Explainable Graph Representation Learning via Graph Pattern Analysis

Anonymous authors
Paper under double-blind review

Abstract

Explainable artificial intelligence (XAI) is an important area in the AI community, and interpretability is crucial for building robust and trustworthy AI models. While previous work has explored model-level and instance-level explainable graph learning, there has been limited investigation into explainable graph representation learning. In this paper, we focus on representation-level explainable graph learning and answer a fundamental question: *What specific information about a graph is captured in graph representations?* Our approach is inspired by graph kernels, which evaluate graph similarities by counting substructures within specific graph patterns. First, we present an unsupervised ensemble graph kernel method for representation or similarity explanation, which however has limitations such as ignoring node features and being computationally expensive. To address these limitations, we introduce a deep learning framework for learning and explaining graph representations through graph pattern analysis. We start by sampling graph substructures of various patterns. Then, we learn the representations of these patterns and combine them using a weighted sum, where the weights indicate the importance of each graph pattern's contribution. Note that our method can be both unsupervised and supervised and is a one-shot explanation, not specified to single samples or predictions. We also theoretically analyze the robustness and generalization ability of our models. Importantly, the generalization analysis shows that incorporating multiple graph patterns lowers the generalization error bound. In our experiments, we show how to learn and explain graph representations for real-world data using pattern analysis. Additionally, we compare our method against multiple baselines in both supervised and unsupervised learning tasks to demonstrate its superiority in terms of accuracy.

1 Introduction

The field of explainable artificial intelligence (XAI) (Došilović et al., 2018; Adadi & Berrada, 2018; Angelov et al., 2021; Hassija et al., 2024) is gaining significant attention in both AI and science communities. Interpretability is crucial for creating robust and trustworthy AI models, especially in critical domains like transportation, healthcare, law, and finance. Graph learning is an important area of AI that particularly focuses on graph-structured data widely exist in social science, biology, chemistry, etc. Explainable graph learning (XGL) (Kosan et al., 2023) can be generally classified into two categories: model-level methods and instance-level methods.

Model-level methods of XGL provide transparency by analyzing the model behavior. Examples include XGNN (Yuan et al., 2020), GLG-Explainer (Azzolin et al., 2022), and GCFExplainer (Huang et al., 2023). Instance-level methods of XGL offer explanations tailored to specific predictions, focusing on why particular instances are classified in a certain manner. For instance, GNNExplainer (Ying et al., 2019) identifies a compact subgraph structure crucial for a GNN's prediction. PGExplainer (Luo et al., 2020) trains a graph generator to incorporate global information and parameterize the explanation generation process. AutoGR (Wang et al., 2021) introduces an explainable AutoML approach for graph representation learning. MotifExplainer (Yu & Gao, 2022) identifies critical motifs (small subgraphs) in a graph. UNR-Explainer (Kang et al., 2024) identifies the top-k most important nodes in a graph to determine the most significant subgraph as the counterfactual explanation. More about XGL can be found in the Appendix C.1.

However, these works mainly focus on enhancing the transparency of GNN models or identifying the most important substructures that contribute to predictions. The exploration of representation-level explainable graph learning (XGL) is limited. We propose explainable graph representation learning and ask a fundamental question: **What specific information about a graph is captured in graph representations?** Formally, if we represent a graph G as a d-dimensional vector $\boldsymbol{g}$, our goal is to understand what specific information about the graph G is embedded in the representation $\boldsymbol{g}$. This problem is important and has practical applications. Some graph patterns are highly practical and crucial in various real-world tasks, and we want this information to be captured in representations. For instance, in molecular chemistry, bonds between atoms or functional groups often form cycles (rings), which indicate a molecule's properties and can be used to generate molecular fingerprints (Morgan, 1965; Alon et al., 2008; Rahman et al., 2009; O'Boyle & Sayle, 2016). Similarly, cliques characterize protein complexes in Protein-Protein Interaction networks and help identify community structures in social networks (Girvan & Newman, 2002; Jiang et al., 2010; Fox et al., 2020).

Although some previous works such as (Kosan et al., 2023) aimed to find the most critical subgraph S by solving optimization problems based on perturbation-based reasoning, either factual or counterfactual, this kind of approach assumes that the most important subgraph S mainly contributes to the representation $\boldsymbol{g}$, neglecting other aspects of the graph, which doesn't align well with our goal of thoroughly understanding graph representations. Analyzing all subgraphs of a graph G is impractical due to their vast number. To address the challenge, we propose to group the subgraphs into different graph patterns, like paths, trees, cycles, cliques, etc, and then analyze the contribution of each graph pattern to the graph representation $\boldsymbol{g}$.

Our idea of pattern analysis is inspired by graph kernels, which compare substructures of specific graph patterns to evaluate the similarity between two graphs (Kriege et al., 2020). For example, random walk kernels (Borgwardt et al., 2005; Gärtner et al., 2003) use path patterns, sub-tree kernels (Da San Martino et al., 2012; Smola & Vishwanathan, 2002) examine tree patterns, and graphlet kernels (Pržulj, 2007; Shervashidze et al., 2009) focus on graphlet patterns. The graph kernel involves learning a pattern counting representation vector $\boldsymbol{h}$, which counts the occurrences of substructures of a specific pattern within the graph G. While the pattern counting vector $\boldsymbol{h}$ is an explainable representation, it has some limitations, such as the high dimensionality and ignorance of node features.

There also exist some representation methods based on subgraphs and substructures, such as Subgraph Neural Networks (SubGNN) (Kriege & Mutzel, 2012), Substructure Assembling Network (SAN) (Zhao et al., 2018), Substructure Aware Graph Neural Networks (SAGNN) (Zeng et al., 2023a), and Mutual Information (MI) Induced Substructure-aware GRL (Wang et al., 2020). However, these methods mainly focus on increasing expressiveness and do not provide explainability for representation learning. We will discuss the details in the Appendix C.2.

In this work, we propose a novel framework to learn and explain graph representations via graph pattern analysis. We start by sampling graph substructures of various patterns. Then, we learn the representations of these patterns and combine them adaptively, where the weights indicate the importance of each graph pattern's contribution. We also provide theoretical analyses of our methods, including robustness and generalization. Additionally, we compare our method against multiple baselines in both supervised and unsupervised learning tasks to demonstrate its effectiveness and superiority. Our contributions are summarized as follows:

- Unlike previous model-level and instance-level XGL, we introduce a new problem — representation-level explainable graph learning. This problem focuses on understanding what specific information about a graph is embedded within its representations in unsupervised learning.

- We propose two strategies to learn and explain graph representations, including a graph ensemble kernel method and a pattern analysis GNN method. The latter involves using GNNs to learn the representations of each pattern and evaluate its contribution to the ensemble graph representation.

- We provide robust analyses and generalization analysis for our methods theoretically. Particularly, our generalization analysis shows adding graph patterns lowers the generalization error bound.

2 Notations

In this work, we use x, $\boldsymbol{x}$, $\boldsymbol{X}$, and $\mathcal{X}$ (or X) to denote scalar, vector, matrix, and set, respectively. We denote $[n] = \{1, 2, ..., n\}$. Let $G = (V, E)$ be a graph with n nodes and d-dimensional node features $\{\boldsymbol{x}_v \in \mathbb{R}^d \mid v \in V\}$. We denote $\boldsymbol{A} \in \{0,1\}^{n \times n}$ the adjacency matrix and $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^\top \in \mathbb{R}^{n \times d}$ the node features matrix. Let $\mathcal{G} = \{G_1, \ldots, G_N\}$ be a dataset of N graphs belonging C classes, where $G_i = (V_i, E_i)$. For G_i, we denote its number of nodes as n_i, the one-hot graph label as $\boldsymbol{y}_i \in \{0,1\}^C$, the graph-level representation as a vector $\boldsymbol{g}_i \in \mathbb{R}^d$, the adjacency matrix as $\boldsymbol{A}_i$, and the node feature matrix as $\boldsymbol{X}_i$. Let $S = (V_S, E_S)$ be a subgraph of graph $G = (V, E)$ such that $V_S \subseteq V$ and $E_S \subseteq E$. The the adjacency matrix of S is denoted as $\boldsymbol{A}_S \in \{0,1\}^{|V_S| \times |V_S|}$ and the node feature matrix of S is sampled from the rows of $\boldsymbol{X}$, denoted as $\boldsymbol{X}_S \in \mathbb{R}^{|V_S| \times d}$.

The graph pattern is defined as a set of all graphs that share certain properties, denoted as $\mathcal{P} = \{P_1, P_2, \ldots, P_i, \ldots\}$, where P_i is the i-th example of this pattern. In this work, the graph patterns are basic graph families such as paths, trees, cycles, cliques, etc. Detailed mathematical definitions for some of these patterns are provided in Appendix B. For example:

- $\mathcal{P}_{\text{path}} = \{\text{ph}_1, \text{ph}_2, \ldots, \text{ph}_i, \ldots\}$ is a path pattern with ph_i as a path of length i.
- $\mathcal{P}_T = \{T_1, T_2, \ldots, T_i, \ldots\}$ is a tree pattern where T_i is the i-th tree.
- $\mathcal{P}_{\text{gl}} = \{\text{gl}_1, \text{gl}_2, \ldots, \text{gl}_i, \ldots\}$ is a graphlet pattern where gl_i is the i-th graphlet.

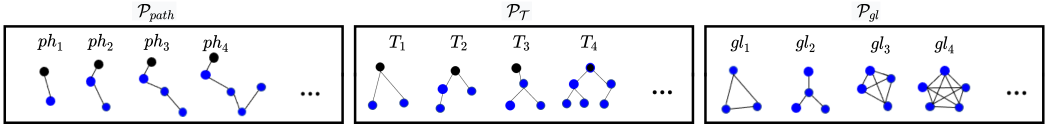


Figure 1: Examples of graph patterns: $\mathcal{P}_{\text{path}}$, $\mathcal{P}_{\text{T}}$ and $\mathcal{P}_{\text{gl}}$

Figure 1 illustrates some intuitive examples of graph patterns. Notably, there are overlaps among different patterns; for instance, the graph $T_3 \in \mathcal{P}_T$ and $\text{gl}_2 \in \mathcal{P}_{\text{gl}}$ are identical, being both a tree and a graphlet. Overlaps are inevitable due to the predefined nature of these basic graph families in graph theory. We denote a set of M different patterns as $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m, \ldots, \mathcal{P}_M\}$. Given the pattern $\mathcal{P}_m$ and the graph G_i, the pattern sampling set is denoted as $\mathcal{S}_i^{(m)}$ and the pattern representation is denoted as $\boldsymbol{z}_i^{(m)} \in \mathbb{R}^d$.

3 Learning Explainable Graph Representations via Ensemble Graph Kernel

In this section, we learn and explain the pattern counting graph representation via graph kernels.

3.1 Pattern Counting Kernel

A graph kernel $K : \mathbb{G} \times \mathbb{G} \to \mathbb{R}$ aims to evaluate the similarity between two graphs. Let G_i and G_j be two graphs in the graph dataset $\mathcal{G}$ and let $\mathcal{H}$ be a high-dimensional vector space. The key to a graph kernel is defining a mapping from the graph space to the high-dimensional vector space as $\phi : \mathbb{G} \to \mathcal{H}$, where $\boldsymbol{h}_i = \phi(G_i)$ and $\boldsymbol{h}_j = \phi(G_j)$. Then, the graph kernel can be defined as the inner product of $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, i.e., $K(G_i, G_j) := \boldsymbol{h}_i^\top \boldsymbol{h}_j$. The most widely used mapping ϕ is the one counting the occurrences of each example in the pattern $\mathcal{P}$ within graph G. The corresponding pattern counting vector is defined as follows.

Definition 3.1 (Pattern Counting Vector)**.** Given a graph G and a pattern $\mathcal{P} = \{P_1, P_2, \ldots, P_i, \ldots\}$, a pattern counting mapping $\phi : \mathbb{G} \to \mathcal{H}$ is defined as

$$\boldsymbol{h} = \phi(G; \mathcal{P}), \text{ with } \boldsymbol{h} = [h^{(1)}, h^{(2)}, \ldots, h^{(i)}, \ldots], \tag{1}$$

where $h^{(i)}$ is the number of occurrences of pattern example P_i as a substructure within graph G. We call $\boldsymbol{h}$ a pattern counting vector of G related to pattern $\mathcal{P}$.

Then the pattern counting kernel $K_{\mathcal{P}} : \mathbb{G} \times \mathbb{G} \to \mathbb{R}$ based on pattern $\mathcal{P}$ can be defined.

Definition 3.2 (Pattern Counting Kernel)**.** Given the a pattern counting mapping $\phi(G; \mathcal{P})$, a pattern counting kernel is defined as

$$K_{\mathcal{P}}(G_i, G_j) := \langle \phi(G_i; \mathcal{P}), \phi(G_j; \mathcal{P}) \rangle = \boldsymbol{h}_i^\top \boldsymbol{h}_j \tag{2}$$

The pattern counting kernel $K_{\mathcal{P}}$ is uniquely determined by the pattern $\mathcal{P}$. For example, if $\mathcal{P}$ is selected as the path pattern $\mathcal{P}_{\text{path}}$, we obtain a random walk kernel (Borgwardt et al., 2005; Gärtner et al., 2003). If $\mathcal{P}$ is the tree pattern $\mathcal{P}_T$, we get a sub-tree kernel (Da San Martino et al., 2012; Smola & Vishwanathan, 2002). Similarly, if $\mathcal{P}$ is the graphlet pattern $\mathcal{P}_{\text{gl}}$, we derive a graphlet kernel (Pržulj, 2007).

3.2 Pattern Analysis using Graph Kernels

Let $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_M\}$ be a set of M different graph patterns. For instance, $\mathcal{P}_1$ represents the path pattern and $\mathcal{P}_2$ represents the tree pattern. Then, we can define a set of M different graph kernels as $\{K_{\mathcal{P}_1}, K_{\mathcal{P}_2}, \ldots, K_{\mathcal{P}_M}\}$. Since the pattern counting kernel $K_{\mathcal{P}_m}$ is uniquely determined by the pattern $\mathcal{P}_m$, we can analyze the importance of pattern $\mathcal{P}_m$ by evaluating the importance of its pattern counting kernel $K_{\mathcal{P}_m}$. To achieve this, we define a learnable ensemble kernel as follows:

Definition 3.3 (Learnable Ensemble Kernel)**.** Let $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_m, ..., \lambda_M]^\top$ be a positive weight parameter vector. The ensemble kernel matrix $\boldsymbol{K}(\boldsymbol{\lambda}) \in \mathbb{R}^{|\mathcal{G}| \times |\mathcal{G}|}$ is defined as the weighted sum of M different kernels $\{K_{\mathcal{P}_1}, K_{\mathcal{P}_2}, \ldots, K_{\mathcal{P}_M}\}$. Given two graphs G_i and G_j in $\mathcal{G}$, the element at the i-th row and j-th column of $\boldsymbol{K}(\boldsymbol{\lambda})$ is given by

$$K_{ij}(\boldsymbol{\lambda}) := \sum_{m=1}^{M} \lambda_m \, K_{\mathcal{P}_m}(G_i, G_j), \;\; \text{s.t} \sum_{m=1}^{M} \lambda_m = 1, \;\; \text{and} \;\; \lambda_m \geq 0, \;\; \forall m \in [M]. \tag{3}$$

Here, the weight parameter λ_m indicates the importance of the kernel $K_{\mathcal{P}_m}$ as well as the corresponding graph pattern $\mathcal{P}_m$ within the dataset $\mathcal{G}$. Instead of the constrained optimization (3), we may consider replacing λ_m with $\exp(w_m) / \sum_{m=1}^{M} \exp(w_m)$ such that the constraints are satisfied inherently, which leads to an unconstrained optimization in terms of $\boldsymbol{w} = [w_1, \ldots, w_M]^\top$. In the following context, for convenience, we just focus on (3), though all results are applicable to the unconstrained optimization. To obtain the weight parameter $\boldsymbol{\lambda}$, we provide the supervised and unsupervised loss functions as follows.

Supervised Contrastive Loss Following (Oord et al., 2018), given a kernel matrix $\boldsymbol{K}(\boldsymbol{\lambda}) \in \mathbb{R}^{N \times N}$, we define the supervised InfoNEC as follows

$$\mathcal{L}_{\text{SCL}}(\boldsymbol{\lambda}) = -\sum_{i \neq j} \mathbb{I}_{[\boldsymbol{y}_i = \boldsymbol{y}_j]} \left(\log K_{ij}(\boldsymbol{\lambda}) - \log \left[\sum_{k} \mathbb{I}_{[\boldsymbol{y}_i = \boldsymbol{y}_k, i \neq k]} K_{ik}(\boldsymbol{\lambda}) + \mu \sum_{k} \mathbb{I}_{[\boldsymbol{y}_i \neq \boldsymbol{y}_k]} K_{ik}(\boldsymbol{\lambda}) \right] \right), \tag{4}$$

where $\mathbb{I}_{[\cdot]}$ is an indicator function and $\mu > 0$ is a hyperparameter.

Unsupervised KL Divergence Inspired by (Xie et al., 2016), given a kernel matrix $\boldsymbol{K} \in \mathbb{R}^{N \times N}$, we define the unsupervised KL divergence loss as follows

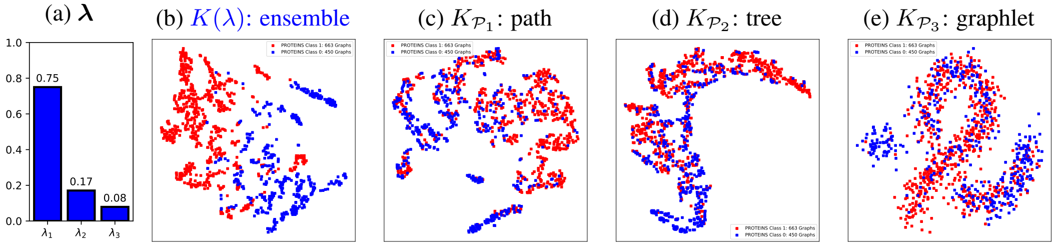
$$\mathcal{L}_{\text{KL}}(\boldsymbol{\lambda}) = \mathbb{KL}(\boldsymbol{K}(\boldsymbol{\lambda}), \boldsymbol{K}'(\boldsymbol{\lambda})), \;\; \text{with} \;\; \boldsymbol{K}'_{ij}(\boldsymbol{\lambda}) = \frac{K_{ij}^2(\boldsymbol{\lambda}) / r_j}{\sum_{j'} K_{ij'}^2(\boldsymbol{\lambda}) / r_{j'}} \;\; \text{and} \;\; r_j = \sum_{j} K_{ij}(\boldsymbol{\lambda}), \tag{5}$$

where r_j are soft cluster frequencies. By minimizing the KL divergence, the model adjusts the parameters $\boldsymbol{\lambda}$ to more accurately represent the natural clustering property of the dataset.

We use the $\mathcal{L}_{\text{SCL}}$ or $\mathcal{L}_{\text{KL}}$ as our loss function, i.e., $\mathcal{L}_{\text{ker}}(\boldsymbol{\lambda}) = \mathcal{L}_{\text{SCL}}(\boldsymbol{K}(\boldsymbol{\lambda}))$ or $\mathcal{L}_{\text{KL}}(\boldsymbol{K}(\boldsymbol{\lambda}))$, when the graphs are labeled or unlabeled. Then the weight parameter $\boldsymbol{\lambda}$ can be obtain by solving

$$\boldsymbol{\lambda}^* = \text{argmin}_{\,\mathbf{1}_M^\top \boldsymbol{\lambda} = 1, \; \boldsymbol{\lambda} \geq 0} \;\; \mathcal{L}_{\text{ker}}(\boldsymbol{\lambda}), \tag{6}$$

where $\boldsymbol{\lambda}^* = [\lambda_1^*, ..., \lambda_m^*, ...\lambda_M^*]^\top$ and λ_m^* indicates the importance of kernel $K_{\mathcal{P}_m}$ as well as pattern $\mathcal{P}_m$. In Figure 2, we can see that the ensemble Kernel performs better than each single kernel and the pattern analysis identifies the importance of each kernel as well as the related graph pattern. We call this method pattern-based XGL with ensemble graph kernel, abbreviated as **PXGL-EGK**. This method not only yields explainable similarity learning but also provides an approach to selecting graph kernels and their hyperparameters automatically if we consider different kernel types with different hyperparameters.



$\sum_{m=1}^{M} \lambda_m K_{\mathcal{P}_m}$ ($\lambda_1 = 0.7502$) ($\lambda_2 = 0.1707$) ($\lambda_3 = 0.07912$)

Figure 2: t-SNE visualizations of different kernel embeddings for the dataset PROTEINS.

3.3 Limitations of Pattern Counting Vector

The pattern counting vector $\boldsymbol{h}$ from Definition 3.1 is easy to understand and its importance can be evaluated using the weight parameter $\boldsymbol{\lambda}^*$ from (6). However, it cannot directly explain the representation of graph G due to the following limitations, which are also the limitations of the proposed PXGL-EGK.

- **Ignoring Node Features:** $\boldsymbol{h}$ captures the topology of G but ignores node features $\boldsymbol{X}$. As shown by previous GNN works, node features are crucial for learning graph representations.
- **High Dimensionality:** The pattern set $\mathcal{P} = \{P_1, P_2, \ldots, P_i, \ldots\}$ can be vast, making $\boldsymbol{h}$ high-dimensional and impractical for many tasks.
- **High Computational Complexity:** Counting patterns P_i in G is time-consuming due to the large number of patterns in $\mathcal{P}$. The function $\phi(G; \mathcal{P})$ needs to be run for each new graph. In addition, in PXGL-EGK, the computation of the M kernel matrices of size $|\mathcal{G}| \times |\mathcal{G}|$ is very expensive especially when $|\mathcal{G}|$ is large.
- **Lacking Implicit Information and Strong Expressiveness:** $\boldsymbol{h}$ is fixed and not learnable. GNN (Kipf & Welling, 2016) shows that message passing can learn implicit information and provide better representations, which should be considered if possible.

4 Learning Explainable Graph Representations via GNNs

In this section, we address the limitations pointed out in Section 3.3 by proposing a GNN framework to learn and explain graph representations via pattern analysis. We first present the definitions of the pattern sample set, pattern representation, and ensemble representation and then show the objective functions of unsupervised and supervised learning.

Definition 4.1 (Pattern Sample Set). A $\mathcal{P}$-pattern sample set $\mathcal{S}$ of a given graph G is defined as

$$\mathcal{S} := \{S_1, S_2, \ldots, S_q, \ldots, S_Q\}, \tag{7}$$

where S_q, $q \in [Q]$, is a subgraph of pattern $\mathcal{P}$ (see the examples in Figure 1) randomly sampled from G using some sampling function Φ[1].

Definition 4.2 (Pattern Representation). Let $\mathcal{S}$ be a $\mathcal{P}$-pattern sample set of a graph G. For each subgraph $S \in \mathcal{S}$, denote its node set, adjacency matrix, and node feature matrix as V_S, $\boldsymbol{A}_S$, and $\boldsymbol{X}_S$ respectively. Let $F : \{0,1\}^{|V_S| \times |V_S|} \times \mathbb{R}^{|V_S| \times d} \to \mathbb{R}^{d'}$ be a pattern representation learning function parameterized by $\mathcal{W}$, then the $\mathcal{P}$-pattern representation $\boldsymbol{z} \in \mathbb{R}^{d'}$ of G is defined as

$$\boldsymbol{z} = \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} F(\boldsymbol{A}_S, \boldsymbol{X}_S; \mathcal{W}). \tag{8}$$

[1] The specific Φ follows `https://ysig.github.io/GraKeL/0.1a8/`

The pattern representation learning function F could be any graph neural network such as GCN (Kipf & Welling, 2016), GIN (Xu et al., 2018), and graph transformer (Rampášek et al., 2022). In this paper, we use GCN only for convenience. Because of the presence of node features, the chance that overlaps occur between patterns is tiny. Nevertheless, we can use the WL-test (Huang & Villar, 2021) in each sampling phase to ensure that new samples are unique from existing ones, which is efficient as the subgraphs are small.

Finally, the ensemble representation $\boldsymbol{g}$ is a weighted sum of the M pattern representations as follows.

Definition 4.3 (Ensemble Representation)**.** Given a graph G and consider a set of M different patterns $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m, \ldots, \mathcal{P}_M\}$, we denote $\boldsymbol{z}^{(m)}$ the $\mathcal{P}_m$-pattern representation obtained from the $\mathcal{P}_m$-pattern set $\mathcal{S}^{(m)}$ using a pattern representation learning function F_m. Let $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_m, \ldots, \lambda_M]^\top$ be a parameter vector, where $\mathbf{1}_M^\top \boldsymbol{\lambda} = 1$ and $\lambda_m \geq 0 \ \forall \ m \in [M]$. Then the ensemble representation $\boldsymbol{g} \in \mathbb{R}^{d'}$ of G is defined as

$$\boldsymbol{g} = \sum_{m=1}^{M} \lambda_m \boldsymbol{z}^{(m)}, \quad \text{with} \quad \boldsymbol{z}^{(m)} = \frac{1}{|\mathcal{S}^{(m)}|} \sum_{S \in \mathcal{S}^{(m)}} F_m(\boldsymbol{A}_S, \boldsymbol{X}_S; \mathcal{W}^{(m)}), \quad \forall \, m \in [M]. \tag{9}$$

Note that instead of explicitly considering the constraints for $\boldsymbol{\lambda}$, we can use the same softmax trick in computing the ensemble kernel (3) to simplify the problem.

Let $\mathbb{W} := \{\mathcal{W}^{(1)}, \mathcal{W}^{(2)}, \ldots, \mathcal{W}^{(m)}, \ldots, \mathcal{W}^{(M)}\}$ be the parameters of the M GNNs. In unsupervised representation learning, we define the similarity between two graphs' ensemble representations as $K_{ij}(\boldsymbol{\lambda}, \mathbb{W}) = \exp\left(-\gamma \|\boldsymbol{g}_i - \boldsymbol{g}_j\|^2\right)$, where $\gamma > 0$ is a hyperparameter. Then similar to (5), we minimize the following objective function to optimize $\mathbb{W}$

$$\mathcal{L}_{\text{KL}}(\boldsymbol{\lambda}, \mathbb{W}) = \mathbb{KL}(\boldsymbol{K}(\boldsymbol{\lambda}, \mathbb{W}), \boldsymbol{K}'(\boldsymbol{\lambda}, \mathbb{W})) \tag{10}$$

where the computation of $\boldsymbol{K}'$ is the same as that in (5).

In supervised learning, given a graph $G \in \mathcal{G}$ with ensemble representation $\boldsymbol{g}$, denote $\boldsymbol{y} \in \{0, 1\}^C$ the ground truth label. Let $\hat{\boldsymbol{y}} \in [0, 1]^C$ be the predicted label given by a softmax classifier $f_c : \mathbb{R}^d \to \mathbb{R}^C$ parameterized by $\boldsymbol{W}_C$, i.e., $\hat{\boldsymbol{y}} = f_c(\boldsymbol{g})$. Let ℓ_{CE} be the multi-class cross-entropy loss, i.e., $\ell_{\text{CE}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{c=1}^{C} y_c \log \hat{y}_c$. Then we minimize the following objective to optimize the parameters $\bar{\mathbb{W}} = \{\mathbb{W}, \boldsymbol{W}_C\}$:

$$\mathcal{L}_{\text{CE}}(\boldsymbol{\lambda}, \bar{\mathbb{W}}) = \frac{1}{N} \sum_{i=1}^{N} \ell_{\text{CE}}\left(\boldsymbol{y}_i, f_c(\boldsymbol{g}_i)\right) \tag{11}$$

Let $\boldsymbol{\lambda}^* = [\lambda_1^*, \ldots, \lambda_m^*, \ldots, \lambda_M^*]^\top$ be the optimal $\boldsymbol{\lambda}$ obtained from minimizing (10) or (11). λ_m^* indicates the contribution of the pattern representation $\boldsymbol{z}^{(m)}$ to the ensemble graph representation $\boldsymbol{g}$. In Figure 4, we visualize the $\boldsymbol{g}$ and each $\boldsymbol{z}^{(m)}$ and show that the ensemble representation $\boldsymbol{g}$ performs the best and the λ_m^* explains the contribution of each pattern representation $\boldsymbol{z}^{(m)}$ to learning $\boldsymbol{g}$. For convenience, we call this method pattern-based XGL with GNNs, abbreviated as **PXGL-GNN**.

5 Theoretical Analysis

In this section, we analyze the robustness property, generalization ability, and computational complexity of our methods theoretically, which not only is important to understand the proposed methods but also provides theoretical support for the effectiveness of the proposed methods. We defer the detailed proof to Appendices D and E.

5.1 Robustness Analysis

Following (O'Bray et al., 2021), a learning method should be robust to small perturbations. Let Δ_A and Δ_X be perturbations on the adjacency matrix and node attributes of a graph G whose representation is denoted as $\boldsymbol{g}$. Then the perturbed graph is $\tilde{G} = (\boldsymbol{A} + \Delta_A, \boldsymbol{X} + \Delta_X)$, of which the representation is denoted as $\tilde{\boldsymbol{g}}$. We seek the upper bound of $\|\tilde{\boldsymbol{g}} - \boldsymbol{g}\|$ and want to know how it is related to Δ_A and Δ_X as well as the representation learning function F. Without loss of

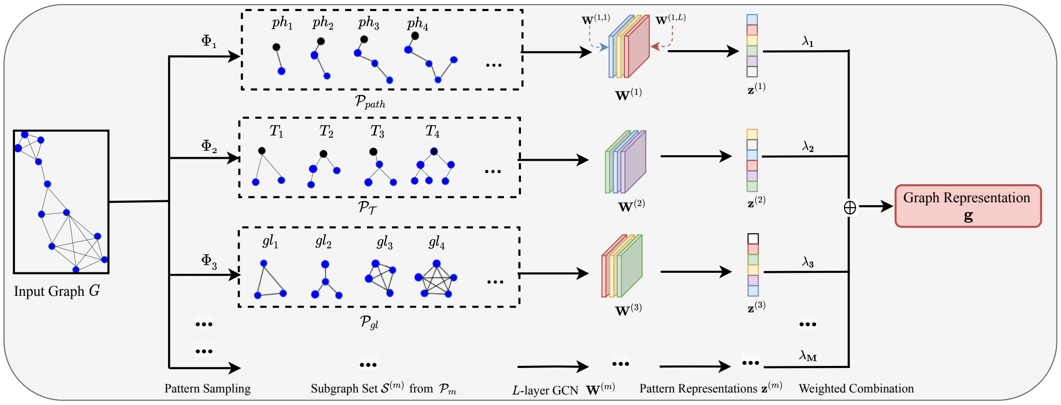


Figure 3: Proposed GNN framework for computing the ensemble graph representation

generality, we assume that G has n nodes, F is an L-layer GCN (Kipf & Welling, 2016), and all the activation functions are $\sigma(\cdot)$. For each pattern $\mathcal{P}_m$, the parameter set of $F(\boldsymbol{A}, \boldsymbol{X}; \mathcal{W}^{(m)})$ are $\mathcal{W}^{(m)} = \{\boldsymbol{W}^{(m,1)}, \ldots, \boldsymbol{W}^{(m,L)}\}$, where $\boldsymbol{W}^{(m,l)}$ denotes the parameter matrix in the l-th layer. We further assume that for each pattern $\mathcal{P}_m$, the output vector representation is obtained by the average pooling. Then we have the following theorem.

Theorem 5.1. *Let $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \Delta_A$ and $\tilde{\boldsymbol{X}} = \boldsymbol{X} + \Delta_X$. Suppose $\|\boldsymbol{A}\|_2 \leq \beta_A$, $\|\boldsymbol{X}\|_F \leq \beta_X$, $\|\boldsymbol{W}^{(m,l)}\|_2 \leq \beta_W$ for all $m \in [M]$ and $l \in [L]$, and $\sigma(\cdot)$ is ρ-Lipschitz continuous. Let α be the minimum node degree of G, and $\Delta_D := \boldsymbol{I} - diag(\mathbf{1}^\top(\boldsymbol{I} + \boldsymbol{A} + \Delta_A))^{\frac{1}{2}} diag(\mathbf{1}^\top \boldsymbol{A})^{-\frac{1}{2}}$. Let $\bar{\beta}_A = 1 + \beta_A$. Then the representation robustness of PXGL-GNN to perturbations Δ_A and Δ_X is shown as*

$$\|\tilde{\boldsymbol{g}} - \boldsymbol{g}\| \leq \frac{1}{\sqrt{n}} \rho^L \beta_W^L (\bar{\beta}_A + \|\Delta_A\|_2)^{L-1} (1+\alpha)^{-L} \left[(\bar{\beta}_A + 2\|\Delta_A\|_2) \|\Delta_X\|_F + 2L\beta_X \bar{\beta}_A \|\Delta_D\|_2 \right]$$

The bound reveals that PXGL-GNN is sensitive to the graph structure perturbation Δ_A when L is large and is relatively not sensitive to the feature matrix perturbation on Δ_X. On the other hand, when α, the minimum node degree, is larger, the method is more robust.

5.2 Generalization Analysis

Following (Bousquet & Elisseeff, 2002; Feldman & Vondrak, 2019), we use uniform stability to derive the generalization bound for PXGL-GNN. Let $\boldsymbol{\lambda}$ and $\mathbb{W}$ be known parameters. The supervised loss ℓ_{CE} in (11) is guaranteed with a uniform stability parameter η. For convenience, we let $\ell(\boldsymbol{\lambda}, \bar{\mathbb{W}}; G) := \ell_{\text{CE}}(\boldsymbol{y}, \hat{\boldsymbol{y}})$. Considering the empirical risk $\mathcal{E}[\ell(\boldsymbol{\lambda}, \bar{\mathbb{W}}; \mathcal{G})] := \frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{\lambda}, \bar{\mathbb{W}}; G_i)$ and true risk $\mathbb{E}[\ell(\boldsymbol{\lambda}, \bar{\mathbb{W}}; G)]$, we have the following high-probability generalization bound: for constant c and $\delta \in (0, 1)$,

$$\mathbf{Pr}\left[\left|\mathbb{E}[\ell_{\text{CE}}(\boldsymbol{\lambda}, \bar{\mathbb{W}}; G) - \mathcal{E}[\ell_{\text{CE}}(\boldsymbol{\lambda}, \bar{\mathbb{W}}; \mathcal{G})]\right| \geq c \left(\eta \log(N) \log\left(\frac{N}{\delta}\right) + \sqrt{\frac{\log(1/\delta)}{N}} \right) \right] \leq \delta. \quad (12)$$

Let $\mathcal{D} := \{G_1, \ldots, G_N\}$ be the training data. By removing the i-th graph G_i, we get $\mathcal{D}^{\backslash i} = \{G_1, \ldots, G_{i-1}, G_{i+1}, \ldots, G_N\}$. Let $\boldsymbol{\lambda}_{\mathcal{D}}$ and $\bar{\mathbb{W}}_{\mathcal{D}} := \{\boldsymbol{W}_C, \boldsymbol{W}_{\mathcal{D}}^{(m,l)}, \forall m \in [M], l \in [L]\}$ be the parameters trained on $\mathcal{D}$. Let $\boldsymbol{\lambda}_{\mathcal{D}^{\backslash i}}$ and $\bar{\mathbb{W}}_{\mathcal{D}^{\backslash i}} := \{\boldsymbol{W}_{C^{\backslash i}}, \boldsymbol{W}_{\mathcal{D}^{\backslash i}}^{(m,l)}, \forall m \in [M], l \in [L]\}$ be the parameters trained on $\mathcal{D}^{\backslash i}$. We aim to find an η such that

$$|\ell_{\text{CE}}(\boldsymbol{\lambda}_{\mathcal{D}}, \bar{\mathbb{W}}_{\mathcal{D}}; G) - \ell_{\text{CE}}(\boldsymbol{\lambda}_{\mathcal{D}^{\backslash i}}, \bar{\mathbb{W}}_{\mathcal{D}^{\backslash i}}; G)| \leq \eta \quad (13)$$

We have the following result for η.

Theorem 5.2. *Suppose $\max\{\max_{m\in[M],l\in[L]} \|\boldsymbol{W}_{\mathcal{D}}^{(m,l)}\|_2, \max_{m\in[M],l\in[L]} \|\boldsymbol{W}_{\mathcal{D}^{\backslash i}}^{(m,l)}\|_2\} \leq \hat{\beta}_W$ and $\max_{m\in[M],l\in[L]} \|\boldsymbol{W}_{\mathcal{D}}^{(m,l)} - \boldsymbol{W}_{\mathcal{D}^{\backslash i}}^{(m,l)}\|_2 \leq \hat{\beta}_{\Delta W}$, $\|\boldsymbol{W}_C - \boldsymbol{W}_{C^{\backslash i}}\|_2 \leq \gamma_{\Delta C}$, $\|\boldsymbol{W}_{C^{\backslash i}}\|_2 \leq \gamma_C$.*

Suppose the f_c in ℓ_{CE} (11) is a linear classifier, which is τ-Lipschitz continuous. Suppose Thus the η for estimation error (12) and uniform stability (13) is:

$$\eta = \frac{\tau}{\sqrt{n}} \rho^L \hat{\beta}_W^{L-1} \beta_X (1+\beta_A)^L (1+\alpha)^{-L} \left[\hat{\beta}_W \gamma_{\Delta C} + \gamma_C \left(2\hat{\beta}_W + L\hat{\beta}_{\Delta W} \right) \right] \tag{14}$$

Invoking (14) into (12), we obtain the generalization error bound of our model. We see that when α is larger and β_A, β_X are smaller, the generalization ability is stronger. It is worth noting that in the proof (see (35)) of the theorem, we used an aggressive relaxation such that $\boldsymbol{\lambda}$ was not present in η. By keeping $\boldsymbol{\lambda}$, we can obtain

$$\eta = \frac{\tau}{\sqrt{n}} \rho^L \hat{\beta}_W^{L-1} \beta_X (1+\beta_A)^L (1+\alpha)^{-L} \left[\hat{\beta}_W \gamma_{\Delta C} + \gamma_C \left(\hat{\beta}_W \left\| \boldsymbol{\lambda}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}} \right\| + L\hat{\beta}_{\Delta W} \left\| \boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}} \right\| \right) \right] \tag{15}$$

Since $\|\boldsymbol{\lambda}_{\mathcal{D}}\|_1 = \|\boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\|_1 = 1$, when M is larger, $\|\boldsymbol{\lambda}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\|$ and $\|\boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\|$ are potentially smaller. This means that when we include more graph patterns, the generalization bound of our PXGL-GNN becomes tighter, which potentially leads to higher classification accuracy.

5.3 Time and Space Complexity

Given a dataset with N graphs (each has n nodes and e edges), we select M different patterns and sample Q subgraphs of each pattern. First, our PXGL-EGK requires computing M kernel matrices, of which the space complexity is $\mathcal{O}(MN^2)$ and the time complexity is $\mathcal{O}(N^2 \sum_{m=1}^{M} \psi_i)$, where ψ_i denotes the time complexity of the m-th graph kernel. For instance, the time complexities of the graphlet kernel, shortest path kernel, and Weisfeiler-Lehman Subtree kernel are $\mathcal{O}(n^k)$, $\mathcal{O}(n^4)$, and $\mathcal{O}(hn + he)$ respectively, where k and h are some kernel-specific hyperparameters. When N is large, the method has high time and space complexities. Regarding PXGL-GNN, suppose each representation learning function F_m is an L-layer GCN, of which the width is linear with d. For both supervised and unsupervised learning, suppose the batch size and the number of iterations in the optimization are B and T respectively. Then, in supervised learning, the space complexity and time complexity are $\mathcal{O}(BMQ(e + nd) + MLd^2 + Cd)$ and $\mathcal{O}(TBMQL(ed + nd^2) + NQ \sum_{m=1}^{M} \vartheta_m)$ respectively, where ϑ_m denotes the time complexity of generating a sample of the m-th pattern. For instance, when the m-th pattern is graphlets with size $k \in \{3, 4, 5\}$, we have $\vartheta_m \leq nu^{k-1}$ (Shervashidze et al., 2009), where u denotes the maximum node degree of the graph. In unsupervised learning, the space complexity and time complexity are $\mathcal{O}(BMQ(e+nd) + MLd^2 + Cd + B^2)$ and $\mathcal{O}(TBMQL(ed + nd^2) + TB^2 + NQ \sum_{m=1}^{M} \vartheta_m)$ respectively. PXGL-GNN is scalable to large graph datasets because the complexities are linear with BMQ and B^2 and ϑ_m are controllable.

6 Related Works

Due to space limitation, we introduce previous works on explainable graph learning (XGL), graph representation learning (GCL), and graph kernels in Appendix C.

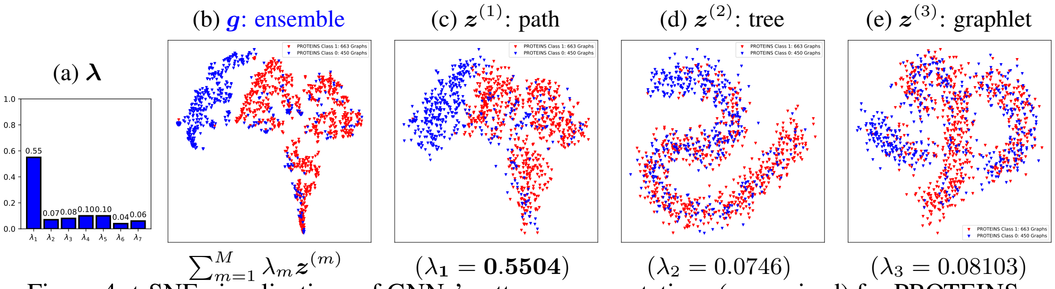
7 Experiments

We test our method on the TUdataset (Morris et al., 2020) for both supervised and unsupervised learning tasks, as shown in Table 1. Our goal is to learn explainable graph representations. We provide the weight parameter $\boldsymbol{\lambda}$ and visualize the ensemble representation $\boldsymbol{g}$ and the pattern representation $\boldsymbol{z}^{(m)}$. We use seven graph patterns: paths, trees, graphlets, cycles, cliques, wheels, and stars, sampling $Q = 50$ subgraphs for each. We use a 5-layer GCN for the representation learning function F and a 3-layer DNN with softmax for classification function f_c. We repeat the experiments ten times

Table 1: Statistics of Datasets

Name	# of graphs	# of classes	# of nodes	node labels	node attributes
MUTAG	188	2	17.9	yes	no
PROTEINS	1113	2	39.1	yes	yes
DD	1178	2	284.32	yes	no
NCI1	4110	2	29.9	yes	no
COLLAB	5000	3	74.49	no	no
IMDB-B	1000	2	19.8	no	no
REDDIT-B	2000	2	429.63	no	no
REDDIT-M5K	4999	5	508.52	no	no

and report the average value with standard deviation. Due to the space limitation, the results of PXGL-EGK and other figures are shown in Appendix F.



(a) $\boldsymbol{\lambda}$

(b) $\boldsymbol{g}$: ensemble

(c) $\boldsymbol{z}^{(1)}$: path

(d) $\boldsymbol{z}^{(2)}$: tree

(e) $\boldsymbol{z}^{(3)}$: graphlet

$\sum_{m=1}^{M} \lambda_m \boldsymbol{z}^{(m)}$ $(\lambda_1 = \mathbf{0.5504})$ $(\lambda_2 = 0.0746)$ $(\lambda_3 = 0.08103)$

Figure 4: t-SNE visualizations of GNNs' pattern representations (supervised) for PROTEINS.

7.1 Supervised Learning

We conduct supervised XGL via pattern analysis by solving optimization with the classification loss (11). The dataset is split into 80% training, 10% validation, and 10% testing data. The weight parameter $\boldsymbol{\lambda}$, indicating each pattern's contribution to graph representation learning, is reported in Table 2. We also visualize the graph representation $\boldsymbol{g}$ and three pattern representations $\boldsymbol{z}^{(m)}$ of PROTEINS in Figure 4. Results show the paths pattern is most important for learning $\boldsymbol{g}$, and the ensemble representation $\boldsymbol{g}$ outperforms single pattern representations $\boldsymbol{z}^{(m)}$.

Table 2: $\boldsymbol{\lambda}$ of supervised PXGL-GNN. The largest value is **bold** and the second largest value is blue.

Pattern	MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M5K
paths	0.095 ± 0.014	**0.550 ± 0.070**	0.093 ± 0.012	0.022 ± 0.002	**0.587 ± 0.065**	0.145 ± 0.018	0.131 ± 0.027	0.027 ± 0.003
trees	0.046 ± 0.005	0.074 ± 0.009	0.054 ± 0.006	0.063 ± 0.008	0.105 ± 0.013	0.022 ± 0.003	0.055 ± 0.007	0.025 ± 0.003
graphlets	0.062 ± 0.008	0.081 ± 0.011	0.125 ± 0.015	0.101 ± 0.013	0.063 ± 0.008	0.084 ± 0.011	0.026 ± 0.003	0.054 ± 0.007
cycles	**0.654 ± 0.085**	0.099 ± 0.013	0.094 ± 0.012	0.176 ± 0.022	0.022 ± 0.003	0.123 ± 0.016	0.039 ± 0.005	0.037 ± 0.005
cliques	0.082 ± 0.011	0.098 ± 0.012	**0.572 ± 0.073**	**0.574 ± 0.075**	0.134 ± 0.017	**0.453 ± 0.054**	0.279 ± 0.069	0.256 ± 0.067
wheels	0.026 ± 0.003	0.039 ± 0.005	0.051 ± 0.007	0.012 ± 0.002	0.068 ± 0.009	0.037 ± 0.004	0.036 ± 0.005	0.023 ± 0.003
stars	0.035 ± 0.005	0.056 ± 0.007	0.011 ± 0.002	0.052 ± 0.007	0.021 ± 0.003	0.136 ± 0.017	**0.447 ± 0.006**	**0.578 ± 0.033**

We compare our method with classical GNNs including GIN (Xu et al., 2018), DiffPool (Ying et al., 2018), DGCNN (Zhang et al., 2018), GRAPHSAGE (Hamilton et al., 2017), subgraph-based GNNs including SubGNN (Kriege & Mutzel, 2012), SAN (Zhao et al., 2018), SAGNN (Zeng et al., 2023a), and recent methods including S2GAE (Tan et al., 2023) and ICL (Zhao et al., 2024). The accuracies in Table 3 show that our method performs the best.

Table 3: Graph Classification Accuracy (%). The best accuracy is **bold** and the second best is blue.

Method	MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M5K
GIN	84.53 ± 2.38	73.38 ±2.16	76.38 ±1.58	73.36 ±1.78	75.83 ± 1.29	72.52 ± 1.62	83.27 ± 1.30	52.48 ± 1.57
DiffPool	86.72 ± 1.95	76.07 ±1.62	77.42 ±2.14	75.42 ±2.16	78.77 ± 1.36	73.55 ± 2.14	84.16 ± 1.28	51.39 ± 1.48
DGCNN	84.29 ± 1.16	75.53 ±2.14	76.57 ±1.09	74.81 ±1.53	77.59 ± 2.24	72.19 ± 1.97	86.33 ± 2.29	53.18 ± 2.41
GRAPHSAGE	86.35 ± 1.31	74.21 ±1.85	79.24 ±2.25	77.93 ±2.04	76.37 ± 2.11	73.86 ± 2.17	85.59 ± 1.92	51.65 ± 2.55
SubGNN	87.52 ± 2.37	76.38 ±1.57	82.51 ±1.67	82.58 ±1.79	81.26 ± 1.53	71.58 ± 1.20	88.47 ± 1.83	53.27 ± 1.93
SAN	92.65 ± 1.53	75.62 ±2.39	81.36 ±2.10	83.07 ±1.54	82.73 ± 1.92	75.27 ± 1.43	90.38 ± 1.54	55.49 ± 1.75
SAGNN	93.24 ± 2.51	75.61 ±2.28	84.12 ±1.73	81.29 ±1.22	79.94 ± 1.83	74.53 ± 2.57	89.57 ± 2.13	54.11 ± 1.22
ICL	91.34 ± 2.19	75.44 ±1.26	82.77 ±1.42	83.45 ±1.78	81.45 ± 1.21	73.29 ± 1.46	90.13 ± 1.40	56.21 ± 1.35
S2GAE	89.27 ± 1.53	76.47 ±1.12	84.30 ±1.77	82.37 ±2.24	82.35 ± 2.34	75.77 ± 1.72	90.21 ± 1.52	54.53 ± 2.17
PXGL-GNN	**94.87 ± 2.26**	**78.23 ±2.46**	**86.54 ±1.95**	**85.78 ±2.07**	**83.96 ± 1.59**	**77.35 ± 2.32**	**91.84 ± 1.69**	**57.36 ± 2.14**

7.2 Unsupervised Learning

We conduct unsupervised XGL via pattern analysis by solving optimization (with the KL divergence loss (10). The weight parameter $\boldsymbol{\lambda}$ for XGL is reported in Table 4. The visualization of unsupervised XGL results are in Appendix F.4. Results show that the ensemble representation $\boldsymbol{g}$ outperforms single pattern representations $\boldsymbol{z}^{(m)}$.

For clustering performance, we use clustering accuracy (ACC) and Normalized Mutual Information (NMI). Baselines include four kernels: Random walk kernel (RW) (Borgwardt et al., 2005),

Table 4: $\boldsymbol{\lambda}$ of unsupervised PXGL-GNN. The largest value is **bold** and the second largest value is blue.

Pattern	MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M5K
paths	0.085 ± 0.021	**0.463 ± 0.057**	0.083 ± 0.010	0.023 ± 0.001	**0.478 ± 0.046**	0.153 ± 0.018	0.101 ± 0.007	0.084 ± 0.006
trees	0.027 ± 0.005	0.082 ± 0.008	0.069 ± 0.007	0.042 ± 0.002	0.127 ± 0.017	0.082 ± 0.009	0.060 ± 0.003	0.036 ± 0.002
graphlets	0.074 ± 0.009	0.085 ± 0.010	0.172 ± 0.020	0.105 ± 0.012	0.055 ± 0.006	0.098 ± 0.011	0.025 ± 0.002	0.055 ± 0.005
cycles	**0.546 ± 0.065**	0.095 ± 0.011	0.108 ± 0.013	0.276 ± 0.033	0.022 ± 0.002	0.124 ± 0.014	0.043 ± 0.005	0.028 ± 0.003
cliques	0.197 ± 0.023	0.207 ± 0.025	**0.527 ± 0.063**	**0.482 ± 0.058**	0.243 ± 0.029	**0.423 ± 0.051**	0.212 ± 0.061	0.157 ± 0.067
wheels	0.032 ± 0.003	0.036 ± 0.004	0.018 ± 0.002	0.013 ± 0.001	0.044 ± 0.005	0.035 ± 0.004	0.036 ± 0.003	0.025 ± 0.013
stars	0.039 ± 0.004	0.032 ± 0.002	0.023 ± 0.003	0.059 ± 0.007	0.031 ± 0.001	0.085 ± 0.010	**0.455 ± 0.019**	**0.585 ± 0.022**

Sub-tree kernels (Da San Martino et al., 2012; Smola & Vishwanathan, 2002), Graphlet kernels (Pržulj, 2007), Weisfeiler-Lehman (WL) kernels (Kriege & Mutzel, 2012); and three unsupervised graph representation learning methods with Gaussian kernel in (10): InfoGraph (Sun et al., 2019), GCL (You et al., 2020), GraphACL (Luo et al., 2023). The results are in Table 5. Our method outperformed all competitors in almost all cases.

Table 5: ACC and NMI of Graph Clustering. The best ACC is **bold** and the the second best ACC is blue. The best NMI is green and the second best NMI is with *.

Method	Metric	MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M5K
RW	ACC	0.724 ±0.023	0.718 ± 0.019	0.529 ± 0.017	0.519 ±0.025	0.596 ±0.019	0.669 ±0.028	≥ 1 day	≥ 1 day
	NMI	0.283 ±0.008	0.226 ± 0.008	0.207 ± 0.003	0.218 ±0.009	0.356* ±0.002	0.295 ±0.006	≥ 1 day	≥ 1 day
sub-tree	ACC	0.716 ±0.017	0.683 ± 0.023	0.563 ± 0.026	0.532 ±0.016	0.533 ±0.021	0.627 ±0.022	≥ 1 day	≥ 1 day
	NMI	0.217 ±0.005	0.167 ± 0.004	0.225 ± 0.005	0.295 ±0.004	0.198 ±0.005	0.254 ±0.007	≥ 1 day	≥ 1 day
Graphlet	ACC	0.727 ±0.020	0.654 ± 0.017	**0.581 ± 0.014**	0.526 ±0.032	0.525 ±0.026	0.617 ±0.019	≥ 1 day	≥ 1 day
	NMI	0.225 ±0.003	0.131 ± 0.009	0.320 ± 0.009	0.273 ±0.005	0.217 ±0.003	0.210 ±0.004	≥ 1 day	≥ 1 day
WL	ACC	0.695 ±0.031	0.647 ± 0.032	0.517 ± 0.020	0.517 ±0.028	0.569 ±0.017	0.635 ±0.017	≥ 1 day	≥ 1 day
	NMI	0.185 ±0.007	0.135 ± 0.001	0.192 ± 0.008	0.234 ±0.007	0.253 ±0.007	0.261 ±0.003	≥ 1 day	≥ 1 day
InfoGraph	ACC	0.729 ±0.021	0.716 ± 0.019	0.549 ± 0.035	0.535 ±0.012	0.597 ±0.020	0.624 ±0.016	0.582 ±0.023	0.597 ±0.019
	NMI	0.236 ±0.005	0.231 ± 0.003	0.266 ± 0.004	0.263 ±0.005	0.311 ±0.008	0.198 ±0.005	0.206 ±0.006	0.286* ±0.006
GCL	ACC	0.761 ±0.014	0.723 ± 0.025	0.563 ± 0.016	0.558 ±0.010	0.582 ±0.015	0.653 ±0.024	0.573 ±0.015	0.582 ±0.017
	NMI	0.337 ±0.003	0.258 ± 0.002	0.289 ± 0.009	0.341 ±0.002	0.293 ± 0.009	0.253 ±0.008	0.195 ±0.005	0.266 ±0.005
GraphACL	ACC	0.742 ±0.023	0.731 ± 0.027	0.572 ± 0.027	0.522 ±0.013	0.554 ±0.013	0.679 ±0.013	0.594 ±0.014	0.567 ±0.023
	NMI	0.347* ±0.007	0.274* ± 0.008	0.312 ± 0.003	0.260 ±0.007	0.236 ±0.006	0.315* ±0.007	0.215* ± 0.006	0.238 ± 0.009
PXGL-GNN	ACC	**0.778 ±0.029**	**0.746 ± 0.019**	0.576 ± 0.035	**0.564 ±0.013**	**0.612 ±0.014**	**0.686 ±0.027**	**0.616 ±0.017**	**0.608 ±0.023**
	NMI	0.352 ±0.006	0.292 ± 0.010	0.317* ± 0.003	0.327* ±0.008	0.372 ±0.007	0.324 ±0.011	0.224 ± 0.009	0.295 ±0.012

8 Conclusion

This paper studied the explainability of graph representations. We proposed two strategies to learn and explain effective graph representations. The first one is based on graph ensemble kernel and the second one is based GNNs that learns from different graph patterns such as path, tree, etc. We also provide some theoretical analysis for the proposed method, including robustness analysis and generalization bound. The experiments showed that our method not only provides higher accuracy of classification and clustering than its competitors but also yields explainable results.

References

Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and S Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24(13): i241–i249, 2008.

Alireza Amouzad, Zahra Dehghanian, Saeed Saravani, Maryam Amirmazlaghani, and Behnam Roshanfekr. Graph isomorphism u-net. *Expert Systems with Applications*, 236:121280, 2024.

Plamen P Angelov, Eduardo A Soares, Richard Jiang, Nicholas I Arnold, and Peter M Atkinson. Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(5):e1424, 2021.

Steve Azzolin, Antonio Longa, Pietro Barbiero, Pietro Liò, and Andrea Passerini. Global explainability of gnns via logic combination of learned concepts. *arXiv preprint arXiv:2210.07147*, 2022.

Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pp. 8–pp. IEEE, 2005.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *Journal of Machine Learning Research*, 23(89):1–64, 2022.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33:10383–10395, 2020.

Giovanni Da San Martino, Nicolo Navarin, and Alessandro Sperduti. A tree-based kernel for graphs. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pp. 975–986. SIAM, 2012.

Kaize Ding, Yancheng Wang, Yingzhen Yang, and Huan Liu. Eliciting structural and semantic global knowledge in unsupervised graph contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7378–7386, 2023.

Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pp. 0210–0215. IEEE, 2018.

Jingcan Duan, Siwei Wang, Pei Zhang, En Zhu, Jingtao Hu, Hu Jin, Yue Liu, and Zhibin Dong. Graph anomaly detection via multi-scale contrastive learning networks with augmented view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7459–7467, 2023.

Vitaly Feldman and Jan Vondrak. High probability generalization bounds for uniformly stable algorithms with nearly optimal rate. In *Conference on Learning Theory*, pp. 1270–1279. PMLR, 2019.

Aasa Feragen, Niklas Kasenburg, Jens Petersen, Marleen de Bruijne, and Karsten Borgwardt. Scalable kernels for graphs with continuous attributes. *Advances in neural information processing systems*, 26, 2013.

Jacob Fox, Tim Roughgarden, C Seshadhri, Fan Wei, and Nicole Wein. Finding cliques in social networks: A new distribution-free model. *SIAM journal on computing*, 49(2):448–464, 2020.

Fabrizio Frasca, Beatrice Bevilacqua, Michael Bronstein, and Haggai Maron. Understanding and extending subgraph gnns by rethinking their symmetries. *Advances in Neural Information Processing Systems*, 35:31376–31390, 2022.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*, pp. 129–143. Springer, 2003.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

Lorenzo Giusti, Teodora Reu, Francesco Ceccarelli, Cristian Bodnar, and Pietro Liò. Cin++: Enhancing topological message passing. *arXiv preprint arXiv:2306.03561*, 2023.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

Vikas Hassija, Vinay Chamola, Atmesh Mahapatra, Abhinandan Singal, Divyansh Goel, Kaizhu Huang, Simone Scardapane, Indro Spinelli, Mufti Mahmud, and Amir Hussain. Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, 16(1): 45–74, 2024.

Shohei Hido and Hisashi Kashima. A linear-time graph kernel. In *2009 Ninth IEEE International Conference on Data Mining*, pp. 179–188. IEEE, 2009.

Ningyuan Teresa Huang and Soledad Villar. A short tutorial on the weisfeiler-lehman test and its variants. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8533–8537. IEEE, 2021.

Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, and Ambuj Singh. Global counterfactual explainer for graph neural networks. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pp. 141–149, 2023.

Suk-Geun Hwang. Cauchy's interlace theorem for eigenvalues of hermitian matrices. *The American mathematical monthly*, 111(2):157–159, 2004.

Chuntao Jiang, Frans Coenen, and Michele Zito. Finding frequent subgraphs in longitudinal social network data using a weighted graph mining approach. In *Advanced Data Mining and Applications: 6th International Conference, ADMA 2010, Chongqing, China, November 19-21, 2010, Proceedings, Part I 6*, pp. 405–416. Springer, 2010.

Hyunju Kang, Geonhee Han, and Hogun Park. Unr-explainer: Counterfactual explanations for unsupervised node representation learning models. In *The Twelfth International Conference on Learning Representations*, 2024.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Risi Kondor and Horace Pan. The multiscale laplacian graph kernel. *Advances in neural information processing systems*, 29, 2016.

Mert Kosan, Samidha Verma, Burouj Armgaan, Khushbu Pahwa, Ambuj Singh, Sourav Medya, and Sayan Ranu. Gnnx-bench: Unravelling the utility of perturbation-based gnn explainers through in-depth benchmarking. *arXiv preprint arXiv:2310.01794*, 2023.

Nils Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483*, 2012.

Nils M Kriege, Fredrik D Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5:1–42, 2020.

O-Joun Lee et al. Transitivity-preserving graph representation learning for bridging local connectivity and role-based similarity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12456–12465, 2024.

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.

Xiao Luo, Wei Ju, Yiyang Gu, Zhengyang Mao, Luchen Liu, Yuhui Yuan, and Ming Zhang. Self-supervised graph-level representation learning with adversarial contrastive learning. *ACM Transactions on Knowledge Discovery from Data*, 2023.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International conference on machine learning*, pp. 4363–4371. PMLR, 2019.

Harry L Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of chemical documentation*, 5(2):107–113, 1965.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL `www.graphlearning.io`.

Leslie O'Bray, Max Horn, Bastian Rieck, and Karsten Borgwardt. Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions. *arXiv preprint arXiv:2106.01098*, 2021.

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

Noel M O'Boyle and Roger A Sayle. Comparing structural fingerprints using a literature-based similarity benchmark. *Journal of cheminformatics*, 8:1–14, 2016.

Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.

Syed Asad Rahman, Matthew Bashton, Gemma L Holliday, Rainer Schrader, and Janet M Thornton. Small molecule subgraph detector (smsd) toolkit. *Journal of cheminformatics*, 1:1–13, 2009.

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.

Caihua Shan, Yifei Shen, Yao Zhang, Xiang Li, and Dongsheng Li. Reinforcement learning enhanced explainer for graph neural networks. *Advances in Neural Information Processing Systems*, 34:22523–22533, 2021.

Xiao Shen, Dewang Sun, Shirui Pan, Xi Zhou, and Laurence T Yang. Neighbor contrastive learning on learnable graph augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9782–9791, 2023.

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pp. 488–495. PMLR, 2009.

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and SVN Vishwanathan. Hash kernels for structured data. *Journal of Machine Learning Research*, 10(11), 2009.

Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgiannis. Grakel: A graph kernel library in python. *J. Mach. Learn. Res.*, 21 (54):1–5, 2020.

Geri Skenderi, Hang Li, Jiliang Tang, and Marco Cristani. Graph-level representation learning with joint-embedding predictive architectures. *arXiv preprint arXiv:2309.16014*, 2023.

Alex Smola and SVN Vishwanathan. Fast kernels for string and tree matching. *Advances in neural information processing systems*, 15, 2002.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34:15920–15933, 2021.

Qiaoyu Tan, Ninghao Liu, Xiao Huang, Soo-Hyun Choi, Li Li, Rui Chen, and Xia Hu. S2gae: self-supervised graph autoencoders are generalizable learners with graph masking. In *Proceedings of the sixteenth ACM international conference on web search and data mining*, pp. 787–795, 2023.

Quang Truong and Peter Chin. Weisfeiler and lehman go paths: Learning topological features via path complexes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 15382–15391, 2024.

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

Cédric Vincent-Cuaz, Rémi Flamary, Marco Corneli, Titouan Vayer, and Nicolas Courty. Template based graph neural network with optimal transport distances. *Advances in Neural Information Processing Systems*, 35:11800–11814, 2022.

Pengyang Wang, Yanjie Fu, Yuanchun Zhou, Kunpeng Liu, Xiaolin Li, and Kien A Hua. Exploiting mutual information for substructure-aware graph representation learning. In *IJCAI*, pp. 3415–3421, 2020.

Xin Wang, Shuyi Fan, Kun Kuang, and Wenwu Zhu. Explainable automated graph representation learning with hyperparameter importance. In *International Conference on Machine Learning*, pp. 10727–10737. PMLR, 2021.

Chunyu Wei, Yu Wang, Bing Bai, Kai Ni, David Brady, and Lu Fang. Boosting graph contrastive learning via graph contrastive saliency. In *International conference on machine learning*, pp. 36839–36855. PMLR, 2023.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pp. 478–487. PMLR, 2016.

Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. Infogcl: Information-aware graph contrastive learning. *Advances in Neural Information Processing Systems*, 34:30414–30425, 2021.

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

Yihang Yin, Qingzhong Wang, Siyu Huang, Haoyi Xiong, and Xiang Zhang. Autogcl: Automated graph contrastive learning via learnable view generators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8892–8900, 2022.

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.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.

Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *International Conference on Machine Learning*, pp. 12121–12132. PMLR, 2021.

Zhaoning Yu and Hongyang Gao. Motifexplainer: a motif-based graph neural network explainer. *arXiv preprint arXiv:2202.00519*, 2022.

Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 430–438, 2020.

Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International conference on machine learning*, pp. 12241–12252. PMLR, 2021.

Dingyi Zeng, Wanlong Liu, Wenyu Chen, Li Zhou, Malu Zhang, and Hong Qu. Substructure aware graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11129–11137, 2023a.

Liang Zeng, Lanqing Li, Ziqi Gao, Peilin Zhao, and Jian Li. Imgcl: Revisiting graph contrastive learning on imbalanced node classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11138–11146, 2023b.

Jiaxing Zhang, Dongsheng Luo, and Hua Wei. Mixupexplainer: Generalizing explanations for graph neural networks with data augmentation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3286–3296, 2023a.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. Spectral feature augmentation for graph contrastive learning and beyond. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11289–11297, 2023b.

Xiaohan Zhao, Bo Zong, Ziyu Guan, Kai Zhang, and Wei Zhao. Substructure assembling network for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Zhe Zhao, Pengkun Wang, Haibin Wen, Yudong Zhang, Zhengyang Zhou, and Yang Wang. A twist for graph classification: Optimizing causal information flow in graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17042–17050, 2024.

A Appendix

You may include other additional sections here.

B Math Definitions of Patterns

In our work, graph patterns refer to as subgraphs with practical meanings. Let $G = (V, E)$ be a graph. A subgraph $S = (V_S, E_S)$ of G is defined such that $V_S \subseteq V$ and $E_S \subseteq E \cap (V_S \times V_S)$. The math definitions of graph patterns are as follows:

- **Paths:** S is a *path* if there exists a sequence of distinct vertices $v_1, \ldots, v_k \in V_S$ such that $E_S = ((v_i, v_{i+1}) : i = 1, \ldots, k-1)$.
- **Trees:** S is a *tree* if it is connected and contains no cycles, i.e., it is acyclic and $|E_S| = |V_S| - 1$.
- **Graphlets:** S is a *graphlet* if it is a small connected induced subgraph of G, typically consisting of 2 to 5 vertices.
- **Cycles:** S is a *cycle* if there exists a sequence of distinct vertices $v_1, \ldots, v_k \in V_S$ such that $E_S = ((v_i, v_{i+1}) : i = 1, \ldots, k-1) \cup ((v_k, v_1))$.
- **Cliques:** S is a *clique* if every two distinct vertices in V_S are adjacent, thus $E_S = ((v_i, v_j) : v_i, v_j \in V_S, i \neq j)$.
- **Wheels:** S is a *wheel* if it consists of a cycle with vertices $v_1, \ldots, v_{k-1}$ and an additional central vertex v_k such that v_k is connected to all vertices of the cycle.
- **Stars:** S is a *star* if it consists of one central vertex v_c and several leaf vertices $v_1, \ldots, v_{k-1}$, where each leaf vertex is only connected to v_c. Thus, $E_S = ((v_c, v_i) : i = 1, \ldots, k-1)$.

C Related Works

In this section, we introduce previous works on explainable graph learning (XGL), graph representation learning (GRL), and graph kernels.

C.1 Explainable Graph Learning (XGL)

Explainable artificial intelligence (XAI) is a rapidly growing area in the AI community (Došilović et al., 2018; Adadi & Berrada, 2018; Angelov et al., 2021; Hassija et al., 2024). Explainable graph learning (XGL) (Kosan et al., 2023) can be roughly classified into two categories: model-level methods and instance-level methods.

Model-level Model-level or global explanations aim to understand the overall behavior of a model by identifying patterns in its predictions. For example, XGNN(Yuan et al., 2020) trains a graph generator to create graph patterns that maximize a certain prediction, providing high-level insights into GNN behavior. GLG-Explainer(Azzolin et al., 2022) combines local explanations into a logical formula over graphical concepts, offering human-interpretable global explanations aligned with ground-truth or domain knowledge. GCFExplainer(Huang et al., 2023) uses global counterfactual reasoning to find representative counterfactual graphs, providing a summary of global explanations through vertex-reinforced random walks on an edit map of graphs.

Instance-level Instance-level methods offer explanations tailored to specific predictions, focusing on why particular instances are classified in a certain manner. For instance, GNNExplainer (Ying et al., 2019) identifies a compact subgraph structure and a small subset of node features crucial for a GNN's prediction. PGExplainer (Luo et al., 2020) trains a graph generator to incorporate global information and uses a deep neural network (DNN) to parameterize the explanation generation process. SubgraphX (Yuan et al., 2021) efficiently explores different subgraphs using Monte Carlo tree search to explain predictions. RG-Explainer (Shan et al., 2021) constructs a connected explanatory subgraph by sequentially adding nodes, consistent with the message passing scheme. MixupExplainer (Zhang et al., 2023a) introduces a general form of Graph Information Bottleneck (GIB) to address distribution shifting issues in post-hoc graph explanation. AutoGR (Wang et al., 2021) introduces an explainable AutoML approach for graph representation learning. UNR-Explainer (Kang et al., 2024) identifies the top-k most important nodes in a graph to determine the most significant subgraph. It is a classic instance-level explainable graph learning method focused on node representation. However, this task is entirely different from our approach, as it addresses node-level representation rather than representation-level explainability. For this reason, we did not include a comparison.

C.2 Graph Representation Learning

Graph representation learning is crucial for transforming complex graphs into vectors, particularly for tasks like classification. The methods for graph representation learning are mainly classified into two categories: supervised and unsupervised learning.

Supervised Representation Learning Most GNNs can be used in supervised graph representation learning tasks by aggregating all the node embeddings into a graph representation using a readout function (Hamilton, 2020; Chami et al., 2022). Besides traditional GNNs like GCN (Kipf & Welling, 2016), GIN (Xu et al., 2018), and GAT (Veličković et al., 2017), recent works include: Template-based Fused Gromov-Wasserstein (FGW) (Vincent-Cuaz et al., 2022) computes a vector of FGW distances to learnable graph templates, acting as an alternative to global pooling layers. Path Isomorphism Network (PIN) (Truong & Chin, 2024) introduces a graph isomorphism test and a topological message-passing scheme operating on path complexes. Graph U-Net (Amouzad et al., 2024) proposes GIUNet for graph classification, combining node features and graph structure information using a pqPooling layer. Unified Graph Transformer Networks (UGT) (Lee et al., 2024) integrate local and global structural information into fixed-length vector representations using self-attention. CIN++ (Giusti et al., 2023) enhances topological message passing to account for higher-order and long-range interactions, achieving state-of-the-art results. Graph Joint-Embedding Predictive Architectures (Graph-JEPA) (Skenderi et al., 2023) use masked modeling to learn embeddings for subgraphs and predict their coordinates on the unit hyperbola in the 2D plane.

Unsupervised Representation Learning Unsupervised methods aim to learn graph representations without labeled data. Notable methodologies include: InfoGraph (Sun et al., 2019) emphasizes mutual information between graph-level and node-level representations. Graph Contrastive Learning techniques (You et al., 2020; Suresh et al., 2021; You et al., 2021) enhance graph representations through diverse augmentation strategies. AutoGCL (Yin et al., 2022) introduces learnable graph view generators. GraphACL (Luo et al., 2023) adopts a novel self-supervised approach. InfoGCL (Xu et al., 2021) and SFA (Zhang et al., 2023b) focus on information transfer and feature augmentation in contrastive learning. Techniques like GCS (Wei et al., 2023), NCLA (Shen et al., 2023), S^3-CL (Ding et al., 2023), and ImGCL (Zeng et al., 2023b) refine graph augmentation and learning methods. GRADATE (Duan et al., 2023) integrates subgraph contrast into multi-scale learning networks.

GNNs using Subgraphs and Substructures Our pattern analysis method samples subgraphs from different graph patterns to conduct explainable graph representation learning. The key novelty and contribution of our paper is that graph pattern analysis provides explainability for representations. We discuss other GNN methods based on subgraphs and substructures here: Subgraph Neural Networks (SubGNN) (Kriege & Mutzel, 2012) learn disentangled subgraph representations using a novel subgraph routing mechanism, but they sample subgraphs randomly, lacking explainability. Substructure Aware Graph Neural Networks (SAGNN) (Zeng et al., 2023a) use cut subgraphs and return probability to capture structural information but focus on expressiveness rather than explainability. Mutual Information (MI) Induced Substructure-aware GRL (Wang et al., 2020) maximizes MI between original and learned representations at both node and graph levels but does not provide explainable representation learning. Substructure Assembling Network (SAN) (Zhao et al., 2018) hierarchically assembles graph components using an RNN variant but lacks explainability in representation learning.

Several works focus on analyzing the expressiveness of methods by their ability to count substructures, but they do not provide explainable representation learning. For example: (Chen et al., 2020) analyze the expressiveness of MPNNs (Gilmer et al., 2017) and 2nd-order Invariant Graph Networks (2-IGNs) (Maron et al., 2019) based on their ability to count specific subgraphs, highlighting tasks that are challenging for classical GNN architectures but not focusing on explainability. (Frasca et al., 2022) compare the expressiveness of SubGNN (Kriege & Mutzel, 2012) and 2-IGNs (Maron et al., 2019) using symmetry analysis, establishing a link between Subgraph GNNs and Invariant Graph Networks.

C.3 Graph Kernels

Graph kernels evaluate the similarity between two graphs. Over the past decades, numerous graph kernels have been proposed (Siglidis et al., 2020). We classify them into two categories: pattern counting kernels and non-pattern counting kernels.

Pattern Counting Kernels Pattern counting kernels compare specific substructures within graphs to evaluate similarity (Kriege et al., 2020). For examples, Random walk kernels (Borgwardt et al., 2005; Gärtner et al., 2003) measure graph similarity by counting common random walks between graphs. Shortest-path kernels(Borgwardt & Kriegel, 2005) compare graphs using the shortest distance matrix generated by the Floyd-Warshall algorithm, based on edge values and node labels. Sub-tree kernels (Da San Martino et al., 2012; Smola & Vishwanathan, 2002) decompose graphs into ordered Directed Acyclic Graphs (DAGs) and use tree kernels extended to DAGs. Graphlet kernels (Pržulj, 2007) count small connected non-isomorphic subgraphs (graphlets) within graphs and compare their distributions. Weisfeiler-Lehman subtree kernels (Kriege & Mutzel, 2012) use small subgraphs, like graphlets, to compare graphs, allowing flexibility to compare vertex and edge attributes with arbitrary kernel functions.

Non-pattern Counting Kernels Non-pattern counting kernels evaluate graph similarity without relying on specific substructure counts. For examples, Neighborhood hash kernel (Hido & Kashima, 2009) use binary arrays to represent node labels and logical operations on connected node labels. This kernel has linear time complexity. GraphHopper kernel (Feragen et al., 2013) compare shortest paths between node pairs using kernels on nodes encountered while hopping along shortest paths. Graph hash kernel (Shi et al., 2009) use hashing for efficient kernel computation, suitable for data streams and sparse feature spaces, with deviation bounds from the exact kernel matrix. Multiscale Laplacian Graph (MLG) kernel (Kondor & Pan, 2016) account for structure at different scales using

Feature Space Laplacian Graph (FLG) kernels, applied recursively to subgraphs. They introduce a randomized projection procedure similar to the Nystrom method for RKHS operators.

D Proof for Robustness Analysis

Let Δ_A and Δ_X be some perturbations on adjacency matrix and node attributes, then the perturbed graph is denoted as $\tilde{G} = (\boldsymbol{A} + \Delta_A, \boldsymbol{X} + \Delta_X)$. Let $\boldsymbol{g}$ be the graph representation of G and $\tilde{\boldsymbol{g}}$ be the graph representation of $\tilde{G}$. The robustness analysis is to find the upper bound of $\|\tilde{\boldsymbol{g}} - \boldsymbol{g}\|$.

Assumptions and Notations: Let $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \Delta_A$ and $\tilde{\boldsymbol{X}} = \boldsymbol{X} + \Delta_X$. We suppose that $\|\boldsymbol{A}\|_2 \leq \beta_A$, $\|\boldsymbol{X}\|_F \leq \beta_B$ and $\|\boldsymbol{W}^{(m,l)}\|_2 \leq \beta_W$, $(\forall\, m \in [M],\ l \in [L])$, the activation $\sigma(\cdot)$ of GCN is ρ-Lipschitz continuous. We denote the minimum node degree of G as α, the effects of structural perturbation as $\kappa = \min(\boldsymbol{1}^\top \Delta_A)$, and $\Delta_D := \boldsymbol{I} - \text{diag}(\boldsymbol{1}^\top(\boldsymbol{I} + \boldsymbol{A} + \Delta_A))^{\frac{1}{2}} \text{diag}(\boldsymbol{1}^\top \boldsymbol{A})^{-\frac{1}{2}}$.

Theorem: Our conclusion for robustness analysis is as follows:

$$\|\tilde{\boldsymbol{g}} - \boldsymbol{g}\| \leq \frac{1}{\sqrt{n}} \rho^L \beta_W^L \beta_X (1+\alpha)^{-L} (1 + \beta_A + \|\Delta_A\|_2)^L \Big(1 + 2L\|\Delta_D\|_2 + L(1 + \beta_A + \|\Delta_A\|_2)^{-1} \|\Delta_A\|_2)\Big) \tag{16}$$

To provide a clearer analysis, we first use the whole graph G and $\tilde{G}$ as the input of the pattern representation learning function F without sampling the subgraphs. Then we consider using the subgraph sampling to analyze $\boldsymbol{g}$ and $\tilde{\boldsymbol{g}}$ and finally finish the proof of robustness analysis.

D.1 Learning Pattern Representations using the Whole Graph without Sampling

In this section, we first consider using the whole graph G and $\tilde{G}$ as the input of the pattern representation learning function F without sampling the subgraphs, i.e., we analyze $F(\boldsymbol{A}, \boldsymbol{X}; \mathcal{W}^{(m)})$ and $F(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{X}}; \mathcal{W}^{(m)})$.

Representation Learning Function F In theoretical analysis, we suppose the pattern representation learning function F is a L-layer GCN (Kipf & Welling, 2016) with an average pooling avg-pool : $\mathbb{R}^{n \times d} \to \mathbb{R}^d$ as the output layer. The pattern learning function for the pattern $\mathcal{P}_m$ is denoted as $F(\boldsymbol{A}, \boldsymbol{X}; \mathcal{W}^{(m)})$, where $\mathcal{W}^{(m)} = \{\boldsymbol{W}^{(m,1)}, ..., \boldsymbol{W}^{(m,l)}, ..., \boldsymbol{W}^{(m,L)}\}$ and $\boldsymbol{W}^{(m,l)}$ is the trainable parameter of the l-th layer. We use the adjacency matrix $\boldsymbol{A}$ and node feature matrix $\boldsymbol{X}$ of G as the input. Then the self-connected adjacency matrix is $\hat{\boldsymbol{A}} = \boldsymbol{I} + \boldsymbol{A}$, the diagonal matrix is $\hat{\boldsymbol{D}} = \text{diag}(\boldsymbol{1}^\top \hat{\boldsymbol{A}})$, then the normalized self-connected adjacency matrix is $\boldsymbol{U} = \hat{\boldsymbol{D}}^{-\frac{1}{2}} \hat{\boldsymbol{A}} \hat{\boldsymbol{D}}^{-\frac{1}{2}}$. Let $\sigma(\cdot)$ be an activation function, then the hidden embedding $\boldsymbol{X}^{(m,L)}$ of the l-th layer is defined as follows

$$\boldsymbol{X}^{(m,l)} = \underbrace{\sigma(\boldsymbol{U} ... \sigma(\boldsymbol{U}}_{l \text{ times}} \boldsymbol{X} \underbrace{\boldsymbol{W}^{(m,1)}) ... \boldsymbol{W}^{(m,l)})}_{l \text{ times}}, \ \ \forall\, l \in [L], \tag{17}$$

The pattern representation $\boldsymbol{z}^{(m)}$ of pattern $\mathcal{P}_m$ is obtained by

$$\boldsymbol{z}^{(m)} = F(\boldsymbol{A}, \boldsymbol{X}; \mathcal{W}^{(m)}) = \text{avg-pool}(\boldsymbol{X}^{(m,L)}) = \frac{1}{n} \boldsymbol{1}^\top \boldsymbol{X}^{(m,L)} \tag{18}$$

For a perturbed graph $\tilde{G}$, we use $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{X}}$ to denote the adjacency matrix and feature matrix respectively. The corresponding self-connected adjacency matrix is $\hat{\boldsymbol{A}}' = \boldsymbol{I} + \tilde{\boldsymbol{A}}$ and the degree matrix as $\hat{\boldsymbol{D}}' = \text{diag}(\boldsymbol{1}^\top \hat{\boldsymbol{A}}')$. Then the normalized self-connected adjacency matrix is $\tilde{\boldsymbol{U}} = \hat{\boldsymbol{D}}'^{-\frac{1}{2}} \hat{\boldsymbol{A}}' \hat{\boldsymbol{D}}'^{-\frac{1}{2}}$. The l-th layer hidden embedding of $\tilde{G}$ is defined as follows

$$\tilde{\boldsymbol{X}}^{(m,l)} = \underbrace{\sigma(\tilde{\boldsymbol{U}} ... \sigma(\tilde{\boldsymbol{U}}}_{l \text{ times}} \tilde{\boldsymbol{X}} \underbrace{\boldsymbol{W}^{(m,1)}) ... \boldsymbol{W}^{(m,l)})}_{l \text{ times}}, \ \ \forall\, l \in [L], \tag{19}$$

The perturbed pattern representation $\tilde{\boldsymbol{z}}^{(m)}$ of pattern $\mathcal{P}_m$ is obtained by

$$\tilde{\boldsymbol{z}}^{(m)} = F(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{X}}; \mathcal{W}^{(m)}) = \text{avg-pool}(\tilde{\boldsymbol{X}}^{(m,L)}) = \frac{1}{n} \boldsymbol{1}^\top \tilde{\boldsymbol{X}}^{(m,L)} \tag{20}$$

Lemma D.1. *Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two square matrices, $\|\cdot\|_2$ be the spectral norm and $\|\cdot\|_F$ be the Frobenius norm , then $\|\boldsymbol{X}\|_2 \leq \|\boldsymbol{X}\|_F$, $\|\boldsymbol{XY}\|_2 \leq \|\boldsymbol{X}\|_2\|\boldsymbol{Y}\|_2$ and $\|\boldsymbol{XY}\|_F \leq \|\boldsymbol{X}\|_2\|\boldsymbol{Y}\|_F$.*

Lemma D.2 (Inequalities)**.** *Some inequalities that will be used in our proof:*

$$\begin{aligned}
&\|\boldsymbol{U}\|_2 \leq (1+\alpha)^{-1}(1+\beta_A)\\
&\|\tilde{\boldsymbol{U}}\|_2 \leq (1+\alpha+\kappa)^{-1}(1+\beta_A+\|\Delta_A\|_2)\\
&\|\Delta_U\|_2 \leq 2(1+\beta_A)(1+\alpha)^{-1}\|\Delta_D\|_2 + (1+\alpha+\kappa)^{-1}\|\Delta_A\|_2\\
&\|\Delta_{X^{(m,l)}}\|_F \leq \rho^l\beta_W^l\beta_X(1+\alpha)^{-l}(1+\beta_A+\|\Delta_A\|_2)^l\Big(1+2l\|\Delta_D\|_2 + l(1+\beta_A+\|\Delta_A\|_2)^{-1}\|\Delta_A\|_2\Big)
\end{aligned}$$

Proof. Since the minimum node degree of G is α, then we have $\|\hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2 \leq (1+\alpha)^{-\frac{1}{2}}$. Since $\|\boldsymbol{A}\|_2 \leq \beta_A$, then $\|\hat{\boldsymbol{A}}\|_2 \leq 1+\beta_A$. We have

$$\|\boldsymbol{U}\|_2 \leq \|\hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2\|\hat{\boldsymbol{A}}\|_2\|\hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2 \leq (1+\alpha)^{-1}(1+\beta_A). \tag{21}$$

Similarly, since the effects of structural perturbation is $\kappa = \min(\boldsymbol{1}^\top\Delta_A)$, we have $\|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2 \leq (1+\alpha+\kappa)^{-\frac{1}{2}}$. Since $\|\tilde{\boldsymbol{A}}'\|_2 \leq \|\hat{\boldsymbol{A}}\|_2 + \|\Delta_A\|_2 \leq 1+\beta_A+\|\Delta_A\|_2$, we obtain

$$\|\tilde{\boldsymbol{U}}\|_2 \leq \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2\|\hat{\boldsymbol{A}}'\|_2\|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2 \leq (1+\alpha+\kappa)^{-1}(1+\beta_A+\|\Delta_A\|_2). \tag{22}$$

Letting $\Delta_U = \tilde{\boldsymbol{U}} - \boldsymbol{U}$, we have

$$\begin{aligned}
\|\Delta_U\|_2 &= \|\tilde{\boldsymbol{U}} - \boldsymbol{U}\|_2 = \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}(\hat{\boldsymbol{A}}+\Delta_A)\hat{\boldsymbol{D}}'^{-\frac{1}{2}} - \hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2\\
&= \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}'^{-\frac{1}{2}} - \hat{\boldsymbol{D}}'^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}} + \hat{\boldsymbol{D}}'^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}} - \hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}} + \hat{\boldsymbol{D}}'^{-\frac{1}{2}}\Delta_A\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2\\
&\leq \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\hat{\boldsymbol{A}}(\hat{\boldsymbol{D}}'^{-\frac{1}{2}} - \hat{\boldsymbol{D}}^{-\frac{1}{2}})\|_2 + \|(\hat{\boldsymbol{D}}'^{-\frac{1}{2}} - \hat{\boldsymbol{D}}^{-\frac{1}{2}})\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2 + \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\Delta_A\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2\\
&\leq (\|\hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2 + \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2)\|\hat{\boldsymbol{A}}\|_2\|\hat{\boldsymbol{D}}'^{-\frac{1}{2}} - \hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2 + \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2\|\Delta_A\|_2\|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2\\
&\leq ((1+\alpha)^{-\frac{1}{2}} + (1+\alpha+\kappa)^{-\frac{1}{2}})(1+\beta_A)\|\hat{\boldsymbol{D}}'^{-\frac{1}{2}} - \hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2 + (1+\alpha+\kappa)^{-1}\|\Delta_A\|_2\\
&\leq 2(1+\beta_A)(1+\alpha)^{-\frac{1}{2}}\|\hat{\boldsymbol{D}}'^{-\frac{1}{2}} - \hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2 + (1+\alpha+\kappa)^{-1}\|\Delta_A\|_2\\
&\leq 2(1+\beta_A)(1+\alpha)^{-\frac{1}{2}}(1+\alpha+\kappa)^{-\frac{1}{2}}\|\boldsymbol{I} - \hat{\boldsymbol{D}}'^{\frac{1}{2}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2 + (1+\alpha+\kappa)^{-1}\|\Delta_A\|_2\\
&= 2(1+\beta_A)(1+\alpha)^{-\frac{1}{2}}(1+\alpha+\kappa)^{-\frac{1}{2}}\|\Delta_D\|_2 + (1+\alpha+\kappa)^{-1}\|\Delta_A\|_2\\
&\leq 2(1+\beta_A)(1+\alpha)^{-1}\|\Delta_D\|_2 + (1+\alpha+\kappa)^{-1}\|\Delta_A\|_2
\end{aligned} \tag{23}$$

where $\Delta_D = \boldsymbol{I} - \hat{\boldsymbol{D}}'^{\frac{1}{2}}\hat{\boldsymbol{D}}^{-\frac{1}{2}} = \boldsymbol{I} - \text{diag}(\boldsymbol{1}^\top(\boldsymbol{I}+\boldsymbol{A}+\Delta_A))^{\frac{1}{2}}\text{diag}(\boldsymbol{1}^\top\boldsymbol{A})^{-\frac{1}{2}}$.

The $\boldsymbol{X}^{(m,l)}$ is the hidden embedding of the l-layer GCN of $F(\boldsymbol{A},(X);\mathcal{W}^{(m,l)})$, which is the representation learning function related to $\mathcal{P}_m$. Then we have

$$\begin{aligned}
\|\boldsymbol{X}^{(m,l)}\|_F &= \|\sigma(\boldsymbol{U}\boldsymbol{X}^{(m,l-1)}\boldsymbol{W}^{(m,l)})\|_F\\
&\leq \rho\|\boldsymbol{U}\boldsymbol{X}^{(m,l-1)}\boldsymbol{W}^{(m,l)}\|_F\\
&\leq \rho\|\boldsymbol{U}\|_2\|\boldsymbol{X}^{(m,l-1)}\|_F\|\boldsymbol{W}^{(m,l)}\|_2\\
&\leq \rho\beta_W(1+\alpha)^{-1}(1+\beta_A)\|\boldsymbol{X}^{(m,l-1)}\|_F\\
&\leq \rho^l\beta_W^l(1+\beta_A)^l(1+\alpha)^{-l}\|\boldsymbol{X}\|_F\\
&\leq \rho^l\beta_W^l\beta_X(1+\beta_A)^l(1+\alpha)^{-l}
\end{aligned} \tag{24}$$

For $\Delta_{X^{(m,l)}} = \tilde{\boldsymbol{X}}^{(m,l)} - \boldsymbol{X}^{(m,l)}$, we have

$$\begin{aligned}
\|\Delta_{X^{(m,l)}}\|_F &= \|\tilde{\boldsymbol{X}}^{(m,l)} - \boldsymbol{X}^{(m,l)}\|_F \\
&= \|\sigma(\tilde{\boldsymbol{U}}\tilde{\boldsymbol{X}}^{(m,l-1)}\boldsymbol{W}^{(l)}) - \sigma(\boldsymbol{U}\boldsymbol{X}^{(l-1)}\boldsymbol{W}^{(l)})\|_F \\
&\le \rho\|\tilde{\boldsymbol{U}}\tilde{\boldsymbol{X}}^{(m,l-1)} - \boldsymbol{U}\boldsymbol{X}^{(m,l-1)}\|_F\|\boldsymbol{W}^{(m,l)}\|_2 \\
&\le \rho\beta_W\left(\|\tilde{\boldsymbol{U}}\|_2\|\Delta_{X^{(m,l-1)}}\|_F + \|\Delta_U\|_2\|\boldsymbol{X}^{(m,l-1)}\|_F\right) \\
&\le \rho^2\beta_W^2\|\tilde{\boldsymbol{U}}\|_2^2\|\Delta_{X^{(m,l-2)}}\|_F + \rho^2\beta_W^2\|\tilde{\boldsymbol{U}}\|_2\|\Delta_U\|_2\|\boldsymbol{X}^{(m,l-2)}\|_F + \rho\beta_W\|\Delta_U\|_2\|\boldsymbol{X}^{(m,l-1)}\|_F \\
&\le \rho^l\beta_W^l\|\tilde{\boldsymbol{U}}\|_2^l\|\Delta_X\|_F + \sum_{k=1}^{l}\rho^k\beta_W^k\|\tilde{\boldsymbol{U}}\|_2^{k-1}\|\Delta_U\|_2\|\boldsymbol{X}^{(m,l-k)}\|_F \\
&\le \rho^l\beta_W^l(1+\beta_A+\|\Delta_A\|_2)^{l-1}(1+\alpha)^{-l}\left[(1+\beta_A+2\|\Delta_A\|_2)\|\Delta_X\|_F + 2l\beta_X(1+\beta_A)\|\Delta_D\|_2\right]
\end{aligned} \tag{25}$$

□

D.2 Learning Graph Representations via Sampling Subgraphs

In this section, we consider learning the graph representation $\boldsymbol{g}$ and $\tilde{\boldsymbol{g}}$ respectively by sampling subgraphs of graph patterns. That is, we analyse $F(\boldsymbol{A}_S, \boldsymbol{X}_S; \mathcal{W}^{(m)})$ and $F(\tilde{\boldsymbol{A}}_{\tilde{S}}, \tilde{\boldsymbol{X}}_{\tilde{\boldsymbol{S}}}; \mathcal{W}^{(m)})$. And then we provide the upper bound of $\|\tilde{\boldsymbol{g}} - \boldsymbol{g}\|$.

Let S be a subgraph of graph G and $\tilde{S}$ be a subgraph of graph $\tilde{G}$. Let Δ_{A_S} and Δ_{X_S} be some perturbations on adjacency matrix and node attributes, then the perturbed graph is denoted as $\tilde{S} = (\boldsymbol{A}_S + \Delta_{A_S}, \boldsymbol{X}_S + \Delta_{X_S})$.

Assumptions and Notations: Let $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \Delta_A$ and $\tilde{\boldsymbol{X}} = \boldsymbol{X} + \Delta_X$. We suppose that $\|\boldsymbol{A}\|_2 \le \beta_A$, $\|\boldsymbol{X}\|_F \le \beta_B$ and $\|\boldsymbol{W}^{(m,l)}\|_2 \le \beta_W$, $(\forall\ m \in [M],\ l \in [L])$, the activation $\sigma(\cdot)$ of GCN is ρ-Lipschitz continuous. We denote the minimum node degree of G as α, the effects of structural perturbation as $\kappa = \min(\boldsymbol{1}^\top\Delta_A)$, and $\Delta_D := \boldsymbol{I} - \text{diag}(\boldsymbol{1}^\top(\boldsymbol{I} + \boldsymbol{A} + \Delta_A))^{\frac{1}{2}}\text{diag}(\boldsymbol{1}^\top\boldsymbol{A})^{-\frac{1}{2}}$. We present the following useful lemmas.

Lemma D.3 (Eigenvalue Interlacing Theorem (Hwang, 2004))**.** *Suppose $A \in \mathbb{R}^{n\times n}$ is symmetric. Let $B \in \mathbb{R}^{m\times m}$ with $m < n$ be a principal submatrix (obtained by deleting both the i-th row and i-th column for some value of i). Suppose A has eigenvalues $\lambda_1 \le \cdots \le \lambda_n$ and B has eigenvalues $\beta_1 \le \cdots \le \beta_m$. Then*

$$\lambda_k \le \beta_k \le \lambda_{k+n-m} \quad \textit{for } k = 1, \cdots, m.$$

Lemma D.4. *Since $\boldsymbol{X}_S$ and Δ_{X_S} are submatrices of $\boldsymbol{X}$ and Δ_X respectively, then we have*

$$\|\boldsymbol{X}_S\|_F \le \|\boldsymbol{X}\|_F, \ \textit{ and } \ \|\Delta_{X_S}\|_F \le \|\Delta_X\|_F.$$

Let $\Delta_{D_S} := \boldsymbol{I} - \textit{diag}(\boldsymbol{1}^\top(\boldsymbol{I} + \boldsymbol{A}_S + \Delta_{A_S}))^{\frac{1}{2}}\textit{diag}(\boldsymbol{1}^\top\boldsymbol{A}_S)^{-\frac{1}{2}}$. Base on the Eigenvalue Interlacing Theorem, for any subgraph S of graph G, since $\boldsymbol{A}_S$, Δ_{A_S}, Δ_{D_S} are principal submatrices of $\boldsymbol{A}$, Δ_A, Δ_D respectively, then we have

$$\|\boldsymbol{A}_S\|_2 \le \|\boldsymbol{A}\|_2 \le \beta_A, \quad \|\Delta_{A_S}\|_2 \le \|\Delta_A\|_2, \quad \|\Delta_{D_S}\|_2 \le \|\Delta_D\|_2.$$

Notations: For a subgraph S of graph G, the self-connected adjacency matrix is $\hat{\boldsymbol{A}}_S = \boldsymbol{I} + \boldsymbol{A}_S$, the degree matrix is $\hat{\boldsymbol{D}}_S = \text{diag}(\boldsymbol{1}^\top\hat{\boldsymbol{A}}_S)$, and the normalized self-connected adjacency matrix is $\boldsymbol{U}_S = \hat{\boldsymbol{D}}_S^{-\frac{1}{2}}\hat{\boldsymbol{A}}_S\hat{\boldsymbol{D}}_S^{-\frac{1}{2}}$.

For a subgraph $\tilde{S}$ of graph $\tilde{G}$, we define some notations here. We denote the self-connected adjacency matrix as $\hat{\boldsymbol{A}}'_{\tilde{S}} = \boldsymbol{I} + \tilde{\boldsymbol{A}}_{\tilde{S}}$, the diagonal matrix as $\hat{\boldsymbol{D}}'_{\tilde{S}} = \text{diag}(\boldsymbol{1}^\top\hat{\boldsymbol{A}}'_{\tilde{S}})$, and the normalized self-connected adjacency matrix as $\tilde{\boldsymbol{U}}_{\tilde{S}} = \hat{\boldsymbol{D}}'^{-\frac{1}{2}}_{\tilde{S}}\hat{\boldsymbol{A}}'_{\tilde{S}}\hat{\boldsymbol{D}}'^{-\frac{1}{2}}_{\tilde{S}}$. We also denote $\Delta_{U_S} = \tilde{\boldsymbol{U}}_{\tilde{S}} - \boldsymbol{U}_S$ and $\Delta_{X_S^{(m,l)}} = \tilde{\boldsymbol{X}}_{\tilde{S}}^{(m,l)} - \boldsymbol{X}_S^{(m,l)}$.

Lemma D.5 (Inequalities)**.** *Base on Lemma D.4, for any subgraph S of graph G, the inequalities in the Lemma D.2 still holds for S, shown as follows:*

$$
\begin{aligned}
&\|\boldsymbol{U}_S\|_2 \leq (1+\alpha)^{-1}(1+\beta_A) \\
&\|\tilde{\boldsymbol{U}}_S\|_2 \leq (1+\alpha+\kappa)^{-1}(1+\beta_A+\|\Delta_A\|_2) \\
&\|\Delta_{U_S}\|_2 \leq 2(1+\beta_A)(1+\alpha)^{-1}\|\Delta_D\|_2 + (1+\alpha+\kappa)^{-1}\|\Delta_A\|_2 \\
&\|\boldsymbol{X}_S^{(m,l)}\|_F \leq \rho^l \beta_W^l \beta_X (1+\beta_A)^l (1+\alpha)^{-l} \\
&\|\Delta_{X_S^{(m,l)}}\|_F \leq \rho^l \beta_W^l (1+\beta_A+\|\Delta_A\|_2)^{l-1}(1+\alpha)^{-l}\left[(1+\beta_A+2\|\Delta_A\|_2)\|\Delta_X\|_F + 2l\beta_X(1+\beta_A)\|\Delta_D\|_2\right]
\end{aligned}
\tag{26}
$$

Proof. The proof is mainly based on Lemma D.4.

Similar to (21), we have

$$
\|\boldsymbol{U}_S\|_2 \leq \|\hat{\boldsymbol{D}}_S^{-\frac{1}{2}}\|_2 \|\hat{\boldsymbol{A}}_S\|_2 \|\hat{\boldsymbol{D}}_S^{-\frac{1}{2}}\|_2 \leq \|\hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2 \|\hat{\boldsymbol{A}}\|_2 \|\hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2 \leq (1+\alpha)^{-1}(1+\beta_A). \tag{27}
$$

Similar to (22), we have

$$
\begin{aligned}
\|\tilde{\boldsymbol{U}}_S\|_2 &\leq \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}_{\tilde{S}}\|_2 \|\hat{\boldsymbol{A}}'\|_2 \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}_{\tilde{S}}\|_2 \leq \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2 \|\hat{\boldsymbol{A}}'\|_2 \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2 \\
&\leq (1+\alpha+\kappa)^{-1}(1+\beta_A+\|\Delta_A\|_2).
\end{aligned}
\tag{28}
$$

Similar to (23), we have

$$
\begin{aligned}
\|\Delta_U\|_2 &\leq (\|\hat{\boldsymbol{D}}_S^{-\frac{1}{2}}\|_2 + \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}_{\tilde{S}}\|_2)\|\hat{\boldsymbol{A}}\|_2 \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}_{\tilde{S}} - \hat{\boldsymbol{D}}_S^{-\frac{1}{2}}\|_2 + \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}_{\tilde{S}}\|\Delta_A\|_2 \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}_{\tilde{S}}\|_2 \\
&\leq (\|\hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2 + \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2)\|\hat{\boldsymbol{A}}\|_2 \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}} - \hat{\boldsymbol{D}}^{-\frac{1}{2}}\|_2 + \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2 \|\Delta_A\|_2 \|\hat{\boldsymbol{D}}'^{-\frac{1}{2}}\|_2 \\
&\leq 2(1+\beta_A)(1+\alpha)^{-1}\|\Delta_D\|_2 + (1+\alpha+\kappa)^{-1}\|\Delta_A\|_2
\end{aligned}
\tag{29}
$$

Similar to (24), we have

$$
\begin{aligned}
\|\boldsymbol{X}_S^{(m,l)}\|_F &\leq \rho \|\boldsymbol{U}_S\|_2 \|\boldsymbol{X}_S^{(m,l-1)}\|_F \|\boldsymbol{W}^{(m,l)}\|_2 \\
&\leq \rho \|\boldsymbol{U}\|_2 \|\boldsymbol{X}^{(m,l-1)}\|_F \|\boldsymbol{W}^{(m,l)}\|_2 \\
&\leq \rho^l \beta_W^l \beta_X (1+\beta_A)^l (1+\alpha)^{-l}
\end{aligned}
\tag{30}
$$

Similar to (25), we have

$$
\begin{aligned}
\|\Delta_{X_S^{(m,l)}}\|_F &\leq \rho^2 \beta_W^2 \|\tilde{\boldsymbol{U}}_S\|_2^2 \|\Delta_{X_S^{(l-2)}}\|_F + \rho^2 \beta_W^2 \|\tilde{\boldsymbol{U}}_S\|_2 \|\Delta_{U_S}\|_2 \|\boldsymbol{X}_S^{(l-2)}\|_F + \rho\beta_W \|\Delta_{U_S}\|_2 \|\boldsymbol{X}_S^{(l-1)}\|_F \\
&\leq \rho^2 \beta_W^2 \|\tilde{\boldsymbol{U}}\|_2^2 \|\Delta_{X^{(l-2)}}\|_F + \rho^2 \beta_W^2 \|\tilde{\boldsymbol{U}}\|_2 \|\Delta_U\|_2 \|\boldsymbol{X}^{(l-2)}\|_F + \rho\beta_W \|\Delta_U\|_2 \|\boldsymbol{X}^{(l-1)}\|_F \\
&\leq \rho^l \beta_W^l (1+\beta_A+\|\Delta_A\|_2)^{l-1}(1+\alpha)^{-l}\left[(1+\beta_A+2\|\Delta_A\|_2)\|\Delta_X\|_F + 2l\beta_X(1+\beta_A)\|\Delta_D\|_2\right]
\end{aligned}
\tag{31}
$$

□

Finally, we can prove our theorem of robustness analysis in the main paper using Lemma D.5 as follows.

Proof. Given a pattern sampling set $\mathcal{S}^{(m)}$, we assume the S^* satisfies

$$
S^* = \underset{S \in \mathcal{S}^{(m)}}{\arg\max} \|\Delta_{X_S^{(m,L)}}\|_F.
$$

Since the Lemma D.5 holds for any subgraph S, we have

$$
\|\Delta_{X_{S^*}^{(m,l)}}\|_F \leq \rho^l \beta_W^l (1+\beta_A+\|\Delta_A\|_2)^{l-1}(1+\alpha)^{-l}\left[(1+\beta_A+2\|\Delta_A\|_2)\|\Delta_X\|_F + 2l\beta_X(1+\beta_A)\|\Delta_D\|_2\right]
$$

Then the upper bound of $\|\tilde{\boldsymbol{g}}-\boldsymbol{g}\|$ is given by

$$
\begin{aligned}
\|\tilde{\boldsymbol{g}}-\boldsymbol{g}\| &= \left\|\sum_{m=1}^{M}\lambda_m\left(\tilde{\boldsymbol{z}}^{(m)}-\boldsymbol{z}^{(m)}\right)\right\| \le \sum_{m=1}^{M}\lambda_m\left\|\tilde{\boldsymbol{z}}^{(m)}-\boldsymbol{z}^{(m)}\right\| \\
&= \frac{1}{Q}\sum_{m=1}^{M}\lambda_m\left\|\sum_{S\in\mathcal{S}^{(m)}}F(\tilde{\boldsymbol{A}}_S,\tilde{\boldsymbol{X}}_S;\mathcal{W}^{(m)})-\sum_{S\in\mathcal{S}^{(m)}}F(\boldsymbol{A}_S,\boldsymbol{X}_S;\mathcal{W}^{(m)})\right\| \\
&\le \frac{1}{Q}\sum_{m=1}^{M}\lambda_m\sum_{S\in\mathcal{S}^{(m)}}\left\|F(\tilde{\boldsymbol{A}}_S,\tilde{\boldsymbol{X}}_S;\mathcal{W}^{(m)})-F(\boldsymbol{A}_S,\boldsymbol{X}_S;\mathcal{W}^{(m)})\right\| \\
&= \frac{1}{Q}\sum_{m=1}^{M}\lambda_m\sum_{S\in\mathcal{S}^{(m)}}\frac{1}{n}\left\|\mathbf{1}^\top(\tilde{\boldsymbol{X}}_S^{(m,L)}-\boldsymbol{X}_S^{(m,L)})\right\|_F \\
&\le \frac{1}{Q}\sum_{m=1}^{M}\lambda_m\frac{1}{n}\sum_{S\in\mathcal{S}^{(m)}}\|\mathbf{1}\|\left\|\tilde{\boldsymbol{X}}_S^{(m,L)}-\boldsymbol{X}_S^{(m,L)}\right\|_F \\
&= \frac{1}{Q\sqrt{n}}\sum_{m=1}^{M}\lambda_m\sum_{S\in\mathcal{S}^{(m)}}\left\|\Delta_{X_S^{(m,L)}}\right\|_F \\
&\le \frac{1}{Q\sqrt{n}}\sum_{m=1}^{M}\lambda_m\, Q\left\|\Delta_{X_{S^*}^{(m,L)}}\right\|_F \\
&\le \frac{1}{\sqrt{n}}\rho^l\beta_W^l(1+\beta_A+\|\Delta_A\|_2)^{l-1}(1+\alpha)^{-l}\left[(1+\beta_A+2\|\Delta_A\|_2)\|\Delta_X\|_F+2L\beta_X(1+\beta_A)\|\Delta_D\|_2\right]
\end{aligned}
\tag{32}
$$

□

E Proof for Generalization Analysis of Supervised Loss

Before providing our theorem, we need to provide the classification loss function f_c.

Classification loss function f_c: We use a linear classifier with parameter $\boldsymbol{W}_C \in \mathbb{R}^{d\times C}$ and use softmax as the activation function as the classification function f_c, i.e., $\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{g}\boldsymbol{W}_C)$. We suppose that $\|\boldsymbol{W}_C\|_2 \le \beta_C$.

Then the classification loss is as follows

$$
\ell_{\text{CE}}(\boldsymbol{\lambda},\mathbb{W}) = \text{cross-entropy}(\boldsymbol{y},\hat{\boldsymbol{y}}) = \text{cross-entropy}(\boldsymbol{y},\text{softmax}(\boldsymbol{g}\boldsymbol{W}_C)). \tag{33}
$$

To simplify the proof, we rewrite supervised loss $\ell_{\text{CE}}(\boldsymbol{\lambda},\mathbb{W})$ function as

$$
\varphi(\boldsymbol{g}\boldsymbol{W}_C) := \text{cross-entropy}(\boldsymbol{y},\hat{\boldsymbol{y}}) = \text{cross-entropy}(\boldsymbol{y},\text{softmax}(\boldsymbol{g}\boldsymbol{W}_C)).
$$

Lemma E.1. *Let $\boldsymbol{v}$ be a vector, there exits a positive constant τ such that $\varphi(\boldsymbol{v})$ is a τ-Lipschitz continuous function.*

Generalization Error Let $\mathcal{D} := \{G_1, ..., G_{|\mathcal{D}|}\}$ be the training data. By removing the i-th graph of $\mathcal{D}$, we have $\mathcal{D}^{\setminus i} = \{G_1, ..., G_{i-1}, G_{i+1}, ..., G_{|\mathcal{D}|-1}\}$. Let $\boldsymbol{\lambda}_{\mathcal{D}}$ and $\bar{\mathbb{W}}_{\mathcal{D}} := \{\boldsymbol{W}_C, \boldsymbol{W}_{\mathcal{D}}^{(m,l)},\ \forall\, m \in [M],\ l \in [L]\}$ be the parameters trained on $\mathcal{D}$. Let $\boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}$ and $\bar{\mathbb{W}}_{\mathcal{D}^{\setminus i}} := \{\boldsymbol{W}_{C^{\setminus i}}, \boldsymbol{W}_{\mathcal{D}^{\setminus i}}^{(m,l)},\ \forall\, m \in [M],\ l \in [L]\}$ be the parameters trained on $\mathcal{D}^{\setminus i}$. Then our goal is to find a η such that

$$
|\ell_{\text{CE}}(\boldsymbol{\lambda}_{\mathcal{D}},\bar{\mathbb{W}}_{\mathcal{D}};G) - \ell_{\text{CE}}(\boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}},\bar{\mathbb{W}}_{\mathcal{D}^{\setminus i}};G)| \le \eta \tag{34}
$$

Theorem E.2. *Given a graph G, let $\boldsymbol{g}$ be the graph representations learned with parameter $\boldsymbol{\lambda}_{\mathcal{D}}$ and $\bar{\mathbb{W}}_{\mathcal{D}}$ and $\boldsymbol{g}^{\setminus i}$ be the graph representations learned with parameter $\boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}$ and $\bar{\mathbb{W}}_{\mathcal{D}^{\setminus i}}$.*

To simplify the proof, we denote that $\hat{\beta}_W = \max(\hat{\beta}_{W\mathcal{D}}, \hat{\beta}_{W\mathcal{D}^{\setminus i}})$, where

$$
\hat{\beta}_{W\mathcal{D}} = \max_{m\in[M],l\in[L]}\|W_{\mathcal{D}}^{(m,l)}\|_2,\ \ \textit{and}\ \ \hat{\beta}_{W\mathcal{D}^{\setminus i}} = \max_{m\in[M],l\in[L]}\|W_{\mathcal{D}^{\setminus i}}^{(m,l)}\|_2.
$$

We also denote that

$$\hat{\beta}_{\Delta W} = \max_{m\in[M],l\in[L]} \|\mathcal{W}_{\mathcal{D}}^{(m,l)} - \mathcal{W}_{\mathcal{D}^{\setminus i}}^{(m,l)}\|_2.$$

Then we have

$$\eta = \frac{\tau}{\sqrt{n}}\rho^L \hat{\beta}_W^{L-1}\beta_X(1+\beta_A)^L(1+\alpha)^{-L}\left[\hat{\beta}_W\|\boldsymbol{W}_C - \boldsymbol{W}_{C^{\setminus i}}\|_2 + \|\boldsymbol{W}_{C^{\setminus i}}\|_2\left(\hat{\beta}_W\|\boldsymbol{\lambda}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\| + L\hat{\beta}_{\Delta W}\|\boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\|\right)\right]$$

Proof. We provide two lemmas used in our proof

Lemma E.3. $\|\boldsymbol{g}\| \leq \frac{1}{\sqrt{n}}\rho^L\hat{\beta}_W^L\beta_X(1+\beta_A)^L(1+\alpha)^{-L}$

Lemma E.4.

$$\|\boldsymbol{g} - \boldsymbol{g}^{\setminus i}\| \leq \frac{1}{\sqrt{n}}\rho^L\hat{\beta}_W^{L-1}\beta_X(1+\beta_A)^L(1+\alpha)^{-L}\left(\hat{\beta}_W\|\boldsymbol{\lambda}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\| + L\hat{\beta}_{\Delta W}\|\boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\|\right)$$

The main proof of our Theorem

$$\begin{aligned}
&|\ell_{\text{CE}}(\boldsymbol{\lambda}_{\mathcal{D}}, \bar{\mathbb{W}}_{\mathcal{D}}; G) - \ell_{\text{CE}}(\boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}, \bar{\mathbb{W}}_{\mathcal{D}^{\setminus i}}; G)| = \|\varphi(\boldsymbol{g}^{\setminus i}\boldsymbol{W}_{C^{\setminus i}}) - \varphi(\boldsymbol{g}\boldsymbol{W}_C)\| \\
\leq&\tau\|\boldsymbol{g}\boldsymbol{W}_C - \boldsymbol{g}^{\setminus i}\boldsymbol{W}_{C^{\setminus i}}\| \\
=&\tau\|\boldsymbol{g}\boldsymbol{W}_C - \boldsymbol{g}\boldsymbol{W}_{C^{\setminus i}} + \boldsymbol{g}\boldsymbol{W}_{C^{\setminus i}} - \boldsymbol{g}^{\setminus i}\boldsymbol{W}_{C^{\setminus i}}\| \\
\leq&\tau\|\boldsymbol{g}\|\|\boldsymbol{W}_C - \boldsymbol{W}_{C^{\setminus i}}\|_2 + \tau\|\boldsymbol{g} - \boldsymbol{g}^{\setminus i}\|\|\boldsymbol{W}_{C^{\setminus i}}\|_2 \\
\leq&\tau\|\boldsymbol{W}_C - \boldsymbol{W}_{C^{\setminus i}}\|_2\frac{1}{\sqrt{n}}\rho^L\hat{\beta}_W^L\beta_X(1+\beta_A)^L(1+\alpha)^{-L} \\
&+\tau\|\boldsymbol{W}_{C^{\setminus i}}\|_2\frac{1}{\sqrt{n}}\rho^L\hat{\beta}_W^{L-1}\beta_X(1+\beta_A)^L(1+\alpha)^{-L}\left(\hat{\beta}_W\|\boldsymbol{\lambda}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\| + L\hat{\beta}_{\Delta W}\|\boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\|\right) \\
=&\frac{\tau}{\sqrt{n}}\rho^L\hat{\beta}_W^{L-1}\beta_X(1+\beta_A)^L(1+\alpha)^{-L}\left[\hat{\beta}_W\|\boldsymbol{W}_C - \boldsymbol{W}_{C^{\setminus i}}\|_2 + \|\boldsymbol{W}_{C^{\setminus i}}\|_2\left(\hat{\beta}_W\|\boldsymbol{\lambda}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\| + L\hat{\beta}_{\Delta W}\|\boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\|\right)\right]
\end{aligned} \tag{35}$$

Since $\sum_{i=1}^{M}\lambda_i \leq 1$ and $\lambda_i \geq 0$, we have $\|\boldsymbol{\lambda}\| \leq 1$ and $\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\| \leq 2$. This finished the proof. □

E.1 Proof for Lemmas

Lemma E.5. *Let $\boldsymbol{v}$ be a vector, there exits a positive constant τ such that $\varphi(\boldsymbol{v})$ is a τ-Lipschitz continuous function.*

Proof. **Step 1: Softmax is Lipschitz** The softmax function is known to be Lipschitz continuous. Specifically, there exists a constant K such that:

$$\|\text{softmax}(v) - \text{softmax}(w)\|_1 \leq L_1\|v - w\|_2,$$

where $\|\cdot\|_1$ is the ℓ_1-norm and $\|\cdot\|_2$ is the ℓ_2-norm. For the ℓ_1-norm, L_1 can be bounded by 1, but generally, for different norms, the exact Lipschitz constant might vary.

Step 2: Cross-Entropy is Lipschitz on the Simplex Given $\mathbf{q} = \text{softmax}(v)$ and $\mathbf{r} = \text{softmax}(w)$, we need to check the Lipschitz continuity of the cross-entropy loss function with respect to these distributions:

$$|\text{cross-entropy}(\mathbf{p}, \mathbf{q}) - \text{cross-entropy}(\mathbf{p}, \mathbf{r})| \leq L_2\|\mathbf{q} - \mathbf{r}\|.$$

The cross-entropy loss is a convex function and it is smooth with respect to the probability distributions $\mathbf{q}$ and $\mathbf{r}$. Given the boundedness of the probability values (since $\mathbf{q}$ and $\mathbf{r}$ lie in the probability simplex), the gradient of the cross-entropy loss is also bounded.

Combining Steps Since both the softmax function and the cross-entropy loss function are Lipschitz continuous, their composition will also be Lipschitz continuous. Therefore, there exists a constant $\tau = L_1L_2$ such that:

$$|\varphi(v) - \varphi(w)| \leq \tau\|v - w\|.$$

Hence, $\varphi(v) = \text{cross-entropy}(\text{softmax}(v))$ is τ-Lipschitz continuous. □

Lemma E.6. $\|\boldsymbol{g}\| \leq \frac{1}{\sqrt{n}}\rho^L \hat{\beta}_W^L \beta_X (1+\beta_A)^L (1+\alpha)^{-L}$

Proof. Given a pattern sampling set $\mathcal{S}^{(m)}$, we assume the S^* satisfies

$$S^* = \underset{S\in\mathcal{S}^{(m)}}{\arg\max} \|\boldsymbol{X}_S^{(L)}\|_F.$$

Since the Lemma D.5 holds for any subgraph S, then we have

$$\|\boldsymbol{X}_{S^*}^{(m,l)}\|_F \leq \rho^l \hat{\beta}_W^l \beta_X (1+\beta_A)^l (1+\alpha)^{-l}.$$

Then, we have

$$\begin{aligned}
\|\boldsymbol{g}\| &= \|\sum_{m=1}^M \lambda_m\, \boldsymbol{z}^{(m)}\| \leq \sum_{m=1}^M \lambda_m\, \|\boldsymbol{z}^{(m)}\| \\
&= \frac{1}{Q}\sum_{m=1}^M \lambda_m \,\| \sum_{S\in\mathcal{S}^{(m)}} F(\boldsymbol{A}_S, \boldsymbol{X}_S; \mathcal{W}^{(m)})\| \\
&\leq \frac{1}{Q}\sum_{m=1}^M \lambda_m \sum_{S\in\mathcal{S}^{(m)}} \|F(\boldsymbol{A}_S, \boldsymbol{X}_S; \mathcal{W}^{(m)})\| \\
&= \frac{1}{Q}\sum_{m=1}^M \lambda_m \sum_{S\in\mathcal{S}^{(m)}} \frac{1}{n}\|\boldsymbol{1}^\top(\boldsymbol{X}_S^{(m,L)})\|_F \\
&\leq \frac{1}{Q}\sum_{m=1}^M \lambda_m \frac{1}{n}\sum_{S\in\mathcal{S}^{(m)}} \|\boldsymbol{1}\|_2\|\boldsymbol{X}_S^{(m,L)}\|_F \\
&= \frac{1}{Q\sqrt{n}}\sum_{m=1}^M \lambda_m \sum_{S\in\mathcal{S}^{(m)}} \|\boldsymbol{X}_S^{(m,L)}\|_F \\
&\leq \frac{1}{\sqrt{n}}\sum_{m=1}^M \lambda_m \,\|\boldsymbol{X}_{S^*}^{(m,L)}\|_F \\
&\leq \frac{1}{\sqrt{n}}\rho^L \hat{\beta}_W^L \beta_X (1+\beta_A)^L (1+\alpha)^{-L}
\end{aligned} \tag{36}$$

□

Lemma E.7.

$$\|\boldsymbol{g} - \boldsymbol{g}^{\setminus i}\| \leq \frac{1}{\sqrt{n}}\rho^L \hat{\beta}_W^{L-1} \beta_X (1+\beta_A)^L (1+\alpha)^{-L} \left(\hat{\beta}_W \|\boldsymbol{\lambda}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\| + L\hat{\beta}_{\Delta W}\|\boldsymbol{\lambda}_{\mathcal{D}^{\setminus i}}\|\right)$$

Proof. To simplify the proof, we denote

$$\begin{aligned}
\hat{\beta}_W &= \max\{\max_{m\in[M],l\in[L]} \|\boldsymbol{W}_{\mathcal{D}}^{(m,l)}\|_2, \max_{m\in[M],l\in[L]} \|\boldsymbol{W}_{\mathcal{D}^{\setminus i}}^{(m,l)}\|_2\} \\
\hat{\beta}_{\Delta W} &= \max_{m\in[M],l\in[L]} \|\mathcal{W}_{\mathcal{D}}^{(m,l)} - \mathcal{W}_{\mathcal{D}^{\setminus i}}^{(m,l)}\|_2.
\end{aligned} \tag{37}$$

Let $\boldsymbol{X}_{S\mathcal{D}}^{(m,l)}$ be the embedding features of the l-th layer GCN with the parameter $\mathcal{W}_{\mathcal{D}}^{(m)}$ learned from dataset $\mathcal{D}$. Let $\boldsymbol{X}_{S\mathcal{D}^{\setminus i}}^{(m,l)}$ be the embedding features of the l-th layer GCN with the parameter $\mathcal{W}_{\mathcal{D}^{\setminus i}}^{(m)}$ learned from dataset $\mathcal{D}^{\setminus i}$.

We denote $\boldsymbol{Z}_{\mathcal{D}} = [\boldsymbol{z}_{\mathcal{D}}^{(1)}, ..., \boldsymbol{z}_{\mathcal{D}}^{(m)}]^\top$ and $\boldsymbol{Z}_{\mathcal{D}^{\setminus i}} = [\boldsymbol{z}_{\mathcal{D}^{\setminus i}}^{(1)}, ..., \boldsymbol{z}_{\mathcal{D}^{\setminus i}}^{(m)}]^\top$. Let

$$q_1 = \underset{m\in[M]}{\arg\max} \|\boldsymbol{z}_{\mathcal{D}}^{(m)}\|, \quad q_2 = \underset{m\in[M]}{\arg\max} \|\boldsymbol{z}_{\mathcal{D}}^{(l_2)} - \boldsymbol{z}_{\mathcal{D}^{\setminus i}}^{(l_2)}\|.$$

Then we have

$$\|\boldsymbol{Z}_{\mathcal{D}}\|_2 \leq \|\boldsymbol{z}_{\mathcal{D}}^{(q_1)}\|, \quad \|\boldsymbol{Z}_{\mathcal{D}} - \boldsymbol{Z}_{\mathcal{D}^{\setminus i}}\|_2 \leq \|\boldsymbol{z}_{\mathcal{D}}^{(q_2)} - \boldsymbol{z}_{\mathcal{D}^{\setminus i}}^{(q_2)}\|.$$

Similar to (36), we have

$$\|\boldsymbol{z}_{\mathcal{D}}^{(q_1)}\| \leq \frac{1}{\sqrt{n}}\rho^L \hat{\beta}_W^L \beta_X (1+\beta_A)^L (1+\alpha)^{-L} \tag{38}$$

Denote $\Delta_{X_{S\mathcal{D}}^{(q_2,l)}} := \boldsymbol{X}_{S\mathcal{D}}^{(q_2,l)} - \boldsymbol{X}_{S\mathcal{D}\backslash i}^{(q_2,l)}$, then, similar to inequality (25) we have

$$\begin{aligned}
\|\Delta_{X_{S\mathcal{D}}^{(q_2,l)}}\|_F &= \|\sigma(\boldsymbol{U}_S \boldsymbol{X}_{S\mathcal{D}}^{(q_2,l-1)} \mathcal{W}_{\mathcal{D}}^{(q_2)}) - \sigma(\boldsymbol{U}_S \boldsymbol{X}_{S\mathcal{D}\backslash i}^{(q_2,l)} \mathcal{W}_{\mathcal{D}\backslash i}^{(q_2)})\|_F \\
&\leq \rho\|\boldsymbol{U}_S\|_2 \|\boldsymbol{X}_{S\mathcal{D}}^{(q_2,l-1)} W_{\mathcal{D}}^{(q_2,l-1)} - \boldsymbol{X}_{S\mathcal{D}\backslash i}^{(q_2,l-1)} W_{\mathcal{D}\backslash i}^{(q_2,l-1)}\|_F \\
&\leq \rho\|\boldsymbol{U}_S\|_2 \|\boldsymbol{X}_{S\mathcal{D}}^{(q_2,l-1)} W_{\mathcal{D}}^{(q_2,l-1)} - \boldsymbol{X}_{S\mathcal{D}}^{(q_2,l-1)} W_{\mathcal{D}\backslash i}^{(q_2,l-1)} + \boldsymbol{X}_{S\mathcal{D}}^{(q_2,l-1)} W_{\mathcal{D}\backslash i}^{(q_2,l-1)} - \boldsymbol{X}_{S\mathcal{D}\backslash i}^{(q_2,l-1)} W_{\mathcal{D}\backslash i}^{(q_2,l-1)}\|_F \\
&\leq \rho\|\boldsymbol{U}_S\|_2 \|\boldsymbol{X}_{S\mathcal{D}}^{(q_2,l-1)} (W_{\mathcal{D}}^{(q_2,l-1)} - W_{\mathcal{D}\backslash i}^{(q_2,l-1)}) + (\boldsymbol{X}_{S\mathcal{D}}^{(q_2,l-1)} - \boldsymbol{X}_{S\mathcal{D}\backslash i}^{(q_2,l-1)}) W_{\mathcal{D}\backslash i}^{(q_2,l-1)}\|_F \\
&\leq \rho\|\boldsymbol{U}_S\|_2 (\|\boldsymbol{X}_{S\mathcal{D}}^{(q_2,l-1)}\|_F \|W_{\mathcal{D}}^{(q_2,l-1)} - W_{\mathcal{D}\backslash i}^{(q_2,l-1)}\|_2 + \|\boldsymbol{X}_{S\mathcal{D}}^{(q_2,l-1)} - \boldsymbol{X}_{S\mathcal{D}\backslash i}^{(q_2,l-1)}\|_F \|W_{\mathcal{D}\backslash i}^{(q_2,l-1)}\|_2) \\
&= \rho\|\boldsymbol{U}_S\|_2 \hat{\beta}_W \|\Delta_{X_{S\mathcal{D}}^{(q_2,l-1)}}\|_F + \rho\|\boldsymbol{U}_S\|_2 \hat{\beta}_{\Delta W} \|\boldsymbol{X}_{S\mathcal{D}}^{(q_2,l-1)}\|_F \\
&\leq \rho^l \|\boldsymbol{U}_S\|_2^l \hat{\beta}_W^l \|\Delta_{X_{S\mathcal{D}}^{(q_2,0)}}\|_F + \sum_{k=1}^{l} \rho^k \|\boldsymbol{U}_S\|_2^k \hat{\beta}_W^{k-1} \hat{\beta}_{\Delta W} \|\boldsymbol{X}_{S\mathcal{D}}^{(q_2,l-k)}\|_F
\end{aligned} \tag{39}$$

where $\|\Delta_{X_{S\mathcal{D}}^{(q_2,0)}}\|_F = \|\boldsymbol{X}_S - \boldsymbol{X}_S\|_F = 0$. We can directly use the inequality (24), such that

$$\|\boldsymbol{X}_{S\mathcal{D}}^{(m,l)}\|_F \leq \rho^l \hat{\beta}_W^l \beta_X (1+\beta_A)^l (1+\alpha)^{-l} \tag{40}$$

Thus, we continue the proof

$$\begin{aligned}
\|\Delta_{X_{S\mathcal{D}}^{(q_2,l)}}\|_F &\leq \rho^l \|\boldsymbol{U}_S\|_2^l \hat{\beta}_W^l \|\Delta_{X_{S\mathcal{D}}^{(q_2,0)}}\|_F + \sum_{k=1}^{l} \rho^k \|\boldsymbol{U}_S\|_2^k \hat{\beta}_W^{k-1} \hat{\beta}_{\Delta W} \|\boldsymbol{X}_{S\mathcal{D}}^{(q_2,l-k)}\|_F \\
&\leq l\rho^l (1+\alpha)^{-l} (1+\beta_A)^l \hat{\beta}_W^{l-1} \hat{\beta}_{\Delta W} \beta_X
\end{aligned} \tag{41}$$

Also similar to (D.5), we have

$$\begin{aligned}
\|\boldsymbol{z}_{\mathcal{D}}^{(q_2)} - \boldsymbol{z}_{\mathcal{D}\backslash i}^{(q_2)}\| &= \|F(\boldsymbol{A}_S, \boldsymbol{X}_S; \mathcal{W}_{\mathcal{D}}^{(q_2)}) - F(\boldsymbol{A}_S, \boldsymbol{X}_S; \mathcal{W}_{\mathcal{D}\backslash i}^{(q_2)})\| \\
&= \frac{1}{n}\|\mathbf{1}^\top(\boldsymbol{X}_{S\mathcal{D}}^{(q_2,L)}) - \mathbf{1}^\top(\boldsymbol{X}_{S\mathcal{D}\backslash i}^{(q_2,L)})\| \\
&= \frac{1}{\sqrt{n}}\|\boldsymbol{X}_{S\mathcal{D}}^{(q_2,L)} - \boldsymbol{X}_{S\mathcal{D}\backslash i}^{(q_2,L)}\|_F = \frac{1}{\sqrt{n}}\|\Delta_{X_{S\mathcal{D}}^{(q_2,L)}}\|_F \\
&\leq \frac{L}{\sqrt{n}}\rho^L (1+\alpha)^{-L} (1+\beta_A)^L \hat{\beta}_W^{L-1} \hat{\beta}_{\Delta W} \beta_X
\end{aligned} \tag{42}$$

Finally, we have

$$\begin{aligned}
\|\boldsymbol{g} - \boldsymbol{g}^{\backslash i}\| &= \|\boldsymbol{\lambda}_{\mathcal{D}}^\top \boldsymbol{Z}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}\backslash i}^\top \boldsymbol{Z}_{\mathcal{D}\backslash i}\| \\
&= \|\boldsymbol{\lambda}_{\mathcal{D}}^\top \boldsymbol{Z}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}\backslash i}^\top \boldsymbol{Z}_{\mathcal{D}} + \boldsymbol{\lambda}_{\mathcal{D}\backslash i}^\top \boldsymbol{Z}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}\backslash i}^\top \boldsymbol{Z}_{\mathcal{D}\backslash i}\| \\
&= \|(\boldsymbol{\lambda}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}\backslash i})^\top \boldsymbol{Z}_{\mathcal{D}} + \boldsymbol{\lambda}_{\mathcal{D}\backslash i}^\top (\boldsymbol{Z}_{\mathcal{D}} - \boldsymbol{Z}_{\mathcal{D}\backslash i})\| \\
&\leq \|\boldsymbol{\lambda}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}\backslash i}\| \|\boldsymbol{Z}_{\mathcal{D}}\|_2 + \|\boldsymbol{\lambda}_{\mathcal{D}\backslash i}\| \|\boldsymbol{Z}_{\mathcal{D}} - \boldsymbol{Z}_{\mathcal{D}\backslash i}\|_2 \\
&\leq \|\boldsymbol{\lambda}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}\backslash i}\| \|\boldsymbol{z}_{\mathcal{D}}^{(q_1)}\| + \|\boldsymbol{\lambda}_{\mathcal{D}\backslash i}\| \|\boldsymbol{z}_{\mathcal{D}}^{(q_2)} - \boldsymbol{z}_{\mathcal{D}\backslash i}^{(q_2)}\| \\
&\leq \|\boldsymbol{\lambda}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}\backslash i}\| \frac{1}{\sqrt{n}}\rho^L \hat{\beta}_W^L \beta_X (1+\beta_A)^L (1+\alpha)^{-L} \\
&\quad + \|\boldsymbol{\lambda}_{\mathcal{D}\backslash i}\| \frac{L}{\sqrt{n}}\rho^L (1+\alpha)^{-L} (1+\beta_A)^L \hat{\beta}_W^{L-1} \hat{\beta}_{\Delta W} \beta_X \\
&= \frac{1}{\sqrt{n}}\rho^L \hat{\beta}_W^{L-1} \beta_X (1+\beta_A)^L (1+\alpha)^{-L} \left(\hat{\beta}_W \|\boldsymbol{\lambda}_{\mathcal{D}} - \boldsymbol{\lambda}_{\mathcal{D}\backslash i}\| + L\hat{\beta}_{\Delta W} \|\boldsymbol{\lambda}_{\mathcal{D}\backslash i}\|\right)
\end{aligned} \tag{43}$$

□

F More Experimental Results

In this section, we present additional experiments and supplementary figures.

F.1 Evaluating the Ensemble Kernel (PXGL-EGK)

Here, we compare our ensemble kernel (PXGL-EGK) as defined in Definition 3.3 with individual kernels $K_{\mathcal{P}}$. We report the results as follows. Specifically, we use three pattern counting kernels in the ensemble method: Random Walk (RW) kernels (Borgwardt et al., 2005; Gärtner et al., 2003), Sub-tree kernels (Da San Martino et al., 2012; Smola & Vishwanathan, 2002), and Graphlet kernels (Pržulj, 2007). Since graph kernels are unsupervised learning methods, we compare the clustering accuracy and Normalized Mutual Information (NMI) of each kernel, as shown in Table 6. The result shows that PXGL-EGK outperform each individual kernels it used.

Table 6: ACC and NMI of Graph Clustering. The best ACC is **bold** and the best NMI is green.

Method	Metric	MUTAG	PROTEINS	DD	IMDB-B
RW	ACC	0.743 ± 0.052	0.712 ± 0.021	0.516 ± 0.015	0.658 ± 0.014
	NMI	0.238 ± 0.016	0.268 ± 0.016	0.187 ± 0.002	0.266 ± 0.019
Sub-tree	ACC	0.729 ± 0.013	0.692 ± 0.027	0.542 ± 0.016	0.612 ± 0.018
	NMI	0.195 ± 0.047	0.151 ± 0.028	0.229 ± 0.015	0.242 ± 0.013
Graphlet	ACC	0.735 ± 0.026	0.636 ± 0.017	0.568 ± 0.013	0.614 ± 0.012
	NMI	0.214 ± 0.019	0.154 ± 0.026	0.285 ± 0.011	0.214 ± 0.025
PXGL-EGK	ACC	**0.761** ± **0.025**	**0.721** ± **0.028**	**0.572** ± **0.025**	**0.672** ± **0.023**
	NMI	0.328 ± 0.046	0.321 ± 0.019	0.296 ± 0.013	0.310 ± 0.021

F.2 Sensitivity Analysis

Sensitivity of PXGL-GNN to Q Here we use the MUTAG dataset to show the sensitivity of accuracy and time cost to the number of samples Q for each pattern. We see that the time cost is roughly linear with Q and the accuracy is not sensitive to Q when it is larger than 5.

Table 7: Impact of sampling number Q on MUTAG dataset (20 epochs, 7 patterns)

Q	3	5	7	10	15
Accuracy (%)	87.63 ± 1.42	94.87 ± 2.26	94.26 ± 2.17	95.35 ± 1.89	95.33 ± 2.48
Training Time (s)	636s	877s	1035s	1563s	2351s

Sensitivity of PXGL-GNN to L In the following table, we use three datasets to show the accuracy of the graph classification of our PXGL-EGK model with different number of layers L. The results reveal that the model performs best at $L = 5$. With fewer layers, the model lacks sufficient capacity for representation; with more layers, the model is too complex and has overfitting performances. This is consistent with our theoretical analysis, since when the model is complex the gap between training error and the testing error becomes large.

Table 8: Impact of the number of layers of GNN

Model	L = 1	L = 3	L = 5	L = 7	L = 9
MUTAG	81.44 ± 1.29	86.73 ± 2.78	94.87 ± 2.26	91.25 ± 1.14	89.66 ± 1.15
PROTEINS	62.17 ± 1.53	67.22 ± 1.16	78.23 ± 2.46	73.21 ± 1.98	71.07 ± 1.63
DD	75.36 ± 1.21	79.35 ± 1.20	86.54 ± 1.95	82.36 ± 1.24	82.17 ± 1.54

Sensitivity of PXGL-GNN to pattern combination The following table shows the classification accuracy given by PXGL-GNN with different combinations of patterns on the MUTAG dataset. We see that by including more patterns, the classification accuracy tends to be higher.

Table 9: Classification accuracy of PXGL-GNN with different pattern combinations on MUTAG dataset. The best performance is shown in **bold**.

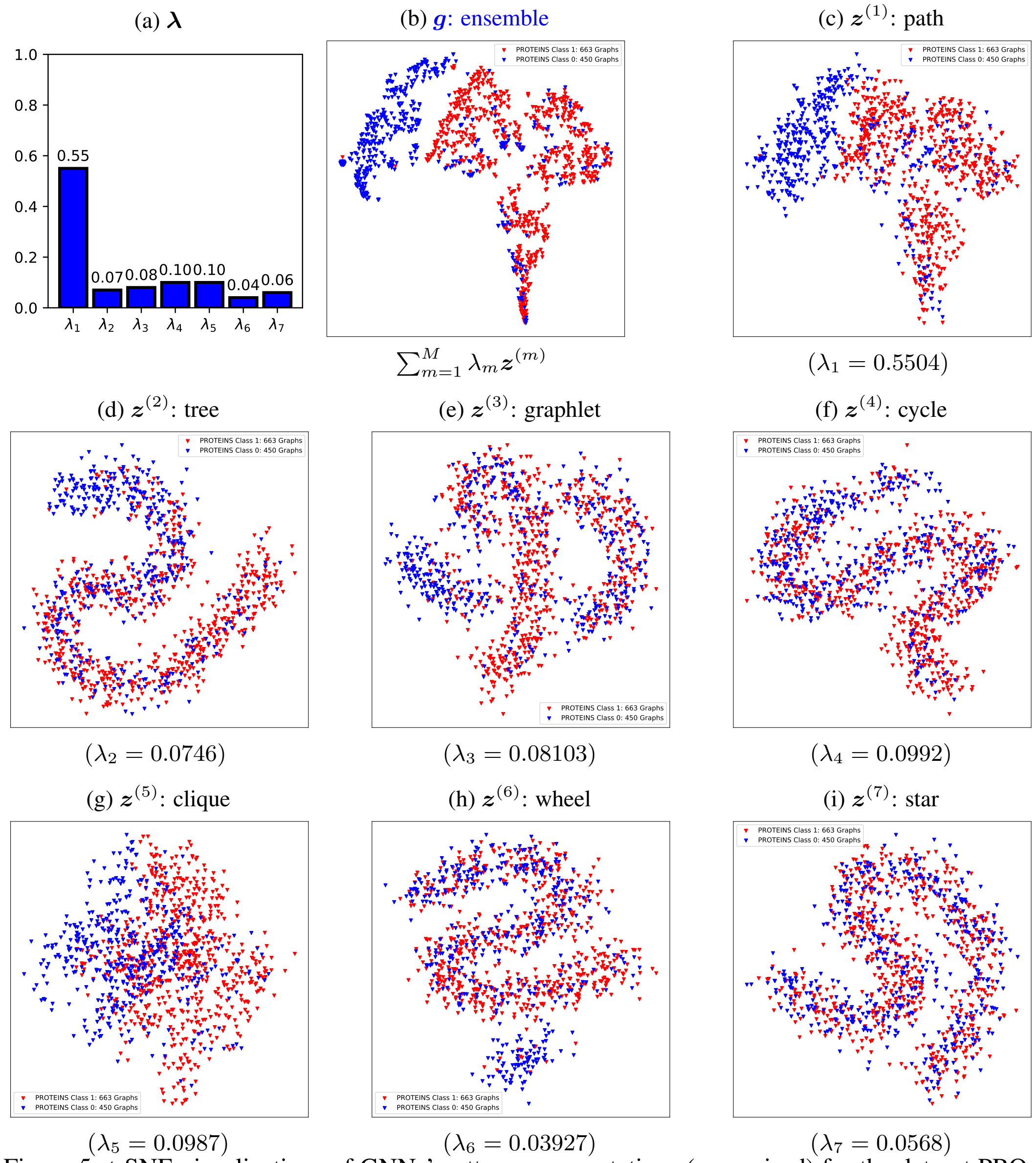
Pattern Combinations	Accuracy (%)	λ weights
Paths only	80.47 ± 1.24	1.0
Trees only	86.39 ± 2.73	1.0
Cycles only	89.24 ± 1.76	1.0
Paths + Trees	87.11 ± 2.93	0.274 / 0.716
Paths + Cycles	91.62 ± 1.14	0.207 / 0.793
Trees + Cycles	92.31 ± 2.65	0.325 / 0.675
All Patterns	94.87 ± 2.26	0.095/0.046/0.654

F.3 Supervised Learning

In this section, we provide the figures to visualize weight vector $\boldsymbol{\lambda}$, graph representation $\boldsymbol{g}$ and pattern representations $\boldsymbol{z}^{(m)}$ learned by solving the supervised loss (11).

F.4 Unsupervised Learning

In this section, we provide the figures to visualize weight vector $\boldsymbol{\lambda}$, graph representation $\boldsymbol{g}$ and pattern representations $\boldsymbol{z}^{(m)}$ learned by solving the unsupervised loss (10).



(a) $\boldsymbol{\lambda}$

(b) $\boldsymbol{g}$: ensemble

$\sum_{m=1}^{M} \lambda_m \boldsymbol{z}^{(m)}$

(c) $\boldsymbol{z}^{(1)}$: path

$(\lambda_1 = 0.5504)$

(d) $\boldsymbol{z}^{(2)}$: tree

$(\lambda_2 = 0.0746)$

(e) $\boldsymbol{z}^{(3)}$: graphlet

$(\lambda_3 = 0.08103)$

(f) $\boldsymbol{z}^{(4)}$: cycle

$(\lambda_4 = 0.0992)$

(g) $\boldsymbol{z}^{(5)}$: clique

$(\lambda_5 = 0.0987)$

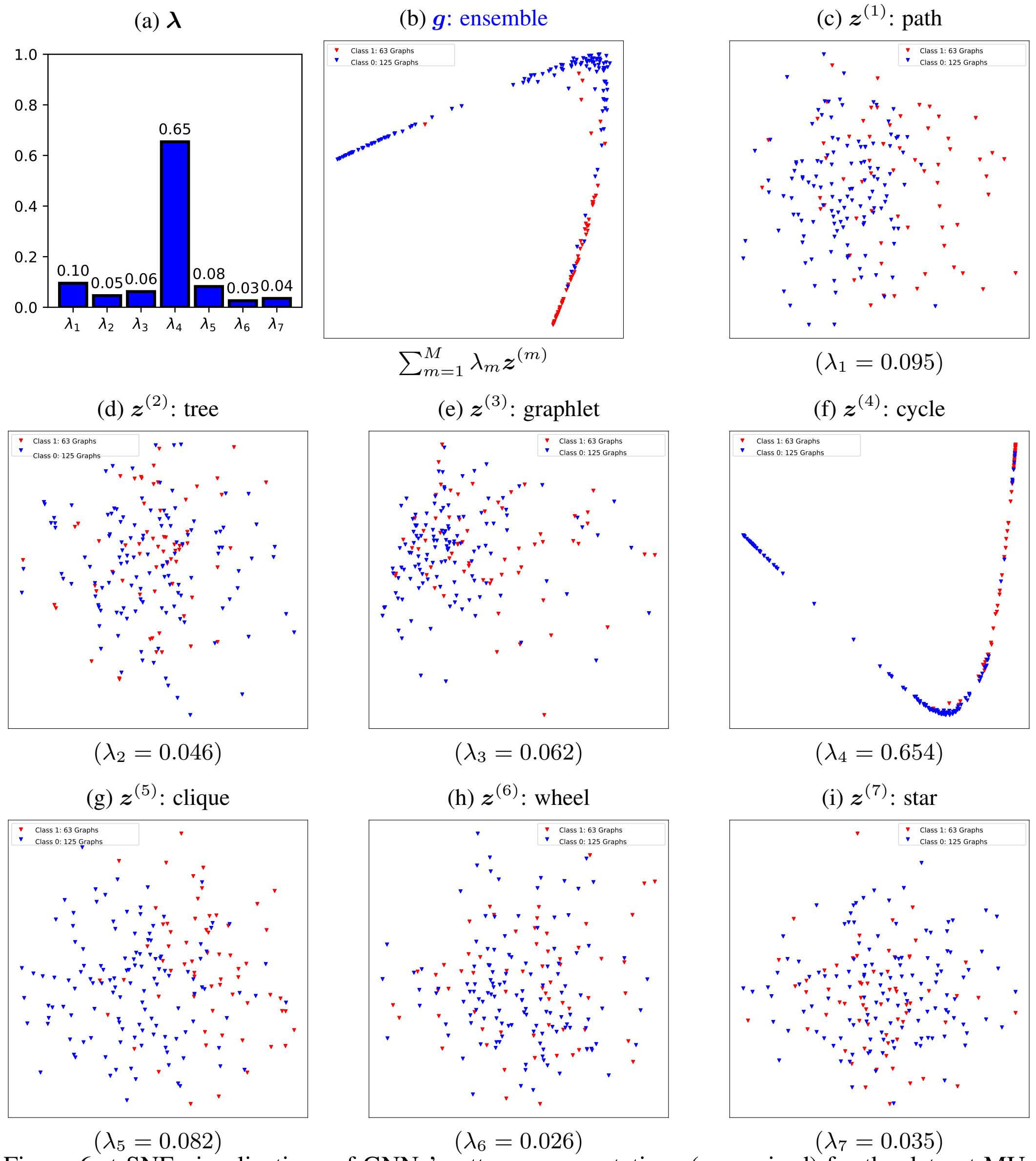
(h) $\boldsymbol{z}^{(6)}$: wheel

$(\lambda_6 = 0.03927)$

(i) $\boldsymbol{z}^{(7)}$: star

$(\lambda_7 = 0.0568)$

Figure 5: t-SNE visualizations of GNNs' pattern representations (supervised) for the dataset PROTEINS.



$\sum_{m=1}^{M} \lambda_m \boldsymbol{z}^{(m)}$

$(\lambda_1 = 0.095)$

$(\lambda_2 = 0.046)$

$(\lambda_3 = 0.062)$

$(\lambda_4 = 0.654)$

$(\lambda_5 = 0.082)$

$(\lambda_6 = 0.026)$

$(\lambda_7 = 0.035)$

Figure 6: t-SNE visualizations of GNNs' pattern representations (supervised) for the dataset MUTAG.

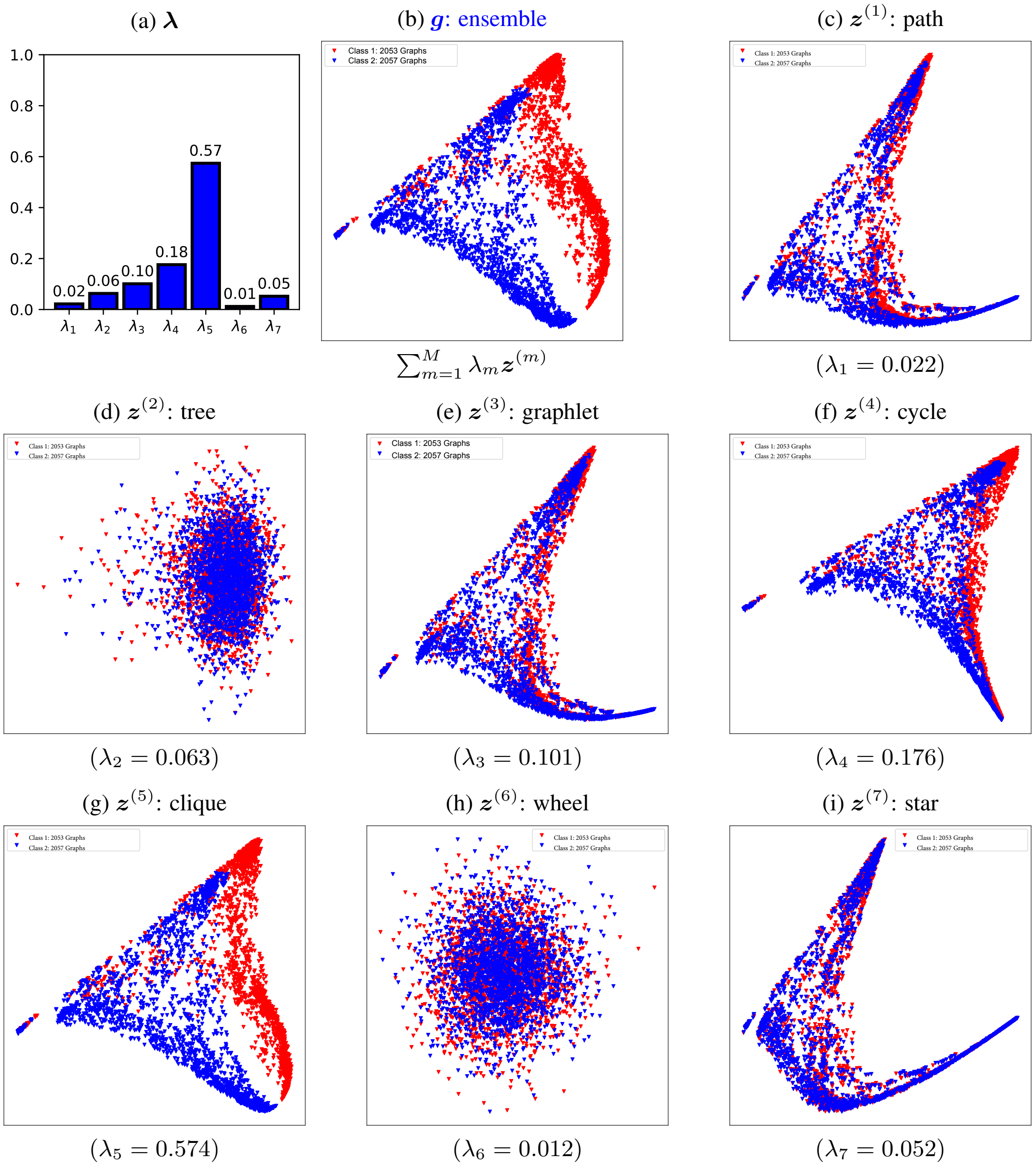
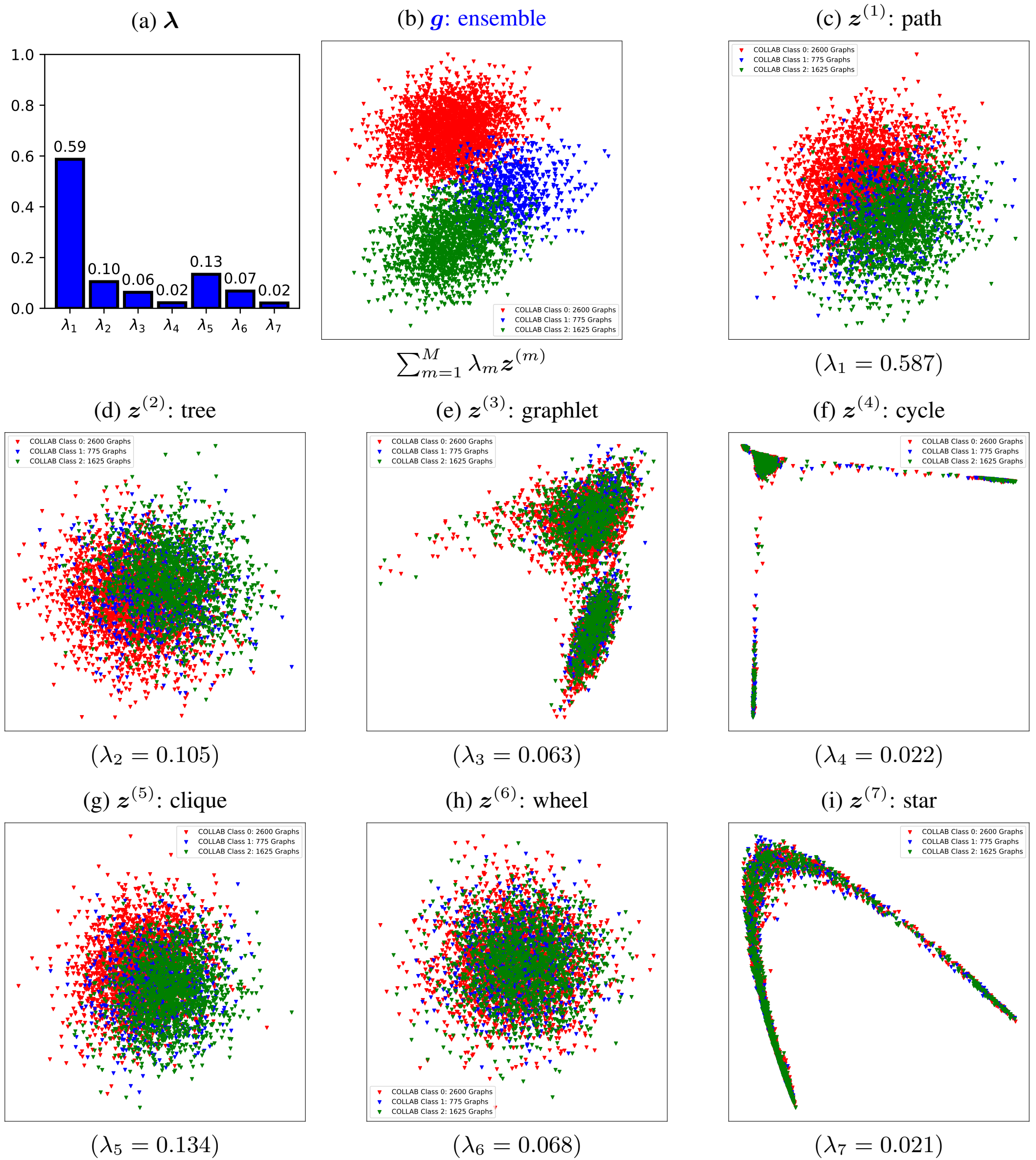


Figure 7: t-SNE visualizations of GNNs' pattern representations (supervised) for the dataset NCI1.



(a) $\boldsymbol{\lambda}$

(b) $\boldsymbol{g}$: ensemble

$\sum_{m=1}^{M} \lambda_m \boldsymbol{z}^{(m)}$

(c) $\boldsymbol{z}^{(1)}$: path

$(\lambda_1 = 0.587)$

(d) $\boldsymbol{z}^{(2)}$: tree

$(\lambda_2 = 0.105)$

(e) $\boldsymbol{z}^{(3)}$: graphlet

$(\lambda_3 = 0.063)$

(f) $\boldsymbol{z}^{(4)}$: cycle

$(\lambda_4 = 0.022)$

(g) $\boldsymbol{z}^{(5)}$: clique

$(\lambda_5 = 0.134)$

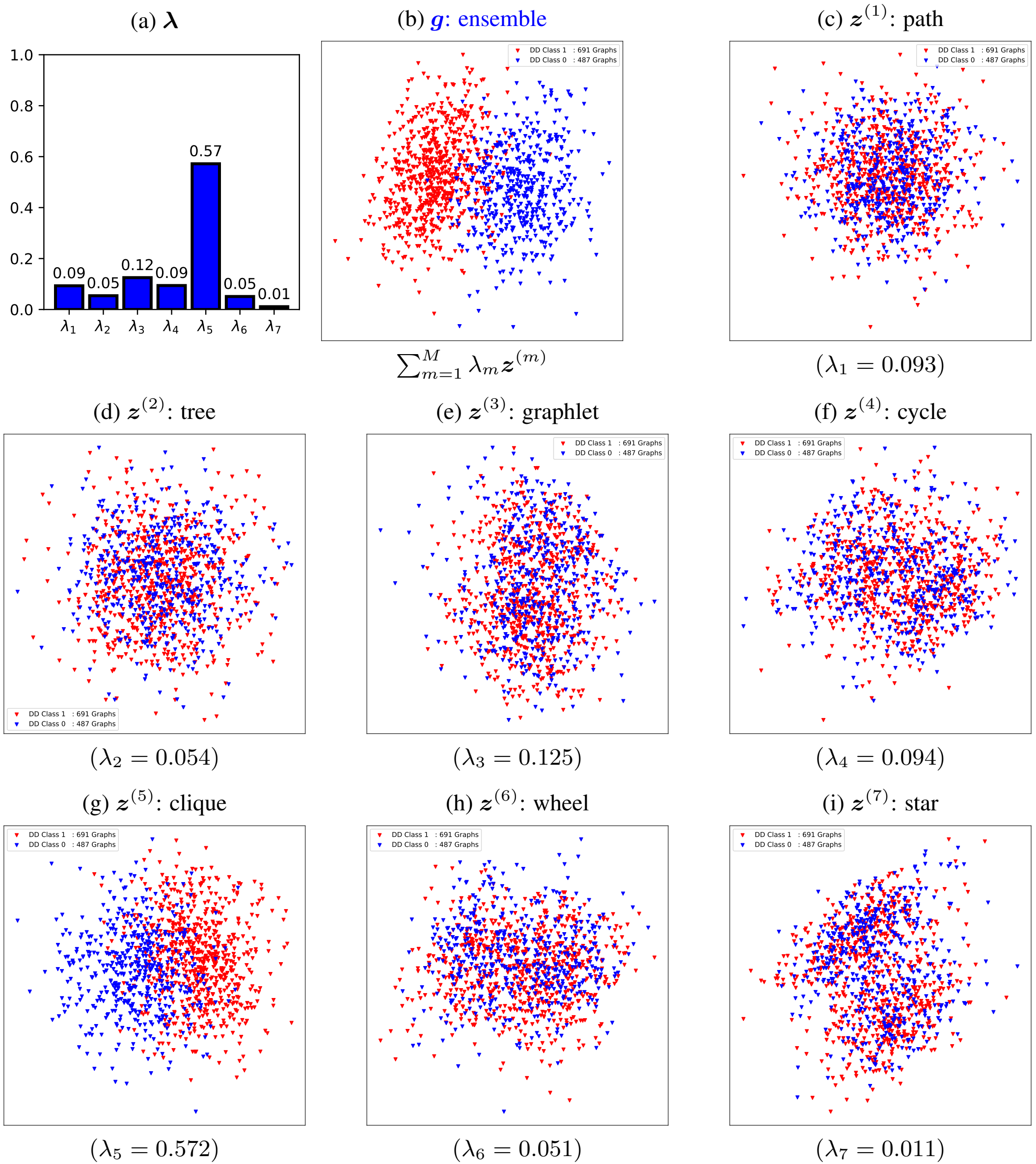
(h) $\boldsymbol{z}^{(6)}$: wheel

$(\lambda_6 = 0.068)$

(i) $\boldsymbol{z}^{(7)}$: star

$(\lambda_7 = 0.021)$

Figure 8: t-SNE visualizations of GNNs' pattern representations (supervised) for the dataset COLLAB.


(a) $\boldsymbol{\lambda}$

(b) $\boldsymbol{g}$: ensemble

$\sum_{m=1}^{M} \lambda_m \boldsymbol{z}^{(m)}$

(c) $\boldsymbol{z}^{(1)}$: path

$(\lambda_1 = 0.093)$

(d) $\boldsymbol{z}^{(2)}$: tree

$(\lambda_2 = 0.054)$

(e) $\boldsymbol{z}^{(3)}$: graphlet

$(\lambda_3 = 0.125)$

(f) $\boldsymbol{z}^{(4)}$: cycle

$(\lambda_4 = 0.094)$

(g) $\boldsymbol{z}^{(5)}$: clique

$(\lambda_5 = 0.572)$

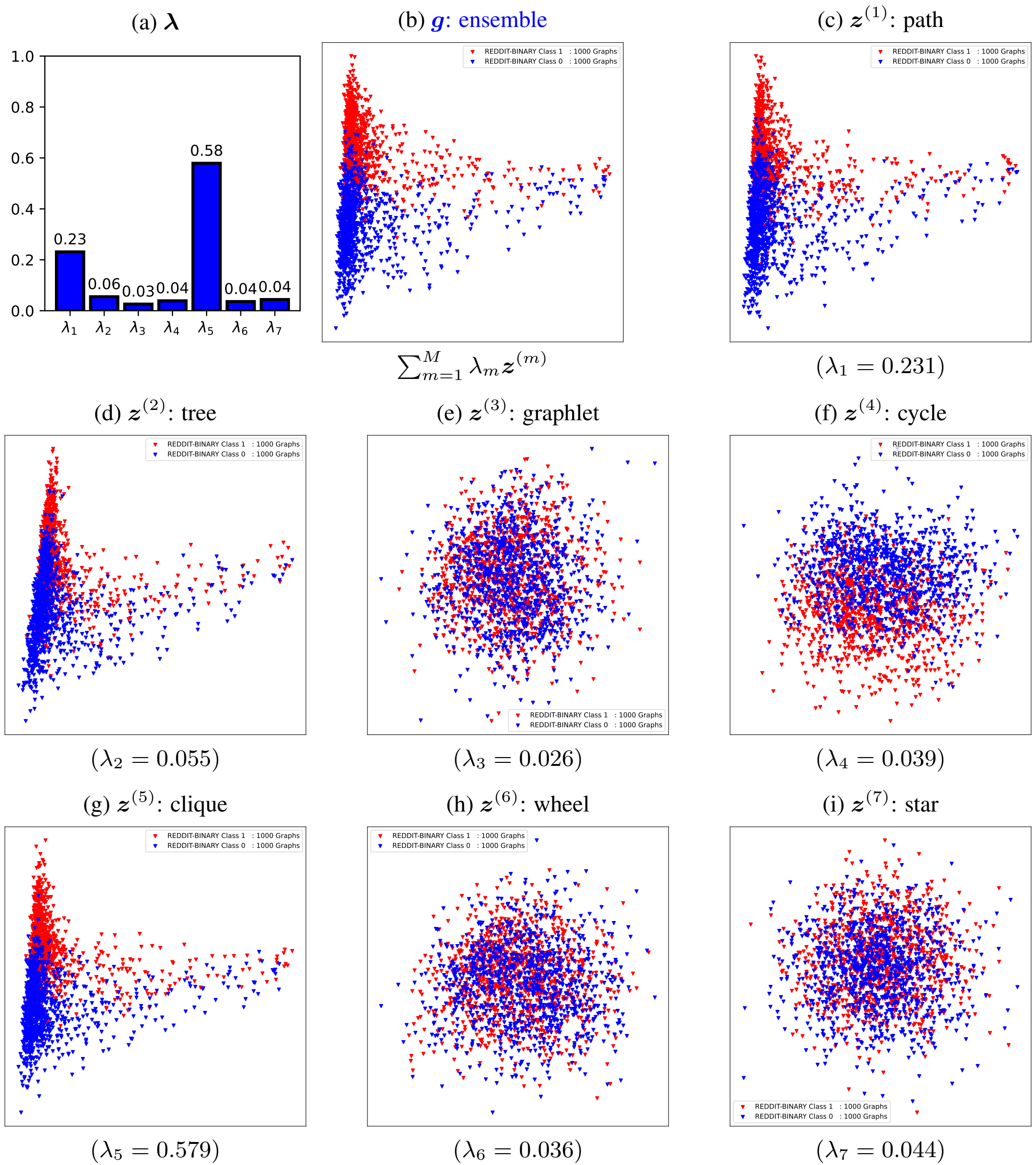
(h) $\boldsymbol{z}^{(6)}$: wheel

$(\lambda_6 = 0.051)$

(i) $\boldsymbol{z}^{(7)}$: star

$(\lambda_7 = 0.011)$

Figure 9: t-SNE visualizations of GNNs' pattern representations (supervised) for the dataset DD.



(a) $\boldsymbol{\lambda}$

(b) $\boldsymbol{g}$: ensemble

$\sum_{m=1}^{M} \lambda_m \boldsymbol{z}^{(m)}$

(c) $\boldsymbol{z}^{(1)}$: path

$(\lambda_1 = 0.231)$

(d) $\boldsymbol{z}^{(2)}$: tree

$(\lambda_2 = 0.055)$

(e) $\boldsymbol{z}^{(3)}$: graphlet

$(\lambda_3 = 0.026)$

(f) $\boldsymbol{z}^{(4)}$: cycle

$(\lambda_4 = 0.039)$

(g) $\boldsymbol{z}^{(5)}$: clique

$(\lambda_5 = 0.579)$

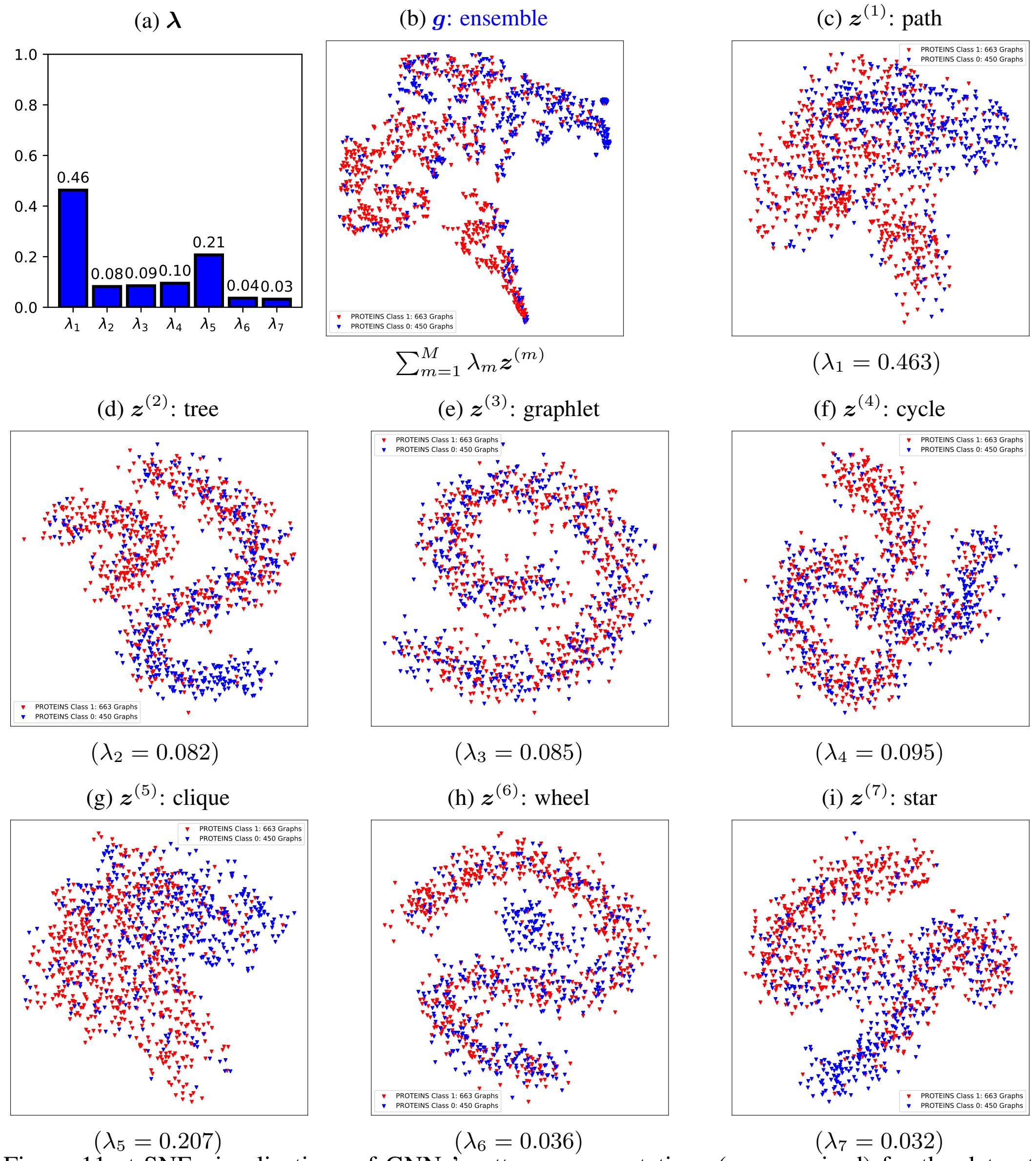
(h) $\boldsymbol{z}^{(6)}$: wheel

$(\lambda_6 = 0.036)$

(i) $\boldsymbol{z}^{(7)}$: star

$(\lambda_7 = 0.044)$

Figure 10: t-SNE visualizations of GNNs' pattern representations (supervised) for the dataset REDDIT-BINARY.



(a) $\boldsymbol{\lambda}$

(b) $\boldsymbol{g}$: ensemble

$\sum_{m=1}^{M} \lambda_m \boldsymbol{z}^{(m)}$

(c) $\boldsymbol{z}^{(1)}$: path

$(\lambda_1 = 0.463)$

(d) $\boldsymbol{z}^{(2)}$: tree

$(\lambda_2 = 0.082)$

(e) $\boldsymbol{z}^{(3)}$: graphlet

$(\lambda_3 = 0.085)$

(f) $\boldsymbol{z}^{(4)}$: cycle

$(\lambda_4 = 0.095)$

(g) $\boldsymbol{z}^{(5)}$: clique

$(\lambda_5 = 0.207)$

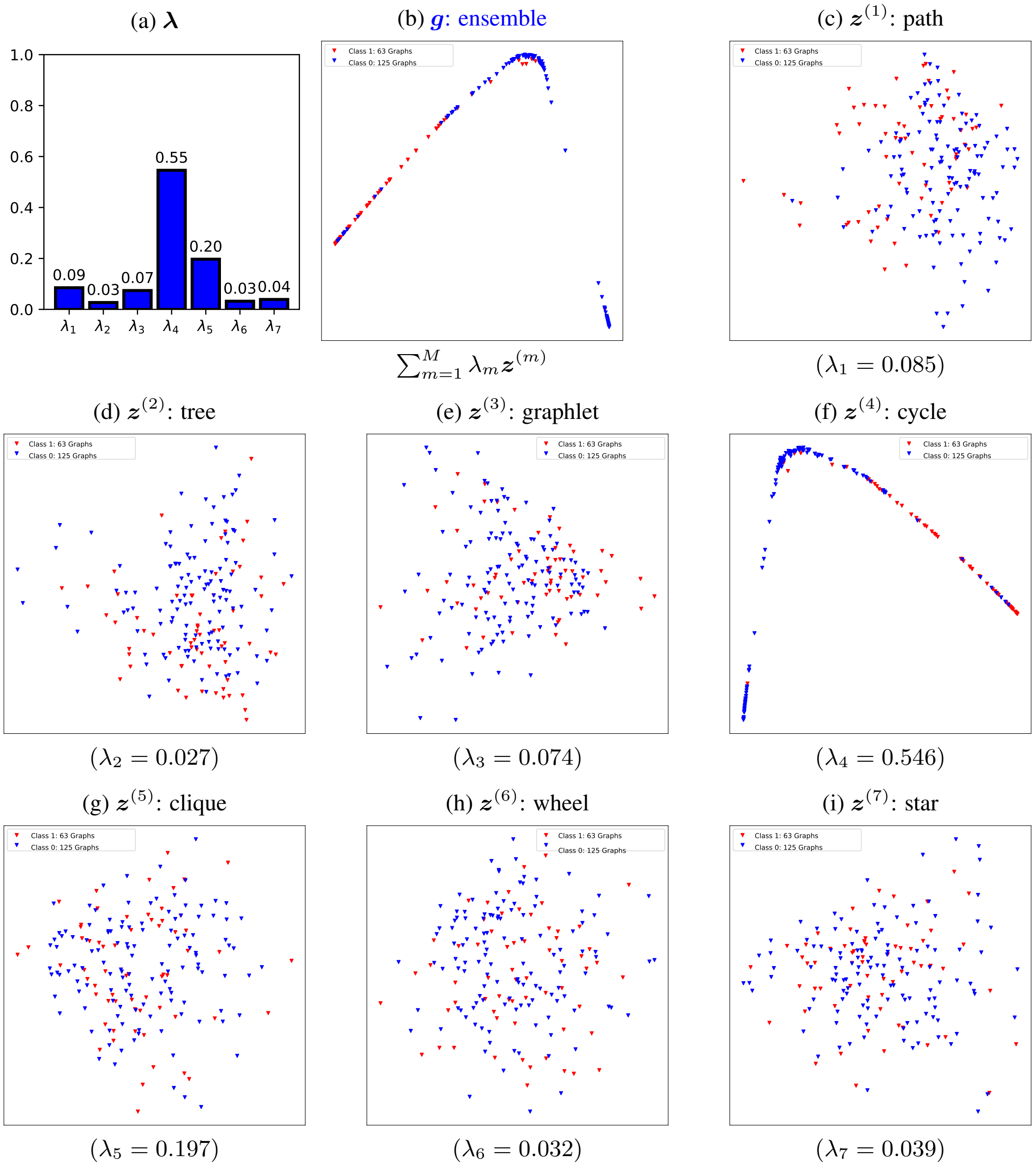
(h) $\boldsymbol{z}^{(6)}$: wheel

$(\lambda_6 = 0.036)$

(i) $\boldsymbol{z}^{(7)}$: star

$(\lambda_7 = 0.032)$

Figure 11: t-SNE visualizations of GNNs' pattern representations (unsupervised) for the dataset PROTEINS.



(a) $\boldsymbol{\lambda}$

(b) $\boldsymbol{g}$: ensemble

$\sum_{m=1}^{M} \lambda_m \boldsymbol{z}^{(m)}$

(c) $\boldsymbol{z}^{(1)}$: path

$(\lambda_1 = 0.085)$

(d) $\boldsymbol{z}^{(2)}$: tree

$(\lambda_2 = 0.027)$

(e) $\boldsymbol{z}^{(3)}$: graphlet

$(\lambda_3 = 0.074)$

(f) $\boldsymbol{z}^{(4)}$: cycle

$(\lambda_4 = 0.546)$

(g) $\boldsymbol{z}^{(5)}$: clique

$(\lambda_5 = 0.197)$

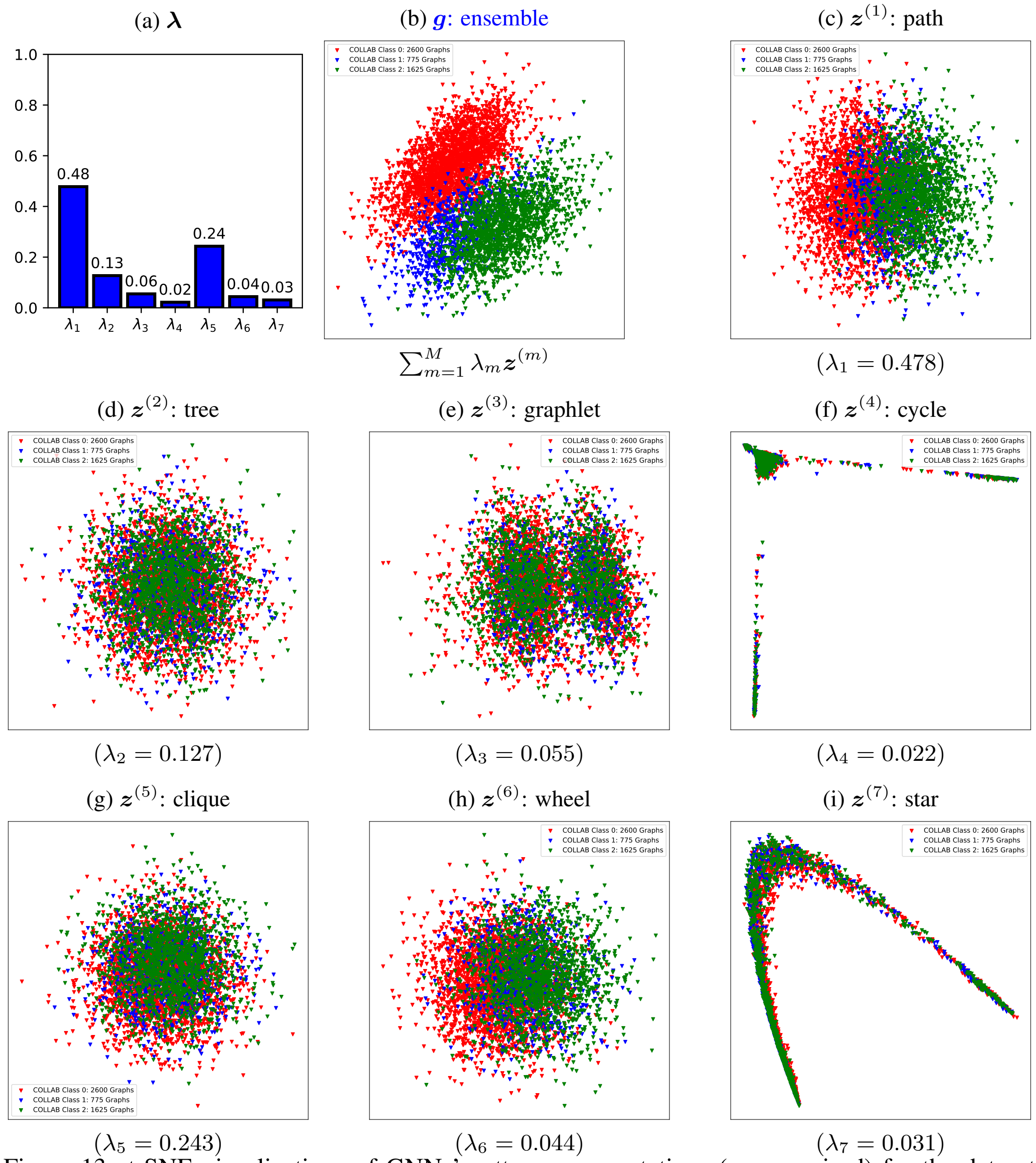
(h) $\boldsymbol{z}^{(6)}$: wheel

$(\lambda_6 = 0.032)$

(i) $\boldsymbol{z}^{(7)}$: star

$(\lambda_7 = 0.039)$

Figure 12: t-SNE visualizations of GNNs' pattern representations (unsupervised) for the dataset MUTAG.



(a) $\boldsymbol{\lambda}$

(b) $\boldsymbol{g}$: ensemble

$\sum_{m=1}^{M} \lambda_m \boldsymbol{z}^{(m)}$

(c) $\boldsymbol{z}^{(1)}$: path

($\lambda_1 = 0.478$)

(d) $\boldsymbol{z}^{(2)}$: tree

($\lambda_2 = 0.127$)

(e) $\boldsymbol{z}^{(3)}$: graphlet

($\lambda_3 = 0.055$)

(f) $\boldsymbol{z}^{(4)}$: cycle

($\lambda_4 = 0.022$)

(g) $\boldsymbol{z}^{(5)}$: clique

($\lambda_5 = 0.243$)

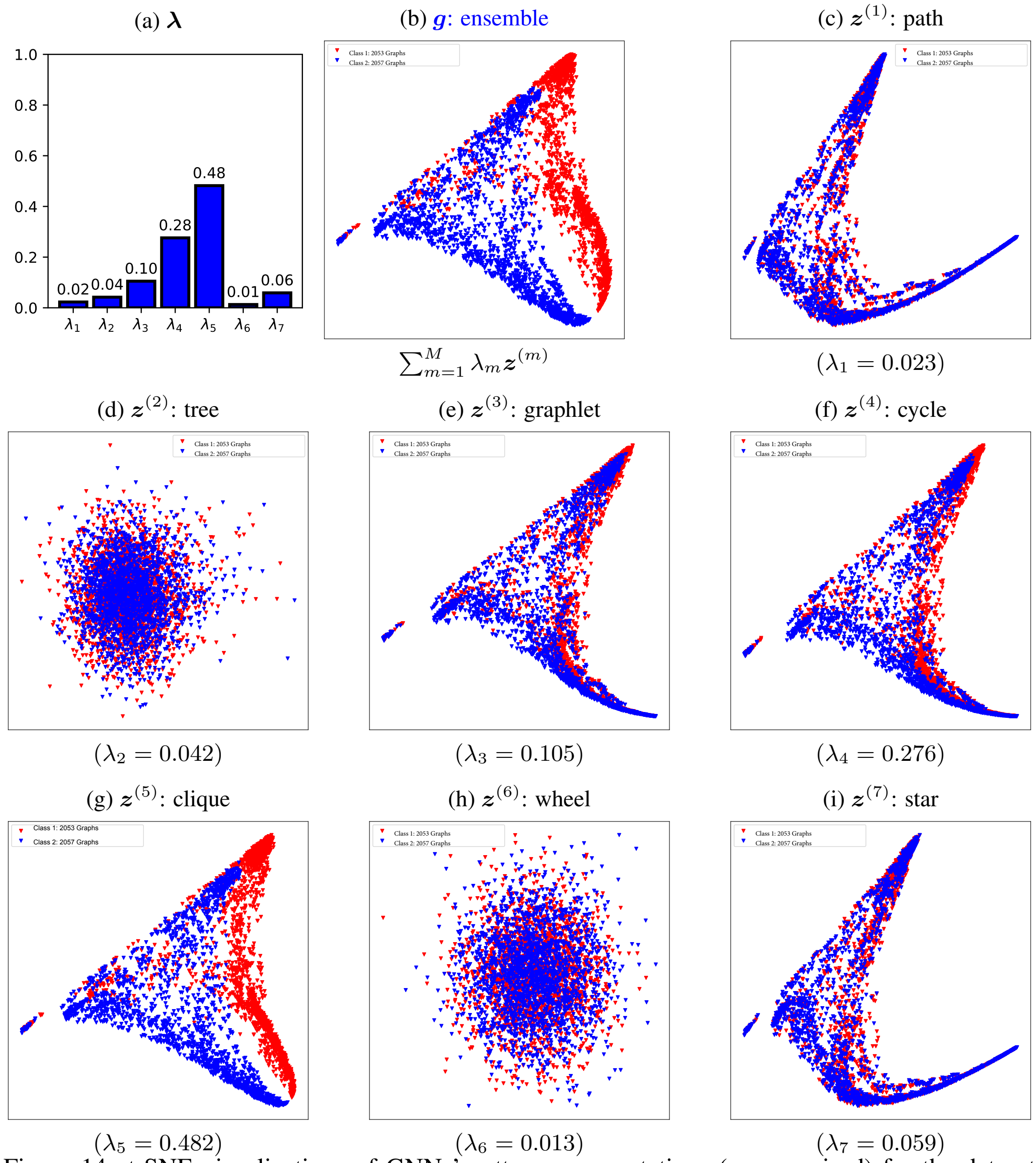
(h) $\boldsymbol{z}^{(6)}$: wheel

($\lambda_6 = 0.044$)

(i) $\boldsymbol{z}^{(7)}$: star

($\lambda_7 = 0.031$)

Figure 13: t-SNE visualizations of GNNs' pattern representations (unsupervised) for the dataset COLLAB.



(a) $\boldsymbol{\lambda}$

(b) $\boldsymbol{g}$: ensemble

$\sum_{m=1}^{M} \lambda_m \boldsymbol{z}^{(m)}$

(c) $\boldsymbol{z}^{(1)}$: path

$(\lambda_1 = 0.023)$

(d) $\boldsymbol{z}^{(2)}$: tree

$(\lambda_2 = 0.042)$

(e) $\boldsymbol{z}^{(3)}$: graphlet

$(\lambda_3 = 0.105)$

(f) $\boldsymbol{z}^{(4)}$: cycle

$(\lambda_4 = 0.276)$

(g) $\boldsymbol{z}^{(5)}$: clique

$(\lambda_5 = 0.482)$

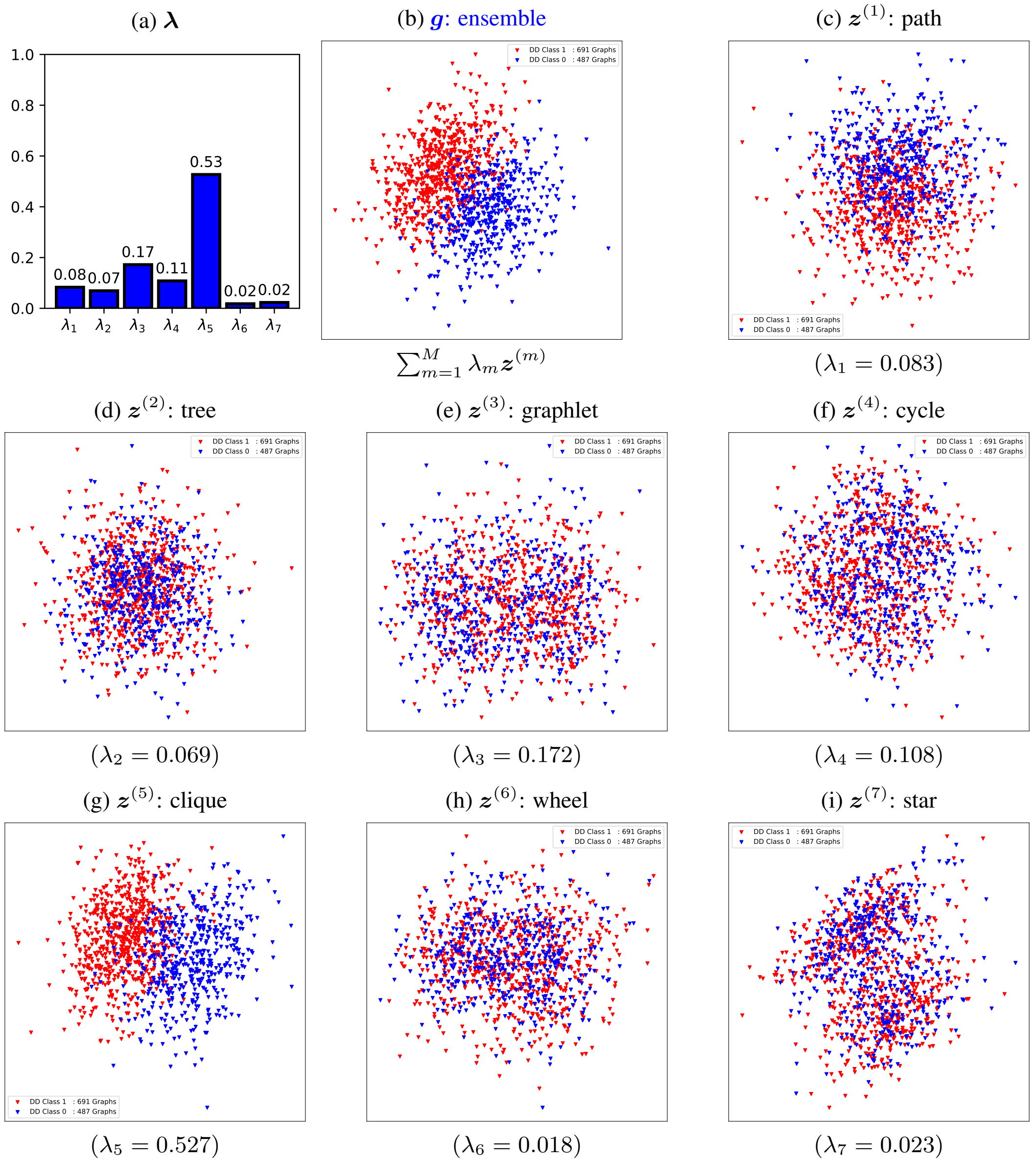
(h) $\boldsymbol{z}^{(6)}$: wheel

$(\lambda_6 = 0.013)$

(i) $\boldsymbol{z}^{(7)}$: star

$(\lambda_7 = 0.059)$

Figure 14: t-SNE visualizations of GNNs' pattern representations (unsupervised) for the dataset NCI1.



(a) $\boldsymbol{\lambda}$

(b) $\boldsymbol{g}$: ensemble

(c) $\boldsymbol{z}^{(1)}$: path

$\sum_{m=1}^{M} \lambda_m \boldsymbol{z}^{(m)}$

($\lambda_1 = 0.083$)

(d) $\boldsymbol{z}^{(2)}$: tree

(e) $\boldsymbol{z}^{(3)}$: graphlet

(f) $\boldsymbol{z}^{(4)}$: cycle

($\lambda_2 = 0.069$)

($\lambda_3 = 0.172$)

($\lambda_4 = 0.108$)

(g) $\boldsymbol{z}^{(5)}$: clique

(h) $\boldsymbol{z}^{(6)}$: wheel

(i) $\boldsymbol{z}^{(7)}$: star

($\lambda_5 = 0.527$)

($\lambda_6 = 0.018$)

($\lambda_7 = 0.023$)

Figure 15: t-SNE visualizations of GNNs' pattern representations (unsupervised) for the dataset DD.