Adversarial Separation Network for Cross-Network Node Classification

Xiaowen Zhang*
Yuntao Du*
zhangxw@smail.nju.edu.cn
duyuntao@smail.nju.edu.cn
State Key Laboratory for Novel Software Technology at Nanjing University
Nanjing, Jiangsu, China

Rongbiao Xie
State Key Laboratory for Novel Software Technology at Nanjing University
Nanjing, Jiangsu, China
rongbiaoxie@smail.nju.edu.cn

Chongjun Wang†
State Key Laboratory for Novel Software Technology at Nanjing University
Nanjing, Jiangsu, China
chjwang@nju.edu.cn

ABSTRACT

Node classification is an important yet challenging task in various network applications, and many effective methods have been developed for a single network. While for cross-network scenarios, neither single network embedding nor traditional domain adaptation can directly solve the task. Existing approaches have been proposed to combine network embedding and domain adaptation for cross-network node classification. However, they only focus on domain-invariant features, ignoring the individual features of each network, and they only utilize 1-hop neighborhood information (local consistency), ignoring the global consistency information. To tackle the above problems, in this paper, we propose a novel model, *Adversarial Separation Network* (ASN), to learn effective node representations between source and target networks. We explicitly separate domain-private and domain-shared information. Two domain-private encoders are employed to extract the domain-specific features in each network and a shared encoder is employed to extract the domain-invariant shared features across networks. Moreover, in each encoder, we combine local and global consistency to capture network topology information more comprehensively. ASN integrates deep network embedding with adversarial domain adaptation to reduce the distribution discrepancy across domains. Extensive experiments on real-world datasets show that our proposed model achieves state-of-the-art performance in cross-network node classification tasks compared with existing algorithms.

CCS CONCEPTS

• **Information systems** → Social networks; • **Computing methodologies** → Machine learning.

KEYWORDS

cross-network node classification, transfer learning, domain adaptation, graph embedding

ACM Reference Format:
Xiaowen Zhang, Yuntao Du, Rongbiao Xie, and Chongjun Wang. 2021. Adversarial Separation Network for Cross-Network Node Classification. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.*

1 INTRODUCTION

The node classification task is to predict the node labels by network topology structure or information derived from the nodes in the network [44]. So far, many effective and feasible methods have been proposed [16, 46]. However, they are mostly for a single network and do not consider the cross-network scenarios. When given a new graph with no labels, even it is similar to an existing graph, the data distribution shift across them would pose an obstacle for applying a node classification model learned from an existing (source) network to a new (target) network [43]. An example of cross-network node classification is illustrated in Figure 1. Cross-network node classification can benefit various actual applications. *e.g.* , in social network, there exist a source network where all users are associated with some labels indicating their interests, and a target network where no users have observable labels. By transferring knowledge from the labeled source network to the unlabeled target network, we could discover the interest of the target user.

Domain adaptation has attracted much attention in recent years, which aims to leverage the labeled information from a source domain to improve the performance of the unlabeled target domain [27]. Many approaches are proposed for domain adaptation, making it widely used in computer vision (CV) [24, 31] and natural language processing (NLP) [4, 40]. Despite the significant achievements, the application of domain adaptation to network structure data (graph) is still difficult and challenging. Since each node in the graph has a complicated relationship (*i.e.*, edges) with others, which violates the assumption of traditional domain adaptation that the data sample is independent and identically distributed (IID) in each domain. Therefore, existing domain adaptation algorithms perform poorly in cross-network node classification tasks since they cannot directly model the network structure information (The specific analysis is shown in section 5.4).

Currently, there are some attempts to apply domain adaptation for cross-network node classification based on network structure data [8]. They aim to learn low-dimensional node representations and reduce domain discrepancy by combining deep network embedding and domain adaptation. CDNE [35] proposes to learn label-discriminative and network-invariant representations based on Maximum Mean Discrepancy (MMD). ACDNE [34] integrates adversarial domain adaptation [10] with network embedding to learn node representation. Besides, AdaGCN [6] and UDAGCN [43] are

*Both authors contributed equally to this research.
†Corresponding author

CIKM' 21, November 1–5,2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482228

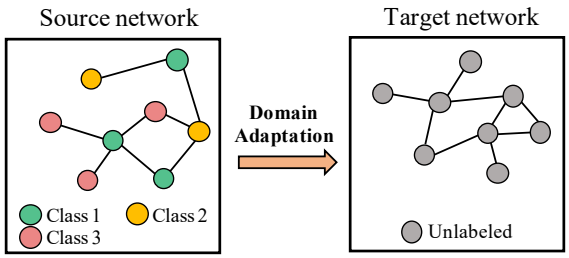


Figure 1: An example of cross-network node classification

proposed to leverage graph convolutional networks (GCN) [19] and domain adaptation to reduce the domain discrepancy. However, these methods either learn to map feature representations from one domain to another, or learn to extract domain-invariant features, but ignore the individual features of each network, which can degrade the performance of transfer learning tasks across networks.

Besides, addressing the cross-network node classification task still faces some **challenges**:

(1) Existing cross-network node classification approaches mainly use traditional GCN to learn node representation. Unfortunately, it only considers the 1-hop neighbor nodes (local consistency) [18], while global consistency information is not well utilized. In fact, global consistency is also critical as the node may be influenced by its neighborhoods with different distances [41]. Therefore, how to explore local and global information in a network and how to integrate them is worth thinking about.

(2) Previous cross-network domain adaptation methods all attempt to learn shared features that can be migrated across networks (such as author organization and publisher in paper citation networks). But they neglect the influences of each domain's private features that cannot be transferred to another (such as the paper ID in paper citation networks). How to explicitly capture features that are unique to each network and shared between networks is very important.

To address the first challenge, we adopt a dual GCN framework, including a local GCN to extract local 1-hop neighbor features, and a global GCN to capture global topological information of the network. To address the second, we adopt the idea of Domain Separation Networks [2] and assume that explicitly separating domain-private features can improve the model's ability to extract the domain-invariant features. The idea of private and public features has been proposed in image recognition tasks before [32]. But as far as we know, in cross-network adaptation, we are the first to explicitly separate shared features from the private features based on graph data across networks. Specifically, we separate domain-private and domain-shared features by designing a private encoder for each network and a shared encoder across networks, and force the two encoders to extract different features. Besides, to ensure the extracted features can retain topological information of the original network, we introduce a decoder in each domain to reconstruct the network structure from the concatenation of domain-private and domain-shared features.

In this paper, we propose a novel model named *Adversarial Separation Network* (ASN) for cross-network node classification tasks. It aims to learn effective node representations between source and target networks by integrating deep network embedding with adversarial domain adaptation. Our principal **contributions** can be summarized as follows:

(1) We adopt a dual GCN model to integrate local and global consistency with an attention layer to learn comprehensive node representations across networks effectively.
(2) We design an innovative way to explicitly separate domain-private and domain-shared information, by introducing a shared encoder to extract domain-shared features and two private encoders to extract each domain's unique features.
(3) Extensive experimental results in the real-world datasets verify the effectiveness of the proposed ASN model for cross-network node classification.

2 RELATED WORK

Our work is closely related to domain adaptation and cross-network node classification and next we briefly review them.

2.1 Single Network Node Classification

Network representation learning (network embedding) aims to learn the low-dimensional potential representation of nodes in the network, and can be used for various graph-based tasks, such as classification [1, 15, 22], link prediction [5, 23],clustering[7] and visualization [21].

Some methods based on deep learning have been proposed to learn more informative node representation. DeepWalk [30] is the first network embedding method proposed to use deep learning techniques, which employs random walk sampling to generate the neighborhood of each node. DNGR [3] is based on deep neural network which uses a random surfing strategy to capture network structure information and converts the structural information into PPMI matrix. The above methods only use the network structure information to learn the low-dimensional network representation, but ignore the attributes of nodes. Recently, several attributed network embedding methods [16, 46] have proposed to jointly utilize network structures, node attributes, and available node labels to learn comprehensive network representations.

Although the attributed network embedding methods can capture the relationships between nodes across networks based on node attributes, none of them considered the domain discrepancy across different networks.

2.2 Domain Adaptation

Domain adaptation, a representative method in transfer learning, aims to leverage the information of labeled source domain to improve the performance of unlabeled target domain. In recent years, many approaches based on deep learning are proposed for domain adaptation [9, 42], which can be roughly categorized into two groups: distribution alignment [24, 42] and adversarial domain adaptation [9, 38].

The distribution alignment methods aim to reduce the statistic discrepancy across domains by learning features with less distribution discrepancy. DAN [24] minimizes the feature discrepancy between the last three layers of AlexNet [20] and the multiple-kernel MMD is used to measure the discrepancy. JAN [26] minimizes the joint MMD distance to reduce the conditional distribution discrepancy across domains. Other measures are also adopted such as Kullback-Leibler (KL) divergence, correlation alignment (CORAL) [36] and central moment discrepancy (CMD) [45]. These methods can utilize the deep neural network to extract more transferable features and also have achieved remarkable performance.

In this paper, we focus on adversarial domain adaptation methods, which are motivated by the idea of GAN [11]. The Domain adversarial neural network (DANN) [10] aims to learn domain-invariant features and designs a gradient reversal layer (GRL) to back-propagate the gradients. WDGRL [33] utilizes a neural network to estimate empirical Wasserstein distance across domains. CDAN [25] leverages the discriminative information from the classifier to assist the adversarial adaptation. MADA [29] captures a multimodal structure to support the fine-grained alignment of different data distributions from different domains. However, the above domain adaptation algorithms cannot be directly applied to the cross-network node classification task since they cannot effectively capture the complex network structure information.

2.3 Cross-network Node Classification

Recently, to solve the problem of cross-network node classification, some studies have been working to effectively solve the problem of cross-network node classification by effectively combining domain adaptation and deep network embedding to learn the low-dimensional node representation across networks.

CDNE [35] is proposed to learn label-discriminative and network-invariant representations by incorporating MMD-based domain adaptation into network embedding. ACDNE [34] integrates adversarial domain adaptation based on DANN with deep network embedding to learn node representation. AdaGCN [6] leverages GCN and adversarial domain adaptation to address cross-network node classification. The above methods only consider the local 1-hop neighbor nodes for network embedding, but neglect the network's global consistency information. UDAGCN [43] attempts to solve the issue by adopting dual GCNs to jointly exploit local and global consistency for feature aggregation, while it ignores the individual and specific features of each single network.

3 PROBLEM DEFINITION

In this paper, we focus on node classification tasks on network graphs. Let $\mathcal{G} = (V, E, A, X, Y)$ donate a graph with a set of nodes $V = \{v_i\}_{i=1,\dots,N}$ and a set of edges $E = \{e_{i,j} = (v_i, v_j)\}$ which indicate the relationship between two nodes. Let $A \in \mathbb{R}^{N\times N}$ donate the adjacency matrix of $\mathcal{G}$, which represents the topological structure of a graph where N is the number of nodes, and $A_{i,j} = 1$ if $e_{i,j} = (v_i, v_j) \in E$; otherwise $A_{i,j} = 0$. Let $X \in \mathbb{R}^{N\times O}$ and $Y \in \mathbb{R}^{N\times K}$ donate the node attribute matrix and node label matrix associated with $\mathcal{G}$, where O is the number of node attributes and K is the number of node labels in $\mathcal{G}$.

Source Network Let $\mathcal{G}^S = (V^S, E^S, A^S, X^S, Y^S)$ be a labeled source network with a set of labeled nodes V^S and a set of edges E^S. Y^S is the label matrix for the source graph.

Target Network Let $\mathcal{G}^T = (V^T, E^T, A^T, X^T)$ be a completely unlabeled target domain network with a set of unlabeled nodes V^T and a set of edges E^T.

Cross-Network Node Classification Given a fully labeled source graph $\mathcal{G}^S$ and an unlabeled target graph $\mathcal{G}^T$, the cross-network node classification is to learn appropriate node representations and build a node classifier to accurately classify the nodes in the target network with the assistance of the abundant labeled information from the source network. However, since there are no labels in $\mathcal{G}^T$, it is a challenging task.

4 METHODOLOGY

This section presents our model *Adversarial Separation Network* (ASN) for cross-network node classification.

4.1 Overview of Model Framework

Figure 2 shows the model framework of our proposed *Adversarial Separation Network* (ASN). As shown in Figure 2(a), the proposed model consists of three encoders (*i.e.,* a source encoder E_S, a target encoder E_T and a shared encoder E_H), two decoders (*i.e.,* a source decoder R_S and a target decoder R_T), a domain discriminator D and a node classifier C.

Specifically, the source encoder E_S and target encoder E_T are applied to capture each network's specific features that cannot be shared across networks. While the shared encoder E_H is trained adversarially to extract the shared features in both networks. The decoders (R_S and R_T) aim to reconstruct the original graph according to the features extracted by the corresponding encoders. More importantly, to reduce distribution discrepancy across networks, the domain discriminator D is employed for adversarial domain adaptation. The node classifier C is trained to assist the unlabeled target network in carrying out node classification tasks.

Besides, as shown in Figure 2(b) and (c), each encoder contains a local GCN (G_l) and a global GCN (G_g). G_l aims to extract the local features by the graph adjacency matrix and G_g aims to capture the global consistency relationship by the Positive Point-wise Mutual Information (PPMI) matrix [47].

4.2 Private and Shared Node Representation

As mentioned before, all existing domain adaptation approaches only focus on creating a map [35] or learning shared representations across domains [43], they ignore the individual features of each domain. To settle this issue, we apply the idea of Domain Separation Networks [2] for node classification, and we assume that explicitly separating features that are unique (private) to each domain can improve the model's ability to extract the shared features across domains.

4.2.1 **Difference Loss.** The difference loss is applied to encourage the shared and private encoders to encode different aspects of both domain's features. We define the difference loss via a soft subspace orthogonality constraint between the private and shared node representations of each domain. Let $Z^S_{sh} = E_H(X^S, A^S, P^S)$ and $Z^T_{sh} = E_H(X^T, A^T, P^T)$ be the matrices that are hidden *shared* representations of source and target graph network respectively. P^S and P^T are the PPMI matrices of source network and target network, respectively. Similarly, let $Z^S_{pr} = E_S(X^S, A^S, P^S)$ and $Z^T_{pr} = E_T(X^T, A^T, P^T)$ be the matrices that are hidden *private* representations respectively. The difference loss ensures orthogonality and separation between the private and shared node representations of each network:

$$\begin{aligned}\min_{E_S,E_T,E_H} \mathcal{L}_{diff} &= \min_{E_S,E_H} \mathcal{L}^S_{diff} + \min_{E_T,E_H} \mathcal{L}^T_{diff} \\ &= \|{Z^S_{sh}}^\top Z^S_{pr}\|^2_F + \|{Z^T_{sh}}^\top Z^T_{pr}\|^2_F \end{aligned} \tag{1}$$

where $\|\cdot\|^2_F$ is the squared Frobenius norm.

4.2.2 **Reconstruction Loss.** To force extracted features to maintain the structure information of the original network, we introduce the reconstruction loss and propagate it back to the source and target encoders. The decoder should make the reconstructed adjacency matrix ($\hat{A^S}$ or $\hat{A^T}$) as similar as possible to the original adjacency matrix (A^S or A^T), since the adjacency matrix determines the structure of the graph. Thus each domain's private encoder and decoder are trained to minimize the cross entropy loss:

$$\begin{aligned}\min_{E_S,E_T,R_S,R_T} \mathcal{L}_{recon} &= \min_{E_S,R_S} \mathcal{L}^S_{recon} + \min_{E_T,R_T} \mathcal{L}^T_{recon} \\ &= -\mathbb{E}_{a^s_{ij}\sim A^S}[a^s_{ij} log \hat{a^s_{ij}} + (1-a^s_{ij}) log(1-\hat{a^s_{ij}})] \\ &\quad -\mathbb{E}_{a^t_{ij}\sim A^T}[a^t_{ij} log \hat{a^t_{ij}} + (1-a^t_{ij}) log(1-\hat{a^t_{ij}})] \end{aligned} \tag{2}$$

where a^s_{ij} or a^t_{ij} (0 or 1) represents the original value of an element in the adjacency matrix A^S or A^T, and $\hat{a^s_{ij}}$ or $\hat{a^t_{ij}}$ (between 0 and 1) represents the value of the corresponding element in the reconstructed adjacency matrix $\hat{A^S} = R_S([Z^S_{pr}, Z^S_{sh}])$ or $\hat{A^T} = R_T([Z^T_{pr}, Z^T_{sh}])$, where $[\cdot,\cdot]$ represents the operation of concatenating two tensors together.

4.3 Local and Global Node Representation

As shown in Figure 2(b) and (c), in the process of node representation learning, to capture the local and global information in each network, we introduce a dual-GCNs framework: local GCN (G_l) and global GCN (G_g). We feed both the source network and target network into the node representation learning module.

4.3.1 **Local GCN.** By directly utilizing the proposed GCN model in [19], we formulate the local GCN G_l as a type of feed-forward neural network. Given the node attribute matrix X and graph adjacency matrix A, the output

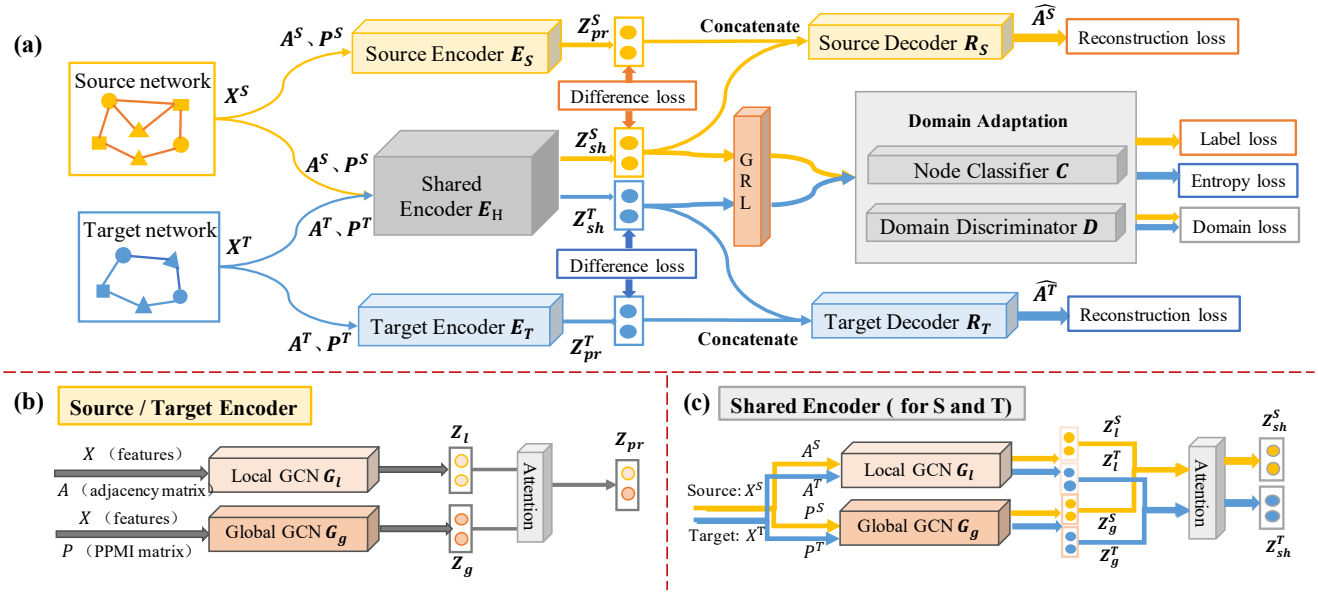


Figure 2: The framework of *Adversarial Separation Network* (ASN) for cross-network node classification

of the i-th hidden layer $Z_l^{(i)}$ of the network G_l is defined as:

$$Z_l^{(i)}(X) = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}Z_l^{(i-1)}\tilde{W}^{(i)}) \tag{3}$$

where $\sigma(\cdot)$ denotes the activation function, $\tilde{A} = A + I_n$ is the adjacent matrix with self-loops ($I_n \in \mathbb{R}^{N\times N}$ is an identity matrix) and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. And $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{\frac{1}{2}}$ is the symmetric normalized adjacency matrix. $Z_l^{(i-1)}$ represents the output of the (i-1)-th layer and $Z_l^0 = X$. $\tilde{W}^{(i)}$ represent the trainable weight parameters of the i-th layer.

4.3.2 **Global GCN.** To capture the global consistency relationship, we introduce a global GCN and follow [47] to employ the PPMI matrix to measure the structural proximity between nodes within k steps in a network.

In a network $\mathcal{G}$, we first obtain the transition probability matrix $\mathcal{T}$ through random walk ($\mathcal{T}_{ij}$ represents the transition probability of visiting node v_j from v_i after k steps). We then aggregate an overall transition probability matrix by weighting closer neighborhoods more. Next, the PPMI matrix $P \in \mathbb{R}^{N\times N}$ associated with network $\mathcal{G}$ is calculated to measure the topological proximity between a pair of nodes v_i and v_j. A higher positive value of P_{ij} indicates that node v_i has a strong network connection with v_j within k steps in network $\mathcal{G}$; otherwise, $P_{ij} = 0$. For the detailed calculation of P, please refer to [47].

After calculating P to measure the topological proximity between nodes in a network, we donate the P-based convolutional network as G_g and the output of the i-th hidden layer $Z_g^{(i)}$ of network G_g is defined as follows:

$$Z_g^{(i)}(X) = \sigma(D^{-\frac{1}{2}}PD^{-\frac{1}{2}}Z_g^{(i-1)}W^{(i)}) \tag{4}$$

where P is the PPMI matrix, $D_{ii} = \sum_j P_{ij}$ is the normalized matrix, $Z_g^{(i-1)}$ is the output of the (i-1)-th layer and $Z_g^0 = X$. $W^{(i)}$ are the trainable weight parameters of the i-th layer.

4.3.3 **Attention.** As embeddings from local and global GCN contribute differently, we add an attention layer to capture the significance of each embedding to produce a unified node representation.

For each network, after performing the local and global node embedding module, we obtain Z_l and Z_g, which represent the output features of G_l and G_g. To compute the weight coefficients $\omega_1 \in \mathbb{R}$ and $\omega_2 \in \mathbb{R}$ for Z_l and Z_g respectively, we assign a linear transformation layer fc as the attention function:

$$\omega_i = fc([Z_l, Z_g]), i = 1, 2 \tag{5}$$

Then we normalize the weights ω_i with a softmax layer:

$$\omega_i = \frac{exp(\omega_i)}{\sum_{i=1}^{2} exp(\omega_i)} \tag{6}$$

After summing the weights and the corresponding features, we can get the final output Z of the encoder:

$$Z = \omega_1 Z_l + \omega_2 Z_g \tag{7}$$

Note that each encoder contains a local GCN and a global GCN. For example, for shared encoder E_H, after performing the local and global node embedding module on the source domain graph, Z_l^S and Z_g^S are obtained. After conducting Equation (5)-(7), we can obtain the final output Z_{sh}^S.

4.4 Cross-Network Node Classification

To better learn knowledge across domains to assist in node classification tasks, we adopt a node classifier $C(\cdot)$ to learn label-discriminative node representations, by incorporating the node classification loss for the labeled source network and the entropy loss for the unlabeled target network.

4.4.1 **Node Classification Loss.** The classification loss trains the model to predict the node labels we are ultimately interested in. Since the target domain is unlabeled, the classification loss is applied only to the source network and we need to minimize the classification loss:

$$\min_{E_H, C} \mathcal{L}_{cls} = \frac{1}{N_s}\sum_{i=1}^{N_s} L(f(x_i^s), y_i^s)) \tag{8}$$

where $L(\cdot)$ is the cross-entropy loss and N_s is the number of nodes in the source network. y_i^s denotes the ground-truth label of node v_i^s and $f(x_i^s) = C(E_H(x_i^s, A^S, P^S))$ is the the predicted probability of node v_i^s.

4.4.2 **Entropy Loss.** To directly access unlabeled target data and force the node classifier $C(\cdot)$ to pass through the low-density regions of the target feature space, we employ an entropy loss for the target network. Following [12], we achieve it via minimizing the conditional entropy with respect to the predicted probability on the target network:

$$\min_{E_H, C} \mathcal{L}_{ent} = \frac{1}{N_t} \sum_{i=1}^{N_t} H(f(x_i^t)) \tag{9}$$

where $H(f(x_i^t)) = -\sum_{k=1}^{K} f_k(x_i^t) \log f_k(x_i^t)$ and $H(\cdot)$ is the entropy function, and N_t is the number of nodes in the target network. $f(x_i^t) = C(E_H(x_i^t, A^T, P^T))$ is the predicted probability for x_i^t and $f_k(x_i^t)$ is the probability of predicting the node v_i^t to class k in the target network.

4.5 Adversarial Domain Adaptation

To make the shared node representations learned by our model to be domain-invariant, we employ an adversarial domain adaptation approach. The domain discriminator $D(\cdot)$ is utilized to distinguish whether the shared features are from the source network or the target network. Meanwhile, the shared encoder E_H tries to produce domain-invariant features that make the domain discriminator into believing that there is no difference between Z_{sh}^S and Z_{sh}^T.

As a result, domain-invariant features can be extracted from the outputs of shared encoder E_H, which are expected to reduce the distribution discrepancy across networks. Thus, by utilizing nodes from source and target networks for training, the objective of domain adversarial loss is defined as:

$$\mathcal{L}_{adv} = \mathbb{E}_{x^s \sim \mathcal{G}^S} \log D(z_{sh}^s) + \mathbb{E}_{x^t \sim \mathcal{G}^T} \log[1 - D(z_{sh}^t)] \tag{10}$$

where $D(z_{sh}^s)$ and $D(z_{sh}^t)$ donate the predicted domain labels of node from source and target domains, respectively. $z_{sh}^s = E_H(x^s, A^S, P^S)$ and $z_{sh}^t = E_H(x^t, A^T, P^T)$ are the node embedding outputs of shared encoder E_H.

The adversarial learning procedure is a min-max game. The domain discriminator D aims to minimize the domain adversarial loss, while the shared encoder E_H is trained to maximize the loss. Thus, the objective of domain adaptation is:

$$\max_{E_H} \min_{D} \mathcal{L}_{adv} \tag{11}$$

4.6 Overall Objectives

To better transfer knowledge from the source network to the target network, our model consists of three goals to enable classifying nodes in the target network. The first is to encode both domain-private and domain-shared features by explicitly capturing information that is unique to each network and shared between networks. The second is to learn label discriminative features to classify the source graphs well. And the third is to learn domain-invariant node representations via adversarial domain adaptation approach.

By integrating all the objectives mentioned before, the goal of ASN is to optimize the overall objectives as follows:

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda_e \mathcal{L}_{ent} + \lambda_d \mathcal{L}_{adv} + \lambda_f \mathcal{L}_{diff} + \lambda_r \mathcal{L}_{recon} \tag{12}$$

where λ_d, λ_e, λ_f and λ_r represent the trade-off parameters to balance the importances of different loss terms. To achieve adversarial training, we apply gradient reversal layer (GRL) [10] instead of training the shared encoder and the discriminator alternately. GRL behaves as the identity function during the forward propagation and inverts the gradient sign during the backward propagation, hence driving the parameters to maximize the output.

4.7 Algorithm Description

Our algorithm is illustrated in Algorithm 1. Given a labeled source network $\mathcal{G}^S = (V^S, E^S, A^S, X^S, Y^S)$ and an unlabeled target network $\mathcal{G}^T = (V^T, E^T, A^T, X^T)$, our goal is to obtain the optimal model that can classify the target network correctly.

Algorithm 1 Adversarial Separation Network for Cross-Network Node Classification

Input: source network: $\mathcal{G}^S = (V^S, E^S, A^S, X^S, Y^S)$ and target network: $\mathcal{G}^T = (V^T, E^T, A^T, X^T)$
Parameter: learning rate lr, trade-off parameters $\lambda_d, \lambda_f, \lambda_r$ and λ_e
Output: optimal encoders E_S, E_T, E_H and optimal node classifier C

1: Compute PPMI matrix P^S for source network.
2: Compute PPMI matrix P^T for target network.
3: **while** not convergence **do**
4: **In source network $\mathcal{G}^S$:**
5: Learn local node embedding Z_l^S by local GCN G_l using Eq.(3)
6: Learn global node embedding Z_g^S by global GCN G_g using Eq.(4)
7: Learn private node embedding Z_{pr}^S by source encoder E_S using Eq.(5)-(7)
8: Learn shared node embedding Z_{sh}^S by shared encoder E_H using Eq.(5)-(7)
9: Learn reconstructed graph adjacency matrix $\hat{A^S}$ by source decoder R_S
10: **In target network $\mathcal{G}^T$:**
11: Learn local node embedding Z_l^T by local GCN G_l using Eq.(3)
12: Learn global node embedding Z_g^T by global GCN G_g using Eq.(4)
13: Learn private node embedding Z_{pr}^T by target encoder E_T using Eq.(5)-(7)
14: Learn shared node embedding Z_{sh}^T by shared encoder E_H using Eq.(5)-(7)
15: Learn reconstructed graph adjacency matrix $\hat{A^T}$ by target decoder R_T
16: Compute difference loss $\mathcal{L}_{diff}$ using Eq (1).
17: Compute reconstruction loss $\mathcal{L}_{recon}$ using Eq (2).
18: Compute node classification loss $\mathcal{L}_{cls}$ using Eq (8).
19: Compute target entropy loss $\mathcal{L}_{ent}$ using Eq (9).
20: Compute domain adversarial loss $\mathcal{L}_{adv}$ across networks using Eq (10)-(11).
21: Back-propagate loss gradient using Eq (12).
22: Update weights of all used models.
23: **end while**

Firstly, we compute the PPMI matrices [47] P^S and P^T for the source and target network (Line 1-2). Then we employ a dual GCN framework (G_l and G_g) to capture the local and global consistency relationship of each network (Line 5-6 and Line 11-12). Also, we adopt private and shared encoders to learn private and shared node representations for source and target networks (Line 7-8 and Line 13-14). Two decoders for source and target network are employed to reconstruct the graph adjacency matrix (Line 9 and Line 15). Next, the difference loss, reconstruction loss, node classification loss, entropy loss, and domain classification loss are computed by learned node representations (Line 16-20). Finally, the trainable parameters of ASN are updated by Adam optimizer (Line 21-22).

After the model finally converges or a maximum training iteration has been reached, we can employ the optimized encoders to generate comprehensive node representations for both networks. Then we employ the optimized node classifier to predict node labels for the target network.

5 EXPERIMENTS

In this section, we describe datasets, experimental setting, and performance analysis. The source code of ASN is provided in https://github.com/yuntaodu/ASN.

Datasets	#Nodes	#Edges	#Attributes	#Union Attributes	#Labels
DBLPv7	5484	8130	4412	6775	5
ACMv9	9360	15602	5571	6775	5
Citationv1	8935	15113	5379	6775	5

Table 1: Statistics of the network datasets

5.1 Datasets

In the experiments, we performed feature extraction on the original AMiner datasets [37](http://www.arnetminer.org/data), and obtained three citation network dataset. The details of the datasets are displayed in Table 1. For each dataset, we extracted the papers published in different periods, *i.e.,* DBLPv7 (between years 2004 and 2008), ACMv9 (between years 2000 and 2010) and Citationv1 (before the year 2008). It is obvious that they inherently have varied data distributions.

We consider the citation networks as undirected networks with each node representing a paper and each edge indicating a citation relation between two papers. Each paper belongs to one of the following five categories according to its research topics, including "Databases", "Artificial Intelligence", "Computer Vision", "Information Security", and "Networking". Six cross-network classification tasks are conducted, including C→D, A→D, D→C, A→C, D→A, and C→A, where D, C, A represent DBLPv7, Citationv1 and ACMv9, respectively.

5.2 Compared Methods

What's more, we select compared methods from related research lines, including the following four types:

(1) Single Network Embedding: ANRL [46] designs a neighbor enhancement autoencoder and an attribute-aware skip-gram model to learn the node representations. **LANE** [16] constructs the similarity matrices of node attributes, network structure and labels, and projects them into a unified embedding space.

(2) Traditional Domain Adaptation: MMD [13] is used to reduce the distribution discrepancy between the source domain and the target domain based on the maximum mean discrepancy. **DANN** [10] designs a GRL between feature extractor and domain discriminator and integrates an adversarial mechanism into domain adaptation.

(3) GCN-based Node Classification: GCN [19] is a classical model for node classification, which uses Laplace matrix, eigendecomposition and Fourier transform to convolve on graph network. **GAT** [39] uses an attention mechanism to determine the importance of each neighbor node to the central node when aggregating the node's neighbor information. **GraphSAGE** [14] aims to learn node representation by sampling and aggregating node features from local neighbors, rather than training a separate embedding for each node.

(4) Cross-Network Node Classification: Source-Only is trained only on the source domain with Local GCN and Global GCN. We evaluate the performance of the target domain without domain adaptation. **UDAGCN** [43] adopts a dual graph convolutional network to jointly exploits local and global consistency for feature aggregation. **AdaGCN** [6] uses adversarial domain adaptation and GCN to jointly model network structures and node attributes for learning representations. UDAGCN, AdaGCN, and ASN all utilize GCN to extract features, while CDNE [35], and ACDNE [34] are based on their self-designed models to extract node features, which is not the point we focus on. Therefore, ASN only compares with UDAGCN and AdaGCN.

5.3 Experimental Settings

We follow the standard evaluation protocols for cross-network node classification task [43]. All the labeled source nodes and unlabeled target nodes are used for model training for cross-network node classification methods and traditional domain adaptation methods. For single network embedding and GCN-based node classification methods, the model is trained in the source network and tested on the target network. The average classification accuracy is calculated after 10 times repetitions on six domain adaptation tasks. We ran all the comparison methods on the datasets we used.

Task	lr	λ_d	λ_r	λ_f
A →D	3×10^{-2}	5×10^{-1}	10^{-1}	10^{-4}
A →C	2×10^{-2}	5×10^{-1}	10^{-3}	10^{-4}
D →A	2×10^{-2}	5×10^{-1}	10^{-1}	10^{-4}
D →C	3×10^{-2}	5×10^{-1}	10^{-1}	10^{-4}
C →A	3×10^{-2}	5×10^{-1}	1	10^{-2}
C →D	2×10^{-2}	5×10^{-1}	10^{-1}	10^{-2}

Table 2: Optimal hyperparameters for each cross-network node classification task where A, D and C represent ACMv9, DBLPv7 and Citationv1, respectively.

In the experiments, we implement our proposed methods using Pytorch [28] and train the model with Adam optimizer [17]. We train the model for 600 epochs, and to prevent overfitting, we perform a learning rate decaying by assigning the weight decay value as 5×10^{-4} to stabilize training.

In both source and target networks, for all encoders, we all use GCNs that contain two hidden layers to learn node representation. The hidden dimensionality for the GCN layer is set as $d_1^G = 128, d_2^G = 16$. And the dropout rate for each layer is set to 0.5. For a fair comparison, the same dimensionality is also set for other compared methods based on GCN.

For local GCN G_l, we input the feature matrix X and the adjacency matrix A into the first layer GCN with dimension 128, and then input the obtained output into the second layer GCN with dimension 16 to obtain the final local node representation. For global GCN, we first get the PPMI matrix P when measuring the global topological proximities between nodes. Then we input the feature matrix X and the PPMI matrix P into the first layer GCN with dimension 128, and then input the obtained output into the second layer GCN with dimension 16 to obtain the final global node representation.

For the size of k in PPMI matrix, if k is too small, only the close neighbors can be extracted, eventually the PPMI matrix tends to be similar to the adjacency matrix. If k is too large, the neighbors far away from the current node will be captured, which causes over-smoothing. Finally, the representation of all nodes tends to be the same, leading to the over-smoothing problem. In our paper, we set k by searching $k \in \{2, 3, 4, 5\}$ and eventually we set $k = 3$ to get the final PPMI matrix.

The node classifier only contains one hidden layer with 16 units. For the domain discriminator, it is constructed with two hidden layers with dimensionalities as $d_1^D = 16, d_2^D = 10$ and the dropout rate is set to 0.3. Besides, for the domain private decoder, we use the inner-product as a decoder to reconstruct the adjacent matrix of the original graph. The learning rates for the models in our method will be described in the next subsection. We train the model on V100 GPU, and it costs 20 minutes to train the model.

5.4 Hyperparameters Setting

There are five hyperparameters in our model, namely the learning rate lr, weight balance parameters $\lambda_d, \lambda_r, \lambda_f, \lambda_e$ which represent the trade-off parameters to balance the importances of domain adversarial loss $\mathcal{L}_{adv}$, reconstruction loss $\mathcal{L}_{recon}$, difference loss $\mathcal{L}_{diff}$ and entropy loss $\mathcal{L}_{ent}$.

For λ_e, we set it to $\frac{epoch}{max_{epoch}} * 0.01$, where $epoch$ represents the number of current iteration and max_{epoch} represents the maximum training iteration. For other hyperparameters, we evaluate all tasks through grid search on the hyperparameter space with a wide range of values for regularization parameters $\lambda_d \in [0, 1]$, $\lambda_r \in [0, 1]$ and $\lambda_f \in [0, 1]$. The optimal

Compared Methods	A → D	C → D	D → A	C → A	A → C	D → C	Average
DeepWalk	0.4078	0.3130	0.3880	0.3974	0.4260	0.4847	0.4028
LANE	0.5706	0.5857	0.5362	0.5627	0.5802	0.5695	0.5675
ANRL	0.6648	0.6603	0.6308	0.6446	0.6841	0.6664	0.6585
GCN	0.6435	0.7018	0.6172	0.7036	0.7448	0.7049	0.6860
GAT	0.6800	0.6479	0.5948	0.675	0.714	0.7097	0.6702
GraphSAGE	0.7094	0.7154	0.6521	0.6912	0.7303	0.7117	0.7017
MMD	0.6745	0.7012	0.6379	0.6865	0.7048	0.6910	0.6827
DANN	0.6874	0.7128	0.6272	0.6704	0.7222	0.7019	0.6870
Source-only	0.6435	0.7018	0.6172	0.7036	0.7448	0.7049	0.6860
UDAGCN	0.6832	0.7004	0.6761	0.6885	0.7192	0.7351	0.7004
AdaGCN	0.6932	0.7089	0.6792	0.6830	0.6932	0.7777	0.7059
ASN	**0.7604**	**0.7797**	**0.7032**	**0.7470**	**0.8112**	**0.7827**	**0.7641**

Table 3: Node classification accuracy comparisons on six cross-domain tasks.

hyperparameter settings that achieve the best result for each task are shown in Table 2. We can observe that extensive hyperparameter-tuning is not necessary to obtain state-of-the-art performance. The hyperparameters are restricted in a certain range, *i.e.* $lr = \{2 \times 10^{-2}, 3 \times 10^{-2}\}$, $\lambda_d = 5 \times 10^{-1}$, $\lambda_r = \{10^{-3}, 10^{-1}, 1\}$, $\lambda_f = \{10^{-4}, 10^{-2}\}$.

5.5 Performance Analysis

The accuracy of different methods on cross-domain node classification tasks is listed in Table 3. It can be easily observed that our method ASN outperforms all baselines in six tasks. From the results, we have the following observations:

(1) Single Network Embedding VS. Cross-network Node Classification. DeepWalk obtains the worst performance among all the baselines since it only utilizes the network structure information. LANE and ANRL perform better as they fully consider the network structure, node features and labels. However, ASN still outperforms them by a large margin. This is because single network embedding only utilizes source network but does not consider the distribution discrepancy across networks, which shows the necessity of reducing the domain discrepancy in cross-network node classification.

(2) Traditional GCN-based Methods VS. GCN-based Methods with Domain Adaptation. As we can see, GCN-based methods (GCN, GAT and GraphSAGE) have better performances than single network embedding, which shows the powerful advantages of graph convolutional neural networks in node embedding. However, ASN still performs much better than them, this is because we make full use of domain adaptation to better guide the target network to perform node classification tasks. Obviously, domain adaptation is critical for cross-network node classification tasks.

(3) Traditional Domain Adaptation VS. Cross-network Adaptation. MMD and DANN perform better than single network embedding which indicates that considering the domain discrepancy across networks is crucial. However, they still perform worse than cross-network methods. It suggests that traditional domain adaptation methods cannot handle cross-network node classification tasks due to their inability to leverage network structure information. Thus, considering complicated network structure relationships between nodes is essential for node classification in cross-network scenarios.

(4) Adversarial Domain Adaptation for Cross-Network Node Classification. Compared with the source-only method, UDAGCN and AdaGCN perform better in most cases as they adopt a powerful adversarial domain adaptation approach to conduct cross-network node classification. The outperformance of ASN over the source-only method also verifies this.

Components	D → A	C →D
Local GCN	0.4957	0.5383
Global GCN	0.4250	0.4442
Local + global (without attention)	0.6430	0.7314
Local + global (with attention)	**0.7032**	**0.7797**

Table 4: Ablation study for components

(5) Private and Shared Encoding for Cross-Network Node Classification. Compared with state-of-the-art methods (UDAGCN and AdaGCN) in cross-network scenarios, our model ASN can reach the best performance and beat all baselines on six cross-network tasks. It indicates that our proposed ASN can better capture the underlying representations of the domain-private and domain-shared features. Thus by explicitly separating node representations private to each domain and shared between both domains is of great importance to learn the domain-invariant and label discriminative node representations, which is beneficial for node classification.

5.6 Ablation Study

To investigate the contributions of different components and losses in our proposed model, we conducted ablation studies on two representative tasks.

Ablation Study for Components As shown in Table 4, we conducted an ablation study to investigate the contributions of different components. The worse performances on the models with only local GCN and only global GCN confirm the superiority of dual GCN structure, which combines the local and global consistency information to learn a comprehensive node representation. Besides, the attention layer used to balance the importance between local and global consistency is also necessary for the node representation learning, thus helping the node classification task on the target network.

Ablation Study for Losses Using all components (including local GCN, global GCN and the attention layer), we conduct ablation study on losses. As shown in Table 5, every single loss has its indispensable contribution. $\mathcal{L}_{cls}$+$\mathcal{L}_{ent}$ performs better than $\mathcal{L}_{cls}$ *(source-only)* which reflects utilizing target data by minimizing entropy loss is useful. The improvement of $\mathcal{L}_{cls}$+$\mathcal{L}_{ent}$+$\mathcal{L}_{adv}$ shows that reducing domain discrepancy by adversarial

Loss	D →A	C → D
$\mathcal{L}_{cls}$ (source-only)	0.6172	0.7018
$\mathcal{L}_{cls}+\mathcal{L}_{ent}$	0.6242	0.7160
$\mathcal{L}_{cls}+\mathcal{L}_{ent}+\mathcal{L}_{adv}$	0.6675	0.7235
$\mathcal{L}_{cls}+\mathcal{L}_{ent}+\mathcal{L}_{adv}+\mathcal{L}_{diff}$	0.6841	0.7412
$\mathcal{L}_{cls}+\mathcal{L}_{ent}+\mathcal{L}_{adv}+\mathcal{L}_{recon}$	0.6839	0.7425
Ours (all)	**0.7032**	**0.7797**

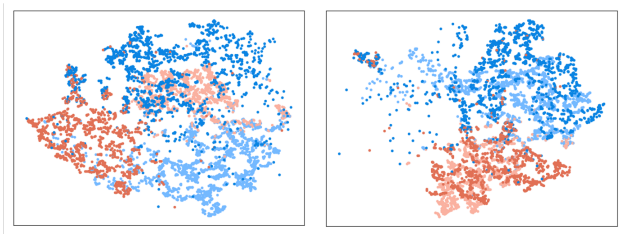
Table 5: Ablation study for losses

domain adaptation is important for node classification in the target network. By integrating *difference loss* $\mathcal{L}_{diff}$ into the base model can obtain much better performance. Thus the soft subspace orthogonality constraint between private and shared encoder is beneficial. The improvement by using *reconstruction loss* $\mathcal{L}_{recon}$ proves that reconstruction between encoder and decoder is effective. Finally, a combination of all losses can achieve the best results.

5.7 Visualization

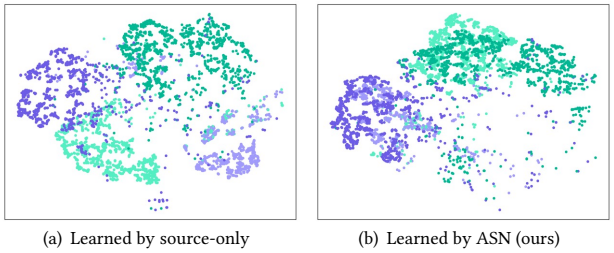
To show feature transferability, as shown in Figure 3, we employ t-SNE toolkit [21] to visualize node representations in a 2-D space.

For clear presentation, as previous method [6], we only visualize nodes from two classes in the task of DBLPv7→Citationv1. As shown in Figure 3, the dark *red* and *blue* points represent nodes of "Databases" and "Artificial Intelligence" respectively from the source network, while the light *red* and *blue* points are from the target network. In Figure 4, the dark *green* and *purple* points represent nodes of "Computer vision" and "Data mining" from the source network, while the light points are from the target network.



(a) Learned by source-only (b) Learned by ASN (ours)

Figure 3: Visualization of node representations learned by source-only model and ASN from DBLPv7→Citationv1 in class "Databases" and "Artificial Intelligence".



(a) Learned by source-only (b) Learned by ASN (ours)

Figure 4: Visualization of node representations learned by source-only model and ASN from DBLPv7→Citationv1 in class "Computer vision" and "Data mining".

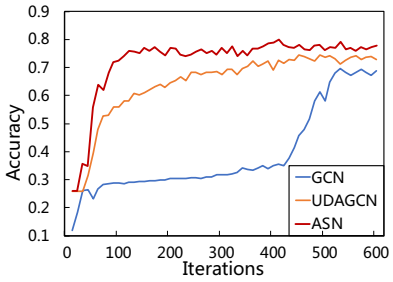


Figure 5: Convergence of different methods

Baselines	GCN	UDAGCN	ASN
# Parameters	1743014	1739078	5229064
Runing time (ms / iteration)	72.8	100.2	169.8

Table 6: Comparisons on computational complexity

Specifically, it can be observed that the source-only model can not address the distribution shift across different networks. As shown in Figure 3(a), nodes from different domains and categories, *e.g.* red and blue points, are mixed together. And in Figure 4(a), nodes from different domains, *e.g.* dark green and light green points, are far from each other.

In contrast, from Figure 3(b) and Figure 4(b) which show the node representations learned by our model ASN, we can observe that nodes from the same category in different domains are clustered together. For instance, in Figure 3(b), dark red points cluster with light red points, and dark blue points cluster with light blue points. Since nodes from the same categories from different networks are well clustered together, it indicates that adversarial domain adaptation can effectively reduce the distribution divergence across networks. Moreover, the boundary between the two clusters are clear, which reflects that the learned node representations are discriminative. Thus, the node representations learned by ASN are label-discriminative and network-invariant as we expected.

5.8 Convergence verification

We conduct an experiment to test the parameter numbers and running time as shown in Table 6. The results show that although the parameter numbers of ASN are three times than others, it takes less than twice for the running time. And we also provide the converge curves of task A → C in Figure 5. As we can see, the convergence speed and accuracy of ASN are superior to others and ASN only takes 101 iterations to reach convergence, while UDAGCN takes 400 iterations.

6 CONCLUSION

In this paper, we study the problem of cross-network node classification by adversarial domain adaptation. To address the incapability of existing traditional domain adaptation algorithms applied to network structure data, we propose a novel model named *Adversarial Separation Network* (ASN) to learn effective node representations for cross-network node classification. Our proposed model contains a shared encoder between the source and target networks and two private encoders and decoders which are unique to each network. We explicitly separate the domain-shared features from the domain-private features to learn more efficient node representations. Also, to better exploit both local and global information of the networks, we adopt a dual GCN framework (including a local GCN and a global GCN) with an attention layer to learn more comprehensive node representations. Extensive experiments on three real-world datasets show that compared with existing algorithms, our proposed method ASN achieves state-of-the-art performance among all cross-network node classification tasks.

7 ACKNOWLEDGMENTS

This paper is supported by the National Key Research and Development Program of China (Grant No. 2018YFB1403400), the National Natural Science Foundation of China (Grant No. 61876080), the Key Research and Development Program of Jiangsu(Grant No. BE2019105), the Collaborative Innovation Center of Novel Software Technology and Industrialization at Nanjing University.

REFERENCES

[1] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. 2011. Node classification in social networks. In Social network data analytics. Springer, 115–148.
[2] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain Separation Networks. Advances in Neural Information Processing Systems (2016), 343–351.
[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep neural networks for learning graph representations. AAAI Conference on Artificial Intelligence 16 (2016), 1145–1152.
[4] Francine Chen and Yan-Ying Chen. 2019. Adversarial Domain Adaptation Using Artificial Titles for Abstractive Title Generation. In 57th Annual Meeting of the Association for Computational Linguistics. 2197–2203.
[5] William Cukierski, Benjamin Hamner, and Bo Yang. 2011. Graph-based features for supervised link prediction. In The 2011 International Joint Conference on Neural Networks. IEEE, 1237–1244.
[6] Quanyu Dai, X. Shen, X. Wu, and D. Wang. 2019. Network Transfer Learning via Adversarial Domain Adaptation with Graph Convolution. ArXiv abs/1909.01541 (2019).
[7] Chris HQ Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D Simon. 2001. A min-max cut algorithm for graph partitioning and data clustering. In Proceedings 2001 IEEE international conference on data mining. IEEE, 107–114.
[8] Meng Fang, Jie Yin, and Xingquan Zhu. 2013. Transfer learning across networks for collective classification. In IEEE 13th International Conference on Data Mining. IEEE, 161–170.
[9] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In International conference on machine learning. PMLR, 1180–1189.
[10] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Franois Laviolette, Mario Marchand, and Victor Lempitsky. 2017. Domain-Adversarial Training of Neural Networks. Journal of Machine Learning Research 17, 1 (2017), 2096–2030.
[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. Advances in Neural Information Processing Systems 3 (2014), 2672–2680.
[12] Yves Grandvalet and Yoshua Bengio. 2004. Semi-supervised learning by entropy minimization. Advances in neural information processing systems 17 (2004), 529–536.
[13] A. Gretton, K. M. Borgwardt, MJ Rasch, B. Schlkopf, and A.J. Smola. 2007. A Kernel Method for the Two-Sample-Problem. Advances in Neural Information Processing Systems 1 (2007).
[14] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. arXiv preprint arXiv:1706.02216 (2017).
[15] Fenyu Hu, Yanqiao Zhu, Shu Wu, Liang Wang, and Tieniu Tan. 2019. Hierarchical graph convolutional networks for semi-supervised node classification. arXiv preprint arXiv:1902.06667 (2019).
[16] X Huang, J Li, and X Hu. 2017. Label informed attributed network embedding. 10th ACM International Conference on Web Search and Data Mining (2017), 731–739.
[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations (ICLR) (2015).
[18] Thomas Kipf and M. Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. International Conference on Learning Representations (ICLR) (2017).
[19] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. International Conference on Learning Representations (ICLR) (2017).
[20] A. Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. Commun. ACM 60 (2012), 84 – 90.
[21] Van Der Maaten Laurens and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. Journal of Machine Learning Research 9, 2605 (2008), 2579–2605.
[22] Geng Li, Murat Semerci, Bülent Yener, and Mohammed J Zaki. 2012. Effective graph classification based on topological and label attributes. Statistical Analysis and Data Mining: The ASA Data Science Journal 5, 4 (2012), 265–283.
[23] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. Journal of the American society for information science and technology 58, 7 (2007), 1019–1031.
[24] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In International conference on machine learning. PMLR, 97–105.
[25] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. 2018. Conditional adversarial domain adaptation. In Advances in neural information processing systems. 1640–1650.
[26] Mingsheng Long, Han Zhu, J. Wang, and Michael I. Jordan. 2017. Deep Transfer Learning with Joint Adaptation Networks. In ICML.
[27] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering (2010).
[28] Adam Paszke, S. Gross, Francisco Massa, A. Lerer, and etl. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems.
[29] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. 2018. Multi-adversarial domain adaptation. In AAAI Conference on Artificial Intelligence, Vol. 32.
[30] B Perozzi, R Al-Rfou, and S Skiena. 2014. DeepWalk: Online learning of social representations. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2014), 701–710.
[31] Ariadna Quattoni, Michael Collins, and Trevor Darrell. 2008. Transfer learning for image classification with sparse prototype representations. In 2008 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 1–8.
[32] E. Sanchez, M. Serrurier, and M. Ortner. 2020. Learning Disentangled Representations via Mutual Information Estimation. In ECCV.
[33] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. 2018. Wasserstein distance guided representation learning for domain adaptation. In AAAI Conference on Artificial Intelligence, Vol. 32.
[34] Xiao Shen, Quanyu Dai, Fu Lai Chung, Wei Lu, and Kup Sze Choi. 2020. Adversarial Deep Network Embedding for Cross-network Node Classification. AAAI Conference on Artificial Intelligence (2020).
[35] Xiao Shen, Quanyu Dai, Sitong Mao, Fu-Lai Chung, and Kup-Sze Choi. 2021. Network Together: Node Classification via Cross-Network Deep Network Embedding. IEEE Transactions on Neural Networks and Learning Systems 32, 5 (2021), 1935–1948.
[36] Baochen Sun and Kate Saenko. 2016. Deep CORAL: Correlation Alignment for Deep Domain Adaptation. In ECCV Workshops.
[37] Jie Tang, J Zhang, L Yao, J Li, L Zhang, and Z Su. 2008. Arnetminer: extraction and mining of academic social networks. 14th ACM SIGKDD international conference on Knowledge discovery and data mining (2008).
[38] E Tzeng, J Hoffman, K Saenko, and E Darrell. 2017. Adversarial Discriminative Domain Adaptation. IEEE Conference on Computer Vision and Pattern Recognition (2017), 7167–7176.
[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 (2017).
[40] Dong Wang and Thomas Fang Zheng. 2015. Transfer learning for speech and language processing. In Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA). IEEE, 1225–1237.
[41] Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. 2020. Direct Multi-hop Attention based Graph Neural Network. arXiv preprint arXiv:2009.14332 (2020).
[42] Jindong Wang, Wenjie Feng, Yiqiang Chen, Han Yu, Meiyu Huang, and Philip S Yu. 2018. Visual domain adaptation with manifold embedded distribution alignment. In 26th ACM international conference on Multimedia. 402–410.
[43] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. 2020. Unsupervised Domain Adaptive Graph Convolutional Networks. In The Web Conference 2020. 1457–1467.
[44] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings.. In International conference on machine learning. 40–48.
[45] W. Zellinger, Thomas Grubinger, E. Lughofer, T. Natschläger, and Susanne Saminger-Platz. 2017. Central Moment Discrepancy (CMD) for Domain-Invariant Representation Learning. International Conference on Learning Representations (ICLR) (2017).
[46] Z Zhang, H Yang, and J Bu. 2018. ANRL: Attributed Network Representation Learning via Deep Neural Networks. International Joint Conference on Artificial Intelligence 18 (2018), 3155–3161.
[47] Chenyi Zhuang and Qiang Ma. 2018. Dual graph convolutional networks for graph-based semi-supervised classification. In The World Wide Web Conference. 499–508.