
Towards Unsupervised Open-Set Graph Domain Adaptation via Dual Reprogramming

Zhen Zhang^{1,2}

¹Nanjing University

²National University of Singapore

zhen_zhang@nju.edu.cn

Bingsheng He

National University of Singapore

Singapore

dcsheb@nus.edu.sg

Abstract

Unsupervised Graph Domain Adaptation has become a promising paradigm for transferring knowledge from a fully labeled source graph to an unlabeled target graph. Existing graph domain adaptation models primarily focus on the closed-set setting, where the source and target domains share the same label spaces. However, this assumption might not be practical in the real-world scenarios, as the target domain might include classes that are not present in the source domain. In this paper, we investigate the problem of unsupervised open-set graph domain adaptation, where the goal is to not only correctly classify target nodes into the known classes, but also recognize previously unseen node types into the unknown class. Towards this end, we propose a novel framework called GraphRTA, which conducts reprogramming on both the graph and model sides. Specifically, we reprogram the graph by modifying target graph structure and node features, which facilitates better separation of known and unknown classes. Meanwhile, we also perform model reprogramming by pruning domain-specific parameters to reduce bias towards the source graph while preserving parameters that capture transferable patterns across graphs. Additionally, we extend the classifier with an extra dimension for the unknown class, thus eliminating the need of manually specified threshold in open-set recognition. Comprehensive experiments on several public datasets demonstrate that our proposed model can achieve satisfied performance compared with recent state-of-the-art baselines. Our source codes and datasets are publicly available at <https://github.com/cszhangzhen/GraphRTA>.

1 Introduction

Graph Neural Networks (GNNs) have demonstrated impressive capabilities in a wide range of graph-based tasks, such as node classification [25, 14, 48], social recommendation [54, 6, 57], molecular generation [59, 34, 44] and point cloud processing [22, 43, 45], etc. Despite their great success, these GNN models often suffer from severe performance degradation when confronted with distribution shifts in graphs, such as changes in the underlying structures, node features, and label distributions [52, 30, 28]. *Unsupervised Graph Domain Adaptation* has become a promising strategy for addressing the domain shift problem by transferring knowledge across domains without relying on labels in the target domain. The majority of existing models are developed under the *closed-set* setting [62, 51, 66], which assumes that the source and target domains share the same set of classes.

However, such a strict assumption is unrealistic in real-world applications, since the target domain might introduce new classes that are absent from the source domain [64, 53], leading to significant challenges in identifying unseen samples. For instance, fraud detection models trained on previously known fraudulent behaviors from the source domain might struggle in the emerging target domain, as fraud schemes are continually evolving with fraudsters frequently developing new tactics.

Therefore, treating all novel category instances as known classes will significantly compromise the model’s ability to generalize across diverse environments. Towards this end, *open-set* graph domain adaptation [49, 33] has been proposed to accurately classify known node types into their corresponding classes while simultaneously identifying unseen node types into an unknown class. It promotes better generalization capabilities, making the model applicable across a variety of applications.

There exist some recent endeavors to explore the unsupervised open-set graph domain adaptation task [49, 40, 33, 32, 53]. The main procedure involves dividing target instances into the known and unknown groups based on their prediction entropy, then aligning the known group with the source domain. To distinguish between these two groups and enable open-set recognition, a predefined threshold is often employed, assuming that the unknown instances will exhibit higher entropy than the known ones. While promising, these methods depend heavily on manually set threshold and one threshold cannot fit all, which makes them difficult to adapt to different distributions. Additionally, they also struggle with learning clear decision boundaries, since they mainly focus on aligning source domain with the target known group, which may result in inadequate separation of the target unknown group. Hence, more efforts are necessary to tackle the challenges of open-set graph domain adaptation, particularly in recognizing and separating the target unknown group.

To address the aforementioned challenges, we propose a novel framework named GraphRTA (*Reprogram To Adapt*), which performs dual reprogramming from the graph and the model perspectives. Specifically, we reprogram the target graph by refining its structure and node features to explicitly reduce the distribution shift and encourage a clearer separation between the known and unknown groups. At the same time, we reprogram the model by pruning domain-specific parameters based on their gradients, thereby mitigating bias towards to the source graph, while retaining parameters that capture transferable patterns between the source and target graphs. Furthermore, we augment the classifier by adding an extra dimension for the unknown class, which removes the necessity for a manually specified threshold in open-set recognition. Extensive experiments across multiple public datasets indicates that our proposed model achieves superior performance in comparison to the latest state-of-the-art baselines.

In summary, our key contributions are as follows:

- We explore the challenge of unsupervised open-set graph domain adaptation, which is more practical in the real-world scenarios yet remains under-explored in the graph community.
- We are the first to reprogram both the model and the graph to improve the alignment and separation processes, offering an architecture-agnostic solution that can be applied across various GNN architectures.
- Experimental results show that GraphRTA outperforms or matches the SOTA baselines, highlighting the effectiveness of our approach in addressing the challenges associated with the open-set graph domain adaptation task.

2 Related Work

Graph Neural Networks. During the past decade, GNNs have demonstrated remarkable capability in tackling a wide range of graph learning tasks. Following the message passing framework, numerous types of GNNs have been developed, which can be broadly classified into spectral and spatial approaches [56, 50, 1]. Spectral methods, such as ChebConv [4] and Spec-GN [58], derive graph convolution operator based on spectral graph theory. In contrast, spatial models like GCN [25], GraphSAGE [14] and GAT [48] perform convolution by directly aggregating information from neighboring nodes. For comprehensive insights into these models, readers may refer to comprehensive surveys on GNNs [55, 23]. Despite their impressive performance, GNNs often depend on high-quality labeled data, which could be challenging to obtain in real-world applications. Additionally, their performance can significantly deteriorate when encountering distribution discrepancies. To mitigate this limitation, recent work has focused on adapting models trained on label-rich source domains to unlabeled target domains, thereby enhancing their generalization capabilities.

Closed-Set Graph Domain Adaptation. Although domain adaptation has been extensively investigated in computer vision [11, 60, 40] and natural language processing [39, 5, 61], research on graph domain adaptation remains in its early stages. Current graph domain adaptation methods primarily aim to learn domain-invariant representations, typically employing statistical matching techniques

such as maximum mean discrepancy [13] or central moment discrepancy [63], or by adopting adversarial learning mechanism [38] for implicit alignment. More specifically, UDAGCN [52] utilizes a combination of local and global graph encoders alongside adversarial training to achieve domain-invariant representations. GRADE [51] introduces a graph subtree discrepancy metric to reduce distribution shifts between source and target graphs, while SpecReg [62] applies spectral regularization to support theory-grounded graph domain adaptation. StruRW [30] proposes an edge re-weighting strategy to mitigate conditional structure shifts. A2GNN [28] highlights the inherent adaptability of graph neural networks by decoupling its transformation and propagation layers. Nevertheless, all of these aforementioned models assume that both the source and target graphs share the same label space [65, 29]. Unfortunately, this is impractical in real-world scenarios, as the target graph may contain new classes that do not exist in the source graph.

Open-Set Graph Domain Adaptation. Open-set domain adaptation extends closed-set domain adaptation by recognizing target novel classes that are not present in the source domain, meanwhile accurately classifying target instances that belong to the source label space [36, 41, 27, 26]. Recent models utilize a threshold to designate low-confidence samples as unknown, while aligning the source domain with the known portion of the target domain via adversarial training [19, 49, 33, 18]. Among them, DANCE [40] applies self-supervised neighborhood clustering to align each target sample with either a neighboring instance or a source prototype. PGL [33] utilizes a progressive approach to gradually reject target samples and align conditional distributions through episodic training. OpenWGL [53] introduces an uncertainty-based node representation learning framework that employs a constrained variational graph autoencoder to filter out unknown instances with a multi-sampling approach. OpenWRF [16] integrates out-of-distribution detection techniques with neighborhood information from the graph to identify novel classes. G2Pxy [64] generates both the internal and external unknown proxies via mixup to predict the distribution of novel classes in open-set learning. SDA [49] groups target representations into several clusters and employs a separate domain alignment strategy to align each target sample with either a target cluster center or a source prototype. In contrast, our model is designed from both a model-centric and a data-centric perspective, which effectively generalizes to the target domain.

3 The Proposed Model

3.1 Notations and Problem Definition

For unsupervised open-set graph domain adaptation, we are given a labeled source graph $\mathcal{G}_s = (\mathbf{X}_s, \mathbf{A}_s, \mathbf{Y}_s)$ containing n_s nodes and an unlabeled target graph $\mathcal{G}_t = (\mathbf{X}_t, \mathbf{A}_t)$ with n_t nodes, where the source and target graphs are sampled from different probability distributions, i.e., $\mathbb{P}(\mathcal{G}_s) \neq \mathbb{P}(\mathcal{G}_t)$. The feature matrix $\mathbf{X} \in \mathbb{R}^{n \times f}$ represents node attribute information, and the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ indicates the connectivity information between nodes. The source graph includes a set of classes \mathcal{C}_s forming the node label matrix $\mathbf{Y}_s \in \mathbb{R}^{n_s \times |\mathcal{C}_s|}$. Meanwhile, the target graph is associated with an additional set of classes $\mathcal{C}_{t \setminus s}$, collectively labeled as ‘unknown’, resulting in a total of $|\mathcal{C}_t| = |\mathcal{C}_s| + 1$ classes. We decompose the GNN model $\Phi(\cdot)$ into two fundamental parts: the feature extractor $f_w(\cdot)$, which transforms the graph into the node representation space, and the classifier $g_\phi(\cdot)$, which assigns class labels based on these node representations. Therefore, the GNN model $\Phi(\cdot)$ can be represented as $\Phi = f_w \circ g_\phi$. Our problem can then be formulated as follows:

Given a graph neural network Φ , a labeled source graph \mathcal{G}_s with label set \mathcal{C}_s and an unlabeled target graph \mathcal{G}_t with label space \mathcal{C}_t , where \mathcal{C}_s is a subset of \mathcal{C}_t , our goal of unsupervised graph domain adaptation is to train the model Φ using \mathcal{G}_s and \mathcal{G}_t to accurately classify target nodes when they belong to a label in \mathcal{C}_s , while marking nodes as ‘unknown’ if their labels fall outside of \mathcal{C}_s .

3.2 Graph Neural Networks Revisiting

Current GNNs operate within the message passing framework [25, 14, 48], which performs convolution by iteratively aggregating representations from its local neighborhood. Taking GCN [25] as an example, the node representations at layer l can be calculated as follows:

$$\mathbf{Z}^l = f_w(\mathcal{G}, \mathbf{W}^l) = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{Z}^{l-1} \mathbf{W}^l). \quad (1)$$

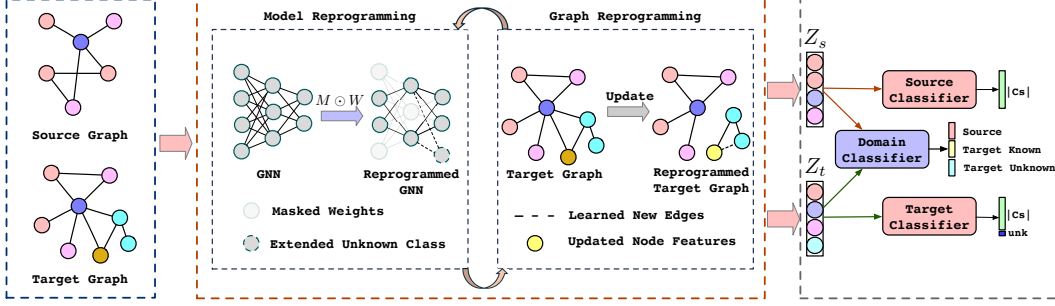


Figure 1: The pipeline of the proposed GraphRAT model. In the model reprogramming module, some nodes and connections are faded, indicating that these weights have been pruned during the reprogramming process. In the graph reprogramming module, node features are updated, and edges are dynamically modified (deleted or added), as indicated by dashed lines. Additionally, domain adversarial learning is incorporated to categorize instances into three distinct groups: source, target-known, and target-unknown, thereby enhancing the model’s generalization capacity.

Here, $\sigma(\cdot)$ denotes the non-linear activation function, i.e., ReLU or LeakyReLU. The matrix $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ represents the adjacent matrix with self-connections, while $\tilde{\mathbf{D}}$ is the diagonal degree matrix of $\tilde{\mathbf{A}}$. $\mathbf{W}^l \in \mathbb{R}^{d_{l-1} \times d_l}$ indicates a matrix of trainable parameters. As demonstrated in Section 4.3, our proposed model is architecture agnostic and can be employed across diverse GNN frameworks.

3.3 Domain-Agnostic Model Reprogramming

We present an overview of the proposed GraphRTA framework in Figure 1. Unsupervised open-set graph domain adaptation involves training a model on a labeled source graph and an unlabeled target graph, where the target graph contains novel classes not present in the source graph. Since the model is trained using labeled data from the source graph while having no access to labels in the target graph, a primary challenge is the risk of bias toward the source domain. This discrepancy can lead the model to overemphasize source-specific patterns, thereby limiting its ability to generalize effectively to the target domain.

Motivated by the lottery ticket hypothesis [9, 2, 35, 3], which demonstrates that only a subset of parameters is crucial for generalization, we propose to reprogram the graph neural network $f_w(\cdot)$ by selectively masking its weights. Specifically, we introduce differentiable masks \mathbf{M}^l to indicate the insignificant elements within the weights \mathbf{W}^l of each layer. Therefore, the node representations \mathbf{Z} at layer l is computed as follows:

$$\mathbf{Z}^l = f_w(\mathcal{G}, \tilde{\mathbf{W}}^l) = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{Z}^{l-1} (\mathbf{W}^l \odot \mathbf{M}^l)), \quad (2)$$

where \odot denotes the element-wise product. Hyperparameter ρ is utilized to control the sparsity of the masks, determining the proportion of weights to be retained. To quantify the importance of each weight element, we calculate its gradient $\nabla \mathbf{M}^l$ with respect to the loss function. The absolute value of these gradients is then used as the importance score, reflecting how much each weight contributes to generalization in the target domain. Weights with smaller gradients typically contribute less to reducing the loss and may capture domain-specific patterns that do not help in adapting to the target domain and recognition of unseen classes. By masking such weights, the reprogrammed model can focus better on transferable features that generalize across domains. Hence, we set the lowest ρ percent of gradient values in \mathbf{M}^l to zero, leaving the remaining elements at 1. These sparse masks are then applied to prune \mathbf{W}^l , resulting in the reprogrammed sparse model.

To handle open-set recognition, we further reprogram the output layer by augmenting an extra dimension for the unknown class as follows:

$$g_\phi(\mathbf{z}) = [\phi^\top \mathbf{z}, \hat{\mathbf{w}}^\top \mathbf{z}], \quad (3)$$

where $\phi \in \mathbb{R}^{d \times |\mathcal{C}_s|}$ represents the closed-set classifier and $\hat{\mathbf{w}} \in \mathbb{R}^{d \times 1}$ denotes the linear projection layer for the unknown class. The augmented logits are then passed through a softmax layer to generate the posterior probabilities, with the final prediction assigned to the class with the highest probability. This mechanism is designed to effectively distinguish between known and unknown

classes through using a dynamic threshold based on the input node representation \mathbf{z} , whereas existing models typically rely on a manually defined threshold, making them less adaptable to domain discrepancies. Through these model reprogramming procedures, we develop a domain-agnostic GNN model capable of identifying open-set classes.

3.4 Distribution-Aware Graph Reprogramming

While our proposed model reprogramming module can help mitigate source bias, it does not address the fundamental domain shift that arises from the structural and feature differences between the source and target graph data. Existing approaches often overlook the fact that the domain shift is inherently caused by the input graph’s unique characteristics. This discrepancy makes it challenging for the GNN model to generalize well across domains in the open-set scenario, as the target graph may contain novel structures, features or classes that the source domain does not encompass.

To overcome this limitation, we further propose performing graph reprogramming, where the target graph itself is refined to improve compatibility between the source and target domains. Through this approach, we modify the target graph’s structure and node features to better align with the source domain, while differentiating the known and unknown groups within the target domain [66]. This strategy not only mitigates domain shift but also strengthens the model’s ability to generalize and recognize the unseen classes. More specifically, we implement graph reprogramming by applying transformation functions to adjust both the target graph structure and node features:

$$\hat{\mathbf{X}}_t = \psi_x(\mathbf{X}_t), \hat{\mathbf{A}}_t = \psi_a(\mathbf{A}_t), \quad (4)$$

where $\psi_x(\cdot)$ represents the transformation function applied to update node features, and $\psi_a(\cdot)$ denotes the function for modifying the graph structure by adding or removing edges. Although numerous approaches, like graph structure learning methods [7, 21, 31], can be employed to adjust the graph data, we choose two simple, direct transformation strategies described below, with additional options explored in the ablation study in Appendix B.

For node features, we define the transformation function $\psi_x(\mathbf{X}_t) = \mathbf{X}_t + \Delta\mathbf{X}_t$, where $\Delta\mathbf{X}_t \in \mathbb{R}^{n_t \times f}$ represents a set of continuous, learnable parameters. This simple formulation allows for either masking node features (setting them to zero) or modifying their values, enhancing the models flexibility in refining node representations. For graph structure, the transformation function $\psi_a(\cdot)$ is modeled as $\psi_a(\mathbf{A}_t) = \mathbf{A}_t \oplus \Delta\mathbf{A}_t$, where $\Delta\mathbf{A}_t \in \mathbb{R}^{n_t \times n_t}$ is a binary matrix that adjusts the graph’s structure by adding or removing edges. The operation \oplus denotes element-wise exclusive OR (XOR), where if both the corresponding values in \mathbf{A}_t and $\Delta\mathbf{A}_t$ are 1, the XOR operation results in 0, effectively deleting the edge. Conversely, if one value is 0 and the other is 1, an edge is added. To regulate the changes to the graph structure, we impose a constraint on the total number of modifications made to the adjacency matrix. Particularly, the sum of the entries in $\Delta\mathbf{A}_t$ is limited by a pre-defined budget \mathcal{B} , i.e., $\sum \Delta\mathbf{A}_t \leq \mathcal{B}$, preventing excessive changes that might deviate too much from the original graph and ensuring computational efficiency. Through these graph reprogramming processes, we dynamically update the target graph to mitigate distribution shifts and facilitate open-set recognition.

3.5 Training Procedure

In this section, we detail the process of updating the model reprogramming and graph reprogramming modules using the proposed losses. Following previous works [12, 20, 33, 46], we adopt a domain adversarial learning framework to match the source and target distributions through feature alignment. In an open-set scenario, where the target domain contains unknown classes not present in the source domain, such alignment approach can cause negative transfer due to class set mismatches. Existing methods focus on aligning source and target features within the known class set, ignoring any alignment signal from target instances that belong to unknown classes. As a result, the classifier is unable to establish a clear decision boundary for unknown classes, as target-unknown instances remain entangled with known ones in the aligned feature space.

To this end, we aim at explicitly pushing target-unknown features apart from both the source and target-known features, while ensuring alignment between source and target-known features. Specifically, we first calculate the entropy value for each target instance based on the known classes as

follows:

$$e_i = \mathcal{H}(\mathbf{p}_i) = - \sum_{k=1}^{|\mathcal{C}_s|} \mathbf{p}_{ik} \log(\mathbf{p}_{ik}), \quad (5)$$

where \mathbf{p}_i is the output probability generated by the classifier $g_\phi(\cdot)$. The entropy value, which quantifies the uncertainty, serves as an indicator for open-set recognition, where a high entropy suggests the instance may belong to the unknown class. After normalizing the entropy values, we model them as being generated by a mixture of two Beta distributions to capture the overall characteristics of the target graph [20]:

$$p(e_i) = \mu_{tk} \cdot p(e_i|tk) + \mu_{tu} \cdot p(e_i|tu), \quad (6)$$

where e_i represents the entropy value for node v_i . $p(e_i|tk)$ and $p(e_i|tu)$ denote the probability density functions for the target-known (tk) and target-unknown groups (tu), respectively. Meanwhile, μ_{tk} and μ_{tu} are the mixing coefficients for these two distributions. Then, we perform posterior inference by fitting a Beta mixture model using the Expectation-Maximization (EM) algorithm:

$$p(tk|e_i) = \frac{\mu_{tk} \cdot p(e_i|tk)}{\mu_{tk} \cdot p(e_i|tk) + \mu_{tu} \cdot p(e_i|tu)}. \quad (7)$$

Thus, we estimate the probability that an instance belongs to the target-known group based on its entropy value without using any thresholds and $p(tu|e_i) = 1 - p(tk|e_i)$.

After estimating the probability of each target instance belonging to either the target-known or target-unknown group, we can classify all the instances into three distinct domains for the domain adversarial learning framework, i.e., *source*, *target-known*, and *target-unknown*. To achieve this goal, we introduce a domain discriminator $d_\theta(\cdot)$, implemented as a multi-layer perceptron (MLP), which engages in a minimax game with the feature extractor $f_w(\cdot)$. The feature extractor works to learn node representations that make it challenging for the discriminator to identify the origin of each node. We implement adversarial training using a Gradient Reversal Layer (GRL) [12], which promotes the maximization of feature representations. Meanwhile, the domain discriminator $d_\theta(\cdot)$ is optimized by minimizing the cross-entropy loss to effectively classify domains as follows:

$$\mathcal{L}_{adv} = - \frac{1}{n_s + n_t} \sum_{i=1}^{n_s+n_t} \sum_{k=1}^3 \mathbf{y}_{ik} \log(\hat{\mathbf{y}}_{ik}), \quad (8)$$

where $\mathbf{y}_i = [1, 0, 0]$ when the node is from the source graph, and $\mathbf{y}_i = [0, p(tk|e_i), p(tu|e_i)]$ when the node is from the target graph. $\hat{\mathbf{y}}_i$ represents the domain prediction for node v_i . Thus, the adversarial learning loss simultaneously aligns and segregates the three sets to learn domain-invariant representations.

Furthermore, we update the model reprogramming module by utilizing the label information in the source graph as follows:

$$\mathcal{L}_{cls} = \sum_{i=1}^{n_s} \mathcal{L}_{ce}(\Phi(\mathbf{x}_i), y_i) + \lambda \mathcal{L}_{ce}(\Phi(\mathbf{x}_i) \setminus y_i, |\mathcal{C}_s| + 1), \quad (9)$$

where \mathcal{L}_{ce} represents the cross-entropy loss, while $\Phi(\cdot)$ denotes the reprogrammed model. λ is a trade-off hyper-parameter. The first term focuses on optimizing the augmented output to match the ground truth labels, thereby preserving performance on the closed set. For the second term, $\Phi(\mathbf{x}_i) \setminus y_i$ removes the probability associated with the ground truth label and aligns it with class $|\mathcal{C}_s| + 1$, which can be regarded as a simplified mixup approach. By masking out the ground truth, we ensure that the model is explicitly trained to classify instances as unknown when they do not align with any of the known class patterns. For graph reprogramming module, we incorporate entropy minimization loss to encourage confident predictions for the unlabeled target instances, while simultaneously distinguishing target-known features from target-unknown features as follows:

$$\mathcal{L}_{ent} = \sum_{i=1}^{n_t} \mathcal{H}(\Phi(\mathbf{x}_i)) + \mathcal{L}_{ce}(\hat{\mathbf{y}}_i, p(tu|e_i)), \quad (10)$$

where $\mathcal{H}(\cdot)$ is the entropy function defined in Eq.(5), while minimizing $\mathcal{L}_{ce}(\hat{\mathbf{y}}_i, p(tu|e_i))$ enables the graph to generate discriminative features specifically for target-unknown instances. Therefore, the overall loss function is:

$$\mathcal{L} = \mathcal{L}_{adv} + \mathcal{L}_{cls} + \mathcal{L}_{ent}. \quad (11)$$

4 Experiments

4.1 Experimental Settings

Datasets. To thoroughly assess the performance of our proposed GraphRTA, we conduct experiments using three categories of publicly available datasets. An overview of these dataset characteristics is provided in Table 1, with more detailed descriptions as follows.

The first category comprises three *citation* datasets, i.e., DBLPv7(**D**), Citationv1(**C**), and ACMv9(**A**) [28], where nodes represent individual papers, while edges indicate citation relationships. Particularly, DBLPv7 encompasses DBLP papers published between 2004 and 2008, Citationv1 contains articles from Microsoft Academic Graph up to 2008, and ACMv9 consists of papers published by ACM from 2000 to 2010. Each paper is classified into one of the five distinct research topics: Databases, Artificial Intelligence, Computer Vision, Information Security, and Networking.

Table 1: Dataset statistics.

Datasets	#Nodes	#Edges	#Feat	#Class
DBLPv7	5,484	8,117		
Citationv1	8,935	15,098	6,775	5
ACMv9	9,360	15,556		
ogbn-arxiv	169,343	1,166,243	128	40
Cornell	183	298		
Texas	183	325	1,703	5
Wisconsin	251	515		

We further include the *ogbn-arxiv* dataset [17], which is composed of computer science papers from arXiv. Each paper is represented by a 128-dimensional feature vector, derived by averaging the embeddings of the words in its title and abstract. We partition this dataset into three temporal domains according to the publication years: 1950-2016 (*Arxiv I*), 2016-2018 (*Arxiv II*), and 2018-2020 (*Arxiv III*). The task involves classifying each paper into one of 40 predefined subject areas under the temporal shifts.

Lastly, we incorporate the *WebKB* dataset [37], a webpage dataset from computer science departments across various universities. Among them, we select three heterophily graphs (i.e., *Cornell*, *Texas*, and *Wisconsin*), where nodes represent web pages and edges indicate hyperlinks between them. Each webpage is denoted by bag-of-words features, then we categorize them into the following five groups: student, project, course, staff, or faculty.

Baselines. We compare our proposed GraphRTA against a broad set of recent baselines across three key categories. (1) *Graph Neural Networks*: This category includes traditional GNN models like GCN [25], SAGE [14], and GAT [48]. They are trained on the source graph and then evaluated directly on the target graph without adaptations between domains. For open-set recognition, a threshold is applied to identify instances belonging to unseen categories. (2) *Closed-Set Graph Domain Adaptation*: Approaches in this group focus on graph domain adaptation within the closed-set settings, where the source and target graphs share the same label space. We compare our model with several recent methods including UDAGCN [52], GRADE [51], SpecReg [62], StruRW [30], and A2GNN [28]. Similarly, we employ a predefined threshold to identify open-set instances. (3) *Open-Set Graph Learning or Domain Adaptation*: This group of methods is designed for scenarios where the target graph contains categories that do not exist in the source graph. We consider DANCE [40], OpenWGL [53], PGL [33], OpenWRF [16], G2Pxy [64], SDA [49] and UAGA [42] for comparisons. These approaches are strong baselines for assessing our model’s capability to transfer knowledge to the target domain and generalize effectively to unseen categories.

Implementation Details. In this work, we adopt the experimental setup used in prior researches [64, 49], where a portion of classes is reserved as “unknown” with the remaining classes regarded as “known”. Specifically, $|\mathcal{C}_s|$ is 3 in Citation and WebKB datasets, and 30 for the ogbn-arxiv dataset. The domain adaptation models are trained using labeled source nodes from the known classes along with unlabeled nodes from the target graph. Among them, 70% of the labeled source nodes are utilized for training, 10% are set aside for validation, and the remaining 20% serve as a sanity check. The final evaluation is conducted on the target nodes. For a fair comparison, we utilize the baselines’ publicly available source codes and tune their hyperparameters to their optimal values using the validation set. Our proposed GraphRTA is implemented using PyTorch Geometric [8] and optimized with the Adam optimizer [24]. Hyperparameters for learning rate, weight decay and λ are searched within the ranges of $[0.1, 0.01, 0.001, 1e^{-4}, 1e^{-5}]$, and the sparse constraint ρ is explored within the interval $[0, 1]$. The experiments are repeated five times, and performance metrics are reported as the

Table 2: Node classification accuracy and H-score (mean \pm std) for citation datasets. The best results are shown in bold with the second-best results underlined.

Models	ACMv9 \rightarrow Citationv1		ACMv9 \rightarrow DBLPv7		Citationv1 \rightarrow ACMv9		Citationv1 \rightarrow DBLPv7		DBLPv7 \rightarrow ACMv9		DBLPv7 \rightarrow Citationv1	
	Acc	HS	Acc	HS	Acc	HS	Acc	HS	Acc	HS	Acc	HS
GCN	40.64 \pm 0.98	41.02 \pm 2.11	45.84 \pm 1.06	50.20 \pm 1.26	47.10 \pm 0.49	49.05 \pm 0.74	51.48 \pm 0.42	56.13 \pm 0.22	43.90 \pm 1.50	44.47 \pm 2.80	39.26 \pm 0.70	37.96 \pm 1.34
SAGE	38.24 \pm 0.80	36.89 \pm 2.10	41.77 \pm 1.05	45.20 \pm 1.41	43.90 \pm 1.56	45.63 \pm 2.18	47.14 \pm 0.82	51.65 \pm 0.83	41.66 \pm 0.47	42.04 \pm 1.32	39.62 \pm 0.22	40.32 \pm 0.40
GAT	32.01 \pm 0.73	21.51 \pm 2.36	34.84 \pm 1.25	32.90 \pm 2.66	36.38 \pm 0.66	30.67 \pm 1.66	34.56 \pm 0.64	32.22 \pm 1.53	35.85 \pm 1.91	25.66 \pm 5.30	32.87 \pm 1.41	20.91 \pm 3.92
UDAGCN	44.78 \pm 4.12	20.94 \pm 6.21	55.07 \pm 1.03	50.05 \pm 7.90	53.38 \pm 1.53	53.56 \pm 6.00	62.36 \pm 4.19	43.21 \pm 4.08	47.28 \pm 1.49	39.21 \pm 1.16	52.38 \pm 1.18	46.92 \pm 6.26
GRADE	57.23 \pm 1.06	59.49 \pm 1.16	56.12 \pm 0.65	58.14 \pm 1.07	57.86 \pm 0.29	60.41 \pm 0.26	61.94 \pm 0.38	64.21 \pm 0.48	54.93 \pm 0.40	57.73 \pm 0.41	54.60 \pm 0.64	57.36 \pm 0.55
SpecReg	51.31 \pm 5.60	36.70 \pm 1.49	58.17 \pm 2.06	60.15 \pm 0.41	56.58 \pm 1.22	56.36 \pm 0.86	63.68 \pm 5.82	59.62 \pm 0.59	53.30 \pm 4.05	53.12 \pm 6.84	55.72 \pm 2.43	49.43 \pm 7.41
StruRW	46.47 \pm 4.63	42.36 \pm 4.25	46.91 \pm 1.95	46.08 \pm 3.60	51.91 \pm 0.60	45.38 \pm 0.13	56.19 \pm 0.10	54.08 \pm 1.19	48.86 \pm 0.62	43.85 \pm 1.74	51.08 \pm 0.91	43.25 \pm 1.29
A2GNN	42.53 \pm 2.07	41.66 \pm 3.69	<u>60.43\pm0.52</u>	62.74 \pm 0.82	57.21 \pm 1.03	57.12 \pm 0.63	63.45 \pm 0.31	65.37 \pm 0.59	<u>57.64\pm1.82</u>	60.68\pm2.27	41.09 \pm 1.20	43.52 \pm 1.24
DANCE	57.77 \pm 0.64	60.94 \pm 0.76	58.01 \pm 0.47	61.31 \pm 0.62	58.76 \pm 0.36	61.33 \pm 0.55	62.97 \pm 0.65	65.42 \pm 1.20	55.97 \pm 0.62	58.90 \pm 0.60	55.77 \pm 0.56	58.95 \pm 0.70
OpenWGL	49.98 \pm 0.62	5.57 \pm 1.56	52.43 \pm 0.62	7.86 \pm 0.69	48.37 \pm 0.50	4.07 \pm 0.99	55.68 \pm 0.35	3.49 \pm 0.76	48.04 \pm 1.08	25.13 \pm 4.89	49.52 \pm 0.68	22.05 \pm 2.29
PGL	54.42 \pm 1.04	57.86 \pm 1.12	48.43 \pm 1.12	53.15 \pm 1.23	51.87 \pm 0.69	54.71 \pm 0.72	53.83 \pm 0.66	59.01 \pm 0.75	49.27 \pm 0.80	51.74 \pm 0.83	52.82 \pm 0.87	56.16 \pm 0.93
OpenWRF	53.53 \pm 2.59	31.05 \pm 4.57	48.16 \pm 1.63	35.45 \pm 3.62	47.01 \pm 3.32	33.08 \pm 6.46	52.81 \pm 1.43	31.96 \pm 1.24	47.27 \pm 1.54	19.38 \pm 4.03	57.31 \pm 1.14	34.06 \pm 9.77
G2Pxy	59.75 \pm 0.49	54.47 \pm 1.33	56.26 \pm 0.73	49.63 \pm 2.41	58.49 \pm 1.03	58.56 \pm 1.36	61.42 \pm 0.47	59.13 \pm 0.54	54.48 \pm 0.58	54.82 \pm 0.78	56.36 \pm 0.69	54.02 \pm 1.75
SDA	58.23 \pm 4.67	59.97 \pm 6.74	59.06 \pm 4.75	56.34 \pm 1.23	57.33 \pm 6.22	58.85 \pm 8.58	<u>63.55\pm0.91</u>	<u>65.53\pm2.00</u>	57.66\pm0.61	<u>60.54\pm1.20</u>	57.27 \pm 2.72	<u>59.73\pm3.87</u>
UAGA	53.37 \pm 6.72	<u>61.34\pm1.16</u>	52.11 \pm 2.96	67.50\pm2.95	52.25 \pm 4.81	60.59 \pm 5.95	52.73 \pm 5.13	64.81 \pm 4.50	48.14 \pm 1.47	55.73 \pm 3.98	47.97 \pm 0.24	<u>52.16\pm2.18</u>
GraphRTA	66.26\pm0.93	66.33\pm1.69	62.33\pm0.68	<u>64.42\pm1.10</u>	60.93\pm2.63	62.89\pm2.46	63.87\pm1.97	65.99\pm1.87	56.91 \pm 2.50	59.41 \pm 2.22	60.11\pm1.98	62.33\pm1.53

Table 3: Node classification accuracy and H-score (mean \pm std) for ogbn-arxiv and WebKB datasets. OOM means out-of-memory. ‘-’ indicates cases where baseline methods encounter errors because their predefined strategies are not satisfied.

Models	Arxiv I \rightarrow Arxiv II		Arxiv I \rightarrow Arxiv III		Arxiv II \rightarrow Arxiv III		Cornell \rightarrow Wisconsin		Texas \rightarrow Cornell		Texas \rightarrow Wisconsin	
	Acc	HS	Acc	HS	Acc	HS	Acc	HS	Acc	HS	Acc	HS
GCN	44.82 \pm 0.20	41.08 \pm 0.74	41.57 \pm 0.22	41.05 \pm 0.66	45.65 \pm 0.37	41.04 \pm 0.89	21.19 \pm 0.17	0.20 \pm 0.45	38.46 \pm 8.43	25.19 \pm 11.0	21.83 \pm 6.29	13.32 \pm 9.15
SAGE	44.95 \pm 0.15	37.83 \pm 0.71	42.75 \pm 0.15	38.63 \pm 0.64	49.60 \pm 0.12	38.11 \pm 0.49	18.56 \pm 2.86	10.30 \pm 10.7	34.75 \pm 3.31	11.66 \pm 4.91	24.22 \pm 10.7	14.48 \pm 9.01
GAT	44.81 \pm 0.13	34.31 \pm 1.39	42.05 \pm 0.30	34.97 \pm 0.76	46.49 \pm 0.17	36.35 \pm 0.78	20.96 \pm 0.21	0.40 \pm 0.54	27.97 \pm 3.00	15.89 \pm 9.15	9.48 \pm 2.31	8.08 \pm 2.24
UDAGCN	31.90 \pm 2.27	33.68 \pm 3.48	27.71 \pm 0.86	28.37 \pm 1.63	35.09 \pm 1.29	39.53 \pm 1.56	19.28 \pm 6.57	5.94 \pm 5.51	29.18 \pm 1.87	7.35 \pm 5.28	22.78 \pm 5.19	3.93 \pm 4.12
GRADE	43.01 \pm 0.20	47.18 \pm 0.29	39.28 \pm 0.37	44.19 \pm 0.41	42.80 \pm 0.17	46.09 \pm 0.21	17.05 \pm 11.5	12.52 \pm 1.26	28.96 \pm 7.54	21.52 \pm 11.5	24.46 \pm 7.32	23.84 \pm 4.67
SpecReg	37.80 \pm 1.90	31.76 \pm 1.64	28.14 \pm 4.59	29.03 \pm 3.87	46.60 \pm 0.29	31.70 \pm 4.27	20.79 \pm 3.24	19.93 \pm 3.12	31.69 \pm 3.34	13.53 \pm 9.15	19.20 \pm 4.72	10.28 \pm 6.14
StruRW	37.47 \pm 1.93	40.67 \pm 2.09	36.17 \pm 0.27	40.54 \pm 0.45	42.10 \pm 0.44	43.62 \pm 0.50	16.57 \pm 2.26	16.22 \pm 3.72	42.02 \pm 7.51	40.98\pm7.86	16.01 \pm 2.47	11.46 \pm 6.17
A2GNN	42.07 \pm 0.14	45.00 \pm 0.24	38.92 \pm 0.16	43.14 \pm 0.17	42.26 \pm 0.53	45.18 \pm 0.17	19.12 \pm 2.56	17.40 \pm 3.32	<u>44.37\pm0.71</u>	31.59 \pm 6.44	14.98 \pm 0.77	6.22 \pm 2.58
DANCE	OOM	OOM	OOM	OOM	OOM	OOM	21.52 \pm 23.4	3.11 \pm 0.49	20.77 \pm 0.00	0.00 \pm 0.00	4.22 \pm 0.21	1.65 \pm 1.50
OpenWGL	32.58 \pm 0.58	1.45 \pm 0.40	32.99 \pm 1.49	1.31 \pm 0.37	35.46 \pm 2.61	0.04 \pm 0.08	16.57 \pm 2.49	14.09 \pm 2.30	33.22 \pm 5.77	28.99 \pm 5.65	10.51 \pm 4.80	8.76 \pm 3.56
PGL	41.50 \pm 0.25	46.32 \pm 0.28	38.38 \pm 0.14	43.31 \pm 0.16	40.28 \pm 0.24	45.43 \pm 0.27	18.24 \pm 7.16	0.00 \pm 0.00	21.20 \pm 0.45	0.00 \pm 0.00	28.45 \pm 5.93	0.00 \pm 0.00
OpenWRF	32.58 \pm 0.58	1.45 \pm 0.40	32.99 \pm 1.49	1.31 \pm 0.37	35.46 \pm 2.61	0.04 \pm 0.08	<u>26.29\pm7.11</u>	6.84 \pm 5.81	21.64 \pm 3.81	4.58 \pm 5.03	<u>26.93\pm11.4</u>	5.99 \pm 4.65
G2Pxy	31.13 \pm 4.55	28.45 \pm 1.13	24.79 \pm 1.49	16.28 \pm 3.25	34.93 \pm 3.03	37.27 \pm 6.75	-	-	-	-	-	-
SDA	39.77 \pm 0.34	42.60 \pm 0.15	36.37 \pm 0.21	39.44 \pm 0.21	41.53 \pm 0.20	46.03 \pm 0.18	-	-	-	-	-	-
UAGA	32.92 \pm 0.16	23.69 \pm 0.16	32.24 \pm 0.13	22.98 \pm 0.13	39.16 \pm 0.39	29.79 \pm 0.41	-	-	-	-	-	-
GraphRTA	47.70\pm1.39	50.79\pm2.79	45.52\pm2.00	46.25\pm0.40	52.37\pm1.49	48.42\pm1.94	33.46\pm4.43	34.36\pm1.76	52.18\pm0.38	<u>35.64\pm0.83</u>	29.34\pm1.21	30.08\pm2.96

mean along with standard deviations for both accuracy and H-score [10]. The H-score combines the accuracies of the target known and target unknown classes to provide a balanced assessment of the model’s performance as follows:

$$HS = \frac{2 \times Acc_{tk} \times Acc_{tu}}{Acc_{tk} + Acc_{tu}}, \quad (12)$$

where Acc_{tk} denotes accuracy on the target known classes, and Acc_{tu} indicates accuracy on the target unknown classes. A higher score means balanced performance across known and unknown target categories, offering a more comprehensive evaluation metric.

4.2 Results and Analyses

We present the overall results in Table 2 and Table 3. As we can see, our proposed GraphRTA consistently demonstrates superior performance across a variety of scenarios. Within the three baseline categories, standard GNN models show limited performance, due to their lack of mechanisms to address distribution shifts between source and target graphs. Closed-set adaptation baselines, which account for these distribution shifts, yield better results than standard GNNs but struggle to handle open-set classes effectively. In contrast, open-set baselines generally perform well by incorporating strategies to recognize and manage previously unseen classes in the target graph, though they are still outperformed by GraphRTA in most cases.

We observe that the H-score provides a more comprehensive evaluation metric than accuracy. Several baselines achieve high accuracy scores but suffer from low H-scores, reflecting their difficulty in accurately identifying open-set instances. For example, while OpenWGL achieves an accuracy of 49.98% in the scenario of ACMv9 \rightarrow Citationv1, its H-score is only about 5.57%, illustrating its limitations in open-set recognition. Additionally, most models struggle with heterophilous datasets like ogbn-arxiv and WebKB, which pose additional challenges due to their weakly correlated characteristics. Two recent baselines encounter issues in this context, for instance, G2Pxy fails to meet

Table 4: Classification H-score with different known classes.

Classes	A→C	A→D	C→A	C→D	D→A	D→C
$C_{\text{known}} = 2$	53.39±1.87	45.29±3.46	41.10±2.63	41.54±2.92	44.48±2.86	54.18±4.41
$C_{\text{known}} = 3$	66.33±1.69	64.42±1.10	62.89±2.46	65.99±1.87	59.41±2.22	62.33±1.53
$C_{\text{known}} = 4$	65.74±0.54	63.28±0.69	61.63±0.64	67.20±0.63	57.12±0.35	59.82±0.69

Table 5: Classification H-score with different openset class detection strategies.

Methods	A→C	A→D	C→A	C→D	D→A	D→C
GraphRTA _{thres}	63.07±0.13	61.21±1.07	61.00±1.52	63.89±1.35	58.88±0.72	59.73±1.10
GraphRTA	66.33±1.69	64.42±1.10	62.89±2.46	65.99±1.87	59.41±2.22	62.33±1.53

the predefined rules for node embedding mixup, while SDA struggles to form meaningful clusters under these conditions. In contrast, GraphRTA overcomes these limitations by leveraging model and graph reprogramming, enabling it to deliver robust performance without such constraints.

4.3 Ablation Studies

Impact of Different Known Classes. We have also conducted additional ablation studies to evaluate the impact of varying the number of known classes on the model’s performance. Specifically, we aim to understand how the size of the known class subset influences the model’s ability to discriminate between known and unknown categories. As the number of known classes increases, the boundary between known and unknown nodes becomes more complex, potentially affecting the generalization capability of the domain adaptation process. The results, reported in terms of the H-score on the citation dataset, are presented in Table 4. When the number of known classes is very small (e.g., only 2 known classes), open-set models have limited supervision to guide effective feature alignment and decision boundary formation, leading to less stable performance. In contrast, as more known classes are introduced, the model benefits from richer supervision, enabling more discriminative representations and improving robustness against the presence of unknown classes. These observations highlight that our model becomes progressively more reliable and less sensitive to open-set uncertainty as the number of known classes increases.

Alternative Open-set Detection Strategies. We further implemented an entropy-based thresholding variant, which replaces the 3-class classification scheme with a confidence-based decision rule. Specifically, samples with prediction entropy exceeding a pre-defined threshold are regarded as belonging to unknown classes. We then compared this variant with our proposed approach across all experimental settings. The results, measured by the H-score in Table 5, show that our method consistently outperforms the entropy-based thresholding strategy, demonstrating its superior ability to distinguish between known and unknown nodes in the open-set graph domain adaptation scenario.

Impact of Different GNN Architectures.

As discussed in Section 3.2, our proposed GraphRTA is designed to be compatible with a range of GNN architectures, allowing flexibility across models. We examine its performance using 3 prominent GNN frameworks: GCN [25], GraphSAGE [14], and GAT [48], with results presented in Table 6. These results indicate that while all architectures benefit from the proposed dual reprogramming approach, their performance varies across different datasets, highlighting the influence of architectural design on adaptation capabilities. Among them, GAT achieves the lowest performance, which adapt poorly from the source to target graphs due to its attention mechanism. The multi-head attention paradigm also requires substantial parameter tuning, which may hinder adaptation. Interestingly, GCN, despite its simpler design, consistently performs well, demonstrating resilience across most tasks.

Table 6: H-scores with different GNN architectures.

Architectures	D→C	D→A	A→C
GraphRTA _{GCN}	62.33±1.53	59.41±2.22	66.33±1.69
GraphRTA _{SAGE}	59.09±2.66	55.45±1.49	63.11±1.48
GraphRTA _{GAT}	54.17±6.60	51.82±9.14	57.51±6.53

Visualization. To better understand the quality of the learned node representations, we use t-SNE [47] to project them into a 2-D space and visualize the results in Figure 2 through scatter plots. More specifically, the three known classes are represented by red, blue, and green, while the unknown class is depicted in orange. The vanilla GCN [25], without any adaptation, struggles to

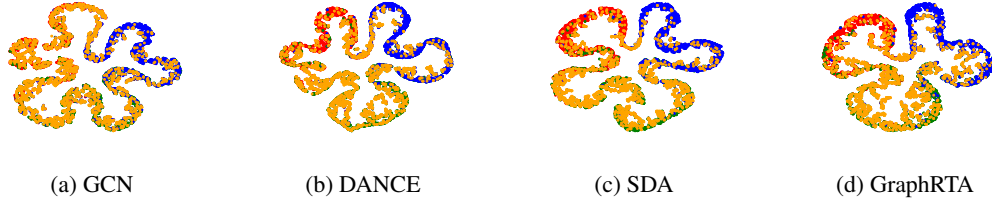
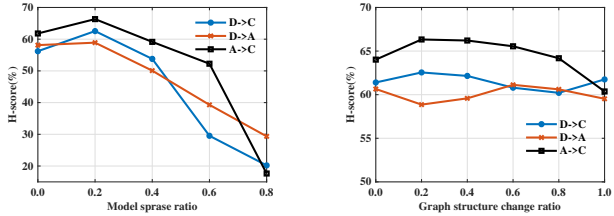


Figure 2: Visualization of node representations in the target graph for the citation dataset ($A \rightarrow C$) with the unknown class highlighted in orange.

produce distinct clusters, leading to significant overlap between nodes of the unknown class and those of the known classes. This overlap occurs because the model lacks mechanisms to address distribution shifts between the source and target graphs. In contrast, two representative open-set baselines, DANCE [40] and SDA [49], are able to identify nodes from the target unknown class, but their boundaries are blurred, with most nodes from unknown classes often blending into known class clusters. Our proposed GraphRTA achieves relatively clearer separation, generating compact clusters for known classes while effectively isolating open-set instances.

Hyper-parameter Sensitivity. In this section, we analyze the effects of two important hyperparameters: the sparsity ratio ρ within the model reprogramming module and the structural modification ratio β in the graph reprogramming module. As illustrated in Figure 3, the sparsity ratio significantly influences the model’s performance; as the masking ratio increases, the performance declines sharply. This impairs the model’s ability to capture essential patterns due to excessive weight masking. In contrast, the performance impact of adjusting the graph structure change ratio is relatively robust, suggesting that the model can adapt to moderate structural alterations in the graph without significant degradation. Additional analyses of other hyperparameters and representation visualizations are provided in the Appendix B.



(a) Model reprogramming ratio (b) Graph reprogramming ratio

5 Conclusion

This paper studies unsupervised open-set graph domain adaptation, an under-explored area in the graph community, where the target graph introduces new classes that are not present in the source graph. To address the source bias and distributional shift problems, we propose a novel framework named GraphRTA that conducts dual reprogramming at the model as well as the graph levels. Through extensive evaluations on a variety of public datasets, we further show that our proposed GraphRTA consistently outperforms or matches the performance of recent state-of-the-art models. In future work, we aim to extend our framework to address additional challenges in graph adaptation, such as source-free open-set graph domain adaptation, semi-supervised open-set graph domain adaptation, and out-of-distribution detection, thereby broadening its applicability and enhancing its robustness as well as generalization in real-world scenarios.

6 Border Impacts and Limitations

This paper advances the field of machine learning, particularly in Open-Set Graph Domain Adaptation, which enables graph models to adapt to new, unseen classes across domains. It has broad applications, including fraud detection, biological network analysis, and recommendation systems. While our work improves model robustness and generalization, potential societal impacts include both benefits (e.g., better adaptability in real-world graph-based systems) and challenges (e.g., risks of biased adaptation or misclassification in high-stakes applications). However, we do not identify any immediate or specific societal concerns.

Acknowledgments and Disclosure of Funding

This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

References

- [1] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3496–3507, 2021.
- [2] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *International conference on machine learning*, pages 1695–1706. PMLR, 2021.
- [3] Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Jingjing Liu, and Zhangyang Wang. The elastic lottery ticket hypothesis. *Advances in Neural Information Processing Systems*, 34:26609–26621, 2021.
- [4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.
- [5] Chenxiao Dou, Xianghui Sun, Yaoshu Wang, Yunjie Ji, Baochang Ma, and Xiangang Li. Domain-adapted dependency parsing for cross-domain named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12737–12744, 2023.
- [6] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.
- [7] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems*, 34:22667–22681, 2021.
- [8] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.
- [9] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.
- [10] Bo Fu, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Learning to detect open classes for universal domain adaptation. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*, pages 567–583. Springer, 2020.
- [11] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.
- [12] 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*, 17(59):1–35, 2016.
- [13] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.
- [14] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.
- [15] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. *arXiv preprint arXiv:2305.19523*, 2023.

- [16] Marcel Hoffmann, Lukas Galke, and Ansgar Scherp. Open-world lifelong graph learning. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2023.
- [17] 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. *Advances in neural information processing systems*, 33:22118–22133, 2020.
- [18] Chowdhury Sadman Jahan and Andreas Savakis. Unknown sample discovery for source free open set domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1067–1076, 2024.
- [19] Lalit P Jain, Walter J Scheirer, and Terrance E Boult. Multi-class open set recognition using probability of inclusion. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part III 13*, pages 393–409. Springer, 2014.
- [20] JoonHo Jang, Byeonghu Na, Dong Hyeok Shin, Mingi Ji, Kyungwoo Song, and Il-Chul Moon. Unknown-aware domain adversarial learning for open-set domain adaptation. *Advances in Neural Information Processing Systems*, 35:16755–16767, 2022.
- [21] Wei Jin, Tong Zhao, Jiayuan Ding, Yozen Liu, Jiliang Tang, and Neil Shah. Empowering graph representation learning with test-time graph transformation. *arXiv preprint arXiv:2210.03561*, 2022.
- [22] Yongcheng Jing, Chongbin Yuan, Li Ju, Yiding Yang, Xinchao Wang, and Dacheng Tao. Deep graph reprogramming. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24345–24354, 2023.
- [23] Shima Khoshraftar and Aijun An. A survey on graph representation learning methods. *ACM Transactions on Intelligent Systems and Technology*, 15(1):1–55, 2024.
- [24] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [25] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [26] Jogendra Nath Kundu, Naveen Venkat, Ambareesh Revanur, R Venkatesh Babu, et al. Towards inheritable models for open-set domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12376–12385, 2020.
- [27] Hong Liu, Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Qiang Yang. Separate to adapt: Open set domain adaptation via progressive separation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2927–2936, 2019.
- [28] Meihan Liu, Zeyu Fang, Zhen Zhang, Ming Gu, Sheng Zhou, Xin Wang, and Jiajun Bu. Rethinking propagation for unsupervised graph domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13963–13971, 2024.
- [29] Meihan Liu, Zhen Zhang, Jiachen Tang, Jiajun Bu, Bingsheng He, and Sheng Zhou. Revisiting, benchmarking and understanding unsupervised graph domain adaptation. *Advances in Neural Information Processing Systems*, 37:89408–89436, 2024.
- [30] Shikun Liu, Tianchun Li, Yongbin Feng, Nhan Tran, Han Zhao, Qiang Qiu, and Pan Li. Structural re-weighting improves graph domain adaptation. In *International Conference on Machine Learning*, pages 21778–21793. PMLR, 2023.
- [31] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference 2022*, pages 1392–1403, 2022.
- [32] Yadan Luo, Zijian Wang, Zhuoxiao Chen, Zi Huang, and Mahsa Baktashmotlagh. Source-free progressive graph learning for open-set domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):11240–11255, 2023.

- [33] Yadan Luo, Zijian Wang, Zi Huang, and Mahsa Baktashmotlagh. Progressive graph learning for open-set domain adaptation. In *International Conference on Machine Learning*, pages 6468–6478. PMLR, 2020.
- [34] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *International conference on machine learning*, pages 7192–7203. PMLR, 2021.
- [35] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR, 2020.
- [36] Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 754–763, 2017.
- [37] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.
- [38] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [39] Mohammad Rostami, Digbalay Bose, Shrikanth Narayanan, and Aram Galstyan. Domain adaptation for sentiment analysis using robust internal representations. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11484–11498, 2023.
- [40] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, and Kate Saenko. Universal domain adaptation through self supervision. *Advances in neural information processing systems*, 33:16282–16292, 2020.
- [41] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 153–168, 2018.
- [42] Xiao Shen, Zhihao Chen, Shirui Pan, Shuang Zhou, Laurence T Yang, and Xi Zhou. Open-set cross-network node classification via unknown-excluded adversarial graph domain alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 20398–20408, 2025.
- [43] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719, 2020.
- [44] Hannes Stärk, Dominique Beaini, Gabriele Corso, Prudencio Tossou, Christian Dallago, Stephan Günnemann, and Pietro Liò. 3d infomax improves gnns for molecular property prediction. In *International Conference on Machine Learning*, pages 20479–20502. PMLR, 2022.
- [45] Shyam A Tailor, René De Jong, Tiago Azevedo, Matthew Mattina, and Partha Maji. Towards efficient point cloud graph neural networks through architectural simplification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2095–2104, 2021.
- [46] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.
- [47] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- [48] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [49] Yu Wang, Ronghang Zhu, Pengsheng Ji, and Sheng Li. Open-set graph domain adaptation via separate domain alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 9142–9150, 2024.

- [50] 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.
- [51] Jun Wu, Jingrui He, and Elizabeth Ainsworth. Non-iid transfer learning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10342–10350, 2023.
- [52] 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.
- [53] Man Wu, Shirui Pan, and Xingquan Zhu. Openwgl: Open-world graph learning. In *2020 IEEE international conference on data mining (icdm)*, pages 681–690. IEEE, 2020.
- [54] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 346–353, 2019.
- [55] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.
- [56] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.
- [57] Liangwei Yang, Zhiwei Liu, Yingdong Dou, Jing Ma, and Philip S Yu. Consisrec: Enhancing gnn for social recommendation via consistent neighbor aggregation. In *Proceedings of the 44th international ACM SIGIR conference on Research and development in information retrieval*, pages 2141–2145, 2021.
- [58] Mingqi Yang, Yanming Shen, Rui Li, Heng Qi, Qiang Zhang, and Baocai Yin. A new perspective on the effects of spectrum in graph neural networks. In *International Conference on Machine Learning*, pages 25261–25279. PMLR, 2022.
- [59] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.
- [60] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Universal domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2720–2729, 2019.
- [61] Wangjie You, Pei Guo, Juntao Li, Kehai Chen, and Min Zhang. Efficient domain adaptation for non-autoregressive machine translation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 13657–13670, 2024.
- [62] 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.
- [63] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. *arXiv preprint arXiv:1702.08811*, 2017.
- [64] Qin Zhang, Zelin Shi, Xiaolin Zhang, Xiaojun Chen, Philippe Fournier-Viger, and Shirui Pan. G2pxy: generative open-set node classification on graphs with proxy unknowns. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 4576–4583, 2023.
- [65] Zhen Zhang, Meihan Liu, and Bingsheng He. Pygda: A python library for graph domain adaptation. *arXiv preprint arXiv:2503.10284*, 2025.
- [66] Zhen Zhang, Meihan Liu, Anhui Wang, Hongyang Chen, Zhao Li, Jiajun Bu, and Bingsheng He. Collaborate to adapt: Source-free graph domain adaptation via bi-directional adaptation. In *Proceedings of the ACM on Web Conference 2024*, pages 664–675, 2024.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Please refer to the abstract and Section [1](#).

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Please refer to Section [6](#).

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: Our paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: Please refer to Section 4 and our code URL released in the abstract.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The data and code are available at the URL provided in the abstract.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so No is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Please refer to Section 4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Please refer to Table 2 and Table 3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Please refer to Appendix A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We make sure the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Please refer to Section 6.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have cited necessary assets.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: Our paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

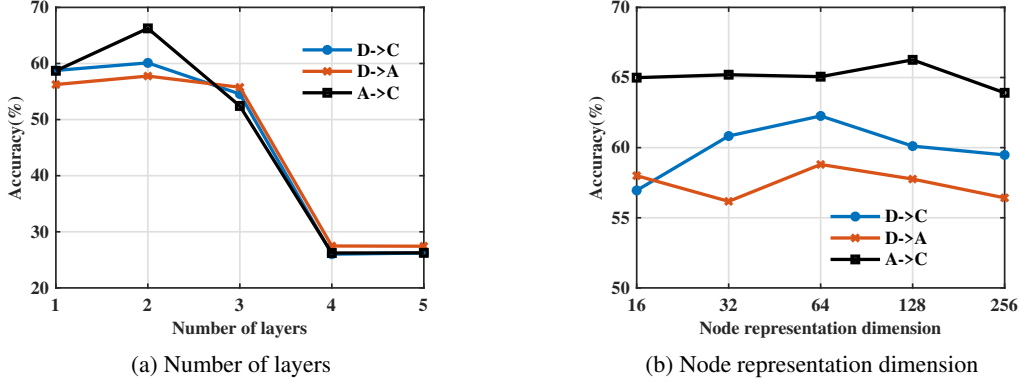


Figure 4: Accuracy under different layers and dimension.

Table 7: Classification H-score with different components.

Architectures	D→C	D→A	A→C
GraphRTA _{w/o MR}	59.86±0.38	56.31±0.68	64.93±0.24
GraphRTA _{w/o GR}	60.34±0.62	56.71±1.29	62.95±1.97
GraphRTA	62.33±1.53	56.91±2.50	66.33±1.69

A Running Environment

Our experiments are conducted on a Linux server with 2 AMD EPYC 7543 CPU@2.80GHz, 512G RAM and one NVIDIA A100-SXM4-80GB GPU. The proposed model is implemented with Pytorch 1.13.1 in Python 3.8 using Pytorch Geometric 2.4.0.

B Additional Ablation Studies and Analyses

In this section, we conduct a series of ablation studies to comprehensively assess the effectiveness of our proposed GraphRTA framework.

Sensitivity to Two Key Hyper-Parameters. We begin by analyzing the model’s sensitivity to two critical hyper-parameters: the number of layers L and the node representation dimension d . As shown in Figure 4, model performance initially improves with an increase in the number of layers but starts to degrade beyond a certain point (i.e., 2 layers). This decline is attributed to overfitting, as deeper layers may overly adapt to the training data while failing to generalize effectively. In contrast, the model exhibits consistent robustness to variations in the node representation dimension, highlighting its ability to perform well across a range of dimensional configurations.

Effectiveness of Model and Graph Reprogramming. Next, we further investigate the individual contributions of the model reprogramming (i.e., MR) and graph reprogramming (i.e., GR) components. Results presented in Table 7 provide insights into the impact of excluding these components. Specifically, the configuration labeled as “w/o MR” excludes the model reprogramming module, while “w/o GR” omits the graph reprogramming module. A configuration without either component demonstrates the most significant performance degradation, underscoring the necessity of incorporating both modules. These findings highlight that the dual reprogramming strategy is critical for mitigating domain shifts and improving model adaptability.

Table 8: H-score with different graph reprogramming strategies.

Architectures	D→C	D→A
GraphRTA _{w SUBLIME [31]}	59.72±1.03	54.28±1.48
GraphRTA _{w SLAPS [7]}	60.65±0.65	55.47±1.62
GraphRTA	62.33±1.53	56.91±2.50

Table 9: MMD comparison before and after graph reprogramming.

Methods	A→C	A→D	C→A	C→D	D→A	D→C
Before	0.0381	0.0402	0.0368	0.0399	0.0371	0.0378
After	0.3520	0.3932	0.5081	0.4682	0.3608	0.4477

Table 10: LLM integration for open-set detection.

Methods	Arxiv-I→Arxiv-II	Arxiv-I→Arxiv-III	Arxiv-II→Arxiv-III
A2GNN	45.00±0.24	43.14±0.18	45.18±0.17
A2GNN + LLM-explanations	46.40±0.45	43.34±0.18	48.81±0.49
SDA	42.60±0.15	39.44±0.21	46.03±0.18
SDA + LLM-explanations	44.10±1.75	40.55±0.10	48.50±0.69
GraphRTA	50.79±2.79	46.25±0.40	48.42±1.94
GraphRTA + LLM-explanations	51.56±1.23	47.59±0.48	50.18±0.79

Comparison with Alternative Graph Reprogramming Strategies. We also evaluate two alternative graph reprogramming approaches, SUBLIME [31] and SLAPS [7], both of which employ self-supervised learning techniques. The results, summarized in Table 8, reveal that our approach outperforms these strategies. SUBLIME relies on GNN-based node similarity learning followed by KNN-based sparsification to generate a sparse adjacency matrix. This dependence on nearest neighbors limits its flexibility and performance. SLAPS, on the other hand, employs a denoising auto-encoder loss for graph reconstruction, which introduces constraints that may not generalize well across different datasets. By avoiding such dependencies, our method achieves superior versatility and effectiveness, as demonstrated in the experiments.

How Does Graph Reprogramming Affect the Feature Distribution of Unknown Classes? We measure the distributional divergence between known and unknown class features using the Maximum Mean Discrepancy (MMD) metric. This analysis provides a quantitative assessment of whether our approach enhances the separability of known and unknown samples in the latent space. The results shown in Table 9 indicate that graph reprogramming significantly increases the distributional distance between the two groups after adaptation, suggesting that it effectively promotes clearer feature separation and mitigates overlap between known and unknown domains.

C Potential of LLM Integration for Unknown Class Detection

Integrating large language models (LLMs) into open-set graph domain adaptation could offer promising opportunities, particularly for datasets where nodes are associated with rich textual information. To explore this direction, we investigate the impact of LLM-generated semantic explanations on unknown class detection. Specifically, we adopt the prompt strategy from TAPE [15] to generate semantic explanations for each node in the ogbn-arxiv dataset. For each node, we construct textual features by concatenating the title, abstract, and the corresponding LLM-generated explanation, thereby enriching the semantic content available to the model.

We then evaluate the effectiveness of these enhanced node features using three representative open-set graph adaptation models. Each experiment is conducted five times with different random seeds, and we report the average H-score along with the standard deviation to ensure statistical reliability. The results, summarized in Table 10, show that incorporating LLM-generated explanations consistently improves performance across all evaluated models. This improvement highlights that LLMs can provide complementary semantic signals that help distinguish between known and unknown classes, thereby enhancing the robustness of open-set graph domain adaptation. Overall, these findings suggest that leveraging LLM-generated semantic knowledge can substantially benefit open-set scenarios, especially in text-rich graph datasets. We consider this a promising direction for future research.

D Complexity Analysis

Let \mathcal{G}_s denote the source graph consisting of n_s nodes and e_s edges, \mathcal{G}_t represent the target graph with n_t nodes and e_t edges. Assume that the node representation dimension is d and the graph neural network has L layers. Then, the time complexity associated with encoding the feature representations of both the source and target graphs is given by $\mathcal{O}(Ld^2(n_s + n_t) + Ld(e_s + e_t))$. For the model reprogramming phase, the complexity of sorting the gradient magnitudes is $\mathcal{O}(Ld\log(d))$. In the context of graph reprogramming, the complexity for transforming node features is $\mathcal{O}(n_t d)$, while the refinement of the graph structure takes a complexity of $\mathcal{O}(e_t)$. Furthermore, additional computation arises from the posterior inference process, which is characterized by a complexity of $\mathcal{O}(n_t T)$, where T denotes the number of iterations in the Expectation-Maximization (EM) procedure, assumed to be a constant. Thus, the overall computational complexity of the proposed framework falls within the same order of magnitude as existing methods.

Finally, to enhance clarity about our training procedure, we present a comprehensive step-by-step outline in Algorithm 1.

Algorithm 1 GraphRTA’s Training Strategy

- 1: **Input:** Given a labeled source graph \mathcal{G}_s and an unlabeled target graph \mathcal{G}_t , graph neural network $\Phi = f_w \circ g_\phi$
 - 2: **Output:** Target graph predictions $\mathbf{Y}_t \in \mathbb{R}^{n_t \times |\mathcal{C}_s| + 1}$
 - 3: Randomly initialize weights of f_w and g_ϕ
 - 4: **while** not reached the maximum epochs **do**
 - 5: **for** batch data from source and target graph **do**
 - 6: Fix the parameters of graph reprogramming
 - 7: Conduct model reprogramming with Eq.(2)
 - 8: **end for**
 - 9: Update model as reprogrammed GNN
 - 10: **for** batch data from source and target graph **do**
 - 11: Fix the parameters of model reprogramming
 - 12: Conduct graph reprogramming with Eq.(4)
 - 13: **end for**
 - 14: Update target graph as reprogrammed graph
 - 15: **end while**
 - 16: Compute $\mathbf{Y}_t \in \mathbb{R}^{n_t \times |\mathcal{C}_s| + 1}$ with graph neural network
-