# Generative Explanation for Graph Neural Network: Methods and Evaluation

Jialin Chen[1], Kenza Amara[2], Junchi Yu[3], Rex Ying[1],
[1]Department of Computer Science, Yale University
[2]Department of Computer Science, ETH Zurich
[3]Institute of Automation, Chinese Academy of Sciences
[1]{jialin.chen,rex.ying}@yale.edu,[2]kenza.amara@ai.ethz.ch,[3]yujunchi2019@ia.ac.cn

**Abstract**

Graph Neural Networks (GNNs) achieve state-of-the-art performance in various graph-related tasks. However the black-box nature often limits their interpretability and trustworthiness. Numerous explanation methods have been proposed to uncover the decision-making logic of GNNs, by generating underlying explanatory substructures. In this paper, we conduct a comprehensive review of the existing explanation methods for GNNs from the perspective of graph generation. Specifically, we propose a unified optimization objective for current generative explanation methods, comprising two sub-objectives: Attribution and Information constraints. We further demonstrate their specific manifestations in different generative model architectures and explanation scenarios. With the unified objective of the explanation problem, we reveal the shared characteristics and distinctions among current methods, laying the foundation for future methodological advancements. Empirical results demonstrate the advantages and limitations of different approaches in terms of explanation performance, efficiency, and generalizability.

## 1 Introduction

Graph Neural Networks (GNNs) have emerged as a powerful tool for studying graph-structured data in various applications, such as social networks, drug discovery, and recommendation systems [55, 10, 40, 11, 9, 46, 13]. The explainability and trustworthiness of GNNs are crucial for their successful deployment in real-world scenarios, especially in high-stake applications, such as anti-money laundering, fraud detection, and healthcare forecasting [51, 31, 1]. Explanations for GNNs aim to discover the reasoning logic behind their predictions, making them more understandable and transparent to users. Explanations also help identify potential biases and build trust in the decision-making process of the model. Furthermore, they aid users in understanding complex graph-structured data, leading to improved outcomes in various applications through better feature extraction [54, 12, 47].

Numerous explanation methods have been extensively studied for GNNs, including gradient-based attribution methods [32, 6, 36], perturbation-based methods [48, 41, 52, 34, 16], *etc.* However, most of these methods optimize individual explanations for a specific instance, lacking global attention to the overall dataset and the ability to generalize to unseen instances. To tackle this challenge, generative explainability methods have emerged recently, which instead formulate the explanation task as a distribution learning problem. Generative explainability methods aim to learn the underlying distributions of the explanatory graphs across the entire graph dataset [42, 22, 28, 50], providing a more holistic approach to GNN explanations.

Current surveys in the field of Graph Neural Networks (GNNs) explainability primarily focus on the taxonomy and evaluation of explanation methods [51, 1, 33], as well as broader trustworthy aspects such as robustness, privacy, and fairness [44, 54, 23, 47]. The emerging generative explainability methods prompt us to consider the potential advantages of incorporating distribution learning into the optimization objective, such as better explanation efficiency and generalizability.

Our work stands apart from previous works by thoroughly investigating the different mechanisms for generating explanations. We explore a comprehensive range of graph generation techniques that have been employed in GNN explanation tasks, including cutting-edge techniques such as the Variational Graph Autoencoder (VGAE) and denoising diffusion models. Our study begins by elucidating the core design considerations of different generative models and employs a novel generative perspective to unify a group of effective GNN explanation approaches. The key insight lies in a unified optimization objective, which includes an *Attribution* constraint and an *Information* constraint to ensure that the generated explanations are sufficiently succinct and relevant to the predictions. We subsequently delve into the details of the practical designs of the *Attribution* and *Information* constraints to facilitate the analysis of the connections and potential extensions of current generative explainability methods. The proposed unified optimization objective also empowers GNN users to efficiently design effective generative explainability methods.

Comprehensive experiments on synthetic and real-world datasets demonstrate the advantages and drawbacks of these existing methods. Specifically, our results show that generative approaches are empirically more efficient during the inference stage. Meanwhile, generative approaches achieve the best generalization capacity compared to other non-generative approaches.

This paper is structured as follows. First, we introduce the notations and problem setting in Sec. 2. Then, we propose a standard optimization objective with *Attribution* constraint and *Information* constraint to unify generative explanation methods in Sec. 3.1. Detailed expressions of these two constraints are elaborated in Sec. 3.2 and Sec. 3.3, respectively. In Sec. 3.4, we discuss how to generalize the proposed framework to extensive explanation scenarios, e.g.counterfactual and model-level explanations. Additionally, we present a taxonomy of representative works with different generative backbones in Sec. 4. Finally, we conduct comprehensive evaluations and demonstrate the potential of deep generative methods for GNN explanation in Section 5.

## 2 Preliminaries

### 2.1 Notations and Definitions

Given a well-trained GNN model $f$ (base model) and an instance (i.e.a node or a graph) of the dataset, the objective of the explanation task is to identify concise graph substructures that contribute the most to the model's predictions. The given graph (or $N$-hop neighboring subgraph of the given node) can be represented as a quadruplet $G(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E})$, where $\mathcal{V}$ is the node set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set. $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_n}$ and $\mathbf{V} \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ denote the feature matrices for nodes and edges, respectively, where $d_n$ and $d_e$ are the dimensions of node features and edge features. In this work, we focus on structural explanation, i.e.we keep the dimensions of node and edge features unchanged. The notations used throughout this paper are summarized in Table 1. Depending on the specific explanation scenario, we define the explanation graphs with different target labels as follows.

**Definition 2.1 (Explanation Graph)** *Given a well-trained GNN $f$ and an instance represented as $G(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{V})$, an explanation graph $G_e(\mathcal{V}_e, \mathcal{E}_e, \mathbf{X_e}, \mathbf{E_e})$ for the instance is a compact subgraph of $G$, such that $\mathcal{V}_e \subseteq \mathcal{V}$, $\mathcal{E}_e \subseteq \mathcal{E}$, $\mathbf{X_e} = \{X_j | v_j \in \mathcal{V}_e\}$ and $\mathbf{E_e} = \{E_k | e_k \in \mathcal{E}_e\}$, where $v_j$ and $X_j$ denote the graph node and the corresponding node feature, $e_k$ and $E_k$ denote the graph edge and the corresponding edge feature. Explanation graph $G_e$ is expected to be compact and result in the same predicted label $Y^*$ as the label of $G$ made by $f$, i.e.$Y^* = Y_f(G_e) = Y_f(G)$, where $Y_f(\cdot)$ denotes the predicted label made by the model $f$.*

**Definition 2.2 (Counterfactual Explanation Graph)** *Given a well-trained GNN $f$ and an instance $G$, a counterfactual explanation graph $G_{ce}$ is as close as possible to the original graph $G$, while it results in a different predicted label $Y^*$ with the label of $G$ predicted by $f$, i.e.$Y^* = Y_f(G_{ce}) \neq Y_f(G)$.*

| Notation | Description |
|---|---|
| $G(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E})$ | a graph with nodes $\mathcal{V}$, edges $\mathcal{E}$, node features $\mathbf{X}$ and edge features $\mathbf{E}$ |
| $v_j \in \mathcal{V}$ | a graph node |
| $e_k \in \mathcal{E}$ | a graph edge |
| $G_e$ | an explanation graph |
| $G_{ce}$ | a counterfactual explanation graph |
| $G_m^c$ | a model-level explanation graph for the target class $c$ |
| $\mathcal{G} = \{G^1, \cdots, G^M\}$ | the graph set of $M$ input instances |
| $\mathcal{G}_e = \{G_e^1, \cdots, G_e^M\}$ | the set of $M$ generated explanation graphs |
| $\mathbf{X} = \{X_1, \cdots, X_{\lvert\mathcal{V}\rvert}\} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times d_n}$ | the node feature matrix with $d_n$ feature dimensions |
| $\mathbf{E} = \{E_1, \cdots, E_{\lvert\mathcal{E}\rvert}\} \in \mathbb{R}^{\lvert\mathcal{E}\rvert \times d_e}$ | the edge feature matrix with $d_e$ feature dimensions |
| $A \in \{0,1\}^{\lvert\mathcal{V}\rvert \times \lvert\mathcal{V}\rvert}$ | the unweighted adjacency matrix |
| $\mathcal{Y}$ | the label space |
| $f$ | the well-trained GNN to be explained (base model) |
| $Y^s \in \{0, 1, \cdots, \lvert\mathcal{Y}\rvert\}$ | the original label of $s$, where $s$ can be a node or a graph |
| $Y_f(s) \in \{0, 1, \cdots, \lvert\mathcal{Y}\rvert\}$ | the predicted label of $s$ by $f$ |
| $Y^* \in \{0, 1, \cdots, \lvert\mathcal{Y}\rvert\}$ | the predicted label during the explanation stage |
| $P_f(s) \in [0,1]^{\lvert\mathcal{Y}\rvert}$ | the output probability vector of $s$ by $f$ |
| $f(s) \in \mathbb{R}^{\lvert\mathcal{Y}\rvert}$ | the output logit vector of $s$ by $f$ |
| $g_\theta(\cdot) : \mathcal{G} \to \mathcal{G}_e$ | an explanation generator with parameters $\theta$ |
| $p(\cdot \mid G)$ | the distribution of the explanation graphs for a given $G$ |

Table 1: Summary of the notations

**Definition 2.3 (Model-level Explanation Graph)** *Given a set of graph $\mathcal{G} = \{G^1, \cdots, G^M\}$, where each $G^j \in \mathcal{G}$ has the same label c predicted by the well-trained GNN $f$, a model-level explanation graph $G_m^c$ is a distinctive subgraph pattern that commonly appears in $\mathcal{G}$, and is predicted as the same label c, i.e. $Y^* = Y_f(G_m^c) = c$.*

## 2.2 General Explanation for Graph Neural Network

Given a graph $G$ and the corresponding label $Y^*$ in specific explanation scenarios, generating explanation graphs can be formulated as an optimization problem that maximizes the mutual information between the generated graph and the target label $Y^*$ with the following objective:

$$\begin{aligned} G_e^* &= \underset{G_e}{\operatorname{argmax}}\, MI(Y^*, G_e) = \underset{G_e}{\operatorname{argmax}}\, H(Y^*) - H(Y^*|G_e) \\ &= \underset{G_e}{\operatorname{argmin}}\, H(Y^*|G_e) = \underset{G_e}{\operatorname{argmin}} -\mathbb{E}_{Y^*|G_e} \log P(Y^*|G_e), \end{aligned} \tag{8}$$

where $MI(\cdot, \cdot)$ denotes the mutual information function, $H(\cdot)$ denotes the entropy function, $P(Y^*|G_e)$ measures the probability that $G_e$ is predicted as the label $Y^*$.

**Instance-dependent Explainers**. Early efforts develop explanation frameworks for GNNs that optimize an explanation for each individual instance. For example, the Gradient-based methods [6, 32] evaluate the node and edge importance with the gradient norm of prediction node and edge features. Nodes and edges with higher gradient norms are considered more important and are included in the explanation subgraph of the final prediction. Other methods utilize more advanced frameworks such as mask optimization [48], surrogate model [39], and Monte Carlo Tree Search [52] to search the explanation subgraphs for each individual instance.

Although instance-dependent explainers partly reveal the behavior of GNNs, there are several limitations. Since these methods optimize explanations for individual graphs, they require significant computation and lack holistic knowledge about how the GNN model behaves across the entire dataset. Furthermore, the learning modules in instance-dependent explainers cannot be generalized to explain the predictions for unseen instances, since the parameters are specific for individual instances.

## 3 Generative Framework for Graph Explanations

### 3.1 Unified Optimization Objective

To overcome the aforementioned limitations, recent research has develop approaches that leverage deep generative methods to explain GNNs. Instead of optimizing an explanation for individual instances, the generative methods aim to generate explanations for new graphs by learning a strategy to search for the most explanatory subgraphs across the whole dataset. Formally, given a set of input graphs $\mathcal{G}$, the generative explainer learns the distribution of the underlying explanation graphs $p(G_e|G)$ using a parameterized subgraph generator $g_\theta : \mathcal{G} \to \mathcal{G}_e$. After training, the subgraph generator is capable of identifying the explanation subgraphs that are most important to the desired graph labels:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \log P_{Y^*}(G_e|\theta, G), \tag{9}$$

where $P_{Y^*}(G_e|\theta, G)$ is the probability that the generated $G_e = g_\theta(G)$ is a valid explanation for the desired label $Y^*$. In addition to the validity requirement, an ideal explanation graph should be sparse and compact compared with the given graph. Directly optimizing Eq. 9 leads to a trivial solution where $G_e = G$, as the input graph is most informative for the graph label. To obtain a compact explanation, we impose an information constraint $\mathcal{L}_{\text{INFO}}(G_e, G)$ that restricts the amount of information contained in the generated explanation subgraph, thereby ensuring the conciseness and brevity of the explanations. The overall objective of generative explanation is

$$\min_\theta - \log P_{Y^*}(G_e|\theta, G) + \mathcal{L}_{\text{INFO}}(G_e, G) := \mathcal{L}_{\text{ATTR}}(G_e, Y^*) + \mathcal{L}_{\text{INFO}}(G_e, G). \tag{10}$$

We name the first term in Eq. 10 the attribution loss $\mathcal{L}_{\text{ATTR}}(G_e, Y^*)$, which measures whether $G_e$ captures the most important substructures for the desired label $Y^*$. $\mathcal{L}_{\text{ATTR}}$ is typically the cross-entropy loss for categorical $Y^*$ and mean squared loss for continuous $Y^*$.

**Connection With Variational Auto-encoder**. $\mathcal{L}_{\text{INFO}}$ in Eq. 10 can be set as the variational constraint, i.e.$\mathcal{L}_{\text{INFO}}(G, G_e) := \mathrm{D}_{\text{KL}}\left(q_\theta(G_e|G) \| Q(G_e)\right)$, where $\mathrm{D}_{\text{KL}}$ denotes Kullback–Leibler divergence, $Q(G_e)$ is the prior distribution of the generated explanation graph $G_e$, and $q_\theta(G_e|G)$ is the variational approximation to $g_\theta(G_e|G)$, variational constraint drives the posterior distribution of $G_e$ generated by $g_\theta(\cdot|G)$ to its prior distribution, thus restricting the information contained in $G_e$ in the process. The overall objective is

$$\mathcal{L} = \mathbb{E}_G - \log P_{Y^*}(G_e|\theta, G) + \mathrm{D}_{\text{KL}}\left(q_\theta(G_e|G) \| Q(G_e)\right), \tag{11}$$

In this case, the objective of generative explanation shares similar spirits with the Variational Auto-encoder (VAE) [20]. Recall the optimization objective of VAE is

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{z \sim q_\phi(z|G)} - \log\left(p_\varphi(G|z)\right) + \mathrm{D}_{\text{KL}}\left(q_\phi(z|G) \| q(z)\right), \tag{12}$$

where $q_\phi$ is the encoder that maps graph $G$ into a latent space, then the decoder $p_\varphi$ recovers the original graph $G$ based on the latent representation $z$. $q(z)$ is the prior distribution of the latent representation, which is usually a Gaussian distribution. Notably, the generative explanation approach with the variational constraint as $\mathcal{L}_{\text{INFO}}$ (Eq. 11) is a variant of Variational Auto-encoder (VAE) (Eq. 12), albeit with two fundamental distinctions. Firstly, VAE aims to learn the distribution of the original graph, whereas generative explanation focuses on learning the underlying distribution of explanatory structures. Secondly, VAE constrains the distribution of latent representation $z$, while the generative explanation constrains the posterior distribution of $G_e$. These distinctions highlight the methodologies for generalizing generative models to the task of GNN explainability.

## 3.2 Taxonomies of Generative Models

In this section, we will discuss several taxonomies of generative models that have been employed in the field of GNN explainability. These models aim to learn the probability distribution of the underlying explanatory substructures by training across the entire graph dataset.

**Mask Generation (MG) [27, 49, 29, 43]** The mask generation method is to optimize a mask generator $g_\theta$ to generate the edge mask $M$ for the input graph $G$. The elements of the mask represent the importance score of the corresponding edges, which is further employed to select the important substructures of the input graph as explanations. The mask generator is usually a graph encoder followed by a multi-layer perception (MLP), which first embeds edge representations $h_{e_i}$ for each edge and then generates the sampling probability $p_i$ for edge $e_i$. The mask $m_i \in \{0, 1\}$ of $e_i$ is sampled from the Bernoulli distribution $\mathrm{Bern}(p_i)$. Since the sampling process is non-differentiable, the Gumbel-Softmax trick is usually employed for continuous relaxation as follows:

$$m_i = \sigma((\log \epsilon - \log(1-\epsilon) + \log(p_i) - \log(1-p_i))/\tau), \epsilon \sim \mathrm{Uniform}(0, 1) \tag{13}$$

where $\tau$ is the temperature, $\sigma$ is the sigmoid function. When $\tau$ goes to zero, $m_i$ is close to the discrete Bernoulli distribution. The explanation $G_e$ is obtained by applying the edge mask $M$ to the input graph $G$, i.e.$G_e = M \odot G = g_\theta(G) \odot G$, where $\odot$ is element-wise multiplication. Given an input graph $G$ and the desired label $Y^*$, the parameter $\theta$ of the mask generator $g_\theta$ is optimized by minimizing the following attribution loss:

$$\mathcal{L}_{\text{ATTR}} = -\mathbb{E}_G \log P_{Y^*}(G_e|G, \theta) \text{ with } G_e = g_\theta(G) \odot G, \tag{14}$$

which is equivalent to the cross entropy between the output probability $P_f(G_e)$ made by the base GNN $f$ and the desired label $Y^*$.

**Variational Graph Autoencoder (VGAE) [25, 28, 26]** Variational Graph Autoencoder (VGAE) [20] is a variational autoencoder for graph-structured data, where the encoder $q_\phi(\cdot)$ and the decoder $p_\theta(\cdot)$ are typically parameterized by graph neural networks. VGAE can be used to learn the distribution of the underlying explanations and thus generate explanation graphs for unseen instances. The encoder maps an input graph $G$ to a probability distribution over a latent space. The decoder then samples from the latent space and recovers an explanation graph by $G_e = p_\theta(z)$. The attribution loss of VGAE for generating explanation graphs is

$$\mathcal{L}_{\text{ATTR}} = \mathbb{E}_{z \sim q_\phi(z|G)} - \log P_{Y^*}(G_e|\theta, z, G) + \mathrm{D}_{\mathrm{KL}}(q_\phi(z|G) \| q(z)). \tag{15}$$

The standard VGAE shown in Eq. 12 aims to generate realistic graphs. On the contrary, the VGAE-based explainer maximizes the likelihood of the valid explanation graph $G_e$ for the desired label $Y^*$. The former term in Eq. 15 evaluates whether the explanation graph $G_e$ captures the most important structures for $Y^*$. It can be replaced by the cross entropy between the output probabilities $P_f(G_e)$ and $Y^*$ as CLEAR [28], or the cross entropy between the generated graph $G_e$ and ground-truth explanations as GEM [25]. The second term of KL divergence is a model-specific constraint that drives the posterior distribution $q_\phi(z|G)$ to the prior distribution $q(z)$, which is usually a Gaussian distribution.

**Generative Adversarial Networks (GAN) [24]** Generative Adversarial Network (GAN) is a type of generative model that does not include an explicit encoder component. Instead, GANs consist of a generator $g_\theta$ that creates an explanation graph $G_e = g_\theta(z)$ for $z$ sampled from a prior distribution $q(z)$, and a discriminator $d_\phi$ that distinguishes between the input graph $G$ and the generated explanation graph $G_e$. The objective function of a GAN is a min-max game in which the generator $g_\theta$ tries to minimize the function while the discriminator $d_\phi$ tries to maximize it.

$$\mathcal{L}_{\text{ATTR}} = -\mathbb{E}_{z\sim q(z)} \log P_{Y^*}(G_e|\theta, z, G) + \log d_\phi(G) + \mathbb{E}_{z\sim q(z)} \log(1 - d_\phi(g_\theta(z))), \tag{16}$$

where the first term can be the cross entropy between $P_f(g_\theta(z))$ and the desired label $Y^*$. $d_\theta(\cdot)$ denotes the probability that the discriminator predicts that the input is an explanation. GAN-based Explainer [24] was recently proposed with the first term replaced with the square of the difference between the output logit of $f$ for $G$ and $G_e$, i.e.$\mathbb{E}_{z\sim q(z)} \left(f(G) - f(g_\theta(z))\right)^2$. Once the GAN is well-trained, the generator $g_\theta$ can be employed to generate valid explanation graphs for any unseen instances, given a point in the prior distribution $q(z)$.

**Diffusion [17, 4, 38]** The Diffusion model is a class of generative models that have been used in graph generation tasks to generate realistic graphs, which contains two key components: the forward diffusion process, and the reverse denoising network. Given an original graph $G_0$, the forward diffusion process progressively generates a sequence of noisy graphs $\{G_0, G_1, \cdots, G_T\}$ with increasing levels of noise, and $G_T$ becomes pure noise. Let $\boldsymbol{A}_t = [\boldsymbol{a}_t^{ij}]_{ij}$ denote the one-hot version of the adjacency matrix of $G_t$ at timestep $t$, where $\boldsymbol{a}_t^{ij} \in \{0, 1\}^2$ is a 2-dimensional one-hot encoding of the $ij$-element. The discrete forward diffusion process is defined as $q(\boldsymbol{a}_t^{ij}|\boldsymbol{a}_{t-1}^{ij}) = \text{Cat}(\boldsymbol{a}_t^{ij}; \boldsymbol{P} = \boldsymbol{a}_{t-1}^{ij}\boldsymbol{Q}_t)$, where $\text{Cat}(\boldsymbol{x}, \boldsymbol{P})$ is a categorical distribution over the one-hot vector $\boldsymbol{x}$ with probability vector $\boldsymbol{P}$ and $\boldsymbol{Q}_t \in [0, 1]^{2\times 2}$ is a symmetric transition matrix at timestamp $t$. The forward diffusion is a Markov process that independently performs over all edges in the full adjacency matrix. Therefore, the graph-level diffusion process is $q(G_t|G_{t-1}) = \prod_{ij} q(\boldsymbol{a}_t^{ij}|\boldsymbol{a}_{t-1}^{ij})$ and $q(G_T|G_0) = \prod_{t=1}^{T} q(G_t|G_{t-1})$. The reverse denoising network $g_\theta$ learns to remove the noise and recover the target explanation graph by $G_e = g_\theta(G_t)$ for $t = 1, 2, \cdots, T$. Since an ideal $G_e$ is a compact subgraph of the original graph $G$, it is equivalent to ensuring that the complementary subgraph $G - G_e$ is close to $G$. Therefore, instead of using the generated graph to reconstruct the original graph in the standard diffusion model, we make $G - G_e$ approximate $G$ and take $G_e$ as the output explanation graph for $G$. The loss function is as follows,

$$\mathcal{L} = -\mathbb{E}_{t\in[0,T]}\mathbb{E}_{G_t\sim q(G_t|G_0)} \log P_{Y^*}(G_e|\theta, G_t, G_0) + \mathbb{E}_{t\in[0,T]}\mathbb{E}_{G_t\sim q(G_t|G_0)}\mathcal{L}_{\text{CE}}(G_0 - g_\theta(G_t), G_0). \tag{17}$$

The second term denotes the binary cross entropy loss across all elements in the adjacency matrices of $(G_0 - g_\theta(G_t)$ and $G_0$. Notably, the diffusion-based explainer naturally involves the *Information* constraint into the optimization objective, as $\mathcal{L}_{\text{CE}}$ plays a role of $\mathcal{L}_{\text{INFO}}$ that restricts the size of generated explanation graph $G_e$.

**Reinforcement Learning Approaches** Reinforcement Learning (RL) can be used to learn the distribution of underlying explanation graphs by framing the process of generating an explanation graph as a trajectory of step-wise states. Let $\tau = (s_0, \cdots, s_K) \in \mathcal{T}$ denote a trajectory $\tau$ that consists of states $s_0, \cdots, s_K$ and $\mathcal{T}$ is a set of all possible trajectories. At the $k$-th step, the state $s_k$ refers to a subgraph of the given graph, denoted as $G_k$. $G_0$ is a starting node from the given graph and $G_K$ is the terminal explanation graph. Let $a_k$ denote the action from $s_{k-1}$ to $s_k$, which is usually adding a neighboring edge to the current subgraph $G_{k-1}$. Instead of learning the distribution of the holistic explanation graphs $G_e$, reinforcement learning approaches learn the distribution of the state transition, i.e.the distribution of the selected edge to be added given the current state. The objective of these approaches is to learn a generative agent (policy network) $g_\theta(G_{k-1})$ with parameters $\theta$ that determines the next action by $a_k \sim g_\theta(G_{k-1})$. The reward function is a crucial component of reinforcement learning to address the non-differentiability issue of the sampling process within the generative agent. In the explanation task, the reward function measures the quality of the subgraph $G_k$ for the desired label $Y^*$ given the current subgraph $G_{k-1}$. RCExplainer [42] proposes to take the individual causal effect (ICE) [14] of the action $a_k$ as the reward. GFlowExplainer [22] involves the output probability $P_f(G_k)$ over the desired label $Y^*$ into the reward design. Reinforcement learning is used in conjunction with another probabilistic model to represent the distribution over the states, e.g.Markov Decision Process (MDP), Direct Acyclic Graph (DAG), etc.

- **Markov Decision Process (MDP) [50, 42].** The trajectories of states can be framed as a Markov Decision Process. The generative agent $g_\theta(G_{k-1})$ captures the sequential effect of each edge in the generating process toward a target explanation graph. The attribution loss function is as follows,

$$\mathcal{L}_{\text{ATTR}} = -\mathbb{E}_k[R(G_{k-1}, a_k)] \log P(a_k|\theta, G_{k-1}), \tag{18}$$

  where $R(G_{k-1}, a_k)$ is the reward for the action $a_k$ at state $G_{k-1}$ and $P(a_k|\theta, G_{k-1})$ is the probability of yielding $a_k$ from the distribution $g_\theta(G_{k-1})$. This loss function encourages the generative agent to attach higher probabilities to the edges that bring larger rewards, thus leading to an ideal explanation.

- **Direct Acyclic Graph (DAG) [22].** GflowNet [8] frames the trajectories of the states as a direct acyclic graph and aims to train a generative policy network where the distribution over the states is proportional to a pre-defined reward function. A concept of *flow* is introduced to measure the probability flow along the trajectories. Let $F(\tau)$ denote the flow of the trajectory $\tau$ and $F(s)$ denote the flow of the state $s$, which is the sum of all trajectory flows passing through that state. It satisfies that the inflows of a state $s_k$ equals the outflows of $s_k$. The attribution loss is as follows,

$$\mathcal{L}_{\text{ATTR}}(\tau) = \sum_{s_{k+1}\in\tau} \left( \sum_{(s_k,a_k)\to s_{k+1}} F(s_k, a_k) - \mathbb{1}_{s_{k+1}=s_K} R(s_K) - \mathbb{1}_{s_{k+1}\neq s_K} \sum_{a_{k+1}} F(s_{k+1}, a_{k+1}) \right)^2, \tag{19}$$

  where $\sum_{(s_k,a_k)\to s_{k+1}} F(s_k, a_k)$ and $\sum_{a_{k+1}} F(s_{k+1}, a_{k+1})$ denote the inflows and outflows of a state $s_{k+1}$, respectively. $\mathbb{1}$ is used to check whether $s_{k+1}$ is the terminal state $s_K$. $R(s_K)$ is the reward of the graph $G_K$ corresponding to the terminal state $s_K$. It is provable that the distribution of the terminal states generated by the agent $P(s_K|\theta)$ trained with Eq. 19 is proportional to their rewards.

Typically, reinforcement learning approaches do not rely on an explicit $\mathcal{L}_{\text{INFO}}$ to constrain the sparsity of the generated explanation graph. One common strategy is that we create trajectories $\tau = (s_0, \cdots, s_K)$ by iteratively sampling $a_k \sim g_\theta(G_{k-1})$ and stop this process once the stopping criteria are attained, e.g.$K$ achieves the pre-defined size of explanation graphs.

## 3.3 Taxonomies of Information Constraint

Only maximizing the likelihood of the explanatory subgraph with Eq. 9 typically leads to a trivial solution of the whole input graph, which is unsatisfactory. An ideal explanatory subgraph is supposed to have a small portion of the original graph information as well as be faithful for the prediction. Hence, existing methods [29, 49] introduce an additional information constraint $\mathcal{L}_{\text{INFO}}$ as a regularization term to restrict the information of the generated explanation apart from the attribution loss $\mathcal{L}_{\text{ATTR}}$. The information constraint $\mathcal{L}_{\text{INFO}}$ can be categorized as Size Constraint, Mutual Information Constraint, and Variational Constraint.

**Size Constraints [48]** The size constraint is a straightforward approach to restricting subgraph information. Given an input graph $G$ and the size tolerance $K \in (0, |G|)$, the size constraint is $|G_e| \leq K$. Here, $|\cdot|$ denotes the volume of a graph, and $K$ is an integer hyperparameter to constrain the volume of explanatory subgraphs. This constraint is first introduced in GNNExplainer [48]. Since applying the same size constraint to different graph sizes is problematic, some work employs the sparsity constraint $k \in (0, 1)$ and constrains the volume of explanatory subgraph with $|G_e| \leq k \cdot |G|$. Recent works [28, 26] further utilize a soft version of size constraint, i.e.$\mathcal{L}_{\text{INFO}} = d(G, G_e)$, where $d(G, G_e)$ is the element-wise distance between the adjacency matrices of $G$ and $G_e$.

Although the size constraint is intuitive, it has several limitations. Firstly, the topological size is insufficient in measuring the subgraph information as they ignore the information within node and edge features. Secondly, one has to choose different sparsity tolerance $k$ to achieve the best explanation performance, which is difficult due to the trade-off between the sparsity and validity of the explanations.

**Mutual Information Constraint [49]** The mutual information constraint restricts the subgraph information by reducing the relevance between the original graphs and the explanatory subgraphs. Given the graph $G$ and the explanatory subgraph $G_e$, the mutual information constraint is formulated as:

$$\mathcal{L}_{\text{INFO}} = MI(G, G_e) = \mathbb{E}_{p(G)}\mathbb{E}_{p(G_e|G)} \log \frac{p(G_e|G)}{p(G_e)}. \tag{20}$$

Here $MI(x, y)$ is the mutual information of two random variables. Minimizing Eq. 20 reduces the relevance between $G$ and $G_e$. Thus, the explanation generator tends to leverage limited input graph information to generate the explanation subgraph. Compared with the size constraints, the mutual information constraint is more fundamental in information measurement and flexible to different graph sizes. However, mutual information is intractable to compute, making the constraint impractical to use. One solution is resorting to computationally expensive estimation techniques, such as the Donsker-Varadhan representation of mutual information [7, 49].

**Variational Constraint [29]** Since the mutual information constraint is intractable, the variational constraint is proposed by deriving a tractable variational upper bound of the mutual information constraint. One can plug a prior distribution $q(G_e)$ into $MI(G, G_e)$ as the variational approximation to $p(G_e|G)$:

$$\begin{aligned} MI(G, G_e) &= \mathbb{E}_{p(G)}\mathbb{E}_{p(G_e|G)} \log \frac{p(G_e|G)}{p(G_e)} = \mathbb{E}_{p(G)}\mathbb{E}_{p(G_e|G)} \log \frac{p(G_e|G)}{q(G_e)} - \mathrm{D}_{\mathrm{KL}}(q(G_e) \| p(G_e)) \\ &\leq \mathbb{E}_{p(G)}\mathbb{E}_{p(G_e|G)} \log \frac{p(G_e|G)}{q(G_e)} := \mathcal{L}_{\text{VC}}. \end{aligned} \tag{21}$$

Here, the inequality is due to the non-negative nature of the Kullback–Leibler (KL) divergence. The posterior distribution $p(G_e|G) = \prod_{i=1}^{N} p(e_i|\theta)$ is factorized into the multiplication of the marginal distributions of edge sampling $p(e_i|\theta)$, which is parameterized with the generative explanation network $\theta$. The prior distribution $q(G_e) = \prod_{i=1}^{N} q(e_i)$ is factorized into the prior distributions of edge sampling $q(e_i)$. $N$ is the total edge number. In practice, $q(e_i)$ is usually chosen as the Bernoulli distribution or Gaussian distribution. Thus, the variational constraint is an upper bound of the mutual information $MI(G, G_e)$ that can be simplified as

$$\mathcal{L}_{\text{INFO}} = \mathcal{L}_{\text{VC}} = \sum_{i=1}^{N} \mathrm{D}_{\mathrm{KL}}(p(e_i|\theta) \| q(e_i)). \tag{22}$$

## 3.4 Extension of Explanation Scenarios

**Counterfactual Explanation** Most explanation methods focus on discovering the prediction-relevant subgraph to explain GNNs based on Eq. 9. Although these methods can highlight the important substructures for the predictions, they cannot answer the *counterfactual* problem such as: "Will the removal of certain substructure lead to prediction change of GNNs?" Counterfactual explanations provide insightful information on how the model prediction would change if some event had occurred differently, which is crucial in some real-world scenarios, e.g.drug design and molecular modification [30, 18, 45]. Given an input graph $G$, the goal of the generative counterfactual explanation is to train a subgraph generator $g_\theta$ to generate a minimal substructure of the input. If the substructure is removed, the prediction of GNNs will change the most. Formally, the objective for counterfactual explanations is:

$$\mathcal{L}_{\text{CF}} = -\log P_{Y^*}(G_{ce}|\theta, G) + \mathcal{L}_{\text{INFO}}(G, \overline{G_{ce}}) \tag{23}$$

Here, $\overline{G_{ce}}$ denotes the generated substructure, which is also the modification applied to $G$ to obtain a counterfactual explanation $\overline{G_{ce}} = G - G_{ce}$. $\mathcal{L}_{\text{INFO}}$ is a regularization term that constrains the information amount contained in $\overline{G_{ce}}$ to be minimal compared with the input graph $G$.

**Connection to Graph Adversarial Attack**. The counterfactual explanation methods can capture the vulnerability of GNN's prediction since the counterfactual explanation subgraph leads to the prediction change. This problem setting is similar to graph adversarial attacks as they both aim to alter the prediction behavior of the pre-trained GNN by modifying testing graphs. Recall that graph adversarial attacks modify the node features or graph structures to decrease the average performance of a pre-trained GNN. However, the explanation methods change the prediction of each testing sample by instance-level subgraph deletion instead of decreasing the overall testing performance after a one-time graph modification [2].

**Connection to Graph Out-of-distribution Generalization**. Deep learning models have been found to rely on spurious patterns in the input to make predictions. These patterns are often unstable under distribution shifts, leading to a drop in performance when testing on out-of-distribution (OOD) data. To address this issue, counterfactual augmentation has been proposed as an effective method for improving the OOD generalization ability of deep learning models. This technique involves minimally modifying the training data to change their labels and training the model with both the original and counterfactually augmented data. For graphs, counterfactually augmented subgraphs can be generated by removing subgraphs to create the complementary subgraph, which is a natural form of counterfactual augmentation. However, this approach has received less attention in the context of graph neural networks, presenting an avenue for future research.

**Model-level Explanation** The goal of model-level explanation in the context of GNNs is to identify important graph patterns that contribute to the decision boundaries of the model. Unlike instance-level explanation, model-level explanation provides insights into the general behavior of the model across a range of input graphs with the same predicted label. A brute-force approach to finding these patterns is to mine the subgraphs that commonly appear in graphs with the same predicted label. However, this is computationally expensive due to the exponentially large search space. Recently, generative methods have been proposed to generate model-level explanations, such as reinforcement learning [50] and probabilistic generative models [43].

In these approaches, a generator function $g_\theta(\cdot)$ is used to generate the model-level explanation $G_m$ for a given predicted label $Y^*$ based on a set of graphs $\mathcal{G}$ via $G_m \sim g_\theta(\mathcal{G}, Y^*)$. The optimization objective for the generator function is to minimize the negative log-likelihood of the explanation given the graphs, while also ensuring that the generated explanation is a compact and recurrent substructure in the set of input graphs:

$$\mathcal{L} = -\log P_{Y^*}(G_m|\theta, \mathcal{G}) + \mathcal{L}_{\text{INFO}}(G_m, \mathcal{G}). \tag{24}$$

The first term in Eq. 24 measures whether $G_m$ captures the most determinant graph patterns for the prediction of $Y^*$. The generator $g_\theta(\cdot)$ can be modeled by other applicable generative models discussed in Sec. 3.2. $\mathcal{L}_{\text{INFO}}(G_m, \mathcal{G})$ ensures that $G_m$ is a compact substructure that commonly appears in $\mathcal{G}$.

## 4 Method Taxonomy

The information constraints $\mathcal{L}_{\text{INFO}}$ in Sec. 3.3 and the attribution constraints $\mathcal{L}_{\text{ATTR}}$in Sec. 3.2 can be combined to construct an overall optimization objective for GNN explainability. We provide a comprehensive comparison

Table 2: A comprehensive summary of existing generative explanation methods for Graph Neural Networks. RL-MDP denotes the reinforcement learning approach based on Markov Decision Process and RL-DAG denotes the reinforcement learning approach based on Direct Acyclic Graph.

| **Method** | **Generator** | **Information Constraint** | **Level** | **Scenario** | **Output** |
|---|---|---|---|---|---|
| PGExplainer [27] | Mask Generation | size | instance | factual | E |
| GIB [49] | Mask Generation | mutual information | instance | factual | N |
| GSAT [29] | Mask Generation | variational | instance | factual | E |
| GNNInterpreter [43] | Mask Generation | size | model | factual | N / E / NF |
| GEM [25] | VGAE | size | instance | factual | E |
| CLEAR [28] | VGAE | size | instance | counterfactual | E / NF |
| OrphicX [26] | VGAE | variational & size | instance | factual | E |
| D4Explainer | Diffusion | size | instance & model | counterfactual | E |
| GANExplainer [24] | GAN | - | instance | factual | E |
| RCExplainer [42] | RL-MDP | size | instance | factual | SUBGRAPH |
| XGNN [50] | RL-MDP | size | model | factual | SUBGRAPH |
| GFlowExplainer [22] | RL-DAG | size | instance | factual | SUBGRAPH |

and summary of existing generative explanation methods and their corresponding generators and information constraints in Table 2. Most existing approaches focus on instance-level factual explanations, while CLEAR [28] focuses on counterfactual explanation and D4Explainer is applicable for both counterfactual and model-level explanations. GIB [49] proposes to deploy mutual information between the generated explanation graph and the original graph as the information constraint, while GSAT [29] utilizes the variational constraint. We further compare the outputs of these approaches (the last column in Table 2), where E denotes outputting edge importance with continuous values, N denotes node importance with continuous values, NF denotes the importance of node features and SUBGRAPH denotes hard masks for discrete explanatory subgraphs.

## 5 Evaluation

### 5.1 Experimental setting

**Datasets** We evaluate the explainability methods on both synthetic and real-world datasets in different domains, including MUTAG, BBBP, MNIST, BA-2Motifs and BA-MultiShapes. **BA-2Motifs** [27] is a synthetic dataset with binary graph labels. The house motif and the cycle motif give class labels and thus are regarded as ground-truth explanations for the two classes. **BA-MultiShapes** [5] is a more complicated synthetic dataset with multiple motifs. Class 0 indicates that the instance is a plain BA graph or a BA graph with a house, a grid, a wheel, or the three motifs together. On the contrary, Class 1 denotes BA graphs with two of these three motifs. **MUTAG** is a collection of $\sim$3000 nitroaromatic compounds and it includes binary labels on their mutagenicity on Salmonella typhimurium. The chemical fragments -NO2 and -NH2 in mutagen graphs are labeled as ground-truth explanations [27]. The Blood–brain barrier penetration **BBBP** dataset includes binary labels for over 2000 compounds on their permeability properties. In molecular datasets, node features encode the atom type and edge features encode the type of bonds that connect atoms. **MNIST75sp** contains graphs that are converted from images in MNIST [21] using superpixels. In these graphs, the nodes represent the superpixels, and the edges are determined by the spatial proximity between the superpixels. The coordinates and intensity of the corresponding superpixel construct the node features. Dataset statistics are summarized in Table 3.

**GNN models** For each dataset, we first train a GNN model. We have tested four GNN models: GCN [19], GIN[15], GAT [37], and GraphTransformer [35]. We only display results for the GraphTransformer model for the real-world datasets and the GIN model for the synthetic datasets since they give the highest accuracy scores

| | MUTAG | BBBP | MNIST75sp | BA-2Motifs | BA-MultiShapes |
|---|---|---|---|---|---|
| # graphs | 2,951 | 2,039 | 70,000 | 1,000 | 1,000 |
| # node features | 14 | 9 | 5 | 1 | 10 |
| # edge features | 1 | 3 | 1 | 1 | 1 |
| Avg # nodes | 30 | 24 | 67 | 25 | 40 |
| Avg # edges | 61 | 52 | 541 | 51 | 87 |
| Avg degree | 2.0 | 2.1 | 7.9 | 2.0 | 2.2 |
| # classes | 2 | 2 | 10 | 2 | 2 |
| GNN performance | 0.94 | 0.92 | 0.96 | 1.00 | 0.71 |

Table 3: Dataset statistics and accuracy performance of the GNN model on the test set

on the test sets respectively. GraphTransformer and GIN give high accuracy on the real-world and synthetic datasets respectively, with a reasonable training time and fast convergence. Unlike GCN, GraphTransformer and GIN have also the advantage of taking edge features, extending their use to more complex graph datasets. The network structure of the GNN model for graph classification is a series of 3 layers with ReLU activation, followed by a max pooling layer to get graph representations before the final fully connected layer. We adopt the Adam optimizer with an initial learning rate of 0.001. We split train/validation/test with 80/10/10$\%$ for all datasets. Each model is trained for 200 epochs with an early stop. The accuracy performances of GNN models are shown in Table 3. The results show that the designed GNN models are sufficiently powerful for graph classifications on both synthetic and real-life datasets.

**Explainability methods** We compare non-generative methods: Saliency [6], Integrated Gradient [36], Occlusion [53], Grad-CAM [32], GNNExplainer [48], PGMExplainer [39], and SubgraphX [52], with generative ones: PGExplainer [27], GSAT [29], GraphCFE (CLEAR) [28], D4Explainer and RCExplainer [42]. Following GraphFramEx [3], we define an explanation as an edge mask on the existing edges in the initial graph to be explained. First, this constraint facilitates the comparison of very diverse explainability methods. Moreover, in the context of our study, all datasets are expected to be explained by some entities that already exist in the initial graphs, i.e.motifs in synthetic datasets and groups of atoms in molecular datasets. We follow the original setting to train PGExplainer, GSAT, and RCExplainer. We implement the diffusion-based explainer as introduced in Sec. 3.2, and name it D4Explainer. D4Explainer generates an explanatory graph that can contain additional edges that are not in the initial graph. To keep consistent, we retrieve the common edges with the initial graph to evaluate D4Explainer in this work. GraphCFE is a simplified version of CLEAR [28] without the causality component, which is an explainability method for counterfactual explanations. Indeed, the causal models introduced in [28] are constructed from simulations because it is hard to get the ground-truth causal model from datasets. Since we ignore the existence of any causal model in our datasets, we decide not to focus on the causality and use only the CLEAR-VAE backbone, i.e.GraphCFE, in this work. We retrieve the important edges by subtracting the counterfactual explanation generated by GraphCFE from the initial graph. The remaining edges have weights of 1, while the rest have weights of 0.

**Metrics** To evaluate the explainability methods, we use the systematic evaluation framework GraphFramEx [3]. We evaluate the methods on the faithfulness measure $fidelity{-}_{acc}$, which is defined as

$$fidelity{-}_{acc} = \frac{1}{N}\sum_{i=1}^{N}\left|\mathbb{1}(\hat{Y}_f(G^i) = Y^i) - \mathbb{1}(\hat{Y}_f(G_e^i) = Y^i)\right|,$$

where $G^i$ and $G_e^i$ denote the initial graph and the explanatory graph, respectively. $fidelity{-}_{acc}$ measures if the generated explanatory subgraph is faithful to the initial graph, i.e.leads to the same GNN prediction.

## 5.2 Instance-level explanations

**Faithfulness** We conducted a comprehensive comparison of the faithfulness between generative and non-generative methods using three real-world datasets (BBBP, MUTAG, and MNIST) and two synthetic datasets (BA-2Motifs and BA-MultiShapes). The results, depicted in Figure 1, indicate that generative methods are generally performing the same or better than non-generative methods. Specifically, for MNIST, generative methods outperform non-generative methods across the board. In the cases of MUTAG and BA-2Motifs, the generative methods RCExplainer, GraphCFE, and GSAT closely follow Grad-CAM and Occlusion in terms of faithfulness. Regarding BBBP and BA-MultiShapes, both generative and non-generative methods exhibit similar results. Consequently, generative methods achieve state-of-the-art performance on benchmark graph datasets. Furthermore, we demonstrate that generative methods possess additional desirable properties, such as efficiency and generalization capacity, which make them more appealing than non-generative methods.

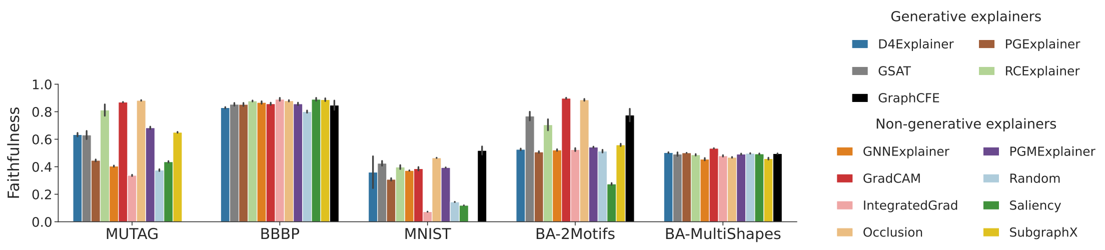

Figure 1: Faithfulness of explainability methods. On the y-axis, we report the faithfulness computed as $1 - fidelity-_{acc}$. On the x-axis, generative methods are always on the left-hand side of the methods (the bars). If the score is close to 1, the explanation is very faithful. The score is averaged over all explanations with less than 20 edges to enforce sparse and human-intelligible explanations.

**Efficiency** To measure the efficiency of explainability methods, we report the computation time to produce an explanation for a new instance in Figure 2. Comparing generative methods with other learnable methods (e.g.GNNExplainer, PGMExplainer) in Figure 2, we observe that once the model is trained, generative explainability methods require shorter inference time than non-generative ones in general. The time is reported in logarithmic scale and generative methods always have inference times of the order of $10^0$ or less, except for the case of RCExplainer for MNIST. The advantage of shorter inference time is especially pronounced on large-scale datasets, e.g.MNIST. We also report the time required to train a generative model from scratch in Table 4.

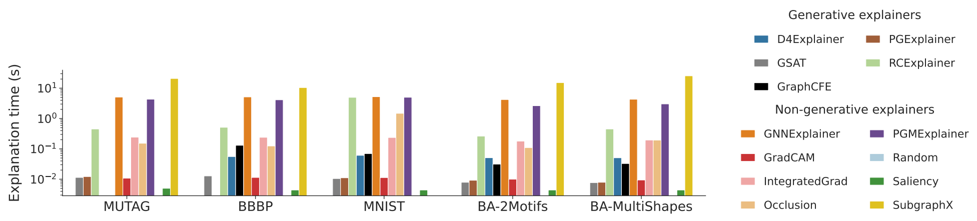

Figure 2: Inference time of explainability methods to explain one single graph. The computation time is averaged over 100 explanations over 5 seeds and reported in logarithmic scale.

**Generalization** To compare generative and non-generative explainability methods on their generalization capacity, we split the datasets into seen and unseen data. The split ratio is 90/10%. We further split the seen data into training, validation, and test set. The GNN model and the generative explainability methods are trained on

| | **D4Explainer** | **GraphCFE** | **GSAT** | **PGExplainer** | **RCExplainer** |
|---|---|---|---|---|---|
| **BA-2Motifs** | 475.3 | 320.9 | 23.1 | 11.6 | 194.0 |
| **BA-MultiShapes** | 309.3 | 211.8 | 20.0 | 17.2 | 251.0 |
| **BBBP** | 385.6 | 1350.0 | - | 26.0 | 303.4 |
| **MNIST** | 934.6 | 929.5 | 41.4 | 28.6 | 3271.0 |
| **MUTAG** | 253.1 | - | 79.8 | 27.7 | 434.6 |
| Mean | 471.6 | 703.1 | 41.1 | 22.2 | 890.8 |

Table 4: Training times (s) of the generative methods with 1 GPU (Nvidia GeForce RTX 2080)

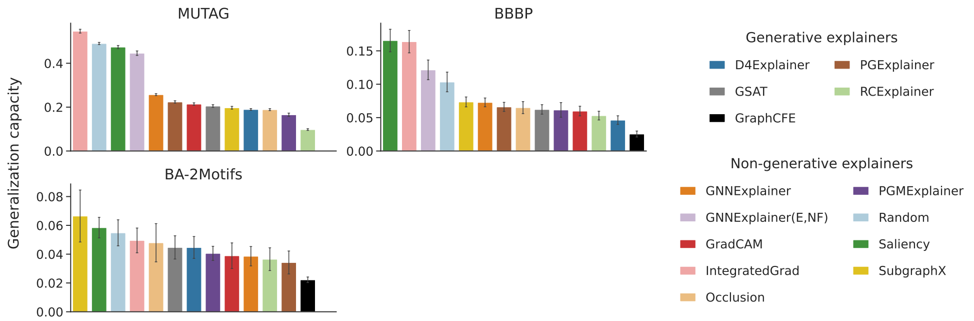

Figure 3: Generalization capacity of explainability methods is computed by subtracting the performance on data seen during training and the performance on unseen data. The lower the discrepancy reported on the y-axis, the better the method can generalize to unseen data. GNNExplainer indicates the explanations at only edge-level and GNNExplainer(E,NF) represents the explanations for both edges and node features.

the seen data. For non-generative methods, we explain 100 graphs from the seen dataset. Then, we test the trained methods on the unseen data. In Figure 3, we report the scores discrepancies between the test set of the seen data and the 10% unseen data for each explainability method. We also visualize the standard error on the five random seeds in Figure 3. Methods with higher absolute score discrepancies cannot generalize well to unseen data, while the ones with lower score discrepancies have a powerful generalization capacity. We can observe from Figure 3 that generative explainability methods have lower scores than non-generative methods across three datasets in general, which demonstrates the better generalization capacity.

## 6 Conclusion

In this paper, we present a comprehensive review of explanation methods for Graph Neural Networks (GNNs) from the perspective of graph generation. By proposing a unified optimization objective for generative explanation methods, encompassing Attribution and Information constraints, we provide a framework to analyze and compare existing approaches. Our study reveals shared characteristics and distinctions among current methods, laying the foundation for future advancements in the field. Moreover, we highlight the advantages and limitations of different approaches in terms of explanation performance, efficiency, and generalizability through empirical results. Notably, generative-based approaches demonstrate enhanced efficiency and generalizability compared to instance-dependent methods. Overall, our work contributes to the advancement of transparent and trustworthy graph-based models, paving the way for improved outcomes in various applications through better feature extraction and understanding of complex graph-structured data.

## References

[1] Chirag Agarwal, Owen Queen, Himabindu Lakkaraju, and Marinka Zitnik. Evaluating explainability for graph neural networks. Scientific Data, 10(1):144, 2023.

[2] Ulrich Aïvodji, Alexandre Bolot, and Sébastien Gambs. Model extraction from counterfactual explanations. arXiv preprint arXiv:2009.01884, 2020.

[3] Kenza Amara, Rex Ying, Zitao Zhang, Zhihao Han, Yinan Shan, Ulrik Brandes, Sebastian Schemm, and Ce Zhang. Graphframex: Towards systematic evaluation of explainability methods for graph neural networks. arXiv preprint arXiv:2206.09677, 2022.

[4] Maximilian Augustin, Valentyn Boreiko, Francesco Croce, and Matthias Hein. Diffusion visual counterfactual explanations. arXiv preprint arXiv:2210.11841, 2022.

[5] Steve Azzolin, Antonio Longa, Pietro Barbiero, Pietro Liò, and Andrea Passerini. Global explainability of gnns via logic combination of learned concepts. arXiv preprint arXiv:2210.07147, 2022.

[6] Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks. CoRR, abs/1905.13686, 2019.

[7] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: mutual information neural estimation. arXiv preprint arXiv:1801.04062, 2018.

[8] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In NeurIPS, 2021.

[9] Pietro Bongini, Monica Bianchini, and Franco Scarselli. Molecular generative graph neural networks for drug discovery. Neurocomputing, 450:242–252, 2021.

[10] Anshika Chaudhary, Himangi Mittal, and Anuja Arora. Anomaly detection using graph neural networks. In 2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon), pages 346–350. IEEE, 2019.

[11] Dawei Cheng, Fangzhou Yang, Sheng Xiang, and Jin Liu. Financial time series forecasting with multi-modality graph neural network. Pattern Recognition, 121:108218, 2022.

[12] Enyan Dai, Tianxiang Zhao, Huaisheng Zhu, Junjie Xu, Zhimeng Guo, Hui Liu, Jiliang Tang, and Suhang Wang. A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. arXiv preprint arXiv:2204.08570, 2022.

[13] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In The world wide web conference, pages 417–426, 2019.

[14] Madelyn Glymour, Judea Pearl, and Nicholas P Jewell. Causal inference in statistics: A primer. John Wiley & Sons, 2016.

[15] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks, 2020.

[16] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. CoRR, abs/2001.06216, 2020.

[17] Guillaume Jeanneret, Loïc Simon, and Frédéric Jurie. Diffusion models for counterfactual explanations. In Proceedings of the Asian Conference on Computer Vision, pages 858–876, 2022.

[18] José Jiménez-Luna, Francesca Grisoni, and Gisbert Schneider. Drug discovery with explainable artificial intelligence. Nature Machine Intelligence, 2(10):573–584, 2020.

[19] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. CoRR, abs/1609.02907, 2016.

[20] Thomas N Kipf and Max Welling. Variational graph auto-encoders. arXiv preprint arXiv:1611.07308, 2016.

[21] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[22] Wenqian Li, Yinchuan Li, Zhigang Li, Jianye Hao, and Yan Pang. Dag matters! gflownets enhanced explainer for graph neural networks. arXiv preprint arXiv:2303.02448, 2023.

[23] Yiqiao Li, Jianlong Zhou, Sunny Verma, and Fang Chen. A survey of explainable graph neural networks: Taxonomy and evaluation metrics. arXiv preprint arXiv:2207.12599, 2022.

[24] Yiqiao Li, Jianlong Zhou, Boyuan Zheng, and Fang Chen. Ganexplainer: Gan-based graph neural networks explainer. arXiv preprint arXiv:2301.00012, 2022.

[25] Wanyu Lin, Hao Lan, and Baochun Li. Generative causal explanations for graph neural networks. In International Conference on Machine Learning, pages 6666–6679. PMLR, 2021.

[26] Wanyu Lin, Hao Lan, Hao Wang, and Baochun Li. Orphicx: A causality-inspired latent variable model for interpreting graph neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13729–13738, 2022.

[27] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. In NeurIPS, 2020.

[28] Jing Ma, Ruocheng Guo, Saumitra Mishra, Aidong Zhang, and Jundong Li. Clear: Generative counterfactual explanations on graphs. arXiv preprint arXiv:2210.08443, 2022.

[29] Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In International Conference on Machine Learning, pages 15524–15543. PMLR, 2022.

[30] Tri Minh Nguyen, Thomas P Quinn, Thin Nguyen, and Truyen Tran. Counterfactual explanation with multi-agent reinforcement learning for drug target prediction. arXiv preprint arXiv:2103.12983, 2021.

[31] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10772–10781, 2019.

[32] Phillip E. Pope, Soheil Kolouri, Mohammad Rostami, Charles E. Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In CVPR, pages 10772–10781, 2019.

[33] Mario Alfonso Prado-Romero, Bardh Prenkaj, Giovanni Stilo, and Fosca Giannotti. A survey on graph counterfactual explanations: Definitions, methods, evaluation. arXiv preprint arXiv:2210.12089, 2022.

[34] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for NLP with differentiable edge masking. CoRR, abs/2010.00577, 2020.

[35] Yunsheng Shi, Zhengjie Huang, Wenjin Wang, Hui Zhong, Shikun Feng, and Yu Sun. Masked label prediction: Unified massage passing model for semi-supervised classification. CoRR, abs/2009.03509, 2020.

[36] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In ICML, volume 70, pages 3319–3328, 2017.

[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[38] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. arXiv preprint arXiv:2209.14734, 2022.

[39] Minh N. Vu and My T. Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. In NeurIPS, 2020.

[40] Jianian Wang, Sheng Zhang, Yanghua Xiao, and Rui Song. A review on graph neural network methods in financial applications. arXiv preprint arXiv:2111.15367, 2021.

[41] Xiang Wang, Ying-Xin Wu, An Zhang, Xiangnan He, and Tat-Seng Chua. Towards multi-grained explainability for graph neural networks. In Proceedings of the 35th Conference on Neural Information Processing Systems, 2021.

[42] Xiang Wang, Yingxin Wu, An Zhang, Fuli Feng, Xiangnan He, and Tat-Seng Chua. Reinforced causal explainer for graph neural networks. IEEE Trans. Pattern Anal. Mach. Intell., 2022.

[43] Xiaoqi Wang and Han-Wei Shen. Gnninterpreter: A probabilistic generative model-level explanation for graph neural networks. arXiv preprint arXiv:2209.07924, 2022.

[44] Dana Warmsley, Alex Waagen, Jiejun Xu, Zhining Liu, and Hanghang Tong. A survey of explainable graph neural networks for cyber malware analysis. In 2022 IEEE International Conference on Big Data (Big Data), pages 2932–2939. IEEE, 2022.

[45] Geemi P Wellawatte, Aditi Seshadri, and Andrew D White. Model agnostic generation of counterfactual explanations for molecules. Chemical science, 13(13):3697–3705, 2022.

[46] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. Drug Discovery Today: Technologies, 37:1–12, 2020.

[47] Bingzhe Wu, Jintang Li, Junchi Yu, Yatao Bian, Hengtong Zhang, CHaochao Chen, Chengbin Hou, Guoji Fu, Liang Chen, Tingyang Xu, et al. A survey of trustworthy graph learning: Reliability, explainability, and privacy protection. arXiv preprint arXiv:2205.10014, 2022.

[48] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In NeurIPS, pages 9240–9251, 2019.

[49] Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. In International Conference on Learning Representations, 2021.

[50] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. XGNN: towards model-level explanations of graph neural networks. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, KDD, pages 430–438, 2020.

[51] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. CoRR, 2020.

[52] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. ArXiv, 2021.

[53] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13, pages 818–833. Springer, 2014.

[54] He Zhang, Bang Wu, Xingliang Yuan, Shirui Pan, Hanghang Tong, and Jian Pei. Trustworthy graph neural networks: Aspects, methods and trends. arXiv preprint arXiv:2205.07424, 2022.

[55] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. AI open, 1:57–81, 2020.