Domain Adaptation and Out-of-Distribution Detection for Network Data

by

Yu Wang

(Under the Direction of Pengsheng Ji and Sheng Li)

Abstract

Node classification on networks, i.e., graphs, is an essential topic in the graph machine learning field with ubiquitous applications in natural science, social science, industries, etc. In this dissertation, we study a few graph machine learning cases where target data are potentially out-of-distribution (OOD) compared to in-distribution (ID) training data. Among various OOD problems, one important field is domain adaptation, where the source and target data (i.e., domains) share the same feature space but different distributions. Current domain adaptation methods for node classification only focus on the closed-set setting, where source and target domains share the same label space. Moreover, only a few pioneering graph OOD node classification methods have been proposed, which leave out the task of OOD detection (also known as anomaly detection). We first address a novel open-set graph domain adaptation learning problem, which can leverage source domains with rich labels to deal with classification tasks in an unlabeled target domain. Specifically, we develop an algorithm

for efficient knowledge transfer from a labeled source graph to an unlabeled target graph under a separate domain alignment strategy, to learn discriminative feature representations for the target graph. Our goal is to not only correctly classify target nodes into the known classes, but also classify unseen types of nodes into an unknown class. Experimental results on real-world datasets show that our method outperforms existing methods on graph domain adaptation. Secondly, we study a graph OOD semi-supervised node classification setting, where our developed method solves both the OOD detection problem and the node classification problem at the same time. Our method can detect OOD nodes from ID nodes while classifying both ID and OOD nodes into their categories. Methods of data augmentation such as Mixup and anomaly detection such as Deep Support Vector Data Description are adapted for graph data and its OOD detection and node classification tasks. We extensively compare our method with others on the graph OOD benchmark datasets and demonstrate our method's effectiveness.

INDEX WORDS: [Network data, graph neural networks, domain adaptation, out-of-distribution learning, node classification, transfer learning, semi-supervised learning, graph mining, network analysis]

Domain Adaptation and Out-of-Distribution Detection for Network Data

by

Yu Wang

B.S., Nanjing University, China, 2015

M.S., University of Georgia, 2018

A Dissertation Submitted to the Graduate Faculty of the University of Georgia in Partial Fulfillment of the Requirements for the Degree.

Doctor of Philosophy

Athens, Georgia

2023

Domain Adaptation and Out-of-Distribution Detection for Network Data

by

Yu Wang

Major Professors:	Pengsheng Ji
	Sheng Li
Committee:	Liang Liu
	Yuan Ke

Electronic Version Approved:

Ron Walcott

Dean of the Graduate School

The University of Georgia

May 2023

Acknowledgments

My deepest and sincerest thanks go to my advisors, Dr. Pengsheng Ji and Dr. Sheng Li. Dr. Ji guided me into the fascinating research field of statistical machine learning on network data. Dr. Li's vision inspired and guided me to start research on domain adaptation and to learn and solve those challenging problems. I could not thank them enough for everything. Without their support, guidance, and mentorship, I wouldn't be the person I am today, nor would this dissertation have been possible. I have learned so much and hope to continue learning from them in the future.

I also wish to thank my committee members, Dr. Liang Liu and Dr. Yuan Ke, for their kind guidance, comments, suggestions, and support throughout the years.

I would like to thank my collaborator, Ronghang Zhu, for his tremendous help and encouragement on my research projects. His expertise in domain adaptation provided invaluable insights into my research.

Thank you to all the professors, classmates, and friends who have been along with my PhD journey. Pursuing a PhD is never easy; all the good company makes this experience rewarding, fun, and unforgettable.

Thanks to Mom and Dad for always being there. Thank you to my twin sister Qi, who is with me since Day 1 and still is my best friend today. Finally, I want to thank my husband Cheng, who is my strongest support system in every

aspect. My PhD journey is truly lightened by his everyday encouragement, jokes, and much more.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Graph machine learning

Graph data, also known as network data, is a unique non-Euclidean data structure[1]. Graph data represents entities (objectives) and their interactions (relationships). Entities are represented as a set of nodes or vertices, and the interactions are represented as edges or links. A lot of real-world datasets are in the form of graphs, such as social networks, citation networks, knowledge graphs, protein-protein interaction networks, molecules, and the World Wide Web, just to name a few. Examples of possible machine learning problems for graph data can be mainly divided into three different categories: node level (e.g., node classification, node regression, node clustering), edge level (e.g., link prediction), and graph level (e.g., graph classification). Machine learning on network data has been a rapidly growing area of research in the past a few decades. It aims to extract useful information from the graph/node/edge attributes and the graph structure, which can lead to better understanding, prediction, and decision-making from the data.

[1] In this dissertation, we use the two terms, graph and network, interchangeably to refer to graph-structured data.

Researchers from different fields of study have been continuously developing graph related methods. From a statistical learning perspective, graph related problems can be categorized into two main categories, supervised and unsupervised learning. In recent years, different problem settings emerged such as semi-supervised and self-supervised learning, which are more realistic settings in a lot of applications, and corresponding methods came up as well (Zhai et al., 2019, Chen et al., 2020). Traditional statistical methods demonstrate good performance on some problems such as community detection (Z. Wang et al., 2020), node classification (Bhagat et al., 2011), and link prediction (Kumar et al., 2020). However, those methods have been facing substantial challenges when dealing with more complex and large-scale systems, as well as constantly evolving problem settings. Thus, we need to develop new methodologies to solve novel problems.

Graph Neural Networks (GNNs), first brought up by Scarselli et al., 2009, are a large group of deep learning models that generalize classical deep learning concepts to graph data and enable neural networks to reason about objects and their relations. GNNs can operate on the graph structure and node features simultaneously, allowing them to capture the local and global structure of the graph. Due to its promising performance and large potential, numerous GNN architectures and applications have been brought up to improve the methodologies of solving graph related research questions. Nowadays, GNNs have been constantly showing that they hold state-of-art performances in a lot of graph machine learning studies. In this dissertation, all proposed methodologies are based on GNN models.

1.2 A review of GNNs: frameworks and open problems

1.2.1 Original GNN model

GNNs are largely motivated by two things: convolutional neural networks (CNNs) and graph embedding (Zhou et al., 2018). CNNs are brought up by LeCun et al., 1998 which emphasize the local connection, shared weights and the use of multi-layer, while graph embedding learns to represent graph nodes, edges or subgraphs in low-dimensional vectors. These two aspects can both be applied to collectively aggregate information from graph structure, thus to solve the graph problems. The goal of GNNs is to generalize classical deep learning concepts to graph data and enable neural networks to reason about objects and their relations.

The idea of GNNs comes from Scarselli et al., 2009, which extends the existing neural networks into the graph area. GNNs do propagation guided by the graph structure instead of using it as part of features. Generally speaking, GNNs train a model that updates the hidden state of nodes by a weighted sum of the states of their neighborhood and then uses the updated embeddings to solve the corresponding task, such as node classification, graph classification, link prediction, etc.

For a set of graph data, let $\mathbf{h}_v$ be an s-dimension vector of node v and can be used to produce an output $\mathbf{o}_v$ such as the node label. Let f be a parametric function (local transition function) that is shared among all nodes and updates the node state according to the input neighborhood, and g be the local output function that describes how the output is produced. Define:

$$\mathbf{h}_v = f\left(\mathbf{x}_v, \mathbf{x}_{co[v]}, \mathbf{h}_{ne[v]}, \mathbf{x}_{ne[v]}\right) \tag{1.1}$$

$$\mathbf{o}_v = g\left(\mathbf{h}_v, \mathbf{x}_v\right) \tag{1.2}$$

where $\mathbf{x}_v$, $\mathbf{x}_{co[v]}$, $\mathbf{h}_{ne[v]}$, $\mathbf{x}_{ne[v]}$ are the features of v, the features of its edges, the states, and the features of the nodes in the neighborhood of v, respectively.

Let $\mathbf{H}$, $\mathbf{O}$, $\mathbf{X}$, and $\mathbf{X}_N$ be the vectors constructed by stacking all of the states, outputs, features, and node features, respectively. The compact form will be:

$$\mathbf{H} = F(\mathbf{H}, \mathbf{X}) \tag{1.3}$$

$$\mathbf{O} = G\left(\mathbf{H}, \mathbf{X}_N\right) \tag{1.4}$$

where F, the global transition function, and G, the global output function are stacked versions of f and g for all nodes in a graph, respectively. The value of $\mathbf{H}$ is the fixed point of Eq. 1.3 and is uniquely defined with the assumption that F is a contraction map.

Let $\mathbf{H}^t$ denotes the t-th iteration of $\mathbf{H}$. The iterative scheme for GNN to compute the state of $(t+1)$-th iteration of $\mathbf{H}$ is defined as:

$$\mathbf{H}^{t+1} = F\left(\mathbf{H}^t, \mathbf{X}\right). \tag{1.5}$$

After defining the framework of GNN, the next step is to learn the parameters of f and g. Let $\mathbf{t}_v$ be the target information for a specific node, the loss can be written as follow:

$$\text{loss} = \sum_{i=1}^{p} \left(\mathbf{t}_i - \mathbf{o}_i\right) \tag{1.6}$$

Algorithm 1 Algorithm for GNNs

1: Update the states $\mathbf{h}_v^t$ iteratively by Eq. 1.1 until a time T. They approach the fixed point solution of Eq. 1.3: $\mathbf{H}(T) \approx \mathbf{H}$.
2: Compute the gradient of weights $\mathbf{W}$ from the loss in Eq. 1.6.
3: Update weights $\mathbf{W}$ according to the gradient computed in the last step.

where p is the number of supervised nodes. Then we can learn the weights based on a gradient-descent strategy and is shown in Algorithm 1.

The original GNN is a promising idea but also has some limitations, including: it is inefficient to update the hidden states of nodes iteratively for the fixed point; it uses the same parameters in the iterations; edge features cannot be effectively used and modeled. Therefore, a lot of its variants were proposed afterwards.

1.2.2 Variants of GNNs

The variants of GNNs can be grouped into three different categories: GNN variants operating on different graph types, variants modifying propagation rules, and variants using advanced training methods. Below we will introduce them briefly.

1. GNN variants for different graph types.

   Graphs can be directed or undirected. The original GNN method is applied to undirected graphs, while for directed graphs the edges that connecting two nodes contains more information than the mutual "connection" meaning. Therefore, Kampffmeyer et al., 2019 proposed to use two kinds of weight matrix to incorporate more precise structural information for the two groups on different sides of the edges. Also, different

methods were brought up for heterogeneous graphs, and graphs with edge information, as well as dynamic graphs.

2. GNN variants modifying propagation rules.

   A lot of propagation rules have been proposed to improve the quality of learning representations. One important idea, convolution, has been incorporated into the GNN model. Methods using the convolution can be divided into two groups: spectral approaches and non-spectral (spatial) approaches. Spectral approaches include ChebNet by Hammond et al., 2011 and GCN by Kipf and Welling, 2017. For all of the spectral approaches, the learned filters depend on the graph Laplacian eigenbasis, which depends on the graph structure. Thus, these models trained on a specific structure could not be directly applied to a graph with a different structure.

   Non-spectral approaches define convolutions directly on the graph instead of the graph Laplacian, operating on spatially close neighbors. Examples include Neural FPs by Duvenaud et al., 2015, DCNN by Atwood and Towsley, 2016, GraphSAGE by Hamilton et al., 2017, etc.

   Also, there are other works such as gated graph neural network (GGNN) by Y. Li et al., 2015 that uses the gate mechanism, graph attention network (GAT) by Veličković et al., 2017 which incorporates the attention mechanism into the propagation step.

3. GNN variants using advanced training methods.

   Some of the methods, such as GCN, require the full graph Laplacian, which is computational-consuming for large graphs. To target this problem, works on sampling have been proposed. The ideas include sampling

methods (neighbor sampling, importance-based sampling, and parameterized and trainable sampling, etc.), receptive field control, data augmentation, and unsupervised training (e.g., graph auto-encoders (AEs) that aim at representing nodes into low-dimensional vectors by an unsupervised training manner).

1.2.3 General frameworks

Besides the GNN variants mentioned above, several general frameworks are proposed to integrate different models into one single framework. Figure 1.1 shows a general design pipeline for a GNN model.

1. **Message passing neural networks (MPNNs)**. The MPNN framework is a general framework for supervised learning on graphs by Gilmer et al., 2017, which unifies various GNN and GCN approaches. This framework incorporates some of the most popular models for graph data, such as spectral and non-spectral approaches in graph convolution, GGNN, interaction networks, molecular graph convolutions, deep tensor neural networks and so on.

   The model has two phases: a message passing phase and a readout phase. The message passing phase is also known as the propagation step, which runs for T time steps and is defined in terms of message function M_t and vertex update function U_t. For iteration t, the messages $\mathbf{m}_v^t$ and the updating functions of hidden states $\mathbf{h}_v^t$ are defined as:

$$\begin{aligned} \mathbf{m}_v^{t+1} &= \sum_{w \in \mathcal{N}_v} M_t \left(\mathbf{h}_v^t, \mathbf{h}_w^t, \mathbf{e}_{vw} \right) \\ \mathbf{h}_v^{t+1} &= U_t \left(\mathbf{h}_v^t, \mathbf{m}_v^{t+1} \right) \end{aligned} \tag{1.7}$$

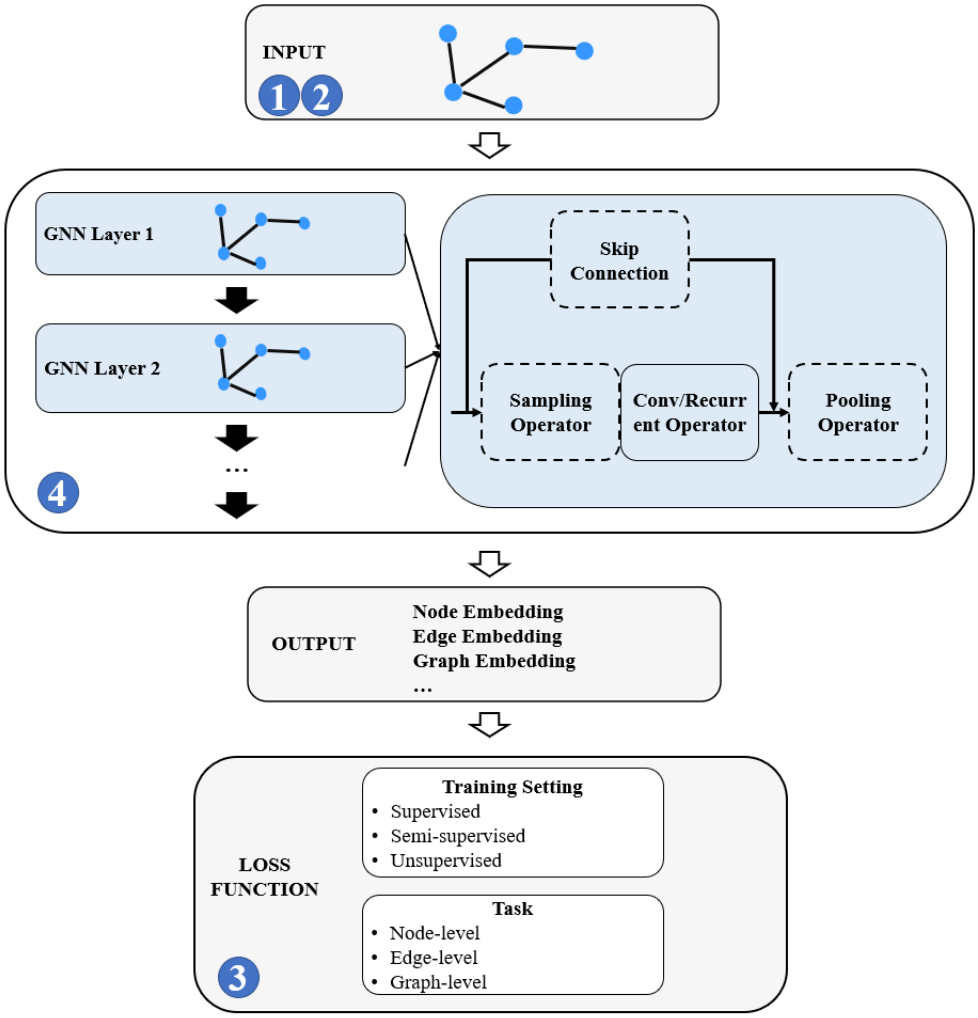

Figure 1.1: GNN design pipeline. Inspired by Zhou et al., 2018.

where $\mathbf{e}_{vw}$ represents features of the edge from node v to w. The readout phase computes a feature vector for the whole graph using the readout function R:

$$\hat{\mathbf{y}} = R\left(\left\{\mathbf{h}_v^T | v \in G\right\}\right) \tag{1.8}$$

where T is the total time steps. The message function M_t, vertex update function U_t and readout function R are generalized expressions and thus can generate different methods.

2. **Non-local neural networks (NLNNs)**. Different from the previous framework, NLNNs capture long-range dependencies with deep neural networks. The non-local operation generalizes the classical non-local mean operation in computer vision. The set of positions can be in space, time or spacetime. The definition of non-local operation can be different, thus leading to various ways of calculating the hidden state of nodes.

3. **Graph Networks (GNs)**. The GN framework generalizes and extends various GNN models. A core part of GN is calle the GN block, which contains three "update" functions, ϕ, and three "aggregation" functions, ρ. The design of GNs has three principles: flexible representations, configurable within-block structure and composable multi-block architectures.

1.2.4 Major GNN applications

Since graph data exists in numerous fields, GNNs have been applied into different areas to solve the real-world problems. Zhou et al., 2018 give a comprehensive summary of GNN applications in various fields of study. Some scenarios include:

1. Structural scenarios where the data has explicit relational structure, such as molecular structures and knowledge graphs.

2. Non-structural scenarios where the relational structure is not explicit, such as image and text. For example, in the image classification problem, we can use knowledge graphs as extra information to guide zero-short recognition classification. Also, we can use the similarity between images in the dataset to solve the few-shot recognition problem by building a weighted full-connected image network based on the similarity and do message passing in the graph.

3. Other scenarios such as generative models and combinatorial optimization problems. NetGAN by Bojchevski et al., 2018 is one of the first work to build neural graph generative model, which generates graphs via random walks.

1.2.5 Open problems

Currently, there are still open problems in the GNN area. First, the GNNs usually have a shallow structure with no more than three layers. Second, dynamic GNN is still being researched and has open problems due to the temporal changing structure of the data. Third, there is still no optimal methods to generate graphs from non-structural raw data such as images and texts. Fourth, it is a challenge for GNNs to scale up, just like many other graph embedding methods. Fifth, graph data in real life are usually gigantic, meaning that labeling the constantly emerging and evolving data can be a huge burden and potential issue. This leads to a fast growing research interest in transfer learning, self-supervised learning, unsupervised learning, etc. In conclusion, there have been

huge improvements during the past few years for the GNN research area but also many challenges to be explored in order to solve some of the most critical open problems.

1.3 Focus of dissertation

This dissertation focuses on the topic of out-of-distribution (OOD) learning on graph data. OOD learning deals with those situations where training and test data follow different distributions. These situations are also called "distribution shifts". Although general OOD machine learning methods have been extensively studied, OOD learning on graph data is still an emerging field with limited findings and resources. This could be due to the natural complexity of graph data: graph data is not structured like images, and it contains multiple types of information such as node attributes, edge attributes, and graph structure. Therefore, it can be hard to directly utilize the existing OOD machine learning methods designed for other types of data. Also, there are very limited benchmark datasets available for graph OOD tasks, both on node level and graph level.

In the meantime, there has been increasing evidence that test nodes do not necessarily follow the same distribution as training nodes. The same can happen to the response variable (e.g., in a node classification problem, the category of a node). In those case, the performance would drop and harm the predictions and applications of graph models. Thus, there is an emerging need for graph OOD learning algorithms to generalize well on OOD data and eliminate the potential performance drop on new data.

Generally, distribution shifts are categorized into two types, i.e., covariate shift and concept shift (Quinonero-Candela et al., 2008, Moreno-Torres

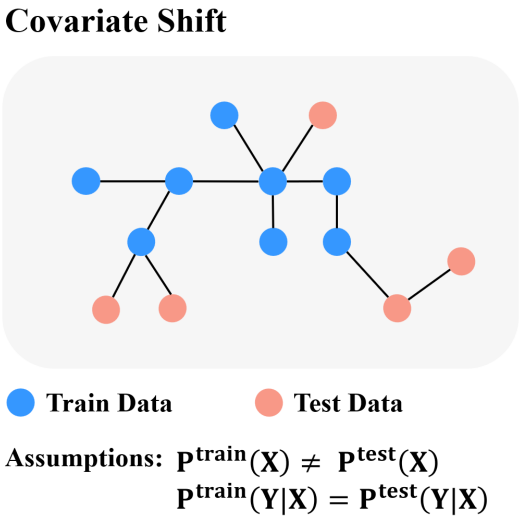

Figure 1.2: Illustration of covariate shift on graph data.

et al., 2012). Let $Y \in \mathcal{Y}$ be the response variable which a model is trained to predict for and $X \in \mathcal{X}$ be the covariate variable. Then the joint distribution $P(Y, X)$ can be written as $P(Y|X)P(X)$. In covariate shift (Figure 1.2), the distributions of X are different between training and testing data, while $P(Y|X)$ stays the same. Covariate shift can be mathematically expressed as $P^{\text{train}}(X) \neq P^{\text{test}}(X)$ and $P^{\text{train}}(Y|X) = P^{\text{test}}(Y|X)$, where $P^{\text{train}}(\cdot)$ and $P^{\text{test}}(\cdot)$ denote training and testing distributions. In concept shift (Figure 1.3), $P(Y|X)$, the conditional distribution, has been shifted while the input covariate stays the same. Concept shift can be expressed as $P^{\text{train}}(X) = P^{\text{test}}(X)$ and $P^{\text{train}}(Y|X) \neq P^{\text{test}}(Y|X)$. There is an essential difference between these two types of distribution shift, and it is natural to have different strategies to solve the OOD problem for them. Gui et al., 2022 have designed graph benchmark data for both covariate shift and concept shift, showing that existing graph OOD methods perform differently across two types of shifts.

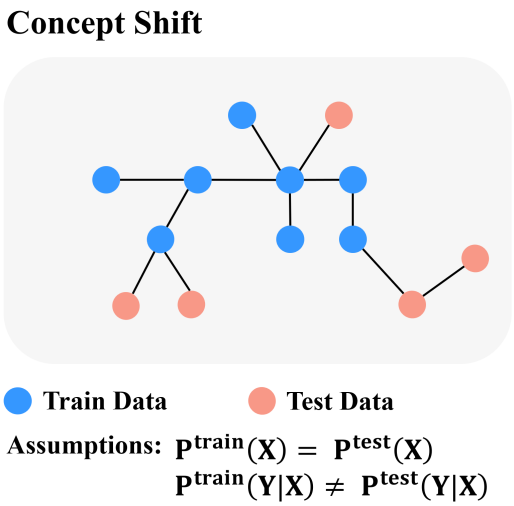

Figure 1.3: Illustration of concept shift on graph data.

1.4 Structure of dissertation

In this chapter, a literature review of graph data related research, especially GNNs, has been summarized and discussed. In the next few chapters, we solve two of the graph out-of-distribution problems, i.e., graph domain adaptation and out-of-distribution detection under the semi-supervised node classification setting. Chapter 2 is focused on addressing one novel and challenging problem, the open-set graph domain adaptation (OS-GDA). An illustration of this problem is shown in Figure 1.4. OS-GDA is a combination of both covariate shift and concept shift, which makes the problem more realistic and challenging. Chapter 3 is focused on out-of-distribution detection under the concept shift scenario, where a combination of semi-supervised node classification problem and an out-of-distribution detection problem are solved at the same time. The last chapter summarizes the works in this dissertation and points out future work directions.

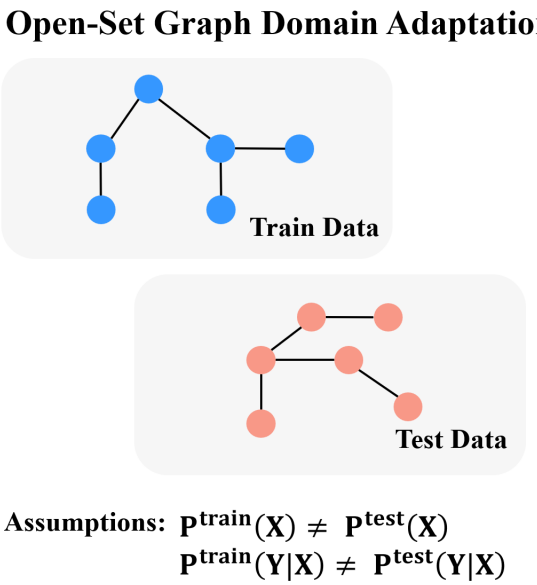

Figure 1.4: Illustration of open-set graph domain adaptation (OS-GDA) problem.

Chapter 2

Open-Set Graph Domain Adaptation via Separate Domain Alignment

2.1 Background

Domain adaptation has become an attractive learning paradigm, as it can leverage source domains with rich labels to deal with classification tasks in an unlabeled target domain. A few recent studies develop domain adaptation approaches for graph-structured data. In the case of node classification task, current domain adaptation methods only focus on the closed-set setting, where source and target domains share the same label space. A more practical assumption is that the target domain may contain new classes that are not included in the source domain. Therefore, in this chapter, we introduce a novel and challenging problem for graphs, i.e., open-set domain adaptive node classification, and propose a new approach to solve it. Specifically, we develop an algorithm

for efficient knowledge transfer from a labeled source graph to an unlabeled target graph under a separate domain alignment (SDA) strategy, in order to learn discriminative feature representations for the target graph. Our goal is to not only correctly classify target nodes into the known classes, but also classify unseen types of nodes into an unknown class. Experimental results on real-world datasets show that our method outperforms existing methods on graph domain adaptation.

Many top-performing machine learning models are trained on large-scale labeled data. However, in practice, labels can be hard to obtain due to the huge cost and/or considerable difficulty of labeling. To handle these challenges, domain adaptation (DA) (Pan and Yang, 2010) is proposed to transfer knowledge from a labeled dataset, namely source domain, to an unlabeled dataset, namely target domain, while domain divergence always exists among source and target domains. DA has drawn much attention in recent years for several reasons. First, traditional machine learning methods require a large amount of labeled data for model training, but unlabeled data from new domains constantly emerge. Second, compared to labeling the data from every new domain, it is more efficient to transfer knowledge from a similar domain that already has sufficient labels. Third, studies have shown promising performance on knowledge transfer using domain adaptation techniques in multiple fields, such as computer vision (Long et al., 2018a), natural language processing (Jiang and Zhai, 2007), and time series data analysis (R. Zhao et al., 2021).

Graph data, which efficiently represents the relationship (edges) between objects (nodes), is ubiquitous and has various applications in the real world. One of the most important learning tasks on graph data is node classification, where the algorithms learn to predict the category of each node. Examples of

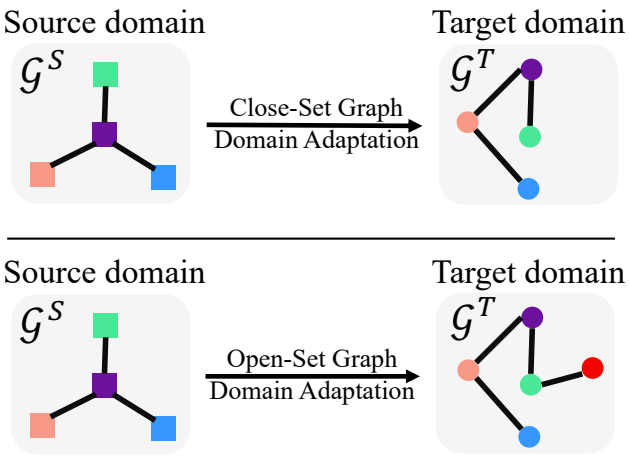

Figure 2.1: Problem setting difference between closed-set and open-set graph domain adaptation. Different colors denote different categories. Different shapes represent different domains.

this task are found in diverse areas, including social networks (Bhagat et al., 2011), citation networks (Ji and Jin, 2016), protein–protein association networks (Szklarczyk et al., 2018), and product co-purchasing networks (Bhatia et al., 2016). Although many algorithms have been developed for supervised and semi-supervised graph learning, the topic of cross-network (or cross-graph) domain adaptation has been largely underexplored. In recent years, a few graph domain adaptation methods are brought up (Dai et al., 2022; Shen et al., 2021; M. Wu et al., 2020). They focus on the closed-set cross-network node classification problem, holding the assumption that the target graph contains nodes of categories only in the source graph, however, which is unrealistic. In many real-world applications, the target graph always contains novel nodes that are out of the label space of the source graph. Inspired by this, we introduce the concept of open-set (Geng et al., 2020) cross-network node classification. Specifically, we allow the target graph to contain nodes of unknown classes that don't belong to the source label space. Meanwhile, we require the learned model to not only

correctly classify target node if its label is in the source label space but also to be able to identify it as "unknown". The difference between closed-set domain adaptation and open-set graph domain adaptation is illustrated in Figure 2.1.

To address the new challenging problem, i.e., Open-Set Graph Domain Adaptation (OS-GDA), we propose a novel separate domain alignment (SDA) framework. An overview is shown in Figure 2.2. Rather than directly aligning source and target domains without considering target unknown class nodes, SDA provides different domain alignment strategies for different target nodes. Specifically, we roughly split the target nodes into two groups based on their entropy values of the classifier's outputs, i.e., certain group and uncertain group. The data in certain group have smaller entropy values compared with threshold while those in uncertain group have larger entropy values. In principle, entropy estimates the prediction uncertainty. The smaller the entropy, the higher the certainty of the prediction. For certain group, we utilize adversarial learning to align them to the source domain. For the uncertain group, we propose a neighbor center clustering method to better separate target data from known classes and those from unknown classes. In this way, the target data coming from known classes would align to the source domain while those from unknown classes would be far from data of known classes.

Our contributions are summarized as follows:

- We introduce a practical and challenging task, namely open-set graph domain adaptation (OS-GDA), that allows target graphs to contain unknown class nodes.
- We propose the Separate Domain Alignment (SDA) framework, which provides suitable domain alignment strategies for different target nodes.

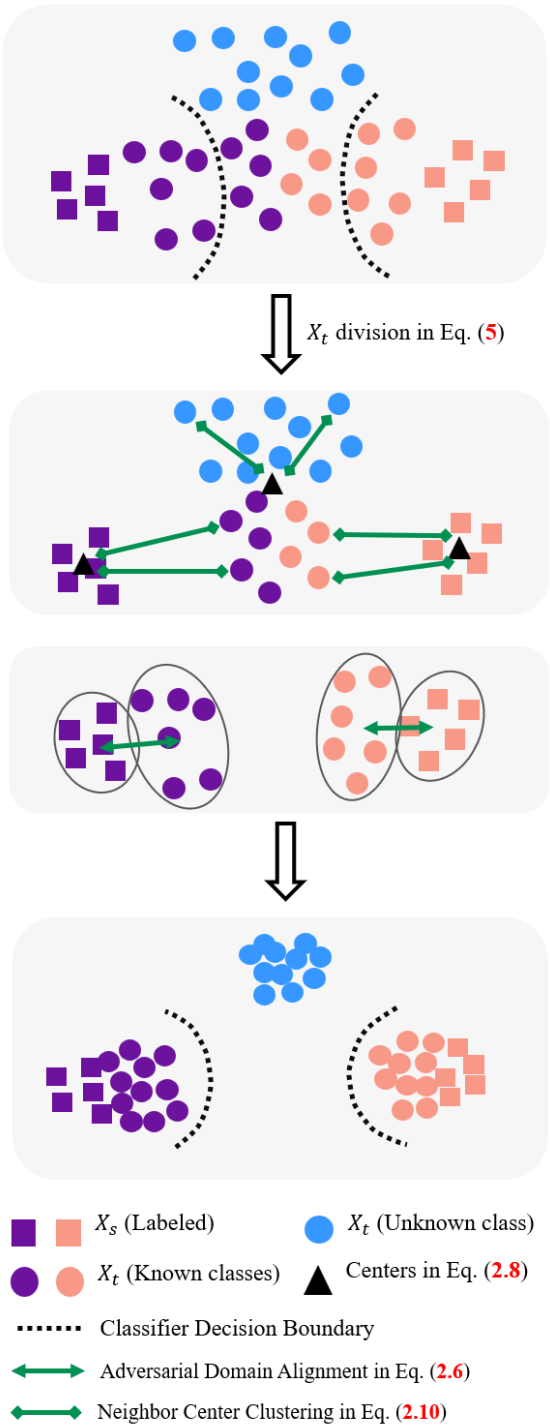

Figure 2.2: Illustration of the proposed separate domain alignment (SDA), which consists of three parts: (1) dividing target nodes into two groups, i.e., certain group and uncertain group; (2) for certain group, we utilize adversarial domain alignment to align target nodes to source nodes; (3) for uncertain group, we propose a neighbor center clustering loss to cluster target nodes. Best viewed in color.

- We conduct extensive experiments and show that our method successfully tackles the novel OS-GDA problem and surpasses all baselines with large margins.

2.2 Related work

2.2.1 Graph Neural Networks

Since GNN was first brought up to extend general neural network methods to graph domains (Scarselli et al., 2009), numerous GNN algorithms have been developed and shown impressive performance in graph learning tasks such as node or edge classification, link prediction, community detection, and regression. Representative GNN methods include ChebNet (Defferrard et al., 2016), graph convolutional network (GCN) (Kipf and Welling, 2017), and GraphSAGE (Hamilton et al., 2017), which leverage the node adjacency matrix and analogously define convolution operators on graphs in either spectral or spatial spaces. After GCN bridged the gap between spectral-based methods and spatial-based methods, spatial-based methods have also gained huge popularity due to its attractive efficiency, flexibility, and generality (Hamilton et al., 2017; Z. Wu et al., 2021). However, these GNN modes would fail to solve the cross-network problem due to the domain divergence between training graph and testing graph (M. Wu et al., 2020).

2.2.2 Closed-set domain adaptation

Many DA methods have been proposed and achieved success in various fields. Most existing methods belong to closed-set domain adaptation (CS-DA), which aims to reduce the domain divergence between source and target domains and

extract domain-invariant features (Ganin and Lempitsky, 2015; Long et al., 2015). They can be roughly divided into two categories: (1) moment matching based methods which statistically match the data distributions such as maximum mean discrepancy (MMD) (Gretton et al., 2012) and CORrelation ALignment (CORAL) (Sun et al., 2016); (2) adversarial learning based methods which play a minimax game between feature extractor and domain discriminator to learn domain-invariant features such as Domain Adversarial Neural Networks (DANN) (Ganin et al., 2016) and Conditional Domain Adversarial Networks (CDAN) (Long et al., 2018a).

Compared to DA in CV and NLP fields, graph domain adaptation is a relatively new topic. Only a few methods have been proposed for the closed-set graph domain adaptation (CS-GDA), i.e., closed-set cross-network node classification. All of them are based on the adversarial domain alignment idea and try to mitigate the domain divergence between source and target graphs. Cross-network deep network embedding (CDNE) (Shen et al., 2021) utilizes autoencoder as the network and employs MMD to learn domain-invariant embeddings for cross-network node classification. Adversarial domain adaptation with graph convolutional networks (AdaGCN) (Dai et al., 2022) utilizes GCN and adversarial domain adaptation to jointly model network structures and node attributes. Unsupervised domain adaptive graph convolutional networks (UDAGCN) (M. Wu et al., 2020) proposes a dual GCN to jointly exploit local and global consistency for feature aggregation. Adversarial Separation Network (ASN) (X. Zhang et al., 2021) utilizes an adversarial separation network to explicitly separate domain-private and domain-shared information by introducing both a shared encoder and two private encoders. However, all of these methods

require the source and target graphs to share the same label space, which is not practical in real-world applications.

2.2.3 Open-set domain adaptation

In early open-set domain adaptation (OS-DA) definition (Panareda Busto and Gall, 2017), both source and target domains have private label spaces, respectively, and the common label space is known. Later, the setting of OS-DA (Saito et al., 2018) is adjusted by claiming no source private label space, which means target label space contains source label space. In other words, source label space is the subset of target label space. The goal of OS-DA is to learn a model with source and target domains that can not only correctly classify target data if it belongs to source label space but also successfully identify target data from unknown classes. Recent OS-DA (Bucci et al., 2020; Liu et al., 2019) methods mainly focus on the later challenging setting.

Different from CS-DA, open-set domain adaptation (OS-DA) (Panareda Busto and Gall, 2017; Saito et al., 2018) allows target domain to include data that are out of source label space. In other words, source label space is the subset of target label space. The goal of OS-DA is to learn a model with source and target domains that can not only correctly classify target data if it belongs to source label space but also successfully identify target data from unknown classes.

In the graph domain adaptation field, existing methods, e.g., CDNE, UDA-GCN, and ASN, hold the strong assumption that source and target graphs share the same label space, which is not realistic in real-world applications. Oftentimes, this assumption cannot hold due to the fact that target graph contains unknown class nodes for source label space. This leads to the open-set problem in cross-network node classification, which allows target data to contain nodes

with classes that are out of source label space. To the best of our knowledge, this challenging problem has not been solved yet.

Most efforts on domain adaptation have been focused on the closed-set domain adaptation (CSDA) (e.g., Dong et al., 2020), whose assumption is that source and target domains share the same label space. This is not always appropriate in the real-world scenarios. It is a more realistic assumption that data from new domains contain both classes that are seen in the source domain and some novel classes that are never seen before. If using a CSDA method, we will incorrectly classify samples from unseen classes into one of the existing classes. Following that, open-set domain adaptation (OSDA) has been widely investigated (Kundu et al., 2020; Panareda Busto and Gall, 2017; Saito et al., 2018), assuming that the label space of source domain is a subset of target domain's label space. OSDA methods attempt to identify outliers that are unseen during training (Hendrycks and Gimpel, 2016; Liang et al., 2018). Also, multiple studies show that self-supervision and contrastive learning are useful to separate outliers from inliers (Hendrycks et al., 2019; Tack et al., 2020).

In the graph domain adaptation field, all the pioneering methods mentioned above (CDNE, AdaGCN, UDAGCN, and ASN) are developed for CSDA. Similarly, this is not always appropriate - often times there are new node classes in the target domain graph that are not seen in the source domain graph, which is an OSDA problem. To the best of our knowledge, this more challenging and more realistic problem of open-set domain adaptive cross-network node classification has not been solved yet. To tackle the OSDA problem, we must separate target nodes that do not belong to any of the known classes. This is challenging because we do not have any prior knowledge about the novel classes. Meanwhile, we also need to align the distributions of source and target nodes of

known classes. These two tasks have to be optimized together to achieve ideal performance.

2.3 Method

Unlike the closed-set graph domain adaptation (CS-GDA) problem, we target at open-set problem in graph domain adaptation where the target domain contains categories that do not belong to source label space. Compared with CS-GDA, open-set graph domain adaptation (OS-GDA) is more challenging and practical, since we cannot always guarantee all target nodes in the source label space. As a result, the task of OS-GDA is not only to align distributions of source and target domains, but also to identify target nodes that are out of source label space. Here, we propose a novel separate domain alignment (SDA) scheme to address the OS-GDA problem. Figure 2.3 illustrates the overall framework. Firstly, we introduce the preliminary. Then, we recap the details of local and global node embedding. Last, we illustrate our newly proposed SDA and summarize the overall pipeline of our framework.

2.3.1 Preliminaries

In OS-GDA problem, we focus on the graph node classification task. We assume an undirected graph $\mathcal{G}$ with node set V and edge set $E \subseteq V \times V$. $N_v = |V|$ and $N_e = |E|$ denote the number of nodes and edges in graph. Let $\mathbf{A} \in R^{N_v \times N_v}$ be the adjacency matrix of $\mathcal{G}$, where each element $\mathbf{A}_{ij} = \mathbf{A}(i,j)$ indicates the connectivity of node v_i and node v_j. $\mathbf{A}_{i,j} = 1$ if edge $(v_i, v_j) \in E$, otherwise $\mathbf{A}_{i,j} = 0$. $\mathbf{X} \in R^{N_v \times d}$ represents the content features of V where d is the feature dimension. Y is the label set for V that comes from label space C.

Source Graph: Let $\mathcal{G}_s = (V_s, E_s, \mathbf{A}_s, \mathbf{X}_s, Y_s)$ indicate the labeled source network with node set V_s, edge set E_s, and label matrix Y_s with label space C_s.

Target Graph: Let $\mathcal{G}_t = (V_t, E_t, \mathbf{A}_t, \mathbf{X}_t)$ be the unlabeled target network with unlabeled node set V_t and edge set E_t. The target label space is denoted as C_t.

Open-Set Graph Domain Adaptation: Different from CS-GDA which requires source and target graphs to share the same label space, i.e., $C_s = C_t$, OS-GDA relaxes this claim by allowing target graph to contain nodes from classes out of source label space, i.e., $C_s \subset C_t$. Specifically, we denote the known label space shared by both domains as $C = C_s \cap C_t = C_s$ and the unknown label space for target graph as $\bar{C}_t = C_t \setminus C_s$. The goal of OS-GDA is to train the model with $\mathcal{G}_s$ and $\mathcal{G}_t$, classify target nodes into $|C| + 1$ categories, where $\bar{C}_t$ are gathered as one unknown class, and require the learned model to classify the target node correctly if it is associated with a label in C, or identify it as "unknown" otherwise. In general, the model consists of three modules, i.e., G, F, and O. Here, $G : x \to g$ represents the graph feature extractor that maps the content feature of node x into an embedding space, $F : g \to f$ is the classifier using input embedding to predict the category, and domain discriminator O is for adversarial domain alignment.

2.3.2 Recap of local and global node embedding

Many methods (M. Wu et al., 2020; Xu et al., 2021; X. Zhang et al., 2021; Y. Zhang et al., 2020) have proven the advantage of combining local and global graph information to learn the semantic node embedding. The core idea is to both recover the local 1-hop neighbor information and extract the global topological features from the given graph. Furthermore, the attention mechanism is

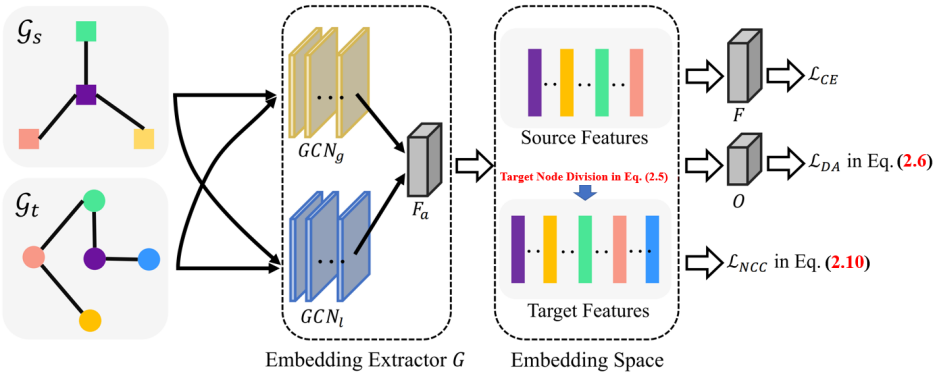

Figure 2.3: Illustration of our novel Separate Domain Alignment (SDA) scheme for open-set graph domain adaptation, which includes three losses: cross-entropy loss $\mathcal{L}_{CE}$, domain alignment loss $\mathcal{L}_{DA}$, and neighbor center clustering loss $\mathcal{L}_{NCC}$, and three modules: feature extractor $G = \{GCN_g, GCN_l, F_a\}$, classifier F, and domain discriminator O.

adopted to aggregate the local and global embeddings to capture the semantic information from both aspects. Specifically, the graph convolution network (GCN) (Kipf and Welling, 2017) is directly utilized to capture the graph local information while the positive pointwise mutual information (PPMI) based GCN (Zhuang and Ma, 2018) is applied to excavate the graph global information.

Local GCN (GCN_l)

To capture the local information in a graph, We directly utilize the GCN model proposed by Kipf and Welling, 2017 and formulate the local GCN GCN_l as a type of feed-forward neural networks. Give the graph $\mathcal{G} = (V, E, \mathbf{A}, \mathbf{X}, Y)$, the output of the i-th hidden layer $\mathbf{Z}_l^{(i)}$ of the GCN_l is defined as:

$$\mathbf{Z}_l^{(i)}(\mathbf{X}) = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}_l^{(i-1)}\mathbf{W}_l^{(i)}), \tag{2.1}$$

where $\sigma(\cdot)$ is the activation function, $\tilde{\mathbf{A}} = \mathbf{A}+\mathbf{I}_N$ denotes the adjacency matrix with self-loops ($\mathbf{I}_N$ is an identity matrix), $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ and $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is the symmetric normalized adjacency matrix, $\mathbf{Z}_l^{(i-1)}$ indicates the output of the $(i-1)$-th layer and $\mathbf{Z}_l^0 = \mathbf{X}$, and $\mathbf{W}_l^{(i)}$ represents the learnable parameters of the i-th layer.

Global GCN (GCN_g)

To excavate the global topological features, we introduce the PPMI-based GCN (Zhuang and Ma, 2018) which utilizes the PPMI matrix $\mathbf{P}$ to evaluate the topological proximity between nodes within k steps in a given graph $\mathcal{G}$. Please refer to Zhuang and Ma, 2018 for more details of $\mathbf{P}$.

With calculated $\mathbf{P}$, we also formulate the global GCN GCN_g as a type of feed-forward neural networks, which is defined as follows:

$$\mathbf{Z}_g^{(i)}(\mathbf{X}) = \sigma(\mathbf{D}^{-\frac{1}{2}}\mathbf{P}\mathbf{D}^{-\frac{1}{2}}\mathbf{Z}_g^{(i-1)}\mathbf{W}_g^{(i)}), \tag{2.2}$$

where $\sigma(\cdot)$ is the activation function, $\mathbf{P}$ denotes the PPMI matrix, $\mathbf{D}_{ii} = \sum_j \mathbf{P}_{ij}$ is the normalized matrix, $\mathbf{Z}_g^{(i-1)}$ represents the output of the $(i-1)$-th layer and $\mathbf{Z}_g^{(0)} = \mathbf{X}$, and $\mathbf{W}_g^{(i)}$ is the trainable parameters of the i-th layer.

Embedding Attention (F_a)

To further excavate the contribution of both embeddings, i.e., local embedding $\mathbf{Z}_l$ and global embedding $\mathbf{Z}_g$, and generate a unified node embedding space, an attention layer F_a is introduced. It takes $\mathbf{Z}_l$ and $\mathbf{Z}_g$ as input and produces weight coefficients α_1 and α_2 for $\mathbf{Z}_l$ and $\mathbf{Z}_g$, respectively:

$$[\alpha_1, \alpha_2] = F_a([\mathbf{Z}_l, \mathbf{Z}_g]). \tag{2.3}$$

The unified node embedding is the combination of local and global embeddings with their corresponding weight coefficients:

$$\mathbf{Z} = \frac{\exp(\alpha_1)}{\exp(\alpha_1) + \exp(\alpha_2)}\mathbf{Z}_l + \frac{\exp(\alpha_2)}{\exp(\alpha_1) + \exp(\alpha_2)}\mathbf{Z}_g. \tag{2.4}$$

Figure 2.3 illustrates the details of how to extract the node embedding. For simplicity, we utilize $G = \{GCN_l, GCN_g, F_a\}$ to denote the node embedding extractor.

2.3.3 Separate Domain Alignment

Previous CS-GDA methods (M. Wu et al., 2020; X. Zhang et al., 2021) utilize adversarial learning to align source and target graphs and minimize the entropy value of target nodes to force the classifier to pass through the low-density regions of the target embedding space. As a result, these methods cannot be applied to our proposed OS-GDA problem. Without considering the target nodes from unknown label space $\bar{C}_t$, directly aligning source and target domains would lead to a negative knowledge transfer.

To solve the problem, we propose a novel separate domain alignment (SDA) scheme that enables the model to align known-class target nodes to source nodes while separating unknown-class target nodes from known-class target nodes. SDA consists of two alignment operations, i.e., adversarial domain alignment and neighbor center clustering, for different target nodes.

Inspired by Grandvalet and Bengio, 2004, we use entropy to evaluate the uncertainty of classifier prediction. The lower the entropy value, the more certain the prediction. Target nodes with lower entropy value would be more likely to belong to known label space C while those with higher entropy are more likely

to come from unknown label space $\bar{C}_t$. We employ the threshold γ to adaptively divide the target nodes into two groups, i.e, certain group and uncertain group, by utilizing the entropy $e_t = H(f_t)$ where $H(f_t) = -\sum_{i=1}^{|C_s|} f_t^i \log(f_t^i)$, as the follows:

$$x_t \in \begin{cases} Group_c, & e_t < \gamma \\ Group_u, & e_t \geq \gamma. \end{cases} \tag{2.5}$$

Instead of tuning the hyper-parameter γ to divide target nodes, we set γ to $\frac{\log(|C_s|)}{2}$ where $|C_s|$ denotes the number of source classes and $\log(|C_s|)$ is the maximum entropy value of the classifier.

Adversarial domain alignment for certain group

For target nodes from certain group $Group_c$, there is a high probability that these nodes belong to known label space C. We utilize adversarial domain alignment to learn domain invariant embedding for nodes from C. Specifically, we employ a domain discriminator $O(\cdot)$ to play a minimax game with embedding extractor $G(\cdot)$. The objective function is defined as:

$$\mathcal{L}_{adv} = E_{x_s \in X_s} \log(O(G(x_s)) + E_{x_t \in Group_c} \log(1 - O(G(x_t)). \tag{2.6}$$

The domain discriminator tries to identify the source and target nodes while the embedding extractor aims to fool the domain discriminator. The overall process is:

$$\min_O \max_G \mathcal{L}_{adv}. \tag{2.7}$$

Neighbor center clustering for uncertain group

For target nodes from uncertain group $Group_u$, directly utilizing adversarial learning to align them to source domain could cause negative transfer, because $Group_u$ contains target nodes from both known and unknown classes. Violently enforcing target unknown class nodes to align with source nodes will deteriorate the learned domain-invariant embedding. To address this challenge, we exploit a novel neighbor center clustering (NCC) to better identify target nodes from known class and those from unknown class while softly aligning target nodes to source nodes. The main idea of our NCC is to move each target node in $Group_u$ either to source class centers or to cluster centers in $Group_u$. The target unknown class nodes are more likely to share similar semantic information with the centers which are close to ground truth unknown class centers. Likewise, those from known class would possibly have similar characteristics with the known class centers.

Given the uncertain group $Group_u$, we utilize K-means to group them into K clusters and obtain corresponding embedding centers $\{\mu_t^1, ..., \mu_t^K\}$. Meanwhile, we utilize the weight vectors $W_f = [w_f^1, ..., w_f^{|C_s|}]$ in the classifier F as source class centers. Let M represents the center matrix which consists of cluster centers from $Group_u$ and source class centers:

$$M = [\mu_t^1, ..., \mu_t^K, w_f^1, ..., w_f^{|C_s|}], \tag{2.8}$$

where both μ_t^i and w_f^j are L2-normalized.

Given a target node embedding g_t^i from $Group_u$ and the center matrix M, the probability that the j-th center m_j in M is the neighbor center of g_t^i is,

$$p_{i,j} = \frac{\exp\left(\langle g_t^i, m_j \rangle / \tau\right)}{\sum_{k=1}^{K+|C_s|} \exp\left(\langle g_t^i, m_k \rangle / \tau\right)}, \tag{2.9}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product between two vectors to measure their similarity, τ denotes the temperature parameter which is empirically set as 0.05. Eventually, the neighbor center clustering loss is formulated as:

$$\mathcal{L}_{ncc} = -\sum_{i=1}^{N_{group_u}} \sum_{i=j}^{K+|C_s|} p_{i,j} \log\left(p_{i,j}\right), \tag{2.10}$$

where N_{group_u} is the number of target nodes in uncertain group $Group_u$. By minimizing the above loss, the learned target embedding space will be more discriminative and benefit the target unknown class identification.

2.3.4 Overall Objectives

Our overall loss objective consists of three items. Cross-entropy loss $\mathcal{L}_{ce}$ is applied for the source graph. Adversarial domain alignment loss $\mathcal{L}_{adv}$ and neighbor center clustering loss $\mathcal{L}_{ncc}$ in Eq. (2.6) and Eq. (2.10) are used for source and target graphs. Therefore, the overall loss function can be written as:

$$\mathcal{L}_{SDA} = \mathcal{L}_{ce} + \mathcal{L}_{adv} + \beta \mathcal{L}_{ncc}, \tag{2.11}$$

where β is hyper-parameters for $\mathcal{L}_{ncc}$. Our method procedure is summarized in Algorithm 2.

Algorithm 2 Seperate Domain Alignment (SDA) Algorithm

Input: $\mathcal{G}_s$, $\mathcal{G}_t$, γ, β, K, initialized G, F, O.
Output: Learned G and F

1: **for** $epoch = 1$ to $epochs$ **do**
2: Extract node embedding by utilizing G.
3: Apply Eq. (2.5) for target node division.
4: Apply Eq. (2.7) for target certain group alignment.
5: Apply Eq. (2.10) for target uncertain group alignment.
6: **end for**
7: **return** G and F

2.3.5 Inference

In the testing phase, given each target node x_t with its classifier output f_t. If its entropy value is larger than the threshold γ, which is the same as the one in Eq. (2.5), x_t will be marked as unknown class. Otherwise, it will be assigned to a class in the source label space C_s depending on f_t.

2.4 Experiments

In this section, we evaluate the effectiveness of our method as the following: Firstly, we introduce the experimental settings. Next, we compare our method to other methods. Then, we provide an extensive ablative study investigating each of our proposed modules. Last, we present the hyper-parameters sensitivity study.

2.4.1 Experimental settings

Datasets: We leverage three commonly used paper citation networks, i.e., **A**CMv9 (between years 2000 and 2010), **D**BLPv7 (between years 2004 and 2008), and

Table 2.1: Statistics of datasets for experiment.

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

Citationv1(before the year 2008), provided by ArnetMiner (Tang et al., 2008) and construct graphs based on these citation networks following X. Zhang et al., 2021. These networks come from three different original sources (ACM, DBLP, and Microsoft Academic Graph, respectively, suggested by the dataset names). we utilize the same dataset processing techniques as X. Zhang et al., 2021. Each node in the citation networks belongs to one of the following five categories based on its research topic: "Databases" (**DB**), "Artificial Intelligence" (**AI**), "Computer Vision" (**CV**), "Information Security" (**IS**), and "Networking" (**NW**). To induce the domain divergence between different citation networks, the extracted nodes are from different publish time window, i.e., DBLPv7 is between years 2004 and 2008, ACMv9 is between years 2000 and 2010, and Citationv1 is before year 2008. The node attributes are obtained by extracting sparse bag-of-words features from the article title. Note that the original dimensions of attributes are different. We unify the dimension of attributes by following X. Zhang et al., 2021. Tables 2.1 and 2.2 show the details of the datasets. There exists domain discrepancy between each pair of them. We consider them as undirected networks with each node indicating a paper and each edge denoting a citation relation between two nodes. Each node belongs to one of five categories according to its research topics, including "Artificial Intelligence", "Computer Vision", "Databases", "Information Security", and "Networking".

Table 2.2: Label distribution of datasets for experiment.

Datasets	Label Distribution (%)				
	1	2	3	4	5
DBLPv7	21.66	32.97	23.83	6.05	15.48
ACMv9	20.47	29.56	22.54	8.58	18.85
Citationv1	25.32	26.00	22.43	7.72	18.53

Table 2.3: Enumerating target unknown classes label space.

# Situation	1	2	3	4	5
Unknown Classes	DB, AI	DB, CV	DB, IS	DB, NW	AI, CV
# Situation	6	7	8	9	10
Unknown Classes	AI, IS	AI, NW	CV, IS	CV, NW	IS, NW

Protocols: We introduce an open-set protocol for the experiment by setting the size of source label space C_s to 3 and keeping the size of target label space C_t as 5. The experiment is conducted under six domain adaptation tasks: A→D, D→A, A→C, C→A, C→D, and D→C. The size of source label space is 3 and the size of target label space is 5. For each task, we evaluate the method under every possible target unknown class label space, which includes 10 situations shown in Table 2.3. The reported results are calculated by averaging over 10 runs with different target unknown classes label space.For each domain adaptation task, we evaluate all methods by enumerating every possible C_s, which include 10 combinations.

Baselines: We compare with four main streams of the state-of-the-art methods: (1) *Graph Node Classification* methods, namely Graph Convolutional Networks (**GCN**) (Kipf and Welling, 2017) and **GraphSAGE** (Hamilton et al., 2017). (2) *Unsupervised Domain Adaptation* methods, namely Domain Adver-

sarial Neural Networks (**DANN**) (Ganin et al., 2016) and Conditional Domain Adversarial Network (**CDAN**) (Long et al., 2018b). (3) *Open-set Domain Adaptation* methods, namely Open Set Domain Adaptation by Backpropagation (**OSDAB**) (Saito et al., 2018) and Domain Adaptative Neighborhood Clustering via Entropy optimization (**DANCE**) (Saito et al., 2020). (4) *Closed-set Graph Domain Adaptation* methods, Unsupervised Domain Adaptive Graph Convolutional Networks (**UDAGCN**) (M. Wu et al., 2020) and Adversarial Separation Network (**ASN**) (X. Zhang et al., 2021).

Evaluation Metrics: We use four metrics, i.e., average class accuracy over all classes (Acc), average class accuracy on known classes (Acc_k), average class accuracy on unknown class (Acc_u), and h-score (HS) (Fu et al., 2020), to evaluate the performance of all methods. The Acc is the mean of per-class accuracy over known and unknown classes, which would fail to truly discover the ability of unknown class identification for the methods. Because it gives equal weight for per known class accuracy and unknown class accuracy which leads the value of Acc to be dominated by known classes accuracy. Thus, we introduce the HS to address the importance of both Acc_k and Acc_u by computing the harmonic mean of them:

$$HS = \frac{2 \times Acc_k \times Acc_u}{Acc_k + Acc_u}. \tag{2.12}$$

HS value is high only when both Acc_k and Acc_u are high. In our experiment, we report the averaged results of 10 runs by enumerating every possible source label space C_s.

Implementation Details: Our implementation is based on Pytorch (Paszke et al., 2019)[2]. We utilize the same graph embedding extractor structure as ASN (X. Zhang et al., 2021) which includes local GCN GCN_l, global GCN GCN_g, and attention layer F_a. Both GCN_l and GCN_g are two-layer structures, the hidden

[2] https://pytorch.org/.

dimensions for two layers in GCN_l and GCN_g are set as 128 and 16, respectively. The dropout rate is defined as 0.5. For a fair comparison, the same dimensions are also set for other baselines. We optimize the model for 100 epochs by using Adam optimizer with the learning rate of 0.005, momentum of 0.9, and weight decay of 5×10^{-4}. The hyper-parameter γ, τ and β are set as $\frac{\log(3)}{2}$, 0.05, and 0.05, respectively. The number of steps k in PPMI matrix for GCN_g is defined as 3, which is the same as ASN.

2.4.2 Results and analysis

Quantitative comparisons are shown in Table 2.4 from the aspects of Acc and HS, and Table 2.5 from the aspects of Acc_k and Acc_u. We group the methods on top four rows compared with ours on the last row. On the first row, there are two graph node classification methods, i.e., GCN and GraphSAGE. On the second row, there are two unsupervised domain adaptation methods: DANN and CDAN. On the third row, we present two state-of-the-art open-set domain adaptation methods in the computer vision field (on image data), which are DANCE and OSDAB. On the fourth row, we present two cutting-edge closed-set graph domain adaptation methods: UDAGCN and ASN.

In Table 2.4, we observe that our method consistently outperforms all the compared methods with a significant margin. For example, we get 4.92% better than ASN and 24.30% better than UDAGCN in terms of Acc. Checking HS, we see 8.39% and 27.68% performance gains compared with ASN and UDAGCN. Further, surprisingly the performances of GCN and GraphSAGE are approaching or surpassing UDAGCN and ASN in terms of HS, which reveals the negative transfer problem among CS-GDA methods when target domain contains unknown classes. This phenomenon is caused by violently

Table 2.4: Results (%) on six open-set graph domain adaptation tasks in terms of Acc and HS.

Methods	A → D		D → A		A → C		C → A		C → D		D → C		Average	
	Acc	HS	Acc	HS	Acc	HS	Acc	HS	Acc	HS	Acc	HS	Acc	HS
GCN	45.10	41.80	38.95	39.52	46.36	43.91	44.14	43.66	48.45	44.61	42.26	41.25	44.21	42.46
GraphSAGE	48.26	46.22	43.14	42.84	50.60	49.04	48.13	46.36	51.72	48.66	47.20	46.70	48.17	46.64
DANN	33.30	28.07	34.58	36.53	39.64	41.22	34.47	34.42	36.92	41.88	35.20	35.46	35.68	34.16
CDAN	31.13	21.65	29.03	27.76	30.99	26.00	31.72	30.91	35.69	30.47	28.62	21.82	31.20	26.44
DANCE	60.54	25.99	**53.27**	39.53	63.23	39.15	60.44	35.88	64.29	28.98	57.62	39.50	59.90	34.84
OSDAB	28.56	11.27	26.20	12.91	29.32	11.15	27.80	7.34	33.81	18.89	28.63	14.16	29.05	12.62
UDAGCN	36.20	26.59	31.90	12.31	37.44	32.01	35.64	22.76	41.88	36.48	35.50	25.09	36.43	25.87
ASN	56.40	37.55	47.49	43.93	59.88	49.82	57.51	47.87	56.65	45.62	56.97	46.19	55.81	45.16
Ours (SDA)	**61.60**	**49.22**	51.36	**50.86**	**64.47**	**55.27**	**61.67**	**55.89**	**67.51**	**55.35**	**57.74**	**54.72**	**60.73**	**53.55**

Table 2.5: Results (%) on six open-set graph domain adaptation tasks in terms of Acc_k and Acc_u.

Methods	A → D		D → A		A → C	
	Acc_k	Acc_u	Acc_k	Acc_u	Acc_k	Acc_u
GCN	47.65	37.44	38.22	41.12	48.40	40.25
GraphSAGE	49.55	**44.36**	42.88	43.92	51.87	46.78
DANN	36.16	24.69	31.49	43.85	37.11	47.24
CDAN	35.51	18.00	29.31	28.21	33.33	23.98
DANCE	**74.27**	19.35	**59.25**	35.32	**74.57**	29.21
OSDAB	35.25	8.61	32.06	8.62	36.40	8.07
UDAGCN	37.29	32.93	38.97	10.70	36.78	39.42
ASN	63.41	35.35	47.12	48.58	64.74	45.30
Ours (SDA)	68.23	41.68	49.68	**56.40**	69.75	**48.64**

Methods	C → A		C → D		D → C		Average	
	Acc_k	Acc_u	Acc_k	Acc_u	Acc_k	Acc_u	Acc_k	Acc_u
GCN	44.57	42.85	51.41	39.56	43.04	39.91	45.55	40.19
GraphSAGE	49.44	44.20	53.87	45.28	47.19	47.22	49.13	45.29
DANN	31.79	42.52	41.37	23.00	34.11	38.48	35.34	36.63
CDAN	32.09	30.58	38.81	26.33	31.95	18.62	33.50	24.28
DANCE	**70.51**	30.23	**79.06**	19.98	**66.57**	30.65	**70.70**	27.47
OSDAB	35.67	4.19	40.28	14.38	33.62	13.65	35.55	9.59
UDAGCN	40.55	20.88	41.82	42.06	37.86	28.42	38.88	29.07
ASN	61.91	44.35	62.01	40.45	61.60	43.04	60.13	42.85
Ours (SDA)	63.34	**56.66**	73.74	**48.82**	57.84	**57.44**	63.76	**51.61**

enforcing source and target data to align without considering the difference between their label spaces.

Different from Table 2.4, Table 2.5 demonstrates the advantage of our methods from the aspects of Acc_k and Acc_u. We organize the experimental results in the same layout as Table 2.4. In this evaluation, we again find that our method consistently and significantly outperforms all the compared methods regarding Acc_u. It is worth noting that DANCE has the best performance regarding Acc_k throughout the tasks but performs poorly on identifying the target nodes from unknown classes. Compared to the most competitive opponent ASN, our method surpasses by 3.63% on Acc_k and 8.76% on Acc_u. The same trend of negative transfer problem is also observed comparing the first and third groups. Furthermore, compared with all baselines, we find that our method holds the significant advantage of not only correctly classifying target nodes that belong to known classes but also successfully identifying the target nodes from unknown classes.

2.4.3 Ablation study

We conduct ablation studies to examine the effectiveness of the proposed $\mathcal{L}_{SDA}$ in Eq. (2.11) and show the results in Table 2.6. Firstly, compared with $\mathcal{L}_{ce}$, both $\mathcal{L}_{ce} + \mathcal{L}_{ncc}$ and $\mathcal{L}_{ce} + \mathcal{L}_{adv}$ gain significant improvement over four evaluation metrics. Especially, We can see 6.97% and 9.64% performance gains in terms of HS, which proves the effectiveness of proposed domain alignment strategies for target nodes from different groups. Additionally, comparing "Ours" to $\mathcal{L}_{ce} + \mathcal{L}_{ncc}$ and $\mathcal{L}_{ce} + \mathcal{L}_{adv}$, we achieve another significant improvement over four evaluation metrics, which further verifies the power of our SDA. Overall, each of the incremental combination demonstrates the effectiveness of the com-

Table 2.6: Ablation study for separate domain alignment on D→A domain adaptation task. $\mathcal{L}_{ce}$ is the cross-entropy loss objective. $\mathcal{L}_{adv}$ represents the adversarial domain alignment loss objective. $\mathcal{L}_{ncc}$ denotes the neighbor center clustering loss objective.

Loss Objectives	Acc_k	Acc_u	Acc	HS
$\mathcal{L}_{ce}$	44.03	42.42	43.69	38.70
$\mathcal{L}_{ce} + \mathcal{L}_{ncc}$	49.45	48.74	49.27	45.67
$\mathcal{L}_{ce} + \mathcal{L}_{adv}$	48.32	54.04	49.69	48.34
$\mathcal{L}_{ce} + \mathcal{L}_{ncc} + \mathcal{L}_{adv}$ (Ours)	49.68	56.40	51.36	50.86

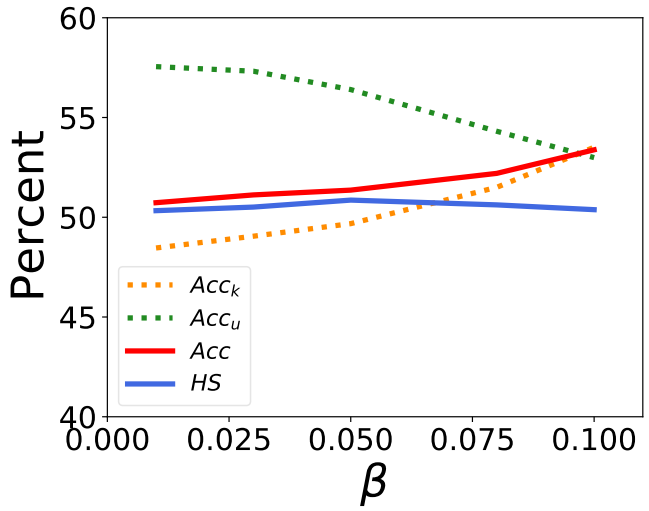

Figure 2.4: The impact of the value β, i.e., the hyper-parameter in Eq. (2.11), on domain adaptation task D→A.

ponents, i.e., the adversarial domain alignment loss $\mathcal{L}_{adv}$ for target nodes from certain group and neighbor center clustering loss $\mathcal{L}_{ncc}$ for target nodes from uncertain group, suggesting the SDA has the advantage of handling open-set graph domain adaptation problem.

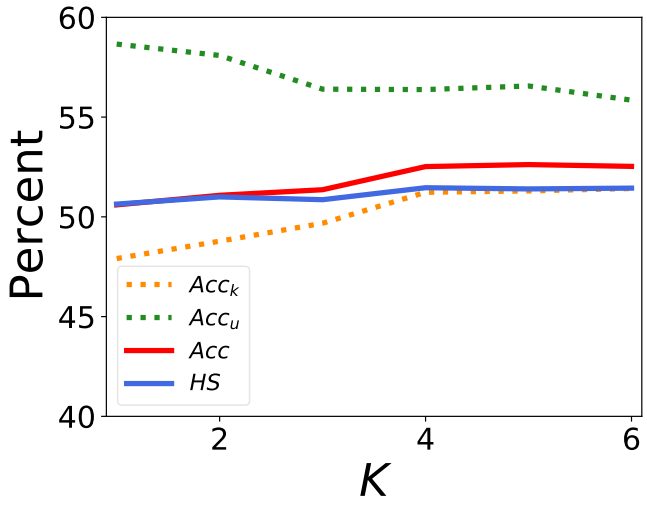

Figure 2.5: The impact of the value K, i.e., the number of clusters in target uncertain group, on domain adaptation task D→A.

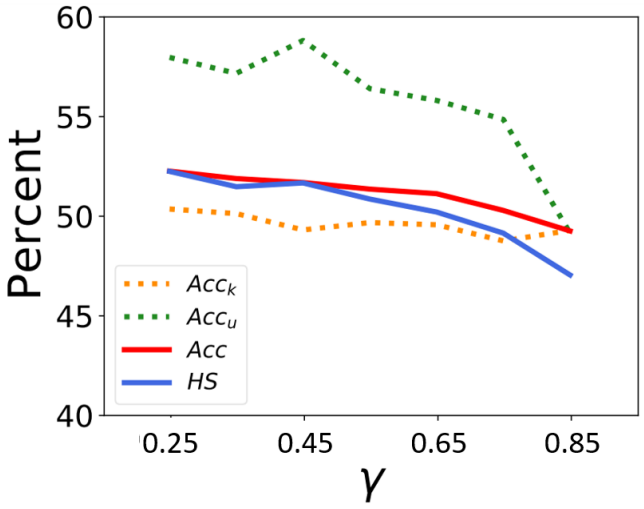

Figure 2.6: The impact of the value γ, i.e., the hyper-parameter in Eq. 2.5), on domain adaptation task D→A

2.4.4 Hyper-parameter sensitivity study

We evaluate the sensitivity of hyper-parameters β in Eq. (2.11), number of clusters K in target uncertain group, and γ in Eq. 2.5. We show the performance of our method on domain adaptation task D→A. β is selected from $\{0.01, 0.03, 0.05, 0.08, 0.10\}$. K is picked from $\{1, 2, 3, 4, 5, 6\}$. We varied γ, which is set as $log(3)/2 = 0.549$, by adding δ to γ. To fully investigate the influence of γ, we use $\delta \in \{-0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3\}$. The values we pick for γ are $\{0.249, 0.349, 0.449, 0.549, 0.649, 0.749, 0.849\}$.

As shown in Figure 2.4, we observe that HS is relatively stable in the range $[0.01, 0.05]$ while slightly degraded in the range $[0.05, 0.10]$. The Acc and Acc_k are continually increased by increasing the value of β while Acc_u is gradually decreased.

Figure 2.5 demonstrates the performance of our method under different number of clusters in target uncertain group, i.e., K. We also observe that the HS is slightly increased when increasing the value of K, Acc and Acc_k are increased, and Acc_u is gradually decreased.

The larger γ is, the more target unknown classes nodes would be divided into the certain group $Group_c$. The smaller γ is, the more target known classes nodes would be divided into the uncertain group $Group_u$. From Figure 2.6, we observe that the performance of our method is continuously decreasing in terms of all evaluation metrics when γ is increasing. This phenomenon is caused by dividing plenty of target unknown classes nodes into $Group_c$. As the adversarial learning will be applied to $Group_c$ and source domain to alleviate the domain divergence, the target unknown classes nodes have a side effect on the domain alignment which has been observed in existing closed-set graph domain adapta-

tion methods (M. Wu et al., 2020; X. Zhang et al., 2021) and hence deteriorate the model's performance on our open-set graph domain adaptation problem.

Overall, our method is less sensitive to the two hyper-parameters, β and K, from the aspects of Acc and HS. Our method is sensitive to the hyper-parameter γ; a higher γ would result in some performance drop. The reason has been explained in the above paragraph.

2.5 Conclusion

In this work, we propose to address a new and challenging problem, namely open-set graph domain adaptation (OS-GDA), where target graph is allowed to contain nodes that are out of source label space. A separate domain alignment (SDA) scheme is newly introduced to effectively resolve the open-set cross-network node classification problem. We jointly consider two different domain alignment strategies for different target nodes to sufficiently learn the well-aligned discriminative embedding space, which further improves the capability of the model on OS-GDA. Extensive experiments show that our method achieves significant performance gain over the state-of-the-art methods.

Chapter 3

Semi-Supervised Node Classification with OOD Detection

3.1 Background

Although graph machine learning has been extensively studied in various field including both academia and industry, most of the research problems hold the in-distribution (ID) assumption, i.e., training data and testing data are from the same distribution. But in some practical cases, it is very likely that testing data are out-of-distribution (OOD). Recently, OOD detection or OOD generalization, have drawn increasing attention to the machine learning research community. However, current works mainly solve the problems on Euclidean data (e.g., images). For graph data, the OOD problem remains under-explored.

There are essentially two challenges for OOD research (in our case, the node classification problem) on graph data: 1) the unique characteristic of graph

structure information (relationships between entities) needs to be considered when designing methods, compared to other type of data that only have their own features for each entity; 2) there are interactions between ID and OOD data in a graph, meaning that many GNN models would embed neighbor nodes to be more similar, making the OOD detection problem harder.

Not only are there fewer OOD methods proposed for graph data, but also that very limited benchmark data for OOD detection/generalization are available nowadays. For example, Gui et al., 2022 attempt to provide some comprehensive graph benchmarks for researchers. Before that, very little efforts have been put into for extensively benchmarking graph data. We are motivated to utilize the latest benchmark data and evaluate its quality.

In this chapter, we design a method to better solve the semi-supervised node classification problem, under the assumption that in the testing data, there are both ID and OOD nodes.

Our contributions are summarized as follows:

- We propose a Mixup By Domain (MixupD) framework for the semi-supervised node classification for graphs with their test data containing OOD nodes, which outperforms other baselines in most cases.
- We adapt the Deep Support Vector Data Description (Deep SVDD) method on graph OOD problem and demonstrate promising results on OOD detection.
- We utilize some of the most recent newly-developed graph OOD benchmark data to evaluate the data quality and performances on baseline methods and our method.

3.2 Related work

3.2.1 OOD generalization on graphs

Currently, there are mainly three classes of existing methodologies regarding OOD generalization on graphs (H. Li et al., 2022). They are described below with examples.

- Structure-wise graph data augmentation. Graph topology is one crucial information for graph learning, therefore it is natural to perform data augmentation utilizing the graph structure. The data augmentation can happen on the node level or edge level. One example is to remove or add edges into the graph.
- Feature-wise graph data augmentation. One example would be to manipulate the node features by giving them perturbation during training, which has shown its effectiveness on OOD generalization.
- Mixed-type graph data augmentation. This is currently the most popular direction, since it can combine the advantages of both structure-wise and feature-wise graph augmentation methods and have shown state-of-the-art performance.

3.2.2 The idea of Mixup

Mixup, along with its variants (Verma et al., 2019; H. Zhang et al., 2017), has been both theoretically and empirically shown to perform well on OOD generalization in the fields of computer vision (L. Zhang et al., 2020) and natural language processing (Guo, 2020). Mixup augment data by generating new instances based on the interpolation of the given instances. Figures 3.1 and 3.2

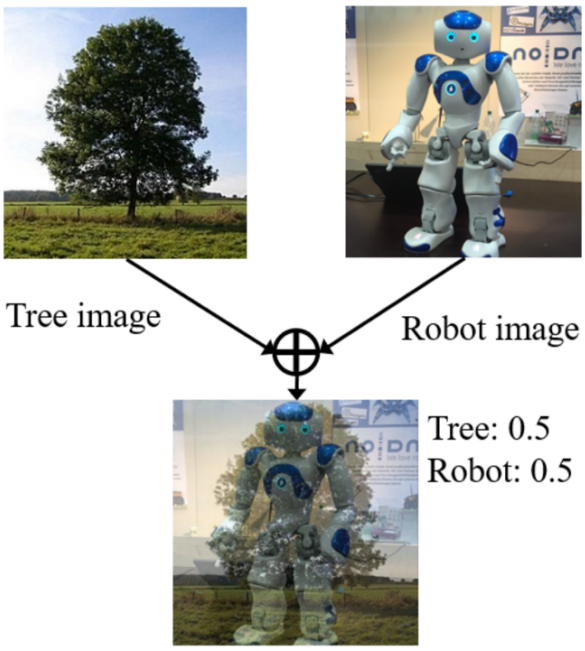

Figure 3.1: For image classification, the existing Mixup generates synthetic images by interpolating both image pixels and labels (Source: Y. Wang et al., 2021).

demonstrate the difference between the Mixup for image data and network data. In the figures, two instances perform Mixup to generate a new instance. It is quite intuitive for interpolating two images, since it is simply a linear combination of the pixel values for image data (Figure 3.1). For graph data, however, it becomes complicated due to the fact that nodes are connected. Moreover, GNN models usually update node features by their neighbors, meaning that "mixing" two nodes actually means a mix for multiple nodes, which should be carefully designed (Figure 3.2). Y. Wang et al., 2021 design a Mixup framework specifically for graph data, aiming to solve both the node classification problem and the graph classification problem.

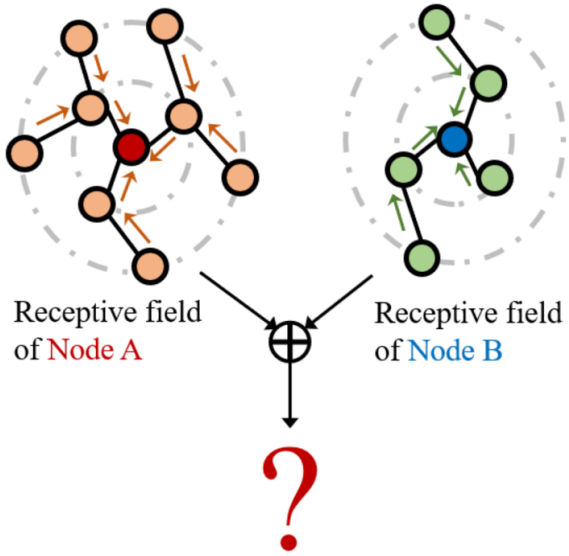

Figure 3.2: For node classification, to mix a pair of nodes A (red) and B (blue), we need to mix their receptive field subgraphs (Source: Y. Wang et al., 2021).

3.2.3 OOD detection

OOD detection, or anomaly detection (AD), is the task of identifying unusual samples in the data. One of the classic methods for OOD detection is Support Vector Data Description (SVDD), which utilizes a hypersphere to separate the data in feature space (Tax and Duin, 2004). Deep SVDD is highly inspired by SVDD, which then becomes a popular OOD detection technique that uses a deep neural network to learn a compact representation of normal data and detect anomalies based on dissimilarity scores (Ruff et al., 2018). Deep SVDD is an unsupervised method that minimizes a loss function combining reconstruction error and regularization to achieve a tight boundary around normal data points. Deep SVDD has been used in various applications; L. Zhao and

Akoglu, 2021 have shown that the incorporation of Deep SVDD demonstrate promising performances on graph-level OOD detection tasks.

3.3 Method

3.3.1 Preliminaries

Given a pair of samples (x_i, y_i) and (x_j, y_j), where x denotes the input feature and y denotes the one-hot class label. Mixup produces a synthetic sample as below.

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j, \tag{3.1}$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j, \tag{3.2}$$

where λ is a hyper-parameter controlling the weight of Mixup and $\lambda \in [0, 1]$. Mixup uses Equations 3.1 and 3.1 to extend the training data distribution by calculating an interpolation of features, which should lead to interpolations of the associated labels. During training, Mixup randomly picks one sample and then pairs it up with another sample drawn from the same mini-batch.

In our problem, let a graph be $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges. For a node i, its input attribute vector is $\mathbf{x}_i$, and its neighborhood node set $\mathcal{N}(i) = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$. We will use GNN as the backbone model for our method as it is the state-of-the-art solution for node classification. Briefly, GNNs train and compute an updated representation for each node $\mathbf{h}_i^{(l)}$ at layer l through a message passing mechanism:

$$\mathbf{h}_i^{(l)} = \text{AGGREGATE}\left(\mathbf{h}_i^{(l-1)}, \left\{\mathbf{h}_j^{(l-1)} \mid j \in \mathcal{N}(i)\right\}, \mathbf{W}^{(l)}\right) \tag{3.3}$$

where $\mathbf{W}^{(l)}$ is the trainable weights for layer l, and AGGREGATE is an aggregation function, which varies in different GNN models. $\mathbf{h}_i^{(0)} = \mathbf{x}_i$ is the input of the model. To tackle the node classification problem, GNNs minimize the classification loss (e.g., cross-entropy) by updating the nodes' representations through the GNN layers. Let L be the total number of layers in a GNN model.

3.3.2 Mixup By Domain (MixupD)

Gui et al., 2022 develop an OOD benchmark, known as GOOD, specifically designed for graphs and graph-related tasks. GOOD explicitly makes distinctions between covariate and concept shifts and has data splits that accurately reflect different shifts. Overall, GOOD contains 11 datasets with 17 domain selections, among which 4 datasets are designed for node-level multi-class classification task with 6 domain selections. For a specific domain selection, the nodes in the graph can be inherently different (i.e., OOD) since each node i has its own domain label. Largely motivated by GOOD, we try to utilize domain information for more efficient Mixup and hopefully better OOD generalization. The key part is that when performing Mixup with node i, we select the other node j by selecting a node with different domain labels. The intuition behind this is that for better OOD generalization, it can be more helpful to interpolate across domains instead of within domains. If we mix two nodes randomly, it is still

very likely to mix nodes from the same domain. Algorithm 3 shows the details of our method.

Algorithm 3 Mixup By Domain (MixupD) for Node Classification

Input: Graph $G = (\mathcal{V}, \mathcal{E})$ of a mini-batch, with node attributes $\{\mathbf{x}_i \mid i \in \mathcal{V}\}$, a GNN model with the aggregation function AGGREGATE$(\cdot)$, hyper-parameter α for the distribution of λ, the domain labels $\{\mathbf{d}_i \mid i \in \mathcal{V}\}$, the ground truth labels $\{\mathbf{y}_i \mid i \in \mathcal{V}\}$.
Output: The trained parameters of GNN: $\left\{\mathbf{W}^{(l)}\right\}_l$.

for $i \leftarrow 1$ to $\#\mathcal{V}$ **do**
  $\mathbf{h}_i^{(0)} \leftarrow \mathbf{x}_i$
end for
for $l \leftarrow 1$ to $L-1$ **do**
  for $i \leftarrow 1$ to $\#\mathcal{V}$ **do**
    $\mathbf{h}_i^{(l)} \leftarrow \text{AGGREGATE}\left(\mathbf{h}_i^{(l-1)}, \left\{\mathbf{h}_j^{(l-1)} \mid j \in \mathcal{N}(i)\right\}, \mathbf{W}^{(l)}\right)$
  end for
end for
for $i \leftarrow 1$ to $\#\mathcal{V}$ **do**
  Sample j from $\mathcal{V}$ where $\mathbf{d}_j \neq \mathbf{d}_i$
  $\lambda \leftarrow \text{Beta}(\alpha, \alpha)$
  $\tilde{\mathbf{x}}_{ij} \leftarrow \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j$
  $\tilde{\mathbf{y}}_{ij} \leftarrow \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j$
  $\tilde{\mathbf{h}}_{ij}^{(0)} \leftarrow \tilde{\mathbf{x}}_{ij}$
  for $l \leftarrow 1$ to L **do**
    $\tilde{\mathbf{h}}_{ij,i}^{(l)} \leftarrow \text{AGGREGATE}\left(\tilde{\mathbf{h}}_{ij}^{(l-1)}, \left\{\mathbf{h}_k^{(l-1)} \mid k \in \mathcal{N}(i)\right\}, \mathbf{W}^{(l)}\right)$
    $\tilde{\mathbf{h}}_{ij,j}^{(l)} \leftarrow \text{AGGREGATE}\left(\tilde{\mathbf{h}}_{ij}^{(l-1)}, \left\{\mathbf{h}_k^{(l-1)} \mid k \in \mathcal{N}(j)\right\}, \mathbf{W}^{(l)}\right)$
    $\tilde{\mathbf{h}}_{ij}^{(l)} \leftarrow \lambda \tilde{\mathbf{h}}_{ij,i}^{(l)} + (1-\lambda)\tilde{\mathbf{h}}_{ij,j}^{(l)}$
  end for
end for
Calculate classification loss $\mathcal{L}$ on $\left\{\tilde{\mathbf{h}}_{ij}^{(L)}, \tilde{\mathbf{y}}_{ij} \mid i \in \mathcal{V}\right\}$.
Back-propagation on $\left\{\mathbf{W}^{(l)}\right\}_l$ for minimizing $\mathcal{L}$.

3.3.3 MixupD+DeepSVDD

To add in the OOD detection model in our framework, we incorporate Deep SVDD on top of Algorithm 3. The main difference is that we add a few more parameters and a corresponding loss $\mathcal{L}_{DeepSVDD}$:

$$R^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \max\left\{0, \left\|\phi\left(\boldsymbol{x}_i; \mathcal{W}\right) - \boldsymbol{c}\right\|^2 - R^2\right\} + \frac{\lambda}{2} \sum_{\ell=1}^{L} \left\|\boldsymbol{W}^{\ell}\right\|_F^2, \quad (3.4)$$

where Equation 3.4 is $\mathcal{L}_{DeepSVDD}$. $\phi(\cdot; \mathcal{W}) : \mathcal{X} \to \mathcal{F}$ is a neural network with $L \in N$ hidden layers and set of weights $\mathcal{W} = \left\{\boldsymbol{W}^1, \ldots, \boldsymbol{W}^L\right\}$ for some input space $\mathcal{X} \subseteq R^d$ and output space $\mathcal{F} \subseteq R^p$ for nodes in a graph. The goal of minimizing $\mathcal{L}_{DeepSVDD}$ is to jointly learn the network parameters $\mathcal{W}$ together with minimizing the volume of a data-enclosing hypersphere in output space $\mathcal{F}$ that is characterized by radius $R > 0$ and center $c \in \mathcal{F}$. Hyperparameter $\nu \in (0, 1]$ controls the trade-off between the volume of the sphere and violations of the boundary, i.e., allowing some points to be mapped outside the sphere.

3.4 Experiments

In this section, we evaluate the effectiveness of our method as the following: Firstly, we introduce the experimental settings. Second, we compare our method to other baseline methods and analyze the results.

3.4.1 Experimental settings

Datasets

We used 4 datasets from GOOD and 6 domain selections from the four datasets. They are described as below. **GOOD-Cora** is a citation network adapted from the full Cora dataset. The dataset consists of a citation network graph that includes nodes representing individual scientific publications and edges repre-

senting citation links. The objective is to classify publications into one of 70 categories. Two domain selections were used for splitting the dataset: word and degree. The former is the word diversity determined by the quantity of selected words used in a publication, independent of its label. The latter is based on the node degree in the graph, which ensures that a paper's popularity does not influence its classification.

GOOD-Arxiv is a citation dataset adapted from OGB[3]. The input data consists of a directed graph that depicts the citation network among ArXiv papers in the field of computer science (CS). In the graph, nodes represents individual ArXiv papers, while directed edges indicate citations. The objective is to predict the subject area of ArXiv CS papers, which involves classifying them into one of 40 categories. The dataset is partitioned based on two domain selections: time (publication year) and node degree.

[3] https://ogb.stanford.edu/.

GOOD-WebKB is a university webpage network dataset. The network comprises nodes that correspond to webpages, with node features representing the words found on each page, while edges indicate hyperlinks between webpages. The objective is to classify webpages into one of five categories. The dataset is partitioned based on the domain of the university, which means that webpage classification is based on word content and link connections rather than university-specific features.

GOOD-CBAS is a synthetic dataset modified from BA-Shapes. The input data comprises a graph that was generated by attaching 80 house-like motifs to a 300-node Barabási–Albert base graph. The objective is to classify the role of nodes, which could be the top, middle, or bottom node of a house-like motif or a node from the base graph. This forms a 4-class classification task. Unlike using constant node features, the dataset features colored node attributes. As

a result, OOD algorithms must handle color differences in covariate splits and color-label correlations in concept splits.

Baselines

We adopt two baselines, namely empirical risk minimization (ERM) and Mixup, for comparison with our method. We choose these two methods as baselines because 1) ERM is a simple GCN model, which works as a fundamental baseline for all other graph OOD methods; 2) Mixup is shown to be the top performer among all baseline methods in the experiments by Gui et al., 2022, and we aim to compare with the best existing method.

Evaluation metrics

There are two tasks in our problem setting: node classification and OOD detection.

Given that the first one is a multi-class classification task, we consider two metrics for the first task: average class accuracy over all classes for (1) OOD test nodes only; (2) ID and OOD test nodes. These two metrics represent both the ability to classify nodes into the correct classes and the ability to generalize on OOD data.

For the second task, we choose AUC as the metric to evaluate the ability to detect whether a node is ID or OOD, which is a standard metric for OOD detection.

Implementation details

Our implementation is based on Pytorch (Paszke et al., 2019)[4]. Training environment is set the same as GOOD (Gui et al., 2022)[5]. We use GCN as GNN

[4] https://pytorch.org/.

[5] https://github.com/divelab/GOOD/.

backbones for all the experiments. Each task has been run ten times to obtain 10 replicates, and we report the average of the 10 runs in the final results.

Note that since the baseline methods do not solve the OOD detection problem, for comparison, we build a simple logistic regression model to classify the nodes as ID or OOD, using their features updated by their models. We do a 10%/90% split for the OOD detection training. This is not ideal as we will need some nodes' ID/OOD label to train the OOD detection model; however, the baseline methods do not consider this challenging problem, and this is one way to make a relatively fair comparison.

For training MixupD+DeepSVDD, we have two losses during model training: $\mathcal{L}$ and $\mathcal{L}_{DeepSVDD}$. We optimize these two losses in turns to avoid the negative effect between these two optimization tasks.

3.4.2 Results and analysis

Table 3.1 shows the results of our experiment. It can be seen that our method performs best on GOOD-Cora and GOOD-WebKB. Surprisingly, ERM outperforms the rest for GOOD-Arxiv and GOOD-CBAS, by a relatively large margin. The results show that our method MixupD outperforms the original Mixup method, which prove its effectiveness on improving OOD generalization. The accuracies for the combined test for both ID and OOD nodes is overall better than the accuracies for the OOD test nodes only, which is within expectation since it is easier to predict ID test nodes given the training data are ID.

Table 3.2 shows the effectiveness of incorporating Deep SVDD for dataset GOOD-WebKB. Interestingly, the MixupD+DeepSVDD method not only outperforms other baselines regarding the OOD detection task, but also on the

Table 3.1: Experiment results (Accuracy in %) on node classification for data(domain selection).

	Cora (degree)		Cora (word)	
	OOD test	Combined test (ID&OOD)	OOD test	Combined test (ID&OOD)
ERM	60.57	62.32	64.60	64.84
Mixup	63.36	64.82	64.44	65.33
Ours	**63.70**	**65.09**	**65.43**	**66.44**
	Arxiv (degree)		Arxiv (time)	
	OOD test	Combined test (ID&OOD)	OOD test	Combined test (ID&OOD)
ERM	**62.86**	**65.16**	**67.43**	**68.77**
Mixup	61.28	62.25	64.84	66.38
Ours	59.71	61.79	62.77	65.15
	CBAS (color)		WebKB (university)	
	OOD test	Combined test (ID&OOD)	OOD test	Combined test (ID&OOD)
ERM	**82.86**	**86.57**	26.70	38.64
Mixup	64.57	79.64	30.83	44.97
Ours	66.07	80.54	**32.66**	**46.69**

classification tasks for both ID and OOD nodes. Specifically, MixupD+DeepSVDD performs significantly better than MixupD in terms of the classification accuracy on OOD test nodes. Furthermore, the variance of OOD detection AUC for baseline methods are significantly larger than our MixupD+DeepSVDD method. For example, the AUC of 10 runs for MixupD has a standard deviation of 3.23%, while for MixupD+DeepSVDD is 1.51%, meaning that MixupD+DeepSVDD has a consistently better performance on OOD detection than others.

Table 3.2: Experiment results (Accuracy and AUC in %) on node classification for data(domain selection).

	WebKB (university)		
	OOD test	Combined test (ID&OOD)	OOD detection
ERM	26.70	38.64	50.57
Mixup	30.83	44.97	48.81
Ours	32.66	46.69	59.58
Ours+DeepSVDD	**37.89**	**47.63**	**63.42**

3.5 Future work

In this chapter, we aim to solve the semi-supervised node classification problems where nodes can be ID or OOD. We also attempt to simultaneously conduct OOD detection in the method. We specifically study the concept shift case using benchmark datasets, GOOD, developed by Gui et al., 2022. We develop Mixup By Domain (MixupD) as well as MixupD+DeepSVDD to solve both OOD generalization and OOD detection problem and study their effectiveness. Experiments show overall superior performance of our methods compared with other baseline methods. Extensive runs of the experiments on the benchmark data provided by GOOD indicate that currently there is still not a single optimal solution for the OOD problem across all datasets. However, our method outperforms baselines in most cases and is the first to solve the OOD detection problem at the same time, which shows promising performance than any other existing baselines.

There are several directions where this research can be continuously refined: (1) we can extend the problem to both concept shift and covariate shift OOD data; (2) we can fine tune the Mixup by Domain; (3) more examination can

be done in terms of why there are performance discrepancies across different GOOD benchmark datasets (e.g., GOOD-Cora and GOOD-Arxiv).

Chapter 4

Conclusion

In this dissertation, we study a few graph machine learning cases where target graph data are OOD compared to the ID training data. Among various OOD problems, we study two challenging and realistic topics, i.e., domain adaptation and OOD detection, and propose solutions to those novel problems. The methods proposed in this dissertation have shown state-of-the-art performance and can be applied to solve the real-world problems. Hopefully, works presented here can help to reduce labeling effort on large data and to detect abnormal data in an efficient and effective way. We also point out some promising future directions in this research field and look forward to seeing this field keep growing.

Bibliography

Atwood, J., & Towsley, D. (2016). Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 1993–2001.

Bhagat, S., Cormode, G., & Muthukrishnan, S. (2011). Node classification in social networks. In C. C. Aggarwal (Ed.), *Social network data analytics* (pp. 115–148). Springer US. https://doi.org/10.1007/978-1-4419-8462-3_5

Bhatia, K., Dahiya, K., Jain, H., Kar, P., Mittal, A., Prabhu, Y., & Varma, M. (2016). The extreme classification repository: Multi-label datasets and code. http://manikvarma.org/downloads/XC/XMLRepository.html

Bojchevski, A., Shchur, O., Zügner, D., & Günnemann, S. (2018). Netgan: Generating graphs via random walks. *International Conference on Machine Learning*, 610–619.

Bucci, S., Loghmani, M. R., & Tommasi, T. (2020). On the effectiveness of image rotation for open set domain adaptation. *European Conference on Computer Vision*, 422–438.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., & Hinton, G. E. (2020). Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, *33*, 22243–22255.

Dai, Q., Wu, X.-M., Xiao, J., Shen, X., & Wang, D. (2022). Graph transfer learning via adversarial domain adaptation with graph convolution. *IEEE Transactions on Knowledge and Data Engineering*.

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf

Dong, J., Cong, Y., Sun, G., Liu, Y., & Xu, X. (2020). Cscl: Critical semantic-consistent learning for unsupervised domain adaptation. *European Conference on Computer Vision*, 745–762.

Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., & Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292*.

Fu, B., Cao, Z., Long, M., & Wang, J. (2020). Learning to detect open classes for universal domain adaptation. *ECCV*, 567–583.

Ganin, Y., & Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. *International conference on machine learning*, 1180–1189.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., & Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, *17*(1), 2096–2030.

Geng, C., Huang, S.-j., & Chen, S. (2020). Recent advances in open set recognition: A survey. *IEEE transactions on pattern analysis and machine intelligence*, *43*(10), 3614–3631.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. *International Conference on Machine Learning*, 1263–1272.

Grandvalet, Y., & Bengio, Y. (2004). Semi-supervised learning by entropy minimization. *NeurIPS*, 529–536.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., & Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, *13*(1), 723–773.

Gui, S., Li, X., Wang, L., & Ji, S. (2022). GOOD: A graph out-of-distribution benchmark. *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. https://openreview.net/forum?id=8hHg-zs_p-h

Guo, H. (2020). Nonlinear mixup: Out-of-manifold data augmentation for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(04), 4044–4051.

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*.

Hammond, D. K., Vandergheynst, P., & Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, *30*(2), 129–150.

Hendrycks, D., & Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. https://doi.org/10.48550/ARXIV.1610.02136

Hendrycks, D., Mazeika, M., Kadavath, S., & Song, D. (2019). Using self-supervised learning can improve model robustness and uncertainty. *Advances in neural information processing systems*, *32*.

Ji, P., & Jin, J. (2016). Coauthorship and citation networks for statisticians. *The Annals of Applied Statistics*, *10*(4), 1779–1812. https://doi.org/10.1214/15-AOAS896

Jiang, J., & Zhai, C. (2007). Instance weighting for domain adaptation in nlp.

Kampffmeyer, M., Chen, Y., Liang, X., Wang, H., Zhang, Y., & Xing, E. P. (2019). Rethinking knowledge graph propagation for zero-shot learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11487–11496.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*. https://openreview.net/forum?id=SJU4ayYgl

Kumar, A., Singh, S. S., Singh, K., & Biswas, B. (2020). Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, *553*, 124289.

Kundu, J. N., Venkat, N., Revanur, A., V, R. M., & Babu, R. V. (2020). Towards inheritable models for open-set domain adaptation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, H., Wang, X., Zhang, Z., & Zhu, W. (2022). Out-of-distribution generalization on graphs: A survey. *arXiv preprint arXiv:2202.07987*.

Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2015). Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Liang, S., Li, Y., & Srikant, R. (2018). Enhancing the reliability of out-of-distribution image detection in neural networks. *International Conference on Learning Representations*. https://openreview.net/forum?id=H1VGkIxRZ

Liu, H., Cao, Z., Long, M., Wang, J., & Yang, Q. (2019). Separate to adapt: Open set domain adaptation via progressive separation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2927–2936.

Long, M., Cao, Y., Wang, J., & Jordan, M. (2015). Learning transferable features with deep adaptation networks. *International conference on machine learning*, 97–105.

Long, M., Cao, Z., Wang, J., & Jordan, M. I. (2018a). Conditional adversarial domain adaptation. *Advances in neural information processing systems*, *31*.

Long, M., Cao, Z., Wang, J., & Jordan, M. I. (2018b). Conditional adversarial domain adaptation. *Advances in neural information processing systems*, *31*.

Moreno-Torres, J. G., Raeder, T., Alaiz-Rodrıguez, R., Chawla, N. V., & Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern recognition*, *45*(1), 521–530.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Panareda Busto, P., & Gall, J. (2017). Open set domain adaptation. *Proceedings of the IEEE international conference on computer vision*, 754–763.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Neural Information Processing Systems*, *32*.

Quinonero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (2008). *Dataset shift in machine learning*. Mit Press.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., & Kloft, M. (2018). Deep one-class classification. *International conference on machine learning*, 4393–4402.

Saito, K., Kim, D., Sclaroff, S., & Saenko, K. (2020). Universal domain adaptation through self supervision. *Advances in neural information processing systems*, *33*, 16282–16292.

Saito, K., Yamamoto, S., Ushiku, Y., & Harada, T. (2018). Open set domain adaptation by backpropagation. *Proceedings of the European Conference on Computer Vision (ECCV)*, 153–168.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80. https://doi.org/10.1109/TNN.2008.2005605

Shen, X., Dai, Q., Mao, S., Chung, F.-L., & Choi, K.-S. (2021). Network together: Node classification via cross-network deep network embedding. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(5), 1935–1948. https://doi.org/10.1109/TNNLS.2020.2995483

Sun, B., Feng, J., & Saenko, K. (2016). Return of frustratingly easy domain adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *30*(1).

Szklarczyk, D., Gable, A. L., Lyon, D., Junge, A., Wyder, S., Huerta-Cepas, J., Simonovic, M., Doncheva, N. T., Morris, J. H., Bork, P., Jensen, L. J., & Mering, C. v. (2018). STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Research*, *47*(D1), D607–D613. https://doi.org/10.1093/nar/gky1131

Tack, J., Mo, S., Jeong, J., & Shin, J. (2020). Csi: Novelty detection via contrastive learning on distributionally shifted instances. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (pp. 11839–11852). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/8965f76632d7672e7d3cf29c87ecaa0c-Paper.pdf

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., & Su, Z. (2008). Arnetminer: Extraction and mining of academic social networks. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 990–998. https://doi.org/10.1145/1401890.1402008

Tax, D. M., & Duin, R. P. (2004). Support vector data description. *Machine learning*, *54*, 45–66.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., & Bengio, Y. (2019). Manifold mixup: Better representations by interpolating hidden states. *International conference on machine learning*, 6438–6447.

Wang, Y., Wang, W., Liang, Y., Cai, Y., & Hooi, B. (2021). Mixup for node and graph classification. *Proceedings of the Web Conference 2021*, 3663–3674.

Wang, Z., Liang, Y., & Ji, P. (2020). Spectral algorithms for community detection in directed networks. *arXiv preprint arXiv:2008.03820*.

Wu, M., Pan, S., Zhou, C., Chang, X., & Zhu, X. (2020). Unsupervised domain adaptive graph convolutional networks. *Proceedings of The Web Conference 2020*, 1457–1467. https://doi.org/10.1145/3366423.3380219

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

Xu, M., Wang, H., Ni, B., Guo, H., & Tang, J. (2021). Self-supervised graph-level representation learning with local and global structure. *International Conference on Machine Learning*, 11548–11558.

Zhai, X., Oliver, A., Kolesnikov, A., & Beyer, L. (2019). S4l: Self-supervised semi-supervised learning. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1476–1485.

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhang, L., Deng, Z., Kawaguchi, K., Ghorbani, A., & Zou, J. (2020). How does mixup help with robustness and generalization? *arXiv preprint arXiv:2010.04819*.

Zhang, X., Du, Y., Xie, R., & Wang, C. (2021). Adversarial separation network for cross-network node classification. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2618–2626.

Zhang, Y., Deng, W., Wang, M., Hu, J., Li, X., Zhao, D., & Wen, D. (2020). Global-local gcn: Large-scale label noise cleansing for face recognition.

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7731–7740.

Zhao, L., & Akoglu, L. (2021). On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights. *Big Data*.

Zhao, R., Xia, Y., & Zhang, Y. (2021). Unsupervised sleep staging system based on domain adaptation. *Biomedical Signal Processing and Control*, *69*, 102937. https://doi.org/https://doi.org/10.1016/j.bspc.2021.102937

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2018). Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*.

Zhuang, C., & Ma, Q. (2018). Dual graph convolutional networks for graph-based semi-supervised classification. *Proceedings of the 2018 World Wide Web Conference*, 499–508.