Unsupervised Domain Adaptive Graph Convolutional Networks

Man Wu
Dept. of Computer & Electrical Engineering and Computer Science, Florida Atlantic University
mwu2019@fau.edu

Shirui Pan*
Faculty of Information Technology, Monash University
shirui.pan@monash.edu

Chuan Zhou
AMSS, Chinese Academy of Sciences
School of Cyber Security, University of Chinese Academy of Sciences
zhouchuan@amss.ac.cn

Xiaojun Chang
Faculty of Information Technology, Monash University
cxj273@gmail.com

Xingquan Zhu
Dept. of Computer & Electrical Engineering and Computer Science, Florida Atlantic University
xzhu3@fau.edu

ABSTRACT

Graph convolutional networks (GCNs) have achieved impressive success in many graph related analytics tasks. However, most GCNs only work in a single domain (graph) incapable of transferring knowledge from/to other domains (graphs), due to the challenges in both graph representation learning and domain adaptation over graph structures. In this paper, we present a novel approach, unsupervised domain adaptive graph convolutional networks (UDA-GCN), for domain adaptation learning for graphs. To enable effective graph representation learning, we first develop a dual graph convolutional network component, which jointly exploits local and global consistency for feature aggregation. An attention mechanism is further used to produce a unified representation for each node in different graphs. To facilitate knowledge transfer between graphs, we propose a domain adaptive learning module to optimize three different loss functions, namely source classifier loss, domain classifier loss, and target classifier loss as a whole, thus our model can differentiate class labels in the source domain, samples from different domains, the class labels from the target domain, respectively. Experimental results on real-world datasets in the node classification task validate the performance of our method, compared to state-of-the-art graph neural network algorithms.

CCS CONCEPTS

• **Information systems** → *Social networks.*

KEYWORDS

Domain Adaptation, graph convolutional networks, node classification

ACM Reference Format:
Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. 2020. Unsupervised Domain Adaptive Graph Convolutional Networks. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380219

*Corresponding Author.

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380219

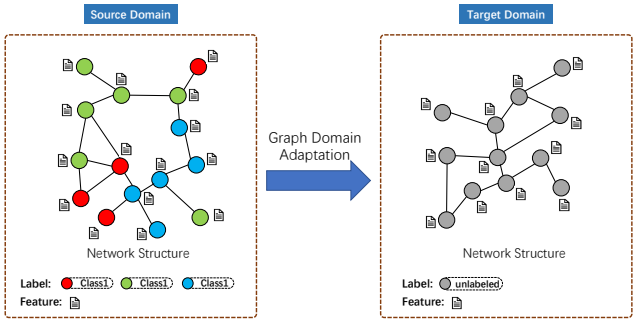


Figure 1: An example of unsupervised graph domain adaptation. Given a graph in a target domain where all nodes are unlabeled, domain adaptation aims to transfer knowledge from the graph of an existing domain where nodes are labeled to classify nodes in the target domain.

1 INTRODUCTION

Node classification is an important yet challenging task in various network[1] applications, including social networks [2], protein-protein interaction networks [11], and citation networks [18]. Many research efforts have been made in the past decade to develop reliable and efficient methods for node classification tasks [44]. However, most of existing methods mainly focus on graph representations for nodes from a single graph, and they have largely overlooked the generalisation of the classification model to a completely new graph. As a result, when the new graph is collected, even it is very similar to an existing graph, we have to relabel the nodes in the graph and rebuild a classifier model for the node classification task. The ineffectiveness of existing learning frameworks for graph data calls for transferable models that enable knowledge adapted from a source graph to a target graph.

Domain adaptation, which aims to support transfer learning from a source domain with sufficient label information to a target domain with plenty of unlabeled data by minimizing their domain

[1]Networks and graphs are interchangeable terms in the paper when referred to the graph structure data.

discrepancy, has already attracted a lot of interests from the fields of Computer Vision [20][21] and Natural Language Processing [16][7]. However, applying domain adaptation to network analysis like classifying nodes across networks has not been sufficiently investigated. Given a source network having fully labeled nodes and a target network without any labeled data, the objective of unsupervised graph domain adaptation is to take advantage of the rich labeled information from the source network to help build an accurate node classifier for the target network. An example of unsupervised graph domain adaptation is illustrated in Figure 1.

Currently, most researches of domain adaptation concentrate on CV and NLP fields, which can not be directly applied on node classification problems. The reasons are twofold: First, these methods are usually designed for CV and NLP tasks, where samples (e.g. images and sequences) are independent and identically distributed, resulting in little requirement for model rotational invariance. However, network structured data, where nodes are connected with edges representing their relations, require models with rotational invariance because of the phenomenon known as graph isomorphism. Therefore, existing methods can not model network structural information, which is the core of node classification. Second, most existing domain adaptation models learn discriminative representations in a supervised manner, in which the value of loss function is only associated with each single sample's absolute position in their feature space. Network embedding for node classification, alternatively, usually aims to learn multipurpose representation in an unsupervised manner by preserving the relative position of all node pairs, resulting in increased difficulty in optimization.

Recently, there are some attempts to apply domain adaptation ideas for graph-structured data. The CDNE [31] algorithm learns transferable node embedding for cross network learning tasks by minimizing the maximum mean discrepancy (MMD) loss. However, it cannot jointly model network structures and node attributes, which limits its modeling capacity. To utilize the network structure for cross-network node classification, the AdaGCN algorithm [6] uses graph convolutional networks as a feature extractor to learn node representations, and utilize adversarial learning strategy to learn domain invariant node representation. Though it seems reasonable to exploit GCNs and adversarial learning jointly to enhance the performance of cross domain node classification for graph-structured data, these existing methods still cannot deal with three levels of challenges effectively below.

(1) In data structure level, many existing methods, graph convolutional networks (GCNs) [18] in particular, only consider the direct (the local consistency) neighbour nodes for knowledge embedding, the global consistency information has not been well investigated yet. In practice, the global consistency relationship is vitally important. For instance, in a real social network, each individual is a member of several communities and can be influenced by her/his neighborhoods with different distances around her/him, ranging from local consistency relationship (e.g. families, friends), to global consistency relationship (e.g. society, nation states). Thus global consistency relationship should also be exploited to obtain a comprehensive representation of the node for graph learning collaboratively.
(2) In representation learning level, most existing graph learning methods lean the node representation based on the local consistency relationship. However, as mentioned above, the global consistency relationship cannot be neglected. Thus, in our scenario, how to combine the local and global relationship to capture a comprehensive representation of the node is vitally important. Ideally, this should be done within the end-to-end learning framework.
(3) In domain adaptive learning level, the existing graph domain adaptations only utilize domain labels to help train a domain classifier to model the global distribution of source and target domains, where a Gradient Reversal Layer [13] was proposed to train domain classifier to discern whether a sample is from the source domain or the target domain. Meanwhile, the source domain data was utilized to train a source classifier for source domain classification learning. However, they do not consider the target domain and ignore the semantic information contained in target domain samples, which is the key component in cross-domain learning. Therefore, the source domain information, domain information and target domain information should be considered collaboratively to learn the domain-invariant and semantic representations.

To address the above limitations, we propose an Unsupervised Domain Adaptive Graph Convolutional Networks (UDA-GCN) for cross-domain node classification by modeling the local and global consistency relationship of each graph, and combining source information, domain information and target information into a unified deep model. Our approach consists of three key components: (1) in data structure level, the local and global consistency relationship of each graph are utilized to assist in the training of node embedding module. (2) in representation learning level, an inter-graphed based attention mechanism is proposed to combine the local and global relationship for a comprehensive node representation of each domain. (3) in domain adaptive learning level, we advocate a domain adaptive learning approach to exploit the source information, domain information and target information jointly, which can be utilized to learn domain-invariant and semantic representations effectively to reduce the domain discrepancy for cross-domain node classification. Empirical results on three public real datasets demonstrate that UDA-GCN outperforms the state-of-the-art cross-domain node classification methods. Our contributions can be summarized as follows:

- We present a novel unsupervised graph domain adaptation problem, and propose an effective graph convolutional network algorithm to solve it.
- We propose a novel method to integrate local and global consistency with an attention mechanism to learn effective node embedding across networks.
- We design a new way to exploit source information and target information with different loss functions, so that domain-invariant and semantic representations can be effectively learned to reduce the domain discrepancy for cross-domain node classification.
- We evaluate our method on real-world datasets, and the results demonstrate that the proposed model outperforms the baseline methods.

2 RELATED WORK

Our work is closely related to graph neural networks and cross domain classification. We briefly review these works in this section.

2.1 Graph Neural Networks

Network node representation generally aims to map nodes with higher proximities in a network closer to each other in the low-dimensional latent space, which is based on network topology structure only or with side information. For topology structure only embedding methods, most of existing works focused on preserving network structures and properties in embedding vectors [28] [33] [14]. Line [33] and SDNE [39] seek to preserve the first-order and second-order proximities between nodes based on the first-order and second-order neighbors. DeepWalk [28] employs the random walk sampling strategy to generate the neighborhood of each node. Then, some deep learning approaches [4, 30] have been employed to learn more similar feature representations for nodes which can more easily reach each other within K steps. Aside from topology structure only methods, many approaches are proposed to incorporate side information such as node features [25] [43] [46].

Recently, graph neural networks have attracted attention all around the world, which are designed to use deep learning architectures on graph-structured data [40, 42, 47, 48]. Many solutions are proposed to generalize well-established neural network models that work on regular grid structure to deal with graphs with arbitrary structures [24, 38, 41]. GCN [18] is a deep convolutional learning paradigm for graph-structured data which integrates local node features and graph topology structure in convolutional layers. GAT [37] improves GCN by leveraging attention mechanism to aggregate features from the neighbors of a node with discrimination. However, most of existing methods mainly focus on learning representations for nodes from a single network. As a result, when transferring models across networks to handle the same task, they may suffer from the embedding space drift [8] and the embedding distribution discrepancy [35]. Moreover, most of these methods can only utilize the direct neighbourhood information (the local consistency relationship), while the high-order proximities which can capture the global consistency information are always be neglected [3].

Graph domain adaptation v.s. Inductive Learning. It is worthy to note that there are graph neural networks in recent years on learning inductive representation for node classification. GraphSAGE [15], for example, presents different aggregation methods for feature extraction and can be applied to learn the embedding for nodes which are not in the training process. Unlike the inductive learning methods which only use the training set to train the model (the training data and testing data are separate), the domain adaptation approach is to feed the training data and testing data together into a network.

Different from the previous approaches, we focus on domain adaptation to implement the node classification task across two networks. Furthermore, we employ a dual graph convolutional networks to capture the local and global consistency relationship of each graph for node representation learning.

2.2 Cross-Domain classification

Domain adaptation is a subtopic of transfer learning, which aims to learn machine learning models transferable on different but relevant domains sharing same label space [23]. Many approaches are proposed for cross-domain classification, which can be roughly categorized into four groups: (1) Instance re-weighting approaches aim to identify the training samples in the source domain that are most relevant to the target domain by instance re-weighting and importance sampling. Then the re-weighted source instances are used for training a target domain model [16]. (2) Co-training methods bridge the gap between the source domain and the target domain by slowly adding target features and the most reliable examples of the current algorithm in the training set [5]. (3) Kernel methods explore multiple kernels to induce an optimal learning space and learn a kernel function and a robust classifier by minimizing the distribution mismatch between the labeled and unlabeled samples from the source and target domains [9]. (4) Feature representation based methods are designed to map different domains into a common shared space and contract their feature distributions as close as possible [7, 20, 32, 50]. Among them, deep feature representation based methods have attracted a lot of attention in recent years due to its effectiveness. They can be categoried into three branches, i.e., discrepancy-based methods [20][36], reconstruction-based methods [49][17], and adversarial-based methods [13][35][27]. For cross-domain learning, many methods use an adversarial objective to reduce domain discrepancy [12, 20]. Among which, the domain adversarial neural network (DANN) [13] learns domain invariant features by a minimax game between the domain classifier and the feature extractor, using a gradient reversal layer to back-propagate the gradients computed from the domain classifier.

Recently, domain adaptation have been utilized for graph-structured data [6, 31, 45]. CDNE [31] learns transferable node embeddings for cross network learning tasks by minimizing the maximum mean discrepancy (MMD) loss. However, it cannot jointly model network structures and node attributes, which might limit its modeling capacity. To utilize the network structure for cross-network node classification, some researches [6, 45] attempt to use graph convolutional networks as feature extractor to learn node representations, and utilize adversarial learning strategy to learn domain invariant node representations, which obtain the promising performance. However, the above methods only use the GCN which considers the direct (the local consistency) neighbour nodes for knowledge embedding, and neglect the global consistency information of network for cross domain node classification.

In this paper, we propose an end-to-end Unsupervised Domain Adaptive Graph Convolutional Networks (UDA-GCN) for cross-domain node classification by jointly modeling local and global consistency relations of each graph, domain information, source domain information and target domain information as a unified learning framework.

3 PROBLEM DEFINITION AND OVERALL FRAMEWORK

This section defines the problem to be addressed and introduces notations used throughout the paper as summarized in Table 1. Then we present the overall framework for the problem.

Table 1: Notations.

Notation	Description
$G = (V, E, X, Y)$	a weighted attributed graph
V	node set of G
A	the adjacency matrix of G
X	feature matrix of G
Y	the label matrix of G
N	the number of nodes in G
C	the number of node categories
$G^s = (V^s, E^s, X^s, Y^s)$	a fully labeled source graph
$G^t = (V^t, E^t, X^t)$	a completely unlabeled target graph
V^s	a set of all labeled nodes of G^s
E^s	a set of edges of G^s
N^s	the number of nodes in G^s
$Y^s \in \mathbb{R}^{N^s \times C}$	a label matrix of G^s
V^t	a set of fully unlabeled nodes of G^t
E^t	a set of edges of G^t
N^t	the number of nodes in G^t
F	the frequency matrix
P	the point-wise mutual information (PPMI) matrix
Z^s	source graph node representation
Z^t	target graph node representation
att^s	attention coefficient of source domain
att^t	attention coefficient of target domain
f_s, f_d, f_t	source classifier, domain classifier, target classifier
λ	the domain adaptation rate
γ_1, γ_2	the balance parameters of the overall objective

3.1 Problem Statement

Node Classification on Graphs: In this paper, we focus on node classification on graphs. A graph is represented as $G = (V, E, X, Y)$, where $V = \{v_i\}_{i=1,\cdots,N}$ is a vertex set representing the nodes in a graph, and $e_{i,j} = (v_i, v_j) \in E$ is an edge indicating the relationship between two nodes. The topological structure of a graph G can be represented by an adjacency matrix A, where $A_{i,j} = 1$ if $(v_i, v_j) \in E$; otherwise $A_{i,j} = 0$. $x_i \in X$ indicates content features associated with each node v_i. $Y \in \mathbb{R}^{N \times C}$ is a label matrix of G, where N is the number of nodes in G and C is the number of node categories. If a node $v_i \in V$ is associated with label l , $Y^l_{(i)} = 1$; otherwise, $Y^l_{(i)} = 0$.

Source Domain Graph: Let $G^s = (V^s, E^s, X^s, Y^s)$ be a fully labeled source network with a set of all labeled nodes V^s and a set of edges E^s. $Y^s \in \mathbb{R}^{N^s \times C}$ is a label matrix of G^s, where $N^s = |V^s|$ is the number of nodes in G^s and C is the number of node categories.

Target Domain Graph: Similarly, the target network is represented as $G^t = (V^t, E^t, X^t)$, which is a completely unlabeled target network with a set of unlabeled nodes V^t and a set of edges E^t.

Unsupervised Domain Adaptive Node Classification: Given an unlabeled target network G_t and a fully labeled source network G_s, the cross domain node classification is to build a classifier f to accurately classify the nodes in the target network with the assistance of the fully labeled source network. However, this is a challenging task due to the lack of labels for G_t.

3.2 Overall Framework

In order to leverage cross-domain graphs to learn a classifier for node classification, we propose an Unsupervised Domain Adaptive Graph Convolutional Networks (UDA-GCN) to reduce the distribution gap and induce a low-dimensional feature representation shared across domains. Our framework, as shown in Figure 2, mainly consists of the following three components:

- **Node Representation Learning**. In order to learn the better representation of each node, we employ a dual graph convolutional networks to capture the local and global consistency relationship of each graph.
- **Inter-Graph Attention**. We develop an inter-graph attention approach to automatically determine the weights (att^s,att^t) of the source and target graph representations from the local and global GCN layers, respectively.
- **Domain Adaptive Learning for Cross-Domain Node Classification**. To enable cross-domain classification, we advocate a domain adaptive learning approach to train three classifiers. The first one is source classifier, and aims to minimize the classification loss on the source domain data. The second one is domain classifier, and a domain adversarial loss is utilized to enforce the differentiation between the source and target domains. The third one is target classifier, and as there's no label in the target domain, an entropy loss is placed on the target classifier in order to obtain a better semantic information of of the target domain. By doing so, the domain adaptive learning can maximally utilize the domain information and target domain information to learn domain-invariant and semantic representations effectively to reduce the domain discrepancy for cross-domain node classification.

4 METHODOLOGY

This section presents our unsupervised domain adaptive graph convolutional networks for cross-domain node classification.

4.1 Node Embedding Module

In order to encode the semantic information of each node (to capture the local and global information of the graph), the node representation learning procedure consists of two graph neural networks. For local consistency, we introduce the convolutional method using the graph adjacency matrix A. For global consistency, we propose another convolutional method based on a random walk. We feed both the source graph and target graph into our node embedding module.

4.1.1 Local Consistency Network ($Conv_A$). By directly utilizing the GCN method proposed by [15], we formulate the $Conv_A$ as a type of feed-forward neural network. Given the input feature matrix X and adjacency matrix A, the output of the i-th hidden layer of the network Z is defined as:

$$Conv_A^{(i)}(X) = Z^{(i)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{\frac{1}{2}}Z^{(i-1)}W^{(i)}\right) \quad (1)$$

where $\tilde{A} = A + I_n$ is the adjacent matrix with self-loops ($I_n \in \mathbb{R}^{n \times n}$ is the identity matrix), and $\tilde{D}_{i,i} = \sum_j \tilde{A}_{i,j}$. Accordingly, $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{\frac{1}{2}}$ is

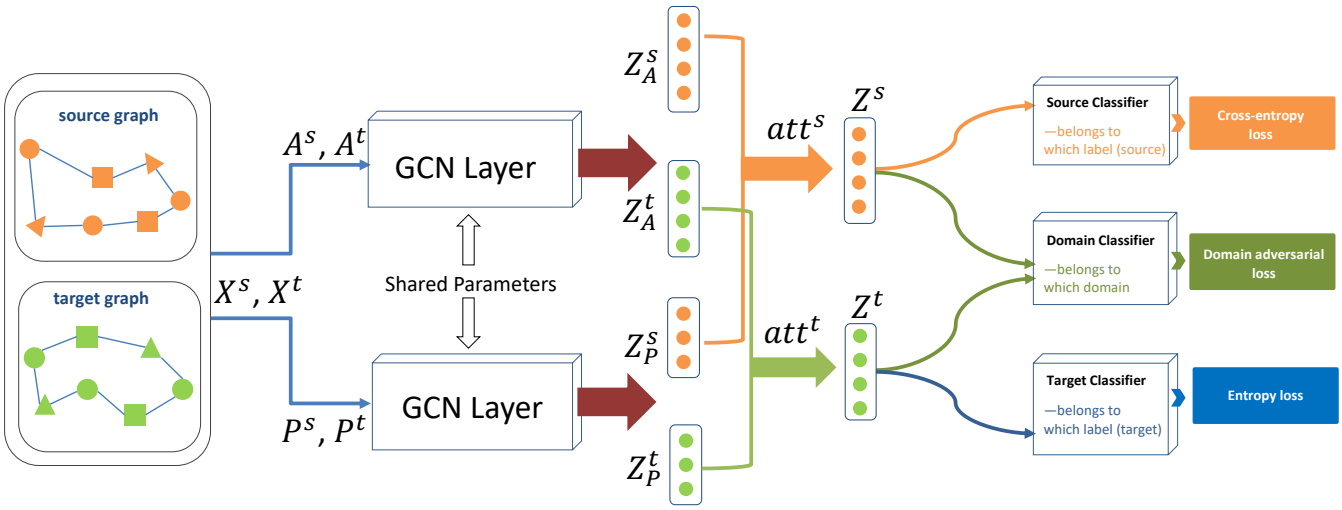


Figure 2: The overall architecture of the proposed Unsupervised Domain Adaptive Graph Convolutional Networks (UDA-GCN) for Cross Domain Node Classification. The input consists of graphs from source and target domains. Our UDA-GCN model consists of three components: (1) UDA-GCN first uses a dual graph convolutional networks to capture the local and global consistency relationship of each graph. (2) Then, a inter-graphed based attention mechanism is proposed to combine the output of different convolved data transformations. (3) Finally, we introduce three classifiers working together to learn domain-invariant and semantic representations effectively for training the source classifier, domain classifier and the target classifier, respectively. For more details, please refer to the content in Section 4.

the normalized adjacency matrix. $Z^{(i-1)}$ is the output of the $(i-1)$-th layer, and $Z^{(0)} = X$. $W^{(i)}$ are the trainable parameters of the network, and $\sigma(\cdot)$ denotes the activation function.

4.1.2 Global Consistency Network ($Conv_P$). In addition to $Conv_A$ which defined by the adjacent matrix A, we introduce a PPMI-based convolution method to encode the gloable information, which is denoted as a matrix $P \in \mathbb{R}^{N\times N}$.

Before obtaining the matrix P, we first calculate a frequency matrix F using the random walk. Random walks have been used as a similarity measure for a variety of problems in recommendation [29], graph classification [1], and semi-supervised learning [44]. Here, we use the random walk to calculate the semantic similarities between nodes. We then calculate P and explain why it leverages knowledge from the frequency to semantics based on F . Finally, we define the P-based graph convolution function $Conv_P$.

Frequency matrix F: The Markov chain describing the sequence of nodes visited by a random walker is called a random walk. If the random walker is on node x_i at time t , we define the state as $s(t) = x_i$. The transition probability of jumping from the current node x_i to one of its neighbors x_j is denoted as $p(s(t+1) = x_j | s(t) = x_i)$. In our problem setting, given the adjacency matrix A, we assign:

$$p(s(t+1) = x_j | s(t) = x_i) = A_{i,j} / \sum_j A_{i,j} \tag{2}$$

Point-wise mutual information matrix (PPMI) P: After calculating the frequency matrix F, the i-th row in F is the row vector $F_{i,:}$ and the j-th column in F is the column vector $F_{:,j}$. $F_{i,:}$ corresponds to a node x_i and $F_{:,j}$ corresponds to a context c_j. The contexts are defined as all nodes in X. The value of an entry $F_{i,j}$ is the number of times that x_i occurs in context c_j. Based on F , we calculate the PPMI matrix $P \in \mathbb{R}^{N\times N}$ as:

$$p_{i,j} = \frac{F_{i,j}}{\sum_{i,j} F_{i,j}}, \tag{3}$$

$$p_{i,*} = \frac{\sum_j F_{i,j}}{\sum_{i,j} F_{i,j}}, \tag{4}$$

$$p_{*,j} = \frac{\sum_i F_{i,j}}{\sum_{i,j} F_{i,j}}, \tag{5}$$

$$P_{i,j} = max\left\{log\left(\frac{p_{i,j}}{p_{i,*}p_{*,j}}\right), 0\right\} \tag{6}$$

Since our node embedding module consists of two networks, in addition to the $Conv_A$, which is based on the similarity defined by the adjacency matrix A, another network $Conv_P$ is derived from the similarity defined by the PPMI matrix P. This neural network is given by:

$$Conv_P^{(i)}(X) = Z^{(i)} = \sigma\left(D^{-\frac{1}{2}} P D^{-\frac{1}{2}} Z^{(i-1)} W^{(i)}\right) \tag{7}$$

where P is the PPMI matrix and $D_{i,i} = \sum_j P_{i,j}$ for normalization. Obviously, applying diffusion based on such a node-contextual matrix P ensures global consistency. Additionally, by using the same neural network structure as $Conv_A$, the two can be combined very concisely. Therefore, source and target graphs are fed into the parameters-shared node embedding module to learn representations.

4.2 Inter-Graph Attention

After performing the node embedding module for the source and target graph, we obtain four embeddings, Z_A^s, Z_P^s for source graph and Z_A^t,Z_P^t for target graph. We need to aggregate embeddings from different graphs to produce a unified representation. For each domain, as embeddings from the local and global consistency networks contribute differently to learning the representation, we propose an *Inter-Graph Attention* scheme to capture the significance of each embedding from each domain.

Specially, we use the original input X^s and X^t as the key of the attention mechanism. We then perform attention on each domain output (Z_A^s,Z_P^s for source domain and Z_A^t,Z_P^t for target domain), two attention coefficient att^s and att^t are computed by an attention function f for each domain, respectively:

$$att_A^k = f(Z_A^k, JX^k), \tag{8}$$

$$att_P^k = f(Z_P^k, JX^k), \tag{9}$$

where k denotes that the output is from the source domain s or the target domain t, J is a shared weight matrix to make the input X^k have the same dimension with the output Z_A^k and Z_P^k. Then we further normalize the weight att^k with a softmax layer.

$$att_A^k = \frac{exp\left(att_A^k\right)}{exp\left(att_A^k + att_P^k\right)}, \tag{10}$$

$$att_P^k = \frac{exp\left(att_P^k\right)}{exp\left(att_A^k + att_P^k\right)}, \tag{11}$$

After implementing the attention, we can get the final output Z^s and Z^t:

$$Z^s = att_A^s Z_A^s + att_P^s Z_P^s. \tag{12}$$

$$Z^t = att_A^t Z_A^t + att_P^t Z_P^t. \tag{13}$$

4.3 Domain Adaptive Learning for Cross-Domain Node Classification

To better learn a knowledge transfer across different domains to assist in the node classification task, our proposed model consists of a adversarial module, a source classifier as well as a target classifier working together to learn both class discriminative and domain invariant node representations, thus enabling classifying nodes in the target network. The overall objective is as follows:

$$\mathcal{L}(Z^s, Y^s, Z^t) = \mathcal{L}_S(Z^s, Y^s) + \gamma_1 \mathcal{L}_{DA}(Z^s, Z^t) + \gamma_2 \mathcal{L}_T(Z^t) \tag{14}$$

The γ_1, γ_2 are the balance parameters. The $\mathcal{L}_S$, $\mathcal{L}_{DA}$ and $\mathcal{L}_T$ represent the source classifier loss, the domain classifier loss and the target classifier loss, respectively. The details are introduced as follows.

4.3.1 Source Classifier Loss. The source classifier loss $\mathcal{L}_S(f_s(Z^s), Y^s)$ is to minimize the cross-entropy loss for the labeled data in the source domain:

$$\mathcal{L}_S(f_s(Z^s), Y^s) = -\frac{1}{N_s} \sum_{i=1}^{N_s} y_i \log(\hat{y_i}), \tag{15}$$

where y_i denotes the label of the i-th node in the source domain, $\hat{y_i}$ are the classification prediction for the i-th source labeled node v_i^s, respectively.

4.3.2 Domain Classifier Loss. The domain classifier loss $\mathcal{L}_{DA}(Z^s, Z^t)$ enforces that the node representation after the node feature extraction process from source domain network G^s and target domain network G^t are similar. To achieve this, we learn a domain classifier $f_d(Q_\lambda(Z^s, Z^t); \theta_D)$ parameterized by θ_D with an adversarial training scheme, which tries to disciminative if a node is from G^s or G^t. On the one hand, we would like the source classifier f_s can classify each node into the correct class via minimizing Eq. (15). On the other hand, we would like that node representations from different domains are similar, so that the domain classifier cannot differentiate if the node comes from G^t or G^s. In our paper, we use Gradient Reversal Layer (GRL) [13] for adversarial training. Mathematically, we define the GRL as $Q_\lambda(x) = x$ with a reversal gradient $\frac{\partial Q_\lambda(x)}{\partial x} = -\lambda I$. Learning a GRL is adversarial in such a way that: on the one side, the reversal gradient enforces $f_s(Z^s)$ to be maximized; on the other side, θ_D is optimized by minimizing the cross-entropy domain classifier loss:

$$\mathcal{L}_{DA} = -\frac{1}{N^s + N^t} \sum_{i=1}^{N^s+N^t} m_i \log(\hat{m}_i) + (1 - m_i)\log(1 - \hat{m}_i) \tag{16}$$

where $m_i \in \{0, 1\}$ denotes the groundtruth, and $\hat{m}_i$ denotes the domain prediction for the i-th document in the source domain and target domain, respectively.

4.3.3 Target Classifier Loss. In the target domain, an entropy loss is placed on the target classifier. Unlike the source classifier, we do not use cross-entropy as the label loss, because we do not have the class information for the unsupervised learning in the target domain. In order to utilize the data in the target domain, we employ an entropy loss for the target classifier f_t:

$$\mathcal{L}_T(f_t(Z^t)) = -\frac{1}{N^t} \sum_{i=1}^{N^t} \hat{y_i} \log(\hat{y_i}), \tag{17}$$

where $\hat{y_i}$ are the classification prediction for the i-th node v_i^t in the target domain.

$\mathcal{L}_S(Z^s, Y^s)$, $\mathcal{L}_{DA}(Z^s, Z^t)$ and $\mathcal{L}_T(Z^t)$ are jointly optimized via our objective function in Eq. (14), and all parameters are optimized using the standard backpropagation algorithms.

4.4 Algorithm Description

Our algorithm is illustrated in Algorithm 1. Given a source graph $G^s = (V^s, E^s, X^s, Y^s)$ and a target graph $G^t = \left(V^t, E^t, X^t\right)$, our goal is to obtain the node representations of the source graph and the target graph Z^s and Z^t, respectively. Firstly, we employ a dual graph convolutional networks to capture the local and global consistency relationship of each graph (Step 2-8). Here, the original input are X^s and X^t, the output are Z_A^s,Z_P^s for source domain and Z_A^t,Z_P^t for target domain. Then we propose an inter-graph attention scheme to the output of each domain and we obtain the final output of node representations Z^s and Z^t (Step 9). Finally, by employing the source classifier, domain classifier and target classifier, we can maximally utilize the domain information and label information to

Algorithm 1 UDA-GCN training algorithm for cross domain node classification

Require:
source domain network $G^s = (V^s, E^s, X^s, Y^s)$;
target domain network $G^t = (V^t, E^t, X^t)$;
the numbers of GCN layers L;
Ensure:
$[Z^s, Z^t]$: output embeddings
$[f_s, f_d, f_t]$: source classifier; domain classifier; target classifier;
1: **while** not convergence **do**
2: **for** i=1 to L **do**
3: $Z_A^i \leftarrow Conv_A^{(i)}\left(Z^{(i-1)}\right)$
4: $Z_P^i \leftarrow Conv_P^{(i)}\left(Z^{(i-1)}\right)$
5: **if** $i = 1$ **then**
6: $Z^{(i-1)} \leftarrow X$
7: **end if**
8: **end for**
9: $[Z_A^s, Z_P^s, Z_A^t, Z_P^t] \leftarrow$ Obtain four embeddings for source domain and target domain.
10: $[Z^s, Z^t] \leftarrow$ Learn output embeddings for source domain and target domain using Eqs. (12) and (13)
11: $f_s \leftarrow$Learn source classifier from Z^s and Y^s using Eq. (15)
12: $f_d \leftarrow$ Learn domain classifier from Z^s and Z^t using Eq. (16)
13: $f_t \leftarrow$ Learn target classifier from Z^t using Eq. (17)
14: Back-propagate loss gradient from Z^s, Z^t and Y^s using Eq. (14)
15: Update weights
16: **if** early stopping condition satisfied **then**
17: Terminate
18: **end if**
19: **end while**

learn domain-invariant and semantic representations effectively to reduce the domain discrepancy for cross-domain node classification (Step 11-13).

4.5 Time Complexity Analysis

Given a graph with n nodes and m edges, if the adjacent matrix is sparse, the time complexity for the graph convolution operation of GCN is $O(m)$. In our model, we use the point-wise mutual information (PPMI) matrix instead of the adjacency matrix for propagation. Because the PPMI matrix is not guaranteed to be sparse, its complexity is subject to a dense matrix complexity: $O(n^2)$. In addition, because we employ a dual GCN consisting of the sparse adjacency matrix and a dense PPMI matrix, the overall time complexity of Dual GCN moudle is $O(m + n^2)$.

5 EXPERIMENTS

In this section, we will first describe benchmark datasets, baselines and experimental setting, and then report the algorithm performance.

5.1 Benchmark Datasets

We conduct experiments on three real-world networks. We constructed graphs based on datasets provided by ArnetMiner [34]. The details of the experimental datasets are displayed in Table 2. DBLPv8, ACMv9 and Citationv2 are three paper citation networks from different original sources (DBLP, ACM and Microsoft Academic Graph respectively) and for each dataset, we extracted the papers published in different periods, i.e., DBLPv8 (after year 2010), ACMv9 (between years 2000 and 2010), and Citationv2 (before year 2008). In our experiments, we consider them as undirected networks and each edge representing a citation relation between two papers. We classify papers to some of the following six categories according to its research topics, including "Database", "Data mining", "Artificial intelligent", "Computer vision", "Information Security" and "High Performance Computing". We evaluate our proposed model by conducting multi-label classification on these three network domains through six transfer learning tasks including C→D, A→D, D→C, A→C, D→A, and C→A, where D, A, C denote DBLPv8, ACMv9 and Citationv2, respectively.

Table 2: Statistics of the experimental datasets.

Dataset	# of Nodes	# of Edges	# of Features	# of Labels
DBLPv8	5578	7341	7537	6
ACMv9	7410	11135	7537	6
Citationv2	4715	6717	7537	6

5.2 Baselines

In order to make a fair comparison and demonstrate the effectiveness of our proposed model, we employ the following methods as baselines. We compare our approach with both state-of-the-art single-domain node classification models as well as cross-domain models with the necessary domain adaption.

State-of-the-art single-domain node classification models:

- DeepWalk [28]: It is a classics single network embedding method which employs the random walk sampling strategy to generate the neighborhood of each node and extends Skip-Gram model to learn low-dimensional node representation.
- LINE [33]: LINE can preserve both first-order and second-order proximities for the undirected network through modeling node co-occurrence probability and node conditional probability.
- GraphSAGE [15]: The spectral clustering algorithm of SFA is adapted to co-cluster all words into the shared clusters for domain adaptation.
- DNNs: DNNs is a multi-layer perceptron (MLP) which only uses the node features.
- GCN [18]: GCN is a deep convolutional network for graph-structured data, which integrates network topology, node features and observed labels into an end-to-end learning framework.

Cross-domain node classification models with adaption:

- DGRL [13]: The feature generator is a 2-layer perceptron to obtain the representation of each node. A gradient reverse layer (GRL) is added for domain classification..
- AdaGCN [6]: The feature generator is a GCN architecture and a gradient reverse layer (GRL) is added to train a domain classifier.

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

Methods	C→D	A→D	D→C	A→C	D→A	C→A	**Average**
DeepWalk	0.1397	0.1798	0.2840	0.2284	0.3649	0.2005	0.2329
LINE	0.2216	0.1972	0.2539	0.2848	0.4117	0.1895	0.2598
GraphSAGE	0.6151	0.7228	0.6312	0.4333	0.6961	0.4413	0.5900
DNNs	0.4035	0.4279	0.5065	0.3832	0.5904	0.3669	0.4464
GCN	0.6250	0.6486	0.6259	0.4327	0.6945	0.4474	0.5790
DGRL	0.4229	0.4303	0.5018	0.3877	0.5947	0.3799	0.4529
AdaGCN	0.6388	0.7142	0.6399	0.4448	0.7045	0.4494	0.5986
UDA-GCN	**0.7182**	**0.7341**	**0.7281**	**0.4770**	**0.7617**	**0.4603**	**0.6466**

Table 4: Classification accuracy comparisons between UDA-GCN variants on six cross-domain tasks.

Methods	C→D	A→D	D→C	A→C	D→A	C→A
UDA-GCN$\neg p$	0.6522	0.7305	0.6522	0.4520	0.7103	0.4495
UDA-GCN$\neg d$	0.7054	0.7162	0.6719	0.4651	0.7215	0.4559
UDA-GCN$\neg t$	0.6882	0.7305	0.6880	0.4651	0.7393	0.4559
UDA-GCN	**0.7182**	**0.7341**	**0.7281**	**0.4770**	**0.7617**	**0.4603**

Table 5: The summary of the variants of graph convolutional networks. The symbol × indicates the algorithm exploits the corresponding information.

Methods	local	global	domain loss	target loss
GraphSAGE	×			
GCN	×			
AdaGCN	×		×	
UDA-GCN$\neg p$	×		×	×
UDA-GCN$\neg d$	×	×		×
UDA-GCN$\neg t$	×	×	×	
UDA-GCN	×	×	×	×

5.3 Experimental Settings

All deep learning algorithms are implemented in Pytorch [26] and are trained with Adam optimizer. We follow the evaluation protocol in unsupervised domain adaptation [6, 19] and evaluate all approaches through grid search on the hyperparameter space and report the best results of each approach. We use all labeled source samples and all unlabeled target samples. For all cross domain node classification tasks, we use the same set of parameter configurations unless otherwise specified. For each deep approach, we use a fixed learning rate $1e^{-4}$. For GCN, AdaGCN and UDA-GCN, the GCNs of both the source and target networks contain two hidden layers ($L = 2$) with structure as $128 - 16$. For DeepWalk and LINE, node representations are first learned and then a one-vs-rest logistic regression classifier is trained with labeled nodes of source domain. Here, the dimension of node representations for them are all set to 128 for fair comparison. For GraphSAGE, we also adapt it to the inductive setting, and train in the source domain. Here, we utilize the Pytorch version implemented by the geometric deep learning extension library [10]. DNNs and DGRL have similar parameter settings with GCN and AdaGCN, respectively. The adaptation rate λ is the following schedule: $\lambda = \min(\frac{2}{1+\exp(-10p)} - 1, 0.1)$, and the p is changing from 0 to 1 within the training process as in [13]. The balance parametersγ_1, γ_2 are set 1, 0.8, respectively. The dropout rate for each GCN layer is set to 0.3.

5.4 Cross-Domain Classification Results

Table 3 lists the accuracy of different methods on cross-domain node classification tasks. From the results, we have the following observations:

(1) The DeepWalk and LINE obtain the worst performance among these baselines since they only utilize the network structure information rather than node features. The DNNs also obtain worse performance than the other methods. This is because the traditional DNNs only consider node features, and do not capture the graph structure information for better node representation.
(2) Graph-based methods (GCN and GraphSAGE) have better performances than the traditional two-step network embedding methods (DeepWalk and LINE), which shows that the end-to-end graph convolutional neural networks encoding both local graph structure and features of nodes have competitive advantages than traditional models in cross domain node classification.
(3) DGRL and AdaGCN have better performance than the traditional single-domain node classification methods (DNNs and GCN), confirming the superiority of domain-loss in cross-domain node classification.
(4) The proposed UDA-GCN model consistently beats all the baselines on six cross-domain tasks. It demonstrates that the proposed domain adaptive graph neural network can better capture the underlying representation of the documents and reduce the distribution gap across domains by jointly modeling local and global consistency relations of each graph, domain information, source domain information and target domain information as a unified learning framework.

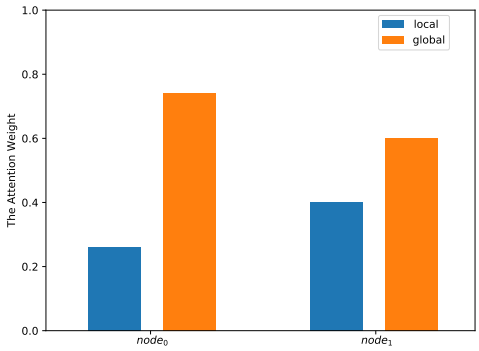


Figure 3: The importance of attention mechanism. $node_0$ and $node_1$ are two random nodes of the network.

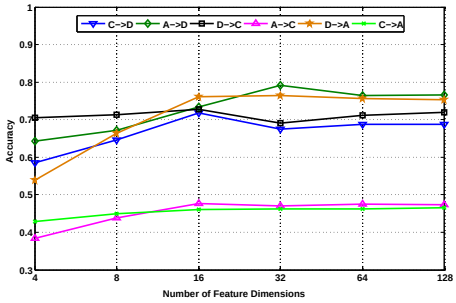


Figure 4: Impact of feature dimensions of node output embeddings

5.5 Analysis of UDA-GCN Components

Because the proposed UDA-GCN contains multiple key components, in this section, we compare variants of UDA-GCN with respect to the following aspects to demonstrate the effectiveness of UDA-GCN — (1) the effect of the dual graph neural network module, (2) the impact of domain-adversarial loss, and (3) the impact of the target classifier loss. The following UDA-GCN variants are designed for comparison.

- UDA-GCN¬p: A variant of UDA-GCN with the global GCN layer of DAGNN being removed, and only using the local GCN layer.
- UDA-GCN¬d: A variant of UDA-GCN with the gradient reverse layer of UDA-GCN (*i.e.*, domain classifier) being removed.
- UDA-GCN¬t: A variant of UDA-GCN with the target classifier loss of DAGNN being removed.

Different variants of graph neural networks model are summarized in Table 5, where the symbol × indicates the algorithm exploits the corresponding information. The ablation study results are shown in Table 4.

5.5.1 Effects of the global GCN layer module. We compare UDA-GCN with UDA-GCN¬p to investigate the effectiveness of the novel dual GCN approach employed in our paper. From the result, we find that the UDA-GCN model performs better than UDA-GCN¬p, which confirms the superiority of the dual GCN which combines the local and global relationship to capture a comprehensive representation of the node.

5.5.2 Impact of domain-adversarial loss. In order to verify the effectiveness of the domain-adversarial loss, we compare UDA-GCN model and UDA-GCN¬d. From Table 4, we can easily observe the UDA-GCN model performs significantly better than UDA-GCN¬d. This confirms that the usage of domain-adversarial loss can learn a superior representation for nodes from different domains.

5.5.3 Impact of the target classifier loss. In order to show the superiority of the target classifier, we design a variant model UDA-GCN¬t. The only difference between UDA-GCN¬t and UDA-GCN is that UDA-GCN¬t do not use the information of the target domain which is also the core information in cross-domain learning. The results in Table 4 show the performances of the node classification task on both datasets are improved when the target information are used, indicating the effectiveness of the target classifier loss.

5.6 Parameter Analysis

5.6.1 Attention Mechanism. By employing the inter-graph attention method to combine the local and global relationship for a comprehensive node representation, UDA-GCN obtains better results than AdaGCN.

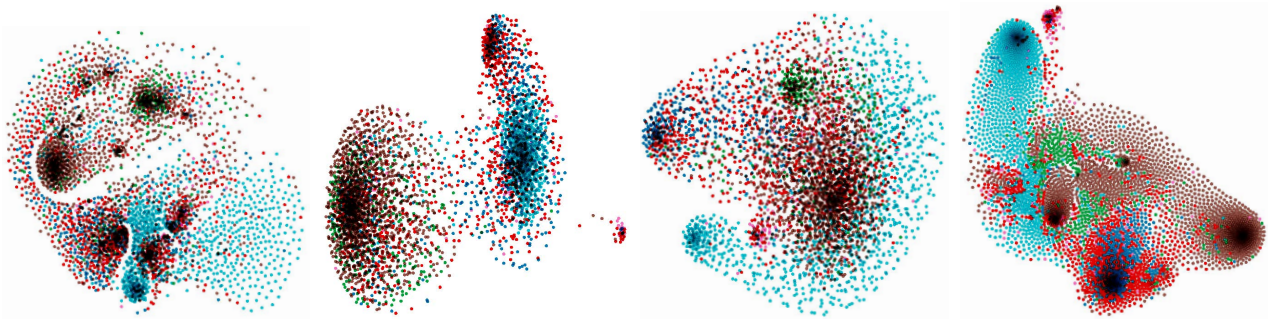
Fig. 3 shows the importance vectors of two nodes (the $node_0$ and $node_1$, which are the randomly selected in this experiment). Note that the importance vector is node-wise, that is, each type of embedding plays different roles for different nodes. For $node_0$, the global embedding is much more important than the local embedding, while for $node_1$, the model pays more attention to the local embedding. Regarding the motivation of the attention mechanism, we find that the attention-based integration can perform better than the simple averaging-based or concatenation integration. The attention mechanism can capture the importance of different types of embeddings and learn the optimal integration weights of two types of representations.

5.6.2 Impact of feature dimensions of node output embeddings Z^s and Z^t . We set the number of feature dimensions of source output embedding Z^s as the same as that of target output embedding Z^t. UDA-GCN uses 2-layer GCNs with structure as $128 - 16$, and feature dimensions d of node output embeddings is 16. We vary d from 4 to 128 and report the results of six cross domain node classification tasks, respectively in Figure 4. When d increases from 4 to 128, the accuracy of target domain is improved on both tasks. Furthermore, only slight differences can be observed with different d, and the increase of d does not necessarily result in performance improvements from 16 to 128. The results show that with sufficient feature dimensions ($d \geq 16$), UDA-GCN is stable with the number of feature dimensions.

5.6.3 Visualization. An important application of network representation is to create meaningful visualizations that layout a network on a two dimensional space. We visualize the learned embedding for the target domain dataset. For simplicity, we only visualize two learned embeddings in the DBLPv8→ACMv9 and DBLPv8→Citationv2 to validate the effectiveness of our proposed model. For each approach, we map the the learned embedding

(a) DNNs (b) GraphSAGE (c) GCN (d) UDA-GCN

Figure 5: Visualization of the domain adaptive embedding learning results using t-SNE [22] (The source domain is DBLPv8 and the target domain is ACMv9).



(a) DNNs (b) GraphSAGE (c) GCN (d) UDA-GCN

Figure 6: Visualization of the domain adaptive embedding learning results using t-SNE [22] (The source domain is DBLPv8 and the target domain is Citationv2).

vectors to a 2-D space with the T-distributed Stochastic Neighbor Embedding (t-SNE) [22] method. t-SNE can project each high-dimensional objects into a 2-dimensional or 3-dimensional space, where similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability. By t-SNE, the visualization result can preserve the similarity between the learned embeddings.

Figs. 5 and 6 compare the visualization results of different approaches. We can observe that the visualization using DNNs is not very meaningful, where many nodes belonging to the same class are not clustered together and clusters are overlapped. GraphSAGE and GCN perform better than DNNs and their t-SNE results show more meaningful clusters than DNNs. However, the boundaries of most clusters are still hard to find. For the proposed UDA-GCN method, clusters are much more clear and many obvious boundaries can be found between clusters, which shows UDA-GCN can generate more meaningful layout of the network than other approaches.

6 CONCLUSIONS

In this paper, we studied the problem of unsupervised graph domain adaptation. We argued that most existing graph neural networks only learn models in a single graph and fail to consider the knowledge transfer across graphs. In this paper, we presented a novel unsupervised domain adaptive graph convolutional networks (UDA-GCN) to enable knowledge adaptation between graphs. By employing a dual graph convolutional networks to exploit both local and global relations of the graphs, we are able to learn better representation for nodes in both source and target graphs. The inter-graph attention mechanism presented here further generates a unified embedding for down-stream node classification task. By using a cross-entropy loss for source domain classification, a domain adversarial loss for domain discrimination, and an entropy loss for target domain information absorption, we are able to reduce the domain discrepancy and enable efficient domain adaptation. Experimental results on three real-world graph datasets show that our algorithm outperforms existing methods for cross domain network node classification.

ACKNOWLEDGMENTS

This research is supported by the U.S. National Science Foundation (NSF) through Grants IIS-1763452 and CNS-1828181, the NSFC (No. 61872360), the Youth Innovation Promotion Association CAS (No. 2017210), and Australian Research Council through grant DE190100626.

REFERENCES

[1] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 475–486.

[2] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. 2011. Node classification in social networks. In *Social network data analytics*. Springer, 115–148.

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. ACM, 891–900.

[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep neural networks for learning graph representations. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[5] Minmin Chen, Kilian Q Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *NIPS*. 2456–2464.

[6] Quanyu Dai, Xiao Shen, Xiao-Ming Wu, and Dan Wang. 2019. Network Transfer Learning via Adversarial Domain Adaptation with Graph Convolution. *arXiv preprint arXiv:1909.01541* (2019).

[7] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2007. Co-clustering based classification for out-of-domain documents. In *SIGKDD*. 210–219.

[8] Lun Du, Yun Wang, Guojie Song, Zhicong Lu, and Junshan Wang. 2018. Dynamic Network Embedding: An Extended Approach for Skip-gram based Network Embedding.. In *IJCAI*. 2086–2092.

[9] Lixin Duan, Ivor W Tsang, and Dong Xu. 2012. Domain transfer multiple kernel learning. *IEEE TPAMI* 34, 3 (2012), 465–479.

[10] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[11] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein interface prediction using graph convolutional networks. In *Advances in Neural Information Processing Systems*. 6530–6539.

[12] Yaroslav Ganin and Victor Lempitsky. 2014. Unsupervised domain adaptation by backpropagation. *arXiv:1409.7495* (2014).

[13] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *JMLR* 17, 1 (2016), 2096–2030.

[14] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[16] Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *ACL*. 264–271.

[17] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. 2017. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1857–1865.

[18] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[19] Lianghao Li, Xiaoming Jin, and Mingsheng Long. 2012. Topic correlation analysis for cross-domain text classification. In *AAAI*. 998–1004.

[20] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. 2015. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791* (2015).

[21] Mingsheng Long, Guiguang Ding, Jianmin Wang, Jiaguang Sun, Yuchen Guo, and Philip S Yu. 2013. Transfer sparse coding for robust image representation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 407–414.

[22] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[23] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430* (2009).

[24] Shirui Pan, Ruiqi Hu, Sai-fu Fung, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Learning graph embedding with adversarial training methods. *IEEE Transactions on Cybernetics* (2019).

[25] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407* (2018).

[26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic Differentiation in PyTorch. In *NIPS Autodiff Workshop*.

[27] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. 2018. Multi-adversarial domain adaptation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[28] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[29] Alain Pirotte, Jean-Michel Renders, Marco Saerens, et al. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge & Data Engineering* 3 (2007), 355–369.

[30] Xiao Shen and Fu-Lai Chung. 2017. Deep network embedding with aggregated proximity preserving. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. ACM, 40–43.

[31] Xiao Shen and Fu Lai Chung. 2019. Network Embedding for Cross-network Node Classification. *arXiv preprint arXiv:1901.07264* (2019).

[32] Baochen Sun and Kate Saenko. 2016. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*. Springer, 443–450.

[33] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 1067–1077.

[34] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 990–998.

[35] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7167–7176.

[36] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474* (2014).

[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[38] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In *IJCAI*. 3670–3676.

[39] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1225–1234.

[40] Man Wu, Shirui Pan, Xingquan Zhu, Chuan Zhou, and Lei Pan. 2019. Domain-Adversarial Graph Neural Networks for Text Classification. In *ICDM*. 648–657.

[41] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019. A comprehensive survey on graph neural networks. *arXiv:1901.00596* (2019).

[42] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *IJCAI*. ijcai.org, 1907–1913.

[43] Linchuan Xu, Xiaokai Wei, Jiannong Cao, and Philip S Yu. 2018. On exploring semantic meanings of links for embedding social networks. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 479–488.

[44] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861* (2016).

[45] Yizhou Zhang, Guojie Song, Lun Du, Shuwen Yang, and Yilun Jin. 2019. DANE: Domain Adaptive Network Embedding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. 4362–4368.

[46] Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. 2018. ANRL: Attributed Network Representation Learning via Deep Neural Networks.. In *IJCAI*, Vol. 18. 3155–3161.

[47] Shichao Zhu, Chuan Zhou, Shirui Pan, Xingquan Zhu, and Bin Wang. 2019. Relation Structure-Aware Heterogeneous Graph Neural Network. In *ICDM*. 1534–1539.

[48] Shichao Zhu, Lewei Zhou, Shirui Pan, Chuan Zhou, Guiying Yan, and Bin Wang. 2020. GSSNN: Graph Smoothing Splines Neural Networks. In *AAAI*.

[49] Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. 2015. Supervised representation learning: Transfer learning with deep autoencoders. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[50] Fuzhen Zhuang, Ping Luo, Zhiyong Shen, Qing He, Yuhong Xiong, Zhongzhi Shi, and Hui Xiong. 2010. Collaborative dual-plsa: mining distinction and commonality across multiple domains for text classification. In *CIKM*. 359–368.