Knowledge and Information Systems
https://doi.org/10.1007/s10115-023-01942-2

REGULAR PAPER

Node classification across networks via category-level domain adaptive network embedding

Boshen Shi[1,2] · Yongqing Wang[1] · Jiangli Shao[1,2] · Huawei Shen[1] · Yangyang Li[3,4] · Xueqi Cheng[1]

Received: 8 December 2022 / Revised: 27 May 2023 / Accepted: 10 July 2023

Abstract
To improve the performance of classifying nodes on unlabeled or scarcely-labeled networks, the task of node classification across networks is proposed for transferring knowledge from similar networks with rich labels. As data distribution shift exists across networks, domain adaptive network embedding is proposed to overcome such challenge by learning network-invariant and discriminative node embeddings, in which domain adaptation technique is applied to network embedding for reducing domain discrepancy. However, existing works merely discuss category-level domain discrepancy which is crucial to better adaptation and classification. In this paper, we propose category-level domain adaptive network embedding. The key idea is minimizing intra-class domain discrepancy and maximizing inter-class domain discrepancy between source and target networks simultaneously. To further enhance classification performance on target network, we reduce embedding variation inside each class and enlarge it between different classes. Graph attention network is adopted for learning network embeddings. In addition, a novel pseudo-labeling strategy for target network is developed to better compute category-level information. Theoretical analysis guarantees the effectiveness of our model. Furthermore, extensive experiments on real-world datasets show that our model achieves the state-of-art performance, in particular, outperforming existing domain adaptive network embedding models by up to 32%.

Keywords Node classification across networks · Graph neural networks · Domain adaptation · Transfer learning

✉ Boshen Shi
shiboshen19s@ict.ac.cn

✉ Yongqing Wang
wangyongqing@ict.ac.cn

1 Data Intelligence System Research Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

2 School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China

3 National Engineering Research Center for Risk Perception and Prevention (NEL-RPP), CAEIT, Beijing 100041, China

4 Academy of Cyber, Beijing 100085, China

Published online: 30 July 2023

1 Introduction

Node classification is a critical yet challenging task, primarily due to the difficulties and expense involved in acquiring abundant high-quality labels. Contemporary network embedding techniques, such as graph neural networks [1–3], typically depend on a substantial quantity of labels to achieve satisfactory classification performance (e.g., labeling ratio equals to 53.7% in ogbn-arxiv dataset [4]). This dependency could result in suboptimal performance in scenarios with unlabeled or sparsely-labeled networks. To mitigate these issues, we propose the task of node classification across networks. Given a target network where nodes need to be classified, we can select several source networks from similar domains to impart rich label information, thus transferring valuable knowledge to the target network. In this way, we can potentially elevate the classification performance on the target network. Consider the example of social bot detection. Given that the acquirement of ground-truth bots often necessitates extensive domain knowledge and the involvement of human experts, an alternative approach may lie in constructing a cross-network node classification task that incorporates additional social networks furnished with abundant bot labels [5, 6].

Node classification across networks presents novel challenges to existing network embedding methods, which are tailored for single networks. A key hurdle is the distribution shift in node embeddings across networks, which inhibits the inductive application of models trained on source networks to target networks [7]. Consequently, domain adaptive network embedding treats each network as a unique domain, implementing domain adaptation techniques on network embeddings to reduce domain discrepancy and foster the learning of discriminative node embeddings. It is a transductive method which learns node embeddings jointly on source and target networks during training. Based on the metric used to measure domain discrepancy, it can be categorized into two classes. One research direction explicitly models domain discrepancy through Euclidean Distance and minimizes such discrepancy during training [8, 9]. Conversely, the other line of research employs adversarial training to minimize discrepancy measurement, including but not limited to, Jensen–Shannon Divergence [10, 11] and Wasserstein Distance [12, 13]. Despite these advancements, the majority of existing methods mainly minimize global domain discrepancy calculated on the overall embeddings across source and target networks, leaving domain discrepancy at category-level largely ignored. Therefore, even though the overall node embeddings may overlap in the embedding space, there could be negative instances where nodes from the same class across networks are incorrectly separated, or nodes belonging to different classes across networks are incorrectly overlapped. Figure 1 illustrates this phenomenon.

Therefore, better embeddings for node classification across networks should satisfy two conditions. First, embeddings in the same class are approximately overlapped, regardless of their originating network (intra-class). Second, embeddings in different classes are separated across networks (inter-class). As the existing methods ignore fine-grained category-level information, which is defined as node embedding distribution conditioned on label distribution, they leave intra-class domain discrepancy relatively large and inter-class domain discrepancy relatively small. We mitigate such problem by explicitly modeling intra-class and inter-class domain discrepancy followed by minimizing and maximizing them, respectively. Moreover, as target network labeling information is required to compute category-level domain discrepancy, it is essential to design a proper pseudo-labeling strategy for unlabeled target network. We follow two principles. First, intra-class domain discrepancy should be further reduced after pseudo-labeling. Second, neighboring nodes tend to have the same

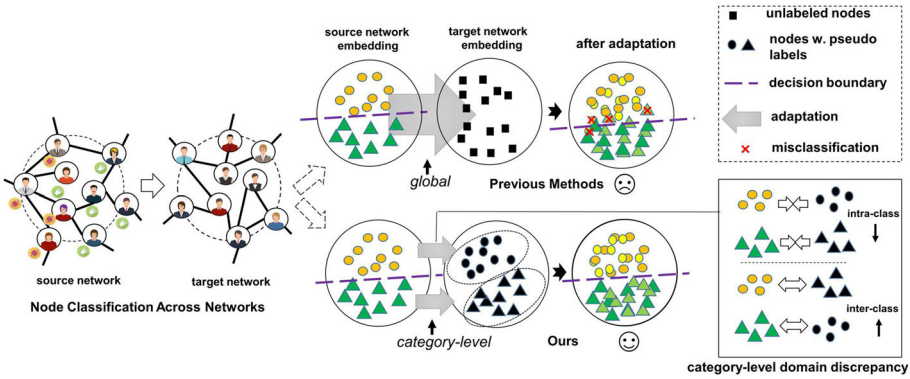

Fig. 1 Previous methods: "Despite the minimization of global domain discrepancy, the incorrect overlap of node embeddings can still occur, potentially leading to misclassification. Ours: We alleviate misclassification by exploiting category-level information

label. Thus, both embedding distribution of source and target networks and complex network topology are taken into account to generate pseudo-labels.

In this work, we explore the novel problem of node classification across networks by proposing **category-level** domain adaptive network embedding. Generally, we consider category-level information from two aspects. First, we minimize intra-class domain discrepancy and maximize inter-class domain discrepancy between source and target networks. Considering such category-level domain discrepancy facilitates the transfer of label knowledge from source networks, thereby enhancing adaptation performance. Specifically, we utilize category-level Maximum Mean Discrepancy(MMD) to compute intra-class and inter-class domain discrepancy between network embeddings. As target networks are usually unlabeled, we design a pseudo-labeling strategy in order to obtain category-level information. In particular, target nodes are firstly labeled by their nearest source class centers in the embedding space. To leverage network homophily (i.e., neighboring nodes tend to have the same label), such pseudo-labels are further propagated on target networks for smoothing. Subsequently, we further make node embeddings in target networks more discriminative by reducing embedding variation inside each class and enlarging it between different classes. The integration of category-level information within target network facilitates enhanced classification performance. More precisely, as the proposed pseudo-labeling strategy prefers to assign nearby nodes in the embedding space with the same labels, it contributes to reducing embedding variation inside each class on target networks. Furthermore, we apply entropy loss to node predictions on target networks, with the aim of enlarging embedding variation between different classes. Finally, we establish the effectiveness of the proposed model theoretically via generalization bound and conduct various experiments to verify its efficacy in real-world scenarios.

To summarize, our contributions are listed as follows:

1. We propose **category-level** domain adaptive network embedding by minimizing intra-class domain discrepancy and maximizing inter-class domain discrepancy between source and target networks. To further enhance classification performance, we also reduce embedding variation inside each class and enlarge it between different classes on target network.

2. We propose a novel pseudo-labeling strategy for domain adaptive network embedding, in which both embedding distribution and network topology are exploited.
3. Both theoretical analysis and extensive experiments on real-world datasets verify the effectiveness of the proposed model.

2 Related work

2.1 Network embedding

Network embedding aims to transform networks into continuous embedding space, preserving both network topology and node content information. The conventional methods include matrix factorization [14], random walk [15, 16] and graph neural networks(GNN) [1–3]. Most of them concentrate on a single network, leaving the scenarios of multiple networks under explored. Recently research has studied the problem of network embedding across multiple networks by positing that at least a portion of nodes are shared across networks [17–19]. In such a case, the embeddings of shared nodes should be consistent across networks. However, domain-adaptive network embedding methods do not make this assumption about shared nodes and, as a result, are applicable in a broader range of scenarios.

2.2 Domain adaptation

Domain adaptation refers to the process that adapting source domains with sufficient labels to target domain with plenty of unlabeled data by minimizing their domain discrepancy [7, 20]. Recently, domain adaptation models utilizing deep neural networks have succeeded in addressing cross-domain classification tasks in many fields including natural language processing (NLP) [21–23] and computer vision (CV) [24–26]. Two main approaches are identified in the literature [20]. **Discrepancy-based** models measure domain discrepancy by distance metric and minimize such distance to make learned features domain-invariant. Metric includes Maximum Mean Discrepancy(MMD) [24, 27–30], covariance [31], central moment discrepancy [32], Kullback–Leibler Divergence [33] and so on. Besides, **Adversarial-based** models are based on an adversarial loss which encourages samples from different domains to be non-discriminative with respect to domain labels [34]. Domain adaptation cannot be directly applied to network structured data because it assumes samples are independent and identically distributed. Thus, it is necessary to study how to apply domain adaptation technique to network embedding for solving node classification across networks.

2.3 Domain adaptive network embedding

To alleviate the existing distribution shift between source and target networks, domain adaptive network embedding adapts node embeddings in the target domain by capturing the domain discrepancy and exploiting the knowledge from source domain while preserving the intrinsic structure and properties of the target domain. It is worth mentioning that we have noticed a parallel line of research in the field of graph out-of-distribution generalization [39–42]. It is a problem of generalizing a model to predict on node sets that belong to unknown distributions. To summarize, domain adaptive network embedding is a technique that can be used to address the above-mentioned problem and enhance the performance of a model on target domain data, whereas graph out-of-distribution generalization is a problem or task.

Table 1 Summary of domain adaptive network embedding

Discrepancy measurement	Model	Encoder	Category-level
JSD	ACDNE [35]	AutoEncoder	No
	UDAGCN [10]	Dual-GNN	No
	ASN [11]	Dual-GNN	No
	GRADE [36]	GNN	No
Wasserstein distance	AdaGCN [13]	GNN	No
	SpecReg [12]	GNN	No
χ^2 divergence	DANE [37]	GNN	No
Euclidean distance	CDNE [9]	Stacked AutoEncoder	Yes
MMD	GNA [38]	Linear Transformation	No
Mix	GraphAE [8]	GNN	No

Mix means the model could adopt multiple measurements

Although both methods contribute to enhancing the generalization ability of models, they focus on different tasks and scenarios.

Among the existing studies, most models only focus on global domain discrepancy, leaving category-level information ignored. Typically, adversarial training [34, 43, 44] is adopted to learn global domain discrepancy. Among them, a group of models adopts Jensen–Shannon Divergence (JSD) as discrepancy measurement [8, 10, 11, 35, 35, 36]. Particularly, UDAGCN [10] and ASN [11] implement dual GNN structure to capture both local and high-order network topology. ASN further separates domain-private and domain-shared features by designing a private encoder for each network and a shared encoder across networks. GRADE [36] further considers topology differences between networks by combining discrepancy measurement with Weisfeiler–Lehman graph isomorphism test. In addition, other models adopt Wasserstein Distance [12, 13] as discrepancy measurement. Among them, SpecReg [12] improves over AdaGCN [13] by adding spectral regularizations to smooth graph signals for better adaptation. Apart from the models discussed above, DANE [37] takes Pearson χ^2 divergence as measurement to avoid the instability of adversarial learning. GNA [38] incorporates graph kernels with MMD for better computing domain discrepancy between networks with temporal information. To sum up, Table 1 offers a summary of domain adaptive network embedding methods.

For models considering category-level information, CDNE [9] utilizes Euclidean Distance to measure global domain discrepancy together with intra-class domain discrepancy and jointly minimizes them. However, it ignores inter-class discrepancy. Furthermore, Euclidean Distance is inaccurate for computing distribution shift. Moreover, its pseudo-labeling strategy does not explicitly take network topology into account, which degrades the quality of pseudo-labels.

3 Problem statement

Generally, a network G can be represented by $G = (A, X, Y)$ and V, E denote node set and edge set, respectively. Network topology is represented by adjacency matrix A. The $X \in \mathbb{R}^{N \times M}$ denotes node attribute matrix, where $N = |V|$ and M is the dimension of input

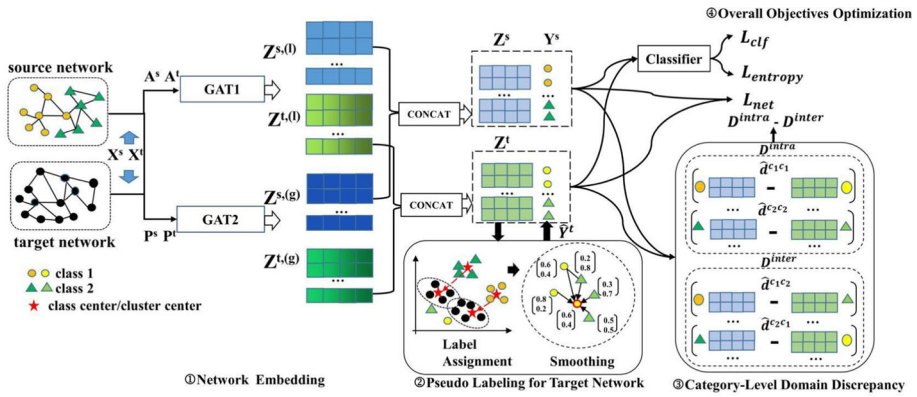

Fig. 2 Framework of the proposed model, which comprises four integral components: (1) A dual GNN network embedding module (2) A pseudo-labeling module (3) A domain discrepancy computation module, and (4) An overall objectives optimization module

features. The $Y \in \mathbb{R}^{N \times C}$ is 0–1 label matrix, and C is the number of labels. $Y_{i,c} = 1$ denotes node i is associated with label c.

In this paper, we study node classification problem across networks. We assume there is one source domain with source network G^s and one target domain with target network G^t. We focus on a more challenging task that G^t is totally unlabeled. We also follow two basic assumptions in domain adaptation [7]. First, the node features X^s and X^t are sampled from the same feature space. Second, two networks share the same set of labels. Thus, G^s and G^t can be represented as $G^s = (A^s, X^s, Y^s)$ and $G^t = (A^t, X^t)$, respectively. Given G^s and G^t, domain adaptive network embedding aims to learn an embedding function f_{emb} which learns node embeddings Z^s and Z^t in $\mathbb{R}^h$ and a classifier f_c which predicts target labels Y^t based on Z^s, Z^t and Y^s.

4 Methods

The proposed method can be explained in four integral parts, i.e., network embedding, pseudo-labeling, category-level domain discrepancy measurement and objective optimization. The model framework is illustrated in Fig. 2.

4.1 Network embedding

The network embedding function f_{emb} incorporates dual GNN structure proposed by [10] to capture both local and high-order network topology, which is implemented with Graph Attention Network(GAT) [2]. Compared with GCN, GAT introduces attention-based aggregation layer to capture different importance of neighborhood for a central node during forward propagation. As previous studies [10, 11] have empirically proved, incorporating high-order network information could improve adaptation performance. Besides, f_{emb} is typically shared between source and target networks for better adaptation [37].

Specifically, f_{emb} utilizes two GATs to compute embedding Z on each network. One GAT aggregates 1-hop neighbors from adjacency matrix at each layer. Given X and A, node embeddings computed by adjacency matrix are denoted as $Z^{(l)} = GAT_1(X, A)$, where

$Z^{(l)} \in \mathbb{R}^{N\times h}$. The other GAT aggregates high order neighbors from Positive Pointwise Mutual Information (PPMI) matrix [10] at each layer. PPMI matrix P measures structural proximity between nodes within K steps in a network and is computed through random walk. A higher positive value of $P_{i,j}$ indicates that vertex i has a strong connection with j within K steps in network. Thus, given X and P, node embeddings computed by PPMI matrix are denoted as $Z^{(g)} = GAT_2(X, P)$, where $Z^{(g)} \in \mathbb{R}^{N\times h}$. Finally, the node embeddings Z are obtained by concatenating $Z^{(l)}$ and $Z^{(g)}$. Concatenation coefficients w_1 and w_2 are then computed by applying a linear transformation matrix $W : \mathbb{R}^{2h} \to \mathbb{R}^2$:

$$w_i = \frac{\exp\left(W\left(Z^{(l)} \| Z^{(g)}\right)\right)_i}{\sum_{j=1,2}\exp\left(W\left(Z^{(l)} \| Z^{(g)}\right)\right)_j}$$

where $\|$ denotes concatenation operator and Z is calculated as:

$$Z = w_1 Z^{(l)} + w_2 Z^{(g)} \tag{1}$$

4.2 Pseudo-labeling

As we reduce intra-class domain discrepancy during training, nodes with the same label across networks should ideally be proximate in the embedding space. This process inspires us that we could initially assign target nodes with labels of their nearest source class centers, thereby enhancing the accuracy of the pseudo-labels generated. The increased accuracy will in turn benefit subsequent adaptation and classification tasks.

Specifically, we first compute C source class centers $O^{s,c}$ on source node embeddings: $O^{s,c} = \sum_i Y^s_{i,c} Z^s_i / \sum_j Y^s_{j,c}$, where $Y^s_{i,c} = 1$ if node i has label c and $Y^s_{i,c} = 0$ otherwise. After that we apply Gaussian Mixture Model(GMM) to target node embeddings for computing K clusters, where $K = C$. We use Φ to indicate cluster probability matrix obtained from GMM, where $\Phi_{i,k}$ refers to the probability of target node i belonging to the kth cluster. The $O^{t,k}$ denotes center of the kth target cluster, and $O^{t,k} = \sum_i \Phi_{i,k} Z^t_i / \sum_j \Phi_{j,k}$. Motivated by the above idea, we assign pseudo label c to the kth target cluster by choosing source class center $O^{s,c}$ which is the nearest to $O^{t,k}$:

$$c = \underset{c'}{\operatorname{argmin}}\, Dist(O^{t,k}, O^{s,c'}) \tag{2}$$

where cosine dissimilarity $Dist(x, y) = (1 - \langle x, y\rangle / \|x\|\|y\|)/2$ is adopted and $\langle\cdot,\cdot\rangle$ denotes inner product.

After label assignment, pseudo label probability matrix $\hat{Y}^t$ could be induced from Φ accordingly. For example, if $\Phi_i = (0.2, 0.5, 0.3)$ for target node i and the three clusters are labeled as $\{2, 0, 1\}$ by Eq. 2, respectively, then $\hat{Y}^t_i = (0.5, 0.3, 0.2)$. As label assignment may ignore network homophily property, $\hat{Y}^t$ is noisy. Thus, we iteratively propagate $\hat{Y}^t$ on target network until convergence [45] for smoothing:

$$\hat{Y}^t = r * \hat{Y}^t + (1 - r) * \tilde{A}^t \hat{Y}^t \tag{3}$$

where $\tilde{A}^t = {D^t}^{-\frac{1}{2}} A^t {D^t}^{-\frac{1}{2}}$ and D^t denotes node degree matrix.

The proposed pseudo-labeling strategy has two significant advantages. Firstly, by assigning labels of the nearest source class centers to target nodes, it ensures that nodes within the same class are in close proximity to each other across networks. This helps to decrease intra-class domain discrepancy. Secondly, the strategy prefers to assign labels to nearby nodes in the embedding space, helping to minimize variation within each class on target networks.

4.3 Category-level domain discrepancy

After pseudo-labeling, we compute both intra-class domain discrepancy and inter-class domain discrepancy based on MMD.

Given samples drawn from marginal distributions P and Q, respectively, MMD performs a kernel two-sample test to determine whether to accept the null hypothesis $P = Q$ or not [46]. The basic idea behind MMD is that if the generating distributions are identical, all the statistics are the same as well. Formally, MMD defines the difference between two distributions with their mean embeddings in the reproducing kernel Hilbert space (RKHS):

$$d_{\mathcal{H}}(P, Q) = \sup_{\phi}(\mathbb{E}_p(\phi(Z^s)) - \mathbb{E}_q(\phi(Z^t)))_{\mathcal{H}} \tag{4}$$

where ϕ maps original embeddings to RKHS $\mathcal{H}$ with a characteristic kernel k. Kernel k means $k(Z^s, Z^t) = \langle \phi(Z^s), \phi(Z^t) \rangle$. In practice, an estimate of 4 compares the square distance between the empirical kernel mean embeddings [46].

Theoretically, intra-class domain discrepancy of class c refers to MMD between node embeddings from two networks within the same class c, denoted as $d_{\mathcal{H}}^{cc}$. To compute the empirical value of $d_{\mathcal{H}}^{cc}$, i.e., $\hat{d}_{\mathcal{H}}^{cc}$, we need to compute the square distance between the weighted empirical kernel mean embeddings. We explicitly model the probabilities of nodes i, j belonging to each class c on both source and target networks, which are denoted as $w_i^{s,c}$ and $w_j^{t,c}$, respectively. $\sum_{i=1}^{N^s} w_i^{s,c} = \sum_{j=1}^{N^t} w_j^{t,c} = 1$. For labeled source network, probability $w_i^{s,c}$ is computed as: $w_i^{s,c} = Y_{i,c}^s / \sum_k Y_{k,c}^s$. For target network, $w_j^{t,c} = \hat{Y}^t{}_{j,c} / \sum_k \hat{Y}^t{}_{k,c}$. As label prior probability is not always equal between networks, modeling such terms is necessary for more accurate computation. Thus, $\hat{d}_{\mathcal{H}}^{cc}$ is computed as:

$$\begin{aligned}\hat{d}_{\mathcal{H}}^{cc}(P, Q) &= \| \sum_{Z_i \in Z^s, y_i = c} w_i^{s,c} \phi(Z_i) - \sum_{Z_j \in Z^t, \hat{y}_j = c} w_j^{t,c} \phi(Z_j) \|_{\mathcal{H}}^2 \\ &= \sum_{i=1}^{N^{s,c}} \sum_{j=1}^{N^{s,c}} w_i^{s,c} w_j^{s,c} k_{i,j} + \sum_{i=1}^{N^{t,c}} \sum_{j=1}^{N^{t,c}} w_i^{t,c} w_j^{t,c} k_{i,j} - \sum_{i=1}^{N^{s,c}} \sum_{j=1}^{N^{t,c}} w_i^{s,c} w_j^{t,c} k_{i,j}\end{aligned}$$

where $k_{i,j} = k(Z_i, Z_j)$ and $N^{s,c}$, $N^{t,c}$ denote the number of nodes belonging to class c on two networks. In conclusion, as there are C classes in all, the intra-class domain discrepancy D^{intra} is computed as:

$$D^{\text{intra}} = \frac{1}{C} \sum_c \hat{d}_{\mathcal{H}}^{cc} \tag{5}$$

Inter-class domain discrepancy of classes c and c' refers to MMD between node embeddings from two networks in different classes, denoted as $d_{\mathcal{H}}^{cc'}$. Maximizing inter-class domain discrepancy prevents embeddings coming from two networks in different classes from being overlapped and therefore, makes them more discriminative. Similarly, we empirically compute $\hat{d}_{\mathcal{H}}^{cc'}$ as:

$$\hat{d}_{\mathcal{H}}^{cc'}(P, Q) = \| \sum_{\substack{Z_i \in Z^s \\ y_i = c}} w_i^{s,c} \phi(Z_i) - \sum_{\substack{Z_j \in Z^t \\ \hat{y}_j = c'}} w_j^{t,c'} \phi(Z_j) \|_{\mathcal{H}}^2$$

As there are C classes in all, inter-class domain discrepancy D^{inter} is computed as:

$$D^{\text{inter}} = \frac{1}{C(C-1)} \sum_{c} \sum_{\substack{c' \\ c' \neq c}} \hat{d}_{\mathcal{H}}^{cc'} \tag{6}$$

4.4 Overall objectives

The overall optimization objectives consist of category-level domain discrepancy, classification loss, entropy loss and network reconstruction loss. Category-level domain discrepancy is computed as $D^{\text{intra}} - D^{\text{inter}}$ because we want to minimize D^{intra} and maximize D^{inter}.

To exploit knowledge from labeled source network, we train a classifier by minimizing cross-entropy loss L_{clf} between classification prediction $y_i^{s,p}$ and ground truth labels y_i^s on every source node i:

$$L_{\text{clf}} = -\frac{1}{N^s} \sum_{i=1}^{N^s} y_i^s \log(y_i^{s,p})$$

To enhance the discriminative capabilities of node embeddings from unlabeled target network, we employ entropy loss L_{entropy} which makes classification prediction $y_j^{t,p}$ on each node j has the tendency to become one-hot vector; thus, it encourages decision boundary to cross the low-density regions in embedding space and enlarge variation between classes on target network [10]:

$$L_{\text{entropy}} = -\frac{1}{N^t} \sum_{j=1}^{N^t} y_j^{t,p} \log(y_j^{t,p})$$

In addition, we also apply first-order loss function L_{net} to source and target networks for preserving neighboring topology, which is proposed in LINE [16]. $L_{\text{net}} = L_{\text{net}}^s + L_{\text{net}}^t$.

$$L_{\text{net}}^{s/t} = -\sum_{(i,j) \in E^{s/t}} \log \sigma(Z_j^{d\,T} Z_i^d) - Q_{\text{neg}} \mathbb{E}_{k \sim P_{\text{neg}}(N)} \log \sigma(-Z_k^{d\,T} Z_i^d)$$

where Q_{neg} is the number of negative samples and $P_{\text{neg}}(N)$ is a noise distribution where negative samples are sampled. We set $P_{\text{neg}}(N) \propto d_i^{0.75}$ where d_i is degree of node i.

By integrating all the objectives mentioned above, the overall loss function is as follows:

$$\text{Loss} = \alpha(D^{\text{intra}} - D^{\text{inter}}) + \beta L_{\text{clf}} + \gamma_1 L_{\text{entropy}} + \gamma_2 L_{\text{net}} \tag{7}$$

where α, β, γ_1 and γ_2 are trade-off parameters to balance different objectives. Algorithm 1 illustrates model training in one epoch, where we iteratively sample S nodes belonging to each class on both networks as one mini-batch and update the model according to Eq. 7.

4.5 Time complexity

Given a graph with $|V|$ nodes, $|E|$ edges and M-dimension node features together with a two-layer GAT with hidden dims h_1, h_2, the worst complexity of f_{emb} is $O(2|V|h_1(h_2 + M) + (h_1 + h_2)(|E| + |V|^2))$ as PPMI matrix is approximately dense in the worst cases. As the complexity of other components is no more than $O(|V|)$ or $O(|E|)$, the overall complexity is comparable to other domain adaptive network embedding models utilizing PPMI matrix.

5 Theoretical analysis

In this section, we prove that even if target network is unlabeled, our model could bound target training error according to source training error and gradually converge to optimal point. We discuss in the embedding space as model encodes both network topology and adaptation results into node embeddings.

Definition 1 $D(P1, P2)$ measures the distance between two probability distributions and satisfies triangular inequality: $D(P1, P2) \leq D(P1, P3) + D(P3, P2)$. $D(P1, P2)$ could also represent classification error when $P1$ and $P2$ stand for prediction and groundtruth labeling functions, respectively.

Definition 2 f_S and f_T denote the groundtruth labeling functions on two networks. h^* denotes the best model trained with Algorithm 1.

Definition 3 $\epsilon_S(h^*, f_S)$ and $\epsilon_T(h^*, f_T)$ are training errors on source and target networks, and they are marked as $\epsilon_S(h^*)$ and $\epsilon_T(h^*)$ for simplicity. Specifically, $\epsilon_S(h^*)$ is computed as:

$$\epsilon_S(h^*) = \mathbb{E}_z[D(f_S, h^*)] = \int_Z P^s(Z) D(f_S, h^*) \mathrm{d}Z$$

and $\epsilon_T(h^*)$ is computed in a similar way. $\epsilon_T(h^*)$ is incomputable in practice as target network is unlabeled.

Definition 4 Let h_t^* denote ideal target predictor: $h_t^* = argmin_{h_t} \epsilon_T(h_t)$. h_t^* is incomputable in practice as target network is unlabeled.

Lemma 1 introduces three basic properties of labeling functions.

Lemma 1 *Two properties are hold h^* and h_t^*:*

$$D(h^*, f_S) \leq \delta_1 \quad D(h_t^*, f_T) \leq \delta_2$$

where δ_1 and δ_2 are small constants. Besides, another property holds for f_S and f_T:

$$D(f_S, f_T) \leq \lambda$$

Firstly, we assume the difference between the best predictor and groundtruth labeling function to be small enough. $D(h^*, f_S) \leq \delta_1$ is because we use all labeled source nodes for training and we test on the same set of nodes, while $D(h_t^*, f_T) \leq \delta_2$ is because h_t^* is the ideal best target predictor. Second, we assume the difference between groundtruth labeling functions to be small enough, which is a common assumption for domain adaptation [47]. In practice, as classifier is shared between networks, such property is guaranteed.

Based on Lemma 1, we introduce Theorem 1 to prove that our model bounds target training error in terms of source training error.

Theorem 1

$$\epsilon_T(h^*) \leq \epsilon_S(h^*) + (\lambda + \delta_1)\mathcal{M} + \lambda \tag{8}$$

where $\mathcal{M} = \int_Z \sum_y^C |P^t(y)P^t(Z|y) - P^s(y)P^s(Z|y)| \mathrm{d}Z$

Term $\mathcal{M}$ represents intra-class domain discrepancy, which is equal to D^{intra}. As our model gradually reduces D^{intra} and source training error $\epsilon_S(h^*)$ during training, we gradually reduces the upper bound of target training error. As a result, target training error decreases accordingly. Finally, h^* has a high classification accuracy on unlabeled target network. It is worth mentioning that when utilizing category-level information, explicitly modeling probability of nodes belonging to each class $P^s(y)$ and $P^t(y)$ is necessary because we do not assume such label prior probability is equal between networks. Thus, we model them by $w^{s,c}$ and $w^{t,c}$ in Eqs. 5, 6, respectively.

Proof

$$
\begin{aligned}
&\epsilon_T(h^*) - \epsilon_S(h^*) \\
&= \epsilon_T(h^*, f_T) - \epsilon_S(h^*, f_T) + \epsilon_S(h^*, f_T) - \epsilon_S(h^*, f_S) \\
&\leq \int_Z |P^t(Z) - P^s(Z)| D(h^*, f_T) \mathrm{d}Z + \epsilon_S(h^*, f_S) + \epsilon_S(f_S, f_T) - \epsilon_S(h^*, f_S) \\
&\leq \int_Z |P^t(Z) - P^s(Z)| (D(h^*, f_S) + \lambda) \mathrm{d}Z + \lambda \\
&= (\lambda + \delta_1) \int_Z \sum_y^C |P^t(y) P^t(Z|y) - P^s(y) P^s(Z|y)| \mathrm{d}Z + \lambda \\
&= (\lambda + \delta_1) \mathcal{M} + \lambda
\end{aligned}
$$

□

Then, we introduce Theorem 2 to show that our model approximates to the ideal target predictor and could converge to optimal point.

Theorem 2

$$
\epsilon_T(h^*) \leq \epsilon_T(h_t^*) + (2\delta_2 + \lambda)\mathcal{M} + \delta_1 + \delta_2 + \lambda \tag{9}
$$

Similarly with Theorem 1, as our model h^* minimizes $\mathcal{M}$, it gradually approximates to ideal target predictor h_t^*. Thus, our model achieves remarkable results when it predicts target labels.

Proof

$$
\begin{aligned}
&\epsilon_T(h^*) - \epsilon_T(h_t^*) \\
&= \epsilon_T(h^*, f_T) - \epsilon_S(h_t^*, f_S) + \epsilon_S(h_t^*, f_S) - \epsilon_T(h_t^*, f_T) \\
&\leq \epsilon_T(h^*, h_t^*) + \epsilon_T(h_t^*, f_T) - \epsilon_S(h_t^*, f_S) + \epsilon_S(h_t^*, f_T) + \epsilon_S(f_S, f_T) - \epsilon_T(h_t^*, f_T) \\
&\leq \int_Z |P^t(Z) - P^s(Z)| (D(f_S, f_T) + D(f_T, h_t^*)) \mathrm{d}Z \\
&\quad + \delta_2 \int_Z |P^t(Z) - P^s(Z)| \mathrm{d}Z + \delta_1 + \delta_2 + \lambda \\
&\leq (2\delta_2 + \lambda)\mathcal{M} + \delta_1 + \delta_2 + \lambda
\end{aligned}
$$

□

Algorithm 1 The proposed model

Require: G^s, G^t, P^s, P^t; hyper parameters; learning rate lr; maximum iterative times ITER; initialized models f_{emb} and f_c, $\theta = \{\theta_{emb}, \theta_c\}$.
Ensure: Target network prediction Y^t
1: Compute node embeddings Z^s and Z^t by Eq. 1
2: Generate pseudo-labels $\hat{Y}^t$ by Eq. 3. $G^t \leftarrow G^t \cup \hat{Y}^t$
3: iter $\leftarrow 0$
4: **while** iter<ITER **do**
5: Sample mini-batch from G^s and G^t
6: Compute $loss$ in terms of Eq. 7
7: Update model: $\theta = \theta - lr * \nabla_\theta loss$
8: Update node embeddings Z^s and Z^t
9: iter $\leftarrow$ iter + 1
10: **end while**
11: $Y^t \leftarrow$ argmax $f_c(f_{emb}(X^t, A^t))$

6 Experiments

6.1 Datasets

We adopt both citation networks and social networks to evaluate the proposed model.

- **Citation**: This dataset is widely used by domain adaptive network embedding models. It is extracted from three AMiner [48] datasets formed in different periods and contains three networks [9]. In each network, node represents an article, and each edge indicates the citation of one article to another. An article can have multiple labels indicating its relevant research topics including "Databases", "Artificial Intelligence", "Computer Vision", "Information Security", and "Networking". The sparse bag-of-words features extracted from the article title were utilized as the attributes for each article. We integrate word dictionaries from three datasets into one dictionary to make sure their features have the same dimension and each dimension represents the same meaning.
- **Twitch**: It contains five networks which are collected separately from different countries or regions on Twitch platform [49]. Each network represents user-user relationship where node corresponds to Twitch users and edge corresponds to mutual friendship. The associated label is binary which indicates whether a streamer uses explicit language. Node features are games liked, location and streaming habits.

Networks in citation or twitch dataset share the same feature space and label space, but they vary in network topology, node attribute distribution and label distribution. We build classification tasks between every two networks which belong to the same dataset. Dataset description is provided in Table 2.

6.2 Baselines

We select baseline models from relevant fields including the following types:

- **Unsupervised network embedding** We use Variational Graph Auto-Encoder(VGAE) [50] as unsupervised network embedding baseline, and it is only trained on target network regardless of source network.
- **Single network embedding** We train GNNs w./w.o. dual structure on labeled source network and directly apply it to unlabeled target network without adaptation. Both GCN and GAT are adopted.

- **Traditional domain adaptation methods** We choose DAN [27] from discrepancy-based methods and DANN [34] from adversarial-based methods.
- **Domain adaptive network embedding** We choose the state-of-the-art models from two aspects (1) utilizing category-level information: CDNE and (2) utilizing global information based on GNN: AdaGCN [13], DANE [37], UDAGCN [10], ASN [11], GRADE [36] and SpecReg [12].

6.3 Implementation details

We use all source labels for training and keep no label on target network for all tasks unless otherwise stated. The GNNs applied in both baselines and ours have two convolution layers with hidden dims 128–16 (citation) or 64–16 (twitch) except for CDNE and AdaGCN which follow their original settings. Node classifier is one-layer MLP for all models. In our proposed model, PPMI matrix is computed with K=3. To compute MMD practically, we use the sum of multiple Gaussian RBF kernels for kernel $k(\cdot,\cdot)$ and the kernel number is set to 10. Besides, the training process is implemented by SGD optimizer with momentum of 0.9 and weight decay of 0.0005 and the convergence is estimated by early stopping. Learning rate lr is set to 0.02 (citation) and 0.01 (twitch). We set smoothing coefficient r equals to 0.1 for citation and 0.7 for twitch. Micro-F1 and Macro-F1 are used as metric in the experiments. Micro-F1 assigns equal weight to each sample and is defined as follows:

$$
\begin{aligned}
F1_{mi} &= \frac{2 \cdot \mathrm{Pr} \cdot \mathrm{Re}}{\mathrm{Pr} + \mathrm{Re}} \\
\mathrm{Pr} &= \frac{\sum_{c=1}^{C} \mathrm{TP}(c)}{\sum_{c=1}^{C} (\mathrm{TP}(c) + \mathrm{FP}(c))} \\
\mathrm{Re} &= \frac{\sum_{c=1}^{C} \mathrm{TP}(c)}{\sum_{c=1}^{C} (\mathrm{TP}(c) + \mathrm{FN}(c))}
\end{aligned}
$$

where $\mathrm{TP}(c)$, $\mathrm{FP}(c)$ and $\mathrm{FN}(c)$ indicate the total number of true positives, false positives, and false negatives associated with class c, respectively. Similarly, Macro-$F1$ assigns equal weight to each class and can be defined as follows:

$$
\begin{aligned}
F1_{ma} &= \frac{1}{C} \sum_{c=1}^{C} \frac{2 \cdot \mathrm{Pr}(c) \cdot \mathrm{Re}(c)}{\mathrm{Pr}(c) + \mathrm{Re}(c)} \\
\mathrm{Pr}(c) &= \frac{\mathrm{TP}(c)}{\mathrm{TP}(c) + \mathrm{FP}(c)}, \\
\mathrm{Re}(c) &= \frac{\mathrm{TP}(c)}{\mathrm{TP}(c) + \mathrm{FN}(c)}
\end{aligned}
$$

For trade-off terms, $\gamma_1 = 0.5$ and $\gamma_2 = 0.2$ are the same among all tasks. For citation dataset, A→C, C→A, D→A and C→D, $\alpha = 10$ and $\beta = 3$; on D→C, $\alpha = 10$ and $\beta = 1$; on A→D, $\alpha = 20$ and $\beta = 3$. For twitch dataset, we set $\alpha = 50$ and $\beta = 4$ on all tasks.

6.4 Performance analysis

Firstly, the results on citation dataset are revealed in Table 3. It is obvious that the overall performance of unsupervised network embedding is much lower due to lack of supervision. Moreover, single network embedding and traditional domain adaptation methods are not

Table 2 Dataset statistics

Datasets	Type	#Nodes	#Edges	#Features	#Labels
Acmv9	Citation	9360	15,602	6775	5
Citationv1		8935	15,113		
Dblpv7		5484	8130		
ENGB(EN)	Twitch	7126	35,324	3170	2
PTBR(PT)		1912	31,299		
RU		4385	37,304		
ES		4648	59,383		
FR		6549	112,667		

Table 3 Averaged micro-F1 scores over 10 runs on citation dataset

	A→C	A→D	C→A	C→D	D→A	D→C	AVG(↑)	VAR(↓)
VGAE	0.3766	0.293	0.2819	0.293	0.2819	0.3766	0.3172	0.0021
GCN	0.7244	0.6750	0.6991	0.7079	0.6455	0.6994	0.6919	0.0008
Dual GCN	0.7509	0.6978	0.7158	0.7241	0.6696	0.7201	0.7131	0.0007
GAT	0.5904	0.5614	0.6191	0.6159	0.6013	0.6949	0.6138	0.0020
Dual GAT	0.6431	0.6100	0.6258	0.6901	0.6797	0.6308	0.6466	0.0010
DAN	0.5714	0.5297	0.5565	0.5590	0.5003	0.5223	0.5399	0.0007
DANN	0.5673	0.5535	0.5553	0.5785	0.5311	0.5627	0.5581	**0.0003**
CDNE	0.7891	0.7203	0.7752	0.7415	0.7659	0.7961	0.7647	0.0008
UDAGCN	0.7328	0.6983	0.6589	0.7177	0.5238	0.6873	0.6698	0.0058
ASN	0.8203	**0.7686**	0.7919	0.7693	0.7217	0.7913	0.7772	0.0011
AdaGCN	0.7305	0.6916	0.6895	0.7258	0.6656	0.7171	0.7034	0.0006
DANE	0.7023	0.6603	0.6700	0.6674	0.6817	0.7449	0.6878	0.0010
GRADE	0.7561	0.7225	0.6903	0.7484	0.6654	0.7256	0.7181	0.0009
SpecReg	0.6214	0.6453	0.6278	0.6530	0.5819	0.6449	0.6291	0.0007
Ours	**0.8313**	0.7568	**0.7985**	**0.7932**	**0.7689**	**0.8272**	**0.7960**	0.0009

We also compute the average score and variance over all six tasks
We mark the best performance within a task with bold letter and we also underline the second best result

comparable to domain adaptive network embedding methods because they fail to capture either domain discrepancy or network topology. In particular, although DAN and DANN utilize domain adaptation technique, they are still not comparable to single network embedding methods mainly because they fail to model network topology. Thus, we further compare different domain adaptive network embedding methods on twitch dataset and report the results in Table 4. Table 5 illustrates the complementary results on both datasets with Macro-F1.

As Tables 3, 4 and 5 reveal, the proposed model outperforms all domain adaptive network embedding baselines on most cross-network node classification tasks. Compared to GNN-based domain adaptive network embedding baselines, we achieve performance gain up to 10% on citation dataset and 13% on twitch dataset with Micro-F1. Such benefit could reach 32% and 17% on two datasets with Macro-F1, respectively. It should be noted that although some baselines perform well with Micro-F1, their performance drops significantly with Macro-F1, such as ASN. Another competitive baseline, GRADE, underperforms on citation datasets,

Table 4 Averaged micro-F1 scores over 10 runs on twitch dataset

	EN→RU	EN→PT	EN→ES	EN→FR	RU→EN	RU→PT	RU→ES	RU→FR	PT→EN	PT→RU	PT→ES
CDNE	0.7328	0.6512	0.6543	0.6313	0.5456	0.6457	0.6871	0.6113	0.4549	0.6542	0.7070
UDAGCN	0.7366	**0.6543**	**0.7063**	0.6283	0.5328	0.6425	0.6797	0.6217	0.5162	0.7048	0.6952
ASN	0.7166	0.6542	**0.7063**	0.6251	0.5167	0.6414	0.6846	0.6314	0.5216	0.7078	0.6816
DANE	0.6143	0.5733	0.6713	0.6173	0.5697	0.6161	0.6306	0.5785	0.4818	0.6070	0.6114
AdaGCN	0.7354	0.6459	0.6444	0.6326	0.5457	0.6314	**0.7074**	0.6204	0.5213	0.7348	0.7014
GRADE	0.7349	0.6453	0.6938	0.6262	0.5717	0.6407	0.7038	**0.6337**	0.5252	0.7248	0.6851
SpecReg	0.6829	0.6129	0.5957	0.5817	0.5475	**0.6553**	0.7071	0.6313	**0.5465**	0.5265	0.5824
Ours	**0.7554**	0.6389	0.7001	**0.637**	**0.5808**	0.6548	0.7010	0.6325	0.5320	**0.7411**	**0.708**

	PT→FR	ES→EN	ES→PT	ES→RU	ES→FR	FR→EN	FR→ES	FR→RU	FR→PT	AVG (↑)	VAR (↓)
CDNE	0.6213	0.5333	0.5457	0.5451	0.5686	0.5456	0.5925	0.6451	0.5457	0.6062	0.0050
UDAGCN	0.6211	0.5524	0.5118	0.7311	0.6311	0.5530	0.7065	0.7252	0.6443	0.6407	0.0055
ASN	0.6286	0.4832	0.6438	0.7218	0.6302	0.5509	0.6749	0.7022	0.6496	0.6393	0.0049
DANE	0.5405	0.5723	**0.6682**	0.6376	0.5608	0.5585	0.6639	0.6076	0.6094	0.6026	**0.0021**
AdaGCN	0.6218	0.5045	0.6412	0.7289	0.6228	0.5461	0.6863	0.7077	0.6501	0.6425	0.0049
GRADE	**0.6330**	**0.5650**	0.6373	0.7348	0.6298	**0.5716**	0.6987	0.7153	0.6435	0.6507	0.0037
SpecReg	0.5356	0.4460	0.4852	0.5155	0.4699	0.5229	0.5822	0.5408	0.6207	0.5694	0.0048
Ours	0.6317	0.5559	0.6543	**0.7465**	**0.6314**	0.5639	**0.7074**	**0.7298**	**0.6558**	**0.6593**	0.0043

We also compute the average score and variance over all twenty tasks
We mark the best performance within a task with bold letter and we also underline the second best result

Table 5 Averaged macro-F1 on citation dataset and twitch dataset

	Citation			Twitch				
	A→{C,D}	C→{A,D}	D→{A,C}	EN→{RU,PT,ES,FR}	RU→{EN,PT,ES,FR}	PT→{EN,RU,ES,FR}	ES→{EN,RU,PT,FR}	FR→{EN,RU,PT,ES}
CDNE	0.7339	0.7425	0.7698	0.5317	0.5213	0.5342	0.5205	0.5219
UDAGCN	0.5970	0.5635	0.4781	0.5526	0.5481	0.5498	0.5592	0.5608
ASN	0.7728	0.7625	0.6886	0.5704	0.5203	0.5411	0.5285	0.5496
DANE	0.6019	0.5645	0.6147	0.5597	0.5532	0.5332	0.5604	0.5501
AdaGCN	0.6618	0.6769	0.5821	0.5206	0.4362	0.4821	0.4596	0.4868
GRADE	0.5803	0.5153	0.5489	0.5721	0.5584	**0.5626**	0.5680	**0.5704**
SpecReg	0.7677	0.6295	0.5307	0.5106	0.4397	0.5478	0.4792	0.5617
Ours	**0.7742**	**0.7841**	**0.7905**	**0.5779**	**0.5637**	0.5480	**0.5711**	0.5687

For example, A→{C,D} refers to averaged score on task A→C and A→D
We mark the best performance within a task with bold letter and we also underline the second best result

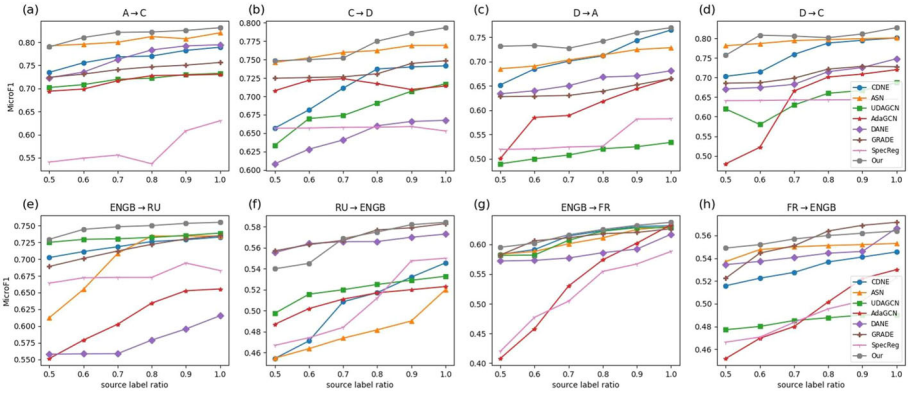

Fig. 3 Model performance changes with ratio of source network labels gradually increasing from 50 to 100%. From left to right: (1) Up: A→C, C→D, D→A, D→C (2) Bottom:EN→RU, RU→EN, EN→FR, FR→EN

despite exhibiting strong performance on the twitch dataset. To summarize, as the proposed model consistently outperforms on both datasets under various metrics, we could prove the superiority of exploiting category-level information. It is worth mentioning that such superiority is more significant on citation dataset, likely due to its larger class variety.

Compared with CDNE on two datasets, we achieve performance gain up to 3% and 11% with Micro-F1, while such benefit reaches 4% and 5% with Macro-F1. The main reason is threefold. First, CDNE ignores inter-class domain discrepancy, which would mix up nodes in different classes from two networks. CDNE assumes that nodes belonging to any given class are equally probable, and it does not factor in the probabilities associated with nodes belonging to specific classes when computing intra-class domain discrepancy. Consequently, such computation is inaccurate. Moreover, the pseudo-labeling strategy of CDNE does not model embedding distribution or network topology, which could reduce the accuracy of pseudo-labels.

In addition, we further valid the robustness of our model from two aspects. At first, we compute the variance among all classification tasks in the same dataset. Since the difficulty of tasks can vary due to differences in feature distribution, label distribution, and network topology, an effective model should exhibit minimal fluctuation across tasks while maintaining competitive performance. As Table 3 and Table 4 reveal, our model not only displays relatively lower variance compared to competitive baselines but also sustains superior performance, corroborating the model's stability and efficiency. Furthermore, we also conduct experiment on the cases where the source network is not fully labeled. We vary labeling ratio from 0.5 to 1 and compare the performance of different models. Figure 3 reveals the result, which indicates that our model is minimally affected by a reduction in the number of labeled source nodes, thereby preserving its competitiveness. This resilience can be attributed to the effective use of category-level information, which enhances the transfer of labeling knowledge (Fig. 4).

We subsequently verify the convergence of our model from two distinct perspectives. Firstly, we study the change of D^{intra} during training and report the results in Fig. 5, given its crucial role in indicating the quality of adaptation. According to Theorems 1 and 2, as our model gradually decreases $D^{\text{intra}}(\mathcal{M})$ during training and finally, reaches the lowest D^{intra}, it could reach a more tighter upper bound for target error $\epsilon_T(h)$ after convergence. Therefore, our model achieves the best performance. Furthermore, we study the convergence speed in

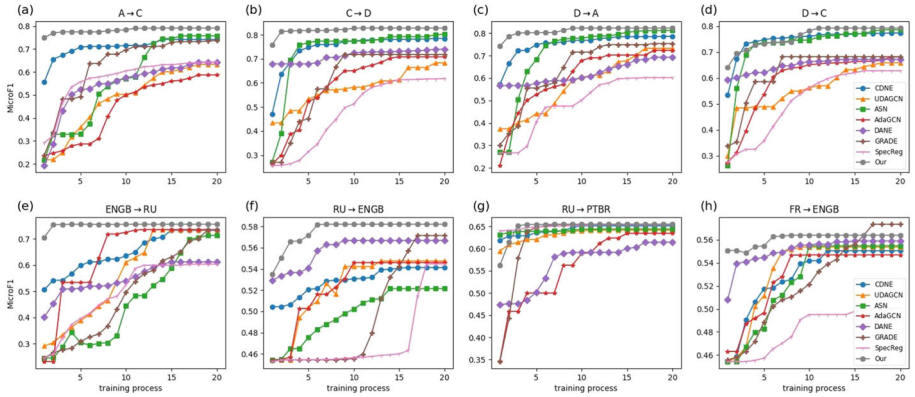

Fig. 4 The micro-f1 increases with training.From left to right: (1) Up: C→D, D→C, A→C, C→A (2) Bottom: EN→RU, RU→EN, RU→PT, FR→EN

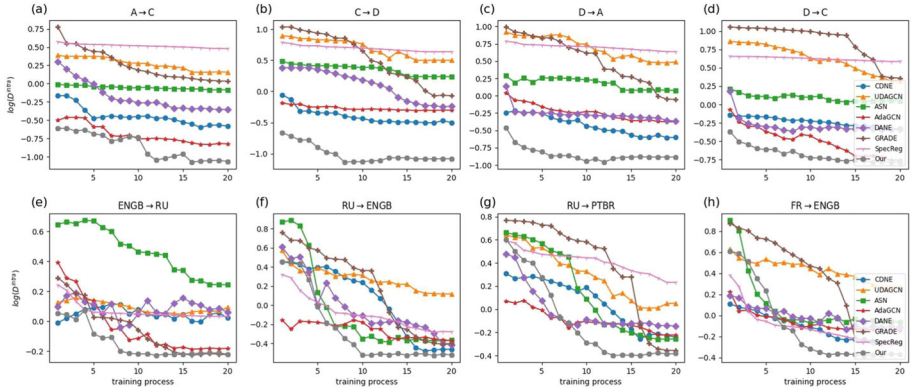

Fig. 5 Intra-class domain discrepancy computed with groundtruth labels decreases with training. From left to right: (1) Up: C→D, D→C, A→C, C→A (2) Bottom: EN→RU, RU→EN, RU→PT, FR→EN

terms of epochs. Although our model updates multiple times with mini-bathes in one epoch, it converges much faster overall as Fig. 4 shows.

Lastly, we observe an interesting phenomenon from Table 3 that without adaptation, the performance of GAT is much worse than GCN, but it surpasses GCN after applying adaptation. This phenomenon could be attributed to the initial lack of effective learning of target embeddings, leading to the calculation of noisy weight coefficients based on these embeddings, thus failing to accurately represent the importance of neighboring nodes.

6.5 Ablation study

We demonstrate the necessity and effectiveness of each model component based on eight variants and report the results in Table 6. First of all, we exclude adaptation from our model, i.e., remove $D^{\text{intra}} - D^{\text{inter}}$ from objective Eq. 7. This results in a significant drop in performance, underscoring the criticality of domain adaptation. Sequentially, we remove two important components for adaptation in turn. The first removed component is inter-class domain discrepancy, which results in a performance drop of about 1.4% and 2.8% on two

Table 6 Ablation study

	Original	w/o. adaptation	w/o. D^{inter}	w/o. weighting	clf-pseudo	w/o. smoothing	w/o. L_{entropy}	w/o. L_{net}	GCN-f_{emb}
Citation	0.7947	0.6576	0.7807	0.7258	0.7022	0.7782	0.7716	0.7771	0.7852
Twitch	0.6593	0.5655	0.6315	0.6178	0.6031	0.6507	0.6469	0.6472	0.6544

The MicroF1 results are computed as an average over all tasks, specifically six for the citation dataset and twenty for the twitch dataset

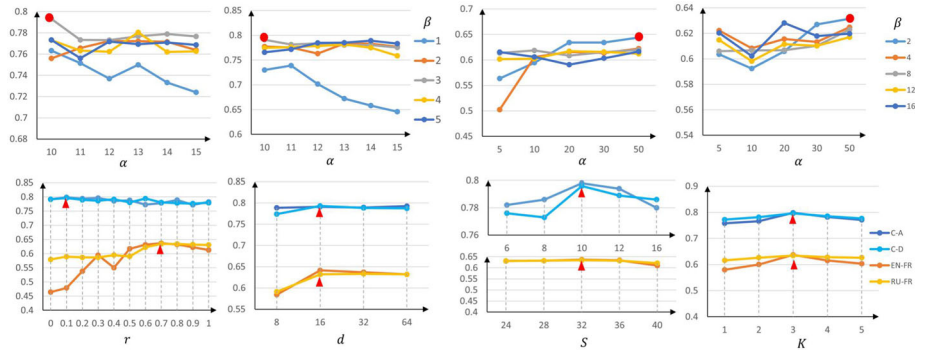

Fig. 6 Parameter search for α, β, S, r and K. Up: from left to right C→D, C→A, EN→FR, RU→FR, search for α, β. Bottom: from left to right r, d, S, K. The y-axis refers to Micro-F1 score in all plots, and red marks indicate the optimal point

datasets, respectively. The second removed component is probability of nodes belonging to each class, where we instead assume an equal class-probability for all nodes, which results in a 7% and 4% performance decline on two datasets, respectively. As Theorem 1 and 2 have demonstrated that such probability is indispensable for theoretical guarantee, the noticeable performance drop is reasonable.

We then validate the effectiveness of pseudo-labeling strategy. Firstly, we directly use classifier predictions on target nodes as pseudo-labels. This variant performs the worst because neither network homophily nor embedding distribution is explicitly modeled. Furthermore, the unstable performance of classifier in the initial stage also degrades the quality of pseudo-labels. We further keep our strategy but remove label smoothing and the variant performs slightly worse than original one, which proves exploiting network homophily is necessary. Furthermore, we prove the effectiveness of L_{entropy} and L_{net} by removing them, respectively, in another two variants which cause performance drop of about 1–2%.

Lastly, we implement a GCN version by replacing GAT in f_{emb} with GCN for further evaluation. It is worth mentioning that although GCN version has a slight performance drop compared to the original model, it still achieves the best overall result against baselines and the performance improvement is still significant, which achieves 0.8–11.5% on citation dataset and 1.2–5.2% on twitch dataset, respectively. Such phenomenon also proves the superiority of exploiting category-level information.

6.6 Parameter sensitivity

We study the sensitiveness of important parameters and report the result in Fig. 6. First of all, as α and β are two important trade-off terms to balance between adaptation and classification, we study their impacts on the performance. When domain discrepancy term is empirically much smaller than classification error, increasing α could boost performance (as seen with the twitch dataset), whereas it might diminish performance when the magnitudes of two terms are approximately similar(as seen with the citation dataset). The value of β should also be smaller when discrepancy term is empirically small. Regarding S, we expect the mini-batch size, $2 \cdot S \cdot C$, not to be too large because larger size would increase more noise when computing domain discrepancy. Thus, although S is larger on twitch dataset, excessively increasing S would degrade performance on both datasets. For label smoothing parameter r, we set it to a

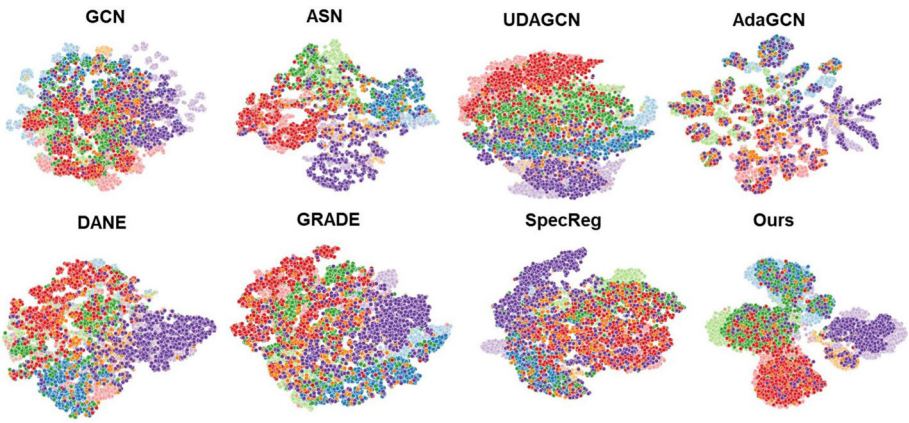

Fig. 7 Visualization of node embeddings in task D→C. Different colors stand for different classes, while dark and light of the same color represent the same class on two networks

small value when the empirical phenomenon of network homophily is remarkable to further smooth pseudo-labels, but basically $r = 0.8$ could boost performance on both datasets. Lastly, we study the impact of embedding dimension h and the step K of PPMI matrix. We find that increasing h to 16 or K to 3 could boost performance, but keep increasing their values would not bring more benefits and would increase complexity.

6.7 Visualization

In Fig. 7, we present a visualization of the embeddings output by GNN-based models, including the proposed model, using t-SNE [51]. One striking observation from the visualization of the proposed model is the close alignment of dark and light shades of the same color. This indicates that our model effectively minimizes intra-class domain discrepancy between networks, clustering most node embeddings from both networks within the same class. Additionally, the distinct boundaries between different colors attest to the discriminative nature of the embeddings produced by the proposed model, thereby demonstrating it effectively maximizes intra-class domain discrepancy and enlarges inter-class embedding variation. Such phenomenon guarantees a better classification performance. In contrast, as evidenced in Fig. 7, the baselines fall short in both adaptation and classification performance. These visualization results corroborate the superiority of our approach which leverages category-level information.

7 Conclusion

We propose category-level domain adaptive network embedding via Graph Attention Network for node classification across networks. The key idea is to minimize intra-class domain discrepancy and maximize inter-class domain discrepancy between source and target networks, concurrently reducing intra-class variation and enlarging inter-class variation within target network at the same time. Theoretical analysis and experiments prove the effectiveness of our model. Since our current model only utilizes a single network as source network, gener-

alizing to multi-source scenario is challenging and should be more applicable. Also, adapting our model to heterogeneous networks will be beneficial for more practical applications.

Acknowledgements This work was funded by the National Natural Science Foundation of China under grant number U21B2046. This work is also supported by the fellowship of China Postdoctoral Science Foundation 2022M713206.

Author Contributions BS: Conceptualization, methodology, software, validation, investigation, writing-original draft, writing-review and editing. YW: Validation, formal analysis, resources, writing-review and editing. JS: Methodology, visualization, data curation, investigation, writing-review and editing. HS: Supervision, project administration, funding acquisition. YL: Supervision, project administration, funding acquisition. XC: Supervision, project administration, funding acquisition All authors reviewed the manuscript.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907
2. Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. Statistics 1050:20
3. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. Adv Neural Inf Process Syst 30:66
4. Hu W, Fey M, Zitnik M, Dong Y, Ren H, Liu B, Catasta M, Leskovec J (2020) Open graph benchmark: datasets for machine learning on graphs. Adv Neural Inf Process Syst 33:22118–22133
5. Feng S, Wan H, Wang N, Li J, Luo M (2021) Twibot-20: a comprehensive twitter bot detection benchmark. In: Proceedings of the 30th ACM international conference on information & knowledge management, pp 4485–4494
6. Feng S, Tan Z, Wan H, Wang N, Chen Z, Zhang B, Zheng Q, Zhang W, Lei Z, Yang S et al (2022) Twibot-22: towards graph-based twitter bot detection. arXiv preprint arXiv:2206.04564
7. Pan SJ, Yang Q (2009) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
8. Guo G, Wang C, Yan B, Lou Y, Feng H, Zhu J, Chen J, He F, Yu P (2022) Learning adaptive node embeddings across graphs. IEEE Trans Knowl Data Eng 6:66
9. Shen X, Dai Q, Mao S, Chung F-L, Choi K-S (2020) Network together: Node classification via cross-network deep network embedding. IEEE Trans Neural Netw Learn Syst 32(5):1935–1948
10. Wu M, Pan S, Zhou C, Chang X, Zhu X (2020) Unsupervised domain adaptive graph convolutional networks. In: Proceedings of the web conference 2020, pp 1457–1467
11. Zhang X, Du Y, Xie R, Wang C (2021) Adversarial separation network for cross-network node classification. In: Proceedings of the 30th ACM international conference on information & knowledge management, pp 2618–2626
12. You Y, Chen T, Wang Z, Shen Y (2023) Graph domain adaptation via theory-grounded spectral regularization. In: The eleventh international conference on learning representations
13. Dai Q, Wu X-M, Xiao J, Shen X, Wang D (2022) Graph transfer learning via adversarial domain adaptation with graph convolution. IEEE Trans Knowl Data Eng 6:66
14. Wang D, Cui P, Zhu W (2016) Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1225–1234
15. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710
16. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: large-scale information network embedding. In: Proceedings of the 24th international conference on World Wide Web, pp 1067–1077
17. Jiang M (2021) Cross-network learning with partially aligned graph convolutional networks. In: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, pp 746–755

18. Ni J, Chang S, Liu X, Cheng W, Chen H, Xu D, Zhang X (2018) Co-regularized deep multi-network embedding. In: Proceedings of the 2018 World Wide Web conference, pp 469–478
19. Sun J, Zhang Y (2019) Multi-graph convolutional neural networks for representation learning in recommendation. In: IEEE ICDM
20. Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H, He Q (2020) A comprehensive survey on transfer learning. Proc IEEE 109(1):43–76
21. Karisani P (2022) Multiple-source domain adaptation via coordinated domain encoders and paired classifiers. Proc AAAI Conf Artif Intell 36(7):7087–7095
22. Grangier D, Iter D (2022) The trade-offs of domain adaptation for neural language models. In: Proceedings of the 60th annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp 3802–3813
23. Martins P, Marinho Z, Martins AF (2022) Efficient machine translation domain adaptation. In: Proceedings of the 1st workshop on semiparametric methods in NLP: decoupling logic from knowledge, pp 23–29
24. Tzeng E, Hoffman J, Zhang N, Saenko K, Darrell T (2014) Deep domain confusion: maximizing for domain invariance. arXiv preprint arXiv:1412.3474
25. Li Z, Zhao X, Zhao C, Tang M, Wang J (2022) Transfering low-frequency features for domain adaptation. In: 2022 IEEE international conference on multimedia and expo (ICME). IEEE, pp 1–6
26. Shen Y, Yang Y, Yan M, Wang H, Zheng Y, Guibas LJ (2022) Domain adaptation on point clouds via geometry-aware implicits. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 7223–7232
27. Long M, Cao Y, Wang J, Jordan M (2015) Learning transferable features with deep adaptation networks. In: International conference on machine learning. PMLR, pp 97–105
28. Long M, Wang J, Ding G, Sun J, Yu PS (2013) Transfer feature learning with joint distribution adaptation. In: Proceedings of the IEEE international conference on computer vision, pp 2200–2207
29. Kang G, Jiang L, Yang Y, Hauptmann AG (2019) Contrastive adaptation network for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4893–4902
30. Zhu Y, Zhuang F, Wang J, Ke G, Chen J, Bian J, Xiong H, He Q (2020) Deep subdomain adaptation network for image classification. IEEE Trans Neural Netw Learn Syst 32(4):1713–1722
31. Sun B, Saenko K (2016) Deep coral: Correlation alignment for deep domain adaptation. In: Computer Vision–ECCV 2016 workshops: Amsterdam, The Netherlands, October 8–10 and 15–16, 2016, Proceedings, Part III 14. Springer, pp 443–450
32. Zellinger W, Grubinger T, Lughofer E, Natschläger T, Saminger-Platz S (2017) Central moment discrepancy (cmd) for domain-invariant representation learning. arXiv preprint arXiv:1702.08811
33. Yu Q, Hashimoto A, Ushiku Y (2021) Divergence optimization for noisy universal domain adaptation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2515–2524
34. Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H, Laviolette F, Marchand M, Lempitsky V (2016) Domain-adversarial training of neural networks. J Mach Learn Res 17(1):2030–2096
35. Shen X, Dai Q, Chung F-l, Lu W, Choi K-S (2020) Adversarial deep network embedding for cross-network node classification. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 2991–2999
36. Wu J, He J, Ainsworth E (2022) Non-iid transfer learning on graphs. arXiv preprint arXiv:2212.08174
37. Zhang Y, Song G, Du L, Yang S, Jin Y (2019) Dane: domain adaptive network embedding. In: IJCAI
38. Li H, Tong H, Weng Y (2022) Domain adaptation in physical systems via graph kernel. In: Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining, pp 868–876
39. Li H, Wang X, Zhang Z, Zhu W (2022) Ood-gnn: out-of-distribution generalized graph neural network. IEEE Trans Knowl Data Eng 6:66
40. Fan S, Wang X, Shi C, Cui P, Wang B (2021) Generalizing graph neural networks on out-of-distribution graphs. arXiv preprint arXiv:2111.10657
41. Chen Y, Zhang Y, Bian Y, Yang H, Kaili M, Xie B, Liu T, Han B, Cheng J (2022) Learning causally invariant representations for out-of-distribution generalization on graphs. Adv Neural Inf Process Syst 35:22131–22148
42. Wu Y-X, Wang X, Zhang A, He X, Chua T-S (2022) Discovering invariant rationales for graph neural networks
43. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein generative adversarial networks. In: International conference on machine learning. PMLR, pp 214–223
44. Mao X, Li Q, Xie H, Lau RY, Wang Z, Paul Smolley S (2017) Least squares generative adversarial networks. In: Proceedings of the IEEE international conference on computer vision, pp 2794–2802
45. Zhu X, Ghahramani Z (2002) Learning from labeled and unlabeled data with label propagation

46. Gretton A, Borgwardt KM, Rasch MJ, Schölkopf B, Smola A (2012) A kernel two-sample test. J Mach Learn Res 13(1):723–773
47. Ben-David S, Blitzer J, Crammer K, Kulesza A, Pereira F, Vaughan JW (2010) A theory of learning from different domains. Mach Learn 79(1):151–175
48. Tang J, Zhang J, Yao L, Li J, Zhang L, Su Z (2008) Arnetminer: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 990–998
49. Rozemberczki B, Allen C, Sarkar R (2021) Multi-scale attributed node embedding. J Complex Netw 9(2):1–22
50. Kipf TN, Welling M (2016) Variational graph auto-encoders. arXiv preprint arXiv:1611.07308
51. Van der Maaten L, Hinton G (2008) Visualizing data using t-sne. J Mach Learn Res 9(11):66