Incremental Unsupervised Domain Adaptation on Evolving Graphs

Hsing-Huan Chung
Department of Electrical and Computer Engineering
The University of Texas at Austin
hhchung@utexas.edu

Joydeep Ghosh
Department of Electrical and Computer Engineering
The University of Texas at Austin
jghosh@utexas.edu

Abstract

Non-stationary data distributions in evolving graphs can create problems for deployed graph neural networks (GNN), such as fraud detection GNNs that can become ineffective when fraudsters alter their patterns. The aim of this study is to investigate how to incrementally adapt graph neural networks to incoming, unlabeled graph data after training and deployment. To achieve this, we propose a new approach called graph contrastive self-training (GCST) that combines contrastive learning and self-training to alleviate performance drop. To evaluate the effectiveness of our approach, we conduct a comprehensive empirical evaluation on four diverse graph datasets, comparing it to domain-invariant feature learning methods and plain self-training methods. Our contribution is three-fold: we formulate and study incremental unsupervised domain adaptation on evolving graphs, present an approach that integrates contrastive learning and self-training, and conduct a comprehensive empirical evaluation of our approach, which demonstrates its stability and superiority over other methods. The code is available at https://github.com/hsinghuan/iuda-graph.

1 Introduction

Dynamic data distributions in real-world graphs present a challenge for machine learning models. For instance, fraudsters may alter their patterns on transaction graphs to evade detection by existing fraud detection models, causing a decline in the model performance. While it would be ideal to periodically retrain the models in a supervised fashion, the labels may not always be immediately available in real-world scenarios. For example, the inspection of risk analysts to obtain the labels for fraud cases may take days or even more than a week (Dal Pozzolo et al., 2015; Tax et al., 2021), making it too late for the models to adapt instantly. Adaptation without the presence of labels post-deployment is therefore essential to reliable machine learning systems. In this study, we investigate how to adapt graph neural networks (GNNs) to incoming, unlabeled graph data after training and deployment.

The central focus of our work is on addressing an incremental unsupervised domain adaptation problem (Hoffman et al., 2014; Wulfmeier et al., 2018) in the context of node classification in temporally evolving graphs. The task consists of training a GNN on a labeled source graph snapshot and deploying it in a non-stationary environment, where it must be continually adapted to unseen, unlabeled target graph snapshots. Our objective is to design an adapter that minimizes the decline in performance and maximizes average accuracy over time. This particular problem has received limited attention in the existing literature, despite its great importance for real-world applications.

In this paper, we propose graph contrastive self-training (GCST) that leverages a synergistic combination of contrastive learning and self-training. Our adapter is given a graph neural network $f_{\theta,\phi} = g_\theta \circ h_\phi$ where h_ϕ is the graph encoder and g_θ is the prediction head. At each adaptation stage t, the adapter initializes the encoder and prediction head with parameters obtained from the previous stage, $h_{\phi_{t-1}}$ and $g_{\theta_{t-1}}$. It jointly minimizes the supervised loss on the source graph, pseudo-label loss on the target graph, and contrastive loss on both the source and target graph. The supervised loss and pseudo-label loss maintain the discriminability of the graph neural network. The contrastive loss disentangles class and domain information in the feature space to enable the prediction head to generalize across domains. Note that we do not apply domain-invariant feature learning objectives (Ganin et al., 2016), which have been the dominant approach for deep domain adaptation. Domain-invariant feature learning aims to learn an encoder that produces source and target features that are indistinguishable, while also maximizing source domain accuracy. However, such an objective has been shown to be detrimental to target performance when there exists label shift (Zhao et al., 2019), which is a common phenomenon in real-world graphs. Our experimental results demonstrate the stability and superior performance of our approach compared to domain-invariant feature learning and plain self-training methods.

Our contribution is three-fold:

- We formulate and study incremental unsupervised domain adaptation on evolving graphs, which is essential for real-world applications, yet has received limited attention in prior research.
- We present graph contrastive self-training (GCST) that integrates contrastive learning and self-training to effectively adapt graph neural networks to unlabeled, evolving graph data
- We conduct a comprehensive empirical evaluation of our approach, comparing its performance against both domain-invariant feature learning methods and plain self-training methods on four diverse graph datasets. The results demonstrate the stability and superiority of our approach.

2 Related Work

Unsupervised domain adaptation. Given a source and a target distribution, unsupervised domain adaptation (UDA) is the task of learning a classifier on labeled source data and unlabeled target data that performs well on the target distribution. Domain-invariant feature learning is a dominant approach that aims to jointly minimize the discrepancy between the distributions of source features and target features as well as the source classification loss (Ganin et al., 2016; Sun & Saenko, 2016; Saito et al., 2018; Long et al., 2015; 2017; 2018). Nevertheless, the limitations of domain-invariant feature learning under label shift have been revealed by impossibility theorems (Zhao et al., 2019). Self-training has appeared to be another competitive paradigm in unsupervised domain adaptation, which utilizes pseudo-labeling or entropy minimization to generalize the source classifier to the target domain (Liu et al., 2021; Berthelot et al., 2022; Zou et al., 2018; 2019; French et al., 2018; Prabhu et al., 2021). Another alternative to domain-invariant feature learning is contrastive learning followed by fine-tuning, which disentangles domain and class information in the feature space such that the classifier can simultaneously classify both source and target data (Shen et al., 2022). We take a synergistic approach combining contrastive learning and self-training to solve incremental UDA on evolving graphs.

Incremental unsupervised domain adaptation. Incremental unsupervised domain adaptation is the task of adapting a model to a continually changing environment as opposed to a single and discrete target domain. A pioneering work by Hoffman et al. (2014) extends manifold learning-based domain adaptation to this particular problem by learning a time-varying transformation between source and target points. Wulfmeier et al. (2018) propose to incrementally adapt a convolutional neural network feature extractor to the newest domains using domain adversarial learning. Under this problem setting, some previous studies also work on preventing catastrophic forgetting during adaptation (Bobu et al., 2018; Liu et al., 2020; Tang et al., 2021; Rostami, 2021). Gradual domain adaptation (GDA) is another closely related setting where the source-trained classifier adapts to the target domain by leveraging intermediate domains (Kumar et al., 2020; Abnar et al., 2021; Wang et al., 2022; He et al., 2022). Algorithms for GDA can be directly applied to our problem if we consider every time step as both a target domain and an intermediate domain for the future. A standard GDA algorithm, gradual self-training (GST) (Kumar et al., 2020), pseudo-labels the closest intermediate domain and re-trains the model on the pseudo-labeled data to gradually adapt to the target domain. Our approach inherits the property of GST in terms of pseudo-labeling over time, but considers updating pseudo-labels within each time step and maintaining the source loss component during adaptation.

Unsupervised domain adaptation on graphs. UDA on graphs is a less studied area compared to UDA on images. Most existing studies work on cross-network node classification which aims to transfer knowledge from a sufficiently-labeled source graph to a weakly-labeled target graph. AdaGCN (Dai et al., 2022), UDA-GCN (Wu et al., 2020), DANE (Song et al., 2022), and GNN-SpecReg (You et al., 2023) integrate domain adversarial loss into the training of graph convolutional networks. On the other hand, DGDA (Cai et al., 2021) applies a variational graph autoencoder to disentangle a node feature into three components: domain, semantic, and randomness, and uses the semantic part for classification. To the best of our knowledge, we are the first to study incremental UDA on evolving graphs.

3 Problem Formulation

3.1 Graph Neural Networks

We consider a graph $\mathcal{G} = \{\boldsymbol{A}, \boldsymbol{X}\}$, where $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix and N is the number of nodes. The node feature matrix $\boldsymbol{X} \in \mathbb{R}^{N\times F}$ has a node feature dimension of F. We denote a GNN for node classification as $f_{\theta,\phi} = g_\theta \circ h_\phi$, where h_ϕ is the graph encoder for computing node representations and g_θ is the prediction head for the downstream task. The GNN takes $\mathcal{G}$ as an input and outputs $\hat{\boldsymbol{Y}} = f_{\theta,\phi}(\mathcal{G}) \in \mathbb{R}^{N\times C}$, which represents the probabilistic predictions on all nodes, where C is the number of classes.

Suppose there is a set of indices $\mathcal{V}$ indicating the labeled nodes, and we denote the associated labels as $\boldsymbol{y}$. If we want to evaluate the node classification accuracy or impose a supervised training loss, we can compare $\boldsymbol{y}$ with $\hat{\boldsymbol{Y}}_{\mathcal{V}}$, where $\hat{\boldsymbol{Y}}_{\mathcal{V}}$ is a sub-matrix of $\hat{\boldsymbol{Y}}$ that specifies the predictions on labeled nodes.

3.2 Incremental Unsupervised Domain Adaptation on Evolving Graphs

We consider a realistic scenario where a GNN is trained on the labeled source graph but faces gradual distribution shifts after deployment. The source distribution is denoted as P_0 while the evolving target distributions are denoted as $P_1, P_2, \ldots, P_T$. We assume that the distribution shift is gradual: for some $\epsilon > 0, \rho(P_t, P_{t+1}) < \epsilon \,\forall\, 0 \leq t < T$, where ρ is some distance function for two distributions. A sequence of graph snapshots $\mathbf{G} = \{\mathcal{G}^{(0)}, \mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(T)}\}$ sampled from the evolving distributions are presented to the GNN and adapter over time. In the source training stage ($t = 0$), the GNN is trained with the labeled source graph snapshot $(\mathcal{G}^{(0)}, \boldsymbol{y}^{(0)})$. After source training, the adapter has access to unlabeled target graph snapshots $\mathcal{G}^{(t)}, 1 \leq t \leq T$ in a sequential manner.

We adopt the prequential evaluation protocol, also known as test-then-train, which is commonly used in studying concept drifts on data streams (Lu et al., 2019). In the scenario where a stream of graphs is directed towards the GNN and prediction requests are made, the model is required to provide predictions and then update itself accordingly to adapt to any distribution shift. In more concrete terms, the model is first tested on $(\mathcal{G}^{(1)}, \boldsymbol{y}^{(1)})$, then adapted on $\mathcal{G}^{(1)}$. In subsequent, it is tested on $(\mathcal{G}^{(2)}, \boldsymbol{y}^{(2)})$, and then adapted on $\mathcal{G}^{(2)}$, and so on and so forth. Note that the labels of the target data are only used for evaluation, not for adaptation. The goal of the adapter is to incrementally adapt a graph neural network $f_{\theta,\phi} = g_\theta \circ h_\phi$ that performs well over time, i.e.

$$\min_{\theta,\phi} \frac{1}{T} \sum_{i=1}^{T} \mathbb{E}_{\mathcal{G}^{(t)}, \boldsymbol{y}^{(t)} \sim P_t} l(f_{\theta,\phi}(\mathcal{G}^{(t)})_{\mathcal{V}^{(t)}}, \boldsymbol{y}^{(t)}) \tag{1}$$

where l is the loss function or evaluation metric and $\mathcal{V}^{(t)}$ is the index set of labeled nodes in $\mathcal{G}^{(t)}$.

4 Graph Contrastive Self-Training (GCST)

In this section, we describe our training and adaptation procedure. Before deployment and adaptation, the graph neural network $f_{\theta,\phi}$ is trained on the labeled source graph $\mathcal{G}^{(0)}$ in a supervised manner using the following objective:

$$\theta_0, \phi_0 = \arg\min_{\theta,\phi} L_{\mathcal{G}^{(0)}}(\theta, \phi) = \arg\min_{\theta,\phi} \sum_{v \in \mathcal{V}^{(0)}} l(f_{\theta,\phi}(\mathcal{G}^{(0)})_v, \boldsymbol{y}_v^{(0)}) \tag{2}$$

We use the cross-entropy loss between probabilistic predictions and labels as the loss function l. After deployment at stage t, the adapter adapt the GNN on $\mathcal{G}^{(t)}$ to produce θ_t and ϕ_t.

4.1 Self-Training

The goal of self-training is to generalize the GNN from source to future target domains utilizing pseudo-labels, which are relatively reliable under the assumption of gradual distribution shift. Our self-training loss contains two components, a source supervised loss $L_{\mathcal{G}^{(0)}}(\theta, \phi)$ and a target pseudo-label loss $PL_{\mathcal{G}^{(t)}}(\theta, \phi)$. Pseudo-labeling assigns a hard label to each node v by taking the class predicted with maximum confidence, i.e. $\arg\max_c \{\tilde{\boldsymbol{Y}}_{v,c}\}$ where $\tilde{\boldsymbol{Y}}$ is the probabilistic prediction matrix for pseudo-labeling. Practically, we can specify a threshold value $\tau \in [0, 1)$ to mask out unlabeled nodes whose prediction confidences are below τ. The general self-training loss that contains source supervised loss and target pseudo-label loss can be written as:

$$\begin{aligned} ST_{\mathcal{G}^{(0)},\mathcal{G}^{(t)}}(\theta, \phi) &= L_{\mathcal{G}^{(0)}}(\theta, \phi) + PL_{\mathcal{G}^{(t)}}(\theta, \phi) \\ &= \sum_{v \in \mathcal{V}^{(0)}} l(f_{\theta,\phi}(\mathcal{G}^{(0)})_v, \boldsymbol{y}_v^{(0)}) + \sum_{v=1}^{N^{(t)}} \mathbf{1}_{\max_c \tilde{\boldsymbol{Y}}_{v,c}^{(t)} \geq \tau} l(f_{\theta,\phi}(\mathcal{G}^{(t)})_v, \arg\max_c \{\tilde{\boldsymbol{Y}}_{v,c}^{(t)}\}) \end{aligned} \tag{3}$$

where $N^{(t)}$ is the number of nodes in $\mathcal{G}^{(t)}$ and $\tilde{\boldsymbol{Y}}^{(t)}$ is a probabilistic prediction matrix on $\mathcal{G}^{(t)}$ for pseudo-labeling.

We can further define two versions of pseudo-label loss depending on $\tilde{\boldsymbol{Y}}^{(t)}$ which are fixed pseudo-label loss $FPL_{\mathcal{G}^{(t)}}(\theta, \phi)$ and updated pseudo-label loss $UPL_{\mathcal{G}^{(t)}}(\theta, \phi)$. The first version is calculated with fixed pseudo-labels, which means the pseudo-labels at stage t are assigned by $f_{\theta_{t-1},\phi_{t-1}}$ and fixed during adaptation, i.e.

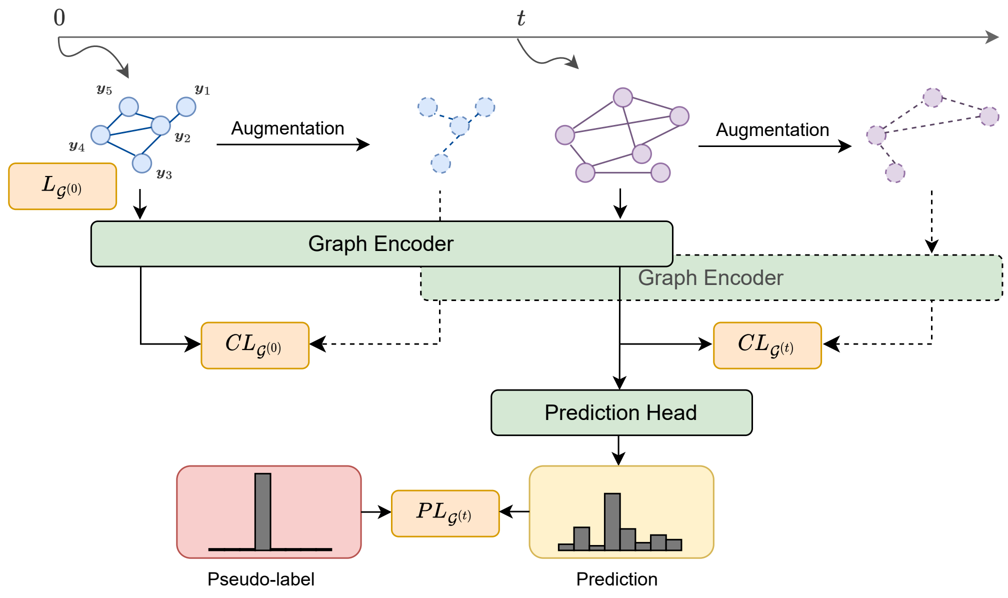


Figure 1: The procedure of graph contrastive self-training (GCST). GCST augments the source and current target graph snapshots and passes the two pair of graphs through graph encoders for contrastive training. In addition, GCST applies self-training using the labeled source graph and the pseudo-labeled target graph, where the pseudo-labels can be assigned by either the model trained in the previous stage or the current model.

$\tilde{\mathbf{Y}}^{(t)} = f_{\theta_{t-1},\phi_{t-1}}(\mathcal{G}^{(t)})$. By using the fixed pseudo-label loss, we believe the model trained in the previous stage is accurate enough to pseudo-label the current graph given that the distribution shift is gradual. Alternatively, we can update pseudo-labels with current model parameters during adaptation, i.e. $\tilde{\mathbf{Y}}^{(t)} = f_{\theta,\phi}(\mathcal{G}^{(t)})$. Using both $L_{\mathcal{G}^{(0)}}(\theta, \phi)$ and $UPL_{\mathcal{G}^{(t)}}(\theta, \phi)$ is essentially the classic pseudo-label approach for semi-supervised learning (Lee, 2013). On the other hand, only using $FPL_{\mathcal{G}_{(t)}}(\theta, \phi)$ will lead to gradual self-training(Kumar et al., 2020).

4.2 Graph Contrastive Learning

Although the pseudo-labels are sufficiently reliable under gradual distribution shifts, mislabeling a certain portion of nodes is inevitable. We introduce a contrastive learning objective that learns the underlying structure of graphs independent of pseudo-label quality, which can in turn assist self-training. The adapter minimizes the contrastive loss on the target graph $\mathcal{G}^{(t)}$ and the source graph $\mathcal{G}^{(0)}$ to learn node representations.

We follow the framework given by MVGRL (Hassani & Khasahmadi, 2020). Given a graph $\mathcal{G}$, the adapter first applies a graph structural augmentation function A to it, producing an alternative view of the graph, $A(\mathcal{G})$. $\mathcal{G}$ and $A(\mathcal{G})$ are passed through two separate graph encoders, h_ϕ and $h_{\phi'}$, both initialized with ϕ_{t-1}. Let $M_\mathcal{G}$ be a mask function that filters out representations of uncommon nodes between $\mathcal{G}$ and $A(\mathcal{G})$. We can obtain two sets of representations of the common nodes, $\boldsymbol{H}_+ = M_\mathcal{G}(h_\phi(\mathcal{G}))$ and $\boldsymbol{H}'_+ = M_\mathcal{G}(h_{\phi'}(A(\mathcal{G})))$. Afterwards, the two sets of representations are passed through an average readout function $AvgR$, which simply takes an average over the node dimension in $\boldsymbol{H}$ to produce a global graph representation. MVGRL maximizes the mutual information between the node representations of a view and the graph representation of the other view, i.e. $MI(\mathbf{H}_+, AvgR(\mathbf{H}'_+))$ and $MI(\mathbf{H}'_+, AvgR(\mathbf{H}_+))$. To obtain node representations of negative samples, the node features of two graph views are randomly shuffled and passed through the encoders and mask, producing $\mathbf{H}_-$ and $\mathbf{H}'_-$. MVGRL minimizes the mutual information between the negative sample node representations of a view and the graph representation of the other view, i.e. $MI(\mathbf{H}_-, AvgR(\mathbf{H}'_+))$ and $MI(\mathbf{H}'_-, AvgR(\mathbf{H}_+))$. The overall contrastive loss can be written as:

$$CL_\mathcal{G}(\phi, \phi') = -MI(\mathbf{H}_+, AvgR(\mathbf{H}'_+)) - MI(\mathbf{H}'_+, AvgR(\mathbf{H}_+)) + MI(\mathbf{H}_-, AvgR(\mathbf{H}'_+)) + MI(\mathbf{H}'_-, AvgR(\mathbf{H}_+)) \tag{4}$$

We follow the implementation of MVGRL, which uses a bilinear discriminator to model the mutual information between two representations. There are two differences between our implementation and the original MVGRL. Firstly, the original MVGRL uses graph diffusion as the augmentation function. We instead use random node dropout due to

Algorithm 1 Graph Contrastive Self-Training (GCST)

```
Require: $\theta_0, \phi_0, \{\mathcal{G}^{(0)}, \mathcal{G}^{(1)} \dots \mathcal{G}^{(T)}\}, \boldsymbol{y}^{(0)}, numIters, \eta, \alpha, \tau$
  for $t \leftarrow 1$ to $T$ do
      $\theta, \phi, \phi' \leftarrow \theta_{t-1}, \phi_{t-1}, \phi_{t-1}$
      for $i \leftarrow 1$ to $numIters$ do
          if update pseudo-labels then
              $\tilde{\boldsymbol{Y}}^{(t)} \leftarrow f_{\theta,\phi}(\mathcal{G}^{(t)})$
          else
              $\tilde{\boldsymbol{Y}}^{(t)} \leftarrow f_{\theta_{t-1},\phi_{t-1}}(\mathcal{G}^{(t)})$
          end if
          Use Eq.3 to compute $ST_{\mathcal{G}^{(0)},\mathcal{G}^{(t)}}$
          $\boldsymbol{H}_+^{(0)}, \boldsymbol{H}'^{(0)}_+ \boldsymbol{H}_+^{(t)}, \boldsymbol{H}'^{(t)}_+ \leftarrow M_{\mathcal{G}^{(0)}}(h_\phi(\mathcal{G}^{(0)})), M_{\mathcal{G}^{(0)}}(h_\phi(A(\mathcal{G}^{(0)}))), M_{\mathcal{G}^{(t)}}(h_\phi(\mathcal{G}^{(t)})), M_{\mathcal{G}^{(t)}}(h_\phi(A(\mathcal{G}^{(t)})))$
          $\mathcal{G}_-^{(0)}, \mathcal{G}_-^{(t)} \leftarrow$ randomly shuffle node features in $\mathcal{G}^{(0)}, \mathcal{G}^{(t)}$
          $\boldsymbol{H}_-^{(0)}, \boldsymbol{H}'^{(0)}_- \boldsymbol{H}_-^{(t)}, \boldsymbol{H}'^{(t)}_- \leftarrow M_{\mathcal{G}_-^{(0)}}(h_\phi(\mathcal{G}_-^{(0)})), M_{\mathcal{G}_-^{(0)}}(h_\phi(A(\mathcal{G}_-^{(0)}))), M_{\mathcal{G}_-^{(t)}}(h_\phi(\mathcal{G}_-^{(t)})), M_{\mathcal{G}_-^{(t)}}(h_\phi(A(\mathcal{G}_-^{(t)})))$
          Use Eq.4 to compute $CL_{\mathcal{G}^{(0)}}$ and $CL_{\mathcal{G}^{(t)}}$
          $(\theta, \phi, \phi') \leftarrow (\theta, \phi, \phi') - \eta \nabla_{\theta,\phi,\phi'}[ST_{\mathcal{G}^{(0)},\mathcal{G}^{(t)}}(\theta, \phi) + \alpha(CL_{\mathcal{G}^{(0)}}(\phi, \phi') + CL_{\mathcal{G}^{(t)}}(\phi, \phi'))]$
      end for
      $\theta_t, \phi_t \leftarrow \theta, \phi$
  end for
```

the high computation cost of graph diffusion on large graphs. Secondly, the final node representations of MVGRL for downstream classification are the sum of $h_\phi(G)$ and $h_{\phi'}(G)$. In our setting, we are looking for a graph encoder that pairs well with the prediction head and let the adapter carry it to the next stage. We let that graph encoder be h_ϕ.

Contrastive pre-training followed by fine-tuning has been shown to be effective for unsupervised domain adaptation (Shen et al., 2022). We do not take this two-stage approach because contrastive pre-training without downstream classification loss will flush the knowledge of the GNN gained in the previous stage, which is detrimental to gradual self-training. To get the best of both worlds, we optimize h_ϕ with respect to the contrastive loss and self-training loss simultaneously.

4.3 Overall Objective

The overall objective of Graph Contrastive Self-Training at the adaptation stage t is the following:

$$\theta_t, \phi_t, \phi'_t = \arg\min_{\theta,\phi,\phi'} ST_{\mathcal{G}^{(0)},\mathcal{G}^{(t)}}(\theta, \phi) + \alpha(CL_{\mathcal{G}^{(0)}}(\phi, \phi') + CL_{\mathcal{G}^{(t)}}(\phi, \phi')) \tag{5}$$

where α is a hyper-parameter that balances the self-training loss and the contrastive loss. θ is initialized with θ_{t-1}, and both ϕ and ϕ' are initialized with ϕ_{t-1}. θ_t and ϕ_t will be carried to stage $t+1$ for testing and adaptation. We denote GCST with fixed pseudo-labels as GCST-FPL and GCST with iteratively updated pseudo-labels as GCST-UPL. We list the overall procedure in Algorithm 1.

5 Experiments

5.1 Datasets

5.1.1 Synthetic Datasets

We generate two synthetic datasets with Stochastic Block Model (SBM) to simulate structural and node feature distribution shifts. The label distributions are fixed throughout stages, and the intensity of the input distribution shift from one stage to the next remains constant.

SBMB. SBMB contains a sequence of 20 graphs where the connection probabilities between blocks change over time. The task is a 3-way classification problem. Each graph contains 1000 nodes, with 600 of them belonging to class 1, 300 to class 2, and the remaining 100 to class 3. The node features are 10-dimensional and are obtained by sampling from three 10-dimensional Gaussian distributions with different means. Node feature distribution is fixed across all graphs. We group the graphs into 1 source stage (1-4) and 8 test stages (5-6/7-8/9-10/11-12/13-14/15-16/17-18/19-20).

Table 1: Overall results on four datasets. The metric for Elliptic is the F1 score, while accuracy is used for the rest. For each method, we compute the average performance over stages for each run. After conducting five runs with different random seeds, we calculate the average and standard deviation of the average performances over time. We underline the results that underperform the non-adaptive baseline, Fixed.

Method	**SBMB**	**SBMF**	**ogbn-arxiv**	**Elliptic**
Fixed	0.896±0.0040	0.894±0.0177	0.558±0.0050	0.452±0.0109
GST	0.909±0.0014	0.899±0.0103	0.579±0.0042	0.452±0.0043
CBST	0.899±0.0040	0.901±0.0071	0.560±0.0059	$\underline{0.396}$±0.0084
CRST	0.899±0.0045	$\underline{0.849}$±0.0069	0.576±0.0059	$\underline{0.441}$±0.0078
DANN	**0.928**±0.0030	$\underline{0.879}$±0.0078	$\underline{0.510}$±0.0187	0.455±0.0203
JAN	0.913±0.0073	**0.929**±0.0052	$\underline{0.506}$±0.0144	0.479±0.0074
Deep CORAL	0.905±0.0052	0.896±0.0064	$\underline{0.551}$±0.0040	0.497±0.0086
UDA-GCN	0.912±0.0035	$\underline{0.856}$±0.0069	$\underline{0.506}$±0.0082	$\underline{0.437}$±0.0104
DANE	0.927±0.0060	$\underline{0.870}$±0.0080	$\underline{0.534}$±0.0149	0.455±0.0156
GCST-FPL	0.918±0.0045	0.921±0.0123	**0.585**±0.0051	0.469±0.0189
GCST-UPL	0.911±0.0012	0.921±0.0044	**0.582**±0.0032	**0.511**±0.0086

SBMF. SBMF contains a sequence of 20 graphs in which the node feature distribution changes over time. The task is binary classification and the node features are generated from rotating mixtures of two 2D-Gaussians. Each graph consists of 1000 nodes, with 700 nodes belonging to class 1 and the remaining 300 belonging to class 2. The block matrix remains fixed across all graphs. The groupings of the graph sequence are the same as SBMB.

A more detailed description of the synthetic datasets can be found in Appendix A.1.

5.1.2 Real-world Datasets

We use two real-world graph datasets to test different adaptation algorithms in the wild.

ogbn-arxiv (Hu et al., 2020)[1]. ogbn-arxiv is an academic citation graph where each node represents a paper and each edge represents a citation link. The task is classifying the papers into one of 40 categories. We select the graph snapshot up till 2012 (denoted as $\sim$ 2012) for the source stage followed by 4 test stages: $\sim$ 2014, $\sim$ 2016, $\sim$ 2018, $\sim$ 2020.

Elliptic[2]. Elliptic is a collection of 49 transaction graphs obtained from the Bitcoin blockchain. Each graph consists of nodes that represent individual transactions and edges that indicate the flow of Bitcoins between two transactions. Each node is labeled as being created by either a "licit", "illicit", or "unknown" entity. The task is to detect illicit nodes. We follow Wu et al. (2022) and drop the first six snapshots due to extreme data imbalance. We group the remaining 43 graphs into 1 source stage (7-19) and 6 test stages (20-24/25-29/30-34/35-39/40-44/45-49).

We list the full dataset statistics in Appendix A.2.

5.2 Baselines

Fixed. We compare against fixing the source-trained model across all test stages without adaptation.

Domain-Invariant Feature Learning. We compare against Domain Adversarial Neural Network (DANN) (Ganin et al., 2016), Joint Adaptation Network (JAN) (Long et al., 2017), Deep CORAL (Sun & Saenko, 2016), Unsupervised Domain Adaptive Graph Convolutional Network (UDA-GCN) (Wu et al., 2020), and Domain Adaptive Network Embedding (DANE) (Song et al., 2022).

Self-Training. We compare against Class-Balanced Self-Training (CBST) (Zou et al., 2018), Confidence-Regularized Self-Training (CRST) (Zou et al., 2019), and Gradual Self-Training (GST) (Kumar et al., 2020).

For all methods, $f_{\theta,\phi}$ at stage t is initialized with θ_{t-1} and ϕ_{t-1}. The auxiliary models such as the bilinear model in GCST and the domain classifiers in DANN and UDA-GCN are randomly initialized at each stage.

[1]https://ogb.stanford.edu/docs/nodeprop/ogbn-arxiv

[2]https://www.kaggle.com/datasets/ellipticco/elliptic-data-set

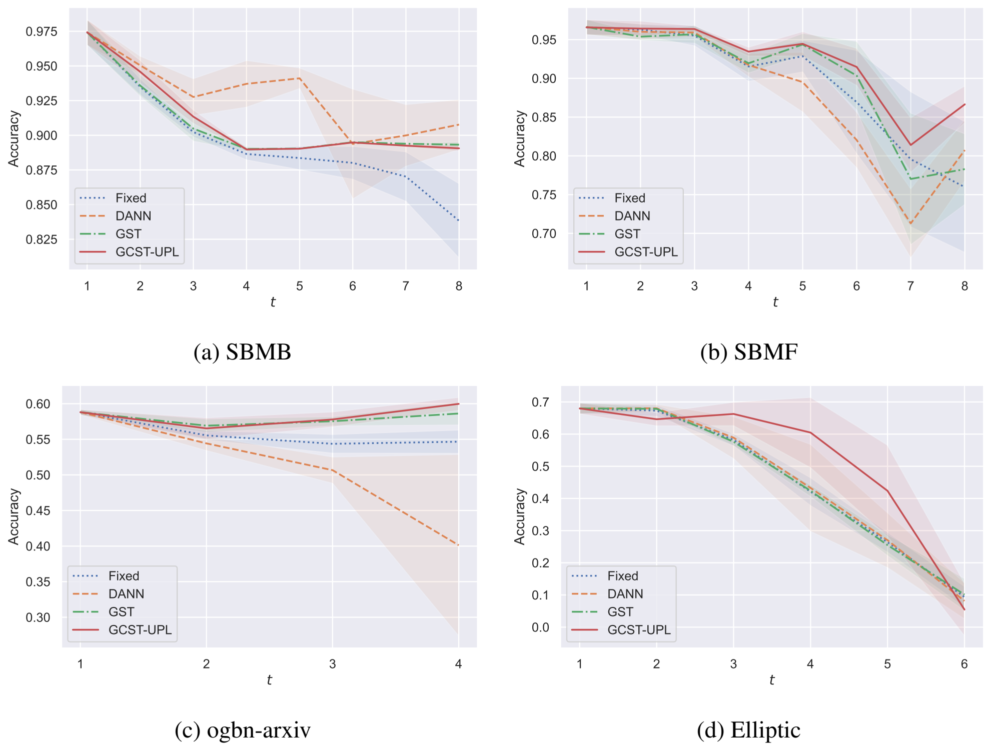


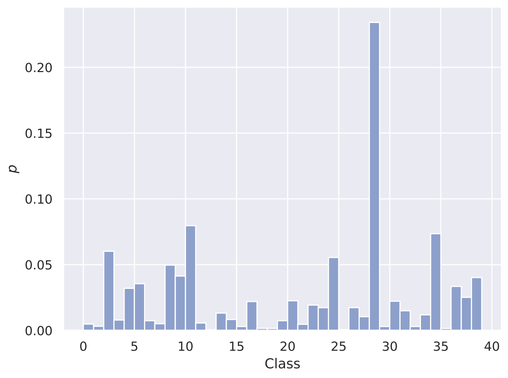
Figure 2: Stage-wise adaptation performances. To maintain readability, we present a representative method out of each class of methods (fixed, self-training, domain-invariant feature learning, and GCST). For each method, we plot its average performance and its range of plus or minus twice the standard deviation. While DANN is the best-performing method on SBMB, it underperforms Fixed on SBMF and ogbn-arxiv. GST shows a steady improvement over Fixed on SBMB, SBMF, and ogbn-arxiv, but performs similarly to Fixed on Elliptic. GCST-UPL emerges as the best-performing method most of the time on SBMF, ogbn-arxiv, and Elliptic, outperforming the other three methods. The stage-wise results of all the methods are included in Appendix A.5.

5.3 Experimental Settings

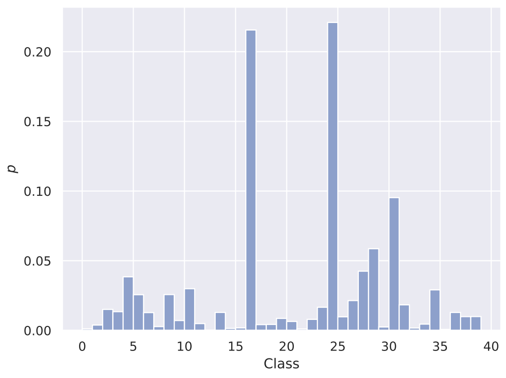
We use a two-layer GraphSAGE (Hamilton et al., 2017) coupled with a two-layer multi-layer perceptron for all experiments. At each stage, we split the data into training and validation sets based on time. To make full utilization of data, the validation set of the current stage will be included in the training set at the next stage. The validation set is used for model selection at each stage. Since labels are not available for adaptation, we use the information maximization (IM) validator on the predicted logits, which encourages diverse and confident predictions. For multi-graph datasets such as SBMB, SBMF, and Elliptic, the snapshot for test and adaptation is a disconnected graph and each component is a graph in the given grouping at that stage. For ogbn-arxiv, only the newly introduced nodes at each stage are tested and the k-hop subgraph of those nodes is considered the target graph where k is the number of layers in the graph encoder. We utilize the clustering and subgraph training method from ClusterGCN (Chiang et al., 2019) to scale up MVGRL to ogbn-arxiv. Finally, we use F1 score as the evaluation metric for Elliptic and accuracy for SBMB, SBMF, and ogbn-arxiv. We list the implementation details of the GNN and adapters in Appendix A.3 and A.4.

5.4 Results

The overall results are shown in Table 1 and the stage-wise performances of representative methods are shown in Fig. 2. By looking at the performance curve of Fixed in Fig. 2, we can observe the intensities of distribution shifts on different datasets. The accuracy drops on SBMB and ogbn-arxiv without adaptation are about 14% and 4%, respectively. In contrast, the accuracy drop on SBMF and the F1 score drop on Elliptic are about 20% and 0.6, presenting more challenging environments to the model. According to the summary presented in Table 1, GCST outperforms all baselines on the two real-world datasets, Elliptic and ogbn-arxiv, while ranking second or third on the synthetic datasets. GCST is the only class of methods that consistently improves upon Fixed while others occasionally underperform or do not improve.



(a) Before 2012



(b) 2019 and 2020

Figure 3: Label shift on ogbn-arxiv. Prior to 2012, papers labeled as information theory (cs.IT, class 28) were the majority and occupied more than 20% of the total papers. In 2019 and 2020, its proportion decreased to 6% while the majority became machine learning (cs.LG, class 24) and computer vision (cs.CV, class 16).

Table 2: Results on synthetic datasets with label shift. We show the averages and standard deviations over 5 runs.

Method	**SBMB-ys**	**SBMF-ys**
Fixed	0.798±0.0162	0.870±0.0375
GST	0.933±0.0147	0.913±0.0193
DANN	0.868±0.0065	0.896±0.0036
JAN	0.826±0.0143	<u>0.822</u>±0.0149
Deep CORAL	0.814±0.0083	<u>0.865</u>±0.0286
UDA-GCN	0.880±0.0249	0.885±0.0028
DANE	0.900±0.0126	0.872±0.0111
GCST-FPL	0.900±0.0329	0.938±0.0057
GCST-UPL	**0.955**±0.0110	**0.941**±0.0042

Among all self-training baselines, GST seems to be the most robust method, even though it does not improve the performance by a significant margin. Conversely, CBST and CRST appear to be relatively unstable, particularly on datasets with severe distribution shifts like SBMF and Elliptic. GCST outperforms these plain self-training methods on all four datasets. On Elliptic, the self-training baselines are struggling to beat Fixed, whereas GCST-UPL is the only method that achieves an F1 score greater than 0.5, surpassing Fixed by 0.06. GCST-FPL slightly outperforms GCST-UPL on SBMB and ogbn-arxiv, the two datasets with milder shifts, indicating that the fixed pseudo-labels assigned by the model trained in the previous stage are reliable under such circumstances. However, GCST-UPL is significantly better than GCST-FPL on Elliptic, with a 0.042 F1 score difference. This observation highlights the necessity of iteratively updating pseudo-labels under severe distribution shifts.

With respect to domain-invariant feature learning methods, it is worth noting that they suffer disastrous performance drops on ogbn-arxiv. DANN, JAN, Deep CORAL, UDA-GCN, and DANE underperform Fixed 4.8%, 5.2%, 0.7%, 5.2%, and 2.4% on ogbn-arxiv, respectively. This is likely due to the fragility of domain-invariant feature learning to label shift and the label shift nature of ogbn-arxiv which is shown in Fig. 3. We suspect that the success of some domain-invariant feature learning methods on SBMB and SBMF is because of the well-controlled label distributions. To test our hypothesis, we generate label-shifted versions of SBMB and SBMF, named SBMB-ys and SBMF-ys. For SBMB-ys, the ratio between the three classes gradually changes from 6:3:1 to 5:1:4. For SBMF-ys, the ratio gradually changes from 7:3 to 3:7. Configurations other than label distributions remain the same as the original synthetic datasets. We test the domain-invariant feature learning methods along with Fixed, GST, GCST-FPL, and GCST-UPL on SBMB-ys and SBMF-ys. The results are shown in Table 2. GCST-UPL is the best-performing method on both datasets, followed by GST and GCST-FPL while domain-invariant feature learning methods perform the worst. JAN is the most competitive method on SBMF, but its performance on SBMF-ys is 4.8% lower than Fixed, indicating its vulnerability to label shift. Conversely, GCST-UPL maintains its accuracy even when there is a shift in the label distribution, making it well-suited for many real-world scenarios where the label distributions are not fixed.

Table 3: Cosine similarities between source and target classifiers (S v.s. T), source and domain classifiers (S v.s. D), and target and domain classifiers (T v.s. D).

Dataset	**S v.s. T**	**S v.s. D**	**T v.s. D**
SBMF	0.840	0.049	0.033
Elliptic	0.653	-0.069	-0.070

Table 4: Ablation study on GCST-FPL (upper half) and GCST-UPL (lower half). The middle row is the result of not using pseudo-labels. The check mark indicates that the component associated with its respective column is being used. The averages and standard deviations of 5 runs are presented. The stage-wise results are included in Appendix A.6.

Components					**Datasets**			
Update Pseudo-Label	$\mathcal{G}^{(0)}$	CL	Gradual	$PL_{\mathcal{G}^{(t)}}$	**SBMB**	**SBMF**	**ogbn-arxiv**	**Elliptic**
-	-	✓	✓	✓	0.910$\pm$0.0015	0.884$\pm$0.0188	0.581$\pm$0.0055	0.455$\pm$0.0242
-	✓	-	✓	✓	0.908$\pm$0.0010	0.911$\pm$0.0125	0.582$\pm$0.0024	0.452$\pm$0.0143
-	✓	✓	-	✓	0.908$\pm$0.0023	0.912$\pm$0.0105	0.576$\pm$0.0054	0.455$\pm$0.0087
-	✓	✓	✓	✓	**0.918**$\pm$0.0045	**0.921**$\pm$0.0123	**0.585**$\pm$0.0051	**0.469**$\pm$0.0189
-	✓	✓	✓	-	0.904$\pm$0.0032	0.906$\pm$0.0066	0.506$\pm$0.0135	0.468$\pm$0.0044
✓	-	✓	✓	✓	0.847$\pm$0.0396	0.757$\pm$0.0366	0.396$\pm$0.0135	0.255$\pm$0.0513
✓	✓	-	✓	✓	0.907$\pm$0.0008	0.912$\pm$0.0087	0.576$\pm$0.0050	0.465$\pm$0.0207
✓	✓	✓	-	✓	**0.911**$\pm$0.0015	0.912$\pm$0.0097	0.577$\pm$0.0132	0.444$\pm$0.0105
✓	✓	✓	✓	✓	**0.911**$\pm$0.0012	**0.921**$\pm$0.0044	**0.582**$\pm$0.0032	**0.511**$\pm$0.0086

We further analyze whether the features learned by GCST disentangle domain and class information, which is an existing phenomenon in contrastive pre-training followed by fine-tuning (Shen et al., 2022). We train three linear classifiers on the feature space using logistic regression: (1) Source classifier, which is trained on source features; (2) Target classifier, which is trained on target features; (3) Domain classifier, which predicts whether a feature is from source or target domain. Suppose the domain and class information are perfectly disentangled and the source classifier is able to generalize to the target domain, then the source and target classifiers should be well aligned to each other and orthogonal to the domain classifier. Therefore, we compute the cosine similarities between source and target classifiers (S v.s. T), source and domain classifiers (S v.s. D), and target and domain (T v.s. D) classifiers on SBMF and Elliptic. In SBMF and Elliptic, we have selected $t = 6$ and $t = 4$ respectively as the target domains for feature analysis. The results are presented in Table 3. The cosine similarity between the source and target classifiers is high on both datasets (> 0.6), whereas the absolute cosine similarities of S v.s. D and T v.s. D are relatively low (< 0.1). These results indicate that GCST approximately disentangles domain and class information.

In summary, GCST is the most consistent and effective approach overall. It outperforms plain self-training baselines, such as GST, CBST, and CRST, by incorporating a contrastive learning objective that enhances the learning of representations. In contrast, the representations of plain self-training baselines solely rely on pseudo-labels, which tend to deteriorate over time. In cases where the label distribution remains unchanged, domain-invariant feature learning may be more effective than our method (e.g. DANN on SBMB and JAN on SBMF). However, in real-world networks where label distributions are not fixed, domain-invariant feature learning methods perform worse than our method. We further demonstrate that domain-invariant feature learning is fragile when dealing with the label-shifted synthetic datasets, SBMB-ys and SBMF-ys. In these cases, GCST significantly outperforms domain-invariant feature learning methods by a considerable margin.

5.5 Ablation Study and Analyses

In this section, we conduct an ablation analysis of GCST to better understand the impact of its components. GST ignores source data during adaptation and makes the model shift toward the target distribution. To examine how ignoring labeled source data like GST does affect the overall performance of GCST, we remove $L_{\mathcal{G}^{(0)}}$ and $CL_{\mathcal{G}^{(0)}}$ from the objective function. We also consider dropping contrastive loss, which would make GCST degenerate to self-training. In order to study the effect of gradual adaptation, we experiment with direct adaptation from the source domain to future domains instead of adapting from the previous domain. Contrastive pre-training followed by fine-tuning (Shen et al., 2022) does not include any pseudo-labeling or self-training techniques. Therefore, we are also interested in how the pseudo-label loss contributes to GCST. The ablation study results are presented in Table 4.

Ignoring labeled source data has a slight effect on GCST-FPL. However, it greatly reduces the performances of GCST-UPL. Its accuracy on ogbn-arxiv decreases from 58.2% to 39.6% and its F1 score on Elliptic drops from 0.511 to 0.255. The reason is that without the information provided by labeled data, the iterative updates of pseudo-labels will accumulate and amplify errors. Therefore, if we choose to update pseudo-labels, labeled source data must be included in the adaptation process.

The contrastive loss component consistently improves the performance of GCST across all datasets. For example, the F1 score of GCST-UPL without contrastive loss on Elliptic is 0.465, whereas adding the contrastive loss component can boost the F1 score to 0.511. This performance boost highlights the benefits of contrastive loss in aiding self-training, particularly when the distribution shift is significant, as in the case of Elliptic.

Gradual adaptation from the previous domain is also crucial to the effectiveness of GCST. If we directly adapt from the labeled source domain, the performances drop in almost all cases. The performance difference is especially significant under strong distribution shifts such as Elliptic, where the F1 score of GCST-UPL drops about 0.07. By adapting from the previous domain, the model parameters are initialized close to optimality at the current domain, which can yield more accurate pseudo-labels at the start of the iteration and lead to better results than direct adaptation.

The middle row of Table 4 shows that removing the pseudo-label loss negatively affects GCST. The effect on ogbn-arxiv is the most significant compared to other datasets, showing a 7.6% accuracy drop. This is likely due to the number of nodes at each stage rapidly increasing in ogbn-arxiv but remaining at the same level in the other datasets. As a result, ignoring pseudo-labels on these vast unlabeled data would lead to more performance degradation from the complete GCST.

In conclusion, our ablation study demonstrates that each component in GCST is essential to its strong performance. The complete versions of GCST-FPL and GCST-UPL are the top-performing methods in all datasets, with GCST-UPL being more stable against large distribution shifts. Therefore, we recommend using GCST-UPL instead of GCST-FPL if only one method is to be chosen.

6 Conclusion

This paper addresses the challenge of adapting a graph neural network (GNN) to evolving graphs in the absence of labels. Our proposed solution, graph contrastive self-training (GCST), combines contrastive learning and self-training to enable the gradual adaptation of GNNs to unlabeled, evolving graphs. Our experimental results demonstrate the superiority and stability of our approach compared to domain-invariant feature learning and plain self-training methods. Additionally, we conduct an ablation study to investigate the individual roles of labeled source graph, contrastive loss, gradual adaptation, and pseudo-label loss in GCST. Our findings suggest that all components are critical to the strong performance of GCST, and omitting any of them would result in degraded performance.

7 Acknowledgement

This research is supported by the Schlumberger Centennial Chair Endowment awarded to Joydeep Ghosh.

References

Samira Abnar, Rianne van den Berg, Golnaz Ghiasi, Mostafa Dehghani, Nal Kalchbrenner, and Hanie Sedghi. Gradual domain adaptation in the wild:when intermediate distributions are absent. *arXiv preprint arXiv:2106.06080*, 2021.

David Berthelot, Rebecca Roelofs, Kihyuk Sohn, Nicholas Carlini, and Alexey Kurakin. Adamatch: A unified approach to semi-supervised learning and domain adaptation. In *International Conference on Learning Representations*, 2022.

Andreea Bobu, Eric Tzeng, Judy Hoffman, and Trevor Darrell. Adapting to continuously shifting domains. *International Conference on Learning Representations Workshop*, 2018.

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.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

Quanyu Dai, Xiao-Ming Wu, Jiaren Xiao, Xiao Shen, and Dan Wang. Graph transfer learning via adversarial domain adaptation with graph convolution. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. Credit card fraud detection and concept-drift adaptation with delayed supervised information. In *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015.

Geoff French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. In *International Conference on Learning Representations*, 2018.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 2016.

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

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Yifei He, Haoxiang Wang, and Han Zhao. Generative gradual domain adaptation with optimal transport. *Principles of Distribution Shift (PODS) Workshop, ICML*, 2022.

Judy Hoffman, Trevor Darrell, and Kate Saenko. Continuous manifold based adaptation for evolving visual domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

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. In *Advances in Neural Information Processing Systems*, 2020.

Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding self-training for gradual domain adaptation. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013.

Hong Liu, Mingsheng Long, Jianmin Wang, and Yu Wang. Learning to adapt to evolving domains. In *Advances in Neural Information Processing Systems*, 2020.

Hong Liu, Jianmin Wang, and Mingsheng Long. Cycle self-training for domain adaptation. In *Advances in Neural Information Processing Systems*, 2021.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, 2018.

Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

Viraj Prabhu, Shivam Khare, Deeksha Kartik, and Judy Hoffman. Sentry: Selective entropy optimization via committee consistency for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Mohammad Rostami. Lifelong domain adaptation via consolidated internal distribution. In *Advances in Neural Information Processing Systems*, 2021.

Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Kendrick Shen, Robbie M Jones, Ananya Kumar, Sang Michael Xie, Jeff Z. Haochen, Tengyu Ma, and Percy Liang. Connect, not collapse: Explaining contrastive learning for unsupervised domain adaptation. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Guojie Song, Yizhou Zhang, Lingjun Xu, and Haibing Lu. Domain adaptive network embedding. *IEEE Transactions on Big Data*, 2022.

Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision – ECCV 2016 Workshops*, 2016.

Shixiang Tang, Peng Su, Dapeng Chen, and Wanli Ouyang. Gradient regularized contrastive learning for continual domain adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Niek Tax, Kees Jan de Vries, Mathijs de Jong, Nikoleta Dosoula, Bram van den Akker, Jon Smith, Olivier Thuong, and Lucas Bernardi. Machine learning for fraud detection in e-commerce: A research agenda. In *Deployable Machine Learning for Security Defense*, 2021.

Haoxiang Wang, Bo Li, and Han Zhao. Understanding gradual domain adaptation: Improved analysis, optimal path and beyond. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. Unsupervised domain adaptive graph convolutional networks. In *Proceedings of The Web Conference*, 2020.

Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. In *International Conference on Learning Representations*, 2022.

Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Incremental adversarial domain adaptation for continually changing environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. Graph domain adaptation via theory-grounded spectral regularization. In *The Eleventh International Conference on Learning Representations*, 2023.

Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Yang Zou, Zhiding Yu, B.V.K. Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Yang Zou, Zhiding Yu, Xiaofeng Liu, B.V.K. Vijaya Kumar, and Jinsong Wang. Confidence regularized self-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

A Additional Details for Section 5

A.1 Synthetic Datasets

SBMB. SBMB contains a sequence of 20 graphs for 3-way node classification where the connection probabilities between each community change over time. Each graph is generated using a Stochastic Block Model specified by a block matrix. The block matrix of the first graph is $[[0.7, 0.2, 0.1], [0.2, 0.8, 0.3], [0.1, 0.3, 0.9]]$. The block matrix of the last graph is $[[0.8, 0.4, 0.3], [0.4, 0.6, 0.2], [0.3, 0.2, 0.5]]$. The block matrices of the graphs in between are the linear interpolations of the two block matrices. Node features of three classes are sampled from three 10-dimensional Gaussian distributions with means at each dimension being -1, 0, and 1. We introduce random label flipping with a probability 0.01 to increase the difficulty.

SBMF. SBMF contains a sequence of 20 graphs for binary node classification where the node feature distributions change over time. Each graph is generated using a Stochastic Block Model specified by the same block matrix, $[[0.5, 0.4], [0.4, 0.6]]$. In the first graph, the node features of the two classes are sampled from two Normal distributions centered at $[\cos 45°, \sin 45°]$ and $[\cos 225°, \sin 225°]$, respectively. The centers gradually rotate to $[\cos 315°, \sin 315°]$ and $[\cos 135°, \sin 135°]$ by the end of the graph sequence. We also introduce random label flipping with a probability 0.01 to this dataset.

A.2 Dataset Statistics

Table 5: Dataset statistics of **SBMB** at each stage.

Stage	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
# Nodes									
Test		2000	2000	2000	2000	2000	2000	2000	2000
Train	3000	2000	2000	2000	2000	2000	2000	2000	
Val	1000	1000	1000	1000	1000	1000	1000	1000	
# Edges									
Test		924860	946420	968666	989158	1009028	1028616	1051078	1070456
Train	1333350	915096	934040	959016	979032	999138	1019076	1039238	
Val	455090	464854	477234	486884	497010	506900	516440	528280	

Table 6: Dataset statistics of **SBMF** at each stage.

Stage	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
# Nodes									
Test		2000	2000	2000	2000	2000	2000	2000	2000
Train	3000	2000	2000	2000	2000	2000	2000	2000	
Val	1000	1000	1000	1000	1000	1000	1000	1000	
# Edges									
Test		933498	932878	934204	934196	933420	931784	934610	932456
Train	1401068	933802	931784	934260	934836	934096	932068	933046	
Val	466738	466434	467528	467472	466832	466156	465872	467436	

Table 7: Dataset statistics of **ogbn-arxiv** at each stage. The dataset statistics of ogbn-arxiv are listed differently from the other three datasets. ogbn-arxiv is originally a single graph dataset. At each time step, we take a snapshot of the graph at that time. Since the citation graph is ever-growing, the accumulated nodes and edges are monotonically increasing over time as shown in the first two rows. At each time step, we only need to test and adapt to the newly introduced nodes (subgraph). We list the statistics of those nodes in the lower three rows.

Stage	**0**	**1**	**2**	**3**	**4**
# Accumulated Nodes	23836	41125	69499	120740	169343
# Accumulated Edges	91468	198802	464838	1230830	2315598
# Test Nodes		17289	28374	51241	48603
# Train Nodes	17401	14570	21189	37781	
# Val Nodes	6435	9154	16399	29799	

Table 8: Dataset statistics of **Elliptic** at each stage.

Stage	**0**	**1**	**2**	**3**	**4**	**5**	**6**
# Nodes							
Test		22479	11854	15461	20857	27001	19646
Train	45120	19204	14683	15752	20843	22614	
Val	5482	8757	5928	5637	5651	10038	
# Edges							
Test		25436	12735	16620	24021	31333	22158
Train	51333	21681	16185	16820	24071	25840	
Val	5953	9708	6258	6058	6008	11501	

A.3 Model Architecture

We use a two-layer GraphSAGE coupled with a two-layer multi-layer perceptron (MLP) for all experiments. Exponential Linear Unit (ELU) activation is placed after all layers except the output layer. Two dropout layers with a rate of 0.2 are placed in between the two SAGE convolution layers and the two linear layers, right after the ELU activations. The dimensional hyper-parameters vary across datasets. We list the configurations below.

A.3.1 SBMB and SBMF

- GraphSAGE
  - Hidden layer dimension: 64
  - Embedding dimension: 64
- MLP
  - Hidden layer dimension: 16

A.3.2 ogbn-arxiv

- GraphSAGE
  - Hidden layer dimension: 128
  - Embedding dimension: 256
- MLP
  - Hidden layer dimension: 64

A.3.3 Elliptic

- GraphSAGE
  - Hidden layer dimension: 128
  - Embedding dimension: 128
- MLP
  - Hidden layer dimension: 32

A.4 Adapter Implementation and Hyper-parameters

GCST. We set the node dropout rate to 0.2. On ogbn-arxiv, we use the clustering and subgraph training technique (Chiang et al., 2019). We partition the current graph snapshot into 32 subgraphs and sample 16 of them in each mini-batch. The candidate contrastive tradeoff coefficients α are $[0.01, 0.05, 0.1, 0.5, 1]$ and the candidate threshold values τ are $[0.1, 0.3, 0.5, 0.7, 0.9]$.

Self-Training. The candidate threshold values τ for GST are $[0.1, 0.3, 0.5, 0.7, 0.9]$. CBST and CRST progressively increase the threshold value from 0.2 to 0.5 with an 0.05 increment each time. We follow the recommendation by Zou et al. (2019) and use the KL-divergence between the uniform distribution and softmax output as the confidence regularizer in CRST. The regularizer tradeoff coefficient is set to 1 on ogbn-arxiv and 0.5 on SBMB, SBMF, and Elliptic.

Domain-Invariant Feature Learning Methods. In the implementation of DANN, we use a warm-up scheduler for the coefficient of the reversed gradient. The exact warm-up schedule is $\lambda(\frac{2}{1+\exp(-10p)} - 1)$, where p is the current epoch divided by the total number of epochs and λ is a hyper-parameter which specifies the maximum coefficient of the reversed gradients. The candidate values for λ are $[0.1, 0.3, 0.5, 0.7, 0.9]$ and the total number of epochs is 500. Regarding JAN and Deep CORAL, we set the candidate joint MMD and correlation tradeoff coefficients to be $[0.5, 1, 5]$. We follow the original implementation of UDA-GCN, setting the warm-up schedule of the reversed gradient coefficient to $\min(p, 0.05)$ and the coefficient of entropy loss to $0.01p$. For DANE, the candidate coefficient values for the adversarial regularizer and the supervised loss are $[0.1, 1]$ and $[1, 10]$, respectively. The total number of epochs for both UDA-GCN and DANE is 500.

By the end of each stage, the IM validator evaluates the information maximization score of the outcomes of the adapter hyper-parameters and selects the best model to be carried to the next stage. For all methods, we use the Adam optimizer with a learning rate 0.001 for adaptation. The random seeds for the five runs are 42, 10, 45, 5, and 3.

A.5 Stage-wise Performances of All Methods

Table 9: Stage-wise results on **SBMB (first half)**. All methods have the same performance level in stage 1 and then gradually diverge to a maximum of 5.1% accuracy difference in stage 4.

Stage	**1**	**2**	**3**	**4**
Method				
Fixed	0.974±0.0047	0.934±0.0030	0.902±0.0018	0.886±0.0022
GST	0.974±0.0047	0.935±0.0040	0.904±0.0046	0.890±0.0010
CBST	0.974±0.0047	0.929±0.0033	0.903±0.0031	0.889±0.0045
CRST	0.974±0.0047	0.926±0.0085	0.903±0.0034	0.890±0.0037
DANN	0.974±0.0047	0.950±0.0034	0.927±0.0067	0.937±0.0085
JAN	0.974±0.0047	0.937±0.0073	0.914±0.0119	0.907±0.0183
Deep CORAL	0.974±0.0047	0.929±0.0122	0.911±0.0095	0.892±0.0052
UDA-GCN	0.974±0.0047	0.934±0.0024	0.913±0.0063	0.906±0.0088
DANE	0.974±0.0047	0.951±0.0125	0.932±0.0064	0.906±0.0222
GCST-FPL	0.974±0.0047	0.949±0.0110	0.925±0.0043	0.907±0.0034
GCST-UPL	0.974±0.0047	0.945±0.0042	0.913±0.0025	0.889±0.0008

Table 10: Stage-wise results on **SBMB (second half)**. The 5.1% accuracy difference between DANN and Fixed enlarges to 6.9% by the final stage.

Stage	**5**	**6**	**7**	**8**
Method				
Fixed	0.883±0.0043	0.880±0.0060	0.870±0.0090	0.838±0.0013
GST	0.890±0.0002	0.895±0.0000	0.893±0.0002	0.893±0.0011
CBST	0.883±0.0049	0.882±0.0051	0.875±0.0084	0.859±0.0125
CRST	0.882±0.0056	0.882±0.0050	0.877±0.0061	0.857±0.0138
DANN	0.941±0.0038	0.893±0.0199	0.899±0.0112	0.907±0.0091
JAN	0.900±0.0176	0.909±0.0074	0.895±0.0138	0.871±0.0219
Deep CORAL	0.889±0.0048	0.889±0.0029	0.883±0.0061	0.873±0.0096
UDA-GCN	0.891±0.0115	0.907±0.0086	0.885±0.0032	0.887±0.0060
DANE	0.925±0.0105	0.912±0.0318	0.900±0.0121	0.918±0.0166
GCST-FPL	0.903±0.0084	0.903±0.0045	0.897±0.0063	0.890±0.0041
GCST-UPL	0.890±0.0002	0.894±0.0002	0.892±0.0006	0.890±0.0011

Table 11: Stage-wise results on **SBMF (first half)**. SBMF exhibits a greater distribution shift than SBMB. By stage 4, the best-performing methods, GCST-FPL and GCST-UPL, already outperform the worst method by roughly 7.7%.

Stage	**1**	**2**	**3**	**4**
Method				
Fixed	0.965±0.0050	0.962±0.0037	0.955±0.0067	0.915±0.0097
GST	0.966±0.0050	0.953±0.0025	0.956±0.0059	0.919±0.0064
CBST	0.968±0.0044	0.963±0.0038	0.950±0.0058	0.909±0.0114
CRST	0.968±0.0044	0.934±0.0237	0.913±0.0070	0.857±0.0173
DANN	0.966±0.0050	0.960±0.0052	0.959±0.0030	0.917±0.0077
JAN	0.965±0.0050	0.959±0.0032	0.953±0.0041	0.923±0.0029
Deep CORAL	0.965±0.0050	0.960±0.0039	0.952±0.0063	0.919±0.0038
UDA-GCN	0.967±0.0048	0.923±0.0279	0.923±0.0396	0.889±0.0194
DANE	0.967±0.0048	0.961±0.0060	0.949±0.0172	0.907±0.0051
GCST-FPL	0.965±0.0050	0.966±0.0039	0.960±0.0041	0.935±0.0055
GCST-UPL	0.965±0.0050	0.964±0.0050	0.963±0.0021	0.934±0.0030

Table 12: Stage-wise results on **SBMF (second half)**. Most methods experience a significant accuracy drop at stage 7, while some of them are able to recover at the final stage.

Stage	**5**	**6**	**7**	**8**
Method				
Fixed	0.928±0.0103	0.869±0.0335	0.795±0.0437	0.759±0.0423
GST	0.943±0.0073	0.903±0.0223	0.770±0.0426	0.786±0.0229
CBST	0.934±0.0033	0.905±0.0090	0.778±0.0354	0.802±0.0274
CRST	0.842±0.0260	0.780±0.0130	0.761±0.0212	0.737±0.0148
DANN	0.895±0.0194	0.820±0.0181	0.712±0.0225	0.807±0.0200
JAN	0.939±0.0153	0.924±0.0121	0.878±0.0115	0.887±0.0108
Deep CORAL	0.931±0.0104	0.877±0.0180	0.806±0.0291	0.754±0.0180
UDA-GCN	0.915±0.0174	0.889±0.0291	0.606±0.0216	0.733±0.0411
DANE	0.900±0.0186	0.800±0.0407	0.622±0.0297	0.851±0.0312
GCST-FPL	0.943±0.0209	0.909±0.0214	0.815±0.0428	0.872±0.0134
GCST-UPL	0.944±0.0080	0.914±0.0121	0.814±0.0178	0.866±0.0120

Table 13: Stage-wise results on **ogbn-arxiv**. In the final stage, GCST-FPL achieves the highest accuracy, followed by GCST-UPL, with both methods having accuracies of around 60%. Conversely, DANN and JAN exhibit accuracies of around 40% in the final stage, indicating their vulnerability to label shift.

Stage	**1**	**2**	**3**	**4**
Method				
Fixed	$0.588_{\pm 0.0021}$	$0.555_{\pm 0.0034}$	$0.543_{\pm 0.0067}$	$0.546_{\pm 0.0083}$
GST	$0.588_{\pm 0.0021}$	$0.569_{\pm 0.0049}$	$0.575_{\pm 0.0034}$	$0.586_{\pm 0.0082}$
CBST	$0.588_{\pm 0.0021}$	$0.561_{\pm 0.0059}$	$0.551_{\pm 0.0100}$	$0.540_{\pm 0.0188}$
CRST	$0.588_{\pm 0.0021}$	$0.554_{\pm 0.0092}$	$0.568_{\pm 0.0051}$	$0.593_{\pm 0.0241}$
DANN	$0.588_{\pm 0.0021}$	$0.544_{\pm 0.0052}$	$0.506_{\pm 0.0094}$	$0.401_{\pm 0.0639}$
JAN	$0.588_{\pm 0.0021}$	$0.556_{\pm 0.0039}$	$0.516_{\pm 0.0132}$	$0.364_{\pm 0.0552}$
Deep CORAL	$0.588_{\pm 0.0021}$	$0.555_{\pm 0.0039}$	$0.540_{\pm 0.0058}$	$0.522_{\pm 0.0179}$
UDA-GCN	$0.588_{\pm 0.0021}$	$0.482_{\pm 0.0130}$	$0.475_{\pm 0.0097}$	$0.481_{\pm 0.0156}$
DANE	$0.588_{\pm 0.0021}$	$0.512_{\pm 0.0279}$	$0.523_{\pm 0.0167}$	$0.512_{\pm 0.0241}$
GCST-FPL	$0.588_{\pm 0.0021}$	$0.570_{\pm 0.0048}$	$0.579_{\pm 0.0050}$	$0.602_{\pm 0.0118}$
GCST-UPL	$0.588_{\pm 0.0021}$	$0.565_{\pm 0.0076}$	$0.577_{\pm 0.0054}$	$0.599_{\pm 0.0047}$

Table 14: Stage-wise results on **Elliptic**. Among all the methods, only GCST-UPL is capable of achieving an F1 score of 0.6 until stage 4, whereas most other methods have an F1 score lower than 0.5.

Stage	**1**	**2**	**3**	**4**	**5**	**6**
Method						
Fixed	$0.679_{\pm 0.0088}$	$0.672_{\pm 0.0091}$	$0.582_{\pm 0.0099}$	$0.421_{\pm 0.0220}$	$0.263_{\pm 0.0158}$	$0.093_{\pm 0.0167}$
GST	$0.679_{\pm 0.0088}$	$0.678_{\pm 0.0031}$	$0.576_{\pm 0.0066}$	$0.423_{\pm 0.0093}$	$0.255_{\pm 0.0168}$	$0.101_{\pm 0.0253}$
CBST	$0.679_{\pm 0.0088}$	$0.619_{\pm 0.0109}$	$0.446_{\pm 0.0131}$	$0.333_{\pm 0.0316}$	$0.203_{\pm 0.0137}$	$0.096_{\pm 0.0115}$
CRST	$0.679_{\pm 0.0088}$	$0.670_{\pm 0.0052}$	$0.541_{\pm 0.0370}$	$0.411_{\pm 0.0224}$	$0.238_{\pm 0.0126}$	$0.109_{\pm 0.0166}$
DANN	$0.679_{\pm 0.0088}$	$0.680_{\pm 0.0066}$	$0.588_{\pm 0.0330}$	$0.432_{\pm 0.0680}$	$0.270_{\pm 0.0430}$	$0.081_{\pm 0.0283}$
JAN	$0.679_{\pm 0.0088}$	$0.642_{\pm 0.0110}$	$0.625_{\pm 0.0114}$	$0.470_{\pm 0.0284}$	$0.426_{\pm 0.0286}$	$0.034_{\pm 0.0192}$
Deep CORAL	$0.679_{\pm 0.0088}$	$0.648_{\pm 0.0120}$	$0.614_{\pm 0.0063}$	$0.544_{\pm 0.0254}$	$0.455_{\pm 0.0201}$	$0.041_{\pm 0.0239}$
UDA-GCN	$0.679_{\pm 0.0088}$	$0.657_{\pm 0.0141}$	$0.528_{\pm 0.0164}$	$0.413_{\pm 0.0214}$	$0.242_{\pm 0.0212}$	$0.102_{\pm 0.0263}$
DANE	$0.679_{\pm 0.0088}$	$0.576_{\pm 0.0360}$	$0.553_{\pm 0.0715}$	$0.541_{\pm 0.0298}$	$0.342_{\pm 0.0525}$	$0.038_{\pm 0.0144}$
GCST-FPL	$0.679_{\pm 0.0088}$	$0.677_{\pm 0.0078}$	$0.597_{\pm 0.0145}$	$0.458_{\pm 0.0411}$	$0.284_{\pm 0.0233}$	$0.121_{\pm 0.0318}$
GCST-UPL	$0.679_{\pm 0.0088}$	$0.646_{\pm 0.0107}$	$0.662_{\pm 0.0186}$	$0.604_{\pm 0.0550}$	$0.423_{\pm 0.0714}$	$0.054_{\pm 0.0414}$

A.6 Stage-wise Performances of the Ablation Study

Table 15: Stage-wise results of the ablation study on **SBMB (first half)**. In stage 4, the performance of GCST-UPL w/o $\mathcal{G}^{(0)}$ exhibits a significant decrease in accuracy when compared to GCST-UPL with a difference of 6.1$\%$. In contrast, other variations of GCST do not show any notable difference in their performance. This highlights the importance of using the labeled source graph when updating pseudo-labels.

Stage	**1**	**2**	**3**	**4**
Method				
GCST-FPL w/o $\mathcal{G}^{(0)}$	0.974±0.0047	0.938±0.0046	0.907±0.0051	0.890±0.0019
GCST-FPL w/o CL	0.974±0.0047	0.936±0.0026	0.906±0.0026	0.890±0.0017
GCST-FPL w/o gradual	0.974±0.0047	0.949±0.0110	0.913±0.0021	0.891±0.0024
GCST-FPL	0.974±0.0047	0.949±0.0110	0.925±0.0043	0.907±0.0034
GCST w/o $PL_{\mathcal{G}^{(t)}}$	0.974±0.0047	0.937±0.0088	0.904±0.0038	0.887±0.0008
GCST-UPL w/o $\mathcal{G}^{(0)}$	0.974±0.0047	0.952±0.0065	0.852±0.0794	0.828±0.0787
GCST-UPL w/o CL	0.974±0.0047	0.934±0.0018	0.903±0.0019	0.887±0.0018
GCST-UPL w/o gradual	0.974±0.0047	0.945±0.0033	0.911±0.0074	0.888±0.0007
GCST-UPL	0.974±0.0047	0.945±0.0042	0.913±0.0025	0.889±0.0008

Table 16: Stage-wise results of the ablation study on **SBMB (second half)**. The accuracy of GCST-UPL w/o $\mathcal{G}^{(0)}$ continues to drop to 78.7% in the final stage. GCST w/o $PL_{\mathcal{G}^{(t)}}$ also demonstrates a 3.4% accuracy drop compared to the full GCST. This result emphasizes the importance of pseudo-labeling in GCST.

Stage	**5**	**6**	**7**	**8**
Method				
GCST-FPL w/o $\mathcal{G}^{(0)}$	0.890±0.0004	0.895±0.0000	0.893±0.0002	0.892±0.0012
GCST-FPL w/o CL	0.889±0.0014	0.893±0.0021	0.891±0.0024	0.888±0.0039
GCST-FPL w/o gradual	0.890±0.0006	0.892±0.0017	0.889±0.0025	0.869±0.0145
GCST-FPL	0.903±0.0084	0.903±0.0045	0.897±0.0063	0.890±0.0041
GCST w/o $PL_{\mathcal{G}^{(t)}}$	0.887±0.0023	0.891±0.0035	0.886±0.0048	0.866±0.0157
GCST-UPL w/o $\mathcal{G}^{(0)}$	0.801±0.0511	0.790±0.0582	0.790±0.0579	0.787±0.0594
GCST-UPL w/o CL	0.888±0.0025	0.892±0.0021	0.890±0.0018	0.887±0.0032
GCST-UPL w/o gradual	0.890±0.0005	0.895±0.0000	0.892±0.0007	0.890±0.0029
GCST-UPL	0.890±0.0002	0.894±0.0002	0.892±0.0006	0.890±0.0011

Table 17: Stage-wise results of the ablation study on **SBMF (first half)**. In this synthetic dataset, which exhibits a more severe distribution shift than SBMB, the use of the labeled source graph is found to be crucial for the performance of GCST-FPL as well. By stage 4, it is observed that GCST-FPL w/o $\mathcal{G}^{(0)}$ demonstrates a 2.6% lower accuracy than GCST-FPL.

Stage	**1**	**2**	**3**	**4**
Method				
GCST-FPL w/o $\mathcal{G}^{(0)}$	0.968±0.0044	0.953±0.0070	0.959±0.0062	0.909±0.0213
GCST-FPL w/o CL	0.967±0.0048	0.963±0.0037	0.958±0.0033	0.927±0.0057
GCST-FPL w/o gradual	0.974±0.0047	0.949±0.0110	0.913±0.0021	0.891±0.0024
GCST-FPL	0.965±0.0050	0.966±0.0039	0.960±0.0041	0.935±0.0055
GCST w/o $PL_{\mathcal{G}^{(t)}}$	0.967±0.0048	0.966±0.0031	0.955±0.0043	0.921±0.0092
GCST-UPL w/o $\mathcal{G}^{(0)}$	0.968±0.0044	0.847±0.0964	0.760±0.0515	0.701±0.0220
GCST-UPL w/o CL	0.967±0.0048	0.962±0.0052	0.957±0.0000	0.923±0.0080
GCST-UPL w/o gradual	0.974±0.0047	0.945±0.0033	0.911±0.0074	0.888±0.0007
GCST-UPL	0.965±0.0050	0.964±0.0050	0.963±0.0021	0.934±0.0030

Table 18: Stage-wise results of the ablation study on **SBMF (second half)**. The accuracies of the versions of not using the labeled source graph continue to drop below 80%. Not using contrastive loss leads to a 5% drop from the full version, highlighting the importance of contrastive learning under larger distribution shifts.

Stage	**5**	**6**	**7**	**8**
Method				
GCST-FPL w/o $\mathcal{G}^{(0)}$	0.941±0.0131	0.868±0.0435	0.708±0.0436	0.766±0.0425
GCST-FPL w/o CL	0.941±0.0101	0.904±0.0224	0.806±0.0440	0.822±0.0245
GCST-FPL w/o gradual	0.890±0.0006	0.892±0.0017	0.889±0.0025	0.869±0.0145
GCST-FPL	0.943±0.0209	0.909±0.0214	0.815±0.0428	0.872±0.0134
GCST w/o $PL_{\mathcal{G}^{(t)}}$	0.939±0.0063	0.902±0.0143	0.785±0.0344	0.814±0.0116
GCST-UPL w/o $\mathcal{G}^{(0)}$	0.747±0.0398	0.659±0.0264	0.660±0.0468	0.714±0.0746
GCST-UPL w/o CL	0.943±0.0061	0.911±0.0073	0.823±0.0486	0.810±0.0226
GCST-UPL w/o gradual	0.890±0.0005	0.895±0.0000	0.892±0.0007	0.890±0.0029
GCST-UPL	0.944±0.0080	0.914±0.0121	0.814±0.0178	0.866±0.0120

Table 19: Stage-wise results of the ablation study on **ogbn-arxiv**. In the final stage, GCST w/o $PL_{\mathcal{G}^{(t)}}$ demonstrates the second lowest accuracy, primarily due to its disregard of a vast number of unlabeled nodes in the later stages.

Stage	**1**	**2**	**3**	**4**
Method				
GCST-FPL w/o $\mathcal{G}^{(0)}$	0.588±0.0021	0.573±0.0055	0.577±0.0063	0.585±0.0095
GCST-FPL w/o CL	0.588±0.0021	0.566±0.0011	0.575±0.0046	0.600±0.0038
GCST-FPL w/o gradual	0.588±0.0021	0.570±0.0039	0.572±0.0051	0.575±0.0132
GCST-FPL	0.588±0.0021	0.570±0.0048	0.579±0.0050	0.602±0.0118
GCST w/o $PL_{\mathcal{G}^{(t)}}$	0.588±0.0021	0.466±0.0311	0.496±0.0136	0.483±0.0151
GCST-UPL w/o $\mathcal{G}^{(0)}$	0.588±0.0021	0.382±0.0347	0.294±0.0139	0.318±0.0111
GCST-UPL w/o CL	0.588±0.0021	0.564±0.0072	0.571±0.0060	0.580±0.0086
GCST-UPL w/o gradual	0.588±0.0021	0.569±0.0031	0.570±0.0078	0.583±0.0497
GCST-UPL	0.588±0.0021	0.565±0.0076	0.577±0.0054	0.599±0.0047

Table 20: Stage-wise results of the ablation study on **Elliptic**. Until stage 4, GCST-UPL is the only method that achieves an F1 score greater than 0.6. However, this advantage would be significantly diminished if either the contrastive loss or the labeled source graph were excluded. Specifically, the exclusion of the contrastive loss results in an F1 score of 0.452, while the exclusion of the labeled source graph leads to an F1 score of 0.148.

Stage	**1**	**2**	**3**	**4**	**5**	**6**
Method						
GCST-FPL w/o $\mathcal{G}^{(0)}$	0.679±0.0088	0.674±0.0082	0.575±0.0268	0.427±0.0671	0.266±0.0475	0.107±0.0218
GCST-FPL w/o CL	0.679±0.0088	0.672±0.0053	0.579±0.0120	0.425±0.0368	0.259±0.0202	0.098±0.0260
GCST-FPL w/o gradual	0.679±0.0088	0.673±0.0049	0.579±0.0137	0.440±0.0137	0.262±0.0107	0.098±0.0190
GCST-FPL	0.679±0.0088	0.677±0.0078	0.597±0.0145	0.458±0.0411	0.284±0.0233	0.121±0.0318
GCST w/o $PL_{\mathcal{G}^{(t)}}$	0.679±0.0088	0.673±0.0094	0.591±0.0161	0.482±0.0101	0.298±0.0181	0.082±0.0124
GCST-UPL w/o $\mathcal{G}^{(0)}$	0.679±0.0088	0.269±0.1498	0.267±0.0855	0.148±0.0993	0.124±0.0578	0.045±0.0136
GCST-UPL w/o CL	0.679±0.0088	0.659±0.0154	0.614±0.0227	0.452±0.0497	0.330±0.0548	0.053±0.0555
GCST-UPL w/o gradual	0.679±0.0088	0.638±0.0170	0.595±0.0359	0.357±0.0344	0.229±0.0198	0.163±0.0518
GCST-UPL	0.679±0.0088	0.646±0.0107	0.662±0.0186	0.604±0.0550	0.423±0.0714	0.054±0.0414