circ f_{\theta_{\text{down}}^{(t)}} (G_{\text{down}}),$$

where $\mathcal{L}_{\text{down}}$ is the loss of the downstream task, and $g \circ f = g(f(\cdot))$ denotes the composition, and $\mathbf{Y}_P$ is the predictions of unlabeled nodes. For efficiency concerns, the optimization process in the first line is initialized at $(\theta, \phi) = (\theta_{\text{refine}}^{(t)}, \phi_{\text{down}}^{(t)})$.

Steps 1 and 2 are performed iteratively. By doing this, pre-trained model refinement and downstream fine-tuning mutually boost the capability of each other, ultimately boosting the downstream performance.

Efficiency improvement. The computation of refinement loss requires all node pairs, thus leading to a high computation cost. To improve efficiency, we propose to sample two specific categories of critical node pairs. (1) The first category involves node pairs in which either one or both nodes possess labels or high-confidence predictions. They are deemed reliable for learning and can accelerate the training process. This set of node pairs forms P_{certain}. (2) The second category includes node pairs where one node is reliable (*i.e.*, labeled or predicted with high confidence) and the other is the downstream classifier uncertain with. In this way, the information in the reliable nodes can diffuse to unlabeled nodes with low prediction confidence. This set of node pairs constitutes $P_{\text{uncertain}}$.

Given the above two node pair sets, we argue that different node pairs should be paid with different attention during the fine-tuning phase proceeds. Initially, the predictions of the downstream classifier may not be accurate enough, in which case we focus on P_{certain}. As the fine-tuning phase proceeds, we can gradually trust the predictions provided by the downstream classifier, and add $P_{\text{uncertain}}$ to estimate refinement loss while paying less attention to labeled node pairs.

In view of this, a time-varying strategy is developed to assign weights to different node pairs. Specifically, the weight of a node pair (v_i, v_j) at t-th iteration is $a_{ij} = \cos(\frac{\pi t}{2T}) p_i p_j$, if $(v_i, v_j) \in P_{\text{certain}}$ and $a_{ij} = \sin(\frac{\pi t}{2T}) p_i (1 - p_j)$, if $(v_i, v_j) \in P_{\text{uncertain}}$, where p_i and p_j are the probabilities of sampling node v_i and v_j respectively, and T is the total number of iterations.

Finally, the refinement loss at t-th iteration is computed as

$$\mathcal{L}_{\text{refine}} = \sum_{(v_i,v_j) \in P_{\text{certain}} \cup P_{\text{uncertain}}} a_{ij} \rho \text{Sim}(f(v_i), f(v_j)) \mathbf{1}(y_{\mathcal{D}}^*(v_i, v_j) = y_{\mathcal{P}}^*(v_i, v_j)).$$

Theoretical analysis. Finally, we theoretically demonstrate that pre-trained model refinement can achieve a lower classification loss on the downstream task than traditional fine-tuning.

Theorem 2 (informal)**.** *Under certain theoretical assumptions, the loss for the downstream task $\mathcal{L}_{down}(\theta_{refine}) \leq \mathcal{L}_{down}(\theta_{pre\text{-}train})$, where θ_{refine} is the model parameter after pre-trained model refinement and $\theta_{pre\text{-}train}$ is the pre-trained model parameter.*

The formal statement of Theorem 2 as well as the proof can be found in Appendix A.5.

5 Experiments

In the experiments, we evaluate the performance of Bridge-Tune with different pre-trained models and on different downstream tasks. We present our setup in §5.1, and the comparison results in terms of performance and runtime in §5.2. Additional experimental results are presented in Appendix A.4. Our codes are available at https://github.com/zjunet/Bridge-Tune.

5.1 Experimental Setup

Datasets. We use a total of 12 downstream datasets for evaluation: US-Airport, Brazil-Airport, Europe-Airport, H-index, Wisconsin, Texas, Cora, Cornell, DD242, DD68, DD687, and the large-scale dataset Ogbarxiv. Since our focus is not on the pre-training stage, in the experiments we directly use the graph pre-trained models that have already been trained on their corresponding datasets.

Baselines. A total number of 13 baselines are considered in the experiments, and they can be roughly categorized into three groups: naïve fine-tuning, advanced fine-tuning, and prompt-tuning. For **naïve fine-tuning**, we compare with (1) Fine-tune: graph encoder initialized with pre-trained parameters is trained end-to-end with downstream classifier; (2) Freeze: graph encoder's parameters are frozen during fine-tuning; and (3) Rand: graph encoder is learning from scratch. **For advanced fine-tuning**, we compare with (1) L2_penalty, L2_SP and L2_SP_Fisher (Xuhong, Grandvalet, and Davoine 2018): parameter regularization-based models; (2) DELTA, Feature (DELTA w/o Att) (Li et al. 2019) and GTOT (Zhang et al. 2022): representation regularization-based models; (3) SupCon (Khosla et al. 2020): a supervised contrastive learning method, which can also be adopted during fine-tuning; (4) L2P (Lu et al. 2021): While not exactly a fine-tuning strategy as it adjusts the pre-training stage to benefit downstream, we include it for a comprehensive comparison; Note that except for GTOT and L2P, the other baselines are originally designed for convolutional neural networks, so we adapt them to our settings by changing the backbone model to the GNN used in our framework. **For prompt-tuning**, we compare with GPPT (Sun et al. 2022a) and GraphPrompt (Liu et al. 2023b).

Settings. We fine-tune on a variety of graph pre-trained models: GCC, GraphCL, EdgePred, and ContextPred. We set the learning rate as 5, 0.1, 0.1, 0.1 when fine-tuning GCC (Qiu

Model \ Dataset	US-Airport	Brazil-Airport	Europe-Airport	H-index	Wisconsin	Texas	DD242	DD68	DD687	Cornell	Cora	Ogbarxiv
Task consistency	0.249	0.245	0.248	0.501	0.321	0.370	0.075	0.081	0.067	0.370	0.179	0.077
Fine-tune	66.21(4.23)	73.24(11.60)	58.62(7.35)	81.90(1.59)	59.82(7.69)	63.95(8.67)	14.49(2.94)	12.39(3.74)	8.26(4.16)	48.04(5.94)	29.54(1.44)	18.62(1.92)
Freeze	63.78(5.11)	69.31(14.02)	53.43(7.57)	74.88(0.97)	53.39(8.47)	62.31(10.80)	14.79(3.17)	12.77(4.46)	11.31(2.74)	47.57(5.65)	29.98(1.82)	14.56(7.60)
Rand	63.44(3.22)	71.73(13.56)	57.35(4.72)	81.21(1.62)	56.89(8.11)	59.70(12.75)	12.31(2.17)	13.81(3.12)	11.18(3.25)	**50.82(3.27)**	29.80(1.74)	18.61(1.88)
L2_penalty	64.63(2.31)	67.60(11.93)	55.38(5.37)	79.88(0.75)	55.84(7.62)	56.33(3.46)	16.59(2.28)	12.26(2.80)	8.98(3.60)	42.54(10.81)	31.68(0.84)	19.27(0.21)
L2_SP	63.70(3.73)	67.02(8.42)	54.41(5.53)	78.56(1.97)	53.43(5.53)	56.95(8.59)	13.71(2.07)	9.16(2.19)	7.04(2.03)	36.49(8.95)	36.60(3.22)	19.28(0.17)
L2_SP_Fisher	64.71(3.94)	71.25(12.42)	57.14(4.81)	80.32(1.44)	57.71(6.52)	60.76(13.11)	19.32(3.83)	13.54(4.25)	8.97(2.09)	44.30(5.80)	43.28(2.21)	/
DELTA	63.37(3.13)	62.56(11.56)	55.95(6.23)	75.42(1.02)	55.39(4.31)	56.36(9.82)	14.57(2.78)	11.62(2.77)	9.93(3.00)	45.35(5.95)	39.33(2.09)	/
Feature (DELTA w/o Att)	62.70(1.84)	62.78(1.60)	53.18(7.08)	72.04(2.17)	55.36(5.94)	64.06(6.32)	14.72(1.73)	9.81(3.18)	7.86(3.85)	47.40(9.43)	37.26(1.48)	19.34(0.13)
GTOT	65.82(3.81)	74.60(14.10)	58.43(4.93)	81.00(1.36)	54.14(6.14)	62.95(7.99)	20.33(3.33)	14.45(2.26)	8.28(2.49)	45.91(8.82)	44.13(2.33)	19.32(0.16)
SupCon	66.56(4.18)	76.21(13.15)	59.88(6.67)	81.00(1.47)	60.60(8.20)	67.28(7.83)	14.88(1.77)	9.54(3.69)	7.17(1.71)	37.69(11.29)	35.42(3.13)	19.36(0.31)
L2P	55.90(3.80)	68.43(16.48)	57.51(4.20)	80.82(1.43)	59.37(9.04)	62.32(4.26)	9.15(2.68)	8.59(1.41)	7.26(2.67)	27.89(15.97)	28.97(3.55)	19.13(0.35)
GPPT	64.03(4.13)	65.66(13.82)	53.12(9.00)	74.66(1.90)	47.42(5.22)	57.90(7.46)	13.63(1.31)	10.83(2.98)	6.35(3.20)	43.60(11.57)	35.45(1.43)	19.85(0.19)
GraphPrompt	62.86(5.90)	60.99(16.06)	50.35(10.46)	73.40(1.85)	51.40(7.27)	59.06(7.59)	13.79(1.83)	11.75(3.00)	5.93(2.40)	39.83(8.19)	36.52(1.97)	19.78(0.25)
Bridge-Tune	**68.99(4.96)**	**77.86(13.95)**	**61.88(5.22)**	**82.66(0.96)**	**62.23(8.44)**	**70.00(5.82)**	**22.97(2.64)**	**16.00(5.60)**	**12.82(4.14)**	50.32(9.37)	**44.17(3.37)**	**20.49(4.35)**

Table 1: Micro F1 scores of different fine-tuning strategies on pre-trained GCC model under the downstream task of node classification. The notation "/" means out of memory or no convergence for more than three days. The p-values comparing our model with competitive baseline GTOT are much smaller than 0.05, which indicates our model significantly outperforms baselines.

et al. 2020), GraphCL (You et al. 2020), EdgePred (Hamilton, Ying, and Leskovec 2017), and ContextPred (Hu et al. 2020b) respectively. We utilize mini-batch training and the batch size is 32. The total iterations of fine-tuning is 30, alternating between one iteration of pre-trained model refinement and one iteration of downstream fine-tuning. When defining P_{certain}, we regard nodes with prediction confidence higher than 0.5 as high-confidence nodes. The numbers reported in all the experiments are the mean and standard deviation over 10 trials. More details can be found in Appendix A.4.

		Node Classification	Link Prediction
GCC	Task consistency	0.249	0.980
	GTOT	65.82(3.81)	90.76(0.12)
	Bridge-Tune	68.99(4.96)	94.97(0.35)
	improvement	4.82% ↑	4.64% ↑
GraphCL	Task consistency	0.249	0.980
	GTOT	59.58(3.13)	63.81(2.37)
	Bridge-Tune	60.50(2.84)	65.71(1.80)
	improvement	1.54% ↑	2.98% ↑
EdgePred	Task consistency	0.242	1.000
	GTOT	61.09(6.63)	62.46(1.87)
	Bridge-Tune	61.34(2.03)	63.15(1.20)
	improvement	0.41% ↑	1.10% ↑
ContextPred	Task consistency	0.292	0.103
	GTOT	60.72(2.58)	64.54(2.52)
	Bridge-Tune	61.86(2.47)	65.29(0.98)
	improvement	1.88% ↑	1.16% ↑

Table 2: Micro F1 on different downstream tasks (in columns), given different pre-trained models (in rows) and on dataset US-Airport.

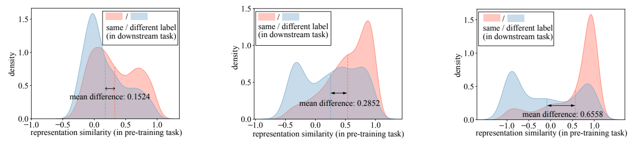
5.2 Experimental Results

Comparison: fine-tuning baselines. Table 1 presents the node classification performance after fine-tuning on the graph pre-trained model GCC. Our model beats the best baseline by an average of +4.68%. In contrast, advanced fine-tuning baselines often cannot help fine-tuning, and even perform worse than directly fine-tuning the models (*i.e.*, the naïve fine-tuning strategies). One possible reason is that most baselines simply focus on regularizing the parameters and representations, and thus cannot essentially diminish the impact of task difference. The unsatisfactory results of SupCon suggest that those methods tailored for supervised learning might not be suitable in "pre-train and fine-tune" paradigm. Recent prompt-tuning approaches also fall short, possibly because they assume pre-training and fine-tuning are conducted on the same dataset.

Comparison: different pre-trained models and downstream tasks. We also conduct experiments using different pre-trained models and on various downstream tasks. Due to space limitations, we only report the comparison between our model and the best baseline GTOT. The results in Table 2 show that our model achieves a significant improvement under various scenarios. More results and details can be found in Appendix A.4.

Ablation study. To demonstrate the effectiveness of each component in our model, we conduct ablation studies on (1) Bridge-Tune-a, which removes the time-varying strategy; (2) Bridge-Tune-P, which does not consider predictions of unlabeled nodes when estimating refinement loss. The Micro F1 score of node classification for Bridge-Tune, Bridge-Tune-a, Bridge-Tune-P, fine-tuned on GCC on US-Airport dataset is 68.99%, 67.25% and 68.32% respectively. The superiority of Bridge-Tune compared with Bridge-Tune-a and Bridge-Tune-P highlights the importance of time-varying strategy and our estimation of refinement loss.

Case study. To examine whether the proposed pre-trained model refinement can help diminish the impact of the difference between pre-training and downstream tasks, we conduct the following analysis. Figure 4 presents the distribution of pre-trained representation similarity of two nodes in negative pairs. We adopt the pre-trained model GCC on the node classification task on US-Airport. The red (or blue) distribution records the similarity distribution of negative pairs whose two nodes are (or are not) from the same class in downstream task. We first observe a significant difference between pre-training and downstream task: As shown in Figure 4(a), We observe that the similarity distributions of negative pairs within the same class (in red) and across different classes (in blue) in the downstream task are indistinguishable. Then, we can see that our pre-trained model refinement indeed helps diminish the task difference: these two distributions are pulled apart in Figure 4(c). We also note that traditional fine-tuning is less distinctive than ours in diminishing the task difference (see Figure 4(b)). Additional results can be found in Appendix A.4.



(a) After pre-train. (b) After fine-tune. (c) After refinement.

Figure 4: The distribution of representation similarity (cosine similarity) of two nodes in negative pairs. The red (or blue) distribution records the similarity distribution of negative pairs whose two nodes are (or not) from the same class in downstream task. The means are shown in dashed vertical lines.

Comparison: runtime. Table 3 presents the runtime for Bridge-Tune and the best baselines in the three categories (naïve fine-tuning, advanced fine-tuning, and prompt-tuning). Bridge-Tune has the best downstream performance while the runtime is comparable to naïve fine-tuning. In contrast, GraphPrompt is the fastest due to fewer learnable parameters but its performance is not satisfactory, GTOT stands as the most competitive baseline but comes with a higher computation burden. A comprehensive time complexity analysis is available in Appendices A.3 and A.4.

	Fine-tune	GraphPrompt	GTOT	Bridge-Tune
pre-trained model refinement	-	-	-	28.65(1.45)
downstream fine-tuning	150.66(4.47)	45.41(3.89)	478.09(9.26)	144.86(8.65)
total	150.66(4.47)	45.41(3.89)	478.09(9.26)	173.51(8.85)

Table 3: Runtime (sec) comparison during the fine-tuning of GCC for node classification on US-Airport. All models are trained till convergence, where the convergence condition is defined as the point where the increase in accuracy on the training set is less than 0.01.

6 Related Work

Graph fine-tuning strategy. Various fine-tuning strategies have been proposed recently, and most research works concentrate on the image and text domains. These works can be roughly categorized into parameter regularization (Xuhong, Grandvalet, and Davoine 2018) and representation regularization (Li et al. 2019; Chen et al. 2019; Kou et al. 2020; Flamary et al. 2016; Xu et al. 2020). However, due to the special structure of graph data, these methods are not directly applicable in the graph domain. Fine-tuning in the graph domain is a promising, yet largely unexplored, research direction. Zhang et al. (2022) adapts representation regularization to graph domain, and the regularizer is inspired by some distances in optimal transport. Though the performance is promising, the use of optimal transport requires a high computational cost, thus not applicable for large-scale models. As another example, Xia et al. (2022) focuses on molecular graphs, and proposes a new regularization tailored to pre-trained molecular model, but this approach can only be applied to molecular graphs.

As a newly developed research direction, prompt-tuning has attracted considerable attention recently. It designs and refines prompts to guide the behavior of pre-trained models towards specific downstream tasks (Liu et al. 2023a). Motivated by this research trend, graph prompt-tuning has received growing research attention. In the context of graph domain, GPPT (Sun et al. 2022a) proposes task token and structure token as prompt template for the node classification applications. GraphPrompt (Liu et al. 2023b) employs a learnable prompt to actively guide downstream tasks using task-specific aggregation. These methods focus on link prediction as the pre-training task and node classification as the downstream task. It is not straightforward to extend these techniques to different tasks, which largely limits their application scope. A very recent paper (Sun et al. 2023) introduces a prompt approach to match various pre-training strategies, but it still lacks explicit consideration of the difference between the pre-training and downstream tasks.

Another line of research, though focusing on the pre-training phase, also attempts to improve downstream performance (Han et al. 2021; Lu et al. 2021). They propose to incorporate auxiliary tasks during pre-training phase so that the pre-trained model is more amenable when adopted to downstream. However, the inclusion of auxiliary tasks potentially compromises the pre-trained model's capacity to generalize across various downstream tasks.

Task similarity. Task similarity refers to the similarity between two machine learning tasks. The research on task similarity was initiated by scholars in the computer vision field, and is used to further improve model performance. Taskonomy (Zamir et al. 2018) delves into the relationship between visual tasks by employing task affinity normalization. Task2vec (Achille et al. 2019) introduces a method to generate task representations. A few follow-up studies apply Task2vec to the graph domain. One such work is GraphGym (You, Ying, and Leskovec 2020), which calculates task similarity by training a collection of anchor models. All the aforementioned works, however, work on supervised tasks, and thus are not applicable to graph pre-training with unlabeled data.

7 Conclusion

We introduce the task consistency measure to quantify the similarity between the graph pre-training and downstream tasks. Such a measure indicates the extent to which a downstream task can benefit from a given pre-training task. Moreover, to diminish the potential impact of task inconsistency, a novel fine-tuning strategy, Bridge-Tune, is proposed, in which the key step aims to mitigate the distinction between the pre-training and downstream tasks. The proposed concepts are theoretically justified, and extensive experiments suggest the superiority of Bridge-Tune on various pre-trained GNN models and downstream tasks.

Acknowledgements

This work is supported by NSFC (No. 62206056, No. 62176233) and the Fundamental Research Funds for the Central Universities.

References

Achille, A.; Lam, M.; Tewari, R.; Ravichandran, A.; Maji, S.; Fowlkes, C. C.; Soatto, S.; and Perona, P. 2019. Task2vec: Task embedding for meta-learning. In *ICCV*, 6430–6439.

Chen, S.; Wang, J.; Chen, Y.; Shi, Z.; Geng, X.; and Rui, Y. 2020. Label distribution learning on auxiliary label space graphs for facial expression recognition. In *CVPR*, 13984–13993.

Chen, X.; Wang, S.; Fu, B.; Long, M.; and Wang, J. 2019. Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning. In *NeurIPS*.

Flamary, R.; Courty, N.; Tuia, D.; and Rakotomamonjy, A. 2016. Optimal transport for domain adaptation. *IEEE Trans. Pattern Anal. Mach. Intell*, 1: 1853–1865.

Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; and Lempitsky, V. 2016. Domain-adversarial training of neural networks. *JMLR*, 17(1): 2096–2030.

Geng, X. 2016. Label distribution learning. *TKDE*, 28(7): 1734–1748.

Gu, Q.; Li, Z.; and Han, J. 2012. Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

Han, X.; Huang, Z.; An, B.; and Bai, J. 2021. Adaptive transfer learning on graph neural networks. In *SIGKDD*, 565–574.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020a. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 22118–22133.

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

Huang, X.; Yang, Y.; Wang, Y.; Wang, C.; Zhang, Z.; Xu, J.; Chen, L.; and Vazirgiannis, M. 2022. Dgraph: A large-scale financial dataset for graph anomaly detection. In *NeurIPS*, 22765–22777.

Ju, M.; Zhao, T.; Wen, Q.; Yu, W.; Shah, N.; Ye, Y.; and Zhang, C. 2023. Multi-task self-supervised graph neural networks enable stronger task generalization. In *ICLR*.

Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; and Krishnan, D. 2020. Supervised contrastive learning. In *NeurIPS*, 18661–18673.

Kornblith, S.; Norouzi, M.; Lee, H.; and Hinton, G. 2019. Similarity of neural network representations revisited. In *ICML*, 3519–3529.

Kou, Z.; You, K.; Long, M.; and Wang, J. 2020. Stochastic normalization. In *NeurIPS*, 16304–16314.

Li, X.; Xiong, H.; Wang, H.; Rao, Y.; Liu, L.; and Huan, J. 2019. Delta: Deep Learning Transfer using Feature Map with Attention for Convolutional Networks. In *ICLR*.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2023a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. In *ACM Computing Surveys*, volume 55, 1–35. ACM New York, NY.

Liu, S.; Wang, H.; Liu, W.; Lasenby, J.; Guo, H.; and Tang, J. 2022. Pre-training Molecular Graph Representation with 3D Geometry. In *ICLR*.

Liu, Z.; Yu, X.; Fang, Y.; and Zhang, X. 2023b. GraphPrompt: Unifying Pre-Training and Downstream Tasks for Graph Neural Networks. In *WWW*, 417–428.

Lu, Y.; Jiang, X.; Fang, Y.; and Shi, C. 2021. Learning to pre-train graph neural networks. In *AAAI*, 4276–4284.

Ma, R.; Xu, J.; Zhang, X.; Zhang, H.; Zhao, Z.; Zhang, Q.; Huang, X.-J.; and Wei, Z. 2023. One-Model-Connects-All: A Unified Graph Pre-Training Model for Online Community Modeling. In *EMNLP*, 15034–15045.

McCallum, A. K.; Nigam, K.; Rennie, J.; and Seymore, K. 2000. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 2(3): 127–163.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*.

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. In *SIGKDD*, 1150–1160.

Ribeiro, L. F.; Saverese, P. H.; and Figueiredo, D. R. 2017. struc2vec: Learning node representations from structural identity. In *SIGKDD*, 385–394.

Rong, Y.; Bian, Y.; Xu, T.; Xie, W.; Wei, Y.; Huang, W.; and Huang, J. 2020. Self-supervised graph transformer on large-scale molecular data. In *NeurIPS*, 12559–12571.

Rossi, R. A.; and Ahmed, N. K. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*, 4292–4293.

Sun, M.; Zhou, K.; He, X.; Wang, Y.; and Wang, X. 2022a. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *SIGKDD*, 1717–1727.

Sun, X.; Cheng, H.; Li, J.; Liu, B.; and Guan, J. 2023. All in One: Multi-Task Prompting for Graph Neural Networks. In *SIGKDD*.

Sun, Y.; Deng, H.; Yang, Y.; Wang, C.; Xu, J.; Huang, R.; Cao, L.; Wang, Y.; and Chen, L. 2022b. Beyond homophily: structure-aware path aggregation graph neural network. In *IJCAI*, 2233–2240.

Sun, Y.; Wang, X.; Liu, Z.; Miller, J.; Efros, A. A.; and Hardt, M. 2019. Test-time training for out-of-distribution generalization. In *ICLR*.

Wang, B.; Guo, J.; Li, A.; Chen, Y.; and Li, H. 2021. Privacy-preserving representation learning on graphs: A mutual information perspective. In *SIGKDD*, 1667–1676.

Wu, Z.; Xiong, Y.; Yu, S. X.; and Lin, D. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 3733–3742.

Xia, J.; Zheng, J.; Tan, C.; Wang, G.; and Li, S. Z. 2022. Towards effective and generalizable fine-tuning for pre-trained molecular graph models. *bioRxiv*.

Xu, J.; Huang, R.; Jiang, X.; Cao, Y.; Yang, C.; Wang, C.; and Yang, Y. 2023. Better with Less: A Data-Centric Prespective on Pre-Training Graph Neural Networks. In *NeurIPS*.

Xu, J.; Yang, Y.; Chen, J.; Jiang, X.; Wang, C.; Lu, J.; and Sun, Y. 2022. Unsupervised adversarially robust representation learning on graphs. In *AAAI*, 4290–4298.

Xu, R.; Liu, P.; Wang, L.; Chen, C.; and Wang, J. 2020. Reliable weighted optimal transport for unsupervised domain adaptation. In *CVPR*, 4394–4403.

Xuhong, L.; Grandvalet, Y.; and Davoine, F. 2018. Explicit inductive bias for transfer learning with convolutional networks. In *ICML*, 2825–2834.

You, J.; Ying, Z.; and Leskovec, J. 2020. Design space for graph neural networks. In *NeurIPS*, 17009–17021.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph contrastive learning with augmentations. In *NeurIPS*, 5812–5823.

Zamir, A. R.; Sax, A.; ; Shen, W. B.; Guibas, L.; Malik, J.; and Savarese, S. 2018. Taskonomy: Disentangling Task Transfer Learning. In *CVPR*. IEEE.

Zhang, J.; Xiao, X.; Huang, L.-K.; Rong, Y.; and Bian, Y. 2022. Fine-tuning graph neural networks via graph topology induced optimal transport. *arXiv preprint arXiv:2203.10453*.

Zheng, Y.; Pan, S.; Lee, V.; Zheng, Y.; and Yu, P. S. 2022. Rethinking and scaling up graph contrastive learning: An extremely efficient approach with group discrimination. In *NeurIPS*, 10809–10820.