Towards Robust Heterogeneous Graph Explanations under Structural Perturbations

Yifan Lu
Hangzhou Dianzi University
MBZUAI
Abu Dhabi, UAE
Yifan.Lu@mbzuai.ac.ae

Pengfei Jiao*
School of Cyberspace
Hangzhou Dianzi University
Hangzhou, China
pjiao@hdu.edu.cn

Xuan Guo
College of Intelligence and Computing
Tianjin University
Tianjin, China
guoxuan@tju.edu.cn

Ziyun Zou
MBZUAI
Abu Dhabi, UAE
Ziyun.Zou@mbzuai.ac.ae

Yiwei Wang
University of California, Merced
Merced, California, USA
yiweiwang2@ucmerced.edu

Mengzhou Gao
School of Cyberspace
Hangzhou Dianzi University
Hangzhou, China
mzgao@hdu.edu.cn

Huaming Wu
Center for Applied Mathematics
Tianjin University
Tianjin, China
whming@tju.edu.cn

Imran Razzak
MBZUAI
Abu Dhabi, UAE
Imran.Razzak@mbzuai.ac.ae

Abstract

Explaining the decision-making process of Graph Neural Networks (GNNs) is essential for improving their transparency and reliability. However, real-world graphs are often heterogeneous and subject to structural noise, posing severe challenges to the robustness of existing explanation methods. To address these issues, we propose RoHeX, a Robust Heterogeneous GNN Explainer that enhances explanation quality under noisy conditions. RoHeX begins with a theoretical analysis revealing how different heterogeneous GNN architectures amplify structural perturbations through message passing. Building on this insight, we design a denoising variational inference framework that filters noisy structures and learns robust latent graph representations. Furthermore, we incorporate relation-aware heterogeneous semantics into the explanation generation process, formulating explanation as an optimization problem under the graph information bottleneck principle. This formulation enables RoHeX to balance fidelity and compactness, producing explanations that are both semantically meaningful and structurally stable. Comprehensive experiments on multiple real-world heterogeneous graphs demonstrate that RoHeX consistently surpasses state-of-the-art baselines in explanation fidelity, robustness to structural perturbations, and explainability.

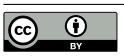
CCS Concepts

• **Information systems** → **Web mining**; *Data mining*; • **Computing methodologies** → *Machine learning*.

Keywords

Graph Neural Network; Heterogeneous Graph; Explainability

*Corresponding author.



This work is licensed under a Creative Commons Attribution 4.0 International License.
WWW '26, Dubai, United Arab Emirates
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792516

ACM Reference Format:
Yifan Lu, Pengfei Jiao, Xuan Guo, Ziyun Zou, Yiwei Wang, Mengzhou Gao, Huaming Wu, and Imran Razzak. 2026. Towards Robust Heterogeneous Graph Explanations under Structural Perturbations. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3774904.3792516

1 Introduction

Graph Neural Networks (GNNs) have become indispensable for modeling complex relational data on the Web, supporting a wide range of applications such as online social networks [3], recommendation systems [48], and content understanding platforms [1]. By integrating node attributes with the structural information of web-scale graphs, GNNs have achieved remarkable success in tasks including node classification [22, 51], link prediction [19, 26], and community detection [15, 17]. However, as these models increasingly influence user-facing decisions on the Web, such as content recommendation, search ranking, and misinformation detection, their opaque decision-making process has raised critical concerns regarding transparency, fairness, and trustworthiness [25, 28, 35]. Understanding why a GNN produces a particular prediction is therefore essential for ensuring accountability and reliability in Web-scale intelligent systems.

To address this, GNN explainers aim to reveal the decision rationale behind model predictions by identifying critical substructures. Existing approaches fall into two categories: post-hoc methods [10, 29, 34], which explain a pretrained model without modifying it, and built-in methods [32, 44, 49], which generate explanations during model training. Post-hoc methods are more flexible and generalizable, while built-in methods often yield task-specific insights. However, both struggle under complex, noisy conditions—especially in heterogeneous graphs, which are the norm in real-world settings [36, 39].

Heterogeneous graphs composed of diverse node and edge types pose substantial challenges for explainability [16, 20]. Their intertwined semantics and irregular structures make subgraph extraction inherently difficult. In addition, real-world graphs often contain

noise such as irrelevant or missing edges, which further complicates the pursuit of robust model explanations [4, 6]. The diversity of data distributions, attribute spaces, and application domains across heterogeneous graphs also constrains the generalizability of existing explainers. Although post-hoc methods offer greater flexibility and broader applicability, they remain sensitive to structural irregularities and noise that may distort the model reasoning process. Furthermore, noise intensifies structural inconsistencies and alters node importance, thereby diminishing the effectiveness of conventional approaches that rely on rigid constraints such as subgraph size, connectivity, or budget [21, 52].

In this paper, we propose RoHeX, a Robust Heterogeneous Graph Neural Network Explainer. First, we theoretically analyze how noise amplification occurs in heterogeneous GNNs. Second, we introduce a denoising variational inference module that learns robust latent representations by filtering noise in the input graph. Third, we design a heterogeneous explanation generator based on relation-aware attention, which captures the rich semantics across node and edge types. Finally, by integrating the Graph Information Bottleneck (GIB) principle, we reframe explanation as an information-theoretic optimization problem to better handle irregular structures. We validate RoHeX's performance through extensive experiments on multiple real-world datasets, demonstrating its superior ability to handle noise and generate high-quality explanations compared to state-of-the-art GNN explainers. The contributions of this paper are as follows:

- We present the first systematic analysis of how noise interacts with heterogeneity in GNN explainers, theoretically proving that existing heterogeneous GNNs amplify structural noise and degrade explainability.
- We propose a robust heterogeneous GNN explainer RoHeX that integrates denoising variational inference with a relation-aware explanation generator to jointly handle heterogeneity and noise in explanation generation.
- Extensive experiments on multiple real-world heterogeneous graphs demonstrate that RoHeX consistently outperforms existing explainers in both explanation fidelity and robustness under noisy conditions.

2 Related Work

2.1 GNN Explainability.

Existing GNN explanation methods can generally be categorized into post-hoc and built-in approaches. Post-hoc methods aim to interpret pretrained GNNs without altering their parameters, as in GraphMask [31] and Vu et al. [34]. Surrogate model-based methods, including LIME [10] and Pereira et al. [29], approximate GNN behavior using interpretable surrogate models. Other approaches learn explanation masks directly: GNNExplainer [41] optimizes feature and edge masks to obtain the most informative subgraph; PGExplainer [21] and PGM-Explainer [33] both employ parametric or probabilistic models to estimate edge importance and capture node dependencies for generating explanatory subgraphs. MixupExplainer [45] extends the Graph Information Bottleneck (GIB) framework by introducing label-independent subgraph sampling, effectively mitigating distribution shift during explanation generation. Built-in methods incorporate interpretability directly into the model architecture or training objective. Prototype-based frameworks such as PGIB [32] and ProtGNN [49] learn representative subgraphs that serve as class prototypes. Graph generation-based methods like XGNN [44] explicitly construct subgraphs that maximize the model's confidence in a target class, visualizing the decision process. GOAt [18] decomposes model outputs into scalar products over node and edge features to quantify feature contributions, while xPath [14] employs a graph rewiring mechanism to identify causal nodes and influence paths, offering fine-grained explanations. AMExplainer [47] introduces an adversarial training scheme that balances sparsity and fidelity, improving both the clarity and efficiency of generated explanations. Despite these advances, existing approaches still face challenges in achieving faithful, scalable, and robust explanations across diverse graph domains, motivating continued research toward more generalizable and principled GNN interpretability frameworks.

2.2 Heterogeneous Graph Neural Networks

Heterogeneous Graph Neural Networks can be broadly categorized into meta-path-based and neighborhood aggregation-based methods. Meta-path-based approaches decompose heterogeneous graphs into multiple homogeneous subgraphs using predefined meta-paths to capture specific semantic relations, followed by intra-subgraph message passing and inter-path aggregation, as in HAN [37], MAGNN [7], Paths2Pair [9], MAGNET [38], and SeHGNN [40]. In contrast, neighborhood aggregation-based methods directly aggregate information from heterogeneous neighbors using type-specific aggregation strategies, with representative examples including RGCN [30], NARS [42], MHGCN [43], Simple-HGN [23], and HERO [27].

3 Preliminary and Problem Definition

3.1 Heterogeneous Graph

A heterogeneous graph, denoted as $\mathcal{G} = (\mathbf{A}, \mathbf{X}, \mathcal{A}, \mathcal{R}, \Phi)$, encompasses multiple types of nodes $\mathcal{V}$ and edges $\mathcal{E}$, where $\mathbf{A}$ is the corresponding adjacency matrix, $\mathbf{X}$ represents node features, $\mathcal{A}$ denotes the set of node types, $\mathcal{R}$ signifies the set of edge types, and Φ represents the set of meta-paths. A meta-path $\phi \in \Phi$ is a path of edges connecting various types of nodes from node v_1 to node v_{l+1}, such as $\mathcal{A}_1 \xrightarrow{\mathcal{R}_1} \mathcal{A}_2 \xrightarrow{\mathcal{R}_2} \dots \xrightarrow{\mathcal{R}_l} \mathcal{A}_{l+1}$, where l denotes the length of the meta-path. The label set of $\mathcal{G}$ is denoted as $\mathbf{Y}$, comprising C categories. Meanwhile, a heterogeneous graph has two mapping functions $\psi(v) : \mathcal{V} \to \mathcal{A}$ and $\varphi(e) : \mathcal{E} \to \mathcal{R}$ that correspond to nodes and edges, respectively.

3.2 Heterogeneous Graph Neural Network Explainer

Given a trained HGNN model f as the subject of explanation and a heterogeneous graph $\mathcal{G}$, the objective of the HGNN explainer is to identify the most influential subgraph $\mathcal{G}_s = (\mathbf{A}_s, \mathbf{X}, \mathcal{A}_s, \mathcal{R}_s)$. Here, $\mathbf{A}_s$ represents the adjacency matrix formed by nodes $\mathcal{V}_s$ and $\mathcal{E}_s$ which significantly contribute to prediction. For the original prediction of GNN model f, it can be formalized as follows:

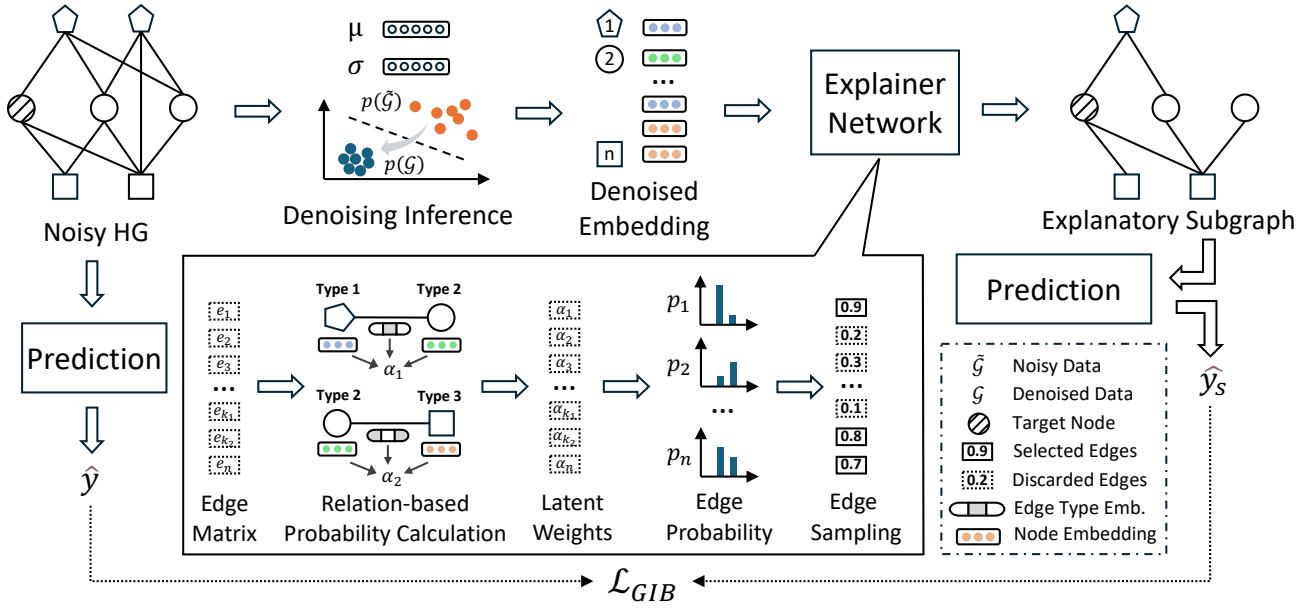


Figure 1: The architecture of our proposed RoHeX. First, the denoised node representations are obtained from the noisy graph via denoising variational inference. Then, the Explainer Network employs the heterogeneous relation-based importance computation method to obtain the weights for different edges. The top k percent of edges are selected as important edges to generate the explanatory subgraph. Finally, the generated explanatory subgraph and the original graph are respectively input into heterogeneous GNN models to obtain predictions, which are used to compute the loss function.

$$\hat{y} = \underset{c \in C}{\operatorname{argmax}} P_f(c|\mathbf{A}, \mathbf{X}, \mathcal{A}, \mathcal{R}), \tag{1}$$

where $P_f(\cdot)$ represents the prediction function of the HGNN model f. Current research indicates that graph structural information is crucial for classification tasks [21, 53].

Therefore, our RoHeX focuses on exploring structural noise when generating explanations. The excellent explanation should contain the most critical information to approximate the predicted labels and outcome changing when predicting the remaining part of the original graph, as follows:

$$\underset{c \in C}{\operatorname{argmax}} P_f(c|\mathbf{A}_s, \mathbf{X}, \mathcal{A}_s, \mathcal{R}_s) = \hat{y}. \tag{2}$$

4 Methodology

Figure 1 illustrates the overall framework of RoHeX. We begin by applying a denoising variational graph encoder to obtain a robust latent representation of the input graph $\mathcal{G}$. Node embeddings sampled from the latent structural distribution are used to construct edge representations. These edge features are then fed into a heterogeneous relation-aware attention module, which estimates the importance of each edge by modeling the semantics of different edge types. Based on the learned importance scores, RoHeX generates a compact and informative explanatory subgraph. Finally, the entire process is optimized under the graph information bottleneck objective, which adaptively promotes structural sparsity and robustness against irregularities.

4.1 Controllable Structural Perturbation for Heterogeneous Graphs

Real-world graph noise often arises from missing or spurious edges and is typically modeled via random edge addition or deletion [24, 46, 50]. Building upon this idea, we introduce a controllable and heterogeneous-aware structural perturbation strategy, a heuristic but flexible method designed to simulate realistic noise, maintain comparability with prior work, and enable targeted evaluation under adversarial or highly irregular settings. Notably, this method can be changed into a graph structure attack method to achieve structural corruption.

Per-relation stochastic operator. For each relation $r \in \mathcal{R}$, let $\mathbf{A}_r \in \{0,1\}^{|\mathcal{V}_r| \times |\mathcal{V}_r|}$ denote the adjacency matrix of that relation. We define a random perturbation tensor Ξ_r whose entries are drawn independently as:

$$\Xi_{ij}^r \sim \begin{cases} \text{Bernoulli}(\eta_r^+), & (i,j) \in \mathcal{E}_r, \\ \text{Bernoulli}(\eta_r^-), & (i,j) \notin \mathcal{E}_r. \end{cases} \tag{3}$$

Here η_r^+ and η_r^- represent the deletion and addition probabilities for relation r. The perturbed adjacency is obtained by:

$$\widetilde{\mathbf{A}}_r = \mathbf{A}_r - \mathbf{A}_r \odot \Xi_r + (1 - \mathbf{A}_r) \odot \Xi_r, \tag{4}$$

where $\odot$ denotes elementwise multiplication. This unified formulation simultaneously handles edge removal and insertion in a single stochastic operator.

Global budget and relation-wise calibration. To ensure comparability across relations and avoid excessive corruption, we impose a

global perturbation budget B representing the total number of modified edges. This budget is allocated to each relation proportionally to its structural significance:

$$B_r = B \cdot \frac{s_r}{\sum_{r' \in \mathcal{R}} s_{r'}}, \qquad s_r = \frac{|\mathcal{E}_r|}{|\mathcal{E}|} \cdot \log |\mathcal{E}_r|. \tag{5}$$

The per-type probabilities are normalized accordingly:

$$\eta_r^+ = \frac{B_r^+}{|\mathcal{E}_r|}, \qquad \eta_r^- = \frac{B_r^-}{|\mathcal{V}_r|^2 - |\mathcal{E}_r|}, \quad \text{s.t. } B_r^+ + B_r^- = B_r. \tag{6}$$

Theorem 4.1 (Node Degree under Controllable Structural Perturbation). *Let d_i^r denote the original degree of node v_i under relation $r \in \mathcal{R}$, and let $d_i^{r\,'}$ be its degree after the perturbation. Under controllable structural perturbation with deletion probability η_r^+ and addition probability η_r^-, the expected perturbed degree satisfies:*

$$\mathbb{E}[d_i^{r\,'}] = d_i^r (1 - \eta_r^+) + (|\mathcal{V}_r| - 1 - d_i^r)\, \eta_r^-. \tag{7}$$

Summing over all relation types yields the total expected degree:

$$\mathbb{E}[d_i'] = d_i + \sum_{r \in \mathcal{R}} \left(\eta_r^- (|\mathcal{V}_r| - 1 - d_i^r) - \eta_r^+ d_i^r \right), \tag{8}$$

where $d_i = \sum_{r \in \mathcal{R}} d_i^r$ is the original total degree of node v_i.

Proof. Fix a relation $r \in \mathcal{R}$. For node pair (i, j), the perturbed adjacency entry in Eq. (4) can be expressed as:

$$\widetilde{A}_{ij}^r = A_{ij}^r (1 - \Xi_{ij}^r) + (1 - A_{ij}^r)\, \Xi_{ij}^r, \tag{9}$$

where $\Xi_{ij}^r \sim \text{Bernoulli}(\eta_r^+)$ if $A_{ij}^r = 1$ (edge deletion) and $\Xi_{ij}^r \sim \text{Bernoulli}(\eta_r^-)$ if $A_{ij}^r = 0$ (edge addition). The perturbed degree of node v_i under relation r is then:

$$d_i^{r\,'} = \sum_{j \neq i} \widetilde{A}_{ij}^r. \tag{10}$$

Taking expectations and applying linearity of expectation gives:

$$\mathbb{E}[d_i^{r\,'}] = \sum_{j \neq i} \mathbb{E}[\widetilde{A}_{ij}^r] = \sum_{j \neq i} \begin{cases} 1 - \eta_r^+, & A_{ij}^r = 1, \\ \eta_r^-, & A_{ij}^r = 0. \end{cases} \tag{11}$$

Since $\left|\{j \neq i : A_{ij}^r = 1\}\right| = d_i^r$ and $\left|\{j \neq i : A_{ij}^r = 0\}\right| = |\mathcal{V}_r| - 1 - d_i^r$, we obtain

$$\mathbb{E}[d_i^{r\,'}] = d_i^r (1 - \eta_r^+) + (|\mathcal{V}_r| - 1 - d_i^r)\, \eta_r^-, \tag{12}$$

which is Eq. (7). Summing over all relations $r \in \mathcal{R}$ and noting $d_i = \sum_r d_i^r$ yields Eq. (8). □

4.2 Noise Analysis and Denoising Variational Inference

We investigate the impact of noise on different approaches for HGNNs. We categorize common HGNNs into two classes: meta-path-based and neighborhood aggregation-based methods. Meta-path-based methods typically require defining a meta-path ϕ, and then capturing information along different relations following the meta-path structure, aggregating this information, such as MAGNET [38] and Paths2Pair [9]. Neighborhood aggregation-based methods simultaneously consider the neighbor node types and edge types and use specific aggregation functions to combine information from different types. Common neighborhood aggregation methods include MHGCN [43] and Simple-HGN [23]. However, these two categories of methods differ in their efficiency of noise propagation. Previous work [46] analyzes the impact of adversarial attacks on message-passing performance. In this paper, we further extend the analysis to the setting of structural noise and theoretically characterize its influence capacity.

Definition 4.2 (Noise influence in HGNN). Given a heterogeneous graph $\mathcal{G}$ and a target node v_i, the influence of a neighbor node $v_j \in \mathcal{N}_i$ is defined as the structural contribution weight assigned to v_j during the node-level aggregation of v_i's embedding.

Theorem 4.3 (Noise Amplification Effect in HG). *In HG, compared to neighborhood aggregation-based methods, meta-path-based methods can significantly amplify the effect of noisy edges. Specifically, for a node v_i and a newly added noisy edge e_{ij}, the factor by which its influence changes is $\frac{d_{v_i}+k}{d_{v_i}+1}$, where k is the degree of the new neighbor v_j under the noise and d_{v_i} is the degree of v_i. When $k > d_{v_i}$, this factor is significantly greater than* 1.

The complete proof of Theorem 4.3 is provided in Appendix A. Based on Theorem 4.3, we employ a neighborhood aggregation method to encode heterogeneous graph and mitigate noise. Given noisy graph data $\tilde{\mathcal{G}}$, our objective is to obtain a denoised version of the standard graph data $\mathcal{G}$. VGAE [13] uses variational inference to derive statistical properties of the graph. The statistical data of latent variables in VGAE can be efficiently inferred from the latent space rather than the observation space, which provides robust graph information. For the standard graph $\mathcal{G}$, it initially generates latent variables $\mathbf{Z}$ from a prior distribution $p(\mathbf{Z})$, such as a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. Second, the data $\mathcal{G}$ is generated using a conditional distribution $p(\mathcal{G}|\mathbf{Z})$. VGAE optimizes its parameters by maximizing the likelihood of the observed data, which as follows:

$$\text{KL}(q_\Psi(\mathbf{Z}|\mathcal{G})||p_\theta(\mathbf{Z}|\mathcal{G})) + \mathcal{L}(\Psi, \theta; \mathcal{G}), \tag{13}$$

where Ψ is the encoder and θ represents the parameters to be optimized. Then, the evidence lower bound $\mathcal{L}(\Psi, \theta; \mathcal{G})$ can be expressed as follows:

$$\begin{aligned} \mathcal{L}(\Psi, \theta; \mathcal{G}) &= \mathbb{E}_{q_\Psi(\mathbf{Z}|\mathcal{G})} \left[\log \frac{p_\theta(\mathbf{Z}, \mathcal{G})}{q_\Psi(\mathbf{Z}|\mathcal{G})} \right] \\ &= \mathbb{E}_{q_\Psi(\mathbf{Z}|\mathcal{G})} \left[\log p_\theta(\mathbf{Z}|\mathcal{G}) \right] - \text{KL}\left(q_\Psi(\mathbf{Z}|\mathcal{G}) \| p(\mathbf{Z}) \right). \end{aligned} \tag{14}$$

Variational inference enhances the model's robustness and generalization capabilities [5, 11]. However, due to the differing distributions between noisy heterogeneous graph data and standard graph data, the obtained distribution tends to align with the noisy distribution, potentially misleading the GNN explainer into generating incorrect explanatory subgraphs. Therefore, we introduce a denoising module during the process of variational inference. The original encoder part is modified to:

$$q'_\Psi(\mathbf{Z}|\mathcal{G}) = \int q_\Psi(\mathbf{Z}|\tilde{\mathcal{G}}) q(\tilde{\mathcal{G}}|\mathcal{G}) \mathrm{d}\tilde{\mathcal{G}}, \tag{15}$$

where Ψ is the encoder based on $\tilde{\mathcal{G}}$, and $q(\tilde{\mathcal{G}}|\mathcal{G}) = \prod_{r \in \mathcal{R}} q(\tilde{\mathbf{A}}_r|\mathbf{A}_r)$. During this process, the evidence lower bound is expressed as:

$$\mathcal{L}_d = \mathbb{E}_{q'_\Psi(\mathbf{Z}|\mathcal{G})} [\log \frac{p_\theta(\mathbf{Z}, \mathcal{G})}{q'_\Psi(\mathbf{Z}|\mathcal{G})}]. \tag{16}$$

As we need to derive the distribution of the noisy graph data $\tilde{\mathcal{G}}$, this lower bound can be further refined as:

$$\mathcal{L}_d = \mathbb{E}_{q'_\Psi(\mathbf{Z}|\mathcal{G})}[\log \frac{p_\theta(\mathbf{Z},\mathcal{G})}{q'_\Psi(\mathbf{Z}|\mathcal{G})}] \geq \mathbb{E}_{q'_\Psi(\mathbf{Z}|\mathcal{G})}\left[\log \frac{p_\theta(\mathcal{G},\mathbf{Z})}{q_\Psi(\mathbf{Z}|\tilde{\mathcal{G}})}\right]$$
$$= \mathbb{E}_{q'_\Psi(\mathbf{Z}|\mathcal{G})}[\log p_\theta(\mathcal{G}|\mathbf{Z})] - \mathbb{E}_{q\left(\tilde{\mathcal{G}}|\mathcal{G}\right)}[\mathrm{KL}(q_\Psi(\mathbf{Z}|\tilde{\mathcal{G}})||p(\mathbf{Z})]. \tag{17}$$

The detailed derivation is in the Appendix B. Compared to VGAE, the denoising variational inference models the posterior probability $p(\mathbf{Z}|\mathcal{G})$ using a Gaussian Mixture Model, whereas VGAE models $p(\mathbf{Z}|\mathcal{G})$ using a Gaussian distribution. Additionally, during the optimization process, there is a constraint that forces $q_\Psi(\mathbf{Z}|\tilde{\mathcal{G}})$ to approximate the standard Gaussian distribution $p(\mathbf{Z})$. Consequently, our method can significantly improve the model's robustness and produce high-quality graph data. We further employ the Monte Carlo sampling method to approximate the objective, which can be effectively optimized using gradient descent as follows:

$$\mathcal{L}_d \approx \frac{1}{K}\sum_{k=1}^{K}\sum_{r\in\mathcal{R}} \log \frac{p_\theta(\mathcal{G}_r,\mathbf{Z})}{q_\Psi(\mathbf{Z}|\tilde{\mathcal{A}}_r)}, \tag{18}$$

where K is the number of samples sampled during the simulation. After denoising variational inference, we input the sampled robust representations $\mathbf{Z}$ into the heterogeneous explanation generator, where the complex semantics on the heterogeneous graph are learned. Before delving into that, we introduce the Graph Information Bottleneck.

4.3 Graph Information Bottleneck

As mentioned in the introduction, noise exacerbates the irregularity of graph structures and alters node importance. Therefore, previous methods imposing structural regularity constraints on explanatory subgraphs are infeasible under noise influence. We exploit the Graph Information Bottleneck (GIB) to enable the explainer network to adaptively handle structural irregularities. The objective of GIB is to obtain the optimal explanatory subgraph $\mathcal{G}_s$. From an information-theoretic perspective, GIB limits the amount of information carried by the explanatory subgraph $\mathcal{G}_s$, rather than imposing simple structural constraints. Simultaneously, nodes may require scattered edges across the graph to jointly explain their predictive function, rather than constraining connectedness. Consequently, GIB adaptively explores the entire graph without imposing any potentially biased restrictions. GIB can be formulated as:

$$\min_{\mathcal{G}_s\subset G} -\mathrm{I}(\hat{y};\mathcal{G}_s) + \beta\,\mathrm{I}(\mathcal{G};\mathcal{G}_s), \tag{19}$$

where $\mathrm{I}(\cdot;\cdot)$ denotes mutual information, and β controls the trade-off between the two terms. Since the information in $\mathcal{G}_s$ can be continually optimized, the explain task can be characterized as an optimization task guided by GIB.

The GIB principle aims to obtain the minimum sufficient information about the graph $\mathcal{G}$. The first term maximizes the mutual information between the label and the explanatory subgraph, ensuring $\mathcal{G}_s$ contains as much information about the label as possible. The second term minimizes the mutual information between the input graph and the explanatory subgraph, ensuring $\mathcal{G}_s$ contains the minimum information about the input graph. Next, we introduce the Heterogeneous Explanation Generator, describing how each term is optimized during training under the GIB principle.

4.4 Heterogeneous Explanation Generator

We begin by modeling the explanatory subgraph as a Gilbert random graph [8], where edges are conditionally independent. Following the literature [21], we define an adjacency matrix-like edge matrix E_s, where each element e_{ij} is a binary variable indicating whether the edge is included in the subgraph. When there is an edge (i,j) from v_i to v_j, $e_{ij}=1$, otherwise $e_{ij}=0$.

To capture the rich semantics of heterogeneous graphs, pairwise node interactions alone are insufficient. Thus, we incorporate heterogeneous semantics learning into the explanatory subgraph generation process. We incorporate heterogeneous edge type information into the attention computation by extending the standard graph attention mechanism. Specifically, we assign an edge type embedding $\mathbf{r}_{\varphi(e)}$ for each edge type $\varphi(e)$, and simultaneously utilize the edge type embeddings and node embeddings to compute the attention coefficient α_{ij}:

$$\alpha_{ij} = \frac{\exp\left(\mathrm{ReLU}\left(\boldsymbol{a}^T[\boldsymbol{W}\boldsymbol{z}_i\|\boldsymbol{W}\boldsymbol{z}_j\|\boldsymbol{W}_r\boldsymbol{r}_{\varphi(e_{ij})}]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\mathrm{ReLU}\left(\boldsymbol{a}^T[\boldsymbol{W}\boldsymbol{z}_i\|\boldsymbol{W}\boldsymbol{z}_k\|\boldsymbol{W}_r\boldsymbol{r}_{\varphi(e_{ik})}]\right)\right)}, \tag{20}$$

where $\boldsymbol{W}_r$ is a learnable weight matrix for type embeddings. Edge type embedding is a one-hot encoding derived from each edge type. This attention coefficient α_{ij} integrates both heterogeneous node and edge type semantics, offering a more comprehensive representation.

Next, we define the heterogeneous random graph variable. The probability of the heterogeneous explanatory subgraph can be factorized as:

$$p(\mathcal{G}) = \prod_{(i,j)\in E_s} p(e_{ij}|\varphi(e_{ij})), \tag{21}$$

where $e_{ij}\sim \mathrm{Bern}(\pi_{ij})$ and π_{ij} is the edge existence probability inferred via α_{ij}. To enable backpropagation through discrete edge selections, we adopt the reparameterization trick using a hard-concrete relaxation:

$$e_{ij} = \mathrm{Sigmoid}\Big(\frac{\log\epsilon - \log(1-\epsilon) + \alpha_{ij}(\varphi(e_{ij}))}{\tau}\Big), \quad \epsilon\sim\mathrm{Uniform}(0,1), \tag{22}$$

where τ is a temperature coefficient to smooth the optimization, and α_{ij} from Eq. 20 adds heterogeneous information into the explanatory subgraph. When $\alpha_{ij} = \log\frac{\pi_{ij}}{1-\pi_{ij}}$, we have $\lim_{\tau\to 0} p(e_{ij}=1) = \frac{\exp(\alpha_{ij})}{1+\exp(\alpha_{ij})}$, so we can obtain the explanatory subgraph $\mathcal{G}_s$ since $p(e_{ij}=1)=\pi_{ij}$.

This results in a continuous probability matrix $\mathbf{M_p}\in\mathbb{R}^{N\times N}$, where each entry $[\mathbf{M_p}]_{ij}=\pi_{ij}$ denotes the likelihood of including edge (i,j). We then construct the soft explanatory subgraph:

$$\mathcal{G}_s = (\mathbf{A_s} = \mathbf{M_p}\odot\mathbf{A}, \mathbf{X}, \mathcal{A}_s, \mathcal{R}_s). \tag{23}$$

To optimize the explainer, we adopt the Graph Information Bottleneck (GIB) principle, balancing predictive fidelity and information

Table 1: The comparison of RoHeX and baselines under different ratios of random structural noise. We use bold font to mark the best score. The second best score is marked with underline.

Dataset	Metric	Noise	PGExplainer	GNNExplainer	PGM-Explainer	V-InfoR	AMExplainer	Hete-PGE	xPath	RoHeX
DBLP	MAE	10%	1.2158±0.0062	0.8530±0.0009	1.0704±0.0007	1.1930±0.0030	1.4862±0.0026	0.8719±0.0008	0.8162±0.0032	**0.8359±0.0029**
		20%	1.2179±0.0089	0.9080±0.0007	1.2046±0.0006	1.1960±0.0025	1.5697±0.0008	0.8896±0.0005	0.9651±0.0042	**0.8743±0.0018**
		30%	1.2449±0.0059	1.2613±0.0011	1.3313±0.0002	1.2312±0.0026	1.7133±0.0006	1.1913±0.0006	1.1405±0.1032	**0.8827±0.0034**
		40%	1.2451±0.0068	1.3389±0.0008	1.3401±0.0005	1.2530±0.0010	1.9345±0.0005	0.9268±0.0006	1.2852±0.0002	**0.9014±0.0015**
	RMSE	10%	1.6775±0.0054	1.2968±0.0006	1.3855±0.0005	1.6481±0.0027	1.9219±0.0048	1.2814±0.0004	1.3175±0.0020	**1.2416±0.0017**
		20%	1.6815±0.0666	1.3072±0.0004	1.5280±0.0003	1.6511±0.0020	2.0419±0.0006	1.2859±0.0002	1.4341±0.0025	**1.2750±0.0013**
		30%	1.6999±0.0024	1.8470±0.0007	1.6497±0.0001	1.6781±0.0020	2.1842±0.0003	1.6532±0.0003	1.5752±0.1198	**1.2792±0.0022**
		40%	1.7060±0.0042	1.9043±0.0004	1.6681±0.0001	1.6885±0.0008	2.3616±0.0002	1.3100±0.0004	1.6819±0.0001	**1.2889±0.0008**
ACM	MAE	10%	0.7624±0.0080	0.3449±0.0003	0.2155±0.0009	0.7639±0.0001	0.3895±0.0005	0.8091±0.0003	0.3900±0.0001	**0.2129±0.0009**
		20%	0.7751±0.0162	0.3951±0.0001	0.3732±0.0003	0.7913±0.0004	0.6746±0.0224	0.8183±0.0005	0.3985±0.0003	**0.2483±0.0010**
		30%	0.7867±0.0152	0.5087±0.0003	0.5932±0.0006	0.8064±0.0003	0.7077±0.0221	0.8220±0.0002	0.4164±0.0012	**0.3140±0.0019**
		40%	0.7913±0.0181	0.6496±0.0002	0.7932±0.0007	0.8154±0.0005	0.7181±0.0164	0.8310±0.0001	0.4292±0.0006	**0.3163±0.0015**
	RMSE	10%	1.0258±0.0037	0.6831±0.0005	**0.5121±0.0004**	1.0145±0.0002	0.6241±0.0003	1.0740±0.0001	0.7750±0.0002	0.5662±0.0012
		20%	1.0307±0.0069	0.7791±0.0002	0.6893±0.0002	1.0366±0.0004	0.8213±0.0649	1.0778±0.0004	0.7458±0.0003	**0.6177±0.0008**
		30%	1.0423±0.0127	0.9506±0.0001	0.8793±0.0004	1.0705±0.0007	0.8412±0.0698	1.0731±0.0001	0.7768±0.0009	**0.6669±0.0013**
		40%	1.0431±0.0124	1.1306±0.0002	1.0766±0.0004	1.0786±0.0003	0.8474±0.0713	1.0856±0.0001	0.8396±0.0007	**0.6909±0.0008**
Freebase	MAE	10%	0.7189±0.0096	0.9012±0.0002	0.9190±0.0003	0.5957±0.0322	0.9312±0.0004	0.7760±0.0007	0.9006±0.0127	**0.3885±0.0010**
		20%	0.7237±0.0078	0.9108±0.0003	0.9401±0.0001	0.6822±0.0527	0.9709±0.1209	0.7812±0.0005	0.9247±0.0092	**0.4441±0.0012**
		30%	0.7285±0.0041	0.9126±0.0003	0.9530±0.0004	0.7249±0.0329	1.0089±0.1651	0.7908±0.0003	0.9263±0.0043	**0.4694±0.0021**
		40%	0.7370±0.0019	0.9378±0.0007	0.9587±0.0009	0.7894±0.0111	1.0531±0.0005	0.8030±0.0001	0.9301±0.0061	**0.4880±0.0014**
	RMSE	10%	1.0616±0.0071	1.2886±0.0001	1.2432±0.0001	1.0375±0.0233	1.2589±0.0004	1.1064±0.0003	1.2466±0.0056	**0.8251±0.0018**
		20%	1.0635±0.0051	1.2983±0.0002	1.2549±0.0001	1.1172±0.0566	1.2987±0.1848	1.1117±0.0004	1.2435±0.0108	**0.8854±0.0010**
		30%	1.0689±0.0039	1.2995±0.0002	1.2747±0.0002	1.1487±0.0277	1.3391±0.2381	1.1200±0.0002	1.2721±0.0034	**0.9035±0.0012**
		40%	1.0803±0.0018	1.3217±0.0005	1.2838±0.0004	1.1929±0.0054	1.3830±0.0004	1.1315±0.0001	1.2643±0.0072	**0.9217±0.0007**

compression. The GIB objective 19 is upper-bounded as:

$$-\mathrm{I}(\hat{y};\mathcal{G}_s)+\beta\,\mathrm{I}(\mathcal{G};\mathcal{G}_s) \le -\mathbb{E}_{p(\mathcal{G}_s,\hat{y})}\left[\log p_f(\hat{y}|\mathcal{G}_s)\right]+\mathrm{H}(\hat{y}) + \beta\mathbb{E}_{p(\mathcal{G})}\left[\mathrm{KL}(p_\alpha(\mathcal{G}_s|\mathcal{G})||q(\mathcal{G}_s))\right], \tag{24}$$

where f is the GNN model and α is the explain model, see Appendix B for detailed derivation. Since $\mathrm{H}(\hat{y})$ is constant, the objective function can be expressed as follows:

$$\mathcal{L}_{GIB} = -\mathbb{E}_{p(\mathcal{G}_s,\hat{y})}\left[\log p_f(\hat{y}|\mathcal{G}_s)\right] + \beta\mathbb{E}_{p(\mathcal{G})}\left[\mathrm{KL}(p_\alpha(\mathcal{G}_s|\mathcal{G})||q(\mathcal{G}_s))\right]. \tag{25}$$

The total loss combines the GIB objective with the denoising loss from the variational graph encoder:

$$\mathcal{L} = \mathcal{L}_d + \mathcal{L}_{GIB}. \tag{26}$$

4.5 Complexity Analysis.

The cost of each iteration comprises two parts: (1) the variational inference process and (2) the heterogeneous explanation generation. The time complexity of the first step is $O(N^2+E)$, and the space complexity is $O(N)$, as this step requires storing the robust node representations. The time complexity of the second step is $O(E)$, and the space complexity is $O(E)$. Therefore, the overall time complexity of RoHeX is $O(N^2+E)$, and the space complexity is $O(N+E)$.

5 Experiment

In this section, we evaluate the performance of the proposed RoHeX and state-of-the-art baselines on the node classification task. We then analyze the contributions of different components of RoHeX and demonstrate why RoHeX is robust to noise and capable of generating explanations that incorporate heterogeneous information.

5.1 Experiment Settings

Datasets and Baselines. We evaluate the effectiveness of our RoHeX on three real-world datasets, namely the academic citation datasets DBLP and ACM, and the knowledge graph dataset Freebase. Since there are no existing robust heterogeneous explainer, we select three types of baselines: the surrogate method PGM-Explainer, the perturbation-based methods GNNExplainer, PGExplainer and AMExplainer, and the V-infor method studying robustness on homogeneous graphs. We used the heterogeneous graph path explainer, xPath, and extended our heterogeneous explanation generator to PGExplainer, referred to as Hete-PGE, for comparison.

Evaluation. The performance of the explainer is evaluated based on the quality of the generated explanatory subgraphs, which are assessed in terms of their contribution to the original prediction. We employ two key metrics: fidelity and sparsity. Fidelity quantifies the reduction in prediction confidence when the explanatory subgraph is removed from the input graph, whereas sparsity measures the proportion of edges retained in the explanatory subgraph $\mathcal{G}_s$ relative to the original input graph. Specifically, we use the Mean Absolute Error (MAE, $\frac{1}{N}\sum_{i=1}^{N}\left|\hat{y}_i-\hat{y}_i^{\mathcal{G}_s}\right|$) and Root Mean Squared Error (RMSE, $\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i-\hat{y}_i^{\mathcal{G}_s}\right)^2}$) as proxy metrics for fidelity. We then compare the performance of various baseline methods under different sparsity levels, where N denotes the number of nodes or graphs, $\hat{y}_i$ is the original prediction, and $\hat{y}_i^{\mathcal{G}_s}$ represents the prediction derived from the explanatory subgraph.

Implementation Details. We conduct experiments under different proportions of random noise scenarios. To ensure randomness, the deletion rate and false addition rate are set to be equal. For the

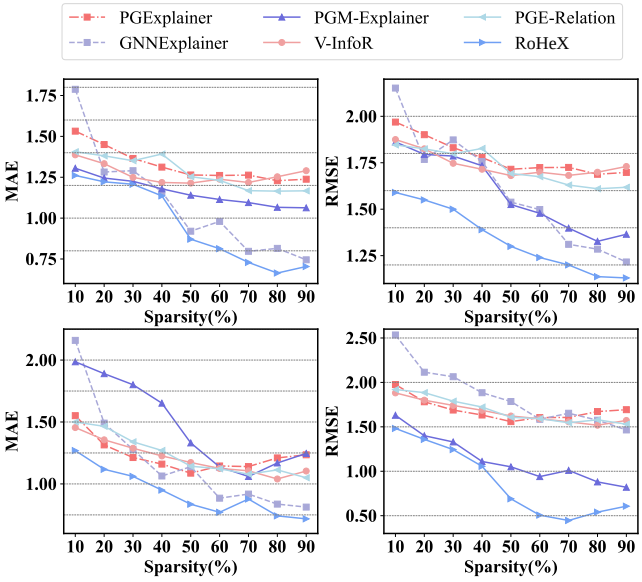


Figure 2: Fidelity-Sparsity Curve on the DBLP dataset. The first row is the result without noise, and the second row is the result with 20% noise budget.

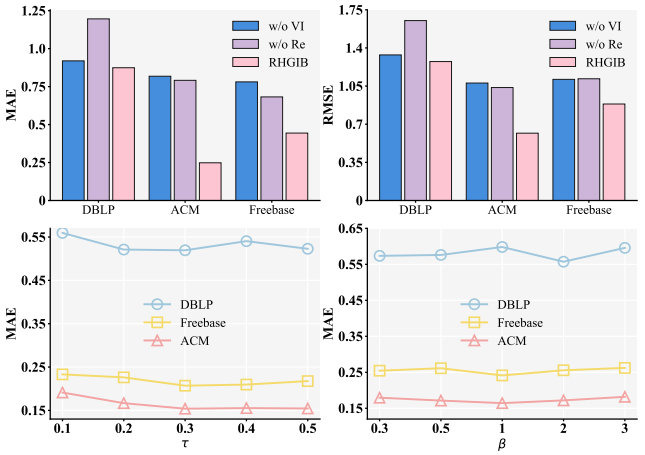


Figure 3: Ablation study on three datasets and the influence of hyperparameters τ and β on RoHeX.

baselines, we select the best-performing parameters for heterogeneous datasets based on the original settings. For fair comparison, we chose the most basic HGNN architecture as the base model, which only includes structural aggregation and relational learning modules. For our RoHeX, we use Adam as the optimizer with a learning rate of 1e-4. We set the hidden dimension for variational inference to 64, the output dimension to 32, and the edge weight output dimension to 32. Each experiment is repeated 5 times, and we report the mean and variance as the results. Descriptions of the datasets, baselines, base HGNN model architecture, and parameter settings are provided in the Appendix C.

5.2 Overall Performance under Structural Perturbations

Table 1 shows the experimental results on the heterogeneous graphs with different budgets of structural perturbations. We find that RoHeX outperforms other baselines in most experimental results, achieving the best performance on the DBLP and Freebase datasets. Taking 30% noise ratio as an example, RoHeX shows 25.9% lower MAE and 22.4% lower RMSE than the second-best method on the DBLP dataset, 38.2% lower MAE and 24.1% lower RMSE on the ACM dataset, and 35.2% lower MAE and 15.4% lower RMSE on the Freebase dataset. xPath performs excellently in multiple scenarios, indicating that designing corresponding modules for heterogeneous graphs is essential. We can observe that Hete-PGE achieves second best performance multiple times on the DBLP dataset and outperforms many baselines on other datasets, demonstrating the effectiveness of our proposed heterogeneous explanation generator in considering rich semantics on heterogeneous graphs. Due to the similar edge type distribution in the DBLP dataset, the dataset exhibits higher heterogeneity, which enhances the module's ability to capture heterogeneous information. Simultaneously, as a plug-and-play module, it can be conveniently extended to other parameterized explanation methods for generating explanations on heterogeneous graphs. On the medium and small-scale datasets DBLP and ACM, explanation methods based on raw features (e.g., GNNExplainer) are more susceptible to noise, potentially because raw features are more easily affected in smaller graphs. Since RoHeX generates robust graph representations, it can better mitigate the influence of noise, which is also why the latent representation-based explainer V-InfoR performs well in multiple scenarios. Under the guidance of GIB, our method can adaptively select important edges while excluding redundant and noisy edges, thereby generating the best explanations for the prediction model.

5.3 Fidelity-Sparsity Analysis

We further investigate RoHeX's performance at different sparsity levels. We provide the Fidelity-Sparsity curve on the DBLP dataset as shown in Figure 2. RoHeX consistently outperforms other baselines across all sparsity levels, indicating that our method can generate the best explanations. As the sparsity increases from 0 to 1, the overall trend of all curves is downward, i.e., decreasing error. When the sparsity is extremely low, e.g., 10%, our method significantly outperforms other baselines, suggesting that RoHeX can identify the truly critical subgraphs. We further find that although the overall performance improves as the sparsity level increases, there are still some cases where the performance drops with increasing sparsity, such as PGExplainer. We conjecture that this may be because in the subgraph generation process, when the sparsity increases to a point where all edges with high importance scores have been selected, forcing higher sparsity will begin to select unimportant edges, which can be viewed as noisy edges, leading to degraded performance. As the sparsity continues to increase, this adverse effect is offset. Since AMExplainer and xPath contain specific Sparsity settings to generate explanations, they were not included in the experiment.

Table 2: Prediction performance of HGNN in different noise scenarios.

Dataset	Method	10%		20%		30%		40%	
		Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
DBLP	Original	92.64	92.16	92.64	92.16	92.64	92.16	92.64	92.16
	Noisy	81.58	80.34	73.41	70.26	67.18	62.02	62.99	55.91
	1-hop	57.28	55.92	54.22	52.40	52.18	49.98	50.56	48.40
	2-hop	57.64	55.65	42.07	38.55	41.76	38.06	39.01	34.01
ACM	Original	92.32	92.40	92.32	92.40	92.32	92.40	92.32	92.40
	Noisy	33.52	19.28	33.28	18.82	31.96	16.14	31.96	16.14
	1-hop	77.10	77.31	76.39	76.51	75.40	75.57	67.94	67.81
	2-hop	52.19	51.73	49.12	50.65	42.38	36.42	38.73	30.07
Freebase	Original	68.99	63.57	68.99	63.57	68.99	63.57	68.99	63.57
	Noisy	67.89	61.90	66.05	59.29	62.94	53.74	57.50	46.02
	1-hop	58.52	53.23	57.62	51.94	57.50	51.21	57.01	51.14
	2-hop	44.66	31.12	42.90	27.24	41.10	24.15	39.46	21.62

5.4 Model Analysis

We investigate the contributions of different components in RoHeX. Specifically, we study (a) the effectiveness of the denoising variational inference module, and (b) the effectiveness of the relation-based importance module. We use 'w/o VI' to denote the model without the denoising variational inference module, and 'w/o Re' to denote the model without the relation-based importance module. For the latter case, we replace it with the common concatenation operation, i.e., $\alpha_{ij} = \text{MLP}[(\mathbf{z}_i, \mathbf{z}_j])$. The experiments are conducted under 20% noise budget, and the first row of Figure 3 shows the results after ablation. We find that without the denoising variational inference module, the model relies on the original features and graph structure for prediction, failing to mitigate the influence of noise, leading to performance degradation. When the model loses the ability to learn heterogeneous relationships, the process of generating explanation subgraphs struggles to recognize the complex semantics in heterogeneous graphs. All edges are treated as the same type, and the model explains solely based on node interactions. This demonstrates the necessity of our proposed relation importance module.

We further analyze the impact of two parameters τ and β on model performance. τ controls the approximation degree of e_{ij} distribution to the Bernoulli distribution, ranging within $[0.1, 0.5]$. β balances the information recovery strength and information filtering strength in the optimization objective, and we select values from $\{0.3, 0.5, 1, 2, 3\}$. The second row of Figure 3 shows the effects of these hyperparameters on RoHeX across three datasets. We can observe that the best results of τ all appear around 0.3. That is, when $\tau = 0.3$, the continuity and approximation degree in Eq. 22 reach the best trade-off. Secondly, RoHeX is not very sensitive to β that controls the constraint strength in Eq. 25, validating that our used GIB constraint can adapt to different data scenarios and achieve superior performance.

5.5 Effectiveness of Perturbation Strategy

We analyze the perturbation performance of controllable structural perturbation for heterogeneous graphs under different noise scenarios, as shown in Table 2. Compared to the original results, our structural perturbation method significantly impacts the decision-making process of HGNN on the whole graph (Noisy), 1-hop subgraph (1-hop), and 2-hop subgraph (2-hop). An interesting finding is that 30% noise is enough to cause the performance of HGNN on the ACM dataset to drop to its lowest. Moreover, we observe that the performance degradation becomes more pronounced as the perturbation expands from the global graph to larger neighborhood scopes. Specifically, both the 1-hop and 2-hop diffusion amplify the propagation of structural noise, leading to accumulated distortion of neighborhood information and further reducing classification accuracy. This phenomenon verifies our Theorem 4.3, which states that noise in HGNNs can greatly affect message passing and amplify its impact. It also highlights the necessity of our proposed denoising variational inference module to effectively mitigate such noise propagation and stabilize explain process.

6 Conclusion

In this work, we focus on the problem of explaining heterogeneous graph neural network under noise. We are the first to study this problem, theoretically proving that heterogeneous graph neural network have an amplifying effect on noise, and propose RoHeX to mitigate the influence of noise and obtain explanatory subgraphs based on heterogeneous relations. Specifically, RoHeX employs denoising variational inference to obtain robust graph representations and parameterizes the explanatory subgraph generation process with heterogeneous semantics. It integrates type information to capture the complexity of diverse node and edge types. Extensive experiments on real-world datasets demonstrate RoHeX's superiority over other state-of-the-art baselines. Future work will explore the impact of node feature–level noise on the performance and robustness of heterogeneous graph neural network explainers.

Acknowledgments

This work was supported in part by the Zhejiang Provincial Natural Science Foundation under Grant LMS25F030011, in part by the Zhejiang Province Key R&D Program Project under Grant No.2025C01023 and in part by the National Natural Science Foundation of China under Grant 62372146.

References

[1] Samy Benslimane, Jérôme Azé, Sandra Bringay, Maximilien Servajean, and Caroline Mollevi. 2023. A text and GNN based controversy detection method on social media. *World Wide Web* 26, 2 (2023), 799–825.

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1247–1250.

[3] Yuwei Cao, Hao Peng, Zhengtao Yu, and S Yu Philip. 2024. Hierarchical and incremental structural entropy minimization for unsupervised social event detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8255–8264.

[4] Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. 2022. Towards robust graph neural networks for noisy graphs with sparse labels. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 181–191.

[5] Haoyi Fan, Fengbin Zhang, Yuxuan Wei, Zuoyong Li, Changqing Zou, Yue Gao, and Qionghai Dai. 2021. Heterogeneous hypergraph variational autoencoder for link prediction. *IEEE transactions on pattern analysis and machine intelligence* 44, 8 (2021), 4125–4138.

[6] James Fox and Sivasankaran Rajamanickam. 2019. How robust are graph neural networks to structural noise? *arXiv preprint arXiv:1912.10206* (2019).

[7] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of the web conference 2020*. 2331–2341.

[8] Edgar N Gilbert. 1959. Random graphs. *The Annals of Mathematical Statistics* 30, 4 (1959), 1141–1144.

[9] Jinquan Hang, Zhiqing Hong, Xinyue Feng, Guang Wang, Guang Yang, Feng Li, Xining Song, and Desheng Zhang. 2024. Paths2Pair: Meta-path Based Link Prediction in Billion-Scale Commercial Heterogeneous Graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5082–5092.

[10] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. 2022. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering* 35, 7 (2022), 6968–6972.

[11] Daniel Im Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. 2017. Denoising criterion for variational auto-encoding framework. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[14] Tong Li, Jiale Deng, Yanyan Shen, Luyu Qiu, Huang Yongxiang, and Caleb Chen Cao. 2023. Towards fine-grained explainability for heterogeneous graph neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 8640–8647.

[15] Yuan Li, Xiuxu Chen, Yuhai Zhao, Wen Shan, Zhengkui Wang, Guoli Yang, and Guoren Wang. 2024. Self-training gnn-based community search in large attributed heterogeneous information networks. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2765–2778.

[16] Huan Liu, Pengfei Jiao, Mengzhou Gao, Chaochao Chen, and Di Jin. 2025. Heterogeneous Temporal Hypergraph Neural Network. *arXiv preprint arXiv:2506.17312* (2025).

[17] Yonghao Liu, Lan Huang, Bowen Cao, Ximing Li, Fausto Giunchiglia, Xiaoyue Feng, and Renchu Guan. 2024. A simple but effective approach for unsupervised few-shot graph classification. In *Proceedings of the ACM on Web Conference 2024*. 4249–4259.

[18] Shengyao Lu, Keith G Mills, Jiao He, Bang Liu, and Di Niu. 2024. GOAt: Explaining Graph Neural Networks via Graph Output Attribution. In *The Twelfth International Conference on Learning Representations*.

[19] Yifan Lu, Mengzhou Gao, Huan Liu, Zehao Liu, Wei Yu, Xiaoming Li, and Pengfei Jiao. 2023. Neighborhood overlap-aware heterogeneous hypergraph neural network for link prediction. *Pattern Recognition* 144 (2023), 109818.

[20] Yifan Lu, Ziyun Zou, Belal Alsinglawi, Islam Al-Qudah, Izzat Alsmadi, Feilong Tang, Pengfei Jiao, Shoaib Jameel, and Imran Razzak. 2025. HeSRN: Representation Learning On Heterogeneous Graphs via Slot-Aware Retentive Network. *arXiv preprint arXiv:2510.09767* (2025).

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

[22] Yuankai Luo, Lei Shi, and Xiao-Ming Wu. 2024. Classic gnns are strong baselines: Reassessing gnns for node classification. *Advances in Neural Information Processing Systems* 37 (2024), 97650–97669.

[23] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 1150–1160.

[24] Jiaqi Ma, Shuangrui Ding, and Qiaozhu Mei. 2020. Towards more practical adversarial attacks on graph neural networks. *Advances in neural information processing systems* 33 (2020), 4756–4766.

[25] Ke Ma, Yizhou Fang, Jean-Baptiste Weibel, Shuai Tan, Xinggang Wang, Yang Xiao, Yi Fang, and Tian Xia. 2025. Phys-Liquid: A Physics-Informed Dataset for Estimating 3D Geometry and Volume of Transparent Deformable Liquids. *arXiv preprint arXiv:2511.11077* (2025).

[26] Yuren Mao, Yu Hao, Xin Cao, Yunjun Gao, Chang Yao, and Xuemin Lin. 2025. Boosting GNN-Based Link Prediction via PU-AUC Optimization. *IEEE Transactions on Knowledge and Data Engineering* (2025).

[27] Yujie Mo, Feiping Nie, Ping Hu, Heng Tao Shen, Zheng Zhang, Xinchao Wang, and Xiaofeng Zhu. 2024. Self-supervised heterogeneous graph learning: a homophily and heterogeneity view. In *The Twelfth International Conference on Learning Representations*.

[28] Peter Müller, Lukas Faber, Karolis Martinkus, and Roger Wattenhofer. 2024. GraphChef: Decision-Tree Recipes to Explain Graph Neural Networks. In *The Twelfth International Conference on Learning Representations*.

[29] Tamara Pereira, Erik Nascimento, Lucas E Resck, Diego Mesquita, and Amauri Souza. 2023. Distill n'Explain: explaining graph neural networks using simple surrogates. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 6199–6214.

[30] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*. Springer, 593–607.

[31] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. 2020. Interpreting Graph Neural Networks for NLP With Differentiable Edge Masking. In *International Conference on Learning Representations*.

[32] Sangwoo Seo, Sungwon Kim, and Chanyoung Park. 2024. Interpretable prototype-based graph information bottleneck. *Advances in Neural Information Processing Systems* 36 (2024).

[33] Minh Vu and My T Thai. 2020. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *Advances in neural information processing systems* 33 (2020), 12225–12235.

[34] Minh N Vu and My T Thai. 2023. Limitations of Perturbation-based Explanation Methods for Temporal Graph Neural Networks. In *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE, 618–627.

[35] Senzhang Wang, Jun Yin, Chaozhuo Li, Xing Xie, and Jianxin Wang. 2024. V-InFoR: a robust graph neural networks explainer for structurally corrupted graphs. *Advances in Neural Information Processing Systems* 36 (2024).

[36] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and S Yu Philip. 2022. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data* 9, 2 (2022), 415–436.

[37] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The world wide web conference*. 2022–2032.

[38] Xin-Cheng Wen, Cuiyun Gao, Jiaxin Ye, Yichen Li, Zhihong Tian, Yan Jia, and Xuan Wang. 2023. Meta-path based attentional graph learning model for vulnerability detection. *IEEE Transactions on Software Engineering* (2023).

[39] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE Transactions on Knowledge and Data Engineering* 34, 10 (2020), 4854–4873.

[40] Xiaocheng Yang, Mingyu Yan, Shirui Pan, Xiaochun Ye, and Dongrui Fan. 2023. Simple and efficient heterogeneous graph neural network. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 10816–10824.

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

[42] Lingfan Yu, Jiajun Shen, Jinyang Li, and Adam Lerer. 2020. Scalable graph neural networks for heterogeneous graphs. *arXiv preprint arXiv:2011.09679* (2020).

[43] Pengyang Yu, Chaofan Fu, Yanwei Yu, Chao Huang, Zhongying Zhao, and Junyu Dong. 2022. Multiplex heterogeneous graph convolutional network. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2377–2387.

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

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

[46] Mengmei Zhang, Xiao Wang, Meiqi Zhu, Chuan Shi, Zhiqiang Zhang, and Jun Zhou. 2022. Robust heterogeneous graph neural networks against adversarial attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 4363–4370.

[47] Wei Zhang, Xiaofan Li, and Wolfgang Nejdl. 2024. Adversarial Mask Explainer for Graph Neural Networks. In *Proceedings of the ACM on Web Conference 2024*.

861–869.
[48] Yi Zhang, Yiwen Zhang, Yuchuan Zhao, Shuiguang Deng, and Yun Yang. 2024. Dual Variational Graph Reconstruction Learning for Social Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2024).
[49] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Cheekong Lee. 2022. Protgnn: Towards self-explaining graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 9127–9135.
[50] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1399–1407.
[51] Ziyun Zou, Yinghui Jiang, Lian Shen, Juan Liu, and Xiangrong Liu. 2025. Loha: Direct graph spectral contrastive learning between low-pass and high-pass views. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 13492–13500.
[52] Ziyun Zou, Lian Shen, Yanhao Li, Yifan Lu, Juan Liu, and Xiangrong Liu. 2025. RETAIN: Reliable Topology Augmentation for both Heterophilic and Homophilic Graphs. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1–5.
[53] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2847–2856.

A Proof of Theorem 4.3

In Graph Neural Network, a node representation is typically updated by aggregating information from its neighboring nodes. This process can be described as a message passing mechanism, where each node receives messages from neighboring nodes and updates its representation based on these messages. To avoid cases where the influence is overly amplified during the aggregation process, the messages from neighboring nodes are typically normalized. A common normalization approach is to multiply each neighbor message by the inverse of its degree. Assuming that each node influence on neighbors is equal, a higher-degree node will distribute its influence evenly among all neighbors. Therefore, the influence received by each neighbor should be proportional to the inverse of the node degree. In contrast, in random walk models, the transition probability between nodes is inversely proportional to the node degree. That is, the probability of a node reaching a particular neighbor is the inverse of its degree.

Given a heterogeneous graph $\mathcal{G}$, let v_i be a node with degree d_{v_i}. A noisy edge e_{ij} is added to the graph, where v_j is a new neighbor with degree d_{v_j} and k specific-type neighbors that match a given meta-path ϕ.

For meta-path-based methods:

(a) Before adding the noisy edge, the influence of v_i is assumed to be a combination of the influences from its d_{v_i} existing neighbors $v_1, v_2, ..., v_{d_{p_i}}$ in $\mathcal{G}$. The influence of each neighbor v_n on v_i can be represented as:

$$I_{\text{ori1}} = \sum_{n=1}^{d_{v_i}} \frac{1}{d_{v_i}} \tag{27}$$

(b) After adding the noisy edge (v_i, v_j), v_i is directly connected to v_j, and the influence of v_j will propagate to its k neighbors. The influence on each neighbor of v_j changes in the following manner v_i:

$$I_{\text{new1}} = \sum_{i=1}^{d_{v_i}} \frac{1}{d_{v_i}+1} + \frac{k}{d_{v_i}+1} = \frac{d_{v_i}}{d_{v_i}+1} + \frac{k}{d_{v_i}+1} = \frac{d_{v_i}+k}{d_{v_i}+1} \tag{28}$$

For neighborhood aggregation-based methods:

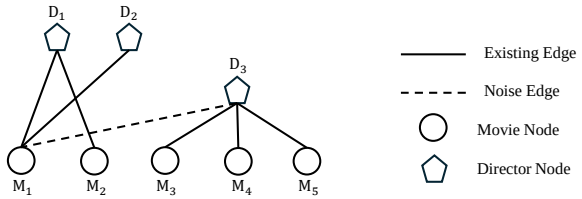


Figure 4: The illustrative example for noise against HGNNs on Movie-Director graph.

(a) Before adding the noisy edge, the influence on the direct neighbors of v_i is given by:

$$I_{\text{ori2}} = \sum_{n=1}^{d_{v_i}} \frac{1}{d_{v_i}} \tag{29}$$

(b) After adding the noisy edge (v_i, v_j), the neighbors of v_i increase to d_{v1}+1, and the influence on each of its neighbors changes to:

$$I_{\text{new2}} = \sum_{n=1}^{d_{v_i}+1} \frac{1}{d_{v_i}+1} \tag{30}$$

The multiplicative relationship ξ of the influence propagation between the meta-path-based method and the neighborhood aggregation method is:

$$\xi = \frac{\frac{I_{\text{new1}}}{I_{\text{ori1}}}}{\frac{I_{\text{new2}}}{I_{\text{ori2}}}} = \frac{\frac{d_{v_i}+k}{d_{v_i}+1}}{1} = \frac{d_{v_i}+k}{d_{v_i}+1} \tag{31}$$

Consequently, when $k > d_{v_i}$, the multiplicative factor ξ is significantly greater than 1. This indicates that in general heterogeneous graphs, meta-path-based approaches are far more susceptible to the influence of noisy edges compared to neighborhood aggregation-based approaches. This substantiates that meta-path-based methods can significantly amplify the effect of noisy edges to a greater extent than neighborhood aggregation methods.

We also provide the illustrative example for noise against HGNNs on Movie-Director graph in Figure 4. The meta-path used is M-D-M. In neighborhood aggregation-based HGNNs, the noise-introduced nodes are not directly considered as 1-hop neighbors of M_1. Consequently, under the influence of noise, the 2-hop neighbors M_3, M_4, M_5 can only affect M_1 through the 1-hop neighbor D_3. However, in meta-path-based HGNNs, all neighbors under the M-D-M meta-path are aggregated with equal weight $\frac{1}{5}$, thereby enlarging the effect of the noisy edge $\langle M_1, D_3 \rangle$ to $\frac{3}{5}$ (the total weight of the noisy neighbors M_3, M_4, M_5).

B Detailed Derivation

First, we give the detailed derivation of Eq. 14. We introduce the Kullback-Leibler (KL) divergence. The KL divergence is a measure used to quantify the difference between two probability distributions. Let us consider two continuous random variables with probability distributions P and Q, and their corresponding probability density functions denoted as $p(x)$ and $q(x)$, respectively. If we aim

to approximate $p(x)$ using $q(x)$, the KL divergence can be expressed as:

$$\mathrm{KL}(P||Q) = \int p(x) \log \frac{p(x)}{q(x)} dx. \tag{32}$$

Because the logarithmic function is convex, the value of KL divergence is nonnegative. Then, Eq. 14 can be written as:

$$\begin{aligned}\mathcal{L}(\Psi, \theta; \mathcal{G}) &= \mathbb{E}_{q_\Psi(\mathbf{Z}|\mathcal{G})}[\log \frac{p_\theta(\mathbf{Z}, \mathcal{G})}{q_\Psi(\mathbf{Z}|\mathcal{G})}] \\ &= \mathbb{E}_{q_\Psi(\mathbf{Z}|\mathcal{G})}[\log p_\theta(\mathbf{Z}|\mathcal{G}) \cdot \frac{p(\mathbf{Z})}{q_\Psi(\mathbf{Z}|\mathcal{G})}] \\ &= \mathbb{E}_{q_\Psi(\mathbf{Z}|\mathcal{G})}[\log p_\theta(\mathbf{Z}|\mathcal{G})] - \mathrm{KL}(q_\Psi(\mathbf{Z}|\mathcal{G})||p(\mathbf{Z})).\end{aligned} \tag{33}$$

Second, the lower bound of denoising variational inference in Eq. 17 can be derived as:

$$\begin{aligned}\mathcal{L}_d &= \mathbb{E}_{q'_\Psi(\mathbf{Z}|\mathcal{G})}[\log \frac{p_\theta(\mathbf{Z}, \mathcal{G})}{q'_\Psi(\mathbf{Z}|\mathcal{G})}] \geq \mathbb{E}_{q'_\Psi(\mathbf{Z}|\mathcal{G})}\left[\log \frac{p_\theta(\mathcal{G}, \mathbf{Z})}{q_\Psi(\mathbf{Z}|\tilde{\mathcal{G}})}\right] \\ &= \mathbb{E}_{q'_\Psi(\mathbf{Z}|\mathcal{G})}[\log p_\theta(\mathcal{G}|\mathbf{Z}) + \log p(\mathbf{Z}) - \log q_\Psi(\mathbf{Z}|\tilde{\mathcal{G}})] \\ &= \mathbb{E}_{q'_\Psi(\mathbf{Z}|\mathcal{G})}[\log p_\theta(\mathcal{G}|\mathbf{Z})] - \mathbb{E}_{q'_\Psi(\mathbf{Z}|\mathcal{G})}\left[\log \frac{q_\Psi(\mathbf{Z}|\tilde{\mathcal{G}})}{p(\mathbf{Z})}\right] \\ &= \mathbb{E}_{q'_\Psi(\mathbf{Z}|\mathcal{G})}[\log p_\theta(\mathcal{G}|\mathbf{Z})] - \mathbb{E}_{q(\tilde{\mathcal{G}}|\mathcal{G})}\mathbb{E}_{q_\Psi(\mathbf{Z}|\mathcal{G})}\left[\log \frac{q_\Psi(\mathbf{Z}|\tilde{\mathcal{G}})}{p(\mathbf{Z})}\right] \\ &= \mathbb{E}_{q'_\Psi(\mathbf{Z}|\mathcal{G})}[\log p_\theta(\mathcal{G}|\mathbf{Z})] - \mathbb{E}_{q\left(\tilde{\mathcal{G}}|\mathcal{G}\right)}[\mathrm{KL}(q_\Psi(\mathbf{Z}|\tilde{\mathcal{G}}))||p(\mathbf{Z})].\end{aligned} \tag{34}$$

Third, we derive an upper bound for GIB in Eq. 24. We decompose the mutual information:

$$\mathrm{I}(\hat{y}; \mathcal{G}_s) = \mathbb{E}_{p(\hat{y}, \mathcal{G}_s)}\left[\log \frac{p(\hat{y}, \mathcal{G}_s)}{p(\hat{y})p(\mathcal{G}_s)}\right], \tag{35}$$

$$\mathrm{I}(\mathcal{G}; \mathcal{G}_s) = \mathbb{E}_{p(\mathcal{G}, \mathcal{G}_s)}\left[\log \frac{p(\mathcal{G}, \mathcal{G}_s)}{p(\mathcal{G})p(\mathcal{G}_s)}\right]. \tag{36}$$

The GIB objective can be written as:

$$\begin{aligned}&-\mathrm{I}(\hat{y}; \mathcal{G}_s) + \beta \mathrm{I}(\mathcal{G}; \mathcal{G}_s) \\ &= -\mathbb{E}_{p(\hat{y}, \mathcal{G}_s)}\left[\log \frac{p(\hat{y}, \mathcal{G}_s)}{p(\hat{y})p(\mathcal{G}_s)}\right] + \beta \mathbb{E}_{p(\mathcal{G}, \mathcal{G}_s)}\left[\log \frac{p(\mathcal{G}, \mathcal{G}_s)}{p(\mathcal{G})p(\mathcal{G}_s)}\right] \\ &= -\mathbb{E}_{p(\hat{y}, \mathcal{G}_s)}\left[\log \frac{p(\hat{y}|\mathcal{G}_s)p(\mathcal{G}_s)}{p(\hat{y})p(\mathcal{G}_s)}\right] + \beta \mathbb{E}_{p(\mathcal{G}, \mathcal{G}_s)}\left[\log \frac{p(\mathcal{G}_s|\mathcal{G})p(\mathcal{G})}{p(\mathcal{G})p(\mathcal{G}_s)}\right] \\ &= -\mathbb{E}_{p(\hat{y}, \mathcal{G}_s)}\left[\log \frac{p(\hat{y}|\mathcal{G}_s)}{p(\hat{y})}\right] + \beta \mathbb{E}_{p(\mathcal{G}, \mathcal{G}_s)}\left[\log \frac{p(\mathcal{G}_s|\mathcal{G})}{p(\mathcal{G}_s)}\right] \\ &= -\mathbb{E}_{p(\mathcal{G}_s)}\mathbb{E}_{p(\hat{y}|\mathcal{G}_s)}\left[\log \frac{p(\hat{y}|\mathcal{G}_s)}{p(\hat{y})}\right] + \beta \mathbb{E}_{p(\mathcal{G})}\mathbb{E}_{p(\mathcal{G}_s|\mathcal{G})}\left[\log \frac{p(\mathcal{G}_s|\mathcal{G})}{p(\mathcal{G}_s)}\right].\end{aligned} \tag{37}$$

Using Jensen's inequality and assuming that $p_f(\hat{y}|\mathcal{G}_s)$ is an approximation of $p(\hat{y}|\mathcal{G}_s)$, we can get:

$$\begin{aligned}&-\mathbb{E}_{p(\mathcal{G}_s)}\mathbb{E}_{p(\hat{y}|\mathcal{G}_s)}\left[\log \frac{p(\hat{y}|\mathcal{G}_s)}{p(\hat{y})}\right] \\ &\leq -\mathbb{E}_{p(\mathcal{G}_s)}\mathbb{E}_{p(\hat{y}|\mathcal{G}_s)}[\log p_f(\hat{y}|\mathcal{G}_s)] - \mathbb{E}_{p(\hat{y})}[\log p(\hat{y})] \\ &= -\mathbb{E}_{p(\mathcal{G}_s, \hat{y})}\left[\log p_f(\hat{y}|\mathcal{G}_s)\right] + \mathrm{H}(\hat{y}).\end{aligned} \tag{38}$$

We introduce explain models:

$$\begin{aligned}&\beta \mathbb{E}_{p(\mathcal{G})}\mathbb{E}_{p(\mathcal{G}_s|\mathcal{G})}\left[\log \frac{p(\mathcal{G}_s|\mathcal{G})}{p(\mathcal{G}_s)}\right] \\ &= \mathbb{E}_{p(\mathcal{G})}\mathbb{E}_{p(\mathcal{G}_s|\mathcal{G})}\left[\log \frac{p_\alpha(\mathcal{G}_s|\mathcal{G})}{p(\mathcal{G}_s)} \cdot \frac{p(\mathcal{G}_s|\mathcal{G})}{p_\alpha(\mathcal{G}_s|\mathcal{G})}\right] \\ &= \mathbb{E}_{p(\mathcal{G})}\mathbb{E}_{p(\mathcal{G}_s|\mathcal{G})}\left[\log \frac{p_\alpha(\mathcal{G}_s|\mathcal{G})}{p(\mathcal{G}_s)}\right] + \\ &\quad \mathbb{E}_{p(\mathcal{G})}\mathbb{E}_{p(\mathcal{G}_s|\mathcal{G})}\left[\log \frac{p(\mathcal{G}_s|\mathcal{G})}{p_\alpha(\mathcal{G}_s|\mathcal{G})}\right].\end{aligned} \tag{39}$$

The second term is the KL divergence:

$$\begin{aligned}&\mathbb{E}_{p(\mathcal{G})}\mathbb{E}_{p(\mathcal{G}_s|\mathcal{G})}\left[\log \frac{p(\mathcal{G}_s|\mathcal{G})}{p_\alpha(\mathcal{G}_s|\mathcal{G})}\right] \\ &= \mathbb{E}_{p(\mathcal{G})}\left[\mathrm{KL}(p(\mathcal{G}_s|\mathcal{G})\|p_\alpha(\mathcal{G}_s|\mathcal{G}))\right] \geq 0.\end{aligned} \tag{40}$$

Therefore,

$$\begin{aligned}\mathrm{I}(\mathcal{G}; \mathcal{G}_s) &\leq \mathbb{E}_{p(\mathcal{G})}\mathbb{E}_{p_\alpha(\mathcal{G}_s|\mathcal{G})}\left[\log \frac{p_\alpha(\mathcal{G}_s|\mathcal{G})}{q(\mathcal{G}_s)}\right] \\ &= \mathbb{E}_{p(\mathcal{G})}\left[\mathrm{KL}(p_\alpha(\mathcal{G}_s|\mathcal{G})\|q(\mathcal{G}_s))\right].\end{aligned} \tag{41}$$

Combined with our previous derivation of the first term, we can get:

$$\begin{aligned}-\mathrm{I}(\hat{y}; \mathcal{G}_s) + \beta \mathrm{I}(\mathcal{G}; \mathcal{G}_s) \leq -\mathbb{E}_{p(\mathcal{G}_s, \hat{y})}\left[\log p_f(\hat{y}|\mathcal{G}_s)\right] + \mathrm{H}(\hat{y}) + \\ \beta \mathbb{E}_{p(\mathcal{G})}\left[\mathrm{KL}(p_\alpha(\mathcal{G}_s|\mathcal{G})||q(\mathcal{G}_s))\right].\end{aligned} \tag{42}$$

C Experiment Supplement

Table 3: Statistics of Datasets.

Dataset	DBLP	ACM	Freebase
Nodes	26,128	10,942	43,854
Edges	239,566	547,872	151,034
Node Types	4	4	4
Edge Types	6	8	6
Classes	4	3	3

C.1 Datasets

We conduct experiments on three real-world datasets. According to the Heterogeneous Graph Benchmark [23] settings, we randomly split the nodes with proportions of 24%, 6%, and 70% for training, validation, and testing, respectively. The statistics of the three datasets are shown in Table 3.

- **DBLP**[1] is a computer science bibliography network that contains four types of nodes: Paper (P), Author (A), Term (T), and Venue (V). The authors in this network are from four research areas (*Database, Data Mining, Artificial Intelligence,* and *Information Retrieval*).
- **ACM**[2] is a citation network that contains four types of nodes: Paper (P), Author (A), Term (T), and Subject (S). The papers in this network are divided into three classes (*Database, Wireless Communication,* and *Data Mining*).

[1] https://dblp.uni-trier.de
[2] http://dl.acm.org

- **Freebase** [2] is a knowledge graph that contains four types of nodes: Movie (M), Actor (A), Director (D) and Writer (W).

C.2 Baselines

Next, we provide details on the baselines used in our experiments.

- **PGExpaliner** [21] is a parameterized explainer that learns a mask for each edge to obtain edge importance scores.
- **GNNExplainer** [41] maximizes the mutual information between the model's prediction on the original input and the masked input by masking features and edges.
- **PGM-Explainer** [33] employs a Bayesian network-based approach, treating vertices in the input graph as random variables to fit the GNN model's predicted label.
- **V-InfoR** [35] utilizes a parametric method, learning edge masks on the latent representations to identify important edges.
- **AMExplainer** [47] optimizes GNN explanations by leveraging adversarial networks to achieve both sparsity and prediction accuracy, ensuring compact and faithful explanation sets.
- **Hete-PGE** is an extension of PGExplainer, where we replace the initial concatenation with a relation-based attention learning module to enable learning of heterogeneous semantics.
- **xPath** [14] provides fine-grained explanations by identifying causal nodes and their influence paths through a novel graph rewiring algorithm, offering detailed insights into the model's decision-making process.

C.3 Base Heterogeneous Graph Neural Network Architecture

Table 4: Node classification result using our heterogeneous Graph Neural Network.

Dataset	DBLP	ACM	Freebase
Micro-F1	92.64±0.14	92.32±0.12	68.99±0.20
Macro-F1	92.16±0.19	92.40±0.11	63.57±0.36

In the experiment, we use a basic heterogeneous Graph Neural Network, which encodes the input graph through 2 layers of GCN, and then used a layer of attention learning module to learn different heterogeneous relations. For a heterogeneous graph, the feature spaces of different types of nodes are usually different. We use a mapping function to map the features of different types into a common feature space, as shown below:

$$\mathbf{z}_v = \mathbf{W}_m \mathbf{x}_v^A + \mathbf{b}_m, \tag{43}$$

where $A \in \mathcal{A}$ is the node type of node v, $\mathbf{W}_m$ is a learnable weight, and $\mathbf{b}_m$ is the bias. Then, in the shared space, we use GCN to obtain the node embeddings:

$$\mathbf{Z}^{(l)} = \mathrm{GCN}(\mathbf{Z}^{(l-1)}, \mathbf{A}), \mathbf{Z}^{(0)} = \mathbf{Z}_v. \tag{44}$$

To learn the heterogeneous semantics of the heterogeneous graph, we introduce a type vector $\boldsymbol{\gamma}_v$ and learn relation information through an attention module:

$$\begin{gathered} \boldsymbol{\gamma}_i^q = \mathbf{W}_r^q \boldsymbol{\gamma}_i, \boldsymbol{\gamma}_j^k = \mathbf{W}_r^k \boldsymbol{\gamma}_j, \\ score_{ij}^{\boldsymbol{\gamma}} = \boldsymbol{\gamma}_i^q \boldsymbol{\gamma}_j^k, \end{gathered} \tag{45}$$

where $\mathbf{W}_r^q$ and $\mathbf{W}_r^k$ are learnable weights. The attention of the nodes can be computed as follows:

$$\begin{gathered} q_i = \mathbf{W}_q^z \mathbf{z}_i, k_j = \mathbf{W}_k^z \mathbf{z}_j, \\ \widehat{\alpha}_{ij} = \frac{\exp(\mathrm{LeakyReLU}(a^T [q_i \parallel k_j]))}{\sum_{j' \in \mathcal{N}_i} \exp(\mathrm{LeakyReLU}(a^T [q_i \parallel k_{j'}]))}. \end{gathered} \tag{46}$$

where $\mathbf{W}_q^z$ and $\mathbf{W}_k^z$ are learnable weights. The final prediction can be expressed as:

$$\begin{gathered} score_{ij} = \widehat{\alpha}_{ij} + \beta score_{ij}^{\boldsymbol{\gamma}}, \\ \mathbf{Z}_{\mathbf{H}}^{(l)} = \mathrm{LayerNorm}(\mathbf{Z}_{\mathbf{H}}^{(l-1)} + score_{ij} \cdot \mathbf{Z}_{\mathbf{H}}^{(l-1)}), \\ \hat{Y} = P_f(\mathbf{Z}_{\mathbf{H}}^{(l)}; \theta_p). \end{gathered} \tag{47}$$

where θ_p is the parameter of the predictor. The basic prediction results are shown in Table 4.

The experiments are conducted on an L20 GPU with 48GB of memory. Our CPU is an Intel(R) Xeon(R) Platinum 8457C. We utilized PyTorch version 1.13.1 and DGL version 1.1.1.

C.4 Parameter Setting

For the base heterogeneous Graph Neural Network, we use Adam [12] as the optimizer, LeakyReLU with a negative slope $s = 0.2$ as the activation function, a learning rate of 1e-4, and a dropout rate of 0 for Freebase and 0.5 for other datasets. The hidden dimension is set to 256. Our training is performed for 100 epochs.