Interpretable Graph Model with Prototype-Based Graph Information Bottleneck

Sangwoo Seo[1], Sungwon Kim[1] and Chanyoung Park[1,†]

[1]Korea Advanced Institute of Science and Technology (KAIST), 291, Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea

Abstract

The success of Graph Neural Networks (GNNs) has led to a need for understanding their decision-making process and providing explanations for their predictions, which has given rise to explainable AI (XAI) that offers transparent explanations for black-box models. Recently, the use of prototypes has successfully improved the explainability of models by learning prototypes to imply training graphs that affect the prediction. However, these approaches tend to provide prototypes with excessive information from the entire graph, leading to the exclusion of key substructures or the inclusion of irrelevant substructures, which can limit both the interpretability and the performance of the model in downstream tasks. In this work, we propose a novel framework of explainable GNNs, called interpretable graph model with **P**rototype-Based **G**raph **I**nformation **B**ottleneck, that incorporates prototype learning within the information bottleneck framework to provide prototypes with the key subgraph from the input graph that is important for the model prediction. Extensive experiments demonstrate that PGIB outperforms state-of-the-art methods in terms of both prediction performance and explainability.

Keywords

Interpretability, Graph Neural Networks, Explainable AI

1. Introduction

In general, explainability can be viewed from two perspectives: 1) ***improving the interpretability*** by providing explanations for the model's *predictions*, and 2) ***providing the reasoning process*** behind the model prediction by giving explanations for the model's *training process*. *Improving the interpretability* in GNNs involves detecting important substructures during the inference phase, which is useful for tasks such as identifying functional groups in molecular chemistry [1, 2, 3, 4]. On the other hand, it is also important to *provide the reasoning process* for why the model predicts in a certain way so as to understand the model in a more fundamental level. Through this reasoning process, we can visualize and analyze how the model makes correct or incorrect decisions, thus obtaining crucial information for improving its performance.

Some approaches to explore the reasoning process can be generally classified into two main categories: 1) ***post-hoc*** approaches, and 2) ***built-in*** approaches. *Post-hoc* approaches [5, 6, 7] focus on exploring the model outputs by visualizing the degree of activation of neurons based on gradients. However, the post-hoc approaches require a separate explanatory model for each trained model, resulting in the need for a new explanatory model for additional training

HI-AI@KDD, Human-Interpretable AI Workshop at the KDD 2024, 26th of August 2024, Barcelona, Spain

† Corresponding author.

tkddn8974@kaist.ac.kr (S. Seo); swkim@kaist.ac.kr (S. Kim); cy.park@kaist.ac.kr (C. Park)

https://sung-won-kim.github.io/ (S. Kim); https://dsail.kaist.ac.kr/professor/ (C. Park)

0009-0004-3151-8061 (S. Seo); 0000-0001-8605-2618 (S. Kim); 0000-0002-5957-5816 (C. Park)

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

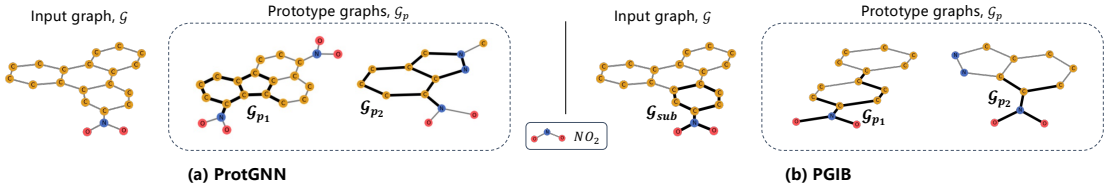


Figure 1: Comparison of the learned prototypes between ProtGNN and PGIB.

data and different models [8, 9]. In order to address the aforementioned challenges, *built-in* approaches aim to integrate the generation of explanations into the model training process. One such approach is prototype learning, which involves learning prototypes that represent each class of the input data, which are then compared with new instances to make predictions. ProtGNN [10], for instance, measures the similarity between the embedding of an input graph and each prototype, providing explanations through the similarity calculation and making predictions for the input graph based on its similarity with the learned prototypes.

However, since ProtGNN compares the graph-level embedding of an input graph with the learned prototypes, the model overlooks the key substructures in the input graph while also potentially including uninformative substructures. This not only results in a degradation of the interpretability of the reasoning process, but also limits the performance on the downstream tasks. Figure 1(a) shows the prototype graphs in the training set (i.e., $\mathcal{G}_p$, denoted as bold edges) detected by ProtGNN for an input molecule (i.e., $\mathcal{G}$) that belongs to the "mutagenic" class. Despite the NO_2 structure being the key functional group for classifying a given molecule as "mutagenic," $\mathcal{G}_p$ detected by ProtGNN tends to include ring structures (i.e., uninformative substructures), and exclude NO_2 structures (i.e., key substructures) in learned prototypes, which is mainly due to the fact that the input graph $\mathcal{G}$ is considered in the whole graph-level. As a result, it is crucial to identify a key subgraph within the input graph that holds essential information for the learning of prototypes. Among the various solutions for detecting important subgraphs, the Information Bottleneck (IB) has emerged as one of the most effective methods [11, 12, 13]. We aim to approach the IB principle from the perspective of prototypes to convey important substructure information to the prototypes.

To this end, we propose a novel framework of explainable GNNs, called Interpretable Graph Model with **P**rototype-based **G**raph **I**nformation **B**ottleneck (PGIB). The main idea is to incorporate prototype learning within the IB framework, which enables the prototypes to capture the key subgraph of the input graph detected by the IB framework. This enables the learning of prototypes $\mathcal{G}_p$ based on the key subgraph $\mathcal{G}_{sub}$, leading to a more precise explanation of the reasoning process and improvement in the performance on the downstream tasks. In Figure 1(b), PGIB is shown to successfully detect the key subgraph $\mathcal{G}_{sub}$ that includes NO_2.

Our main contributions can be summarized as follows: **1)** We propose an effective approach, PGIB, that not only improves the interpretability of the reasoning process, but also the overall performance in downstream tasks by incorporating the prototype learning in a process of detecting key subgraphs based on the IB framework. **2)** We provide theoretical background with our method that utilizes interpretable prototypes in the process of optimizing $\mathcal{G}_{sub}$. **3)** Extensive experiments, including qualitative analysis, demonstrate that PGIB outperforms state-of-the-art methods in terms of both prediction performance and explainability.

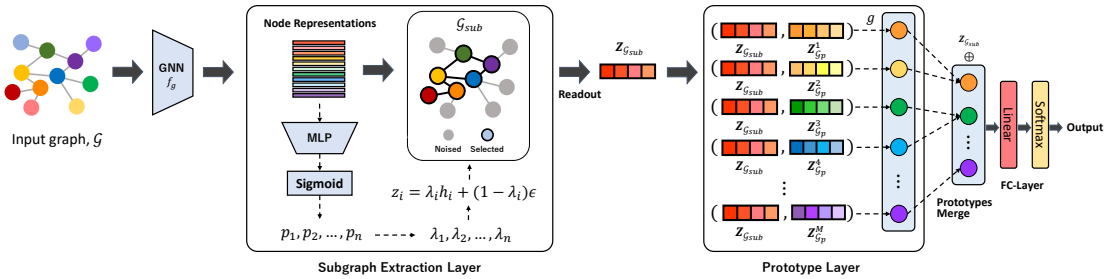


Figure 2: The architecture of our proposed PGIB. PGIB generates a subgraph $\mathcal{G}_{sub}$ by injecting noise to identify core subgraphs, and it is used to compute similarity scores between prototypes in the prototype layer. The trained prototypes play a crucial role in visualizing the reasoning processes during training in an interpretable manner. PGIB also involves merging pairs of similar prototypes to decrease the number of prototypes. Finally, the integrated prototypes are utilized to predict the graph labels in the fully connected layer.

2. Notations.

We use $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ to denote a graph, where $\mathcal{V}$, $\mathcal{E}$, $\mathbf{A}$ and $\mathbf{X}$ denote the set of nodes and edges, the adjacency matrix and node features, respectively. We assume that each node $v_i \in \mathcal{V}$ is associated with a feature vector $\mathbf{x}_i$, which is the i-th row of $\mathbf{X}$. We use $\{(\mathcal{G}_1, y_1), \cdots, (\mathcal{G}_N, y_N)\}$ to denote the set of N graphs with its corresponding labels. The graph labels are given by a set of K classes $\mathcal{C} = \{1, 2, \ldots, K\}$, and the ground truth label of a graph $\mathcal{G}_i$ is denoted by $y_i \in \mathcal{C}$. We use $\mathcal{G}_{sub}$ to denote a subgraph of $\mathcal{G}$, and use $\bar{\mathcal{G}}_{sub}$ to denote the complementary structure of $\mathcal{G}_{sub}$ in $\mathcal{G}$. We also introduce the prototype layer, which consists of a set of prototypes $\mathcal{Z}_p = \left\{\mathbf{z}^1_{\mathcal{G}_p}, \mathbf{z}^2_{\mathcal{G}_p}, \cdots, \mathbf{z}^M_{\mathcal{G}_p}\right\}$, where M is the total number of prototypes, and each prototype $\mathbf{z}^m_{\mathcal{G}_p}$ is a learnable parameter vector that serves as the latent representation of the prototypical part (i.e., $\mathcal{G}_p$) of graph $\mathcal{G}$. We allocate J prototypes for each class, i.e., $M = K \times J$.

3. Methodology

3.1. Prototype-based Graph Information Bottleneck

PGIB is a novel explainable GNN framework that incorporates the prototype learning within the IB framework, thereby enabling the prototypes to capture the essential key subgraph of the input graph detected by the IB framework. More precisely, we reformulate the GIB objective [11] by decomposing the first term, i.e., $I(Y; \mathcal{G}_{sub})$, with respect to the prototype $\mathcal{G}_p$ using the chain rule of mutual information in order to examine the impact of the joint information between $\mathcal{G}_{sub}$ and $\mathcal{G}_p$ on Y as follows:

$$\min_{\mathcal{G}_{sub}} \underbrace{-I(Y; \mathcal{G}_{sub}, \mathcal{G}_p) + I(Y; \mathcal{G}_p | \mathcal{G}_{sub})}_{\text{Section 3.3}} + \beta \underbrace{I(\mathcal{G}; \mathcal{G}_{sub})}_{\text{Section 3.2}}. \tag{1}$$

Please refer to Appendix A.1 for a detailed proof of Equation 1. In the following sections, we describe how each term is optimized during training.

3.2. Subgraph Extraction Layer (Minimizing $I(\mathcal{G};\mathcal{G}_{sub})$)

The goal of the subgraph extraction layer is to extract an informative subgraph $\mathcal{G}_{sub}$ from $\mathcal{G}$ that contains minimal information about $\mathcal{G}$. We minimize $I(\mathcal{G};\mathcal{G}_{sub})$ by training the model to inject noise into insignificant subgraphs $\bar{\mathcal{G}}_{sub}$, while injecting less noise into informative ones $\mathcal{G}_{sub}$ [14]. Specifically, given the representation of node v_i, i.e., $\mathbf{h}_i$, we compute the probability p_i, which is then used to replace the representation h_i to obtain the final representation $\mathbf{z}_i$ as:

$$\begin{aligned} p_i &= \text{Sigmoid}(\text{MLP}(h_i)) \\ z_i &= \lambda_i h_i + (1-\lambda_i)\epsilon, \ \text{ where } \lambda_i \sim \text{Bernoulli}(p_i) \text{ and } \epsilon \sim \mathcal{N}(\mu_{\mathbf{h}_i}, \sigma^2_{\mathbf{h}_i}). \end{aligned} \tag{2}$$

Following [14], we minimize the upper bound of $I(\mathcal{G};\mathcal{G}_{sub})$ as:

$$I(\mathcal{G};\mathcal{G}_{sub}) \leq \mathbb{E}_{\mathcal{G}}(-\frac{1}{2}\log A + \frac{1}{2|\mathcal{V}_{\mathcal{G}}|}A + \frac{1}{2|\mathcal{V}_{\mathcal{G}}|}B^2) =: \mathcal{L}^1_{\text{MI}}(\mathcal{G},\mathcal{G}_{sub}), \tag{3}$$

where $A = \sum_{i=1}^{|\mathcal{V}_{\mathcal{G}}|}(1-\lambda_i)^2$ and $B = \frac{\sum_{i=1}^{|\mathcal{V}_{\mathcal{G}}|}\lambda_i(\mathbf{h}_i - \mu_{\mathbf{h}_i})}{\sigma_{\mathbf{h}_i}}$. Thus, minimizing $\mathcal{L}^1_{\text{MI}}$ allows us to minimize the upper bound of $I(\mathcal{G};\mathcal{G}_{sub})$.

3.3. Prototype Layer (Minimizing $-I(Y;\mathcal{G}_{sub},\mathcal{G}_p) + I(Y;\mathcal{G}_p|\mathcal{G}_{sub})$)

The prototype layer involves allocation of a fixed number of prototypes for each class. The prototypes are required to capture the most significant graph patterns within each class. To begin with, we define the similarity score between the prototype $\mathbf{z}_{\mathcal{G}_p}$ and the embedding $\mathbf{z}_{\mathcal{G}_{sub}}$ obtained from noise injection as : $g(\mathbf{z}_{\mathcal{G}_{sub}}, \mathbf{z}_{\mathcal{G}_p}) = \log\left[\left(\|\mathbf{z}_{\mathcal{G}_{sub}} - \mathbf{z}_{\mathcal{G}_p}\|_2^2 + 1\right) / \left(\|\mathbf{z}_{\mathcal{G}_{sub}} - \mathbf{z}_{\mathcal{G}_p}\|_2^2 + \epsilon\right)\right]$ where $\mathbf{z}_{\mathcal{G}_p}$ is the prototype and shares the same dimension as $\mathbf{z}_{\mathcal{G}_{sub}}$.

3.3.1. Minimizing $-I(Y;\mathcal{G}_{sub},\mathcal{G}_p)$

We derive the lower bound of $I(Y;\mathcal{G}_{sub},\mathcal{G}_p)$ as follows:

Proposition 1. **(Lower bound of $I(Y;\mathcal{G}_{sub},\mathcal{G}_p)$)** Given significant subgraph $\mathcal{G}_{sub}$ for a graph $\mathcal{G}$, its label information Y, prototype graph $\mathcal{G}_p$ and similarity function γ, we have

$$\begin{aligned} I(Y;\mathcal{G}_{sub},\mathcal{G}_p) &= \mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log p\left(Y|\mathcal{G}_{sub},\mathcal{G}_p\right)\right] - \mathbb{E}_Y\left[\log p(Y)\right] \\ &\geq \mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log p\left(Y|\gamma\left(\mathcal{G}_{sub},\mathcal{G}_p\right)\right)\right] - \mathbb{E}_Y\left[\log p(Y)\right] \\ &\geq \mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log q_\theta\left(Y|\gamma\left(\mathcal{G}_{sub},\mathcal{G}_p\right)\right)\right] \\ &=: -\mathcal{L}_{\text{cls}}(q_\theta\left(Y|\gamma\left(\mathcal{G}_{sub},\mathcal{G}_p\right)\right) \end{aligned} \tag{4}$$

where $q_\theta\left(Y|\gamma\left(\mathcal{G}_{sub},\mathcal{G}_p\right)\right)$ is the variational approximation to the true posterior $p\left(Y|\gamma\left(\mathcal{G}_{sub},\mathcal{G}_p\right)\right)$.

3.3.2. Minimizing $I(Y;\mathcal{G}_p|\mathcal{G}_{sub})$

We decompose $I(Y;\mathcal{G}_p|\mathcal{G}_{sub})$ into the sum of two terms as follows:

$$I(Y;\mathcal{G}_p|\mathcal{G}_{sub}) = I(\mathcal{G}_p;Y,\mathcal{G}_{sub}) - I(\mathcal{G}_{sub};\mathcal{G}_p). \tag{5}$$

It is important to note that the first term, i.e., $I(\mathcal{G}_p;Y,\mathcal{G}_{sub})$, eliminates the information about Y related to $\mathcal{G}_{sub}$ from $\mathcal{G}_p$. However, since our goal is not to solely minimize $I(\mathcal{G}_p;Y,\mathcal{G}_{sub})$ but to ensure the interpretability of the prototype $\mathcal{G}_p$, including this term leads to diminished interpretability of $\mathcal{G}_p$. Consequently, we excluded the first term during training, and only consider the second term, i.e., $-I(\mathcal{G}_{sub};\mathcal{G}_p)$, to simultaneously guarantee the interpretability of

both $\mathcal{G}_{sub}$ and $\mathcal{G}_p$. A detailed derivation for Equation 5 is given in Appendix A.3. To minimize $-I(\mathcal{G}_{sub}; \mathcal{G}_p)$, we introduce two different approaches.

1) Variational IB-based approach. We obtain the upper bound of $-I(\mathcal{G}_{sub}; \mathcal{G}_p)$ using the variational IB-based approach as follows:

$$-I(\mathcal{G}_{sub}; \mathcal{G}_p) \leq \mathbb{E}_{\mathcal{G}_{sub}, \mathcal{G}_p}[-\log q_\phi(\mathcal{G}_p|\mathcal{G}_{sub})] := \mathcal{L}^2_{\text{MI}}(\mathcal{G}_{sub}, \mathcal{G}_p), \tag{6}$$

where $q_\phi(\mathcal{G}_p|\mathcal{G}_{sub})$ is the variation approximation of $p(\mathcal{G}_p|\mathcal{G}_{sub})$. Equation 6 shows that the maximization of the mutual information $I(\mathcal{G}_{sub}; \mathcal{G}_p)$ can be attained by minimizing $\mathcal{L}^2_{\text{MI}}(\mathcal{G}_{sub}, \mathcal{G}_p)$. We select a single-layer linear transformation as a modeling option for q_ϕ to minimize the information loss of $\mathcal{G}_{sub}$ when predicting $\mathcal{G}_p$.

2) Contrastive learning-based approach. Recent studies on contrastive learning [15, 16, 17] have proven that minimizing contrastive loss is equivalent to maximizing the mutual information between two variables. Hence, we additionally propose a variant of PGIB, i.e., $\text{PGIB}_{\text{cont}}$, that minimizes the contrastive loss. The contrastive loss is defined as follows:

$$\mathcal{L}^2_{\text{MI}} = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{\sum\limits_{j:\mathbf{z}_{\mathcal{G}_p^j} \in \mathbb{P}_{y_i}} \exp(g(\mathbf{z}_{\mathcal{G}_{sub}^i}, \mathbf{z}_{\mathcal{G}_p^j})/\tau)}{\sum\limits_{k:\mathbf{z}_{\mathcal{G}_p^k} \notin \mathbb{P}_{y_i}} \exp(g(\mathbf{z}_{\mathcal{G}_{sub}^i}, \mathbf{z}_{\mathcal{G}_p^k})/\tau)}. \tag{7}$$

where τ is the temperature hyperparameter, n denotes the number of graphs in a batch, and j and k indicate indices of positive and negative samples, respectively. $\mathbb{P}_{y_i}$ is the set of prototypes that belong to class y_i. We confer interpretability to $\mathcal{G}_p$ by increasing its similarity with $\mathcal{G}_{sub}$.

3.4. Prediction Layer

We compute the predicted probability $\boldsymbol{\pi} \in \mathbb{R}^K$ by passing similarity scores $\mathbf{r} \in \mathbb{R}^M$ and $\mathbf{z}_{\mathcal{G}_{sub}}$ through a linear layer with weights ω. Then we calculate cross-entropy classification loss as:

$$\mathcal{L}_{\text{cls}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{K} \mathbb{I}(y_i = c)\log(\pi_c). \tag{8}$$

Finally, we define the total objective as follows: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \alpha_1\mathcal{L}^1_{\text{MI}} + \alpha_2\mathcal{L}^2_{\text{MI}} + \alpha_3\mathcal{L}_{\text{con}}$, where $\mathcal{L}_{\text{con}}$ is defined in Appendix A.4.3 and a detailed ablation study is provided in Appendix A.5.2.

4. Experimental Results

Graph Classification. Experimental results for graph classification are presented in the Table 1. We have the following observations: **1)** All variants of PGIB outperform the baselines including both the prototype-based and IB-based methods on all datasets. Notably, PGIBs incorporate the crucial information of the key subgraph, which significantly contributes to enhancing the classification performance. $\text{PGIB}_{\text{cont}}$ achieves a significant improvement of up to 5.6% compared to the runner-up baseline. **2)** We observe that $\text{PGIB}_{\text{cont}}$ performs relatively better than PGIB. We attribute this to the nature of the contrastive loss, which is generally shown to be effective in classifying instances between different classes, allowing the prototypes learned based on the contrastive loss to be more distinguishable from one another.

Graph Interpretation. Figure 3(a) presents the visualization of subgraphs in Mutag dataset. The NO_2 functional group is known to be a cause of mutagenicity [1, 26]. In the figure, the bold edges connect the nodes that the models consider important. The NO_2 group in Mutag is

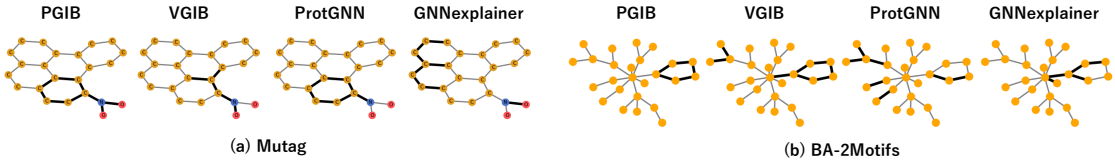
Table 1
Evaluation on graph classification (accuracy).

Dataset	Methods								
	GCN [18]	GIN [19]	GAT [20]	ProtGNN [10]	GIB [12]	VGIB [14]	GSAT [21]	**PGIB**	**PGIB$_{\text{cont}}$**
MUTAG [22]	74.50±7.89	80.50±7.89	73.50±7.43	80.50±9.07	79.00±6.24	81.00±6.63	80.88±8.94	85.00±7.07	**85.50±5.22**
PROTEINS [23]	72.83±4.23	70.30±4.84	71.35±4.85	73.83±4.22	75.25±5.92	73.66±3.32	69.64±4.71	77.14±2.19	**77.50±2.42**
NCI1 [24]	73.16±3.49	75.04±2.08	66.05±1.03	74.13±2.10	64.65±6.78	63.75±3.37	68.13±2.64	77.65±2.20	**78.25±2.13**
DD [25]	72.53±4.51	72.04±3.62	70.81±4.33	69.15±4.33	72.61±8.26	72.77±5.63	71.93±2.74	73.36±1.80	**73.70±2.14**

Table 2: Evaluation on graph interpretation (Fidelity scores).

Method	$\mathcal{F}_- \downarrow$				$\mathcal{F}_+ \uparrow$			
	RLM	HLM-CLint	QED	DRD2	RLM	HLM-CLint	QED	DRD2
GNNexplainer [1]	0.478	0.616	0.498	0.433	0.694	0.778	0.602	0.740
PGexplainer [4]	0.502	0.620	0.560	0.540	0.632	0.692	0.598	0.686
GIB [12]	0.483	0.643	0.525	0.428	0.654	0.781	0.601	0.724
VGIB [14]	0.463	0.579	0.487	0.424	0.765	0.792	0.627	0.756
PGIB$_{\text{cont}}$	0.441	0.593	0.459	0.406	0.747	0.772	0.613	0.771
PGIB$_{\text{cont}}$ + merge	**0.415**	**0.543**	**0.447**	**0.379**	**0.765**	**0.796**	**0.635**	**0.781**

Figure 3: Explanation visualizations on Mutag (a) and BA-2Motifs (b)



correctly identified by PGIB, while other baselines fail to recognize all NO_2 groups or include other unnecessary substructures. Figure 3(b) presents the visualization in BA-2Motifs dataset. We observe that PGIB accurately recognizes motif graphs containing the label information such as a five-node cycle, but other models have difficulty in detecting complete motifs.

We also perform quantitative experiments using the Fidelity metric (i.e., $\mathcal{F}_-$ and $\mathcal{F}_+$) [5, 7]. The Fidelity metric quantifies the extent to which explanations accurately capture the important components. In other words, they measure how well the predictions made solely based on the extracted subgraph (i.e., $\mathcal{F}_-$) and the remaining subgraph (i.e., $\mathcal{F}_+$) mimic the predictions made based on the entire graph, respectively. Hence, a low value of $\mathcal{F}_-$ and a high value of $\mathcal{F}_-$ indicate better explainability of the model. Table 2 shows the fidelity scores on four datasets at the sparsity score of $k = 0.5$. Our proposed model outperforms both post-hoc and built-in state-of-the-art explanation models in all datasets. Furthermore, merging prototypes achieves significant improvements in terms of interpretability. This implies that decreasing the number of prototypes can eliminate uninformative substructures and emphasize key substructures, which increases the interpretability of the extracted subgraphs.

5. Conclusion

We propose a novel framework of explainable GNNs, called interpretable graph model with Prototype-based Graph Information Bottleneck (PGIB), that integrates prototype learning into the information bottleneck framework. The main idea of PGIB is to learn prototypes that capture subgraphs containing key structures relevant to the label information. Experimental results show that PGIB achieves improvements not only in the performance on downstream tasks, but also provides more precise explanation of the reasoning process.

References

[1] Z. Ying, D. Bourgeois, J. You, M. Zitnik, J. Leskovec, Gnnexplainer: Generating explanations for graph neural networks, in: Advances in Neural Information Processing Systems, 2019, pp. 9240–9251.

[2] H. Yuan, J. Tang, X. Hu, S. Ji, Xgnn: Towards model-level explanations of graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 430–438.

[3] M. Vu, M. T. Thai, Pgm-explainer: Probabilistic graphical model explanations for graph neural networks, in: Advances in Neural Information Processing Systems, 2020, pp. 12225–12235.

[4] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, X. Zhang, Parameterized explainer for graph neural network, in: Advances in Neural Information Processing Systems, 2020, pp. 19620–19631.

[5] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, H. Hoffmann, Explainability methods for graph convolutional neural networks, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 10772–10781.

[6] G. Jaume, P. Pati, B. Bozorgtabar, A. Foncubierta, A. M. Anniciello, F. Feroce, T. Rau, J.-P. Thiran, M. Gabrani, O. Goksel, Quantifying explainers of graph neural networks in computational pathology, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 8106–8116.

[7] H. Yuan, H. Yu, S. Gui, S. Ji, Explainability in graph neural networks: A taxonomic survey, IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).

[8] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, J. K. Su, This looks like that: deep learning for interpretable image recognition, Advances in neural information processing systems 32 (2019).

[9] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature machine intelligence 1 (2019) 206–215.

[10] Z. Zhang, Q. Liu, H. Wang, C. Lu, C. Lee, Protgnn: Towards self-explaining graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, 2022, pp. 9127–9135.

[11] T. Wu, H. Ren, P. Li, J. Leskovec, Graph information bottleneck, Advances in Neural Information Processing Systems 33 (2020) 20437–20448.

[12] J. Yu, T. Xu, Y. Rong, Y. Bian, J. Huang, R. He, Graph information bottleneck for subgraph recognition, in: International Conference on Learning Representations, 2021.

[13] N. Lee, D. Hyun, G. S. Na, S. Kim, J. Lee, C. Park, Conditional graph information bottleneck for molecular relational learning, arXiv preprint arXiv:2305.01520 (2023).

[14] J. Yu, J. Cao, R. He, Improving subgraph recognition with variational graph information bottleneck, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 19396–19405.

[15] Y. Tian, D. Krishnan, P. Isola, Contrastive multiview coding, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16, Springer, 2020, pp. 776–794.

[16] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, Y. Shen, Graph contrastive learning with

augmentations, Advances in neural information processing systems 33 (2020) 5812–5823.
[17] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, Y. Bengio, Learning deep representations by mutual information estimation and maximization, arXiv preprint arXiv:1808.06670 (2018).
[18] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).
[19] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, arXiv preprint arXiv:1810.00826 (2018).
[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).
[21] S. Miao, M. Liu, P. Li, Interpretable and generalizable graph learning via stochastic attention mechanism, in: International Conference on Machine Learning, PMLR, 2022, pp. 15524–15543.
[22] M. Rupp, A. Tkatchenko, K.-R. Müller, O. A. Von Lilienfeld, Fast and accurate modeling of molecular atomization energies with machine learning, Physical review letters 108 (2012) 058301.
[23] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, H.-P. Kriegel, Protein function prediction via graph kernels, Bioinformatics 21 (2005) i47–i56.
[24] N. Wale, I. A. Watson, G. Karypis, Comparison of descriptor spaces for chemical compound retrieval and classification, Knowledge and Information Systems 14 (2008) 347–375.
[25] P. D. Dobson, A. J. Doig, Distinguishing enzyme structures from non-enzymes without alignments, Journal of molecular biology 330 (2003) 771–783.
[26] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, C. Hansch, Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity, Journal of medicinal chemistry 34 (1991) 786–797.
[27] F. Doshi-Velez, B. Kim, A roadmap for a rigorous science of interpretability, arXiv preprint arXiv:1702.08608 2 (2017).
[28] T. Wang, Gaining free or low-cost interpretability with interpretable partial substitute, in: International Conference on Machine Learning, PMLR, 2019, pp. 6505–6514.
[29] D. Rymarczyk, Ł. Struski, J. Tabor, B. Zieliński, Protopshare: Prototypical parts sharing for similarity discovery in interpretable image classification, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1420–1430.
[30] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, nature 550 (2017) 354–359.
[31] J. J. Irwin, B. K. Shoichet, Zinc- a free database of commercially available compounds for virtual screening, Journal of chemical information and modeling 45 (2005) 177–182.

A. Appendix

A.1. Prototype-based Graph Information Bottleneck - Eq. 1

The GIB objective is :

$$\min_{\mathcal{G}_{sub}} -I(Y;\mathcal{G}_{sub}) + \beta I(\mathcal{G};\mathcal{G}_{sub}). \tag{9}$$

For the first term $-I(Y;\mathcal{G}_{sub})$, by definition:

$$-I(Y;\mathcal{G}_{sub}) = -\mathbb{E}_{Y,\mathcal{G}_{sub}}\left[\log \frac{p(Y,\mathcal{G}_{sub})}{p(Y)p(\mathcal{G}_{sub})}\right]. \tag{10}$$

We allow the involvement of $\mathcal{G}_p$ in Eq. 10 as follows:

$$\begin{aligned}
-I(Y;\mathcal{G}_{sub}) &= -\mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log \frac{p(Y,\mathcal{G}_{sub},\mathcal{G}_p)}{p(\mathcal{G}_{sub},\mathcal{G}_p)p(Y)}\right] - \mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log \frac{p(Y,\mathcal{G}_{sub})p(\mathcal{G}_{sub},\mathcal{G}_p)}{p(Y,\mathcal{G}_{sub},\mathcal{G}_p)p(\mathcal{G}_{sub})}\right] \\
&= -\mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log \frac{p(Y,\mathcal{G}_{sub},\mathcal{G}_p)}{p(\mathcal{G}_{sub},\mathcal{G}_p)p(Y)}\right] + \mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log \frac{p(Y,\mathcal{G}_{sub},\mathcal{G}_p)p(\mathcal{G}_{sub})}{p(Y,\mathcal{G}_{sub})p(\mathcal{G}_{sub},\mathcal{G}_p)}\right] \\
&= -\mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log \frac{p(Y,\mathcal{G}_{sub},\mathcal{G}_p)}{p(\mathcal{G}_{sub},\mathcal{G}_p)p(Y)}\right] + \mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log \frac{p(Y,\mathcal{G}_p|\mathcal{G}_{sub})}{p(Y|\mathcal{G}_{sub})p(\mathcal{G}_p|\mathcal{G}_{sub})}\right].
\end{aligned} \tag{11}$$

By the definition of conditional probability, we have the following equation:

$$-I(Y;\mathcal{G}_{sub}) = -I(Y;\mathcal{G}_{sub},\mathcal{G}_p) + I(Y;\mathcal{G}_p|\mathcal{G}_{sub}). \tag{12}$$

Finally, we have the following equation by combining Eq. 9 and Eq. 12:

$$\min_{\mathcal{G}_{sub}} -I(Y;\mathcal{G}_{sub},\mathcal{G}_p) + I(Y;\mathcal{G}_p|\mathcal{G}_{sub}) + \beta I(\mathcal{G};\mathcal{G}_{sub}). \tag{13}$$

A.2. Proof of Proposition 1

In this section, we provide a proof for Proposition 1 in the main paper.

For the term $I(Y;\mathcal{G}_p,\mathcal{G}_{sub})$,

$$\begin{aligned}
I(Y;\mathcal{G}_p,\mathcal{G}_{sub}) &= \mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log \frac{p\left(Y|\mathcal{G}_{sub},\mathcal{G}_p\right)}{p(Y)}\right] \\
&\geq \mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log \frac{p\left(Y|\gamma(\mathcal{G}_{sub},\mathcal{G}_p)\right)}{p(Y)}\right].
\end{aligned} \tag{14}$$

We introduce a variational approximation $q_\theta\left(Y|\gamma\left(\mathcal{G}_{sub},\mathcal{G}_p\right)\right)$ of $p\left(Y|\gamma\left(\mathcal{G}_{sub},\mathcal{G}_p\right)\right)$.

$$
\begin{aligned}
I(Y;\mathcal{G}_p,\mathcal{G}_{sub}) &\geq \mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log\frac{q_\theta\left(Y|\gamma(\mathcal{G}_{sub},\mathcal{G}_p)\right)}{p(Y)}\right]+\mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log\frac{p\left(Y|\gamma(\mathcal{G}_{sub},\mathcal{G}_p)\right)}{q_\theta\left(Y|\gamma(\mathcal{G}_{sub},\mathcal{G}_p)\right)}\right]\\
&=\mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log\frac{q_\theta\left(Y|\gamma(\mathcal{G}_{sub},\mathcal{G}_p)\right)}{p(Y)}\right]+\mathbb{E}_{\mathcal{G}_{sub},\mathcal{G}_p}\left[KL\left[p\left(Y|\gamma(\mathcal{G}_{sub},\mathcal{G}_p)\right)\|q_\theta\left(Y|\gamma(\mathcal{G}_{sub},\mathcal{G}_p)\right)\right]\right].
\end{aligned}
\tag{15}
$$

According to the non-negativity of KL divergence, we have:

$$
\begin{aligned}
I(Y;\mathcal{G}_p,\mathcal{G}_{sub}) &\geq \mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log\frac{q_\theta\left(Y|\gamma(\mathcal{G}_{sub},\mathcal{G}_p)\right)}{p(Y)}\right]\\
&=\mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log q_\theta\left(Y|\gamma(\mathcal{G}_{sub},\mathcal{G}_p)\right)\right]-\mathbb{E}_Y\left[\log p(Y)\right]\\
&=\mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log q_\theta\left(Y|\gamma(\mathcal{G}_{sub},\mathcal{G}_p)\right)\right]+H(Y).
\end{aligned}
\tag{16}
$$

Finally, we have the following equation as:

$$
I(Y;\mathcal{G}_p,\mathcal{G}_{sub})\geq\mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log q_\theta\left(Y|\gamma(\mathcal{G}_{sub},\mathcal{G}_p)\right)\right].
\tag{17}
$$

A.3. Decomposition of $I(Y;\mathcal{G}_p|\mathcal{G}_{sub})$ - Eq. 5

For the term $I(Y;\mathcal{G}_p|\mathcal{G}_{sub})$, by definition:

$$
I(Y;\mathcal{G}_p|\mathcal{G}_{sub})=\mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log\frac{p(Y,\mathcal{G}_p|\mathcal{G}_{sub})}{p(Y|\mathcal{G}_{sub})p(\mathcal{G}_p|\mathcal{G}_{sub})}\right].
\tag{18}
$$

By the definition of conditional probability, we have the following equation:

$$
I(Y;\mathcal{G}_p|\mathcal{G}_{sub})=\mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log\frac{p(Y,\mathcal{G}_{sub},\mathcal{G}_p)p(\mathcal{G}_{sub})}{p(Y,\mathcal{G}_{sub})p(\mathcal{G}_{sub},\mathcal{G}_p)}\right].
\tag{19}
$$

We allow the involvement of $\mathcal{G}_p$ in Eq. 19 as follows:

$$
\begin{aligned}
I(Y;\mathcal{G}_p|\mathcal{G}_{sub}) &= \mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log\frac{p(Y,\mathcal{G}_{sub},\mathcal{G}_p)p(\mathcal{G}_{sub})p(\mathcal{G}_p)}{p(Y,\mathcal{G}_{sub})p(\mathcal{G}_{sub},\mathcal{G}_p)p(\mathcal{G}_p)}\right]\\
&=\mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log\frac{p(Y,\mathcal{G}_{sub},\mathcal{G}_p)}{p(Y,\mathcal{G}_{sub})p(\mathcal{G}_p)}+\log\frac{p(\mathcal{G}_{sub})p(\mathcal{G}_p)}{p(\mathcal{G}_{sub},\mathcal{G}_p)}\right]\\
&=\mathbb{E}_{Y,\mathcal{G}_{sub},\mathcal{G}_p}\left[\log\frac{p(Y,\mathcal{G}_{sub},\mathcal{G}_p)}{p(Y,\mathcal{G}_{sub})p(\mathcal{G}_p)}\right]-\mathbb{E}_{\mathcal{G}_{sub},\mathcal{G}_p}\left[\log\frac{p(\mathcal{G}_{sub},\mathcal{G}_p)}{p(\mathcal{G}_{sub})p(\mathcal{G}_p)}\right].
\end{aligned}
\tag{20}
$$

Finally, we have the following equation as:

$$
I(Y;\mathcal{G}_p|\mathcal{G}_{sub})=I(\mathcal{G}_p;Y,\mathcal{G}_{sub})-I(\mathcal{G}_{sub};\mathcal{G}_p).
\tag{21}
$$

A.4. Interpretability Stabilization

A.4.1. Merging Prototypes

Since the number of prototypes for each class is determined before training, some of the learned prototypes may share similar semantics, which negatively affects the model interpretability for which the small size and low complexity are desirable [27, 28]. Inspired by [29], we propose a method to effectively merge the prototypes for graph-structured data, which, in turn, enhances the explanation of the reasoning process and improves performance on downstream tasks while reducing model complexity. The main idea is to merge prototypes based on the similarity between prototype pairs using the embeddings $\mathbf{z}_{\mathcal{G}sub}$. This similarity utilizes all training subgraphs (i.e., $\bigcup_{\mathcal{G}\in\mathcal{X}} \mathcal{G}_{sub}$, where $\mathcal{X}$ is the training set) by measuring the disparity between $g(\mathbf{z}_{\mathcal{G}_{sub}}, \mathbf{z}_{\mathcal{G}_p^i})$ and $g(\mathbf{z}_{\mathcal{G}_{sub}}, \mathbf{z}_{\mathcal{G}_p^j})$ as follows:

$$h(\mathbf{z}_{\mathcal{G}_p^i}, \mathbf{z}_{\mathcal{G}_p^j}) = \left[\sum_{\mathcal{G}\in\mathcal{X}} \left(g(\mathbf{z}_{\mathcal{G}_{sub}}, \mathbf{z}_{\mathcal{G}_p^i}) - g(\mathbf{z}_{\mathcal{G}_{sub}}, \mathbf{z}_{\mathcal{G}_p^j})\right)^2\right]^{-1}. \tag{22}$$

Then, for every pair $(\mathbf{z}_{\mathcal{G}_p^i}, \mathbf{z}_{\mathcal{G}_p^j})$ that falls within the highest ξ percent of similar pairs, the prototype $\mathbf{z}_{\mathcal{G}_p^j}$ and its corresponding weights $\omega(\mathbf{z}_{\mathcal{G}_p^j})$ are removed, and the weights $\omega(\mathbf{z}_{\mathcal{G}_p^i})$ are updated to the sum of $\omega(\mathbf{z}_{\mathcal{G}_p^i})$ and $\omega(\mathbf{z}_{\mathcal{G}_p^j})$. We combine the ξ percentage of the most similar prototype pairs based on the calculated similarity scores.

A.4.2. Prototype Projection

Since the learned prototypes are embedding vectors that cannot be directly interpreted, we project each prototype $\mathbf{z}_{\mathcal{G}_p}$ onto the nearest latent training subgraph from the same class. This process establishes a conceptual equivalence between each prototype and a training subgraph, thereby enhancing interpretability of the prototypes. Specifically, we update prototype $\mathbf{z}_{\mathcal{G}_p}$ of class k $(i.e., \mathbf{z}_{\mathcal{G}_p} \in \mathbb{P}_k)$ by performing the following operation:

$$\mathbf{z}_{\mathcal{G}_p} \leftarrow \arg\min_{\tilde{\mathbf{z}}\in\mathbb{Z}} \|\tilde{\mathbf{z}} - \mathbf{z}_{\mathcal{G}_p}\|_2, \text{ where } \mathbb{Z} = \left\{\tilde{\mathbf{z}} : \text{Readout}\{f_g(\tilde{\mathcal{G}})\}, \tilde{\mathcal{G}} \in \text{Subgraph}(\mathcal{G}_i)\ \forall i \textit{ s.t. } y_i = k\right\}. \tag{23}$$

In the Equation 23, we use Monte Carlo Tree Search (MCTS) [30] to explore training subgraphs $\tilde{\mathcal{G}}$ during prototype projection.

A.4.3. Connectivity Loss

For an input graph $\mathcal{G}$, we construct a node assignment $S_{\mathcal{G}} \in \mathbb{R}^{|\mathcal{V}_{\mathcal{G}}|\times 2}$ based on the probability values that are computed by Equation 2. Specifically, $S_{\mathcal{G}}[j, 0]$ and $S_{\mathcal{G}}[j, 1]$ denote the probability of node $v_j \in \mathcal{V}_{\mathcal{G}}$ belonging to $\mathcal{G}_{sub}$ and $\bar{\mathcal{G}}_{sub}$, respectively. Following [12], poor initialization of the matrix S may result in the proximity of its elements $S[j, 0]$ and $S[j, 1]$ for $\forall v_j \in \mathcal{V}_{\mathcal{G}_i}$, leading to an unstable connectivity of $\mathcal{G}_{sub}$. This instability can have adverse effects on the subgraph generation process. To enhance the interpretability of $\mathcal{G}_{sub}$ by inducing a compact topology, we utilize a batch-wise loss function as follows:

$$\mathcal{L}_{\text{con}} = \|\text{Norm}(S_{\text{B}}^T \mathbf{A}_{\text{B}} S_{\text{B}}) - I_2\|_F \tag{24}$$

where $S_{\mathrm{B}} \in \mathbb{R}^{\sum_{i=1}^{n} |\mathcal{V}_{\mathcal{G}_i}| \times 2}$ and $\mathbf{A}_{\mathrm{B}} \in \mathbb{R}^{\sum_{i=1}^{n} |\mathcal{V}_{\mathcal{G}_i}| \times \sum_{i=1}^{n} |\mathcal{V}_{\mathcal{G}_i}|}$ are the node assignment and the adjacency matrix at the batch level, respectively. I_2 is 2-by-2 identity matrix, $\|\cdot\|_F$ is the Frobenius norm and $\mathrm{Norm}(\cdot)$ is the row normalization. Minimizing $\mathcal{L}_{\mathrm{con}}$ indicates that if v_j is in $\mathcal{G}_{sub}$ its neighbors also have a high probability to be in $\mathcal{G}_{sub}$, while if v_i is in $\mathcal{G}_{sub}$, its neighbors have a low probability to be in $\bar{\mathcal{G}}_{sub}$.

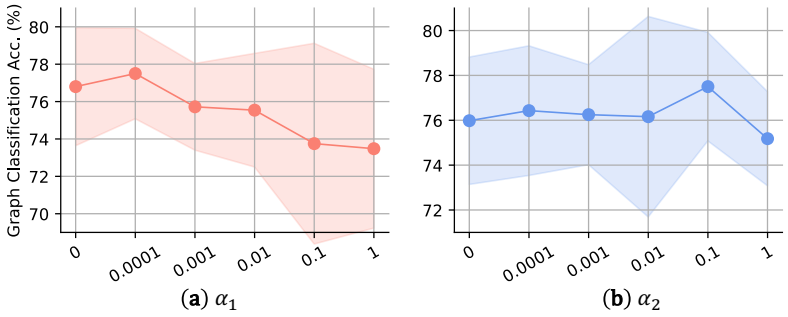
A.5. Additional Experiments

In this section, we present our additional experiments including hyperparameter Analysis (Section A.5.1) ablation study (Section A.5.2), analysis of the different graph readout functions (Section A.5.3), fidelity scores over sparsity (Section A.5.4), analysis of the hyperparameters α_1, α_2, α_3 and J (Section A.5.5) and reasoning process (Section A.5.6). All of our experiments were performed with one NVIDIA GeForce A6000.

A.5.1. Hyperparameter Analysis

In Figure 4, we conduct a sensitivity analysis on the hyperparameters α_1 and α_2 of the final loss (Equation **??**) relevant to mutual information. Note that α_1 and α_2 are related to minimizing $I(\mathcal{G}; \mathcal{G}_{sub})$ and maximizing $I(\mathcal{G}_{sub}; \mathcal{G}_p)$, respectively. **1)** Figure 4(a) shows a significant decrease in performance when α_1 becomes large, i.e., when the model focuses on compressing the subgraphs. This is because too much compression of subgraphs results in the loss of important information, ultimately having a negative impact on the downstream performance. However, when $\alpha_1 = 0$ (i.e., $\mathcal{G} = \mathcal{G}_{sub}$; no compression at all), uninformative information would be included in $\mathcal{G}_{sub}$, which incurs a performance degradation. **2)** Figure 4(b) visualizes the change in performance depending on α_2. A small value of α_2 prevents sufficient transmission of information from $\mathcal{G}_{sub}$ to $\mathcal{G}_p$, whereas excessive value of α_2 allows the influence of $\mathcal{G}_{sub}$ to dominate the prototypes $\mathcal{G}_p$, both of which lead to a performance deterioration.

Figure 4: Impact of α_1 and α_2 on PROTEINS dataset.

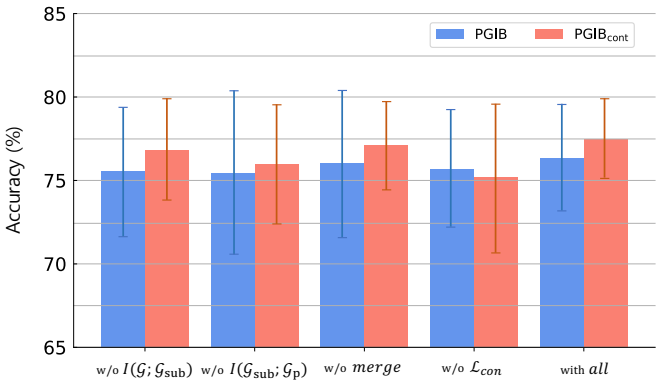


(**a**) α_1 (**b**) α_2

A.5.2. Ablation Study

We perform ablation studies to examine the effectiveness of our model (i.e., PGIB and PGIB$_{\text{cont}}$). In Figure 5, the "*with all*" setting represents our final model that includes all the components. We conducted ablation studies on losses related to mutual information (i.e., $I(\mathcal{G};\mathcal{G}_{sub})$ and $I(\mathcal{G}_{sub};\mathcal{G}_p)$), merging prototypes, and the connectivity loss $\mathcal{L}_{\text{con}}$. We have the following observations: **1)** The performance of the models significantly deteriorates when the terms related to mutual information, $I(\mathcal{G};\mathcal{G}_{sub})$ and $I(\mathcal{G}_{sub};\mathcal{G}_p)$, are not considered, compared to our final model. Specifically, if we exclude the consideration of $\mathcal{G}_{sub}$ when constructing the prototypes $\mathcal{G}_p$ (i.e., without maximizing $I(\mathcal{G}_{sub};\mathcal{G}_p)$), the representations of the prototypes that directly contribute to the final predictions cannot effectively obtain the informative information from the subgraph $\mathcal{G}_{\text{sub}}$, resulting in a deterioration of performance. Moreover, if we fail to incorporate the minimal sufficient information from the entire graph $\mathcal{G}$ into $\mathcal{G}_{sub}$ (i.e., without minimizing $I(\mathcal{G};\mathcal{G}_{sub})$), there is a higher likelihood of prototypes obtaining uninformative information, which can ultimately lead to a deterioration in performance. **2)** Merging prototypes improves both PGIB and PGIB$_{\text{cont}}$ by enhancing the distinguishability of the remaining prototypes. This process not only enhances the interpretability of the prototypes but also results in improved classification accuracy. By merging similar prototypes, important features are emphasized through the aggregation of weights from both prototypes. This results in a more precise and effective representation of the data, enhancing the model's interpretability and accuracy in classification performance (i.e., "*with all*" setting performs better than "*w/o merge*" setting.). **3)** The connectivity loss $\mathcal{L}_{\text{con}}$, which promotes the construction of more realistic subgraphs by inducing a compact topology, has a significant impact on the performance. This improvement can be attributed to the fact that subgraphs relevant to the target often form connected components in real-world datasets. Therefore, incorporating the connectivity loss leads to improved performance by ensuring that the subgraph maintains realistic connectivity patterns.

Figure 5: Ablation studies on PGIB.



A.5.3. Analysis of the Different Graph Readout Functions

We conduct experiments on graph classification using different readout functions for PGIB. In Table 3, we show the classification performance based on three readout functions: max-pooling,

mean-pooling, and sum-pooling, for both the PGIB and $\text{PGIB}_{\text{cont}}$. Table 3 demonstrates that max-pooling achieves the best performance in all datasets except for MUTAG dataset.

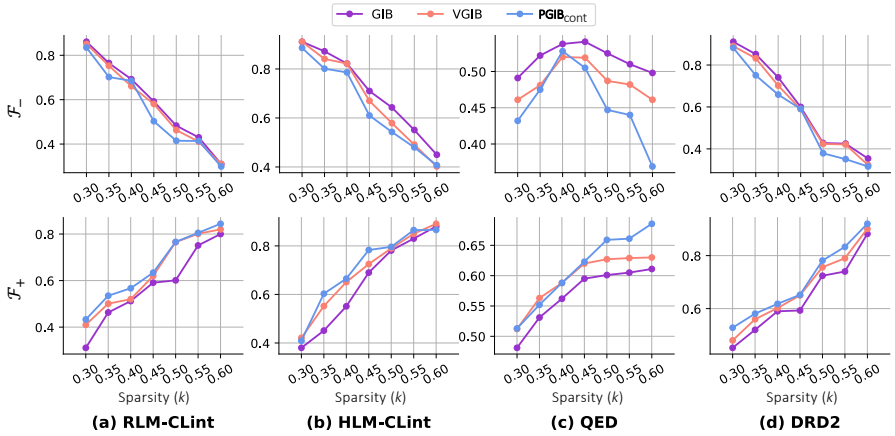
Table 3
Evaluation on the Different Graph Readout Functions (accuracy).

Dataset	Methods					
	PGIB			$\text{PGIB}_{\text{cont}}$		
	MaxPool	MeanPool	SumPool	MaxPool	MeanPool	SumPool
MUTAG	80.50 ± 7.07	**86.50 ± 7.84**	80.50 ± 10.39	85.50 ± 5.22	**88.50 ± 6.34**	86.50 ± 7.43
PROTEINS	**77.14 ± 2.19**	72.32 ± 5.17	60.89 ± 12.07	**77.50 ± 2.42**	68.39 ± 4.40	66.07 ± 4.79
NCI1	**77.65 ± 2.20**	77.59 ± 7.41	63.96 ± 8.37	**78.25 ± 2.13**	77.52 ± 2.94	61.82 ± 3.96
DD	**73.36 ± 1.80**	67.56 ± 4.62	68.99 ± 4.56	**73.70 ± 2.14**	63.78 ± 5.40	64.12 ± 6.50

A.5.4. Fidelity Scores over Sparsity

Figure 6 visualizes the comparison of fidelity scores over various sparsity scores of subgraphs. To ensure a fair comparison, the fidelity scores are compared under the same subgraph sparsity, as the difference between the predictions of the original graph and subgraph strongly depends on the level of sparsity. We observe that $\text{PGIB}_{\text{cont}}$ achieves the best performance in most sparsity environments on the four datasets.

Figure 6: Comparisons of fidelity scores over sparsity scores k.



A.5.5. Analysis of the Hyperparameters

We have conducted additional qualitative analysis. In this analysis, we compare the effects of different choices for α_1, α_2, α_3 (i.e., loss weights) and J (i.e., the number of prototypes for each class) at a more fine-grained level.

A.4.4.1 Visualization of $\mathcal{G}_{sub}$ based on α_1

Figure 4 shows a large value of α_1 reduces the performance. For further analysis, we visualize the subgraph $\mathcal{G}_{sub}$ at different values of α_1. The parameter α_1 has an impact on the compression of the subgraph from the entire input graph. In Figure 7, when α_1 is large, $\mathcal{G}_{\text{sub}}$ receives too much compressed information from $\mathcal{G}$, causing the loss of important data. It prevents $\mathcal{G}_p$ from containing label-relevant information, ultimately resulting in a negative impact on downstream performance.

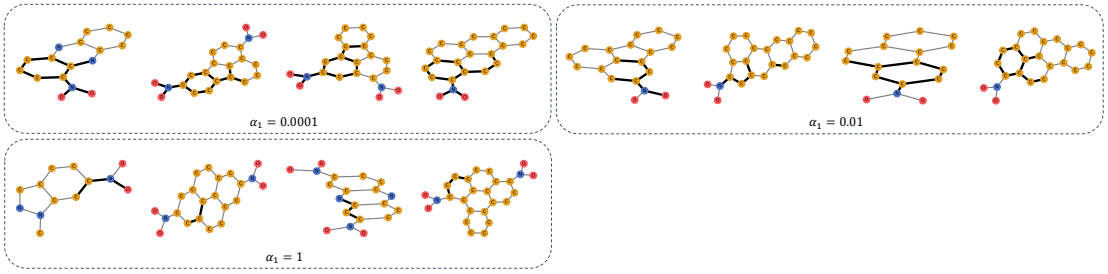


Figure 7: Visualization of $\mathcal{G}_{sub}$ based on α_1 on MUTAG dataset

A.4.4.2 Visualization of $\mathcal{G}_p$ based on α_2

We have performed the qualitative analysis on α_2 to provide a better understanding of its impact. It is crucial that the prototypes not only contain key structural information from the input graph but also ensure a certain level of diversity since each class is represented by multiple prototypes. In Figure 8, when we fix α_1 at 0.1, the diversity of prototypes varies based on the degrees of α_2. Specifically, when α_2 becomes 1, the diversity of prototypes decreases, leading to a decline in the interpretability of the reasoning process and the overall model performance. This finding highlights the importance of selecting proper α_2 to ensure both interpretability and performance are optimized.

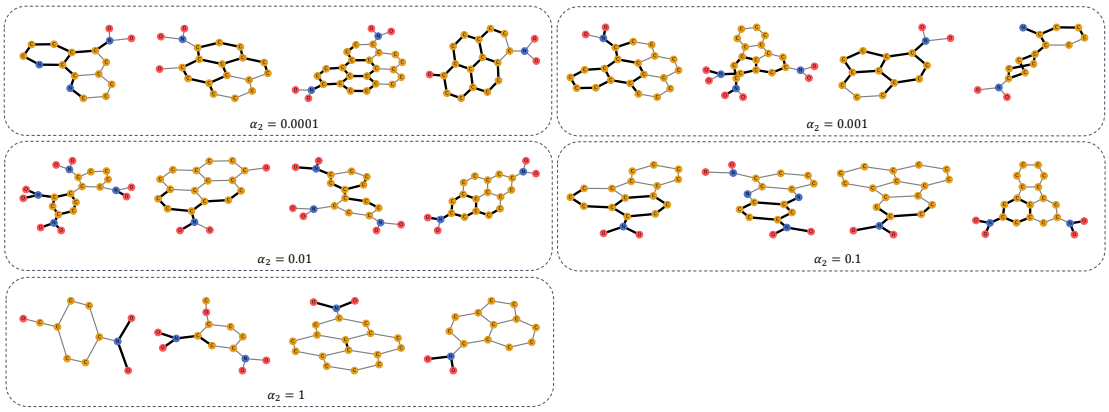


Figure 8: Visualization of $\mathcal{G}_p$ based on α_2 on MUTAG dataset

A.4.4.3 Visualization of $\mathcal{G}_{sub}$ based on α_3

We mentioned that α_3 is associated with the connectivity loss and plays a crucial role in influencing the interpretability of $\mathcal{G}_{sub}$ by promoting compact topology in Section **??**. To verify this, we visualize the subgraph $\mathcal{G}_{sub}$ at different values of α_3. In Figure 9, when we exclude the connectivity loss from the loss function (i.e., set α_3 to 0), $\mathcal{G}_{sub}$ tends to consist of non-connected components. As a result, due to the wide and scattered range of detected subgraphs, the absence of connectivity loss results in the formation of unrealistic subgraphs.

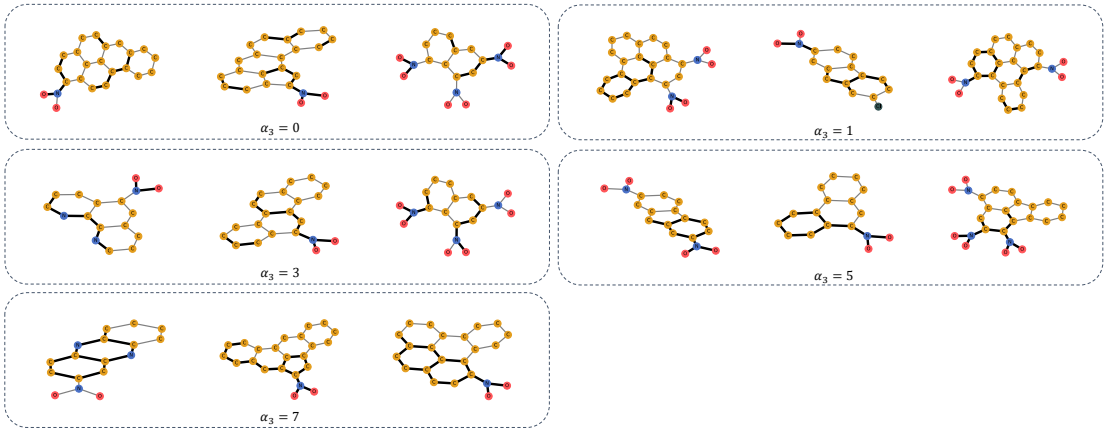


Figure 9: Visualization of $\mathcal{G}_{sub}$ based on α_3 on MUTAG dataset

A.4.4.4 Visualization of $\mathcal{G}_p$ based on J

We conducted interpretation visualizations of $\mathcal{G}_p$ based on the number of prototypes for each class in Figure 10. When the number of prototypes is small (as seen in the case with 3 prototypes), the prototypes do not contain diverse substructures. This limitation arises from making predictions using a restricted number of prototypes. On the other hand, if the number of prototypes is large (as shown in the case with 9 prototypes), a greater diversity of prototypes can be achieved because various and complex information can be obtained from $\mathcal{G}_{sub}$.

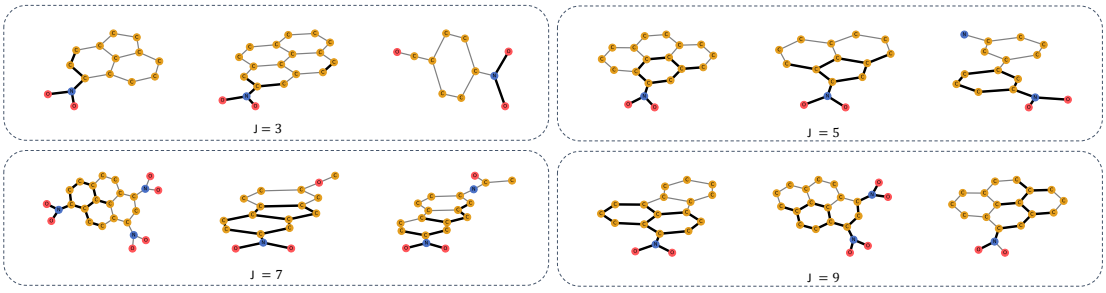


Figure 10: Visualization of $\mathcal{G}_p$ based on the number of prototypes (J) on MUTAG dataset

A.5.6. Reasoning Process

We illustrate the reasoning process on two datasets, i.e., MUTAG and BA2Motif, in Figure 11. PGIB detects important subgraphs, and obtains similarity scores between subgraph $\mathcal{G}_{sub}$ and prototype $\mathcal{G}_p$. Then, PGIB computes the "points contributed" to predicting each class by multiplying the similarity score between $\mathcal{G}_{sub}$ and $\mathcal{G}_p$, with the weight assigned to each prototype in the prediction layer. Lastly, PGIB outputs the class with the highest total point among all the classes. We have the following observations in the reasoning process: **1)** PGIB identifies the specific substructures within $\mathcal{G}$ that contain label-relevant information by extracting $\mathcal{G}_{sub}$ from $\mathcal{G}$. **2)** PGIB identifies which training graphs play a crucial role in the predictions by conducting prototype projection to visualize the training graph that closely resembles the prototype. In other words, since each prototype is projected onto the nearest training graph, we can identify the training graph that had the most influence on predicting the target graph through the prototypes. **3)** PGIB identifies the influence of each "points contributed" on the final prediction by examining the total point to each class.

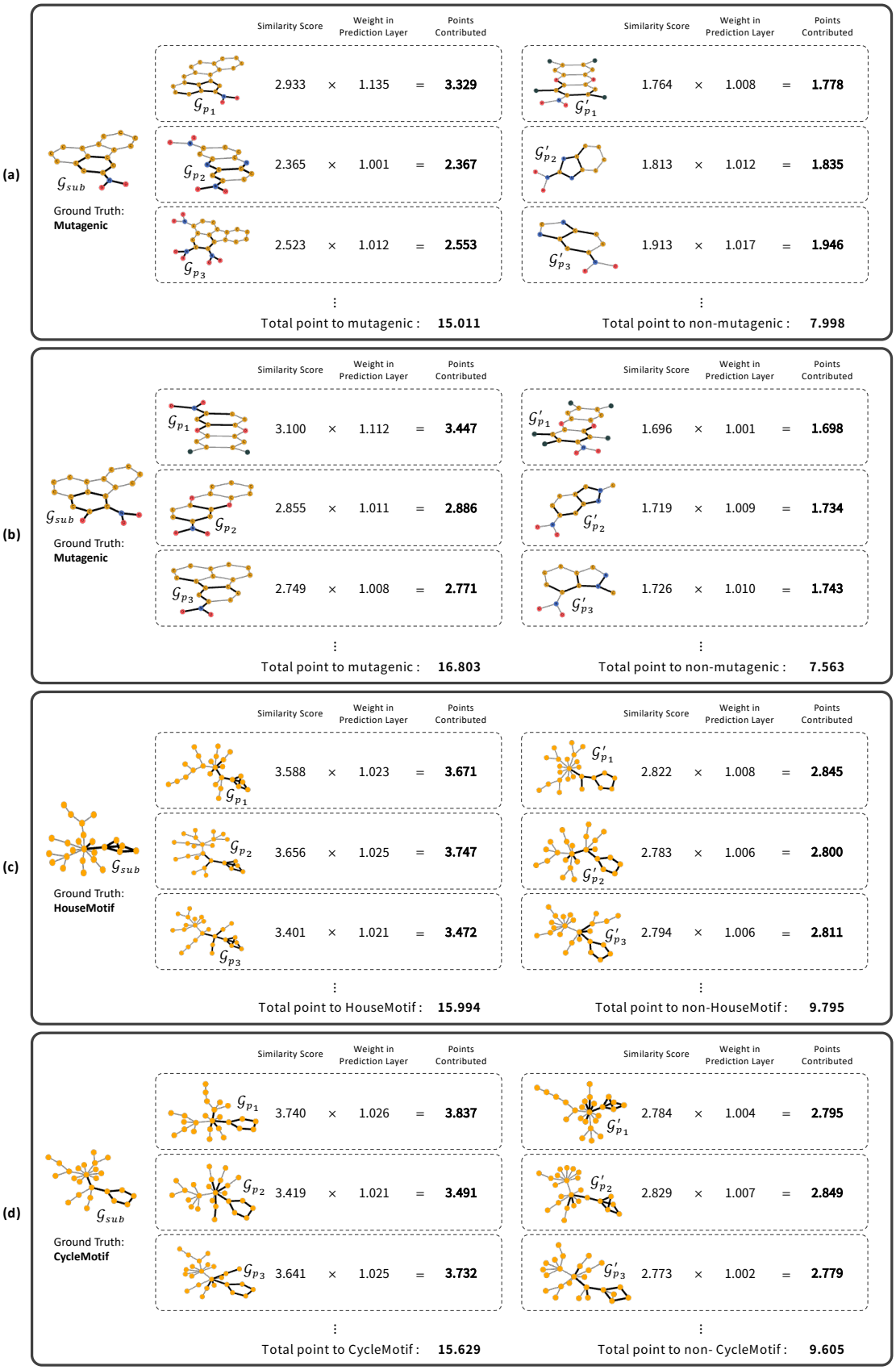


Figure 11: Reasoning process on MUTAG (a-b) and BA-2Motifs (c-d) datasets.

A.6. Baselines

In this part, we provide details on the baselines used in our experiments.

- **ProtGNN** [10] utilizes prototypes to explain prediction results by potentially representing training graphs in graph neural networks. Specifically, ProtGNN measures the similarity between the embedding in the input graph and each prototype, and makes predictions based on the similarity. Additionally, ProtGNN can be extended to ProtGNN+ by incorporating a module that samples the subgraphs from the input graph to visualize the most similar subgraph to each prototype.
- **GIB** [12] utilizes the information bottleneck principle to detect important subgraphs in graph-structured data. Specifically, this method aims to extract subgraph embeddings by restricting the amount of information within the subgraph and retaining only the important information. During the training of graph neural networks, GIB encourages the recognition of important subgraphs within the graph data and performs graph classification tasks based on this recognition.
- **VGIB** [14] introduces noise injection into the graph information bottleneck. VGIB aims to enhance subgraph recognition by incorporating randomness into the graph data. This addition of randomness helps acquire diverse subgraph representations and captures the inherent uncertainty in the data, leading to enhanced performance in both classification tasks and subgraph recognition tasks.
- **GSAT** [21] aims to achieve interpretable and generalizable graph learning through a stochastic attention mechanism. It probabilistically estimates attention weights for the relationships between nodes and edges in a graph. The probabilistic attention mechanism enables the model to learn shared characteristics across domains and enhance its generalization performance.
- **GNNexplainer** [1] is a post-hoc interpretation model designed to interpret the prediction results of GNN models. Specifically, GNNExplainer optimizes a masking algorithm to maximize the mutual information with the existing label information. Its goal is to make the masked subgraph's prediction as close as possible to the original graph, which helps to detect substructures significant for predictions.
- **PGexplainer** [4] parameterizes the underlying structure as an edge distribution and generates the explanatory graph by sampling. PGExplainer collectively explains multiple instances by utilizing deep neural networks to parameterize the process of generating explanations. It enables the interpretability of the GNN model's behavior by adjusting the weight parameters of the GNN model, which allows it to be readily applied in an inductive setting.

Table 4
Source code links of the baseline methods

Methods	Source code
ProtGNN	https://github.com/zaixizhang/ProtGNN
GIB	https://github.com/Samyu0304/graph-information-bottleneck-for-Subgraph-Recognition
VGIB	https://github.com/Samyu0304/Improving-Subgraph-Recognition-with-Variation-Graph-Information-Bottleneck-VGIB-
GSAT	https://github.com/Graph-COM/GSAT
GNNExplainer	https://github.com/RexYing/gnn-model-explainer
PGExplainer	https://github.com/flyingdoog/PGExplainer

A.7. Datasets

In this section, we provide details on the datasets used during training.

- **MUTAG** [22] consists of 188 molecular graphs, which are used to predict the properties of mutagenicity in chemical structures. The graph labels are determined by the mutagenicity of Salmonella typhimurium.
- **PROTEINS** [23] includes 1113 protein structures and is utilized for the classification of proteins into enzymes or non-enzyme. Each node represents an amino acid in the protein molecule, and edges connect nodes if the distance between the amino acids is less than 6 angstroms.
- **NCI1** [24] contains 4110 chemical compounds specifically designed for anticancer testing. Each chemical compound is labeled as either positive or negative based on its response to cell lung cancer.
- **DD** [25] consists of 1178 protein structures labeled as either an enzyme or a non-enzyme, similar to the previous dataset.
- **BA2Motifs** [4] is a synthetic dataset used for graph classification. Each graph is constructed based on a random graph generated using the Barabási–Albert (BA) model. It is then connected to one of two types of motifs: a house motif and a five-node cycle motif. The label of each graph is determined to belong to one of two classes based on the attached motif.
- **ZINC** [31] is a database of commercially accessible compounds used for virtual screening. It contains over 230 million purchasable compounds in a 3D format that can be docked readily.

A.8. Limitations and Societal Impacts

PGIB does not incorporate domain knowledge, so domain-specific information cannot be conveyed to the extracted subgraph. For example, the extracted key subgraph may not necessarily correspond to a biologically or chemically existing functional group. It can cause unrealistic subgraphs to significantly affect the overall training of the model, including prototype training and final performance.

With the advancement and increasing sophistication of explainable artificial intelligence (XAI), these limitations may have a broader societal impact. There is a potential risk of excessive dependence on XAI systems, leading to a decrease in human autonomy and decision-making. Blindly accepting the decisions of AI systems without critically evaluating XAI undermines human judgment and agency, which can potentially result in inappropriate or harmful behavior. For example, if a non-existent functional group is unquestioningly accepted from the model, it can lead to an erroneous understanding of the algorithm as a whole, with incorrect judgment about the functional group.

A.9. Algorithm

Algorithm 1: Overview of PGIB Training

Input: Training dataset $\{(\mathcal{G}_i, y_i)\}_{i=1}^{n}$, prototype merge epoch T_m, prototype merge period τ, the number of prototypes M, the number of classes K, hyper-parameters of the weights of the losses α_1, α_2, and α_3

for *training epochs* $t = 1, 2, \ldots, T$ **do**
- $\mathcal{G}_{sub} \leftarrow \arg\min_{\mathcal{G}_{sub}} I(\mathcal{G}; \mathcal{G}_{sub})$ by injecting noise into subgraph in Eq. 3 // $\mathcal{L}_{\text{MI}}^1$
- Evaluate the loss $\mathcal{L}_{\text{con}}$ in Eq. 24
- $r_m \leftarrow g(\mathbf{z}_{\mathcal{G}_{sub}}, \mathbf{z}_{\mathcal{G}_p}^m)$
- Minimize $-I(\mathcal{G}_{sub}; \mathcal{G}_p)$ in Eq. 6 or 7 // $\mathcal{L}_{\text{MI}}^2$
- Evaluate the loss $\mathcal{L}_{\text{cls}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{K}\mathbb{I}(y_i = c)\log(\pi_c)$
- **if** *Merge* $=$ *True* ***and*** $t > T_m$ ***and*** $t\%\tau = 0$ **then**
  - Calculate prototype similarity $h(\mathbf{z}_{\mathcal{G}_p^i}, \mathbf{z}_{\mathcal{G}_p^j}) = \left[\sum_{\mathcal{G}\in\mathcal{X}}(g(\mathbf{z}_{\mathcal{G}_{sub}}, \mathbf{z}_{\mathcal{G}_p^i}) - g(\mathbf{z}_{\mathcal{G}_{sub}}, \mathbf{z}_{\mathcal{G}_p^j}))^2\right]^{-1}$
  - Perform prototype-merge
- **end**
- Total loss $\mathcal{L} = \mathcal{L}_{\text{cls}} + \alpha_1\mathcal{L}_{\text{MI}}^1 + \alpha_2\mathcal{L}_{\text{MI}}^2 + \alpha_3\mathcal{L}_{\text{con}}$
- Update model parameters by gradient descent

end