Enhancing Graph Learning Interpretability through Modulating Cluster Information Flow

Jiayi Yang[a], Wei Ye[b,*], Xin Sun[c], Rui Fan[b], Jungong Han[d]

[a]*College of Electronic and Information Engineering, Tongji University, Shanghai, 201804, China*
[b]*College of Electronic and Information Engineering, Shanghai Institute of Intelligent Science and Technology, State Key Laboratory of Autonomous Intelligent Unmanned Systems, Tongji University, Shanghai, 201804, China*
[c]*Faculty of Data Science, City University of Macau, Macau, 999078, China*
[d]*Department of Automation, Tsinghua University, Beijing, 100084, China*

Abstract

Interpretable graph learning is essential for scientific applications that depend on learning models to extract reliable insights from graph-structured data. Recent efforts to explain GNN predictions focus on identifying vital substructures, such as subgraphs. However, existing approaches tend to misclassify the neighboring irrelevant nodes as part of the vital subgraphs. To address this, we propose **C**luster **I**nformation **Flow** **G**raph **N**eural **N**etworks (CIFlow-GNN), a *built-in* model-level method that provides accurate interpretable subgraph explanations by modulating the cluster information flow. CIFlow-GNN incorporates two modules, i.e., the graph clustering module and the cluster prototype module. The graph clustering module partitions the nodes according to their connectivity in the graph topology and their similarity in cluster features. Specifically, we introduce a cluster feature loss to regulate information flow at the cluster level. We prove that the proposed cluster feature loss is a lower bound of the InfoNCE loss. Optimizing the cluster feature loss reduces the mutual information among clusters and achieves the modulation of cluster information flow. Subsequently, the graph prototype module uses prototypes as a bridge to select important clusters as vital subgraphs by integrating information across all graphs. To ensure accurate corre-

*Corresponding author
Email addresses: 2111125@tongji.edu.cn (Jiayi Yang), yew@tongji.edu.cn (Wei Ye), sunxin1984@ieee.org (Xin Sun), ranger_fan@outlook.com (Rui Fan), jghan@tsinghua.edu.cn (Jungong Han)

spondence between clusters and prototypes, we further modulate the cluster information flow at the prototype level. Experimental studies on both synthetic and real-world datasets demonstrate that our proposed CIFlow-GNN can identify vital subgraphs effectively and efficiently.

Keywords: Graph neural networks, model-level explanation, cluster information flow, graph clustering, graph classification

---

1. Introduction

Graph neural networks (GNNs) [1–3] have gained significant attention due to their effectiveness in learning node and graph representations. Graph classification [4] is a fundamental problem in GNNs with applications in fields such as biochemistry [5] and social network analysis [6]. However, most GNNs are black-box models that lack the ability to explain their predictions in the graph classification task, limiting their applicability in critical areas like medical diagnosis [6]. Therefore, it is necessary to investigate the interpretability of GNNs.

Over the past few years, interest has been growing in exploring interpretable GNNs. These explainers provide explanations for GNNs by identifying the vital substructures, such as edges [7–9], nodes [10–12], and subgraphs [13–16], which are most relevant to the graph label. Compared with explainers focusing on edges or nodes, those that use subgraphs usually impose explicit connectivity constraints among nodes. Some works [13, 14] extract a single connected subgraph for each graph, while others [15, 16] employ regularization terms to capture multiple connected components as vital subgraphs. Since graph properties are often determined by subgraphs, using subgraphs as intrinsic explanations for graph predictions is more intuitive and effective. For example, in molecular graphs, the functional groups -NO_2 and -NH_2 play a critical role in mutagenicity, while -OH impacts solubility. Current research in interpreting GNNs through subgraph recognition can be categorized into two main branches: *post-hoc* approaches and *built-in* approaches. *Post-hoc* approaches first pretrain GNNs and then optimize a separate explanatory module to generate explanations while freezing the pretrained GNNs. However, since the explanatory module is not jointly trained with

the pretrained GNNs, it may fail to provide or even approximate the optimal solution [17]. In contrast, *built-in* methods jointly optimize the GNNs and the explanatory module to enhance graph learning interpretability.

Nevertheless, research on *built-in* approaches for subgraph recognition remains limited. ProtGNN [14] assigns prototypes to each class and makes predictions for input graphs by measuring the similarity between the graph embeddings and the trained prototype embeddings. It then introduces a prototype projection module that maps each prototype to the nearest training subgraphs as explanations using Monte Carlo Tree Search (MCTS) [18]. Since the prototypes in ProtGNN capture graph-level characteristics, the identified subgraphs may include uninformative substructures. Recently, some approaches [15, 16, 19] have leveraged the concept of Graph Information Bottleneck (GIB) [19, 20] to identify compressed yet informative subgraphs. Compared with ProtGNN, they first partition the input graph into the predicted vital subgraphs and irrelevant complements. Then, they employ a selective noise injection method that injects less noise into the predicted vital subgraphs, while injecting more noise into the complements. The noise injection method, combined with the GIB objective function, modulates the label-relevant information flow from the input graph to the predicted subgraphs in the graph learning process. Concurrently, by preserving more label-relevant information in the selected node representations, the identification of vital subgraphs is enhanced. In addition, PGIB [16] conveys information of vital subgraphs to the prototypes and makes predictions based on the similarity between the identified subgraph embeddings and the trained prototype embeddings.

However, nodes adjacent to the vital subgraphs inevitably aggregate information from them during message passing in GNNs, leading to their misclassification as part of the vital subgraphs. As shown in Fig. 1, since the GIB approaches cannot prevent this issue, nodes on the ring structures connected to the vital functional groups receive label-relevant information and are mistakenly predicted as vital subgraphs. Therefore, it is crucial to reduce the information flow between vital subgraphs and their complements, especially the irrelevant neighboring nodes. From the perspective of clustering, the similarities of nodes in each vital subgraph should be strengthened while those between vital subgraphs and their complements should be weakened. In this paper,

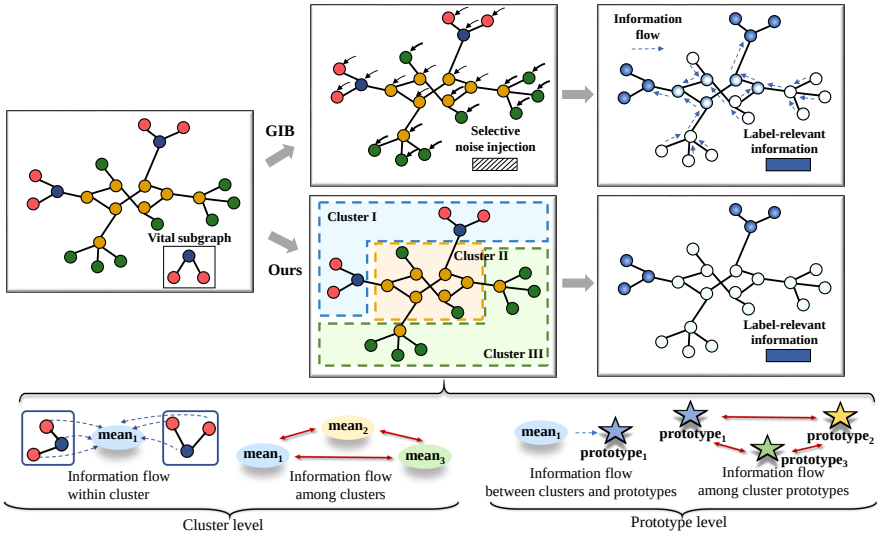


Figure 1: Comparison of the information flow of GIB methods and the cluster information flow of CIFlow-GNN. The GIB methods employ a selective noise injection method to modulate the label-relevant information flow among nodes from the input graph to the predicted subgraphs. In contrast, our method modulates cluster information flow by identifying clusters and adjusting inter-/intra-cluster interactions at both the cluster and prototype levels.

we focus on identifying the most important cluster as the vital subgraphs rather than extraneous neighboring nodes.

To this end, we propose **C**luster **I**nformation **Flow** **G**raph **N**eural **N**etworks (CIFlow-GNN), a *built-in* model-level method that provides vital subgraphs as explanations for GNNs by modulating the cluster information flow. In general, CIFlow-GNN introduces a novel graph clustering module to partition the graphs and a model-level cluster prototype module to select the most important clusters as the vital subgraphs for prediction. In the graph clustering module, we use spectral clustering [21] to partition the nodes into different clusters according to the graph topology. Simultaneously, we modulate the cluster information flow at the cluster level as shown in Fig. 1, by pulling nodes in each cluster close to their cluster centroid and pushing apart different cluster centroids. Theoretically, we prove that our cluster feature loss is a lower bound of the InfoNCE loss [22]. By optimizing the cluster feature loss, the mutual information among clusters is reduced, realizing the modulation of cluster information flow.

Subsequently, in the cluster prototype module, we design CIFlow-GNN as a model-level explainer. We integrate information from all graphs to refine the clusters and generate cluster importance masks for each graph. To achieve this, we classify clusters into shared prototypes and introduce two learnable matrices to learn the relative importance of prototypes within and across classes. It is worth noting that the prototypes act as a bridge between classes and clusters, therefore improper management of the cluster-prototype correspondence can lead to degradation in the accuracy of the cluster importance mask. To mitigate this, we modulate the cluster information flow at the prototype level. To be specific, we strengthen the connection between clusters and prototypes while simultaneously forcing diverse prototypes. Finally, we select the cluster corresponding to the highest score in the cluster importance mask as the interpretable vital subgraph, which may contain multiple connected components. Our main contributions are as follows:

- We enhance the interpretability of graph learning by modulating the cluster information flow at both the cluster and prototype levels.
- We propose a model-level approach that leverages prototypes as a bridge to identify the important clusters as vital subgraphs for each graph.
- Empirical results show that CIFlow-GNN outperforms other state-of-the-art GNN explainers on various datasets in the graph interpretation and classification tasks.

2. Related Work

As the application of GNNs continues to expand, understanding the rationale behind their predictions becomes increasingly critical. Existing research on GNN interpretation can be categorized into two branches: *post-hoc* approaches and *built-in* approaches, with most focusing on the former. Among the *post-hoc* approaches, PGExplainer [23] learns a parameterized mask generator to identify important edges, whereas GAFExplainer [24] improves the interpretability of edge masks through node attribute augmentation and feature fusion. EiG-Search [7] proposes a training-free,

linear-complexity method that ranks edges by importance. GSCExplainer [8] adopts edge-cut partitioning and contrastive learning to alleviate the absence of ground-truth explanations. ConfExplainer [9] explicitly integrates confidence estimation into the explanation generation process by introducing a confidence matrix to quantify the reliability of each edge. Recent works extend the explanation granularity from edges to subgraphs for a more holistic understanding of GNN decisions. SubgraphX [13] leverages Shapley values to access subgraph importance and leverages MCTS [18] for efficient exploration. PAGE [25] clusters graph embeddings using a Gaussian mixture model and selects the top k class-discriminative graph embeddings near the cluster centroid. It then iteratively searches for a common subgraph for a given class among the selected graphs via a scoring function. FORGE [26] enhances the accuracy and faithfulness of explanations by leveraging higher-order structural information. GraphTrail [27] evaluates the global importance of subgraph concepts via Shapley values, and employs symbolic regression to translate GNN predictions into human-interpretable Boolean formulas. While Shapley values focus on individual node importance, they neglect inter-node interactions. To overcome this limitation, MAGE [28] employs the Myerson-Taylor interaction index to embed graph structure into node and interaction attributions. Furthermore, FlowX [29] treats all continuous node sequences as message flows and explains GNNs by identifying and quantifying important message flows using Shapley values via a flow sampling scheme. In contrast, CIFlow-GNN depicts the cluster information flow from a macro perspective, akin to GIB methods [15, 16, 19].

Far fewer works address *built-in* interpretability. SUNNY-GNN [30] leverages structural augmentation and contrastive learning to generate sufficient and necessary explanations. Recently, GIB methods [15, 16, 19] have been proposed to encourage identified subgraphs to be informative to the graph labels while minimizing mutual information with the input graphs, based on the information bottleneck theory [20]. However, these methods struggle to classify vital subgraphs from adjacent nodes correctly. To address this, GIP [31] employs Normalized Cut (Ncut) [21, 32] to partition graphs into clusters and uses random walk graph kernels to compute similarity between clusters and T learnable subgraphs. Our method is also inspired by the Ncut problem and introduces a robust orthogonal loss function to enhance clustering performance.

3. Preliminaries

Notations. A graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ denotes the set of vertices, $\mathcal{E}$ is the set of edges, and $\mathbf{X} = [\mathbf{x}_1; \ldots; \mathbf{x}_n] \in \mathbb{R}^{n \times r}$ is the feature matrix. Each graph $\mathcal{G}$ is labeled by a ground-truth one-hot vector $\mathbf{y} \in \{0, 1\}^C$, where C is the number of graph classes. The adjacency and degree matrices are $\mathbf{A}, \mathbf{D} \in \{0, 1\}^{n \times n}$, respectively. The normalized Laplacian matrix is defined as $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{I}$ is the identity matrix. Matrix $\mathbf{L}$ can be decomposed as $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal$, where $\mathbf{U} \in \mathbb{R}^{n \times n}$ contains the eigenvectors, and $\mathbf{\Lambda} = \text{diag}([\lambda_1, \ldots, \lambda_n])$ is the diagonal matrix of eigenvalues. For a graph with self-loops, the adjacency matrix is modified to $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and the degree matrix becomes $\widetilde{\mathbf{D}}$. The normalized adjacency matrix with self-loops $\widetilde{\mathbf{W}}$ is expressed as $\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$. Detailed notations used in this paper are listed in Table 1.

Table 1: Notations and their descriptions.

Notations	**Descriptions**
K	Number of clusters of $\mathcal{G}$
C	Number of graph classes
M	Number of prototypes for all the graphs
L	Number of node embedding layers
$\mathbf{X} \in \mathbb{R}^{n \times r}$	Raw node features of $\mathcal{G}$
$\mathbf{A}, \mathbf{D}, \mathbf{L} \in \mathbb{R}^{n \times n}$	Adjacency/ Degree/ Normalized Laplacian matrix of $\mathcal{G}$
$\widetilde{\mathbf{W}} \in \mathbb{R}^{n \times n}$	Normalized adjacency matrix with self-loops of $\mathcal{G}$
$\mathbf{H} \in \mathbb{R}^{n \times d}$	Node embeddings of $\mathcal{G}$
$\mathbf{S} \in \mathbb{R}^{n \times K}$	Soft cluster assignment matrix of $\mathcal{G}$
$\boldsymbol{\mu}_k \in \mathbb{R}^d$	Centroid embedding for the k-th cluster of $\mathcal{G}$
$\mathbf{E} \in \mathbb{R}^{K \times d}$	Cluster embeddings of $\mathcal{G}$
$\mathbf{Q} \in \mathbb{R}^{K \times M}$	Prototype assignment matrix of $\mathcal{G}$
$\mathbf{P}_{\text{intra}}, \mathbf{P}_{\text{inter}}, \mathbf{P} \in \mathbb{R}^{C \times M}$	Intra/ Inter/ Whole correspondence matrices for all the graphs
$\mathbf{o} \in \mathbb{R}^K$	Cluster importance mask of $\mathcal{G}$
$\mathbf{M} \in \mathbb{R}^{M \times d}$	Prototype embeddings for all the graphs
$\mathbf{E}_{\mathcal{G}}, \widetilde{\mathbf{E}}_{\mathcal{G}} \in \mathbb{R}^{Kd}$	Graph embeddings/ Masked graph embeddings of $\mathcal{G}$
$\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^C$	Class prediction/ Masked class prediction of $\mathcal{G}$

GNN Explanation. In the graph classification task, a GNN model f is trained to classify a graph $\mathcal{G}$ into its corresponding graph label $\mathbf{y}$, i.e., $f(\mathcal{G}) \longmapsto \mathbf{y}$. In this paper, we focus on discovering vital subgraphs $\mathcal{G}_{\text{sub}}$ that has the greatest impact on the label of graph $\mathcal{G}$, serving as explanations for the graph. From a mutual information perspective, we maximize the mutual information I between $\mathcal{G}_{\text{sub}}$ and the graph label $\mathbf{y}$, thereby encouraging $\mathcal{G}_{\text{sub}}$ to retain predictive capabilities regarding the graph label $\mathbf{y}$.

$$\max_{\mathcal{G}_{\text{sub}}} I\left(\mathcal{G}_{\text{sub}}, \mathbf{y}\right) =: \min_{\mathcal{G}_{\text{sub}}, g, f} \mathcal{L}_{\text{CE}}(g \circ f(\mathcal{G}_{\text{sub}}), \mathbf{y}) \tag{1}$$

Here, an explanatory module g is introduced to identify $\mathcal{G}_{\text{sub}}$. *Post-hoc* approaches first pretrain GNNs model f, and then optimize the explanatory module g while freezing model f. In contrast, *built-in* approaches jointly optimize both f and g.

4. Method: CIFlow-GNN

While existing methods tend to misclassify the neighboring irrelevant nodes as part of the vital subgraphs, we propose to explicitly separate them by graph clustering. To enhance clustering quality, we modulate the cluster information flow at both the cluster and prototype levels. As shown in Fig. 2, CIFlow-GNN consists of two modules: a graph clustering module and a cluster prototype module. The graph clustering module uses spectral clustering to partition the graph into dense subgraphs (clusters) while simultaneously modulating the cluster information flow at the cluster level. The cluster prototype module further refines clusters across graphs and learns cluster importance masks at the prototype level. In the following, we introduce the architecture of CIFlow-GNN in detail.

4.1. Graph Clustering Module

In this module, each graph is partitioned into K clusters through node assignments informed by structural connectivity and feature similarity. To encourage the formation of semantically meaningful clusters, we introduce two complementary objectives: a cluster connectivity loss and a cluster feature loss. These objectives jointly guide the clustering process by modulating the information flow at the cluster level within each graph. The nodes in the clusters are then aggregated into cluster embeddings.

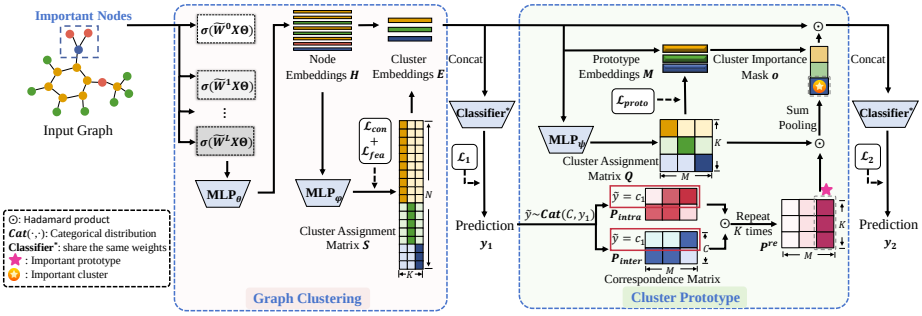


Figure 2: Overview of the proposed CIFlow-GNN framework.

4.1.1. Cluster Assignment Matrix

The goal of this step is to obtain cluster assignment matrix $\mathbf{S}$. First, we compute cluster-friendly node embeddings at layer l following [33]:

$$\mathbf{H}^{(l)} = \sigma\left(\widetilde{\mathbf{W}}^{l}\mathbf{X}\mathbf{\Theta}\right) \tag{2}$$

where $\mathbf{\Theta}$ is the weight matrix. The initial embeddings, $\mathbf{H}^{(0)} = \mathbf{X}\mathbf{\Theta}$, can be gained by applying a linear layer on the node feature matrix $\mathbf{X}$. This design forms cluster structures in the embedding space [33].

Then, to better leverage the intermediate node embeddings, we propagate $l = L$ times to generate a sequence of intermediate matrices $\left\{\mathbf{H}^{(0)}, \mathbf{H}^{(1)}, ..., \mathbf{H}^{(L)}\right\}$. The final node embeddings $\mathbf{H} \in \mathbb{R}^{n\times d}$ is achieved by combining all the intermediate node embedding matrices with Multi-Layer Perceptrons (MLP):

$$\mathbf{H} = \mathrm{MLP}_{\theta}\left(\mathrm{concat}\left(\mathbf{H}^{(l)} \mid l = 0, 1, \ldots, L\right)\right) \tag{3}$$

Finally, we partition each graph into K clusters and aggregate the embeddings of the nodes in each cluster into a single vector using the cluster assignment matrix. To learn the soft cluster assignment matrix $\mathbf{S} \in \mathbb{R}^{n\times K}$, we use an MLP followed by a row-normalized softmax function:

$$\mathbf{S} = \mathrm{Softmax}(\mathrm{MLP}_{\varphi}(\mathbf{H})) \tag{4}$$

where $\mathbf{S}_{i,k}$ represents the probability that node v_i belongs to the k-th cluster.

While Eq. (4) assigns nodes to clusters, it lacks explicit constraints on preserving structural and feature properties within clusters. To address this, we employ the

cluster connectivity loss to partition the graph into densely connected clusters and the cluster feature loss to modulate the cluster information flow at the cluster level, i.e., intra-cluster node similarities are strengthened and inter-cluster node similarities are weakened.

4.1.2. Cluster Connectivity Loss

To ensure nodes in each cluster are densely connected, we optimize the loss of spectral clustering, specifically the normalized cut (Ncut) [21]:

$$\min_{\mathbf{T}\in\mathbb{R}^{n\times K}} \mathrm{Tr}(\mathbf{T}^\intercal \widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{L}}\widetilde{\mathbf{D}}^{-1/2}\mathbf{T}) \quad \text{subject to} \quad \mathbf{T}^\intercal\mathbf{T} = \mathbf{I} \tag{5}$$

where the matrix $\mathbf{T}$ contains the first K eigenvectors of $\widetilde{\mathbf{L}} = \mathbf{I} - \widetilde{\mathbf{W}}$ as columns. We define column vector $\mathbf{z}^{(0)}$ to be a column of matrix $\mathbf{H}^{(0)}$. Then, $\mathbf{z}^{(l)}$ can be computed recursively as $\mathbf{z}^{(l)} = \widetilde{\mathbf{W}}\mathbf{z}^{(l-1)}$. This leads to the following expression:

$$\begin{aligned}\mathbf{z}^{(l)} &= \widetilde{\mathbf{W}}\mathbf{z}^{(l-1)} = \widetilde{\mathbf{W}}^2\mathbf{z}^{(l-2)} = \cdots = \widetilde{\mathbf{W}}^l\mathbf{z}^{(0)} \\ &= \widetilde{\mathbf{W}}^l(c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + \cdots + c_n\mathbf{u}_n) \\ &= c_1\lambda_1^l\mathbf{u}_1 + c_2\lambda_2^l\mathbf{u}_2 + \cdots + c_n\lambda_n^l\mathbf{u}_n\end{aligned} \tag{6}$$

where $[\lambda_1, \lambda_2, \ldots, \lambda_n]$ are the eigenvalues and $[\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n]$ are the eigenvectors of $\widetilde{\mathbf{W}}$, which means each column of $\mathbf{H}^{(l)}$ is a combination of the eigenvectors of $\widetilde{\mathbf{W}}$. Since $\widetilde{\mathbf{W}}$ and $\widetilde{\mathbf{L}}$ share the same eigenvectors, $\mathbf{H}$ can be considered as a combination of the eigenvectors of $\widetilde{\mathbf{L}}$. Consequently, as $\mathbf{S}$ is constructed by combining the columns of $\mathbf{H}$, we treat $\mathbf{S}$ as spectral embeddings, i.e., $\mathbf{T}$ in Eq. (5). This approach avoids the explicit, expensive, and non-differentiable eigendecomposition of the Laplacian matrix. The cluster connectivity loss is defined as follows:

$$\mathcal{L}_{\text{con}} = \min_{\mathbf{S}\in\mathbb{R}^{n\times K}} \underbrace{\mathrm{Tr}(\mathrm{Norm}(\mathbf{S})^\intercal\widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{L}}\widetilde{\mathbf{D}}^{-1/2}\mathrm{Norm}(\mathbf{S}))}_{\text{spectral loss}} + \underbrace{\|\mathrm{Norm}(\mathbf{S})^\intercal\mathrm{Norm}(\mathbf{S}) - \mathbf{I}_K\|_F}_{\text{orthogonal loss}} \tag{7}$$

where $\mathrm{Norm}(\mathbf{S})$ is obtained by applying the Euclidean norm to each column of matrix $\mathbf{S}$ and $\|\cdot\|_F$ denotes the Frobenius norm. Notably, our orthogonal loss allows for partitioning the clusters into varying sizes, making it robust for real-world scenarios.

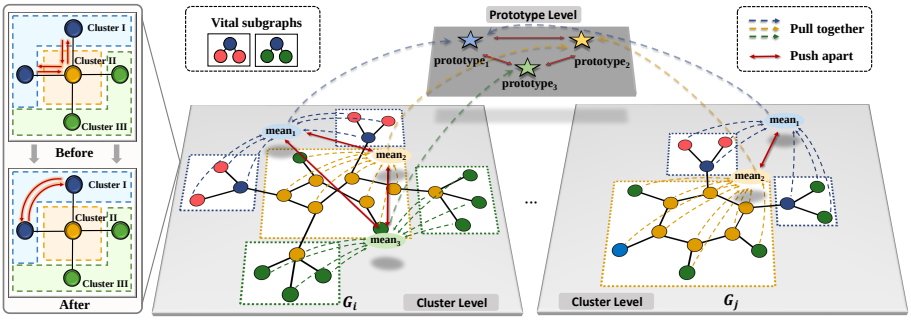


Figure 3: Illustration of modulating the cluster information flow in CIFlow-GNN at both the cluster level ($\mathcal{L}_{\text{fea}}$) and the prototype level ($\mathcal{L}_{\text{proto}}$). As a model-level method, the prototypes are shared across clusters within all graphs.

4.1.3. Cluster Feature Loss

To ensure that nodes within the same cluster exhibit similar embeddings and collectively represent specific group characteristics, whereas dissimilar clusters maintain distinct properties, we design a cluster feature loss. This loss pulls each node close to its cluster centroid while pushing apart different cluster centroids:

$$\mathcal{L}_{\text{fea}} = \sum_{k=1}^{K} \frac{1}{|\mathrm{C}_k|} \sum_{i \in \mathrm{C}_k} \left\| \mathbf{h}_i - \boldsymbol{\mu}_k \right\|^2 - \frac{1}{K-1} \sum_{k=1}^{K} \sum_{\hat{k}=1, \hat{k} \neq k}^{K} \left\| \boldsymbol{\mu}_k - \boldsymbol{\mu}_{\hat{k}} \right\|^2, \tag{8}$$
$$\mathrm{C}_k = \{\{i\} \mid \mathbf{S}_{i,k} = \max(\mathbf{S}_{i,:}),\ i = 1, 2, \ldots, N\}$$

where $\|\cdot\|$ denotes the Euclidean norm, C_k represents the set of nodes assigned to the k-th cluster, $\mathbf{h}_i$ denotes the embeddings of node i, and $\boldsymbol{\mu}_k/\boldsymbol{\mu}_{\hat{k}}$ is the centroid of the $k/\hat{k}$-th cluster. The term $\mathbf{S}_{i,k}$ denotes the probability that node v_i in graph $\mathcal{G}$ belongs to the k-th cluster. A node v_i is allocated to cluster C_k if it exhibits the highest probability of membership in that cluster. Since employing argmax for this allocation yields a non-differentiable problem, we sample from the matrix $\mathbf{S}$ using a categorical distribution, i.e., $\tilde{\mathbf{S}}_i \sim \mathbf{Cat}(K, \mathbf{S}_i)$. Then, we apply the reparameterization trick [34] as follows.

$$\tilde{\mathbf{S}}_{i,k} = \begin{cases} 1, & \sum_{j=0}^{k-1} \mathbf{S}_{i,j} < \xi \leq \sum_{j=0}^{k} \mathbf{S}_{i,j} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where $\xi \sim \mathcal{U}(0,1)$ (uniform distribution). Finally, $\boldsymbol{\mu}_k \in \mathbb{R}^d$ is computed as $\boldsymbol{\mu}_k = \frac{1}{|\mathrm{C}_k|} \mathrm{SUM}_{\text{col}}(\tilde{\mathbf{S}}_{:,k} \odot \mathbf{H})$ using the sum pooling operation applied to each column.

Our cluster feature loss is a proxy of the well-known InfoNCE loss [22] in contrastive learning [35]. Optimizing the cluster feature loss reduces the mutual information among clusters, realizing the modulation of cluster information flow. We analyze the theoretical connection between cluster feature loss and the InfoNCE loss. Specifically, we treat the nodes and their corresponding cluster centroid as positive pairs, while considering the nodes and the centroids from other clusters as negative pairs. Then we have the following theorem:

Theorem 1. *The cluster feature loss is a lower bound of the InfoNCE loss. Formally,* $\mathcal{L}_{fea} \leq \frac{K}{K-1}\mathcal{L}_{InfoNCE}$*, where* K *is the number of clusters.*

Proof. See the proof in the Appendix. □

In Fig. 3, we use two graphs from the Mutagenicity dataset to illustrate the whole process of modulating the cluster information flow. In graph $\mathcal{G}_i$, the two subgraphs in cluster I (blue dashed box) are vital subgraphs to the graph label prediction. However, they are separated by the nodes in cluster II (yellow dashed box), thus causing an information mixture in every message-passing step. To address this, we partition the graph into $K = 3$ clusters and modulate both the intra- and inter-cluster information flow at the cluster level, which facilitates direct information flow within the subgraphs in the same cluster while mitigating the information mixture across distinct clusters.

4.1.4. Cluster Embeddings

The embeddings of the clusters for each graph are computed as:

$$\mathbf{E} = \mathbf{S}^{\intercal}\mathbf{H} \tag{10}$$

where the k-th row of $\mathbf{E} \in \mathbb{R}^{K\times d}$ is the embedding of the k-th cluster. The graph embedding $\mathbf{E}_{\mathcal{G}} \in \mathbb{R}^{Kd}$ is formed by concatenating each row of the cluster embeddings $\mathbf{E}$. We feed $\mathbf{E}_{\mathcal{G}}$ into a classifier to compute class prediction $\mathbf{y}_1$.

4.2. Cluster Prototype Module

Our CIFlow-GNN aims to identify label-relevant subgraphs (clusters) that serve as explanations for graph classification tasks. Thus, the importance of each cluster should

be evaluated. A straightforward way is to apply an MLP to the cluster embeddings $\mathbf{E}$ in each graph and learn their importance values. However, such an instance-level method lacks a global understanding of the whole dataset. In this module, we use prototype learning to learn a model-level cluster importance mask for each cluster. The prototypes are designed to capture common representative patterns of similar clusters across graphs. Nevertheless, improper cluster-prototype correspondence could introduce irrelevant cluster information through backpropagation and thus lead to misclassification of neighboring irrelevant nodes. To mitigate this, we propose a cluster prototype loss to modulate the information flow at the prototype level.

4.2.1. Cluster Importance Mask

The cluster importance mask $\mathbf{o} \in \mathbb{R}^K$ represents the significance values of clusters within each graph. To compute this mask, we develop a three-step framework that integrate information from all graphs. First, we classify all the clusters across different classes in the dataset into different prototypes and learn the prototype assignment matrix. Then, we introduce two correspondence matrices to demonstrate the relative importance of prototypes both within a class and across classes. Finally, we compute the mask $\mathbf{o}$ by performing a Hadamard product over the above matrices. We allocate a pre-determined number M of prototypes for all clusters across the dataset.

The first step in the module is to learn the prototype assignment matrix $\mathbf{Q} \in \mathbb{R}^{K\times M}$ for each graph, where $\mathbf{Q}_{k,m}$ represents the probability that the k-th cluster belongs to the m-th prototype. We use an MLP combined with a row-normalized softmax function to compute matrix $\mathbf{Q}$:

$$\mathbf{Q} = \text{Softmax}(\text{MLP}_{\psi}(\mathbf{E})) \tag{11}$$

The next step involves learning the correspondence matrix $\mathbf{P} \in \mathbb{R}^{C\times M}$ between the classes and the prototypes, where C is the number of graph classes. In particular, we introduce two learnable matrices: $\mathbf{P}_{\text{intra}} \in \mathbb{R}^{C\times M}$ and $\mathbf{P}_{\text{inter}} \in \mathbb{R}^{C\times M}$. The y-th row of $\mathbf{P}_{\text{intra}}$ is softmax-normalized, representing the relative importance of prototypes within the y-th class. Similarly, the m-th column of $\mathbf{P}_{\text{inter}}$ is softmax-normalized, indicating the relative importance of the m-th prototype across different classes. Note that the m-th prototype is crucial to the y-th class when both $[\mathbf{P}_{\text{intra}}]_{c,m}$ and $[\mathbf{P}_{\text{inter}}]_{c,m}$ have large val-

ues. Therefore, we define the correspondence matrix $\mathbf{P}$ that displays the corresponding significance of prototypes for each class using the Hadamard product:

$$\mathbf{P} = \mathbf{P}_{\text{intra}} \odot \mathbf{P}_{\text{inter}} \tag{12}$$

A direct way to select the row in $\mathbf{P}$ that indicates the significance of prototypes is to apply the argmax operation to the class prediction $\mathbf{y}_1$. However, the argmax operation is inherently non-differentiable, which prevents the cluster prototype module from being jointly optimized with the parameters in the graph clustering module. To address this, we sample from the class prediction $\mathbf{y}_1$ by the categorical distribution, i.e., $\tilde{\mathbf{y}} \sim \mathbf{Cat}(C, \mathbf{y}_1)$. Then, we apply the reparameterization trick [34] to enable differentiability. Specifically, we reparameterize the pseudo-label $\tilde{\mathbf{y}}$ as:

$$\tilde{\mathbf{y}}[c] = \begin{cases} 1, & \sum_{j=0}^{c-1} \mathbf{y}_1[j] < \xi \leq \sum_{j=0}^{c} \mathbf{y}_1[j] \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where $\xi \sim \mathcal{U}(0, 1)$ (uniform distribution), $[c]$ indexes the c-th entry of a vector. The c-th row in matrix $\mathbf{P}$ corresponding to $\tilde{\mathbf{y}}[c] = 1$ is selected as the target row. Subsequently, we replicate the selected row K times to form the matrix $\mathbf{P}^{\text{re}} \in \mathbb{R}^{K \times M}$, with the size K chosen to be compatible with the matrix $\mathbf{Q}$ in Eq. (14).

The final step involves obtaining a customized cluster importance mask $\mathbf{o} = [o_1, ..., o_K]^T \in \mathbb{R}^K$ for the K clusters in each graph. We consider the importance of the k-th cluster indirectly by the integration of the probability it corresponds to the prototypes and the probability of these prototypes corresponds to classes. The cluster importance mask $\mathbf{o}$ is computed via the Hadamard product of matrices $\mathbf{Q}$ and $\mathbf{P}^{\text{re}}$, followed by sum pooling on each row:

$$\mathbf{o} = \text{SUM}_{\text{row}}(\mathbf{Q} \odot \mathbf{P}^{\text{re}}) \tag{14}$$

where a larger value of o_k indicates greater importance of the k-th cluster.

As shown in the framework in Fig. 2, the input graph is partitioned into $K = 3$ clusters using the graph clustering module. The resulting cluster embeddings are then assigned to $M = 3$ prototypes, as defined in Eq. (11), to generate the prototype assignment matrix $\mathbf{Q}$. In this example, the first, second, and third clusters are mapped

to the first, second, and third prototypes, respectively. Simultaneously, by employing the reparameterization trick on prediction $\mathbf{y_1}$, we derive the pseudo-label $\tilde{\mathbf{y}}[c_1] = 1$ for the input graph, as specified in Eq. (13). The c_1-th row of the correspondence matrix $\mathbf{P}$ is selected and replicated $K = 3$ times to form matrix $\mathbf{P}^{\text{re}}$. From $\mathbf{P}^{\text{re}}$, it is evident that the third prototype is the most significant for the given class. Next, Eq. (14) is applied to compute the importance values for each cluster. In this example, the third cluster, which aligns most closely to the third prototype in matrix $\mathbf{Q}$, achieves the highest cluster significance value o_3. Consequently, the third cluster is identified as the interpretable vital subgraph in the input graph.

4.2.2. Cluster Prototype Loss

As a model-level approach, our method establishes connections between similar clusters by associating them with shared prototypes, thereby facilitating their alignment across different graphs. Therefore, it reduces the risk of bias in identifying important clusters due to incorrect predictions in individual graphs and further promotes clustering refinement. Nevertheless, this prototype learning method concurrently introduces information from other less relevant prototypes when computing cluster importance masks.

The significance of a cluster to a specific class is indirectly represented through its association with the prototypes, i.e., matrix $\mathbf{Q}$, as derived from Eq. (11), and the importance of these prototypes to the class, i.e., matrix $\mathbf{P}^{\text{re}}$, as specified in Eq. (14). A prototype is designed to capture the common characteristics of a group of clusters within the dataset and each cluster should generally be close to one prototype. However, the prototype assignment matrix $\mathbf{Q}$ is a soft matrix containing continuous values between $[0, 1]$. During the calculation of the cluster importance mask $\mathbf{o}$, interference from other prototypes may occur if the row values of $\mathbf{Q}$ do not closely resemble one-hot encodings. To ensure accurate correspondence assignment between clusters and prototypes, we design a cluster prototype loss to clear the membership of each cluster by modulating inter-cluster information flow at the prototype level.

First, we define the prototype embeddings $\mathbf{M} \in \mathbb{R}^{M \times d}$, which are shared among clusters across all graphs in the dataset. As shown in Fig. 4, $\mathcal{G}_i$ and $\mathcal{G}_j$ are partitioned

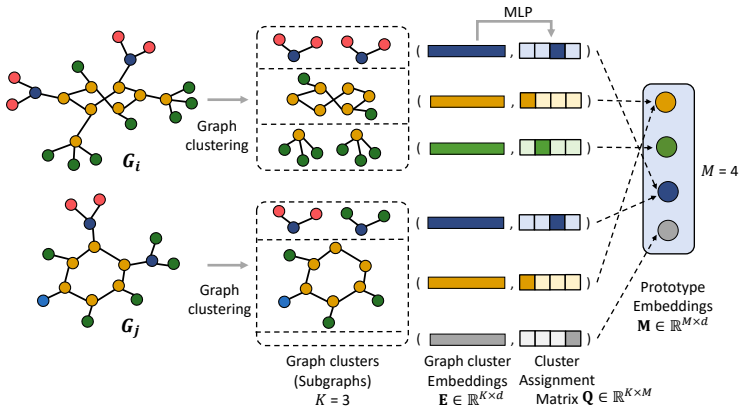


Figure 4: Illustration of the prototype embeddings computation process.

into $K = 3$ clusters, generating corresponding cluster embeddings $\mathbf{E}$ and prototype assignment matrix $\mathbf{Q}$. Then, similar clusters from $\mathcal{G}_i$ and $\mathcal{G}_j$ are mapped to the same prototype in the latent space based on the matrix $\mathbf{Q}$, yielding the prototype embeddings. Specifically, for a batch of size B, the set of prototype assignment matrices is $\left\{\mathbf{Q}^{(1)}, \ldots, \mathbf{Q}^{(B)}\right\}$, where $\mathbf{Q}^{(b)} \in \mathbb{R}^{K \times M}$ represents the prototype assignment matrix of the b-th graph. By concatenating these matrices along the first dimension, we obtain $\mathbf{Q}^{(\mathrm{All})} \in \mathbb{R}^{BK \times M}$. Similarly, concatenating all cluster embeddings $\left\{\mathbf{E}^{(1)}, \ldots, \mathbf{E}^{(B)}\right\}$ yields $\mathbf{E}^{(\mathrm{All})} \in \mathbb{R}^{BK \times d}$. The prototype embeddings are then computed as follows:

$$\mathbf{M} = \frac{\mathbf{Q}^{(\mathrm{All})^T} \mathbf{E}^{(\mathrm{All})}}{\mathrm{SUM}_{\mathrm{col}}(\mathbf{Q}^{(\mathrm{All})})} \tag{15}$$

where $\mathrm{SUM}_{\mathrm{col}}(\cdot)$ denotes the sum pooling operation applied to each column. The normalization term $\mathrm{SUM}_{\mathrm{col}}(\mathbf{Q}^{(\mathrm{All})})$ ensures that the prototype embeddings are not influenced by the varying number of clusters associated with each prototype.

Then, we increase the separation among different prototype embeddings as demonstrated at the prototype level in Fig. 3. As CIFlow-GNN is a *built-in* approach, regulating the prototype information through backpropagation not only drives prototype assignment matrix $\mathbf{Q}$ closer to one-hot encodings but also strengthens the separation of clusters. Additionally, we ensure that each prototype is associated with at least one cluster by enforcing that each column of matrix $\mathbf{Q}^{(\mathrm{All})}$ contains at least one value close

to 1. Our cluster prototype loss function is as follows:

$$\mathcal{L}_{\text{proto}} = -\frac{1}{2M(M-1)} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} \left\| \mathbf{M}_{i,:} - \mathbf{M}_{j,:} \right\|^2 + \frac{1}{M} \sum_{i=1}^{M} \min(\mathbb{1} - \mathbf{Q}_{:,i}^{(\text{All})}) \tag{16}$$

where $\mathbf{M}_{i,:}$ represents the embedding of the i-th prototype, $\mathbb{1}$ is an all-ones matrix, and $\mathbf{Q}^{(\text{All})}$ is the concatenation of prototype assignment matrices between the clusters and prototypes across a batch of graphs.

4.2.3. Masked Cluster Embeddings

We use the vector $\mathbf{o} \in \mathbb{R}^K$ to mask the graph cluster embeddings $\mathbf{E} = \{\mathbf{E}_{k,:}\}_{k=1}^K$, preserving only the predictive portion $\widetilde{\mathbf{E}} = \{\widetilde{\mathbf{E}}_{k,:}\}_{k=1}^K$ by the Hadamard product:

$$\widetilde{\mathbf{E}}_{k,:} = \mathbf{E}_{k,:} \odot o_k \quad k = 1, \ldots, K \tag{17}$$

Similar to $\mathbf{E}_{\mathcal{G}}$, the masked graph embeddings $\widetilde{\mathbf{E}}_{\mathcal{G}}$ is constructed by sequentially concatenating each row of the masked cluster embeddings $\widetilde{\mathbf{E}}$. The resulting $\widetilde{\mathbf{E}}_{\mathcal{G}}$ is then fed into the same classifier as used for $\mathbf{E}_{\mathcal{G}}$ to compute masked class prediction $\mathbf{y}_2$.

4.3. Model Training

Our goal is to learn a *built-in* interpretable GNN that can identify vital subgraphs while maintaining comparable accuracy. For better prediction performance, we minimize the cross-entropy loss on the training dataset:

$$\mathcal{L}_1 = \mathcal{L}_{\text{CE}}(\mathbf{y}_1, \mathbf{y}) \tag{18}$$

where $\mathbf{y} \in \{0, 1\}^C$ is the one-hot ground truth label and $\mathbf{y}_1 \in \mathbb{R}^C$ is the class prediction using the graph embeddings $\mathbf{E}_{\mathcal{G}}$. For better interpretation performance, we consider a cluster accuracy loss that encourages the clusters selected by the cluster prototype module to preserve more label-relevant information and achieve a comparable classification performance as using all the clusters:

$$\mathcal{L}_2 = \mathcal{L}_{\text{CE}}(\mathbf{y}_2, \mathbf{y}) \tag{19}$$

where $\mathbf{y}_2 \in \mathbb{R}^C$ is the prediction using the masked graph embeddings $\widetilde{\mathbf{E}}_{\mathcal{G}}$. Since the pseudo-label $\tilde{\mathbf{y}}$, which serves as one of the inputs for the cluster prototype module,

Table 2: Statistics of four datasets for graph interpretation.

	Ground truth subgraph	Graph number	Class number
Labeled-Motif	Tree motif (class 0); Grid motif, Hexagon motif (class 1)	2,000	2
Mutagenicity [36]	Nitro groups (-NO_2), Amino groups (-NH_2)	4,337	2
Solubility [37]	Hydroxyl groups (-OH)	1,144	2
Benzene [38]	Benzene structure (excluding hydrogen atoms)	12,000	2

is derived from prediction $\mathbf{y}_1$, improving class prediction performance consequently enhancing interpretation. To sum up, in conjunction with the previous constraints on the cluster connectivity loss $\mathcal{L}_{\text{con}}$ and cluster feature loss $\mathcal{L}_{\text{fea}}$, as well as the cluster prototype loss $\mathcal{L}_{\text{proto}}$ in Eq. (7), Eq. (8), and Eq. (16), the full objective is defined as follows:

$$\mathcal{L} = \mathcal{L}_1 + \lambda_2 \mathcal{L}_2 + \lambda_{\text{con}} \mathcal{L}_{\text{con}} + \lambda_{\text{fea}} \mathcal{L}_{\text{fea}} + \lambda_{\text{proto}} \mathcal{L}_{\text{proto}} \tag{20}$$

where λ_2, λ_{con}, λ_{fea}, λ_{proto} are hyperparameters controlling the importance of each part.

5. Experimental Evaluation

5.1. Baselines and Hyperparameters

We choose the following methods that integrate explanation functionality internally as baselines, including 6 *post-hoc* models (PGExplainer [23], SubgraphX [13], EiG-Search [7], MAGE [28], GAFExplainer [24], and ConfExplainer [9]), and 4 *built-in* models (ProtGNN [14], VGIB [15], PGIB [16], and GIP [31]). Since the subgraphs generated by GIP do not directly correspond to node sets in the original graph, we report only its classification performance and efficiency results. Moreover, we exclude the confidence loss term for ConfExplainer in the graph classification task, as ground-truth edge labels are not available.

For CIFlow-GNN, we adopt Adam with a learning rate of 0.01, training 350 epochs for graph classification and 100 epochs for graph interpretation, with batch sizes of 32 and 128, respectively. The hyperparameters λ_{con} and λ_{fea} are selected from $\{0.01, 0.05, 0.1, 0.3\}$, λ_2 is selected from $\{0.1, 0.3, 0.5, 1\}$, and λ_{proto} is selected from

Table 3: Interpretation performance (ROC AUC ↑) of different methods on 4 interpretation datasets. The best results for each dataset are highlighted in **bold**, while the runner-up results are indicated with underlines.

Dataset	Methods									
	PG Explainer	Sub graphX	ProtGNN	VGIB	PGIB	EiG-Search	MAGE	GAF Explainer	Conf Explainer	CIFlow-GNN
Labeled-Motif	77.1±5.1	93.9±2.2	58.0±4.7	68.4±12.6	41.5±1.4	94.0±3.3	93.2±2.7	50.3±4.5	91.2±1.6	**95.6±2.4**
Mutagenicity	62.5±1.3	61.6±2.0	67.8±6.6	64.6±1.8	53.2±2.5	76.4±0.9	75.8±2.0	63.6±4.9	77.3±1.9	**79.0±2.9**
Solubility	77.9±4.9	83.4±4.2	56.2±12.1	67.5±4.7	65.0±2.6	85.9±4.7	88.2±2.4	85.2±3.6	80.5±3.5	**93.2±3.9**
Benzene	48.2±1.4	77.1±3.1	66.4±9.1	62.2±2.4	61.3±1.8	77.6±3.0	77.4±4.1	22.2±8.3	75.3±4.4	**79.9±4.2**

$\{0.05, 0.1, 0.15\}$. The number of node embedding layers L is selected from $\{2, 3, 4, 5, 6\}$. The number of clusters per graph K is selected from $\{2, 3, 4, 5\}$. We select the number of prototypes M from C to KC. For all comparison methods, the hyperparameters are configured based on the settings described in the respective original papers. All experiments are conducted on a server equipped with a dual-core Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz, 256 GB memory, an Nvidia GeForce RTX 3090 GPU, and the Ubuntu 18.04.1 LTS operating system. We make our code and datasets publicly available at https://github.com/YJYTJ/CIFlow-GNN.

5.2. Graph Interpretation

5.2.1. Datasets and Experimental Settings

We evaluate our CIFlow-GNN using 4 datasets for GNN interpretation, including a synthetic dataset and 3 real-world biological datasets. We split the 3 real-world datasets randomly into training/validation/test sets in a ratio of 80%/10%/10%. In Table 2, we detail the statistics of the datasets for graph interpretation.

5.2.2. Interpretation Performance for Graphs

To evaluate the explanations for graphs, we report ROC AUC on 4 interpretation datasets with ground truth. We calculate the ROC AUC between the ground truth subgraphs related to graph properties and the vital subgraphs selected by our method. Each experiment is repeated 3 times.

Table 4: Interpretation performance (ROC AUC ↑) on two variants of CIFlow-GNN: a) only-$\mathbf{P}_{\text{inter}}$, which only utilizes $\mathbf{P}_{\text{inter}}$ as the correspondence matrix; and b) only-$\mathbf{P}_{\text{intra}}$, which only utilizes $\mathbf{P}_{\text{intra}}$ as the correspondence matrix in the cluster prototype module.

Dataset	only $\mathbf{P}_{\text{inter}}$	only $\mathbf{P}_{\text{intra}}$	CIFlow-GNN
Labeled-Motif	94.2±3.7	95.0±1.5	**95.6±2.4**
Mutagenicity	73.5±2.8	78.6±1.4	**79.0±2.9**
Solubility	89.8±4.1	80.3±3.6	**93.2±3.9**
Benzene	77.8±5.3	75.6±3.7	**79.9±4.2**

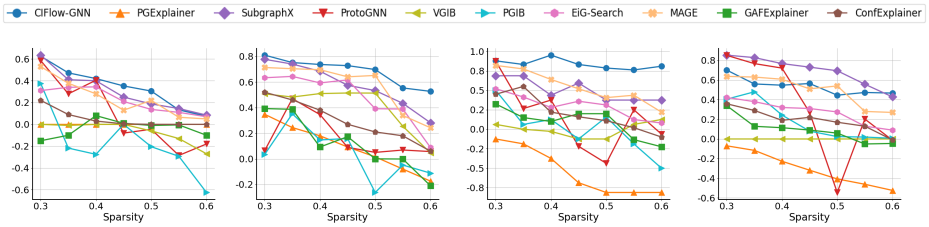
As shown in Table 3, our method significantly outperforms the baselines by 3.1% ↑ on average and up to 5.7% ↑ at most. CIFlow-GNN achieves excellent results on the Labeled-Motif dataset, indicating its effectiveness on datasets with clearly separable graph structures and distinct cluster features. The suboptimal performance of GAFExplainer on the Benzene dataset may be attributed to the scarcity of informative node attributes in the dataset, as the model primarily depends on node feature information for edge mask generation. As shown in Table 4, the variant only-$\mathbf{P}_{\text{inter}}$ achieves the second-best performance on the Solubility and Benzene datasets, while only-$\mathbf{P}_{\text{intra}}$ ranks second on the Labeled-Motif and Mutagenicity datasets. These results highlight the effectiveness of our approach, which incorporates the relative importance of prototypes both within a specific class and among different classes.

5.2.3. Interpretation Performance for Graph Classification Models

We adopt the fidelity metric $\mathcal{F}$, as proposed in [16], to assess the faithfulness of explanations to the GNN models. Formally, let y_i and $\hat{y}_i$ denote the ground truth and predicted values for the i-th input graph. We use a binary importance mask $m_i \in \mathbb{R}^n$, where a value of 1 indicates important nodes and its complement mask $1-m_i$ represents the remaining nodes. The fidelity score is computed as follows:

$$\mathcal{F} = \frac{1}{N}\sum_{i=1}^{N}(\mathbb{I}\left(\hat{y}_i^{m_i} = y_i\right) - \mathbb{I}(\hat{y}_i^{1-m_i} = y_i)) \tag{21}$$

where N is the number of test graphs. Moreover, we use *Sparsity* to quantify the proportion of nodes selected in the explanations. Effective explanations should yield a



(a) Labeled-Motif (b) Mutagenicity (c) Solubility (d) Benzene

Figure 5: Comparisons of fidelity scores ($\mathcal{F} \uparrow$) across different sparsity levels. PGExplainer, EiG-Search, and GAFExplainer are edge-based methods, whereas others are subgraph-based methods similar to ours.

high $\mathcal{F}$ score at a given level of sparsity. The sparsity is computed as follows:

$$Sparsity = \frac{1}{N}\sum_{i=1}^{N}(1 - \frac{|\mathcal{M}_i|}{|\mathcal{G}_i|}) \tag{22}$$

where $|\mathcal{M}_i|$ and $|\mathcal{G}_i|$ mean the number of important nodes in the mask m_i and the number of nodes in $\mathcal{G}_i$, respectively.

Fig. 5 illustrates the fidelity scores of CIFlow-GNN and baseline methods across varying sparsity levels. As shown in the figure, CIFlow-GNN achieves the best model interpretability on the Labeled-Motif, Solubility, and Mutagenicity datasets, and secures the second-best performance at higher sparsity levels on the Benzene dataset. Additionally, we observe that subgraph-based methods generally outperform their edge-based counterparts. An exception is EiG-Search, as it achieves relatively high fidelity despite being an edge-based approach, possibly because it is the only training-free method among the compared approaches.

5.2.4. Interpretation Performance of Information Evaluation

An effective GNN explainer should be capable of distinguishing vital subgraphs and complementary components. To this end, we introduce GInfo, an information evaluation metric that is a variant of the InfoNCE loss [22]. It is computed as follows:

$$\text{GInfo} = \frac{1}{NK}\sum_{i=1}^{N}\sum_{k=1}^{K}\frac{1}{|\mathrm{C}_k|}\sum_{v\in\mathrm{C}_k}\log\frac{\varepsilon\cdot\exp(\text{sim}(\mathbf{h}_v,\boldsymbol{\mu}_k))}{\exp(\text{sim}(\mathbf{h}_v,\boldsymbol{\mu}_k)) + \frac{1}{K-1}\sum_{\hat{k}=1,\hat{k}\neq k}^{K}\exp(\text{sim}(\mathbf{h}_v,\boldsymbol{\mu}_{\hat{k}}))} \tag{23}$$

Table 5: Information evaluation (GInfo ↓) of different methods on 4 interpretation datasets. The best results for each dataset are highlighted in **bold**, while the runner-up results are indicated with underlines.

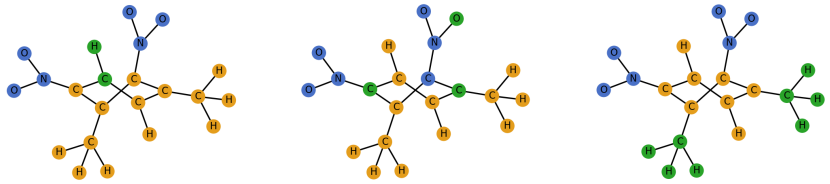
Dataset	Methods									
	PG Explainer	Sub graphX	ProtGNN	VGIB	PGIB	EiG-Search	MAGE	GAF Explainer	Conf Explainer	CIFlow -GNN
Labeled-Motif	0.62±0.01	0.32±0.02	0.58±0.01	0.32±0.01	0.44±0.01	0.48±0.01	0.45±0.01	0.52±0.02	0.37±0.01	**0.04±0.02**
Mutagenicity	0.61±0.01	0.54±0.06	0.59±0.01	0.20±0.03	0.60±0.01	0.56±0.02	0.54±0.02	0.47±0.02	0.17±0.02	**0.13±0.01**
Solubility	0.45±0.07	0.49±0.06	0.49±0.09	0.30±0.01	0.50±0.02	0.51±0.06	0.47±0.04	0.38±0.03	0.42±0.06	**0.28±0.02**
Benzene	0.42±0.07	0.59±0.01	0.61±0.01	0.33±0.04	0.52±0.01	0.55±0.06	0.54±0.02	0.44±0.03	0.40±0.04	**0.30±0.04**

where $\mathbf{h}_v$ denotes the embedding of node v, and $\boldsymbol{\mu}_k$ represents the centroid of the cluster C_k. The Euclidean norm is used as the similarity function $\mathrm{sim}(\cdot)$, with similarity values normalized using min-max normalization. The hyperparameter ε is set to $1+e$, keeping the metric within the range of $[0, 1]$. We repeat each experiment 3 times.

For a node v, the centroid of its cluster serves as the positive sample, while the centroids of other clusters act as negatives. A GInfo value closer to 0 indicates greater information disparity. As shown in Table 5, CIFlow-GNN outperforms all baselines across the datasets, with notable improvements on the Labeled-Motif and Mutagenicity datasets. Among the baselines, VGIB performs competitively, while PGIB shows suboptimal performance. This is likely because PGIB only maps the identified subgraphs to the prototypes while neglecting to project complement components onto prototypes, and fails to maintain separation between prototype embeddings. In contrast, our approach maps each cluster to a prototype and enforces prototype diversity. ConfExplainer achieves the second-best performance on the Mutagenicity dataset. It employs a confidence-aware GIB mechanism that effectively directs information flow toward vital subgraphs. As shown in the table, other methods demonstrate weaker capabilities in distinguishing between vital subgraphs and complementary components.

5.2.5. Effect of cluster connectivity loss and cluster feature loss

We perform graph clustering by jointly optimizing cluster connectivity loss and cluster feature loss, leveraging both the graph structure and cluster features. Fig. 6 shows the impact of the two losses on a molecular graph from the Mutagenicity dataset.



(a) only $\mathcal{L}_{\text{con}}$ (b) only $\mathcal{L}_{\text{fea}}$ (c) both losses

Figure 6: Illustration of the impact of cluster connectivity loss ($\mathcal{L}_{\text{con}}$) and cluster feature loss ($\mathcal{L}_{\text{fea}}$) on graph clustering performance for a molecule in the Mutagenicity dataset.

Table 6: Statistical significance of cluster connectivity loss ($\mathcal{L}_{\text{con}}$) and cluster feature loss ($\mathcal{L}_{\text{fea}}$) across 4 datasets, evaluated via t-tests. The table reports interpretation performance (ROC AUC ↑) and corresponding p-values for CIFlow-GNN with and without each loss. A p-value < 0.05 indicates that the comparison is statistically significant. The p-values that represent statistically significant results are highlighted in **bold**.

Dataset	$\lambda_{\text{con}} = 0$	$\lambda_{\text{con}} = 0.3$	p-value	$\lambda_{\text{fea}} = 0$	$\lambda_{\text{fea}} = 0.3$	p-value
Labeled-Motif	97.1±1.5	97.9±1.0	0.1464	92.7±1.0	99.3±0.23	$\mathbf{8.74 \times 10^{-8}}$
Mutagenicity	63.1±3.4	70.7±7.2	**0.0456**	58.8±9.5	73.6±8.4	**0.0013**
Solubility	65.5±4.8	78.4±9.0	**0.0195**	61.2±17.8	79.3±13.5	**0.0232**
Benzene	66.6±7.5	78.3±1.5	**0.0029**	75.7±3.1	79.5±2.3	**0.0190**

Specifically, (a) uses only the cluster connectivity loss ($\lambda_{\text{con}} = 0.3$), (b) uses only the cluster feature loss ($\lambda_{\text{fea}} = 0.3$), and (c) combines both losses ($\lambda_{\text{con}} = 0.3$ and $\lambda_{\text{fea}} = 0.3$). As shown in Fig. 6(a), the benzene ring and the functional groups -CH_3 are incorrectly grouped into the same cluster due to the Ncut concept inherent in spectral clustering. Fig. 6(b) shows that the carbon atoms on the benzene ring connected to the functional groups form distinct clusters. This occurs because their embeddings incorporate information from neighboring functional groups, leading to differences compared to other carbon atoms. The cluster feature loss promotes separation among clusters, thus further amplifying this discrepancy. Fig. 6(c) demonstrates a clearer and more accurate partitioning of functional groups that exhibits stronger alignment with the physio-chemical properties of the molecules in the real world.

We further conduct t-tests to evaluate the effectiveness of the cluster connectivity loss $\mathcal{L}_{\text{con}}$ and cluster feature loss $\mathcal{L}_{\text{fea}}$, with results summarized in Table 6. Specifically,

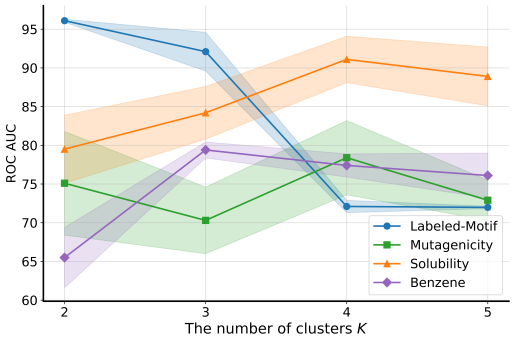
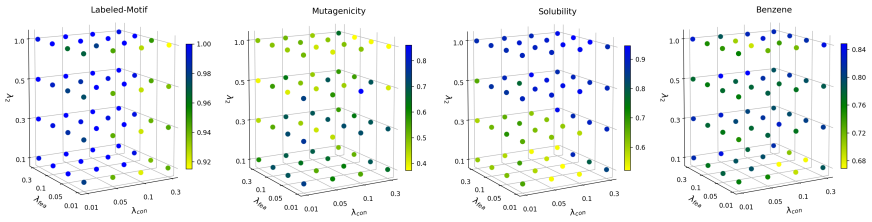


Figure 7: Interpretation performance (ROC AUC↑) on 4 datasets ($C = 2$) across different cluster numbers ($K = 2, 3, 4$) over 3 experiments.

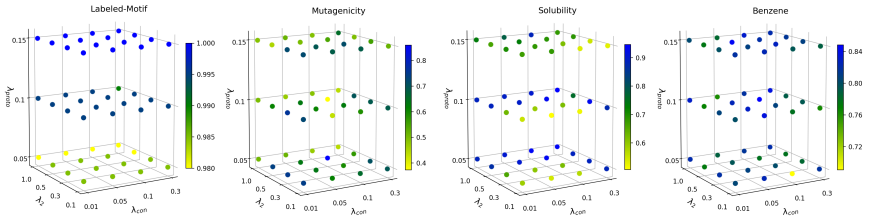
we run 10 trials to compare the interpretation performance (ROC AUC) of CIFlow-GNN with/without cluster connectivity loss ($\lambda_{\text{con}} = 0.3/\lambda_{\text{con}} = 0$), and calculate the corresponding p-values. A similar comparison is conducted to CIFlow-GNN with/without cluster feature loss ($\lambda_{\text{fea}} = 0.3/\lambda_{\text{fea}} = 0$). Both losses have a statistically significant effect on the model performance across all datasets except the Labeled-Motif dataset, where the p-value for cluster connectivity loss is 0.1464 (> 0.05). This is likely due to the varying node features across motifs in Labeled-Motif, which makes cluster feature loss the dominant factor in the clustering process. The notably lower p-value for cluster feature loss on this dataset supports the hypothesis. Overall, these findings affirm the statistical significance of both losses in improving model performance.

5.2.6. Hyperparameter Sensitivity of Interpretation Performance

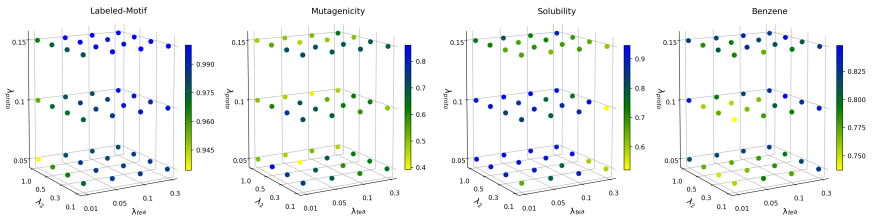
As defined in the objective function Eq. (20), CIFlow-GNN employs 4 key hyperparameters: λ_2, λ_{con}, λ_{fea}, and λ_{proto}. To evaluate the impact of their interactions on the model stability, we conduct sensitivity analyses across 4 datasets. For each analysis, we fix one hyperparameter and vary the values of the remaining three. The results are shown in Fig. 8, in which the dot color represents interpretation performance (ROC AUC ↑). As illustrated in the figure, the model exhibits superior performance on the Labeled-Motif dataset when the hyperparameters λ_2, λ_{fea}, and λ_{proto} are set to relatively large values. For Mutagenicity dataset, higher values of λ_{con}, λ_{fea}, and λ_{proto} yield the best results. On Solubility dataset, the optimal performance is achieved when λ_2, λ_{con},



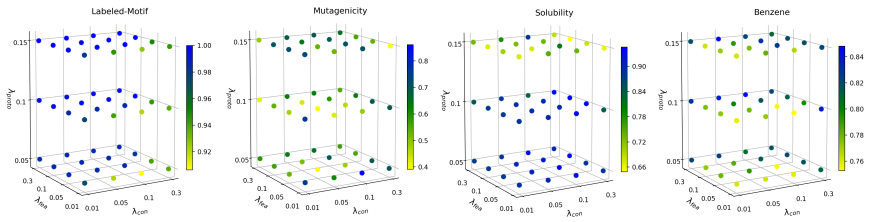
(a) Interpretation performance with respect to the parameters λ_2, λ_{con}, and λ_{fea} on 4 datasets



(b) Interpretation performance with respect to the parameter λ_2, λ_{con}, and λ_{proto} on 4 datasets



(c) Interpretation performance with respect to the parameter λ_2, λ_{fea}, and λ_{proto} on 4 datasets



(d) Interpretation performance with respect to the parameter λ_{con}, λ_{fea}, λ_{proto} on 4 datasets

Figure 8: Interpretation performance (ROC AUC ↑) with respect to the hyperparameters λ_2, λ_{con}, λ_{fea}, and λ_{proto} on 4 datasets. These 4-D figures use color to represent ROC AUC values, indicating the fourth dimension.

Table 7: Optimal CIFlow-GNN hyperparameters for interpretation (ROC AUC ↑).

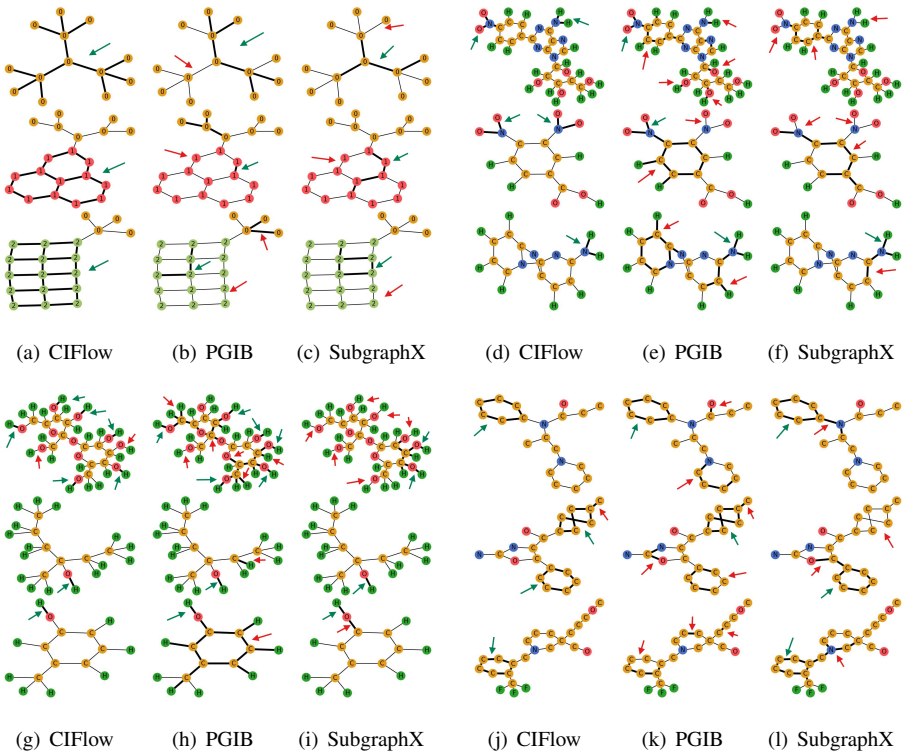
Dataset	Hyperparameters						
	λ_2	λ_{con}	λ_{fea}	λ_{proto}	K	M	L
Labeled-Motif	0.3	0.05	0.1	0.15	2	3	4
Mutagenicity	0.3	0.01	0.1	0.05	4	6	4
Solubility	1	0.05	0.01	0.05	4	5	5
Benzene	0.5	0.1	0.3	0.1	3	5	4

and λ_{fea} are large, while λ_{proto} remains small. Finally, the Benzene dataset benefits most from larger λ_{con} and λ_{fea}.

We further perform a hyperparameter analysis on the number of clusters K in Fig. 7. The results show that Labeled-Motif achieves the best ROC AUC at $K = 2$, Benzene at $K = 3$, and the remaining two datasets at $K = 4$. These findings demonstrate that the selection of K is determined not only by C, but also by the structural and feature characteristics of individual graphs. In addition, we provide the optimal hyperparameter configurations of CIFlow-GNN for achieving the best interpretation performance (ROC AUC ↑) on 4 interpretation datasets in Table 7.

5.2.7. Visualization of Label-relevant Subgraphs

Fig. 9 illustrates the label-relevant subgraphs identified by CIFlow-GNN and two comparison models: PGIB and SubgraphX on 4 datasets. PGIB is a subgraph-based *built-in* method and SubgraphX is a subgraph-based *post-hoc* method. As shown in the figure, both PGIB and SubgraphX misclassify nodes adjacent to the vital subgraphs as part of them. SubgraphX only identifies tightly connected subgraphs and fails to handle cases where multiple subgraphs are disconnected. In contrast, our CIFlow-GNN demonstrates superior performance in identifying label-relevant subgraphs. In Fig. 9(a)-(c), CIFlow-GNN accurately separates vital subgraphs from their complements in the Labeled-Motif dataset. Fig. 9(d)-(f) present that CIFlow-GNN is capable of detecting both -NO_2 and -NH_2 structures simultaneously in the Mutagenicity dataset. Fig. 9(g)-(i) show CIFlow-GNN can identify multiple non-connected -OH motifs within a single graph in the Solubility dataset. Fig. 9(j)-(l) demonstrate CIFlow-



(a) CIFlow (b) PGIB (c) SubgraphX (d) CIFlow (e) PGIB (f) SubgraphX

(g) CIFlow (h) PGIB (i) SubgraphX (j) CIFlow (k) PGIB (l) SubgraphX

Figure 9: Visualization of label-relevant subgraphs on Labeled-Motif (a-c), Mutagenicity (d-f), Solubility (g-i), and Benzene (j-l) datasets. Bold edges connect the nodes that the models consider important. Green arrows mark correctly predicted subgraphs, while red arrows highlight incorrect predictions, including both missed important subgraphs and falsely identified unimportant subgraphs.

GNN not only identifies benzene rings but also distinguishes motifs resembling benzene rings, such as a ring composed of one nitrogen atom and five carbon atoms.

5.3. Graph Classification

5.3.1. Datasets and Experimental Settings

We adopt 10 real-world benchmark graph datasets to evaluate our method for graph classification performance, including 8 biological datasets and 2 social media datasets. For a fair comparison, we follow the cross-validation procedure proposed in [39] which involves using a 10-fold cross-validation for model evaluation and an inner holdout method with 90%/10% training/validation split for model selection.

Table 8: Classification performance (Accuracy ↑) of different methods on 10 benchmark datasets. In general, CIFlow-GNN outperforms the baseline methods on 8 out of 10 datasets. The best results per benchmark are highlighted in **bold**, while the runner-up results are indicated with underlines. OOT indicates Out-Of-Time, meaning the execution time exceeded 24 hours. OOM denotes Out-Of-Memory, indicating that the method exceeded available memory resources.

Dataset	Methods										
	PG Explainer	Sub graphX	ProtGNN	VGIB	PGIB	GIP	EiG-Search	MAGE	GAF Explainer	Conf Explainer	CIFlow -GNN
MUTAG	78.6±11.4	61.7±18.2	77.6±4.7	72.1±21.6	76.6±6.8	83.5±5.1	82.3±9.0	82.7±7.2	72.9±10.9	82.5±6.7	**89.4±7.1**
BZR	82.9±2.7	69.5±21.3	82.5±2.5	79.8±3.9	83.7±5.8	80.9±7.0	85.1±3.3	84.9±5.1	77.0±4.5	83.1±5.2	**86.9±3.7**
BZR_MD	69.0±3.3	63.1±10.7	67.6±9.0	51.6±6.2	61.4±10.0	63.4±5.5	65.7±9.8	67.1±7.8	54.9±5.7	60.8±7.1	**70.8±7.3**
DHFR	69.0±6.4	61.9±4.9	70.8±2.3	51.7±10.5	77.4±4.1	74.5±10.9	76.2±6.1	77.5±5.3	63.1±5.4	71.8±8.2	**79.2±3.4**
COX2	72.2±2.7	77.1±9.3	79.2±3.1	80.0±4.2	75.8±2.3	79.4±2.8	80.2±5.9	80.6±6.5	78.4±2.3	75.5±3.8	**82.0±2.6**
PROTEINS	72.5±3.5	72.6±11.9	74.3±2.6	68.4±6.5	74.7±3.1	**79.5±3.5**	74.1±5.0	73.1±5.1	62.1±5.5	72.5±5.2	75.2±5.3
NCI1	66.8±9.1	52.8±3.5	75.5±1.1	52.2±4.6	73.8±2.1	74.1±2.9	73.7±1.7	73.9±2.0	54.4±3.1	68.0±6.1	**75.8±2.5**
DD	68.0±4.2	OOT	**81.8±1.8**	67.5±8.1	67.3±3.3	OOM	70.9±5.9	OOT	55.9±4.6	74.8±2.7	77.6±2.5
IMDB-B	62.2±4.0	67.3±6.2	58.3±7.3	50.8±3.7	57.0±2.9	66.5±4.5	70.3±4.7	69.8±5.0	49.4±6.5	65.6±5.4	**71.7±5.6**
IMDB-M	40.0±6.5	46.7±2.3	36.0±5.7	33.9±3.1	37.0±3.9	41.9±4.3	45.9±3.7	46.1±3.5	45.1±2.5	44.9±4.5	**47.9±3.4**

5.3.2. Prediction Performance for Graph Classification Models

As shown in Tabel 8, our method achieves the best results on 8 out of 10 datasets while securing the second-best performance on the PROTEINS and DD datasets. CIFlow-GNN achieves an improvement of over 2% across multiple datasets, including MUTAG, BZR, BZR_MD, DHFR, and IMDB-MULTI. Specifically, it outperforms the runner-up method by 7.1% on the MUTAG dataset. This demonstrates that our method achieves not only high interpretability but also outstanding predictive performance. In comparison, ProtGNN achieves the highest accuracy on the DD dataset and ranks as the runner-up on the NCI1 dataset. However, its performance degrades significantly on the IMDB-BINARY and IMDB-MULTI datasets. A potential reason for this performance gap is that ProtGNN is better suited for identifying simpler prototypes in biological datasets while struggling to adapt to the complexity of social media datasets. EiG-Search achieves the second-best performance on the BZR and IMDB-BINARY datasets, while MAGE ranks second on the DHFR and COX2 datasets. Additionally,

Table 9: Efficiency studies of different methods on MUTAG and DD datasets. The unit of running time is second (s). The best results are highlighted in **bold**, while the runner-up results are indicated with underlines. OOT indicates Out-Of-Time, meaning the execution time exceeded 6 hours. OOM denotes Out-Of-Memory, indicating that the method exceeded available memory resources.

Methods		PGExplainer	SubgraphX	ProtGNN	VGIB	PGIB	GIP	EiG-Search	MAGE	GAFExplainer	ConfExplainer	CIFlow-GNN
MUTAG	Pre-training	13.4	13.6	0	0	0	0	13.6	13.9	106.4	14.1	0
	Training	$\mathbf{1.6\times10^{-4}}$	40.9	3.7×10^{-2}	5.3×10^{-3}	4.3×10^{-3}	0.75	8.9	9.9	9.1×10^{-2}	0.82	$\underline{4.1\times10^{-3}}$
DD	Pre-training	218.7	106.2	0	0	0	0	102.1	105.9	113.7	103.5	0
	Training	$\mathbf{4.6\times10^{-4}}$	3766.4	160.4	$\underline{5.4\times10^{-3}}$	4.9	OOM	353.8	OOT	9.4×10^{-2}	1.1	5.7×10^{-3}

GIP achieves the highest accuracy on the PROTEINS dataset, demonstrating its supriority on datasets with prominent cluster structures.

5.4. Efficiency Studies

We evaluate the efficiency of our proposed method on two graph datasets: the small dataset MUTAG, with an average of 17.9 nodes per graph, and the large dataset DD, with an average of 284.3 nodes per graph. Using 100 graphs, we report the averaging time cost required to generate explanations for each graph in 1 epoch. Additionally, we report the pre-training time over 350 epochs for each *post-hoc* method following [13]. Each experiment is repeated 3 times, and the results are summarized in Table 9.

Among the six *post-hoc* methods, MAGE and SubgraphX exhibit significantly high computational costs. This is mainly attributed to the calculation of the Myerson–Taylor index and the use of Monte Carlo Tree Search (MCTS) for subgraph exploration, respectively. In addition, GAFExplainer requires a longer pretraining time compared with the other *post-hoc* methods, since it employs a GAT with 8 attention heads as its GNN module, whereas the others adopt the GIN. The *built-in* methods are all subgraph-based. Our method is significantly faster than ProtGNN and PGIB, as both rely on MCTS to update prototype values, whereas our method does not. GIP encounters an OOM issue on the DD dataset due to its use of dense adjacency matrices for computing the random walk kernels. Among all the methods, the time cost of PGExplainer is shorter than ours. However, PGExplainer is an edge-based method where the se-

lected edges are often loosely connected, and it also requires a pre-training step. In summary, our method demonstrates good computational efficiency compared to other subgraph-based approaches.

6. Conclusion

We propose a novel framework for enhancing the interpretability of GNNs, named Cluster Information Flow Graph Neural Networks (CIFlow-GNN), which modulates the cluster information flow at both the cluster and prototype levels. Experimental results show that CIFlow-GNN achieves high-quality graph interpretation and classification performance, along with competitive computational efficiency compared to other subgraph-based approaches. However, there are still limitations in our work. Firstly, our work primarily targets graphs with well-defined structural patterns, such as molecular graphs. Due to the lack of ground-truth labels for vital subgraphs in existing social network datasets, we have not been able to evaluate interpretability performance in those domains. Secondly, we select the cluster corresponding to the highest score in the cluster importance mask as the vital subgraph. For cases involving multiple important subgraphs distributed across different clusters, we can employ a threshold to select the Top-A clusters. For example, if the sum of the top-A cluster importance scores exceeds a predefined threshold (e.g., 0.5) times the total sum of all cluster importance scores, we treat these Top-A clusters as the vital subgraphs. However, determining the optimal threshold is non-trivial and left for future work.

7. Acknowledgments

We thank the anonymous reviewers for their valuable and constructive comments. This work was partially supported by the National Natural Science Foundation of China under Grants 62176184 and 62473288, the Fundamental Research Funds for the Central Universities, the Xiaomi Young Talents Program, and the Science and Technology Development Fund, Macao SAR No. 0006/2024/RIA1.

Appendix A. The proof of Theorem 1

Proof. Without loss of generality, we calculate $\mathcal{L}_{\text{fea}}$ for the first cluster.

$$
\begin{aligned}
\mathcal{L}_{\text{fea}}^{k=1} &= \frac{1}{|\mathrm{C}_1|}\sum_{i\in\mathrm{C}_1}\left\|\mathbf{h}_i-\boldsymbol{\mu}_1\right\|^2-\frac{1}{K-1}\sum_{\hat{k}=2}^{K}\left\|\boldsymbol{\mu}_1-\boldsymbol{\mu}_{\hat{k}}\right\|^2 \\
&= \frac{1}{|\mathrm{C}_1|}\sum_{i\in\mathrm{C}_1}\left\|\mathbf{h}_i-\boldsymbol{\mu}_1\right\|^2-\frac{1}{K-1}\sum_{\hat{k}=2}^{K}\left\|\frac{1}{|\mathrm{C}_1|}\sum_{i\in\mathrm{C}_1}\mathbf{h}_i-\boldsymbol{\mu}_{\hat{k}}\right\|^2 \\
&= \frac{1}{|\mathrm{C}_1|}(\sum_{i\in\mathrm{C}_1}\left\|\mathbf{h}_i-\boldsymbol{\mu}_1\right\|^2-\frac{1}{K-1}\sum_{\hat{k}=2}^{K}\left\|\sum_{i\in\mathrm{C}_1}(\mathbf{h}_i-\boldsymbol{\mu}_{\hat{k}})\right\|^2) \\
&= \frac{1}{|\mathrm{C}_1|}(\sum_{i\in\mathrm{C}_1}\left\|\mathbf{h}_i-\boldsymbol{\mu}_1\right\|^2-\frac{1}{K-1}\sum_{\hat{k}=2}^{K}[\sum_{i\in\mathrm{C}_1}\left\|\mathbf{h}_i-\boldsymbol{\mu}_{\hat{k}}\right\|^2+2\sum_{i,j\in\mathrm{C}_1,i\neq j}(\mathbf{h}_i-\boldsymbol{\mu}_{\hat{k}})^T(\mathbf{h}_j-\boldsymbol{\mu}_{\hat{k}})]) \\
&= \frac{1}{|\mathrm{C}_1|}(\sum_{i\in\mathrm{C}_1}\left\|\mathbf{h}_i-\boldsymbol{\mu}_1\right\|^2-\frac{1}{K-1}\sum_{i\in\mathrm{C}_1}\sum_{\hat{k}=2}^{K}\left\|\mathbf{h}_i-\boldsymbol{\mu}_{\hat{k}}\right\|^2)+A \qquad (*) \\
&\leq \frac{1}{|\mathrm{C}_1|}\sum_{i\in\mathrm{C}_1}(\left\|\mathbf{h}_i-\boldsymbol{\mu}_1\right\|^2-\frac{1}{K-1}\sum_{\hat{k}=2}^{K}\left\|\mathbf{h}_i-\boldsymbol{\mu}_{\hat{k}}\right\|^2) \\
&= \frac{K}{K-1}\cdot\frac{1}{|\mathrm{C}_1|}\sum_{i\in\mathrm{C}_1}(\left\|\mathbf{h}_i-\boldsymbol{\mu}_1\right\|^2-\frac{1}{K}\sum_{\hat{k}=1}^{K}\left\|\mathbf{h}_i-\boldsymbol{\mu}_{\hat{k}}\right\|^2) \\
&= -\frac{K}{K-1}\cdot\frac{1}{|\mathrm{C}_1|}\sum_{i\in\mathrm{C}_1}(\log\exp\left(-\left\|\mathbf{h}_i-\boldsymbol{\mu}_1\right\|^2\right) \\
&\quad -[\frac{1}{K}\sum_{\hat{k}=1}^{K}\log\exp\left(-\left\|\mathbf{h}_i-\boldsymbol{\mu}_{\hat{k}}\right\|^2\right)])\\
&\leq -\frac{K}{K-1}\cdot\frac{1}{|\mathrm{C}_1|}\sum_{i\in\mathrm{C}_1}(\log\exp\left(-\left\|\mathbf{h}_i-\boldsymbol{\mu}_1\right\|^2\right)-\log[\frac{1}{K}\sum_{\hat{k}=1}^{K}\exp\left(-\left\|\mathbf{h}_i-\boldsymbol{\mu}_{\hat{k}}\right\|^2\right)]) \qquad (**) \\
&= -\frac{K}{K-1}\cdot\frac{1}{|\mathrm{C}_1|}\sum_{i\in\mathrm{C}_1}\log\frac{\exp\left(-\left\|\mathbf{h}_i-\boldsymbol{\mu}_1\right\|^2\right)}{\frac{1}{K}\sum_{\hat{k}=1}^{K}\exp\left(-\left\|\mathbf{h}_i-\boldsymbol{\mu}_{\hat{k}}\right\|^2\right)} \\
&= \frac{K}{K-1}\mathcal{L}_{\text{InfoNCE}}^{k=1} \qquad (A.1)
\end{aligned}
$$

Since $\mathbf{h}_i$ and $\mathbf{h}_j$ progressively approach $\boldsymbol{\mu}_1$ throughout the optimization process, the term A in Eq. $(*)$ can be approximated as:

$$
\begin{aligned}
A &= -\frac{2}{|\mathrm{C}_1|(K-1)}\sum_{\hat{k}=2}^{K}\sum_{i,j\in\mathrm{C}_1,i\neq j}(\mathbf{h}_i-\boldsymbol{\mu}_{\hat{k}})^T(\mathbf{h}_j-\boldsymbol{\mu}_{\hat{k}}) \\
&\approx -\frac{2}{|\mathrm{C}_1|(K-1)}\sum_{\hat{k}=2}^{K}\sum_{i,j\in\mathrm{C}_1,i\neq j}\|\boldsymbol{\mu}_1-\boldsymbol{\mu}_{\hat{k}}\|^2\leq 0 \qquad (A.2)
\end{aligned}
$$

Additionally, we use Jensen's inequality in Eq. (∗∗). By aggregating the loss across all clusters, we derive the following inequality:

$$\mathcal{L}_{\text{fea}} \leq \frac{K}{K-1}\mathcal{L}_{\text{InfoNCE}} \tag{A.3}$$

Thus, the cluster feature loss serves as a lower bound of the InfoNCE loss, thereby completing the proof.

□

References

[1] X. Pan, X. Han, C. Wang, Z. Li, S. Song, G. Huang, C. Wu, A unified framework for convolution-based graph neural networks, Pattern Recognition (2024) 110597.

[2] M. Jiang, G. Liu, Y. Su, X. Wu, Self-attention empowered graph convolutional network for structure learning and node embedding, Pattern Recognition 153 (2024) 110537.

[3] Y. Wang, V. D. Calhoun, G. D. Pearlson, P. Kochunov, T. G. van Erp, Y. Du, A graph transformer-based foundation model for brain functional connectivity network, Pattern Recognition (2025) 111988.

[4] X. Fan, M. Gong, Y. Xie, F. Jiang, H. Li, Structured self-attention architecture for graph-level representation learning, Pattern Recognition 100 (2020) 107084.

[5] X.-b. Ye, Q. Guan, W. Luo, L. Fang, Z.-R. Lai, J. Wang, Molecular substructure graph attention network for molecular property identification in drug discovery, Pattern Recognition 128 (2022) 108659.

[6] Y. Leng, L. Yu, Incorporating global and local social networks for group recommendations, Pattern Recognition 127 (2022) 108601.

[7] S. Lu, B. Liu, K. G. Mills, J. He, D. Niu, Eig-search: Generating edge-induced subgraphs for GNN explanation in linear time, in: ICML, 2024.

[8] Z. Wang, J. Guo, J. Liang, J. Liang, S. Cheng, J. Zhang, Graph segmentation and contrastive enhanced explainer for graph neural networks, in: AAAI, Vol. 39, 2025, pp. 21393–21401.

[9] J. Zhang, X. Liu, D. Luo, H. Wei, Is your explanation reliable: Confidence-aware explanation on graph neural networks, KDD (2025).

[10] A. Duval, F. D. Malliaros, Graphsvx: Shapley value explanations for graph neural networks, in: ECML PKDD, Springer, 2021, pp. 302–318.

[11] M. Vu, M. T. Thai, Pgm-explainer: Probabilistic graphical model explanations for graph neural networks, NeurIPS 33 (2020) 12225–12235.

[12] W. He, M. N. Vu, Z. Jiang, M. T. Thai, An explainer for temporal graph neural networks, in: GLOBECOM, IEEE, 2022, pp. 6384–6389.

[13] H. Yuan, H. Yu, J. Wang, K. Li, S. Ji, On explainability of graph neural networks via subgraph explorations, in: ICML, 2021, pp. 12241–12252.

[14] Z. Zhang, Q. Liu, H. Wang, C. Lu, C. Lee, Protgnn: Towards self-explaining graph neural networks, in: AAAI, Vol. 36, 2022, pp. 9127–9135.

[15] J. Yu, J. Cao, R. He, Improving subgraph recognition with variational graph information bottleneck, in: CVPR, 2022, pp. 19396–19405.

[16] S. Seo, S. Kim, C. Park, Interpretable prototype-based graph information bottleneck, NeurIPS 36 (2024).

[17] S. Miao, M. Liu, P. Li, Interpretable and generalizable graph learning via stochastic attention mechanism, in: ICML, 2022, pp. 15524–15543.

[18] 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 (7676) (2017) 354–359.

[19] J. Yu, T. Xu, Y. Rong, Y. Bian, J. Huang, R. He, Graph information bottleneck for subgraph recognition, ICLR (2021).

[20] N. Tishby, F. C. Pereira, W. Bialek, The information bottleneck method, Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing (2000) 368–377.

[21] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on pattern analysis and machine intelligence 22 (8) (2000) 888–905.

[22] A. v. d. Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, arXiv preprint arXiv:1807.03748 (2018).

[23] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, X. Zhang, Parameterized explainer for graph neural network, NeurIPS 33 (2020) 19620–19631.

[24] W. Hu, J. Wu, Q. Qian, Gafexplainer: Global view explanation of graph neural networks through attribute augmentation and fusion embedding, IEEE Transactions on Knowledge and Data Engineering (2025).

[25] Y.-M. Shin, S.-W. Kim, W.-Y. Shin, Page: prototype-based model-level explanations for graph neural networks, IEEE transactions on pattern analysis and machine intelligence (2024).

[26] A. Sinha, S. Vennam, C. Sharma, P. Kumaraguru, Higher order structures for graph explanations, in: AAAI, Vol. 39, 2025, pp. 20514–20521.

[27] B. Armgaan, M. Dalmia, S. Medya, S. Ranu, Graphtrail: Translating gnn predictions into human-interpretable logical rules, NeurIPS 37 (2024) 123443–123470.

[28] N. Bui, H. T. Nguyen, V. A. Nguyen, R. Ying, Explaining graph neural networks via structure-aware interaction index, in: ICML, 2024.

[29] S. Gui, H. Yuan, J. Wang, Q. Lao, K. Li, S. Ji, Flowx: Towards explainable graph neural networks via message flows, IEEE Transactions on Pattern Analysis and Machine Intelligence (2023).

[30] J. Deng, Y. Shen, Self-interpretable graph learning with sufficient and necessary explanations, in: AAAI, Vol. 38, 2024, pp. 11749–11756.

[31] Y. Wang, S. Liu, T. Zheng, K. Chen, M. Song, Unveiling global interactive patterns across graphs: Towards interpretable graph neural networks, in: KDD, 2024, pp. 3277–3288.

[32] J. Wang, M. Liu, F. Nie, X. Li, Normalized cut co-clustering with out-of-sample extension, Pattern Recognition (2025) 111881.

[33] W. Ye, Z. Huang, Y. Hong, A. Singh, Graph neural diffusion networks for semi-supervised learning, arXiv preprint arXiv:2201.09698 (2022).

[34] W. Guo, J. Yang, H. Yin, Q. Chen, W. Ye, Picnn: A pathway towards interpretable convolutional neural networks, in: AAAI, Vol. 38, 2024, pp. 2003–2012.

[35] H. Zhong, J. Wu, C. Chen, J. Huang, M. Deng, L. Nie, Z. Lin, X.-S. Hua, Graph contrastive clustering, in: ICCV, 2021, pp. 9224–9233.

[36] 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 (2) (1991) 786–797.

[37] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, R. P. Adams, Convolutional networks on graphs for learning molecular fingerprints, NeurIPS 28 (2015).

[38] B. Sanchez-Lengeling, J. Wei, B. Lee, E. Reif, P. Wang, W. Qian, K. McCloskey, L. Colwell, A. Wiltschko, Evaluating attribution for graph neural networks, NeurIPS 33 (2020) 5898–5910.

[39] F. Errica, M. Podda, D. Bacciu, A. Micheli, A fair comparison of graph neural networks for graph classification, in: ICLR, 2020.