Unified Graph Neural Networks Pre-training for Multi-domain Graphs

Mingkai Lin, Xiaobin Hong, Wenzhong Li*, Sanglu Lu

State Key Laboratory for Novel Software Technology, Nanjing University
Nanjing, China
{mingkai,xiaobinhong}@smail.nju.edu.cn, {lwz,sanglu}@nju.edu.cn

Abstract

Graph Neural Networks (GNNs) have proven effective and typically benefit from pre-training on accessible graphs to enhance performance on tasks with limited labeled data. However, existing GNNs are constrained by the "one-domain-one-model" limitation, which restricts their effectiveness across diverse graph domains. In this paper, we tackle this problem by developing a method called Multi-Domain Pre-training for a Unified GNN Model (MDP-GNN). This method is based on the philosophical notion that *everything is interconnected*, suggesting that a latent meta-domain exists to encompass the diverse graph domains and their interconnections. MDP-GNN seeks to identify and utilize this meta-domain to train a unified GNN model through three core strategies. Firstly, it integrates node feature semantics from different domains to create unified representations. Secondly, it employs a bi-level learning strategy to build a domain-synthesized network that identifies latent connections to facilitate cross-domain knowledge transfer. Thirdly, it uses Wasserstein distance to map diverse domains into the common meta-domain for graph distribution alignment. We validate the effectiveness of MDP-GNN through theoretical analysis and extensive experiments on four real-world graph datasets, showing its superiority in enhancing GNN performance across diverse domains.

Introduction

Graph Neural Networks (GNNs) have proven highly effective for analyzing graph-structured data, which inherently contains intricate relational information (Kipf and Welling 2017; Shi et al. 2020; Lin et al. 2022). GNNs show substantial potential in various real-world networking systems (Wei et al. 2023; Zheng et al. 2023; Corso et al. 2024) for leveraging techniques such as neighborhood aggregation and message passing to generate comprehensive node embeddings for diverse machine learning tasks (Rossi et al. 2022; Chowdhury et al. 2023; Boll et al. 2024).

To reduce the requirement for extensive manual annotations and the costs associated with retraining models from scratch, graph pre-training has become a crucial approach for enabling knowledge transfer across various graph learning tasks. Conventional GNN pre-training has two processes: pre-training to leverage substantial amounts of unlabeled structural data to capture latent knowledge, and fine-tuning to apply pre-trained knowledge to specific downstream tasks. Over time, pre-training strategies have evolved significantly. Early methods were relatively simple (Hu et al. 2020b; Jin et al. 2020; Liu et al. 2022), while more recent approaches have become increasingly sophisticated (Veličković et al. 2018; Lu et al. 2021; Xia et al. 2022a). Moreover, recent advancements in large language models (LLMs) have led to visible innovations in graph learning. A key development is the introduction of prompt techniques, which guide models toward producing desired outputs. This has resulted in the emergence of graph prompting methods (Sun et al. 2023; Liu et al. 2023b; Fang et al. 2024; Zhao et al. 2024a), which standardize downstream tasks into a general template used as a pretext task. By tuning prompt tokens to align with pretext knowledge, these methods facilitate the adaptation of models to various tasks.

Another notable characteristic of LLMs is broad versatility across various domains. This versatility is challenging for existing GNN models, typically pre-trained and tuned on the same or closely related domains with similar feature spaces and graph distributions. While recent studies have integrated LLMs with GNNs by leveraging textual descriptions as a unifying medium for multi-domain graphs (Zhao et al. 2024b), these approaches are limited to text-attributed graphs, which are not always available. Thus, it is crucial for GNNs to leverage richer knowledge from heterogeneous, text-free graph domains, utilizing only latent feature vectors to develop general graph foundation models applicable across diverse downstream graph domains. Therefore, this paper explores a more comprehensive approach to *GNN multi-domain pre-training*. We focus on training GNN models using multi-domain graphs characterized by varying latent features and distributions (i.e., citation, social, and co-purchasing networks). These models aim to generalize across different downstream-domain graphs using only a limited number of labels. To achieve this, several significant challenges should be addressed.

(1) Heterogeneous domain discrepancy occurs because varying feature spaces and data distributions across different graph domains limit the effectiveness of a unified GNN model. **(2) Cross-domain knowledge extraction** is complex, as it requires capturing unique characteristics of individual graphs while integrating insights across diverse

*The corresponding author is Wenzhong Li (lwz@nju.edu.cn).

datasets. **(3) Theoretical performance analysis** is lacking, which is crucial for substantiating the capability of GNNs to generalize effectively across diverse domains in practice.

The persisting challenges highlight the necessity of our holistic framework, MDP-GNN, i.e., Multi-Domain Pre-training for a Unified GNN Model. Our approach is founded on the philosophical concept that "everything is interconnected," implying the existence of an underlying meta-domain that unifies the variety of graph domains. Different domains can be considered part of the same overarching domain due to their interconnections, and thus it is possible to learn a unified GNN model from them jointly. It is vital to uncover and leverage the information that illustrates the shared meta-domain where these diverse domains belong. MDP-GNN aims to discover and harness this meta-domain through three strategic approaches, as depicted in Fig. 1.

First, node features from various graph domains are unified through an integration approach, that concatenates the original features in a fixed position with an additional representation learned by a conversion function. This additional representation helps unify node semantic features across diverse domains and serves as a soft prompt during tuning. Second, for extracting graph structure information, MDP-GNN employs a bi-level training method to learn a domain-synthesized network, where parameterized pivot nodes are trained to predict interconnections for diverse domains. This approach not only captures the underlying relationships among various graph domains but also further forms a latent meta-domain. Lastly, MDP-GNN minimizes the Wasserstein distance to make the diverse graph distributions closer to the meta-domain. Aligning the distributions of multi-domain graphs ensures that diverse graph domains can be viewed and analyzed from a unified perspective. These strategies collectively enhance the generalization performance of the GNN model, enabling it to extract knowledge from multi-domain graphs effectively.

The contributions of our work are summarized as follows.

- We tackle the challenge of training a unified GNN model across multi-domain graphs characterized by disparate feature semantics and heterogeneous graph distributions.
- We present MDP-GNN, a novel framework designed to integrate node features from different graph domains, construct the domain-synthesized network to uncover the underlying meta-domain and employ Wasserstein distance for graph distribution alignment. It enhances GNN generalization ability across diverse graph domains.
- We demonstrate the effectiveness of MDP-GNN through theoretical validation and empirical experiments across four distinct real-world graph domains. Our framework significantly surpasses current state-of-the-art methods.

Related Works

Graph Neural Networks (GNNs) have become crucial in graph representation learning, with applications in different domains like social networks (Wei et al. 2023; Zhang et al. 2022; Yang et al. 2024), knowledge graphs (Liang et al. 2024; Tan et al. 2023), and recommendation systems (Yu et al. 2023; Zheng et al. 2023; Sharma et al. 2024). Effective GNN architectures include the Graph Convolutional Network (GCN) (Kipf and Welling 2017), GraphSAGE, Graph Attention Network (GAT) (Veličković et al. 2017), and Graph Isomorphism Network (GIN) (Xu et al. 2018). Additionally, Transformer-based GNN models like the Graph Transformer (GT) (Shi et al. 2020) and NodeFormer (Wu et al. 2022) have been further developed. These models have shown significant success in supervised tasks (Rossi et al. 2022; Wang et al. 2024).

To reduce annotation costs and alleviate the need for training from scratch, graph pre-training has emerged as a promising paradigm for the efficient training of GNN models (Xia et al. 2022b; Hu et al. 2020c; Lu et al. 2021; Lin et al. 2023). This paradigm acquires intrinsic graph knowledge from easily accessible graphs by methods, ranging from predicting masking edges and node attributes (Hu et al. 2020b; Jin et al. 2020) to more complex tasks like contrastive learning (Veličković et al. 2018; Xia et al. 2022a; Luo et al. 2023). The pre-trained GNN model is subsequently fine-tuned for specific downstream domains by updating the pre-trained weights. Inspired by the success of large language models (LLMs) (Zhao et al. 2023) in natural language processing (NLP) and computer vision (CV), similar strategies have been explored in graph learning. LLMs have demonstrated remarkable capabilities in language understanding and generation through prompt-based techniques (Zhu et al. 2023; Liu et al. 2023a; Sohn et al. 2023), which guide the model to produce desired outputs. This success has led to the exploration to enhance the GNN performance of downstream tasks through prompt-tuning (Sun et al. 2023; Liu et al. 2023b; Fang et al. 2024; Zhao et al. 2024a). These methods involve freezing the pre-trained GNN model and updating well-designed graph prompts.

However, these approaches are often limited to the same or closely related domains and lack broad versatility as the foundation models in LLMs across different domains. Recent works, such as OFA (Liu et al. 2024), GraphGPT (Tang et al. 2024a), and HiGPT (Tang et al. 2024b), utilized natural language to derive and align node features across various domains. However, they are restricted to specific text-attributed graphs, making them less effective for the more prevalent text-free graphs. Additionally, recent work (Zhao et al. 2024a) introduced GCOPE for cross-domain pre-training by adding fully connected nodes as coordinators to each domain and using Singular Value Decomposition (SVD) to align feature dimensions. However, GCOPE lacks theoretical validation and may lead to both positive and negative knowledge transfers, as well as semantic inconsistencies in node representations. Therefore, in this work, we propose a general framework, MDP-GNN, for pre-training unified GNNs in heterogeneous, text-free graph domains.

Problem Formulation

Graph pre-training is the process of training a GNN model using a predefined task. This training can be decomposed into a graph model $\text{GNN}(\mathcal{G})$ to extract latent embeddings from the graph $\mathcal{G}$, a task-specific auxiliary function f to predict the labels based on the embeddings and ℓ to be the loss function to measure the difference between the predictions

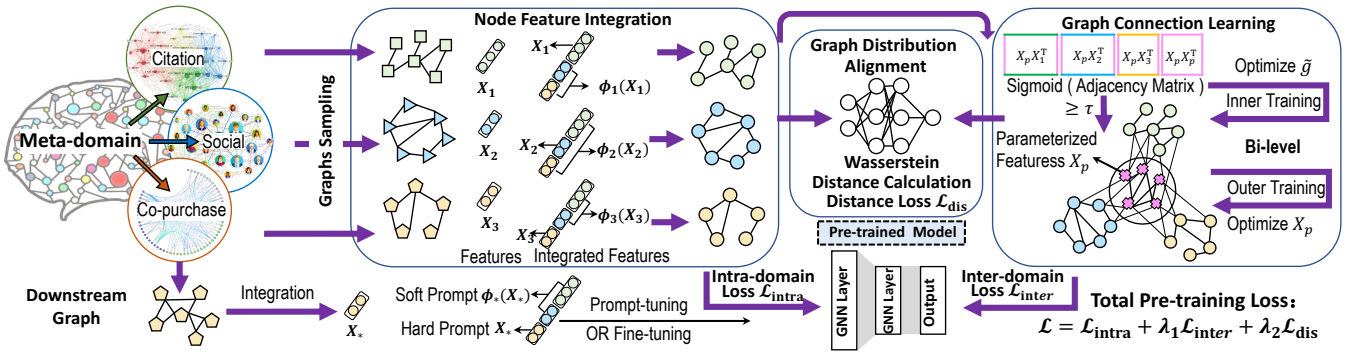


Figure 1: The framework of MDP-GNN for multi-domain pre-training.

and corresponding true labels $\mathcal{Y}$. Thus the entire training process can be formulated as:

$$\min \ \ell(f(\text{GNN}(\mathcal{G})), \mathcal{Y}) := \min g(\mathcal{G}, \mathcal{Y}) \tag{1}$$

In multi-domain pre-training, we have K different graph domains with heterogeneous features and label spaces. The empirical graphs of these domains are denoted as $\mathcal{G}_k (k = 1, 2, \ldots, K)$. Each graph $\mathcal{G}_k$ is represented as $\mathcal{G}_k = (X_k, A_k)$, where $X_k \in \mathbb{R}^{N_k \times d_k}$ represents the node features with space $\mathcal{X}^{d_k}$ for N_k nodes, and $A_k \in \mathbb{R}^{N_k \times N_k}$ is the adjacency matrix of the graph. The objective is to extract knowledge simultaneously from these heterogeneous graph domains into a unified GNN model, which can transfer the synthesized knowledge to a downstream task $\mathcal{G}_*$ through specific tuning approaches. Compared to the conventional single-domain pre-training process, the proposed multi-source pre-training can produce a unified GNN model for multiple graph domains with better generalization.

Methodology

The proposed MDP-GNN framework is illustrated in Fig. 1. It firstly uses *node feature integration* to align node feature space with the same semantic meanings so that they can be learned by a unified GNN model. Then it processes *graph connection learning* to identify the meta-domain with the domain-synthesized graph. Later, *graph distribution alignment* narrows the Wasserstein distribution distance between the multi-domain graphs and meta-domain. In this way, MDP-GNN can produce a unified GNN model with better generalization capability. The details are as follows.

Node Feature Integration The primary challenge in processing heterogeneous graph domains is the heterogeneity in node feature spaces. Typically, node features vary not only in dimension but also in semantic meaning, making it difficult for a single GNN to process them simultaneously. While feature reduction methods like Singular Value Decomposition (SVD) can adjust feature dimensions, they fail to address semantic mismatches and often lose original information. To tackle this issue, we propose a feature integration phase, where we assign a consistent global semantic meaning for each node across different pre-trained domains to generate new and unified features. Specifically, each domain retains its original node features X_k in a designated position, while the remaining positions are filled by converted features based on the conversion function $\phi_k(\cdot)$ on X_k. This approach ensures that all node features across domains are semantically aligned. Thus, for graph $\mathcal{G}_k$, the integrated node features $X'_k \in \mathbb{R}^{N_k \times D}$ with $D = \sum d_k$ are represented as:

$$X'_k = \left[\phi_k(X_k)[0 : \sum_{i=0}^{k-1} d_i], X_k, \phi_k(X_k)[\sum_{i=0}^{k+1} d_i : \sum_{i=0}^{K} d_i] \right]. \tag{2}$$

Here, $\phi_k(X_k)$ represents the converted features with shape $N_k \times (D - d_k)$. We also denote the whole process to transform graph $\mathcal{G}_k$ as $\phi_k(\mathcal{G}_k)$. The function $\phi_k(\cdot)$ ensures that features from different domains are mapped into a consistent semantic space, thereby resolving discrepancies in feature dimensions and meanings across domains through co-training with the respective real features. In this way, we can derive the intra-domain knowledge by loss $\mathcal{L}_{\text{intra}}$ from the intra-domain graphs $\{\phi_k(\mathcal{G}_k)\}$:

$$\mathcal{L}_{\text{intra}} = \sum_k g(\phi_k(\mathcal{G}_k), \mathcal{Y}_k). \tag{3}$$

Graph Connection Learning Identifying and leveraging inter-domain knowledge is essential for maintaining generalization across domains. This requires techniques to bridge the gaps between disparate graph datasets. This aligns with the meta-domain concept that encompasses distinct graphs and their interconnections. Therefore, recovering these connections is critical in the process.

To achieve this, we use a link prediction method to simulate connections. Ideally, this would describe links between every pair of nodes from different domains with a time complexity of $O((\sum_k N_k)^2)$, which is time-consuming. To reduce the complexity, we introduce N_p additional nodes with features X_p as trainable parameters to predict the connection to each domain node. These nodes serve as pivots among different domains to facilitate connections between them, thereby reducing the complexity to $O((\sum_k N_k)N_p)$. Thus the adjacency matrix between pivot nodes and original graphs is given by $\sigma(X_p X_k^T)(k = 1, 2, \ldots, K)$ where $\sigma(\cdot)$ is the Sigmoid function that represents link existence probabilities in $[0, 1]$. Additionally, the adjacency matrix between pivot nodes is represented as $\sigma(X_p X_p^T)$.

We denote the whole domain-synthesized netwotk as $\mathcal{M}(\{\phi_k(\mathcal{G}_k)\}, X_p)$, which is optimized using a bi-level training approach:

$$X_p^* = \underset{X_p}{\text{argmin}} \sum_k \tilde{g}^*(\phi_k(\mathcal{G}_k), \mathcal{Y}_k) + \tilde{g}^*(\mathcal{M}(\{\phi_k(\mathcal{G}_k)\}, X_p), \mathcal{Y}_k),$$

$$\text{where } \tilde{g}^* = \underset{\tilde{g}}{\operatorname{argmin}} \sum_k \tilde{g}(\phi_k(\mathcal{G}_k), \mathcal{Y}_k) + \tilde{g}(\mathcal{M}(\{\phi_k(\mathcal{G}_k)\}, X_p), \mathcal{Y}_k). \tag{4}$$

The primary objective is to optimize the loss for all graph pairs $\phi_k(\mathcal{G}_k)$ and $\mathcal{M}(\{\phi_k(\mathcal{G}_k)\}, X_p)$, corresponding to the label $\mathcal{Y}_k$. In the inner optimization loop, an initialized function $\tilde{g}$ is optimized to minimize the objective, resulting in an optimized function $\tilde{g}^*$. This optimization process is crucial as it fine-tunes the GNN model to identify commonalities based on the shared structures among the graphs $\{\phi_k(\mathcal{G}_k)\}$ and establishes a foundation to refine the synthesized graph for better model generalization capabilities. Notably, the function $\tilde{g}$ can be initialized as the current pre-trained GNN model. In the outer optimization loop, the objective is to optimize the parameter X_p, which directly influences the synthesized graph structure. The aim is to develop a more effective domain-synthesized network that mitigates the negative effects of inappropriate node connections. This goal is achieved by minimizing the objective with respect to the parameter X_p, using the optimal function $\tilde{g}^*$ obtained from the inner loop as a reference.

Through the bi-level optimization process, $\mathcal{M}(\cdot)$ is trained to extract an optimal domain-synthesized network $\mathcal{M}(\{\phi_k(\mathcal{G}_k)\}, X_p^*)$ that effectively captures the underlying structure of the input graphs. This optimization enhances the overall performance and generalization capability during pre-training. Moreover, the cross-domain adjacency matrices, which represent edges with at least one endpoint belonging to pivot nodes, have elements ranging between $[0, 1]$. These matrices are all filled out, so they are still computationally expensive and impractical to handle. Thus we introduce sparsity by masking elements below a non-negative threshold τ, thereby refining the domain-synthesized network to $\mathcal{G}_\mathcal{M}$. In this way, we can derive the inter-domain knowledge by pre-training the resulting graph $\mathcal{G}_\mathcal{M}$ for loss $\mathcal{L}_{\text{inter}}$:

$$\mathcal{L}_{\text{inter}} = \sum_k g(\mathcal{G}_\mathcal{M}, \mathcal{Y}_k). \tag{5}$$

Graph Distribution Alignment The distribution of graph data is critical for understanding the unique characteristics of different graph domains. Standardizing these diverse graph distributions to a meta-domain enables GNNs to more effectively learn common patterns and knowledge across various domains, thereby improving performance and robustness. To achieve this, we employ Wasserstein distance, specifically designed to measure the distance between two discrete distributions. Formally, the Wasserstein distance with the first moment is defined as:

$$W_1(\mathbb{P}_{\text{src}}, \mathbb{P}_{\text{tgt}}) = \sup_{\|w\|_{Lip} \le 1} \mathbb{E}_{x \sim \mathbb{P}_{\text{src}}}[w(x)] - \mathbb{E}_{x \sim \mathbb{P}_{\text{tgt}}}[w(x)], \tag{6}$$

where $\mathbb{P}_{\text{src}}, \mathbb{P}_{\text{tgt}}$ are the real distributions for source and target domains respectively, and $\|w\|_{Lip} \le 1$ is the Lipschitz norm of w. The equation indicates the existence of an optimal 1-Lipschitz function w that separates $\mathbb{P}_{\text{src}}$ and $\mathbb{P}_{\text{tgt}}$, with its maximum expectation representing the first Wasserstein distance. A trained neural network w^* can serve as a Wasserstein discriminator to fit such a 1-Lipschitz function, taking one sample as input and outputs a real number. The discriminator can be obtained by maximizing the formula $\mathbb{E}_{x \sim \mathbb{P}_{\text{src}}}[w(x)] - \mathbb{E}_{x \sim \mathbb{P}_{\text{tgt}}}[w(x)]$, where the expectation $\mathbb{E}(\cdot)$ can be approximated by the empirical average value.

This phase intends to minimize the Wasserstein distance between $\phi_k(\mathcal{G}_k)$ and $\mathcal{G}_\mathcal{M}$. In this paper, we can utilize a two-layer GCN followed by a multilayer perceptron (MLP) to serve as the discriminator w according to (You et al. 2023). It uses graph data as input and outputs a real number for each node instance. In this way, the distance can be achieved by minimizing the distribution distance loss $\mathcal{L}_{\text{dis}}$:

$$\begin{aligned} \mathcal{L}_{\text{dis}} &= \sum_k W_1(\phi_k(\mathcal{G}_k), \mathcal{G}_\mathcal{M}) = \sum_k [\,\overline{w^*(\phi_k(\mathcal{G}_k))} - \overline{w^*(\mathcal{G}_\mathcal{M})}], \\ \text{where } w^* &= \underset{w}{\operatorname{argmax}} \sum_k [\,\overline{w(\phi_k(\mathcal{G}_k))} - \overline{w(\mathcal{G}_\mathcal{M})}]. \end{aligned} \tag{7}$$

It quantifies the graph distribution distance between the multi-domain graphs and the meta-domain graph. Minimizing $\mathcal{L}_{\text{dis}}$ can also be viewed as a bi-level process. Firstly, the distance function w can be maximized to the supremum to obtain w^*, which represents the Wasserstein distance of the graph distributions. Secondly, $\mathcal{L}_{\text{dis}}$ is then minimized by freezing w^* and optimizing $\{\phi_k\}$ so that the distribution distance can be narrowed down.

The Overall Process Combining above three phases, with the hyperparameters λ_1, λ_2, the total loss to optimize is:

$$\mathcal{L} = \mathcal{L}_{\text{intra}} + \lambda_1 \mathcal{L}_{\text{inter}} + \lambda_2 \mathcal{L}_{\text{dis}}. \tag{8}$$

[Training] Given that most graph domains are massive in knowledge, we adopt a graph sampling approach for training to ensure efficiency. The overall training process is in Alg. 1.

Algorithm 1: Pre-training Process for MDP-GNN

Input: Heterogeneous K graph domains; Pivot nodes number N_p;Sparsity parameter τ; Loss parameters λ_1, λ_2
Output: Pre-trained GNN model

```
Initialize functions {φ_k}, g and trainable embedding X_p
while not converged do
    Sample subgraphs {G_k} from source domains
    Integrate node features for {G_k} with {φ_k} [Eq.(2)]
    Calculate pre-training loss L_intra for intra-domain graphs {φ_k(G_k)} [Eq.(3)]
    Bi-level training to optimize X_p [Eq.(4)]
    Construct the graph G_M based on optimized X_p
    Calculate pre-training loss L_inter for the inter-domain graph G_M [Eq.(5)]
    Compute the Wasserstein distance loss L_dis [Eq.(7)]
    Compute the total loss L [Eq.(8)]
    Backpropagate L to update {φ_k}, g
end while
return Pre-trained GNN model
```

[Tuning] The returned GNN model is designed to transfer the extracted knowledge to downstream tasks flexibly. For graphs from domains encountered during the pre-training stage, we process feature integration on the downstream graph by placing the original node feature X_* in its position and converted features $\phi_*(X_*)$ in the remaining positions. For fine-tuning, ϕ_* can be jointly optimized with the pre-trained GNN model. For prompt-tuning, the integrated features act as a combination of a hard prompt X_* to identify

the domain and a soft prompt $\phi_*(X_*)$ to adapt the knowledge. The pre-trained GNN model can also be applied to unseen domains by using a vanilla feature converting function for fine-tuning and prompt-tuning.

Theoritical Analysis

The data in each graph domain $\mathcal{G}_k(k = 1, 2, \cdots, K)$ can be viewed as i.i.d samples with their labels $\{(G_n^k, Y_n^k)\}_{n=1}^{N_k} \sim \mathcal{G}_k$ (Sebag et al. 2019; Verma and Zhang 2019; Zhu et al. 2021). Therefore, the conversion methods (SVD or feature integration w.r.t $\{\phi_k\}$) convert them into $\{(\tilde{G}_n^k, Y_n^k)\}_{n=1}^{N_k} \sim \tilde{\mathcal{G}}_k$ with unified feature space $\mathcal{X}^D$. Without loss of generalizability, we consider a binary classification task to represent the objective to ease analysis, where each graph domain $\mathcal{G}_k$ corresponds to a unified label domain $\mathcal{Y} = [0, 1]$. Let $\mathcal{H}$ be a hypothesis space, where $h \in \mathcal{H}$ predicts the relation between converted graph domain $\tilde{\mathcal{G}}$ and the label space $\mathcal{Y}$, namely $h = GNN \circ f : \tilde{\mathcal{G}} \mapsto \mathcal{Y}$. For each h, we define the risk under distribution $\tilde{\mathcal{G}}_k$ as $\epsilon_k(h) = \mathbb{P}_{\tilde{G},Y\sim\tilde{\mathcal{G}}_k}(h(\tilde{G}) \neq Y)$. h_k^* denotes the oracle hypothesis with minimal total risk according to $\tilde{\mathcal{G}}_k$. We can derive the following theorems.

Theorem 1 *Suppose function h is C-Lipschitz, constrained by some distance measure. We consider K converted graph distributions $\{\tilde{\mathcal{G}}_k\}$ over $\mathcal{X}^D \times [0; 1]$ and $\mathcal{H} := \{h : \tilde{\mathcal{G}} \mapsto \mathcal{Y}\}$ as the set of bounded real-valued functions with the pseudo-dimension $Pdim(\mathcal{H}) = d$. Let α and γ be in the simplex of dimension K. If $\tilde{S}$ is a set of size N_S which respectively contains $\gamma_k N_S = N_k$ samples from $\tilde{\mathcal{G}}_k$, and $\hat{h}$ is the empirical minimizer of $\sum_k \alpha_k \hat{\epsilon}_k(h)$. Then for any $\delta > 0$, with probability at least $1 - \delta$, the compound empirical error on heterogeneous domains is upper bounded as:*

$$\sum_k \epsilon_k(\hat{h}) \leq \sum_k \epsilon(h_k^*) + 2KR(\alpha, \gamma) + 2\sum_k \sum_t \alpha_k (2CW_1(\tilde{\mathcal{G}}_k, \tilde{\mathcal{G}}_t) + \lambda_{kt}),$$

where $\lambda_{kt} = \min_{h\in\mathcal{H}}(\epsilon_k(h) + \epsilon_t(h))$ and

$$R(\alpha, \gamma) = \sqrt{\frac{2d\sum_k \frac{\alpha_k^2}{\gamma_k} \log \frac{2}{\delta}}{N_S}} + 2\sqrt{\frac{d}{N_S}} \sum_k \frac{\alpha_k}{n\gamma_k\sqrt{\gamma_k}}.$$

Theorem 2 *Suppose function h is C-Lipschitz, constrained by some distance measure. We consider K converted graph distributions $\tilde{\mathcal{G}}_k$ over $\mathcal{X}^D \times [0; 1]$ and $\mathcal{H} := \{h : \mathcal{G} \mapsto \mathcal{Y}\}$ as the set of bounded real-valued functions with the pseudo-dimension $Pdim(\mathcal{H}) = d$. Let α and γ be in the simplex of dimension K. Assume $\mathcal{G}_\mathcal{M}$ is the meta domain encompassing distributions $\{\tilde{\mathcal{G}}_k\}$ with N_S samples drawn from it, where $\gamma_k N_S$ samples are associated with the distribution $\tilde{\mathcal{G}}_k$. Define $\hat{h}_\mathcal{M}$ is the empirical minimizer of empirical risks $\sum_k \alpha_k \hat{\epsilon}_k^\mathcal{M}(h)$, where $\hat{\epsilon}_k^\mathcal{M}(h)$ is the empirical risk for the induced samples from $\mathcal{G}_\mathcal{M}$ associated with $\tilde{\mathcal{G}}_k$, then for any $\delta > 0$, with at least $1 - \delta$ probability the compound empirical error on the meta domain is upper bounded as:*

$$\sum_k \epsilon_k(\hat{h}_\mathcal{M}) \leq \sum_k \epsilon(h_k^*) + 2KR(\alpha, \gamma) + 4C\sum_k W_1(\tilde{\mathcal{G}}_\mathcal{M}, \tilde{\mathcal{G}}_k) + 2\sum_k \sum_t \alpha_k \lambda_{kt}^\mathcal{M},$$

where $\lambda_{kt}^\mathcal{M} = \min_{h\in\mathcal{H}}(\epsilon_k^\mathcal{M}(h) + \epsilon_t(h))$ and

$$R(\alpha, \gamma) = \sqrt{\frac{2d\sum_k \frac{\alpha_k^2}{\gamma_k} \log \frac{2}{\delta}}{N_S}} + 2\sqrt{\frac{d}{N_S}} \sum_k \frac{\alpha_k}{n\gamma_k\sqrt{\gamma_k}}.$$

The proofs of the theorems are included in Appendix A. Theorems 1 and 2 provide the compound empirical errors associated with training on converted graphs and meta-graphs for intra-domain and inter-domain knowledge, respectively, thereby offering insights into the generalization capabilities.

(1) The empirical errors in both theorems are influenced by the Wasserstein distance W_1. Thus in the MDP-GNN framework, we specifically design the module to align graph distributions between multi-domain and meta-domain graphs. It minimizes the Wasserstein distance, thereby improving the bounds of the empirical errors.

(2) The node feature integration method is more effective than traditional methods like SVD and MLP. SVD merely aligns the dimensionality of the feature vectors but ignores semantic mismatches between domains, which struggles to reduce the Wasserstein distance W_1. Although MLP can be trained to mitigate this issue, it may still lose specific domain information during conversion, which can negatively impact terms such as $\epsilon(h_k^*)$, λ_{kt}, and $\lambda_{kt}^\mathcal{M}$ in the bound.

(3) The bound in Theorem 2 can be further enhanced by reducing the term $\lambda_{kt}^\mathcal{M}$, which is influenced by the function $\mathcal{M}(\cdot)$ responsible for constructing a well-connected domain-synthesized network. The objective of minimizing $\lambda_{kt}^\mathcal{M}$, first identifying the optimal function h and then further optimizing $\mathcal{M}(\cdot)$, aligns with the proposed bi-level graph connection learning. Thus, MDP-GNN can effectively reduce the term, thereby improving the generalization bound.

Experiments

Experimental Settings

[Datasets] We evaluate MDP-GNN using four large-scale text-free graphs from distinct domains: (1) *Academic* (Hu et al. 2020a), a citation network with papers indexed by MAG (Wang et al. 2020); (2) *Product* (Hu et al. 2020a), an Amazon product co-purchasing network; (3) *Reddit* (Hamilton, Ying, and Leskovec 2017), a comment graph derived from Reddit; and (4) *Yelp* (Zeng et al. 2019), a social network formed from the Yelp platform. Detailed information about these datasets is provided in Tab. 1. For pre-training, we use the full unlabeled graphs of the heterogeneous domains. For tuning, we randomly extract ten different graphs of 5000 nodes from the respective domains as downstream graphs. The setup allows us to assess the generalization and performance across diverse domains comprehensively.

Dataset	Domain	Nodes	Edges	Features	Classes
Academic	Citation	169,343	1,166,243	128	40
Product	Co-purchase	2,449,029	61,859,140	100	47
Reddit	Comment	232,965	11,606,919	602	41
Yelp	Social	716,847	6,977,410	300	100(m)

Table 1: Statistics on multi-domain networks.

[Baseline Algorithms] We use twelve baselines for comparison, including the state-of-the-arts from three categories.

- *Graph supervised learning methods*, (1) **GIN** (Xu et al. 2018), (2) **GAT** (Veličković et al. 2017), (3) **GraphSage** (Hamilton, Ying, and Leskovec 2017), (4) **GT** (Shi et al. 2020). All of them are trained in an end-to-end manner.

	Methods	5% Labels Known								10% Labels Known							
		Academic		Product		Reddit		Yelp		Academic		Product		Reddit		Yelp	
		F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
Sup. Training	GIN	36.38	88.69	46.00	86.32	68.64	96.88	53.13	75.43	49.17	91.59	56.71	91.25	78.17	98.48	54.22	76.97
	GAT	42.96	88.13	47.97	85.70	55.20	88.11	47.15	68.96	54.80	91.08	58.18	89.42	70.49	94.33	48.60	72.41
	GraphSAGE	38.78	90.17	47.99	87.86	61.13	95.71	57.91	82.20	51.17	92.45	58.06	91.73	74.99	97.94	59.84	84.65
	GT	37.56	88.09	44.76	84.92	44.90	88.47	54.59	78.29	48.08	90.67	54.43	89.53	59.23	94.03	57.07	81.61
Self-sup. Training	EdgePred	39.68	89.83	43.38	85.54	45.40	91.66	55.72	81.26	52.17	92.66	55.51	90.95	60.97	95.76	57.92	83.87
	AttrMask	42.52	89.97	45.80	86.18	50.53	92.91	56.49	81.84	54.69	93.01	56.22	91.17	68.38	96.76	58.67	84.23
	DGI	39.41	89.70	43.46	85.56	43.92	90.91	56.35	81.72	51.92	92.41	55.33	90.98	62.40	95.87	58.47	84.06
	SimGRACE	40.84	89.76	43.07	85.32	44.03	90.79	55.95	81.35	52.87	92.57	54.93	90.88	60.15	95.23	58.28	83.93
Prompt Methods	GraphPrompt	18.35	65.38	14.04	63.12	19.95	69.38	26.80	58.68	18.95	69.10	17.91	68.10	24.56	73.41	25.85	58.92
	ProG	37.46	77.08	20.77	67.90	35.74	81.86	26.84	58.79	45.43	79.35	28.45	75.32	40.49	85.46	25.90	58.79
	GPF	21.42	79.38	22.59	74.68	19.79	75.51	53.39	78.67	31.83	82.92	28.64	80.65	24.79	80.81	54.58	79.97
	GCOPE	43.52	87.71	42.82	82.08	56.30	90.95	52.75	77.53	54.31	91.27	55.24	88.72	68.23	94.80	52.55	77.82
MDP+PT	EdgePred	43.23	91.55	52.14	**88.86**	74.24	97.62	59.00	**83.63**	54.14	93.78	64.05	93.02	83.18	98.90	60.38	**85.31**
	AttrMask	**46.22**	**92.33**	51.97	88.50	**82.77**	**98.14**	56.04	80.44	56.45	93.82	64.37	93.09	**89.24**	**99.07**	57.65	82.15
	DGI	43.00	91.05	51.20	87.83	67.32	96.60	58.11	82.68	54.56	93.39	**64.46**	**93.28**	81.66	98.69	59.09	84.54
	SimGRACE	44.95	90.69	51.34	88.21	65.44	96.33	56.96	81.55	55.35	93.13	63.73	93.10	80.12	98.57	58.41	83.33
MDP+FT	EdgePred	46.16	91.72	50.79	87.34	62.45	95.75	**59.02**	83.43	**56.99**	93.77	60.84	91.71	75.56	98.03	**60.44**	85.15
	AttrMask	45.69	92.14	50.81	87.91	62.41	95.58	58.44	82.89	56.05	**93.91**	61.97	92.28	75.77	97.95	59.94	84.73
	DGI	44.94	91.51	**52.16**	88.10	56.81	94.51	58.24	82.61	56.02	93.63	61.88	92.23	71.82	97.50	60.16	84.70
	SimGRACE	44.45	91.35	51.22	87.48	55.75	93.82	57.97	82.40	56.53	93.29	60.89	91.72	70.76	97.29	59.41	84.25

Table 2: The results (%) for different methods (The bolded results are the best, the underlined results are the second best.).

- *Graph self-supervised learning methods.* (5) **EdgePred** (Hu et al. 2020b) and (6) **AttrMask** (Jin et al. 2020) randomly masked partial edges or node attributes and then trained GNNs to predict them. (7) **DGI** (Veličković et al. 2018) and (8) **SimGRACE** (Xia et al. 2022a) leveraged contrastive learning to capture the latent graph properties.
- *Graph prompt learning methods.* (9) **GraphPrompt** (Liu et al. 2023b), (10) **ProG** (Sun et al. 2023), (11) **GPF** (Fang et al. 2024), (12) **GCOPE** (Zhao et al. 2024a). They can adapt the pretext knowledge with specifically designed learnable prompt tokens.

[Default settings] In this work, a two-layer GT serves as the GNN backbone for MDP-GNN, pre-trained with EdgePred, AttrMask, DGI and SimGRACE respectively. We focus on node classification during tuning, using AUC-ROC and F1-score as evaluation metrics. For the testing graphs, we allocate up to 10% of the known labels for training and another 10% for validation. We tune all the models for 1000 epochs using the Adam optimizer, with a learning rate of 0.005. Each experiment is conducted five times, and the mean results are reported. The conversion functions $\{\phi_k\}$ for feature integration are implemented with MLPs. For the bi-level connection learning of Eq. 4, the inner and outer loops are both set to 10 with a learning rate of 0.001, as well as for the inner step of in Eq. 6. The rest hyper-parameters are set as $\tau = 0.8, \lambda_1 = 1, \lambda_2 = 0.01, N_p = 256$. As for baselines, the backbones are the same as MDP-GNN. Since the graph prompting methods have their own prediction functions, we made no modifications. For baseline algorithms that cannot handle graphs with different feature dimensions, we employ an autoencoder (Hinton and Salakhutdinov 2006) to standardize feature dimensions with minimum information loss.

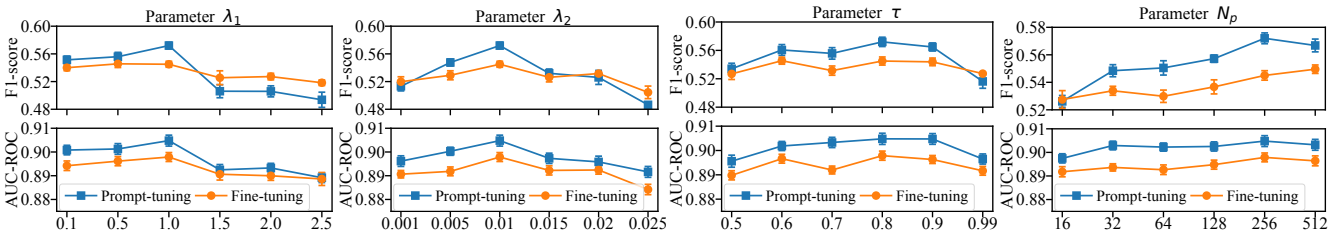
Numerical Results We report the results of baselines and MDP-GNN with prompt-tuning (PT) and fine-tuning (FT).

[Framework Performances] In Tab.2, we present the results tuning in pre-trained domains, with 5% and 10% known labels. MDP-GNN consistently demonstrates superior performance by achieving the best and second-best results, which lie in unified training and prompt mechanisms to adapt pretext knowledge. Notably, prompt-tuning proves more effective than fine-tuning. Tasks of edge prediction and attribute masking are more effective in leveraging heterogeneous pretext knowledge. In contrast, if not well pre-trained in heterogeneous domains, GT with self-supervised training methods performs worse than vanilla supervised models. Prompt learning methods also generally underperform because the pre-training struggles to capture common patterns across heterogeneous domains and they typically freeze the poorly pre-trained GNN parameters. GCOPE shows relatively better results but remains limited when solving domain discrepancies and knowledge extraction.

	Methods	Academic		Product		Reddit		Yelp	
		F1	AUC	F1	AUC	F1	AUC	F1	AUC
Prompt-t.	w/o Fea. Int.	34.05	88.94	42.68	85.58	55.23	94.48	56.97	83.35
	w/o $\mathcal{L}_{\text{intra}}$	39.71	90.54	46.16	87.00	45.37	93.07	56.88	82.01
	w/o $\mathcal{L}_{\text{inter}}$	42.51	91.47	50.50	88.33	68.60	96.83	58.94	83.70
	w/o $\mathcal{L}_{\text{dis}}$	39.95	90.94	50.88	88.32	56.62	95.34	57.83	**83.87**
	MDP-GNN	**43.23**	**91.55**	**52.14**	**88.86**	**74.24**	**97.62**	**59.00**	83.63
Fine-t.	w/o Fea. Int.	39.80	88.52	42.86	84.93	53.22	93.42	57.79	83.05
	w/o $\mathcal{L}_{\text{intra}}$	41.35	90.26	48.62	86.85	48.98	92.56	56.59	80.41
	w/o $\mathcal{L}_{\text{inter}}$	46.06	91.35	50.77	87.66	60.72	95.44	58.46	82.83
	w/o $\mathcal{L}_{\text{dis}}$	46.10	91.42	50.43	**87.97**	51.36	93.45	59.00	83.11
	MDP-GNN	**46.16**	**91.72**	**50.79**	87.34	**62.45**	**95.75**	**59.02**	**83.43**

Table 3: The ablation results (%) for different occasions

[Ablation Study] We evaluate MDP-GNN when removing four important components: (1) Feature Integration; (2) Intra-domain Knowledge; (3) Inter-domain Knowledge; (4) Distribution Alignment. We show the results in Tab.3, with EdgePred for pre-training and 5% known labels for tuning. We find that without these components, the performances of MDP-GNN are mostly weakened to some extent except AUC-ROC when prompt-tuning in Yelp and fine-tuning in Product. The most visible drop occurs without feature integration because we utilize commonly employed SVD instead, which cannot match feature semantic space in diverse domains and loses the capacity to align graph distri-



(a) Parameter λ_1 (b) Parameter λ_2 (c) Parameter τ (d) Parameter N_p

Figure 2: The average performance of EdgePred based MDP-GNN with 5% known labels, as $\lambda_1, \lambda_2, \tau$ and N_p vary

	Methods	5% Labels Known								10% Labels Known							
		Academic		Product		Reddit		Yelp		Academic		Product		Reddit		Yelp	
		F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
Sup. Training	GIN	36.38	88.69	46.00	86.32	68.64	96.88	53.13	75.43	49.17	91.59	56.71	91.25	78.17	98.48	54.22	76.97
	GAT	42.96	88.13	47.97	85.70	55.20	88.11	47.15	68.96	54.80	91.08	58.18	89.42	70.49	94.33	48.60	72.41
	GraphSAGE	38.78	90.17	47.99	87.86	61.13	95.71	57.91	82.20	51.17	92.45	58.06	91.73	74.99	97.94	59.84	84.65
	GT	37.56	88.09	44.76	84.92	44.90	88.47	54.59	78.29	48.08	90.67	54.43	89.53	59.23	94.03	57.07	81.61
Self-sup. Training	EdgePred	40.90	90.01	44.56	86.06	51.36	92.90	56.62	81.93	53.37	92.64	55.89	91.03	67.12	96.55	58.77	84.29
	AttrMask	42.77	89.99	44.01	86.22	53.91	93.74	56.68	81.88	52.96	92.66	55.28	90.85	69.90	96.98	58.70	84.22
	DGI	39.78	89.60	44.31	85.96	48.22	92.00	56.43	81.73	52.91	92.66	56.29	91.17	65.95	96.41	58.55	84.14
	SimGRACE	38.82	89.40	43.52	85.70	48.17	92.31	56.28	81.59	52.26	92.62	55.87	91.18	64.91	96.35	58.63	84.24
Prompt Methods	GraphPrompt	16.72	65.28	20.25	67.83	17.64	66.62	26.80	58.59	20.99	70.04	23.98	74.37	18.95	69.58	25.85	58.73
	ProG	35.92	65.37	19.64	56.69	18.46	60.31	26.80	58.24	39.85	65.29	21.48	59.27	19.70	62.17	25.85	58.17
	GPF	22.59	83.17	24.48	75.10	28.54	82.10	53.56	78.22	34.49	85.19	34.35	81.96	35.30	87.05	55.23	80.01
	GCOPE	41.95	82.43	36.21	80.41	59.55	92.46	53.13	73.80	51.10	84.86	46.63	86.78	66.76	95.29	54.22	75.79
MDP+PT	EdgePred	41.78	91.10	50.66	88.39	71.23	97.46	58.57	83.94	55.75	93.84	62.36	93.00	84.99	99.05	60.16	85.52
	AttrMask	46.29	91.99	51.83	88.52	**80.79**	97.93	54.38	79.11	56.27	94.02	64.20	93.21	90.13	99.27	56.05	80.45
	DGI	38.64	88.33	48.98	87.25	50.79	93.73	57.59	82.83	51.18	92.31	63.95	93.51	79.73	98.65	59.08	84.93
	SimGRACE	42.35	89.93	51.11	88.29	66.90	96.34	56.73	81.10	54.35	92.87	65.00	93.19	79.35	98.43	58.31	82.59
MDP+FT	EdgePred	47.11	91.88	54.58	89.43	74.41	97.91	**60.42**	**85.06**	58.65	94.11	67.07	**94.15**	90.04	99.36	**61.65**	**86.45**
	AttrMask	**47.98**	**92.33**	**55.51**	**89.82**	79.50	**98.38**	60.37	84.80	59.38	**94.25**	**67.45**	94.14	**90.97**	**99.38**	61.43	86.31
	DGI	45.72	91.46	54.52	89.37	71.88	97.73	60.12	84.94	59.26	93.56	66.65	93.90	88.42	99.27	61.48	86.44
	SimGRACE	46.36	91.54	54.72	89.21	73.16	97.67	59.94	84.52	**59.95**	94.01	66.51	94.13	88.70	99.26	60.30	86.11

Table 4: The results (%) for unseen domains (The bolded results are the best, the underlined results are the second best.).

bution. Overall, each component is necessary to help MDP-GNN improve its performance jointly.

[Parameter Analysis] We explore the effects of the parameters λ_1, λ_2, τ, and N_p in Fig.2. We observe that $\lambda_1 = 1$ optimally balances the loss term $\mathcal{L}_{\text{inter}}$, crucial for effective cross-domain knowledge extraction. For λ_2, which regulates domain distribution alignment, the best results are achieved at 0.01. And the values outside this range negatively impact performances. The parameter τ, which controls the sparsity in the domain-synthesized graph, shows optimal performance at 0.8. Lower values cannot capture enough structural information, while higher values introduce noise. Lastly, the number of pivot nodes N_p is critical for modeling the synthesized graph. Increasing N_p generally improves performance. However, we found that the gains plateau when $N_p > 256$, suggesting 256 is optimal between performance and computational efficiency. These carefully tuned parameters jointly enhance MDP-GNN to generalize across various graph domains, ensuring robust and effective learning.

[Unseen Domain Tuning] We show the results of MDP-GNN when tuning in unseen graph domains using a leave-one-out protocol. In this setup, three domains are selected for pre-training, and the remaining domain is used for tuning. The results, presented in Tab. 4, indicate that MDP-GNN consistently outperforms baseline methods, highlighting its robust generalization ability across both seen and unseen domains. As observed in Tab. 2, EdgePred and AttrMask are the most effective pre-training tasks. Notably, in the context of node classification for unseen domains, fine-tuning is specifically recommended, because its results significantly outperform prompt-tuning ones. This likely occurs because fine-tuning allows the model to incorporate domain-specific knowledge, whereas prompt-tuning has a limited capacity to modify the learned pretext knowledge.

Conclusion

This paper introduces the MDP-GNN framework, designed to overcome the traditional GNN limitation of "one-domain-one-model". MDP-GNN capitalizes on the interconnected nature of data to identify a latent meta-domain, enabling effective pre-training across multiple graph domains. The framework integrates node feature semantics from various domains, constructs a domain-synthesized network using a bi-level learning strategy, and applies the Wasserstein distance to align diverse graph distributions. These strategies jointly enhance the GNN capacity for generalization and knowledge transfer across diverse graph domains. Theoretical validation and empirical experiments demonstrate that MDP-GNN significantly outperforms existing state-of-the-art methods, achieving notable improvements.

Acknowledgments

This work was partially supported by the Natural Science Foundation of Jiangsu Province (Grant No. BK20222003), the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Sino-German Institutes of Social Computing. The corresponding author is Wenzhong Li (lwz@nju.edu.cn).

References

Boll, H. O.; Amirahmadi, A.; Ghazani, M. M.; de Morais, W. O.; de Freitas, E. P.; Soliman, A.; Etminani, K.; Byttner, S.; and Recamonde-Mendoza, M. 2024. Graph neural networks for clinical risk prediction based on electronic health records: A survey. *Journal of Biomedical Informatics*, 104616.

Chowdhury, A.; Srinivasan, S.; Mukherjee, A.; Bhowmick, S.; and Ghosh, K. 2023. Improving Node Classification Accuracy of GNN through Input and Output Intervention. *ACM Transactions on Knowledge Discovery from Data (TKDD 2023)*, 18(1): 1–31.

Corso, G.; Stark, H.; Jegelka, S.; Jaakkola, T.; and Barzilay, R. 2024. Graph neural networks. *Nature Reviews Methods Primers*, 4(1): 17.

Fang, T.; Zhang, Y.; Yang, Y.; Wang, C.; and Chen, L. 2024. Universal prompt tuning for graph neural networks. In *Advances in Neural Information Processing Systems (NIPS 2024)*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems (NIPS 2017)*, 30.

Hinton, G. E.; and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786): 504–507.

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. *Advances in neural information processing systems (NIPS 2020)*, 33: 22118–22133.

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

Hu, Z.; Dong, Y.; Wang, K.; Chang, K.-W.; and Sun, Y. 2020c. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2020)*, 1857–1867.

Jin, W.; Derr, T.; Liu, H.; Wang, Y.; Wang, S.; Liu, Z.; and Tang, J. 2020. Self-supervised learning on graphs: Deep insights and new direction. In *International Conference on Learning Representations (ICLR 2020)*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of International Conference on Learning Representations (ICLR 2017)*.

Liang, K.; Meng, L.; Liu, M.; Liu, Y.; Tu, W.; Wang, S.; Zhou, S.; Liu, X.; Sun, F.; and He, K. 2024. A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Lin, M.; Li, W.; Li, D.; Chen, Y.; Li, G.; and Lu, S. 2023. Multi-domain generalized graph meta learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2023)*, 4479–4487.

Lin, M.; Li, W.; Li, D.; Chen, Y.; and Lu, S. 2022. Resource-efficient training for large graph convolutional networks with label-centric cumulative sampling. In *Proceedings of the ACM Web Conference (WWW 2022)*, 1170–1180.

Liu, H.; Feng, J.; Kong, L.; Liang, N.; Tao, D.; Chen, Y.; and Zhang, M. 2024. One For All: Towards Training One Graph Model For All Classification Tasks. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*.

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. *ACM Computing Surveys*, 55(9): 1–35.

Liu, Y.; Jin, M.; Pan, S.; Zhou, C.; Zheng, Y.; Xia, F.; and Philip, S. Y. 2022. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE 2022)*, 35(6): 5879–5900.

Liu, Z.; Yu, X.; Fang, Y.; and Zhang, X. 2023b. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference (WWW 2023)*, 417–428.

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 (AAAI 2021)*, 4276–4284.

Luo, X.; Ju, W.; Gu, Y.; Mao, Z.; Liu, L.; Yuan, Y.; and Zhang, M. 2023. Self-supervised graph-level representation learning with adversarial contrastive learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 18(2): 1–23.

Rossi, A.; Firmani, D.; Merialdo, P.; and Teofili, T. 2022. Explaining Link Prediction Systems based on Knowledge Graph Embeddings. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD 2022)*, 2062–2075.

Sebag, A. S.; Heinrich, L.; Schoenauer, M.; Sebag, M.; Wu, L.; and Altschuler, S. 2019. Multi-Domain Adversarial Learning. In *Seventh annual International Conference on Learning Representations (ICLR 2019)*.

Sharma, K.; Lee, Y.-C.; Nambi, S.; Salian, A.; Shah, S.; Kim, S.-W.; and Kumar, S. 2024. A survey of graph neural networks for social recommender systems. *ACM Computing Surveys*, 56(10): 1–34.

Shi, Y.; Huang, Z.; Feng, S.; Zhong, H.; Wang, W.; and Sun, Y. 2020. Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2020)*.

Sohn, K.; Chang, H.; Lezama, J.; Polania, L.; Zhang, H.; Hao, Y.; Essa, I.; and Jiang, L. 2023. Visual prompt tuning for generative transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2023)*, 19840–19851.

Sun, X.; Cheng, H.; Li, J.; Liu, B.; and Guan, J. 2023. All in One: Multi-Task Prompting for Graph Neural Networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2023)*.

Tan, Z.; Chen, Z.; Feng, S.; Zhang, Q.; Zheng, Q.; Li, J.; and Luo, M. 2023. KRACL: contrastive learning with graph context modeling for sparse knowledge graph completion. In *Proceedings of the ACM Web Conference (WWW 2023)*, 2548–2559.

Tang, J.; Yang, Y.; Wei, W.; Shi, L.; Su, L.; Cheng, S.; Yin, D.; and Huang, C. 2024a. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International Conference on Research and Development in Information Retrieval (SIGIR 2024)*, 491–500.

Tang, J.; Yang, Y.; Wei, W.; Shi, L.; Xia, L.; Yin, D.; and Huang, C. 2024b. Higpt: Heterogeneous graph language model. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2024)*.

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

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2018. Deep Graph Infomax. In *International Conference on Learning Representations (ICLR 2018)*.

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

Wang, K.; Shen, Z.; Huang, C.; Wu, C.-H.; Dong, Y.; and Kanakia, A. 2020. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1): 396–413.

Wang, Y.; Gou, X.; Xu, X.; Geng, Y.; Ke, X.; Wu, T.; Yu, Z.; Chen, R.; and Wu, X. 2024. Scalable Community Search over Large-scale Graphs based on Graph Transformer. In *Proceedings of the 47th International Conference on Research and Development in Information Retrieval (SIGIR 2024)*, 1680–1690.

Wei, X.; Liu, Y.; Sun, J.; Jiang, Y.; Tang, Q.; and Yuan, K. 2023. Dual subgraph-based graph neural network for friendship prediction in location-based social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD 2023)*, 17(3): 1–28.

Wu, Q.; Zhao, W.; Li, Z.; Wipf, D. P.; and Yan, J. 2022. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems (NIPS 2022)*, 35: 27387–27401.

Xia, J.; Wu, L.; Chen, J.; Hu, B.; and Li, S. Z. 2022a. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM Web Conference 2022 (WWW 2022)*, 1070–1079.

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

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations (ICLR 2018)*.

Yang, Y.; Wang, F.; Zhu, E.; Jiang, F.; and Yao, W. 2024. Social Behavior Analysis in Exclusive Enterprise Social Networks by FastHAND. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 18(6): 1–32.

You, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2023. Graph domain adaptation via theory-grounded spectral regularization. In *The eleventh international conference on learning representations (ICLR 2023)*.

Yu, J.; Yin, H.; Xia, X.; Chen, T.; Li, J.; and Huang, Z. 2023. Self-supervised learning for recommender systems: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE 2023)*.

Zeng, H.; Zhou, H.; Srivastava, A.; Kannan, R.; and Prasanna, V. 2019. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *Proceedings of International Conference on Learning Representations (ICLR 2019)*.

Zhang, Y.; Gao, S.; Pei, J.; and Huang, H. 2022. Improving social network embedding via new second-order continuous graph neural networks. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining (KDD 2022)*, 2515–2523.

Zhao, H.; Chen, A.; Sun, X.; Cheng, H.; and Li, J. 2024a. All in one and one for all: A simple yet effective method towards cross-domain graph pretraining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2024)*.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zhao, Z.; Li, Y.; Zou, Y.; Li, R.; and Zhang, R. 2024b. A Survey on Self-Supervised Pre-Training of Graph Foundation Models: A Knowledge-Based Perspective. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2024)*.

Zheng, R.; Qu, L.; Cui, B.; Shi, Y.; and Yin, H. 2023. Automl for deep recommender systems: A survey. *ACM Transactions on Information Systems (TOIS 2023)*, 41(4): 1–38.

Zhu, B.; Niu, Y.; Han, Y.; Wu, Y.; and Zhang, H. 2023. Prompt-aligned gradient for prompt tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2023)*, 15659–15669.

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 (NIPS 2021)*, 34: 1766–1779.