GRAVER: Generative Graph Vocabularies for Robust Graph Foundation Models Fine-tuning

Haonan Yuan[1], **Qingyun Sun**[1]*, **Junhua Shi**[1], **Xingcheng Fu**[2], **Bryan Hooi**[3], **Jianxin Li**[1], **Philip S. Yu**[4]
[1]SKLCCSE, School of Computer Science and Engineering, Beihang University
[2]Key Lab of Education Blockchain and Intelligent Technology, Guangxi Normal University
[3]School of Computing, National University of Singapore
[4]Department of Computer Science, University of Illinois, Chicago
{yuanhn,sunqy,shijunhua,lijx}@buaa.edu.cn
fuxc@gxnu.edu.cn, bhooi@comp.nus.edu.sg, psyu@uic.edu

Abstract

Inspired by the remarkable success of foundation models in language and vision, Graph Foundation Models (GFMs) hold significant promise for broad applicability across diverse graph tasks and domains. However, existing GFMs struggle with unstable few-shot fine-tuning, where both performance and adaptation efficiency exhibit significant fluctuations caused by the randomness in the support sample selection and structural discrepancies between the pre-trained and target graphs. How to fine-tune GFMs robustly and efficiently to enable trustworthy knowledge transfer across domains and tasks is the major challenge. In this paper, we propose **GRAVER**, a novel **G**enerative g**RA**ph **V**ocabulari**E**s for **R**obust GFM fine-tuning framework that tackles the aforementioned instability via generative augmentations. Specifically, to identify transferable units, we analyze and extract key class-specific subgraph patterns by ego-graph disentanglement and validate their transferability both theoretically and empirically. To enable effective pre-training across diverse domains, we leverage a universal task template based on ego-graph similarity and construct graph vocabularies via graphon-based generative experts. To facilitate robust and efficient prompt fine-tuning, we grave the support samples with in-context vocabularies, where the lightweight MoE-CoE network attentively routes knowledge from source domains. Extensive experiments demonstrate the superiority of GRAVER over ***effectiveness***, ***robustness***, and ***efficiency*** on downstream few-shot node and graph classification tasks compared with **15** state-of-the-art baselines.

1 Introduction

Graph-structured data is pervasive across a wide range of domains, including social networks [13, 127], recommendation systems [114, 100], and bioinformatics [50, 17]. Graphs are capable of modeling complex relationships and attributed interactions between entities, making them a powerful abstraction for solving real-world problems, such as fraud detection [60, 128], molecular property prediction [70, 135], and protein-protein interaction modeling [34, 15]. While foundation models such as GPTs [1] and Vision Transformers [38] have achieved impressive generalization capabilities in language and vision through large-scale pre-training, they struggle when applied directly to graphs due to their non-Euclidean nature, structural heterogeneity, and task diversity inherent in graph data [52, 46, 63, 37]. To address this gap, Graph Foundation Models (GFMs) [64, 75, 149] have emerged with the goal of enabling general-purpose learning on graph-structured data. Instead of

*Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

https://doi.org/10.52202/085713-0560

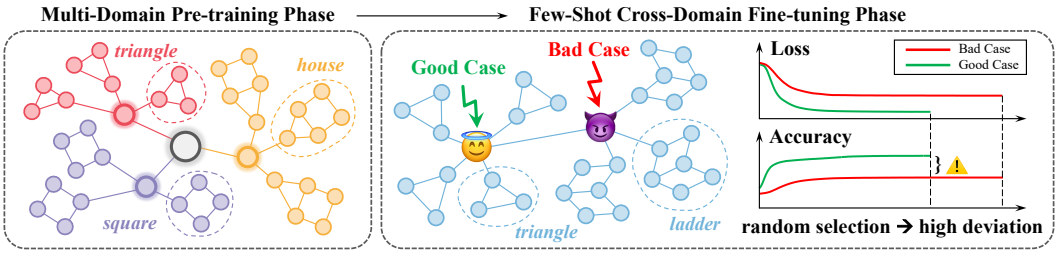


Figure 1: Case study of the fine-tuning instability. When support samples share structural patterns with pre-trained graphs (*e.g.*, triangle), fine-tuning is efficient and stable (Good Case). In contrast, mismatched patterns (*e.g.*, ladder) lead to poor convergence and lower accuracy (Bad Case). Loss and accuracy curves illustrate that random support selection causes high performance deviation, highlighting the need for structure-aware augmentation.

relying on task-specific deep graph models, ideal GFMs are trained once on diverse multi-domain graph datasets and then adapted with minimal supervision to a variety of downstream tasks [76, 5]. Such a ***"pretrain-then-finetune"*** paradigm [86, 23, 42] holds the promise of unifying graph learning across domains by providing a reusable model architecture that generalizes structural knowledge at scale [58, 83, 122, 136, 126].

Existing research on GFMs has addressed prominent challenges like architecture scalability [78, 42], domain alignment [47, 136], prompt design [83, 45], *etc.* Nevertheless, GFMs encounter a more fundamental and pervasive issue: ***robustness*** and ***efficiency*** of few-shot prompt fine-tuning. Specifically, the existing GFMs exhibit significant instability and inefficiency, whereby fine-tuning performance fluctuates heavily due to randomness in the selection of limited support samples and structural discrepancies between pre-trained and target graph domains. We conducted an illustrative case study in Figure 1 to highlight this issue. A synthetic cross-domain node classification setting is constructed with motif-based graphs. The findings reveal a consistent trend: when the randomly selected support nodes exhibit structural patterns similar to those in pre-training (*e.g.*, triangle △), fine-tuning converges faster and yields higher accuracy (Good Case). In contrast, the mismatched structures (*e.g.*, ladder ⧆) lead to slower convergence and degraded performance (Bad Case). This phenomenon is intuitive and commonly observed in practice. Such instability severely undermines the practical applicability and reliability of GFMs in real-world scenarios, where dependable knowledge transfer and rapid adaptation to new tasks and domains are essential.

Research Question: How can Graph Foundation Models be fine-tuned ***robustly*** and ***efficiently*** under random support sets to enable trustworthy knowledge transfer across diverse domains and tasks?

Recent studies attempt to improve GFM's adaptation ability. One line of work focuses on pre-training with transferable patterns [83, 7, 148, 97]. The most relevant GFT [98] proposes a computation-tree-based vocabulary to enable pattern reuse across domains. However, the proposed vocabulary is limited to tree-shaped structures, restricting transferability in multi-domain settings. Another line of research emphasizes generative augmentation [137, 84, 10, 96], which interpolates structures to simulate new training examples. Yet, they often generate task-agnostic structures that misalign with target domains and do not condition on the few-shot support samples, limiting their informativeness. A third group of works [122, 120, 119] introduces modular pre-training with weighted source tokens as prompts to facilitate domain-aware transfer. While effective in improving cross-domain adaptation, these strategies still fall short in scenarios where only a handful of labeled target samples are available. Notably, none of these aforementioned paradigms simultaneously address the dual challenges of generating transferable patterns and ensuring stable fine-tuning under arbitrary few-shot support sets.

To tackle this dilemma, we propose **GRAVER**, a novel **G**enerative g**RA**ph **V**ocabulari**E**s for **R**obust GFM fine-tuning framework. GRAVER directly addresses the instability inherent in few-shot fine-tuning through in-context subgraph augmentations. Specifically: ❶ We first analyze and validate the key transferable patterns from an ego-graph disentanglement perspective, rigorously exploring their transferability theoretical foundations and empirical utility across multiple domains. ❷ By leveraging this insight, we establish adaptive graph vocabularies through graphon-based generative experts in the multi-domain pre-training stage. ❸ At the final fine-tuning stage, GRAVER enhances randomly selected support samples by embedding context-specific generative vocabularies guided by a lightweight MoE-CoE network, which selectively routes pertinent structural and semantic knowledge from relevant source domains and prevents negative transfer. ***Our contributions are:***

https://doi.org/10.52202/085713-0560

- We propose a novel trustworthy GFM with robust and efficient prompt fine-tuning under multi-domain pre-training and cross-domain adaptation. To the best of our knowledge, this is the first GFM with generative graph vocabularies to in-context augment support sets for stable adaptation.
- We design graphon-based generative experts to synthesize structure-feature vocabularies aligned with transferable subgraphs, and introduce a MoE-CoE routing mechanism to dynamically assemble source-domain knowledge for context-aware augmentation during fine-tuning.
- Extensive experiments demonstrate its superior ***effectiveness***, ***robustness***, and ***efficiency*** on downstream few-shot node and graph classification compared with **15** state-of-the-art baselines.

2 Related Work

2.1 Multi-Domain Pre-trained Graph Foundation Models (Appendix G.1)

Graph foundation models have made rapid progress by using self-supervised pre-training to extract transferable patterns from large-scale graphs [57, 29, 104]. Pre-training tasks such as node masking, edge prediction, and contrastive learning have enabled methods like GCC [69], GPT-GNN [30], GPPT [82], L2P-GNN [61], and others [105, 36, 109, 8, 7, 113] to serve as the strong initializations for downstream tasks. However, most of these methods assume the structural or semantic similarity between pre-training and target domains [32, 135], making them sensitive to distribution shifts.

To address this, recent studies have investigated cross-domain graph learning, aiming to transfer knowledge between dissimilar domains. Models such as GCOPE [136], OMOG [54], PGPRec [112], UniAug [88], and UniGraph [23] focus on learning domain-invariant representations. While effective for single-source transfer, these methods often struggle to generalize across multiple heterogeneous domains and may rely on task-specific assumptions that limit scalability.

Toward multi-domain foundation models, recent work explores unified pre-training strategies across diverse domains. OFA [54] and HiGPT [87] use LLMs to encode node information as text, which improves alignment but restricts applicability to text-attributed graphs [95, 85]. GCOPE [136] introduces virtual domain bridges but lacks semantic alignment. MDGPT [123] incorporates domain tokens and SVD-based projection, though it does not ensure semantic consistency across domains.

2.2 Graph Foundation Models Fine-tuning with Prompt (Appendix G.2)

Fine-tuning the pre-trained graph foundation models is a widely adopted strategy to adapt general representations to downstream tasks [130, 84, 139, 108, 85, 115, 33, 35, 129]. Traditional methods typically update full model or selected components, which can be effective but often incur high costs.

To improve fine-tuning efficiency, parameter-efficient fine-tuning (PEFT) techniques update a small set of parameters [143, 89, 19, 106] or introduce lightweight modules [49, 147, 110]. These approaches preserve most pre-trained knowledge while reducing computational load. However, their generalization across domains remains limited due to the difficulty of handling domain shifts.

Prompt-based fine-tuning offers an alternative by conditioning the model on learned input prompts rather than altering parameters [55, 141, 140, 11]. In graph learning, prompt tuning leverages structured embeddings to guide the model toward task-specific patterns [82, 145, 77, 132], achieving better robustness and adaptability with minimal updates. Based on this paradigm, numerous graph prompt methods have been developed [82, 83, 58, 18, 145, 14, 31, 123, 118, 149], which incorporate learnable prompts to highlight structural or semantic cues, improving transferability across tasks.

2.3 GFM Pre-training with Transferable Patterns (Appendix G.3)

Graph Foundation Models aim to capture structural patterns that generalize across diverse domains and tasks [69, 142, 83, 7, 148, 97, 96]. A key challenge is defining transferable subgraph units for pre-training, since graphs do not contain obvious tokens like words or image patches.

One approach addresses this by identifying common substructures such as motifs or computation trees as a reusable graph vocabulary [142, 146, 47, 138, 144, 97, 117]. For example, GFT [98] encodes each node's computation tree and discretizes it through vector quantization. This enables the model to represent graphs in terms of recurring structural units, which improves generalization across domains and mitigates knowledge conflicts during transfer.

https://doi.org/10.52202/085713-0560

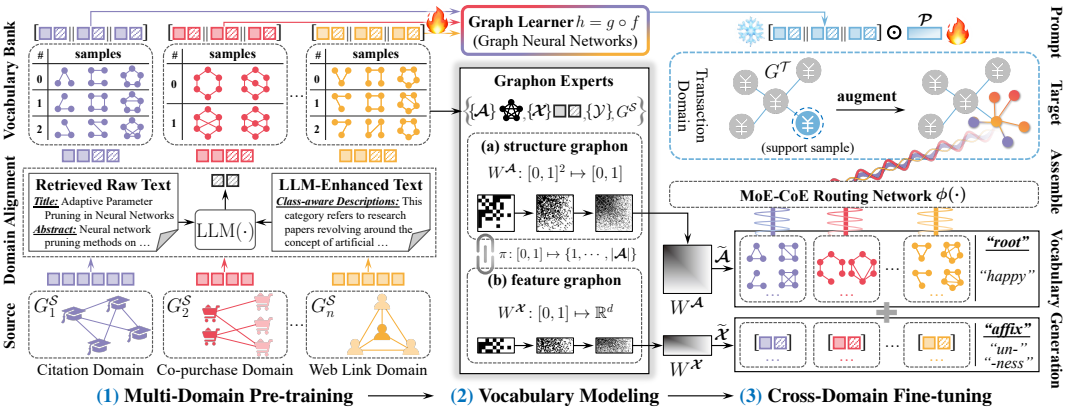


Figure 2: Framework of GRAVER. **(1)** aligns semantics via LLM-enhanced raw text, and disentangles ego-graph into transferable patterns. **(2)** models structure-feature tokens with graphon experts. **(3)** assembles class-aware vocabularies to augment support samples for robust cross-domain fine-tuning.

Another approach focuses on mining frequent motifs during pre-training [135, 4, 133]. MICRO-Graph [133] trains GNNs using motif-based contrastive objectives, while MGSSL [135] reconstructs motif segments to learn interpretable structural blocks, achieving strong performance in molecular prediction tasks. Other methods learn universal structural patterns from synthetic graphs [113].

2.4 GFM Fine-tuning with Generative Augmentations (Appendix G.4)

To improve the adaptability of pre-trained graph models, recent work explores generative augmentation techniques that synthesize new data aligned with the target graph distribution [30, 137, 7, 84, 115, 86, 111]. Unlike traditional perturbations on edges or features, these approaches aim to produce structurally meaningful examples to support robust fine-tuning.

Recent efforts have explored generative strategies to enhance the robustness and adaptability of graph foundation models. One line of work leverages graphons, continuous functions that define graph distributions [2, 68, 72, 12, 71, 84]. For instance, $\mathcal{G}$-Mixup [21] interpolates between class-wise graphons to produce realistic hybrid graphs that improve generalization and noise robustness. Another direction introduces generative fine-tuning objectives to mitigate the mismatch between pre-training and downstream distributions [48, 84, 9, 53], such as G-Tuning [84], which preserves the generative structure of downstream graphs to avoid overfitting. In addition, subgraph-level augmentations [3, 131, 41, 101, 24, 22] have been proposed to help models decouple subgraph semantics from global context; for example, [57] replaces subgraph environments during training to improve interpretability and robustness in molecular tasks. Complementing these efforts, mixture-of-experts architectures[27, 94, 59, 99, 20, 123] such as GraphMETRO [99] deploy multiple expert networks specialized on distinct structural regimes, with routing mechanisms to adaptively select experts for diverse input distributions.

3 Preliminary

Notations. A graph is denoted as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set. Let $\mathbf{A} \in \{0,1\}^{N_i \times N_i}$ be the adjacency matrix and $\mathbf{X} \in \mathbb{R}^{N_i \times d_i}$ be the node feature matrix, where $N = |\mathcal{V}|$ is the number of nodes and d_i denotes input feature dimension. $\mathbf{H}$ means hidden embeddings of $\mathbf{X}$.

Pre-training and Few-shot Fine-tuning. Given n source graphs $\{G^{\mathcal{S}}\} \in \mathcal{G}^{\mathcal{S}}$ from multi-domains $\{D^{\mathcal{S}}\} \in \mathcal{D}^{\mathcal{S}}$ with their labels $\{Y^{\mathcal{S}}\} \in \mathcal{Y}^{\mathcal{S}}$, the pre-training goal is to train a graph learner $h = g \circ f$ on multi-domain graph datasets composed of a GNN-based graph encoder f and a discriminator g, after which the pre-training parameter $\mathbf{\Theta}^\star$ is frozen. During fine-tuning, given a set of target graphs $\{G^{\mathcal{T}}\} \in \mathcal{G}^{\mathcal{T}}$ from target domain $\{D^{\mathcal{T}}\} \in \mathcal{D}^{\mathcal{T}}$ (seen or unseen) with m randomly selected labels $\{Y^{\mathcal{T}}\} \in \mathcal{Y}^{\mathcal{T}}$ (support set) under the m-shot setting ($m \ll \sum_{i=1}^{n} N_i$), the goal is to adapt the pre-trained GFM to predict the labels of the unlabeled samples (query set) correctly by learnable graph prompts.

4 Proposed GFM Framework: GRAVER

In this section, we elaborate on the proposed GRAVER with its framework illustrated in Figure 2.

4.1 Pre-training with Transferable Graph Vocabulary

A central challenge in constructing a transferable graph vocabulary for robust GFM fine-tuning lies in identifying subgraph patterns (graph vocabulary) that generalize across tasks and domains.

Multi-Domain Alignment. Graphs in different domains present heterogeneous features with inconsistent dimensions and semantics. To establish a unified representation space, we introduce a set of shared aligners that consequently unify dimension- and semantic-wise features. Given a graph $G_i^{\mathcal{S}} = (\mathcal{V}_i, \mathcal{E}_i)$ $(i \leqslant n)$ with feature matrix $\mathbf{X}_i^{\mathcal{S}} \in \mathbb{R}^{N_i \times d_i}$ from source domain, the aligners perform:

$$\widehat{\mathbf{X}}_i^{\mathcal{S}} \in \mathbb{R}^{N_i \times d} = \mathbf{W}_i^{\top}\left(\mathcal{F}\left(\mathbf{X}_i^{\mathcal{S}} \,\|\, \mathrm{LLM}\left(\mathbf{X}_i^{\mathcal{S}}\right)\right)\right), \quad \text{for all } G_i^{\mathcal{S}} \in \{G^{\mathcal{S}}\}, \tag{1}$$

where $\mathrm{LLM}(\cdot)$ is implemented by retrieving feature-related raw texts and enhancing class-aware semantics by large language models (GPT-4 and BertTokenizer) [47]. $\mathcal{F}$ implemented by truncated SVD [80] aligns feature dimensions, followed by a learnable projection $\mathbf{W}_i$ that normalizes semantics.

Factor-aware Ego-Graph Disentanglement. Motivated by the observation that localized graph patterns often emerge from multiple latent semantic factors [44], such as social affiliation, topic preference, structural role, *etc.*, each factor-specific subgraph forms a distinct semantic unit, analogous to a vocabulary in linguistics. Treating these units holistically during message-passing and aggregation may obscure the unique semantics of each vocabulary, significantly weakening the discriminability and transferability of pre-trained knowledge. To explicitly identify transferable patterns and prepare a "vocabulary bank" for modeling, we decompose the ego-graph $\mathbf{g}_u^{\mathcal{S}}$ of a given node u into K factor-aware subgraphs (vocabularies) with a differentiable neighbor soft-routing mechanism [62, 56].

Denote the multi-channel encoder as $f_{\mathbf{\Theta}}$: $\mathbf{g}_u^{\mathcal{S}} \mapsto \{\mathbf{h}_{u,k}^{\mathcal{S}} \in \mathbb{R}^d\}_{k=1}^K$, where each channel k encodes a disentangled factor of the ego-neighbors. The t-th soft-routing iteration aims to find an attention:

$$\alpha_{v\to k}^{(t)} \propto \mathrm{Softmax}_k\left(\langle \mathbf{h}_{u,k}^{\mathcal{S}(t)}, \mathbf{h}_{v,k}^{\mathcal{S}(t)} \rangle / \tau\right), \quad \text{s.t. } \sum\nolimits_{k=1}^{K} \alpha_{v\to k}^{(t)} = 1, \tag{2}$$

$$\text{and} \quad \mathbf{h}_{u,k}^{\mathcal{S}(0)} = \sigma\left(\mathbf{W}_k^{\top}\widehat{\mathbf{x}}_u^{\mathcal{S}} + \mathbf{b}_k\right) / \|\sigma\left(\mathbf{W}_k^{\top}\widehat{\mathbf{x}}_u^{\mathcal{S}} + \mathbf{b}_k\right)\|_2, \quad \text{iff } \|\sigma\left(\mathbf{W}_k^{\top}\widehat{\mathbf{x}}_u^{\mathcal{S}} + \mathbf{b}_k\right)\|_2 \geqslant \rho. \tag{3}$$

where τ is the temperature, $\mathbf{W}_k$ and $\mathbf{b}_k$ are projections. The routed neighbors form K vocabularies:

$$\mathbf{g}_u^{\mathcal{S}(t)} = \left\{\mathbf{g}_{u,k}^{\mathcal{S}(t)}\right\}_{k=1}^{K}, \quad \mathbf{g}_{u,k}^{\mathcal{S}(t)} = \left(\mathcal{V}_{u,k}^{(t)}, \mathcal{E}_{u,k}^{(t)}\right), \quad \mathcal{V}_{u,k}^{(t)} = \left\{v \in \mathcal{N}(u) \mid \alpha_{v\to k}^{(t)}\right\}. \tag{4}$$

The updated $\mathbf{h}_u^{\mathcal{S}}$ is the concatenation of that from each disentangled ego-graph over T iterations:

$$\mathbf{h}_u^{\mathcal{S}(T)} = \|_{k=1}^{K} \mathbf{h}_{u,k}^{\mathcal{S}(T)}, \quad \mathbf{h}_{u,k}^{\mathcal{S}(t+1)} := \mathbf{h}_{u,k}^{\mathcal{S}(t)} + \sum\nolimits_{v \in \mathcal{V}_{u,k}^{(t)}} \left[\alpha_{v\to k}^{(t)} \mathbf{h}_{v,k}^{\mathcal{S}(t)}\right]. \tag{5}$$

Semantic-Independence Promotion. To ensure each channel captures distinct semantic factors, we derive a mutual information regularizer between each disentangled vocabulary, which we implemented by leveraging a contrastive-based variational upper bound with the InfoNCE estimator [67]:

$$\mathcal{R}_{\mathrm{MI}}^{u} = \sum\nolimits_{i\neq j} I\left(\mathbf{h}_{u,i}^{\mathcal{S}}; \mathbf{h}_{u,j}^{\mathcal{S}}\right) \overset{\text{Proof C.1}}{\leqslant} \sum\nolimits_{i\neq j} \mathbb{E}_{u\in\mathcal{B}}\left[-\log\left(\mathrm{Softmax}_v\left(\langle \mathbf{h}_{u,i}^{\mathcal{S}}, \mathbf{h}_{v,j}^{\mathcal{S}} \rangle / \tau\right)\Big|_{v=u}\right)\right], \tag{6}$$

where $\mathcal{B}$ is a batch of sampled nodes from source domains. Intuitively, $\mathcal{R}_{\mathrm{MI}}^{u}$ constrains vocabularies to be non-aligned across nodes, thus discouraging shared semantics and keeping distinguished attributes.

Vocabulary Transferability Justification. To investigate how different vocabulary combinations contribute to pre-training, we evaluate their transferability by measuring the stability of encoder f.

Proposition 1 (**Vocabulary Transferability**). *Given any nodes u, v, and assume $\|\widehat{\mathbf{x}}_u^{\mathcal{S}} - \widehat{\mathbf{x}}_v^{\mathcal{S}}\|_2 \leqslant \epsilon$. The semantic discrepancies $\Delta = \|f(\mathbf{g}_u^{\mathcal{S}}) - f(\mathbf{g}_v^{\mathcal{S}})\|_2$ is upper bounded by:*

$$\Delta \overset{[a]\ \text{Proof C.2}}{\leqslant} \left[\min_{\Pi\in S_K} \sum\nolimits_{k=1}^{K} \left\|\mathbf{h}_{u,k}^{\mathcal{S}} - \mathbf{h}_{v,\Pi(k)}^{\mathcal{S}}\right\|_2^2 + \psi(\mathcal{R}_{\mathrm{MI}}^{u,v})\right]^{\frac{1}{2}} \overset{[b]\ \text{Proof C.3}}{\leqslant} \epsilon\sqrt{K}\left(\frac{C_\sigma L_{\mathbf{W}} L_s}{4\rho\tau}\right)^{T}, \tag{7}$$

where Π denotes a permutation over channels, $\psi(\cdot)$ is a slack function controls bound tightness. C_σ, $L_{\mathbf{W}}$, and L_s are Lipschitz constants for activation function σ, weight matrix $\mathbf{W}$, and cosine similarity $\langle\cdot,\cdot\rangle$ (inner product), respectively. ρ is the lower bound of L_2-norm in Eq. (3).

https://doi.org/10.52202/085713-0560

Proposition 1 states that the upper bound of semantic discrepancy is governed by a specific combination of vocabularies. This demonstrates that disentangled ego-graphs can function as independent and minimal semantic units, facilitating effective knowledge transfer and providing us with theoretical foundations for vocabulary-based generative augmentations.

Objectives of Self-supervised Pre-training. Inspired by prior studies [58], we adopt a fully self-supervised contrastive link prediction as the pre-training task. To address task heterogeneity, we build upon a universal task template based on ego-graph semantic similarity, which unifies both link prediction with node- or graph-level classification tasks under a shared framework.

Specifically, we construct a set of quadruples $(u, v^+, v^-, \mathbf{y}_u)$ sampled non-redundantly from $\{G^{\mathcal{S}}\}$, where v^+, v^- are the positive and negative neighbor that link (do not link) to u. The binary label $\mathbf{y}_u$ denotes the link existence and is determined by the pairwise similarity discriminator $g = \mathrm{MLP}(\langle \cdot, \cdot \rangle)$. The objective is to encourage higher semantic similarity between u and its positive neighbor $\mathbf{h}^{\mathcal{S}}_{v^+}$, while suppressing that with the negative neighbor $\mathbf{h}^{\mathcal{S}}_{v^-}$. The training objective is formalized as:

$$\mathcal{L}_{\text{pre}}(\mathbf{\Theta}; \mathbf{H}^{\mathcal{S}}) = -\sum_{(u,v^+,v^-)} \left[\log \frac{\exp(g(\mathbf{h}^{\mathcal{S}}_u, \mathbf{h}^{\mathcal{S}}_{v^+})/\tau)}{\exp(g(\mathbf{h}^{\mathcal{S}}_u, \mathbf{h}^{\mathcal{S}}_{v^+})/\tau) + \exp(g(\mathbf{h}^{\mathcal{S}}_u, \mathbf{h}^{\mathcal{S}}_{v^-})/\tau)} \right] + \lambda \cdot \sum\nolimits_u \mathcal{R}^u_{\text{MI}}, \quad (8)$$

where λ is the trade-off hyperparameter. $\mathbf{\Theta}^\star$ is fixed once pre-training converges.

4.2 Vocabulary Modeling with Generative Graphon Experts

We denote each vocabulary extracted from the pre-training dataset $G^{\mathcal{S}}$ as a tuple $(\boldsymbol{\mathcal{A}}_i, \boldsymbol{\mathcal{X}}_i, \mathcal{Y}_i, G^{\mathcal{S}})$, where $\boldsymbol{\mathcal{A}}_i$ and $\boldsymbol{\mathcal{X}}_i$ represent its adjacency and node feature matrix, $\mathcal{Y}_i$ is class label. To enhance the generalization of disentangled vocabularies, enabling efficient and robust generative augmentation during fine-tuning, we establish a ***"vocabulary bank"*** equipped with two complementary graphon experts: $W^{\boldsymbol{\mathcal{A}}}$ dedicated to modeling vocabularies by structure tokens and $W^{\boldsymbol{\mathcal{X}}}$ to feature tokens.

Structure Token Modeling. For each class $c \in \{1, \cdots, \mathcal{C}\}$, we collect adjacencies $\{\boldsymbol{\mathcal{A}}^{(c)}_i\}^{N_c}_{i=1}$ for each disentangled vocabulary, and approximate the structure token via a nonparametric graphon:

$$W^{\boldsymbol{\mathcal{A}}}_c \colon [0,1]^2 \mapsto [0,1], \quad W^{\boldsymbol{\mathcal{A}}}_c(u, v) = \frac{1}{N_c} \sum\nolimits_{i=1}^{N_c} \boldsymbol{\mathcal{A}}^{(c)}_i[\pi_i(u), \pi_i(v)], \quad (9)$$

where $\pi_i \colon [0,1] \mapsto \{1, \cdots, |\boldsymbol{\mathcal{A}}_i|\}$ is a measure-preserving mapping estimated via empirical node ordering (*e.g.*, node degree). $|\boldsymbol{\mathcal{A}}_i|$ denotes the number of nodes in this vocabulary structure token, and we align each vocabulary with the same number of nodes by utilizing node padding [21, 7].

Feature Token Modeling. Simultaneously, for each class c, we estimate a feature token graphon to model the distribution of node features conditioned on latent positions (coordinates):

$$W^{\boldsymbol{\mathcal{X}}}_c \colon [0,1] \mapsto \mathbb{R}^d, \quad W^{\boldsymbol{\mathcal{X}}}_c(u) = \frac{1}{N_c} \sum\nolimits_{i=1}^{N_c} \boldsymbol{\mathcal{X}}^{(c)}_i[\pi_i(u), :], \quad (10)$$

where π_i is shared with $W^{\boldsymbol{\mathcal{A}}}_c$. This joint token-level parameterization establishes a critical foundation for a hierarchical collaboration mechanism introduced in the next section. By coherently generating both structure and feature tokens, the unified modeling facilitates more stable knowledge transfer.

Conditional Vocabulary Generation. In linguistics, a word typically consists of a ***root*** and ***affixes***, where the root conveys semantics, and the affixes determine its grammatical or functional properties. In a similar vein, a graph vocabulary follows a comparable ***"root + affix"*** paradigm: ***root*** (structures) determines its structural semantics (triad, grid, tree, *etc.*), and ***affix*** (features) represents its domain-specific properties. For instance, a triad can indicate a stable social relationship, but in molecular contexts, it can indicate chemical instability. Based on this understanding, we next illustrate how to generate a graph vocabulary conditioned on a given class within a specific domain.

For dataset $G^{\mathcal{S}}$, given a class label c, we can sample $(\widetilde{\boldsymbol{\mathcal{A}}}, \widetilde{\boldsymbol{\mathcal{X}}})$ as a new synthetic graph vocabulary by first drawing n' latent positions: $\mathbf{u}_1, \cdots, \mathbf{u}_{n'} \sim \mathcal{U}[0,1]$, where $\mathcal{U}$ is the uniform distribution. Then:

$$\widetilde{\boldsymbol{\mathcal{A}}}[i,j] \sim \mathrm{Bern}\left(W^{\boldsymbol{\mathcal{A}}}_c(\mathbf{u}_i, \mathbf{u}_j)\right), \ \widetilde{\boldsymbol{\mathcal{X}}}[i,:] = W^{\boldsymbol{\mathcal{X}}}_c(\mathbf{u}_i), \quad (11)$$

where Bern denotes the Bernoulli distribution. The generated vocabulary inherits both the structural priors and semantic traits of the specified class and domain. We depict this process in Figure 2 (right).

We theoretically demonstrate that the generated vocabularies converge in distribution to their empirical distribution, ensuring transferability and semantic consistency. We formulate this in Proposition 2.

Proposition 2 (Generation Distributional Convergence). *Let $\mathbf{g}_c^{\text{emp}}$ be the empirical distribution over n'-node vocabulary subgraphs of class c, collected from the disentangled vocabulary bank:*

$$\mathbf{g}_c^{\text{emp}} := \frac{1}{N_c}\sum\nolimits_{i=1}^{N_c} \delta_{(\boldsymbol{\mathcal{A}}_i^{(c)}, \boldsymbol{\mathcal{X}}_i^{(c)})}, \quad \text{each } (\boldsymbol{\mathcal{A}}_i^{(c)}, \boldsymbol{\mathcal{X}}_i^{(c)}) \in \{0,1\}^{n'\times n'} \times \mathbb{R}^{n'\times d}, \tag{12}$$

where $\delta_{(\cdot)}$ denotes Dirac measure, representing a point mass probability distribution. Assume the true underlying structure and feature functions $W_c^{\boldsymbol{\mathcal{A}}\star}$, $W_c^{\boldsymbol{\mathcal{X}}\star}$ are bounded, Lipschitz, and satisfy permutation equivariance. If the estimators $W_c^{\boldsymbol{\mathcal{A}}}$, $W_c^{\boldsymbol{\mathcal{X}}}$ converge uniformly in L_∞ norm, then the total variation (TV) distance between $\mathbf{g}_c^{\text{gen}} = (\widetilde{\boldsymbol{\mathcal{A}}}_c, \widetilde{\boldsymbol{\mathcal{X}}}_c)$ and $\mathbf{g}_c^{\text{emp}}$ vanishes as $N_c \to \infty$:

$$\|\mathbf{g}_c^{\text{gen}} - \mathbf{g}_c^{\text{emp}}\|_{\text{TV}} \to 0. \quad \text{(Proof C.4)} \tag{13}$$

4.3 Fine-tuning with Augmented Support Samples

Given any sample (node or graph) $G_i^{\mathcal{T}} = (\mathcal{V}_i, \mathcal{E}_i)$ $(i \leqslant m)$ with feature $\mathbf{X}_i^{\mathcal{T}} \in \mathbb{R}^{N_i \times d_i}$ from the target domain $D_j \in \{D^{\mathcal{T}}\}$, we utilize the shared $\text{LLM}(\cdot)$, aligners $\mathbf{W}_i$, $\mathcal{F}$ to obtain unified $\widehat{\mathbf{X}}_i^{\mathcal{T}} \in \mathbb{R}^{N_i \times d}$.

MoE-CoE Network for Selective Augmentation. Fine-tuning GFMs under few-shot settings is highly non-trivial. The scarcity of labeled samples makes it difficult to adapt large pre-trained models by tuning limited parameters, often resulting in unstable performance and overfitting. To alleviate this, we explicitly incorporate transferable knowledge from source domains while avoiding negative transfer, through a hierarchical routing mechanism termed as MoE-CoE. We propose a two-stage architecture consisting of a Mixture-of-Experts (MoE) layer and a Collaboration-of-Experts (CoE) layer, which together determine how pre-trained vocabulary tokens are reused for target adaptation.

MoE addresses the question of ***where to route*** by selecting relevant domains, where routing weights are assigned across n source-specific vocabulary banks to enable coarse-level domain alignment. CoE addresses the question of ***how to compose*** by promoting collaboration among class-level token experts within each selected domain, which allows multiple tokens to contribute jointly, capturing complementary semantics and enhancing representation coverage. Specifically, assigned weights are:

$$\mathbf{S}_{\text{M}} \in \mathbb{R}^n = \text{Softmax}\left(\mathbf{W}_{\text{M}}^\top \cdot \phi(\widehat{\mathbf{X}}_i^{\mathcal{T}})\right), \quad \mathbf{S}_{\text{C}} \in \mathbb{R}^{\mathcal{C}} = \text{Softmax}\left(\mathbf{W}_{\text{C}}^\top \cdot (\phi(\widehat{\mathbf{X}}_i^{\mathcal{T}} \| \widetilde{\boldsymbol{\mathcal{X}}})\right), \tag{14}$$

where $\mathbf{W}_{\text{M}}$ and $\mathbf{W}_{\text{C}}$ are projections, $\phi(\cdot)$ is a learnable network. The composed vocabulary is:

$$\widetilde{\mathbf{g}}_i^{\text{gen}} \overset{\text{def}}{=} (\widetilde{\boldsymbol{\mathcal{A}}}_i^{\text{gen}}, \widetilde{\boldsymbol{\mathcal{X}}}_i^{\text{gen}}) = \sum\nolimits_{i=1}^{n} \mathbf{S}_{\text{M},i}^\top \Big(\sum\nolimits_{c=1}^{\mathcal{C}} \mathbf{S}_{\text{C},c}^\top \cdot \mathbf{g}_c^{\text{gen}} \Big), \quad \widetilde{G}_i^{\mathcal{T}} = G_i^{\mathcal{T}} \oplus \widetilde{\mathbf{g}}_i^{\text{gen}}, \tag{15}$$

where $\oplus$ denotes augmenting the support sample G_i with the generated vocabulary by overlapping on the max-degree node for aggregation-enhanced alignment. To mitigate load imbalance and encourage sparse activation in the MoE-CoE routing architecture, the weight assignment vectors $\mathbf{S}_{\text{M}}$, $\mathbf{S}_{\text{C}}$ are optimized in a supervised manner by minimizing their selective uncertainty:

$$\mathcal{L}_{\text{MoE-CoE}}(\mathbf{S}_{\text{M}}, \mathbf{S}_{\text{C}}) = -\sum\nolimits_{i=1}^{n} \mathbf{S}_{\text{M},i}^\top \log(\mathbf{S}_{\text{M},i}) - n \sum\nolimits_{c=1}^{\mathcal{C}} \mathbf{S}_{\text{C},c}^\top \log(\mathbf{S}_{\text{C},c}). \tag{16}$$

Objectives of Few-shot Fine-tuning. Motivated by prompt learning [6], we introduce learnable graph prompts as few-shot adapters to retrieve relevant knowledge:

$$\mathbf{H}_i^{\mathcal{T}} = f_{\boldsymbol{\Theta}}^\star(\boldsymbol{\mathcal{P}}_{\boldsymbol{\Omega}} \odot \widetilde{G}_i^{\mathcal{T}}), \tag{17}$$

where $\boldsymbol{\mathcal{P}}$ is a graph prompt initialized with tunable parameter $\boldsymbol{\Omega}$, optimized over m-shot augmented support samples. The graph prompt serves as a lightweight adapter, selectively injecting transferable semantics while preserving domain specificity. To support both node- and graph-level classification, we adopt a unified task template across pre-training and fine-tuning based on ego-graph similarity.

Given m $(m \ll \sum_{i=1}^{n} N_i)$ vocabulary-augmented support samples $\{(\widetilde{G}_i^{\mathcal{T}}, Y_i^{\mathcal{T}})\}_{i=1}^m$, where $\mathbf{H}^{\mathcal{T}}$ denotes the node (for node task) or ego-graph embeddings (for graph task), the fine-tuning objective is transformed into determining the similarity between the query sample and class prototype embedding:

$$\mathcal{L}_{\text{cls}}(\boldsymbol{\Omega}; \mathbf{H}^{\mathcal{T}}) = -\sum_{(\widetilde{G}_i^{\mathcal{T}}, Y_i^{\mathcal{T}})} \left[\log \frac{\exp\left(g(\mathbf{H}_i^{\mathcal{T}}, \overline{\mathbf{H}}_{Y_i}^{\mathcal{T}})/\tau\right)}{\sum_{Y_j \in \{Y^{\mathcal{T}}\}} \exp\left(g(\mathbf{H}_i^{\mathcal{T}}, \overline{\mathbf{H}}_{Y_j}^{\mathcal{T}})/\tau\right)} \right], \tag{18}$$

where $\overline{\mathbf{H}}_{Y_i}^{\mathcal{T}}$ is the class prototype for samples in class Y_i. The overall fine-tuning objective is:

$$\mathcal{L}_{\text{ftn}} = \mathcal{L}_{\text{cls}}(\boldsymbol{\Omega}; \mathbf{H}^{\mathcal{T}}) + \mu \cdot \mathcal{L}_{\text{MoE-CoE}}(\mathbf{S}_{\text{M}}, \mathbf{S}_{\text{C}}), \tag{19}$$

where μ is the trade-off hyperparameter.

https://doi.org/10.52202/085713-0560

5 Experiments

In this section, we conduct extensive experiments to evaluate ***effectiveness***, ***efficiency***, and ***robustness*** of the proposed GRAVER[2]. We focus on the following research questions:

- **RQ1:** How effective and robust is the few-shot classification fine-tuning? (▷ Section 5.2)
- **RQ2:** Which module contributes most to the classification performance? (▷ Section 5.3)
- **RQ3:** How time-efficient in the few-shot fine-tuning phase? (▷ Section 5.4)
- **RQ4:** How LLMs enhance domain alignment to enable zero-shot transferability? (▷ Section 5.5)
- **RQ5:** How sensitive is the performance to hyperparameter fluctuations? (▷ Section 5.6)

5.1 Experimental Settings

Datasets. To highlight the multi-domain pre-training capability of GFMs, we select **seven** benchmark graph datasets from **three** distinct domains. It is worth noting that this differs from conventional experimental settings, which typically treat a single dataset as a domain. Specifically:

- **Citation Domain:** `Cora` [65], `CiteSeer` [16], `PubMed` [73], and the large-scale `ogbn-arXiv` [28].
- **Co-purchase Domain:** `ogbn-Tech` and `ogbn-Home` from large-scale co-purchase network [28].
- **Web Link Domain:** `Wiki-CS` [66], a hyperlink network constructed from a subset of Wikipedia.

Baselines. We compare GRAVER with **15** state-of-the-art baselines from **four** primary categories.

- **Vanilla Graph Neural Networks:** including `GCN` [40] and `GAT` [91] without pre-training.
- **Self-Supervised Graph Pre-training:** `GCC` [69], `DGI` [92], `InfoGraph` [81], `GraphCL` [116], `DSSL` [102], and `GraphACL` [103], which utilize self-supervised objectives to pre-train and finetune.
- **Prompt-based Graph Fine-tuning:** we select pioneering prompt-based fine-tuning baselines including `GPPT` [82], `GraphPrompt` [58], `GPF` [14], and `ProNoG` [121].
- **Multi-Domain Graph Pre-training:** we select the most closely related baselines on multi-domain graph pre-training and finetuning, including `GCOPE` [136], `MDGPT` [123], and `SAMGPT` [119].

Multi-Domain Pre-training Settings. We mainly focus on the challenging task of adapting GFMs to unseen datasets and domains in downstream applications. To better characterize the setting, we explicitly distinguish between ***cross-dataset*** and ***cross-domain*** adaptation:

- **Cross-Dataset:** pre-training and fine-tuning on ***different*** datasets, which are from the ***same*** domain. Specifically, take **Citation**, **Co-purchase**, and **Web Link** as the pre-training domains. When each dataset is used as the target alternatively, the remainings are collectively used as the source datasets.
- **Cross-Domain:** extends to a more challenging scenario, where the target dataset belongs to an ***unseen*** domain different from the source domain. Specifically, we alternately set **Citation**, **Co-purchase**, and **Web Link** as the target domain, while the remaining two domains are the source.

Few-shot Fine-tuning Settings. We evaluate node and graph classification under the m-shot setting, where m labeled samples per class are randomly selected. As each dataset contains a single large graph, we follow prior work [61, 118, 123] to extract ego-graphs centered on target nodes for graph classification, assigning labels based on the central nodes. Accuracy (Acc.) is used for evaluation.

5.2 RQ1: Few-Shot Node and Graph Classification

One-Shot Classification (Table 1)**.** ❶ Overall, GRAVER achieves the best accuracy, demonstrating a clear superiority. Specifically, it outperforms the runner-up with average gains of 2.8% (node) and 3.2% (graph), which are consistent across both dataset and domain transfer scenarios, indicating effective source knowledge utilization even under extreme one-shot conditions. ❷ Compared to strong baselines like `SAMGPT` and `MDGPT`, GRAVER achieves significantly higher performance by over 4% in both tasks. The performance gap widens further in harder domains like `ogbn-arXiv` and `Wiki-CS`, demonstrating its broad applicability. Notably, GRAVER still leads with large margins in the more challenging cross-domain settings due to domain shift. ❸ On robustness, GRAVER consistently exhibits an obviously smaller standard deviation compared to baselines with an average relative decrease of 54.0% (node) and 54.6% (graph). This suggests GRAVER maintains stable performance regardless of the random support set selection, thanks to the generative vocabulary augmentations, and is beneficial for real-world deployment. ❹ Additional results under the **five-shot** setting, presented in Appendix F.1, consistently exhibit the same performance trend as observed in the one-shot scenario.

[2]Codes are available at: https://github.com/RingBDStack/GRAVER.

Table 1: Accuracy (% ± std for 20 runs) of **one-shot classification**. Best results are presented in **bold** and the runner-ups are underlined. CR = Cora, CS = CiteSeer, PM = PubMed, arXiv = ogbn-arXiv, Tech = ogbn-tech, Home = ogbn-home, Wiki = Wiki-CS. Color denotes domain.

Source	Cross-Dataset				Cross-Domain		
	CS PM Home Wiki	CR PM Home Wiki	CR CS Home Wiki	CR CS PM Home Wiki	CR CS PM Wiki	CR CS PM Home	Home Tech Wiki
Model / Target	CR	CS	PM	Tech	Home	Wiki	arXiv
Node Classification							
GCN (bb.) [40]	28.40 ± 4.62	29.25 ± 3.39	40.33 ± 6.90	61.59 ± 5.13	53.89 ± 3.35	36.74 ± 2.53	28.58 ± 5.39
GAT [91]	29.72 ± 5.17	29.31 ± 3.47	40.51 ± 3.95	61.98 ± 5.23	51.70 ± 3.96	36.24 ± 4.19	29.03 ± 5.75
GCC [69]	32.47 ± 4.55	32.78 ± 3.85	41.66 ± 3.27	64.74 ± 3.78	55.36 ± 5.86	37.66 ± 3.80	30.42 ± 5.54
DGI [92]	30.77 ± 3.92	31.41 ± 4.11	39.97 ± 5.90	63.76 ± 3.77	53.31 ± 2.58	39.20 ± 5.67	32.15 ± 4.91
GraphCL [116]	33.64 ± 5.75	28.20 ± 3.13	39.03 ± 8.67	62.44 ± 6.55	51.55 ± 8.20	38.05 ± 3.30	31.81 ± 5.04
DSSL [102]	29.76 ± 4.55	30.84 ± 7.62	39.99 ± 6.29	61.27 ± 5.21	51.90 ± 6.52	37.58 ± 6.78	28.14 ± 5.17
GraphACL [103]	35.92 ± 5.61	33.88 ± 6.69	42.73 ± 5.26	66.70 ± 4.29	58.01 ± 6.32	40.94 ± 6.65	35.57 ± 4.29
GPPT [82]	32.38 ± 5.74	31.78 ± 4.61	41.97 ± 5.48	65.81 ± 6.56	57.81 ± 5.36	40.97 ± 5.41	34.22 ± 6.51
GraphPrompt [58]	38.02 ± 6.52	34.57 ± 6.34	46.19 ± 7.63	70.11 ± 6.62	60.99 ± 7.19	44.02 ± 6.74	40.74 ± 6.56
GPF [14]	40.01 ± 8.21	40.07 ± 7.64	47.58 ± 5.83	70.07 ± 6.55	59.05 ± 5.85	44.62 ± 8.68	40.13 ± 5.07
ProNoG [121]	43.67 ± 6.33	40.34 ± 7.11	51.35 ± 7.16	73.49 ± 7.34	62.77 ± 6.10	45.65 ± 6.44	41.15 ± 4.21
GCOPE [136]	36.27 ± 3.93	40.42 ± 4.64	44.75 ± 4.67	71.26 ± 5.95	60.39 ± 6.08	41.80 ± 4.96	40.00 ± 6.53
MDGPT [123]	42.55 ± 6.84	37.92 ± 7.18	51.03 ± 8.99	72.10 ± 7.12	62.69 ± 7.29	45.93 ± 6.24	43.33 ± 5.75
SAMGPT [119]	46.79 ± 6.54	38.65 ± 6.35	51.92 ± 9.50	73.60 ± 7.55	64.32 ± 7.02	46.03 ± 6.98	45.27 ± 5.05
GRAVER (ours)	**48.00 ± 2.52**	**41.13 ± 3.00**	**55.30 ± 3.03**	**77.62 ± 2.94**	**67.12 ± 3.18**	**49.33 ± 2.44**	**49.25 ± 3.43**
Graph Classification							
GCN (bb.) [40]	40.07 ± 4.78	29.53 ± 5.74	45.25 ± 7.32	81.16 ± 4.56	58.82 ± 2.48	38.46 ± 4.47	46.05 ± 4.42
GAT [91]	36.01 ± 5.09	25.98 ± 7.84	41.03 ± 5.77	77.75 ± 4.92	59.12 ± 5.97	38.49 ± 5.11	40.80 ± 6.87
InfoGraph [81]	42.18 ± 5.17	30.18 ± 4.11	49.10 ± 5.42	81.62 ± 8.35	63.31 ± 3.14	40.98 ± 4.24	43.96 ± 6.48
GraphCL [116]	39.56 ± 5.79	32.61 ± 6.54	47.71 ± 7.04	82.27 ± 4.46	63.77 ± 4.66	46.14 ± 3.10	47.61 ± 5.02
DSSL [102]	41.46 ± 7.91	32.92 ± 5.24	47.28 ± 7.71	83.66 ± 4.99	60.56 ± 5.39	41.09 ± 5.18	47.50 ± 5.20
GraphACL [103]	40.60 ± 5.00	33.17 ± 7.53	50.48 ± 5.77	80.98 ± 3.78	62.38 ± 8.05	45.01 ± 5.62	46.43 ± 6.32
GraphPrompt [58]	41.71 ± 3.45	38.76 ± 7.06	50.61 ± 6.39	84.23 ± 2.43	61.37 ± 5.64	43.98 ± 8.22	47.98 ± 5.40
GPF [14]	41.25 ± 9.41	38.18 ± 8.22	47.83 ± 8.17	83.72 ± 5.42	67.63 ± 4.72	47.16 ± 5.08	47.27 ± 4.16
ProNoG [121]	51.72 ± 8.20	39.87 ± 5.89	54.13 ± 8.93	82.55 ± 3.19	68.95 ± 4.20	47.20 ± 4.83	52.11 ± 5.58
GCOPE [136]	55.91 ± 7.37	40.99 ± 9.01	54.36 ± 8.63	82.58 ± 2.39	67.45 ± 4.39	49.26 ± 5.25	62.90 ± 3.64
MDGPT [123]	52.77 ± 6.71	41.01 ± 9.72	55.47 ± 8.27	83.91 ± 4.06	70.46 ± 5.21	50.24 ± 7.49	59.29 ± 4.25
SAMGPT [119]	53.25 ± 4.28	42.39 ± 7.28	57.66 ± 6.34	85.08 ± 6.39	73.69 ± 5.06	52.10 ± 3.65	62.28 ± 4.05
GRAVER (ours)	**59.20 ± 2.38**	**45.50 ± 2.00**	**61.70 ± 2.58**	**87.33 ± 1.84**	**75.46 ± 2.24**	**55.90 ± 2.33**	**67.10 ± 3.26**

m-Shot Classification (Figure 3)**.** To assess performance under varying m, we compare with three strongest baselines from Table 1: GCOPE, MDGPT, and SAMGPT. Results show: ❶ GRAVER consistently outperforms three baselines across nearly all m-shot settings. ❷ The advantage is most pronounced under low-shot conditions ($m \leqslant 4$), where the generative vocabulary introduces semantically diverse and transferable patterns to enhance adaptation. As m increases, the relative gain narrows due to reduced augmentation benefits and increasing noise. ❸ Similar trends are observed in both cross-dataset and cross-domain scenarios, confirming GRAVER's generalizability across transfer settings. ❹ Additional results of **graph classification** in Appendix F.1 further support these findings.

Robustness against Random Noise (Figure 4)**.** To evaluate robustness under perturbations, we add Gaussian feature noise and random edge modifications to support sets, controlled by parameters λ_f and λ_s. Results show: ❶ GRAVER maintains superior performance under both feature and structure noise, showing robustness to both support sample variability and direct corruption. ❷ While baselines degrade more rapidly with increasing noise, especially in structure, GRAVER shows only gradual decline, or even slight increase, highlighting the stabilizing effect of vocabulary-guided augmentation. ❸ The widening performance gap under higher noise levels indicates that the MoE-CoE architecture effectively filters out irrelevant patterns and reinforces useful knowledge for robust fine-tuning.

5.3 RQ2: Ablation Study

We conduct ablation studies on **three** core components of GRAVER: the semantic-independence promotion (*SIP*), the vocabulary augmentation (*VA*), and the MoE-CoE network (*MC*). Results (Figure 5) show that: ❶ Removing *SIP* leads to a minor yet consistent drop, indicating that encouraging semantic diversity across vocabulary channels is beneficial, though not dominant. ❷ Disabling *VA* causes a major decline, highlighting the importance of generative subgraphs in enhancing support quality under low-shot conditions. ❸ Replacing *MC* with uniform fusion results in significant degradation, confirming that selective routing is crucial for injecting relevant knowledge while suppressing noise. ❹ Additional results on other datasets in Appendix F.2 demonstrate similar conclusions.

https://doi.org/10.52202/085713-0560

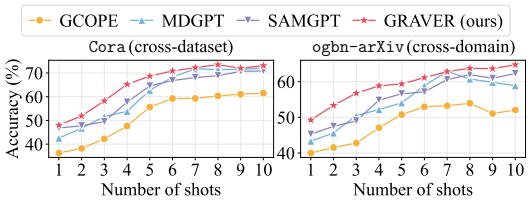


Figure 3: m-shot node classification.

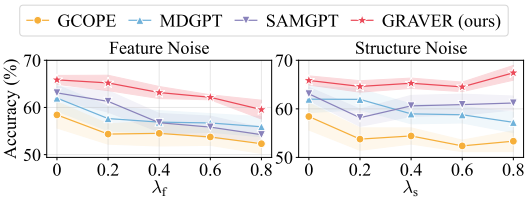


Figure 4: 5-shot graph classification (Wiki-CS).

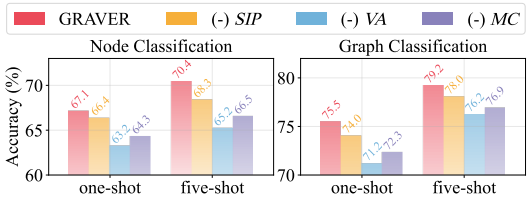


Figure 5: Ablation studies (ogbn-Home).

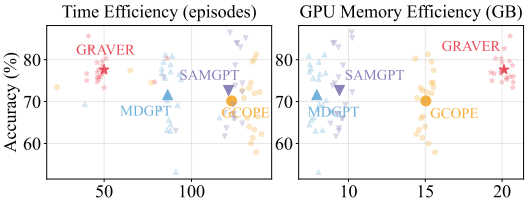


Figure 6: Efficiency analysis (ogbn-Tech).

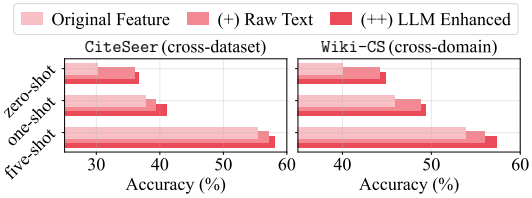


Figure 7: Analysis on LLM (node classification).

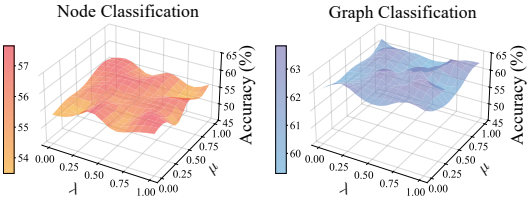


Figure 8: Hyperparameter sensitivity (PubMed).

5.4 RQ3: Efficiency Evaluation for Fine-tuning

In addition to the theoretical complexity analysis presented in Appendix B, we empirically analyze the time and memory efficiency of one-shot node classification, where the results (Figure 6) show that: ❶ GRAVER converges in nearly half the episodes compared to baselines while achieving higher accuracy, demonstrating the efficiency of its generative vocabulary and lightweight MoE-CoE routing design. ❷ Though GRAVER consumes more GPU memory due to maintaining and routing richer vocabulary tokens, the trade-off is acceptable given its superior performance and faster convergence.

5.5 RQ4: LLM-Enhanced Multi-Domain Alignment

We enhance node features by incorporating raw text and LLM-generated class-aware descriptions. Results (Figure 7) indicate that, raw text alone brings substantial improvement in zero-shot settings, as it serves as a domain-agnostic semantic bridge when no labeled data is available. This highlights the crucial role of textual information in aligning feature spaces across domains purely through inherent semantics. Building on this, LLM-enhanced features further boost low-shot performance by injecting class-discriminative signals. These improvements are consistent across both cross-dataset and cross-domain tasks, underscoring the effectiveness of text-based feature enhancement.

5.6 RQ5: Hyperparameter Sensitivity Investigation

We evaluate the sensitivity of GRAVER to two important hyperparameters: λ (for $\mathcal{R}_{\text{MI}}$ in Eq. (8)) and μ (for $\mathcal{L}_{\text{MoE-CoE}}$ in Eq. (19)). Results (Figure 8) show that accuracy reveals only mild fluctuations within a broad range of values for both node and graph classification tasks. This indicates GRAVER is robust to hyperparameter combinations. We provide detailed hyperparameter settings in Appendix E.

6 Conclusion

We propose **GRAVER**, a trustworthy graph foundation model that enables robust and efficient fine-tuning under multi-domain pre-training and cross-domain adaptation. To our knowledge, this is the first framework to leverage generative graph vocabularies for in-context support set augmentation to enhance adaptation stability. GRAVER introduces graphon-based generative experts to synthesize structure-feature vocabularies aligned with transferable subgraphs, and employs a hierarchical MoE-CoE to dynamically assemble source-domain knowledge for context-aware augmentation. Extensive experiments demonstrate its multi-task superiority over 15 state-of-the-art baselines.

https://doi.org/10.52202/085713-0560

Acknowledgments

The corresponding author is Qingyun Sun. This work is supported in part by NSFC under grants No.623B2010, No.62225202, and No.62302023, by NSF under grants III-2106758 and POSE-2346158, by the Fundamental Research Funds for the Central Universities, and by the Academic Excellence Foundation of BUAA for PhD Students. We extend our sincere thanks to all authors for their valuable contributions.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Edo M Airoldi, Thiago B Costa, and Stanley H Chan. Stochastic blockmodel approximation of a graphon: Theory and consistent estimation. *NeurIPS*, 26, 2013.

[3] Emily Alsentzer, Samuel Finlayson, Michelle Li, and Marinka Zitnik. Subgraph neural networks. *NeurIPS*, 33:8017–8029, 2020.

[4] Maciej Besta, Raphael Grob, Cesare Miglioli, Nicola Bernold, Grzegorz Kwasniewski, Gabriel Gjini, Raghavendra Kanakagiri, Saleh Ashkboos, Lukas Gianinazzi, Nikoli Dryden, et al. Motif prediction with graph neural networks. In *KDD*, pages 35–45, 2022.

[5] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020.

[7] 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 *KDD*, pages 142–153, 2023.

[8] Ke-Jia Chen, Jiajun Zhang, Linpu Jiang, Yunyun Wang, and Yuxuan Dai. Pre-training on dynamic graph neural networks. *Neurocomputing*, 500:679–687, 2022.

[9] Zhikai Chen, Haitao Mao, Jingzhe Liu, Yu Song, Bingheng Li, Wei Jin, Bahare Fatemi, Anton Tsitsulin, Bryan Perozzi, Hui Liu, et al. Text-space graph foundation models: Comprehensive benchmarks and new insights. *NeurIPS*, 37:7464–7492, 2024.

[10] Yao Cheng, Yige Zhao, Jianxiang Yu, and Xiang Li. Boosting graph foundation model from structural perspective. *arXiv preprint arXiv:2407.19941*, 2024.

[11] Yu Du, Fangyun Wei, Zihe Zhang, Miaojing Shi, Yue Gao, and Guoqi Li. Learning to prompt for open-vocabulary object detection with vision-language model. In *CVPR*, pages 14084–14093, 2022.

[12] Yutai Duan, Jie Liu, Shaowei Chen, Liyi Chen, and Jianhua Wu. G-Prompt: Graphon-based prompt tuning for graph classification. *Information Processing & Management*, 61(3):103639, 2024.

[13] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *WWW*, pages 417–426, 2019.

[14] Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. Universal prompt tuning for graph neural networks. *NeurIPS*, 36, 2024.

[15] Ziqi Gao, Chenran Jiang, Jiawen Zhang, Xiaosen Jiang, Lanqing Li, Peilin Zhao, Huanming Yang, Yong Huang, and Jia Li. Hierarchical graph learning for protein–protein interaction. *Nature Communications*, 14(1):1093, 2023.

[16] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital libraries*, pages 89–98, 1998.

[17] Vladimir Gligorijević, P Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C Taylor, Ian M Fisk, Hera Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature Communications*, 12(1):3168, 2021.

 https://doi.org/10.52202/085713-0560

[18] Chenghua Gong, Xiang Li, Jianxiang Yu, Cheng Yao, Jiaqi Tan, Chengcheng Yu, and Dawei Yin. Prompt tuning for multi-view graph contrastive learning. *arXiv preprint arXiv:2310.10362*, 2023.

[19] Anchun Gui, Jinqiang Ye, and Han Xiao. G-Adapter: Towards structure-aware parameter-efficient transfer learning for graph transformer networks. *AAAI*, 38(11):12226–12234, 2024.

[20] Zihao Guo, Qingyun Sun, Haonan Yuan, Xingcheng Fu, Min Zhou, Yisen Gao, and Jianxin Li. GraphMoRE: Mitigating topological heterogeneity via mixture of riemannian experts. In *AAAI*, volume 39, pages 11754–11762, 2025.

[21] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. G-Mixup: Graph data augmentation for graph classification. In *ICML*, pages 8230–8248. PMLR, 2022.

[22] Yufei He, Zhenyu Hou, Yukuo Cen, Feng He, Xu Cheng, and Bryan Hooi. Generalizing graph transformers across diverse graphs and tasks via pre-training on industrial-scale data. *arXiv preprint arXiv:2407.03953*, 2024.

[23] Yufei He, Yuan Sui, Xiaoxin He, and Bryan Hooi. UniGraph: Learning a unified cross-domain foundation model for text-attributed graphs. In *KDD*, pages 448–459, 2025.

[24] Van Thuy Hoang and O-Joun Lee. Pre-training graph neural networks on molecules by using subgraph-conditioned graph information bottleneck. In *AAAI*, volume 39, pages 17204–17213, 2025.

[25] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994.

[26] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. GraphMAE: Self-supervised masked graph autoencoders. In *KDD*, pages 594–604, 2022.

[27] Fenyu Hu, Liping Wang, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. GraphDIVE: Graph classification by mixture of diverse experts. In *IJCAI*, pages 2080–2086, 2022.

[28] 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. *NeurIPS*, 33:22118–22133, 2020.

[29] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *ICLR*, 2020.

[30] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. GPT-GNN: Generative pre-training of graph neural networks. In *KDD*, pages 1857–1867, 2020.

[31] Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy S Liang, and Jure Leskovec. PRODIGY: Enabling in-context learning over graphs. *NeurIPS*, 36, 2024.

[32] Qian Huang, Hongyu Ren, and Jure Leskovec. Few-shot relational reasoning via connection subgraph pretraining. *NeurIPS*, 35:6397–6409, 2022.

[33] Renhong Huang, Jiarong Xu, Xin Jiang, Chenglu Pan, Zhiming Yang, Chunping Wang, and Yang Yang. Measuring task similarity and its implication in fine-tuning graph neural networks. *AAAI*, 38(11):12617–12625, 2024.

[34] Kanchan Jha, Sriparna Saha, and Hiteshi Singh. Prediction of protein-protein interaction using graph neural networks. *Scientific Reports*, 12(1):8360, 2022.

[35] Bo Jiang, Hao Wu, Beibei Wang, Jin Tang, and Bin Luo. Reliable and compact graph fine-tuning via graphsparse prompting. *arXiv preprint arXiv:2410.21749*, 2024.

[36] Xunqiang Jiang, Yuanfu Lu, Yuan Fang, and Chuan Shi. Contrastive pre-training of gnns on heterogeneous graphs. In *CIKM*, pages 803–812, 2021.

[37] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *IEEE TKDE*, 2024.

[38] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Computing Surveys*, 54(10s):1–41, 2022.

[39] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[40] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[41] Xiangzhe Kong, Wenbing Huang, Zhixing Tan, and Yang Liu. Molecule generation by principal subgraph mining and assembling. *NeurIPS*, 35:2550–2563, 2022.

[42] Divyansha Lachi, Mehdi Azabou, Vinam Arora, and Eva Dyer. GraphFM: A scalable framework for multi-graph pretraining. *arXiv preprint arXiv:2407.11907*, 2024.

[43] Jaekoo Lee, Hyunjae Kim, Jongsun Lee, and Sungroh Yoon. Transfer learning for deep learning on graph-structured data. In *AAAI*, volume 31, 2017.

[44] Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. Disentangled contrastive learning on graphs. *NeurIPS*, 34:21872–21884, 2021.

[45] Jia Li, Xiangguo Sun, Yuhan Li, Zhixun Li, Hong Cheng, and Jeffrey Xu Yu. Graph intelligence with large language models and prompt learning. In *KDD*, pages 6545–6554, 2024.

[46] Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. A survey of graph meets large language model: progress and future directions. In *IJCAI*, pages 8123–8131, 2024.

[47] Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. ZeroG: Investigating cross-dataset zero-shot transferability in graphs. In *KDD*, pages 1725–1735, 2024.

[48] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *ICML*, pages 3835–3845. PMLR, 2019.

[49] Mingkai Lin, Wenzhong Li, Xiaobin Hong, and Sanglu Lu. Scalable multi-source pre-training for graph neural networks. In *ACM MM*, pages 1292–1301, 2024.

[50] Xuan Lin, Zhe Quan, Zhi-Jie Wang, Tengfei Ma, and Xiangxiang Zeng. KGNN: knowledge graph neural network for drug-drug interaction prediction. In *IJCAI*, pages 2739–2745, 2021.

[51] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *ICLR*, 2024.

[52] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. Towards graph foundation models: A survey and beyond. *arXiv preprint arXiv:2310.11829*, 2023.

[53] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. Graph foundation models: Concepts, opportunities and challenges. *IEEE TPAMI*, 2025.

[54] Jingzhe Liu, Haitao Mao, Zhikai Chen, Wenqi Fan, Mingxuan Ju, Tong Zhao, Neil Shah, and Jiliang Tang. One model for one graph: A new perspective for pretraining with cross-domain graphs. *arXiv preprint arXiv:2412.00315*, 2024.

[55] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[56] Yanbei Liu, Xiao Wang, Shu Wu, and Zhitao Xiao. Independence promoted graph disentangled networks. In *AAAI*, volume 34, pages 4916–4923, 2020.

[57] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. Graph self-supervised learning: A survey. *IEEE TKDE*, 35(6):5879–5900, 2022.

[58] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. GraphPrompt: Unifying pre-training and downstream tasks for graph neural networks. In *WWW*, pages 417–428, 2023.

[59] Zheyuan Liu, Chunhui Zhang, Yijun Tian, Erchi Zhang, Chao Huang, Yanfang Ye, and Chuxu Zhang. Fair graph representation learning via diverse mixture-of-experts. In *WWW*, pages 28–38, 2023.

[60] Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *SIGIR*, pages 1569–1572, 2020.

[61] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. Learning to pre-train graph neural networks. *AAAI*, 35(5):4276–4284, 2021.

 https://doi.org/10.52202/085713-0560

[62] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In *ICML*, pages 4212–4221. PMLR, 2019.

[63] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Michael Galkin, and Jiliang Tang. Graph foundation models. *arXiv preprint arXiv:2402.02216*, 2024.

[64] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Mikhail Galkin, and Jiliang Tang. Position: Graph foundation models are already here. In *ICML*, 2024.

[65] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[66] Péter Mernyei and Cătălina Cangea. Wiki-CS: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

[67] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[68] Francesca Parise and Asuman Ozdaglar. Graphon games. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 457–458, 2019.

[69] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. GCC: Graph contrastive coding for graph neural network pre-training. In *KDD*, pages 1150–1160, 2020.

[70] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *NeurIPS*, 33:12559–12571, 2020.

[71] Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. *NeurIPS*, 33:1702–1712, 2020.

[72] Sayan Saha and Sanghamitra Bandyopadhyay. Graphon-Explainer: Generating model-level explanations for graph neural networks using graphons. *TMLR*, 2024.

[73] 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.

[74] Eli Shamir and Joel Spencer. Sharp concentration of the chromatic number on random graphs $G_{n,p}$. *Combinatorica*, 7:121–129, 1987.

[75] Chuan Shi, Junze Chen, Jiawei Liu, and Cheng Yang. Graph foundation model. *Frontiers of Computer Science*, 18(6), 2024.

[76] Chuan Shi, Cheng Yang, Yuan Fang, Lichao Sun, and Philip S Yu. Lecture-style tutorial: Towards graph foundation models. In *WWW*, pages 1264–1267, 2024.

[77] Reza Shirkavand and Heng Huang. Deep prompt tuning for graph transformers. *arXiv preprint arXiv:2309.10131*, 2023.

[78] Razieh Shirzadkhani, Tran Gia Bao Ngo, Kiarash Shamsi, Shenyang Huang, Farimah Poursafaei, Poupak Azad, Reihaneh Rabbany, Baris Coskunuzer, Guillaume Rabusseau, and Cuneyt Gurcan Akcora. Towards neural scaling laws for foundation models on temporal graphs. *arXiv preprint arXiv:2406.10426*, 2024.

[79] Lin Song, Peter Langfelder, and Steve Horvath. Comparison of co-expression measures: mutual information, correlation, and model based indices. *BMC bioinformatics*, 13:1–21, 2012.

[80] Gilbert W Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35(4):551–566, 1993.

[81] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*, 2020.

[82] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. GPPT: Graph pre-training and prompt tuning to generalize graph neural networks. In *KDD*, pages 1717–1727, 2022.

[83] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. In *KDD*, pages 2120–2131, 2023.

[84] Yifei Sun, Qi Zhu, Yang Yang, Chunping Wang, Tianyu Fan, Jiajun Zhu, and Lei Chen. Fine-tuning graph neural networks by preserving graph generative patterns. *AAAI*, 38(8):9053–9061, 2024.

[85] Yanchao Tan, Zihao Zhou, Hang Lv, Weiming Liu, and Carl Yang. WalkLM: A uniform language model fine-tuning framework for attributed graph embedding. *NeurIPS*, 36, 2024.

[86] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. GraphGPT: Graph instruction tuning for large language models. In *SIGIR*, pages 491–500, 2024.

[87] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. HiGPT: Heterogeneous graph language model. In *KDD*, pages 2842–2853, 2024.

[88] Wenzhuo Tang, Haitao Mao, Danial Dervovic, Ivan Brugere, Saumitra Mishra, Yuying Xie, and Jiliang Tang. Cross-domain graph data scaling: A showcase with diffusion models. *arXiv preprint arXiv:2406.01899*, 2024.

[89] Shiyu Tian, Yangyang Luo, Tianze Xu, Caixia Yuan, Huixing Jiang, Chen Wei, and Xiaojie Wang. KG-Adapter: Enabling knowledge graph integration in large language models through parameter-efficient fine-tuning. In *ACL Findings*, pages 3813–3828, 2024.

[90] Howard G Tucker. A generalization of the Glivenko-Cantelli theorem. *The Annals of Mathematical Statistics*, 30(3):828–830, 1959.

[91] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengi. Graph attention networks. In *ICLR*, 2018.

[92] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

[93] Nicolas Veyrat-Charvillon and François-Xavier Standaert. Mutual information analysis: how, when and why? In *International workshop on cryptographic hardware and embedded systems*, pages 429–443. Springer, 2009.

[94] Haotao Wang, Ziyu Jiang, Yuning You, Yan Han, Gaowen Liu, Jayanth Srinivasa, Ramana Kompella, Zhangyang Wang, et al. Graph mixture of experts: Learning on large-scale graphs with explicit diversity modeling. *NeurIPS*, 36:50825–50837, 2023.

[95] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? *NeurIPS*, 36, 2024.

[96] Shuo Wang, Bokui Wang, Zhixiang Shen, Boyan Deng, and Zhao Kang. Multi-domain graph foundation models: Robust knowledge transfer via topology alignment. In *ICML*, 2025.

[97] Yuxiang Wang, Wenqi Fan, Suhang Wang, and Yao Ma. Towards graph foundation models: A transferability perspective. *arXiv preprint arXiv:2503.09363*, 2025.

[98] Zehong Wang, Zheyuan Zhang, Nitesh Chawla, Chuxu Zhang, and Yanfang Ye. GFT: Graph foundation model with transferable tree vocabulary. *NeurIPS*, 37:107403–107443, 2024.

[99] Shirley Wu, Kaidi Cao, Bruno Ribeiro, James Y Zou, and Jure Leskovec. GraphMETRO: Mitigating complex graph distribution shifts via mixture of aligned experts. *NeurIPS*, 37:9358–9387, 2024.

[100] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. *AAAI*, 33(01):346–353, 2019.

[101] Yucheng Wu, Leye Wang, Xiao Han, and Han-Jia Ye. Graph contrastive learning with cohesive subgraph awareness. In *WWW*, pages 629–640, 2024.

[102] Teng Xiao, Zhengyu Chen, Zhimeng Guo, Zeyang Zhuang, and Suhang Wang. Decoupled self-supervised learning for graphs. *NeurIPS*, 35:620–634, 2022.

[103] Teng Xiao, Huaisheng Zhu, Zhengyu Chen, and Suhang Wang. Simple and asymmetric graph contrastive learning without augmentations. *NeurIPS*, 36, 2024.

[104] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. *IEEE TPAMI*, 45(2):2412–2429, 2022.

[105] Jiarong Xu, Renhong Huang, Xin Jiang, Yuxuan Cao, Carl Yang, Chunping Wang, and Yang Yang. Better with less: A data-active perspective on pre-training graph neural networks. *NeurIPS*, 36:56946–56978, 2023.

[106] Rui Xue, Xipeng Shen, Ruozhou Yu, and Xiaorui Liu. Efficient large language models fine-tuning on graphs. *arXiv preprint arXiv:2312.04737*, 2023.

 https://doi.org/10.52202/085713-0560

[107] Pengwei Yan, Kaisong Song, Zhuoren Jiang, Yangyang Kang, Tianqianjin Lin, Changlong Sun, and Xiaozhong Liu. Empowering dual-level graph self-supervised pretraining with motif discovery. In *AAAI*, volume 38, pages 9223–9231, 2024.

[108] Yuchen Yan, Peiyan Zhang, Zheng Fang, and Qingqing Long. Inductive graph alignment prompt: Bridging the gap between graph pre-training and inductive fine-tuning from spectral perspective. In *WWW*, pages 4328–4339, 2024.

[109] Yaming Yang, Ziyu Guan, Zhe Wang, Wei Zhao, Cai Xu, Weigang Lu, and Jianbin Huang. Self-supervised heterogeneous graph pre-training based on structural clustering. *NeurIPS*, 35:16962–16974, 2022.

[110] Yuhao Yang, Lianghao Xia, Da Luo, Kangyi Lin, and Chao Huang. GraphPro: Graph pre-training and prompt learning for recommendation. In *WWW*, pages 3690–3699, 2024.

[111] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. Language is all a graph needs. In *EACL Fingdings*, pages 1955–1973, 2024.

[112] Zixuan Yi, Iadh Ounis, and Craig Macdonald. Contrastive graph prompt-tuning for cross-domain recommendation. *ACM TOIS*, 42(2):1–28, 2023.

[113] Jun Yin, Chaozhuo Li, Hao Yan, Jianxun Lian, and Senzhang Wang. Train once and explain everywhere: Pre-training interpretable graph neural networks. *NeurIPS*, 36:35277–35299, 2023.

[114] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, pages 974–983, 2018.

[115] Wangyang Ying, Dongjie Wang, Xuanming Hu, Yuanchun Zhou, Charu C Aggarwal, and Yanjie Fu. Unsupervised generative feature transformation via graph contrastive pre-training and multi-objective fine-tuning. In *KDD*, pages 3966–3976, 2024.

[116] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *NeurIPS*, 33:5812–5823, 2020.

[117] Qing Yu, Lixin Zou, Xiangyang Luo, Xiangyu Zhao, and Chenliang Li. Uniform graph pre-training and prompting for transferable recommendation. *ACM TOIS*, 2025.

[118] Xingtong Yu, Yuan Fang, Zemin Liu, and Xinming Zhang. HGPrompt: Bridging homogeneous and heterogeneous graphs for few-shot prompt learning. *AAAI*, 38(15):16578–16586, 2024.

[119] Xingtong Yu, Zechuan Gong, Chang Zhou, Yuan Fang, and Hui Zhang. SAMGPT: Text-free graph foundation model for multi-domain pre-training and cross-domain adaptation. In *WWW*, pages 1142–1153, 2025.

[120] Xingtong Yu, Zhenghao Liu, Yuan Fang, Zemin Liu, Sihong Chen, and Xinming Zhang. Generalized graph prompt: Toward a unification of pre-training and downstream tasks on graphs. *IEEE TKDE*, 2024.

[121] Xingtong Yu, Jie Zhang, Yuan Fang, and Renhe Jiang. Non-homophilic graph pre-training and prompt learning. In *KDD*, 2025.

[122] Xingtong Yu, Chang Zhou, Yuan Fang, and Xinming Zhang. MultiGPrompt for multi-task pre-training and prompting on graphs. In *WWW*, pages 515–526, 2024.

[123] 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.

[124] Haonan Yuan, Qingyun Sun, Xingcheng Fu, Cheng Ji, and Jianxin Li. Dynamic graph information bottleneck. In *WWW*, pages 469–480, 2024.

[125] Haonan Yuan, Qingyun Sun, Xingcheng Fu, Ziwei Zhang, Cheng Ji, Hao Peng, and Jianxin Li. Environment-aware dynamic graph learning for out-of-distribution generalization. *NeurIPS*, 36:49715–49747, 2023.

[126] Haonan Yuan, Qingyun Sun, Junhua Shi, Xingcheng Fu, Bryan Hooi, Jianxin Li, and Philip S Yu. How much can transfer? BRIDGE: Bounded multi-domain graph foundation model with generalization guarantees. In *ICML*, 2025.

[127] Haonan Yuan, Qingyun Sun, Zhaonan Wang, Xingcheng Fu, Cheng Ji, Yongjian Wang, Bo Jin, and Jianxin Li. DG-Mamba: Robust and efficient dynamic graph structure learning with selective state space models. In *AAAI*, volume 39, pages 22272–22280, 2025.

[128] Ge Zhang, Zhao Li, Jiaming Huang, Jia Wu, Chuan Zhou, Jian Yang, and Jianliang Gao. eFraudCom: An e-commerce fraud detection system via competitive graph neural networks. *ACM TOIS*, 40(3):1–29, 2022.

[129] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-BERT: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*, 2020.

[130] Jiying Zhang, Xi Xiao, Long-Kai Huang, Yu Rong, and Yatao Bian. Fine-tuning graph neural networks via graph topology induced optimal transport. In *IJCAI*, pages 3730–3736, 2022.

[131] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. Subgraph federated learning with missing neighbor generation. *NeurIPS*, 34:6671–6682, 2021.

[132] Peiyan Zhang, Yuchen Yan, Xi Zhang, Liying Kang, Chaozhuo Li, Feiran Huang, Senzhang Wang, and Sunghun Kim. GPT4Rec: Graph prompt tuning for streaming recommendation. In *SIGIR*, pages 1774–1784, 2024.

[133] Shichang Zhang, Ziniu Hu, Arjun Subramonian, and Yizhou Sun. Motif-driven contrastive learning of graph representations. *IEEE TKDE*, 36(8):4063–4075, 2024.

[134] Yizhuo Zhang, Heng Wang, Shangbin Feng, Zhaoxuan Tan, Xiaochuang Han, Tianxing He, and Yulia Tsvetkov. Can llm graph reasoning generalize beyond pattern memorization? In *EMNLP Findings*, pages 2289–2305, 2024.

[135] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Chee-Kong Lee. Motif-based graph self-supervised learning for molecular property prediction. *NeurIPS*, 34:15870–15882, 2021.

[136] 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 *KDD*, pages 4443–4454, 2024.

[137] Qifang Zhao, Weidong Ren, Tianyu Li, Xiaoxiao Xu, and Hong Liu. GraphGPT: Graph learning with generative pre-trained transformers. *arXiv preprint arXiv:2401.00529*, 2023.

[138] Tianxiang Zhao, Dongsheng Luo, Xiang Zhang, and Suhang Wang. Multi-source unsupervised domain adaptation on graphs with transferability modeling. In *KDD*, pages 4479–4489, 2024.

[139] Wang Zhili, Di Shimin, Chen Lei, and Zhou Xiaofang. Search to fine-tune pre-trained graph neural networks for graph-level tasks. In *ICDE*, pages 2805–2819. IEEE, 2024.

[140] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, pages 16816–16825, 2022.

[141] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *IJCV*, 130(9):2337–2348, 2022.

[142] Qi Zhu, Carl Yang, Yidan Xu, Haonan Wang, Chao Zhang, and Jiawei Han. Transfer learning of graph neural networks with ego-graph information maximization. *NeurIPS*, 34:1766–1779, 2021.

[143] Qi Zhu, Da Zheng, Xiang Song, Shichang Zhang, Bowen Jin, Yizhou Sun, and George Karypis. Parameter-efficient tuning large language models for graph representation learning. *arXiv preprint arXiv:2404.18271*, 2024.

[144] Xi Zhu, Haochen Xue, Ziwei Zhao, Wujiang Xu, Jingyuan Huang, Minghao Guo, Qifan Wang, Kaixiong Zhou, and Yongfeng Zhang. LLM as GNN: Graph vocabulary learning for text-attributed graph foundation models. *arXiv preprint arXiv:2503.03313*, 2025.

[145] Yun Zhu, Jianhao Guo, and Siliang Tang. SGL-PT: A strong graph learner with graph prompt tuning. *arXiv preprint arXiv:2302.12449*, 2023.

[146] Yun Zhu, Haizhou Shi, Xiaotang Wang, Yongchao Liu, Yaoke Wang, Boci Peng, Chuntao Hong, and Siliang Tang. GraphCLIP: Enhancing transferability in graph foundation models for text-attributed graphs. In *WWW*, pages 2183–2197, 2025.

[147] Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. Efficient tuning and inference for large language models on textual graphs. In *IJCAI*, pages 5734–5742, 2024.

[148] Yun Zhu, Yaoke Wang, Haizhou Shi, Zhenshuo Zhang, Dian Jiao, and Siliang Tang. GraphControl: Adding conditional control to universal graph pre-trained models for graph domain transfer learning. In *WWW*, pages 539–550, 2024.

[149] Chenyi Zi, Haihong Zhao, Xiangguo Sun, Yiqing Lin, Hong Cheng, and Jia Li. ProG: A graph prompt learning benchmark. In *NeruIPS*, 2024.

 https://doi.org/10.52202/085713-0560

Appendix

Table of Contents

A Notations

Notations	Descriptions
$G = (\mathcal{V}, \mathcal{E})$	Graph G with node set $\mathcal{V}$ and edge set $\mathcal{E}$.
$\{G^{\mathcal{S}}\} \in \mathcal{G}^{\mathcal{S}}$, $\{G^{\mathcal{T}}\} \in \mathcal{G}^{\mathcal{T}}$	Graphs sampled from source domain ($\mathcal{S}$) and target domains ($\mathcal{T}$).
$\{Y^{\mathcal{S}}\} \in \mathcal{Y}^{\mathcal{S}}$, $\{Y^{\mathcal{T}}\} \in \mathcal{Y}^{\mathcal{T}}$	Corresponding graph labels in source domain ($\mathcal{S}$) and target domains ($\mathcal{T}$).
$\{D^{\mathcal{S}}\} \in \mathcal{D}^{\mathcal{S}}$, $\{D^{\mathcal{T}}\} \in \mathcal{D}^{\mathcal{T}}$	Source domains and targeted domains.
$\mathbf{A} \in \{0,1\}^{N_i \times N_i}$	Adjacency matrix $\mathbf{A}$, where N_i is the number of nodes.
$\mathbf{X} \in \mathbb{R}^{N_i \times d_i}$	Node feature matrix $\mathbf{X}$, with N_i nodes and d_i input feature dimensions.
$\boldsymbol{\mathcal{A}}$, $\widetilde{\boldsymbol{\mathcal{A}}} \in \mathbb{R}^{n' \times n'}$	Adjacency matrix of disentangled and generated vocabulary with n' nodes.
$\boldsymbol{\mathcal{X}}$, $\widetilde{\boldsymbol{\mathcal{X}}} \in \mathbb{R}^{n' \times h}$	Node feature matrix of disentangled and generated vocabulary with h dim.
$\mathbf{X}^{\mathcal{S}}$, $\widehat{\mathbf{X}}^{\mathcal{S}}$, $\widehat{\mathbf{x}}_i^{\mathcal{S}} \in \mathbb{R}^{\text{-} \times d}$	Source domain graph features, aligned features, and single node features.
$\mathbf{H}^{\mathcal{S}}$, $\mathbf{h}^{\mathcal{S}} \in \mathbb{R}^{K \cdot h}$	Encoded source domain graph features and node features.
$\mathbf{X}^{\mathcal{T}}$, $\widehat{\mathbf{X}}^{\mathcal{T}}$, $\widehat{\mathbf{x}}_i^{\mathcal{T}} \in \mathbb{R}^{\text{-} \times d}$	Target domain graph features, aligned features, and single node features.
$\mathbf{H}^{\mathcal{T}}$, $\mathbf{h}^{\mathcal{T}} \in \mathbb{R}^{K \cdot h}$	Encoded target domain graph features and node features.
$\mathbf{g}_u = \{\mathbf{g}_{u,k}\}_{k=1}^{K}$	Ego-graph and its K disentangled vocabularies of the central node u.
$\mathbf{g}^{\text{emp}}$, $\mathbf{g}^{\text{gen}}$, $\widetilde{\mathbf{g}}$	Observed, generated, and weighted mixed generated vocabularies.
$f_{\boldsymbol{\Theta}}(\cdot) \colon \mathbb{R}^d \mapsto \mathbb{R}^{K \cdot h}$	Graph vocabulary encoder with learnable parameter $\boldsymbol{\Theta}$.
$g(\cdot) \colon \mathbb{R}^{K \cdot h} \times \mathbb{R}^{K \cdot h} \mapsto \mathbb{R}^2$	Graph discriminator composed of an inner product $\langle \cdot, \cdot \rangle$ and a binary MLP.
$h(\cdot) = g \circ f$	The graph learner compound of encoder $f(\cdot)$ and binary discriminator $g(\cdot)$
$\mathcal{F}(\cdot) \colon \mathbb{R}^{d_i} \mapsto \mathbb{R}^d$	Domain feature alignment function from d_i dimension to d.
$\sigma(\cdot)$	Non-linear activation function (PReLU).
$\phi(\cdot)$	Learnable routing network in the MoE-CoE network.
$\psi(\cdot)$	The slack function controls bound tightness.
$\Pi(\cdot)$	The set of permutation functions (disentangle channel mappings).
$\pi(\cdot)$	The measure-preserving mapping estimated via empirical node ordering.
$I(\cdot;\cdot)$	The Shannon Mutual Information term.
$\mathcal{N}(v)$	The neighbor nodes of v (1-hop by default).
$\mathcal{D}_{\text{KL}}$	The Kullback-Leibler divergence.
$\mathcal{D}_{\text{match}}$	The matching distance between disentangled channels.
$\mathcal{D}_{\text{TV}}$	The Total Variation (TV) distance between vocabularies.
$W_c^{\boldsymbol{\mathcal{A}}} \colon [0,1]^2 \mapsto [0,1]$	The nonparametric estimated structure token graphon.
$W_c^{\boldsymbol{\mathcal{X}}} \colon [0,1]^2 \mapsto \mathbb{R}^d$	The nonparametric estimated feature token graphon.
$\mathbf{S}_{\text{M}}$, $\mathbf{S}_{\text{C}}$	Assignment weight vector for the MoE and CoE network, respectively.
$\mathbf{W}$, $\mathbf{W}_{\text{M}}$, $\mathbf{W}_{\text{C}}$	Learnable weight matrix for non-linear projection.
$\mathbf{b}$	The non-linear projection bias.
$\boldsymbol{\mathcal{P}}_{\boldsymbol{\Omega}}$	Graph prompt with learnable parameter $\boldsymbol{\Omega}$.
$\alpha_{i \to k}$	The routing attention (weights) from node i to channel (factor) k.
$\oplus$	The augmentation operations by overlapping with the generated vocabulary.
T	The number of disentanglement routing iterations.
C_σ	The Lipschitz constant for the activation function σ.
$L_{\mathbf{W}}$	The Lipschitz constant for the weight matrix $\mathbf{W}$.
L_s	The Lipschitz constant for the cosine similarity $\langle \cdot, \cdot \rangle$.
Δ	The semantic discrepancies of $\|f(\mathbf{g}_u^{\mathcal{S}}) - f(\mathbf{g}_v^{\mathcal{S}})\|_2$.
δ	The Dirac measure, representing a point mass probability distribution.
ϵ	The upper bound of $\|\widetilde{\mathbf{x}}_u^{\mathcal{S}} - \widetilde{\mathbf{x}}_v^{\mathcal{S}}\|_2$.
ρ	The lower bound of L_2-norm in Eq. (3).
τ	The temperature parameter.
λ	The trade-off hyperparameter for $\mathcal{R}_{\text{MI}}$ in Eq. (8).
μ	The trade-off hyperparameter for $\mathcal{L}_{\text{MoE-CoE}}$ in Eq. (19).
$\mathcal{R}_{\text{MI}}$	The mutual information regularizer between each disentangled vocabulary.
$\mathcal{L}_{\text{pre}}$	The pre-training loss function.
$\mathcal{L}_{\text{MoE-CoE}}$	Loss function for the MoE-CoE network.
$\mathcal{L}_{\text{cls}}$	Loss function for the downstream classification.
$\mathcal{L}_{\text{ftn}}$	Loss function for the fine-tuning phase few-shot training.

https://doi.org/10.52202/085713-0560

B Algorithm and Complexity Analysis

The pre-training pipeline of GRAVER is illustrated in Algorithm 1, and the fine-tuning pipeline is shown in Algorithm 2. Next, we theoretically analyze their computational complexity, respectively.

Algorithm 1: Pre-training pipeline of GRAVER.

Input: n source graphs $\{G^{\mathcal{S}}\} \in \mathcal{G}^{\mathcal{S}}$ from multi-domain $\{D^{\mathcal{S}}\} \in \mathcal{D}^{\mathcal{S}}$, where each of the graph $G_i^{\mathcal{S}}$ is associate with its feature matrix $\mathbf{X}_i \in \mathbb{R}^{N_i \times d_i}$ and adjacency matrix $\mathbf{A} \in \mathbb{R}^{N_i \times N_i}$; Feature dimension aligner $\mathcal{F}(\cdot)$; Domain semantic aligners $\mathbf{W}_i$; Learnable weight matrix $\mathbf{W}_k$ for disentangle channel projection; Number of the latent factors (channels) K; Number of the routing iterations T; Number of the pre-training epochs E_1; Temperature parameter τ; Hyperparameter λ for trading-off the mutual information regularization term $\mathcal{R}_{\text{MI}}$.

Output: Pre-trained graph encoder and discriminator $g(f^{\star}_{\mathbf{\Theta}}(\cdot))$; Structure token graphon $W_c^{\boldsymbol{\mathcal{A}}}$ and feature token graphon $W_c^{\boldsymbol{\mathcal{X}}}$ for each source domain $G^{\mathcal{S}}$ and class c.

Initialize all parameters randomly;
for $e = 1, 2, \cdots, E_1$ **do**
  // domain feature alignment
  Aligned graph feature: $\widehat{\mathbf{X}}_i^{\mathcal{S}} \leftarrow$ Eq. (1) for each $G_i^{\mathcal{S}} \in \{G^{\mathcal{S}}\}$ with $\mathcal{F}$ and $\mathbf{W}_i$;
  // ego-graph disentanglement
  for $t = 1, 2, \cdots, T$ and $k = 1, 2, \cdots, K$ **do**
    Initialize projected embedding $\mathbf{h}_{u,k}^{\mathcal{S}(0)} \leftarrow$ Eq. (3) for each node u;
    Routing attention for neighbor node v to k-th channel (factor): $\alpha_{v \to k}^{(t)} \leftarrow$ Eq. (2);
    Disentangled ego-graphs (vocabularies): $\mathbf{g}_u^{\mathcal{S}(t)} \leftarrow$ Eq. (4) for each node u;
    Updated node disentangled embeddings: $\mathbf{h}_u^{\mathcal{S}(t)} \leftarrow$ Eq. (5) for each node u;
    Calculate the mutual information regularizer: $\mathcal{R}_{\text{MI}}^u \leftarrow$ Eq. (6) for each node u;
  end
  // self-supervised pre-training
  Calculate the overall pre-training loss: $\mathcal{L}_{\text{pre}} \leftarrow$ Eq. (8);
  Update $\mathbf{\Theta}$ by minimizing $\mathcal{L}_{\text{pre}}$ and back-propagation;
  if $e = E_1$ **then**
    // establish the vocabulary bank
    Modeling structure token graphon with observed vocabularies: $W_c^{\boldsymbol{\mathcal{A}}} \leftarrow$ Eq. (9);
    Modeling feature token graphon with observed vocabularies: $W_c^{\boldsymbol{\mathcal{X}}} \leftarrow$ Eq. (10);
  end
end

Complexity Analysis. We analyze the complexity of each part in Algorithm 1 as follows.

- **Dimension-wise feature alignment:** We implement the aligner function $\mathcal{F}: \mathbb{R}^{d_i} \mapsto \mathbb{R}^d$ by the truncated singular value decomposition (SVD) [80]. We omit this computational complexity as it can be pre-processed without increasing the runtime computational burdens.
- **Semantic-wise feature alignment:** We implemented each aligner $\mathbf{W}_i \in \mathbb{R}^{d \times d}$ by a randomly initialized learnable non-linear projection matrix, where the complexity takes $\mathcal{O}(nd^2)$.
- **Ego-graph disentanglement:** We first project the aligned feature of each node to K embedding space, which takes the complexity of $\mathcal{O}(n|\mathcal{V}_i|Kdh)$, where $|\mathcal{V}_i|$ is the average number of nodes in each of the pre-training graphs. Next, the neighbor routing iterations take the complexity of $\mathcal{O}(n|\mathcal{V}_i|T|\mathcal{N}(u)|Kh)$, where $|\mathcal{N}(u)|$ is the number of neighbor nodes of of u. Then, the node disentangled embedding updating takes the complexity of the same $\mathcal{O}(n|\mathcal{V}_i|T|\mathcal{N}(u)|Kh)$. For brevity, denote the average of $|\mathcal{V}_i|$ as n_1, and the average of $|\mathcal{N}(u)|$ as n_2. Thus, the total complexity of the ego-graph disentanglement takes $\mathcal{O}(nn_1Kh(d + Tn_2))$.
- **Pair-wise mutual information regularizer:** For a pair of factors (i, j), the inner product takes the complexity of $\mathcal{O}(|\mathcal{B}|h)$. For each node, we can enumerate $|\mathcal{B}|K(K-1)$ factor pairs. Then, the total complexity accounting on each node u takes the complexity of $\mathcal{O}(nn_1|\mathcal{B}|^2K(K-1)h)$.

- **Pre-training loss calculation:** Suppose there are n_3 direct node neighbors, n_4 indirect neighbors (nodes that are connected) averaged for each node in the source domain graphs, we can sample $n \cdot n_3 \cdot n_4$ quadruples, which is utilized to formalize the self-supervised pre-training contrastive loss. Its computational complexity takes $\mathcal{O}(nn_1(n_3+n_4)Kh)$.
- **Graphon experts estimation:** We implement $\pi_i(\cdot)$ by ordering nodes with degree, which takes the complexity of $\mathcal{O}(N_c n' \log n')$ for each graph $G^{\mathcal{S}}$ and class c. For each coordinate $(u, v) \in \{1, \cdots, n'\}$, the estimation takes the complexity of $\mathcal{O}(N_c)$ for n'^2 coordinate pairs. Thus, the total complexity takes $\mathcal{O}(n\mathcal{C}(N_c n' \log n' + N_c n'^2))$, where $\mathcal{C}$ is the average number of classes.

The overall computational complexity for the pre-training phase is:

$$\mathcal{O}(nd^2) + \mathcal{O}(nn_1Kh(d+Tn_2)) + \mathcal{O}(nn_1|\mathcal{B}|^2K(K-1)h) + \mathcal{O}(nn_1(n_3+n_4)Kh) + \mathcal{O}(n\mathcal{C}(N_c n' \log n' + N_c n'^2)). \quad \text{(B.1)}$$

As $n_1 \gg n$, $n_1 \gg n_2$, $n_2 \doteq n_3$, $n_1 \doteq n_4$, and K, T, $|\mathcal{B}|$, $\mathcal{C}$, N_c, d, h, n' are relatively small constants, the overall computational complexity is approximately reduced to ***quadratic*** with respect to the number of pre-training nodes, which demonstrates the proposed GRAVER is on par with other state-of-the-art graph multi-domain pre-training GFMs.

Algorithm 2: Fine-tuning pipeline of GRAVER.

Input: m nodes class-balanced sampled from graph $G^{\mathcal{T}} \in \mathcal{G}^{\mathcal{T}}$ (for node classification) or m graphs $\{G^{\mathcal{T}}\} \in \mathcal{G}^{\mathcal{T}}$ class-balanced sampled from target domain $\{D^{\mathcal{T}}\} \in \mathcal{D}^{\mathcal{T}}$ (for graph classification), where each of the graph $G_i^{\mathcal{T}}$ is associate with feature matrix $\mathbf{X}_i \in \mathbb{R}^{N_i \times d_i}$ and adjacency matrix $\mathbf{A} \in \mathbb{R}^{N_i \times N_i}$; Pre-trained graph encoder and discriminator $g(f_{\boldsymbol{\Theta}}^{\star}(\cdot))$; Structure token graphon $W_c^{\boldsymbol{\mathcal{A}}}$ and feature token graphon $W_c^{\boldsymbol{\mathcal{X}}}$ for each source domain $G^{\mathcal{S}}$ and class c; Learnable weight matrix $\mathbf{W}_{\text{M}}$ and $\mathbf{W}_{\text{C}}$; Weight assignment vectors $\mathbf{S}_{\text{M}}$ and $\mathbf{S}_{\text{C}}$; Learnable graph prompts $\boldsymbol{\mathcal{P}}_{\boldsymbol{\Omega}}$; Number of the maximum fine-tuning episodes E_2; Temperature parameter τ; Hyperparameter μ for trading-off the MoE-CoE loss term.

Output: Fine-tuned graph encoder and discriminator $g(f_{\boldsymbol{\Theta}}^{\star}(\cdot))$ with $\boldsymbol{\Omega}^{\star}$.

Initialize all parameters randomly;
for $e = 1, 2, \cdots, E_2$ **do**
  // domain feature alignment
  Aligned graph feature: $\widehat{\mathbf{X}}^{\mathcal{T}} \leftarrow$ Eq. (1) for the given $G^{\mathcal{T}}$ (node classification) or $\widehat{\mathbf{X}}_i^{\mathcal{T}} \leftarrow$ Eq. (1) for each $G_i^{\mathcal{T}} \in \{G^{\mathcal{T}}\}$ (graph classification);
  // MoE-CoE network initialization and optimization
  Initialize assigned routing weights: $\mathbf{S}_{\text{C}} \leftarrow$ Eq. (14), $\mathbf{S}_{\text{M}} \leftarrow$ Eq. (14);
  Calculate the MoE-CoE loss: $\mathcal{L}_{\text{MoE-CoE}} \leftarrow$ Eq. (16);
  // support samples augmentation
  Vocabulary generation: $\mathbf{g}_c^{\text{gen}} \leftarrow$ Eq. (11) for each source dataset and class;
  Vocabulary composition: $\widetilde{\mathbf{g}}_i^{\text{gen}} \leftarrow$ Eq. (15) for each supporting samples;
  Vocabulary overlapping: $\widetilde{G}_i^{\mathcal{T}} \leftarrow$ Eq. (15) for each supporting samples;
  Encode target domain node or graph with graph prompts: $\mathbf{H}_i^{\mathcal{T}} \leftarrow$ Eq. (17);
  Calculate downstream classification loss: $\mathcal{L}_{\text{cls}}(\boldsymbol{\Omega}; \mathbf{H}^{\mathcal{T}}) \leftarrow$ Eq. (18);
  // optimization
  Calculate overall fune-tuning loss: $\mathcal{L}_{\text{ftn}} \leftarrow$ Eq. (19);
  Update $\boldsymbol{\Omega}$ by minimizing $\mathcal{L}_{\text{ftn}}$ and back-propagation.
end

Complexity Analysis. We analyze the complexity of each part in Algorithm 2 as follows.

- **Dimension-wise feature alignment:** We apply the same shared aligner function $\mathcal{F}: \mathbb{R}^{d_i} \mapsto \mathbb{R}^d$ which is implemented by the truncated singular value decomposition (SVD) [80]. Similarly, we omit its complexity as it can be preprocessed without increasing the runtime computational burdens.
- **Semantic-wise feature alignment:** We implemented each aligner $\mathbf{W}_i \in \mathbb{R}^{d \times d}$ by a randomly initialized learnable non-linear projection matrix, where the complexity takes $\mathcal{O}(|\mathcal{V}_i|d^2)$ (node clas-

https://doi.org/10.52202/085713-0560

sification) and $\mathcal{O}(m|\mathcal{V}_i|d^2)$ (graph classification). Denote the average of $|\mathcal{V}_i|$ as n_1, the complexity for this step takes $\mathcal{O}(n_1 d^2)$ for node classification, and $\mathcal{O}(mn_1 d^2)$ for graph classification.

- **MoE-CoE weights initialization:** For $\mathbf{S}_{\mathrm{M}}$, the matrix multiplication and Softmax operation takes the complexity of $\mathcal{O}(nn_1 d)$ for each support samples, while $\mathbf{S}_{\mathrm{C}}$ takes the complexity of $\mathcal{O}(nn_1(d+h)\mathcal{C})$ for each support samples. Thus, the total complexity takes $\mathcal{O}(mnn_1 d(\mathcal{C}+1) + mnn_1 h\mathcal{C})$.
- **MoE-CoE self-supervised optimization:** Calculating the uncertainty of $\mathbf{S}_{\mathrm{M}}$ takes the complexity of $\mathcal{O}(n^2 n_1)$, and that of $\mathbf{S}_{\mathrm{C}}$ takes $\mathcal{O}(nn_1\mathcal{C}^2)$. Thus, the total complexity takes $\mathcal{O}(nn_1(n+\mathcal{C}^2))$.
- **Support samples augmentation:** For each source dataset and class, the vocabulary generation takes the complexity of $\mathcal{O}(mn\mathcal{C}(n'^2 + n'h))$. Then, the vocabulary composition takes the complexity of $\mathcal{O}(mnh(\mathcal{C}+1))$. Next, the vocabulary overlapping takes the complexity of $\mathcal{O}(mn' \log n')$. Thus, the total complexity of this step takes $\mathcal{O}(mn\mathcal{C}(n'^2 + n'h) + mnh(\mathcal{C}+1) + mn' \log n')$.
- **Graph prompt insertion:** We initialize graph prompt with learnable parameter $\mathbf{\Omega}$ over the aligned features, which takes the complexity of $\mathcal{O}(n_1 d)$ for node classification and $\mathcal{O}(mn_1 d)$ for graph classification.
- **Augmented support sample encoding:** Similar to the pre-training phase, we first project the aligned feature of each node to K embedding space, which takes the complexity of $\mathcal{O}((n_1 + mn')Kdh)$ for node classification, and $\mathcal{O}(m(n_1 + n')Kdh)$ for graph classification. Next, the neighbor routing iterations take the complexity of $\mathcal{O}((n_1 n_2 + mn')TKh)$ for node classification, and $\mathcal{O}((nn_1 n_2 + mn')TKh)$ for graph classification, where n_2 denotes the average number of node neighbors. Then, the node disentangled embedding updating takes the same complexity as the last step. The total complexity takes $\mathcal{O}((n_1 + mn')Kdh + (n_1 n_2 + mn')TKh)$ for node classification, and $\mathcal{O}(m(n_1 + n')Kdh + (nn_1 n_2 + mn')TKh)$ for graph classification.
- **Fine-tuning classification loss:** Suppose there are n_c classes in total, the contrastive loss takes the computational complexity of $\mathcal{O}(mn_c Kh)$ for both node and graph classification.

The overall computational complexity for the fine-tuning phase of node classification takes:

$$\begin{aligned}\mathcal{O}(n_1 d^2) + \mathcal{O}(mnn_1 d(\mathcal{C}+1) + mnn_1 h\mathcal{C}) + \mathcal{O}(nn_1(n+\mathcal{C}^2)) + \mathcal{O}(mn\mathcal{C}(n'^2 + n'h) \\ + mnh(\mathcal{C}+1) + mn' \log n') + \mathcal{O}(n_1 d) + \mathcal{O}((n_1 + mn')Kdh + (n_1 n_2 + mn')TKh) \\ + \mathcal{O}(mn_c Kh). \quad \text{(B.2)}\end{aligned}$$

The overall computational complexity for the fine-tuning phase of graph classification takes:

$$\begin{aligned}\mathcal{O}(mn_1 d^2) + \mathcal{O}(mnn_1 d(\mathcal{C}+1) + mnn_1 h\mathcal{C}) + \mathcal{O}(nn_1(n+\mathcal{C}^2)) + \mathcal{O}(mn\mathcal{C}(n'^2 + n'h) \\ + mnh(\mathcal{C}+1) + mn' \log n') + \mathcal{O}(mn_1 d) + \mathcal{O}(m(n_1 + n')Kdh \\ + (nn_1 n_2 + mn')TKh) + \mathcal{O}(mn_c Kh). \quad \text{(B.3)}\end{aligned}$$

As $n_1 \gg n$, m, n', and K, T, $\mathcal{C}$, d, h, n_c are relatively small constants, the overall complexity is reduced to ***quasi-linear*** complexity, scaling as $\mathcal{O}(n \log n)$ with the number of nodes in target graphs.

It is worth mentioning that we temporarily defer the theoretical analysis of space complexity, and instead assess the empirical computational cost through efficiency evaluation. As shown in our experiments (see Appendix D for setup details), the pre-training phase of GRAVER exhibits comparable time and memory consumption to strong baselines. Although the fine-tuning phase incurs moderately higher overhead, this can be effectively alleviated via optimized training strategies, enabling a favorable trade-off between performance and efficiency.

To summarize, the proposed GRAVER achieves an efficient training paradigm through a carefully structured pipeline. The pre-training phase, though involving multiple steps such as disentangled representation construction and graphon expert estimation, scales approximately quadratically with the number of nodes in the source graphs. In contrast, the fine-tuning phase benefits from a streamlined and modular design that leverages learned graph vocabularies and MoE-CoE routing, thereby reducing the overall complexity to quasi-linear time with respect to the target graph size. This efficiency, together with our empirical results, demonstrates that the proposed GRAVER maintains strong scalability and practicality for large-scale multi-domain graph learning tasks as a scalable GFM.

C Proofs

C.1 Proof: the Upper Bound of $\mathcal{R}^u_{\text{MI}}$

We first restate $\mathcal{R}^u_{\text{MI}}$ for reference.

Eq. (6):

$$\mathcal{R}^u_{\text{MI}} = \sum_{1\leqslant i<j\leqslant K} I\big(\mathbf{h}^{\mathcal{S}}_{u,i};\mathbf{h}^{\mathcal{S}}_{u,j}\big) \leqslant \sum_{1\leqslant i<j\leqslant K} \mathbb{E}_{u\in\mathcal{B}}\Big[-\log\Big(\text{Softmax}_v\left(\langle \mathbf{h}^{\mathcal{S}}_{u,i}, \mathbf{h}^{\mathcal{S}}_{v,j}\rangle/\tau\right)\Big|_{v=u}\Big)\Big].$$

Proof. We provide a detailed derivation showing how the mutual information between two disentangled representations can be upper bounded using the InfoNCE loss.

Step-1: From Mutual Information to InfoNCE. The mutual information between two random variables $\mathbf{X}$ and $\mathbf{Y}$ is defined as the Kullback-Leibler ($\mathcal{D}_{\text{KL}}$) divergence between their joint distribution $p(\mathbf{x},\mathbf{y})$ and the product of marginals $p(\mathbf{x})p(\mathbf{y})$:

$$I(\mathbf{X};\mathbf{Y}) = \mathcal{D}_{\text{KL}}(p(\mathbf{x},\mathbf{y})\|p(\mathbf{x})p(\mathbf{y})) = \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[\log\frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})p(\mathbf{y})}\right]. \tag{C.1}$$

In practice, Eq. (C.1) is intractable to compute for continuous high-dimensional variables because it requires access to the true densities $p(\mathbf{x},\mathbf{y})$ and $p(\mathbf{x})p(\mathbf{y})$. The InfoNCE estimator [67] provides a tractable lower bound for $I(\mathbf{X};\mathbf{Y})$, based on a noise-contrastive learning setup that distinguishes true positive pairs from negative samples.

Step-2: InfoNCE Objective. Assume we sample $|\mathcal{B}|$ examples $\{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^{|\mathcal{B}|}$ where each $(\mathbf{x}_i,\mathbf{y}_i)\sim p(\mathbf{x},\mathbf{y})$ is drawn from the true joint distribution. One positive pair is drawn from $p(\mathbf{x},\mathbf{y})$, and $|\mathcal{B}|-1$ negative pairs are sampled independently from the marginal $p(\mathbf{x})p(\mathbf{y})$. The InfoNCE objective is defined as:

$$\mathcal{L}_{\text{InfoNCE}} = \mathbb{E}_{(\mathbf{x},\mathbf{y}),\{\mathbf{y}_i^-\}}\left[-\log\frac{s(\mathbf{x},\mathbf{y})}{s(\mathbf{x},\mathbf{y})+\sum_{i=1}^{|\mathcal{B}|-1}s(\mathbf{x},\mathbf{y}_i^-)}\right], \tag{C.2}$$

where $s(\mathbf{x},\mathbf{y})$ is a positive scoring function, typically chosen as $\exp(\langle\mathbf{x},\mathbf{y}\rangle/\tau)$ with temperature τ.

Step-3: Mutual Information Upper Bound. Assume that the positive sample $(\mathbf{x},\mathbf{y})\sim p(\mathbf{x},\mathbf{y})$, and that the negatives $\{\mathbf{y}_i^-\}\sim p(\mathbf{y})$ are *i.i.d.*. From [67], we can rewrite the InfoNCE loss in expectation form over the true joint distribution and the sampling distribution:

$$\mathcal{L}'_{\text{InfoNCE}} = \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[-\log\frac{s(\mathbf{x},\mathbf{y})}{\sum_{\mathbf{y}'}s(\mathbf{x},\mathbf{y}')}\right], \tag{C.3}$$

where the sum in the denominator includes the positive $\mathbf{y}$ and $|\mathcal{B}|-1$ negative samples $\mathbf{y}'\sim p(\mathbf{y})$. The optimal discriminator in this noise contrastive setup yields the bound:

$$I(\mathbf{X};\mathbf{Y}) \leqslant \log|\mathcal{B}| - \mathcal{L}'_{\text{InfoNCE}}. \tag{C.4}$$

This bound becomes tight as $|\mathcal{B}|\to\infty$, then InfoNCE provides a tractable surrogate for minimizing mutual information by replacing $\mathbf{x}$ with $\mathbf{h}^{\mathcal{S}}_{u,i}$, and $\mathbf{y}$ with $\mathbf{h}^{\mathcal{S}}_{u,j}$, respectively.

Step-4: Batch-Based InfoNCE and Independence Regularization. In our setting, we consider the batch $\mathcal{B}$ of size $|\mathcal{B}|$, and two latent factor-specific representations $\mathbf{h}^{\mathcal{S}}_{u,i}$, $\mathbf{h}^{\mathcal{S}}_{u,j}$ over the same node u. The positive pair is $(\mathbf{h}^{\mathcal{S}}_{u,i},\mathbf{h}^{\mathcal{S}}_{u,j})$, while negative samples are the cross-channel representations $\{\mathbf{h}^{\mathcal{S}}_{v,j}\}_{v\neq u}$ from other nodes in the batch. Then, the InfoNCE objective becomes:

$$\mathcal{L}^{(i,j)}_{\text{InfoNCE}} = \mathbb{E}_{u\in\mathcal{B}}\left[-\log\frac{\exp(\langle\mathbf{h}^{\mathcal{S}}_{u,i},\mathbf{h}^{\mathcal{S}}_{u,j}\rangle/\tau)}{\sum_{v\in\mathcal{B}}\exp(\langle\mathbf{h}^{\mathcal{S}}_{u,i},\mathbf{h}^{\mathcal{S}}_{v,j}\rangle/\tau)}\right] \tag{C.5}$$

$$\triangleq \mathbb{E}_{u\in\mathcal{B}}\Big[-\log\Big(\text{Softmax}_v\left(\langle\mathbf{h}^{\mathcal{S}}_{u,i},\mathbf{h}^{\mathcal{S}}_{v,j}\rangle/\tau\right)\Big|_{v=u}\Big)\Big]. \tag{C.6}$$

Hence, by the inequality derived above:

$$I\big(\mathbf{h}^{\mathcal{S}}_{u,i};\mathbf{h}^{\mathcal{S}}_{u,j}\big) \leqslant \log|\mathcal{B}| - \mathcal{L}^{(i,j)}_{\text{InfoNCE}}. \tag{C.7}$$

https://doi.org/10.52202/085713-0560

By discarding the constant $\log|\mathcal{B}|$, we obtain a looser but practical upper bound:

$$I\big(\mathbf{h}^{\mathcal{S}}_{u,i};\mathbf{h}^{\mathcal{S}}_{u,j}\big) \leqslant \mathbb{E}_{u\in\mathcal{B}}\Big[-\log\Big(\mathrm{Softmax}_v\left(\langle\mathbf{h}^{\mathcal{S}}_{u,i},\mathbf{h}^{\mathcal{S}}_{v,j}\rangle/\tau\right)\Big|_{v=u}\Big)\Big], \tag{C.8}$$

$$\Rightarrow \sum_{1\leqslant i<j\leqslant K} I\big(\mathbf{h}^{\mathcal{S}}_{u,i};\mathbf{h}^{\mathcal{S}}_{u,j}\big) \leqslant \sum_{1\leqslant i<j\leqslant K} \mathbb{E}_{u\in\mathcal{B}}\Big[-\log\Big(\mathrm{Softmax}_v\left(\langle\mathbf{h}^{\mathcal{S}}_{u,i},\mathbf{h}^{\mathcal{S}}_{v,j}\rangle/\tau\right)\Big|_{v=u}\Big)\Big]. \tag{C.9}$$

This concludes the proof. □

C.2 Proof: Inequity $[a]$ in Proposition 1

We first restate the inequity $[a]$ in Proposition 1 for reference.

Proposition 1 (Inequity $[a]$):

$$\Delta = \left\|f\big(\mathbf{g}^{\mathcal{S}}_u\big) - f\big(\mathbf{g}^{\mathcal{S}}_v\big)\right\|_2 \leqslant \left[\min_{\Pi\in S_K}\sum_{k=1}^{K}\left\|\mathbf{h}^{\mathcal{S}}_{u,k} - \mathbf{h}^{\mathcal{S}}_{v,\Pi(k)}\right\|_2^2 + \psi(\mathcal{R}^{u,v}_{\mathrm{MI}})\right]^{\frac{1}{2}}.$$

Proof. We next provide a proof of the semantic discrepancies upper bound via subgraph matching and mutual information constraints.

Step-1: Minimal Matching Distance. Since the per-channel routing is independently conducted for each node, the semantic order of channels may not be aligned across nodes. Therefore, we consider the minimal matching distance between their vocabularies (subgraph embeddings):

$$\mathcal{D}_{\text{match}} = \min_{\Pi\in S_k}\sum_{k=1}^{K}\left\|\mathbf{h}^{\mathcal{S}}_{u,k} - \mathbf{h}^{\mathcal{S}}_{v,\Pi(k)}\right\|_2^2, \tag{C.10}$$

where $\Pi \in S_k$ denotes a permutation over channels.

Step-2: Information-theoretic Slack. To encourage disentanglement, we introduce an information-theoretic regularization $\mathcal{R}_{\text{MI}}$ during training to minimize inter-channel mutual information. Ideally, a small $\mathcal{R}_{\text{MI}}$ implies that each channel captures a distinct and independent semantic factor. However, due to optimization limitations, residual dependencies across channels may still exist. These dependencies introduce cross-channel interaction terms in the final embedding space, making the overall semantic discrepancies larger than the matching term alone. To formally capture this effect, we introduce a slack function $\psi(\mathcal{R}_{\text{MI}})$, upper bounding the contribution from such interactions.

To obtain a tractable analytical form, we adopt a linear upper bound of the form:

$$\psi(\mathcal{R}_{\text{MI}}) \leqslant \beta\cdot\mathcal{R}_{\text{MI}}, \tag{C.11}$$

where $\beta > 0$ is a model-dependent constant that reflects how sensitive the final semantic discrepancies are to cross-channel redundancy. This linear approximation is justified under mild distributional assumptions. Specifically, when the per-channel embeddings approximately follow Gaussian or smooth distributions, the single mutual information term between two channels can be bounded in terms of their correlation:

$$I\big(\mathbf{h}^{\mathcal{S}}_{u,i};\mathbf{h}^{\mathcal{S}}_{u,j}\big) \geqslant \frac{1}{2}\log\frac{1}{1-\rho^2}, \tag{C.12}$$

where ρ is the Pearson correlation coefficient. The equality can be satisfied only if $\big(\mathbf{h}^{\mathcal{S}}_{u,i};\mathbf{h}^{\mathcal{S}}_{u,j}\big) \sim \mathcal{N}$ is jointly Gaussian with variances σ_i^2, σ_j^2 and covariance $\rho\sigma_i\sigma_j$. This inequality is commonly used in the literature as a conservative estimate of mutual information [93, 79], since direct computation of $I(\mathbf{h}^{\mathcal{S}}_{u,i};\mathbf{h}^{\mathcal{S}}_{u,j})$ is intractable for arbitrary distributions. We omit its proof for simplicity.

Conversely, high mutual information implies strong coupling (large covariance) between channels, which directly contributes to the cross terms in the semantic discrepancies calculation. Thus, using a linear upper bound allows us to quantify this slack without explicitly modeling the full joint statistics.

Step-3: Upper Bound on Semantic Discrepancy. Combining the matching difference with the information-theoretic slack, we obtain the following upper bound:

$$\Delta = \left\| f\left(\mathbf{g}_u^{\mathcal{S}}\right) - f\left(\mathbf{g}_v^{\mathcal{S}}\right) \right\|_2 \leqslant \left[\mathcal{D}_{\text{match}} + \psi(\mathcal{R}_{\text{MI}}^{u,v})\right]^{\frac{1}{2}} \tag{C.13}$$

$$= \left[\min_{\Pi \in S_K} \sum_{k=1}^{K} \left\| \mathbf{h}_{u,k}^{\mathcal{S}} - \mathbf{h}_{v,\Pi(k)}^{\mathcal{S}} \right\|_2^2 + \psi(\mathcal{R}_{\text{MI}}^{u,v}) \right]^{\frac{1}{2}}. \tag{C.14}$$

This concludes the proof. □

C.3 Proof: Inequity $[b]$ in Proposition 1

We first restate the inequity $[b]$ in Proposition 1 for reference.

Proposition 1 (Inequity $[b]$):

$$\left[\min_{\Pi \in S_K} \sum_{k=1}^{K} \left\| \mathbf{h}_{u,k}^{\mathcal{S}} - \mathbf{h}_{v,\Pi(k)}^{\mathcal{S}} \right\|_2^2 + \psi(\mathcal{R}_{\text{MI}}^{u,v}) \right]^{\frac{1}{2}} \leqslant \epsilon\sqrt{K} \left(\frac{C_\sigma L_{\mathbf{W}} L_s}{4\rho\tau} \right)^T.$$

Proof. We achieve the proof first by further relaxing the conditions in Proof C.2.

Step-1: Further Relaxation of the inequity $[a]$. From Eq. (C.10), we can define the optimal channel matching can be derived with the optimal permutation function as:

$$\Pi^\star = \arg\min_{\Pi \in S_k} \sum_{k=1}^{K} \left\| \mathbf{h}_{u,k}^{\mathcal{S}} - \mathbf{h}_{v,\Pi(k)}^{\mathcal{S}} \right\|_2^2. \tag{C.15}$$

Then, we can reach a more relaxed upper bound:

$$\sum_{k=1}^{K} \left\| \mathbf{h}_{u,k}^{\mathcal{S}} - \mathbf{h}_{v,\Pi^\star(k)}^{\mathcal{S}} \right\|_2^2 \leqslant \sum_{k=1}^{K} \left\| \mathbf{h}_{u,k}^{\mathcal{S}} - \mathbf{h}_{v,k}^{\mathcal{S}} \right\|_2^2. \tag{C.16}$$

In this way, we can rewrite the inequality $[a]$ in Proposition C.2 as:

$$\left[\min_{\Pi \in S_K} \sum_{k=1}^{K} \left\| \mathbf{h}_{u,k}^{\mathcal{S}} - \mathbf{h}_{v,\Pi(k)}^{\mathcal{S}} \right\|_2^2 + \psi(\mathcal{R}_{\text{MI}}^{u,v}) \right]^{\frac{1}{2}} \leqslant \left[\sum_{k=1}^{K} \left\| \mathbf{h}_{u,k}^{\mathcal{S}} - \mathbf{h}_{v,k}^{\mathcal{S}} \right\|_2^2 + \psi(\mathcal{R}_{\text{MI}}^{u,v}) \right]^{\frac{1}{2}}. \tag{C.17}$$

Next, we continue proof with respect to the RHS of Eq. (C.17).

Step-2: Semantic Discrepancy of Single Channel. Assuming $\|\widehat{\mathbf{x}}_u^{\mathcal{S}} - \widehat{\mathbf{x}}_v^{\mathcal{S}}\|_2 \leqslant \epsilon$. The activation function σ, weight matrix $\mathbf{W}$, and cosine similarity $\langle \cdot, \cdot \rangle$ (inner product) are Lipschitz continuous with constants C_σ, $L_{\mathbf{W}}$, and L_s, respectively. ρ is the lower bound of L_2-norm in Eq. (3). Then, for each single channel, the semantic discrepancies in projected features satisfy:

$$\left\| \mathbf{h}_{u,k}^{\mathcal{S}} - \mathbf{h}_{v,k}^{\mathcal{S}} \right\|_2 \leqslant \frac{C_\sigma L_{\mathbf{W}}}{\rho} \left\| \widehat{\mathbf{x}}_u^{\mathcal{S}} - \widehat{\mathbf{x}}_v^{\mathcal{S}} \right\|_2 \leqslant \frac{C_\sigma L_{\mathbf{W}}}{\rho} \cdot \epsilon, \tag{C.18}$$

where the normalization lower bound ensures numerical stability by preventing the denominator from approaching 0. Next, we discuss the discrepancy in the channel routing probability.

The routing attention $\alpha_{v \to k}$ is determined by the cosine similarity $\langle \mathbf{h}_{u,k}^{\mathcal{S}}, \mathbf{h}_{v,k}^{\mathcal{S}} \rangle$:

$$\alpha_{v \to k} \propto \frac{\exp\left(\langle \mathbf{h}_{u,k}^{\mathcal{S}}, \mathbf{h}_{v,k}^{\mathcal{S}} \rangle / \tau\right)}{\sum_{k'} \exp\left(\langle \mathbf{h}_{u,k}^{\mathcal{S}}, \mathbf{h}_{v,k'}^{\mathcal{S}} \rangle / \tau\right)}. \quad \text{(Eq. (2))} \tag{C.19}$$

Differentiating with respect to $\langle \mathbf{h}_{u,k}^{\mathcal{S}}, \mathbf{h}_{v,k}^{\mathcal{S}} \rangle$, we obtain the sensitivity of routing attention:

$$\frac{\partial \alpha_{v \to k}}{\partial \langle \mathbf{h}_{u,k}^{\mathcal{S}}, \mathbf{h}_{v,k}^{\mathcal{S}} \rangle} = \frac{\alpha_{v \to k}(1 - \alpha_{v \to k})}{\tau}. \tag{C.20}$$

https://doi.org/10.52202/085713-0560

Then, the discrepancy in routing attention is bounded by:

$$|\alpha_{v\to k} - \alpha_{u\to k}| \leqslant \frac{L_s}{4\tau} \left\|\mathbf{h}_{u,k}^{\mathcal{S}} - \mathbf{h}_{v,k}^{\mathcal{S}}\right\|_2. \quad (\text{since } \alpha_{v\to k}(1-\alpha_{v\to k}) \leqslant \frac{1}{4}) \tag{C.21}$$

Thus, the semantic discrepancy after routing is:

$$\left\|\mathbf{h}_{u,k}^{\mathcal{S}(T)} - \mathbf{h}_{v,k}^{\mathcal{S}(T)}\right\|_2 \leqslant \epsilon \left(\frac{C_\sigma L_{\mathbf{W}} L_s}{4\rho\tau}\right)^T. \tag{C.22}$$

Step-3: Semantic Discrepancy for K Channels. Integrating Eq. (C.22) into RHS of Eq. (C.17):

$$\left[\sum_{k=1}^{K} \left\|\mathbf{h}_{u,k}^{\mathcal{S}} - \mathbf{h}_{v,k}^{\mathcal{S}}\right\|_2^2 + \psi(\mathcal{R}_{\text{MI}}^{u,v})\right]^{\frac{1}{2}} \leqslant \left[\sum_{k}^{K} \epsilon \left(\frac{C_\sigma L_{\mathbf{W}} L_s}{4\rho\tau}\right)^T + \psi(\mathcal{R}_{\text{MI}}^{u,v})\right]^{\frac{1}{2}} \tag{C.23}$$

$$= \epsilon\sqrt{K}\left(\frac{C_\sigma L_{\mathbf{W}} L_s}{4\rho\tau}\right)^T + \sqrt{\psi(\mathcal{R}_{\text{MI}}^{u,v})}. \tag{C.24}$$

Considering $\rho \to 0$ in Eq. (C.12), which implies the variables are linearly uncorrelated with no shared information. Then, the mutual information regularization term $\mathcal{R}_{\text{MI}}^{u,v} \to 0$. We reach:

$$\left[\sum_{k=1}^{K} \left\|\mathbf{h}_{u,k}^{\mathcal{S}} - \mathbf{h}_{v,k}^{\mathcal{S}}\right\|_2^2 + \psi(\mathcal{R}_{\text{MI}}^{u,v})\right]^{\frac{1}{2}} \leqslant \epsilon\sqrt{K}\left(\frac{C_\sigma L_{\mathbf{W}} L_s}{4\rho\tau}\right)^T. \tag{C.25}$$

By combining Eq. (C.17) and Eq. (C.25), we conclude the proof. □

C.4 Proof: Generation Distributional Convergence (Proposition 2)

We first introduce a lemma on the graph estimator consistency.

Lemma 1 (Consistency of Vocabulary Graphon Estimators). *Assume that the graph vocabulary samples are drawn i.i.d. from a latent generative model with the true underlying structure graphon function $W_c^{\mathcal{A}\star}\colon [0,1]^2 \mapsto [0,1]$ and the feature graphon function $W_c^{\mathcal{X}\star}\colon [0,1] \mapsto \mathbb{R}^d$, both of which are bounded and Lipschitz continuous. Further, assume that the alignment mappings π_i are consistent with the true latent positions up to vanishing approximation error. Then, as $N_c \to \infty$, the estimators satisfy uniform convergence:*

$$\left\|W_c^{\mathcal{A}} - W_c^{\mathcal{A}\star}\right\|_\infty \to 0, \quad \left\|W_c^{\mathcal{X}} - W_c^{\mathcal{X}\star}\right\|_\infty \to 0. \tag{C.26}$$

Proof. Let $u, v \in [0,1]$ be two arbitrary latent positions. We consider the estimation error:

$$\left|W_c^{\mathcal{A}}(u,v) - W_c^{\mathcal{A}\star}(u,v)\right| = \left|\frac{1}{N_c}\sum_{i=1}^{N_c}\left(\boldsymbol{\mathcal{A}}_i^{(c)}[\pi_i(u),\pi_i(v)] - W_c^{\mathcal{A}\star}(u,v)\right)\right|. \tag{C.27}$$

We decompose this difference into bias and variance terms. The random variable $\boldsymbol{\mathcal{A}}_i^{(c)}[\pi_i(u),\pi_i(v)]$ is a Bernoulli variable with success probability $W_c^{\mathcal{A}\star}(u,v) + \nu_i(u,v)$, where $\nu_i(u,v)$ captures the alignment approximation error and latent variability. Under the alignment consistency assumption, we have $|\nu_i(u,v)| \leqslant \nu_{n'}$ for all i, with $\nu_{n'} \to 0$ as the number of nodes in a vocabulary $n' \to \infty$.

Applying Hoeffding's inequality [25], we bound the deviation for fixed (u,v):

$$p\big(\left|W_c^{\mathcal{A}}(u,v) - \mathbb{E}\big[W_c^{\mathcal{A}}(u,v)\big]\right| > \omega\big) \leqslant 2\exp\big(-2N_c\omega^2\big). \tag{C.28}$$

Since the domain $[0,1]^2$ is compact, and $W_c^{\mathcal{A}\star}$ is Lipschitz, we cover the domain with a finite ω-net of size $\mathcal{O}(1/\omega^2)$, and apply the union bound. Then, for any $\eta > 0$, with probability at least $1-\eta$:

$$\sup_{u,v}\left|W_c^{\mathcal{A}}(u,v) - W_c^{\mathcal{A}\star}(u,v)\right| \leqslant \nu_{n'} + \mathcal{O}\left(\sqrt{\frac{\log(1/\eta)}{N_c}}\right), \tag{C.29}$$

where $\nu_{n'}$ accounts for alignment error. Therefore, as $N_c \to \infty$, we conclude that:

$$\left\|W_c^{\boldsymbol{\mathcal{A}}} - W_c^{\boldsymbol{\mathcal{A}}\star}\right\|_\infty \to 0. \tag{C.30}$$

The same argument applies to $W_c^{\boldsymbol{\mathcal{X}}}$. Since the feature graphon is vector-valued and assumed Lipschitz, and the feature vectors are bounded, we apply Azuma-Hoeffding inequality [74] coordinate-wise to each dimension of $\mathbb{R}^d$, and take a union bound. Thus:

$$\left\|W_c^{\boldsymbol{\mathcal{X}}} - W_c^{\boldsymbol{\mathcal{X}}\star}\right\|_\infty \to 0, \tag{C.31}$$

as $N_c \to \infty$. This completes the proof. □

Based on Lemma 1, we then prove Proposition 2. We first restate it for reference.

Proposition 2 (Generation Distributional Convergence) *Let $\mathbf{g}_c^{\text{emp}}$ be the empirical distribution over n'-node vocabulary subgraphs of class c, collected from the disentangled vocabulary bank:*

$$\mathbf{g}_c^{\text{emp}} := \frac{1}{N_c}\sum_{i=1}^{N_c} \delta_{(\boldsymbol{\mathcal{A}}_i^{(c)}, \boldsymbol{\mathcal{X}}_i^{(c)})}, \quad \textit{each } (\boldsymbol{\mathcal{A}}_i^{(c)}, \boldsymbol{\mathcal{X}}_i^{(c)}) \in \{0,1\}^{n'\times n'} \times \mathbb{R}^{n'\times d},$$

where $\delta_{(\cdot)}$ denotes Dirac measure, representing a point mass probability distribution. Assume the true underlying structure and feature functions $W_c^{\boldsymbol{\mathcal{A}}\star}$, $W_c^{\boldsymbol{\mathcal{X}}\star}$ are bounded, Lipschitz, and satisfy permutation equivariance. If the estimators $W_c^{\boldsymbol{\mathcal{A}}}$, $W_c^{\boldsymbol{\mathcal{X}}}$ converge uniformly in L_∞ norm, then the total variation (TV) distance between $\mathbf{g}_c^{\text{gen}} = (\widetilde{\boldsymbol{\mathcal{A}}}_c, \widetilde{\boldsymbol{\mathcal{X}}}_c)$ and $\mathbf{g}_c^{\text{emp}}$ vanishes as $N_c \to \infty$:

$$\|\mathbf{g}_c^{\text{gen}} - \mathbf{g}_c^{\text{emp}}\|_{\text{TV}} \to 0.$$

Proof. Let $\mathbf{u}_1, \cdots, \mathbf{u}_{n'}$ be *i.i.d.* latent positions drawn from $\mathcal{U}[0,1]$. Denote latent vector as $\mathbf{u} = (\mathbf{u}_1, \cdots, \mathbf{u}_{n'}) \in [0,1]^{n'}$. We define the vocabulary generation process via the mapping: $\Phi : \mathbf{u} \mapsto (\widetilde{\boldsymbol{\mathcal{A}}}, \widetilde{\boldsymbol{\mathcal{X}}})$. The convergence assumption on $W_c^{\boldsymbol{\mathcal{A}}}$ and $W_c^{\boldsymbol{\mathcal{X}}}$ are formally established in Lemma 1.

Step-1: Total Variation Distance Decomposition. To study the convergence of $\mathbf{g}_c^{\text{gen}}$ to the empirical distribution $\mathbf{g}_c^{\text{emp}}$, we introduce the limiting (true) distribution $\mathbf{g}_c^\star$ induced by the true graphons $W_c^{\boldsymbol{\mathcal{A}}\star}$, $W_c^{\boldsymbol{\mathcal{X}}\star}$. We then decompose the total variation distance based on the triangle inequality as:

$$\|\mathbf{g}_c^{\text{gen}} - \mathbf{g}_c^{\text{emp}}\|_{\text{TV}} \leqslant \|\mathbf{g}_c^{\text{gen}} - \mathbf{g}_c^\star\|_{\text{TV}} + \|\mathbf{g}_c^\star - \mathbf{g}_c^{\text{emp}}\|_{\text{TV}}. \tag{C.32}$$

We prove that both terms vanish as $N_c \to \infty$, using functional convergence of the graphon estimators.

Step-2: Convergence of $\mathbf{g}_c^{\text{gen}} \to \mathbf{g}_c^\star$. We consider the pushforward measure of latent positions $\mathbf{u}$ through the mapping $\Phi^{(W)}$ that uses estimated graphons $W_c^{\boldsymbol{\mathcal{A}}}$, $W_c^{\boldsymbol{\mathcal{X}}}$, versus the mapping $\Phi^{(\star)}$ that uses the true graphons $W_c^{\boldsymbol{\mathcal{A}}\star}$, $W_c^{\boldsymbol{\mathcal{X}}\star}$. Since $\widetilde{\boldsymbol{\mathcal{A}}}[i,j]$ are conditionally independent Bernoulli trials, and the Bernoulli distribution is Lipschitz with respect to its parameter, we have for every fixed $\mathbf{u}$:

$$\mathcal{D}_{\text{TV}}\big(\varsigma_{n'}^{\boldsymbol{\mathcal{A}}}(\cdot \mid \mathbf{u}), \varsigma_{n'}^{\boldsymbol{\mathcal{A}}\star}(\cdot \mid \mathbf{u})\big) \leqslant \sum_{i<j} \left|W_c^{\boldsymbol{\mathcal{A}}}(\mathbf{u}_i, \mathbf{u}_j) - W_c^{\boldsymbol{\mathcal{A}}\star}(\mathbf{u}_i, \mathbf{u}_j)\right|, \tag{C.33}$$

where $\varsigma_{n'}^{\boldsymbol{\mathcal{A}}}$ be the induced measure over adjacency matrices. By uniform convergence of $W_c^{\boldsymbol{\mathcal{A}}} \to W_c^{\boldsymbol{\mathcal{A}}\star}$, this sum is bounded by $\mathcal{O}(n'^2 \cdot \Delta_{n'}^{\boldsymbol{\mathcal{A}}})$ where $\Delta_{n'}^{\boldsymbol{\mathcal{A}}} = \|W_c^{\boldsymbol{\mathcal{A}}} - W_c^{\boldsymbol{\mathcal{A}}\star}\|_\infty \to 0$. Similarly, for features:

$$\left\|\widetilde{\boldsymbol{\mathcal{X}}}_i - \boldsymbol{\mathcal{X}}_i^\star\right\|_2 = \left\|W_c^{\boldsymbol{\mathcal{X}}}(\mathbf{u}_i) - W_c^{\boldsymbol{\mathcal{X}}^\star}(\mathbf{u}_i)\right\|_2 \leqslant \left\|W_c^{\boldsymbol{\mathcal{X}}} - W_c^{\boldsymbol{\mathcal{X}}\star}\right\|_\infty. \tag{C.34}$$

Hence, via coupling over $\mathbf{u}$, we construct a measure-preserving map such that:

$$\|\mathbf{g}_c^{\text{gen}} - \mathbf{g}_c^\star\|_{\text{TV}} \leqslant C\big(n'^2\Delta_{n'}^{\boldsymbol{\mathcal{A}}} + n'\Delta_{n'}^{\boldsymbol{\mathcal{X}}}\big) \to 0, \tag{C.35}$$

where $\Delta_{n'}^{\boldsymbol{\mathcal{A}}} = \|W_c^{\boldsymbol{\mathcal{A}}} - W_c^{\boldsymbol{\mathcal{A}}\star}\|_\infty$, $\Delta_{n'}^{\boldsymbol{\mathcal{X}}} = \|W_c^{\boldsymbol{\mathcal{X}}} - W_c^{\boldsymbol{\mathcal{X}}\star}\|_\infty$, and $C > 0$ is a constant.

Step-3: Convergence of $\mathbf{g}_c^{\text{emp}} \to \mathbf{g}_c^\star$. By assumption, the vocabulary bank $\{(\boldsymbol{\mathcal{A}}_i^{(c)}, \boldsymbol{\mathcal{X}}_i^{(c)})\}$ are *i.i.d.* samples from $\mathbf{g}_c^\star$. Since the sample space $\mathcal{Z}_{n'} \in \{0,1\}^{n'\times n'} \times \mathbb{R}^{n'\times d}$ is a Polish space (product of compact discrete and Euclidean space), the Glivenko-Cantelli theorem [90] applies. Therefore:

$$\|\mathbf{g}_c^{\text{emp}} - \mathbf{g}_c^\star\|_{\text{TV}} \to 0. \tag{C.36}$$

This completes the proof. □

https://doi.org/10.52202/085713-0560

D Experiment Details

In this section, we present experimental details. The dataset statistics are summarized in Table D.1.

Table D.1: Statistics of the multi-domain graph dataset.

Dataset	Domain	# Nodes	# Edges	# Feature Dimensions	# Classes	Avg. # Deg.
Cora [65]	Citation	2,708	5,429	1,433	7	4.00
CiteSeer [16]	Citation	3,186	4,277	3,703	6	2.57
PubMed [73]	Citation	19,717	44,338	500	3	4.50
ogbn-arXiv [28]	Citation	169,343	1,166,243	128	40	13.77
ogbn-Tech [28]	Co-purchase	47,428	2,077,241	100	3	87.60
ogbn-Home [28]	Co-purchase	9,790	131,841	100	5	26.93
Wiki-CS [66]	Web Link	11,701	216,123	300	10	36.94

D.1 Datasets Details

To emphasize multi-domain pre-training, we select **seven** benchmark graph datasets across **three** distinct domains, in contrast to conventional settings where each dataset is treated as an independent domain. These datasets vary in structure, scale, and feature distribution, providing a diverse and challenging testbed for evaluating pre-trained graph models.

Citation Domain. This domain comprises **four** widely used citation network benchmarks. Each dataset consists of a single graph, where nodes represent academic papers, edges denote citation relationships, and node labels correspond to the category or research topic of each paper.

- Cora [65]: [CC BY 4.0] This dataset contains 2,708 machine learning-related papers from 7 research fields. Each paper is represented by a 1,433-dimensional bag-of-words feature vector, where each dimension indicates a specific term. The graph includes 5,429 citations, resulting in a moderately sparse structure. Due to its small size and clear class boundaries, Cora is widely used for semi-supervised node classification benchmarks.
- CiteSeer [16]: [CC BY-NC-SA 3.0] This dataset comprises 3,186 publications categorized into 6 fields. CiteSeer represents each node with a 3,703-dimensional sparse feature vector capturing word occurrences. The graph contains 4,277 citations and exhibits a sparser structure with fewer average node degrees compared to Cora. The dataset is considered more challenging due to its noisy labels and lower graph connectivity.
- PubMed [73]: [License Unspecified] This biomedical citation network consists of 19,717 research articles on diabetes, divided into 3 classes. Each node is described by a dense 500-dimensional feature vector based on TF-IDF statistics of medical terms. With 44,338 citations, PubMed features a significantly denser graph structure. Its larger scale and domain specificity make it a common benchmark for evaluating scalable graph learning methods.
- ogbn-arXiv [28]: [ODC-BY] A large-scale citation network comprising 169,343 computer science papers from arXiv. Each node is associated with a 128-dimensional feature vector, generated by averaging word2vec embeddings of the paper's title and abstract. The graph contains 1,166,243 citations, and node labels span 40 subject areas. ogbn-arXiv is widely used for benchmarking scalable graph learning methods on real-world data.

Co-purchase Domain. This domain includes **two** networks derived from the ogbn-Product [28], capturing relationships between items frequently co-purchased on Amazon. These datasets are widely used for evaluating recommendation and product classification tasks. Nodes represent products, edges denote co-purchase relationships, and node labels correspond to product categories.

- ogbn-Tech [28]: [Amazon License] A co-purchase sub-network of ogbn-Product [28] related to technology with 47,428 nodes, 2,077,241 edges, and 3 categories. Each node represents a product, edges indicate co-purchase relationships, and node labels correspond to product categories. ogbn-Tech is commonly used for testing recommendation algorithms in the tech product domain.

- ogbn-Home [28]: [Amazon License] A co-purchase sub-network of ogbn-Product [28] related to home usage consisting of 9,790 nodes, 131,841 edges, and 6 categories. Each node represents a product, edges denote co-purchase relationships, and node labels correspond to product categories. ogbn-Home is useful for evaluating recommendation systems in the home goods domain.

Web Link Domain. This domain includes Wiki-CS [66][MIT License], a comprehensive collection of Wikipedia entries with 11,701 nodes and 216,123 web links, which is categorized into 10 distinct areas of computer science. The graph contains nodes representing articles, with edges denoting citation relationships between them. Each node is associated with a feature vector generated from the article's textual content. Wiki-CS is widely used for evaluating graph learning methods, particularly for tasks that involve knowledge construction and domain-specific knowledge extraction.

D.2 Baseline Details

We benchmark GRAVER against **15** state-of-the-art methods spanning **four** major categories: vanilla graph neural networks, self-supervised graph pre-training, prompt-based graph fine-tuning, and multi-domain pre-training frameworks. This comprehensive comparison highlights the effectiveness and robustness of our approach across diverse paradigms.

Vanilla Graph Neural Networks. This category comprises standard graph neural networks trained from scratch on downstream tasks without any pre-training. These models serve as fundamental baselines for evaluating representation learning performance.

- GCN [40] (backbone): A spectral-based method that captures local graph structures via layer-wise neighborhood aggregation. We adopt GCN as a backbone for both node and graph classification.
- GAT [91]: A graph neural network that leverages attention mechanisms to assign adaptive weights to neighbors, enabling selective aggregation of relevant information. We apply GAT to both node and graph classification tasks.

Self-Supervised Graph Pre-training. This category includes baselines that learn transferable representations by optimizing auxiliary tasks without annotations. By capturing intrinsic structural and semantic patterns during pre-training, these models significantly enhance downstream performance.

- GCC [69]: It learns transferable structural representations through subgraph instance discrimination across multiple graphs. By leveraging contrastive learning, GCC captures domain-invariant topological patterns, enhancing generalization to unseen graphs and improving performance in out-of-domain scenarios. It is applied to node classifications.
- DGI [92]: It maximizes mutual information between local patch and global summary, encouraging node representations to retain graph-wide context. Unlike relying on random walks, it is well-suited for both transductive and inductive settings. It is applied to node classifications.
- InfoGraph [81]: It focuses on learning graph-level representations by maximizing mutual information between entire graphs and hierarchical substructures. This design enables effective encoding of both structural and semantic features. It is applied to graph classifications.
- GraphCL [116]: It introduces contrastive learning with a suite of graph augmentations, such as node dropping, edge perturbation, and subgraph sampling, *etc.* These transformations promote the learning of robust and transferable representations under various supervision regimes. It is applied to both node and graph classifications.
- DSSL [102]: It addresses the limitations of homophily assumptions by introducing a latent variable model that decouples semantic content from graph structure. This enables the capture of diverse neighborhood patterns, leading to improved adaptability across homophilic and heterophilic graphs. It is applied to both node and graph classifications.
- GraphACL [103]: It proposes an asymmetric contrastive learning method tailored for heterophilic graphs. By removing the reliance on handcrafted augmentations and homophily priors, it effectively models both local interactions and monophily-based similarities. It is applied to both node and graph classifications.

Prompt-based Graph Fine-tuning. This category includes baselines that inject learnable prompts into pre-trained graph models to steer task-specific adaptation. By introducing lightweight, tunable instructions, prompt-based fine-tuning enhances the model's flexibility and improves knowledge transfer to diverse downstream tasks.

https://doi.org/10.52202/085713-0560

- GPPT [82]: It bridges pre-training and downstream adaptation by introducing token-pair prompts that recast node classification into edge prediction. It employs masked edge prediction as a pre-training task and maintains this objective format during fine-tuning, thereby reducing the task discrepancy and enabling effective prompt-guided transfer. It is applied to node classifications.
- GraphPrompt [58]: It formulates both pre-training and downstream tasks within a unified prompting framework, driven by learnable instructions. This design narrows the objective gap while enhancing label efficiency and transferability. The method provides a general-purpose interface for applying prompts to various graph tasks. It is applied to both node and graph classifications.
- GPF [14]: It presents a universal prompt-based tuning approach compatible with arbitrary pre-trained GNNs. By injecting prompts directly into input feature space, it enables flexible downstream adaptation without architectural constraints. GPF consistently outperforms full-model fine-tuning and tailored prompt strategies. It is applied to both node and graph classifications.
- ProNoG [121]: It targets the challenge of non-homophilic graphs by introducing a conditional network that dynamically adjusts prompt representations based on node-level structural cues. This allows the model to capture both homophilic and heterophilic patterns, achieving strong generalization across real-world graph scenarios. It is applied to both node and graph classifications.

Multi-Domain Graph Pre-training. This category includes the most directly related methods to ours, aiming to learn graph representations that generalize across heterogeneous domains. By exploiting shared structural and semantic patterns, these methods enhance the transferability and robustness of pre-trained models in multi-domain settings.

- GCOPE [136]: It proposes a unified graph pre-training framework that aggregates multiple datasets to distill common structural patterns across domains. By explicitly mitigating negative transfer, which is a common challenge in cross-domain scenarios, it facilitates more effective few-shot adaptation. This method advances the development of general-purpose graph foundation models tailored for multi-domain settings. It is applied to both node and graph classifications.
- MDGPT [123]: It introduces a text-independent multi-domain pre-training framework that aligns heterogeneous graph domains through learnable domain tokens. It further incorporates a dual-prompt strategy, which comprises unifying and mixing prompts to adaptively reconcile domain gaps during fine-tuning. This design enables effective transfer by minimizing semantic conflicts and enhancing the utility of source knowledge. It is applied to both node and graph classifications.
- SAMGPT [119]: It tackles cross-domain adaptation in text-free graphs by aligning structural patterns across domains, in contrast to MDGPT, which focuses on feature alignment using domain tokens. It introduces structure tokens to unify graph aggregation during pre-training, and employs dual prompts to transfer both generalizable and domain-specific structural knowledge. It is applied to both node and graph classifications.

D.3 Experimental Setting Details

D.3.1 Detailed Settings for Section 5.2 (RQ1)

This section examines the effectiveness and robustness of GRAVER in transferring knowledge from multiple source domains to unseen target datasets under the few-shot learning settings, *i.e.*, (C-way) m-shot scenarios (where $1 \leqslant m \leqslant 10$). GRAVER is first pre-trained on source-domain graphs and then fine-tuned with only a limited number of labeled samples from the target domain. To mitigate the instability of fine-tuning and build a trustworthy GFM, we propose a generative augmentation strategy that enriches support samples based on graph vocabularies. This approach ensures both high quality and stability of GFM fine-tuning under any random sample selection.

m-shot Fine-tuning Settings. In each experiment, we randomly sample m labeled instances (nodes or graphs) per class to form the support set used for fine-tuning, while the remaining labeled samples serve as the query set for evaluation. To evaluate robustness and mitigate sampling bias, we perform 20 independent runs with different random seeds, reporting their mean accuracy and standard deviation. We evaluate GRAVER on both node and graph classification tasks. For graph classification, we construct graph-level datasets by extracting 2-hop ego-graphs centered on each node. It is worth noting that GCC, DGI, and GPPT are designed exclusively for node-level classification and are thus only included in node-level comparisons, whereas InfoGraph, an extension of DGI, is evaluated solely on graph-level classification.

Table D.2: Statistics of the multi-domain graph dataset.

Model	Pre-training				Fine-tuning and Downstream Task					
	multi-domain	transferable pattern modeling	LLM-enhanced feature	self-supervised pre-training	cross-domain	multi-task	zero-shot learning	few-shot learning	in-context learning	data augmentation
GCN (bb.) [40]	–	–	–	–	–	–	–	✓	–	–
GAT [91]	–	–	–	–	–	–	–	✓	–	–
GCC [69]	–	–	–	✓	✓	✓	–	–	–	–
DGI [92]	–	–	–	✓	✓	–	–	–	–	–
InfoGraph [81]	–	–	–	✓	–	–	–	–	–	–
GraphCL [116]	–	–	–	✓	–	✓	–	–	–	–
DSSL [102]	–	–	–	✓	–	✓	–	–	–	–
GraphACL [103]	–	–	–	✓	–	✓	–	–	–	–
GPPT [82]	–	–	–	✓	–	✓	–	✓	✓	–
GraphPrompt [58]	–	–	–	✓	–	✓	–	✓	✓	–
GPF [14]	–	–	–	✓	–	✓	–	✓	✓	–
ProNoG [121]	–	–	–	✓	✓	✓	–	✓	✓	–
GCOPE [136]	✓	–	–	✓	✓	–	–	✓	✓	–
MDGPT [123]	✓	✓	–	✓	✓	✓	–	✓	✓	–
SAMGPT [119]	✓	✓	–	✓	✓	✓	–	✓	✓	–
GRAVER (ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Robustness against Random Noise Settings. To further evaluate the robustness of GRAVER under perturbations during fine-tuning, we introduce random noise into the support sets from two complementary aspects: *feature noise* and *structure noise*. For *feature noise*, we add random Gaussian perturbations to each dimension of node features for all support samples, following the formulation $\lambda_{\mathrm{f}} \cdot r \cdot \epsilon_{\mathrm{f}}$, where r denotes the reference amplitude of the original features, $\epsilon_{\mathrm{f}} \sim \mathcal{N}(0, 1)$ is standard Gaussian noise, and $\lambda_{\mathrm{f}} \in \{0.2, 0.4, 0.6, 0.8\}$ controls the noise intensity. For *structure noise*, we randomly modify the support graph structure by simultaneously deleting and adding λ_{s} (%) of the edges, with the same range of λ_{f} values. These controlled perturbations allow us to systematically assess the stability and noise tolerance of GRAVER in few-shot adaptation scenarios.

Cross-Dataset and Cross-Domain Settings. We consider two types of domain adaptation scenarios. For *cross-dataset* transfer, the target dataset is unseen during pre-training but comes from the same domain as the source datasets. For *cross-domain* transfer, the target dataset originates from a different domain altogether. It is worth noting that, while some prior works on cross-domain graph pre-training and fine-tuning [136, 123, 119] treat each dataset as a separate domain, we argue that such treatment is overly coarse. Instead, we define domains based on broader semantic or application categories. As a result, our *cross-dataset* setting corresponds to what previous literature often calls *cross-domain*, whereas our *cross-domain* setting reflects a higher-level shift across distinct fields.

D.3.2 Detailed Settings for Section 5.3 (RQ2)

We investigate contributions of GRAVER's three key components

GRAVER (*w/o SIP*). This variant removes the semantic independence promotion (*SIP*) module, which regularizes mutual information minimization between disentangled vocabulary channels. Specifically, we eliminate the mutual information regularizer $\mathcal{R}_{\mathrm{MI}}$ in Eq. 8 (set $\lambda = 0$), thus removing the constraint that encourages each factor-specific ego-subgraph to capture distinct semantics. This variant evaluates the necessity of encouraging vocabulary disentanglement in the pre-training stage.

GRAVER (*w/o VA*). This variant disables the vocabulary augmentation (*VA*) mechanism during fine-tuning. Specifically, we remove the generative vocabulary $\widetilde{\mathbf{g}}_i^{\mathrm{gen}}$ in Eq. (15) and skip the support augmentation step $G_i^{\mathcal{T}} \oplus \widetilde{\mathbf{g}}_i^{\mathrm{gen}}$. Fine-tuning is conducted directly on raw support graphs $G_i^{\mathcal{T}}$ without in-place augmentation. This ablation tests the utility of injecting class-conditioned generative subgraphs to enrich the few-shot supervision.

GRAVER (*w/o MC*). This variant replaces the hierarchical MoE-CoE routing network (*MC*) with a uniform composition mechanism. Specifically, we set the MoE routing weights $\mathbf{S}_{\mathrm{M}}$ and CoE routing weights $\mathbf{S}_{\mathrm{C}}$ in Eq. (14) to uniform distributions, *i.e.*, $\mathbf{S}_{\mathrm{M}} = [1/n, \cdots, 1/n]$ and $\mathbf{S}_{\mathrm{C}} = [1/N_c, \cdots, N_c]$. This variant examines whether learned adaptive routing is necessary, or whether naive equal-weight fusion suffices for knowledge reuse.

https://doi.org/10.52202/085713-0560

All variants are evaluated on the ogbn-Home dataset (cross-domain) under one-shot and five-shot settings for both node and graph classification. We report the accuracy (%) averaged over 20 runs, along with the standard deviation to measure robustness. Other settings, such as pre-trained weights, support-query splits, and training schedules, are kept identical to ensure fair comparison.

D.3.3 Detailed Settings for Section 5.4 (RQ3)

Convergence Efficiency. We measure the number of training episodes required for the model to converge under the one-shot node classification setting. The convergence is defined as no accuracy increase for 50 consecutive episodes. All models are initialized with the same pre-trained weights and run on the same query set split for fairness.

Memory Usage. We monitor the peak GPU memory consumption (GB) during fine-tuning using NVIDIA's nvidia-smi tool. Measurements are taken over 20 repeated runs and averaged. Analysis includes memory used by prompt embeddings, vocabulary augmentation, routing networks, and model forward/backward passes.

Target Dataset. We perform one-shot node classification on the ogbn-Tech dataset for all efficiency evaluations due to its large graph size and complex feature space. The setting reflects real-world high-load scenarios, making it suitable for assessing practical efficiency.

D.3.4 Detailed Settings for Section 5.5 (RQ4)

(+) Raw Text. For each node in graphs, we retrieve associated raw textual metadata (*e.g.*, paper titles, product names, or page summaries, depending on the specific domain) following [51, 47, 54, 9]. We tokenize the raw text using a pre-trained BERT-Base-Uncased tokenizer, and transform it into a dense vector by taking the mean-pooling over all output token embeddings. This vector is then concatenated with the original node features to form an enriched representation.

(++) LLM Enhanced. Building on the raw text, we further incorporate class-aware textual information. Specifically, for each node, we append a short class-level description (*e.g.*, a node in the "Graph Neural Networks" class may be appended with "This paper belongs to the topic of graph neural networks.") to its raw text. These descriptions are generated manually or retrieved from pre-defined templates, and are consistent within each class. The concatenated text is then encoded with the same BERT-Base-Uncased tokenizer as above. The resulting embedding is concatenated with the original features, similarly to the raw text setting.

Evaluation Protocol and Datasets. In both (+) and (++) settings, the obtained textual embeddings are directly concatenated to the original node feature vectors before alignment. This results in an extended feature space that blends structural and semantic cues. We compare three settings: original structural features, raw text-enhanced features (+), and LLM-enhanced features (++). For all settings, the model architecture and routing modules remain unchanged. Especially, in the ***zero-shot*** setting, no fine-tuning is performed, and predictions are made directly using the pre-trained encoder on the target domain. In the one-shot and five-shot settings, textual features are used alongside support samples for fine-tuning. We report results on CiteSeer (cross-dataset) and Wiki-CS (cross-domain), which are representative benchmarks with rich textual content. The zero-shot setting is evaluated by removing all supervision in the target domain and relying solely on aligned textual semantics.

D.3.5 Detailed Settings for Section 5.6 (RQ5)

Hyperparameter Ranges. For both λ and μ, we sweep over the values $\{0, 0.2, 0.4, 0.6, 0.8\}$, forming a 5×5 grid in the hyperparameter space. This results in 25 configurations, each evaluated independently. To provide a smooth and interpretable view of sensitivity trends, we apply bilinear interpolation over the 5×5 accuracy matrix to obtain a denser and continuous accuracy surface. Figure 8 clearly illustrates how model performance varies with respect to hyperparameter changes.

Evaluation Protocol. For each configuration, we perform five-shot classification on the PubMed dataset for both node and graph tasks. We report the accuracy averaged over 5 random support-query splits per configuration to ensure statistical reliability. The results suggest that GRAVER maintains stable performance over a broad range of λ and μ values, with only mild fluctuations in accuracy. This indicates that the model is relatively robust to hyperparameter tuning.

E Implementation Details

In this section, we provide implementation details of GRAVER and baselines.

E.1 Implementation Details of GRAVER

Pre-training. We pre-train GRAVER for up to 10,000 epochs, with early stopping applied if training loss does not decrease for 50 consecutive epochs to ensure efficiency without compromising performance. The aligned feature dimension is set to 64, and the hidden dimension of the encoder is 256. The number of disentangled channels (factors) K is set to 4, the number of routing iterations T is set to 3, and the number of convolution layers is set to 2. The trade-off hyperparameter λ is tuned within the range of 0 to 1. For optimization, we adopt the Adam optimizer [39], with the learning rate and weight decay selected from the range of 1e-5 to 1e-2 via grid search on the validation set. All model parameters are randomly initialized to eliminate potential bias from pretrained weights.

Fine-tuning. We fine-tune GRAVER for up to 1,000 epochs, using early stopping based on training accuracy with a patience of 50 epochs. All architectural and optimization hyperparameters remain consistent with pre-training. During vocabulary-based augmentation, we generate 15 nodes per class to overlap with the support set. The trade-off hyperparameter μ is tuned within $[0, 1]$ to balance the CoE-MoE objectives. Model parameters are randomly initialized to ensure fair adaptation.

E.2 Implementation Details of Baselines

We provide the baseline implementations with their respective licenses as follows.

- `GCN` [40]: [MIT License] https://github.com/pyg-team/pytorch_geometric.
- `GAT` [91]: [MIT License] https://github.com/pyg-team/pytorch_geometric.
- `GCC` [69]: [MIT License] https://github.com/THUDM/GCC.
- `DGI` [92]: [MIT License] https://github.com/PetarV-/DGI.
- `InfoGraph` [81]: [License Unspecified] https://github.com/sunfanyunn/InfoGraph.
- `GraphCL` [116]: [MIT License] https://github.com/Shen-Lab/GraphCL.
- `DSSL` [102]: [Apache-2.0 License] https://github.com/BUPT-GAMMA/OpenHGNN/blob/main/openhgnn/models/DSSL.py.
- `GraphACL` [103]: [License Unspecified] https://github.com/tengxiao1/GraphACL.
- `GPPT` [82]: [License Unspecified] https://github.com/MingChen-Sun/GPPT.
- `GraphPrompt` [58]: [License Unspecified] https://github.com/Starlien95/GraphPrompt.
- `GPF` [14]: [MIT License] https://github.com/zjunet/GPF.
- `ProNoG` [121]: [License Unspecified] https://github.com/Jaygagaga/ProNoG.
- `GCOPE` [136]: [License Unspecified] https://github.com/cshhzhao/GCOPE.
- `MDGPT` [123]: [License Unspecified] https://anonymous.4open.science/r/MDGPT.
- `SAMGPT` [119]: [License Unspecified] https://github.com/blue-soda/SAMGPT.

For all baselines, we follow the recommended hyperparameters from the original papers or official implementations. If unavailable or suboptimal, we apply careful tuning for best performance. To ensure fairness, we standardize key settings (*e.g.*, hidden dimensions, layers) to match our method, so that performance differences reflect model design rather than external factors.

E.3 Hardware and Software Configurations

We conduct the experiments with the following hardware and software configurations.

- Operating System: Ubuntu 20.04 LTS.
- CPU: Intel(R) Xeon(R) Platinum 8358 CPU@2.60GHz with 1TB DDR4 of Memory.
- GPU: NVIDIA Tesla A100 SMX4 with 80GB of Memory.
- Software: CUDA 10.1, Python 3.8.12, PyTorch[3] 1.9.1, PyTorch Geometric[4] 2.0.1.

[3] https://github.com/pytorch/pytorch
[4] https://github.com/pyg-team/pytorch_geometric

https://doi.org/10.52202/085713-0560

Table F.1: Accuracy (% ± std for 20 runs) of **five-shot node classification**. Best results are presented in **bold** and the runner-ups are underlined. CR = Cora, CS = CiteSeer, PM = PubMed, arXiv = ogbn-arXiv, Tech = ogbn-tech, Home = ogbn-home, Wiki = Wiki-CS. Color denotes domain.

Source	Cross-Dataset				Cross-Domain		
	CS PM Home Wiki	CR PM Home Wiki	CR CS Home Wiki	CR CS PM Home Wiki	CR CS PM Wiki	CR CS PM Home	Home Tech Wiki
Model / Target	CR	CS	PM	Tech	Home	Wiki	arXiv
GCN (bb.) [40]	50.19 ± 4.86	45.89 ± 5.42	50.64 ± 7.46	63.44 ± 5.18	58.24 ± 4.27	47.37 ± 6.06	35.79 ± 5.06
GAT [91]	49.03 ± 7.91	46.11 ± 5.12	52.19 ± 6.29	64.06 ± 6.34	57.40 ± 5.40	47.10 ± 5.42	36.74 ± 4.82
GCC [69]	49.81 ± 8.88	45.36 ± 6.40	53.03 ± 6.53	63.77 ± 5.62	57.94 ± 4.02	45.25 ± 6.92	36.88 ± 3.72
DGI [92]	49.89 ± 6.64	46.52 ± 7.10	53.63 ± 7.12	65.75 ± 6.90	57.07 ± 6.49	46.27 ± 4.02	38.28 ± 5.41
GraphCL [116]	53.17 ± 5.44	48.76 ± 7.65	54.65 ± 4.36	69.71 ± 4.44	56.20 ± 2.51	47.05 ± 6.04	40.41 ± 4.57
DSSL [102]	48.36 ± 6.99	48.48 ± 7.53	54.44 ± 8.08	67.12 ± 5.83	56.05 ± 5.05	49.36 ± 4.80	38.67 ± 3.67
GraphACL [103]	54.50 ± 7.33	50.41 ± 6.72	55.13 ± 4.77	67.12 ± 5.45	59.38 ± 4.98	51.62 ± 4.62	44.22 ± 5.96
GPPT [82]	51.50 ± 5.24	48.48 ± 6.02	55.22 ± 7.85	64.36 ± 4.98	58.34 ± 2.59	50.75 ± 3.32	42.06 ± 4.33
GraphPrompt [58]	54.94 ± 6.73	52.53 ± 5.03	58.03 ± 7.26	69.22 ± 3.98	61.59 ± 1.72	53.50 ± 3.99	49.18 ± 4.58
GPF [14]	54.21 ± 8.39	52.89 ± 7.38	56.38 ± 5.47	70.75 ± 6.38	61.41 ± 5.57	53.43 ± 5.29	49.17 ± 4.99
ProNoG [121]	62.20 ± 6.18	54.93 ± 6.48	57.83 ± 7.71	77.70 ± 6.01	64.98 ± 7.47	53.24 ± 4.39	54.27 ± 3.91
GCOPE [136]	55.61 ± 6.40	56.92 ± 5.77	53.64 ± 8.58	74.96 ± 4.17	62.88 ± 6.93	54.03 ± 5.04	50.77 ± 6.51
MDGPT [123]	62.71 ± 5.98	55.85 ± 3.27	58.72 ± 6.23	77.15 ± 5.10	66.30 ± 5.67	55.08 ± 4.71	54.05 ± 4.06
SAMGPT [119]	64.55 ± 6.72	56.43 ± 4.69	59.05 ± 5.97	79.11 ± 4.27	68.96 ± 4.90	55.95 ± 3.34	56.60 ± 5.02
GRAVER (ours)	**68.68 ± 2.82**	**58.10 ± 2.34**	**61.63 ± 2.54**	**81.63 ± 2.21**	**70.43 ± 2.30**	**57.37 ± 2.21**	**59.37 ± 2.68**

Table F.2: Accuracy (% ± std for 20 runs) of **five-shot graph classification**. Best results are presented in **bold** and the runner-ups are underlined. CR = Cora, CS = CiteSeer, PM = PubMed, arXiv = ogbn-arXiv, Tech = ogbn-tech, Home = ogbn-home, Wiki = Wiki-CS. Color denotes domain.

Source	Cross-Dataset				Cross-Domain		
	CS PM Home Wiki	CR PM Home Wiki	CR CS Home Wiki	CR CS PM Home Wiki	CR CS PM Wiki	CR CS PM Home	Home Tech Wiki
Model / Target	CR	CS	PM	Tech	Home	Wiki	arXiv
GCN (bb.) [40]	52.93 ± 4.08	43.91 ± 5.86	55.36 ± 5.26	77.52 ± 6.08	61.78 ± 5.88	40.17 ± 8.30	41.84 ± 5.57
GAT [91]	49.62 ± 5.09	45.25 ± 7.28	54.49 ± 7.30	79.77 ± 6.41	61.45 ± 6.83	39.44 ± 5.28	34.64 ± 4.07
InfoGraph [81]	54.65 ± 4.93	47.21 ± 5.16	59.70 ± 7.12	78.85 ± 5.88	61.94 ± 3.99	43.01 ± 4.32	40.05 ± 5.83
GraphCL [116]	55.22 ± 5.87	46.41 ± 3.80	59.94 ± 5.44	81.38 ± 4.67	62.75 ± 6.53	44.78 ± 5.90	40.98 ± 5.99
DSSL [102]	53.92 ± 5.05	47.49 ± 5.18	62.43 ± 3.61	78.14 ± 7.23	62.07 ± 7.17	51.67 ± 6.68	46.67 ± 3.55
GraphACL [103]	56.19 ± 7.08	45.18 ± 6.14	60.10 ± 7.08	81.50 ± 5.82	69.69 ± 2.53	48.92 ± 7.40	46.93 ± 7.40
GraphPrompt [58]	53.83 ± 5.19	50.47 ± 4.21	64.38 ± 6.40	83.31 ± 5.31	73.77 ± 4.53	55.64 ± 6.13	56.31 ± 4.81
GPF [14]	54.79 ± 4.73	53.90 ± 5.81	63.08 ± 6.36	84.79 ± 5.44	73.13 ± 7.38	56.54 ± 4.89	55.43 ± 5.76
ProNoG [121]	62.09 ± 5.09	57.64 ± 7.57	66.44 ± 6.17	84.17 ± 7.35	73.17 ± 6.16	59.52 ± 4.65	58.08 ± 3.68
GCOPE [136]	63.86 ± 4.75	58.20 ± 5.78	66.39 ± 3.71	85.99 ± 4.17	72.10 ± 5.36	58.42 ± 5.56	58.14 ± 3.20
MDGPT [123]	65.10 ± 4.18	59.32 ± 6.03	67.60 ± 4.64	84.85 ± 3.39	74.30 ± 7.18	61.99 ± 5.51	61.15 ± 3.24
SAMGPT [119]	68.29 ± 3.36	62.08 ± 5.66	67.94 ± 4.56	85.27 ± 4.62	75.83 ± 4.56	63.09 ± 2.69	63.82 ± 3.41
GRAVER (ours)	**71.07 ± 2.51**	**65.45 ± 2.14**	**71.33 ± 2.39**	**89.32 ± 2.57**	**79.18 ± 2.92**	**65.85 ± 1.88**	**65.28 ± 2.58**

F Additional Results

In this section, we provide additional experiment results.

F.1 Additional Results: Few-Shot Node and Graph Classification

Five-Shot Classification (Table F.1 and Table F.2)**.** We further evaluate GRAVER under the ***five-shot*** settings. Results consistently demonstrate that GRAVER maintains superior performance across both node and graph classification tasks in cross-dataset and cross-domain scenarios, extending the positive trends that are observed in the one-shot setting. Specifically, GRAVER achieves significant performance advantages over baselines, marked by notably higher accuracy and substantially reduced variance, emphasizing its robustness against the randomness inherent in few-shot support set selection. While the performance gap narrows slightly owing to improved baseline performance with increased labeled data, GRAVER continues to outperform competitors. These findings highlight its consistent effectiveness in leveraging generative graph vocabularies to robustly and efficiently adapt to limited labeled data. Overall, the additional results provided here reinforce the reliability and superiority of GRAVER in few-shot learning scenarios.

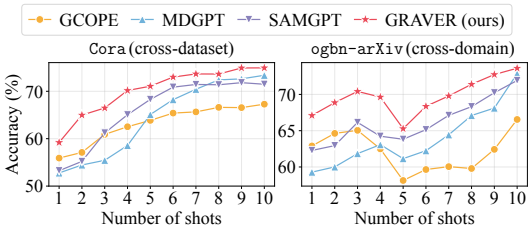


Figure F.1: m-shot graph classification.

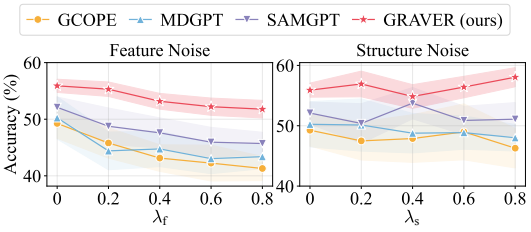


Figure F.2: 1-shot graph classification (Wiki-CS).

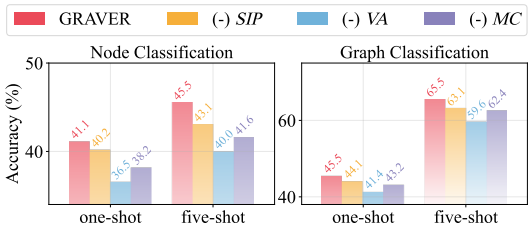


Figure F.3: Ablation studies (CiteSeer).

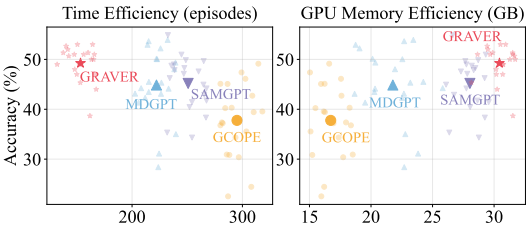


Figure F.4: Efficiency analysis (ogbn-arXiv).

m-Shot Classification (Figure F.1)**.** We additionally analyze the performance under varying shot settings for graph classification tasks. Results further corroborate the conclusions drawn from Figure F.1. Specifically, the performance margin remains evident in the low-shot regime ($m \leqslant 5$), underscoring GRAVER's ability to inject meaningful generative vocabularies even with minimal supervision. Notably, in the cross-domain case, where domain shifts pose a significant challenge, GRAVER's margin remains prominent throughout, highlighting its superior generalization and robustness. These findings reinforce our core claim: generative vocabulary augmentation effectively enhances few-shot adaptation in both low-resource and more relaxed settings.

Robustness against Random Noise (Figure F.2)**.** Compared to the five-shot setting, the one-shot setting reveals a more pronounced performance drop in the presence of noise for baseline methods. In contrast, GRAVER exhibits consistently high accuracy and reduced degradation across all noise levels. Under feature noise, GRAVER shows graceful performance decay, while other models experience sharp declines. Under structure noise, GRAVER not only maintains stability but occasionally improves, likely benefiting from structural diversity introduced by generative augmentation. This contrast highlights GRAVER's robustness advantage, especially in the extremely low-shot regime.

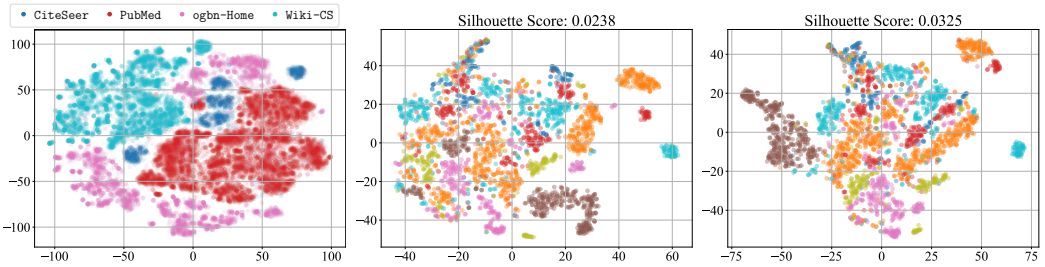
F.2 Additional Results: Ablation Study

Figure F.3 extends the ablation study to the CiteSeer (cross-dataset) setting. Compared to results on ogbn-Home (cross-domain), the relative contributions of the three core components remain consistent. Specifically, removing *VA* causes the most substantial performance drop in both node and graph classification, underscoring its centrality to effective adaptation under extreme supervision scarcity. The degradation from removing *MC* is also evident, indicating the importance of selective knowledge routing. Notably, the performance gaps between full GRAVER and its ablated variants are ***larger*** on CiteSeer than on ogbn-Home, particularly in the five-shot setting. This suggests that GRAVER's mechanisms are especially effective when dealing with smaller, more homogeneous graphs.

F.3 Additional Results: Efficiency Evaluation for Fine-tuning

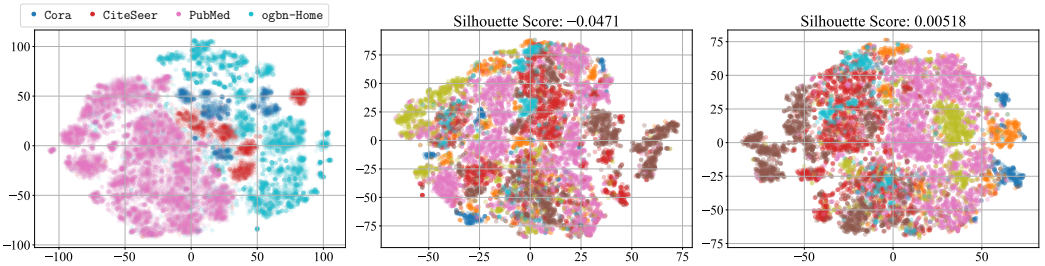
Figure F.4 supplements the efficiency analysis on the large-scale ogbn-arXiv dataset. Compared to the earlier findings on ogbn-Tech, GRAVER continues to demonstrate strong efficiency-performance trade-offs. It consistently converges in fewer episodes and achieves higher accuracy with less variance than baselines. Although GRAVER incurs higher GPU memory usage, its performance gain per memory unit remains the highest among all methods. Notably, the gap in time efficiency is even more pronounced on ogbn-arXiv, suggesting that GRAVER's generative vocabulary and routing design not only stabilize training but also accelerate convergence on large and heterogeneous graphs.

https://doi.org/10.52202/085713-0560



(a) Pre-trained embeddings. (b) Unfinetuned on `Cora`. (c) Fine-tuned on `Cora`.

Figure F.5: Visualization of node embeddings for 1-shot node classification (`Cora`, cross-dataset).



(a) Pre-trained embeddings. (b) Unfinetuned on `Wiki-CS`. (c) Fine-tuned on `Wiki-CS`.

Figure F.6: Visualization of node embeddings for 1-shot node classification (`Wiki-CS`, cross-domain).

F.4 Additional Results: Node Embedding Visualization

Figure F.5 and Figure F.6 provide a qualitative comparison of node embeddings for one-shot node classification under cross-dataset (`Cora`) and cross-domain (`Wiki-CS`) settings, respectively. Across both tasks, we observe consistent improvements in the clustering structure of the embeddings after fine-tuning with GRAVER.

In both settings, the pre-trained embeddings exhibit clear separation among nodes from different source domains, which supports the effectiveness of GRAVER's multi-domain pre-training design. Specifically, the use of LLM-enhanced semantic alignment and shared dimension-wise projection enables graphs with heterogeneous features to be embedded into a unified latent space. Furthermore, the ego-graph disentanglement ensures that factor-specific patterns are preserved across domains. As a result, nodes from different graphs are aligned without sacrificing their structural diversity.

However, despite strong domain-level alignment, the pre-trained embeddings lack strong class-level separability, as evidenced by the relatively low silhouette scores (*e.g.*, 0.0238 on `Cora` and –0.0471 on `Wiki-CS`). Silhouette scores quantitatively measure the class separability of embeddings, where higher values indicate tighter intra-class cohesion and clearer inter-class boundaries. After applying one-shot fine-tuning, silhouette scores increase in both scenarios (*e.g.*, to 0.0325 on `Cora` and 0.00518 on `Wiki-CS`), showing that GRAVER is able to adapt pre-trained embeddings into more class-discriminative structures with minimal supervision. The improvement is more pronounced in the cross-dataset case (`Cora`), while still non-trivial under the harder cross-domain setting (`Wiki-CS`), where semantic and structural gaps are larger.

In summary, GRAVER's pre-training effectively aligns nodes across domains by combining LLM-based semantic projection and ego-graph disentanglement, leading to structured but class-agnostic embeddings. The low silhouette scores reflect poor class separability at this stage. After fine-tuning, the introduction of class-aware vocabulary improves silhouette scores, indicating enhanced class discriminability on top of the well-aligned multi-domain space.

https://doi.org/10.52202/085713-0560

G Additional Related Work

G.1 Multi-Domain Pre-trained Graph Foundation Models

Graph foundation models have rapidly progressed by leveraging self-supervised pre-training to capture universal patterns in graph-structured data [57, 29, 104, 124]. These models aim to distill transferable knowledge from large-scale graphs through tasks such as node feature masking, edge reconstruction, or contrastive instance discrimination. Notable examples include GCC [69], GPT-GNN [30], GPPT [82], L2P-GNN [61], and others [105, 36, 109, 8, 7, 113, 20]. After pre-training, these foundation models are fine-tuned or prompted for specific downstream tasks. However, most existing models are developed under the assumption that the pre-training and downstream domains share similar structural or semantic distributions, often involving homogeneous subgraphs [32] or graphs within a narrow domain [135]. As such, these approaches are vulnerable to domain shift and tend to underperform when directly transferred to out-of-distribution graphs.

To overcome this limitation, recent efforts have explored cross-domain graph learning, where the goal is to adapt knowledge from one domain to another dissimilar domain. Representative models in this direction include GCOPE [136], OMOG [54], PGPRec [112], UniAug [88], and UniGraph [23], which focus on extracting domain-invariant representations to reduce the distribution gap. Despite their effectiveness in single-source transfer settings, these models often lack the capacity to generalize across multiple diverse domains simultaneously. Furthermore, many are designed with task-specific or domain-specific assumptions, limiting their scalability as universal graph foundation models.

To advance toward multi-domain graph foundation models, several recent works explore pre-training strategies that can encode graph knowledge across heterogeneous domains. OFA [54] and HiGPT [87], for example, introduce LLM-guided frameworks that convert node information into natural language for alignment. While promising, these methods are restricted to text-attributed graphs [95, 85] and cannot handle general-purpose structural graphs. Other approaches, such as GCOPE [136], insert virtual nodes to serve as domain bridges, yet these do not achieve semantic feature alignment. MDGPT [123] proposes the use of domain tokens to align multi-domain representations, but it employs SVD-based alignment that overlooks semantic consistency across domains.

G.2 Graph Foundation Models Fine-tuning with Prompt

Fine-tuning pre-trained graph foundation models is a dominant strategy for adapting general-purpose representations to downstream applications. Conventional fine-tuning methods typically update either the full parameter set or selected components of the model to optimize for specific tasks [130, 84, 139, 108, 85, 115, 33, 35, 129, 125]. While such approaches can yield strong task-specific performance, they are often resource-intensive and prone to overfitting.

To address these limitations, parameter-efficient fine-tuning (PEFT) methods have been proposed. Instead of retraining the entire model, PEFT techniques update only a small subset of parameters [143, 89, 19, 106] or incorporate lightweight modules [49, 147, 110] that can be trained with significantly fewer resources. These strategies retain much of the model's pre-trained knowledge while improving adaptability and reducing computational overhead. Despite their advantages, PEFT still faces difficulties in generalizing across domains, where task variability and domain shift may compromise their effectiveness due to the inability to handle inter-domain discrepancies.

In parallel, prompt-based fine-tuning is a promising alternative. Originating from natural language processing [55, 141, 140, 11], prompt learning reframes task adaptation as conditioning the model via learned input tokens or templates, rather than altering model parameters. For graph learning, this paradigm has been extended through the design of graph prompts, which are structured inputs or embeddings that guide the behavior of pre-trained models toward downstream objectives [82, 145, 77, 132]. Prompt tuning enables efficient task adaptation with minimal parameter updates, thus mitigating overfitting and supporting better generalization, especially in few-shot settings.

Building upon this foundation, a variety of prompt learning methods for graphs have been proposed. These include GPPT [82], All in One [83], GraphPrompt [58], Self-Pro [18], SGL-PT [145], GPF-Plus [14], PRODIGY [31], MDGPT [123], HGPROMPT [118], and ProG [149], among others. These models introduce learnable prompts into architectures, allowing them to emphasize salient topological patterns or semantic features relevant to specific tasks, which has shown promise in improving cross-task transferability and enhancing model robustness on structurally complex graphs.

https://doi.org/10.52202/085713-0560

Nevertheless, current graph prompt methods often exhibit limited scalability in multi-domain settings. Many of them are tailored to single-domain tasks or require manual domain-specific prompt designs, restricting their flexibility when faced with diverse or evolving graph distributions. Moreover, the field still lacks a formal understanding of prompt learning in the context of graph structures, in which most methods are empirically driven, without theoretical guarantees or generalization analysis.

G.3 GFM Pre-training with Transferable Patterns

Graph Foundation Models aim to capture structural patterns generalizable across diverse domains and tasks [69, 142, 83, 7, 148, 97, 96]. A core challenge is defining transferable subgraph patterns for pre-training since graphs lack obvious "tokens" like words or patches.

One line of work addresses this by identifying common substructures (motifs, computation trees, *etc.*) as a graph vocabulary that can be learned and reused [142, 146, 47, 138, 144, 97, 117]. For example, Wang et al. [98] propose GFT, which treats each node's computation tree (the tree unrolled from one node's message-passing neighborhood) as a token. By vector-quantizing these computation trees into a discrete vocabulary, GFT's pre-training encourages the model to represent graphs in terms of recurring structural patterns. This approach improved cross-domain generalization and reduced negative knowledge transfer in experiments, illustrating that encoding graphs via transferable pattern "tokens" can serve as a promising pre-training strategy.

Another perspective directly learns frequent subgraph motifs during pre-training [135, 4, 133, 107]. Zhang et al. [133] introduce a motif-driven contrastive framework MICRO-Graph that automatically mines significant subgraph patterns from large graphs. MICRO-Graph clusters similar subgraphs into motif slots and trains a GNN via graph-to-subgraph contrastive learning. By pulling together representations of graphs sharing a motif and pushing apart those without, the GNN learns motif-aware embeddings. This motif-based pre-training yielded improved transfer to downstream tasks like molecular property prediction, as motifs (*e.g.*, functional groups in molecules) carry semantic patterns that generalize across graphs. Similarly, MGSSL [135] incorporates a motif generation objective: a GNN is trained to reconstruct masked substructures and generate likely motif segments within graphs. By capturing the "building blocks" of graphs, MGSSL's multi-level pre-training (at both atom and motif levels) significantly boosted downstream molecular classification performance.

Beyond motifs, researchers have explored universal structural pattern learning for cross-task transfer [43, 142, 148, 134]. For instance, Yin et al. [113] present π-GNN, which is pre-trained on synthetic graphs with known explanatory subgraphs. π-GNN uses a dedicated structural pattern learning module to extract diverse, universal subgraph patterns that occur across graph types, which are then integrated as a comprehensive graph representation. The pre-trained π-GNN can be fine-tuned on real datasets, where it not only improves accuracy but also generalizes its interpretability to new tasks. Notably, a single π-GNN pre-trained on graph-level tasks achieved state-of-the-art explanation quality even on node classification tasks, indicating that the "universal structural patterns" it learned are transferable across task formats.

Early self-supervised graph pre-training methods (*e.g.*, context prediction or graph-level contrastive learning) also implicitly encourage the learning of generalizable structure [26]. However, these often rely on dataset-specific augmentations or predictive tasks. In contrast, the recent trend is to explicitly encode structural commonalities, whether via a subgraph vocabulary [98], motif discovery [133], or synthetic pattern pre-training [113], to build graph foundation models with stronger out-of-distribution robustness. These approaches collectively demonstrate that transferable graph patterns (from small motifs up to rooted computation trees) form a powerful basis for general-purpose graph learning.

G.4 GFM Fine-tuning with Generative Augmentations

After pre-training, adapting graph models to specific downstream tasks can be further enhanced by generative data augmentation techniques [30, 137, 7, 84, 115, 86, 111]. Rather than only tuning on the limited observed graph data, recent works use graph generators to synthesize additional training examples or to adjust the model to the target domain's underlying graph distribution. Such generative augmentations go beyond traditional edge or node perturbations by creating new realistic graph structures, often in a context-aware manner.

One prominent approach is to leverage graphons, which are continuous graph generation functions [2, 68, 72, 12, 71, 84]. Han et al. [21] propose $\mathcal{G}$-Mixup, which adapts Mixup to graphs by interpolating between the underlying graphons of different classes. By augmenting with inter-class hybrid graphs, $\mathcal{G}$-Mixup substantially improved GNN generalization and robustness against noisy labels. The use of a generative graph model is key: it produces realistic graphs that capture smooth transitions between structural patterns of different classes, something unattainable by random edge swaps.

While $\mathcal{G}$-Mixup creates new input graphs, other work focuses on generative fine-tuning objectives to better fit a pre-trained model to the target graph data [48, 84, 9, 53]. A recent example is G-Tuning by Sun et al. [84], which addresses the issue of structural mismatch between pre-training and downstream graphs. They identify that a pre-trained GNN might overfit to patterns from its pre-training graphs and struggle to capture a new graph's structure if the generative processes differ. G-Tuning tackles this by explicitly preserving the "generative patterns" of the downstream graph during fine-tuning. It shows that treating fine-tuning as fitting a generative graph model helps retain broad generalization ability while adapting to new structures.

Another category of generative augmentation focuses on subgraph generation and replacement [3, 131, 41, 101, 24, 22]. These methods generate new sub-components of graphs to enrich training data or to regularize model explanations. For instance, Liu et al. [57] introduce an environment replacement as part of a graph rationalization framework. By training the GNN on both original and these augmented examples, the model learns to identify the subgraph's influence independent of a specific context. Such context-aware subgraph augmentations improve model robustness and interpretability, as the GNN sees multiple generated environments for the same substructure. The success in molecular property prediction tasks demonstrates that generatively augmenting graph contexts can help the model generalize the learned subgraph patterns beyond the limited surroundings.

More broadly, mixture-of-experts (MoE) have been applied to handle diverse generative regimes [27, 94, 59, 99, 20, 123]. Rather than a single generator, multiple generative "experts" can be trained, each specializing in a certain structural pattern or data distribution. Wu et al. [99] develop GraphMETRO, an MoE-based GNN that implicitly performs augmentation by routing each graph through an expert tuned to its distributional subdomain. Each expert produces a representation focusing on a particular structural shift, and a gating network combines their outputs. This idea could be extended by having each expert generate a slight variant of the input and then ensembling the predictions.

In summary, generative augmentations for graph fine-tuning go beyond traditional techniques like edge drops or feature noise by synthesizing structurally meaningful examples or aligning with target distributions. This provides richer training signals that better capture task-specific graph patterns. Empirical studies show that approaches such as graphon-based sampling [21], subgraph replacement [57], and generative alignment [84] consistently improve robustness and generalization. These trends underscore the growing interplay between graph generation and representation learning, enabling more adaptable and data-efficient graph foundation models.

H Limitations and Broader Impact

Limitations. While GRAVER demonstrates strong performance, several peripheral aspects remain to be explored. The vocabulary construction relies on class-aware supervision, which may limit applicability in unsupervised or open-world scenarios. Extending toward unsupervised or weakly supervised vocabulary discovery could enhance generality. In addition, the method is mainly evaluated on classifications, and its effectiveness on more complex reasoning settings, such as link prediction or anomaly detection, remains unclear. Finally, the reuse and compression of disentangled vocabularies across tasks have not been thoroughly examined, which may offer further gains in scalability.

Broader Impact. This work contributes to the growing field of GFMs by addressing a core challenge in robust and efficient few-shot adaptation. The proposed generative vocabulary framework has the potential to improve the stability and scalability of GFM deployment in real-world applications such as scientific discovery, bioinformatics, and recommendation systems, where labeled data is often scarce. By enabling more reliable cross-domain generalization, GRAVER may facilitate broader access to graph-based intelligence capabilities across under-resourced domains. We do not foresee any direct negative societal impacts, as the proposed method does not involve human subjects, sensitive data, or high-risk generative content. Instead, it aims to strengthen the foundation of trustworthy and generalizable graph learning.

 https://doi.org/10.52202/085713-0560

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 abstract and introduction clearly articulate the main contributions, including generative augmentation with transferable vocabularies, graphon-based experts for structure-feature modeling, and robust fine-tuning via MoE-CoE routing. These claims are substantiated by theoretical results (*e.g.*, Proposition 1 and Proposition 2) and extensive experiments (Section 5). The scope and limitations are also reasonably aligned.

   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: The paper includes a dedicated discussion on limitations in Appendix H. It acknowledges the dependency on class-aware supervision during vocabulary construction and generation, which may hinder applicability in unsupervised or open-world settings. Limitations on large-scale domains with highly dynamic distributions are also implied.

   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: The paper provides theoretical assumptions and formal proofs in Appendix for key propositions, such as the vocabulary transferability bound (Proposition 1) and the convergence of generated vocabularies (Proposition 2). Each proof is rigorously derived with necessary assumptions clearly stated (*e.g.*, Lipschitz continuity, permutation invariance).

   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: The paper discloses sufficient experimental details across Sections 5, and references to Appendix D and Appendix E for more settings. The authors provide a public anonymous code repository, which further supports reproducibility.

   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).

 https://doi.org/10.52202/085713-0560

(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 authors share an anonymous code repository (https://anonymous.4open.science/r/GRAVER), which includes code for reproducing key experiment results. Dataset usage and access instructions are also provided or referenced from prior works.

   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, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper thoroughly specifies experimental details in Section 5.1, Appendix D and Appendix E, including baselines, datasets, pre-training/fine-tuning settings, evaluation protocols, hyperparameter selection, and the optimization strategies applied, *etc.*

   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: All main accuracy results are reported with standard deviation across 20 runs, and robustness to noise and hyperparameters is analyzed in Figure 4 and Figure 8. The error bars (std.) and run randomness (few-shot sample selection) is made explicit.

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: The paper includes empirical runtime and GPU memory evaluations in Section 5.4 and Appendix F, where convergence time and memory usage are compared with baselines. The compute resource setups are also listed in Appendix E.3.

   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 paper complies with the NeurIPS Code of Ethics. It uses publicly available datasets with proper citation and does not involve human subjects or sensitive data. There is no evidence of ethical concerns in model design or usage.

   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.

https://doi.org/10.52202/085713-0560

- 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: The paper discusses societal impacts in the broader context of trustworthy graph foundation models in Appendix H. It highlights positive implications in real-world deployment under domain shift and acknowledges potential negative applications if generative vocabularies are misused for adversarial structure injection.

    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 release high-risk models (*e.g.*, LLMs, scraped datasets). Although it involves generative components, they are constrained and do not pose identifiable dual-use risks requiring explicit safeguards.

    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: Datasets and baselines are properly cited and used under appropriate licenses listed in Appendix E.2. The code base will be accompanied by license information once the paper is accepted for publication.

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 GRAVER as a new framework asset, comprising the disentangled vocabulary banks, graphon-based generative experts, and a MoE-CoE routing mechanism. It is well-documented and supported by an anonymized code release.

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: No human subjects or crowdsourced tasks are involved.

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.

https://doi.org/10.52202/085713-0560

- 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: No IRB approval is needed as no human subject research was conducted.

    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: [Yes]

    Justification: The paper explicitly mentions the use of LLMs (*e.g.*, GPT-4, BERT tokenizer) for feature alignment and class-aware text enhancement (Section 4.1), which constitutes a non-standard component of the methodology.

    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.