GEM: A Dual-Scale Architecture for Graph-Level Hierarchical Representation Learning

Zhaowei Wu[1] **Kaizhong Zheng**[1*] **Shuai Jiang**[1]
Guangmingzi Yang[2] **Liangjun Chen**[1] **Badong Chen**[1]

[1]State Key Laboratory of Human-Machine Hybrid Augmented Intelligence,
Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
[2]China Mobile Xiong'an Information and Communication Technology Co., Ltd., China

linus.wu@stu.xjtu.edu.cn kzzheng@xjtu.edu.cn
jessonthy@163.com yangguangmingzi@cmict.chinamobile.com
liangjunchen@xjtu.edu.cn chenbd@mail.xjtu.edu.cn

Abstract

Real-world networks often exhibit small-world properties: dense local clustering yet efficient global propagation. However, current methods struggle to reconcile this hierarchy: Graph Neural Networks (GNNs) suffer from over-smoothing in long-range modeling, while sequence/state-space models often lack the inductive bias to preserve complex graph topology. To bridge this gap, we propose Graph Encoding Mamba (GEM), a dual-scale architecture for graph-level hierarchical representation learning. First, GEM employs a dual-branch encoder that synergizes the local sensitivity of GNNs with the selective scanning of Mamba, effectively capturing both local topology and global shortcuts. Second, a modularity-aware hierarchical pooling mechanism is designed to retain salient community structures during graph coarsening. Crucially, to mitigate structural redundancy from multi-scale fusion, we incorporate an Information Bottleneck objective to distill task-relevant structural patterns, encouraging compact representations and supporting intrinsic interpretability. Extensive experiments on graph-level benchmarks demonstrate that GEM improves performance across biological, social, and brain network datasets.

1 Introduction

Graph-structured data is central to scientific discovery, representing complex systems ranging from molecular structures (Therrien et al., 2025) and brain connectomes (Zheng et al., 2024b) to social networks (Sharma et al., 2024). Many real-world networks exhibit hierarchical small-world organization (Watts and Strogatz, 1998): dense micro-scale motifs form meso-scale communities, which are further connected by sparse shortcuts enabling macro-scale information propagation (see Figure 1). Effective graph-level representation learning therefore requires models that preserve local geometries and community structure while capturing long-range dependencies.

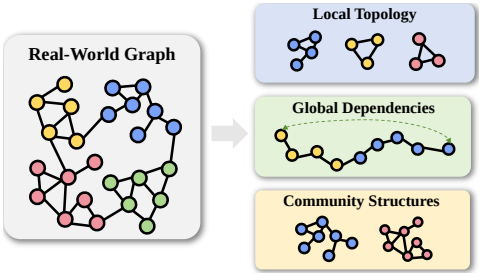


Figure 1: Hierarchical graph structures across local, community, and global scales.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Existing paradigms only partially address this hierarchy. Graph Neural Networks (GNNs) (Kipf, 2016) are effective at local pattern modeling but suffer from over-smoothing and over-squashing when capturing long-range interactions (Alon and Yahav, 2020). Graph Transformers can model global context, but their quadratic complexity limits scalability (Rampášek et al., 2022). State Space Models (SSMs), such as Mamba, provide a linear-time alternative, yet graph linearization may weaken structural inductive bias and local geometric fidelity. This mismatch is further amplified at readout: standard pooling often compresses node representations indiscriminately, collapsing salient meso-scale community structures (Bianchi et al., 2020). Moreover, multi-scale fusion can introduce redundant or spurious interactions, which is undesirable in scientific applications requiring trustworthy predictions (Rudin, 2019).

To address these challenges, we propose Graph Encoding Mamba (GEM), a dual-scale encoding framework with hierarchical readout for graph-level representation learning. GEM is built on the hypothesis that effective graph classification requires jointly preserving three complementary structural scales: local geometric motifs, meso-scale communities, and global shortcuts. Specifically, GEM employs a hybrid parallel encoder, where a GNN branch preserves micro-scale topology and a bidirectional Mamba branch captures macro-scale dependencies with linear graph-size scaling. A modularity-aware pooling strategy then preserves meso-scale community boundaries before graph-level aggregation. Finally, an Information Bottleneck (IB) objective performs structural distillation by filtering redundant interactions from multi-scale fusion and producing compact subgraph rationales for intrinsic interpretability.

Our main contributions are summarized as follows:

- We propose GEM, a graph-level framework that integrates a hybrid GNN-Mamba encoder with modularity-aware hierarchical readout, preserving local topology, modeling global shortcuts, and retaining meso-scale community structure.
- We instantiate the Information Bottleneck with a closed-form Cauchy-Schwarz Quadratic Mutual Information estimator over graph and subgraph embeddings, filtering redundant multi-scale interactions and yielding compact, task-relevant subgraph rationales.
- We demonstrate that GEM learns effective and interpretable graph-level representations, achieving the best mean accuracy across diverse graph domains, producing faithful subgraph rationales, and scaling favorably in memory and runtime.

2 Related Work

Long-Range Dependency Modeling in Graphs: Traditional message-passing GNNs suffer from over-smoothing and limited long-range propagation (Li et al., 2018). In response, Graph Transformers (GTs) employ global self-attention. While models like Graphormer (Ying et al., 2021) and GraphGPS (Rampášek et al., 2022) achieve competitive results, their quadratic complexity motivates scalable variants like Exphormer (Shirzad et al., 2023). Recently, GrokFormer (Ai et al., 2025) integrates spectral filtering to enhance expressivity, though still introducing non-trivial computational overhead. Concurrently, SSMs have emerged as efficient alternatives. Graph-Mamba (Wang et al., 2024) and GMN (Behrouz and Hashemi, 2024) adapt SSMs to graphs via node sequentialization, while GSSC (Huang et al., 2024a) explores permutation equivariance. MbaGCN (He et al., 2025) alternates message aggregation with selective SSMs. In contrast, GEM employs parallel GNN and Mamba branches to preserve local topology and capture long-range dependencies with efficient sequence modeling.

Graph Pooling Methods: Graph pooling is critical for capturing meso-scale community structures in hierarchical small-world networks. While flat pooling (Xu et al., 2018) provides limited hierarchical abstraction, hierarchical approaches pioneered by Diffpool (Ying et al., 2018) have evolved into spectral methods like MinCutPool (Bianchi et al., 2020) and differentiable clustering variants like WGDPool (Xiao et al., 2024). However, these methods often prioritize reconstruction, assignment, or cut objectives, without explicitly preserving meso-scale modular organization. Moreover, conventional aggregation may overlook second-order feature correlations in community representations (Wang and Ji, 2020). In contrast, our modularity-aware pooling encourages topology-aware community boundaries and captures second-order statistical properties.

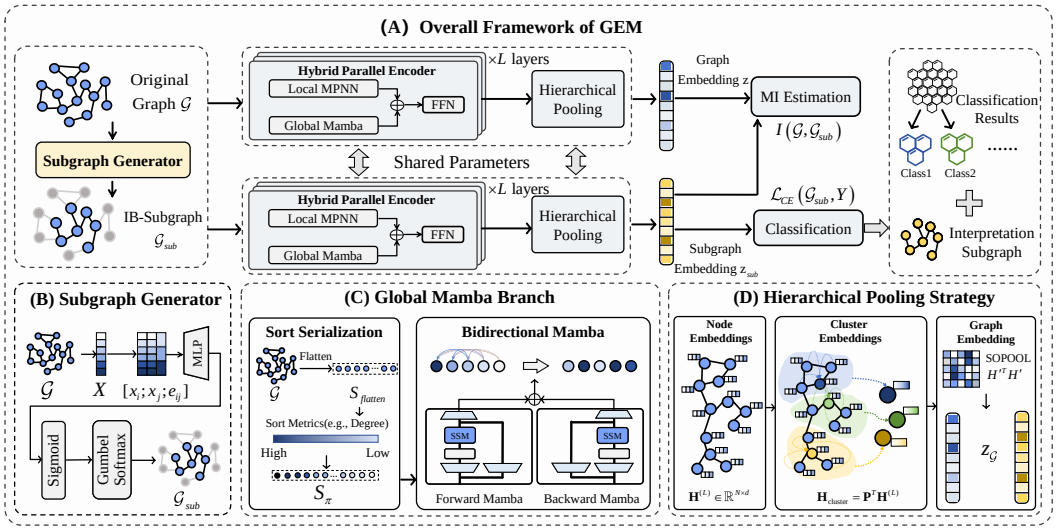


Figure 2: Overview of GEM. (A) Shared training pipeline on the original graph and the distilled subgraph. (B) Subgraph Generator for structural distillation. (C) Bidirectional Mamba for long-range dependency modeling. (D) Hierarchical Pooling for modularity-aware graph summarization.

Interpretability in GNNs: The opacity of GNNs limits their high-stakes adoption (Rudin, 2019). While post-hoc methods (Yuan et al., 2022), such as GNNExplainer (Ying et al., 2019), offer insights, they can suffer from faithfulness gaps (Longa et al., 2025). Consequently, intrinsic interpretability has gained traction, with methods based on the Information Bottleneck (Zheng et al., 2024b) or contrastive sufficiency (Deng and Shen, 2024). GSAT (Miao et al., 2022) further introduces stochastic edge attention under an Information Bottleneck objective to learn interpretable subgraph rationales. Recent advances address specific structural challenges: TopInG (Xin et al., 2025) tackles rationale variability via topological filtration, while SES (Huang et al., 2024b) aligns explanations with predictions through self-supervision. However, these approaches rarely consider the redundancy introduced by integrating local, global, and hierarchical representations. GEM bridges this gap by employing information bottleneck as a structural distillation criterion to filter redundancy from dual-scale fusion and extract compact, task-relevant rationales during prediction.

3 Methodology

3.1 Problem Formulation

Given a graph dataset $\mathcal{D} = \{(\mathcal{G}_i, y_i)\}_{i=1}^{M}$, each graph $\mathcal{G}_i = (\mathbf{A}_i, \mathbf{X}_i, \mathbf{E}_i)$ consists of an adjacency matrix $\mathbf{A}_i \in \{0,1\}^{N_i \times N_i}$, node features $\mathbf{X}_i \in \mathbb{R}^{N_i \times d}$, and optional edge features $\mathbf{E}_i$. The goal of graph classification is to learn a predictor $\Phi : \mathcal{G} \mapsto y$, where $y \in \{1, \ldots, C\}$.

We formulate this task as structural distillation from an Information Bottleneck perspective. Specifically, GEM learns a task-relevant subgraph $\mathcal{G}_{\text{sub}} = f_{\text{sub}}(\mathcal{G})$ and predicts $\hat{y} = f_{\text{cls}}(\mathcal{G}_{\text{sub}})$, aiming to preserve label-sufficient structural information while suppressing task-irrelevant redundancy in multi-scale graph representations. At inference, GEM predicts from the distilled subgraph, while the original-graph embedding is used only for training-time structural compression.

3.2 Overview of GEM

Figure 2 presents the overall architecture of GEM. During training, GEM processes the original graph and its distilled subgraph through a shared Hybrid Parallel Encoder to capture complementary structural signals: a local message-passing branch preserves neighborhood-level geometry, while a global bidirectional Mamba branch models long-range dependencies after graph linearization. The resulting node representations are then summarized by a Hierarchical Pooling module, which identifies community-level structures and applies second-order statistics to produce graph-level embeddings.

This design aligns the model with the hierarchical organization of real-world graphs: local aggregation captures micro-scale interactions, global sequence modeling captures macro-scale shortcuts, and hierarchical pooling preserves meso-scale community structures. The following sections detail these three components and the final training objective.

3.3 Dual-Scale Structural Encoding

Deep message-passing GNNs enlarge receptive fields through repeated neighborhood aggregation, but this often causes over-smoothing and limits long-range dependency modeling. GEM therefore decouples structural encoding into a local GNN branch for *micro-scale* neighborhood geometry and a Bidirectional Mamba branch for *macro-scale* shortcut dependencies, which are fused into a unified node representation.

3.3.1 Micro-Scale: Structure-Aware Message Passing

The local branch preserves micro-scale neighborhood geometry by injecting the learned structural weight w_{ij} into message passing. Instead of treating all observed edges uniformly, GEM conditions each pairwise message on w_{ij}, which is produced by the subgraph generator in Sec. 3.5:

$$\mathbf{m}_{ij}^{(l)} = \mathrm{MLP}_{\mathrm{msg}}\big([\mathbf{h}_i^{(l-1)}; \mathbf{h}_j^{(l-1)}; w_{ij}]\big), \quad \mathbf{h}_{i,\mathrm{local}}^{(l)} = \mathrm{AGG}_{j\in\mathcal{N}(i)}\mathbf{m}_{ij}^{(l)}. \tag{1}$$

Aggregating over local neighborhoods yields $\mathbf{H}_{\mathrm{local}}^{(l)}$, where w_{ij} provides an edge-level structural signal that helps the message function distinguish task-relevant interactions from redundant ones.

3.3.2 Macro-Scale: Bidirectional State Space Modeling

To capture *macro-scale* dependencies beyond local neighborhoods, GEM instantiates the global branch with Mamba, a selective state space model that provides linear sequence complexity. Since Mamba operates on ordered sequences, we first linearize graph nodes by a permutation π. In practice, we sort nodes by descending degree centrality, using high-connectivity nodes as a lightweight proxy for global information flow. This yields a node sequence $S_\pi = \pi(\mathbf{H}^{(l-1)})$.

However, any one-dimensional linearization may introduce directional bias. We therefore use a Bidirectional Mamba block that scans S_π in both forward and backward directions. The backward output is reversed back before fusion to ensure node-wise alignment:

$$\mathbf{H}_{\mathrm{global}}^{(l)} = \mathrm{MLP}\left(\left[\mathrm{SSM}(S_\pi; \overrightarrow{\Theta}); \mathrm{Rev}^{-1}\left(\mathrm{SSM}(\mathrm{Rev}(S_\pi); \overleftarrow{\Theta})\right)\right]\right), \tag{2}$$

where $\overrightarrow{\Theta}$ and $\overleftarrow{\Theta}$ denote the parameters of the forward and backward scans. This bidirectional design reduces directional bias and enables each node to aggregate global contextual information from the serialized graph view. Additional details on the continuous SSM formulation and graph linearization are provided in Appendix A.2.

Since $\mathbf{H}_{\mathrm{global}}^{(l)}$ follows the sequence order S_π, we apply the inverse permutation π^{-1} before fusing it with the local branch:

$$\mathbf{H}^{(l)} = \mathbf{H}^{(l-1)} + \mathrm{MLP}_{\mathrm{fuse}}\left(\mathbf{H}_{\mathrm{local}}^{(l)} + \pi^{-1}(\mathbf{H}_{\mathrm{global}}^{(l)})\right). \tag{3}$$

3.4 Meso-Scale Preservation

Flat readout functions, such as global sum or mean pooling, directly collapse the final node feature matrix $\mathbf{H}^{(L)} \in \mathbb{R}^{N\times d}$ into a graph embedding $\mathbf{z}_{\mathcal{G}}$. While efficient, such first-order aggregation may obscure *meso-scale* modularity, i.e., community structures that bridge local motifs and global topology. To preserve this intermediate scale, GEM introduces a Hierarchical Pooling Strategy that first identifies topology-aware communities and then summarizes their representations through second-order feature statistics.

3.4.1 Modularity-Guided Community Identification

Our objective is to identify structurally coherent communities as meso-scale coarse-graining units. Given node embeddings $\mathbf{H}^{(L)} \in \mathbb{R}^{N\times d}$ and k learnable cluster centroids $\mathbf{C} \in \mathbb{R}^{k\times d}$, we compute a

soft assignment matrix $\mathbf{P} \in \mathbb{R}^{N \times k}$ via a temperature-scaled softmax:

$$P_{ic} = \frac{\exp(\mathbf{h}_i^\top \mathbf{c}_c / \tau)}{\sum_{r=1}^{k} \exp(\mathbf{h}_i^\top \mathbf{c}_r / \tau)}, \tag{4}$$

where $\mathbf{c}_c$ is the c-th cluster centroid and τ controls the sharpness of the assignment distribution.

To make the learned communities topology-aware, we adopt the Newman-Girvan modularity criterion (Newman, 2006). Let $\mathbf{B} = \mathbf{A} - \mathbf{d}\mathbf{d}^\top/(2m)$ be the modularity matrix, where $\mathbf{d}$ is the degree vector and m is the number of edges. Replacing the discrete community indicator in classical modularity with the soft assignment matrix $\mathbf{P}$ yields the differentiable relaxation:

$$\mathcal{L}_{\text{mod}} = -\frac{1}{2m} \text{Tr}(\mathbf{P}^\top \mathbf{B} \mathbf{P}). \tag{5}$$

Minimizing $\mathcal{L}_{\text{mod}}$ encourages nodes within the same soft community to have higher-than-expected connectivity under the modularity null model. The detailed derivation from classical modularity maximization is provided in Appendix A.3. The pooled community features are then obtained as $\mathbf{H}_{\text{cluster}} = \mathbf{P}^\top \mathbf{H}^{(L)} \in \mathbb{R}^{k \times d}$.

3.4.2 Second-Order Statistical Readout

We employ Second-Order Pooling (SOPOOL) (Wang and Ji, 2020) to capture pairwise feature interactions within the learned community representations beyond first-order aggregation. Specifically, the community features are first projected by a learnable mapping $\mathbf{R} \in \mathbb{R}^{d \times d'}$, yielding $\mathbf{H}' = \mathbf{H}_{\text{cluster}}\mathbf{R}$. We then compute the uncentered second-order statistics via the Gram matrix:

$$\mathbf{G} = {\mathbf{H}'}^\top \mathbf{H}' \in \mathbb{R}^{d' \times d'}. \tag{6}$$

The matrix $\mathbf{G}$ aggregates pairwise interactions between feature dimensions across all communities. We flatten its upper triangular part to obtain the final graph representation $\mathbf{z}_{\mathcal{G}}$, aggregating second-order feature statistics across communities.

3.5 Structural Distillation via Information Bottleneck

Although the dual-scale encoder captures both local and global dependencies, multi-scale fusion may introduce task-irrelevant structural redundancy. GEM therefore performs structural distillation from an Information Bottleneck perspective, retaining label-sufficient structures while suppressing unnecessary graph information.

3.5.1 Learnable Subgraph Generator

To achieve this distillation, we design a learnable subgraph generator that assigns each observed edge a differentiable near-binary weight w_{ij}, collectively forming an edge selection matrix $\mathbf{W}$. For each edge (i, j), we first compute an importance probability via an MLP conditioned on the incident node features and optional edge features:

$$p_{ij} = \sigma\left(\text{MLP}_{\text{sub}}\left([\mathbf{x}_i; \mathbf{x}_j; \mathbf{e}_{ij}]\right)\right). \tag{7}$$

We then apply a Binary Concrete (Gumbel-Softmax) relaxation (Maddison et al., 2016) to obtain the edge gate $w_{ij} \in [0, 1]$, which approximates a binary keep/drop decision while remaining differentiable. The resulting gates provide edge-level structural signals in message passing, enabling GEM to distinguish task-relevant connections from redundant ones.

3.5.2 The Information Bottleneck Objective

We constrain the subgraph generator using the IB principle, encouraging the learned subgraph $\mathcal{G}_{\text{sub}}$ to be label-sufficient while remaining compact with respect to the original graph $\mathcal{G}$. This leads to the following IB Lagrangian:

$$\min_{\theta} \mathcal{L}_{\text{IB}} = \underbrace{-I(\mathcal{G}_{\text{sub}}; Y)}_{\text{Sufficiency}} + \beta \underbrace{I(\mathcal{G}; \mathcal{G}_{\text{sub}})}_{\text{Compression}}, \tag{8}$$

where β controls the trade-off between predictive sufficiency and structural compression. We instantiate the two terms with tractable surrogate losses.

Sufficiency Term. The sufficiency term encourages $\mathcal{G}_{\text{sub}}$ to retain label-predictive information. Since $-I(\mathcal{G}_{\text{sub}}; Y) = H(Y|\mathcal{G}_{\text{sub}}) - H(Y)$ and $H(Y)$ is constant, we optimize a cross-entropy surrogate for the conditional entropy:

$$\mathcal{L}_{\text{CE}} = \mathbb{E}_{\mathcal{G}_{\text{sub}},Y}\big[-\log q_\theta(Y|\mathcal{G}_{\text{sub}})\big], \tag{9}$$

where q_θ denotes the classifier.

Compression Term. Directly estimating $I(\mathcal{G}; \mathcal{G}_{\text{sub}})$ is intractable because graph topology is discrete and high-dimensional. We therefore use the dependence between continuous graph-level embeddings $\mathbf{z}_\mathcal{G}$ and $\mathbf{z}_{\text{sub}}$, obtained from the original graph and the distilled subgraph, respectively, as a tractable surrogate. Since $\mathbf{z}_{\text{sub}}$ is computed from the edge-masked graph, gradients from this term are back-propagated to the edge gates $\mathbf{W}$, thereby regularizing the learned subgraph structure.

To obtain a stable non-parametric dependence measure, we use Cauchy-Schwarz Quadratic Mutual Information (CS-QMI), which measures the divergence between the joint distribution and the product of marginals:

$$I_{\text{CS}}(\mathbf{z}_\mathcal{G}; \mathbf{z}_{\text{sub}}) = D_{\text{CS}}\Big(p(\mathbf{z}_\mathcal{G}, \mathbf{z}_{\text{sub}}) \,\|\, p(\mathbf{z}_\mathcal{G})p(\mathbf{z}_{\text{sub}})\Big), \tag{10}$$

where D_{CS} denotes the Cauchy-Schwarz divergence. In practice, given a batch of B graph pairs, we compute the closed-form kernel estimator of CS-QMI using Gaussian Gram matrices over $\mathbf{Z}_\mathcal{G}$ and $\mathbf{Z}_{\text{sub}}$, and minimize it as the structural compression loss:

$$\mathcal{L}_{\text{mi}} = \hat{I}_{\text{CS}}(\mathbf{Z}_\mathcal{G}; \mathbf{Z}_{\text{sub}}). \tag{11}$$

The cross-entropy term prevents trivial collapse, since an over-compressed subgraph would lose label-predictive information and increase $\mathcal{L}_{\text{CE}}$. Detailed definitions of CS divergence and the closed-form estimator are provided in Appendix A.4.

3.6 Training Objective

GEM is trained end-to-end by jointly optimizing predictive sufficiency, structural compression, and topology-aware community preservation:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \beta\mathcal{L}_{\text{mi}} + \alpha\mathcal{L}_{\text{mod}}, \tag{12}$$

where $\mathcal{L}_{\text{CE}}$, $\mathcal{L}_{\text{mi}}$, and $\mathcal{L}_{\text{mod}}$ denote the classification, structural compression, and modularity preservation losses, respectively. The hyperparameters α and β control the strengths of modularity preservation and structural compression.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate GEM on graph-level benchmarks from three domains: *Bioinformatics* (MUTAG, PROTEINS, NCI1), *Social Networks* (COLLAB, REDDIT-BINARY) from TUDataset (Morris et al., 2020), and *Brain Networks* (ABIDE (Di Martino et al., 2014), SRPBS (Tanaka et al., 2021)). Scalability is assessed on MalNet-Tiny (Freitas et al., 2020), with OGB graph benchmarks (Hu et al., 2020) used for throughput profiling. Interpretability is evaluated on BA-2Motifs (Luo et al., 2020) and MUTAG. We further report a lightweight long-range benchmark analysis on LRGB (Dwivedi et al., 2022) in Appendix. Detailed dataset statistics are also provided in Appendix B.1.

Baselines. We compare against four representative categories: *Classic GNNs* (GCN (Kipf, 2016), GAT (Veličković et al., 2017), GIN (Xu et al., 2018)), *Graph Transformers* (Graphormer (Ying et al., 2021), GraphGPS (Rampášek et al., 2022), GrokFormer (Ai et al., 2025)), *Graph Mambas* (Graph-Mamba (Wang et al., 2024), GSSC (Huang et al., 2024a), MbaGCN (He et al., 2025)), and *Interpretable GNNs* (DIR-GNN (Wu et al., 2022), CI-GNN (Zheng et al., 2024a), TopInG (Xin et al., 2025)). All baselines follow their original configurations under the same split strategy.

Implementation & Metrics. GEM is implemented in PyTorch Geometric (Fey and Lenssen, 2019) and trained with AdamW on NVIDIA RTX 4090. We report mean±std over 10 independent runs with

Table 1: Overall performance comparison on graph classification benchmarks. We report mean accuracy (%) with standard deviation as subscript. **Bold** and underline denote the best and second-best mean results, respectively.

Method	MUTAG	PROTEINS	NCI1	REDDIT-B	COLLAB	SRPBS	ABIDE
GCN	$80.00_{\pm 6.86}$	$73.93_{\pm 3.31}$	$73.34_{\pm 3.89}$	$83.50_{\pm 5.53}$	$71.72_{\pm 1.55}$	$82.11_{\pm 4.70}$	$64.18_{\pm 2.37}$
GAT	$81.05_{\pm 2.88}$	$74.11_{\pm 4.47}$	$70.85_{\pm 1.08}$	$79.00_{\pm 3.16}$	$70.28_{\pm 1.54}$	$83.16_{\pm 4.40}$	$65.27_{\pm 1.75}$
GIN	$89.47_{\pm 5.27}$	$76.07_{\pm 1.47}$	$80.66_{\pm 1.16}$	$88.50_{\pm 5.01}$	$75.56_{\pm 2.08}$	$84.21_{\pm 3.72}$	$68.36_{\pm 3.30}$
Graphormer	$88.95_{\pm 4.21}$	$76.85_{\pm 3.31}$	$79.85_{\pm 2.32}$	$91.80_{\pm 2.21}$	$78.24_{\pm 2.13}$	$86.51_{\pm 4.42}$	$67.53_{\pm 3.21}$
GraphGPS	$91.13_{\pm 2.53}$	$77.12_{\pm 1.48}$	$78.54_{\pm 3.04}$	$\underline{92.40}_{\pm 1.53}$	$77.81_{\pm 1.85}$	$\underline{88.23}_{\pm 3.11}$	$65.42_{\pm 2.92}$
GrokFormer	$89.55_{\pm 3.11}$	$75.45_{\pm 3.16}$	$80.48_{\pm 4.31}$	$92.10_{\pm 4.83}$	$\underline{79.14}_{\pm 1.91}$	$87.13_{\pm 3.59}$	$68.20_{\pm 5.41}$
Graph-Mamba	$89.21_{\pm 5.52}$	$74.33_{\pm 2.54}$	$80.11_{\pm 2.41}$	$92.20_{\pm 2.03}$	$78.55_{\pm 2.01}$	$87.52_{\pm 3.51}$	$68.81_{\pm 2.75}$
GSSC	$90.83_{\pm 5.01}$	$\underline{78.81}_{\pm 3.03}$	$78.03_{\pm 1.82}$	$91.50_{\pm 3.12}$	$74.92_{\pm 3.36}$	$88.10_{\pm 2.92}$	$66.14_{\pm 3.35}$
MbaGCN	$90.13_{\pm 4.79}$	$77.51_{\pm 2.92}$	$\underline{81.95}_{\pm 2.10}$	$91.80_{\pm 1.73}$	$77.25_{\pm 2.76}$	$84.33_{\pm 4.58}$	$67.32_{\pm 4.37}$
DIR-GNN	$86.58_{\pm 6.32}$	$77.04_{\pm 2.52}$	$71.42_{\pm 2.21}$	$85.60_{\pm 4.51}$	$75.13_{\pm 3.02}$	$86.86_{\pm 3.82}$	$68.01_{\pm 1.92}$
CI-GNN	$89.12_{\pm 4.51}$	$78.02_{\pm 2.23}$	$79.05_{\pm 5.11}$	$86.10_{\pm 3.82}$	$76.22_{\pm 2.94}$	$87.53_{\pm 3.30}$	$\underline{68.93}_{\pm 2.12}$
TopInG	$\underline{93.02}_{\pm 5.61}$	$75.81_{\pm 2.64}$	$78.55_{\pm 3.49}$	$87.90_{\pm 5.31}$	$75.88_{\pm 3.91}$	$82.53_{\pm 3.51}$	$66.22_{\pm 3.21}$
GEM	$\mathbf{94.74}_{\pm 3.72}$	$\mathbf{81.96}_{\pm 2.78}$	$\mathbf{82.68}_{\pm 2.72}$	$\mathbf{93.40}_{\pm 1.29}$	$\mathbf{80.24}_{\pm 1.01}$	$\mathbf{93.68}_{\pm 4.41}$	$\mathbf{71.09}_{\pm 1.75}$

random 80/10/10 splits. Hyperparameters are selected by validation-set grid search; sensitivity and ordering robustness analyses are reported in Appendix. Graph classification performance is evaluated by *Accuracy*. For interpretability, we measure *AUC* on synthetic motifs and *Fidelity+*/*Fidelity-* on real-world graphs (Longa et al., 2025).

4.2 Performance Comparison

Table 1 reports the graph-level classification performance of GEM across seven benchmarks from bioinformatics, social networks, and brain connectivity domains. GEM achieves the highest mean accuracy on all datasets, indicating consistent gains across diverse graph distributions.

On social network datasets, GEM improves over strong global-modeling baselines, including GraphGPS and GrokFormer, by 1.0 and 1.1 points on REDDIT-BINARY and COLLAB, respectively. On bioinformatics datasets, GEM outperforms the strongest Graph Mamba variants by 3.9, 3.2, and 0.7 points on MUTAG, PROTEINS, and NCI1, suggesting the benefit of complementing sequence-based global modeling with local graph inductive bias. On brain network datasets, GEM obtains larger gains, improving over the second-best methods by 5.45 points on SRPBS and 2.16 points on ABIDE, showing its effectiveness on graphs with domain-specific meso-scale organization. The improvement on SRPBS is further supported by a two-sided paired t-test with Holm correction ($p_{\mathrm{adj}} = 0.033$). These results are consistent with GEM's three-scale design: local message passing preserves fine-grained motifs, bidirectional Mamba captures shortcut-mediated long-range context, and modularity-aware pooling retains community-level organization.

4.3 Ablation Studies

To validate the effectiveness of the key architectural components in GEM, we conduct ablation studies on three representative datasets from different domains: PROTEINS (bioinformatics), REDDIT-BINARY (social networks), and ABIDE (brain networks).

4.3.1 Impact of Core Components

Table 2 reports the ablation results of the major components in GEM. Variants (B) and (C) remove the local GNN branch and the global Mamba branch, respectively; variant (D) replaces the bidirectional Mamba encoder with a unidirectional one; variants (E) and (F) remove $\mathcal{L}_{\mathrm{mod}}$ and $\mathcal{L}_{\mathrm{mi}}$ by setting $\alpha = 0$ and $\beta = 0$, respectively. Removing either the local or global branch leads to clear performance degradation, confirming their complementary roles in modeling local topology and long-range interactions. The relative importance of the two branches is dataset-dependent: removing the global branch causes larger drops on PROTEINS and ABIDE, while removing the local branch yields the largest drop on REDDIT-BINARY.

Replacing bidirectional scanning with Uni-Mamba consistently reduces performance across all three representative datasets, indicating that bidirectional sequence modeling provides additional global context. Disabling either $\mathcal{L}_{\text{mod}}$ or $\mathcal{L}_{\text{mi}}$ also degrades performance, supporting the contribution of meso-scale community preservation and redundancy-controlled structural distillation within the fused representation. Overall, these results show that GEM's gains arise from the joint effect of dual-scale encoding, modularity-aware pooling, and IB-based representation compression. Additional analyses in Appendix B.3 and Appendix B.4 examine the sensitivity to regularization strengths and the robustness of structural node ordering.

Table 2: Ablation study of GEM. Accuracy (%) with standard deviation as subscript.

Model Variant	PROTEINS	REDDIT-B	ABIDE
(A) GEM (Full Model)	**81.96**$_{\pm 2.78}$	**93.40**$_{\pm 1.29}$	**71.09**$_{\pm 1.75}$
(B) w/o Local Branch	78.23$_{\pm 3.15}$	88.20$_{\pm 1.25}$	68.01$_{\pm 2.11}$
(C) w/o Global Branch	74.51$_{\pm 3.58}$	88.90$_{\pm 2.25}$	66.33$_{\pm 2.48}$
(D) Uni-Mamba	80.85$_{\pm 2.91}$	92.60$_{\pm 1.88}$	70.15$_{\pm 1.95}$
(E) w/o $\mathcal{L}_{\text{mod}}$ ($\alpha = 0$)	79.17$_{\pm 3.05}$	90.60$_{\pm 1.91}$	69.25$_{\pm 2.05}$
(F) w/o $\mathcal{L}_{\text{mi}}$ ($\beta = 0$)	80.88$_{\pm 2.99}$	90.70$_{\pm 1.76}$	70.47$_{\pm 1.99}$

4.3.2 Analysis of Pooling Strategy

We compare the proposed hierarchical pooling strategy with seven alternative readout methods, including simple first-order pooling, node selection-based pooling, and advanced global readouts. As shown in Figure 3, our method consistently outperforms all baselines across the three representative datasets. These gains suggest the benefit of the two-stage design, which first identifies meaningful community structures via modularity-guided clustering and then aggregates second-order feature statistics across community representations. This further supports the importance of preserving meso-scale community information before graph-level aggregation.

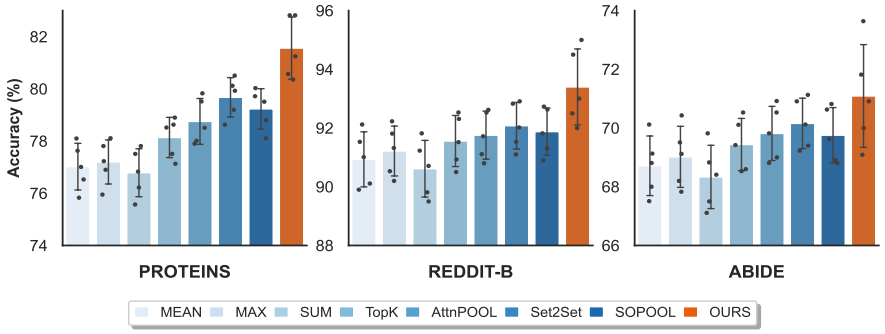


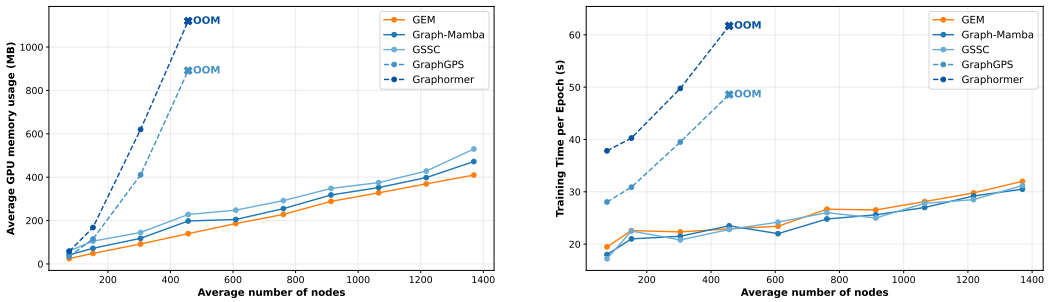
Figure 3: Pooling strategy comparison. Bars show mean accuracy; dots indicate individual runs.

4.4 Analysis of Computational Efficiency

Theoretically, GEM scales approximately linearly with graph size for fixed batch size and model dimensions, while CS-QMI introduces a batch-level Gram-matrix overhead. We evaluate computational efficiency on MalNet-Tiny subsets with increasing average graph sizes, using a fixed batch size of 16 on a single NVIDIA RTX 4090. Figure 4 reports peak GPU memory and per-epoch training time. Transformer-based models, including Graphormer and GraphGPS, exhibit rapidly increasing memory and runtime costs and encounter OOM failures as graph size grows. In contrast, Mamba-based models scale smoothly across the tested graph sizes. GEM achieves the lowest memory footprint among all compared methods and maintains runtime comparable to Graph-Mamba and GSSC. As the average number of nodes increases from 76 to 1,370, GEM's epoch time increases only from 19.5s to 32.0s, suggesting that degree-based linearization and CS-QMI distillation introduce limited overhead over the tested graph sizes. Detailed complexity analysis and additional OGB throughput profiling are provided in Appendix A.5 and Appendix C.1, respectively.

4.5 Interpretability Analysis

Quantitative Analysis. We evaluate explanation quality from two perspectives. *Plausibility* is measured on BA-2Motifs (Luo et al., 2020) using AUC against ground-truth motifs. *Fidelity* is evaluated on MUTAG using Fidelity+ and Fidelity- (Longa et al., 2025), where Fidelity+ measures the necessity of selected substructures and Fidelity- measures whether the retained explanation is sufficient for



(a) Peak GPU memory versus graph size.

(b) Training time per epoch versus graph size.

Figure 4: Scalability analysis on MalNet-Tiny. GEM achieves the lowest peak GPU memory and maintains comparable per-epoch training time as graph size increases, while transformer-based models encounter OOM on larger graphs.

preserving the prediction. We compare GEM with post-hoc explainers, including GNNExplainer and PGExplainer, and intrinsically interpretable baselines, including CI-GNN, TopInG, and GSAT.

As shown in Table 3, GEM achieves the highest AUC on BA-2Motifs, indicating more accurate recovery of ground-truth motifs. On MUTAG, GEM obtains the highest Fidelity+ and the lowest Fidelity-, suggesting that its extracted subgraphs better capture decision-relevant information under both necessity and sufficiency criteria. Compared with GSAT, a closely related IB-based edge-masking method, GEM achieves higher AUC (0.97 vs. 0.95) and Fidelity+ (0.81 vs. 0.77), along with lower Fidelity- (0.11 vs. 0.22). These results support the effectiveness of GEM in extracting faithful, task-relevant subgraph rationales.

Table 3: Quantitative interpretability results. Mean and standard deviation are reported over 5 runs.

Method	PLAUSIBILITY	FIDELITY	
	AUC (↑)	Fid+ (↑)	Fid- (↓)
Post-hoc Baselines			
GNNExplainer	$0.85_{\pm 0.01}$	$0.65_{\pm 0.03}$	$0.32_{\pm 0.04}$
PGExplainer	$0.86_{\pm 0.01}$	$0.69_{\pm 0.02}$	$0.25_{\pm 0.03}$
Built-in Baselines			
CI-GNN	$0.88_{\pm 0.01}$	$0.76_{\pm 0.02}$	$0.15_{\pm 0.02}$
TopInG	$0.95_{\pm 0.02}$	$0.71_{\pm 0.02}$	$0.20_{\pm 0.02}$
GSAT	$0.95_{\pm 0.01}$	$0.77_{\pm 0.02}$	$0.22_{\pm 0.01}$
GEM (Ours)	$\mathbf{0.97}_{\pm 0.01}$	$\mathbf{0.81}_{\pm 0.01}$	$\mathbf{0.11}_{\pm 0.01}$

Qualitative Analysis. We further provide qualitative visualizations to illustrate the explanations produced by GEM. Figure 5 shows an example from BA-2Motifs containing a ground-truth "house" motif. GEM recovers the complete motif structure, whereas the post-hoc baseline PGExplainer highlights only a partial substructure. This behavior is consistent with IB-based structural distillation: the learned edge mask suppresses background BA edges while retaining the class-determining motif as a compact rationale. This example illustrates GEM's ability to extract structurally coherent and task-relevant subgraph rationales.

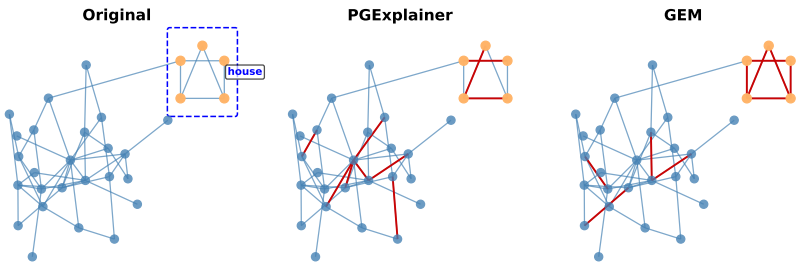


Figure 5: Qualitative comparison on BA-2Motifs. GEM recovers the complete "house" motif (red edges), while PGExplainer identifies only a partial substructure.

5 Conclusion and Limitations

We proposed GEM, a dual-scale framework for hierarchical graph-level representation learning that aligns with the multi-scale organization of real-world graphs. By decoupling local topology modeling and global dependency encoding, preserving meso-scale community structures, and incorporating Information Bottleneck-guided structural distillation, GEM achieves strong classification performance across diverse graph domains, along with favorable efficiency and intrinsic interpretability.

Limitations and Future Work. GEM is primarily designed for graph-level representation learning, where modularity-aware pooling and graph-level readout are central to the architecture. Although we provide a lightweight node-level adaptation to examine transferability, developing task-specialized extensions for broader graph learning scenarios remains future work. Moreover, GEM currently relies on explicit structural ordering for global sequence modeling; adaptive ordering strategies may further improve its flexibility.

Acknowledgments and Disclosure of Funding

This work was supported by Brain Science and Brain-like Intelligence Technology-National Science and Technology Major Project (No.2025ZD0217100) and the National Natural Science Foundation of China (32541016,62436005, U25A20540,62473303).

Competing Interests. The authors declare no competing interests.

References

Guoguo Ai, Guansong Pang, Hezhe Qiao, Yuan Gao, and Hui Yan. Grokformer: Graph fourier kolmogorov-arnold transformers. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 21277–21296. PMLR, 2025.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

Ali Behrouz and Farnoosh Hashemi. Graph mamba: Towards learning on graphs with state space models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 119–130, 2024.

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *International conference on machine learning*, pages 874–883. PMLR, 2020.

Jiale Deng and Yanyan Shen. Self-interpretable graph learning with sufficient and necessary explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 11749–11756, 2024.

Adriana Di Martino, Chao-Gan Yan, Qingyang Li, Erin Denio, Francisco X Castellanos, Kaat Alaerts, Jeffrey S Anderson, Michal Assaf, Susan Y Bookheimer, Mirella Dapretto, et al. The autism brain imaging data exchange: towards a large-scale evaluation of the intrinsic brain architecture in autism. *Molecular psychiatry*, 19(6):659–667, 2014.

Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Scott Freitas, Yuxiao Dong, Joshua Neil, and Duen Horng Chau. A large-scale database for graph representation learning. *arXiv preprint arXiv:2011.07682*, 2020.

Xin He, Yili Wang, Wenqi Fan, Xu Shen, Xin Juan, Rui Miao, and Xin Wang. Mamba-based graph convolutional networks: Tackling over-smoothing with selective state space. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5345–5353. ijcai.org, 2025.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Yinan Huang, Siqi Miao, and Pan Li. What can we learn from state space models for machine learning on graphs? *arXiv preprint arXiv:2406.05815*, 2024a.

Zhenhua Huang, Kunhao Li, Shaojie Wang, Zhaohong Jia, Wentao Zhu, and Sharad Mehrotra. Ses: Bridging the gap between explainability and prediction of graph neural networks. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 2945–2958. IEEE, 2024b.

TN Kipf. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Antonio Longa, Steve Azzolin, Gabriele Santin, Giulia Cencetti, Pietro Liò, Bruno Lepri, and Andrea Passerini. Explaining the explainers in graph neural networks: a comparative study. *ACM Computing Surveys*, 57(5):1–37, 2025.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *International conference on machine learning*, pages 15524–15543. PMLR, 2022.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

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

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

Kartik Sharma, Yeon-Chang Lee, Sivagami Nambi, Aditya Salian, Shlok Shah, Sang-Wook Kim, and Srijan Kumar. A survey of graph neural networks for social recommender systems. *ACM Computing Surveys*, 56(10):1–34, 2024.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, pages 31613–31632. PMLR, 2023.

Saori C Tanaka, Ayumu Yamashita, Noriaki Yahata, Takashi Itahashi, Giuseppe Lisi, Takashi Yamada, Naho Ichikawa, Masahiro Takamura, Yujiro Yoshihara, Akira Kunimatsu, et al. A multi-site, multi-disorder resting-state magnetic resonance image database. *Scientific data*, 8(1):227, 2021.

Félix Therrien, Edward H Sargent, and Oleksandr Voznyy. Using gnn property predictors as molecule generators. *Nature Communications*, 16(1):4301, 2025.

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

Chloe Wang, Oleksii Tsepa, Jun Ma, and Bo Wang. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. *arXiv preprint arXiv:2402.00789*, 2024.

Zhengyang Wang and Shuiwang Ji. Second-order pooling for graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):6870–6880, 2020.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393 (6684):440–442, 1998.

Ying-Xin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. Discovering invariant rationales for graph neural networks. *arXiv preprint arXiv:2201.12872*, 2022.

Zheng Xiao, Hao Chen, Lu Xiao, Pengcheng Wei, Tao Wang, and Kenli Li. Wgdpool: A broad scope extraction for weighted graph data. *Expert Systems with Applications*, 249:123678, 2024.

Cheng Xin, Fan Xu, Xin Ding, Jie Gao, and Jiaxin Ding. Toping: Topologically interpretable graph learning via persistent rationale filtration. In *International Conference on Machine Learning*, 2025.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

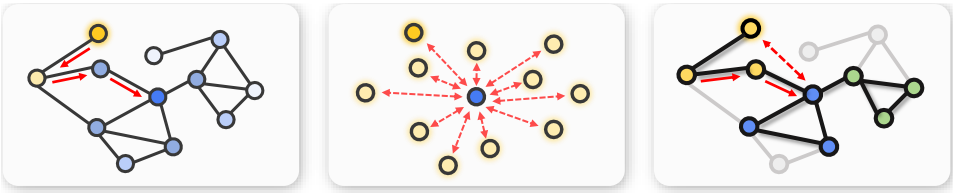
Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5): 5782–5799, 2022.

Kaizhong Zheng, Shujian Yu, and Badong Chen. Ci-gnn: A granger causality-inspired graph neural network for interpretable brain network-based psychiatric diagnosis. *Neural Networks*, 172:106147, 2024a.

Kaizhong Zheng, Shujian Yu, Baojuan Li, Robert Jenssen, and Badong Chen. Brainib: Interpretable brain network-based psychiatric diagnosis with graph information bottleneck. *IEEE Transactions on Neural Networks and Learning Systems*, 2024b.

A Additional Methodological Details

We first provide a conceptual comparison of different graph representation learning paradigms. As illustrated in Figure 6, conventional message-passing GNNs preserve local topology by propagating information along observed edges, but long-range interactions require repeated aggregation and may suffer from over-smoothing or over-squashing. In contrast, sequence-based graph models improve global dependency modeling by linearizing graph nodes, yet the induced sequence interactions may introduce dense virtual dependencies that are not explicitly constrained by the original topology. GEM is designed to combine the strengths of both paradigms: it preserves local graph structure, captures global dependencies, and further suppresses redundant connections while retaining meaningful community organization.



(a) GNN Models (b) Sequence methods (c) GEM (Ours)

Figure 6: Conceptual comparison of graph representation learning paradigms: (a) message-passing GNNs, (b) sequence-based models, and (c) GEM. GEM combines topology-aware local modeling with global dependency encoding while filtering redundant structures.

This conceptual distinction motivates the detailed design choices of GEM. The following sections provide notation summary, additional methodological details on state-space modeling and graph linearization, modularity-guided pooling, the CS-QMI estimator, and computational complexity.

A.1 Notation Summary

Table 4: Notation summary.

Symbol	Meaning
Graphs and representations	
$\mathcal{D} = \{(\mathcal{G}_i, y_i)\}_{i=1}^{M}$	Dataset of M labeled graphs.
$\mathcal{G} = (\mathbf{A}, \mathbf{X}, \mathbf{E})$, $\mathcal{G}_{\text{sub}}$	Original graph and distilled subgraph.
Y, y_i, C	Label variable, sample label, number of classes.
N, $\lvert\mathcal{E}\rvert$, m, d_i, $\mathbf{d}$	Node count, edge count, modularity edge term ($2m = \sum_i d_i$), node degree, degree vector.
d_n, d, L, d_z	Input dim, hidden dim, number of layers, graph-embedding dim.
$\mathbf{H}^{(l)}$, $\mathbf{h}_i^{(l)}$	Layer-l node-feature matrix and node feature.
Subgraph generator and dual-scale encoder	
f_{sub}, p_{ij}, w_{ij}, $\mathbf{W}$	Subgraph generator, edge keep probability, soft gate, gate matrix.
$\mathbf{m}_{ij}^{(l)}$, $\mathcal{N}(i)$	Message term and neighborhood set.
$\mathbf{H}_{\text{local}}^{(l)}$, $\mathbf{H}_{\text{global}}^{(l)}$	Local-branch and global-branch node features.
π, π^{-1}, r_i, $\mathbf{S}_{\pi}^{(l)}$	Node ordering, inverse ordering, ordering score, linearized sequence.
$\text{SSM}(\cdot)$, $\overrightarrow{\Theta}$, $\overleftarrow{\Theta}$	Mamba/SSM encoder and forward/backward parameters.
$\overrightarrow{\mathbf{O}}^{(l)}$, $\overleftarrow{\mathbf{O}}^{(l)}$, $\widetilde{\mathbf{H}}_{\text{global}}^{(l)}$	Forward output, backward output, pre-permutation global features.
SSM details	
$\mathbf{u}(t)$, $\mathbf{s}(t)$, $\mathbf{o}(t)$	Continuous-time SSM input, state, output.
$\mathbf{\Lambda}$, $\mathbf{B}^{\text{ssm}}$, $\mathbf{C}^{\text{ssm}}$, $\mathbf{D}$	SSM transition/input/output/skip matrices.
Δ_t, $\overline{\mathbf{\Lambda}}_t$, $\overline{\mathbf{B}}_t^{\text{ssm}}$	Discretization step and discrete SSM parameters.
Modularity-guided pooling and readout	
k, $\mathbf{C}$, $\mathbf{c}_c$, τ	Community count, centroids, centroid-c, softmax temperature.
$\mathbf{P}$, P_{ic}, $\mathbf{p}_i$, $\mathbf{Y}$, $\delta(c_i, c_j)$	Soft/hard assignments and same-community indicator.
$\mathbf{B}$, Q, Q_{soft}, $\mathcal{L}_{\text{mod}}$	Modularity matrix, hard/soft modularity, modularity loss.
$\mathbf{H}_{\text{cluster}}$, $\mathbf{R}$, $\mathbf{H}'$, $\mathbf{G}$	Cluster features, projection matrix, projected features, second-order Gram matrix.
$\mathbf{z}_{\mathcal{G}}$, $\mathbf{z}_{\text{sub}}$	Original-graph and subgraph embeddings.
Information bottleneck, CS-QMI, and losses	
Φ, f_{cls}, q_θ	Predictor, classifier, classifier distribution.
$I(\cdot;\cdot)$, $H(\cdot)$, D_{CS}, I_{CS}, $\widehat{I}_{\text{CS}}$	MI, entropy, CS divergence, CS-QMI, mini-batch estimator.
n_b, $\mathbf{K}$, $\mathbf{Q}$, κ_σ, σ, $\mathbf{1}$	Batch size (derivation), Gram matrices, Gaussian kernel, bandwidth, all-one vector.
$\mathcal{L}_{\text{IB}}$, $\mathcal{L}_{\text{CE}}$, $\mathcal{L}_{\text{mi}}$, $\mathcal{L}_{\text{total}}$, α, β, $\mathbb{B}$	IB loss, CE loss, compression loss, total loss, loss weights, stability constant.
Complexity symbols	
B, S	Batch size and Mamba state dimension (complexity section).

Table 4 summarizes the main symbols used throughout the paper.

A.2 Details of State Space Modeling and Graph Linearization

This section provides additional details on the global Mamba branch used in GEM. The purpose of this branch is not to replace graph topology with a pure sequence model, but to provide a complementary global view that captures long-range dependencies beyond local message passing. The local GNN branch remains topology-constrained, while the Mamba branch models global contextual interactions over a structurally ordered node sequence.

Continuous State Space Formulation. State Space Models (SSMs) represent a sequence by evolving a latent state through a linear dynamical system. Given an input signal $\mathbf{u}(t)$, the continuous-time SSM is written as

$$\frac{d\mathbf{s}(t)}{dt} = \mathbf{\Lambda}\mathbf{s}(t) + \mathbf{B}^{\text{ssm}}\mathbf{u}(t), \qquad \mathbf{o}(t) = \mathbf{C}^{\text{ssm}}\mathbf{s}(t) + \mathbf{D}\mathbf{u}(t), \tag{13}$$

where $\mathbf{s}(t)$ is the latent state, $\mathbf{u}(t)$ is the input, and $\mathbf{o}(t)$ is the output. We use $\mathbf{\Lambda}$ for the state transition matrix to avoid notational conflict with the graph adjacency matrix $\mathbf{A}$.

To apply Eq. (13) to discrete node sequences, the system is discretized with an input-dependent step size Δ_t. For the t-th sequence element, the discrete recurrence can be written as

$$\mathbf{s}_t = \overline{\mathbf{\Lambda}}_t \mathbf{s}_{t-1} + \overline{\mathbf{B}}_t^{\text{ssm}} \mathbf{u}_t, \qquad \mathbf{o}_t = \mathbf{C}_t^{\text{ssm}} \mathbf{s}_t + \mathbf{D}\mathbf{u}_t, \tag{14}$$

where

$$\overline{\mathbf{\Lambda}}_t = \exp(\Delta_t \mathbf{\Lambda}), \qquad \overline{\mathbf{B}}_t^{\text{ssm}} = \left(\int_0^{\Delta_t} \exp(s\mathbf{\Lambda})\, ds \right) \mathbf{B}_t^{\text{ssm}}. \tag{15}$$

In selective SSMs such as Mamba, Δ_t, $\mathbf{B}_t^{\text{ssm}}$, and $\mathbf{C}_t^{\text{ssm}}$ are input-dependent. This allows the model to selectively preserve or suppress information at each sequence position, which is useful for distinguishing salient structural signals from redundant global context.

Graph Linearization. Since SSMs operate on ordered sequences, GEM first maps the node representations of a graph into a sequence. Given the node representation matrix $\mathbf{H}^{(l-1)} \in \mathbb{R}^{N \times d}$ at layer $l-1$, we compute a structural score for each node. By default, we use degree centrality:

$$r_i = \sum_{j=1}^{N} A_{ij}, \qquad \pi = \text{argsort}\left(\{r_i\}_{i=1}^{N}; \text{descending}\right), \tag{16}$$

where $\pi(t)$ denotes the node placed at the t-th position of the sequence. The linearized node sequence is then

$$\mathbf{S}_{\pi}^{(l)} = \left[\mathbf{h}_{\pi(1)}^{(l-1)}, \mathbf{h}_{\pi(2)}^{(l-1)}, \ldots, \mathbf{h}_{\pi(N)}^{(l-1)}\right]. \tag{17}$$

Degree ordering provides a simple and efficient proxy for structural importance: high-degree nodes often serve as hubs or shortcut carriers in small-world graphs. Importantly, this ordering does not add edges to the graph. It only defines the order in which the global sequence encoder views the node representations.

Bidirectional Selective Scan. A single left-to-right scan may introduce directional bias because graph nodes do not have a natural sequential direction. To reduce this artifact, GEM applies Mamba in both forward and backward directions:

$$\overrightarrow{\mathbf{O}}^{(l)} = \text{SSM}\left(\mathbf{S}_{\pi}^{(l)}; \overrightarrow{\Theta}\right), \qquad \overleftarrow{\mathbf{O}}^{(l)} = \text{Rev}\left(\text{SSM}\left(\text{Rev}(\mathbf{S}_{\pi}^{(l)}); \overleftarrow{\Theta}\right)\right), \tag{18}$$

where $\text{Rev}(\cdot)$ reverses the sequence order, and $\overrightarrow{\Theta}$ and $\overleftarrow{\Theta}$ are the parameters of the forward and backward SSMs. The backward output is reversed back before fusion, so both $\overrightarrow{\mathbf{O}}^{(l)}$ and $\overleftarrow{\mathbf{O}}^{(l)}$ are aligned with the same sequence order.

The sequence-level global representation is then obtained by

$$\widetilde{\mathbf{H}}_{\text{global}}^{(l)} = \text{MLP}_{\text{bi}}\left(\left[\overrightarrow{\mathbf{O}}^{(l)}; \overleftarrow{\mathbf{O}}^{(l)}\right]\right), \tag{19}$$

where $[\cdot;\cdot]$ denotes feature concatenation.

Restoring Node Order and Fusion. Because $\widetilde{\mathbf{H}}^{(l)}_{\text{global}}$ is still arranged according to the sequence order π, we apply the inverse permutation π^{-1} to restore the original node order:

$$\mathbf{H}^{(l)}_{\text{global}} = \pi^{-1}\left(\widetilde{\mathbf{H}}^{(l)}_{\text{global}}\right), \qquad \left(\mathbf{H}^{(l)}_{\text{global}}\right)_{\pi(t)} = \left(\widetilde{\mathbf{H}}^{(l)}_{\text{global}}\right)_t. \tag{20}$$

This step is necessary before combining the global branch with the local message-passing branch, since the local branch operates in the original graph node order.

Finally, the local and global representations are fused through a residual update:

$$\mathbf{H}^{(l)} = \mathbf{H}^{(l-1)} + \text{MLP}_{\text{fuse}}\left(\mathbf{H}^{(l)}_{\text{local}} + \mathbf{H}^{(l)}_{\text{global}}\right). \tag{21}$$

Thus, GEM keeps the local branch explicitly tied to the observed topology, while the global branch supplies long-range context through an efficient sequence scan.

Computational Cost. The graph linearization step requires computing node degrees in $O(|\mathcal{E}|)$ time. Sorting can be implemented in $O(N \log N)$ with a generic sorting routine, and in $O(N + |\mathcal{E}|)$ using degree buckets or counting sort for integer degrees. The bidirectional Mamba scan is linear in the sequence length, giving $O(Nd_s)$ cost for fixed state dimension d_s. Therefore, the global branch avoids the $O(N^2)$ cost of full self-attention while providing graph-level long-range contextual modeling. A full module-wise complexity analysis is provided in Appendix A.5.

A.3 Derivation of Modularity-Guided Pooling

This section derives the modularity-guided pooling objective used in GEM. The goal is to make the learned soft clusters not only feature-aware, but also consistent with the meso-scale community structure of the input graph.

Classical Modularity. We first recall the Newman-Girvan modularity criterion. For an undirected graph with adjacency matrix $\mathbf{A}$, node degrees $d_i = \sum_j A_{ij}$, and $2m = \sum_i d_i$, the modularity of a hard partition $\{c_i\}_{i=1}^N$ is defined as

$$Q = \frac{1}{2m}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(A_{ij} - \frac{d_i d_j}{2m}\right)\delta(c_i, c_j), \tag{22}$$

where $\delta(c_i, c_j) = 1$ if nodes i and j belong to the same community, and 0 otherwise. The term A_{ij} measures the observed connectivity between two nodes, while $\frac{d_i d_j}{2m}$ is the expected connectivity under the configuration-model null graph. Thus, maximizing modularity encourages nodes to be grouped together when their observed connection is stronger than expected by chance.

Let the modularity matrix be

$$\mathbf{B} = \mathbf{A} - \frac{\mathbf{d}\mathbf{d}^\top}{2m}, \tag{23}$$

where $\mathbf{d} = [d_1, \ldots, d_N]^\top$. Then Eq. (22) can be written compactly as

$$Q = \frac{1}{2m}\sum_{i=1}^{N}\sum_{j=1}^{N} B_{ij}\delta(c_i, c_j). \tag{24}$$

Indicator-Matrix Form. Let $\mathbf{Y} \in \{0,1\}^{N\times k}$ denote the hard community assignment matrix, where $Y_{ic} = 1$ means that node i is assigned to community c. Each row of $\mathbf{Y}$ is one-hot. For two nodes i and j, the same-community indicator can be expressed as

$$\delta(c_i, c_j) = \sum_{c=1}^{k} Y_{ic}Y_{jc} = \mathbf{y}_i^\top \mathbf{y}_j, \tag{25}$$

where $\mathbf{y}_i$ is the i-th row of $\mathbf{Y}$. Substituting Eq. (25) into Eq. (24), we obtain

$$\begin{aligned} Q &= \frac{1}{2m} \sum_{i=1}^{N} \sum_{j=1}^{N} B_{ij} \sum_{c=1}^{k} Y_{ic} Y_{jc} \\ &= \frac{1}{2m} \sum_{c=1}^{k} \sum_{i=1}^{N} \sum_{j=1}^{N} Y_{ic} B_{ij} Y_{jc} \\ &= \frac{1}{2m} \operatorname{Tr}\left(\mathbf{Y}^\top \mathbf{B} \mathbf{Y}\right). \end{aligned} \tag{26}$$

Therefore, maximizing classical modularity is equivalent to maximizing the trace term $\operatorname{Tr}(\mathbf{Y}^\top \mathbf{B} \mathbf{Y})$ over discrete assignment matrices.

Differentiable Relaxation. Directly optimizing Eq. (26) is combinatorial because $\mathbf{Y}$ is discrete. GEM instead uses a differentiable soft assignment matrix $\mathbf{P} \in \mathbb{R}^{N \times k}$, where each row is a probability distribution over communities. Given node embeddings $\mathbf{H}^{(L)} \in \mathbb{R}^{N \times d}$ and learnable cluster centroids $\mathbf{C} \in \mathbb{R}^{k \times d}$, the assignment is computed as

$$P_{ic} = \frac{\exp(\mathbf{h}_i^\top \mathbf{c}_c / \tau)}{\sum_{r=1}^{k} \exp(\mathbf{h}_i^\top \mathbf{c}_r / \tau)}, \tag{27}$$

where τ is the temperature controlling assignment sharpness.

Under this relaxation, the hard same-community indicator $\delta(c_i, c_j)$ is replaced by the soft similarity between assignment distributions:

$$\delta(c_i, c_j) \quad \longrightarrow \quad \mathbf{p}_i^\top \mathbf{p}_j = \sum_{c=1}^{k} P_{ic} P_{jc}. \tag{28}$$

This yields the soft modularity objective

$$Q_{\text{soft}}(\mathbf{P}) = \frac{1}{2m} \operatorname{Tr}\left(\mathbf{P}^\top \mathbf{B} \mathbf{P}\right). \tag{29}$$

Since GEM is trained by minimizing losses, we define the modularity-guided pooling loss as

$$\mathcal{L}_{\text{mod}} = -Q_{\text{soft}}(\mathbf{P}) = -\frac{1}{2m} \operatorname{Tr}\left(\mathbf{P}^\top \mathbf{B} \mathbf{P}\right). \tag{30}$$

Minimizing $\mathcal{L}_{\text{mod}}$ encourages nodes with stronger-than-expected connectivity to receive similar community assignments, while discouraging grouping node pairs whose connectivity is weaker than expected under the null model.

Connection to Spectral Relaxation. The above relaxation is also related to spectral modularity maximization. If the discrete assignment matrix $\mathbf{Y}$ is relaxed to a continuous matrix under appropriate normalization constraints, maximizing $\operatorname{Tr}(\mathbf{Y}^\top \mathbf{B} \mathbf{Y})$ leads to the principal eigenspace of the modularity matrix $\mathbf{B}$. GEM does not perform explicit eigendecomposition. Instead, it parameterizes the relaxed assignment matrix through Eq. (27), allowing the community assignments to be learned jointly with the graph encoder in an end-to-end manner.

Use in Hierarchical Pooling. After obtaining the soft assignment matrix, GEM computes community-level representations by assignment-weighted aggregation:

$$\mathbf{H}_{\text{cluster}} = \mathbf{P}^\top \mathbf{H}^{(L)} \in \mathbb{R}^{k \times d}. \tag{31}$$

Equivalently, the representation of community c is

$$\mathbf{h}_c^{\text{cluster}} = \sum_{i=1}^{N} P_{ic} \mathbf{h}_i^{(L)}. \tag{32}$$

Thus, $\mathbf{P}$ serves two roles. First, it defines soft communities used for graph coarsening. Second, through $\mathcal{L}_{\text{mod}}$, it biases these communities toward topology-aware meso-scale structures.

The resulting community embeddings are then passed to the second-order statistical readout module, which captures high-order interactions among communities. In this way, modularity-guided pooling bridges node-level representations and graph-level readout while preserving meso-scale organization.

A.4 Derivation of the CS-QMI Estimator

This section derives the Information Bottleneck objective and the closed-form CS-QMI estimator used for structural distillation in GEM. Unlike regression-oriented CS-IB formulations, GEM is designed for graph classification. Therefore, we use the standard cross-entropy loss for predictive sufficiency and employ CS-QMI only for the structural compression term.

Information Bottleneck for Structural Distillation. Given an input graph $\mathcal{G}$ and its label Y, GEM learns a distilled subgraph $\mathcal{G}_{\mathrm{sub}} = f_{\mathrm{sub}}(\mathcal{G})$. The Information Bottleneck objective encourages $\mathcal{G}_{\mathrm{sub}}$ to be sufficient for predicting Y while discarding task-irrelevant information from $\mathcal{G}$:

$$\min_{\theta} \mathcal{L}_{\mathrm{IB}} = -I(\mathcal{G}_{\mathrm{sub}}; Y) + \beta I(\mathcal{G}; \mathcal{G}_{\mathrm{sub}}). \tag{33}$$

The first term enforces predictive sufficiency, and the second term enforces compression. Since $\mathcal{G}_{\mathrm{sub}}$ is produced by edge gates $\mathbf{W}$, this objective regularizes the learned graph structure rather than only the node features.

Sufficiency Term. For graph classification,

$$-I(\mathcal{G}_{\mathrm{sub}}; Y) = H(Y|\mathcal{G}_{\mathrm{sub}}) - H(Y), \tag{34}$$

where $H(Y)$ is independent of model parameters. Therefore, maximizing $I(\mathcal{G}_{\mathrm{sub}}; Y)$ is equivalent to minimizing the conditional entropy $H(Y|\mathcal{G}_{\mathrm{sub}})$. We use the cross-entropy loss as a tractable surrogate:

$$\mathcal{L}_{\mathrm{CE}} = \mathbb{E}_{(\mathcal{G}_{\mathrm{sub}}, Y)} \left[-\log q_\theta(Y|\mathcal{G}_{\mathrm{sub}}) \right], \tag{35}$$

where q_θ is the classifier. This term prevents trivial collapse: if the learned subgraph removes label-relevant structures, then $\mathcal{G}_{\mathrm{sub}}$ becomes less predictive and $\mathcal{L}_{\mathrm{CE}}$ increases.

Embedding-Level Compression Proxy. Directly estimating $I(\mathcal{G}; \mathcal{G}_{\mathrm{sub}})$ is intractable because graph topology is discrete and high-dimensional. GEM therefore estimates the dependence between their continuous graph-level embeddings. For each graph in a mini-batch, we compute

$$\mathbf{z}_{\mathcal{G}} = \mathrm{Readout}\left(\mathrm{Enc}(\mathcal{G})\right), \qquad \mathbf{z}_{\mathrm{sub}} = \mathrm{Readout}\left(\mathrm{Enc}(\mathcal{G}_{\mathrm{sub}})\right), \tag{36}$$

where the encoder and readout are shared across the original graph and the distilled subgraph. Since $\mathcal{G}_{\mathrm{sub}}$ is obtained through the differentiable edge mask $\mathbf{W}$, gradients from the compression term are back-propagated to the subgraph generator. Thus, reducing the dependence between $\mathbf{z}_{\mathcal{G}}$ and $\mathbf{z}_{\mathrm{sub}}$ encourages the generator to retain a compact, task-relevant structural rationale.

Cauchy-Schwarz Divergence. For two probability density functions p and q, we use the squared-ratio convention of Cauchy-Schwarz divergence:

$$D_{\mathrm{CS}}(p, q) = -\log \frac{\left(\int p(\mathbf{x}) q(\mathbf{x}) d\mathbf{x}\right)^2}{\int p^2(\mathbf{x}) d\mathbf{x} \int q^2(\mathbf{x}) d\mathbf{x}}. \tag{37}$$

Equivalently,

$$D_{\mathrm{CS}}(p, q) = \log \int p^2(\mathbf{x}) d\mathbf{x} + \log \int q^2(\mathbf{x}) d\mathbf{x} - 2 \log \int p(\mathbf{x}) q(\mathbf{x}) d\mathbf{x}. \tag{38}$$

This form avoids auxiliary neural critics and can be estimated directly from mini-batch samples using kernel Gram matrices.

CS-QMI Definition. Mutual dependence between two variables can be measured by comparing their joint distribution with the product of their marginals. For the original graph embedding $\mathbf{z}_{\mathcal{G}}$ and the subgraph embedding $\mathbf{z}_{\mathrm{sub}}$, CS-QMI is defined as

$$I_{\mathrm{CS}}(\mathbf{z}_{\mathcal{G}}; \mathbf{z}_{\mathrm{sub}}) = D_{\mathrm{CS}}\left(p(\mathbf{z}_{\mathcal{G}}, \mathbf{z}_{\mathrm{sub}}), p(\mathbf{z}_{\mathcal{G}}) p(\mathbf{z}_{\mathrm{sub}})\right). \tag{39}$$

Minimizing this quantity encourages the distilled subgraph representation to remove redundant information from the original graph representation, while the cross-entropy term keeps the subgraph label-predictive.

Mini-Batch Estimator. Let a mini-batch contain n_b graph pairs $\{(\mathbf{z}_{\mathcal{G}}^a, \mathbf{z}_{\text{sub}}^a)\}_{a=1}^{n_b}$. To avoid confusion with the modularity matrix $\mathbf{B}$, we use n_b to denote the batch size in this section. Define Gaussian kernel Gram matrices $\mathbf{K}, \mathbf{Q} \in \mathbb{R}^{n_b \times n_b}$ as

$$K_{ab} = \kappa_\sigma\left(\mathbf{z}_{\mathcal{G}}^a, \mathbf{z}_{\mathcal{G}}^b\right), \qquad Q_{ab} = \kappa_\sigma\left(\mathbf{z}_{\text{sub}}^a, \mathbf{z}_{\text{sub}}^b\right), \tag{40}$$

where

$$\kappa_\sigma(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|_2^2}{2\sigma^2}\right). \tag{41}$$

Following the KDE-based estimator of CS-QMI, the three terms in Eq. (38) are estimated as follows. First, the quadratic term of the joint distribution is

$$\int p^2(\mathbf{z}_{\mathcal{G}}, \mathbf{z}_{\text{sub}}) d\mathbf{z}_{\mathcal{G}} d\mathbf{z}_{\text{sub}} \approx \frac{1}{n_b^2} \sum_{a=1}^{n_b} \sum_{b=1}^{n_b} K_{ab} Q_{ab} = \frac{1}{n_b^2} \operatorname{Tr}(\mathbf{KQ}). \tag{42}$$

Second, the quadratic term of the product of marginals is

$$\int p^2(\mathbf{z}_{\mathcal{G}}) p^2(\mathbf{z}_{\text{sub}}) d\mathbf{z}_{\mathcal{G}} d\mathbf{z}_{\text{sub}} \approx \frac{1}{n_b^4} \left(\mathbf{1}^\top \mathbf{K} \mathbf{1}\right) \left(\mathbf{1}^\top \mathbf{Q} \mathbf{1}\right), \tag{43}$$

where $\mathbf{1} \in \mathbb{R}^{n_b}$ is an all-one vector. Third, the cross term between the joint distribution and the product of marginals is

$$\int p(\mathbf{z}_{\mathcal{G}}, \mathbf{z}_{\text{sub}}) p(\mathbf{z}_{\mathcal{G}}) p(\mathbf{z}_{\text{sub}}) d\mathbf{z}_{\mathcal{G}} d\mathbf{z}_{\text{sub}} \approx \frac{1}{n_b^3} \mathbf{1}^\top \mathbf{KQ} \mathbf{1}. \tag{44}$$

Substituting Eqs. (42)–(44) into Eq. (38) gives the closed-form mini-batch estimator:

$$\widehat{I}_{\text{CS}}(\mathbf{Z}_{\mathcal{G}}; \mathbf{Z}_{\text{sub}}) = \log\left(\frac{1}{n_b^2} \operatorname{Tr}(\mathbf{KQ})\right) + \log\left(\frac{1}{n_b^4} (\mathbf{1}^\top \mathbf{K} \mathbf{1})(\mathbf{1}^\top \mathbf{Q} \mathbf{1})\right) - 2\log\left(\frac{1}{n_b^3} \mathbf{1}^\top \mathbf{KQ} \mathbf{1}\right). \tag{45}$$

In implementation, a small constant ε is added inside each logarithm for numerical stability.

Structural Compression Loss. GEM uses the estimator in Eq. (45) as the structural compression loss:

$$\mathcal{L}_{\text{mi}} = \widehat{I}_{\text{CS}}(\mathbf{Z}_{\mathcal{G}}; \mathbf{Z}_{\text{sub}}). \tag{46}$$

The final objective is therefore

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \beta \mathcal{L}_{\text{mi}} + \alpha \mathcal{L}_{\text{mod}}. \tag{47}$$

The three terms play complementary roles: $\mathcal{L}_{\text{CE}}$ preserves label-predictive information, $\mathcal{L}_{\text{mi}}$ suppresses redundant dependence between the original graph and the distilled subgraph, and $\mathcal{L}_{\text{mod}}$ encourages the learned representation to preserve meso-scale community structure.

Computational Cost. The estimator only requires two $n_b \times n_b$ Gram matrices and matrix-vector products, where n_b denotes the mini-batch size. Computing the pairwise Gaussian kernels over graph embeddings requires $O(n_b^2 d_z)$ time, where d_z is the graph embedding dimension. The subsequent Gram-matrix operations require $O(n_b^2)$ time and are therefore dominated by the kernel computation. Thus, the CS-QMI estimator has complexity $O(n_b^2 d_z)$, corresponding to the $O(B^2 d)$ term in the main text when B denotes batch size and d denotes the embedding dimension. This cost is quadratic in batch size but independent of the number of nodes or edges in each graph. Consequently, for fixed or memory-bounded batch sizes, CS-QMI does not change GEM's graph-size scaling.

A.5 Complexity Analysis

Let N and E denote the number of nodes and edges in a graph, d the hidden dimension, B the batch size, and d_z the graph embedding dimension. The graph-size-dependent cost of GEM mainly comes from sparse message passing, subgraph generation, degree-based graph linearization, Mamba scanning, and modularity-aware pooling. The subgraph generator and local message-passing branch operate only on observed edges, leading to $O(Ed)$ complexity. The global branch first orders nodes by degree and then applies bidirectional Mamba scanning, whose sequence modeling cost is linear in

the number of nodes, i.e., $O(Nd)$ for fixed SSM dimension. For modularity-guided pooling, although the modularity matrix can be written in dense form, the objective can be computed using sparse adjacency and degree-vector correction, avoiding explicit $N \times N$ materialization. Therefore, for fixed hidden dimension and number of clusters, the main graph encoding and pooling pipeline scales approximately as $O((N + E)d)$.

The only non-graph-size-linear component is the CS-QMI estimator used for structural compression. Given a batch of B graph pairs, CS-QMI computes Gaussian Gram matrices over graph-level embeddings $\mathbf{Z}_{\mathcal{G}}$ and $\mathbf{Z}_{\mathrm{sub}}$, resulting in $O(B^2 d_z)$ complexity. This term is quadratic in batch size but independent of the number of nodes or edges in each graph. Thus, the overall training complexity can be summarized as

$$O\big((N + E)d + B^2 d_z\big). \tag{48}$$

For fixed or memory-bounded batch sizes, the CS-QMI term does not alter GEM's graph-size scaling, which explains the near-linear memory and runtime behavior observed in the scalability experiments.

B Experimental Setup and Robustness

B.1 Dataset Statistics and Preprocessing

Table 5 summarizes all datasets used in this work. We group them according to their primary experimental roles: main graph-level performance comparison, interpretability evaluation, scalability analysis, OGB throughput profiling, and long-range boundary analysis. The main comparison is conducted on seven real-world graph-level benchmarks, while the remaining datasets are used to evaluate complementary properties such as explanation quality, runtime scalability, dataset-level throughput, and transferability under long-range dependencies.

Table 5: Dataset statistics. The first group contains the main graph-level benchmarks used for performance comparison. Additional datasets are used for interpretability, scalability, throughput profiling, and long-range boundary analysis.

Dataset	**# Graphs**	**Avg. # Nodes**	**Avg. # Edges**	**# Classes/Tasks**	**Primary Usage**
Performance Benchmarks (Real-World)					
MUTAG	188	17.9	19.8	2	Performance
PROTEINS	1,113	39.1	72.8	2	Performance
NCI1	4,110	29.8	32.3	2	Performance
COLLAB	5,000	74.5	2,457.8	3	Performance
REDDIT-BINARY	2,000	429.6	497.8	2	Performance
ABIDE	1,099	116	1,334	2	Performance
SRPBS	184	116	1,334	2	Performance
Interpretability Benchmark (Synthetic)					
BA-2Motifs	1,000	30.0	52.5	2	Interpretability
Scalability Benchmark (Real-World)					
MalNet-Tiny	5,000	1,410.3	2,859.9	5	Efficiency
OGB Throughput Profiling Benchmarks					
ogbg-molhiv	41,127	25.5	27.5	1 task	Throughput
ogbg-molpcba	437,929	26.0	28.1	128 tasks	Throughput
ogbg-ppa	158,100	243.4	2,266.1	37	Throughput
Long-Range Boundary Analysis Benchmarks					
PascalVOC-SP	11,355	479.4	2,710.5	Node labels	Boundary analysis
Peptides-func	15,535	150.9	307.3	10 tasks	Boundary analysis

For the TUDataset benchmarks, we follow the standard graph classification setting. Unless otherwise specified, we use random 80%/10%/10% splits for training, validation, and testing, and report the

mean and standard deviation over 10 independent runs. For datasets without inherent node attributes, such as COLLAB and REDDIT-BINARY, we use one-hot encoded node degrees as node features. For datasets without edge attributes, we assign a constant edge feature of 1 to all observed edges.

For the brain network datasets ABIDE and SRPBS requiring further specific pre-processing, we followed a standard fMRI preprocessing pipeline. This pipeline included signal stabilization, slice-timing and motion correction, spatial normalization, smoothing, temporal filtering (0.01-0.1 Hz), and nuisance signal regression. A functional connectivity matrix was then generated for each subject by calculating the Pearson's correlation between AAL atlas-defined ROI time series to serve as the graph's adjacency matrix.

For MalNet-Tiny, Table 5 reports the statistics of the full dataset used in our study. In the scalability experiment, we further construct induced subgraph subsets with different subsampling ratios, resulting in average graph sizes ranging from 76 to 1,370 nodes. This controlled setting allows us to evaluate GPU memory usage and per-epoch training time as graph size increases.

For OGB graph benchmarks, we use ogbg-molhiv, ogbg-molpcba, and ogbg-ppa for throughput profiling only. We follow the official OGB data loading and preprocessing pipeline, but do not use these datasets to claim predictive superiority. This separation is intentional: the OGB experiments are designed to evaluate dataset-level throughput and the practical effect of the batch-level CS-QMI overhead, rather than downstream predictive performance.

For the LRGB benchmarks, we use PascalVOC-SP and Peptides-func as lightweight boundary analyses of GEM's transferability to long-range graph settings. PascalVOC-SP is a node prediction benchmark on superpixel graphs, whereas Peptides-func is a graph-level multi-label classification benchmark. We therefore use these results to characterize transferability under long-range dependencies rather than to claim task-specialized node-level superiority.

For BA-2Motifs, each graph consists of a Barabási–Albert base graph with an attached class-determining motif. We use two motif types, "house" and "cycle", and the graph label is determined solely by the attached motif. This dataset provides ground-truth rationale structures, making it suitable for evaluating explanation plausibility via AUC.

B.2 Implementation Details and Hyperparameter Search

Computational environment. All experiments are conducted on a Linux server running Ubuntu 20.04.2 LTS. The machine is equipped with an AMD EPYC 7542 32-Core CPU, 377GB RAM, and a single NVIDIA GeForce RTX 4090 GPU with 24GB VRAM. GEM is implemented in Python 3.10 with PyTorch 2.1.1 and PyTorch Geometric 2.6.1. The Mamba branch is implemented with `mamba-ssm` 2.2.2. Other key dependencies include `scikit-learn` 1.6.0, `einops` 0.8.0, NumPy 1.26.3, and NetworkX 3.2.1.

Table 6: Hardware and software configurations used in our experiments.

Component	Version / Specification
Hardware	
CPU	AMD EPYC 7542 32-Core
GPU	NVIDIA GeForce RTX 4090 24GB
RAM	377GB
Software	
OS	Ubuntu 20.04.2 LTS
Python	3.10
PyTorch	2.1.1+cu118
PyTorch Geometric	2.6.1
mamba-ssm	2.2.2
scikit-learn	1.6.0
einops	0.8.0
NumPy	1.26.3
NetworkX	3.2.1

Training protocol. For graph-level classification benchmarks, GEM is trained with AdamW. We report the mean and standard deviation over 10 independent runs using random 80%/10%/10% train/validation/test splits. The validation split is used for hyperparameter selection, and the test split is used only for final evaluation. For all baselines, we follow their original configurations when available and use the same split protocol for fair comparison. The core Mamba parameters are fixed across datasets following the standard Mamba configuration: `d_state`=16, `d_conv`=4, and `expand`=2.

Hyperparameter search. We tune GEM on the validation set using the grid in Table 7. The search covers both architectural hyperparameters and optimization hyperparameters, including the number of GEM layers, hidden dimension, number of clusters, assignment temperature, dropout rate, learning rate, weight decay, and the regularization weights α and β. Importantly, the final hyperparameters are selected separately for each dataset based on validation performance. The value 10^{-3} is used as a robust anchor in sensitivity analysis, but it is not treated as a universal optimum across all datasets. The final dataset-specific configurations are reported in Table 8. The Gaussian kernel bandwidth is determined adaptively within each mini-batch using the mean distance to the $k = 10$ nearest neighbors, excluding each sample itself, and is computed separately for the original-graph and subgraph embeddings.

Table 7: Hyperparameter search space for GEM.

Hyperparameter	**Search Space**
Model Architecture	
Hidden Dimension	$\{64, 96, 128, 256\}$
Number of GEM Layers	$\{2, 3, 4, 5\}$
Number of Clusters (k)	$\{4, 5, 6, 8, 10\}$
Clustering Temperature (τ)	$\{0.5, 1.0, 1.5, 2.0\}$
SOPOOL Output Dimension	$\{16, 32, 64\}$
MLP Layers (Readout)	$\{1, 2, 3\}$
MLP Hidden Dimension	$\{128, 256\}$
Dropout Rate	$\{0.1, 0.2, 0.3, 0.4, 0.5\}$
Training and Optimization	
Batch Size	$\{32, 64\}$
Learning Rate	$\{5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$
Weight Decay	$\{10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$
α ($\mathcal{L}_{\text{mod}}$)	$\{0.001, 0.002, 0.003, 0.005, 0.01\}$
β ($\mathcal{L}_{\text{mi}}$)	$\{0.001, 0.005, 0.01, 0.05\}$

B.3 Sensitivity Analysis

We conduct sensitivity analysis on three representative datasets: PROTEINS, REDDIT-BINARY, and ABIDE. These datasets cover different graph domains and structural scales, ranging from small bioinformatics graphs to larger social networks and dense brain connectivity graphs. We vary one hyperparameter at a time while keeping the remaining settings fixed to the final configurations in Table 8. The results are shown in Figure 7.

Sensitivity to the clustering temperature τ. The clustering temperature τ controls the sharpness of the soft community assignment matrix $\mathbf{P}$. As shown in the left panel of Figure 7, GEM performs best in an intermediate temperature range. When τ is too small, the assignments become overly sharp, which may lead to premature hard clustering and unstable optimization. When τ is too large, the assignments become overly smooth, weakening community boundaries and reducing the effectiveness of modularity-guided pooling. Empirically, PROTEINS and ABIDE achieve their strongest performance around $\tau = 1.0$, while REDDIT-BINARY peaks around $\tau = 1.5$. This indicates that GEM does not require an extremely small temperature; instead, moderate assignment sharpness is sufficient to preserve meso-scale community structure.

Table 8: Final hyperparameter configurations for GEM on each dataset.

Hyperparameter	**MUTAG**	**PROTEINS**	**NCI1**	**COLLAB**	**REDDIT-B**	**ABIDE**	**SRPBS**
Model Architecture							
Hidden Dimension	96	256	256	128	64	128	96
Number of GEM Layers	3	4	3	3	4	3	3
Number of Clusters (k)	4	6	5	6	6	5	5
MLP Layers (Readout)	2	3	3	3	3	2	2
MLP Hidden Dimension	128	256	256	128	256	128	256
Dropout Rate	0.4	0.3	0.2	0.2	0.4	0.5	0.1
Training and Optimization							
Batch Size	32	32	64	32	32	32	32
Learning Rate	0.001	0.005	0.005	0.005	0.001	0.005	0.005
Weight Decay	0.001	0.001	0.001	0.001	0.001	0.001	0.005
α ($\mathcal{L}_{\text{mod}}$)	0.003	0.001	0.001	0.002	0.001	0.001	0.001
β ($\mathcal{L}_{\text{mi}}$)	0.001	0.001	0.005	0.001	0.001	0.001	0.005

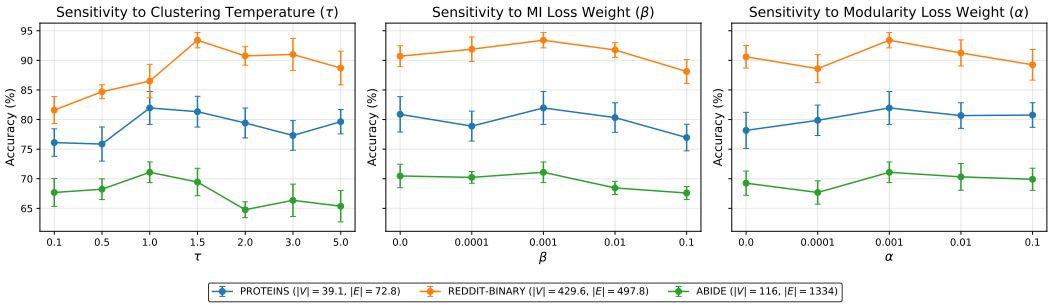


Figure 7: Sensitivity analysis of GEM with respect to the clustering temperature τ, the MI loss weight β, and the modularity loss weight α on PROTEINS, REDDIT-BINARY, and ABIDE. Error bars denote standard deviation over independent runs. GEM is generally stable around intermediate assignment temperatures and small regularization weights, while overly sharp/soft assignments or overly strong compression can degrade performance.

Sensitivity to the MI loss weight β. The hyperparameter β controls the strength of the CS-QMI compression loss $\mathcal{L}_{\text{mi}}$. As shown in the middle panel of Figure 7, all three datasets achieve their best or near-best performance when β is in a small regularization regime, with $\beta = 10^{-3}$ giving the strongest performance among the tested values. Compared with $\beta = 0$, introducing a nonzero compression term improves performance, confirming that IB-based structural distillation helps remove redundant interactions from the dual-scale representation. However, when β becomes too large, especially at $\beta = 0.1$, performance drops across all datasets. This behavior is consistent with the Information Bottleneck trade-off: moderate compression encourages compact and task-relevant subgraphs, whereas overly strong compression can remove label-relevant structures.

Sensitivity to the modularity loss weight α. The hyperparameter α controls the contribution of the modularity-guided pooling loss $\mathcal{L}_{\text{mod}}$. As shown in the right panel of Figure 7, performance generally improves when moving from no modularity regularization or very weak regularization to $\alpha = 10^{-3}$. This supports the role of $\mathcal{L}_{\text{mod}}$ in encouraging topology-aware community assignments. The performance remains relatively stable around small regularization values, while overly large α can slightly degrade performance by over-constraining the learned communities toward topology alone and reducing task adaptivity. These results are consistent with the ablation study in Table 2, where disabling $\mathcal{L}_{\text{mod}}$ reduces performance on all three representative datasets.

Summary. Overall, the sensitivity results show that GEM is not dependent on a narrowly tuned hyperparameter setting. The best performance is obtained with moderate assignment temperature and small regularization weights, which is consistent across datasets with different graph sizes and densities. The trends also support the complementary roles of the two regularizers: $\mathcal{L}_{\text{mi}}$ controls

Table 9: Ordering robustness analysis. We report mean accuracy (%) with standard deviation over 10 runs. Degree ordering is used as the default because of its low computational cost.

Ordering Strategy	PROTEINS	REDDIT-B	ABIDE
Degree	$81.96_{\pm 2.78}$	$93.40_{\pm 1.29}$	$71.09_{\pm 1.75}$
Betweenness	$81.62_{\pm 4.83}$	$91.40_{\pm 3.71}$	$71.35_{\pm 2.04}$
Random	$78.81_{\pm 2.75}$	$89.25_{\pm 3.01}$	$68.61_{\pm 2.32}$

redundant structural information through compression, while $\mathcal{L}_{\mathrm{mod}}$ preserves meso-scale community organization during graph-level readout.

B.4 Ordering Robustness Analysis

The global Mamba branch requires a one-dimensional node sequence as input. In the main experiments, we use degree-based ordering as the default graph linearization strategy because it is simple, deterministic, and computationally efficient. However, a natural concern is whether GEM's performance depends critically on this specific ordering. To examine this, we compare three ordering strategies: degree centrality, betweenness centrality, and random permutation. The results are reported in Table 9.

The results show that structural orderings lead to comparable performance. Degree and betweenness centrality achieve similar accuracy on PROTEINS and ABIDE, while degree ordering performs better on REDDIT-BINARY. This suggests that GEM does not rely on a single handcrafted serialization rule, as long as the ordering preserves meaningful structural information.

Random ordering leads to a larger performance drop, especially on REDDIT-BINARY, indicating that structural information in the ordering is beneficial for global sequence modeling. Nevertheless, the random-ordering variant still performs above the "w/o Global Branch" ablation in Table 2 on all three datasets. This indicates that the global Mamba branch continues to provide useful contextual information even under a non-structural ordering, rather than becoming harmful or entirely ineffective.

We use degree ordering as the default strategy because it provides a favorable trade-off between effectiveness and efficiency. Degree centrality can be computed in linear time with respect to the number of edges, whereas betweenness centrality is substantially more expensive on large graphs. Combined with the bidirectional scan in the global branch and the local message-passing branch, degree-based ordering offers a lightweight and robust graph linearization strategy for GEM.

C Additional Empirical Results

C.1 OGB Throughput Profiling

We further evaluate dataset-level throughput on three OGB graph benchmarks: ogbg-molhiv, ogbg-molpcba, and ogbg-ppa. These experiments are designed for throughput profiling rather than downstream predictive comparison. All models are evaluated under matched hardware settings on a single NVIDIA RTX 4090 GPU, and we report training time per epoch under different batch sizes. This experiment complements the graph-size scalability analysis on MalNet-Tiny by examining whether the batch-level CS-QMI term becomes a practical bottleneck on large-scale benchmark datasets.

As shown in Table 10, GEM maintains competitive per-epoch throughput across the three OGB datasets. On ogbg-molhiv, GEM achieves comparable throughput to GraphGPS and Graph-Mamba at large batch sizes. On ogbg-molpcba, GEM is faster than GraphGPS at $B = 512$ and remains within the same throughput regime as Graph-Mamba. On ogbg-ppa, which contains substantially larger graphs, GEM is able to run at $B = 256$, whereas Graph-Mamba and GraphGPS encounter OOM at this batch size.

These results also suggest that the $O(B^2 d_z)$ CS-QMI term does not dominate practical training time in the tested range. If this batch-level term dominated the per-step cost, increasing the batch size would tend to offset the reduction in the number of training steps, since the number of steps decreases approximately linearly with B while the CS-QMI cost per step grows quadratically with

Table 10: OGB throughput profiling. We report training time per epoch in seconds (s/epoch) under different batch sizes on a single NVIDIA RTX 4090 GPU. OOM denotes out-of-memory failure.

Dataset	Model	Time (s/epoch) ↓			
		B=32	B=128	B=256	B=512
ogbg-molhiv	GEM	89.7	27.1	14.6	9.4
	Graph-Mamba	109.3	40.4	36.8	8.9
	GraphGPS	89.8	23.7	13.5	9.6
ogbg-molpcba	GEM	602.6	255.1	223.9	121.6
	Graph-Mamba	740.8	413.6	120.8	99.3
	GraphGPS	652.9	285.4	215.4	207.1
ogbg-ppa	GEM	245.5	187.5	164.6	OOM
	Graph-Mamba	176.4	890.7	OOM	OOM
	GraphGPS	602.1	191.3	OOM	OOM

B. Empirically, GEM's per-epoch time generally decreases as the batch size increases, indicating that graph encoding and data loading remain the dominant costs under these settings. Therefore, the batch-level CS-QMI overhead does not negate GEM's practical scalability in memory-bounded training regimes.

C.2 Lightweight Long-Range Boundary Analysis

GEM is primarily designed for graph-level hierarchical representation learning, where modularity-aware pooling and graph-level readout are central to the architecture. To better characterize its transferability under long-range graph settings, we further conduct a lightweight boundary analysis on two LRGB benchmarks: PascalVOC-SP and Peptides-func. PascalVOC-SP is a node prediction benchmark on superpixel graphs, while Peptides-func is a graph-level multi-label classification benchmark. Therefore, these experiments are intended to examine whether GEM's hierarchical representations remain useful beyond the main graph classification benchmarks, rather than to claim universal superiority across all graph learning tasks.

For PascalVOC-SP, we adapt GEM to node-level prediction by bypassing the final graph-level readout and augmenting each node representation with its soft community context derived from the assignment matrix $\mathbf{P}$. This keeps the local branch, global Mamba branch, and meso-scale community assignment active, while avoiding extensive task-specific architectural redesign. For Peptides-func, we use GEM in the graph-level setting and evaluate its long-range graph representation ability under the LRGB protocol. The results are reported in Table 11.

Table 11: Lightweight long-range boundary analysis on LRGB benchmarks. PascalVOC-SP is evaluated by F1 score, and Peptides-func is evaluated by average precision (AP). Higher is better for both metrics.

Model	PascalVOC-SP F1 ↑	Peptides-func AP ↑
GCN	$0.1268_{\pm 0.0060}$	$0.5930_{\pm 0.0023}$
GPS	$0.3689_{\pm 0.0131}$	$0.6575_{\pm 0.0049}$
Exphormer	$0.3975_{\pm 0.0037}$	$0.6527_{\pm 0.0043}$
Graph-Mamba	$0.4180_{\pm 0.0120}$	$0.6624_{\pm 0.0079}$
GMN	$0.4192_{\pm 0.0108}$	$0.6972_{\pm 0.0100}$
GEM	$0.4196_{\pm 0.0152}$	$0.6899_{\pm 0.0098}$

As shown in Table 11, GEM achieves competitive results on both LRGB benchmarks without extensive task-specific redesign. On PascalVOC-SP, GEM obtains the best mean F1 score among the compared methods, slightly outperforming Graph-Mamba and GMN. This suggests that the combination of local message passing, global sequence modeling, and soft community context can provide

useful node-level representations under long-range superpixel settings. On Peptides-func, GEM achieves an AP of 0.6899, outperforming GPS, Exphormer, and Graph-Mamba, and approaching the strongest baseline GMN. These results indicate that GEM's dual-scale and community-aware representations transfer meaningfully to long-range graph settings.

Overall, this boundary analysis supports the transferability of GEM's hierarchical representation design while preserving our main positioning: GEM is primarily developed for graph-level representation learning, and broader task-specialized extensions remain future work.

C.3 Additional Qualitative Interpretability Visualization

We provide additional qualitative visualizations as an extension of the example shown in the main text. While the main paper presents a representative comparison between PGExplainer and GEM on a BA-2Motifs graph with a "house" motif, here we extend the comparison to more explanation baselines and an additional motif type. Specifically, we compare GNNExplainer, PGExplainer, DIR-GNN, CI-GNN, and GEM on BA-2Motifs graphs containing either a "house" motif or a "cycle" motif. The results are shown in Figure 8 and Figure 9.

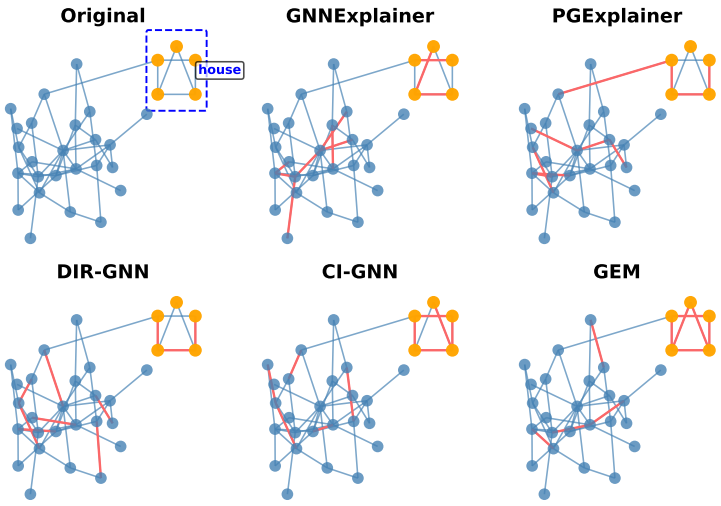


Figure 8: Extended qualitative comparison on a BA-2Motifs graph containing a ground-truth "house" motif. Compared with GNNExplainer, PGExplainer, DIR-GNN, and CI-GNN, GEM more clearly recovers the complete class-determining house motif and produces a structurally coherent rationale.

These visualizations complement the quantitative interpretability results in Table 3. Across both motif types, post-hoc methods such as GNNExplainer and PGExplainer often produce fragmented explanations or include background edges from the BA base graph. Intrinsic baselines such as DIR-GNN and CI-GNN provide more compact rationales, but they may still miss parts of the class-determining motif in these examples. In contrast, GEM better preserves the full motif structure for both the house and cycle cases, yielding explanations that are more complete and structurally coherent.

This behavior is consistent with the role of IB-guided structural distillation. The learned edge mask suppresses task-irrelevant background connections while preserving compact subgraphs that support the graph-level prediction. Therefore, these additional visualizations provide qualitative evidence that GEM can produce intrinsic, task-relevant subgraph rationales, complementing the quantitative plausibility and fidelity results.

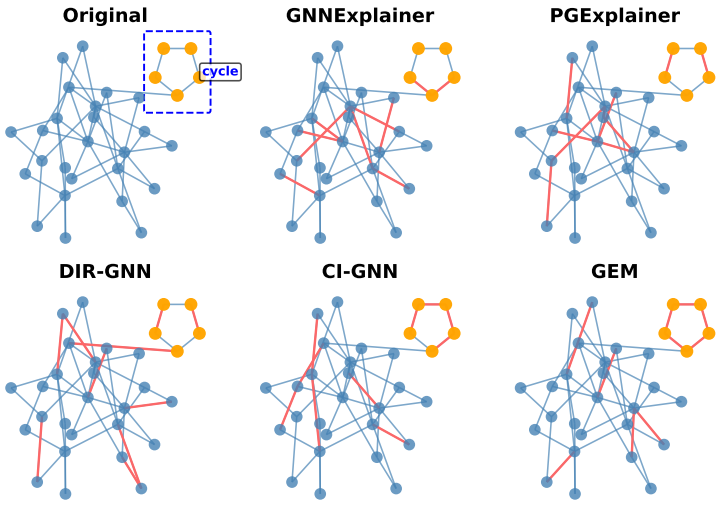


Figure 9: Extended qualitative comparison on a BA-2Motifs graph containing a ground-truth "cycle" motif. GEM recovers the complete cycle motif more coherently, whereas the compared baselines tend to miss motif edges or select additional background edges from the BA base graph.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

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 Section 1 explicitly position GEM as a graph-level hierarchical representation learning framework and state the corresponding architectural, interpretability, and efficiency claims. These claims are supported by the graph-level experiments in Section 4, with boundary transferability analysis and additional robustness results provided in Appendix C.

Guidelines:

- The answer [N/A] 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 [N/A] 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: Section 5 discusses the main limitations of GEM, including its primary focus on graph-level representation learning and its reliance on explicit structural ordering for the global Mamba branch. The paper also identifies task-specialized extensions and adaptive ordering strategies as future directions.

Guidelines:

- The answer [N/A] 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 derivations for the modularity-guided pooling objective and the closed-form CS-QMI estimator in Appendix A.3 and Appendix A.4. The assumptions and notation used in these derivations are summarized in Appendix A.1, and the computational complexity is analyzed in Appendix A.5.

Guidelines:

- The answer [N/A] 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 specifies the model architecture, training objective, datasets, baselines, evaluation metrics, split protocol, and hyperparameter search procedure in Sections 4 and Appendix B.2.

Guidelines:

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

5. **Open access to data and code**

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

   Answer: [Yes]

   Justification: All datasets used in the experiments are publicly available benchmarks cited in the paper. Dataset preprocessing, hyperparameter search spaces, final configurations, and implementation details are described in Appendix B.1 and Appendix B.2.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 describes the datasets, baselines, optimizer, evaluation metrics, and random split protocol. Appendix B.2 further provides the computational environment, training protocol, hyperparameter search space, and final dataset-specific hyperparameters.

   Guidelines:

   - The answer [N/A] 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: The main graph classification results report mean accuracy with standard deviation over 10 independent runs, and the interpretability results report mean and standard deviation over 5 runs. The same reporting convention is used for ablation, robustness, and additional boundary-analysis experiments.

   Guidelines:

   - The answer [N/A] 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: Appendix B.2 reports the hardware and software environment, including CPU, GPU, memory, operating system, Python, PyTorch, PyTorch Geometric, and Mamba versions. Section 4.4 and Appendix C.1 provide memory and wall-clock runtime measurements for scalability and throughput experiments.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research uses publicly available benchmark datasets and does not involve deceptive data collection, unsafe deployment, or direct interaction with human participants.

   Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: The work may have positive societal impact by improving efficient and interpretable graph representation learning for scientific domains such as biological, social, and brain networks. Potential negative impacts could arise if graph predictions or explanations are deployed in high-stakes biomedical or social-network settings without adequate validation, privacy protection, or fairness assessment; the paper therefore presents GEM as a research method rather than a deployed decision system.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk generative models, large pre-trained foundation models, scraped datasets, or systems with obvious dual-use misuse risks. The released asset is an implementation of a graph learning algorithm for benchmark research.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: The paper credits the original creators of the datasets, baselines, and software packages through citations, including TUDataset, ABIDE, SRPBS, MalNet-Tiny, OGB, LRGB, PyTorch Geometric, and Mamba-related baselines. The experiments use these assets as publicly available research benchmarks and do not redistribute restricted raw data.

Guidelines:

- The answer [N/A] 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 main new asset introduced by this work is the GEM implementation.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The paper does not involve crowdsourcing, user studies, or newly collected human-subject data. Experiments use existing public benchmark datasets and do not require participant instructions or compensation.

Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The authors do not conduct new human-subject experiments or collect new participant data. Human-related neuroimaging benchmarks such as ABIDE and SRPBS are used as existing anonymized public datasets according to their benchmark protocols.

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used only for writing polishing, editing, formatting, and discussion assistance in this work.

    Guidelines:

    - The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.