
GraphKeeper: Graph Domain-Incremental Learning via Knowledge Disentanglement and Preservation

Zihao Guo¹, Qingyun Sun^{1*}, Ziwei Zhang¹, Haonan Yuan¹, Huiping Zhuang²,
Xingcheng Fu³, Jianxin Li¹

¹SKLCCSE, School of Computer Science and Engineering, Beihang University

²Shien-Ming Wu School of Intelligent Engineering, South China University of Technology

³Key Lab of Education Blockchain and Intelligent Technology, Guangxi Normal University

{guozh, sunqy, zwzhang, yuanhn, lijx}@buaa.edu.cn

hpzhuang@scut.edu.cn, fuxc@gxnu.edu.cn

Abstract

Graph incremental learning (GIL), which continuously updates graph models by sequential knowledge acquisition, has garnered significant interest recently. However, existing GIL approaches focus on task-incremental and class-incremental scenarios within a single domain. Graph domain-incremental learning (Domain-IL), aiming at updating models across multiple graph domains, has become critical with the development of graph foundation models (GFMs), but remains unexplored in the literature. In this paper, we propose **Graph Domain-Incremental Learning via Knowledge Disentanglement and Preservation (GraphKeeper)**, to address catastrophic forgetting in Domain-IL scenario from the perspectives of embedding shifts and decision boundary deviations. Specifically, to prevent embedding shifts and confusion across incremental graph domains, we first propose the domain-specific parameter-efficient fine-tuning together with intra- and inter-domain disentanglement objectives. Consequently, to maintain a stable decision boundary, we introduce deviation-free knowledge preservation to continuously fit incremental domains. Additionally, for graphs with unobservable domains, we perform domain-aware distribution discrimination to obtain precise embeddings. Extensive experiments demonstrate the proposed GraphKeeper achieves state-of-the-art results with 6.5%~16.6% improvement over the runner-up with negligible forgetting. Moreover, we show GraphKeeper can be seamlessly integrated with various representative GFMs, highlighting its broad applicative potential.

1 Introduction

Graph incremental learning (GIL)[40, 6, 47] aims to continuously update graph models as new graph data arrives, and has attracted increasing attention. Most existing methods target **task-incremental** (Task-IL) and **class-incremental** (Class-IL) settings (Figure 1), where new tasks or classes emerge within a single domain. However, as newly added graphs often come from different domains, the **domain-incremental** (Domain-IL) setting becomes essential. This is especially relevant with the rise of graph foundation models (GFMs), which require integrating diverse graphs from multiple domains to build a comprehensive and evolving knowledge base. Unfortunately, current GIL methods are designed for single-domain scenarios and struggle with Domain-IL. As illustrated in Figure 2, a representative method SSM [45] performs well under Class-IL but fails in the more challenging Domain-IL case. Despite its importance, effective GIL under Domain-IL remains an open problem.

*Corresponding author.

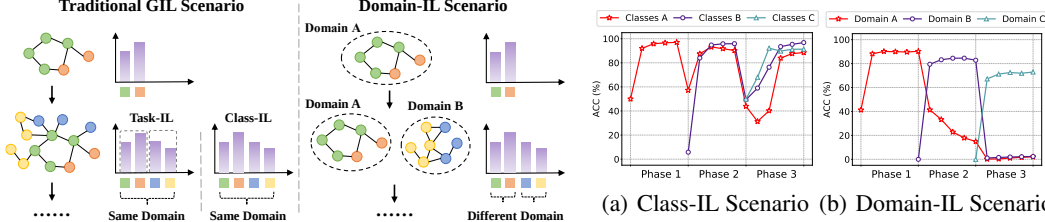


Figure 1: An illustration of traditional GIL (*i.e.*, Task-IL and Class-IL) and Domain-IL scenarios within our representative GIL method, in Class-IL scenario scope studied in this paper.

To tackle this problem, we first analyze the underlying cause of performance degradation in the Domain-IL setting, which is the catastrophic forgetting problem. In typical graph learning models such as GNNs, learning new graphs causes changes in model parameters that manifest in **two aspects**: (1) shifts in the embeddings of previously learned graphs, and (2) deviations in the decision boundary. These changes together lead to the forgetting of prior knowledge. Existing GIL methods attempt to constrain or adapt to such changes, which is relatively feasible in Task-IL and Class-IL scenarios, as the data still resides within a single domain. In contrast, Domain-IL involves substantial structural and semantic discrepancies across domains, making it much harder for GNNs to retain knowledge across diverse graph distributions. This difficulty is also evidenced by the negative transfer observed in multi-domain graph pre-training [2].

Based on the analysis above, we tackle the GIL challenges in Domain-IL scenario from two sides: (1) **Embedding shifts**: *How to learn stable and disentangled representations for different graph domains in GIL?* The embedding shifts introduced by model parameter changes bring the risk of semantic confusion between incremental graphs. To adapt to new domains, more drastic parameter changes occur, making embedding shifts more uncontrollable. (2) **Decision boundary deviations**: *How to effectively retain knowledge from previously learned graph domains?* The deviation in decision boundary is also a significant cause of catastrophic forgetting [51], especially when the model adapts to new domains. Previous GIL methods, which treat embedding learning and prediction as an integrated process, are difficult to constrain the decision boundary effectively.

In this paper, we propose a novel GIL framework **GraphKeeper** to address the catastrophic forgetting in Domain-IL scenario. Specifically: (1) To address the embedding shifts challenge, we propose the domain-specific graph parameter-efficient fine-tuning (PEFT) based on a pre-trained GNN to learn embeddings correspond to graph domains, which ensures parameters for previously learned graph domains remain unaffected when learning new domains. Additionally, we propose intra- and inter-domain disentanglement to avoid confusion in embedding space. (2) To address the decision boundary deviation challenge, we separate the decision module from the embedding model and continuously fit incremental domains through the analytical solution of ridge regression, which prevents drastic changes in the decision boundary while effectively retaining knowledge from previously learned graph domains. (3) Lastly, we perform domain-aware distribution discrimination on graphs with unobservable domain to facilitate the precise embedding and prediction. **Our contributions are:**

- We propose a novel GIL framework GraphKeeper to tackle the catastrophic forgetting in Domain-IL scenario. To the best of our knowledge, we are the first to explore this challenging problem.
- We design the multi-domain graph disentanglement and deviation-free knowledge preservation mechanism, which enables GraphKeeper to learn stable and disentangled representations between different incremental graph domains and maintains stable decision boundary without deviations.
- GraphKeeper achieves 6.5%~16.6% improvement over the runner-up with negligible forgetting. Besides, it can be seamlessly integrated with existing representative GFMs.

2 Related Work

2.1 Graph Incremental Learning

Existing graph incremental learning methods can be categorized into regularization-based, memory replay-based, and parameter isolation-based. Regularization-based methods [16, 31, 4] aim to identify

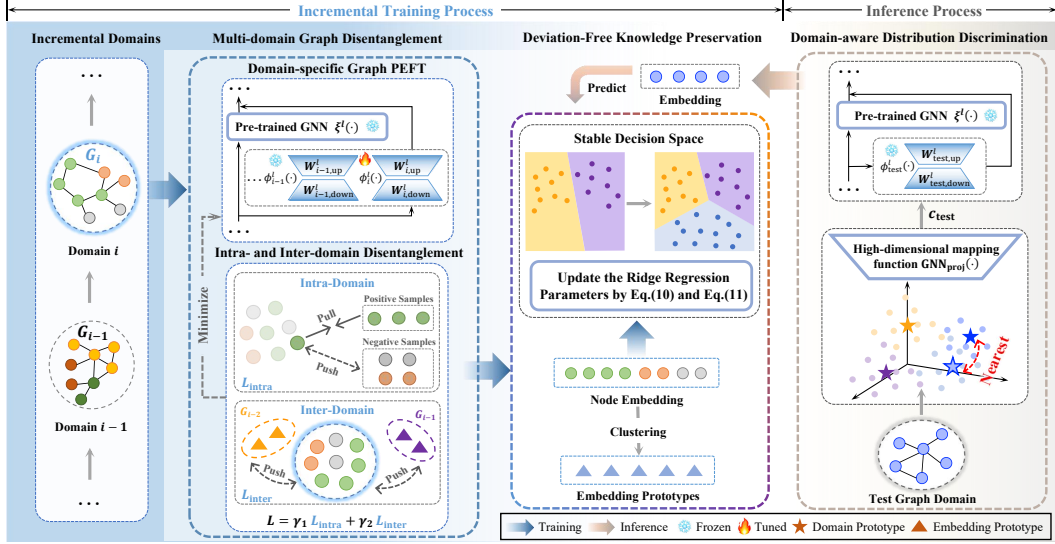


Figure 3: The overall framework of GraphKeeper. (1) The Multi-domain Graph Disentanglement isolates parameters of different graph domains through the domain-specific graph PEFT to prevent embedding shifts, and disentangles embeddings both intra- and inter-domain to prevent confusion. (2) The Deviation-Free Knowledge Preservation continuously fits incremental graph domains while maintaining a stable decision boundary without deviations. (3) The Domain-aware Distribution Discrimination matches graphs with unobservable domain to prototypes of previous domains. Then our method embeds them with corresponding domain-specific PEFT module, and make predictions.

parameters that are important for previous tasks and constrain their changes through penalty terms. Memory replay-based methods [52, 45, 34, 33, 46, 17, 42, 22] are widely recognized for their effectiveness, explicitly preserving representative subsets of graphs from previous tasks and replaying them while learning new tasks. Parameter isolation-based methods [44, 41, 23] adapt to graph data in new tasks through adding trainable parameters. Despite the impressive performance in Task-IL and Class-IL scenarios, these methods do not consider the unique challenges in the Domain-IL scenario.

2.2 Multi-domain Graph Learning

Cross-domain Graph Pre-training. A promising approach for constructing GFM is to pre-train on a large corpus of graphs, and transfer the pre-trained knowledge to downstream graphs [48]. Although existing cross-domain graph pre-training methods [49, 39, 37, 50] can effectively adapt to downstream graphs, the model becomes fixed once pre-training is complete. Continuously learning knowledge from new graphs is essential [47], especially when dealing with multiple downstream graph domains, but it faces the problem of catastrophic forgetting. Our method addresses the challenging Domain-IL scenario in GIL and can be integrated with various graph pre-training methods.

Graph Domain Adaptation. Graph domain adaptation (GDA) [28] aims to leverage knowledge from the source domain to improve the performance in the target domain. Although multiple domains are considered, GDA and Domain-IL scenario in GIL are fundamentally different [36]. For GDA, the source and target domains are simultaneously accessible in most cases, with the focus primarily on target domain performance. In comparison, for Domain-IL scenario, only the domain of current task is accessible, and the objective is to maintain performance across all previously learned domains.

3 Preliminary and Problem Formulation

Graph Incremental Learning. Given a task sequence $S = \{G_1, \dots, G_T\}$, the objective of GIL is to sequentially learn the graphs while ensuring that the model parameters θ_t after learning the current task maintain satisfactory performance on all previous tasks. The objective can be formulated as:

$$\theta_t^* = \arg \min_{\theta} \frac{1}{t} \sum_{i=1}^t \mathcal{L}_i(\theta), \quad (1)$$

where $\mathcal{L}_i$ is i -th task loss. Previous $t-1$ tasks and their data are inaccessible when learning the t -th.

Problem Formulation. We primarily focus on GIL in the Domain-IL scenario, *i.e.*, graphs G_t in $\mathcal{S}$ belong to different domains. This contrasts with Task-IL and Class-IL, where all tasks belong to the same domain. Due to a significant domain gap in both structure and feature semantics, conflicts arise among incremental domains in Domain-IL scenario. Our goal is to learn an encoder $\Phi : \mathcal{V} \mapsto \mathcal{Z}$, where different domains are disentangled in the embedding space $\mathcal{Z}$, representing their respective semantics, and to learn a decoder $\Psi : \mathcal{Z} \mapsto \mathcal{P}$ that accurately maps the disentangled embedding of different graph domains to the label space $\mathcal{P}$, while continuously updating without forgetting.

4 Method

In this section, we present GraphKeeper to address the catastrophic forgetting in the Domain-IL scenario. Our key insight is to decompose the cause of catastrophic forgetting into embedding shifts and decision boundary deviations. The framework is shown in Figure 3. In brief, we first propose multi-domain graph disentanglement (Sec. 4.1) to prevent embedding shifts and confusion across incremental graph domains. Then, we introduce deviation-free knowledge preservation (Sec. 4.2) to continuously fit incremental domains while maintaining a stable decision boundary without deviations. Lastly, we design a domain-aware distribution discrimination mechanism (Sec. 4.3) for graphs with unobservable domains to obtain precise embeddings.

4.1 Multi-domain Graph Disentanglement

To prevent confusion across incremental graph domains, we introduce multi-domain graph disentanglement, which isolates parameters between different graph domains through the domain-specific graph PEFT to avoid embedding shifts, and disentangles embeddings both intra- and inter-domain.

Multi-domain Graph Feature Alignment. Prior to further processing, we first align the feature dimension for graphs from different domains:

$$\tilde{\mathbf{F}}_i = \text{Proj}(\mathbf{F}_i), \quad \tilde{\mathbf{F}}_i \in \mathbb{R}^{|G_i| \times \bar{d}}, \quad (2)$$

where $\mathbf{F}_i$ is the original node features of G_i , $|G_i|$ denotes the number of nodes, $\text{Proj}(\cdot)$ represents a projection operation, and $\bar{d}$ represents the unified feature projection dimension. The projection operation $\text{Proj}(\cdot)$ is realized through truncated singular value decomposition (SVD) [29].

In addition to this dimension alignment, the feature semantics of different domains remain fundamentally distinct. On the one hand, drastic changes in model parameters can occur when adapting to new domains, causing the embedding shifts across graph domains. On the other hand, the embeddings of different graph domains may exhibit unintentional overlap that contradicts their semantics, leading to confusion in prediction. Both factors can lead to the catastrophic forgetting in the Domain-IL scenario. To address these issues, we introduce two core modules as follows.

Domain-specific Graph PEFT. To prevent potential catastrophic forgetting of previous domains due to drastic parameter changes when adapting to new domains, we aim to isolate parameters of different domains. Inspired by the success of low-rank adaptation (LoRA) [10, 14, 9, 38], we propose graph domain-specific PEFT. Specifically, for graph domain in the task sequence, we equip the pre-trained GNN with a LoRA module. For the i -th domain, the l -th layer of the model ψ_i is calculated as:

$$\mathbf{h}^l = \xi^l(\mathbf{h}^{l-1}, \mathbf{W}_i^l) + \phi_i^l(\mathbf{h}^{l-1}, \mathbf{W}_{i,\text{down}}^l \mathbf{W}_{i,\text{up}}^l), \quad (3)$$

where $\xi^l(\cdot)$ denotes the output of the frozen GNN, $\phi_i^l(\cdot)$ denotes the LoRA module of the i -th graph domain, and $\mathbf{W}_{i,\text{down}}^l \in \mathbb{R}^{d^{l-1} \times r}$ and $\mathbf{W}_{i,\text{up}}^l \in \mathbb{R}^{r \times d^l}$ are two low-rank learnable parameter matrices, *i.e.*, r is much smaller than the parameter matrix dimension d^{l-1} and d^l . When learning a new graph domain, we freeze previous domain-specific LoRA parameters, ensuring that learned graph domains remain stable in the embedding space and do not shift with the adaptation to the new domain.

Intra- and Inter-domain Disentanglement. To distinguish semantic differences between different graph domains as well as different classes within the same graph domain, we propose objectives for both intra- and inter-domain disentanglement, aiming to prevent confusion.

(1) Intra-domain Disentanglement. Within a single domain, our goal is to enhance the discriminability of node embeddings across different classes. To achieve this, we introduce an intra-domain

disentanglement objective based on contrastive learning. Specifically, we first generate augmented views for a given G_i :

$$G_i^{\text{aug}} = \text{Aug}(G_i), \quad \mathbf{X}_i = \psi_i(G_i), \quad \mathbf{X}_i^{\text{aug}} = \psi_i(G_i^{\text{aug}}), \quad (4)$$

where $\text{Aug}(\cdot)$ denotes feature and structure augmentation, $\mathbf{X}_i$ and $\mathbf{X}_i^{\text{aug}}$ are learned node embeddings corresponding to the original view and the augmented view, respectively, and ψ_i represents the domain-specific PEFT module in Eq. (3). For each node, we define the positive samples $\mathcal{S}^{\text{pos}}$ as the set of nodes belonging to the same class, and the negative samples $\mathcal{S}^{\text{neg}}$ as the nodes from different classes. The **intra-domain** disentanglement objective can be formulated as:

$$\mathcal{L}_{\text{intra}} = - \sum_{j=1}^{|G_i|} \log \frac{\sum_{o \in \mathcal{S}_j^{\text{pos}}} \exp(\text{sim}(\mathbf{x}_j, \mathbf{x}_o^{\text{aug}}))}{\sum_{o' \in \mathcal{S}_j^{\text{pos}} \cup \mathcal{S}_j^{\text{neg}}} \exp(\text{sim}(\mathbf{x}_j, \mathbf{x}_{o'}^{\text{aug}}))}, \quad (5)$$

where $\text{sim}(\cdot, \cdot)$ denotes cosine similarity. This objective promotes semantic similarity among nodes of the same class while maintaining distinction between nodes of different classes in a single domain.

(2) Inter-domain Disentanglement. With the intra-domain disentanglement, the embeddings for node of each class within a domain are compact to ensure clear discriminability between classes. However, in the Domain-IL scenario, due to the feature semantics gap across different graph domains, there exist risks of unintentional overlap in the embedding space across domains. To prevent catastrophic forgetting caused by this confusion between domains, we next introduce the representation scattering based inter-domain disentanglement objective.

Through freezing of domain-specific LoRA parameters, the embeddings of previous domains remain stable. Consequently, the inter-domain disentanglement objective can be defined as ensuring that the current domain is sufficiently distant from previous domains in the embedding space. However, previous graphs are not accessible in GIL. Considering that the embeddings within each domain are compact and unchanged, after learning an incremental domain, we capture representative embedding prototypes through clustering, which represent the distribution of domains in the embedding space. Therefore, the **inter-domain** disentanglement objective can be expressed by pushing the samples of the current domain away from the embedding prototypes of all previous domains:

$$\mathcal{L}_{\text{inter}} = \frac{1}{|G_i|} \sum_{j=1}^{|G_i|} \sum_{k=1}^{|P|} \frac{1}{\|\mathbf{x}_j - \mathbf{P}_k\|_2^2 + \epsilon}, \quad (6)$$

where $\mathbf{P}$ represents the set of embedding prototypes, and ϵ is a small constant. By minimizing the inter-domain disentanglement objective, the semantics of different domains are clearly separated, with no interference between them.

Overall Objective. The overall optimization objective can be formulated as:

$$\mathcal{L} = \gamma_1 \mathcal{L}_{\text{intra}} + \gamma_2 \mathcal{L}_{\text{inter}}, \quad (7)$$

where γ_1 and γ_2 are hyperparameters for trading-off. By minimizing Eq. (7), we can obtain highly discriminative node embeddings without confusion for both across different domains and across classes within each domain, thereby facilitating accurate knowledge preservation.

4.2 Deviation-Free Knowledge Preservation

Previous GIL methods usually integrate embedding learning and classification through an end-to-end training framework. However, when learning new tasks, the classifier is updated together with the embedding model through back-propagation, leading to the deviation in the decision boundary. Although previous methods have attempted to solve this issue through techniques like memory replay, the limited quantity of memory data is insufficient when faced with the drastic parameter changes required to adapt to new domains.

Ridge Regression for Stable Classifier Updates. We aim to solve the decision boundary deviation issue by separating the classifier from the embedding model, thereby maintaining a stable decision boundary that remains unaffected by new incremental domains, which also aligns with the stable embeddings we introduce in the last section. To this end, we introduce a ridge regression-based knowledge preservation mechanism, which does not require gradient updates through back-propagation,

thus avoiding the catastrophic forgetting caused by decision boundary deviations, inspired by [54, 53]. Specifically, for the first incremental domain, denote the stable and disentangled embedding as $\mathbf{X}_1$. Then, we fit the class labels $\mathbf{Y}_1$ through ridge regression, which is equivalent to:

$$\arg \min_{\mathbf{W}_1} \|\mathbf{Y}_1 - \mathbf{X}_1 \mathbf{W}_1\|_F^2 + \lambda \|\mathbf{W}_1\|_F^2, \quad (8)$$

where $\|\cdot\|_F^2$ indicates the Frobenius norm and λ is the regularization coefficient. We can obtain a closed-form solution for the optimal weight as $\mathbf{W}_1 = (\mathbf{X}_1^\top \mathbf{X}_1 + \lambda \mathbf{I})^{-1} \mathbf{X}_1^\top \mathbf{Y}_1$.

Similarly, for the i -th incremental domain, the optimal ridge regression parameters $\mathbf{W}_i$, which can also retain the knowledge of the previous i domains, can be calculated as:

$$\mathbf{W}_i = (\mathbf{X}_{(1:i)}^\top \mathbf{X}_{(1:i)} + \lambda \mathbf{I})^{-1} \mathbf{X}_{(1:i)}^\top \mathbf{Y}_{(1:i)}. \quad (9)$$

Recursive Update without Historical Data Access. However, historical domains are not accessible in GIL, *i.e.*, we cannot directly calculate Eq. (9). To solve this, we recursively update the parameter matrix only using data from the current graph domain:

$$\mathbf{W}_i = [\mathbf{W}_{i-1} - \mathbf{M}_i \mathbf{X}_i^\top \mathbf{X}_i \mathbf{W}_{i-1} \parallel \mathbf{M}_i \mathbf{X}_i^\top \mathbf{Y}_i], \quad (10)$$

where $\parallel$ indicates the concatenation and $\mathbf{M}_i$ is an intermediate matrix recursively updated as:

$$\mathbf{M}_i = \mathbf{M}_{i-1} - \mathbf{M}_{i-1} \mathbf{X}_i^\top (\mathbf{I} + \mathbf{X}_i \mathbf{M}_{i-1} \mathbf{X}_i^\top)^{-1} \mathbf{X}_i \mathbf{M}_{i-1}, \quad (11)$$

where $\mathbf{M}_1 = (\mathbf{X}_1^\top \mathbf{X}_1 + \lambda \mathbf{I})^{-1}$. The detailed derivations are provided in Appendix A.

Through Eq. (10) and Eq. (11), we can precisely update the model parameters without using historical data while guaranteeing the optimal solution as Eq. (9). For inference, given a graph G_k from the k -th domain, we can obtain the predicted node labels as:

$$\mathbf{Y}_{\text{predict}} = \text{Softmax}(\mathbf{X}_k \mathbf{W}), \quad (12)$$

where $\mathbf{X}_k$ is calculated by Eq. (3). For graph G_{test} with unobservable domain, the corresponding domain-specific PEFT module ϕ_{test} in Eq. (3) is determined by the domain-aware distribution discrimination in Sec. 4.3. Since both the embeddings and the decision boundary remain stable, this significantly reduces the risk of catastrophic forgetting caused by drastic parameter changes.

4.3 Domain-aware Distribution Discrimination

To obtain precise embeddings, it is necessary to match the graphs from different domains with the corresponding PEFT module, which is easy to achieve for the training process, as the domains of the training graphs are observable. However, for the test graph with an unobservable domain, it is difficult to directly match with the corresponding PEFT module. Therefore, we propose a simple yet effective method for domain-aware distribution discrimination. The core idea is to match test graphs with prototypes of different domains.

Primarily, the features across multiple graph domains are at risk of being excessively similar or even overlapping, as shown in Figure G.3, which may cause the test graph to be matched with the wrong domain prototype. To reduce the risk of prototype confusion, we aim to obtain domain prototypes with sufficient discriminability. Specifically, we first transform the features into a high-dimensional space through a randomly initialized and frozen GNN for random mapping:

$$\hat{\mathbf{F}} = \text{GNN}_{\text{proj}}(G, \mathbf{W}_{\text{random}}), \quad (13)$$

where $\hat{\mathbf{F}}$ denotes the transformed features. Though the GNN_{proj} is not trained, this high-dimensional random mapping [30, 19] helps to separate prototypes between different domains, particularly when utilizing the feature distance as the criterion for domain distribution discrimination. The confusion matrix of domain prototypes before and after mapping by Eq. (13) is shown in Figure E.2.

Then, we determine the domain of the test graph by associating it with the nearest domain prototype. Specifically, we calculate the correlation between the test graph and domain prototypes based on the distances, and the domain with the highest correlation is chosen as the domain of the test graph:

$$\mathbf{D}_k = \frac{1}{|G_k|} \sum_{i=1}^{|G_k|} \hat{\mathbf{F}}_i^k, \quad c_{\text{test}} = \arg \max_k (\exp(-\|\mathbf{D}_{\text{test}} - \mathbf{D}_k\|_2^2)), \quad (14)$$

where c_{test} indicates the domain index, D_{test} and D_k are the prototype of the test and k -th incremental graph domain, obtained through average pooling. Our high-dimensional random mapping guarantees a sufficiently distance between the prototypes of different domains, which results in a minimal correlation between the prototype of the test graph and the prototypes of other domains, effectively preventing confusion in domain discrimination.

The overall algorithm and complexity analysis of GraphKeeper are shown in Appendix B.

5 Experiments

5.1 Experimental setup

Datasets and Baselines. To evaluate GraphKeeper², we conduct comprehensive experiments on 15 real-world datasets, where detailed descriptions are in Appendix C.1. We choose a variety of baselines, including general IL methods EWC [13], MAS [1], GEM [18], LWF [15] and representative GIL methods TWP [16], ER-GNN [52], SSM [45], DeLoMe [22], PDGNNs [42], TPP [23]. Additionally, we include two self-designed baselines: Fine-Tune, which continuously training without any IL techniques, and Joint, which utilizes all previous graph domains for training.

Implementation Details. We implement all methods under the GIL benchmark [43]. All results are averaged over 5 independent runs for a fair comparison. More details are provided in Appendix D.

Experimental Settings. To comprehensively evaluate in the Domain-IL scenario, we adopt multiple groups of graph domains. Due to the significant differences between domains, the performance varies to some extent under different incremental orders. Therefore, we set multiple incremental orders for each group and report the average results. Further details are provided in Appendix C.2.

Evaluation Metrics. We adopt two commonly used metrics in GIL: Average Accuracy (AA) and Average Forgetting (AF). Higher AA indicates better performance, and higher AF indicates less forgetting, as detailed in Appendix C.3.

5.2 Performance on Domain-IL Scenario

Analysis. We can observe the following findings from the results of Domain-IL scenario in Table 1:

- While existing GIL methods can handle Task-IL and Class-IL scenarios, they show a significant decline in performance in Domain-IL scenario, which indicates the inability of traditional GIL methods to adapt to the more challenging Domain-IL scenario. In contrast, GraphKeeper achieves state-of-the-art results with 6.5%~16.6% improvement over the runner-up and with negligible forgetting, which benefits from our disentangled and stable representation for different incremental graph domains through the domain-specific graph PEFT to prevent confusion, and performing high-fidelity knowledge preservation.
- GraphKeeper outperforms the Joint baseline even though it has access to all previous graph domains. The results indicate that a single GNN struggles to effectively integrate knowledge from multiple domains. In contrast, GraphKeeper tackles this issue by isolating parameters of different domains through the domain-specific graph PEFT.
- DeLoMe and PDGNNs exhibit relatively passable performance. These methods utilize SGC [35] and APPNP [8] as their backbones, respectively, which sacrifices plasticity to mitigate forgetting. However, this limits their flexibility and applicability. We also try replacing their backbones as GCN and the results are summarized in Table E.3, showing a significant decline in performance.
- TPP utilizes prompts to adapt to different incremental graphs. However, its capability for modeling incremental graph domains is relatively constrained.

5.3 Studies of Integration with GFMs

To further explore the potential of GraphKeeper, we integrate it with two representative GFMs, GCOPE [49] and MDGPT [39], to equip them with the continuous updating capability. Specifically, we first pre-train the GFMs and then incorporate GraphKeeper with them to evaluate the performance in the **few-shot** Domain-IL scenario. The results are summarized in Table 2.

²<https://github.com/RingBDStack/GraphKeeper>

Table 1: The average performance (%) across different incremental orders in Domain-IL scenario. AA indicates Average Accuracy and AF indicates Average Forgetting. $\pm$ represents the standard deviation. The best results are indicated in **bold** and the runner-ups are underlined.

Method	Group 1		Group 2		Group 3		Group 4		Group 5		Group 6	
	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$
Fine-Tune	23.9 $\pm$ 0.1	-72.7 $\pm$ 0.3	17.1 $\pm$ 0.4	-74.4 $\pm$ 0.7	20.9 $\pm$ 0.1	-81.3 $\pm$ 0.2	21.6 $\pm$ 0.6	-77.8 $\pm$ 1.1	19.7 $\pm$ 0.1	-77.8 $\pm$ 0.3	18.6 $\pm$ 0.6	-77.9 $\pm$ 0.6
Joint	66.6 $\pm$ 0.8	-	67.6 $\pm$ 0.4	-	78.0 $\pm$ 0.3	-	75.7 $\pm$ 0.8	-	74.5 $\pm$ 0.3	-	71.5 $\pm$ 0.5	-
EWC	23.3 $\pm$ 0.1	-72.0 $\pm$ 0.3	17.3 $\pm$ 0.4	-74.8 $\pm$ 1.0	20.8 $\pm$ 0.3	-80.9 $\pm$ 0.2	22.4 $\pm$ 0.6	-76.1 $\pm$ 0.8	20.6 $\pm$ 0.2	-77.9 $\pm$ 0.6	18.0 $\pm$ 0.3	-76.6 $\pm$ 0.9
MAS	24.0 $\pm$ 2.4	-70.5 $\pm$ 4.3	16.8 $\pm$ 5.1	-69.8 $\pm$ 5.0	18.9 $\pm$ 2.5	-77.4 $\pm$ 3.0	21.1 $\pm$ 5.0	-75.5 $\pm$ 7.7	18.9 $\pm$ 3.4	-74.0 $\pm$ 3.2	19.8 $\pm$ 2.3	-71.5 $\pm$ 1.1
GEM	23.3 $\pm$ 3.2	-71.7 $\pm$ 5.6	20.1 $\pm$ 3.4	-69.2 $\pm$ 2.8	21.7 $\pm$ 2.3	-80.6 $\pm$ 2.1	21.1 $\pm$ 4.2	-77.4 $\pm$ 4.3	20.0 $\pm$ 2.5	-77.4 $\pm$ 2.1	19.4 $\pm$ 1.8	-76.1 $\pm$ 0.7
LWF	23.7 $\pm$ 2.9	-72.5 $\pm$ 1.3	17.0 $\pm$ 0.7	-73.8 $\pm$ 0.7	20.2 $\pm$ 0.2	-80.7 $\pm$ 0.2	20.1 $\pm$ 0.2	-78.6 $\pm$ 0.6	19.8 $\pm$ 0.2	-77.9 $\pm$ 0.4	18.8 $\pm$ 0.3	-76.0 $\pm$ 0.7
TWP	23.5 $\pm$ 0.1	-71.0 $\pm$ 0.5	16.0 $\pm$ 3.3	-71.0 $\pm$ 1.6	20.0 $\pm$ 2.2	-79.0 $\pm$ 0.7	20.2 $\pm$ 2.0	-79.0 $\pm$ 1.0	18.8 $\pm$ 3.4	-74.4 $\pm$ 1.2	18.2 $\pm$ 2.1	-75.2 $\pm$ 1.0
ER-GNN	23.3 $\pm$ 1.4	-64.8 $\pm$ 3.8	22.7 $\pm$ 2.2	-66.3 $\pm$ 2.9	28.7 $\pm$ 3.0	-66.3 $\pm$ 3.7	23.9 $\pm$ 1.7	-72.9 $\pm$ 2.8	24.8 $\pm$ 2.7	-68.8 $\pm$ 3.9	24.7 $\pm$ 4.7	-67.9 $\pm$ 5.8
SSM	24.1 $\pm$ 7.8	-36.4 $\pm$ 13.1	22.1 $\pm$ 3.6	-35.7 $\pm$ 5.8	15.6 $\pm$ 3.3	-31.0 $\pm$ 4.1	25.3 $\pm$ 2.6	-45.7 $\pm$ 3.8	25.7 $\pm$ 2.8	-39.3 $\pm$ 3.3	16.1 $\pm$ 5.7	-33.9 $\pm$ 4.8
TPP	52.6 $\pm$ 1.8	0.0 $\pm$ 0.0	49.7 $\pm$ 1.5	0.0 $\pm$ 0.0	57.1 $\pm$ 2.5	0.0 $\pm$ 0.0	53.0 $\pm$ 2.9	0.0 $\pm$ 0.0	56.7 $\pm$ 1.0	0.0 $\pm$ 0.0	48.3 $\pm$ 2.4	0.0 $\pm$ 0.0
DeLoMe	49.3 $\pm$ 0.1	-7.7 $\pm$ 0.1	<u>58.2$\pm$5.1</u>	-9.6 $\pm$ 8.5	<u>70.2$\pm$4.1</u>	-1.0 $\pm$ 0.1	<u>73.4$\pm$4.3</u>	-1.1 $\pm$ 0.2	63.2 $\pm$ 5.7	-3.3 $\pm$ 7.1	<u>64.2$\pm$2.2</u>	-4.8 $\pm$ 5.9
PDGNNs	52.4 $\pm$ 0.5	-15.8 $\pm$ 0.6	53.5 $\pm$ 0.3	-7.4 $\pm$ 0.3	65.5 $\pm$ 0.6	-11.4 $\pm$ 0.7	65.5 $\pm$ 0.5	-7.2 $\pm$ 0.8	<u>64.3$\pm$0.3</u>	-8.2 $\pm$ 0.3	60.2 $\pm$ 0.2	-7.2 $\pm$ 0.4
GraphKeeper	69.2$\pm$0.3	-0.4 $\pm$ 0.2	73.1$\pm$1.1	-2.8 $\pm$ 0.7	80.6$\pm$0.4	0.0 $\pm$ 0.1	79.9$\pm$0.5	1.0 $\pm$ 0.5	75.5$\pm$0.9	0.1 $\pm$ 0.5	77.5$\pm$0.7	-0.1 $\pm$ 0.5

Table 2: The results of integrating GraphKeeper with existing GFMs in the **few-shot** Domain-IL senario. AA indicates Average Accuracy and AF indicates Average Forgetting. $\pm$ represents the standard deviation. Better results are indicated in **bold**.

Method	Group 1		Group 2		Group 3		Group 4		Group 5		Group 6	
	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$
GCOPE	20.6 $\pm$ 0.9	-53.5 $\pm$ 2.1	10.5 $\pm$ 0.7	-36.7 $\pm$ 1.0	13.2 $\pm$ 1.2	-47.7 $\pm$ 2.9	12.6 $\pm$ 1.1	-51.0 $\pm$ 1.8	13.6 $\pm$ 1.4	-41.7 $\pm$ 2.9	12.6 $\pm$ 0.9	-43.7 $\pm$ 1.1
GCOPE+Ours	56.8$\pm$1.9	0.2 $\pm$ 0.3	36.6$\pm$1.4	0.3 $\pm$ 0.3	47.4$\pm$4.1	-1.6 $\pm$ 2.8	51.6$\pm$1.6	-0.4 $\pm$ 0.6	44.3$\pm$4.0	0.8 $\pm$ 0.5	44.9$\pm$0.9	-0.8 $\pm$ 0.8
MDGPT	19.9 $\pm$ 1.3	-59.4 $\pm$ 2.1	10.7 $\pm$ 0.5	-42.7 $\pm$ 2.2	12.1 $\pm$ 1.5	-50.4 $\pm$ 2.1	12.8 $\pm$ 1.4	-52.9 $\pm$ 3.1	12.8 $\pm$ 1.3	-47.6 $\pm$ 5.9	12.3 $\pm$ 0.5	-48.8 $\pm$ 3.3
MDGPT+Ours	59.7$\pm$1.2	-1.5 $\pm$ 0.7	32.7$\pm$1.0	-0.4 $\pm$ 0.7	49.5$\pm$2.1	-2.8 $\pm$ 1.1	62.7$\pm$1.7	-1.4 $\pm$ 1.2	50.9$\pm$1.6	-0.7 $\pm$ 0.4	43.9$\pm$1.1	-0.2 $\pm$ 1.0

Analysis. We observe that the original GFMs exhibit low AA and high AF. The high AF indicates that they perform well when trained separately on each graph domain, demonstrating strong few-shot capability. However, they lack the continuous updating capability with severe catastrophic forgetting issues, resulting in low AA. After integrating the GraphKeeper, they show significantly higher AA with negligible forgetting. This demonstrates that GraphKeeper can be seamlessly integrated into existing GFMs, combining their few-shot strength with the continuous updating capability to construct more powerful GFMs. Additionally, GraphKeeper does not require memory replay, which avoids the potential memory explosion problem, particularly on a large corpus of graphs.

5.4 Ablation Study

Ablation Study. We analyze three variants of GraphKeeper. The results are summarized in Figure 4 and Figure E.1.

- **GraphKeeper (w/o DT):** We replace the disentanglement objectives in Eq. (7) with the vanilla contrastive objective that ignores domain-specific semantics.
- **GraphKeeper (w/o PEFT):** We remove the domain-specific graph PEFT module and perform continuous training on the pre-trained GNN model.
- **GraphKeeper (w/o KP):** We remove the deviation-free knowledge preservation module and utilize an MLP as the classifier for end-to-end training.

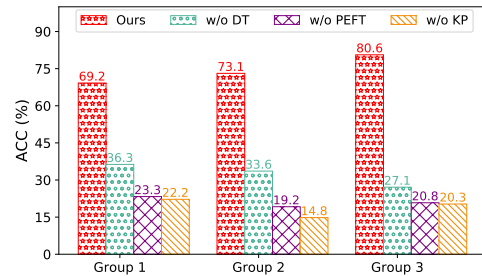


Figure 4: The ablation study results.

Analysis. Overall, we observe that removing any key component of GraphKeeper results in a noticeable drop in performance, highlighting the importance of each module. (1) For GraphKeeper (w/o

DT), removing both intra-domain and inter-domain disentanglement objectives significantly weakens the model’s ability to distinguish between classes and domains. This results in entangled embeddings that cause confusion across domains and hinder generalization. **(2)** For GraphKeeper (w/o PEFT), excluding parameter-efficient fine-tuning removes the isolation of domain-specific parameters. As a result, adaptation to new domains induces large parameter updates that affect previously learned domains, leading to embedding shifts and inconsistency with the preserved knowledge. **(3)** For GraphKeeper (w/o KP), the absence of knowledge preservation allows the classifier parameters to drift throughout training. This drift causes deviations in the decision boundary, disrupting the alignment between the learned embeddings and their corresponding labels. These results confirm that both embedding stability and decision boundary consistency are essential to mitigate catastrophic forgetting in the Domain-IL setting, and that GraphKeeper achieves this through the synergistic effect of disentanglement, parameter isolation, and knowledge preservation.

5.5 Visualisation of Disentangled Embeddings

Embedding confusion is an important cause of catastrophic forgetting in Domain-IL scenario. To provide a more intuitive comparison of the advantages of GraphKeeper, we visualize the embeddings of different incremental graph domains with t-SNE [32] in Figure G.3 and Figure G.4.

Analysis. As shown in Figure G.3, the embeddings produced by GraphKeeper are highly compact with clear boundaries between domains, thanks to the introduction of multi-domain graph disentanglement, which effectively reduces interference across incremental domains. In contrast, PDGNNs and DeLoMe exhibit noticeable domain overlap. Although they replay previous data, they still struggle to avoid conflicts between different domains. To further investigate the evolutions, we also visualize previously learned domains at different learning stages (Figure G.4). It can be observed that the embeddings of PDGNNs and DeLoMe become increasingly entangled over time, while those of GraphKeeper maintain high separability throughout, further corroborating our analysis.

5.6 Hyperparameters Analysis

We analyze the impact of hyperparameters in Figure 5, where r denotes the rank of the parameter matrix in domain-specific graph PEFT module, and γ_1/γ_2 plays the trade-off role between intra- and inter-domain disentanglement objectives.

Analysis. **(1)** For hyperparameter r , a small value limits the number of tunable parameters, making it difficult for the model to adapt effectively to new graph domains. As r increases, the performance improves and achieves strong results while still using far fewer parameters than full-model tuning. **(2)** For hyperparameters γ_1 and γ_2 , when the ratio γ_1/γ_2 decreases, meaning the inter-domain disentanglement objective becomes dominant, the model shows relatively poor performance due to insufficient class discrimination within each domain. On the other hand, when the intra-domain objective dominates, the performance also degrades because the lack of inter-domain disentanglement causes partial confusion between domains. A balanced trade-off between γ_1 and γ_2 yields the best performance.

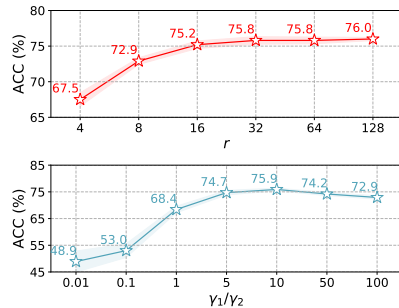


Figure 5: Hyperparameter analysis.

6 Conclusion

In this paper, we propose a novel GIL framework, GraphKeeper, which effectively addresses catastrophic forgetting in the Domain-IL setting by mitigating embedding shifts and decision boundary deviations. The multi-domain graph disentanglement module learns stable and disentangled representations, thereby reducing interference among domains. In parallel, the deviation-free knowledge preservation mechanism maintains a consistent decision boundary across tasks. Moreover, the domain-aware distribution discrimination enables precise embedding of graphs with unknown domain labels. Extensive experiments validate the effectiveness of GraphKeeper and highlight its strong potential for seamless integration into graph foundation models.

Acknowledgements

The corresponding author is Qingyun Sun. Authors of this paper are supported by the National Natural Science Foundation of China through grants No.62302023 and No.62225202, the Fundamental Research Funds for the Central Universities JK2024-07, and the Guangzhou Basic and Applied Basic Research Foundation 2024A04J3681. We owe sincere thanks to all authors for their valuable efforts and contributions.

References

- [1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pages 139–154, 2018.
- [2] Yuxuan Cao, Jiarong Xu, Carl Yang, Jiaan Wang, Yunchao Zhang, Chunping Wang, Lei Chen, and Yang Yang. When to pre-train graph neural networks? from data generation perspective! In *SIGKDD*, pages 142–153, 2023.
- [3] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *NeurIPS*, 32, 2019.
- [4] Yuanning Cui, Yuxin Wang, Zequn Sun, Wenqiang Liu, Yiqiao Jiang, Kexin Han, and Wei Hu. Lifelong embedding learning and transfer for growing knowledge graphs. In *AAAI*, volume 37, pages 4217–4224, 2023.
- [5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.
- [6] Falih Gozi Febrinanto, Feng Xia, Kristen Moore, Chandra Thapa, and Charu Aggarwal. Graph lifelong learning: A survey. *IEEE Computational Intelligence Magazine*, 18(1):32–51, 2023.
- [7] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *The Web Conference*, pages 2331–2341, 2020.
- [8] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.
- [9] Anchun Gui, Jinqiang Ye, and Han Xiao. G-adapter: Towards structure-aware parameter-efficient transfer learning for graph transformer networks. In *AAAI*, volume 38, pages 12226–12234, 2024.
- [10] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.
- [11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [12] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.
- [13] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
- [14] Shengrui Li, Xueting Han, and Jing Bai. Adapterggnn: Parameter-efficient fine-tuning improves generalization in gnns. In *AAAI*, volume 38, pages 13600–13608, 2024.
- [15] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.
- [16] Huihui Liu, Yiding Yang, and Xinchao Wang. Overcoming catastrophic forgetting in graph neural networks. In *AAAI*, volume 35, pages 8653–8661, 2021.
- [17] Yilun Liu, Ruihong Qiu, and Zi Huang. Cat: Balanced continual graph learning with graph condensation. In *ICDM*, pages 1157–1162. IEEE, 2023.
- [18] David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. *NeurIPS*, 30, 2017.

- [19] Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *Advances in Neural Information Processing Systems*, 36, 2024.
- [20] Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.
- [21] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. Query-driven active surveying for collective classification. In *10th international workshop on mining and learning with graphs*, volume 8, page 1, 2012.
- [22] Chaoxi Niu, Guansong Pang, and Ling Chen. Graph continual learning with debiased lossless memory replay. In *ECAI 2024-27th European Conference on Artificial Intelligence, Including 13th Conference on Prestigious Applications of Intelligent Systems, PAIS 2024, Proceedings*, pages 1808–1815. IOS Press BV, 2024.
- [23] Chaoxi Niu, Guansong Pang, Ling Chen, and Bing Liu. Replay-and-forget-free graph class-incremental learning: A task profiling and prompting approach. *Advances in Neural Information Processing Systems*, 37:87978–88002, 2024.
- [24] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.
- [25] Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *CIKM*, pages 1325–1334, 2020.
- [26] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.
- [27] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.
- [28] Bo-Shen Shi, Yong-Qing Wang, Fang-Da Guo, Bing-Bing Xu, Hua-Wei Shen, and Xue-Qi Cheng. Domain adaptation for graph representation learning: Challenges, progress, and prospects. *Journal of Computer Science and Technology*, pages 1–18, 2025.
- [29] Gilbert W Stewart. On the early history of the singular value decomposition. *SIAM review*, 35(4):551–566, 1993.
- [30] Jiaqi Sun, Lin Zhang, Guangyi Chen, Peng Xu, Kun Zhang, and Yujie Yang. Feature expansion for graph neural networks. In *ICML*, pages 33156–33176. PMLR, 2023.
- [31] Li Sun, Junda Ye, Hao Peng, Feiyang Wang, and S Yu Philip. Self-supervised continual graph learning in adaptive riemannian spaces. In *AAAI*, volume 37, pages 4633–4642, 2023.
- [32] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- [33] Junshan Wang, Wenhao Zhu, Guojie Song, and Liang Wang. Streaming graph neural networks with generative replay. In *SIGKDD*, pages 1878–1888, 2022.
- [34] Di Wei, Yu Gu, Yumeng Song, Zhen Song, Fangfang Li, and Ge Yu. Incregnn: Incremental graph neural network learning by considering node and parameter importance. In *DASFAA*, pages 739–746. Springer, 2022.
- [35] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*, pages 6861–6871. PMLR, 2019.
- [36] Man Wu, Xin Zheng, Qin Zhang, Xiao Shen, Xiong Luo, Xingquan Zhu, and Shirui Pan. Graph learning under distribution shifts: A comprehensive survey on domain adaptation, out-of-distribution, and continual learning. *arXiv preprint arXiv:2402.16374*, 2024.
- [37] Lianghao Xia and Chao Huang. Anygraph: Graph foundation model in the wild. *arXiv preprint arXiv:2408.10700*, 2024.
- [38] Zhe-Rui Yang, Jindong Han, Chang-Dong Wang, and Hao Liu. Graphlora: Structure-aware contrastive low-rank adaptation for cross-graph transfer learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 1785–1796, 2025.

- [39] Xingtong Yu, Chang Zhou, Yuan Fang, and Xinming Zhang. Text-free multi-domain graph pre-training: Toward graph foundation models. *arXiv preprint arXiv:2405.13934*, 2024.
- [40] Qiao Yuan, Sheng-Wei Guan, Pin Ni, Tianlun Luo, Ka Lok Man, Prudence Wong, and Victor Chang. Continual graph learning: A survey. *arXiv preprint arXiv:2301.12230*, 2023.
- [41] Peiyan Zhang, Yuchen Yan, Chaozhuo Li, Senzhang Wang, Xing Xie, Guojie Song, and Sunghun Kim. Continual learning on dynamic graphs via parameter isolation. In *SIGIR*, pages 601–611, 2023.
- [42] Xikun Zhang, Dongjin Song, Yixin Chen, and Dacheng Tao. Topology-aware embedding memory for continual learning on expanding networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4326–4337, 2024.
- [43] Xikun Zhang, Dongjin Song, and Dacheng Tao. Cglb: Benchmark tasks for continual graph learning. *NeurIPS*, 35:13006–13021, 2022.
- [44] Xikun Zhang, Dongjin Song, and Dacheng Tao. Hierarchical prototype networks for continual graph representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4622–4636, 2022.
- [45] Xikun Zhang, Dongjin Song, and Dacheng Tao. Sparsified subgraph memory for continual graph representation learning. In *ICDM*, pages 1335–1340. IEEE, 2022.
- [46] Xikun Zhang, Dongjin Song, and Dacheng Tao. Ricci curvature-based graph sparsification for continual graph representation learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.
- [47] Xikun Zhang, Dongjin Song, and Dacheng Tao. Continual learning on graphs: Challenges, solutions, and opportunities. *arXiv preprint arXiv:2402.11565*, 2024.
- [48] Ziwei Zhang, Haoyang Li, Zeyang Zhang, Yijian Qin, Xin Wang, and Wenwu Zhu. Graph meets llms: Towards large graph models. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*, 2023.
- [49] Haihong Zhao, Aochuan Chen, Xiangguo Sun, Hong Cheng, and Jia Li. All in one and one for all: A simple yet effective method towards cross-domain graph pretraining. In *SIGKDD*, pages 4443–4454, 2024.
- [50] Jianan Zhao, Hesham Mostafa, Mikhail Galkin, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. Graphany: A foundation model for node classification on any graph. *arXiv preprint arXiv:2405.20445*, 2024.
- [51] Junhao Zheng, Shengjie Qiu, and Qianli Ma. Learn or recall? revisiting incremental learning with pre-trained language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14848–14877, 2024.
- [52] Fan Zhou and Chengtai Cao. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *AAAI*, volume 35, pages 4714–4722, 2021.
- [53] Huiping Zhuang, Yizhu Chen, Di Fang, Run He, Kai Tong, Hongxin Wei, Ziqian Zeng, and Cen Chen. GacI: Exemplar-free generalized analytic continual learning. In *NeurIPS*, 2024.
- [54] Huiping Zhuang, Zhenyu Weng, Hongxin Wei, Renchunzi Xie, Kar-Ann Toh, and Zhiping Lin. Acil: Analytic class-incremental learning with absolute memorization and privacy protection. *NeurIPS*, 35:11602–11614, 2022.

A Derivation of Eq. (10) and Eq. (11)

We first restate Eq. (10) and Eq. (11) for reference.

Eq. (10):

$$\mathbf{W}_i = [\mathbf{W}_{i-1} - \mathbf{M}_i \mathbf{X}_i^\top \mathbf{X}_i \mathbf{W}_{i-1} \parallel \mathbf{M}_i \mathbf{X}_i^\top \mathbf{Y}_i],$$

where $\parallel$ indicates the concatenation and $\mathbf{M}_i$ is an intermediate matrix.

Eq. (11):

$$\mathbf{M}_i = \mathbf{M}_{i-1} - \mathbf{M}_{i-1} \mathbf{X}_i^\top (\mathbf{I} + \mathbf{X}_i \mathbf{M}_{i-1} \mathbf{X}_i^\top)^{-1} \mathbf{X}_i \mathbf{M}_{i-1},$$

where $\mathbf{M}_1 = (\mathbf{X}_1^\top \mathbf{X}_1 + \lambda \mathbf{I})^{-1}$.

Proof. We begin by defining the closed-form solution of ridge regression at step k as:

$$\mathbf{W}_k = \mathbf{M}_k \mathbf{X}_{(1:k)}^\top \mathbf{Y}_{(1:k)}. \quad (\text{A.1})$$

where $\mathbf{M}_k = (\mathbf{X}_{(1:k)}^\top \mathbf{X}_{(1:k)} + \lambda \mathbf{I})^{-1}$. This can be recursively expressed using the matrix inversion lemma as:

$$\mathbf{M}_k = (\mathbf{M}_{k-1}^{-1} + \mathbf{X}_k^\top \mathbf{X}_k)^{-1}. \quad (\text{A.2})$$

Applying the Woodbury matrix identity, we derive the recursive update of $\mathbf{M}_k$:

$$\mathbf{M}_k = \mathbf{M}_{k-1} - \mathbf{M}_{k-1} \mathbf{X}_k^\top (\mathbf{I} + \mathbf{X}_k \mathbf{M}_{k-1} \mathbf{X}_k^\top)^{-1} \mathbf{X}_k \mathbf{M}_{k-1}, \quad (\text{A.3})$$

which corresponds to Eq. (11). We then turn to the update of $\mathbf{W}_k$. By definition:

$$\begin{aligned} \mathbf{W}_k &= \mathbf{M}_k \mathbf{X}_{(1:k)}^\top \mathbf{Y}_{(1:k)} \\ &= \mathbf{M}_k [\mathbf{X}_{(1:k-1)}^\top \mathbf{Y}_{(1:k-1)} \parallel \mathbf{X}_k^\top \mathbf{Y}_k] \\ &= [\mathbf{M}_k \mathbf{X}_{(1:k-1)}^\top \mathbf{Y}_{(1:k-1)} \parallel \mathbf{M}_k \mathbf{X}_k^\top \mathbf{Y}_k]. \end{aligned} \quad (\text{A.4})$$

We now focus on computing the first term. Substituting Eq. (A.3) into this term yields:

$$\begin{aligned} \mathbf{M}_k \mathbf{X}_{(1:k-1)}^\top \mathbf{Y}_{(1:k-1)} &= \mathbf{M}_{k-1} \mathbf{X}_{(1:k-1)}^\top \mathbf{Y}_{(1:k-1)} \\ &\quad - \mathbf{M}_{k-1} \mathbf{X}_k^\top (\mathbf{I} + \mathbf{X}_k \mathbf{M}_{k-1} \mathbf{X}_k^\top)^{-1} \mathbf{X}_k \mathbf{M}_{k-1} \mathbf{X}_{(1:k-1)}^\top \mathbf{Y}_{(1:k-1)} \\ &= \mathbf{W}_{k-1} - \mathbf{M}_{k-1} \mathbf{X}_k^\top (\mathbf{I} + \mathbf{X}_k \mathbf{M}_{k-1} \mathbf{X}_k^\top)^{-1} \mathbf{X}_k \mathbf{W}_{k-1}. \end{aligned} \quad (\text{A.5})$$

To simplify the matrix expression, we define:

$$\mathbf{K}_k = (\mathbf{I} + \mathbf{X}_k \mathbf{M}_{k-1} \mathbf{X}_k^\top)^{-1}. \quad (\text{A.6})$$

From the identity $\mathbf{K}_k (\mathbf{I} + \mathbf{X}_k \mathbf{M}_{k-1} \mathbf{X}_k^\top) = \mathbf{I}$, we can derive:

$$\mathbf{K}_k = \mathbf{I} - \mathbf{K}_k \mathbf{X}_k \mathbf{M}_{k-1} \mathbf{X}_k^\top. \quad (\text{A.7})$$

Using this, we rewrite the matrix product as:

$$\begin{aligned} \mathbf{M}_{k-1} \mathbf{X}_k^\top (\mathbf{I} + \mathbf{X}_k \mathbf{M}_{k-1} \mathbf{X}_k^\top)^{-1} &= \mathbf{M}_{k-1} \mathbf{X}_k^\top \mathbf{K}_k \\ &= \mathbf{M}_k \mathbf{X}_k^\top. \end{aligned} \quad (\text{A.8})$$

Substituting Eq. (A.8) into Eq. (A.5) gives:

$$\mathbf{M}_k \mathbf{X}_{(1:k-1)}^\top \mathbf{Y}_{(1:k-1)} = \mathbf{W}_{k-1} - \mathbf{M}_k \mathbf{X}_k^\top \mathbf{X}_k \mathbf{W}_{k-1}. \quad (\text{A.9})$$

Finally, substituting Eq. (A.9) back into Eq. (A.4), we obtain:

$$\mathbf{W}_k = [\mathbf{W}_{k-1} - \mathbf{M}_k \mathbf{X}_k^\top \mathbf{X}_k \mathbf{W}_{k-1} \parallel \mathbf{M}_k \mathbf{X}_k^\top \mathbf{Y}_k], \quad (\text{A.10})$$

where $\mathbf{M}_k$ is recursively updated via Eq. (A.3).

This completes the derivation of Eq. (10) and Eq. (11). $\square$

B Algorithms and Complexity Analysis

B.1 Algorithms

The overall training and inference process of GraphKeeper are given in Algorithm 1 and Algorithm 2.

Algorithm 1: The overall training process of GraphKeeper

Input: Graph domain sequence $S = \{G_1, \dots, G_T\}$; Pre-trained GNN ξ ; Number of training epochs E .
Output: Ridge regression parameters W ; Domain-specific LoRA parameters $\{\phi_1, \dots, \phi_T\}$; Domain prototypes $\{D_1, \dots, D_T\}$; Random projection function GNN_{proj} .

- 1 Randomly initialize and frozen the parameters of GNN_{proj} ;
- 2 Frozen the parameters of ξ ;
- 3 Initialize the set P of the embedding prototypes;
- 4 **for** $t = 1, \dots, T$ **do**
 - // Multi-domain Graph Disentanglement
 - 5 Align the feature dimensions of G_t by Eq. (2);
 - 6 Initialize the LoRA parameters ϕ_t for G_t ;
 - 7 **for** $epoch = 1, \dots, E$ **do**
 - 8 | Train the LoRA parameters ϕ_t by Eq. (7);
 - 9 **end**
 - 10 Calculate the embedding X_t by Eq. (3);
 - 11 Add new embedding prototypes into set P by clustering;
 - // Deviation-Free Knowledge Preservation
 - 12 Update the ridge regression parameters W by Eq. (10) and (11);
 - 13 Calculate the domain prototype D_t ;
- 14 **end**

Algorithm 2: The inference process of GraphKeeper

Input: Test graph G_{test} ; Ridge regression parameters W ; Pre-trained GNN ξ ; Domain-specific LoRA parameters $\{\phi_1, \dots, \phi_T\}$; Domain prototypes $\{D_1, \dots, D_T\}$; Random projection function GNN_{proj} .
Output: Predicted result of the test graph G_{test} .

// Domain-aware Distribution Discrimination

- 1 Calculate the test prototype D_{test} ;
- 2 Discriminate the domain of G_{test} by Eq. (14);
- // Inference
- 3 Calculate the embedding X_{test} with ξ and ϕ_{test} by Eq. (3);
- 4 Get the predicted result by Eq. (12);

B.2 Complexity Analysis

For brevity, $\bar{d}$ donates the unified feature dimension, $|G_i|$ donates the number of nodes in the i -th graph domain, and d donates the model's hidden layer dimension. For a single graph domain, with the model containing 2 layers, the complexity of the forward propagation of the frozen pre-trained GNN is $\mathcal{O}(|G|\bar{d}d + |G|d^2)$, and the complexity of LoRA forward and backward propagation is $\mathcal{O}(2(|G|(\bar{d}r + rd) + 2|G|(dr))) = \mathcal{O}(2|G|r(\bar{d} + 3d))$, where r is the rank of LoRA module. The complexity of intra-domain disentanglement loss calculated by Eq. (5) is $\mathcal{O}(|G|^2d)$, and the complexity of inter-domain disentanglement loss calculated by Eq. (6) is $\mathcal{O}(|G||P|d)$, where $|P|$ donates the number of embedding prototypes. Given the number of the training epochs E , the combined complexity of these components is $\mathcal{O}(E(|G|(d(\bar{d} + d + 6r + |G| + |P|) + 2r\bar{d})))$. For Eq. (10) and Eq. (11), the primary computational cost arises from matrix inversion, with a complexity of $\mathcal{O}(|G|^3)$, but this is performed only once. The overall computational complexity is $\mathcal{O}(\sum_{i=1}^T E(|G_i|(d(\bar{d} + d + 6r + |G_i| + |P|) + 2r\bar{d})) + |G_i|^3)$. As the feature dimension $\bar{d}$, the rank r , hidden layer dimension d , the number of embedding prototypes $|P|$, and the training epochs E are relatively small compared to

the number of nodes, the overall computational complexity can then be approximately reduced to $\mathcal{O}(\sum_{i=1}^T (|G_i|^2 + |G_i|^3)) = \mathcal{O}(\sum_{i=1}^T |G_i|^3)$

C Experiment Details

C.1 Datasets

We conduct experiments on 15 real-world datasets, including academic networks (Cora [26], Citeseer [12], PubMed [21], CoauthorCS [27], and DBLP [7]), co-purchase networks (Photo [27] and Computer [27]), web networks (WikiCS [20], Facebook [24], Chameleon [24], and Squirrel [24]), social networks (GitHub [24], LastFMAsia [25], and DeezerEurope [25]), and airline networks (Airport [3]). Statistics of datasets are concluded in Table C.1. All the datasets are consented to by the authors for academic usage. All the datasets do not contain personally identifiable information or offensive content.

Table C.1: Statistics of datasets.

Dataset	# Nodes	# Edges	# Features	# Classes	# Homophily
Cora	2,708	5,429	1,433	7	0.81
Citeseer	3,327	4,732	3,703	6	0.74
PubMed	19,717	44,338	500	3	0.80
CoauthorCS	18,333	163,788	6,805	15	0.81
DBLP	17,716	105,734	1,639	4	0.83
Photo	7,650	119,081	745	8	0.83
Computer	13,752	245,778	767	10	0.78
WikiCS	11,701	431,726	300	10	0.65
Facebook	22,470	342,004	128	4	0.89
Chameleon	2,277	36,101	2,325	5	0.24
Squirrel	5,201	217,073	2,089	5	0.22
GitHub	37,700	578,006	128	2	0.85
LastFMAsia	7,624	55,612	128	18	0.87
DeezerEurope	28,281	185,504	128	2	0.53
Airport	3,188	18,631	4	4	0.72

C.2 Experiment Setting

C.2.1 Domain-IL Scenario

To comprehensively evaluate the effectiveness of methods in Domain-IL scenario, we obtained 6 increment groups of graph domain from 15 datasets, as shown in Table C.2. Group 1 to Group 3 consist of graph domains from the same type (*e.g.*, all are social networks in a group), while Group 4 to Group 6 include graph domains from mixed types. For the group containing graph domains A, B, C, and D, we evaluate in incremental orders of $A \rightarrow B \rightarrow C \rightarrow D$, $B \rightarrow C \rightarrow D \rightarrow A$, $C \rightarrow D \rightarrow A \rightarrow B$, and $D \rightarrow A \rightarrow B \rightarrow C$, respectively. For each graph domain, we set the unified dimension of the features to 512 and split the training set, validation set, and test set in proportions of 60%, 20%, and 20%. For the few-shot setting, we sample 10 labeled nodes from all classes within each graph domain for training.

C.3 Evaluation Metrics

The accuracy matrix is denoted as M , where $M_{i,j}$ represents the accuracy on domain j after learning domain i . We adopt two commonly used metrics: Average Accuracy (AA) and Average Forgetting (AF). The AA is computed as $\frac{1}{T} \sum_{j=1}^T M_{T,j}$, which represents the average performance across all domains after learning all domains. On the other hand, AF quantifies forgetting across domains through $\frac{1}{T-1} \sum_{j=1}^{T-1} (M_{T,j} - M_{j,j})$, which represents the difference between the final performance on each domain and the performance on the domain when it was first learned. Higher AA indicate better performance and higher AF indicates less forgetting.

Table C.2: Statistics of incremental groups of graph domain.

Group	Graph Domains
Group 1	GitHub, LastFMAsia, DeezerEurope
Group 2	WikiCS, Facebook, Chameleon, Squirrel
Group 3	Citeseer, Pubmed, CoauthorCS, DBLP
Group 4	Pubmed, Photo, WikiCS, Airport
Group 5	CoauthorCS, Computer, Chameleon, DeezerEurope
Group 6	Cora, Facebook, LastFMAsia, Squirrel
Group 7	GitHub, LastFMAsia, DeezerEurope, WikiCS, Facebook, Chameleon, Squirrel, Citeseer, Pubmed, CoauthorCS, DBLP
Group 8	Pubmed, Photo, WikiCS, Airport, CoauthorCS, Computer, DeezerEurope, Cora, Facebook, LastFMAsia, Squirrel

C.4 Running Environment

We conduct the experiments with:

- Operating System: Ubuntu 20.04 LTS.
- CPU: Intel(R) Xeon(R) Platinum 8358 CPU@2.60GHz with 1TB DDR4 of Memory.
- GPU: NVIDIA Tesla V100 with 32GB of Memory.
- Software: CUDA 11.7, Python 3.8.0, Pytorch 1.7.1, DGL 0.6.1.

D Implementation Details

D.1 Implementation Details of GraphKeeper

We set the number of model layers to 2 for all methods, with the learning rate set to $5e-2$, the weight decay coefficient set to $5e-4$, and 200 training epochs per incremental graph domain. The parameters of models are optimized by Adam [11]. For GraphKeeper, the pre-trained GNN model is trained through link prediction on the first incremental domain (except for Sec. 5.3), the feature and structure augmentation are achieved by randomly masking a few features and dropping a few edges, and the embedding prototype sampling is implemented through DBSCAN [5]. The hyperparameter r is chosen from $\{4, 8, 16, 32, 64, 128\}$, γ_1 is chosen from $\{0.01, 0.1, 1.0, 5.0, 10.0\}$, γ_2 is chosen from $\{0.001, 0.01, 0.10, 0.50, 1.0\}$.

D.2 Implementation Details of Baselines

We implement all methods under the GIL benchmark [43].

- **EWC** [13], **MAS** [1], **GEM** [18], **LWF** [15], **TWP** [16], **ER-GNN** [52]: <https://github.com/QueuQ/CGLB> [CC BY 4.0 License].
- **SSM** [45]: <https://github.com/QueuQ/SSM> [CC BY 4.0 License].
- **TPP** [23]: <https://github.com/mala-lab/TPP> [MIT License].
- **DeLoMe** [22]: <https://github.com/mala-lab/DeLoMe> [with license unspecified].
- **PDGNNs** [42]: <https://github.com/imZHANGxikun/PDGNNs> [CC BY 4.0 License].

E Additional Experiment Results

E.1 Comparison of Methods with Unified GCN Backbone

Considering that some baselines adopt simplified GNN such as SGC [35] and APPNP [8] as their backbone, they achieve stability at the cost of sacrificing model plasticity. However, constraining the backbone of the model imposes limitations on practical applications, especially when the model architecture cannot be replaced. Thus, we unify the backbone of related methods to GCN for a fair

Table E.3: Comparison of performance with the unified GCN backbone.

Method	Group 1		Group 2		Group 3		Group 4		Group 5		Group 6	
	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$
DeLoMe	49.2 $\pm$ 2.1	-20.9 $\pm$ 3.3	32.2 $\pm$ 1.4	-51.8 $\pm$ 1.7	56.9 $\pm$ 0.3	-19.8 $\pm$ 0.3	40.9 $\pm$ 1.3	-44.8 $\pm$ 1.6	49.1 $\pm$ 1.3	-31.7 $\pm$ 2.1	45.4 $\pm$ 1.5	-35.3 $\pm$ 2.4
PDGNNs	48.0 $\pm$ 2.4	-33.4 $\pm$ 3.8	42.3 $\pm$ 2.5	-39.1 $\pm$ 2.9	53.4 $\pm$ 0.7	-35.5 $\pm$ 0.9	40.9 $\pm$ 1.7	-49.4 $\pm$ 2.4	49.6 $\pm$ 1.8	-36.1 $\pm$ 2.5	45.1 $\pm$ 1.5	-38.1 $\pm$ 2.3
GraphKeeper	69.2$\pm$0.3	-0.4$\pm$0.2	73.1$\pm$1.1	-2.8$\pm$0.7	80.6$\pm$0.4	0.0$\pm$0.1	79.9$\pm$0.5	1.0$\pm$0.5	75.5$\pm$0.9	0.1$\pm$0.5	77.5$\pm$0.7	-0.1$\pm$0.5

Table E.4: Performance comparison on longer Domain-IL sequences.

Method	Group 7 (11 Domains)		Group 8 (12 Domains)	
	AA $\uparrow$	AF $\uparrow$	AA $\uparrow$	AF $\uparrow$
TPP	50.2 $\pm$ 0.9	0.0 $\pm$ 0.0	52.8 $\pm$ 2.5	0.0 $\pm$ 0.0
DeLoMe	> 1 day		> 1 day	
PDGNNs	37.7 $\pm$ 0.1	-13.1 $\pm$ 0.4	45.0 $\pm$ 0.2	-13.6 $\pm$ 0.3
GraphKeeper	73.0$\pm$1.1	-0.7$\pm$0.5	76.5$\pm$0.9	-0.8$\pm$0.2

comparison. The results are summarized in Table E.3. It shows that after unifying the backbone, the performance of the baselines declines to a certain extent, significantly reducing their usability. This also highlights the advantage of our approach, which imposes no restrictions on model architecture, achieving a dual benefit of plasticity and stability.

E.2 Longer Incremental Learning Sequence

We conduct evaluation on two longer incremental sequences, Group 7 and Group 8, containing 11 and 12 graph domains, respectively, which are more challenging. As shown in Table E.4, the advanced baselines suffer significant performance drops, while GraphKeeper maintains strong performance and outperforms the runner-up by 22.8%~23.7%. This demonstrates the scalability of GraphKeeper and its robustness to larger data scales.

E.3 Ablation study

Due to space limitations, the ablation study from Group 4 to Group 6 are presented in Figure E.1, showing that removing any component of GraphKeeper leads to a degradation of the performance, especially for GraphKeeper (w/o PEFT) and GraphKeeper (w/o KP) variants. Specifically, the full model consistently outperforms all ablated variants across the three groups. Notably, removing PEFT or KP causes the most significant drops (*e.g.*, from 79.9% to 21.3%/19.7% in Group 4), highlighting their crucial roles in preserving prior knowledge and supporting cross-domain alignment. The performance of w/o DT is also inferior, confirming the importance of domain transfer modules. These results verify the necessity of each component for robust generalization.

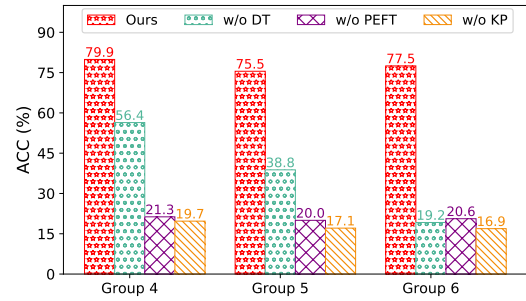


Figure E.1: Ablation study.

E.4 Domain Discrimination

Through random high-dimensional mapping, we achieve linear separability between domain features that may potentially overlap. As shown in Figure E.2, all domains exhibit a significant reduction in pairwise confusion after mapping. This provides a stability guarantee for our subsequent domain prototype matching.

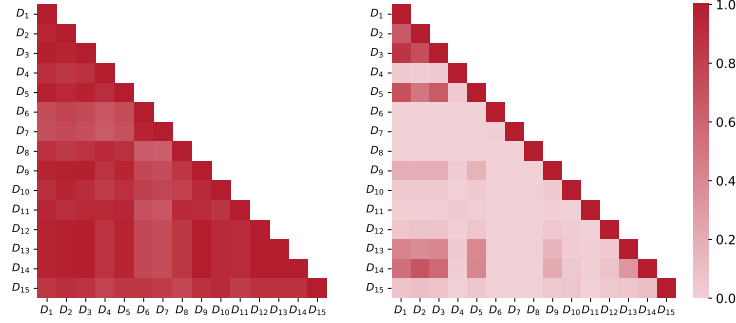


Figure E.2: Confusion matrix of domain prototypes. Deeper color indicates higher proximity. *Left*: Original. *Right*: Ours.

F Further Discussion on Related Work

In Sec. 4.3, we introduce the domain-aware distribution discrimination to match graphs with unobservable domains to previous domain prototypes. Recent work [23] discusses a similar issue, they propose applying Laplace smoothing on original features and matching the test graph with the class prototypes through cosine similarity. Unfortunately, as shown in Figure G.3, the features across multiple domains are at risk of being excessively similar or even overlapping, which may cause the test graph to be matched with the wrong domain prototype. We guarantee a sufficiently distance between the prototypes of different domains through high-dimensional random mapping, use distance metrics in high-dimensional space as the criteria for matching, effectively preventing confusion in domain discrimination.

G Limitations

The limitations of GraphKeeper include the following aspects. First, when evaluating the performance of GraphKeeper in Domain-IL scenario, we follow the traditional setting, where the classes in the new incremental domain are assumed to be entirely novel. However, more general and practical cases, such as partially overlapping classes across domains, have not been thoroughly explored. Additionally, although GraphKeeper can be seamlessly integrated into existing GFMs, enhancing their continual updating capabilities based on their few-shot learning strengths, it does not fully leverage the zero-shot learning potential of GFMs. This limitation is tied to the scenario design, as it does not account for previously unseen domains, an aspect that has been overlooked by all existing methods. In summary, future work will extend the GraphKeeper’s applicability, and investigate more general scenario in GIL.

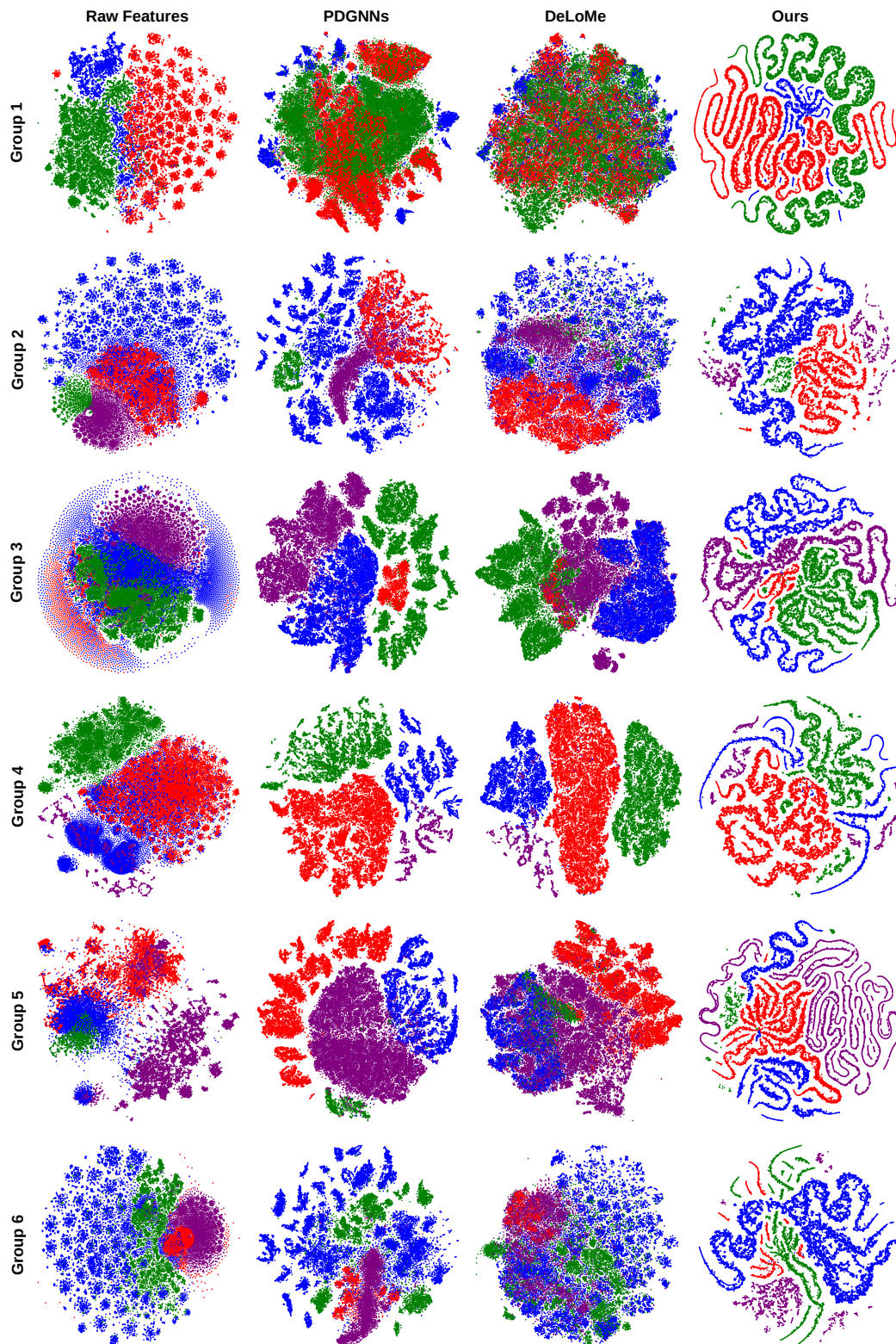


Figure G.3: Visualization of the node embeddings of different graph domains after incrementally learning all graph domains. Each color corresponds to a graph domain. The embeddings produced by GraphKeeper are highly compact with clear boundaries between different graph domains.

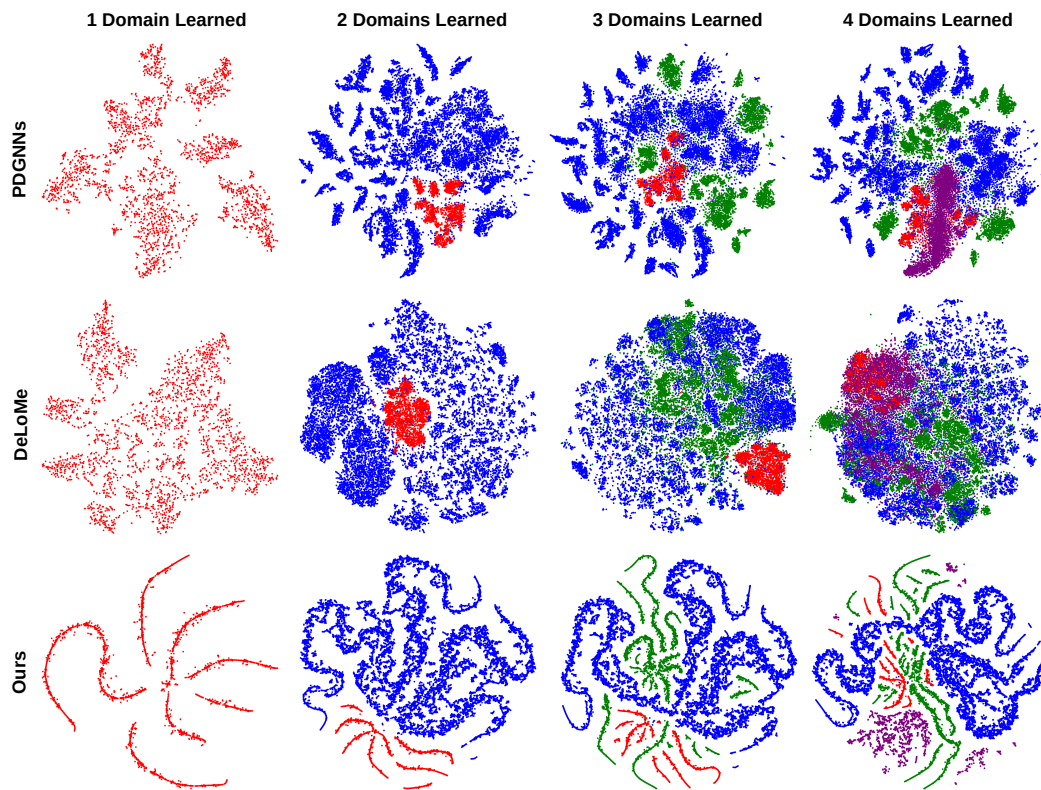


Figure G.4: Visualization of the node embeddings of previously learned domains at different stages of the learning process on Group 6.

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: The main claims made in the abstract and introduction accurately represent the paper's contributions and scope, with all results and findings clearly and correctly reflected.

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: See Appendix G for discussion on limitations.

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: [\[Yes\]](#)

Justification: We provide detailed proof in Appendix A for two theoretical derivations in Eq. (10) and Eq. (11).

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: We have provided the code, datasets, training scripts, and algorithms (See Sec.5.1 and Appendix B), which collectively provide comprehensive details for fully reproducing the experiments and verifying the results presented in the paper.

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 code and datasets is available at: <https://anonymous.4open.science/r/GraphKeeper>.

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: All experimental setting and details are provided in Appendix D.

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: We have provided the variance in the experimental results tables, which allows for an understanding of the variability and reliability of the results.

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: See Appendix C.4 for discussion on experiments compute resources.

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: The research conducted in this paper adheres to all aspects of the NeurIPS Code of Ethics. No ethical issues were identified in the methodology, data collection, or result interpretations.

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: This paper enables adaptive and scalable graph learning, promotes knowledge sharing, and supports privacy-friendly applications without storing past data.

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: The paper does not involve the release of data or models that have a high risk for misuse, hence the question of safeguards does not apply in this context.

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: The paper appropriately cites and respects the licenses and terms of use of the datasets and baseline code used in the work.

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: [\[Yes\]](#)

Justification: The paper introduces GraphKeeper as a new method and provides code access via an anonymous repository.

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: The 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: The 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.