A THE USE OF LARGE LANGUAGE MODELS (LLMS)

In preparing this paper, Large Language Models (LLMs) were used solely as general-purpose writing assistants. Specifically, we employed LLMs to polish the text by carefully correcting grammar, spelling, and occasionally translating our original sentences into fluent academic English when necessary. Importantly, the LLMs were not used to generate new research ideas, formulate novel claims, or alter the meaning of our original sentences. All outputs generated by LLMs were carefully reviewed and verified by the authors for consistency with our intended meaning and to avoid any issues of plagiarism or scientific misconduct.

B RELATED WORK

GNN Explainability and Out-Of-Distribution. Current GNN explanatory methods (Shin et al., 2024; Wang & Shen, 2022; Yu & Gao, 2024; Zhang et al., 2022; Wu et al., 2023; Li et al., 2023) aim to enhance model decision transparency through model-level or instance-level methods (Yuan et al., 2022). Model-level methods (Yuan et al., 2020) enhance GNN explainability by generating input-independent explanations, while instance-level methods (Yuan et al., 2021) identify input features most relevant to the model prediction. In this work, we focus on post-hoc instance-level methods. However, such methods face growing OOD challenges (Zheng et al., 2023; Fang et al., 2023). While some works (Fang et al., 2024; Faber et al., 2020) have already made attempts, but they require that the model be retrained. Separately, traditional mixup-based data augmentation methods (Zhang et al., 2023; Chen et al., 2024; Zhang et al., 2024) also alleviate the OOD problem, while still struggle to ensure the quality of mixup graphs for approximating the original distribution.

Graph Contrastive Learning. Graph Contrastive Learning (GCL) has become a dominant paradigm for graph self-supervised learning, aiming to learn discriminative label-independent representations by maximizing mutual information across different views. For example, GraphCL (You et al., 2020) leverages various data augmentation strategies to enhance the robustness and transferability of unsupervised representations. AutoGCL (Yin et al., 2022) employs learnable view generators, guided by auto-augmentation, to produce semantically consistent yet diverse graph views. SimGRACE (Xia et al., 2022) generates contrastive views via perturbed encoders for representation alignment. Among current post-hoc explanatory methods, RegExplainer (Zhang et al., 2024) is a pioneering work that introduces contrastive learning in the graph regression task. However, a unified task-agnostic explanation framework remains unexplored.

Graph Pooling. Graph pooling methods typically capture global information through hierarchical compression to obtain generalizable and expressive representations. Some methods learn a score function from node features and select the top r nodes to form a new subgraph (Gao & Ji, 2019). For example, SAGPooling (Lee et al., 2019) refines node scoring by integrating structural context via GNNs. GSAPool (Zhang et al., 2021) enhances pooling process by dynamically fusing node features and topology information. DMIPool (Zhao et al., 2023) further captures multi-level dependencies from dual-view representations to improve robustness. MLAP (Itoh et al., 2022) preserves cross-layer structural information through layer-wise pooling, enhancing node distinguishability. Our method introduces pooling strategies to apply hard perturbations to graph structure, extracting more distinguishable explanatory and label-irrelevant subgraphs, ultimately enabling the mixup graphs to more faithfully approximate the original distribution.

C NOTATIONS

In Table 4, we summarized the important notations we used and their descriptions in this paper.

Table 4: Important symbols and notations.

Symbols	Descriptions
G	Graph instance
$\boldsymbol{X}$	Node feature matrix
$\boldsymbol{A}$	Adjacency matrix
Y	Label for G
$\mathcal{G}$	Graph dataset
$\mathcal{V}, \mathcal{E}$	Node set and edge set
$\mathcal{Y}$	Label set
v_i	The i-th node
x_i	The feature of node v_i
n	Number of nodes in G
edge(i, j)	Edge from node i to node j
i	The i-th node index
j	The j-th node index
d	Feature dimension
G^*	Optimal explanatory subgraph
Y^*	Label for G^*
$\boldsymbol{H}^*$, $\boldsymbol{A}^*$	Node embeddings and adjacency matrix of G^*
$I(\cdot)$	Mutual information
α	Hyper parameter for mutual information term in GIB
$\mathcal{N}$	Number of graphs in $\mathcal{G}$
$f(\cdot)$	To-be-explained GNN model
$\boldsymbol{H}$	Node embeddings matrix
$f_\psi(\cdot)$	Explainer network
$f_{\text{enc}}(\cdot)$	Encoder component of f that generates graph representations
$\boldsymbol{h}_G$	Graph representation of G
$\text{DV}(\cdot)$	Donsker–Varadhan variational representation
$T(\cdot)$	Critic function
G_Y	The graph paired with label Y
$\text{sim}(\cdot, \cdot)$	Dot-product similarity between ℓ_2-normalized vectors
r_l	Node preservation ratio at layer l
I^L	Final preserved node indices after L pooling layers
score	Node scores
p_l	Learnable projection vector at layer l
N	Number of sampled neighbors
G'	Sampled neighbor
$\boldsymbol{A}_{\text{our}}^{(\text{mix})}$	Adjacency matrix obtained via structural mixup strategy
$\boldsymbol{H}_{\text{our}}^{(\text{mix})}$	Node embeddings matrix obtained via structural mixup strategy
$G_{\text{our}}^{(\text{mix})}$	Graph obtained via structural mixup strategy
S'	Binary node preserved matrix from G'
$\boldsymbol{W}$	Edge weights matrix
$\boldsymbol{M}_{\text{our}}^{(\text{mix})}$	Edge weights mask for mixup graph
$\mathcal{L}_{\text{InfoNCE}}$	InfoNCE loss
$\mathcal{L}_{\text{BCE}}$	Binary cross-entropy loss
β	Hyper parameter balancing $\mathcal{L}_{\text{InfoNCE}}$ and $\mathcal{L}_{\text{BCE}}$

D ALGORITHMS

Algorithm 1 Training Explainer

Input: A graph dataset $\mathcal{G}$, pretrained GNN model f, explainer network $f_\psi(\cdot)$.
Output: Trained explainer network $f_\psi(\cdot)$.

```
Initialize explainer network f_ψ(·).
for e ∈ epochs do
    for G ∈ 𝒢 do
        Randomly sample N graphs {G_i}_{i=0}^{N-1} from graph dataset 𝒢
        Compare similarity between {G_i}_{i=0}^{N-1} and G to obtain G^+ and {G_i^-}_{i=1}^{N-1}
        G_our^(mix)+ ← Structural Mixup(G, G^+)
        {G_our,i^(mix)-}_{i=1}^{N-1} ← Structural Mixup(G, {G_i^-}_{i=1}^{N-1})
        Generate M_(our)^(mix)+ and {M_our,i^(mix)-}_{i=1}^{N-1} with G_our^(mix)+ and {G_our,i^(mix)-}_{i=1}^{N-1}
        Compute L_InfoNCE(G, G^+, {G_i^-}_{i=1}^{N-1}) with Equation 10
        Compute L_BCE with Equation 11
        Compute overall loss L with Equation 12
    end for
    Update f_ψ(·) with back propagation
end for
Return Trained explainer network f_ψ(·)
```

Algorithm 2 Graph Structural Mixup Algorithm

Input: To-be-explained graph G, G' sampled from graph dataset $\mathcal{G}$.
Output: Graph $G_{\text{our}}^{(\text{mix})}$.

```
Obtain I^L via L-layer graph pooling following Equation 8
Obtain (I')^L via L-layer graph pooling following Equation 8
Generate explanatory subgraphs G* and (G')* based on I^L and (I')^L
Structural Mixup adjacency A_our^(mix) and node embeddings H_our^(mix) matrix with Equation 9
Return G_our^(mix) = (H_our^(mix), A_our^(mix))
```

To address the distribution shift issue in the GIB objective, a graph-pooling-based structural mixup strategy is employed to generate in-distribution, naturally connected mixup graphs $G_{\text{our}}^{(\text{mix})}$ that preserve the information of explanatory subgraphs G^*. Built upon this, contrastive learning is introduced as an optimization objective within GIB to capture underlying representations, thereby approximating the mutual information between G^* and the label Y in a task-agnostic manner. For completeness, we present the pseudocode of our method below, which encompasses both explainer training and the structural mixup strategy.

In Algorithm 1, at each epoch, we first randomly sample N neighbors $\{G_i\}_{i=0}^{N-1}$ from the graph dataset $\mathcal{G}$ for each graph G. Based on the similarity between their representations and that of G, we identify the positive neighbor G^+ and the negative neighbors $\{G_i^-\}_{i=1}^{N-1}$. Then, the structural mixup strategy combines the explanatory subgraph G^* with the label-irrelevant parts of G^+ and $\{G_i^-\}_{i=1}^{N-1}$, producing the positive and negative mixup graphs $G_{\text{our}}^{(\text{mix})+}$ and $\{G_{\text{our},i}^{(\text{mix})-}\}_{i=1}^{N-1}$. For each mixup graph, an MLP is employed to predict the edge weight of each edge, and the structural information obtained from graph pooling process is incorporated to generate the final masks $\boldsymbol{M}_{\text{our}}^{(\text{mix})+}$ and $\{\boldsymbol{M}_{\text{our},i}^{(\text{mix})-}\}_{i=1}^{N-1}$. The explainer is optimized by minimizing the InfoNCE loss Equation 10 and the BCE loss Equation 11, while the overall loss function Equation 12 sums the two objectives with a trade-off parameter. Finally, the explainer parameters $f_\psi(\cdot)$ are updated through backpropagation with the overall loss. Specifically, Algorithm 2 shows the structural mixup process. Given a to-be-explained graph G and a sampled neighbor graph G', where G' can be instantiated as the positive neighbor G^+ or one of the negative neighbors $\{G_i^-\}_{i=1}^{N-1}$, we perform L-layer graph pooling to obtain the preserved node indices I^L and $(I')^L$. Then, the explanatory subgraph G^* is represented as $G^* = (\boldsymbol{H}^*, \boldsymbol{A}^*)$, where $\boldsymbol{H}^* = \boldsymbol{H}(I^L, :)$ and $\boldsymbol{A}^* = \boldsymbol{A}(I^L, I^L)$. Similarly, the explanatory subgraph of G' is obtained in the same way based on $(I')^L$. The adjacency matrix $\boldsymbol{A}_{\text{our}}^{(\text{mix})}$ and node

embeddings $\boldsymbol{H}_{\text{our}}^{(\text{mix})}$ of the mixup graph are obtained by replacing $(\boldsymbol{A}')^*$ with $\boldsymbol{A}^*$, as defined in Equation 9. Finally, the mixup graph $G_{\text{our}}^{(\text{mix})} = (\boldsymbol{H}_{\text{our}}^{(\text{mix})}, \boldsymbol{A}_{\text{our}}^{(\text{mix})})$ is returned.

Computational Complexity Analysis. Given graph G, the time complexity of L-layer graph pooling is $O(L \cdot (|\mathcal{E}|d + nd))$, where d is the feature dimension, $\mathcal{E}$ denotes the edge set, and n is the number of nodes in G. Then, structural replacement replaces the explanatory subgraph via m node indices, with a complexity of $O(m^2)$, where m is the number of final preserved nodes. Edge weights for the mixup graph are generated by an MLP and the time complexity is $O(|\mathcal{E}^{(\text{mix})}| \cdot d)$, where $\mathcal{E}^{(\text{mix})}$ denotes the edge set of the mixup graph. Since $m \ll n$, $|\mathcal{E}| \ll n^2$, and the sizes of $|\mathcal{E}|$ and $|\mathcal{E}^{(\text{mix})}|$ are comparable, the overall time complexity of our method is dominated by $O(L \cdot |\mathcal{E}|d)$.

E Full Experimental Setup

Detailed experimental settings are provided in this section, including implementation details, datasets, and baseline methods. All experiments are conducted on a Linux workstation running Ubuntu 22.10 (kernel 5.19.0-46-generic) equipped with 8 NVIDIA GeForce RTX 4090 GPUs (24 GB each). The system uses NVIDIA driver version 535.86.05 and CUDA 12.2. All code is implemented in Python 3.9.21, using PyTorch 2.0.1+cu118, PyTorch Geometric 2.6.1, torch-scatter 2.1.2+pt20cu118, and torch-sparse 0.6.18+pt20cu118. The complete experimental details can be found in our anonymous repository: https://anonymous.4open.science/r/TAME-main-2FB7.

E.1 Implementation Details

Following the configuration in previous work (Ying et al., 2019; Zhang et al., 2023), we divide each dataset into training, validation, and test sets with a ratio of 0.8, 0.1, and 0.1, respectively. We choose GCN as the backbone model for all experiments, as it demonstrates the most stable performance across different datasets. The detailed comparison of the prediction performance of GCN and other GNN architectures are provided in Appendix H.3. A three-layer GCN is used as the target model, and for graph regression tasks, an additional linear layer is appended. We train the GCN model to a reasonable performance, as shown in the GCN column of Tables 5 and 6. For explanation, we follow the sample selection strategy used in GNNExplainer (Ying et al., 2019) and randomly select 200 graph instances per dataset to reduce computational cost. All explanation methods are optimized using the Adam optimizer with a weight decay of 5×10^{-4} (Kingma, 2014). Regarding hyperparameters, we apply grid search to determine the loss weights β, and set the top-r ratios according to the ground truth explanations. For baseline methods with overlapping hyperparameters, we adopt a unified setting; otherwise, we retain their default configurations. We ensure consistent learning rates and training epochs across all explanation methods. We evaluate the performance of our proposed method against baseline approaches on ground-truth explanation tasks using the AUC-ROC score. To quantitatively assess the effectiveness of our method in addressing distributional shifts, we measure the distances between graph representations using cosine similarity and Euclidean distance.

E.2 Datasets

We use nine synthetic datasets and four real-world datasets in our experiments.

BA-2Motifs (Luo et al., 2020). The BA-2Motifs dataset comprises 1,000 synthetic graphs, each generated from a Barabási–Albert graph and extended with either a house or a five-cycle motif. The label of each graph is determined by its attached motif, forming a binary classification task in which the motif serves as the ground-truth explanation.

BA-HouseGrid (Bui et al., 2024). The BA-HouseGrid dataset employs the house motif and the 3×3 grid motif. These motifs are chosen to minimize structural overlap and ensure that models learn the full motif structure instead of relying on local substructures for prediction.

SPMotif (Wu et al., 2022). In the SPMotif dataset, each graph consists of a base structure (Tree, Ladder, or Wheel) and a motif (Cycle, House, or Crane). A parameter b controls the degree of the spurious correlation between the base structure and the motif, with $b = \frac{1}{3}$ indicating no spurious

correlation. In our experiments, $b = 0.7$. The label and ground-truth explanation for each graph are determined solely by the motif it contains.

BA-HouseAndGrid (Bui et al., 2024). The BA-HouseAndGrid dataset contains Barabási–Albert graph graphs extended with the house motif, the grid motif, or both. Graphs are labeled 1 if they contain both motifs, otherwise, they are labeled 0.

Alkane-Carbonyl (Sanchez-Lengeling et al., 2020). The Alkane-Carbonyl dataset comprises 4,326 molecular graphs partitioned into two classes based on the functional groups. A molecule is labeled positive when it contains both alkane and carbonyl groups, which also serve as the ground-truth explanation.

Fluorid-Carbonyl (Sanchez-Lengeling et al., 2020). The Fluorid-Carbonyl dataset comprises 8,671 molecular graphs partitioned into two classes based on functional groups. A molecule is labeled positive if it contains both fluoride atoms and a carbonyl group, which also serve as the ground-truth explanation.

Benzene (Sanchez-Lengeling et al., 2020). The Benzene dataset comprises 12,000 molecular graphs from the ZINC15 (Sterling & Irwin, 2015) database, partitioned into two classes based on the presence of benzene rings. A molecule is labeled positive if it contains at least one benzene ring. Each benzene ring in the molecule serves as a distinct ground-truth explanation.

BA-Motif-Volume (Zhang et al., 2024). In BA-Motif-Volume dataset, each graph is constructed from a Barabási–Albert graph with an attached five-cycle motif. Node features are assigned random float values in the range [0.00, 100.00], and the regression label for each graph is defined as the sum of node feature values over the motif.

House-Grid-Volume. The House-Grid-Volume dataset is derived from BA-HouseGrid (Bui et al., 2024) by replacing all node features with random floats sampled from [0.00, 100.00]. The regression label for each graph is defined as the sum of node features within its motif.

BA-Motif-Counting (Zhang et al., 2024). The BA-Motif-Counting dataset consists of graphs created by attaching a randomly sampled number of five-cycle motifs (with the number varying from $\{0, \ldots, 10\}$ in our experiment) to a Barabási–Albert random graph. The number of motifs in each graph serving as its regression label.

Triangles (Chen et al., 2020b). The Triangles dataset is constructed following the prior work (Chen et al., 2020b), consisting of 5,000 Erdős–Rényi random graphs denoted as $ER(m, p)$, where $m = 30$ is the number of nodes in each graph and $p = 0.1$ is the edge existence probability. The regression label for each graph is the number of triangles it contains. In our experiments, we use a subset of 1,000 graphs randomly sampled from this dataset.

Crippen (Delaney, 2004). The Crippen dataset contains 1,127 molecules with corresponding aqueous solubility measurements from the Delaney solubility (Delaney, 2004) dataset and assigns node weights using the Crippen model (Wildman & Crippen, 1999). Following prior work (Sanchez-Lengeling et al., 2020), we adopt this dataset and generate edge weights as the average of incident node weights.

House-OrGrid-Volume. The House-OrGrid-Volume dataset is derived from BA-HouseOrGrid (Bui et al., 2024) by replacing all node features with random floats sampled from [0.00, 100.00]. The regression label for each graph is defined as the sum of node features within its motif.

E.3 BASELINES

To assess effectiveness, we incorporate various post-hoc methods, including GNNExplainer, PGExplainer, MetaGNN, MatchExplainer, MixupExplainer, ProxyExplainer, and RegExplainer, as well as the gradient-based GRAD and the attention-based ATT.

GRAD (Ying et al., 2019). GRAD is a gradient-based method that generates weights to edges and nodes by computing the gradients of the loss function of GNN with respect to the adjacency matrix and node features.

ATT (Veličković et al., 2017). ATT is a graph attention network that learns attention weights for edges in the input graph, with these weights are adopted as a proxy measure of edge importance.

GNNExplainer (Ying et al., 2019). GNNExplainer learns soft masks over edges and node features for each instance by minimizing the mutual information between the original graph and the prediction results. The explanatory subgraphs are obtained via element-wise multiplication of the learned soft masks with the original graph.

PGExplainer (Luo et al., 2020). PGExplainer extends GNNExplainer by parameterizing the explanation generation process with a trainable explainer, and generates a soft mask to produce the explanatory subgraph.

MetaGNN (Spinelli et al., 2022). MetaGNN trains GNNs to be inherently interpretable by incorporating a meta-explainer, which generates post-hoc explanations during training.

MatchExplainer (Wu et al., 2023). MatchExplainer explains GNN predictions by matching shared subgraph patterns using graph edit distance (GED) as the similarity metric, and this non-parametric subgraph matching approach inherently avoids optimization bias.

MixupExplainer (Zhang et al., 2023). MixupExplainer mitigates the distribution shift present in previous methods by mixing explanatory subgraphs with the label-irrelevant parts of other randomly sampled graphs in a non-parametric manner.

ProxyExplainer (Chen et al., 2024). ProxyExplainer performs autoencoders to reconstruct the explanatory and label-irrelevant parts, then combines them to generate proxy graphs in a parametric manner, approximating the original distribution to mitigate the OOD issue.

RegExplainer (Zhang et al., 2024). RegExplainer addresses interpretability in graph regression tasks by combining mixup with contrastive learning to mitigate distribution shift and tackle the challenge of continuously ordered labels.

F PROOF OF PROPERTY 1

Proof. The mutual information between the explanatory subgraph G^* and the label Y can be formulated as the Kullback–Leibler (KL) divergence between the joint distribution $p(G^*, Y)$ and the product of its marginals:

$$I(G^*; Y) = \mathrm{KL}(p(G^*, Y) \,\|\, p(G^*)p(Y)).$$

The Donsker–Varadhan (DV) variational representation of KL divergence Poole et al. (2019) provides the following inequality:

$$\mathrm{KL}(p \,\|\, q) = \sup_T \left\{ \mathbb{E}_p[T] - \log \mathbb{E}_q[e^T] \right\},$$

where the supremum is taken over all measurable functions $T : \mathcal{G} \times \mathcal{Y} \to \mathbb{R}$ such that the expectations exist. The bound is tight when $T(x) = \log \frac{p(x)}{q(x)}$.

Applying this representation to the mutual information yields:

$$I(G^*; Y) \geq \mathbb{E}_{p(G^*,Y)}[T(G^*, Y)] - \log \mathbb{E}_{p(G^*)p(Y)} \left[e^{T(G^*,Y)} \right].$$

This variational characterization establishes a theoretical basis for estimating mutual information by choosing tractable parameterizations of the critic function T.

□

G PROOF OF PROPERTY 2

Proof. To instantiate the critic function $T(G^*, Y)$, we leverage the fact that each label Y in the dataset is paired with a real input graph G_Y, i.e., $(G_Y, Y) \sim p(G, Y)$. Instead of encoding the label Y directly, we obtain its representation via the associated graph G_Y, which reflects the semantics of Y in the same space as G^*. This design ensures that both G^* and Y are represented in the same embedding space, while preserving the theoretical foundation of estimating $I(G^*; Y)$. Let $f(\cdot)$ denote an encoder that maps a graph to its representation, and define $\boldsymbol{h}_{G^*} = f(G^*)$ and $\boldsymbol{h}_Y = f(G_Y)$. The similarity between two representations is defined as $\mathrm{sim}(u, v) := u^\top v$, where u and v are ℓ_2-normalized vectors.

We instantiate the Donsker–Varadhan critic function $T : \mathcal{G} \times \mathcal{Y} \to \mathbb{R}$ as:

$$T(G^*, Y) := \mathrm{sim}(\boldsymbol{h}_{G^*}, \boldsymbol{h}_Y) + \log(N{-}1),$$

where N is the total number of samples (including one positive and $N{-}1$ negatives) in each estimation batch. We first evaluate the expectation under the joint distribution $p(G^*, Y)$ (positive pairs):

$$\mathbb{E}_{p(G^*,Y)}[T(G^*, Y)] = \mathbb{E}_{p(G^*,Y)}\left[\mathrm{sim}^+ + \log(N{-}1)\right],$$

where $\mathrm{sim}^+ := \mathrm{sim}(\boldsymbol{h}_{g^*}, \boldsymbol{h}_{y^+})$ denotes the similarity score of the positive pair.

To evaluate the negative term, we apply Jensen's inequality to the expectation under the product of marginals:

$$-\log \mathbb{E}_{p(G^*)p(Y)}\left[e^{T(G^*,Y)}\right] \geq -\mathbb{E}_{G^*\sim p(G^*)}\left[\log \mathbb{E}_{Y\sim p(Y)}\left[e^{T(G^*,Y)}\right]\right].$$

We approximate the inner expectation using $N{-}1$ i.i.d. samples $\{y_j^-\}_{j=1}^{N-1}$ drawn from $p(Y)$:

$$\mathbb{E}_Y[e^{T(G^*,Y)}] \approx \frac{1}{N{-}1}\sum_{j=1}^{N-1}\exp\left(\mathrm{sim}_j^- + \log(N{-}1)\right) = (N{-}1)\cdot\frac{1}{N{-}1}\sum_{j=1}^{N-1}\exp\left(\mathrm{sim}_j^-\right),$$

where $\mathrm{sim}_j^- := \mathrm{sim}(\boldsymbol{h}_{g^*}, \boldsymbol{h}_{y_j^-})$ denotes the similarity between G^* and each negative sample y_j^-.

Taking the logarithm, we obtain:

$$\log \mathbb{E}_Y[e^{T(G^*,Y)}] \approx \log(N{-}1) + \log\left(\frac{1}{N{-}1}\sum_{j=1}^{N-1}\exp\left(\mathrm{sim}_j^-\right)\right),$$

and thus, the negative term becomes:

$$-\log \mathbb{E}_{p(G^*)p(Y)}[e^{T(G^*,Y)}] \geq -\log(N{-}1) - \mathbb{E}_{\mathcal{P}}\left[\log\left(\frac{1}{N{-}1}\sum_{j=1}^{N-1}\exp\left(\mathrm{sim}_j^-\right)\right)\right],$$

where $\mathcal{P} := p(G^*, Y_0)\prod_{j=1}^{N-1} p(Y_j^-)$ denotes the joint sampling distribution comprising one positive pair (G^*, Y_0) and $N{-}1$ independent negative samples $\{Y_j^-\}$.

Combining the positive and negative components of the DV bound:

$$
\begin{aligned}
I(G^*;Y) &\geq \mathbb{E}_{\mathcal{P}}\left[\mathrm{sim}^+ + \log(N{-}1)\right] - \log(N{-}1) - \mathbb{E}_{\mathcal{P}}\left[\log\left(\frac{1}{N{-}1}\sum_{j=1}^{N-1}\exp\left(\mathrm{sim}_j^-\right)\right)\right] \\
&= \mathbb{E}_{\mathcal{P}}\left[\mathrm{sim}^+\right] - \mathbb{E}_{\mathcal{P}}\left[\log\left(\frac{1}{N{-}1}\sum_{j=1}^{N-1}\exp\left(\mathrm{sim}_j^-\right)\right)\right] \\
&= \mathbb{E}_{\mathcal{P}}\left[\log\left(\exp\left(\mathrm{sim}^+\right)\right) - \log\left(\frac{1}{N{-}1}\sum_{j=1}^{N-1}\exp\left(\mathrm{sim}_j^-\right)\right)\right] \\
&= \mathbb{E}_{\mathcal{P}}\left[\log\left(\frac{\exp\left(\mathrm{sim}^+\right)}{\frac{1}{N-1}\sum_{j=1}^{N-1}\exp\left(\mathrm{sim}_j^-\right)}\right)\right] \\
&= \log(N{-}1) + \mathbb{E}_{\mathcal{P}}\left[\log\left(\frac{\exp\left(\mathrm{sim}^+\right)}{\sum_{j=1}^{N-1}\exp\left(\mathrm{sim}_j^-\right)}\right)\right].
\end{aligned}
$$

Since the logarithm is monotonically increasing, augmenting the denominator with the positive term yields a smaller ratio and hence a looser (but valid) bound:

$$
I(G^*;Y) \geq \log(N{-}1) + \mathbb{E}_{\mathcal{P}}\left[\log\left(\frac{\exp\left(\mathrm{sim}^+\right)}{\exp\left(\mathrm{sim}^+\right) + \sum_{j=1}^{N-1}\exp\left(\mathrm{sim}_j^-\right)}\right)\right].
$$

We define the InfoNCE estimator as:

$$
\ell(G^*, Y) := \log\frac{\exp\left(\mathrm{sim}(\boldsymbol{h}_{g^*}, \boldsymbol{h}_{y^+})\right)}{\sum_{j=0}^{N-1}\exp\left(\mathrm{sim}(\boldsymbol{h}_{g^*}, \boldsymbol{h}_{y_j})\right)},
$$

where $\{y_j\}_{j=0}^{N-1}$ includes one positive label $y_0 = y^+$ and $N{-}1$ negative samples y_j^-.

Thus, we arrive at the final variational lower bound:

$$
\boxed{I(G^*;Y) \geq \log(N{-}1) + \mathbb{E}_{\mathcal{P}}\left[\ell(G^*, Y)\right]}.
$$

□

H EXTENSIVE EXPERIMENTS

H.1 ABLATION STUDY

We conduct ablation studies to evaluate the contribution of each component in the TAME framework. Specifically, we design the following variants: 1) w/o Mix: Removes the structral mixup step after extracting the explanation, directly feeding the explanation subgraph into the objective function. 2) w/o CTL: Retains the structral mixup strategy and BCE loss, but replaces the contrastive learning term with cross-entropy loss for classification and mean squared error (MSE) loss for regression. 3) w/o BCE: Removes only the binary mask behavior regularization term from the training loss. All variants are configured with the same hyperparameters as the original TAME, including the learning rate, number of training epochs, and the loss weights β.

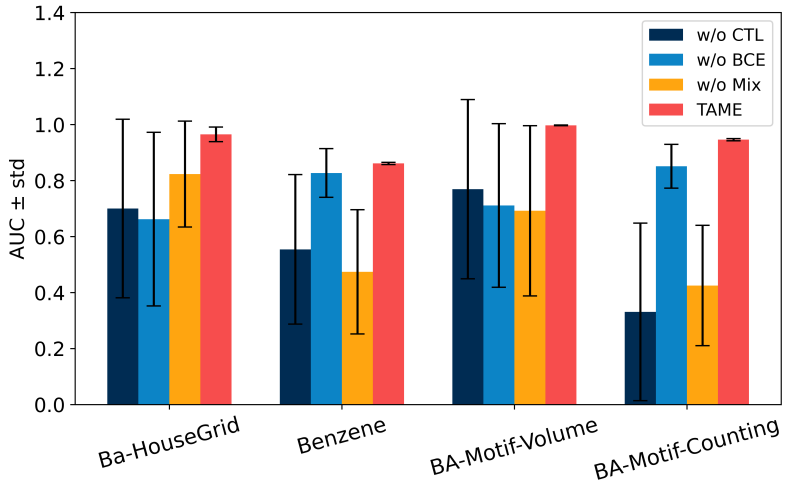


Figure 6: Ablation study of TAME across classification and regression tasks. We evaluate the AUC-ROC performance of the original TAME and its variants. Standard deviations are indicated by black error bars.

We conduct experiments on representative datasets covering both classification and regression tasks, with BA-HouseGrid and Benzene used for classification, and BA-motif-volume and BA-Motif-Counting used for regression. The results are presented in Figure 6. Experimental results show that TAME consistently outperforms all its variants across all datasets, indicating that each component contributes positively to the overall performance. Notably, the performance of the *w/o CTL* variant drops significantly on both classification and regression datasets, indicating that our proposed contrastive learning mechanism plays a critical role in both tasks.

H.2 HYPER-PARAMETER SENSITIVITY STUDY

We further investigate the sensitivity of our proposed method to the hyperparameter β, which controls the weight of the BCE loss. Specifically, we conduct experiments on four representative datasets, including the Ba-HouseAndGrid and Benzene classification datasets, as well as the BA-Motif-Volume and House-Grid-Volume regression datasets. We vary β from 0.1 to 1.0 in increments of 0.1 while keeping the contrastive learning weight fixed at 1.0. The experiments are conducted under ten random seeds, where the solid lines represent the average AUC of our method and the dashed lines correspond to the second-best performing method. The experimental results are presented in Figure 7. The results indicate that TAME maintains stable performance across both classification and regression tasks. For some datasets, the AUC slightly decreases and exhibits larger variance when the weight of the BCE loss is small. This phenomenon can be attributed to the role of the BCE loss in constraining the edge weights to remain faithful to the explanation structure extracted by graph pooling, while simultaneously reducing the weights of non-explanatory structures. Without

this constraint, the edge weights may not be effectively optimized, leading to a decline in performance.

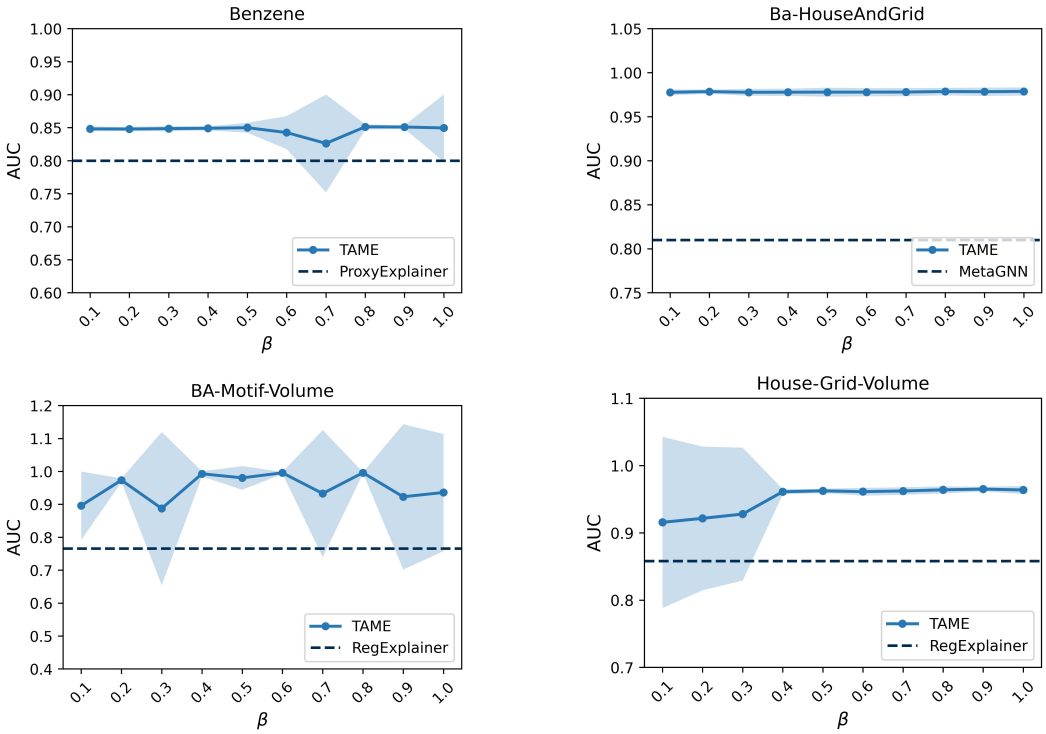


Figure 7: Sensitivity analysis of hyperparameter β for BCE loss.

H.3 BACKBONE MODELS ANALYSIS

We conduct a performance analysis of common backbone models employed by explainability methods on both classification and regression tasks. For the classification task, model performance is evaluated using ACC and Macro_F1 metrics on four synthetic datasets, including BA-2Motifs, BA-HouseGrid, BA-HouseAndGrid, and SPMotif, as well as three real-world datasets including Alkane-Carbonyl, Fluoride-Carbonyl, and Benzene. For the regression task, RMSE and MAPE are adopted as evaluation metrics on five synthetic datasets including BA-Motif-Volume, BA-Motif-Counting, Triangles, House-Grid-Volume, and House-OrGrid-Volume, along with one real-world dataset, Crippen.

Experimental results demonstrate that GCN and GIN achieve strong performance in classification tasks, while GCN consistently attains the best performance across regression benchmarks. Therefore, to ensure that the explanation quality is not confounded by the performance of the backbone model, we choose GCN as the backbone model for explainer evaluation.

Dataset	**GCN**		**GIN**		**GAT**	
	ACC	Macro_F1	ACC	Macro_F1	ACC	Macro_F1
BA-2motifs	99.00	98.98	**100.00**	**100.00**	44.00	30.55
BA-HouseGrid	99.90	99.89	**100.00**	**100.00**	49.10	32.93
BA-HouseAndGrid	98.40	98.39	**99.90**	**99.89**	48.90	32.84
SPMotif	**96.16**	**96.15**	95.88	95.89	34.05	18.98
Alkane-Carbonyl	**95.57**	**95.13**	92.03	91.24	94.69	94.31
Fluoride-Carbonyl	94.35	89.99	**97.23**	**95.03**	83.29	45.44
Benzene	90.58	90.53	**92.66**	**92.65**	79.08	79.06

Table 5: Comparison of GCN, GIN, and GAT on classification datasets using ACC and Macro-F1 metrics.

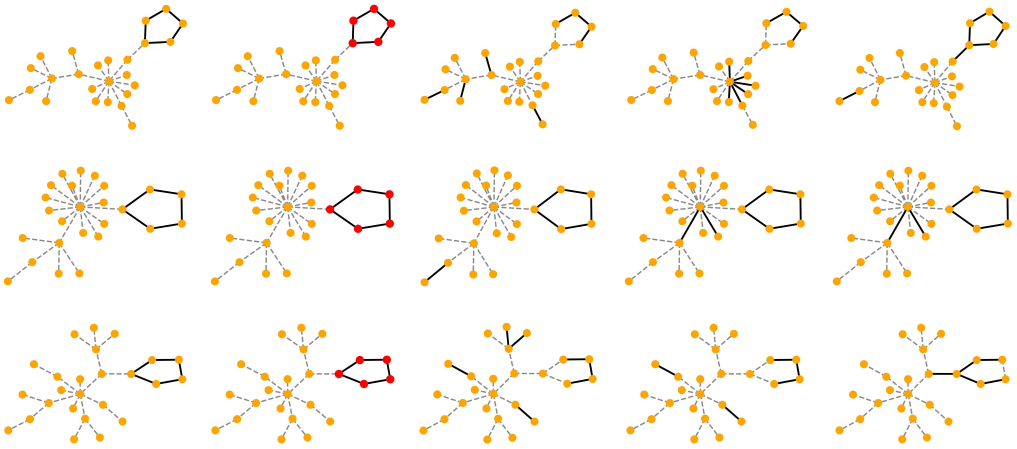
Dataset	**GCN**		**GIN**		**GAT**	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
BA-Motif-Volume	**234.20**	**7.47**	398.96	14.94	485.31	17.45
BA-Motif-Counting	**0.39**	—	1.94	—	2.84	—
Triangles	**2.00**	—	2.46	—	2.55	—
House-Grid-Volume	**11.32**	**2.45**	94.54	29.56	49.85	13.88
House-OrGrid-Volume	**34.15**	**6.69**	152.14	28.17	53.64	10.27
Crippen	**0.92**	**30.71**	1.91	123.17	1.16	74.68

Table 6: Comparison of GCN, GIN, and GAT on regression datasets using RMSE and MAPE metrics. MAPE values are marked with dashes for BA-Motif-Counting and Triangles due to the presence of label-zero samples that make MAPE undefined.

H.4 Extensive Case Study

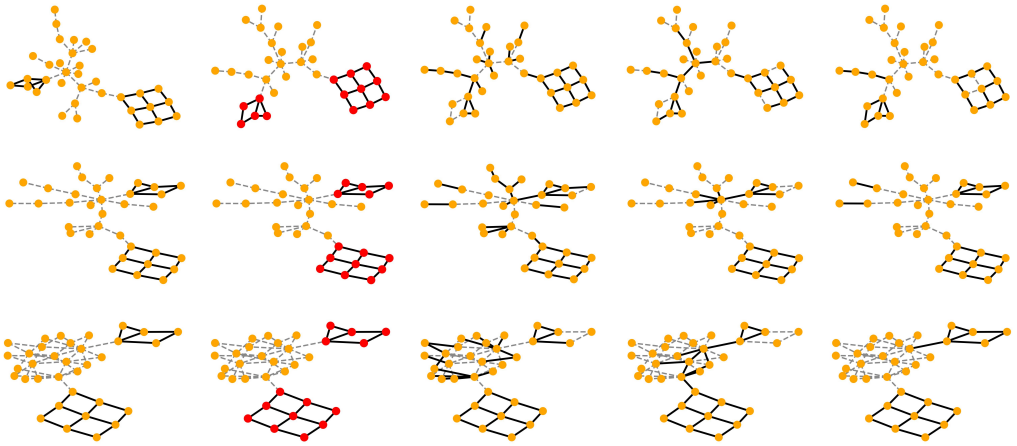
H.4.1 Visualization of Additional Explanation Results

We further present extensive visualization experiments on both classification and regression datasets. The classification datasets include BA-2Motifs, BA-HouseAndGrid, and Benzene, as shown in Figure 8, Figure 9, and Figure 10, respectively. The regression datasets include BA-Motif-Volume, and House-OrGrid-Volume, as presented in Figure 11, and Figure 12, respectively. For each dataset, we randomly select three target graphs to evaluate different explainers.



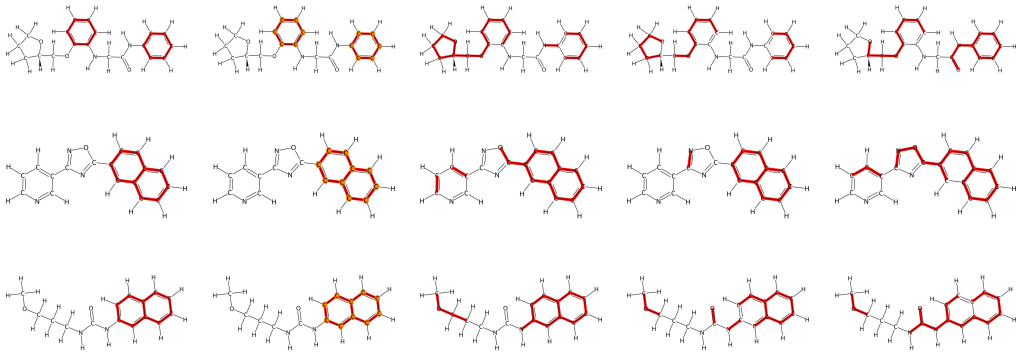
(a) Ground Truth (b) TAME (c) ProxyExplainer (d) MixupExplainer (e) MetaGNN

Figure 8: Visualization of explanation on BA-2motifs.



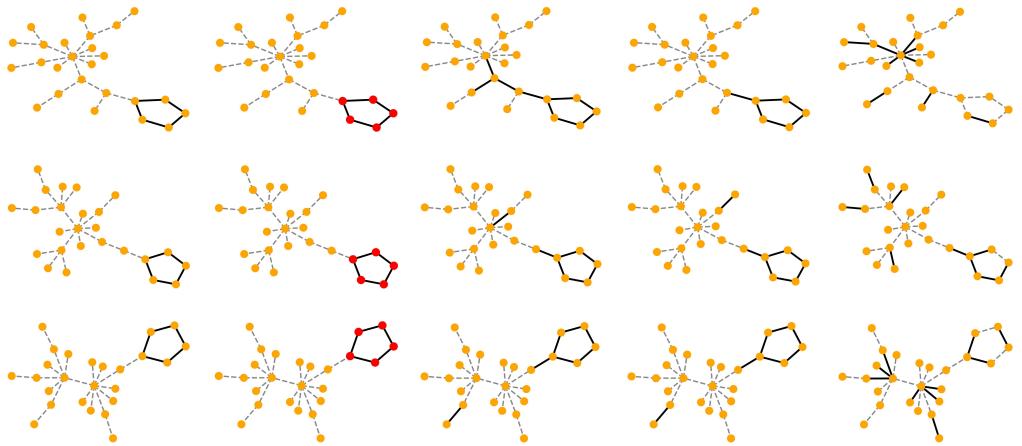
(a) Ground Truth (b) TAME (c) ProxyExplainer (d) MixupExplainer (e) MetaGNN

Figure 9: Visualization of explanation on BA-HouseAndGrid.



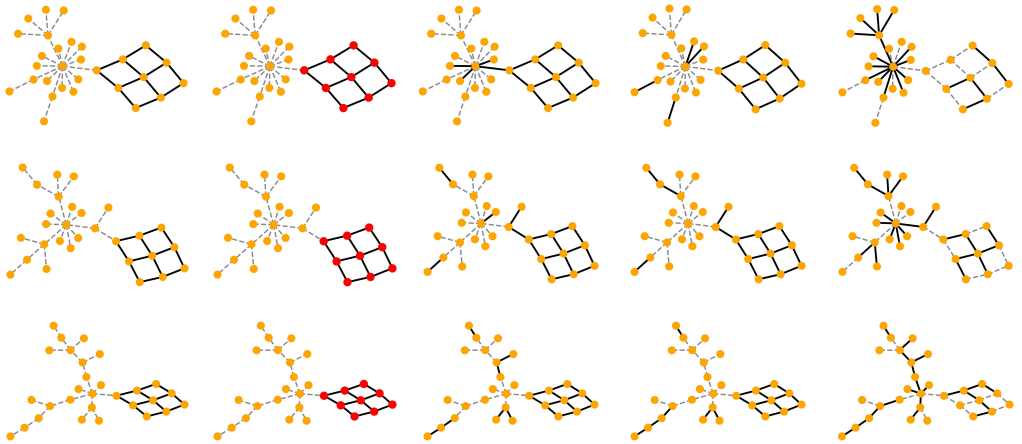
(a) Ground Truth (b) TAME (c) ProxyExplainer (d) MixupExplainer (e) MetaGNN

Figure 10: Visualization of explanation on Benzene.



(a) Ground Truth (b) TAME (c) ProxyExplainer (d) MixupExplainer (e) MetaGNN

Figure 11: Visualization of explanation on BA-Motif-Volume.

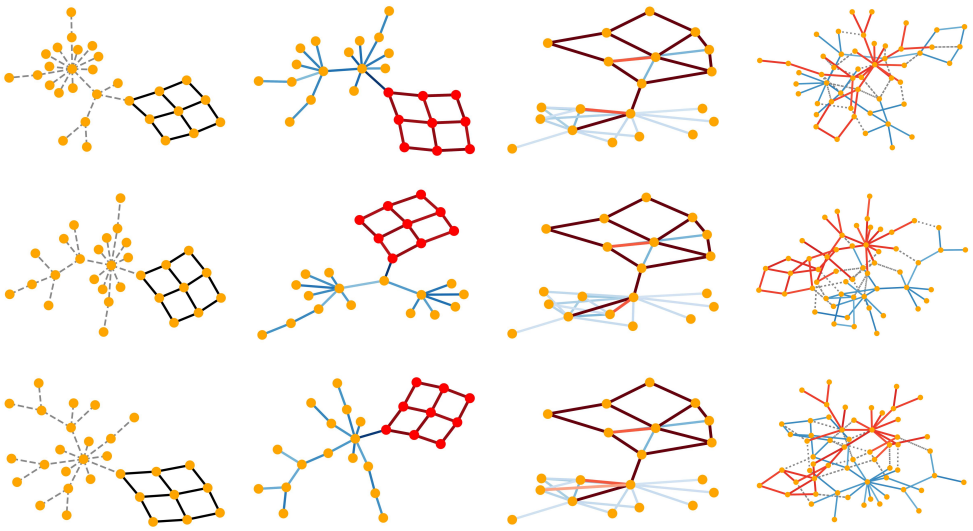


(a) Ground Truth (b) TAME (c) ProxyExplainer (d) MixupExplainer (e) MetaGNN

Figure 12: Visualization of explanation on House-Grid-Volume.

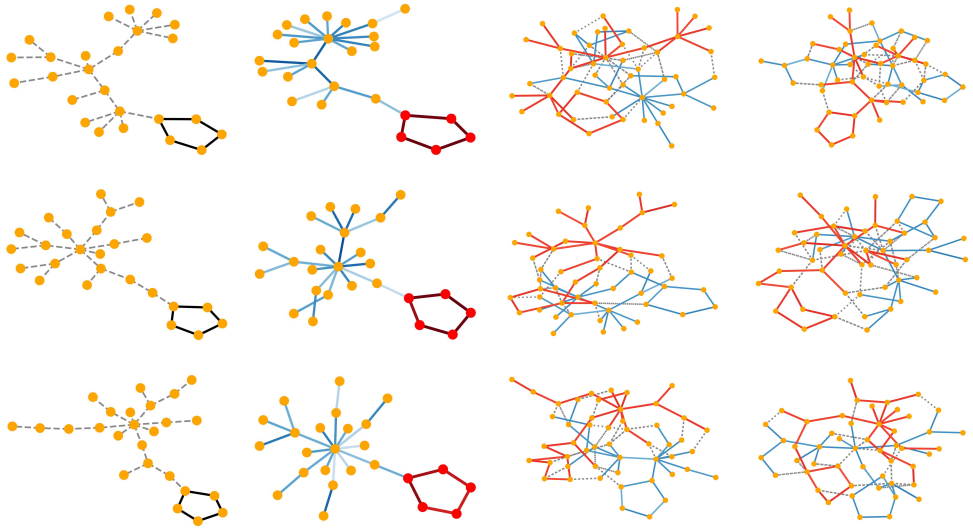
H.4.2 Visualization of Additional mixup Results

We present the mixup results on the BA-HouseGrid classification dataset and the BA-Motif-Volume regression dataset in Figure 13 and Figure 14, respectively. For each dataset, three graph samples are randomly selected to generate the mixup results, where the edge color intensity indicates the corresponding weight magnitude. It should be noted that ProxyExplainer employs a graph generation strategy to produce mixup results, which leads to fully connected edges. To facilitate visualization, we display only the top 64 edges with the highest weights, corresponding to the maximum number of edges considered in this experiment.



(a) Ground Truth (b) TAME (c) ProxyExplainer (d) MixupExplainer

Figure 13: Visualization of mixup graphs on BA-HouseGrid.



(a) Ground Truth (b) TAME (c) RegExplainer (d) MixupExplainer

Figure 14: Visualization of mixup graphs on BA-Motif-Volume.