Generating In-Distribution Counterfactual Explanation for Graph Neural Networks

Linmao Chen[1], Chaobo He[1]*, Junwei Cheng[1], Chunying Li[2], Quanlong Guan[3]

[1]School of Computer Science, South China Normal University
[2]School of Computer Science, Guangdong Polytechnic Normal University
[3]Department of Computer Science, Jinan University
{chenlinmao, hechaobo, jung}@m.scnu.edu.cn,
ChunyingL@gpnu.edu.cn, gql@jnu.edu.cn

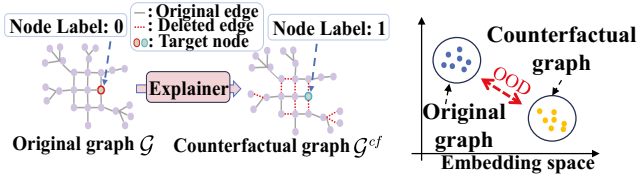
Abstract

Graph Neural Networks (GNNs) have received increasing attention due to their ability to handle graph-structured data, yet their explainability remains a significant challenge. An effective solution is to provide the GNN models with counterfactual explanations, which aim to answer "How should the input instance be perturbed to change the model's prediction?". However, existing works mainly focus on generating explanations that can effectively alter model predictions, while neglecting whether the explanations remain aligned with the original data distribution, leading to the distribution shift problem. To address this problem, we propose a novel method called ICExplainer for generating explanations within the original distribution. Specifically, we introduce graph diffusion-based generative model into the counterfactual reasoning, treating it as an optimization objective for graph distribution learning. Taking insights from variational inference, we use it to estimate the true distribution of the input graphs to retain essential structural and semantic information. The inferred distribution is then utilized as prior knowledge to guide the reverse process, ensuring that generated explanations are both counterfactual and distributionally coherent. Extensive experiments conducted on both synthetic and real-world datasets demonstrate the superior performance of ICExplainer over existing methods.

1 Introduction

Graph neural networks (GNNs) have emerged as a powerful tool for modeling graph-structured data, with widespread applications in real-world scenarios such as drug discovery (Jiang et al. 2020), label learning (Li et al. 2025a,b), cluster analysis (Liang et al. 2025; Cheng et al. 2025), and recommendation systems (Chen et al. 2024a; Zhang et al. 2025). However, like other neural networks, GNNs are also black box models. This limits their further application in high-stake domains that require explainability and transparency (Wu et al. 2022; Longa et al. 2025; Li et al. 2025c). Consequently, numerous methods have been proposed to study its decision-making process. Among these methods, providing counterfactual explanation for GNNs is an important research branch.

*Corresponding author.



(a) Explanation generation (b) OOD problem

Figure 1: Example of counterfactual explanation generation in node-level classification task and OOD problem. (a) illustrates the explanation generation process. The original graph $\mathcal{G}$, after being perturbed by the explainer, forms the corresponding counterfactual explanation graph $\mathcal{G}^{cf}$, The red dashed line indicates the deleted edge, while the node to be explained is highlighted in red and blue; (b) demonstrates the distribution of counterfactual graph $\mathcal{G}^{cf}$ deviating from this of the original graph $\mathcal{G}$.

Counterfactual explanation enhances model explainability by answering the question "How should the input instance be slightly perturbed to alter the model's prediction?" (Verma, Dickerson, and Hines 2020; Schut et al. 2021). Most existing methods address this by iteratively removing edges from the input graph and feeding the modified graphs into GNNs, optimizing the explainer based on the resulting prediction changes (Lucic et al. 2022; Tan et al. 2022; Bajaj et al. 2021). However, these edge removal-based methods fail to account for the in-distribution property of the explanation graphs. As shown in Figure 1, the explanation graph can significantly deviates from the distribution of the original graph, leading to the distribution shift problem, also known as Out-Of-Distribution (OOD) problem. This will cause the following negative impacts:

- The OOD problem undermines the model's predictive capabilities, making it unclear whether changes in the model's predictions are due to the explainer genuinely capturing counterfactual-related property or being influenced by distribution shifts (Amara, El-Assady, and Ying 2023). This results in unreliable and potentially suboptimal explanations.
- It is not suitable for some real-world applications where

domain-specific rules exist (He et al. 2021; Wang and Shen 2023; Liang et al. 2024). For example, in molecule generation, a molecule graph is valid if it adheres to the valence bond theory (Lewis 1933). However, when the explanation graph significantly deviates from the distribution of the original molecule graph, it may not necessarily satisfy these predefined rules.

An effective approach to address this distribution challenge is integrating graph generative modeling (Li et al. 2022; Vignac et al. 2022; Fu et al. 2024) into counterfactual reasoning (Kosan et al. 2023), which can generate explanations that possess counterfactual property while remaining as closely aligned as possible with the original data distribution. Consequently, recent studies have proposed several generative model-based approaches (Ma et al. 2022; Chen et al. 2023). **We present additional related work in Appendix**. CLEAR (Ma et al. 2022) and D4Explainer (Chen et al. 2023) both incorporate graph distribution modeling as an optimization objective to constrain the distribution of generated graphs. However, due to the inherent diversity and complexity of graph-structured input distributions, the aforementioned methods are not always able to accurately capture the underlying data distribution, resulting in the generated counterfactual graphs may still deviate significantly from original graphs, leaving the OOD problem unresolved. Therefore, there is an urgent need for a method that can faithfully model the underlying true input distribution and generate counterfactual explanations that are both effective and distributionally aligned.

To overcome the aforementioned problems, in this work we propose a novel generative model-based method called ICExplainer, which introduces graph diffusion into counterfactual reasoning. Specifically, during the forward diffusion process, we introduce varying levels of random noise to the input graph to generate a series of noisy graphs, and meanwhile apply variational inference to derive the true distribution of the input graph. This distribution then serves as prior knowledge to guide the reverse process of graph diffusion. In this process, noise and edges unrelated to counterfactual property can be removed, while the learned prior provides global semantic guidance for the denoising model, aiding in the reconstruction of original distribution. Finally, the denoising model outputs a fully connected probability adjacency matrix, from which candidate counterfactual explanations can be sampled. Our main contributions are summarized as follows:

- **Problem.** We systematically analyze and address the OOD problem in GNN counterfactual explanation, which is crucial for enhancing the reliability of explanation.
- **Methodology.** We propose a novel method called ICExplainer, which consists of a variational inference-based graph distribution inference process and a counterfactual graph distribution learning process based on graph diffusion model, ensuring the explanations that exhibit both counterfactual validity and in-distribution consistency.
- **Experiments.** Extensive experiments on both synthetic and real-world datasets have shown that ICExplainer outperforms state-of-the-art methods.

2 Preliminaries

Graph Neural Networks. We define the graph $\mathcal{G} = (\mathcal{V}, \boldsymbol{X}, \boldsymbol{A})$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the nodes set and $\boldsymbol{A} \in \{0,1\}^{n\times n}$ is the adjacency matrix. $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ represents the node feature matrix, where d is the node feature dimension and the i-th row is the feature vector of the node v_i. GNNs can learn node representations through message passing mechanism, and each node's representation is updated by aggregating its own representations and those of its neighbors, which is denoted as follows:

$$\boldsymbol{h}_i^{(l)} = AGGR\left(\boldsymbol{h}_i^{(l-1)}, \{\boldsymbol{h}_j^{(l-1)} \Big| j \in N(v_i)\}\right), \quad (1)$$

where $N(v_i)$ is the neighbors set of v_i, $\boldsymbol{h}_i^{(l)}$ is the representation of v_i at layer l. The $AGGR(\cdot)$ typically represents a sum, mean, or concatenation operation.

Counterfactual Explanation and Post-hoc Instance-level Explanation. Given an instance $\mathcal{G}$ and a well-trained GNN f, the goal of counterfactual explanation is to find the minimal perturbation to the original input graph that changes the model prediction. Without loss of generality, we consider the problem of counterfactual explanation for the graph classification tasks, including node-level classification and graph-level classification.

Definition 1 (Counterfactual Explanation)**.** *Let $\mathcal{G}$ be the given graph and $Y_{\mathcal{G}}$ be the prediction of f on $\mathcal{G}$. Our task is to introduce the minimal perturbations to form a new graph $\mathcal{G}^{cf}$ such that $Y_{\mathcal{G}} \neq Y_{\mathcal{G}^{cf}}$. Formally, it can be described as the following optimization problem:*

$$\mathcal{G}^{cf} = \underset{f(\mathcal{G})\neq f(\mathcal{G}^{cf})}{\arg\min}\ dist(\mathcal{G}, \mathcal{G}^{cf}), \quad (2)$$

where $dist(\mathcal{G}, \mathcal{G}^{cf})$ is used to measure the similarity between graphs $\mathcal{G}$ and $\mathcal{G}^{cf}$, which can be specified by the number of changed edges.

Following the existing works (Ying et al. 2019; Luo et al. 2020; Yuan et al. 2023), the method considered in this paper is model-agnostic, treating GNN model as a black box, i.e., the *post-hoc instance-level* explanation method.

Definition 2 (Post-hoc Instance-level GNN Counterfactual Explanation)**.** *Given a well-trained GNN model f, for an arbitrary graph $\mathcal{G} = (\mathcal{V}, \boldsymbol{X}, \boldsymbol{A})$, the goal of post-hoc instance-level GNN explanation is to find a graph $\mathcal{G}^{cf}$ that can explain the prediction of f on $\mathcal{G}$.*

Distribution Shift in Post-hoc Explanation. Existing methods typically input the explanation generated by the explainer into the to-be-explained model f to observe whether the prediction changes. These methods assume that the model f can accurately predict the label of the counterfactual graph $\mathcal{G}^{cf}$, so that the explainer can be optimized based on the predictions of f. However, these methods fail to account for the distributional consistency between the original graph and the counterfactual graph. Namely, let $P_{\mathcal{G}}$ be the distribution of the original graph and $P_{\mathcal{G}^{cf}}$ be the distribution of the counterfactual graph, $P_{\mathcal{G}} \neq P_{\mathcal{G}^{cf}}$. This distributional shift can affect the predictive ability of the model f, resulting in unreliable explanations.

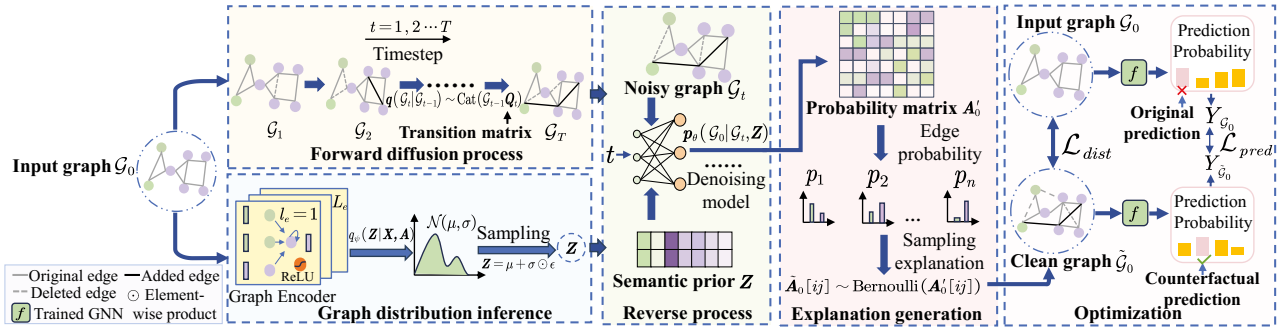


Figure 2: Overall framework of the proposed ICExplainer model. We first introduce variational inference to approximate the true distribution of the input graph, capturing its key information by learning a latent variable $\boldsymbol{Z}$, which will be used to guide the learning of the counterfactual graph distribution. We then apply graph diffusion model to the counterfactual graph distribution learning, which consists of forward diffusion process $\boldsymbol{q}\left(\mathcal{G}_t|\,\mathcal{G}_{t-1}\right)$ and reverse process $\boldsymbol{p}_\theta$. Finally, the generated counterfactual explanation graphs will be optimized under the constraints of distribution loss $\mathcal{L}_{dist}$ and prediction loss $\mathcal{L}_{pred}$.

3 Methodology

In this section, we elaborate on the architecture of ICExplianer, our method for generating counterfactual explanations for GNNs. It consists of a variational inference-based graph distribution inference mechanism and a counterfactual graph distribution learning process based on graph diffusion models. The model architecture is illustrated in the Figure 2.

Graph Distribution Inference

Due to the diversity and complexity of input graph distributions(Fu et al. 2021, 2023), we introduce a variational inference-based graph distribution inference mechanism. The core idea is to use variational inference to approximate the underlying true distribution of the input graphs. Specifically, for each graph $\mathcal{G} = (\mathcal{V}, \boldsymbol{X}, \boldsymbol{A})$, it is assumed that its generation process involves an unobserved continuous random latent variable $\boldsymbol{Z}$ (Paisley, Blei, and Jordan 2012; Kingma, Welling et al. 2013). This process can be divided into two steps: 1) sampling $\boldsymbol{Z}$ from the prior distribution $p\left(\boldsymbol{Z}\right)$, and 2) sampling $\boldsymbol{A}$ from the conditional probability distribution $p\left(\boldsymbol{A}|\,\boldsymbol{Z}\right)$. We need to infer the latent variable $\boldsymbol{Z}$ based on the observed graph $\mathcal{G}$. However, in most cases, the posterior distribution $p\left(\boldsymbol{Z}|\,\boldsymbol{X},\boldsymbol{A}\right)$ is complex and unknown. Therefore, variational inference approximates $p\left(\boldsymbol{Z}|\,\boldsymbol{X},\boldsymbol{A}\right)$ by introducing a variational distribution $q_\psi\left(\boldsymbol{Z}|\,\boldsymbol{X},\boldsymbol{A}\right)$. We assume $q_\psi\left(\boldsymbol{Z}|\,\boldsymbol{X},\boldsymbol{A}\right)$ is a Gaussian distribution, and the encoding process is defined as follows:

$$q_\psi\left(\boldsymbol{Z}|\,\boldsymbol{X},\boldsymbol{A}\right) = \mathcal{N}\left(\boldsymbol{Z}|\,\mu,\sigma\right), \tag{3}$$

where μ and σ represent the mean and standard deviation of the Gaussian distribution calculated by $Encoder_\mu\left(\cdot\right)$ and $Encoder_\sigma\left(\cdot\right)$, respectively. By employing the reparameterization trick, $\boldsymbol{Z}$ can be computed as follows:

$$\boldsymbol{Z} = \mu + \sigma \odot \epsilon, \tag{4}$$

where $\epsilon \in \mathcal{N}\left(0, \boldsymbol{I}\right)$ and $\odot$ denotes the element-wise product operation. To optimize $\boldsymbol{Z}$, the following tractable evidence lower bound (ELBO) can be employed as the optimization objective:

$$\begin{aligned}\log p\left(\boldsymbol{A}\right) \geq \mathcal{L}_{inf} = &-D_{KL}\left[q_\psi\left(\boldsymbol{Z}\mid\boldsymbol{X},\boldsymbol{A}\right)||p\left(\boldsymbol{Z}\right)\right] \\ &+\mathbb{E}_{q_\psi\left(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A}\right)}\left[\log p\left(\boldsymbol{A}\mid\boldsymbol{Z}\right)\right],\end{aligned} \tag{5}$$

where D_{KL} represents the Kullback–Leibler (KL) divergence between the prior distribution $p\left(\boldsymbol{Z}\right)$ and the variational distribution $q_\psi\left(\boldsymbol{Z}|\,\boldsymbol{X},\boldsymbol{A}\right)$. The network used to learn the distribution $q_\psi\left(\boldsymbol{Z}|\,\boldsymbol{X},\boldsymbol{A}\right)$ serves as the encoder, which maps the input graph to a distribution over the latent variable $\boldsymbol{Z}$. Similarly, $p\left(\boldsymbol{A}\mid\boldsymbol{Z}\right)$ acts as the decoder, meaning that given $\boldsymbol{Z}$, it can reconstruct graph $\mathcal{G}$.

The latent variable $\boldsymbol{Z}$ obtained through variational inference captures essential structural and semantic information of the input graph. Therefore, we leverage $\boldsymbol{Z}$ as a semantic prior to guide the reverse process. By injecting this high-fidelity prior, the model ensures that the generated explanation graphs remain aligned with the original data distribution and preserve key in-distribution property.

Counterfactual Graph Distribution Learning

Forward Diffusion Process. To enable the generation of valid counterfactual explanations, we first transform the input graph $\mathcal{G}_0$ (i.e., the original graph $\mathcal{G}$) into a noisy state approximating an Erdős–Rényi random graph $\mathcal{G}_T$ (Erdos, Rényi et al. 1960) via a discrete structural diffusion process (Haefeli et al. 2022). This controlled corruption progressively obscures the original graph's structure, creating a search space from which the model will later reconstructs the corresponding counterfactual explanation graph.

Let $a_t^{ij} \in \{0,1\}^2$ represent the 2-dimensional one-hot encoding of the ij-th element of the adjacency matrix $\boldsymbol{A}_t$. Then, the forward transition for each edge is defined by a categorical distribution (Cat):

$$q\left(\boldsymbol{a}_t^{ij}\middle|\,\boldsymbol{a}_{t-1}^{ij}\right) = \text{Cat}\left(\boldsymbol{a}_t^{ij}; \boldsymbol{P} = \boldsymbol{a}_{t-1}^{ij}\boldsymbol{Q}_t\right), \tag{6}$$

where $\boldsymbol{Q}_t$ is a transition matrix with entries $[\boldsymbol{Q}_t]_{rs} = q\left(a_t^{ij} = s \mid a_{t-1}^{ij} = r\right)$, capturing the probability of edge-state transitions. The t-step transition probability can be computed in closed-form as $q\left(\boldsymbol{a}_t^{ij} \middle| \boldsymbol{a}_0^{ij}\right) = \text{Cat}\left(\boldsymbol{a}_t^{ij}; \boldsymbol{P} = \boldsymbol{a}_0^{ij}\overline{\boldsymbol{Q}}_t\right)$, where $\overline{\boldsymbol{Q}_t} = \boldsymbol{Q}_1\boldsymbol{Q}_2\cdots\boldsymbol{Q}_t$. We model the diffusion of each edge independently. Consequently, the graph-level transition distribution is factorized as:

$$q\left(\mathcal{G}_t|\,\mathcal{G}_{t-1}\right) = \prod_{ij} q\left(\boldsymbol{a}_t^{ij} \middle| \boldsymbol{a}_{t-1}^{ij}\right). \tag{7}$$

Applying this forward diffusion process yields a series of noisy graphs $\{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T\}$. During the reverse process, they are denoised under the guidance of the learned semantic prior $\boldsymbol{Z}$ to generate candidate counterfactual explanation graphs that are both valid and distributionally coherent.

Reverse Process Parameterization and Denoising Model. The reverse process trains a denoising model to approximate the distribution of clean graph (i.e., the counterfactual graph). To stabilize the training procedure, we follow (Austin et al. 2021; Haefeli et al. 2022), and use a network $\text{nn}_\theta(\mathcal{G}_t)$ to predict a distribution $p_\theta(\mathcal{G}_0|\mathcal{G}_t)$ over the clean graph. This parameterization can be expressed as:

$$p_\theta(\mathcal{G}_{t-1}|\mathcal{G}_t) \propto \sum_{\tilde{\mathcal{G}}_0} q(\mathcal{G}_{t-1}|\mathcal{G}_t, \tilde{\mathcal{G}}_0) \cdot \tilde{p_\theta}(\tilde{\mathcal{G}}_0|\mathcal{G}_t). \tag{8}$$

In order to effectively learn the counterfactual graph distributions, we build the denoising model upon the Provably Powerful Graph Network (PPGN) (Maron et al. 2019), chosen for its strong expressiveness. The model input is as follows:

$$\boldsymbol{P_{in}} = \text{Concat}\left(\mathcal{G}_t, \boldsymbol{Z}, \bar{\beta}_t \cdot \boldsymbol{I}\right), \tag{9}$$

where $\mathcal{G}_t$ is the noisy graph, $\boldsymbol{Z}$ is the semantic latent vector and $\bar{\beta}_t$ is the transition probability of edge states. We construct a diagonal matrix $\bar{\beta}_t \cdot \boldsymbol{I}$ to represent the current noise level, and it will pass through an Multilayer Perceptron (MLP) to incorporate time information. We feed $\boldsymbol{P_{in}}$ into a stack of PPGN layers with residual connections and layer normalization. The concatenated output of L PPGN layers can be denoted by:

$$\mathcal{C}\left(\boldsymbol{P_{in}}\right) = \text{Concat}\left(\left\{\text{PPGN}^{(l)}\left(\boldsymbol{P_{in}}\right)\right\}_{l=1}^{L}\right). \tag{10}$$

Finally, we employ an additional MLP_{out} to reduce the dimensionality of $\mathcal{C}\left(\boldsymbol{P_{in}}\right)$ and obtain the final output $\boldsymbol{F_{out}}$:

$$\boldsymbol{F_{out}} = \text{MLP}_{\text{out}}\left(\mathcal{C}\left(\boldsymbol{P_{in}}\right)\right). \tag{11}$$

Counterfactual Explanation Generation. The denoising model outputs a probabilistic adjacency matrix $\boldsymbol{A}_0'$, which represents the model's predicted distribution over clean graph. The elements in the matrix range in [0,1], indicating the probability of each edge's existence in the counterfactual graph. However, adjacency matrix are typically discrete and often assumed to contain only binary values (where $\boldsymbol{A}\left[v_i, v_j\right] = 1$ if an edge exists between v_i and v_j, otherwise = 0). Therefore, based on the probabilities in $\boldsymbol{A}_0'$, we sample a binary discrete matrix using the Bernoulli distribution $\tilde{\boldsymbol{A}}_0\left[ij\right] \sim \text{Bernoulli}\left(\boldsymbol{A}_0'\left[ij\right]\right)$. To allow gradient-based optimization, we apply categorical reparameterization (Jang, Gu, and Poole 2017) to the Bernoulli variable $\tilde{\boldsymbol{A}}_0\left[ij\right]$. This continuous relaxation of the Bernoulli distribution can be formulated as:

$$\tilde{\boldsymbol{A}}_0\left[ij\right] = \delta\left(\frac{\log\epsilon - \log\left(1-\epsilon\right) + \log p - \log\left(1-p\right)}{\tau}\right), \tag{12}$$

where $\epsilon \sim \text{Uniform}\left(0, 1\right)$, δ is the activation function and τ is the temperature used to control the similarity between the relaxed distribution and the Bernoulli distribution. Through the aforementioned sampling process, we obtain the discrete adjacency matrix $\tilde{\boldsymbol{A}}_0$ and its corresponding minimally perturbed counterfactual explanation graph $\tilde{\mathcal{G}}_0$ (i.e.,$\mathcal{G}^{cf}$).

Loss Function Optimization. To optimize the generated explanation graphs, it is necessary to consider both the in-distribution property and the counterfactual property of the generated graphs. Inspired by (Haefeli et al. 2022), we mitigate training instability issues in the model by extending specific parameterizations and reweighting each KL divergence in Eq. (24) (See Appendix). Combined with the learned prior $\boldsymbol{Z}$, we have the following distribution loss:

$$\mathcal{L}_{dist} = -\,\mathbb{E}_{q(\mathcal{G}_0)} \sum_{t=1}^{T} \left(1 - 2\bar{\beta}_t + \frac{1}{T}\right) \mathbb{E}_{q(\mathcal{G}_t|\mathcal{G}_0)} \log p_\theta\left(\mathcal{G}_0 \mid \mathcal{G}_t, \boldsymbol{Z}\right), \tag{13}$$

the reweighting term $1 - 2\bar{\beta}_t + \frac{1}{T}$ linearly assigns more weight to samples with less noise. To ensure that the generated explanation graphs effectively alter the model's predictions, we employ the following prediction loss:

$$\mathcal{L}_{pred} = -\,\mathbb{E}_{\substack{\mathcal{G}_0\sim q,\ t\sim\mathcal{U}(0,T),\\ \mathcal{G}_t\sim q(\cdot|\mathcal{G}_0),\\ \boldsymbol{Z}\sim q_\psi(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A}),\\ \tilde{\mathcal{G}}_0\sim p_\theta(\cdot|\mathcal{G}_t,\boldsymbol{Z})}} \left[\log\left(f(\tilde{\mathcal{G}}_0) = Y_{\tilde{\mathcal{G}}_0}\right)\right], \tag{14}$$

where $Y_{\tilde{\mathcal{G}}_0}$ is the desired classification label, i.e., $Y_{\tilde{\mathcal{G}}_0} \neq Y_{\mathcal{G}_0}$. The Eq. (14) is used to reduce the predicted probability of the generated explanation graph under the original label, thereby satisfying the counterfactual property. Ultimately, our optimization objective function is as follows:

$$\mathcal{L} = \mathcal{L}_{dist} + \mathcal{L}_{inf} + \lambda\mathcal{L}_{pred}, \tag{15}$$

where λ is used to balance the contribution of $\mathcal{L}_{pred}$. The model complexity analysis and training algorithm are provided in Appendix.

Methods	Tree-Circles		Tree-Grid		BA-2motifs		MUTAG	
	Valid. ↑	Spars. ↓	Valid. ↑	Spars. ↓	Valid. ↑	Spars. ↓	Valid. ↑	Spars. ↓
GNNExplainer	$0.475_{\pm 0.006}$	$0.184_{\pm 0.005}$	$0.476_{\pm 0.035}$	$0.399_{\pm 0.005}$	$0.369_{\pm 0.020}$	$0.384_{\pm 0.004}$	$0.167_{\pm 0.007}$	$0.207_{\pm 0.001}$
PGExplainer	$0.254_{\pm 0.071}$	$0.158_{\pm 0.096}$	$0.468_{\pm 0.011}$	$0.402_{\pm 0.009}$	$0.364_{\pm 0.045}$	$0.146_{\pm 0.100}$	$0.324_{\pm 0.035}$	$0.209_{\pm 0.195}$
CF^2	$0.578_{\pm 0.016}$	$0.132_{\pm 0.001}$	$0.695_{\pm 0.003}$	$\underline{0.131}_{\pm 0.000}$	$0.492_{\pm 0.000}$	$0.369_{\pm 0.006}$	$\underline{0.760}_{\pm 0.005}$	$0.459_{\pm 0.004}$
D4Explainer	$\underline{0.905}_{\pm 0.002}$	$\underline{0.107}_{\pm 0.049}$	$\underline{0.883}_{\pm 0.032}$	$0.273_{\pm 0.011}$	$\underline{0.789}_{\pm 0.022}$	$0.200_{\pm 0.004}$	$0.684_{\pm 0.009}$	$0.104_{\pm 0.024}$
ProxyExplainer	$0.628_{\pm 0.024}$	$0.127_{\pm 0.012}$	$0.795_{\pm 0.017}$	$0.203_{\pm 0.016}$	$0.575_{\pm 0.007}$	$\mathbf{0.096}_{\pm 0.001}$	$0.370_{\pm 0.015}$	$\underline{0.087}_{\pm 0.006}$
ICExplainer	$\mathbf{0.975}_{\pm 0.012}$	$\mathbf{0.067}_{\pm 0.010}$	$\mathbf{0.935}_{\pm 0.013}$	$\mathbf{0.08}1_{\pm 0.002}$	$\mathbf{0.937}_{\pm 0.038}$	$\underline{0.107}_{\pm 0.023}$	$\mathbf{0.843}_{\pm 0.022}$	$\mathbf{0.049}_{\pm 0.011}$

Table 1: Validity and Sparsity of ours and baseline explainers on four datasets. Each experiment is conducted five times with random seeds. The best result for each dataset is highlighted in bold, and the second-best ones are underlined.

Methods	Tree-Circles	BA-2motifs	MUTAG
GNNExplainer	$0.505_{\pm 0.002}$	$0.426_{\pm 0.004}$	$0.259_{\pm 0.001}$
PGExplainer	$0.457_{\pm 0.035}$	$0.418_{\pm 0.019}$	$0.358_{\pm 0.027}$
CF^2	$0.520_{\pm 0.003}$	$0.503_{\pm 0.001}$	$\underline{0.617}_{\pm 0.002}$
D4Explainer	$\underline{0.854}_{\pm 0.031}$	$\underline{0.727}_{\pm 0.022}$	$0.537_{\pm 0.051}$
Proxyxplainer	$0.574_{\pm 0.016}$	$0.514_{\pm 0.016}$	$0.300_{\pm 0.013}$
ICExplainer	$\mathbf{0.913}_{\pm 0.011}$	$\mathbf{0.843}_{\pm 0.039}$	$\mathbf{0.630}_{\pm 0.047}$

Table 2: Fidelity evaluation on Tree-Circles, BA-2motifs and MUTAG.

4 Experiments

In this section, we conduct comprehensive experimental studies on benchmark datasets to empirically validate the effectiveness of our method ICExplainer. These experiments primarily aim to address the following research questions:

- **RQ1:** How does the performance of ICExplainer compare to other state-of-the-art methods?
- **RQ2:** Is the OOD problem significant in existing counterfactual explanation methods for GNNs? Can ICExplainer alleviate this problem?
- **RQ3:** How does each component in ICExplainer affect the overall performance of explanation generation?

Experimental Settings

We evaluate the explanation performance on two typical tasks: node-level classification and graph-level classification. For node-level task, we use two synthetic datasets, Tree-Circles and Tree-Grid (Ying et al. 2019). For graph-level task, we use one synthetic dataset, BA-2motifs (Luo et al. 2020), and one real-world dataset, MUTAG (Debnath et al. 1991). We use two general baselines and three state-of-the-art explainers for performance comparison. These include GNNExplainer (Ying et al. 2019), PGExplainer (Luo et al. 2020), CF^2 (Tan et al. 2022), D4Explainer (Chen et al. 2023) and ProxyExplainer (Chen et al. 2024b). For the methods originally designed to provide factual explanations, we construct counterfactual explanations by removing the most important edges identified by these methods. Detailed information regarding the datasets, baselines and base GNN models are listed in the Appendix.

We employ standard metrics to evaluate the quality of explanations, including Validity (Ma et al. 2022), Fidelity (Yuan et al. 2023), and Sparsity (Chen et al. 2023). Validity is defined as the proportion of generated counterfactual explanation that effectively alter the model's predictions, Fidelity is used to measure the magnitude of confidence decline in the original predicted class, and Sparsity measures the proportion of modifications in the counterfactual graph compared to the original graph. We provide the detailed definitions of the above metrics in Appendix.

Performance of Different Methods (RQ1)

To answer RQ1, we compared the proposed method with other baselines, conducting each experiment five times with random seeds. The average Validity, Sparsity, and corresponding standard deviations are shown in Table 1. From the table results, we can draw the following observations:

- From the perspective of validity, our proposed method outperforms other baselines on most datasets. Specifically, compared to the leading baseline, our method achieves an average improvement of 10.8% in validity scores on synthetic datasets and 10.9% on real-world datasets. This demonstrates that our approach consistently captures factors related to counterfactual properties across different datasets, ensuring robust explanation performance.
- From the perspective of sparsity, our proposed method reduces the average by 26.4% compared to the leading baseline, indicating that our model not only finds valid counterfactuals but also minimizes perturbations to the original graph, thereby mitigating the negative impact of distribution shifts on predictions.

Fidelity is also a crucial metric for evaluating the quality of explanations, and we use it for further performance comparison. As shown in Table 2, ICExplainer achieves an average improvement of 8.3% in fidelity on benchmark datasets, consistently outperforming baseline methods.

These observations demonstrate the effectiveness of our proposed method in generating counterfactual explanations. Compared to other strong baselines, ICEXplainer consistently exhibits superior performance across different datasets, and its robustness and adaptability.

	Tree-Circles						Tree-Grid					
Metric	GNNE	PGE	CF^2	D4E	PROE	ICE	GNNE	PGE	CF^2	D4E	PROE	ICE
Deg.	0.1594	0.0468	0.1092	0.0852	0.1005	**0.0383**	0.2087	0.2629	0.1832	0.1819	0.1349	**0.1291**
Clus.	**0.0000**	**0.0000**	0.0005	0.0056	**0.0000**	0.0071	0.0003	0.0001	0.0006	0.0002	**0.0000**	**0.0000**
Spec.	0.0755	0.0767	0.0516	0.0492	0.0442	**0.0390**	0.0785	0.1908	0.0763	0.0362	0.2142	**0.0321**
Sum.	0.2349	0.1235	0.1613	0.1400	0.1447	**0.0844**	0.2875	0.4538	0.2601	0.2183	0.3491	**0.1612**
	BA-2motifs						**MUATG**					
Metric	GNNE	PGE	CF^2	D4E	PROE	ICE	GNNE	PGE	CF^2	D4E	PROE	ICE
Deg.	0.1296	0.0741	0.0985	0.0608	**0.0359**	0.0487	0.2792	**0.0025**	0.2685	0.1793	0.0886	0.0962
Clus.	0.0764	0.0071	0.0369	0.0038	0.0109	**0.0007**	0.0003	0.0014	**0.0000**	0.0006	0.0007	**0.0000**
Spec.	0.0467	0.0197	**0.0192**	0.0365	0.0528	0.0318	0.0219	1.0740	**0.0088**	0.0515	0.0789	0.0667
Sum.	0.2527	0.1009	0.1546	0.1011	0.0996	**0.0812**	0.3014	1.0779	0.2773	0.2314	0.1682	**0.1629**

Table 3: MMD results between counterfactual graphs and the original graphs. GNNE, PGE, D4E, PROE, ICE represent the GNNExplainer, PGExplainer, D4Explainer, ProxyExplaine and ICExplainer, respectively. We report MMD distances of degree distributions (*Deg.*), cluster coefficients (*Clus.*), spectrum distributions (*Spec.*), and the summation (*Sum.*).

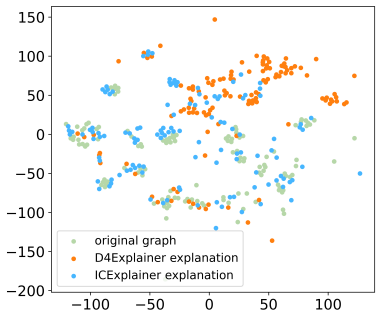


Figure 3: Visualizations of the distribution embeddings.

Alleviating OOD Problem (RQ2)

In this section, we evaluate the ability of ICExplainer to generate in-distribution counterfactual explanation.

Visualizing Distribution Shift. In this section, we employ t-Distributed Stochastic Neighbor Embedding (t-SNE) (Van der Maaten and Hinton 2008) to evaluate the in-distribution property of explanation graphs by visualizing the distributing embeddings of the original graphs and the explanation graphs generated by D4Explainer and ICExplainer on Tree-Grid dataset, respectively.

The visualization results are shown in Figure 3. It can be observed that there is a diversity in the original graph distributions, with most orange points deviate from the green points, indicating that the explanation graphs generated by existing methods exhibit a significant distribution shift issue. Meanwhile, the blue points are more closely aligned with the green points, suggesting that the explanation graphs produced by ICExplainer better conform to the original graph distributions. This demonstrates the effectiveness of our method in mitigating the OOD issue. We provide more quantitative evaluation results on distribution analysis in Appendix.

Measure Distribution Similarity. In this section, we quantitatively evaluate ICExplainer's capability in generating in-distribution explanations. Following previous work (Chen et al. 2024b), we employ Maximum Mean Discrepancy (MMD) to measure the differences in the distributions of various graph statistics between the explanation graphs and the original graphs. Smaller MMD indicates a more similar graph distribution.

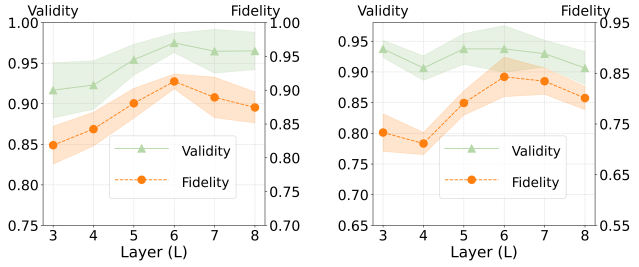
The results are shown in Table 3. Our observations are as follows. First, the MMDs between the explanation graphs generated by most baselines and the original graphs are generally large, suggesting that models trained on the original graphs may not correctly predict OOD explanation graphs. This results in unreliable explanations. Second, the strong baseline method D4Explainer, based on a generative model, also does not perform well across various datasets, indicating that it does not always capture diverse graph distributions. Third, the explanation graphs generated by our method generally have smaller MMDs, demonstrating their in-distribution property and also highlighting ICExplainer's ability to capture different graph distributions.

Ablation Study (RQ3)

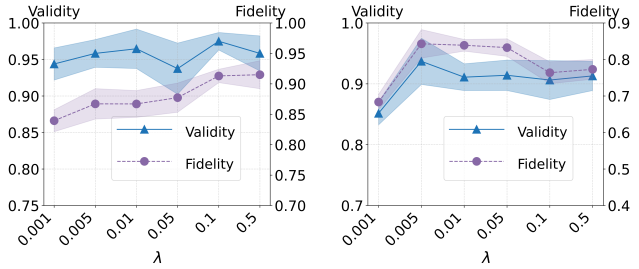
In this section, we conduct ablation studies to investigate the roles of different components in ICExplainer.

Effect of the number of powerful layer L. For L, we vary it among {3,4,5,6,7,8}. Figure 4(a)(b) illustrates the variation in model performance across different L. Overall, the performance of the model improves with an increase in L, but deeper models may increase complexity, leading to overfitting. The best performance is often achieved when $L \in [5, 8]$.

Effect of λ. We vary λ within the range {0.001, 0.005, 0.01, 0.05, 0.1, 0.5}. Figure 4(c)(d) visualizes the change in performance depending on λ. Intuitively, Validity and Fidelity increases with the growth of λ. The Figure 4(c) demonstrates the results that align with expectations, but

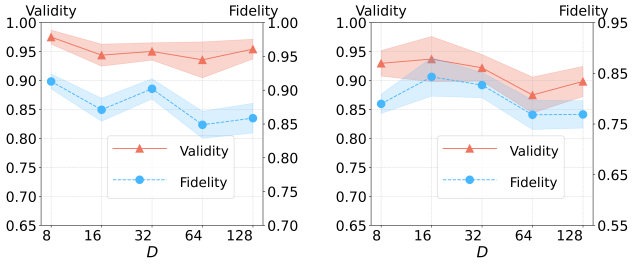


(a) Tree-Circles (b) BA-2motifs

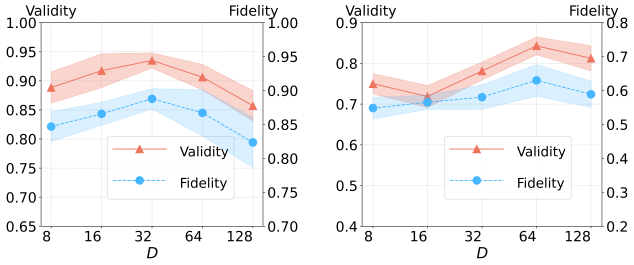


(c) Tree-Circles (d) BA-2motifs

Figure 4: Effect of L and λ.



(a) Tree-Circles (b) BA-2motifs



(c) Tree-Grid (d) MUTAG

Figure 5: Effect of D.

Figure 4(d) presents a different outcome. We attribute this counterintuitive behavior to the fact that GNNs are more susceptible to OOD problem when making predictions on input instances with complex data distributions. Excessively large λ may cause the model to aggressively modify the original graph, leading the generated explanations to deviate from the original distribution and thus producing sub-optimal explanations. We found that when $\lambda \in (0.005, 0.1)$, the model achieves a relatively optimal performance.

Effect of representation dimension D. D represents the dimensionality size of $\boldsymbol{Z}$, we vary it among $\{8, 16, 32, 64, 128\}$. Figure 5 shows the impact of D on model's performance. We found that both excessively large and small dimensions can affect the expressive power of $\boldsymbol{Z}$, leading to fluctuations in model performance. In most cases, a small representation dimension is sufficient to achieve satisfactory performance, while an excessively high dimension may introduce redundant noise information.

Effect of encoder architecture. To evaluate the extent to which encoder selection impact the explanation performance of ICExplainer, we conduct ablation studies on various GNN encoders, including SAGE (Hamilton, Ying, and Leskovec 2017), GAT (Velickovic et al. 2017), GIN (Xu et al. 2018), and GCN (Kipf and Welling 2016). The results are shown in Table 4. We can observe that GCN is the optimal encoder architecture for ICExplainer. Although other encoder architectures exhibit some performance degradation compared to GCN, they still achieve superior results relatively to other baselines. This demonstrates the universality of ICExplainer in encoder architecture selection.

Overall, the ablation experiments demonstrate the positive impact of each component of ICExplainer on the explanation performance, indicating the effectiveness of the individual modules. For complete experimental results, please refer to Appendix.

	Tree-Circles		Tree-Grid	
Encoder	**Valid.** ↑	**Spars.** ↓	**Valid.** ↑	**Spars.** ↓
SAGE	0.961	0.098	0.914	0.115
GAT	0.946	0.061	0.895	0.102
GIN	0.954	0.181	0.898	0.116
GCN	**0.975**	**0.067**	**0.935**	**0.081**
	BA-2motifs		**MUTAG**	
Encoder	**Valid.** ↑	**Spars.** ↓	**Valid.** ↑	**Spars.** ↓
SAGE	0.867	0.213	0.820	0.051
GAT	0.930	0.233	0.806	0.077
GIN	0.922	0.177	0.782	0.127
GCN	**0.937**	**0.107**	**0.843**	**0.049**

Table 4: Ablation study on different encoders

5 Conclusion

In this paper, we investigate the overlooked OOD problem in previous counterfactual explanation methods for GNNs. To address it, we propose ICExplainer for generating counterfactual explanations with in-distribution property. It integrates counterfactual reasoning into the generative graph diffusion model, and guides the reverse process by combining the true distribution obtained through variational inference. Extensive experiments on benchmark synthetic and real-world datasets demonstrate the effectiveness of ICExplainer. Future work will explore the combined impact of structural and feature shifts to further enhance the method's applicability.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under grant 2022YFC3303603, National Natural Science Foundation of China under Grants 62477016, 62377028 and 62077045, and Guangdong Basic and Applied Basic Research Foundation under Grants 2024A1515011758, 2024A1515140144 and 2023B1515120064, and Fundamental Research Funds for the Central Universities under grant 21625102.

References

Amara, K.; El-Assady, M.; and Ying, R. 2023. Ginx-eval: Towards in-distribution evaluation of graph neural network explanations. *arXiv preprint arXiv:2309.16223*.

Austin, J.; Johnson, D. D.; Ho, J.; Tarlow, D.; and Van Den Berg, R. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34: 17981–17993.

Bajaj, M.; Chu, L.; Xue, Z. Y.; Pei, J.; Wang, L.; Lam, P. C.-H.; and Zhang, Y. 2021. Robust counterfactual explanations on graph neural networks. *Advances in Neural Information Processing Systems*, 34: 5644–5655.

Chen, J.; Wu, S.; Gupta, A.; and Ying, R. 2023. D4Explainer: In-distribution Explanations of Graph Neural Network via Discrete Denoising Diffusion. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 78964–78986. Curran Associates, Inc.

Chen, W.; Wu, Y.; Zhang, Z.; Zhuang, F.; He, Z.; Xie, R.; and Xia, F. 2024a. FairGap: Fairness-aware recommendation via generating counterfactual graph. *ACM Transactions on Information Systems*, 42(4): 1–25.

Chen, Z.; Zhang, J.; Ni, J.; Li, X.; Bian, Y.; Islam, M. M.; Mondal, A. M.; Wei, H.; and Luo, D. 2024b. Generating in-distribution proxy graphs for explaining graph neural networks. In *Proceedings of the 41st International Conference on Machine Learning*, 7712–7730.

Cheng, J.; Liang, K.; Feng, P.; Liu, W.; Tang, Y.; and He, C. 2025. Clustering Diffusion Model with Frequency-Signal Modulation for Variational Graph Autoencoders. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Debnath, A. K.; Lopez de Compadre, R. L.; Debnath, G.; Shusterman, A. J.; and Hansch, C. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2): 786–797.

Erdos, P.; Rényi, A.; et al. 1960. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1): 17–60.

Fu, X.; Gao, Y.; Wei, Y.; Sun, Q.; Peng, H.; Li, J.; and Li, X. 2024. Hyperbolic geometric latent diffusion model for graph generation. *arXiv preprint arXiv:2405.03188*.

Fu, X.; Li, J.; Wu, J.; Sun, Q.; Ji, C.; Wang, S.; Tan, J.; Peng, H.; and Yu, P. S. 2021. ACE-HGNN: Adaptive curvature exploration hyperbolic graph neural network. In *2021 IEEE International Conference on Data Mining (ICDM)*, 111–120. IEEE.

Fu, X.; Wei, Y.; Sun, Q.; Yuan, H.; Wu, J.; Peng, H.; and Li, J. 2023. Hyperbolic geometric graph representation learning for hierarchy-imbalance node classification. In *Proceedings of the ACM Web Conference 2023*, 460–468.

Haefeli, K. K.; Martinkus, K.; Perraudin, N.; and Wattenhofer, R. 2022. Diffusion models for graphs benefit from discrete state spaces. *arXiv preprint arXiv:2210.01549*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30.

He, C.; Fei, X.; Cheng, Q.; Li, H.; Hu, Z.; and Tang, Y. 2021. A survey of community detection in complex networks using nonnegative matrix factorization. *IEEE Transactions on Computational Social Systems*, 9(2): 440–457.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparametrization with Gumble-Softmax. In *International Conference on Learning Representations*. OpenReview. net.

Jiang, M.; Li, Z.; Zhang, S.; Wang, S.; Wang, X.; Yuan, Q.; and Wei, Z. 2020. Drug–target affinity prediction using graph neural network and contact maps. *RSC advances*, 10(35): 20701–20712.

Kingma, D. P.; Welling, M.; et al. 2013. Auto-encoding variational bayes.

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

Kosan, M.; Verma, S.; Armgaan, B.; Pahwa, K.; Singh, A.; Medya, S.; and Ranu, S. 2023. GNNX-BENCH: Unravelling the Utility of Perturbation-based GNN Explainers through In-depth Benchmarking. In *The 12th International Conference on Learning Representations*.

Lewis, G. N. 1933. The chemical bond. *The Journal of Chemical Physics*, 1(1): 17–28.

Li, D.; Tan, Z.; Li, Q.; Gan, Z.; Xia, T.; Wang, J.; and Li, X. 2025a. FedRog: Robust Federated Graph Classification for Strong Heterogeneity and High-Noise Scenarios. In *Proceedings of the 33rd ACM International Conference on Multimedia*, 6938–6947.

Li, J.; Fu, X.; Sun, Q.; Ji, C.; Tan, J.; Wu, J.; and Peng, H. 2022. Curvature graph generative adversarial networks. In *Proceedings of the ACM web conference 2022*, 1528–1537.

Li, Q.; Li, X.; Wang, J.; et al. 2025c. From GNN to MLP: Enhancing learning with knowledge-augmented distillation and confidence guidance. *Information Fusion*, 103518.

Li, X.; Gan, Z.; Li, Q.; Qu, B.; Wang, J.; et al. 2025b. Rethinking the impact of noisy labels in graph classification: A utility and privacy perspective. *Neural Networks*, 182: 106919.

Liang, K.; Meng, L.; Li, H.; Wang, J.; Lan, L.; Li, M.; Liu, X.; and Wang, H. 2025. From concrete to abstract: multiview clustering on relational knowledge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Liang, K.; Meng, L.; Liu, M.; Liu, Y.; Tu, W.; Wang, S.; Zhou, S.; Liu, X.; Sun, F.; and He, K. 2024. A survey of knowledge graph reasoning on graph types: Static, dynamic,

and multi-modal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12): 9456–9478.

Longa, A.; Azzolin, S.; Santin, G.; Cencetti, G.; Liò, P.; Lepri, B.; and Passerini, A. 2025. Explaining the explainers in graph neural networks: a comparative study. *ACM Computing Surveys*, 57(5): 1–37.

Lucic, A.; Ter Hoeve, M. A.; Tolomei, G.; De Rijke, M.; and Silvestri, F. 2022. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *International Conference on Artificial Intelligence and Statistics*, 4499–4511. PMLR.

Luo, D.; Cheng, W.; Xu, D.; Yu, W.; Zong, B.; Chen, H.; and Zhang, X. 2020. Parameterized explainer for graph neural network. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 19620–19631.

Ma, J.; Guo, R.; Mishra, S.; Zhang, A.; and Li, J. 2022. CLEAR: generative counterfactual explanations on graphs. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 25895–25907.

Maron, H.; Ben-Hamu, H.; Serviansky, H.; and Lipman, Y. 2019. Provably Powerful Graph Networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2156–2167.

Paisley, J.; Blei, D. M.; and Jordan, M. I. 2012. Variational Bayesian inference with stochastic search. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 1363–1370.

Schut, L.; Key, O.; Mc Grath, R.; Costabello, L.; Sacaleanu, B.; Gal, Y.; et al. 2021. Generating interpretable counterfactual explanations by implicit minimisation of epistemic and aleatoric uncertainties. In *International Conference on Artificial Intelligence and Statistics*, 1756–1764. PMLR.

Tan, J.; Geng, S.; Fu, Z.; Ge, Y.; Xu, S.; Li, Y.; and Zhang, Y. 2022. Learning and Evaluating Graph Neural Network Explanations based on Counterfactual and Factual Reasoning. In *WWW'22: Proceedings of the ACM Web Conference 2022*.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y.; et al. 2017. Graph attention networks. *stat*, 1050(20): 10–48550.

Verma, S.; Dickerson, J.; and Hines, K. 2020. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2(1): 1.

Vignac, C.; Krawczuk, I.; Siraudin, A.; Wang, B.; Cevher, V.; and Frossard, P. 2022. DiGress: Discrete denoising diffusion for graph generation. In *The 11th International Conference on Learning Representations*.

Wang, X.; and Shen, H. W. 2023. GNNInterpreter: A Probabilistic Generative Model-Level Explanation for Graph Neural Networks. In *The 11th International Conference on Learning Representations*.

Wu, B.; Bian, Y.; Zhang, H.; Li, J.; Yu, J.; Chen, L.; Chen, C.; and Huang, J. 2022. Trustworthy graph learning: Reliability, explainability, and privacy protection. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4838–4839.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Ying, Z.; Bourgeois, D.; You, J.; Zitnik, M.; and Leskovec, J. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in Neural Information Processing Systems*, 32.

Yuan, H.; Yu, H.; Gui, S.; and Ji, S. 2023. Explainability in Graph Neural Networks: A Taxonomic Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5): 5782–5799.

Zhang, G.; Yuan, G.; Cheng, D.; Liu, L.; Li, J.; and Zhang, S. 2025. Mitigating propensity bias of large language models for recommender systems. *ACM Transactions on Information Systems*, 43(6): 1–26.