MentorGNN: Deriving Curriculum for Pre-Training GNNs

Dawei Zhou*
Virginia Tech
Blacksburg, Virginia, USA
zhoud@vt.edu

Lecheng Zheng*
University of Illinois
Urbana-Champaign
Urbana, Illinois, USA
lecheng4@illinois.edu

Dongqi Fu
University of Illinois
Urbana-Champaign
Urbana, Illinois, USA
dongqif2@illinois.edu

Jiawei Han
University of Illinois
Urbana-Champaign
Urbana, Illinois, USA
hanj@illinois.edu

Jingrui He
University of Illinois
Urbana-Champaign
Urbana, Illinois, USA
jingrui@illinois.edu

ABSTRACT

Graph pre-training strategies have been attracting a surge of attention in the graph mining community, due to their flexibility in parameterizing graph neural networks (GNNs) without any label information. The key idea lies in encoding valuable information into the backbone GNNs, by predicting the masked graph signals extracted from the input graphs. In order to balance the importance of diverse graph signals (e.g., nodes, edges, subgraphs), the existing approaches are mostly hand-engineered by introducing hyperparameters to re-weight the importance of graph signals. However, human interventions with sub-optimal hyperparameters often inject additional bias and deteriorate the generalization performance in the downstream applications. This paper addresses these limitations from a new perspective, i.e., *deriving curriculum for pre-training GNNs*. We propose an end-to-end model named MentorGNN that aims to supervise the pre-training process of GNNs across graphs with diverse structures and disparate feature spaces. To comprehend heterogeneous graph signals at different granularities, we propose a curriculum learning paradigm that automatically re-weighs graph signals in order to ensure a good generalization in the target domain. Moreover, we shed new light on the problem of domain adaption on relational data (i.e., graphs) by deriving a natural and interpretable upper bound on the generalization error of the pre-trained GNNs. Extensive experiments on a wealth of real graphs validate and verify the performance of MentorGNN.

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**; **Neural networks**.

*Both authors contributed equally to this work.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557393

KEYWORDS

Domain Adaptation, Pre-Training Strategies, GNNs

ACM Reference Format:
Dawei Zhou, Lecheng Zheng, Dongqi Fu, Jiawei Han, and Jingrui He. 2022. MentorGNN: Deriving Curriculum for Pre-Training GNNs. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3511808.3557393

1 INTRODUCTION

In the era of big data, graph presents a fundamental data structure for modeling relational data of various domains, ranging from physics [49] to chemistry [6, 13], from neuroscience [4] to social science [61, 65]. Graph neural networks (GNNs) [16, 33, 55, 61] provide a powerful tool to distill knowledge and learn expressive representations from graph structured data. Despite the successes of GNNs developments, many GNNs are trained in a supervised manner that requires abundant human-annotated data. Nevertheless, in many high-impact domains (e.g., brain networks constructed by fMRI) [3, 13, 15], collecting high-quality labels is quite resource-intensive and time-consuming. To fill this gap, the recent advances [21, 38, 46, 60] have focused on pre-training GNNs by directly learning from proxy graph signals (shown in Figure 1) extracted from the input graphs. The key assumption is that the extracted graph signals are informative and task-invariant, and as such, the learned graph representation can be generalized well to tasks that have not been observed before.

Nevertheless, the current GNN pre-training strategies are still at the early stage and suffer from multiple limitations. Most prominently, the performance of the existing pre-training strategies largely relies on the quality of graph signals. It has been observed that noisy and irrelevant graph signals often lead to negative transfer [48] and marginal improvement in many application scenarios [20]. For example, one may want to predict the chemical properties [6, 13] of a family of novel molecules (e.g., the emerging COVID-19 variants). However, the available data (e.g., the known coronavirus) for pre-training are homologous but with diverse structures and disparate feature spaces. In this case, how can we control the risk of negative transfer and guarantee the generalization performance in the downstream tasks (e.g., characterizing and predicting a new COVID-19 variant)? Even worse, the risk of negative transfer can

be exacerbated when encountering heterogeneous graph signals, i.e., the graph signals can be categorized into distinct types (e.g., class-memberships, attributes, centrality scores, and temporal dependencies) at different granularities (e.g., nodes, edges, and subgraphs) [10, 12, 40, 71, 72]. To accommodate the heterogeneity of graph signals, the current pre-training strategies are often hand-engineered by using some hyperparameters [21, 22]. Consequently, the inductive bias from humans might be injected into the pre-trained models thus deteriorating the generalization performance.

We identify the following two challenges for alleviating the negative transfer phenomenon of pre-training GNNs. First (*C1. Cross-Graph Heterogeneity*), how can we distill the informative knowledge from source graphs and translate it effectively to the target graphs for solving novel tasks? Second (*C2. Graph-Signal Heterogeneity*), how can we combine and tailor complex graph signals to further harmonize their unique contributions and maximize the overall generalization performance? To fill the gap, it is crucial to obtain a versatile GNN pre-training model that can enable knowledge transfer from the source domain to the target domain and carefully exploit the relevant graph signals for downstream tasks.

In this paper, we propose an end-to-end framework, namely MentorGNN, to seamlessly embrace both aforementioned objectives for pre-training GNNs. In particular, to address C1, we propose a multi-scale encoder-decoder architecture that automatically summarizes graph contextual information at different granularities and learns a mapping function across different graphs. Moreover, to address C2, we develop a curriculum learning paradigm [24], in which a teacher model (i.e., graph signal re-weighting scheme) gradually generates a domain-adaptive curriculum to guide the pre-training process of the student model (i.e., GNNs) and enhance the generalization performance in the tasks of interest. In general, our contributions are summarized as follows.

- **Problem.** We formalize the *domain-adaptive graph pre-training* problem and identify multiple unique challenges inspired by the real applications.
- **Algorithm.** We propose a novel method named MentorGNN, which 1) automatically learns a knowledge transfer function and 2) generates a domain-adaptive curriculum for pre-training GNNs across graphs. In addition, we derive a natural and interpretable generalization bound for domain-adaptive graph pre-training.
- **Evaluation.** We systematically evaluate the performance of MentorGNN under two settings: 1) single-source graph transfer and 2) multi-source graph transfer. Extensive results prove the superior performance of MentorGNN under both settings.

The rest of our paper is organized as follows. In Section 2, we introduce the preliminary and problem definition, followed by the details of our proposed framework MentorGNN in Section 3. Experimental results are reported in Section 4. We review the related literature in Section 5. Finally, we conclude this paper in Section 6.

2 PRELIMINARY

Here we introduce the notations (shown in Table 1) and the background of our problem setting. In this paper, we use regular letters to denote scalars (e.g., α), boldface lowercase letters to denote vectors (e.g., $\mathbf{v}$), and boldface uppercase letters to denote matrices (e.g., $\mathbf{A}$). We denote the source graph and the target graph using

Table 1: Symbols and notation.

Symbol	Description
$\mathcal{G}_s, \mathcal{G}_t$	input source and target graphs.
$\mathcal{V}_s, \mathcal{V}_t$	the set of nodes in $\mathcal{G}_s$ and $\mathcal{G}_t$.
$\mathcal{E}_s, \mathcal{E}_t$	the set of edges in $\mathcal{G}_s$ and $\mathcal{G}_t$.
$\mathbf{B}_s, \mathbf{B}_t$	the node attribute matrices in $\mathcal{G}_s$ and $\mathcal{G}_t$.
$\mathbf{X}_s, \mathbf{X}_t$	the hidden representation of $\mathcal{G}_s, \mathcal{G}_t$.
$\mathcal{L}_s, \mathcal{L}_t$	the set of annotated data in $\mathcal{G}_s$ and $\mathcal{G}_t$.
n_s, n_t	# of nodes in $\mathcal{G}_s$ and $\mathcal{G}_t$.
m_s, m_t	# of edges in $\mathcal{G}_s$ and $\mathcal{G}_t$.
$\mathcal{G}_s^{(l)}, \mathcal{G}_t^{(l)}$	the l-th layer coarse graphs of $\mathcal{G}_s, \mathcal{G}_t$.
$\mathbf{X}_s^{(l)}, \mathbf{X}_t^{(l)}$	the hidden representation of $\mathcal{G}_s^{(l)}, \mathcal{G}_t^{(l)}$.
$\mathbf{A}_s^{(l)}, \mathbf{A}_t^{(l)}$	the adjacency matrices of $\mathcal{G}_s^{(l)}, \mathcal{G}_t^{(l)}$.
$\mathbf{P}_s^{(l)}, \mathbf{P}_t^{(l)}$	the perturbation matrices of $\mathcal{G}_s^{(l)}, \mathcal{G}_t^{(l)}$.
$n_s^{(l)}, n_t^{(l)}$	# of supernodes in $\mathcal{G}_s^{(l)}, \mathcal{G}_t^{(l)}$.
L	# of layers.
K	# of graph signal types.
$\odot$	Hadamard product.
$\lvert\cdot\rvert_p$	L_2-norm of a vector.

$\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s, \mathbf{B}_s)$ and $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{B}_t)$, where $\mathcal{V}_s$ ($\mathcal{V}_t$) is the set of nodes, $\mathcal{E}_s$ ($\mathcal{E}_t$) is the set of edges, and $\mathbf{B}_s$ ($\mathbf{B}_t$) is the node attributes in $\mathcal{G}_s$ ($\mathcal{G}_t$). Next, we briefly review the graph pre-training strategies and a theoretical model for domain adaptation.

Pre-training strategies for GNNs. Previous studies [17, 21, 22, 32, 46] have been proposed to use easily-accessible graph signals to capture domain-specific knowledge and pre-train GNNs. These attempts are mostly designed to parameterize and optimize the GNNs by predicting the masked graph signals (e.g., node attributes [17, 32], edges [51], subgraphs [21], and network proximity [22]) from the visible ones in the given graph $\mathcal{G}$. In Figure 1, we instantiate the masked graph signals at the level of nodes, edges, and subgraphs, respectively. To predict a given proxy graph signal $\mathbf{s} \in \mathcal{G}$, we use $f : \mathbf{s} \rightarrow [0, 1]$ to denote the true labeling function and $h : \mathbf{s} \rightarrow [0, 1]$ to denote the learned hypothesis by GNNs. The risk of the hypothesis $h(\cdot)$ can be computed as $\epsilon(h, f) := \mathbb{E}_{\mathbf{s}\in\mathcal{G}}[|h(\mathbf{s}) - f(\mathbf{s})|]$. In general, the overall learning objective of pre-training models for GNNs can be formulated as

$$\arg\max_{\boldsymbol{\theta}} \log h(\mathcal{G}|\hat{\mathcal{G}}, \boldsymbol{\theta}) = \sum_{\mathbf{s}\in\mathcal{G}} \beta_{\mathbf{s}} \log h(\mathbf{s}|\hat{\mathcal{G}}, \boldsymbol{\theta}) \tag{1}$$

where $\hat{\mathcal{G}}$ is the corrupted graph with some masked graph signals $\mathbf{s}$, $\beta_{\mathbf{s}}$ is a hyperparameter that balances the weight of the graph signal $\mathbf{s}$, and $\boldsymbol{\theta}$ is the hidden parameters of the hypothesis $h(\cdot)$. By maximizing Eq. 1, we can encode the contextual information of the selected proxy graph signals into pre-trained GNNs $h(\cdot)$. With that, the end-users can fine-tune the pre-trained GNNs and make predictions in the downstream tasks.

Domain adaptation. Following the conventional notations in domain adaptation, we let $f_s(\cdot)$ and $f_t(\cdot)$ be the true labeling functions in the source and the target domains. Given $f_s(\cdot)$ and $f_t(\cdot)$, $\epsilon_s(h) = \epsilon_s(h, f_s)$ and $\epsilon_t(h) = \epsilon_t(h, f_t)$ denote the corresponding risks with respect to hypothesis $h(\cdot)$. With this, Ben-David et al. proved a domain-adaptive generalization bound in terms of domain discrepancy and empirical risks [1]. To approximate the empirical

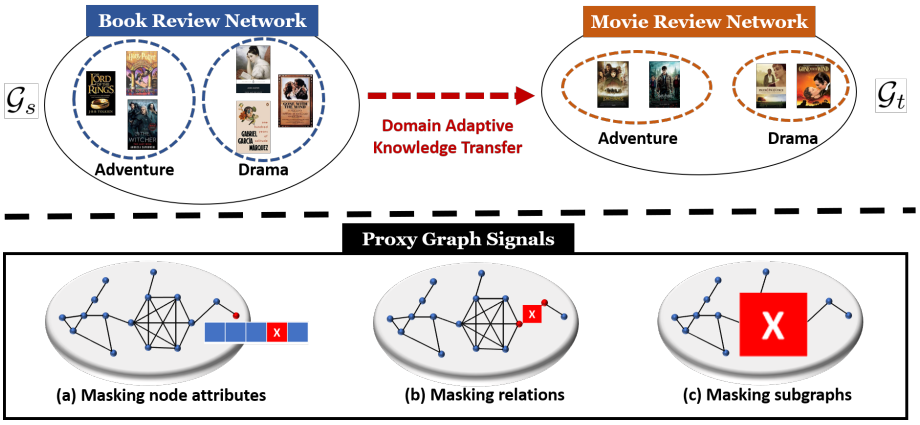


Figure 1: An illustrative example of domain-adaptive graph pre-training. (Top) Domain-adaptive knowledge transfer between the book review network and the movie review network. (Bottom) Proxy graph signals (red masks) at the granularity of nodes, edges, and subgraphs.

risks, one common approach [1] is to assume the data points are sampled i.i.d. from both the source and the target domains. However, this assumption does not hold for the graph-structured data, as the samples in graphs (nodes, edges, subgraphs) are relational and non-i.i.d. in nature. To study the generalization performance of graph mining models, an alternative approach is to define the generalization bound based on the true data distribution [1, 66]. For instance, Theorem 1 [66] provides a population result, which does not rely on the i.i.d. assumption and can be deployed on graphs.

Theorem 1. [66] Let $\langle \mathcal{D}_s, f_s \rangle$ and $\langle \mathcal{D}_t, f_t \rangle$ be the source and the target domains, for any function class $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$ on the input space $\mathcal{X}$, and $\forall h \in \mathcal{H}$, the following inequality holds:

$$\epsilon_t(h) \leq \epsilon_s(h) + d_{\mathcal{H}}(\mathcal{D}_s, \mathcal{D}_t) + \min\{\mathbb{E}_{\mathcal{D}_s}[|f_s - f_t|], \mathbb{E}_{\mathcal{D}_t}[|f_s - f_t|]\}.$$

Problem definition. In this paper, our goal (as shown in Figure 1) is to learn a knowledge transfer function denoted as $g(\cdot)$, such that the knowledge obtained by a GNN model in the source graph $\mathcal{G}_s$ can be transferred to the target graph $\mathcal{G}_t$ and pre-train the GNNs in $\mathcal{G}_t$. Without loss of generality, we assume that 1) the source graph $\mathcal{G}_s$ is well studied by $f_s(\cdot)$ and comes with rich label information for diverse graph signals $\mathbf{s} \in \mathcal{G}_s$, i.e., $\mathcal{L}_s = \{(\mathbf{s}, y) | \mathbf{s} \in \mathcal{G}_s, y \in \mathcal{Y}_s\}$; and 2) the task of interest in the target graph $\mathcal{G}_t$ is novel, and we are only given a handful of annotated graph signals, i.e., $\mathcal{L}_t = \{(\mathbf{s}, y) | \mathbf{s} \in \mathcal{G}_t, y \in \mathcal{Y}_t\}$. Given the notations above, we formally define the problem as follows.

Problem 1. ***Domain-Adaptive Graph Pre-Training***
Given: *(i) source graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s, \mathbf{B}_s)$ with rich label information $\mathcal{L}_s$, (ii) target graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{B}_t)$ with scarce label information $\mathcal{L}_t$, and (iii) user-defined graph neural network architecture.*
Find: *the pre-trained model $h(\cdot)$ that leverages the knowledge obtained from the source graph $\mathcal{G}_s$ and the target graph $\mathcal{G}_t$.*

3 PROPOSED FRAMEWORK MENTORGNN

In this section, we introduce our proposed framework MentorGNN (shown in Figure 2) to address Problem 1. The key challenges of Problem 1 lie in the dual-level data heterogeneity, namely *cross-graph heterogeneity* and *graph-signal heterogeneity*. Then, we dive into two major modules of MentorGNN, i.e., cross-graph adaptation (colored in blue in Figure 2) and curriculum learning (colored in orange in Figure 2), that are designed specifically for addressing the dual-level data heterogeneity.

3.1 Cross-Graph Adaptation via Multi-Scale Encoder-Decoder

The core obstruction of cross-graph adaptation lies in how to effectively translate the knowledge learned from $\mathcal{G}_s$ to $\mathcal{G}_t$ without any supervision of cross-graph association (e.g., partial network alignments [5, 62, 63, 65]). Specifically, given $\mathcal{G}_s$ and $\mathcal{G}_t$, we want to learn a transformation function $g(\cdot)$ that leverages both network structures and node attributes over the entire graph and translate the knowledge between $\mathcal{G}_s$ and $\mathcal{G}_t$, i.e., $(\mathcal{V}_t, \mathcal{E}_t, \mathbf{B}_t) \simeq g((\mathcal{V}_s, \mathcal{E}_s, \mathbf{B}_s))$. However, learning the transformation function $g(\cdot)$ may require a large parameter space $O(m_s m_t)$ and become computationally intractable, especially when both $\mathcal{G}_s$ and $\mathcal{G}_t$ are large. In order to alleviate the computational cost, we propose to learn the translation function $g(\cdot)$ at a coarser resolution instead of directly translating knowledge between $\mathcal{G}_s$ and $\mathcal{G}_t$. The underlying intuition is that many real graphs from distinct domains may share similar high-level organizations. For instance, in Figure 1 (Top), the book review network ($\mathcal{G}_s$) and the movie review network ($\mathcal{G}_t$) come from two distinct domains, but may share similar high-level organizations (e.g., alignments between book genres and movie genres). That is to say, the communities (the adventure books) on the book review network may have related semantic meanings to the communities (the adventure movies) on the movie review network.

Motivated by this observation, we develop a multi-scale encoder-decoder architecture (the module colored in blue in Figure 2), which explores the cluster-within-cluster hierarchies of graphs to better characterize the graph signals at multiple granularities. In particular, the encoder $\mathcal{P}$ learns the multi-scale representation of $\mathcal{G}_s$ by pooling the source graph from the fine-grained representation $\mathbf{X}_s$ to the

coarse-grained representation $\mathbf{X}_s^{(L)}$, while the decoder $\mathcal{U}$ aims to reconstruct the target graph from the coarse-grained representation $\mathbf{X}_t^{(L)}$ to the fine-grained representation $\mathbf{X}_t$. In our paper, we set an identical number of layers L in the encoder and the decoder to make $\mathcal{G}_s$ and $\mathcal{G}_t$ comparable with each other.

Encoder: The encoder is defined as a set of differentiable pooling matricies $\mathcal{P} = \{\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(L)}\}$, where $\mathbf{P}^{(l)} \in \mathbb{R}^{n_s^{(l)} \times n_s^{(l+1)}}$, $n_s^{(l)}$ and $n_s^{(l+1)}$ are the number of super nodes at layer l and layer $l+1$. Specifically, following [64], the differentiable pooling matrix $\mathbf{P}^{(l)}$ at layer l is defined as follows.

$$\mathbf{P}^{(l)} = \text{softmax}(\text{GNN}_{l,pool}(\mathbf{A}_s^{(l)}, \mathbf{X}_s^{(l)})) \tag{2}$$

where each entry $\mathbf{P}^{(l)}(i, j)$ indicates the assignment coefficient from node i at layer l to the corresponding supernode j at layer $l+1$, and $\text{GNN}_{l,pool}$ is the corresponding surrogate GNN to generate the assignment matrix $\mathbf{P}^{(l)}$. It is noteworthy that the number of granularity levels $L \geq 2$ and the maximum number of supernodes in each layer $n_s^{(l)}$ are both hyperparameters selected by the end-users. More implementation details are discussed in Section 4.1. With the differentiable pooling matrix $\mathbf{P}^{(l)}$, the l^{th}-layer coarse-grained representation $\mathbf{X}_s^{(l)}$ of $\mathcal{G}_s$ can be approximated by $\mathbf{X}_s^{(l)} = \mathbf{P}^{(l-1)\prime} \ldots \mathbf{P}^{(1)\prime}\mathbf{X}_s$.

Decoder: Different from the encoder, our decoder is composed of a translation function $g(\cdot)$ and a set of differentiable unpooling matrices $\mathcal{U} = \{\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(L)}\}$. To be specific, the translation function $g(\cdot)$ learns a non-linear mapping between $\mathcal{G}_s$ and $\mathcal{G}_t$ at the coarsest layer L. In our implementation, $g(\cdot)$ is parameterized by a multilayer perceptron (MLP), and the differentiable pooling matrix $\mathbf{U}^{(l)} \in \mathbb{R}^{n_t^{(l+1)} \times n_t^{(l)}}$ at each layer l can be computed as follows.

$$\mathbf{U}^{(l)} = \text{softmax}(\text{GNN}_{l,unpool}(\mathbf{A}_t^{(l)}, \mathbf{X}_t^{(l)})) \tag{3}$$

In contrast to $\text{GNN}_{l,pool}$, $\text{GNN}_{l,unpool}$ is a reverse operation that aims to reconstruct $\mathbf{X}^{(l-1)}$ from its coarse representation $\mathbf{X}^{(l)}$. With the learned translation function $g(\cdot)$ and the differentiable unpooling matrices $\mathcal{U}$, the hidden representation of $\mathcal{G}_t$ can be computed by $\hat{\mathbf{X}}_t = \mathbf{U}^{(1)\prime} \ldots \mathbf{U}^{(L)\prime}\hat{\mathbf{X}}_t^{(L)}$, where $\hat{\mathbf{X}}_t^{(L)} = g(\mathbf{X}_s^{(L)})$ is the translated representation from $\mathcal{G}_s$ to $\mathcal{G}_t$ at the L^{th} layer.

3.2 Graph Signal Comprehension via Curriculum Learning

In the presence of various types of graph signals, the current GNNs pre-trained models are mostly designed in the form of a weighted combination of multiple graph signal encoders by incorporating some hyperparameters. However, manually selecting hyperparameters are often challenging and may lead to sub-optimal performance due to the inductive bias from humans. Here we propose a graph signal re-weighting scheme (the module colored in orange in Figure 2) to capture the contribution of each graph signal towards the downstream tasks. The learned sample weighting scheme specifies a curriculum under which the GNNs will be pre-trained gradually from the easy concepts to the hard concepts. In our problem setting, the curriculum is presented as a sequence of graph signals that are extracted from the given graph. Compared with the previous GNNs pre-training methods, MentorGNN automatically 1) assigns an attention weight towards each graph signal without the requirements of the pre-defined hyperparameters, and 2) supervises the pre-training process to control the risk of negative transfer.

Consider a GNN-based pre-training problem with K types of graph signals extracted from graphs at L levels of resolutions, we formulate the learning objective as follows.

$$\mathcal{L} = \arg\min_{\boldsymbol{\Theta}, \boldsymbol{\theta}} \sum_{l=1}^{L} \sum_{k=1}^{K} \sum_{\mathbf{s}_i^{(l,k)} \in \mathcal{G}_t^{(l)}} g_m(\mathbf{s}_i^{(l,k)}; \boldsymbol{\Theta}) \mathcal{J}(\mathbf{Y}_t, h(\hat{\mathbf{X}}_t^{(l)}, \mathbf{s}_i^{(l,k)}, \boldsymbol{\theta})) + G(g_m(\mathbf{s}_i^{(l,k)}, \boldsymbol{\Theta}), \boldsymbol{\lambda}) \tag{4}$$

where $\mathbf{s}_i^{(l,k)}$ denotes the i^{th} sample of the type-k graph signals (e.g., edges, node attributes, centrality scores) extracted from the l^{th}-layer of $\mathcal{G}_t$, $g_m(\mathbf{s}_i^{(l,k)}; \boldsymbol{\Theta})$ is a function that predicts the attention weights for each training graph signal $\mathbf{s}_i^{(l,k)}$, $h(\hat{\mathbf{X}}_t^{(l)}, \mathbf{s}_i^{(l,k)}, \boldsymbol{\theta})$ denotes the pre-trained GNNs using $\mathbf{s}_i^{(l,k)}$, $\mathcal{J}(\mathbf{Y}_t, h(\hat{\mathbf{X}}_t^{(l)}, \mathbf{s}_i^{(l,k)}, \boldsymbol{\theta}))$ denotes the prediction loss of a downstream task over a handful of labeled examples $\mathbf{Y}_t$, and $G(g_m(\mathbf{s}_i^{(l,k)}, \boldsymbol{\Theta}), \boldsymbol{\lambda})$ is the curriculum regularizer parameterized by the learning threshold $\boldsymbol{\lambda}$. It is worthy to mention that such labeled examples can be any type of graph signals, including the class-memberships of nodes, edges, and even subgraphs. The goal of $\mathcal{L}$ is to automatically learn attention weight $g_m(\mathbf{s}_i^{(l,k)}; \boldsymbol{\Theta})$ towards each graph signal $\mathbf{s}_i^{(l,k)}$ in order to guide the pre-training process of GNNs $h(\cdot)$ via curriculum learning, thus mitigating the risk of negative transfer.

Intuitively, the learning objective in Eq. 4 can be interpreted as a teacher-student training paradigm [8, 34, 70, 78]. In particular, the labeling function $h(\hat{\mathbf{X}}_t^{(l)}, \mathbf{s}_i^{(l,k)}, \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ serves as the student model, which aims to learn expressive representation by predicting the graph signal $\mathbf{s}_i^{(l,k)}$; the teacher model $g_m(\mathbf{s}_i^{(l,k)}; \boldsymbol{\Theta})$ parameterized by $\boldsymbol{\Theta}$ aims to measure the importance of each graph signal $\mathbf{s}_i^{(l,k)}$, and to provide guidance to pre-train GNNs on the target graph $\mathcal{G}_t$. To regularize the learning curriculum, one prevalent choice is to employ some pre-defined curriculums [17, 21, 22, 32, 46, 56, 58, 75, 76], which have been extensively explored in the literature [8, 23, 34, 78]. Here we consider a curriculum regularizer derived from the robust non-convex penalties [23].

$$G(g_m(\mathbf{s}_i^{(l,k)}, \boldsymbol{\Theta}), \lambda_1, \lambda_2) = \frac{1}{2}\lambda_2 g_m^2(\mathbf{s}_i^{(l,k)}; \boldsymbol{\Theta}) - (\lambda_1 + \lambda_2) g_m(\mathbf{s}_i^{(l,k)}; \boldsymbol{\Theta})$$

where $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2\}$ are both positive hyperparameters. Note that $G(g_m(\mathbf{s}_i^{(l,k)}, \boldsymbol{\Theta}), \lambda_1, \lambda_2)$ is a convex function in terms of $g_m(\mathbf{s}_i^{(l,k)}, \boldsymbol{\Theta})$ and thus the closed-form solution of our learning curriculum can be easily obtained as follows.

$$g_m(\mathbf{s}_i^{(l,k)}; \boldsymbol{\Theta}^*) = \begin{cases} 1(\mathcal{J}(\mathbf{Y}_t, h(\hat{\mathbf{X}}_t^{(l)}, \mathbf{s}_i^{(l,k)}, \boldsymbol{\theta})) \leq \lambda_1) & \lambda_2 = 0 \\ \min(\max(0, 1 - \frac{\mathcal{J}(\mathbf{Y}_t, h(\hat{\mathbf{X}}_t^{(l)}, \mathbf{s}^{(l,k)}, \boldsymbol{\theta})) - \lambda_1}{\lambda_2}), 1) & \text{Otherwise} \end{cases} \tag{5}$$

The learning threshold $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2\}$ plays a key role in controlling the learning pace of MentorGNN. When $\lambda_2 = 0$, the algorithm will only select the "easy" graph signals of $\mathcal{J}(\mathbf{Y}_t, h(\hat{\mathbf{X}}_t^{(l)}, \mathbf{s}_i^{(l,k)}, \boldsymbol{\theta})) \leq \lambda_1$ in training labeling function $h(\cdot)$, which is close to the binary scheme in self-paced learning [23, 34]. When $\lambda_2 \neq 0$, $g_m(\mathbf{s}_i^{(l,k)}; \boldsymbol{\Theta}^*)$

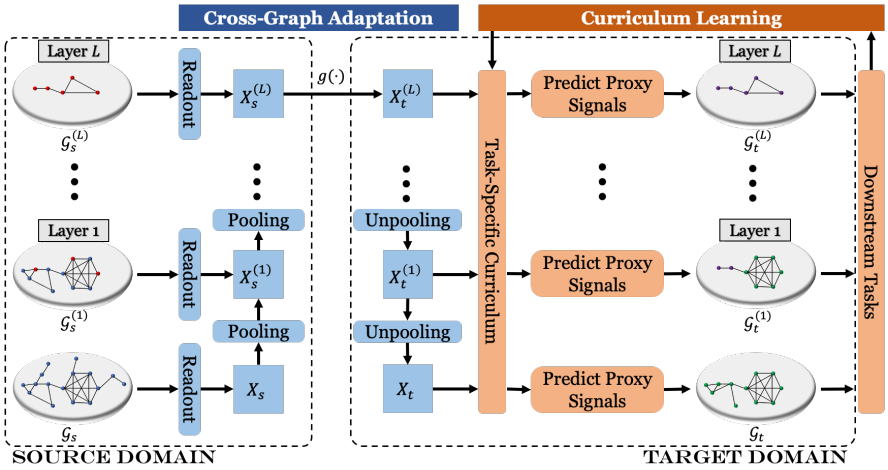


Figure 2: An overview of the MentorGNN, which is composed of two modules, namely cross-graph adaptation (colored in blue) and curriculum learning (colored in orange).

Algorithm 1 The MentorGNN learning framework.

Require: : (i) source graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s, \mathbf{B}_s)$ with rich label information $\mathcal{L}_s$, (ii) target graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{B}_t)$ with scarce label information $\mathcal{L}_t$, (iii) user-defined graph neural network architecture $h(\cdot)$ for pre-training, (iv) parameters λ_1, λ_2, and ξ.

Ensure: : (i) the knowledge transfer function $g(\cdot)$, (ii) the pre-trained model $h(\cdot)$ that leverages the knowledge obtained from both $\mathcal{G}_s$ and $\mathcal{G}_t$.

1: Initialize $\boldsymbol{\theta}$ and $\boldsymbol{\Theta}$.
2: **while** Stopping criterion is not satisfied **do**
3: Fetch a collection of graph signals $\mathbf{s}_i^{(l,k)} \in \mathcal{G}_t$.
4: **if** Update teacher **then**
5: Update $\boldsymbol{\Theta}$ by taking a gradient step for $\mathcal{L}$.
6: Compute learning curriculum $g_m(\mathbf{s}_i^{(l,k)}; \boldsymbol{\Theta})$ via the closed-form solution in Eq. 5.
7: **end if**
8: **if** Update student **then**
9: Update $\boldsymbol{\theta}$ and $g(\cdot)$ by taking a gradient step for $\mathcal{J}(\mathbf{Y}_t, h(\hat{\mathbf{X}}_t^{(l)}, \mathbf{s}^{(l,k)}, \boldsymbol{\theta}))$.
10: Compute $\{\hat{\mathbf{X}}_t^{(1)}, \ldots, \hat{\mathbf{X}}_t^{(L)}\}$ via multi-scale encoder-decoder (Subsection 3.1).
11: **end if**
12: Augment the values of λ_1 and λ_2 by the ratio ξ.
13: **end while**

will return continuous values in $[0, 1]$, and the graph signals with the loss of $\mathcal{J}(\mathbf{Y}_t, h(\hat{\mathbf{X}}_t^{(l)}, \mathbf{s}_i^{(l,k)}, \boldsymbol{\theta})) \leq \lambda_1 + \lambda_2$ will not be selected in pre-training. In practice, we gradually augment the value of $\lambda_1 + \lambda_2$ to enforce MentorGNN learning from the "easy" graph signals to the "hard" graph signals by mimicking the cognitive process of humans and animals [2]. The pseudo-code of MentorGNN is provided in Algorithm 1, which is designed in an alternative-updating fashion [53]. In each iteration, when we train the student model $h(\hat{\mathbf{X}}_t^{(l)}, \mathbf{s}_i^{(l,k)}, \boldsymbol{\theta}))$, we keep $\boldsymbol{\Theta}$ fixed and minimize the prediction loss $\mathcal{J}(\mathbf{Y}_t, h(\hat{\mathbf{X}}_t^{(l)}, \mathbf{s}^{(l,k)}, \boldsymbol{\theta}))$; when we train the teacher model $g_m(\mathbf{s}_i^{(l,k)}; \boldsymbol{\Theta})$, we keep $\boldsymbol{\theta}$ fixed and update the learning curriculum to be used by the student model in the next iteration; at last, we augment the value of Augment the values of λ_1 and λ_2 by the ratio ξ for the next iteration.

3.3 Generalization Bound of MentorGNN

Given a source graph $\mathcal{G}_s$ and a target graph $\mathcal{G}_t$, what generalization performance can we achieve with MentorGNN in pre-training GNNs? Here, we present two approaches towards the generalization bound of MentorGNN in the presence of multiple types of graph signals at different granularities: one by a union bound argument and the other relying on the graph signal learning curriculum. Let $f_s(\cdot)$ be the true labeling function in $\mathcal{G}_s$, and $f_t^{(l,k)}(\cdot)$ be the true labeling function for the k^{th} type graph signals $\mathbf{s}^{(l,k)}$ at the l^{th} level of $\mathcal{G}_t$. One straightforward idea is to leverage the generalization error between $f_s(\cdot)$ and each $f_t^{(l,k)}(\cdot)$ by applying Theorem 1 $l \times k$ times. Following this idea, we can obtain the following worst-case generalization bound of MentorGNN, which largely depends on the largest generalization error between $f_s(\cdot)$ and $f_t^{(l,k)}(\cdot)$.

Corollary 1. (Worst Case) Given $\mathcal{G}_s$ and $\mathcal{G}_t$, $f_s(\cdot)$ is the labeling function in $\mathcal{G}_s$, and $f_t^{(l,k)}(\cdot)$ is the labeling function for the k^{th} type graph signals $\mathbf{s}^{(l,k)}$ at the l^{th} level of granularity in $\mathcal{G}_t$. Then, for any function class $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$, and $\forall h \in \mathcal{H}$, the following inequality holds:

$$\epsilon_t(h) \leq \epsilon_s(h) + d_{\mathcal{H}}(\mathcal{G}_s, \mathcal{G}_t) + \max_{l,k}\{\min\{\mathbb{E}_{\mathcal{G}_s}[|f_s - f_t^{(l,k)}|], \mathbb{E}_{\mathcal{G}_t}[|f_s - f_t^{(l,k)}|]\}\} \quad (6)$$

The worst-case generalization bound shown in Corollary 1 could be pessimistic in practice, especially when the graphs are large and noisy. However, extensive work [21, 22, 46] has empirically shown that leveraging multiple types of graph signals often leads to improved performance in many application domains, even in the presence of noisy data and irrelevant features. The key observation is that the information from multiple types of graph signals

is often redundant and complementary. That is to say, when the majority of graph signals are related, a few irrelevant graph signals may not hurt the overall generalization performance too much. Hence, the natural question is: *can we obtain a better generalization bound than the one shown in Corollary 1?* To answer this question, we present a re-weighting case of the generalization bound for MentorGNN, that is developed based on the obtained learning curriculum $g_m(\mathbf{s}^{(l,k)}, \Theta)$ as follows.

Corollary 2. (Re-weighting Case) Given $\mathcal{G}_s$ and $\mathcal{G}_t$, $f_s(\cdot)$ is the labeling function in $\mathcal{G}_s$, and $f_t^{(l,k)}(\cdot)$ is the labeling function for the k^{th} type graph signals $\mathbf{s}^{(l,k)}$ at the l^{th} level of granularity in $\mathcal{G}_t$. Then, for any function class $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$, $\forall h \in \mathcal{H}$, and $\sum_{l,k,i} g_m(\mathbf{s}_i^{(l,k)}, \Theta) = 1$, the following inequality holds:

$$\epsilon_t(h) \leq \epsilon_s(h) + d_{\mathcal{H}}(\mathcal{G}_s, \mathcal{G}_t) + \sum_{l,k,i} g_m(\mathbf{s}_i^{(l,k)}, \Theta) \cdot \min\{\mathbb{E}_{\mathcal{G}_s}[|f_s - f_t^{(l,k)}|], \mathbb{E}_{\mathcal{G}_t}[|f_s - f_t^{(l,k)}|]\}\}$$

Remark 1: Corollary 1 and Corollary 2 are both based on the true data distribution, and as such, the derived generalization bound might slightly deviate from the empirical results in practice. However, we argue that the state-of-the-art GNNs [33, 55, 61] are reliable and accurate, whose empirical performance is very close to the true labeling function. Therefore, our theoretical results can well approximate the generalization bound of MentorGNN in practice.

Remark 2: Corollary 2 reduces the multi-type multi-granularity of graph signals into an aggregated version with a linear combination using $g_m(\mathbf{s}_i^{(l,k)}, \Theta)$. In fact, the worst-case generalization bound can be considered as a special case of the generalization bound shown in Corollary 2. Based on the following inequality, it is easy to prove that Corollary 2 provides a much tighter generalization bound than Corollary 1.

$$\sum_{l,k,i} g_m(\mathbf{s}_i^{(l,k)}, \Theta) \min\{\mathbb{E}_{\mathcal{G}_s}[|f_s - f_t^{(l,k)}|], \mathbb{E}_{\mathcal{G}_t}[|f_s - f_t^{(l,k)}|]\}\} \leq \max_{l,k}\{\min\{\mathbb{E}_{\mathcal{G}_s}[|f_s - f_t^{(l,k)}|], \mathbb{E}_{\mathcal{G}_t}[|f_s - f_t^{(l,k)}|]\}\}.$$

4 EXPERIMENTS

We evaluate the performance of MentorGNN on both synthetic and real graphs in the following aspects:

- **Effectiveness:** We report comparison results with a diverse set of baseline methods, including state-of-the-art GNNs, graph pre-training models, and transfer learning models, in the single-source graph transfer and the multi-source graph transfer settings.
- **Case study:** We conduct a case study to study the generalization performance of MentorGNN in the dynamic protein-protein interaction graphs.
- **Parameter sensitivity and scalability analysis:** We study the parameter sensitivity and the scalability of MentorGNN on both synthetic and real graphs.

4.1 Experimental Setup

Data sets: Cora [41] data set is a citation network consisting of 2,708 scientific publications and 5,429 edges. Each edge in the graph represents the citation from one paper to another. CiteSeer [41] data set consists of 3,327 scientific publications, which could be categorized into six classes, and this citation network has 9,228 edges. PubMed [45] is a diabetes data set, which consists of 19,717 scientific publications in three classes and 88,651 edges. The Reddit [18] data set was extracted from Reddit posts in September 2014. After pre-processing, three Reddit graphs are extracted and denoted as Reddit1, Reddit2, and Reddit3, respectively. Reddit1 consists of 4,584 nodes and 19,460 edges; Reddit2 consists of 3,765 nodes and 30,494 edges; and Reddit3 consists of 4,329 nodes and 35,191 edges.
Baselines: We compare our method with the following baselines: (1) GCN [33]: graph convolutional network, which is directly trained and tested on the target graph; (2) GAT [55]: graph attention network, which is directly trained and tested on the target graph; (3) DGI [56]: deep graph infomax, which is directly trained and tested on the target graph; (4) GPA [20]: a policy network for transfer learning across graphs. Since GPA is designed for zero-shot setting, we fine-tune the pre-trained model with 100 iterations in the downstream tasks, and then make the final prediction; (5) UDA-GCN [59]: an unsupervised domain-adaptive graph convolutional network; (6) GPT-GNN [22]: a self-supervised attributed graph generative framework to pre-train a GNN by capturing both the structural and semantic properties of the graph; (7) Pretrain-GNN [21]: a self-supervised pre-training model that learns both local and global representations of each node; (8) MentorGNN-V: one variant of MentorGNN, which only considers node attributes as graph signals; and (9) MentorGNN-C: one variant of MentorGNN, which is designed without the curriculum learning module.
Implementation details: The data and code are available in the link*. In the implementation of MentorGNN, we consider two types ($K = 2$) of graph signals, i.e., node attributes and edges, at $L = 3$ levels of granularity. The output dimension of the first level of granularity is 500, and the output dimension of the second level of granularity is 100. We use Adam [31] as the optimizer with a learning rate of 0.005 and a two-layer GAT [55] with a hidden layer of size 50 as our backbone structure. MentorGNN and its variants are trained for a maximum of 2000 episodes. The experiments are performed on a Windows machine with eight 3.8GHz Intel Cores and a single 16GB RTX 5000 GPU.

4.2 Single-Source Graph Transfer

In this subsection, we first consider the single-source graph transfer setting (following [20]), where we are given a single source graph (e.g., Cora [41], CiteSeer [41], and PubMed [45]) and a single target graph (e.g., Reddit1, Reddit 2, Reddit 3 [18]). Our goal is to pre-train GNNs across two graphs and then fine-tune the model to perform node classification on the target graph. We split the data set into training, validation, and test sets with the fixed ratio of 4%:16%:80%, respectively. Each experiment is repeated five times, and we report the average accuracy and the standard deviation of all methods across different data sets in Tables 2 - 4. Results reveal that our proposed method and its variants outperform all baselines. Specifically, we have the following observations: (1) compared with the traditional GNNs, e.g., GCN, GAT and DGI, our method and its variants can further boost the performance by utilizing the knowledge learned from both the source graph and the target graph; (2)

*https://github.com/Leo02016/MentorGNN

Table 2: Accuracy of node classification in the single-source graph transfer setting, from a single source graphs (Cora) to a single target graph (Reddit1 or Reddit2 or Reddit3).

Method	Cora → Reddit1	Cora → Reddit2	Cora → Reddit3
GCN	0.8736 ± 0.0151	0.8996 ± 0.0158	0.8816 ± 0.0050
GAT	0.9420 ± 0.0154	0.9241 ± 0.0094	0.8985 ± 0.0088
DGI	0.7845 ± 0.0208	0.9062 ± 0.0071	0.8388 ± 0.0098
GPA	0.7011 ± 0.0116	0.7157 ± 0.0053	0.7271 ± 0.0063
UDA-GCN	0.9323 ± 0.0106	0.9378 ± 0.0079	0.9528 ± 0.0058
GPT-GNN	**0.9570 ± 0.0097**	0.9614 ± 0.0065	0.9548 ± 0.0092
Pretrain-GNN	0.9484 ± 0.0105	0.9351 ± 0.0100	0.9574 ± 0.0090
MentorGNN-V	0.9448 ± 0.0131	0.9584 ± 0.0045	0.9454 ± 0.0071
MentorGNN-C	0.9508 ± 0.0097	0.9640 ± 0.0060	0.9655 ± 0.0072
MentorGNN	0.9562 ± 0.0059	**0.9815 ± 0.0053**	**0.9741 ± 0.0046**

Table 3: Accuracy of node classification in the single-source graph transfer setting, from a single source graphs (CiteSeer) to a single target graph (Reddit1 or Reddit2 or Reddit3).

Method	CiteSeer → Reddit1	CiteSeer → Reddit2	CiteSeer → Reddit3
GCN	0.8736 ± 0.0151	0.8996 ± 0.0158	0.8816 ± 0.0050
GAT	0.9420 ± 0.0154	0.9241 ± 0.0094	0.8985 ± 0.0088
DGI	0.7845 ± 0.0208	0.9062 ± 0.0071	0.8388 ± 0.0098
GPA	0.6932 ± 0.0105	0.7162 ± 0.0055	0.7273 ± 0.0065
UDA-GCN	0.9027 ± 0.0082	0.9540 ± 0.0092	0.9610 ± 0.0131
GPT-GNN	0.9544 ± 0.0078	0.9528 ± 0.0068	0.9541 ± 0.0100
Pretrain-GNN	0.9386 ± 0.0084	0.9488 ± 0.0098	0.9581 ± 0.0113
MentorGNN-V	0.9589 ± 0.0043	0.9637 ± 0.0072	0.9763 ± 0.0034
MentorGNN-C	0.9521 ± 0.0091	0.9646 ± 0.0064	0.9779 ± 0.0045
MentorGNN	**0.9617 ± 0.0028**	**0.9834 ± 0.0048**	**0.9806 ± 0.0026**

Table 4: Accuracy of node classification in the single-source graph transfer setting, from a single source graphs (PubMed) to a single target graph (Reddit1 or Reddit2 or Reddit3).

Method	PubMed → Reddit1	PubMed → Reddit2	PubMed → Reddit3
GCN	0.8736 ± 0.0151	0.8996 ± 0.0158	0.8816 ± 0.0050
GAT	0.9420 ± 0.0154	0.9241 ± 0.0094	0.8985 ± 0.0088
DGI	0.7845 ± 0.0208	0.9062 ± 0.0071	0.8388 ± 0.0098
GPA	0.6907 ± 0.0114	0.7146 ± 0.0050	0.7210 ± 0.0103
UDA-GCN	0.9381 ± 0.0064	0.9460 ± 0.0335	0.9506 ± 0.0099
GPT-GNN	0.9582 ± 0.0087	0.9546 ± 0.0060	0.9553 ± 0.0063
Pretrain-GNN	0.9405 ± 0.0059	0.9625 ± 0.0131	0.9566 ± 0.0088
MentorGNN-V	0.9574 ± 0.0129	0.9761 ± 0.0059	0.9454 ± 0.0071
MentorGNN-C	**0.9602 ± 0.0028**	0.9749 ± 0.0043	0.9734 ± 0.0104
MentorGNN	0.9575 ± 0.0072	**0.9839 ± 0.0025**	**0.9785 ± 0.0050**

Table 5: Accuracy of node classification in the setting of multi-source graph transfer, from multiple source graphs (Cora, CiteSeer, and PubMed) to a single target graph (Reddit1 or Reddit2 or Reddit3).

Method	Reddit1	Reddit2	Reddit3
GCN	0.8736 ± 0.0151	0.8996 ± 0.0158	0.8816 ± 0.0050
GAT	0.9420 ± 0.0154	0.9241 ± 0.0094	0.8985 ± 0.0088
DGI	0.7845 ± 0.0208	0.9062 ± 0.0071	0.8388 ± 0.0098
GPA	0.7053 ± 0.0091	0.7193 ± 0.0027	0.7308 ± 0.0033
MentorGNN-V	0.9586 ± 0.0054	0.9848 ± 0.0033	0.9801 ± 0.0024
MentorGNN-C	**0.9634 ± 0.0055**	0.9844 ± 0.0045	0.9795 ± 0.0051
MentorGNN	0.9621 ± 0.0015	**0.9857 ± 0.0020**	**0.9811 ± 0.0025**

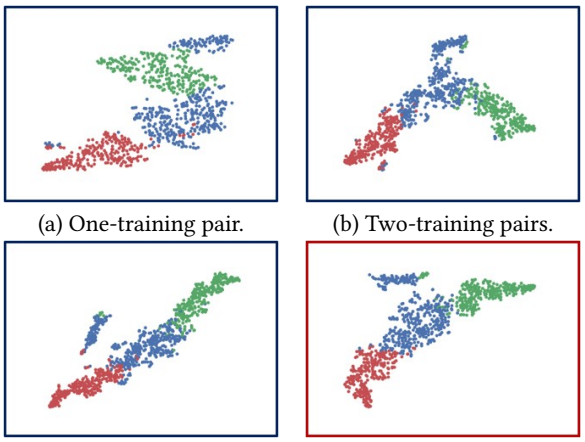
the performance of GPA is worse than GCN, GAT and DGI. One conjecture is that the performance of GPA largely suffers from the label scarcity issue in our setting. In particular, the results of GPA in [20] require 67% labeled samples for training on Reddit1, while we are only given 4% labeled samples of Reddit1 for training. (3) UDA-GCN, GPT-GNN and Pretrain-GNN outperform the traditional GNNs in most cases, which implies that these pre-training algorithms indeed transfer the useful knowledge from the source graph and thus boost the performance of node classification in the target graph; (4) compared with the pre-training algorithms (UDA-GCN, GPT-GNN and Pretrain-GNN), MentorGNN further boosts the performance by more than 3% in the setting of CiteSeer → Reddit2. (5) compared with MentorGNN-V which discards the link signal, MentorGNN could further improve the performance by up to 3.31% by utilizing the structural information of the graph; and (6) compared with MentorGNN-C which does not utilize the curriculum learning, MentorGNN boosts the performance by up to 1.88% in the setting of CiteSeer → Reddit2. Overall, the comparison experiments verify the fact that MentorGNN largely alleviates the impact of negative transfer and improves the generalization performance across all data sets.

4.3 Multi-Source Graph Transfer

In this subsection, we evaluate our proposed model MentorGNN in the multi-Source graph transfer setting, where multiple source graphs are given for pre-training GNNs. In our experiments, we use three source citation networks (i.e., Cora, CiteSeer, and PubMed) as the source graphs, and our goal is to transfer the knowledge extracted from multiple source graphs to improve the performance of the node classification task on a single target graph (e.g., Reddit1, Reddit 2, Reddit 3). Similar to the single-source graph transfer setting, we use the same data split scheme to generate the training set, the validation set, and the test set. The full results in terms of prediction accuracy and standard deviation over five runs are reported in Table 5. Note that some of our baseline graph pre-training methods (i.e., UDA-GCN, GPT-GNN, Pretrain-GNN) are not designed for the multi-source graph transfer setting. Thus, we exclude them from Table 5. In general, our proposed method and its variants outperform all the baselines. Moreover, by combining the results (Table 2 - 4) in the single source-graph setting, it is interesting to see that the performances of GPA and MentorGNN are both slightly improved when we leverage multiple source graphs simultaneously. For instance, when we consider the Reddit1 as the target graph, the accuracy of GPA and our method is improved by 0.42% and 0.72% respectively compared with the single-source graph transfer setting (i.e., Cora → Reddit1).

4.4 Case Study: Metabolic Pattern Modeling on Protein-Protein Interaction Graph

Protein analysis is of great significance in many biological applications [57]. In this case study, we aim to apply MentorGNN to study and capture the metabolic patterns of yeast cells in the dynamic setting. To be specific, given a dynamic protein-protein interaction (PPI) graph, e.g., DPPIN-Breitkreutz [11], it consists of five snapshots $\mathcal{G} = \{\mathcal{G}^1, \ldots, \mathcal{G}^5\}$. Each node represents a protein, and each timestamped edge stands for the interaction between two proteins.



(a) One-training pair. (b) Two-training pairs. (c) Three-training pairs. (d) Ground truth.

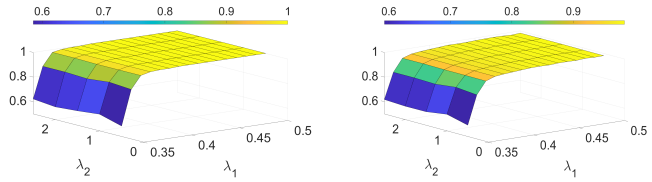
Figure 3: Metabolic pattern modeling on DPPIN-Breitkreutz graph. (a)-(c) show the network layouts of the generated $\mathcal{G}^5$ by learning from $\{\langle \mathcal{G}^1, \mathcal{G}^2 \rangle\}$, $\{\langle \mathcal{G}^1, \mathcal{G}^2 \rangle, \langle \mathcal{G}^2, \mathcal{G}^3 \rangle\}$, and $\{\langle \mathcal{G}^1, \mathcal{G}^2 \rangle, \langle \mathcal{G}^2, \mathcal{G}^3 \rangle, \langle \mathcal{G}^3, \mathcal{G}^4 \rangle\}$, respectively; and (d) shows the network layout of the ground-truth $\mathcal{G}^5$.

In order to train MentorGNN, we use the past snapshot as the source graph and the future snapshot as the target graph. The five snapshots naturally form four pairs of ⟨source graph, target graph⟩ in the chronological order, i.e., $\langle \mathcal{G}^1, \mathcal{G}^2 \rangle, \langle \mathcal{G}^2, \mathcal{G}^3 \rangle, \langle \mathcal{G}^3, \mathcal{G}^4 \rangle, \langle \mathcal{G}^4, \mathcal{G}^5 \rangle$. In our implementation, we treat the first three pairs as the training set and use the remaining one for testing. After obtaining the knowledge transfer function $g(\cdot)$ via MentorGNN, we aim to reconstruct the last snapshot $\mathcal{G}^5$ by adapting from $\mathcal{G}^4$. Figure 3 shows the synthetic $\mathcal{G}^5$ generated by MentorGNN using different portions of the training set as well as the ground-truth $\mathcal{G}^5$. In detail, Figure 3(a)-(c) show the generated $\mathcal{G}^5$ by learning from $\{\langle \mathcal{G}^1, \mathcal{G}^2 \rangle\}$, $\{\langle \mathcal{G}^1, \mathcal{G}^2 \rangle, \langle \mathcal{G}^2, \mathcal{G}^3 \rangle\}$, and $\{\langle \mathcal{G}^1, \mathcal{G}^2 \rangle, \langle \mathcal{G}^2, \mathcal{G}^3 \rangle, \langle \mathcal{G}^3, \mathcal{G}^4 \rangle\}$, respectively; and Figure 3(d) shows the ground-truth $\mathcal{G}^5$ by using t-SNE [54]. Each dot is the projected node in the embedding space, and the different colors correspond to three substructures of the PPI network. In general, we observe that Figure 3(c) is most similar to the ground-truth in Figure 3(d), Figure 3(b) is the next, and Figure 3(a) is the least similar one. Through this comparison, we know that our MentorGNN captures the temporal evolution pattern, and the generated molecule graphs are more trustworthy given more temporal information.

4.5 Parameter Sensitivity Analysis

In this subsection, we study the impact of the learning thresholds λ_1 and λ_2 on MentorGNN. In Figure 4, we conduct the case study in the single-source graph transfer setting of Cora → Reddit2. Particularly, we report the training accuracy and the test accuracy of our model with a diverse range of λ_1 and λ_2. In general, we observe that: (1) the model often achieves the best training accuracy and test accuracy when $\lambda_2 = 1$; and (2) given a fixed λ_2, both the training accuracy and test accuracy are improved with increasing λ_1. It is because the learned teacher model enforces the pre-trained GNNs to learn from the "easy concepts" (i.e., small λ_1) to the "hard concepts"

(i.e., large λ_1). In this way, the pre-trained GNNs encode more and more informative graph signals in the learned graph representation and thus achieve better performance.



(a) Training accuracy (b) Testing accuracy

Figure 4: Parameter analysis w.r.t. the learning thresholds λ_1 and λ_2 on Cora → Reddit2.

4.6 Scalability Analysis

In this subsection, we study the scalability of MentorGNN in the setting of single-source graph transfer, where the source graph is Cora while the target graphs are a collection of synthetic graphs with an increasing number of nodes. To control the size of the target graphs, we generate the synthetic target graphs via ER algorithm [7]. In Figure 5, we present the running time of MentorGNN with the different number of layers ($L = 2, 3, 4$) over the increasing number of nodes in the target graphs. All the results reported in Figure 5 are based on 1000 training epochs. In general, we observe that 1) the complexity of the proposed method is roughly quadratic w.r.t. the number of nodes, and 2) the running time only slightly increases when we increase the number of layers in MentorGNN.

5 RELATED WORK

In this section, we briefly review the related works in the context of pre-training strategies for graphs and domain adaptation.

Pre-Training Strategies for Graphs. To tackle the label scarcity in graph representation learning, pre-training strategies for GNNs have been proposed [14, 21, 22, 25, 28, 39, 42, 46, 47, 52]. The core idea is to capture the generic graph information across different tasks, and transfer it to the target task to reduce the required amount of labeled data and domain-specific features. Some recent methods effectively extract knowledge at the level of both individual nodes and the entire graphs via various techniques, including contrastive predictive coding [9, 27, 29, 30, 36, 67–69], context prediction [22, 46], and mutual information maximization [26, 46, 50]. To be specific, in [21], the authors design three topology-based tasks for pre-training GNNs to learn the transferable structural information, (i.e., node-level pre-training, subgraph-level pre-training, graph-level pre-training); [21] propose a pre-training framework for GNNs by combining the self-supervised pre-training strategy on the node level (i.e., neighborhood prediction and attributes masking) and the supervised pre-training strategy on the graph level (i.e., graph property prediction). However, the current pre-training strategies often lack the capability of domain adaptation, thus limiting their ability to leverage valuable information from other data sources. In this paper, we propose a novel method named MentorGNN that enables pre-training GNNs across graphs and domains.

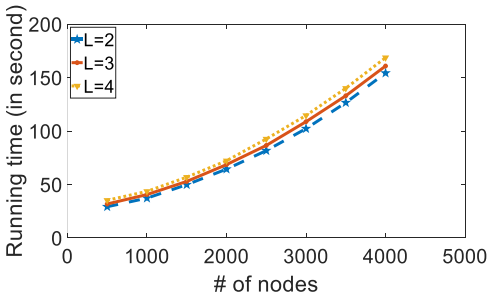


Figure 5: Scalability analysis.

Domain Adaptation. To ensure that the learned knowledge is transferable between the source domains (or tasks) and the target domains (or tasks), many efforts tend to learn the domain-invariant or task-invariant representations, such as [37, 66, 73, 74]. After the domain adaptation demand meets the graph-structured data, the transferability of GNNs has been theoretically analyzed in terms of convolution operations [35]. Then to learn the domain-invariant representations with GNNs, several domain-adaptive GNN models are proposed [19, 43, 44, 59, 77]. For example, UDA-GCN is proposed [59] in the unsupervised learning setting to learn invariant representations via a dual graph convolutional network component. Pragmatically, in order to efficiently label the nodes in a target graph to reduce the annotation cost of training, GPA [20] is proposed to learn a transferable policy with reinforcement learning techniques among fully labeled source domain graphs, which can be directly generalized to unlabeled target domain graphs. Despite the success of domain adaptation on graphs, little effort has been made to analyze the generalization performance of the deep learning models on relational data (e.g., graphs). Here we propose a new generalization bound for domain-adaptive pre-training on graph-structured data.

6 CONCLUSION

Pre-training deep learning models is of key importance in many graph mining tasks. In this paper, we study a novel problem named domain-adaptive graph pre-training, which aims to pre-train GNNs from diverse graph signals across disparate graphs. To address this problem, we present an end-to-end framework named MentorGNN, which 1) automatically learns a knowledge transfer function and 2) generates a domain-adaptive curriculum for pre-training GNNs from the source graph to the target graph. We also propose a new generalization bound for MentorGNN in the domain-adaptive pre-training. Extensive empirical results show that the pre-trained GNNs under MentorGNN with fine-tuning using a few labels achieve significant improvements in the settings of single-source graph transfer and multi-source graph transfer.

ACKNOWLEDGMENTS

This work is supported by National Science Foundation under Award No. IIS-1947203, IIS-2117902, IIS-2137468, IIS-19-56151, IIS-17-41317, and IIS 17-04532 and the C3.ai Digital Transformation Institute. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

REFERENCES

[1] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Mach. Learn.* 79, 1-2 (2010), 151–175.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009 (ACM International Conference Proceeding Series, Vol. 382)*, Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman (Eds.). ACM, 41–48.

[3] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F. Stewart, and Jimeng Sun. 2017. GRAM: Graph-based Attention Model for Healthcare Representation Learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 787–795.

[4] Fabrizio de Vico Fallani, Jonas Richiardi, Mario Chavez, and Sophie Achard. 2014. Graph analysis of functional brain networks: practical issues in translational neuroscience. *Philosophical Transactions of the Royal Society B: Biological Sciences* 369, 1653 (2014), 20130521.

[5] Boxin Du, Si Zhang, Yuchen Yan, and Hanghang Tong. 2021. New Frontiers of Multi-Network Mining: Recent Developments and Future Trend. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 4038–4039.

[6] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.). 2224–2232.

[7] P ERDdS and A R&wi. 1959. On random graphs I. *Publ. math. debrecen* (1959).

[8] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. 2018. Learning to Teach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[9] Shengyu Feng, Baoyu Jing, Yada Zhu, and Hanghang Tong. 2022. Adversarial Graph Contrastive Learning with Information Regularization. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. ACM, 1362–1371.

[10] Dongqi Fu, Liri Fang, Ross Maciejewski, Vetle I. Torvik, and Jingrui He. 2022. Meta-Learned Metrics over Multi-Evolution Temporal Graphs. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*. ACM, 367–377.

[11] Dongqi Fu and Jingrui He. 2021. DPPIN: A biological repository of dynamic protein-protein interaction network data. *arXiv preprint arXiv:2107.02168* (2021).

[12] Dongqi Fu, Zhe Xu, Bo Li, Hanghang Tong, and Jingrui He. 2020. A View-Adversarial Framework for Multi-View Network Embedding. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*. ACM, 2025–2028.

[13] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1263–1272.

[14] Andrey Gritsenko, Yuan Guo, Kimia Shayestehfard, Armin Moharrer, Jennifer Dy, and Stratis Ioannidis. 2021. Graph Transfer Learning. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 141–150.

[15] Maxime Guye, Gaelle Bettus, Fabrice Bartolomei, and Patrick J Cozzone. 2010. Graph theoretical analysis of structural and functional connectivity MRI in normal and pathological brain networks. *Magnetic Resonance Materials in Physics, Biology and Medicine* 23, 5 (2010), 409–421.

[16] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Eng. Bull.* 40, 3 (2017), 52–74.

[17] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 1024–1034.

[18] William L. Hamilton, Justine Zhang, Cristian Danescu-Niculescu-Mizil, Dan Jurafsky, and Jure Leskovec. 2017. Loyalty in Online Communities. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017*. AAAI Press, 540–543.

[19] Xueting Han, Zhenhuan Huang, Bang An, and Jing Bai. 2021. Adaptive Transfer Learning on Graph Neural Networks. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. ACM, 565–574.

[20] Shengding Hu, Zheng Xiong, Meng Qu, Xingdi Yuan, Marc-Alexandre Côté, Zhiyuan Liu, and Jian Tang. 2020. Graph Policy Network for Transferable Active Learning on Graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[21] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[22] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. GPT-GNN: Generative Pre-Training of Graph Neural Networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 1857–1867.

[23] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G. Hauptmann. 2015. Self-Paced Curriculum Learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. AAAI Press, 2694–2700.

[24] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 2309–2318.

[25] Xunqiang Jiang, Tianrui Jia, Yuan Fang, Chuan Shi, Zhe Lin, and Hui Wang. 2021. Pre-training on Large-Scale Heterogeneous Graph. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. ACM, 756–766.

[26] Xunqiang Jiang, Yuanfu Lu, Yuan Fang, and Chuan Shi. 2021. Contrastive Pre-Training of GNNs on Heterogeneous Graphs. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*. ACM, 803–812.

[27] Baoyu Jing, Chanyoung Park, and Hanghang Tong. 2021. HDMI: High-order Deep Multiplex Infomax. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. ACM / IW3C2, 2414–2424.

[28] Baoyu Jing, Yuejia Xiang, Xi Chen, Yu Chen, and Hanghang Tong. 2021. Graph-MVP: Multi-View Prototypical Contrastive Learning for Multiplex Graphs. *arXiv preprint arXiv:2109.03560* (2021).

[29] Baoyu Jing, Yuchen Yan, Yada Zhu, and Hanghang Tong. 2022. COIN: Co-Cluster Infomax for Bipartite Graphs. *arXiv preprint arXiv:2206.00006* (2022).

[30] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised Contrastive Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[31] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[32] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *CoRR* abs/1611.07308 (2016). arXiv:1611.07308

[33] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[34] M. Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-Paced Learning for Latent Variable Models. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*. Curran Associates, Inc., 1189–1197.

[35] Ron Levie, Wei Huang, Lorenzo Bucci, Michael M. Bronstein, and Gitta Kutyniok. 2021. Transferability of Spectral Graph Convolutional Neural Networks. *J. Mach. Learn. Res.* 22 (2021), 272:1–272:59.

[36] Bolian Li, Baoyu Jing, and Hanghang Tong. 2022. Graph Communal Contrastive Learning. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. ACM, 1203–1213.

[37] Bo Li, Yezhen Wang, Shanghang Zhang, Dongsheng Li, Kurt Keutzer, Trevor Darrell, and Han Zhao. 2021. Learning Invariant Representations and Risks for Semi-Supervised Domain Adaptation. (2021), 1104–1113.

[38] Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. 2021. Disentangled Contrastive Learning on Graphs. (2021), 21872–21884.

[39] Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, Hongyu Guo, and Jian Tang. 2022. Pre-training Molecular Graph Representation with 3D Geometry. (2022).

[40] Xin Liu, Haojie Pan, Mutian He, Yangqiu Song, Xin Jiang, and Lifeng Shang. 2020. Neural subgraph isomorphism counting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1959–1969.

[41] Qing Lu and Lise Getoor. 2003. Link-based Classification. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, Tom Fawcett and Nina Mishra (Eds.). AAAI Press, 496–503.

[42] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. 2021. Learning to Pre-train Graph Neural Networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence,*

AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021. AAAI Press, 4276–4284.

[43] Yadan Luo, Zijian Wang, Zi Huang, and Mahsa Baktashmotlagh. 2020. Progressive Graph Learning for Open-Set Domain Adaptation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 6468–6478.

[44] Lukas Hedegaard Morsing, Omar Ali Sheikh-Omar, and Alexandros Iosifidis. 2020. Supervised Domain Adaptation using Graph Embedding. In *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*. IEEE, 7841–7847.

[45] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. 2012. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, Vol. 8.

[46] Nicolò Navarin, Dinh Van Tran, and Alessandro Sperduti. 2018. Pre-training Graph Neural Networks with Kernels. *CoRR* abs/1811.06930 (2018). arXiv:1811.06930

[47] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. ACM, 1150–1160.

[48] Michael T. Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G. Dietterich. 2005. To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning*, Vol. 898. 1–4.

[49] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. 2020. Learning to Simulate Complex Physics with Graph Networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 8459–8468.

[50] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[51] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*. ACM, 1067–1077.

[52] Siyi Tang, Jared Dunnmon, Khaled Kamal Saab, Xuan Zhang, Qianying Huang, Florian Dubost, Daniel Rubin, and Christopher Lee-Messer. 2022. Self-Supervised Graph Neural Networks for Improved Electroencephalographic Seizure Analysis. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[53] Paul Tseng. 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications* 109, 3 (2001), 475–494.

[54] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[55] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[56] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2018. Deep Graph Infomax. *CoRR* abs/1809.10341 (2018).

[57] Jianxin Wang, Xiaoqing Peng, Wei Peng, and Fang-Xiang Wu. 2014. Dynamic protein interaction network construction and applications. *Proteomics* 14, 4-5 (2014), 338–352.

[58] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. 2021. Curgraph: Curriculum learning for graph classification. In *Proceedings of the Web Conference 2021*. 1238–1248.

[59] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. 2020. Unsupervised Domain Adaptive Graph Convolutional Networks. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen (Eds.). ACM / IW3C2, 1457–1467.

[60] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. 2021. InfoGCL: Information-Aware Graph Contrastive Learning. (2021), 30414–30425.

[61] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[62] Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021. Dynamic knowledge graph alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4564–4572.

[63] Yuchen Yan, Si Zhang, and Hanghang Tong. 2021. Bright: A bridging algorithm for network alignment. In *Proceedings of the Web Conference 2021*. 3907–3917.

[64] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. 4805–4815.

[65] Si Zhang and Hanghang Tong. 2016. FINAL: Fast Attributed Network Alignment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM, 1345–1354.

[66] Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J. Gordon. 2019. On Learning Invariant Representations for Domain Adaptation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 7523–7532.

[67] Lecheng Zheng, Dongqi Fu, and Jingrui He. 2021. Tackling oversmoothing of gnns with contrastive learning. *arXiv preprint arXiv:2110.13798* (2021).

[68] Lecheng Zheng, Jinjun Xiong, Yada Zhu, and Jingrui He. 2022. Contrastive Learning with Complex Heterogeneity. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*. ACM, 2594–2604.

[69] Lecheng Zheng, Yada Zhu, Jingrui He, and Jinjun Xiong. 2021. Heterogeneous Contrastive Learning. *CoRR* abs/2105.09401 (2021). arXiv:2105.09401

[70] Dawei Zhou, Jingrui He, Hongxia Yang, and Wei Fan. 2018. SPARC: Self-Paced Network Representation for Few-Shot Rare Category Characterization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Yike Guo and Faisal Farooq (Eds.). ACM, 2807–2816.

[71] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2017. A Local Algorithm for Structure-Preserving Graph Cut. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 655–664.

[72] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2021. High-Order Structure Exploration on Massive Graphs: A Local Graph Clustering Perspective. *ACM Trans. Knowl. Discov. Data* 15, 2 (2021), 18:1–18:26.

[73] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. 2020. A data-driven graph generative model for temporal interaction networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 401–411.

[74] Dawei Zhou, Lecheng Zheng, Jiejun Xu, and Jingrui He. 2019. Misc-GAN: A multi-scale generative model for graphs. *Frontiers in big Data* 2 (2019), 3.

[75] Yao Zhou, Arun Reddy Nelakurthi, and Jingrui He. 2018. Unlearn What You Have Learned: Adaptive Crowd Teaching with Exponentially Decayed Memory Learners. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. ACM, 2817–2826.

[76] Yao Zhou, Arun Reddy Nelakurthi, Ross Maciejewski, Wei Fan, and Jingrui He. 2020. Crowd Teaching with Imperfect Labels. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. ACM / IW3C2, 110–121.

[77] Qi Zhu, Carl Yang, Yidan Xu, Haonan Wang, Chao Zhang, and Jiawei Han. 2021. Transfer Learning of Graph Neural Networks with Ego-graph Information Maximization. (2021), 1766–1779.

[78] Xiaojin Zhu. 2015. Machine Teaching: An Inverse Problem to Machine Learning and an Approach Toward Optimal Education. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, Blai Bonet and Sven Koenig (Eds.). AAAI Press, 4083–4087.