Multi-Domain Generalized Graph Meta Learning

Mingkai Lin, Wenzhong Li*, Ding Li, Yizhou Chen, Guohao Li, Sanglu Lu

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

Abstract

Graph meta learning aims to learn historical knowledge from training graph neural networks (GNNs) models and adapt it to down-stream learning tasks in a target graph, which has drawn increasing attention due to its ability of knowledge transfer and fast adaptation. While existing graph meta learning approaches assume the learning tasks are from the same graph domain but lack the solution for multi-domain adaptation. In this paper, we address the multi-domain generalized graph meta learning problem, which is challenging due to non-Euclidean data, inequivalent feature spaces, and heterogeneous distributions. To this end, we propose a novel solution called MD-Gram for multi-domain graph generalization. It introduces an empirical graph generalization method that uses empirical vectors to form a unified expression of non-Euclidean graph data. Then it proposes a multi-domain graphs transformation approach to transform the learning tasks from multiple source-domain graphs with inequivalent feature spaces into a common domain, where graph meta learning is conducted to learn generalized knowledge. It further adopts a domain-specific GNN enhancement method to learn a customized GNN model to achieve fast adaptation in the unseen target domain. Extensive experiments based on four real-world graph domain datasets show that the proposed method significantly outperforms the state-of-the-art in multi-domain graph meta learning tasks.

Introduction

Graph Neural Networks (GNNs) (Kipf and Welling 2017; Wu et al. 2019; Klicpera, Bojchevski, and Günnemann 2019) are powerful models for learning representation of graphs and have achieved great success in dealing with graph-related applications with data containing rich relational information. They are widely applied in social networks, knowledge graphs, recommendation systems, etc. A large number of GNN models and applications were proposed in the past years (Hamilton, Ying, and Leskovec 2017; Luo, Yan, and Ji 2021; Cheng et al. 2022).

Despite their great success, the existing GNN models are typically designed for specific tasks and they are lack of the ability of quick adaptation to new graph tasks with only a small number of labeled instances (Zhou et al. 2019; Zhang et al. 2022). In recent years, graph meta learning (Huang and Zitnik 2020; Zhang et al. 2022) has been proposed to address this issue. Graph meta learning leverages a distribution of similar GNN based tasks to accumulate transferable knowledge from prior learning experience, which can be used as a strong inductive basics for fast adaptation to down-stream tasks. Based on the learning tasks, related works on graph meta learning can be divided into three categories: node-level (Yao et al. 2020; Liu et al. 2021), edge-level (Bose et al. 2019; Huang and Zitnik 2020), and graph-level (Wang et al. 2021; Guo et al. 2021) meta learning.

The effectiveness of graph meta learning is established on a strong assumption that the learning tasks in meta-training and meta-testing phases are from the same domain and the same distribution (e.g., coming from the similar networks and sharing the same node feature space). Such an assumption is hard to hold in real-world applications, and there is an urgent need to train GNNs with transferable knowledge into unseen domains. For example, we may want to train a GNN model based on diverse open graph datasets (such as citation networks and social networks), and then applied the model for product recommendation (such as a co-purchasing network) where the information available on the target graph is limited due to privacy protection. We refer to such a problem as *multi-domain generalized graph meta learning problem*, which aims to learn prior experiences that can be generalized from multi-source domain graphs and fast adapted to the tasks on unseen target domain with only a small number of labeled samples. To the best of our knowledge, the multi-domain generalized graph meta learning problem has not yet been studied in the literature.

In computer vision (CV), domain generalization techniques have been studied to improve the generalization capability of a model in different domains, and the typical strategies include data augmentation (Honarvar Nazari and Kovashka 2020; Zhou et al. 2020b), domain-invariant representation learning (Li et al. 2018a; Zhou et al. 2020a), and exploiting learning strategies (Dou et al. 2019; Li et al. 2019). Despite the progress, domain generalization for graph meta learning encounters several challenges. First, graph meta learning deals with non-Euclidean graph data consisting of nodes and edges from heterogeneous domains, which is completely different from the regular image data and cannot be processed by the CV's domain generalization meth-

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

ods. Second, the feature space in different graph domains are inequivalent as well as with diverse graph structures and distributions, whereas all the existing domain generalization methods in CV assume homogeneous feature space. In summary, no effective solution has been found to solve the multi-domain generalized graph meta learning problem.

To address the above challenges, we propose a novel multi-domain generalized graph meta learning (MD-Gram) framework, that can learn transferable knowledge from multi-domain graphs and be adapted to unseen domain graphs. Firstly, it introduces an *empirical graph generalization* method that uses empirical vectors to represent nodes and edges of multi-domain graphs, which enables the non-Euclidean graph data to be represented with the same dimensional unified expression. Secondly, it proposes a *multi-domain graphs transformation* method that uses neural networks to transform multiple source-domain graphs into a common domain with minimal Wasserstein distances, which enables the learning tasks from inequivalent feature spaces and heterogeneous graph distributions to be trained with meta learning in the common domain to learn generalized knowledge. Thirdly, it adopts a *domain-specific GNN enhancement* approach to learn a customized GNN meta model, which achieves knowledge transfer and fast adaptation to down-stream learning tasks in the unseen target domain. The efficiency and feasibility of the proposed framework are verified by extensive experiments.

The contributions of our work are summarized as follows.

- We are the first to address the multi-domain generalized graph meta learning problem, which confronts the challenges of training meta GNN models with non-Euclidean graph data on multiple source domains with inequivalent feature spaces and heterogeneous distributions.
- We propose a novel graph meta learning framework called MD-Gram for multi-domain graph generalization. It forms a unified representation for heterogeneous graph data and transforms the multi-domain GNN learning tasks into a common domain for meta learning, which achieves generalized knowledge transfer and fast adaptation for the unseen target graph.
- We implement MD-Gram with Pytorch and test its efficiency and feasibility based on four real-world graph datasets from varying domains. Under the leave-one-domain-out setting, our method achieves faster convergence with fewer labeled samples, and its AUC metric significantly outperforms the state-of-the-art graph meta learning methods.

Related Works

Graph Neural Networks: Recently Graph Neural Networks (GNNs) have become an effective way to learn graph structure with deep neural networks. With the operation of neighboring information aggregation and message passing, they combine node features with graph structure to learn normalized node representations. GNNs gained popularity due to their superior performance in many learning tasks such as node classification to predict node properties based on other node properties in a graph (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017); graph classification to discriminate between graphs of different classes (Bai et al. 2019; Lee, Rossi, and Kong 2018), and link prediction to predict the existence of a relationship between two entities in a network (Liben-Nowell and Kleinberg 2007; Rossi et al. 2022). For its effectiveness and universality, training GNN models more rapidly and efficiently with a small set of labeled instances is becoming more and more necessary and thus meta learning on graphs has caused greater concern.

Graph Meta Learning: Meta learning (or learn to learn) is to learn from prior experiences to form inductive biases for fast adaptation to new tasks (Thrun and Pratt 2012; Schmidhuber, Zhao, and Wiering 1997) and it has been particularly effective in few-shot learning tasks (Finn, Abbeel, and Levine 2017; Lee and Choi 2018). Consequently, meta learning on graphs generally combines the advantages of GNNs and meta learning. There have been many related studies that spanned a variety of methods and applications in recent years. From the view of methods, graph meta learning often used metric based methods that learned task-specific similarity metrics between query data and support set data (Snell, Swersky, and Zemel 2017; Yao et al. 2020; Tan et al. 2022); and optimization-based methods that learned well initialized base-learner to quickly adapt to a new task with gradient computation (Zhou et al. 2019; Wang et al. 2020b). Considering the learning tasks, these works can be divided into three categories: node-level (Yao et al. 2020; Liu et al. 2021; Huang and Zitnik 2020), edge-level (Bose et al. 2019; Huang and Zitnik 2020) and graph-level (Wang et al. 2021; Guo et al. 2021) meta learning. These works focused on specific graph learning tasks and developed corresponding graph meta learning methods for quickly adapting learning knowledge. However, most of the graph meta learning works follow a strong fundamental assumption that all learning tasks are from the same distribution domain. The problem of generalizing prior experiences from multi-domain graphs in meta training and adapting them to unseen domain graphs has not yet been studied in the literature.

Domain Generalization: Domain generalization refers to the problem of training a model from a collection of different domains that can directly generalize to the target domains, which is a crucial learning task in computer vision (Li et al. 2018a,b; Muandet, Balduzzi, and Schölkopf 2013). There have been many works about it, such as utilizing data augmentation as a regularization approach to improve the generalization of DNNs (Honarvar Nazari and Kovashka 2020; Zhou et al. 2020b), domain-invariant representation learning with some statistical metrics (e.g. MMD and Wasserstein distance) (Li et al. 2018a; Zhou et al. 2020a), and exploiting learning strategies to help model generalization (Dou et al. 2019; Li et al. 2019). Domain generalization is more challenging with meta-learning setting (Tseng et al. 2020; Du et al. 2020), and existing works are limited to dealing with regular image data with equivalent feature space, which can not apply to non-Euclidean graph data. One recent work (Hassani 2022) developed a multi-view enhanced method MVG-Meta to learn graph embeddings for cross-domain graph classification, but it still cannot be generalized to multiple heterogeneous graph domains for meta training.

Different from the existing works, we are the first to address the multi-domain generalized graph meta learning problem and proposed a novel solution called MD-Gram that deals with non-Euclidean graph data and adopts multi-domain graphs transformation for graph meta learning.

Problem Formulation

Given a graph $G = (V, A, X)$, where V is the set of nodes, A is an adjacency matrix representing the edges, and X is a feature matrix regarding to nodes. A GNN model follows a neighborhood aggregation strategy, which iteratively updates the presentation of a node by aggregating representations from its neighbors in multiple layers. Specifically, operation in each layer can be formulated as:

$$Z^{h+1} = \mathcal{M}(Z^h, A; W^h), \quad Z^0 = X, \tag{1}$$

where $\mathcal{M}(\cdot)$ is the message aggregation function and W^h is the learnable parameters in the h-th layer.

GNN is commonly used in classification tasks, and the general process can be formulated as:

$$Z = \text{GNN}(A, X; \theta_g), \quad \mathcal{L}(A, X, Y; \theta_g) = \ell(f(Z), Y), \tag{2}$$

where $\ell(\cdot, \cdot)$ measures the difference between predictions and corresponding true labels Y, $f(Z)$ is the prediction function, and $\theta_g = (W^0, W^1, \cdots)$ are learnable parameters of the GNN. In the multi-domain generalized graph meta learning problem, we assume that there are a set of s source graph domains $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_s\}$, where element $\mathcal{D}_k$ represents the kth source graph domain. Similar to the definition in (Pan and Yang 2009), we denote a graph domain as $\mathcal{D}_k = \{\mathcal{X}_k, \mu_k^{node}, \mu_k^{edge}\}$ where $\mathcal{X}_k$ is the feature space, μ_k^{node} is the node probability distribution on $\mathcal{X}_k$, and μ_k^{edge} is the edge probability distribution of connecting two nodes. We denote the target domain as $\mathcal{D}_*$ with similar definition.

Taking the domain of a social network for example, each point in the space $\mathcal{X}_k$ represents a user with an embedding, where μ_k^{node} over $\mathcal{X}_k$ is the probability for the user to take part in the network, and μ_k^{edge} is the probability for two users to establish a friendship.

We consider standard domain generalization that all tasks share the same label space. Assume there are s graphs from the source domains, denoted by $G_{\mathcal{D}_k} = (V_{\mathcal{D}_k}, A_{\mathcal{D}_k}, X_{\mathcal{D}_k})$ $(1 \le k \le s)$ where $A_{\mathcal{D}_k} \in \mathbb{Z}^{|V_{\mathcal{D}_k}| \times |V_{\mathcal{D}_k}|}$ and $X_{\mathcal{D}_k} \in \mathbb{R}^{|V_{\mathcal{D}_k}| \times d_k}$. The learning task on each graph is denoted by $T_{\mathcal{D}_k} = \{Y_{\mathcal{D}_k}, f_k(\cdot)\}$, where Y_k is the label set and $f_k(\cdot)$ is a predictive function.

The multi-domain generalized graph meta learning problem aims to learn task-level knowledge from multi-domain tasks $T_{\mathcal{D}_k}$, and apply such knowledge to new tasks in an unseen domain for fast adaptation, i.e., training an optimal initialized GNN model θ^* for the target domain $\mathcal{D}_*$.

Methodology

The proposed multi-domain generalized graph meta learning (MD-Gram) framework is illustrated in Fig. 1. The meta training consists of three phases: *empirical graph generalization*, *multi-domain graphs transformation*, and *domain-specific GNN enhancement*. In the first phase, it proposes to use empirical vectors to represent the nodes and edges of multi-domain graphs to form a unified and generalized graph expression with the same dimension. In the second phase, it transforms multiple source-domain graphs into a common domain by minimizing their Wasserstein distances. The common domain bridges the inequivalent feature spaces and heterogeneous graph distributions, and it enables graph meta learning in the transformed domain to learn generalized knowledge. In the third phase, it trains a customized GNN from the generalized model by performing domain-specific GNN enhancement to achieve knowledge transfer and fast adaptation to down-stream learning tasks in the unseen target domain, which is further used for meta test. Noted that phase two and three are both trained alternatively until convergence. The detailed processes are as follows.

Empirical Graph Generalization

Recall that a graph domain is expressed by a feature space as well as a node distribution and an edge distribution on the feature space. To cope with the heterogeneity of multiple graph domains, we need to form a unified expression of the heterogeneous node/edge distributions of different graphs. Since their true distributions are unknown, we can observe the instances (i.e., nodes and edges) on a graph, and adopt the empirical vector technique (Rubner et al. 2000) to form an empirical distribution for each graph on a vector space.

Given a graph $G_{\mathcal{D}_k} = \{V_{\mathcal{D}_k}, A_{\mathcal{D}_k}, X_{\mathcal{D}_k}\}$ defined in domain $\mathcal{D}_k$, we use empirical vectors to represent its nodes and edges with the same dimension. Specifically, the empirical vectors are computed by $X_{\mathcal{D}_k}^{node} = \varphi(X_{\mathcal{D}_k})$ and $X_{\mathcal{D}_k}^{edge} = \varphi(X_{\mathcal{D}_k}, A_{\mathcal{D}_k})$ respectively, where φ means the concat operation, namely $x_v^{node} = x_v || x_v$ and $x_{(u,v)}^{edge} = x_u || x_v$. In this way, the graph $G_{\mathcal{D}_k}$ can be generalized into empirical vectors $X_{\mathcal{D}_k}^g = \{X_{\mathcal{D}_k}^{node}, X_{\mathcal{D}_k}^{edge}\} \in \mathbb{R}^{(m_k+n_k)\times 2d_k}$.

With the proposed generalization, the empirical distribution (combining both nodes and edges) on a graph from $\mathcal{D}_k$ can be derived by (Flamary et al. 2016):

$$\hat{\mu}_k^g(x) = \frac{1}{m_k + n_k} \sum\nolimits_{x_i \in X_{\mathcal{D}_k}^g} \delta(x - x_i), \tag{3}$$

where δ is the delta dirac function (Zhao 2011) and m_k, n_k are the numbers of the node and edge empirical vectors.

Multi-Domain Graphs Transformation

Since meta learning is generally conducted on tasks from the same domain and distribution, we need to transform the learning tasks from multiple graph domains into a common domain where meta learning can be conducted smoothly.

Based on the introduced empirical graph distribution, we apply a deep neural network ϕ_k on graph node features $X_{\mathcal{D}_k}$. Thereby the individual graph and its empirical distribution $\hat{\mu}_k^g(x)$ can be transformed into a common domain $\mathcal{D}_c$, whose true distribution is μ_c^g. The process can be briefly denoted as:

$$\phi_k(X_{\mathcal{D}_k}^g) = \{\phi_k(X_{\mathcal{D}_k}^{node}),\ \phi_k(X_{\mathcal{D}_k}^{edge})\} \in \mathbb{R}^{(m_k+n_k)\times 2d}, \tag{4}$$

In the common domain, the graph node features are transformed to a unified space of d dimensions and the transformed empirical graph distribution can also be derived by Eq. (3) as $\phi_k(\hat{\mu}_k^g)$.

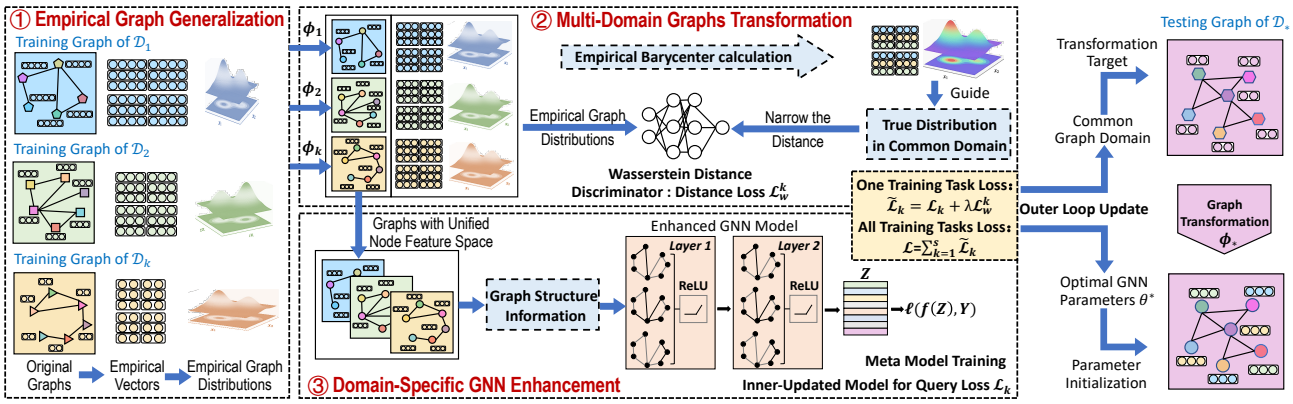


Figure 1: The framework of MD-Gram for multi-domain generalized graph meta learning.

The key question here is to find a common domain so that multiple source distributions can be transformed to it easily. Intuitively, we can merge all source distributions to form a common one that has joint distribution and aggregates knowledge from various domains. For this purpose, we use the Wasserstein barycenter (Agueh and Carlier 2011; Luise et al. 2019) to merge the discrete graph distributions.

Wasserstein distance is developed from the earth mover's distance (EMD) (Rubner et al. 2000) that measuring the distance between two discrete distributions. It can be interpreted as the minimum cost to move one pile of dirt to the other. Given two distribution measures μ_s and μ_t on the metric space $\mathbb{M}$ with finite p-th moment, their Wasserstein distance can be formulated as (Villani 2009)

$$W_p(\mu_s, \mu_t) = (\inf_{\gamma\in\Gamma(\mu_s,\mu_t)} \int_{\mathbb{M}\times\mathbb{M}} \rho(x,y)^p d\gamma(x,y))^{1/p}, \quad (5)$$

where Γ is the set of all measures on $\mathbb{M}\times\mathbb{M}$ with marginals μ_s and μ_t; $\rho(x,y)$ is a distance function for two instances x and y in the set $\mathbb{M}$; and $\gamma(x,y)$ is a randomized policy for transporting a unit quantity of material from a random location x to another location y.

Wasserstein barycenter is the center of a set of probabilistic distributions that minimizes the sum of Wasserstein distances to the elements in that set. It is an intermediate distribution not only aggregating the source probability distributions but also having the ability to preserves the modality of the different datasets, which is ideal to serve as a common domain. Assume an empirical graph distribution in the common domain is denoted by $\hat{\mu}_c^g$ with its empirical vectors $X_{\mathcal{D}_c}^g = \{X_{\mathcal{D}_c}^{node}, X_{\mathcal{D}_c}^{edge}\} \in \mathbb{R}^{(m_c+n_c)\times 2d}$, it can be derived by finding the Wasserstein barycenter of the set of empirical distributions $\{\phi_k(\hat{\mu}_k^g)\}_{k=1}^s$ from source domains:

$$\hat{\mu}_c^g = \arg\min_{\hat{\mu}_c^g} \sum_{k=1}^{s} W_p(\phi_k(\hat{\mu}_k^g), \hat{\mu}_c^g), \quad (6)$$

which can be solved by the WBT algorithm (Montesuma and Mboula 2021).

After the empirical Wasserstein barycenter $\hat{\mu}_c^g$ is obtained, it can serve as a guide for multi-domain graphs transformation. However, the above process still encounters two problems. Firstly, the true distribution μ_c^g for the common domain is still unknown, therefore Eq. (5) cannot be directly applied to transform. Secondly, the search of common domain in Eq. (6) relies on ϕ_k, a neural network to transform the empirical distribution into the common domain, which needs to be trained jointly for Eq. (6) until convergence.

To solve the first problem, we utilize a neural network based approximation to calculate the Wasserstein distance. Owing to Kantorovich-Rubinstein theorem (Rachev et al. 1990), the dual representation of the first Wasserstein distance can be written as:

$$W_1(\mu_s, \mu_t) = \sup_{\|f_w\|_L\le 1} \mathbb{E}_{x\sim\mu_s}[f_w(x)] - \mathbb{E}_{x\sim\mu_t}[f_w(x)], \quad (7)$$

where $\|f_w\|_L \le 1$ is the Lipschitz norm of f_w. The equation shows that there always exists an optimal 1-Lipschitz function f_w that separates μ_s and μ_t, whose maximum expectation is the first Wasserstein distance. This enables us to use a neural network as the Wasserstein discriminator to fit the function f_w. The discriminator takes an empirical vector as input and output a real number, which can be trained by maximizing $\mathbb{E}_{x\sim\mu_s}[f_w(x)] - \mathbb{E}_{x\sim\mu_t}[f_w(x)]$. In this way, the Wasserstein distance can be derived with a neural network parametrized by θ_w^k, which uses $X_{\mathcal{D}_c}^g$ and the following loss to optimize the distance between $\phi_k(\hat{\mu}_k^g)$ and μ_c^g:

$$\begin{aligned}&\mathcal{L}_w^k(\phi_k(\hat{\mu}_k^g), \mu_c^g; \theta_w^k, \theta_{\phi_k})\\ &= \frac{1}{n_k}\sum_{x\in\phi_k(X_{\mathcal{D}_k}^{node})} f_w^k(x) - \frac{1}{n_c}\sum_{x\in X_{\mathcal{D}_c}^{node}} f_w^k(x)\\ &+ \frac{1}{m_k}\sum_{x\in\phi_k(X_{\mathcal{D}_k}^{edge})} f_w^k(x) - \frac{1}{m_c}\sum_{x\in X_{\mathcal{D}_c}^{edge}} f_w^k(x).\end{aligned} \quad (8)$$

To train the neural network ϕ_k parameterized by θ_{ϕ_k} for domain transformation, we can first train f_w^k with only θ_w^k to maximize the supremum, and then minimize $\mathcal{L}_w^k$ with θ_{ϕ_k} by fixing the parameter θ_w^k, which corresponds to the following adversarial min-max process:

$$\min_{\theta_{\phi_k}} \max_{\theta_w^k} \mathcal{L}_w^k(\phi_k(\hat{\mu}_k^g), \mu_c^g; \theta_w^k, \theta_{\phi_k}). \quad (9)$$

The above process is optimized alternatively until the neural networks converge. In this way, multiple source graphs can be transformed to the common domain gradually.

Domain-specific GNN Enhancement

With multi-domain graph transformation, graph meta learning can be conducted in the graphs with unified node feature space to train a generalized GNN meta model. To apply

the model to the unseen target domain, we need to conduct domain-specific GNN enhancement to form a customized GNN model in adaptation to the target graph with diverse structures. There have been many works on adjusting GNNs for varying graph structures (Wen, Fang, and Liu 2021; Bose et al. 2019; Wang et al. 2020a; Perez et al. 2018) and handling distribution shift for single-source graphs (Wu et al. 2022; Yehudai et al. 2021). In this paper, we adopt the GS-Gating method in (Bose et al. 2019) for domain-specific GNN enhancement. Specifically, during message passing of GNN, it uses a GCN with summing pooling operation to compute a graph signature $\psi(G)$ for a feature-wise linear modulation, and trains a sigmoid gating term for the GNN to adaptively learn when to apply the modulation. Therefore the learning task of GNN is reformulated with an extra parameter θ_ψ:

$$Z = \text{GNN}(A, X; \theta_g, \theta_\psi), \mathcal{L}(A, X, Y; \theta_g, \theta_\psi) = \ell(f(Z), Y). \quad (10)$$

Graph Meta Learning Process

Meta Training:

We adopt the popular MAML (Finn, Abbeel, and Levine 2017) meta learning framework to train GNNs. Generally, meta training consists of two procedures: an inner loop that mimics training individual task with partial labeled data, and an outer loop that provides task-level update with task query loss. Thus we design the following training process for meta learning combined with the adversarial process of Eq. (9).

For the inner loop, with current empirical Wasserstein barycenter $\hat{\mu}_c^g$ derived from the transformed distributions $\{\phi_k(\hat{\mu}_k^g)\}_{k=1}^s$, we get the Wasserstein distance between each $\phi_k(\hat{\mu}_k^g)$ and μ_c^g by maximizing a discriminator f_w^k with learning rate α_1 for r updates (i.e., the inner max in Eq. (9)):

$$\theta_w^k = \theta_w^k + \alpha_1 \nabla_{\theta_w^k} \mathcal{L}_w^k(\phi_k(\hat{\mu}_k^g), \mu_c^g; \theta_w^k, \theta_{\phi_k}). \quad (11)$$

Then, we mimic training on each graph task with t loops:

$$\theta_g^{(l)} = \theta_g^{(l-1)} - \alpha_2 \nabla_{\theta_g} \mathcal{L}_k(A_{\mathcal{D}_k}, \phi_k(X_{\mathcal{D}_k}), Y_{\mathcal{D}_k}^{spt}; \theta_g^{(l-1)}, \theta_\psi, \theta_{\phi_k}), \quad (12)$$

where α_2 is the inner learning rate and $\mathcal{L}_k$ is the prediction loss in the support set. Note that above training is focused on the GNN model itself, where θ_g is the updated parameter.

During the outer loop, we hope to train the GNN meta model and narrow down the distance between $\phi_k(\hat{\mu}_k^g)$ and μ_c^g jointly. Therefore, we fix θ_w^k and update θ_{ϕ_k} only in order to minimize the distance loss $\mathcal{L}_w^k(\phi_k(\hat{\mu}_k^g), \mu_c^g; \theta_w^k, \theta_{\phi_k})$ for the outer min in Eq. (9). And meanwhile, we apply the inner-updated meta model on a query set to minimize the prediction loss $\mathcal{L}_k$ through the second-order gradient, which serves as task-level update signals. Thus we can utilize the following weighted loss function for each task:

$$\begin{aligned} \tilde{\mathcal{L}}_k = & \mathcal{L}_k(A_{\mathcal{D}_k}, \phi_k(X_{\mathcal{D}_k}), Y_{\mathcal{D}_k}^{qry}; \theta_g^{(t)}, \theta_\psi, \theta_{\phi_k}) \\ & + \lambda \mathcal{L}_w^k(\phi_k(\hat{\mu}_k^g), \mu_c^g; \theta_w^k, \theta_{\phi_k}), \end{aligned} \quad (13)$$

where λ is a hyperparameter to tune the influence of the task loss and the distance loss. Thereby the total task loss below can be computed to update the parameters $\{\theta_{\phi_k}\}_{k=1}^s, \theta_g, \theta_\psi$:

$$\mathcal{L} = \sum\nolimits_{k=1}^{s} \tilde{\mathcal{L}}_k. \quad (14)$$

Algorithm 1: Meta Training for MD-Gram

Input: Multi-domain graphs $\{G_{\mathcal{D}_k} = (V_{\mathcal{D}_k}, A_{\mathcal{D}_k}, X_{\mathcal{D}_k})\}_{k=1}^s$; Unified node feature dimensions d; Iteration numbers t, r; Learning rates $\alpha_1, \alpha_2, \alpha_3$; The numbers of node and edge vectors for barycenter n_c, m_c; Loss hyperparameter λ
Output: Global parameters for GNN θ_g, θ_ψ; The empirical vectors $X_{\mathcal{D}_c}^g$ with $\hat{\mu}_c^g$ for the common domain

```
Initialize parameters {θ_{φ_k}}_{k=1}^s, θ_g, θ_φ, total loss L = 0
while not converged do
    Calculate {φ_k(μ̂)}_{k=1}^s via parameters {θ_{φ_k}}_{k=1}^s
    Calculate the empirical barycenter: μ̂_c^g with its X_{D_c}^g
    for G_{D_k} in {G_{D_k} = (V_{D_k}, A_{D_k}, X_{D_k})}_{k=1}^s do
        Sample support set Y_{D_k}^{spt} and query set Y_{D_k}^{qry}
        Initialize θ_w^k
        for l = 1 ··· r do
            Calculate L_w^k and update θ_w^k by Eq. (11) with α_1
        end for
        Initialize θ_g^{(0)} = θ_g
        for l = 1 ··· t do
            Calculate L_k and update θ_g^{(l)} by Eq. (12) with α_2
        end for
        Calculate task loss L̃_k by Eq. (13) regarding λ
        L = L + L̃_k
    end for
    Backpropagate L with α_3 to update {θ_{φ_k}}_{k=1}^s, θ_g, θ_ψ
end while
return θ_g, θ_ψ; μ̂_c^g with its X_{D_c}^g
```

With the meta training, we gradually update the barycenter to search for the common domain and train an effective enhanced GNN meta model in the common domain. The overall process of meta training is shown in Alg. 1.

Meta Testing:

During meta testing, by adopting the adversarial process of Eq. (9), we can use empirical vectors to transform the target graph $G_{\mathcal{D}_*}$ from the unseen domain to the common domain as preprocess. Then the well-trained meta model is applied for GNN initialization. Note that the GNN should be finetuned jointly with the transformation parameters.

Experiments

Experimental Settings

Dataset. The experiments are based on four real-world networks from different graph domains: (1) **Product [P]** (Hu et al. 2020): The Ogbn-products from Open Graph Benchmark, which is an Amazon product co-purchasing network

Dataset	Type	#Graphs	Avg.$\lvert N\rvert$	Avg. $\lvert E\rvert$	Feat.
Product	Co-purchasing	5	2000	16,250	100
Yelp	Social	5	2000	38,306	300
Reddit	Post-to-post	5	2000	21,172	602
Academic	Citation	5	2000	6,788	128

Table 1: Datasets statistics.

Methods	10% known edges				20% known edges				30% known edges			
	ARY-P	ARP-Y	APY-R	PRY-A	ARY-P	ARP-Y	APY-R	PRY-A	ARY-P	ARP-Y	APY-R	PRY-A
VGAE	66.95	55.48	51.72	72.31	72.47	71.17	60.27	76.00	71.83	66.96	59.03	74.40
VGAE-F	65.66	67.38	59.41	65.93	71.69	78.97	64.80	71.66	67.67	76.56	65.49	67.94
MAML	70.62	69.87	62.94	74.73	74.93	81.86	69.87	77.28	72.43	79.04	69.78	76.06
Meta-Graph	63.26	75.86	66.09	63.71	64.28	79.92	69.52	64.17	65.27	80.89	72.03	62.52
MVG-Meta	61.68	79.28	55.43	57.52	68.28	82.66	60.61	54.77	73.01	84.03	63.46	56.88
G-META	55.87	71.98	59.29	50.78	57.26	75.21	60.29	49.18	57.39	77.57	61.42	49.49
MI-GNN	60.16	53.92	55.13	65.12	61.99	60.07	59.58	66.10	64.88	61.92	60.97	66.78
MD-Gram	**76.79**	**85.30**	**73.00**	**80.50**	**79.76**	**87.17**	**76.33**	**81.55**	**83.86**	**88.33**	**78.67**	**82.48**

Table 2: The AUC (%) results for 10-epoch updates for different methods with different fractions of known edges.

with product descriptions as features. (2) **Yelp [Y]** (Zeng et al. 2019): A social network formed by users and their friendship of the Yelp website that uses customer reviewers as node features. (3) **Reddit [R]** (Hamilton, Ying, and Leskovec 2017): A graph dataset from the Reddit posts with 50 large communities to build a post-to-post graph. (4) **Academic [A]** (Hu et al. 2020): An academic citation network named ogbn-papers100M from Open Graph Benchmark.

Following the method of (Yao et al. 2020; Wen, Fang, and Liu 2021), we extract 5 graphs from each domain for experiments. The detailed information are summarized in Table 1.

Default Parameters and Baseline Algorithms. For the training setting, we follow the leave-one-domain-out protocol. Specifically, we choose one domain as the test domain and use the remaining domains as the source domains; and the model showing the best performance on all source domains are chosen as the final model. In this way, we have four different datasets, each has 15 multi-domain graphs for meta training and the rest 5 domain graphs for meta testing, e.g., PYR-A denotes PYR for training and A for testing. We consider the few-shot setting for a link prediction task that at most 30% edges is known beforehand, fixed 10% for validation and predict the rest edges following the setting of (Bose et al. 2019; Huang and Zitnik 2020). The VGAE (Kipf and Welling 2016) is used as our basic model consisting of a two-layer GCN as encoder and inner product operation as decoder. The unified node feature dimension is $d = 256$; learning rates are $\alpha_1 = 0.001, \alpha_2 = \alpha_3 = 0.005$; iteration numbers are $r = 20, l = 10$; hyperparameter for weighted loss is $\lambda = 1$. The numbers of node/edge empirical vectors for graphs and barycenter are fixed as 2000/4000 respectively because there is no constraint on them and we only use partial instances to represent the distributions.

Seven baselines including state-of-the-art methods on graph meta leaning for link prediction task are used for performance evaluation. (1) **VGAE** (Kipf and Welling 2016): it is the original VGAE model trained individually on each test graph. (2) **VGAE-F** (Kipf and Welling 2016): it pretrains a VGAE on the training graphs and finetunes it on the test graphs. (3) **MAML** (Finn, Abbeel, and Levine 2017): it directly applied the MAML framework for multi-domain graph meta learning. (4) **Meta-Graph** (Bose et al. 2019): it injects graph signature in VGAE to do few-shot multi-graph link prediction. (5) **G-META** (Huang and Zitnik 2020): it leverages subgraph to learn node embedding and further combines ProtoNet (Snell, Swersky, and Zemel 2017) and MAML for model optimization. (6) **MI-GNN** (Wen, Fang, and Liu 2021): it employs a dual adaptation mechanism to train a graph model with both graph-level and task-level adaptation. (7) **MVG-Meta** (Hassani 2022): it develops a multi-view enhanced GIN encoder to learn embedding for cross-domain datasets. Since the baselines cannot handle graphs with different feature dimensions, we use the Autocoder (Hinton and Salakhutdinov 2006) to adjust them to fixed dimensions (i.e., 256) with minimum information loss.

We report the model accuracy as the AUC score. For each setting, we conduct training for 5 times and take the mean results. The experiments are implemented with Pytorch in Python 3.6.8 and conducted on a PC with Intel Xeon E5-2620 v2 2.10GHz CPU, GeForce RTX 2070 8G GPU and 64GB memory, running the 64-bit CentOS Linux 7.2.

Numerical Results

Ability of Fast Adaptation. We first compare the fast adaptation ability of different algorithms. The adaptation ability indicates that the model can be adapted quickly with the pre-trained parameters, and it corresponds to that the initialized model achieves good performance with fewer training epochs. Therefore we compare the performance in the 10-th epoch to show the adaptation ability. We display the AUC results in Table 2 for 10-epoch gradient updates under 10%, 20% and 30% fractions of known edges as training set. Compared with all baselines, MD-Gram shows the best performance with significant AUC improvements. Compared to the second best method, MD-Gram has a performance improvement of about 6%-10% in ARY-P, 4%-6% in ARP-Y, 6%-10% in APY-R, and 4%-8% in PRY-A. This shows the benefits of learning knowledge from multiple source domains, and transferring such prior knowledge to the target domain really helps to achieve fast adaptation with small samples and fewer training epochs.

Convergence Analysis. We then analyze the convergence for different methods under continuous updating epochs in the setting of 20% known edges, and the results are shown in Fig. 2. From the figure, it can be seen that at the beginning of gradient descent, nearly all models developed from MD-Gram have the best AUC results, which means that the meta-learned GNN can be applied to the target graph to achieve high accuracy without retraining. With increasing finetuning epochs, the AUC metrics of MD-Gram rise more quickly than that of the baselines and steadily reach convergence. Some baselines like VGAE-F, MAML and Meta-Graph fail to offer a good initialized model so their AUCs are at a low level at the beginning, which implies that they

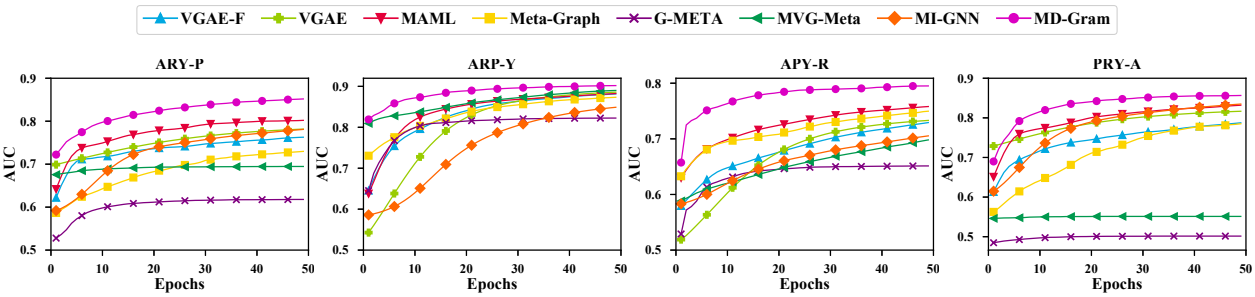


Figure 2: The AUC results of continuous epochs for different methods with 20% known edges.

20% known edges	ARY-P	ARP-Y	APY-R	PRY-A
VGAE	85.23	90.26	80.28	89.04
VGAE-F	85.51	90.39	81.26	84.17
MAML	86.59	90.42	81.94	88.58
Meta-Graph	86.40	90.52	81.50	87.48
MVG-Meta	69.47	89.29	71.61	55.12
G-META	62.23	82.60	65.18	50.15
MI-GNN	88.73	90.12	80.41	**90.80**
MD-Gram	**89.14**	**90.71**	**82.24**	88.19

Table 3: Final AUC (%) of different methods (20% edges).

20% known edges	ARY-P	ARP-Y	APY-R	PRY-A
W/O Graph Gene.	64.57	79.70	68.96	65.09
W/O Domain Trans.	70.59	85.60	72.82	78.70
W/O GNN Enhan.	76.98	**88.79**	74.12	76.09
W/O Edge Vectors	76.41	87.06	74.47	75.76
W/O Preprocess	77.78	87.04	74.72	79.19
MD-Gram	**79.76**	87.17	**76.33**	**81.55**

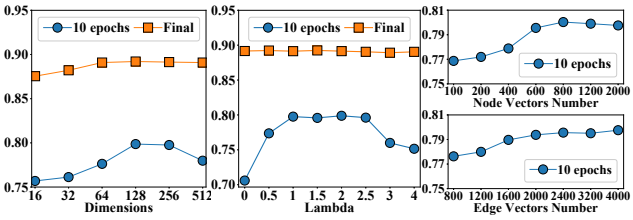
Table 4: The ablation results of 10-epoch AUC (%).

can not exploit the knowledge from different domains effectively. Some baselines like G-META converge to a low accuracy level, probably stuck into local optimum.

Final Performance. We further compare the final results in Table 3 when all models are trained to converge. Due to page limit, we only show the results with 20% edges known, and the full results can be found in the Appendix. It is shown that different methods have significant difference results in their final accuracy, even they are trained with sufficient data samples and epochs in the target graph. The proposed MD-Gram achieves the best final performance except for the PRY-A dataset setting. It implies that multi-domain generalized graph meta learning can not only fast adapt but also improve the final performance of GNNs in the target domain.

Ablation Study. Here we investigate the effective of each module. We evaluate the performance of MD-Gram by removing the following component individually: (1) *graph generalization*; (2) *domain transformation*; (3) *GNN enhancement*. (4) *edge empirical vectors*, and (5) *preprocess in meta testing*. We display the 10-epoch results with the setting of 20% known edges in Table 4. It shows that without these components the performances of MD-Gram are mostly weakened to some extent except "W/O GNN enhancement" in ARP-Y. It implies that each of the components is necessary to jointly help MD-Gram improve its performances.

Hyperparameter Analysis. We then analyze the influences of four hyperparameters in ARY-P: the unified feature dimensions d, the weight of loss λ, the number of empirical vectors m, n, and the results are shown in Fig. 3. Fig. 3(a) shows that when d increases from 16 to 128, the performance improves due to its representation ability enhanced, and its starts to declines when d is more than 256, probably due to more parameters making the model hard to train. Fig. 3(b) shows the curves of varying λ within [0,4]. It seems that the final performance is insensitive to λ, but the curve for 10-epoch update performs the best for λ within [1, 2.5], which provides the best trade-off between domain generalization and local task customization. Fig. 3(c) shows the influence of the number of empirical vectors used for training. Basically, the increasing number of empirical vectors can improve model accuracy, but is also increases the training cost. Therefore a sufficient number of observed instances from the graph is good enough to represent empirical distributions.



(a) Parameter d (b) Parameter λ (c) Parameter m, n

Figure 3: The curves for parameter analysis in ARY-P.

Conclusion

In this paper, we proposed a novel graph meta learning method called MD-Gram for training GNNs from multiple source domains. It formed a unified representation for heterogeneous graph data and transformed the multi-domain GNN learning tasks into a common domain to conduct meta learning to learn generalized knowledge. It further trained a customized domain-specific meta GNN model with a few labeled samples, which achieved generalized knowledge transfer and fast adaptation for down-stream graph learning tasks in the unseen target graph. Extensive experiments verified the efficiency and feasibility of the proposed method.

Acknowledgments

This work was partially supported by the Natural Science Foundation of Jiangsu Province (Project "Research on Frontier Basic Theory and Method of Security Defense for Power Systems with High-dimensional Uncertain Factors", Grant No. BK20222003), the National Natural Science Foundation of China (Grant Nos. 61972196, 61832008, 61832005), the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Sino-German Institutes of Social Computing.

References

Agueh, M.; and Carlier, G. 2011. Barycenters in the Wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2): 904–924.

Bai, Y.; Ding, H.; Qiao, Y.; Marinovic, A.; Gu, K.; Chen, T.; Sun, Y.; and Wang, W. 2019. Unsupervised inductive graph-level representation learning via graph-graph proximity. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2019)*, 1988–1994.

Bose, A. J.; Jain, A.; Molino, P.; and Hamilton, W. L. 2019. Meta-graph: Few shot link prediction via meta learning. *arXiv preprint arXiv:1912.09867*.

Cheng, D.; Yang, F.; Xiang, S.; and Liu, J. 2022. Financial time series forecasting with multi-modality graph neural network. *Pattern Recognition*, 121: 108218.

Dou, Q.; Coelho de Castro, D.; Kamnitsas, K.; and Glocker, B. 2019. Domain generalization via model-agnostic learning of semantic features. *Advances in Neural Information Processing Systems (NIPS 2019)*, 32.

Du, Y.; Zhen, X.; Shao, L.; and Snoek, C. G. 2020. Metanorm: Learning to normalize few-shot batches across domains. In *Proceedings of International Conference on Learning Representations (ICLR 2020)*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of International conference on machine learning (ICML 2017)*, 1126–1135. PMLR.

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

Guo, Z.; Zhang, C.; Yu, W.; Herr, J.; Wiest, O.; Jiang, M.; and Chawla, N. V. 2021. Few-shot graph learning for molecular property prediction. In *Proceedings of the Web Conference (WWW 2021)*, 2559–2567.

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

Hassani, K. 2022. Cross-domain few-shot graph classification. In *Proceedings of the AAAI conference on artificial intelligence (AAAI 2022)*.

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

Honarvar Nazari, N.; and Kovashka, A. 2020. Domain generalization using shape representation. In *Proceedings of European Conference on Computer Vision (ECCV 2020)*, 666–670. Springer.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems (NIPS 2020)*, 33: 22118–22133.

Huang, K.; and Zitnik, M. 2020. Graph meta learning via local subgraphs. *Advances in Neural Information Processing Systems (NIPS 2020)*, 33: 5862–5874.

Kipf, T. N.; and Welling, M. 2016. Variational graph auto-encoders. In *Bayesian Deep Learning Workshop (NIPS 2016)*.

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

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then propagate: Graph neural networks meet personalized pagerank. In *Proceedings of International Conference on Learning Representations (ICLR 2019)*.

Lee, J. B.; Rossi, R.; and Kong, X. 2018. Graph classification using structural attention. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2018)*, 1666–1674.

Lee, Y.; and Choi, S. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *Proceedings of International Conference on Machine Learning (ICML 2018)*, 2927–2936. PMLR.

Li, H.; Pan, S. J.; Wang, S.; and Kot, A. C. 2018a. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR 2018)*, 5400–5409.

Li, Y.; Tian, X.; Gong, M.; Liu, Y.; Liu, T.; Zhang, K.; and Tao, D. 2018b. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV 2018)*, 624–639.

Li, Y.; Yang, Y.; Zhou, W.; and Hospedales, T. 2019. Feature-critic networks for heterogeneous domain generalization. In *Proceedings of International Conference on Machine Learning (ICML 2019)*, 3915–3924. PMLR.

Liben-Nowell, D.; and Kleinberg, J. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7): 1019–1031.

Liu, Z.; Fang, Y.; Liu, C.; and Hoi, S. C. 2021. Relative and absolute location embedding for few-shot node classification on graph. In *Proceedings of the AAAI conference on artificial intelligence (AAAI 2021)*, volume 35, 4267–4275.

Luise, G.; Salzo, S.; Pontil, M.; and Ciliberto, C. 2019. Sinkhorn barycenters with free support via frank-wolfe algorithm. *Advances in neural information processing systems (NIPS 2019)*, 32.

Luo, Y.; Yan, K.; and Ji, S. 2021. Graphdf: A discrete flow model for molecular graph generation. In *Proceedings of International Conference on Machine Learning (ICML 2021)*, 7192–7203. PMLR.

Montesuma, E. F.; and Mboula, F. M. N. 2021. Wasserstein barycenter for multi-source domain adaptation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR 2021)*, 16785–16793.

Muandet, K.; Balduzzi, D.; and Schölkopf, B. 2013. Domain generalization via invariant feature representation. In *Proceedings of the International Conference on Machine Learning (ICML 2013)*, 10–18. PMLR.

Pan, S. J.; and Yang, Q. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10): 1345–1359.

Perez, E.; Strub, F.; De Vries, H.; Dumoulin, V.; and Courville, A. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2018)*, volume 32.

Rachev, S. T.; et al. 1990. *Duality theorems for Kantorovich-Rubinstein and Wasserstein functionals*. Instytut Matematyczny Polskiej Akademi Nauk (Warszawa).

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.

Rubner, Y.; Tomasi, C.; Guibas, C., Leonidas J; and Guibas, L. J. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2): 99–121.

Schmidhuber, J.; Zhao, J.; and Wiering, M. 1997. Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement. *Machine Learning*, 28(1): 105–130.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems (NIPS 2017)*, 30.

Tan, Z.; Ding, K.; Guo, R.; and Liu, H. 2022. Graph few-shot class-incremental learning. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM 2022)*, 987–996.

Thrun, S.; and Pratt, L. 2012. *Learning to learn*. Springer Science & Business Media.

Tseng, H.-Y.; Lee, H.-Y.; Huang, J.-B.; and Yang, M.-H. 2020. Cross-domain few-shot classification via learned feature-wise transformation. In *Proceedings of International Conference on Learning Representations (ICLR 2020)*.

Villani, C. 2009. *Optimal transport: old and new*, volume 338. Springer.

Wang, Y.; Abuduweili, A.; Yao, Q.; and Dou, D. 2021. Property-aware relation networks for few-shot molecular property prediction. *Advances in Neural Information Processing Systems (NIPS 2021)*, 34: 17441–17454.

Wang, Y.; Ma, Y.; Aggarwal, C.; and Tang, J. 2020a. Non-IID Graph Neural Networks. *arXiv preprint arXiv:2005.12386*.

Wang, Y.; Yao, Q.; Kwok, J. T.; and Ni, L. M. 2020b. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3): 1–34.

Wen, Z.; Fang, Y.; and Liu, Z. 2021. Meta-inductive node classification across graphs. In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR 2021)*, 1219–1228.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying graph convolutional networks. In *Proceedings of International conference on machine learning (ICML 2019)*, 6861–6871. PMLR.

Wu, Q.; Zhang, H.; Yan, J.; and Wipf, D. 2022. Handling Distribution Shifts on Graphs: An Invariance Perspective. In *Proceedings of International Conference on Learning Representations (ICLR 2022)*.

Yao, H.; Zhang, C.; Wei, Y.; Jiang, M.; Wang, S.; Huang, J.; Chawla, N.; and Li, Z. 2020. Graph few-shot learning via knowledge transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2020)*, volume 34, 6656–6663.

Yehudai, G.; Fetaya, E.; Meirom, E.; Chechik, G.; and Maron, H. 2021. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning (ICML 2021)*, 11975–11986. PMLR.

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, C.; Ding, K.; Li, J.; Zhang, X.; Ye, Y.; Chawla, N. V.; and Liu, H. 2022. Few-Shot Learning on Graphs: A Survey. *arXiv preprint arXiv:2203.09308*.

Zhao, J.-C. 2011. *Methods for phase diagram determination*. Elsevier.

Zhou, F.; Cao, C.; Zhang, K.; Trajcevski, G.; Zhong, T.; and Geng, J. 2019. Meta-gnn: On few-shot node classification in graph meta-learning. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM 2019)*, 2357–2360.

Zhou, F.; Jiang, Z.; Shui, C.; Wang, B.; and Chaib-draa, B. 2020a. Domain generalization with optimal transport and metric learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR 2020)*.

Zhou, K.; Yang, Y.; Hospedales, T.; and Xiang, T. 2020b. Learning to generate novel domains for domain generalization. In *Proceedings of European Conference on Computer Vision (ECCV 2020)*, 561–578. Springer.