Domain Adaptive Graph Infomax via Conditional Adversarial Networks

Jiaren Xiao, Quanyu Dai, *Member, IEEE,* Xiaochen Xie, *Member, IEEE,* Qi Dou, *Member, IEEE,* Ka-Wai Kwok, *Senior Member, IEEE,* and James Lam, *Fellow, IEEE*

***Abstract*—The emerging graph neural networks (GNNs) have demonstrated impressive performance on the node classification problem in complex networks. However, existing GNNs are mainly devised to classify nodes in a (partially) labeled graph. To classify nodes in a newly-collected unlabeled graph, it is desirable to transfer label information from an existing labeled graph. To address this cross-graph node classification problem, we propose a graph infomax method that is domain adaptive. Node representations are computed through neighborhood aggregation. Mutual information is maximized between node representations and global summaries, encouraging node representations to encode the global structural information. Conditional adversarial networks are employed to reduce the domain discrepancy by aligning the multimodal distributions of node representations. Experimental results in real-world datasets validate the performance of our method in comparison with the state-of-the-art baselines.**

***Index Terms*—Conditional Adversarial Networks, Cross-graph Node Classification, Domain Adaptation, Graph Neural Networks, Mutual Information.**

I. Introduction

NODE classification is a vital task in various real applications, such as the prediction of user characteristics in social networks [1], the classification of protein roles in protein-protein interaction (PPI) networks [2], and the assignment of research topics for publications in citation networks [3]. Since real-world networks are usually sparse, nonlinear, and high-dimensional, it is challenging to learn meaningful information from these networks to facilitate node classification [4]. A general way is to encode graph[1] information, like graph structure and node attributes, into low-dimensional node representations (i.e., embedding vectors) [4], [5]. On top of node representations, node classification can be easily performed by employing classical machine learning techniques, e.g., a logistic regression classifier.

This work was supported in part by the Innovation and Technology Commission (ITC) (MRP/029/20X) and the HKSAR Government under the InnoHK initiative, Hong Kong, via Centre for Garment Production Limited, and in part by the Research Grants Council (RGC) of Hong Kong under Grant 17205721. (*Corresponding authors: Ka-Wai Kwok and James Lam.*)

Jiaren Xiao, Ka-Wai Kwok, and James Lam are with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong, China (e-mail: xiaojr@connect.hku.hk; kwokkw@hku.hk; james.lam@hku.hk).

Quanyu Dai is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China (e-mail: quanyu.dai@connect.polyu.hk).

Xiaochen Xie is with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong, China, and also with Centre for Garment Production Limited, Hong Kong, China (e-mail: xcxie@connect.hku.hk).

Qi Dou is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China (e-mail: qidou@cuhk.edu.hk).

[1]Network and graph are interchangeably used to denote the graph-structured data.

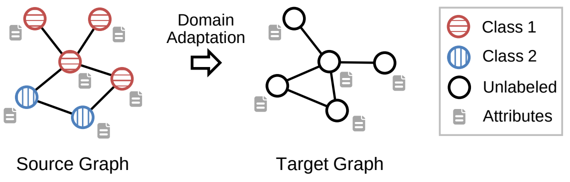


Fig. 1. An example of cross-graph node classification. The research goal is to facilitate node classification in an unlabeled target graph by transferring knowledge from a labeled source graph.

Existing studies mainly focus on node classification in a (partially) labeled graph [2], [3]. In real applications, it is frequently encountered that the nodes are unlabeled in a newly collected graph. A node classification model, which is learned in an existing labeled graph, can sometimes be directly applied to an unlabeled new graph. However, different graphs usually have diverse data distributions of node connections, attributes, and labels, which would degrade the performance of such direct application.

Therefore, it is desirable if a learning model can effectively transfer label information from a labeled graph to assist node classification in an unlabeled graph. Otherwise, when a new graph is collected, to obtain satisfactory node classification performance, we have to label the nodes before rebuilding a learning model. For example, in a newly-formed unlabeled social network, to enable user prediction, it would be beneficial to transfer knowledge from a mature social network which is well annotated. In addition, to classify proteins in a newly-collected PPI network, it would be favorable if the label information in an existing biological network can be utilized. Moreover, the abundant label information in well-established citation databases would be helpful to enable the classification of papers in a newly-constructed citation network.

In this work, we consider the cross-graph node classification problem [6]–[9] illustrated in Figure 1. The research goal is to facilitate node classification in an unlabeled target graph by transferring label information from a labeled source graph. There are no edge connections or common nodes existing between the source and target graphs. With only a part of the node attributes in common, the discrepancy is further enlarged between the source and target graphs.

Since the source and target graphs can be treated as two independent domains, cross-graph node classification has a close relation with domain adaptation research. As a subtopic of transfer learning [10], domain adaptation studies are mostly conducted in the fields of computer vision [11]–[13] and natural language processing [14], [15]. The input data (i.e., images and text) is usually assumed to be independent and identically distributed (i.i.d.). The classical domain adaptation approaches are not directly applicable to the graph-structured data, since the nodes in a graph are highly correlated by edges, thus violating the i.i.d. assumption [16].

There are a few recent studies addressing the challenging problem of cross-graph node classification, such as CDNE [6], ACDNE [7], AdaGCN [8], and UDA-GCN [9]. Although their performance is more preferable than those designed for single-graph learning, *three open questions* remain to be further explored.

- From the global view of a graph, if distant nodes have similar structural roles, they are likely to perform the same function [17], which can be an important predictor for the node labels. Existing methods [6]–[9] manage to make the nodes have similar representations, if they are close in a graph, such as the adjacent nodes or the nodes within K steps. It is achieved by the Laplacian smoothing of graph convolution [18] or the additional constraint based on PPMI matrix [19]. Although these methods reach a certain level of local or global consistency, they still lack consideration about the global structural role of a node.
- Existing studies align the source and target representations by minimizing domain classification error [7], [9] or distribution discrepancy metrics such as the Wasserstein distance [8]. Since the domain alignment is category agnostic in these methods, the aligned node representations are possibly not classification friendly. Moreover, due to the nature of classification problem, the distribution of node representations would be multimodal. These domain adaptation techniques may fail to capture the multimodal structure [20].
- AdaGCN [8] and UDA-GCN [9] employ the graph neural networks (GNNs). They apply spectral graph theory [21] to define filters for graph convolutions that process the whole graph in one go. It may hinder their applications in real-world large-scale graphs which can be directed, signed, or heterogeneous [16].

To address the above three issues, we propose a novel method, named as domain Adaptive Graph Infomax via conditional adversarial networks (i.e., AdaGIn). Given a labeled source graph and an unlabeled target graph, the objective of our method is to classify nodes in the target graph, by jointly utilizing the information in the source and target graphs. Our method consists of *three modules*: a representation learner, a node classifier, and a domain discriminator.

Representation learner employs the spatial GNN layers to compute node representations for the source and target graphs. The spatial GNN layers repeatedly aggregate the local neighborhood information [22]. The generated node representation summarizes a patch of graph centered around this node, thus

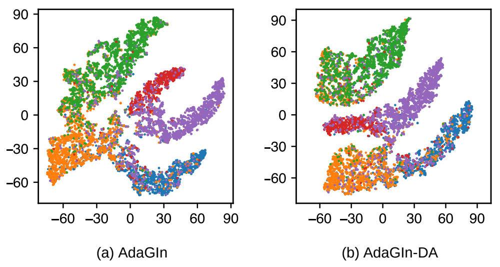


Fig. 2. Visualization of node representations in the target graph (Source graph: Citationv1, Target graph: ACMv9). Each point represents one node. Five colors distinguish various node classes. The sign "-" means domain adaptation component (DA) has been removed from the AdaGIn model.

containing the local structural information. Inspired by DGI [23], the node representation is then driven to preserve the mutual information with the graph-level global representation. Maximization of the local-global mutual information could encourage node representations to capture the global structural properties [23]. Taking node representations as inputs, node classifier produces label predictions. Representation learner and node classifier are trained to minimize the cross-entropy loss of labeled nodes in the source graph, so that the learned node representations can be label-discriminative.

To reduce the domain discrepancy between the source and target graphs, our method applies the conditional domain adversarial networks [20], which model the domain adaptation process as a two-player game. Domain discriminator is trained to distinguish whether an input sample comes from the source or target graph. In contrast, representation learner is optimized to "fool" domain discriminator by generating domain-invariant node representations. To enhance the alignment of multimodal distributions, domain discriminator is conditioned on the discriminative information in the classifier predictions. Figure 2 illustrates that, with the help of domain adaptation, the generated node representations are more meaningful and separable in the 2D space, benefiting the subsequent node classification task.

The proposed method is evaluated using five real-world networks, including three citation networks and two social networks. The *main contributions* are summarized as follows.

- A novel spatial GNN model is proposed to tackle the challenging problem of cross-graph node classification. The overall performance is superior to the state-of-the-art baselines in the benchmark transfer tasks.
- Mutual information maximization is enabled in cross-graph learning, which encourages node representations to encode the global structural information.
- Conditional adversarial networks are introduced to reduce the domain discrepancy by matching the multimodal distributions of node representations.

The rest of this paper is organized as follows. Section II reviews the related literature. Section III presents the proposed method. Section IV reports the experiments and results. Finally, Section V concludes this paper.

II. Related Work

In this section, we introduce the related work in three aspects, including the research on domain adaptation, graph representation learning, and cross-graph learning.

A. Domain Adaptation

As a subtopic of transfer learning, domain adaptation [10] aims at transferring knowledge from a source domain with abundant labels to a target domain that is short of labels. The feature-based domain adaptation approaches, which attract lots of research interests due to their effectiveness, can be categorized into three kinds, including the discrepancy-based [12], [24], the reconstruction-based [25], and the adversarial-based [20], [26], [27].

The adversarial-based methods are considered in this work. To learn domain-invariant features, DANN [26] builds an adversarial training platform, in which representation learner and domain discriminator play a minimax game. WDGRL [27] introduces the Wasserstein distance [28] to measure the domain divergence, in order to improve the gradient property and the generalization bound. Inspired by the recent progress in conditional generative adversarial networks (CGANs) [29], CDAN [20] conditions the domain discriminator on the discriminative information contained in the classifier predictions, which helps match the multimodal distributions in the classification problem.

There are also some recent advances worth noticing. For example, ATM [30] devises a novel loss, named as MDD, to quantify the distribution gap. By optimizing this additional loss, ATM alleviates the equilibrium challenge issue [31], thus improving the performance of adversarial domain adaptation. In addition, Li et al. [32] innovatively employed the adversarial attacks to enable domain adaptation in the absence of source data or target data.

The existing domain adaptation methods usually assume the input samples to be independent and identically distributed (i.i.d.), such as images and text. They are not directly applicable to solving the learning problems on graph-structured data, where highly-correlated nodes violate the i.i.d. assumption [16].

A few recent studies construct graph adjacency matrix based on the pairwise distance of image embeddings/features. Graph matching is then employed to align the domains of image samples. For example, by minimizing the spectral distance of graph Laplacians, LGA [33] aims at forcing the manifold of target domain has a similar connectivity structure as that of the source domain. In addition, Das and Lee [34] applied the first-, second-, and third-order hyper-graph matching to match the images in different domains. These approaches are devised for the alignment of image samples rather than the data that can be naturally represented as a graph, such as the citation and social networks.

B. Graph Representation Learning

Graph representation learning [35] aims at learning low-dimensional node representations to facilitate downstream tasks, such as node classification and link prediction. Node representations are expected to preserve graph structure only [36]–[39], or jointly capture structural properties and side information like node attributes [40], [41]. One recent advancement is known as graph neural networks (GNNs) [16] which design neural networks that directly operate on graphs [2], [3], [42], [43].

GNNs are classified into two categories: spectral approaches and spatial ones. The spectral GNNs devise spectral filters to perform graph convolutions. The classical GCN model [3] simplifies the spectral convolutions and processes the whole graph at the same time. Modern GNNs usually update node representations by iteratively aggregating neighborhood information [22]. This stream of approaches relies on the spatial relations of nodes to propagate information [16]. The computation of node representations can be conducted in a batch of nodes rather than the whole graph. GraphSAGE [2] pioneers in designing various aggregation functions to aggregate information from the sampled neighborhood. In graph attention networks (GAT) [42], node representation is the weighted sum of the representations of all neighboring nodes. Compared with spectral approaches, spatial ones have advantages in the scalability to large graphs and the generality to various graph types [16].

To capture the statistical similarities among data, contrastive learning [44] has been introduced to encode graph into informative representations. Inspired by the Deep InfoMax (DIM) [45], some recent studies (e.g., DGI [23] and InfoGraph [46]) maximize the mutual information between the local representations of substructures (e.g., a patch of graph centered around a node) and the global representations of a graph. By doing so, the local and global representations can be mindful of each other, leading to improved performance on node classification or graph classification.

Existing studies mainly focus on learning in a single graph, ignoring knowledge transfer across graphs. Although a model learned in one graph can sometimes be adapted to perform learning tasks in a new graph, to improve model performance, more efforts are needed to overcome the discrepancy between graphs. To this end, our work jointly explores contrastive learning and domain adaptation to perform transfer learning across graphs.

C. Cross-graph Learning

Some recent research on cross-graph learning assumes the existence of inter-graph connections [47], [48] or common nodes across graphs [49], [50]. Besides, with the target graph partially labeled, the basic assumption of DASGA [51] is that the source and target graphs have similar frequency contents in the label function. DASGA then performs domain adaptation on graphs by learning aligned graph Fourier bases.

Without the above assumptions, some other studies [6]–[9] focus on transfer learning between two independent graphs. The abundant label information in a source graph is expected to be transferred to facilitate node classification in a target graph lacking labels.

NetTr [52] projects the label propagation matrices of the source and target graphs into a common latent space utilizing the nonnegative matrix tri-factorization (NMTF) [53]. Although node attributes are also combined to train the node classifier, the transferrable representations are learned solely based on graph topology, in order to capture structural patterns shared by the source and target graphs. CDNE [6] learns node representations in an autoencoder architecture. The maximum mean discrepancy (MMD) [54] is minimized between the source and target representations to mitigate domain divergence. ACDNE [7] preserves attribute affinity and topology proximity separately using two feature extractors. Gradient reversal layer [26] is employed in ACDNE to learn domain-invariant node representations.

Recently, a few GNN-based methods are proposed to perform cross-graph node classification. AdaGCN [8] employs GCN [3] to learn node representations. This method minimizes the Wasserstein distance [28] between the source and target representations to enable domain adaptation. UDA-GCN [9] develops a dual graph convolutional network to jointly preserve the local and global consistency of a graph. Gradient reversal layer [26] is also used in UDA-GCN to achieve domain adaptation. Unlike these spectral GNNs (i.e., AdaGCN and UDA-GCN), this work devises a spatial GNN model to address this challenging problem. Two recent techniques, mutual information maximization [23] and conditional adversarial networks [20], are employed to capture global graph information and multimodal embedding distribution, respectively.

It is worth noting that a few recent GNN models transfer knowledge following a pre-training and fine-tuning paradigm, without utilizing the adversarial domain adaptation. As one typical example, GCC [55] pre-trains a model for discovering common structural patterns in the absence of node attributes and node labels. Specifically, the pre-training is designed as subgraph instance discrimination in and across multiple source graphs. GCC leverages contrastive learning to distinguish similar subgraph instances from dissimilar ones. The pre-trained model is then fine-tuned on a partially-labeled target graph for node classification task. Differing from GCC, in this work, we consider a transfer learning problem involving two attributed graphs: one fully labeled source graph and one unlabeled target graph. Contrastive learning is employed to maximize the mutual information between node representation and graph-level representation within a single graph (i.e., source graph or target graph). To reduce domain divergence, we make use of conditional adversarial networks.

III. Proposed Method

In this section, we first introduce the research problem and the main notations. Then we present the model architecture and elaborate each module. Next, we provide a theoretical analysis about the adversarial domain adaptation. Finally, we describe the training algorithm followed by a time complexity analysis.

A. Problem Definition and Notations

An information network can be represented as an attributed graph $\mathcal{G}(\boldsymbol{V}, \boldsymbol{A}, \boldsymbol{X}, \boldsymbol{Y})$, in which $\boldsymbol{V} \in \mathbb{R}^N$, $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, $\boldsymbol{X} \in \mathbb{R}^{N\times L}$, and $\boldsymbol{Y} \in \mathbb{R}^{N\times C}$ are node set, adjacency matrix, attribute matrix, and label matrix, respectively. N is the number of nodes in $\boldsymbol{V}$, that is, $N = |\boldsymbol{V}|$. L is the dimension of node attributes. C is the number of node labels. In $\boldsymbol{A}$, $\boldsymbol{X}$, and $\boldsymbol{Y}$, the i-th row contains binary values indicating the edge connections, attributes, and labels of the i-th node $v \in \boldsymbol{V}$, respectively. Specifically, $A_{ij} = 1$ means there is an edge connecting the i-th and j-th nodes; $X_{id} = 1$ means the i-th node has the d-th attribute; and $Y_{ic} = 1$ means the i-th node is associated with the c-th label. The undirected graph is considered in this paper. The degree of the i-th node v is the number of its connected edges, i.e., $\sum_j A_{ij}$. The average degree is calculated by $\sum_i \sum_j A_{ij}/N$, indicating the density of a graph. The main notations of this paper are summarized in Table I.

TABLE I
Main notations.

Notation	Description
$\mathcal{G}$	An attributed graph
$\mathcal{G}^s$, $\mathcal{G}^t$	Source graph and target graph
$\boldsymbol{V}$	Node set of $\mathcal{G}$
$\boldsymbol{A}$	Adjacency matrix of $\mathcal{G}$
$\boldsymbol{x}_v$, $\boldsymbol{X}$	Attribute vector of node $v \in \boldsymbol{V}$ and attribute matrix of $\mathcal{G}$
$\boldsymbol{e}_v$, $\boldsymbol{E}$	Embedding vector of node $v \in \boldsymbol{V}$ and representation matrix of $\mathcal{G}$
$\boldsymbol{y}_v$, $\boldsymbol{Y}$	Label vector of node $v \in \boldsymbol{V}$ and label matrix of $\mathcal{G}$
$\hat{\boldsymbol{y}}_v$, $\hat{\boldsymbol{Y}}$	Label prediction vector of node $v \in \boldsymbol{V}$ and label prediction matrix of $\mathcal{G}$
$\boldsymbol{\mathcal{X}}$	Union attribute set
N	Number of nodes in $\mathcal{G}$
L, l	Attribute dimension and embedding dimension
C	Number of classes in $\boldsymbol{Y}$
U	Number of attributes in $\boldsymbol{\mathcal{X}}$
f_g, f_c, f_d	Representation learner, node classifier, and domain discriminator
$\boldsymbol{\theta}_g, \boldsymbol{\theta}_c, \boldsymbol{\theta}_d$	Sets of parameters in f_g, f_c, and f_d
AGG	Aggregation function
s	Neighborhood sample size
σ	Nonlinear activation function
$\boldsymbol{h}_v$	Representation vector of node $v \in \boldsymbol{V}$
η_0	Initial learning rate
n_e	Maximum training epoch
n_i	Maximum iteration per epoch
$\boldsymbol{B}$	A batch of nodes
λ_1	Mutual information coefficient
λ_2	Domain adaptation coefficient
R_a	Common attribute rate
R_n	Proportion of selected nodes

Graph embedding aims at mapping a node, $v \in \boldsymbol{V}$, to a low-dimensional node representation $\boldsymbol{e}_v$ (i.e., embedding vector). $\boldsymbol{e}_v^\top$ is one row within representation matrix $\boldsymbol{E} \in \mathbb{R}^{N\times l}$, where l is the embedding dimension. Since the label space of nodes is $\mathcal{Y} = \{1, \ldots, C\}$, data distribution $\boldsymbol{e}_v \sim P(\boldsymbol{e}_v)$ would be multimodal, with each mode corresponding to one class. The multimodal structure can be indicated by the t-SNE visualization shown in Figure 2. There are five clusters in total (i.e., $C = 5$), with each cluster indicating one mode. On top of node embeddings, a classifier, $f : \boldsymbol{E} \longmapsto \boldsymbol{Y}$, can be learned to conduct node classification.

In this work, we investigate the cross-graph node classifi-

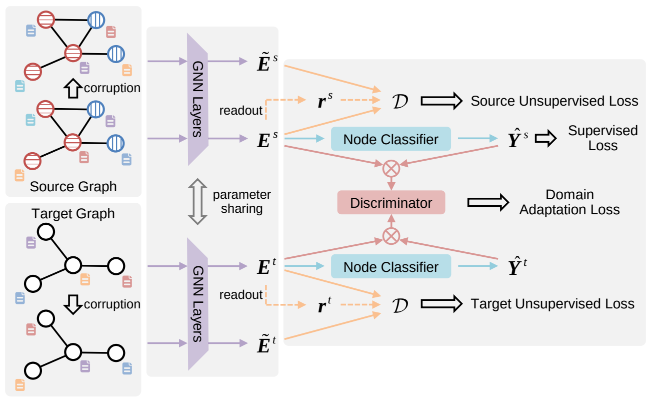


Fig. 3. Architecture of the proposed method. Various node attributes in a graph are distinguished by their colors. Parameter sharing is enabled when employing the GNN layers to process the source and target graphs. Please refer to Section III for more details.

cation problem. Source graph, $\mathcal{G}^s\,(\boldsymbol{V}^s, \boldsymbol{A}^s, \boldsymbol{X}^s, \boldsymbol{Y}^s)$, is fully labeled, that is, the labels of each node in $\mathcal{G}^s$ are known. Target graph, $\mathcal{G}^t\,(\boldsymbol{V}^t, \boldsymbol{A}^t, \boldsymbol{X}^t, \boldsymbol{Y}^t)$, is completely unlabeled, that is, label matrix $\boldsymbol{Y}^t$ is unknown, remaining to be predicted. We use two superscripts, s and t, to denote the source and target graphs, respectively.

In the case that the attribute sets of source and target graphs (i.e., $\boldsymbol{\mathcal{X}}^s$ and $\boldsymbol{\mathcal{X}}^t$) are not exactly the same, we construct a union attribute set $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}^s \cup \boldsymbol{\mathcal{X}}^t$. With $U = |\boldsymbol{\mathcal{X}}|$ representing the total number of attributes, the attribute matrices of source and target graphs can be reformulated as $\boldsymbol{X}^s \in \mathbb{R}^{N^s \times U}$ and $\boldsymbol{X}^t \in \mathbb{R}^{N^t \times U}$, respectively. The union attribute set enables parameter sharing when processing the source and target graphs, since the same learning model can be applied to generate node representations for both graphs. The parameter sharing attempts to learn domain-invariant node representations, and to facilitate knowledge transfer across graphs [56]. Through sharing parameters, the overall model architecture can also be more compact, with the number of learnable parameters being reduced. Based on the union attribute set, we can further define a common attribute rate $R_a = \left|\boldsymbol{\mathcal{X}}^s \cap \boldsymbol{\mathcal{X}}^t\right| / \left|\boldsymbol{\mathcal{X}}^s \cup \boldsymbol{\mathcal{X}}^t\right|$, showing the percentage of common attributes shared by the source and target graphs.

There are no shared common nodes or edge connections existing between the source and target graphs. Therefore, the source and target graphs can be treated as two independent domains. In addition to the varying graph scale, these two domains are also diverse in the distributions of node connections, attributes, and labels. Note that, as the settings in many prior arts [6]–[9], the source and target graphs are required to have the same set of labels, that is, the categories of nodes in these two graphs are the same.

The goal of this work is to mitigate the graph divergence, so that label information in the source graph can be exploited to learn a node classifier that is readily applicable to classify nodes in the unlabeled target graph. To achieve this goal, node representations generated in the learning process need to be domain-invariant and label-discriminative.

B. Overview of Model Architecture

To achieve cross-graph node classification, there are two major challenges. First, the available information (i.e., graph structure, node attributes, and node labels) shall be well exploited to learn node representations that are informative enough for the subsequent node classification task. Second, the domain gap between source and target graphs shall be mitigated, so that the node representations can be shared across domains. Consequently, the node classifier trained on source representations can be readily applicable to predict target nodes using the target representations.

To address the above challenges, we propose a method named as domain Adaptive Graph Infomax via conditional adversarial networks (i.e., AdaGIn). Figure 3 shows the proposed model architecture. The three main modules of this model are as follows.

- **Representation Learner.** The node representations are learned by the GNN layers, which aggregate information from the local neighborhood, thus capturing the graph structure and node attributes simultaneously. To make a node representation mindful of the global structure and other nodes with similar structural roles, inspired by DGI [23], the local-global mutual information is calculated and maximized in the learning process.
- **Node Classifier.** On top of the learned node representations, the node classifier, which is a logistic regression, makes predictions on the node labels.
- **Domain Discriminator.** The domain discriminator aims at telling apart the input samples from the source and target graphs. It competes with the representation learner during adversarial training using the gradient reversal layer, so that the representation learner can generate domain-invariant node representations. The discrepancy between source and target graphs would be reduced. As a result, node classifier trained on the source representations can be more desirable for classifying target nodes with the target representations.

C. Node Representation Learning

The graph neural networks (GNNs) are applied to encode nodes into vector representations (i.e., embedding vectors). In the modern GNNs, a common strategy is to iteratively update the representation of a node by aggregating information from its neighboring nodes [22]. Considering node v in a sampled minibatch of nodes (i.e., $\boldsymbol{B}$), the k-th step/layer of the neighborhood aggregation is formulated as follows.

$$\boldsymbol{h}_S^k = \mathrm{AGG}_k(\boldsymbol{h}_u^{k-1} \mid u \in \boldsymbol{S}_v) = \frac{1}{|\boldsymbol{S}_v|}\sum\nolimits_{u \in \boldsymbol{S}_v} \boldsymbol{h}_u^{k-1} \quad (1)$$

$$\boldsymbol{h}_v^k = \sigma_1\left([\boldsymbol{W}_v^k \boldsymbol{h}_v^{k-1}; \boldsymbol{W}_S^k \boldsymbol{h}_S^k]\right) \quad (2)$$

where AGG_k is the aggregation function at step k, $\boldsymbol{S}_v$ is a set of sampled nodes adjacent to node v, $\boldsymbol{h}_S^k$ is the representation vector of neighborhood at step k, $\boldsymbol{h}_v^k$ is the representation vector of node v at step k, $\boldsymbol{W}_v^k$ and $\boldsymbol{W}_S^k$ are the weight matrices for linear transformations, and σ_1 is the ReLU nonlinear activation (i.e., $\sigma_1(x) = \max(0, x)$).

The node representation is initialized with the node attribute vector, i.e., $\boldsymbol{h}_v^0 = \boldsymbol{x}_v$. Then, the node gradually aggregates information from far neighborhood. After k steps, the node's representation captures the structural information within its k-hop neighbors. Node representation in the final step (i.e., $\boldsymbol{h}_v^K$) is the embedding vector (i.e., $\boldsymbol{e}_v$), which is used for the subsequent node classification.

$$\boldsymbol{e}_v = \boldsymbol{h}_v^K = f_g\left(\boldsymbol{x}_v, \boldsymbol{x}_S\right), v \in \boldsymbol{B}. \tag{3}$$

where f_g is the representation learner (i.e., the spatial GNN layers) and $\boldsymbol{x}_S$ is the attribute matrix of the sampled neighboring nodes.

To enhance the structural constraint applied during the learning process of node representation, the local-global mutual information is calculated and maximized within a graph. Considering a minibatch of nodes (i.e., $\boldsymbol{B}$), the representation of each node is first computed using the GNN layers, i.e., $\boldsymbol{E} = \left[\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{|\boldsymbol{B}|}\right]$. Since the neighborhood information is repeatedly aggregated in the learning process, the produced representation of a node summarizes a patch of the graph centered around this node. Therefore, the node representation can be treated as a local representation of this patch in the graph. A summary vector, $\boldsymbol{r}$, is obtained by a readout function (i.e., $\mathcal{R}$) which simply averages the node representations within this minibatch.

$$\boldsymbol{r} = \mathcal{R}\left(\boldsymbol{E}\right) = \sigma_2\left(\frac{1}{|\boldsymbol{B}|}\sum_{i=1}^{|\boldsymbol{B}|}\boldsymbol{e}_i\right) \tag{4}$$

where σ_2 is the logistic sigmoid nonlinearity (i.e., $\sigma_2\left(x\right) = 1/\left(1+\exp\left(-x\right)\right)$). This summary vector is a kind of global information regarding the nodes within the minibatch and their sampled neighbors. A local-global pair is denoted as $\left(\boldsymbol{e}_i, \boldsymbol{r}\right)$.

The local-global pairs are then treated as positive samples. The local-global mutual information is maximized by classifying these positive samples and the negative counterparts. The negative samples are obtained by pairing the summary vector with the node representations of a corrupted graph. To encourage the positive samples to encode the structural similarities of different nodes, as in DGI [23], the graph is corrupted by row-wisely shuffling attribute matrix $\boldsymbol{X}$, while adjacency matrix $\boldsymbol{A}$ is preserved. The corrupted attributes and original adjacency matrix are fed into the GNN layers to obtain the node representations for negative samples, i.e., $\tilde{\boldsymbol{E}} = \left[\tilde{\boldsymbol{e}}_1, \ldots, \tilde{\boldsymbol{e}}_{|\boldsymbol{B}|}\right]$. Similarly, a negative sample can be denoted as $\left(\tilde{\boldsymbol{e}}_i, \boldsymbol{r}\right)$.

To classify the local-global pairs (i.e., positive samples) and the corresponding corrupted counterparts (i.e., negative samples), a bilinear scoring function, $\mathcal{D}$, is applied to assign a score indicating the possibility of a sample to be positive:

$$\boldsymbol{s}_i = \mathcal{D}\left(\boldsymbol{e}_i, \boldsymbol{r}\right) = \sigma_2\left(\boldsymbol{e}_i^{\top}\boldsymbol{W}_b\boldsymbol{r}\right), i \in \{1, \ldots, |\boldsymbol{B}|\}, \tag{5}$$

$$\tilde{\boldsymbol{s}}_i = \mathcal{D}\left(\tilde{\boldsymbol{e}}_i, \boldsymbol{r}\right) = \sigma_2\left(\tilde{\boldsymbol{e}}_i^{\top}\boldsymbol{W}_b\boldsymbol{r}\right), i \in \{1, \ldots, |\boldsymbol{B}|\}, \tag{6}$$

where $\boldsymbol{W}_b$ is a learnable scoring matrix. Finally, the objective for this binary classification is as follows.

$$\mathcal{L}_{\mathrm{MI}} = -\frac{1}{2|\boldsymbol{B}|}\sum_{i=1}^{|\boldsymbol{B}|}\left\{\log\mathcal{D}\left(\boldsymbol{e}_i, \boldsymbol{r}\right) + \log\left[1 - \mathcal{D}\left(\tilde{\boldsymbol{e}}_i, \boldsymbol{r}\right)\right]\right\} \tag{7}$$

The mutual information between $\boldsymbol{e}_i$ and $\boldsymbol{r}$ will be maximized by minimizing this loss function.

Note that node representations of both the source and target graphs ($\boldsymbol{E}^s$ and $\boldsymbol{E}^t$) are generated in the same way, so the above descriptions are provided without mentioning the specific graph. The loss regarding mutual information (i.e., the unsupervised loss) is calculated for the source and target graphs independently, i.e., $\mathcal{L}_{\mathrm{MI}}^s$ and $\mathcal{L}_{\mathrm{MI}}^t$. The total unsupervised loss is the summation of source and target losses.

$$\mathcal{L}_{\mathrm{UN}} = \mathcal{L}_{\mathrm{MI}}^s + \mathcal{L}_{\mathrm{MI}}^t \tag{8}$$

Furthermore, the same GNN layers is utilized to generate node representations for the source and target graphs, that is, parameter sharing is enabled when processing these two graphs.

D. Node Label Prediction

To make the node embeddings label-discriminative, the supervised signals (i.e., the node labels) in the source graph are incorporated in the learning process. Specifically, we construct a node classifier which is a logistic regression. Node embeddings learned by the representation learner are fed into this classifier to obtain label predictions.

$$\hat{\boldsymbol{y}}_v = f_c\left(\boldsymbol{e}_v\right) = \sigma_3\left(\boldsymbol{W}_c\boldsymbol{e}_v + \boldsymbol{b}_c\right), v \in \boldsymbol{B}. \tag{9}$$

where f_c is the node classifier; $\boldsymbol{W}_c$ and $\boldsymbol{b}_c$ are learnable weight matrix and bias vector, respectively. $\hat{\boldsymbol{y}}_v^{\top}$ is one row in prediction score matrix, $\hat{\boldsymbol{Y}}$. Nonlinear activation, σ_3, is usually a sigmoid activation for multilabel classification or a softmax activation for multiclass classification.

Since only the label information of source graph is available during training, the cross-entropy loss (i.e., the supervised loss) is calculated using the source labels $\boldsymbol{Y}^s$ and the corresponding predictions $\hat{\boldsymbol{Y}}^s$. For multilabel classification, the cross-entropy loss is defined as follows.

$$\mathcal{L}_{\mathrm{CE}} = -\mathbb{E}_{v\in\boldsymbol{B}^s}\sum_{k=1}^{C}\left[Y_{vk}^s\log\left(\hat{Y}_{vk}^s\right) + \left(1 - Y_{vk}^s\right)\log\left(1 - \hat{Y}_{vk}^s\right)\right] \tag{10}$$

where $\boldsymbol{B}^s$ is a batch of nodes sampled from source graph, binary element Y_{vk}^s in label matrix $\boldsymbol{Y}^s$ indicates whether a node $v \in \boldsymbol{B}^s$ belongs to class k, and $\hat{Y}_{vk}^s$ is the corresponding element in prediction score matrix $\hat{\boldsymbol{Y}}^s$. For multiclass classification, the cross-entropy loss is

$$\mathcal{L}_{\mathrm{CE}} = -\mathbb{E}_{v\in\boldsymbol{B}^s}\sum_{k=1}^{C}\left[Y_{vk}^s\log\left(\hat{Y}_{vk}^s\right)\right]. \tag{11}$$

E. Adversarial Domain Adaptation

Conditional adversarial networks [20] are leveraged to mitigate the domain gap between the source and target graphs. Domain discriminator, f_d, is constructed as a multilayer perceptron (i.e., MLP) followed by a sigmoid activation. It is conditioned on the discriminative information contained in the classifier prediction. We construct a joint variable, $\boldsymbol{h}_v$, which is a multilinear map defined as the outer product of classifier prediction (i.e., $\hat{\boldsymbol{y}}_v$) and embedding vector (i.e., $\boldsymbol{e}_v$), that is,

$\boldsymbol{h}_v = \hat{\boldsymbol{y}}_v \otimes \boldsymbol{e}_v$. Domain prediction is obtained by feeding the joint variable into domain discriminator.

$$\hat{d}_v = f_d\left(\boldsymbol{h}_v\right) \tag{12}$$

where $\hat{d}_v$ is a score indicating the probability that a joint variable comes from the source graph.

Multilinear conditioning is employed to capture the cross-covariance between node representation and classifier prediction. By conditioning, domain divergence in both node representation and classifier prediction can be modeled simultaneously. Domain discriminator is then optimized to tell apart whether a joint variable is from the source or target graph. Domain adaptation loss, $\mathcal{L}_{\mathrm{DA}}$, is calculated as follows.

$$\mathcal{L}_{\mathrm{DA}} = -\frac{1}{2}\left\{\mathbb{E}_{v\in\boldsymbol{B}^s}\log\left[f_d\left(\boldsymbol{h}_v^s\right)\right] + \mathbb{E}_{v\in\boldsymbol{B}^t}\log\left[1 - f_d\left(\boldsymbol{h}_v^t\right)\right]\right\} \tag{13}$$

in which $\boldsymbol{B}^s$ and $\boldsymbol{B}^t$ are batches from the source and target graphs; $\boldsymbol{h}_v^s$ and $\boldsymbol{h}_v^t$ are the joint variables of nodes in these two batches.

To reduce the domain discrepancy, representation learner and domain discriminator are trained in an adversarial manner. Specifically, domain discriminator is trained to minimize domain adaptation loss, $\mathcal{L}_{\mathrm{DA}}$, thus improving its capability of distinguishing the source and target samples. In contrast, representation learner is trained to maximize the same loss, so that the produced node representations can be domain-invariant. We apply a gradient reversal layer (GRL) [26] to simultaneous update representation learner and domain discriminator. When minimizing the domain adaptation loss, GRL flips the gradient with respect to the model parameters in representation learner.

F. Overall Loss and Model Training

The overall loss of AdaGIn is comprised of supervised loss, $\mathcal{L}_{\mathrm{CE}}$, unsupervised loss, $\mathcal{L}_{\mathrm{UN}}$, and domain adaptation loss, $\mathcal{L}_{\mathrm{DA}}$.

$$\min_{\boldsymbol{\theta}_g,\boldsymbol{\theta}_c}\left[\mathcal{L}_{\mathrm{CE}} + \lambda_1\mathcal{L}_{\mathrm{UN}} + \lambda_2\max_{\boldsymbol{\theta}_d}\left(-\mathcal{L}_{\mathrm{DA}}\right)\right] \tag{14}$$

where balance coefficients, λ_1 and λ_2, are named as mutual information coefficient and domain adaptation coefficient, respectively; $\boldsymbol{\theta}_g$, $\boldsymbol{\theta}_c$, and $\boldsymbol{\theta}_d$ are the sets of learnable parameters in representation learner, node classifier, and domain discriminator, respectively.

Algorithm 1 outlines the training and testing procedures of AdaGIn. During the minibatch training, we first independently sample a batch of nodes from the source graph and a batch of nodes from the target graph. Then the same GNN layers is employed to compute representations for nodes in these two batches. Next we calculate the unsupervised loss, the supervised loss, and the domain adaptation loss one by one. Finally, the model parameters of AdaGIn are updated via gradient descent based on the overall loss in Eq. 14. When the model converges after a number of epochs, the generated node representations would be label-discriminative and domain-invariant. In the testing stage, the classifier trained on source nodes can be applied to classify target nodes with the target node representations.

Algorithm 1: Algorithm of AdaGIn

```
Input : Fully labeled source graph $\mathcal{G}^s(\boldsymbol{V}^s, \boldsymbol{A}^s, \boldsymbol{X}^s, \boldsymbol{Y}^s)$;
        unlabeled target graph $\mathcal{G}^t(\boldsymbol{V}^t, \boldsymbol{A}^t, \boldsymbol{X}^t)$; batch size
        $N_b$; maximum training epoch $n_e$; maximum
        iteration per epoch $n_i$; coefficients $\lambda_1$ and $\lambda_2$.

1  Initialize model parameters $\boldsymbol{\theta}_g$ for representation learner, $\boldsymbol{\theta}_c$
     for node classifier, and $\boldsymbol{\theta}_d$ for domain discriminator.
2  for epoch $< n_e$ do
3      for iteration $< n_i$ do
4          Sample a batch of source nodes (i.e., $\boldsymbol{B}^s$) and a
             batch of target nodes (i.e., $\boldsymbol{B}^t$);
5          Compute source representations $\boldsymbol{E}^s$ and target
             representations $\boldsymbol{E}^t$ using Eq. 3;
6          Calculate unsupervised loss $\mathcal{L}_{\mathrm{UN}}$ using Eq. 8;
7          Calculate supervised loss $\mathcal{L}_{\mathrm{CE}}$ using Eq. 10 or
             Eq. 11;
8          Calculate domain adaptation loss $\mathcal{L}_{\mathrm{DA}}$ using Eq. 13;
9          Backpropagate the overall loss in Eq. 14 and update
             $\boldsymbol{\theta}_g$, $\boldsymbol{\theta}_c$ and $\boldsymbol{\theta}_d$.
10     end
11 end

Testing: With the optimized model parameters $\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_c$,
         target representation $\boldsymbol{E}^t$ is computed using Eq. 3.
         Target label prediction $\hat{\boldsymbol{Y}}^t$ is subsequently
         calculated using Eq. 9.
```

The time complexity of AdaGIn depends on its three modules, including representation learner (Eq. 3), node classifier (Eq. 9), and domain discriminator (Eq. 12). As the GNN layers follow the neighborhood aggregation strategy, the time complexity of representation learner is similar to the one of GraphSAGE [2] which is $\mathcal{O}\left(\prod_{k=1}^{K} s_k\right)$ for each node. s_k is the neighborhood sample size at search depth k. K is the maximum search depth. The time complexity of node classifier and domain discriminator is proportional to the number of nodes to be processed. Therefore, the overall complexity of AdaGIn is linear to the total number of nodes, that is, the scale of a graph.

G. Theoretical Analysis on Conditional Adversarial Networks

As introduced in Section III-E, a joint variable, $\boldsymbol{h}_v$, is constructed to be the outer product of classifier prediction (i.e., $\hat{\boldsymbol{y}}_v$) and embedding vector (i.e., $\boldsymbol{e}_v$). The label information, conveyed by classifier prediction $\hat{\boldsymbol{y}}_v$, potentially reveals the multimodal structure behind data distribution $\boldsymbol{e}_v \sim P(\boldsymbol{e}_v)$. Therefore, joint variable, $\boldsymbol{h}_v$, is expected to capture the multimodal structure. Following CDAN [20], we provide a generalization error analysis about the conditional adversarial networks. For simplicity, we ignore subscript, v, in the following discussions.

We consider the source and target domains over a fixed node representation space, $\boldsymbol{e}$, and a family of source node classifiers, G, in hypothesis space, $\mathcal{H}$ [26]. Let $\epsilon_S(G) = \mathbb{E}_{(\boldsymbol{e},\boldsymbol{y})\sim S}\left[G(\boldsymbol{e}) \neq \boldsymbol{y}\right]$ be the risk of a hypothesis $G \in \mathcal{H}$ regarding source domain distribution S. Disagreement between hypotheses $G_1, G_2 \in \mathcal{H}$ is denoted as $\epsilon_S(G_1, G_2) = \mathbb{E}_{(\boldsymbol{e},\boldsymbol{y})\sim S}\left[G_1(\boldsymbol{e}) \neq G_2(\boldsymbol{e})\right]$. Let $G^* =$

$\arg\min_G [\epsilon_S(G) + \epsilon_T(G)]$ be the ideal hypothesis that explicitly embodies the notion of adaptability. Note that target domain distribution, T, is different from source domain distribution, S, that is, $S \neq T$. The probabilistic bound [57] of target risk $\epsilon_T(G)$ of hypothesis G can be given with source risk $\epsilon_S(G)$ and domain discrepancy $|\epsilon_S(G, G^*) - \epsilon_T(G, G^*)|$.

$$\begin{aligned}\epsilon_T(G) \leqslant & \epsilon_S(G) + [\epsilon_S(G^*) + \epsilon_T(G^*)] \\ & + |\epsilon_S(G, G^*) - \epsilon_T(G, G^*)|.\end{aligned} \tag{15}$$

The goal of adversarial domain adaptation is to reduce domain discrepancy $|\epsilon_S(G, G^*) - \epsilon_T(G, G^*)|$.

Following the derivations of CDAN [20], domain discrepancy, $|\epsilon_S(G, G^*) - \epsilon_T(G, G^*)|$, has the following upper bound.

$$\begin{aligned}&|\epsilon_S(G, G^*) - \epsilon_T(G, G^*)| \\ &\leqslant \sup_{f_d \in \mathcal{H}_{\text{disc}}} \left|\mathbb{E}_{(\boldsymbol{e}, \hat{\boldsymbol{y}}) \sim S_G}[f_d(\boldsymbol{e}, \hat{\boldsymbol{y}}) = 1] + \mathbb{E}_{(\boldsymbol{e}, \hat{\boldsymbol{y}}) \sim T_G}[f_d(\boldsymbol{e}, \hat{\boldsymbol{y}}) = 0]\right| \\ &= \sup_{f_d \in \mathcal{H}_{\text{disc}}} \left|\mathbb{E}_{\boldsymbol{h} \sim S_G}[f_d(\boldsymbol{h}) = 1] + \mathbb{E}_{\boldsymbol{h} \sim T_G}[f_d(\boldsymbol{h}) = 0]\right|.\end{aligned} \tag{16}$$

in which $\mathcal{H}_{\text{disc}}$ is the family of domain discriminator f_d; $\hat{\boldsymbol{y}} = G(\boldsymbol{e})$ is the label prediction vector; $S_G = (\boldsymbol{e}, G(\boldsymbol{e}))_{\boldsymbol{e} \sim S(\boldsymbol{e})}$ and $T_G = (\boldsymbol{e}, G(\boldsymbol{e}))_{\boldsymbol{e} \sim T(\boldsymbol{e})}$ are the proxies of the joint distributions $S(\boldsymbol{e}, \boldsymbol{y})$ and $T(\boldsymbol{e}, \boldsymbol{y})$, respectively [58]; $\boldsymbol{h} = (\boldsymbol{e}, \hat{\boldsymbol{y}})$ is the joint variable.

The above upper bound has a close relation with domain adaptation loss, $\mathcal{L}_{\text{DA}}$, defined in Eq. 13. Referring to the minimax paradigm shown in Eq. 14, this supremum is achieved in the process of training the optimal discriminator f_d (i.e., $\max_{\boldsymbol{\theta}_d}(-\mathcal{L}_{\text{DA}})$), thus giving an upper bound of domain discrepancy $|\epsilon_S(G, G^*) - \epsilon_T(G, G^*)|$. Simultaneously, node representation, $\boldsymbol{e}$, is generated by representation learner, f_g, to minimize domain discrepancy (i.e., $\min_{\boldsymbol{\theta}_g}(-\mathcal{L}_{\text{DA}})$). Therefore, conditional adversarial networks can theoretically bound and reduce the domain discrepancy, thus aligning the multimodal distributions.

Furthermore, by minimizing the cross-entropy loss (i.e., $\min_{\boldsymbol{\theta}_g} \mathcal{L}_{\text{CE}}$, see Eq. 14), node classifier, f_c, is optimized to reduce source risk, $\epsilon_S(G)$. With the help of adversarial domain adaptation, domain discrepancy $|\epsilon_S(G, G^*) - \epsilon_T(G, G^*)|$ is bounded and minimized, encouraging the source risk to approximate the target risk more closely (see Eq. 15). Therefore, when applying the node classifier trained on the source graph to the target graph, target risk, $\epsilon_T(G)$, would also be reduced, resulting in improved node classification performance in the target graph.

IV. Experiments

In this section, we first evaluate the proposed method on the task of cross-graph node classification. An ablation study is then conducted on the contributions of mutual information maximization, multilinear conditioning, and domain adaptation. Next, we investigate the performance under varying common attribute rates. Finally, we analyze the influences of the critical hyperparameters, including initial learning rate, embedding dimension, and mutual information coefficient.

TABLE II
Summary of datasets.

Dataset	#Nodes	#Edges	Average Degree	#Attributes	#Union Attributes	#Labels
ACMv9	9,360	15,602	3.33	5,571	6,775	5
Citationv1	8,935	15,113	3.38	5,379		
DBLPv7	5,484	8,130	2.96	4,412		
Blog1	2,300	33,471	29.11	8,125	8,189	6
Blog2	2,896	53,836	37.18	8,162		

* "#Nodes" means the number of nodes. The rest can be deduced by analogy.

A. Experimental Setup

1) Datasets: As shown in Table II, following [6]–[8], experiments are conducted on five real-world networks. The three citation networks (i.e., ACMv9, Citationv1, and DBLPv7) are provided by ArnetMiner [59]. They are extracted from ACM, Microsoft Academic Graph, and DBLP, respectively. Papers in these three networks are published in different periods, i.e., after year 2010, before year 2008, and between years 2004 and 2008, respectively. Each citation network is modeled as an undirected graph. Specifically, a node represents one paper. An edge indicates a citation link between two papers, ignoring the direction. The attributes of a node are represented by a bag-of-words vector, indicating keywords extracted from the title of the corresponding paper. The number of union attributes (i.e., 6775) is the total number of attributes in these three graphs. According to the research topics, each node is categorized into some of the five classes, including "Database", "Artificial Intelligence", "Computer Vision", "Information Security", and "Networking".

Since papers in three citation networks are extracted from different databases and published in different time periods, the corresponding data distributions of these networks would also be diverse. Although label categories are the same, the statistics of graph scale, node attributes, and edge connections vary across graphs, indicating the intrinsic discrepancy between graphs. These graphs have no common nodes or edge connections, which further enlarges the divergence. Therefore, it would be challenging to transfer knowledge from one graph to facilitate node classification in another graph.

The two social networks (i.e., Blog1 and Blog2) are disjoint subnetworks from the BlogCatalog dataset [60]. A node represents one blogger. An undirected edge indicates the friendship between two bloggers. Node attribute vector is obtained using the keywords in the blogger's self-description. Each node belongs to one class that is designated by the blogger's interest group. Compared with the citation networks, the social networks have higher average degrees, indicating each node has a larger number of neighbors. Nodes in the social networks also have richer attributes. Since the two social networks are extracted from the same network, they have close attribute distributions.

The proposed method is evaluated by conducting multilabel node classification in six transfer tasks, including D→A, A→D, C→A, A→C, D→C, and C→D. A, C, and D denote ACMv9, Citationv1, and DBLPv7, respectively. The arrow,

TABLE III
COMMON ATTRIBUTE RATE IN EACH TRANSFER TASK.

Transfer Task	D→A A→D	C→A A→C	D→C C→D	B1→B2 B2→B1
#Common Attributes	3,621	4,285	3,783	8,098
#Union Attributes	6,362	6,665	6,008	8,189
Common Attribute Rate	56.92%	64.29%	62.97%	98.89%

"→", indicates the direction of knowledge transfer, that is, from a fully labeled source graph to a completely unlabeled target graph. The evaluation is further conducted by performing multiclass classification in transfer tasks B1→B2 and B2→B1. The two social networks, Blog1 and Blog2, are denoted by B1 and B2, respectively. The common attribute rate in each transfer task can be found in Table III, where the number of union attributes is the total number of attributes in the designated source and target graphs.

2) Baselines: The baselines are of three kinds, including graph neural networks for learning on a single graph, typical domain adaptation methods, and transfer learning methods specifically designed for graph data.

- GCN [3], GAT [42], and GraphSAGE [2]: They are well-known graph neural networks for single-graph node classification. GCN devises a simplified spectral filter to perform convolution on graphs. GraphSAGE samples fixed-size neighbors first, then generates node representations by aggregating neighborhood information. Instead of neighborhood sampling, GAT assigns learnable weights to all neighboring nodes in the aggregation process. These three GNN models are trained on the labeled source graph, and then directedly evaluated on the unlabeled target graph.
- DANN [26], CDAN [20], and WDGRL [27]: These adversarial domain adaption approaches are originally designed for transfer learning on images or text. To process graph data, their representation learning modules are constructed as multilayer perceptrons (MLPs), which only take node attributes as input, ignoring the graph structure.
- GCC [55], NetTr [52], CDNE [6], ACDNE [7], AdaGCN [8], and UDA-GCN [9]: They are designed for cross-graph learning. The GCC model is pre-trained to discover common structural patterns across multiple source graphs, and then fine-tuned on the target graph. NetTr discovers structural representations that are transferrable across graphs. CDNE achieves domain adaptation by incorporating the MMD loss [54] across graphs in an autoencoder architecture. ACDNE utilizes two feature extractors to separately preserve attribute affinity and topology proximity. UDA-GCN develops a dual graph convolutional network. Both ACDNE and UDA-GCN employs the gradient reversal layer [26] to perform domain adaptation. AdaGCN improves the transferrability of GCN [3] by minimizing the Wasserstein distance [28] between the source and target representations.

3) Implementation Details: The proposed method, AdaGIn, is implemented using PyTorch. Table IV presents the hyperparameters selected for each transfer task. Representation learner, f_g, consists of the GNN layers in which the layer sizes are selected in $\{128, 256, 512, 1024, 2048\}$. For example, in transfer task D→A, the dimensions of three GNN layers are 1024, 512, and 128 in sequence, which are denoted as "$1024/512/128$". The dropout rate of each GNN layer is 0.5. Node classifier, f_c, is a logistic regression. Domain discriminator, f_d, is a multilayer perceptron (MLP) containing three hidden layers, with the layer dimensions set as 512, 64, and 16. Its output is of one dimension followed by a sigmoid activation.

We train the AdaGIn model over shuffled minibatches using the Adam optimizer [61], with a batch size of 32. Initial learning rate, η_0, is chosen from $\{0.005, 0.010, 0.015\}$. To prevent overfitting, L_2 regularization is imposed on all learnable weights, with the weight decay term set as 5×10^{-5}. Following DANN [26], learning rate, η_p, is decayed by $\eta_p = \eta_0 \left(1 + 10p\right)^{-0.75}$, where training progress p increases from 0 to 1. In the overall loss function (i.e., Eq. 14), mutual information coefficient, λ_1, is set as 0.1 for the citation graphs and 1.0 for the social graphs. Starting from 0, domain adaptation coefficient, λ_2, is progressively increased by $\lambda_2 = 2\left(1 + \exp\left(-10p\right)\right)^{-1} - 1$. The maximum value of λ_2 is set as 0.2 for the citation graphs.

GCN and AdaGCN are implemented with PyTorch following the AdaGCN paper [8]. We train both models for 200 epochs with learning rates selected in $\{0.005, 0.010, 0.015, 0.020\}$. The GCN model embedded in AdaGCN has three layers with the layer sizes chosen from $\{128, 256, 512, 1024, 2048\}$. Domain adaptation coefficient is searched in $\{0.001, 0.01, 0.1\}$. Gradient penalty coefficient is chosen from $\{1, 10, 100\}$. The training step of domain discriminator is either 10 or 20. The domain discriminator of AdaGCN is removed when calculating the results of GCN.

The setup of GAT follows the original paper [42], while two attention heads are applied in the first layer due to the memory constraint. We adapt GraphSAGE to transductive setting, and employ its max-pooling variant (i.e., GS-pool) due to its preferable performance and computational efficiency [2]. The architectures of DANN and CDAN are similar to AdaGIn, except that their representation learners are MLPs. WDGRL also employs a MLP to generate node representations, with the settings similar to AdaGCN.

With only one source graph in our case, it is unfeasible to pre-train a GCC model, since its pre-training requires multiple source graphs as input. Therefore, we use the pre-trained model[2] provided by the authors to generate node representations for the source and target graphs. Then we train a logistic regression classifier using the labeled source graph. The trained classifier is subsequently applied to classify nodes in the unlabeled target graph.

The results of NetTr are directly taken from the ACDNE paper [7]. In order to calculate standard deviations and perform

[2]https://github.com/THUDM/GCC

TABLE IV
HYPERPARAMETER SELECTION FOR ADAGIN.

Source	Target	Epoch	η_0	Weight Decay	Dropout	Batch Size	f_g	s^*	f_d
D	A	50	0.010	5×10^{-5}	0.5	32	1024/512/128	$\{10, 10, 10\}$	512/64/16
A	D								
C	A		0.015				512/128/128		
A	C		0.010				1024/256/128		
D	C		0.005				1024/512/128		
C	D		0.010				2048/128	$\{10, 10\}$	
B1	B2	100						$\{10, 2\}$	
B2	B1		0.015						

* A set $\{s_1, \ldots, s_K\}$ contains the neighborhood sample size s_k at each search depth $k \in \{1, \ldots, K\}$.

t-test, we execute the official source codes [3] [4] to evaluate CDNE and ACDNE. We find the results of CDNE and ACDNE are slightly better than, or very close to, the reported ones in the ACDNE paper. UDA-GCN is also tested using its source codes[5] with almost every key hyperparameter tuned. There are two or three GNN layers with the layer sizes selected in $\{128, 256, 512, 1024, 2048\}$. The learning rate is swept in $\{0.0001, 0.001, 0.01\}$. The weight decay coefficient is chosen from $\{0.00005, 0.0005, 0.005, 0.05, 0.5\}$.

Following ACDNE, the embedding dimension (i.e., l) is set as 128 for every method except for GCC. The embedding dimension of GCC is fixed as 64 by the provided pre-trained model. For each method, we report the mean values and the standard deviations of F1 scores after five runs with different random seeds. Note that the standard deviations of NetTr are not reported in the ACDNE paper. All the experiments are conducted on a computer with one NVIDIA GeForce GTX 1080Ti GPU (11 GB of RAM), an Intel(R) Core(TM) i7-8700K CPU (6 cores, 3.70 GHz), and 32 GB of RAM.

B. Cross-graph Node Classification

Experiments are conducted under the unsupervised domain adaptation setting, in which the source graph is fully labeled, and the target graph is completely unlabeled. The F1 scores (i.e., Micro-F1 score and Macro-F1 score) are reported to quantify the node classification performance in the target graph. We introduce the experimental results in the citation and social graphs separately.

1) Node Classification in Citation Graphs: As shown in Table V and Table VI, the first group of baselines are GNNs including GCN [3], GAT [42], and GraphSAGE [2]. The GNN models are trained in the source graph and then directly evaluated in the target graph. Domain adaptation techniques are not utilized in these cases. Since the information of target graph is unavailable during training, referring to DANN [26], these baselines are called as the source-only models.

GCN performs best in this group. The strengths of GNN models are twofold. One is that the GNN models explore and capture structural information, like edge connections, to learn meaningful node representations. Such structural relations are commonly seen among graphs. Thus, the GNN model trained in the source graph has the potential to encode important structural properties in the target graph. The other is that the three citation graphs share some common attributes (see Table III). The GNN models considered here can jointly encode node attributes and graph structure to compute node representations. The common attributes help shrink the domain divergence between graphs, thus improving the model performance when testing in the target graph. Further investigation on common attribute rate are presented in Section IV-D.

In the second group of baselines (i.e., DANN [26], CDAN [20], and WDGRL [27]), the models only take node attributes as input, ignoring the structural relations between nodes. It is similar to the way when processing the image data which is assumed to be independent and identically distributed (i.i.d.). Even though domain adaptation techniques are employed in these methods, their performance is generally inferior to other groups, including the source-only models. Therefore, when solely utilizing the node attributes of citation graphs, the generated node representations are not that label-discriminative. Furthermore, in such cases, domain discrepancy would be too large to be effectively reduced by these domain adaptation techniques. Since the nodes in a graph are correlated by edge connections violating the i.i.d. assumption, to improve classification performance, it is crucial to devise methods that incorporate structural information.

In the third group of baselines (i.e., GCC [55], NetTr [52], CDNE [6], ACDNE [7], AdaGCN [8], and UDA-GCN [9]), clear underperformance is observed in GCC. As introduced in Section IV-A3, although node representations can be generated using the pre-trained GCC model, a node classifier cannot be directly trained using the unlabeled target graph. Instead, we have to train the classifier with the labeled source graph first, and then test it in the target graph. It degrades the performance of GCC model. More importantly, GCC cannot exploit node attributes which are crucial for representation learning in the attributed graphs. Specifically, the affinity of node attributes could be a predictor for node labels [7]. The neighborhood attribute information also contributes to learning meaningful node representations [2]. Similarly, NetTr learns transferrable latent representations solely based on graph topology, consequently leading to underperformance.

In Table V and Table VI, AdaGIn ranks first in all six transfer tasks, yielding the highest averaged F1 scores. ACDNE is

[3]https://github.com/shenxiaocam/CDNE
[4]https://github.com/shenxiaocam/ACDNE
[5]https://github.com/mandy976/UDAGCN

TABLE V
MICRO-F1 SCORE (%) OF NODE CLASSIFICATION IN TARGET CITATION GRAPH.

Method	D→A	A→D	C→A	A→C	D→C	C→D	Average
GCN [3]	70.80 (0.67)	72.85 (0.58)	74.84 (0.65)	77.28 (1.26)	75.39 (0.54)	75.45 (0.44)	74.44
GAT [42]	65.64 (0.66)	71.34 (0.46)	70.06 (0.43)	75.80 (0.38)	72.44 (0.36)	73.79 (0.51)	71.51
GraphSAGE [2]	61.10 (1.40)	63.94 (2.66)	68.30 (1.26)	67.34 (1.83)	66.66 (1.83)	69.81 (0.88)	66.19
DANN [26]	51.70 (0.57)	53.84 (0.76)	53.23 (1.07)	53.42 (0.76)	52.42 (0.98)	56.58 (0.81)	53.53
CDAN [20]	51.43 (0.42)	53.78 (0.92)	53.18 (1.27)	53.70 (0.98)	52.64 (0.84)	56.73 (0.72)	53.58
WDGRL [27]	47.75 (1.19)	50.46 (0.71)	51.52 (1.41)	49.70 (1.77)	50.03 (2.01)	51.78 (1.03)	50.21
GCC [55]	27.25 (0.31)	21.85 (0.23)	27.93 (0.13)	23.09 (0.25)	40.43 (0.25)	37.97 (0.11)	29.75
NetTr [52]	56.23	56.30	57.75	58.81	59.11	59.88	58.01
CDNE [6]	77.01 (0.32)	71.99 (0.29)	77.42 (0.63)	79.00 (0.32)	79.51 (0.45)	74.10 (0.24)	76.51
ACDNE [7]	76.30 (0.53)	76.92 (0.44)	79.58 (0.30)	83.31 (0.10)	82.14 (0.23)	77.24 (0.90)	79.25
AdaGCN [8]	73.78 (1.23)	75.15 (0.30)	76.37 (0.79)	81.67 (0.43)	79.61 (0.91)	76.67 (0.24)	77.21
UDA-GCN [9]	65.93 (2.03)	73.33 (1.02)	70.76 (2.36)	75.50 (1.68)	70.94 (2.12)	74.86 (1.34)	71.89
AdaGIn [ours]	**78.07** (0.55)	**77.73** (0.32)	**79.69** (0.46)	**83.49** (0.17)	**83.16** (0.49)	**77.83** (0.09)	**80.00**

* A: ACMv9, C: Citationv1, D: DBLPv7. In each column, the highest F1 score is highlighted in bold and the top two are underlined. The standard deviations are given in parentheses.

TABLE VI
MACRO-F1 SCORE (%) OF NODE CLASSIFICATION IN TARGET CITATION GRAPH.

Method	D→A	A→D	C→A	A→C	D→C	C→D	Average
GCN [3]	69.85 (0.54)	70.35 (0.69)	74.07 (0.66)	75.09 (1.32)	73.22 (0.48)	73.20 (0.54)	72.63
GAT [42]	56.00 (1.75)	64.72 (0.61)	62.91 (0.39)	69.38 (0.62)	64.09 (2.11)	68.12 (0.61)	64.20
GraphSAGE [2]	57.84 (2.06)	59.67 (2.63)	65.99 (1.79)	62.62 (0.80)	60.30 (1.80)	66.88 (1.16)	62.22
DANN [26]	47.61 (0.57)	49.80 (0.60)	50.37 (1.17)	51.39 (0.90)	49.65 (0.94)	52.67 (0.17)	50.25
CDAN [20]	47.00 (0.61)	49.75 (0.34)	49.39 (1.71)	50.83 (0.82)	49.55 (1.03)	52.89 (0.39)	49.90
WDGRL [27]	42.91 (1.49)	45.73 (1.48)	47.75 (2.05)	46.54 (0.99)	46.00 (1.72)	46.67 (0.62)	45.93
GCC [55]	19.17 (0.24)	15.19 (0.23)	21.77 (0.13)	16.69 (0.14)	30.86 (0.21)	28.83 (0.22)	22.09
NetTr [52]	50.99	49.80	53.44	55.46	55.53	55.18	53.40
CDNE [6]	76.37 (0.46)	69.72 (0.23)	77.11 (0.66)	77.08 (0.33)	77.96 (0.41)	71.86 (0.25)	75.02
ACDNE [7]	76.23 (0.55)	75.14 (1.02)	78.97 (0.53)	81.64 (0.06)	80.10 (0.56)	75.88 (0.97)	77.99
AdaGCN [8]	73.62 (1.30)	72.80 (0.51)	75.97 (0.72)	80.13 (0.48)	77.47 (0.88)	74.83 (0.54)	75.80
UDA-GCN [9]	56.64 (5.39)	70.44 (1.81)	69.30 (2.21)	72.95 (1.92)	61.64 (2.72)	70.46 (3.20)	66.91
AdaGIn [ours]	**77.76** (0.53)	**75.57** (0.44)	**79.21** (0.50)	**82.02** (0.27)	**81.67** (0.43)	**76.16** (0.32)	**78.73**

* A: ACMv9, C: Citationv1, D: DBLPv7. In each column, the highest F1 score is highlighted in bold and the top two are underlined. The standard deviations are given in parentheses.

the best performing baseline. In Table V, the relative performance gains of AdaGIn over ACDNE are 2.32%, 1.05%, and 1.24% in transfer tasks D→A, A→D, and D→C, respectively. In the t-test, the corresponding p-values are 1.57×10^{-3}, 4.85×10^{-2}, and 3.81×10^{-2}, respectively. As the p-values are smaller than 0.05, the improvements of AdaGIn over ACDNE are statistically significant in these transfer tasks. In summary, AdaGIn surpasses the prior arts by incorporating the local-global mutual information and conditional adversarial domain adaptation.

2) Node Classification in Social Graphs: As reported in Table VII, the methods discussed above are further tested in the social graphs. The domain discrepancy between Blog1 and Blog2 is reduced by their close attribute distributions (see Section IV-A1). Therefore, the F1 scores of AdaGIn are much higher than those in the citation graphs. AdaGIn outperforms ACDNE in transfer tasks B1→B2 and B2→B1. The corresponding p-values (i.e., 4.95×10^{-3} and 3.57×10^{-2}) are smaller than 0.05, indicating that the improvements are statistically significant. Note that AdaGIn has an improvement over CDAN by incorporating the structural information in the domain adaptation process.

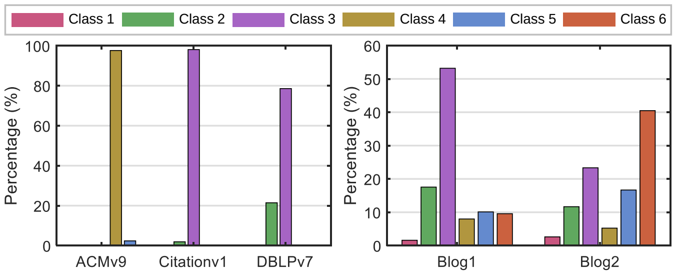
As shown in Table II, compared with the citation graphs, the social graphs have richer node attributes and higher average degrees, which means the nodes have more attributes and neighbors. Therefore, when evaluated in the social graphs, we observe distinct relative performance among the methods.

The spectral GNN methods, AdaGCN and UDA-GCN, have clear underperformance in comparison with AdaGIn. The spectral GNNs exploit the whole neighborhood of a node,

TABLE VII
F1 SCORE (%) OF NODE CLASSIFICATION IN TARGET SOCIAL GRAPH.

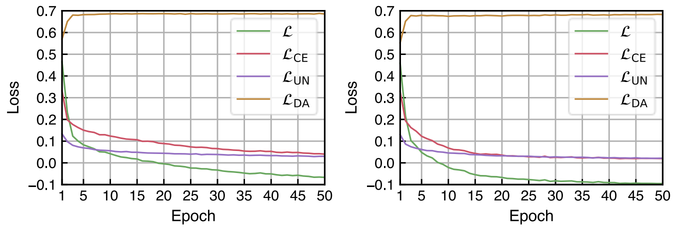
	Source	Target	GCN [3]	GAT [42]	GraphSAGE [2]	DANN [26]	CDAN [20]	WDGRL [27]	GCC [55]	NetTr [52]	CDNE [6]	ACDNE [7]	AdaGCN [8]	UDA-GCN [9]	AdaGIn [ours]
Micro-F1 (%)	B1	B2	68.60 (0.82)	72.85 (0.50)	91.26 (0.05)	91.66 (0.28)	91.66 (0.48)	87.49 (0.65)	27.55 (0.80)	82.18	87.67 (0.68)	95.39 (0.10)	70.12 (0.20)	74.41 (0.80)	**96.15** (0.32)
	B2	B1	69.14 (1.29)	73.23 (0.79)	91.50 (0.08)	91.51 (0.43)	91.53 (0.38)	87.52 (2.41)	26.99 (0.51)	82.17	84.09 (0.44)	94.92 (0.17)	70.01 (0.25)	74.82 (0.76)	**95.25** (0.10)
	Average		68.87	73.04	91.38	91.59	91.60	87.51	27.27	82.18	85.88	95.16	70.07	74.62	**95.70**
Macro-F1 (%)	B1	B2	68.18 (0.98)	72.16 (0.73)	91.15 (0.05)	91.55 (0.29)	91.57 (0.48)	87.37 (0.65)	26.15 (0.68)	81.83	87.55 (0.67)	95.32 (0.10)	69.63 (0.26)	74.21 (0.90)	**96.11** (0.33)
	B2	B1	68.14 (1.54)	72.04 (0.76)	91.29 (0.08)	91.30 (0.43)	91.31 (0.39)	87.32 (2.37)	26.36 (0.53)	81.91	83.95 (0.43)	94.81 (0.18)	69.09 (0.28)	74.10 (0.94)	**95.16** (0.10)
	Average		68.16	72.10	91.22	91.43	91.44	87.35	26.26	81.87	85.75	95.07	69.36	74.16	**95.64**

[*] B1: Blog1, B2: Blog2. In each row, the highest F1 score is highlighted in bold and the top two are underlined. The standard deviations are given in parentheses.



(a) Citation graphs (b) Social graphs

Fig. 4. Percentage of each class nodes in the neighborhood. The central node is the largest degree node of a graph. Class index of the central node is 4, 3, 3, 3, and 6 in ACMv9, Citationv1, DBLPv7, Blog1, and Blog2, respectively.



(a) From C to A (b) From D to A

Fig. 5. Loss over training epochs.

which might introduce certain noise and degrade the model performance. Instead, AdaGIn reduces noise by sampling some of the neighboring nodes. Similarly, among the source-only models, GraphSAGE largely outperforms GCN and GAT with the help of neighborhood sampling. Although GAT is a spatial GNN model, it also exploits the entire neighborhood without sampling [42]. In addition, even though DANN and CDAN ignore the neighborhood information, they still have impressive performance, which reveals that node attributes in the social graphs are informative for learning meaningful node representations.

The benefit of neighborhood sampling in the social graphs can be supported by analyzing the labels of neighboring nodes. It can be seen in Figure 4 that, in the citation graphs, most neighboring nodes belong to the class of the central node. However, in the social graphs, only around 50%, or even less, neighboring nodes are of the same class as the central node. For clarity, we use the same set of class index for the citation and social graphs, although these two kinds of graphs have different definitions for each class. Note that, there are five classes in the citation graphs and six classes in the social graphs, respectively.

GCN and GAT compute the representation of a node as the weighted average of its previous representation and its neighbors'. Such smoothing operation makes the connected nodes have close representations, which leads to similar label predictions in the subsequent classification task [18]. The homophily assumption behind this smoothing operation is that the nodes with edge connections are likely to be of the same class [3]. Figure 4 shows that there are more neighboring nodes violating this assumption in the social graphs. GCN and GAT utilize the complete neighbor set, which would lead to their underperformance in the social graphs. AdaGCN and UDA-GCN follow the way of GCN to compute node representations, thus also encountering performance degradation. Instead, AdaGIn and GraphSAGE sample nodes from the neighborhood. The sampled neighbor set has fewer nodes that belong to classes different from the one of the central node. The classification performance is consequently improved.

3) Loss Curve during Training: Figure 5 shows the curve of each loss with respect to the training epoch. Two representative transfer tasks, C→A and D→A, are presented as examples. According to Eq. 14, the overall loss for training the GNN layers can be computed by $\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda_1 \mathcal{L}_{\mathrm{UN}} - \lambda_2 \mathcal{L}_{\mathrm{DA}}$. There is a steady decrease in the overall loss with a reduced decrease rate, indicating the model training gradually converges. Similar trends can be seen in supervised loss $\mathcal{L}_{\mathrm{CE}}$ and unsupervised loss $\mathcal{L}_{\mathrm{UN}}$. In contrast, with the overall loss minimized during training, domain adaptation loss, $\mathcal{L}_{\mathrm{DA}}$, increases first and converges around five epochs. Note that the GNN layers are trained to raise the domain adaptation loss, thus generating domain-invariant node representations.

C. Ablation Study

In this section, we investigate three key components in the AdaGIn model, including mutual information maximization (MI), multilinear conditioning (CD), and domain adaptation (DA). The investigation is conducted by removing some of the components to construct a variant of AdaGIn (see the left side of Table VIII or Table IX). The contribution of one component can be indicated by the changes in F1 scores caused by its removal. The following AdaGIn variants are constructed for comparison.

TABLE VIII
MICRO-F1 SCORE (%) OF ADAGIN VARIANTS.

MI	CD	DA	Model Variant	D→A	A→D	C→A	A→C	D→C	C→D	B1→B2	B2→B1	Average
✓	✓	✓	AdaGIn	**78.07** (0.55)	**77.73** (0.32)	**79.69** (0.46)	**83.49** (0.17)	**83.16** (0.49)	**77.83** (0.09)	**96.15** (0.32)	**95.25** (0.10)	**83.92**
	✓	✓	AdaGIn-MI	72.91 (0.44)	75.48 (0.60)	76.24 (0.56)	80.99 (0.34)	78.83 (0.65)	76.10 (0.29)	93.29 (0.56)	92.82 (0.15)	80.83
✓		✓	AdaGIn-CD	77.10 (0.48)	76.81 (0.09)	78.97 (0.39)	83.18 (0.24)	82.25 (0.16)	77.38 (0.26)	95.92 (0.06)	94.83 (0.36)	83.31
✓			AdaGIn-DA	76.16 (0.19)	75.46 (0.11)	78.71 (0.21)	82.68 (0.16)	81.95 (0.10)	76.71 (0.45)	93.05 (0.29)	92.66 (0.14)	82.17
		✓	AdaGIn-MI-CD	72.58 (0.36)	74.92 (0.59)	76.01 (0.16)	80.58 (0.40)	78.92 (0.41)	76.56 (0.39)	93.98 (0.31)	93.08 (0.18)	80.83
			AdaGIn-MI-DA	68.44 (0.76)	70.93 (0.54)	73.31 (0.11)	76.64 (0.56)	73.45 (1.33)	73.01 (0.37)	91.91 (0.15)	91.99 (0.11)	77.46

* MI: mutual information maximization, CD: multilinear conditioning, DA: domain adaptation. The signs "✓" and "-" indicate the existence and removal of one component, respectively. The highest Micro-F1 score in each column is highlighted in bold. The standard deviations are given in parentheses.

TABLE IX
MACRO-F1 SCORE (%) OF ADAGIN VARIANTS.

MI	CD	DA	Model Variant	D→A	A→D	C→A	A→C	D→C	C→D	B1→B2	B2→B1	Average
✓	✓	✓	AdaGIn	**77.76** (0.53)	**75.57** (0.44)	**79.21** (0.50)	**82.02** (0.27)	**81.67** (0.43)	**76.16** (0.32)	**96.11** (0.33)	**95.16** (0.10)	**82.96**
	✓	✓	AdaGIn-MI	72.22 (0.67)	73.47 (1.21)	75.64 (0.45)	79.45 (0.43)	76.45 (0.65)	74.49 (0.57)	93.23 (0.58)	92.64 (0.16)	79.70
✓		✓	AdaGIn-CD	76.92 (0.53)	74.85 (0.28)	78.40 (0.27)	81.89 (0.35)	80.74 (0.25)	75.98 (0.30)	95.87 (0.07)	94.71 (0.36)	82.42
✓			AdaGIn-DA	75.39 (0.29)	73.03 (0.39)	78.07 (0.34)	80.84 (0.16)	80.14 (0.18)	74.90 (0.52)	92.98 (0.28)	92.51 (0.15)	80.98
		✓	AdaGIn-MI-CD	71.97 (0.22)	73.19 (1.01)	75.51 (0.09)	79.07 (0.60)	77.45 (0.41)	75.34 (0.67)	93.91 (0.31)	92.89 (0.18)	79.92
			AdaGIn-MI-DA	66.84 (0.91)	68.15 (0.47)	71.78 (0.35)	74.22 (0.41)	71.20 (1.05)	70.91 (0.54)	91.82 (0.14)	91.81 (0.08)	75.84

* MI: mutual information maximization, CD: multilinear conditioning, DA: domain adaptation. The signs "✓" and "-" indicate the existence and removal of one component, respectively. The highest Macro-F1 score in each column is highlighted in bold. The standard deviations are given in parentheses.

- AdaGIn-MI: A variant of AdaGIn without mutual information maximization (MI). The unsupervised loss is removed from the overall loss function (i.e., Eq. 14).
- AdaGIn-CD: A variant of AdaGIn without multilinear conditioning (CD). The domain discriminator is not conditioned on the label predictions. It directly takes the embedding vectors as input and outputs domain predictions.
- AdaGIn-DA: A variant of AdaGIn without domain adaptation (DA). We remove the domain discriminator in the model architecture and the domain adaptation loss in the overall loss function (i.e., Eq. 14).
- AdaGIn-MI-CD: A variant of AdaGIn without mutual information maximization (MI) and multilinear conditioning (CD).
- AdaGIn-MI-DA: A variant of AdaGIn without mutual information maximization (MI) and domain adaptation (DA).

Note that when the domain discriminator is removed, the multilinear conditioning disappears simultaneously. Therefore, AdaGIn-MI-DA is a variant without all the three key components.

1) Performance of Model Variants: As shown in Table VIII and Table IX, the averaged Micro-F1 score of AdaGIn drops by 3.09% (absolute difference) when the MI component is removed (i.e., AdaGIn-MI), which empirically validates the importance of preserving the local-global mutual information. Similarly, in the averaged Micro-F1 score, a 1.75% drop (absolute difference) is caused by the removal of DA component from the AdaGIn model (see AdaGIn-DA). Mutual information maximization is applied in both the source and target graphs. Therefore, although the DA component has been removed, AdaGIn-DA is still optimized with the unsupervised loss calculated in the target graph. Compared with AdaGIn, this helps keep the performance from dropping greatly. To validate this point, we further remove the MI component to obtain a source-only model, i.e., AdaGIn-MI-DA. The information of target graph is entirely unknown when training the source-only model. In comparison with AdaGIn-MI, there is a decrease by 3.37% (absolute difference) in the averaged Micro-F1 score due to the lack of DA component.

The domain adaptation component of AdaGIn conditions domain discriminator on the label predictions. The efficacy of multilinear conditioning is empirically validated by the performance drop due to its removal (see AdaGIn-CD). However, when MI is removed, the improvement of AdaGIn-MI over AdaGIn-MI-CD cannot be consistently observed in all transfer tasks. Mutual information maximization contributes to learning more informative node representations and more accurate label predictions, which is likely to enhance the capability of multilinear conditioning.

In Table VIII and Table IX, we observe the MI component produces larger performance gains when the target graph is of large scale (i.e., ACMv9 and Citationv1). For example, considering the absolute difference between AdaGIn and AdaGIn-MI, when the mutual information is maximized, the Micro-F1 score increases by 5.16% in D→A and 4.33% in D→C which are much higher than 1.73% in C→D. As introduced in Section III-C, neighborhood information is repeatedly aggregated to generate the representation of a node. Therefore, node representation can be treated as the local patch representation

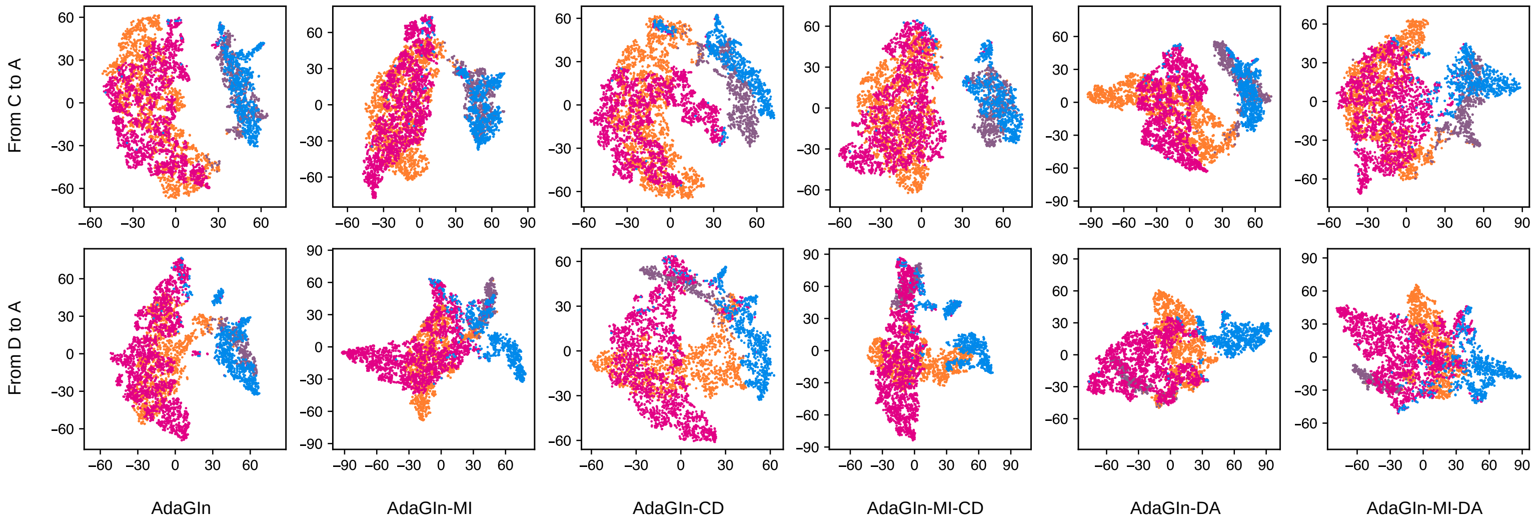


Fig. 6. Node representations visualized in the 2D space using t-SNE (best viewed in color). Each point represents one node. The points in orange and brown are from the source graph, and the points in pink and blue are from the target graph. The orange and pink points belong to "Computer Vision", whereas the brown and blue points belong to "Information Security".

in a graph. To avoid memory overflow, we have to restrict the neighborhood search depth and the sample size in each depth, which results in a limited patch size. Consequently, if a graph consists of a large number of nodes, it would be more difficult for the node representation to capture the global structural information. Under such circumstances, the model performance is more likely to benefit from the maximization of local-global mutual information.

2) Visualization of Node Representations: As a qualitative evaluation, Figure 6 visualizes the node representations in the 2D space using t-SNE [62]. For clarity, two transfer tasks are taken as examples (i.e., C→A and D→A), in which ACMv9 serves as the target graph. We only show the nodes from two classes, i.e., "Computer Vision" and "Information Security". Node representations generated by AdaGIn have the most preferable visualization. Specifically, the same class nodes are projected together, regardless of whether they are from the source or target graph. Furthermore, the clusters of the two classes are separated more clearly. By comparing the visualizations of AdaGIn and its variants, we have the following observations.

- Mutual information maximization (MI) improves the label predictions, consequently making the clusters of the two classes more separable in most cases.
- Domain adaptation (DA), which includes a conditional discriminator, improves the multimodal alignment in the 2D space. Specifically, the clusters of the two classes have a clearer separation. Meanwhile, a larger overlap can be seen in the clusters of the same class.
- Multilinear conditioning (CD) contributes to distinguishing the clusters of different classes and aligning the clusters of the same class together.

3) Effect of Global Information: We further analyze the effect of global information encoded from various domains. Specifically, we use "SMI" and "TMI" to denote the mutual information maximization applied in the source graph and in the target graph, respectively. Then two AdaGIn variants are constructed for comparison.

TABLE X
MODEL PERFORMANCE (MICRO-F1, %) WITH GLOBAL INFORMATION ENCODED FROM VARIOUS DOMAINS.

Model Variant	Citation Graphs	Social Graphs
AdaGIn	**80.00**	**95.70**
AdaGIn-MI	76.76	93.06
AdaGIn-TMI	77.08	94.53
AdaGIn-SMI	79.08	94.72

* TMI: the MI component applied in the target graph, SMI: the MI component applied in the source graph. The highest Micro-F1 score in each column is highlighted in bold.

- AdaGIn-TMI: A variant of AdaGIn without mutual information maximized in the target graph. In other words, MI is merely applied in the source graph to encode the global structural information.
- AdaGIn-SMI: A variant of AdaGIn without applying mutual information maximization in the source graph. From another perspective, MI is solely applied in the target graph to encode the global structural information.

As shown in Table X, when the MI component is removed from one of the graphs, there is a clear performance drop (see AdaGIn-TMI and AdaGIn-SMI). It indicates that the model performance benefits from the MI component regardless of which graph it is applied. Furthermore, the decrease of Micro-F1 score observed in AdaGIn-TMI is larger than the one in AdaGIn-SMI. Since the model is evaluated in the target graph, it would be more beneficial to directly maximize the mutual information in the target graph. With the MI component

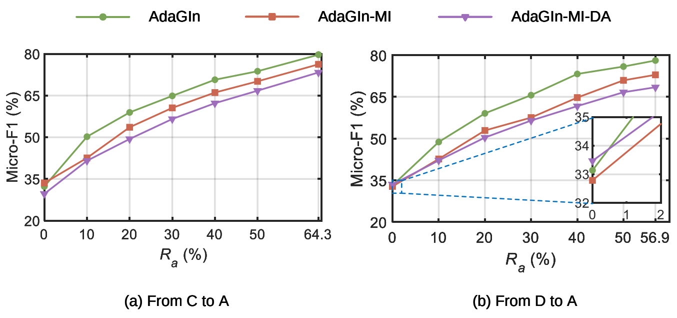


Fig. 7. Micro-F1 score under varying common attribute rate.

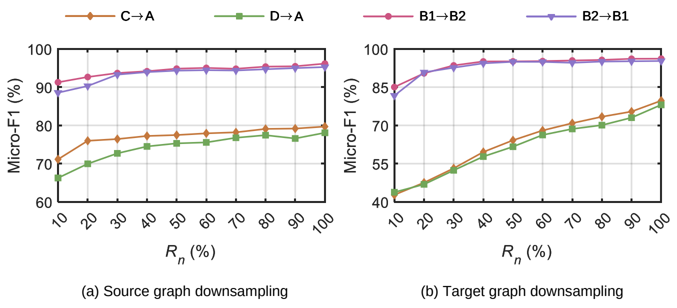


Fig. 8. Analysis of the model performance with the source or target graph downsampled.

removed from both the source graph and the target graph, the Micro-F1 score has the largest drop (see AdaGIn-MI). Therefore, it is desirable to jointly apply the MI component in the source and target graphs. The above observations are found to be consistent in the citation graphs and the social graphs.

D. Distribution Discrepancy Analysis

In this section, we investigate the influence of common attribute rate (i.e., R_a) on the model performance. As introduced in Section III-A, common attribute rate, R_a, is the percentage of common attributes shared by the source and target graphs. We conduct the investigation by removing some of the common attributes. Lower common attribute rate indicates larger distribution discrepancy between the source and target graphs.

In the two representative transfer tasks (i.e., C→A and D→A), the original common attribute rates are 64.29% and 56.92%, respectively (see Table III). In Figure 7, by comparing with the source-only model, AdaGIn-MI-DA, we showcase the performance gain of AdaGIn. A positive performance gain indicates the improvement obtained by mutual information maximization (MI) and domain adaptation (DA). To investigate these two learning strategies individually, we further present the variant AdaGIn-MI. Note that multilinear conditioning is applied when performing domain adaptation.

In Figure 7, the Micro-F1 score of each model decreases with the common attribute rate. The reason is that domain gap is enlarged between the source and target graphs by reducing common attributes. Furthermore, the total number of node attributes also decreases in each graph due to the removal of common attributes. As described in Section III-C, node representation learning relies on node attributes, the reduction of which increases the difficulty of learning informative node representations, consequently degrading the node classification performance. AdaGIn outperforms the source-only model, AdaGIn-MI-DA, in all cases except $R_a = 0$ in transfer task D→A. We explain the reasons by discussing mutual information maximization and domain adaptation individually.

Mutual information maximization improves the classification performance in a wide range of common attribute rates. However, without any common attributes (i.e., $R_a = 0$), we observe marginal improvement in D→A and no improvement in C→A. Local-global mutual information is calculated and maximized using node attributes (see Section III-C). If all common attributes are removed, the remaining node attributes would not be that informative. For example, in transfer task C→A, when $R_a = 0$, there are merely 20.34% and 23.08% node attributes left in Citationv1 and ACMv9, respectively (see Table II and Table III). Under such conditions, mutual information maximization would be weakened, which even results in underperformance.

Domain adaptation also contributes to performance improvement in some cases. However, the performance gain, yielded by applying domain adaptation, sometimes shrinks or even becomes negative (see transfer task D→A), as the common attribute rate gets smaller. As stated in [10], domain adaptation techniques might be unsuccessful when the source and target domains share few similarities. In the worst case, such techniques may have negative impacts on the learning tasks in the target domain, which is referred to as negative transfer.

E. Sample Complexity Analysis

In this section, we analyze how the model performance is influenced by the number of nodes in the source or target graph. We follow [63] to conduct the experiments under domain adaptation scenario. In Figure 8(a), the source graph is downsampled by selecting the same percentage of nodes from each class. Percentage, R_n, represents the proportion of nodes selected from the graph. When $R_n = 100\%$, there are no nodes being removed from a graph. We report the Micro-F1 score in a target graph that is unchanged. Similarly, in Figure 8(b), we fix the source graph and report the F1 scores in a downsampled target graph where only a percentage of nodes per class is preserved. Note that, differing from the i.i.d. images in computer vision studies, when a node is removed from a graph, its connected edges disappear simultaneously.

We summarize the findings in the transfer tasks of citation graphs (i.e., C→A and D→A) first. It can be seen in Figure 8, the Micro-F1 score increases gradually with the percentage of nodes selected. As shown in Figure 4, in the citation graphs, most neighboring nodes are with the same class as the central node, which is in line with the homophily assumption shared by many GNN models. An increased number of neighboring nodes in the same class could support the GNN layers to learn more informative node representations. In the multimodal distribution of node representations, each mode corresponds

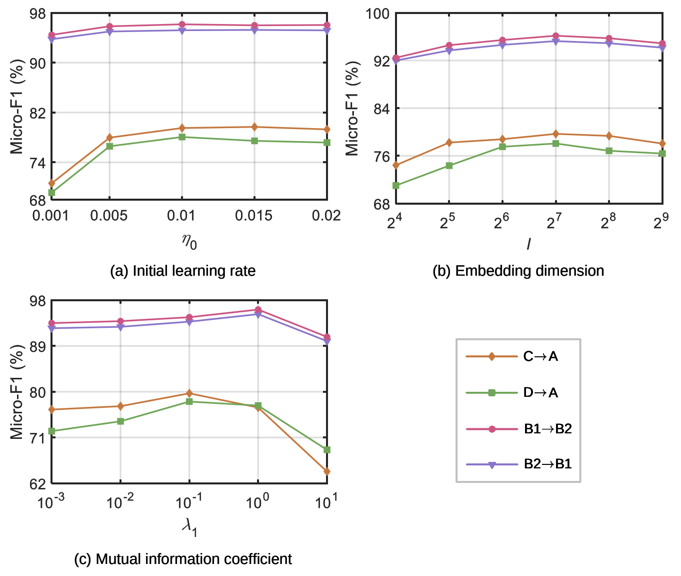


Fig. 9. Impact of hyperparameters.

to one class of nodes (see Section III-A). If there are more nodes of the same class to characterize the features of a mode, the distribution alignment would possibly be improved.

In Figure 8, the downsampling of target graph leads to a steeper performance drop. Since the model performance in target graph is reported, when a part of target nodes is removed, the downsampled target graph becomes less informative, resulting in a large drop in performance. In the case that the source nodes are removed partially, the complete and informative target graph prevents the Micro-F1 score from dropping greatly.

Similar observations are also found in the transfer tasks of social graphs (i.e., B1→B2 and B2→B1). However, the Micro-F1 score only slightly increases when R_n is greater than 40%. Since the social graphs have higher average degrees (see Table II), when R_n reaches 40%, the nodes would already have abundant neighborhood information to assist representation learning. Moreover, as the homophily assumption is not strictly held in the social graphs (see Figure 4), the increase of neighboring nodes might introduce certain noise. In addition, the rich node attributes in social graphs are informative for representation learning. Therefore, with only a few nodes left in the target graph (i.e., $R_n = 10\%$), the performance drops in social graphs are much smaller than those in citation graphs.

F. Hyperparameter Sensitivity Analysis

In this section, we investigate the performance of AdaGIn with regard to three hyperparameters, i.e., initial learning rate η_0, embedding dimension d, and mutual information coefficient λ_1. The goal is to shed light on the hyperparameter configuration. When investigating one hyperparameter, the others are fixed to their default values introduced in Section IV-A3. Figure 9 displays the Micro-F1 scores of transfer tasks in the citation and social graphs. In general, the performance of AdaGIn is more sensitive to these hyperparameters when evaluated in the citation graphs. In B1→B2 and B2→B1, the Micro-F1 scores become stable when the initial learning rate is larger than 0.005. In C→A and D→A, the Micro-F1 score first increases with the initial learning rate, and then decreases after reaching a maximum value. Similar trends can be seen in the other two hyperparameters under all transfer tasks. With embedding dimension set as 128, the highest Micro-F1 score is observed in each transfer task. The optimal mutual information coefficients are 0.1 and 1.0 for transfer tasks in the citation and social graphs, respectively.

V. Conclusion

A novel GNN model has been proposed to address the cross-graph node classification problem. This method enables mutual information maximization in the cross-graph learning, which encourages node representations to capture the global structural information. Conditional adversarial networks are employed to align the multimodal distributions of node representations. Experimental results demonstrate that the proposed method is superior to the state-of-the-art approaches in the benchmark transfer tasks. As a spatial GNN model, the proposed method is promising for many high-impact applications in the real-world large-scale networks, such as protein-protein interaction networks and recommender systems.

References

[1] S. Bhagat, G. Cormode, and S. Muthukrishnan, “Node classification in social networks,” in *Social Network Data Analytics*, C. C. Aggarwal, Ed. Boston, MA: Springer US, 2011, pp. 115–148.
[2] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 1025–1035.
[3] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *Proceedings of International Conference on Learning Representations*, Toulon, France, 2017.
[4] P. Cui, X. Wang, J. Pei, and W. Zhu, “A survey on network embedding,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, May 2019.
[5] P. Goyal and E. Ferrara, “Graph embedding techniques, applications, and performance: A survey,” *Knowledge-Based Systems*, vol. 151, pp. 78–94, Jul. 2018.
[6] X. Shen, Q. Dai, S. Mao, F.-L. Chung, and K.-S. Choi, “Network together: Node classification via cross-network deep network embedding,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 5, pp. 1935–1948, May 2021.
[7] X. Shen, Q. Dai, F.-l. Chung, W. Lu, and K.-S. Choi, “Adversarial deep network embedding for cross-network node classification,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, pp. 2991–2999, Apr. 2020, number: 03.
[8] Q. Dai, X.-M. Wu, J. Xiao, X. Shen, and D. Wang, “Graph transfer learning via adversarial domain adaptation with graph convolution,” *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2022.
[9] M. Wu, S. Pan, C. Zhou, X. Chang, and X. Zhu, “Unsupervised domain adaptive graph convolutional networks,” in *Proceedings of The Web Conference 2020*, ser. WWW ’20. New York, NY, USA: Association for Computing Machinery, Apr. 2020, pp. 1457–1467.
[10] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[11] M. Long, G. Ding, J. Wang, J. Sun, Y. Guo, and P. S. Yu, “Transfer sparse coding for robust image representation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 407–414.
[12] M. Long, Y. Cao, J. Wang, and M. I. Jordan, “Learning transferable features with deep adaptation networks,” in *Proceedings of the 32nd International Conference on Machine Learning*, ser. ICML’15. Lille, France: JMLR.org, Jul. 2015, pp. 97–105.

[13] H. Yang, H. He, W. Zhang, and X. Cao, "FedSteg: A federated transfer learning framework for secure image steganalysis," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1084–1094, Apr. 2021.

[14] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Co-clustering based classification for out-of-domain documents," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '07. New York, NY, USA: Association for Computing Machinery, Aug. 2007, pp. 210–219.

[15] J. Jiang and C. Zhai, "Instance weighting for domain adaptation in NLP," in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, 2007.

[16] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[17] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '18. New York, NY, USA: Association for Computing Machinery, Jul. 2018, pp. 1320–1329.

[18] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, New Orleans, USA, 2018, pp. 3538–3545.

[19] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, Dec. 2014, pp. 2177–2185.

[20] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., Dec. 2018, pp. 1647–1657.

[21] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.

[22] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proceedings of International Conference on Learning Representations*, 2019.

[23] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *Proceedings of International Conference on Learning Representations*, 2019.

[24] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv:1412.3474 [cs]*, Dec. 2014.

[25] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: Transfer learning with deep autoencoders," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. Buenos Aires, Argentina: AAAI Press, Jul. 2015, pp. 4119–4125.

[26] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, Jan. 2016.

[27] J. Shen, Y. Qu, W. Zhang, and Y. Yu, "Wasserstein distance guided representation learning for domain adaptation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, number: 1.

[28] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. Sydney, NSW, Australia: JMLR.org, Aug. 2017, pp. 214–223.

[29] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv:1411.1784 [cs, stat]*, Nov. 2014.

[30] J. Li, E. Chen, Z. Ding, L. Zhu, K. Lu, and H. T. Shen, "Maximum density divergence for domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3918–3930, 2021.

[31] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang, "Generalization and equilibrium in generative adversarial nets (GANs)," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. Sydney, NSW, Australia: JMLR.org, Aug. 2017, pp. 224–232.

[32] J. Li, Z. Du, L. Zhu, Z. Ding, K. Lu, and H. T. Shen, "Divergence-agnostic unsupervised domain adaptation by adversarial attacks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[33] Y. Zhang, S. Miao, and R. Liao, "Structural domain adaptation with latent graph alignment," in *Proceedings of the 25th IEEE International Conference on Image Processing*, 2018, pp. 3753–3757.

[34] D. Das and C. George Lee, "Unsupervised domain adaptation using regularized hyper-graph matching," in *Proceedings of the 25th IEEE International Conference on Image Processing*, 2018, pp. 3758–3762.

[35] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv:1709.05584 [cs]*, Apr. 2018.

[36] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2014, pp. 701–710.

[37] S. Liu, B. Wang, L. T. Yang, and P. Yu, "HNF: Hybrid neural filtering based on centrality-aware random walk for personalized recommendation," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2021.

[38] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, Oct. 2015, pp. 891–900.

[39] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225–1234.

[40] Z. Zhang, H. Yang, J. Bu, S. Zhou, P. Yu, J. Zhang, M. Ester, and C. Wang, "ANRL: Attributed network representation learning via deep neural networks," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, ser. IJCAI'18. Stockholm, Sweden: AAAI Press, Jul. 2018, pp. 3155–3161.

[41] J. Xiao, Q. Dai, X. Xie, J. Lam, and K.-W. Kwok, "Adversarially regularized graph attention networks for inductive learning on partially labeled graphs," *arXiv:2106.03393 [cs]*, Jun. 2021.

[42] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proceedings of International Conference on Learning Representations*, 2018.

[43] Z. Zhang, Y. Li, H. Dong, H. Gao, Y. Jin, and W. Wang, "Spectral-based directed graph network for malware detection," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 957–970, Apr. 2021.

[44] Y. Zhu, Y. Xu, Q. Liu, and S. Wu, "An empirical study of graph contrastive learning," *arXiv:2109.01116 [cs]*, Oct. 2021.

[45] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *Proceedings of International Conference on Learning Representations*, 2019.

[46] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *Proceedings of International Conference on Learning Representations*, 2020.

[47] J. Ni, S. Chang, X. Liu, W. Cheng, H. Chen, D. Xu, and X. Zhang, "Co-regularized deep multi-network embedding," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, Apr. 2018, pp. 469–478.

[48] L. Xu, X. Wei, J. Cao, and P. S. Yu, "Embedding of Embedding (EOE): Joint embedding for coupled heterogeneous networks," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '17. New York, NY, USA: Association for Computing Machinery, Feb. 2017, pp. 741–749.

[49] M. Heimann, H. Shen, T. Safavi, and D. Koutra, "REGAL: Representation learning-based graph alignment," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM '18. New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 117–126.

[50] L. Liu, W. K. Cheung, X. Li, and L. Liao, "Aligning users across social networks using network embedding," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI'16. New York, New York, USA: AAAI Press, Jul. 2016, pp. 1774–1780.

[51] M. Pilancı and E. Vural, "Domain adaptation on graphs by learning aligned graph bases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 2, pp. 587–600, 2022.

[52] M. Fang, J. Yin, and X. Zhu, "Transfer learning across networks for collective classification," in *Proceedings of the 13th IEEE International Conference on Data Mining*, Dec. 2013, pp. 161–170, iSSN: 2374-8486.

[53] C. Ding, T. Li, W. Peng, and H. Park, "Orthogonal nonnegative matrix tri-factorizations for clustering," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06. New York, NY, USA: Association for Computing Machinery, Aug. 2006, pp. 126–135.

[54] B. Schölkopf, J. Platt, and T. Hofmann, "A kernel method for the two-sample-problem," in *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*. MIT Press, 2007, pp. 513–520.

[55] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "GCC: Graph contrastive coding for graph neural network pre-training," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 1150–1160.

[56] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 4, pp. 801–814, Apr. 2019.

[57] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, no. 1, pp. 151–175, May 2010.

[58] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, Dec. 2017.

[59] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "ArnetMiner: Extraction and mining of academic social networks," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08. New York, NY, USA: Association for Computing Machinery, Aug. 2008, pp. 990–998.

[60] J. Li, X. Hu, J. Tang, and H. Liu, "Unsupervised streaming feature selection in social media," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, ser. CIKM '15. New York, NY, USA: Association for Computing Machinery, Oct. 2015, pp. 1041–1050.

[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of International Conference on Learning Representations*, Jan. 2015.

[62] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[63] J. Liu, M. Jing, J. Li, K. Lu, and H. T. Shen, "Open set domain adaptation via joint alignment and category separation," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2021.

Jiaren Xiao received the B.Eng. degree in mechanical engineering from Xi'an Jiaotong University, Xi'an, China, in 2015, and the M.Eng. degree in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2018. He is now a Ph.D. candidate at the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong. His research interests include representation learning, transfer learning, and adversarial learning for graph-structured data.

Quanyu Dai (Member, IEEE) is currently a researcher at Huawei Noah's Ark Lab. He received the Ph.D. degree at The Hong Kong Polytechnic University. His research interests include graph-based algorithms and recommender systems. He has publications appeared in the top-tier journals and conferences, such as TKDE, TNNLS, IJCAI, AAAI, WWW and KDD.

Xiaochen Xie (Member, IEEE) received the B.E. degree in automation and the M.E. degree in control science and engineering from Harbin Institute of Technology, Harbin, China, in 2012 and 2014, respectively, and the Ph.D. degree in control engineering from The University of Hong Kong, Hong Kong, in 2018. Her research interests include robust control and filtering, periodic systems, switched systems, intelligent systems and process monitoring.

Qi Dou (Member, IEEE) Dr. Dou is an Assistant Professor at the Department of Computer Science and Engineering, Chinese University of Hong Kong (CUHK). She received B. Eng in Biomedical Engineering at Beihang University in 2014, PhD in Computer Science at CUHK in 2018, and worked as postdoctoral researcher in Department of Computing at Imperial College London during 2018 to 2020. She has focused research interest at the interdisciplinary field of artificial intelligence for medical scenarios, with expertise in medical image analysis and robotic surgery perception.

Ka-Wai Kwok (Senior Member, IEEE) received the B.Eng. and M.Phil. degrees in automation and computer-aided engineering from the Chinese University of Hong Kong, Hong Kong, in 2003 and 2005, respectively, and the Ph.D. degree in Computing from the Hamlyn Center for Robotic Surgery, Department of Computing, Imperial College London, London, U.K., in 2012.

He is currently an Associate Professor with the Department of Mechanical Engineering, University of Hong Kong (HKU), Hong Kong. Prior to joining HKU in 2014, he worked as a Postdoctoral Fellow with Imperial College London in 2012 for surgical robotics research. In 2013, he was awarded the Croucher Foundation Fellowship, which supported his research jointly supervised by advisors from the University of Georgia, Athens, GA, USA, and Brigham and Women's Hospital, Harvard Medical School, Boston, MA, USA.

His research focuses on surgical robotics, intra-operative image processing, and their uses of intelligent and control systems. To date, Dr. Kwok has co-authored 123 publications with >50 clinical fellows and >90 engineering scientists, and 6 out of 14 invention patents licensed/transferred to industrial partners in support for their commercialization. His multidisciplinary work has been recognized by >10 international publication awards, mostly under IEEE, particularly in the largest flagship conferences of robotics: e.g. ICRA Best Conference Paper Award in 2018, and IROS Toshio Fukuda Young Professional Award in 2020. He also serves as an Associate Editor for Journal of Systems and Control Engineering (JSCE), IEEE Robotics and Automation Magazine (RAM), and Annals of Biomedical Engineering (ABME). He is the principal investigator of research group for Interventional Robotic and Imaging Systems (IRIS), HKU.

James Lam (Fellow, IEEE) received a BSc (1st Hons.) degree in Mechanical Engineering from the University of Manchester, and was awarded the Ashbury Scholarship, the A.H. Gibson Prize, and the H. Wright Baker Prize for his academic performance. He obtained the MPhil and PhD degrees from the University of Cambridge. He is a Croucher Scholar, Croucher Fellow, and Distinguished Visiting Fellow of the Royal Academy of Engineering, and Cheung Kong Chair Professor. Prior to joining the University of Hong Kong in 1993 where he is now Chair Professor of Control Engineering, he was a faculty member at the City University of Hong Kong and the University of Melbourne.

Professor Lam is a Chartered Mathematician (CMath), Chartered Scientist (CSci), Chartered Engineer (CEng), Fellow of Institute of Electrical and Electronic Engineers (FIEEE), Fellow of Institution of Engineering and Technology (FIET), Fellow of Institute of Mathematics and Its Applications (FIMA), Fellow of Institution of Mechanical Engineers (FIMechE), and Fellow of Hong Kong Institution of Engineers (FHKIE).

He is Editor-in-Chief of IET Control Theory and Applications, Journal of The Franklin Institute and Proc. IMechE Part I: Journal of Systems and Control Engineering, Subject Editor of Journal of Sound and Vibration, Editor of Asian Journal of Control, Senior Editor of Cogent Engineering, Section Editor of IET Journal of Engineering, Consulting Editor of International Journal of Systems Science, Associate Editor of Automatica and Multidimensional Systems and Signal Processing.

His recent research interests include multi-agent systems, positive systems, Boolean networks, networked control systems, and vibration control. He is a Highly Cited Researcher in Engineering (2014 to 2021), Computer Science (2015), and Cross-Fields (2021).