Shift-Robust Node Classification via Graph Clustering Co-training

Qi Zhu[1], **Chao Zhang**[2], **Chanyoung Park**[3], **Carl Yang**[4], **Jiawei Han**[1]
[1]University of Illinois Urbana-Champaign,
[2]Georgia Institute of Technology, [3]KAIST, [4]Emory University
[1]{qiz3,hanj}@illinois.edu, [2]chaozhang@gatech.edu,
[3]cy.park@kaist.ac.kr, [4]j.carlyang@emory.edu

Abstract

It is widely known that machine learning models only achieve sub-optimal performance when testing data exhibit distribution shift against training *i.e.*, $\Pr_{\text{train}}(X, Y) \neq \Pr_{\text{test}}(X, Y)$. Although Graph Neural Networks (GNNs) have become de facto models for semi-supervised learning tasks, they suffer even more from distribution shift because multiple types of shifts origin from not only node features but graph structures. Existing domain adaptation methods only work for specific type of shifts. In response, we propose **S**hift-**R**obust **N**ode **C**lassification (SRNC) - a unified domain adaptation framework for different kinds of distribution shifts on graph. Specifically, we co-train an unsupervised cluster GNN, which captures the data distribution by *graph homophily* on target graph. Then a shift-robust classifier is optimized on training graph and pseudo samples from target graph, which are provided by cluster GNN. Compared to the existing domain adaptation algorithms on graph, our approach works for both open-set and close-set shifts with convergence guarantees. In our experiments, the classification accuracy is improved at least 3% against the second-best baseline under open-set shifts. On time-evolving graph with close-set shift, existing domain adaption algorithms can barely improve the generalization if not worse. SRNC is still able to mitigate the negative effect ($> 2\%$ absolute improvements) of the shift across different testing-times.

1 Introduction

Graph Neural Networks (GNNs) [11, 27, 7] have achieved enormous success for node classification, these models experience performance drops if training and testing data are not identically and independently distributed (*i.i.d.*) - distribution shift [13]. In real-world GNN applications, the robustness towards distribution shift has become a pressing topic [39, 18, 32] when distribution shift between structures and features can both deteriorate the performance. In this work, we consider two different kinds of shifts according to whether label space Y is changed or not - (1) open-set shift: there are emerging new classes in the graph (*e.g.*, new COVID variant in community transmission); (2) close-set shift: time-augmented test graph has substantially different class distributions (*e.g.*, time-evolving Twitter and Academia graphs).

Despite the popularity and importance of GNNs, only a few domain adaption algorithms are proposed to make GNNs robust to distribution shifts. Most of them are designed to account for a specific type of distribution shift. For example, open-set classification [2, 33, 17] can recognize objects of unseen classes during testing. SRGNN [39] and UDA-GCN [32] are two representative methods adopted from invariant representation learning [15, 16, 35] for close-set shifts. Nevertheless both methods works for only one kind of close-set shift - covariate shift where class-conditional distributions of

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

hidden representations are same during training and testing. Robust graph neural networks against distribution shifts are in great need, but there is still no *go-to* method regarding different kinds of shifts.

Being aware of the aforementioned challenge, we are wondering whether it is possible to design a domain adaption framework with the least amount of assumption of distribution shifts. To the light of this issue, graph structure enjoys numerous nice topological properties, while none of the existing work incorporate these properties to mitigate the limitation of prior knowledge on distribution shifts. As a consequence, these algorithms present even worse result than base model without domain adaptation (see results of domain-invariant baselines in Table 1 for more details).

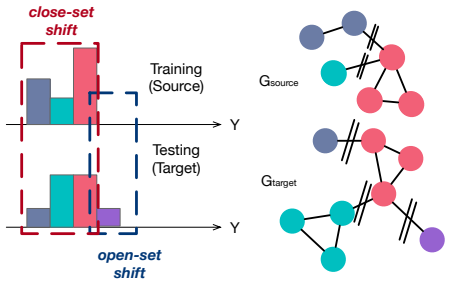


Figure 1: A unified domain adaption perspective for open-set shift and close-set shift. *graph homophily* is universally hold on source and target indicated by $\|$ cut.

In this paper, we propose a unified domain adaption framework for shift-robust node classification, where unlabeled target (testing) graph is used to facilitate model generalization. First, we consider the joint probability on node and label $\Pr(h_i, y_i)$ on graph, where h is the output embedding from a graph neural network. The modeling of joint probability avoids tackling different assumptions on different marginal distributions $\Pr(y_i|h_i)$ one by one. Figure 1 gives a toy example that both open-set and close-set can be interpreted in one joint form. Second, we observe similar degrees of *graph homophily* [38] (*e.g.*, neighbors with same labels) between homophilic source and target graphs. In other words, most real-world applications (*e.g.* time-evolving social networks) would not expect target graph of weak homophily while source graph with strong homophily. Based on *graph homophily*, we propose to use graph clustering to identify latent classes [3, 26] by breaking less frequent edges (possibly heterophily) between potential clusters (right side of Figure 1). If such graph clustering can capture the joint probability on source graph, it would probably generalize well on target graph with comparable degree of homophily. Therefore, our proposed model, which we call SRNC (Section 4.1), co-trains two graph neural networks (GNNs): an shift-robust classification GNN_Θ for unified domain adaption and a clustering GNN_Φ (Section 4.2) for homophily-based clustering - (a) *The clustering memberships inferred by the clustering GNN on source data* $\mathbf{Q}_\Phi(C^s|X^s, A^s)$ *should be close to training conditional distribution* $\Pr_{train}(Y^s|X^s, A^s)$. For instance, on the ride of Figure 2, the KL-divergence between above two probabilities are minimized to push the clustering result consistent with training data; (b) *In the other direction, the classifier is optimized on both training graph and target graph for domain adaptation.* The targets samples are sampled from clustering GNN to improve the generalization on target graph. In SRNC optimization, the two modules improve each other through variational training. Convergence of the iterative optimization between two modules is theoretically justified in Section 4.3.

Our experiments illustrate that the shift-robust classifier trained in proposed way can detect more than 70% open-set samples on three widely used benchmarks. For close-set shift, we use early snapshot (prior to 2011) of ogb-arxiv [9] network as training graph and testing on more recent graphs (14-16/16-18/18-20). SRNC is the only method that can consistently reduce the negative effect (outperform base model) of shift in our evaluation. Meanwhile we show progressively utilizing classifier's prediction as pseudo labels (our ablation without clustering component) on target graph [33, 17] is much worse than proposed method.

To summarize, we propose the first unified domain adaption GNN framework for multiple types of distribution shift. We implement the framework by co-training a shift-robust classifier with a variational cluster. The latter one captures target data distribution and approximates the source class-conditional distribution simultaneously. Extensive experiments show that our method is significantly better out-of-distribution generalization on two different distribution shifts. Additional case study reveals its potential to discover new classes and improve label space design.

2 RELATED WORK

Out-of-Distribution Generalization. In literature, the training and testing distribution is often referred as in-distribution and out-of-distribution when there is a distribution shift. Many research topics are invented to improve the out-of-distributitaon (OOD) generalization, of which two most well-known ones are domain adaptation and domain generalization. Domain generalization [30] requires to train a robust and generalizable predictors on unseen domain assuming training data (graphs) from multiple domains are available. Domain adaption [1] requires only one source domain but the access of the unlabeled target (testing) data. In terms of node classification, the assumption of domain adaption generally holds because it only requires one training graph and usually whole testing graph is available. In this paper, we focus our discussion on domain adaptation.

Close-set Distribution Shift. There has been a recent surge on discussing GNNs generalization [6, 29, 18, 30]. The generalization and stability of GCNs is found to be related with graph filters [29]. The $\mathcal{H}$-divergence [1] and discrepancy measures [19] are invented to measure the generalization bound between source and target domains. To bridge the gap between source and target, Domain Invariant Representation (DIR) Learning aims to minimize these discrepancy measures on source and (unlabeled) target data by adversarial learning [5] or regularization (*e.g.* CMD [35], MMD [15, 16]). In graph neural networks [30], DIR is frequently used for domain adaptation. For example, DANE [37] adds a domain classifier on top of GNN outputs. UDA-GCN [32] further proposes a dual-gnn architecture and inter-graph attention to obtain unified representations across source and target graphs. For covariate shift, SRGNN [39] adopts two shift-robust techniques - CMD and importance sampling for non-IID training data. SOGA [20] assumes a well-trained GNN model without the access of source graph and enforces maximal mutual information and structural proximity on target. Besides, generative algorithm DGDA [4] introduces variational graph auto-encoders to recover and disentangle domain and semantic latent variables for adaptation. We also use unlabeled target data in our framework, but we propose a clustering component to generate pseudo samples specifically for graph structured data.

Open-set Distribution Shift. Early open set recognition and classification [23, 2] require classifiers to detect *unknown* objects during testing on image domain. These studies [8] focus on addressing the overconfidence issue of deep neural networks for unknown classes, such as OpenMax [2] and DOC [25]. On graph structured data, most of the GNNs work under semi-supervised learning. Hence, over confidence issue is less likely to occur. Instead, the topological structure poses challenges for open-set classification. To this end, OpenWGL [33] employs an uncertainty loss in the form of graph reconstruction loss [12] on unlabeled data. PGL [17] address the conditional shifts by episodic pseudo labels and domain adversarial regularization with graph neural networks. Yet, none of open-set learning on graph has explored modeling more than one unseen classes distribution on target graph as ours.

3 Problem Definition & Preliminary

$G = \{V, A, E, X\}$ is defined as a graph with nodes V, their features ($X \in \mathbb{R}^{|V|\times|F|}$) and edges E between nodes (*e.g.*, adjacency matrix A, $A \in \mathbb{R}^{|V|\times|V|}$). Each GNN layer takes node embeddings H and adjacency matrix A as input, aggregates the neighborhood information and outputs representation H^k. For example, in GCN [11],

$$H^k = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{k-1}W^{(k)}\right), \tag{1}$$

where $H^0 = X$, $\hat{A} = A + I$ and $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$.

Given a source graph G^s and the output of a k-layer graph neural network is h^k. Semi-supervised GNNs perform node classification via a cross-entropy loss over labeled source data $\mathcal{D}^s = \{(h_1, y_1), ...(h_n, y_n)\}$.

$$\mathcal{L} = \frac{1}{|\mathcal{D}^s|}\sum_{i=1}^{n}\sum_{j=1}^{N} y_{ij}\log\hat{y}_{ij}, \hat{y}_{ij} = \text{softmax}(f(h_i^k)). \tag{2}$$

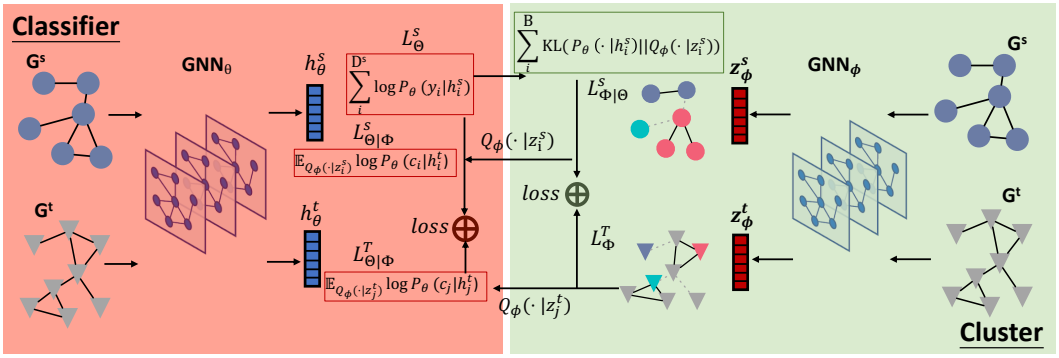


Figure 2: Training framework of SRNC. The left and right block illustrate the training process of classification and clustering GNNs, respectively.

Definition 3.1 (Distribution Shift in Node Classification)**.** *Assume node representations* $H^t = \{h^t_1, ..., h^t_m\}$ *are output from graph neural network on target graph* G^t *with m nodes. The label space on source and target graph are* Y^s *and* Y^t*, respectively. Distribution shift occurs if* $\Pr_{train}(H^s, Y^s) \neq \Pr_{test}(H^t, Y^t)$[1].

In general, there are two basic types of distribution shifts: (1) open-set shift - new classes arise during test-time, *i.e.* $Y^s \subset Y^t$. (2) close-set shift - $Y^s = Y^t$ and joint probability changes between training and testing, $\Pr_{\text{train}}(H^s, Y) \neq \Pr_{\text{test}}(H^t, Y)$.

Definition 3.2 (graph homophily ratio [38])**.** *The graph homophily ratio* $\tau_h = \frac{\{(u,v):(u,v)\in E \wedge y_u = y_v\}}{|E|}$ *is the fraction of edges in a graph which connect nodes with the same class label.*

Recent study [38] has shown *graph homophily* is the main reason of the superior performance of graph neural networks on node classification. Motivated by this phenomenon, we design our shift-robust classifier for domain adaptation on homophilic graphs. When the graph homophily ratio is low, most of the graph neural networks perform not well [38]. Hence, we leave the heterophilic graph extension of our framework as future work.

4 Method

In this section, we present our framework for shift-robust node classification. In Figure 2, SRNC co-trains two GNN modules (1) a semi-supervised classification GNN_Θ on the left side (Section 4.1) and (2) an unsupervised clustering GNN_Φ (Section 4.2) on the right side. The classification GNN is optimized on labeled source graph G^s and pseudo labels on target graph G^t for better generalization against possible distributional shifts. Meanwhile, the clustering GNN is optimized on target graph G^t and regularized with predictions from the classifier on source graph G^s. Finally in Section 4.3 and Algorithm 1, we summarize how we optimize SRNC via iterative optimization between both modules. We also provide convergence analysis of the co-training by Expectation-Maximization (EM). In the remaining of the paper, we call training data as source and testing data as target interchangeably.

4.1 Shift-Robust Node Classification

We first propose the domain adaption loss for semi-supervised node classification towards distribution shifts. Later in the paper, we always denote θ, ϕ as parameter of categorical distribution which are parameterized via Graph Neural Networks. For example, Classifier $\mathbf{P}_\Theta = \mathbf{P}_\theta \circ \text{GNN}_\Theta$ minimize a negative cross-entropy loss on GNN embeddings,

$$\mathbf{P}_\theta(y_i|h_i) = -\log \frac{\exp(h_{i,y_i})}{\sum_{j=1}^{N} \exp(h_{i,j})}, h_i = \text{GNN}_\Theta(x_i, \mathbf{A}), \tag{3}$$

[1]We study the distribution shift in final latent space H and refer $\Pr(H, Y)$ as joint probability in the paper.

where $h_i \in \mathbb{R}^N$ and $h_{i,j}$ is the j-th entry of the vector.

We have source loss $\mathcal{L}^s_\Theta$ on training data (h^s_i, y^s_i), and target loss $\mathcal{L}^t_{\Theta|\Phi}$ on pseudo samples $(h^t_i, \hat{y}^t_i)$ from target (testing) data.

$$\begin{aligned}\mathcal{L}_\Theta = & \underbrace{\sum_{i=1}^{|D^s|} -\log \mathbf{P}_\theta(y^s_i|h^s_i)}_{\mathcal{L}^s_\Theta} \\ & + \underbrace{\mathbb{E}_{i\sim U\{1,|V^t|\},\hat{y_i}^t\sim\mathbf{Q}_\phi(\cdot|z^t_i)}\left[-\log \mathbf{P}_\theta(\hat{y}^t_i|h^t_i)\right]}_{\mathcal{L}^t_{\Theta|\Phi}}.\end{aligned} \tag{4}$$

We denote under-script $\Theta|\Phi$ as the forward pass relies on both GNNs $\{\Theta, \Phi\}$ while backward pass only updates Θ. For example, in Figure 2,loss term $L^s_{\Theta|\Phi}$ receives samples from cluster block $\mathbf{Q}_\Phi$ but the back-propagation only update parameters of GNN_Θ The sampling process in the second term first uniformly sample same amount of nodes as training $\{x^t_i\}_{i=1}^{|\mathcal{D}^s|}$ from target graph. Then we obtain their node embeddings $\{z^t_i\}$ and pseudo labels $\{\hat{y}^t_i\}$ through cluster GNN_Φ and softmax layer $\mathbf{Q}_\phi(y_i|z_i)$. We will discuss how to align the identity of cluster $C = \{1..K\}$ and classes $Y = \{1..N\}$ at the end of clustering co-training (Section 4.2). In left part of Figure 2, $\mathbf{P}_\Theta$ is first trained on source graph and jointly optimized on pseudo target samples. Note that source and target graph can be same (semi-supervised) or different (supervised).

Our framework works for both kinds of distribution shifts that we deem essential in the paper. Specifically, for open-set shift, we set number of cluster larger than known classes, *i.e.* $K > N$. After alignment between clusters and classes, target samples in unmatched clusters are pseudo labeled as new class $\hat{y} = |Y| + 1$. For close-set shift, we simply set number of cluster and classes equal in two modules.

4.2 Graph Clustering Co-training

In this section, we will explain how to approximate the true target data distribution $(h^t_i, {y_i}^t) \sim \mathbb{P}^t$ with $\mathbf{Q}_\phi$ and the connection between *graph homophily* and clustering. The sufficient condition of uniform sampling from target $\mathbb{P}^t$ requires an unbiased estimation of $\mathbb{P}^t(y^t_i|z^t_i)$ if we uniformly sample node i from target graph. Although underlying $\mathbb{P}^t(y^t_i|z^t_i)$ is unknown, similar nodes likely appear in the same cluster based on *graph homophily*. Hence, we decompose the approximation into two steps: (a) homophily-based clustering to group nodes into different clusters (b) co-training mechanism to match the identity of clusters to known label space on source graph.

Modularity [22] measures the strength of division of a graph into clusters by comparing random distributed edges between clusters. Graph homophily ensures both source and target graph yields large modularity if grouped by ground truth labels. Given the cluster membership matrix $\mathcal{C} \in \{0, 1\}^{|V|\times C}$, the modularity measure [22] $\mathcal{S}$ quantifies the divergence between the number of intra-cluster edges and the expected number in a random graph. By maximizing the modularity, the nodes are densely connected within each cluster:

$$\mathcal{S} = \frac{1}{2|E|}\sum_{ij}\left[A_{ij} - \frac{d_i d_j}{2|E|}\right]\delta(c_i, c_j), \tag{5}$$

where c_i and c_j is the cluster membership of node i and j and $d_i \in \mathbb{R}^{|V|}$ is the degree of the node. Direct optimizing the binary cluster membership matrix $\mathcal{C}$ is NP-Hard. In GNN_Φ, we model node k's soft cluster membership c_k following [26]. We parameterize $\mathbf{Q}_\Phi(c_k|x_k, A)$ as $\mathbf{Q}_\phi \circ \text{GNN}_\phi$, using output embeddings and a softmax layer - $\mathbf{Q}_\phi(c_k|z_k) \sim \text{Categorical}(\phi)$,

$$\mathbf{Q}_\phi(c_k|z_k) = \text{softmax}(W^T z_k), z_k = \text{GNN}_\Phi(x_k, \mathbf{A}), \tag{6}$$

$$\mathcal{L}^t_\Phi = \frac{1}{2|E|}\text{Tr}\left(\mathbf{Q}^\intercal_\phi d^\intercal d\mathbf{Q}_\phi - \mathbf{Q}^\intercal_\phi A^t \mathbf{Q}_\phi\right), \tag{7}$$

The optimized cluster distribution is the soft cluster membership output by the GNN in Equation 6.

Given cluster distribution $\mathbf{Q}^t_\Phi(c_i|x^t_i, A^t)$ on target graph, we are interested in its correlations with true $\mathbb{P}^t(y^t_i|x^t_i, A^t)$. In practice, *graph homophily*-based clustering algorithm would work consistently

good on source and target *w.r.t.*, that is, $\mathbf{Q}^s_\Phi(c_i|x^s_i, A^s) \approx \mathbb{P}^s(y^s_i|x^s_i, A^s) \leftrightarrow \mathbf{Q}^t_\Phi(c_i|x^t_i, A^t) \approx \mathbb{P}^t(y^t_i|x^t_i, A^t)$. Besides the modularity clustering, lots of the graph clustering algorithms either explicitly or implicitly preserves *graph homophily*. The choices of clustering algorithms are very flexible as shown in Section D.3.

To approximate the class conditional probability on source, first we need to build a mapping from the graph clusters C to the class labels Y using source graph. The number of clusters is set to be equal or greater than number of classes N ($K \geq N$). Thus, we could build a bipartite mapping between the clusters $\{c^K_{k=1}\}$ and the classes $\{y^N_{n=1}\}$. We use the KL divergence to measure the distance between cluster c_k and class y_n over source data. Specifically, we search for the optimal mapping $\mathcal{M} \in \{0,1\}^{K\times N}$ by solving a linear sum assignment cost [14] between them, defined as follows,

$$\begin{aligned} &\min \sum_k \sum_n T_{k,n}\mathcal{M}_{k,n}, \\ &T_{k,n} = \mathrm{KL}\left(\mathbf{P}_\theta(y_n|H^s)||\mathbf{Q}_\phi(c_k|Z^s)\right), \\ &\text{subject to } \forall \sum_k \mathcal{M}_{k,n} \leq 1, \sum_n \mathcal{M}_{k,n} \leq 1 \end{aligned} \tag{8}$$

Thus we can map some clusters to classes, *i.e.*, $C_L \to Y, C_L \subseteq C, |C_L| = N$. After this step, the clusters on source graph are "colored" in Figure 2. In order to achieve similar class distribution on source data ($\mathbf{Q}^s_\Phi \approx \mathbb{P}^s$), we add a regularization term on clustering,

$$\mathcal{L}^s_{\Phi|\Theta} = \sum_{i=1}^{B} \mathrm{KL}\left(\mathbb{P}^s(\mathbf{y}|h^s_i)||\mathbf{Q}_\phi(\mathbf{c}|z^s_i)\right). \tag{9}$$

If labels are all known in source data, we random sample B nodes and $\{\mathbb{P}^s(y^s_i|h^s_i)\}$ is a one-hot vector. Otherwise (semi-supervised), we use current classifier's inference probability on source data $\mathbf{P}_\theta(\mathbf{y}|h^s_i)$.

4.3 Model Optimization

In this section, we summarize the training objective for classification and clustering modules. Following Equation 4, the overall loss for classification is,

$$\mathcal{L}_\Theta = \mathcal{L}^s_\Theta + \mathcal{L}^t_{\Theta|\Phi} + \alpha\mathcal{L}^s_{\Theta|\Phi}, \tag{10}$$

where the last term (pseudo samples $(x^s_i, \hat{y}^s_i)$ on unknown source nodes) is added to ensure training convergence. We have $\alpha = 1$ for semi-supervised $\alpha = 0$ for full-supervised source data.

Similarly, the overall loss for clustering also includes both source and target data,

$$\mathcal{L}_\Phi = \mathcal{L}^t_\Phi + \mathcal{L}^s_{\Phi|\Theta}, \tag{11}$$

Joint Optimization. As we can see in the loss function, both modules require inference result (freezing parameters) from the other. Therefore, we initialize both models with $\mathcal{L}^s_\Theta$ and $\mathcal{L}^t_{\Theta|\Phi}$, that is pre-training classification with labeled source graph and clustering with unlabeled target graph. We then iteratively train the classification module and clustering module **w.r.t.** $\mathcal{L}_\Theta$ and $\mathcal{L}_\Phi$. In $\mathcal{L}^t_{\Theta|\Phi}$, we sample same amount of nodes as labeled source data $\mathcal{D}^s$ on target. In regularization loss $\mathcal{L}^s_{\Phi|\Theta}$, we train clustering GNN_Φ for T steps and sample B nodes each step. Lastly, Algorithm 1 summarizes the joint optimization algorithm.

Convergence. With the Variational-EM model [21], we now discuss the convergence of graph clustering co-training on source graph. In the previous sections, we use categorical distribution $\mathbf{P}_\theta, \mathbf{Q}_\phi$ over hidden representation H, Z for two modules. To analyze the convergence, we need to detail both terms with all of their parameters as $\mathbf{P}_\Theta(\mathbf{y}|\mathbf{X}^s, \mathrm{A}^s) = \mathbf{P}_\theta(\mathbf{y}|\mathrm{GNN}_\Theta(\mathbf{X}^s, \mathrm{A}^s))$ and $\mathbf{Q}_\Phi(\mathbf{c}|\mathbf{X}^s, \mathrm{A}^s) = \mathbf{Q}_\phi(\mathbf{c}|\mathrm{GNN}_\Phi(\mathbf{X}^s, \mathrm{A}^s))$ For simplicity, we denote $\mathbf{G} = (\mathbf{X}^s, \mathrm{A}^s)$ as inputs on source data. In this analysis, clustering GNN serves as a variational distribution to approximate an intractable probability distribution $\mathbf{P}_\Phi$. Essentially, we optimize the evidence lower bound (ELBO) of the log-likelihood as follows,

$$\begin{aligned} &\log \mathbf{P}_\Theta(\mathbf{y}|\mathbf{X}^s, \mathrm{A}^s) \geq \\ &\mathbb{E}_{\mathbf{Q}_\Phi(\mathbf{y}|\mathbf{X}^s, \mathrm{A}^s)}\left[\log \mathbf{P}_\Theta(\mathbf{y}|\mathbf{X}^s, \mathrm{A}^s) - \log \mathbf{Q}_\Phi(\mathbf{y}|\mathbf{X}^s, \mathrm{A}^s)\right], \end{aligned} \tag{12}$$

In t-th E-step, regarding the Equation 12, the optimal $\mathbf{Q}_\Phi^{(t+1)}$ is,

$$\log \mathbf{Q}_\Phi^{(t+1)}(\mathbf{y}|\mathbf{G}) = \mathbb{E}_{\mathbf{Q}_\Phi^{(t)}(\mathbf{y}|\mathbf{G})}\left[\log \mathbf{P}_\Theta^{(t)}(\mathbf{y}|\mathbf{G})\right] + \text{const}, \tag{13}$$

The solution of the above E-step is,

$$\mathbf{Q}_\Phi^{(t+1)}(\mathbf{y}|\mathbf{G}) = \operatorname{argmin} \mathrm{KL}(\mathbf{P}_\Theta^{(t)}(\mathbf{y}|\mathbf{G})||Q_\Phi^{(t)}(\mathbf{y}|\mathbf{G})).$$

In practice, we achieve this update by sample B nodes on source graph, which is exactly the regularization term $\mathcal{L}^s_{\Phi|\Theta}$ (Equation 9).

In t-th M-step, we update the output distribution on source from classifier $\mathbf{P}_\Theta^{(t+1)}$ with $\mathbf{Q}_\Phi^{(t+1)}(\mathbf{y}|\mathbf{G})$. To estimate the expectation term in Equation 12, we sample $(x_j^s, \hat{y}_j^s) \sim \mathbf{Q}_\Phi$ to compute the log-likelihood and update parameter Θ of the classifier. Given the alignment between cluster and classes in Section 4.2. Interestingly, we describe the same sampling process on target in $\mathcal{L}^t_{\Theta|\Phi}$ and now we add $\mathcal{L}^s_{\Theta|\Phi}$ in final loss (Equation 10) of classifier to accomplish the M-step. Therefore, at each episode, the joint optimization is equivalent to perform variational EM alternatively. The convergence is proven in the original paper [21].

Discussion. Micro-graph [36] employs similar clustering and EM optimization between node and motif representations for the purpose of contrastive learning. The main difference is that we adopt EM between classification and clustering to achieve domain invariant learning under similar homophily ratios. Limitation of our method are two folds: (a) it requires access of test data as other domain adaptation algorithm (b) the performance of clustering co-training is not guaranteed when testing graph is more heterophily compared with training.

Complexity. Compared with common domain adaptation algorithms, our main difference in computation comes from graph clustering co-training. Hereby, we analyze the additional computation cost of $\mathbf{Q}$ in each episode. Let $\mathcal{O}(\mathcal{E})$ be the time GNNs take to compute a single node embedding and class or cluster membership for each node. Assuming there are $|V^t|$ nodes in target graph and node degree is constant order (*i.e.* $\mathcal{O}(|V^t|) \neq \mathcal{O}(\mathcal{E})$), the pre-training of clustering takes $\mathcal{O}(|V^t| \cdot \mathcal{E})$. During iterative optimization, the KL-divergence regularization costs $\mathcal{O}(2 \cdot \mathcal{E} \cdot B \cdot T)$, T is the number steps in Algorithm 1. Overall, the extra complexity $\mathcal{O}((|V^t| + 2 \cdot B \cdot T) \cdot \mathcal{E})$ is in linear of target graph size $|V^t|$, because usually $2 \cdot B \cdot T \ll |V^t|$. It is the same as most other domain adaption algorithms [33, 32], where target node representations are calculated each episode as well.

5 Experiments

5.1 Experimental Setting

Datasets. In our experiments, we perform semi-supervised node classification on four benchmark networks (see Table 4). These four networks are: Cora, Citeseer, PubMed [24] and ogb-arxiv [9]. We conduct open-set shift experiments on first three datasets and close-set shift on ogb-arxiv, because ogb-arxiv naturally has distribution shift across different time periods.

Parameter Settings and Reproducibility. In experiments, we use the GCN [11] and add self-loop in graph for all compared algorithms including ours. The Adam SGD [10] optimizer is used for training with learning rate as 0.01 and weight decay as 0.0005. The hidden size of all GNN models including SRNC is 128. We set cluster number as 16 in cluster GNN_Φ for open-set experiment and same as number of classes in close-set experiment. The pre-training epochs of episode 0 (Algorithm 1) for classification and clustering are set as 200 and 1000, respectively. Besides, B is set as the same as number of training nodes and $T = 500$ in co-training. Note that none of Cora, Citeseer and Pubmed has 16 classes, which do not favor our method as a fair comparison.

To provide a comprehensive study of SRNC, we design two ablations of our algorithms:

- SRNC **w.o** Φ: a bootstrapping ablation, where we pseudo label the same amount of data on target using model θ predictions instead of co-trained clustering Φ in Equation 4.
- SRNC Ep.1: we stop the algorithm after episode 1 to verify the effectiveness of iterative optimization.

Table 1: Performance under close-set shift on ogb-arxiv.

Method	ogb-arxiv		
	2014-2016	2016-2018	2018-2020
DGI	52.6 ± 0.4	48.3 ± 1.9	50.9 ± 1.4
DGI-DANN	48.9 ± 1.5	44.4 ± 3.1	28.2 ± 0.7
DGI-CMD	44.5 ± 0.6	36.5 ± 1.0	31.0 ± 1.9
DGI-SRGNN	50.5 ± 1.8	49.7 ± 2.7	47.7 ± 2.2
GCN	56.2 ± 0.5	55.7 ± 0.8	53.8 ± 1.2
GCN-DANN	54.3 ± 1.0	50.4 ± 3.2	46.2 ± 5.0
GCN-CMD	50.7 ± 0.6	48.7 ± 1.5	50.0 ± 2.3
GCN-SRGNN	54.4 ± 0.6	53.3 ± 1.1	55.0 ± 1.1
GCN-UDA	57.3 ± 0.4	56.5 ± 0.5	57.5 ± 1.6
GCN-EERM	50.4 ± 1.6	50.4 ± 2.7	51.0 ± 2.8
SRNC **w.o** Φ	57.3 ± 0.2	58.0 ± 0.8	55.6 ± 1.7
SRNC Ep.1	56.9 ± 0.1	56.0 ± 0.4	54.5 ± 0.1
SRNC	$\mathbf{58.1 \pm 0.3}$	$\mathbf{58.7 \pm 0.8}$	$\mathbf{59.1 \pm 1.3}$

5.2 Close-Set Shift on Dynamic Graph

Compared Algorithms. On close-set shift, we provide the result from multiple close-set domain adaption methods on both supervised and unsupervised GNNs: GCN, DGI [28] with DANN [5], CMD [35], SRGNN [39], UDAGCN [32] and EERM [34]. The first three methods are model-agnostic, so we applied to both GNNs. The latter two require end-to-end training witn GNNs.

Evaluation Setting. On ogb-arxiv, we split the train, validation and test nodes by year. There are total 12974 (-2011, train), 28151 (2011-2014, validation), 28374 (2014-2016, test set 1), 51241 (2016-2018, test set 2) and 48603 (2018-2020, test set 3) paper, respectively. In testing, each test graph contains all previous nodes such that train graph is a subset of all test graphs. This dynamic graph setting is close to real world applications, where model is trained on an earlier snapshot (source) and deployed on more recent ones (target).

Results. Most of close-set domain adaption baselines in comparison (*i.e.* DANN, CMD, SRGNN), focus on learning invariant feature representations between source and target. However, as we stressed before, conditional shifts presumably exist in real-world graphs so regularization (CMD) and adversarial head (DANN) make the performance even worse compared with original model. Then we are wondering whether unsupervised representation learning is more robust to (close-set) distribution shifts. We optimize the mutual information objective in DGI on training graph to obtain testing set node embeddings and train a multi-layer perceptron (MLP) classifier with training data. From Table 1, we can observe the testing performance of DGI is worse than GCN across different testing periods. Similarly, existing domain invariant approach only makes the result worse on DGI. The performance drop is even larger than their GCN counterparts. It is probably because the distribution shifts happen mostly in GNN encoders and adaption method can not mitigate the shifts in pre-trained graph encoders.

The performance of SRNC is significantly better than other baselines across all test period because we are able to capture the joint distribution $\Pr_{\text{test}}(X, Y)$ on target data. Besides our method, GCN-UDA is the second best method, it also considers graph structure information via inter-graph attention and therefore, outperforms DANN (domain classifier only) a lot. EERM reports the similar performance of the original paper on ogb-arxiv dataset, but only present consistent but not superior performance. As ablations, the bootstrapped pseudo labels (SRNC **w.o.** Φ) from classifier can only marginally improve the performance and the improvement on GCN is much smaller as shifts become larger(2018-2020 split). SRNC Ep.1 also reports worse results since the algorithm is not converged yet. Our results demonstrate that our framework is effective under different close-set shift introduced by dynamic graph.

5.3 Open-Set Classification

Compared algorithms. Since most of the close-set domain adaption algorithms cannot deal with open-set shift, we apply a different set of open-set classification baselines: THS [8], DOC [25], PGL [17], OpenWGL [33].

Table 2: Open-set classification on three different citation networks. Numbers reported are all percentage (%).

Method	Cora			Citeseer			PubMed		
	Micro-F1↑	Macro-F1↑	ΔF1↓	Micro-F1↑	Macro-F1↑	ΔF1↓	Micro-F1↑	Macro-F1↑	ΔF1↓
DGI-IID	83.4± 3.0	81.1 ± 1.7	0	75.3 ± 2.3	68.9 ± 4.9	0	80.2 ± 0.6	80.2 ± 0.5	0
DGI-THS	70.2 ± 4.2	68.9 ± 4.0	13.2	62.9 ± 5.1	56.4 ± 10.7	12.4	63.8 ± 7.4	58.0 ± 3.1	16.4
DGI-DOC	71.2 ± 3.6	70.7 ± 3.0	12.2	56.6 ± 8.1	57.0 ± 8.5	18.7	58.0 ± 4.6	57.4 ± 2.3	22.2
GCN-IID	82.0 ± 3.0	79.7 ± 1.6	0	75.0 ± 2.4	68.0 ± 4.4	0	79.3 ± 0.3	78.8 ± 0.3	0
GCN-THS	72.4 ± 3.7	71.7 ± 3.7	9.6	66.7 ± 3.4	61.5 ± 7.0	8.3	64.2± 2.8	58.9 ± 6.2	15.1
GCN-DOC	72.8 ± 3.4	72.8 ± 3.0	9.2	66.0 ± 5.0	63.8 ± 7.1	9.0	58.5 ± 7.0	47.5 ± 2.0	20.8
GCN-PGL	72.1 ± 4.4	70.9 ± 4.8	9.9	67.0 ± 5.2	60.0 ± 9.4	8.0	63.6 ± 3.8	57.8 ± 7.0	15.7
OpenWGL	66.7 ± 6.1	64.3 ± 5.7	15.3	64.5 ± 3.8	56.1 ± 7.0	11.5	64.2 ± 2.9	64.1 ± 2.5	15.1
SRNC **w.o** Φ	71.7 ± 6.4	70.2 ± 3.6	10.3	65.5 ± 4.7	56.2 ± 4.5	9.5	65.8 ± 1.6	60.5 ± 7.4	13.5
SRNC Ep.1	76.0 ± 4.7	75.2 ± 2.9	6.0	69.2 ± 5.8	60.4 ± 6.0	1.9	67.3 ± 5.1	68.0 ± 3.9	12.0
SRNC	**77.4 ± 4.0**	**75.9 ± 3.6**	**4.6**	**70.7 ± 4.0**	**63.4 ± 7.4**	**4.3**	**69.1 ± 4.4**	**69.4 ± 2.5**	**10.2**

Evaluation Setting. We create open-set shift by removing 3 classes on Cora/Citeseer and one class on PubMed from training data. In testing, nodes from the masked classes are all re-labeled as the unknown class. For each known class, we random sample 20 nodes for training and report the mean and standard deviation of micro-F1 and macro-F1 for 10 runs. Besides, we report the performance of GCN and DGI with the unknown label in training data (GCN-IID and DGI-IID) in Table 2, and relative performance drop (of other methods) in Micro (ΔF1). In validation set, we have nodes from the unknown class and use it to select the best hyper parameters all of the baselines and SRNC. For example, in PGL, we use validation to pick the best threshold α among each episodes $\{\alpha_k\}$ and label nodes with lower probabilities into unknown class. SRNC has an explicit class for unknown, so we use the Micro-F1 on validation to stop the iterative optimization.

Results. Compared with reference IID version (GCN-IID and DGI-IID), all methods experience performance drop due to open-set shift. Among the baselines, threshold based methods GCN-PGL and GCN-THS perform better than the others. OpenWGL, to our surprise, reports much worse performance in semi-supervised learning , it is probably because the uncertainty difference between known classes and unknown class is not that significant without enough training data. We also notice the open-set shift is more severe in PubMed, although its unknown ratio (1/3) is lower than the other two. In literature, when training data is IID (no open-set shift), unsupervised or contrastive learning is even better than semi-supervised GNNs with the same amount of training data. However, end-to-end supervised GNNs (GCN) is more robust than unsupervised GNNs (DGI) when there is open-set shift. It is consistent with the close-set observations and for the same reason semi-supervised GNNs can enjoy more improvements from domain adaptation algorithms.

On average, SRNC outperforms all the other representative open-set algorithms for at lease 4% and 2% in micro-F1 and macro-F1, respectively. From the comparison between our own ablations, we find that the graph clustering co-training(SRNC vs. SRNC **w.o.** Φ) contributes the most to the performance gains. This ablation performs bootstrapping that randomly samples pseudo unknown nodes from classifier's low confident predictions. In other words, our co-training design is clearly better than bootstrapping and drawing samples from unseen clusters is the key towards better open-set generalization. Moreover, the iterative brings more improvement (1$\sim$3% F1) on top of SRNC Ep.1 ablation since variational EM takes several round to converge in practice.

6 Conclusion

In this paper, we proposed a general framework, SRNC, to enable graph neural networks for distributional shift. Different from existing work on either *open-set shift* or *close-set shift*, SRNC works for both scenario with a novel graph clustering co-training component. Accordingly, SRNC employ a latent variable model (*i.e.* cluster GNN) to model the latent structure and allows the clustering GNN to improve the generalization of shift-robust classification GNN. Future work includes applying the model to applications such as medical diagnosis and molecular dynamics, where robustness towards distribution shifts are in critical need.

References

[1] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Mach. Learn.*, 79(1–2), 2010.

[2] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *CVPR*, pages 1563–1572, 2016.

[3] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*, pages 874–883. PMLR, 2020.

[4] Ruichu Cai, Fengzhu Wu, Zijian Li, Pengfei Wei, Lingling Yi, and Kun Zhang. Graph domain adaptation: A generative view. *arXiv preprint arXiv:2106.07482*, 2021.

[5] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Franccois Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 2016.

[6] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pages 3419–3430. PMLR, 2020.

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

[8] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[9] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[12] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[13] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

[14] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[15] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. ICML'15, 2015.

[16] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In *ICML*. JMLR.org, 2017.

[17] Yadan Luo, Zijian Wang, Zi Huang, and Mahsa Baktashmotlagh. Progressive graph learning for open-set domain adaptation. In *International Conference on Machine Learning*. PMLR.

[18] Jiaqi Ma, Junwei Deng, and Qiaozhu Mei. Subgroup generalization and fairness of graph neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

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

[20] Haitao Mao, Lun Du, Yujia Zheng, Qiang Fu, Zelin Li, Xu Chen, Han Shi, and Dongmei Zhang. Source free unsupervised graph domain adaptation. *arXiv preprint arXiv:2112.00955*, 2021.

[21] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[22] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[23] Walter J Scheirer, Lalit P Jain, and Terrance E Boult. Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2317–2324, 2014.

[24] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[25] Lei Shu, Hu Xu, and Bing Liu. Doc: Deep open classification of text documents. *arXiv preprint arXiv:1709.08716*, 2017.

[26] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *arXiv preprint arXiv:2006.16904*, 2020.

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

[28] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[29] Saurabh Verma and Zhi-Li Zhang. Stability and generalization of graph convolutional neural networks. In *KDD*, 2019.

[30] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Wenjun Zeng, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. *arXiv preprint arXiv:2103.03097*, 2021.

[31] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[32] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. Unsupervised domain adaptive graph convolutional networks. In *Proceedings of The Web Conference 2020*, pages 1457–1467, 2020.

[33] Man Wu, Shirui Pan, and Xingquan Zhu. Openwgl: Open-world graph learning. In *ICDM, 2020, Sorrento, Italy*, 2020.

[34] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. *arXiv preprint arXiv:2202.02466*, 2022.

[35] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. *arXiv*, 2017.

[36] Shichang Zhang, Ziniu Hu, Arjun Subramonian, and Yizhou Sun. Motif-driven contrastive learning of graph representations. *arXiv preprint arXiv:2012.12533*, 2020.

[37] Yizhou Zhang, Guojie Song, Lun Du, Shuwen Yang, and Yilun Jin. Dane: Domain adaptive network embedding. *arXiv preprint arXiv:1906.00684*, 2019.

[38] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *arXiv preprint arXiv:2006.11468*, 2020.

[39] Qi Zhu, Natalia Ponomareva, Jiawei Han, and Bryan Perozzi. Shift-robust gnns: Overcoming the limitations of localized graph training data. *Advances in Neural Information Processing Systems*, 34, 2021.

A Algorithm details

Algorithm 1: Pseudo code for SRNC optimization

/* Episode 0 */
Pre-train the classifier GNN_Θ via $\mathcal{L}^s_\Theta$ and the cluster GNN_Φ via $\mathcal{L}^t_\Phi$;
/* Iterative optimization, episode 1,2...*/
while *Micro-F1 on validation increases* **do**
  /* update cluster GNN_Φ for T steps each episode */
  for *t = 1 to T* **do**
    Sample B nodes (x^s_i, y^s_i) from source
    $\Phi \xleftarrow{-} \nabla_\Phi \mathcal{L}^s_\Phi + \nabla_\Phi \mathcal{L}^t_{\Phi|\Theta}$
  end
  /* update shift-robust classifier GNN_Θ */
  Sample $|\mathcal{D}^s|$ nodes $(x^t_i, \hat{y}^t_i)$ from target
  while $\mathcal{L}_\Theta$ *not converge* **do**
    $\Theta \xleftarrow{-} \nabla_\Theta \mathcal{L}^s_\Theta + \nabla_\Theta \mathcal{L}^t_{\Theta|\Phi} + \nabla_\Theta \alpha \mathcal{L}^s_{\Theta|\Phi}$
  end
end

In addition, we provide the comparison between domain adaptation graph neural networks in Table 3.

Table 3: Comparison of graph neural networks on domain adaptation. *open-set shift*: Recognizing new classes in testing data. *covariate shift*: Handling distribution shifts when class-conditional distribution does not change, $\Pr_{\text{train}}(Y|X) = \Pr_{\text{test}}(Y|X)$. *conditional shift*: Handling distribution shifts when class-conditional distribution does change, $\Pr_{\text{train}}(Y|X) \neq \Pr_{\text{test}}(Y|X)$.

	SRNC	**PGL** [17]	**UDA** [32]	**SRGNN** [39]	**EERM** [34]
open-set	✔	✔			
covariate	✔		✔	✔	✔
conditional	✔	✔			✔

B Experimental Setups

B.1 Dataset Details

Table 4: Overall Dataset Statistics

Dataset	# Nodes	# Edges	# Classes	# Train
Cora	2,708	5,429	7	140
Citeseer	3,327	4,732	6	120
PubMed	19,717	44,325	3	100
ogb-arxiv	169,343	2,501,829	40	12974

B.2 Evaluation Metrics.

In our experiments, we have $\{(x^t, y^t)\}$ in testing (target) data $\mathcal{D}^t$ and calculate Micro-F1 and Macro-F1 on predictions $\hat{y}$ from target data,

$$\text{Micro-F1} = \frac{|x \in D^t \wedge \hat{y} = y^t|}{|(x, y) \in D^t|} \tag{14}$$

$$\text{Macro-F1} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \frac{|x \in D^t \wedge \hat{y} = y^t|}{|x \in D^t \wedge y^t = i|} \tag{15}$$

Notice that in open-set shift, target data $\mathcal{D}^t$ has one more class than training - the unknown class ($N_{\text{test}} = N_{\text{train}} + 1$)).

C Additional Experiments

C.1 Unseen Class Discovery

Table 5: Top paper titles from discovered clusters on DBLP dataset. We choose the two most similar clusters for comparison. **(x)** indicates the paper is not consistent with the area of other papers in the same cluster.

suggested unseen	SRNC	**Constrained K-Means (DGI)**
Computational Mathematics	• a meshless method for solving nonlinear two-dimensional ... domains using radial basis functions with error analysis • application of the ritz-galerkin method ... wave equation • a generalized moving least square reproducing kernel method • multi-rate multicast ... multi-radio wireless mesh networks **(x)** • numerical solution of the higher-order ... variable coefficients equation with variable coefficients	• upper and lower bounds for dynamic cluster assignment for multi-target tracking in heterogeneous wsns**(x)** • multi-rate multicast ... multi-radio wireless mesh networks**(x)** • matrix equations over (r,s)-symmetric and ... symmetric matrices • a tau approach for solution of the ... fractional diffusion equation • a meshless method for solving nonlinear two-dimensional ... domains using radial basis functions with error analysis
Networks & Distributed Systems	• an implementation and evaluation ... mobile ad-hoc networks • trustworthiness among peer ... distributed agreement protocol • reduction of processing ... synchronize multmedia replicas • quorum-based replication of multimedia ... distributed systems • dynamic clusters of servers to reduce total power consumption • experimentation of group communication protocols • a dynamic energy-aware server selection algorithm • trustworthiness ... in peer-to-peer(p2p) overlay networks	• a pattern-based ... for a multi-core system development**(x)** • model checking prioritized timed systems**(x)** • learning-based adaptation to ... reconfigurable network-on-chip **(x)** • design issues in a performance ... embedded multi-core systems **(x)** • the architecture of parallelized cloud-based ... testing system • counterexample-guided assume-guarantee synthesis learning **(x)** • mechanism design ... allocation in non-cooperative cloud systems • a qos-aware uplink scheduling paradigm for lte networks

Use Case. DBLP is a computer science bibliography website. We construct a heterogeneous network upon three types of nodes: **(A)**Author, **(P)**Paper and **(T)**Term. There are five types of papers labeled by the corresponding venues[2] - Data Mining **(DM)**, Database **(DB)**, Artificial Intelligence **(AI)**, Natural Language Processing **(NLP)** and Computer Vision **(CV)**. We sampled 50,000 papers between 2010 and 2015, then use the venue information to label 5567 of them into 5 classes. After filtering out rare authors and terms, there are 20601 papers, 5158 authors and 2000 terms in total. In this subset, we observed only 30 % of 20,000 papers are labeled into known categories. In other words, the majority of nodes in the DBLP graph belong to unseen classes. This is a real situation where people do not know the label space in advance and it poses an open-set challenge for practitioners. One biggest advantange of SRNC is the capability of unseen class discovery clustering GNN GNN_Φ. Unlike any other open-set classification algorithms, a graph clustering component can further partition the unknown objects into different groups based on their structural and semantic similarities.

Results. In Table 5, we show the paper in DBLP dataset that are assigned to the (unmatched) clusters, discovered by our Cluster GNN_Φ and Constrained-KMeans + DGI embeddings. Two methods utilize the same amount of supervision of known classes and we set total number of clusters as 10. Interestingly, from the five clusters that are supposed to belong to unseen classes, we find two shared clusters about "Computational Mathematics" and "Network & Distributed System" from both algorithms. The results show ad-hoc semi-supervised clustering on node embeddings produces erroneous cluster membership even for the top-ranked items. For example, it contains almost half papers from the computer architecture in "Network & Distributed System", whereas unseen classes suggested by our GNN clusterring are much more consistent. Traditional clustering on off-the-shelf network embeddings lack rationales of objects within the same clusters. When our method optimize the ELBO between classification and clustering results on seen classes, it adopts the similar rationales in unseen clusters. The clustering result also provides insight for designing the new label space or active learning paradigms.

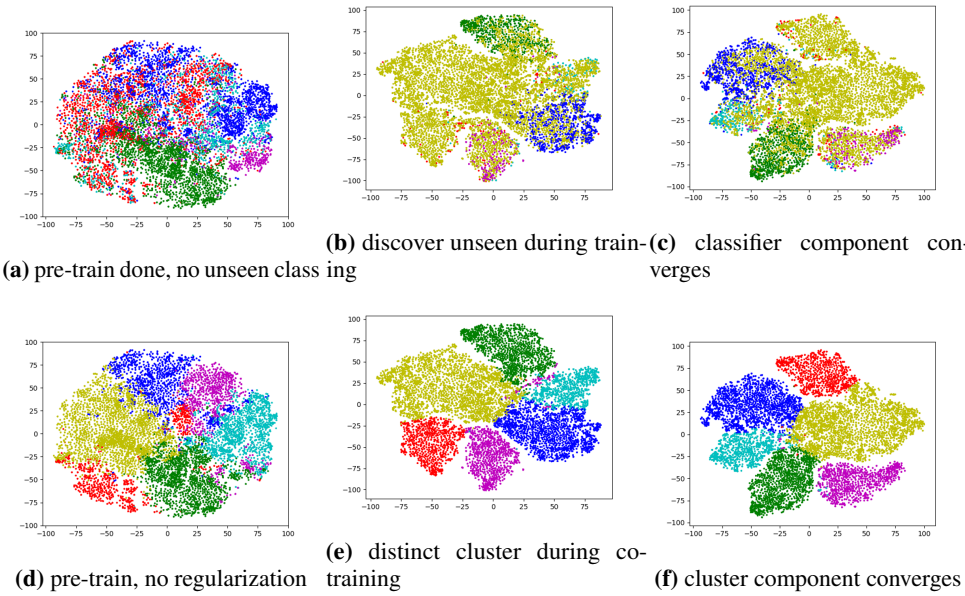
Table 6: Open-set performance improvement when number of cluster changes from 16 to 7 (best hyper-parameter) on Cora.

	Cora (1 unseen class)		Cora (3 unseen classes)	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Δ	+4.73	+3.79	+1.62	+1.15

D Model Analysis and Parameter Study

D.1 Parameter Sensitivity.

Among all the hyper-parameters, the pre-defined number of clusters $|C|$ seems to be the most crucial in open-set scenario. Now we provide the performance of SRNC with varied numbers of clusters k, *i.e.* 7 (optimal) vs. 16 (used in experiment) in Table 6. Both Micro-F1 and Macro-F1 are further boosted, since seven is the ground truth number of classes. On the one hand, SRNC is not sensitive to the choice of $|C|$ and better than baselines regardless, especially when there are multiple missing classes (closer to the real-world setting). On the other, an accurate estimation of $|C|$ can facilitate the model training.



(a) pre-train done, no unseen class

(b) discover unseen during training

(c) classifier component converges

(d) pre-train, no regularization

(e) distinct cluster during co-training

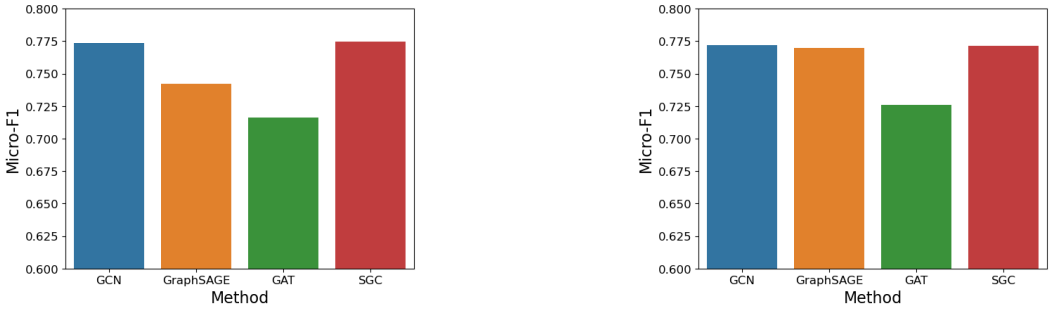
(f) cluster component converges

Figure 3: Visualization of training process on DBLP dataset (Best viewed in color). The same color indicates the aligned class/cluster (Red: AI, Blue: CV, Cyan: DB, Green: DM, Purple: NLP). Color yellow represents the unseen class detected by the classifier.

D.2 Effectiveness of iterative optimization.

In Figure 3, we demonstrate the learning procedure on DBLP dataset. Each column shows the predicted labels from classification GNN (upper) and the cluster assignments from the cluster GNN (lower). We project the node embeddings onto two dimensions using t-SNE. In this example, we set number of clusters to six. In Figure 3a, the pre-trained classification GNN predicts all papers into 5 seen classes. Our cluster GNN shows 6 initial clusters. During the training, the clusters in Figure 3e gradually align with the classifier's predictions in same color. The classifier presented in Figure 3b starts to recognize nodes of unseen class with pseudo samples from cluster corresponds to unseen class (yellow) in Figure 3e. When the training converges, the clusters in Figure 3f are separated clearly.

[2]**DM**: SIGKDD, ICDM **DB**: SIGMOD, VLDB **NLP**:ACL, EMNLP, NAACL **AI**: AAAI, IJCAI, ICML **CV**: ECCV, ICCV, CVPR



(a) Modularity Clustering

(b) DGI+DEC

Figure 4: Varying GNNs and clustering methods on Cora.

D.3 Applicability of SRNC framework.

As we discusses in section 4, SRNC is not tailored for one specific GNN architectures or unsupervised graph clustering algorithms. In this set of experiments, we investigate four different classification models for GNN_θ (GCN [11], GraphSAGE [7], GAT [27], SGC [31]). These four GNN models are widely used for different applications and effective on the Cora dataset we used in this study. Besides, we investigate two rather different clustering algorithms on graph - DMoN [26] (used in main experiment), DGI+DEC [28] in Figure 4. Deep embedding clustering (DEC) is a self-trained clustering algorithms. We use K-means on the DGI node representations to initialize the cluster centers and DEC updates the cluster centers and same soft assignment $\mathcal{Q}$ matrix in Equation 7. We observed that in general, all of these variations are at least (2% on Micro-F1) better than the best performed baseline in the open-set experiments (Table 2). Specifically, four different GNN architectures yield similar performance except GAT, which is a bit complicated for domain adaptation on small graphs. Moreover, two different graph clustering algorithms report close numbers. It validates the convergence of variational EM is empirically reachable as long as the underlying class distributions are well captured by clustering algorithms.