A Survey on Explainability of Graph Neural Networks

Jaykumar Kakkad[*], **Jaspal Jannu**[*]
University of Illinois Chicago, USA
{jkakka4,jjannu2}@uic.edu

Kartik Sharma
Georgia Institute of Technology, USA
ksartik@gatech.edu

Charu Aggarwal
IBM T. J. Watson Research Center, USA
charu@us.ibm.com

Sourav Medya
University of Illinois, Chicago, USA
medya@uic.edu

Abstract

Graph neural networks (GNNs) are powerful graph-based deep-learning models that have gained significant attention and demonstrated remarkable performance in various domains, including natural language processing, drug discovery, and recommendation systems. However, combining feature information and combinatorial graph structures has led to complex non-linear GNN models. Consequently, this has increased the challenges of understanding the workings of GNNs and the underlying reasons behind their predictions. To address this, numerous explainability methods have been proposed to shed light on the inner mechanism of the GNNs. Explainable GNNs improve their security and enhance trust in their recommendations. This survey aims to provide a comprehensive overview of the existing explainability techniques for GNNs. We create a novel taxonomy and hierarchy to categorize these methods based on their objective and methodology. We also discuss the strengths, limitations, and application scenarios of each category. Furthermore, we highlight the key evaluation metrics and datasets commonly used to assess the explainability of GNNs. This survey aims to assist researchers and practitioners in understanding the existing landscape of explainability methods, identifying gaps, and fostering further advancements in interpretable graph-based machine learning.

1 Introduction

Recent years have seen a tremendous rise in the use of Graph Neural Networks (GNNs) for real-world applications, ranging from healthcare [129, 130], drug design [111, 52, 91], recommender systems [9], and fraud detection [79]. Predictions made in these domains have a substantial impact and therefore require to be highly trustworthy. In the realm of deep learning, one effective approach to enhance trust in these predictions is to provide an explanation supporting them [87]. These explanations elucidate the model's predictions for human understanding and can be generated through various methods. For instance, they may involve identifying important substructures within the input data [56, 82, 122], providing additional examples from the training data [12], or constructing counterfactual examples by perturbing the input to produce a different prediction outcome [55, 93, 9].

The interpretability of deep learning models is influenced by the characteristics of the input domain as the content and the complexity of explanations can vary depending on the inputs. When it comes to explaining predictions made by graph neural networks (GNNs), several challenges arise. First, since graphs are combinatorial data structures, finding important substructures by evaluating different combinations that maximize a certain prediction becomes difficult. Second, attributed graphs contain

[*]Both authors contributed equally.

Preprint. Under review.

both node attributes and edge connectivity, which can influence the predictions and they should be considered together in explanations. Third, explanations must be adaptable to different existing GNN architectures. Lastly, explanations for the local tasks (e.g., node or edge level) may differ from those for global tasks (e.g., graph level). Due to these challenges, explaining graph neural networks is non-trivial and a large variety of methods have been proposed in the literature to tackle it [115, 69, 107, 77, 5, 97, 56, 119, 93, 4, 73]. With the increasing use of GNNs in critical applications such as healthcare and recommender systems, and the consequent rise in their explainability methods, we provide an updated survey of the explainability of GNNs. Additionally, we propose a novel taxonomy that categorizes the explainability methods for GNNs, providing a comprehensive overview of the field.

Existing surveys on GNN explainability predominantly focus on either factual methods [120, 13, 47] or counterfactual methods [26] but not both. These surveys thus lack a comprehensive overview of the different methods in the literature by limiting the discussion to specific methods [120, 47] or only discussing them under the broad umbrella of trustworthy GNNs [13, 104]. Our survey aims to bridge this gap by providing a comprehensive and detailed summary of existing explainability methods for GNNs. We include both factual as well as counterfactual methods of explainability in GNNs. To enhance clarity and organization, we introduce a novel taxonomy to categorize these methods for a more systematic understanding of their nuances and characteristics.

1.1 Graph Neural Networks

Graph neural networks (GNNs) have been used to learn powerful representations of graphs [42, 96, 28]. Consider a graph G given by $G = (V, E)$, where V denotes the set of n nodes and E denotes the set of m edges. We can create an adjacency matrix $\mathbf{A} \in [0, 1]^{n \times n}$ such that $A_{ij} = 1$ if $(i, j) \in E$ and 0 otherwise. Each node may have attributes, given by the matrix $\mathbf{X} \in \mathbb{R}^{n \times F}$ such that each row i stores the F-dimensional attribute vector for node i. A GNN model $\mathcal{M}$ embeds each node $v \in V$ into a low-dimensional space $\mathbf{Z} : \mathbb{R}^{n \times d}$ by following this message passing rule for k steps as

$$\mathbf{Z}_v^{(k+1)} = \textsc{Update}_{\Phi}(\mathbf{Z}_v^{(k)}, \textsc{Agg}(\{\textsc{Msg}_{\Theta}(\mathbf{Z}_v^{(k)}, \mathbf{Z}_u^{(k)}) : (u, v) \in E\})), \tag{1}$$

such that $\mathbf{Z}^0 = \mathbf{X}$ and $\mathbf{Z} := \mathbf{Z}^{(k)}$. Different instances of the update $\textsc{Update}_{\Phi}$, aggregation $\textsc{Agg}_{\Phi}$ and message generator $\textsc{Msg}_{\Theta}$ functions give rise to different GNN architectures. For example, GCN [42] has an identity message, a mean aggregation, and weighted update functions, while GAT [96] learns an attention-based message generation instead. These embeddings are trained for a specific task $\mathcal{T}$ that can be either supervised (e.g., node classification, graph classification, etc.) or unsupervised (e.g., self-supervised link prediction, clustering, etc.).

1.2 Explainability in ML

Problem 1 (Explainability [8]) *Consider a supervised task $\mathcal{T}$ with the aim of learning a mapping from $\mathcal{X}$ to $\mathcal{Y}$, and a model $\mathcal{M}$ trained for this task. Given a set of $(\mathbf{x}, y)$ pairs $\subseteq (\mathcal{X}, \mathcal{Y})$ and the model $\mathcal{M}$, generate an explanation $\mathbf{e}$ from a given set $\mathcal{D}_E$ such that $\mathbf{e}$ "explains" the prediction $\hat{y} = \mathcal{M}(\mathbf{x})$.*

These explanations can be either *local* to a single test input $(\mathbf{x}, y)$ or *global* when they explain prediction over a specific dataset $\mathcal{D}' \subseteq (\mathcal{X}, \mathcal{Y})$. Further, the explanation can be generated either *post-hoc* (*i.e.*, after the model training) or *ante-hoc* where the model itself is *self-interpretable*, *i.e.*, it explains its predictions. With some exceptions, post-hoc explanations usually consider a black-box access to the model while self-interpretable methods update the model architecture and/or training itself. We can further differentiate the explanation methods based on their content, *i.e.*, the explanation set $\mathcal{D}_E$. *Local explanations* only consider the local neighborhood of the given data instance while *global explanations* are concerned about the model's overall behavior and thus, searches for patterns in the model's predictions. On the other hand, explanations can also be *counterfactual*, where the aim is to explain a prediction by providing a contrasting example that changes it.

2 Overview

With the widespread adoption of GNNs across various applications, the demand for explaining their predictions has grown substantially. Moreover the GNN-based models are becoming more complex [62]. Recently, the community has witnessed a surge in efforts dedicated to the explainability of GNNs. These methods exhibit variations in terms of explanation types, utilization of model information, and training procedures, among other factors. We organize and categorize these methods to develop a deeper understanding of the existing works and provide a broad picture of their applicability in different scenarios.

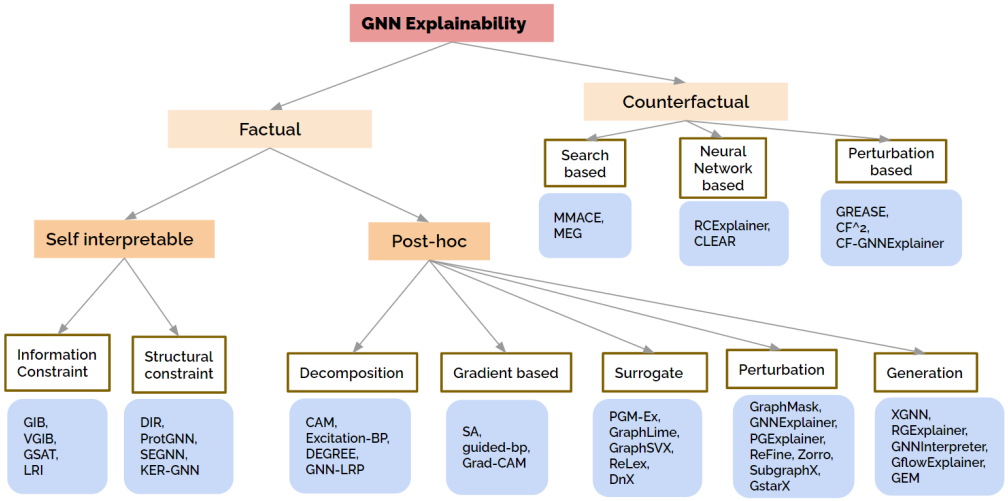


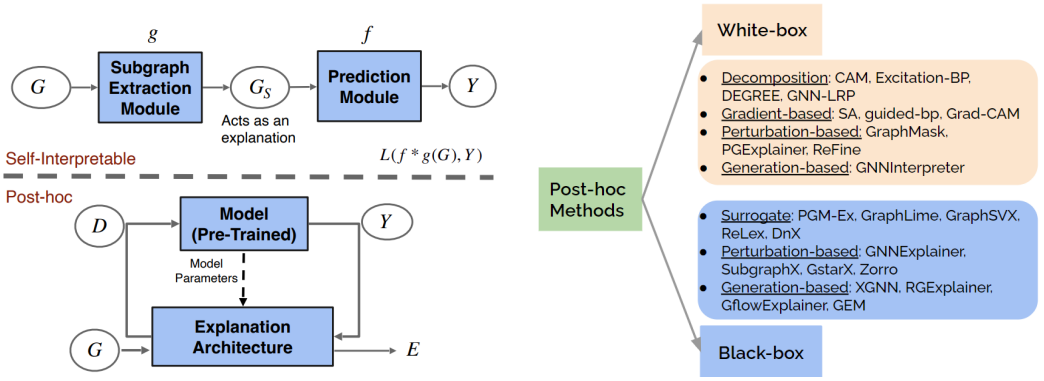
Figure 1: **Overview of the Schema. (1) Factual.** Information constraints: GIB [118], VGIB [116], GSAT [69], LRI [70]; Structural Constraints: DIR [107], ProtGNN [125], SEGNN [12], KER-GNN [21]; Decomposition: CAM [77], Excitation-BP [77], DEGREE [22], GNN-LRP [84]; Gradient-based: SA [5] , Guided-BP [5] , Grad-CAM [77]; Surrogate: PGM-Ex [56], GraphLime [34], GraphSVX [17], ReLex [124], DnX [75]; Perturbation-based: GNNExplainer[115], GraphMask [82], PGExplainer [56], ReFine [100], ZORRO [23], SubgraphX [122], GstarX [123]; Generation: XGNN [119], RGExplainer [85], GNNInterpreter [101], GFlowExplainer [46], GEM [49]; **(2) Counterfactual.** Search-based: MMACE [102] , MEG [73]; Neural Network-based: RCExplainer [4], CLEAR [57]; Perturbation-based: GREASE [9], CF2 [93], CF-GNNexplainer [55]

Main Schema: Factual and Counterfactual Methods. Figure 1 provides an overview of the broad categorization of the existing works. Based on the type of explanations, we first make two broad categories: (1) Factual and (2) Counterfactual. Factual methods aim to find an explanation in the form of input features with the maximum influence over the prediction. These explanations can be a set of either node features or a substructure (set of nodes/edges) or both. On the other hand, counterfactual methods provide an explanation by finding the smallest change in the input graph that changes the model's prediction. Hence, counterfactual explanations can be used to find a set of similar features that can alter the prediction of the model.

Organization. In the following sections, we describe each category in detail and provide summary of various explainability methods in each category. In Sec. 3, we describe the factual approaches which are further classified into self-interpretable and post-hoc categories. In Sec. 4, the counterfactual methods are categorized into perturbation-based, neural network-based and search-based methods. Sec. 5 presents three special categories of explainers such as temporal, global and causality-based. In Sec. 6, we overview the explainer methods that are relevant for specific applications in different domains such as in social networks, biology, and computer security. Lastly, we review widely used datasets in Sec. 7 and evaluation metrics in Sec. 8.

3 Factual

We classify the factual explainer methods broadly into two categories based on the nature of the integration of the explainability architecture with the main model as follows.



(a) Self-Interpretable & Post-hoc (b) Categorization of Post-hoc methods

Figure 2: (a) **Self-interpretable and post-hoc architectures :** In self-interpretable methods, the subgraph extraction module g uses constraints to find an informative subgraph G_s from the input graph G. The prediction module f uses this G_s to predict the label Y. In contrast, Post-hoc methods consider model as pre-trained with fixed weights. For any instance G, post-hoc methods generate explanation using model's input $\mathcal{D}$, output Y and in some cases the model's internal parameters. (b) **White-box and Black-box post-hoc methods:** Methods are shown in the individual categories. Decomposition-based: CAM [77], Excitation-BP [77], DEGREE [22], GNN-LRP [84]; Gradient-based: SA [5] , Guided-BP [5] , Grad-CAM [77]; Surrogate: PGM-Ex [56], GraphLime [34], GraphSVX [17], ReLex [124], DnX [75]; Perturbation-based: GNNExplainer [115], GraphMask [82], PGExplainer [56], ReFine [100], ZORRO [23], SubgraphX [122], GstarX [123]; Generation-based: XGNN [119], RGExplainer [85], GNNInterpreter [101], GFlowExplainer [46], GEM [49].

- **Post-hoc:** Post-hoc methods do not have the explainable architecture inbuilt into the model to attribute a model's prediction to the input. As seen in Figure 2a, the explainability architecture (EA) is separated from the model which is pre-trained with fixed weights. For any instance G, post-hoc methods generate an explanation using the model's input $\mathcal{D}$, output Y and sometimes even internal parameters of the model. Note that different EAs use different inputs $\mathcal{D}$ that are fed to the model. Post-hoc methods might not be always accurate as they may end up extracting features that are spuriously correlated with the task [125, 81, 69, 45].
- **Self-interpretable:** Contrary to post-hoc methods, self-interpretable explainable methods design explainability architecture directly inside the model. As seen in Figure 2a, these methods usually have two modules. The subgraph extraction module (the function g) uses constraints to find an informative subgraph G_s from the input graph G. Then, the prediction module f uses G_s to predict label Y. G_s also acts as an explanation. Both modules are trained together with an objective $L(f \circ g(G), Y)$ to minimize the loss between prediction $f \circ g(G)$ and label Y. One major drawback of self-interpretable models is that the good interpretability of the models is often at the cost of the prediction accuracy [69].

3.1 Post-hoc

We divide the post-hoc methods based on their approaches used to find explanation into the following categories: a) Decomposition-based methods (Sec. 3.1.1), b) Gradient-based methods (Sec. 3.1.2), c) Surrogate methods (3.1.3), d) Perturbation-based methods (Sec. 3.1.4), e) Generation-based methods (3.1.5). The post-hoc methods can also be categorized based on their requirement to access the internal parameters of the model. As seen in figure 2b, this division results into the following categories of the methods: **white-box** and **black-box**.

White-box: These methods require access to internal model parameters or embeddings to provide explanations. For instance, all decomposition-based methods (Sec. 3.1.1) require model parameters such as node weights of each layer to compute an importance score of different parts of the input. Even gradient based methods (Sec. 3.1.2) require access to the gradients. Thus, all methods in these categories are considered as white-box methods and they are not suitable in cases where a model's internal parameters are inaccessible.

Black-box: Contrary to the white-box methods, black-box methods do not require access to the model's internal parameters. For instance, all approaches in the category of surrogate methods (Sec.

Table 1: Key highlights of *decomposition-based* methods

Method	Parameters	Form of Explanation	Task
CAM [77]	Node embedding of last layer and MLP weights	Node importance	Graph Classification
Excitation-BP [77]	Weights of all GNN layers	Node importance	Node and Graph Classification
DEGREE [22]	Decomposes messages and requires all parameters	Similar nodes	Node and Graph Classification
GNN-LRP [84]	Weights of all layers	Collection of edges	Node and Graph Classification

3.1.3) generate a local dataset using the model's input and output. Since these methods do not require access to the model parameters, all of them can be categorized as black-box methods.

3.1.1 Decomposition-based Methods

These methods consider the prediction of the model as a score that is decomposed and distributed backwards in a layer by layer fashion till it reaches the input. The score of different parts of the input can be construed as its importance to the prediction. However, the decomposition technique can vary across methods. They also require internal parameters of the model to calculate the score. Hence, these explanation methods are considered as white-box methods. Table 1 provides a summary of these methods.

One of the decomposition-based methods, **CAM** [77] aims at constructing the explanation of GNNs that have a Global Average Pooling (GAP) layer and a fully connected layer as the final classifier. Let e_n be the final embedding of node n just before the GAP layer and w^c be the weight vector of the classifier for the class C. The importance score of the node n is computed as $(w^c)^T e_n$. This means that the node's contribution to the class score y^c is taken as the importance. It is clear that this method is restricted to GNNs that have a GAP layer and perform only the graph classification task.

Another method, **Excitation-BP** [77] considers that the final probability of the prediction can be decomposed into excitations from different neurons. The output of a neuron can be intuitively understood as the weighted sum of excitations from the connected neurons in the previous layer combined with a non-linear function where the weights are the usual neural network parameters. With this, the output probability can be distributed to the neurons in the previous layer according to the ratios of these weights. Finally, the importance of a node is obtained by combining the excitations of all the feature maps of that node.

Contrary to other methods, **DEGREE** [22] finds explanation in the form of subgraph structures. First, it decomposes the message passing feed-forward propagation mechanism of the GNN to find a contribution score of a group of target nodes. Next, it uses an agglomeration algorithm that greedily finds the most influential subgraph as the explanation. **GNN-LRP** [84] is based on the concept that the function modeled by GNN is a polynomial function in the vicinity of a specific input. The prediction score is decomposed by approximating the higher order Taylor expansion using layer-wise relevance propagation [3]. This differs from other decomposition-based methods not only in the decomposition technique but also in the score attribution. While other methods attribute scores to nodes or edges, GNN-LRP attributes scores to walks i.e., a collection of edges.

3.1.2 Gradient-based Methods

The gradient-based explainer methods follow the following key idea. Gradients represent the rate of change, and the gradient of the prediction with respect to the input represents how sensitive the prediction is to the input. This sensitivity is seen as a measure of importance. We provide a summary of these methods in Table 2.

Sensitivity Analysis (SA) [5] is one of the earlier methods to use gradients to explain GNNs. Let x be an input, which can be a node or an edge feature vector, $SA(x)$ be its importance, and ϕ be the GNN model, then the importance is computed as $SA(x) \propto ||\nabla_x \phi(x)||^2$. The intuition behind this method is based on the aforementioned sensitivity to the input. **Guided Backpropagation (Guided-BP)** [5], a slightly modified version of the previous method SA, follows a similar idea except the fact that the negative gradients are clipped to zero during the backpropagation. This is done to preserve only inputs that have an excitatory effect on the output. Intuitively, since positive and negative gradients have opposing effect on the output, using both of them could result in less accurate explanations.

Table 2: Key highlights of *gradient-based* methods

Method	Explanation Type	Task	Explanation Target	Datasets Evaluated
SA [5]	Instance level	Graph classification Node classification	Nodes, Node features Edges, Edge features	Infection, ESOL [15]
Guided-BP [5]	Instance level	Graph classification Node classification	Nodes, Node features Edges, Edge features	Infection, ESOL [15]
Grad-CAM [77]	Instance level	Node classification	Nodes, Node features	BBBP, BACE, TOX21 [40]

The method **Grad-CAM** [77] builds upon **CAM** [77] (see Section 3.1.1), and uses gradients with respect to the final node embeddings to compute the importance scores. The importance score is $(\frac{1}{N}\sum_{n=1}^{N}\nabla_{e_n}(y^c))^T e_n$, where e_n is the final embedding of node n just before the GAP layer, and w^c is the weight vector of the classifier for class C, $g = \frac{1}{N}\sum_{n=1}^{N} e_n$ is the vector after the GAP layer, and the final class score y^c of C is $w^T g$. This removes the restriction about the necessity of GAP layer. This equation shows that the importance of each node is computed as the weighted sum of the feature maps of the node embeddings, where the weights are the gradients of the output with respect to the feature maps.

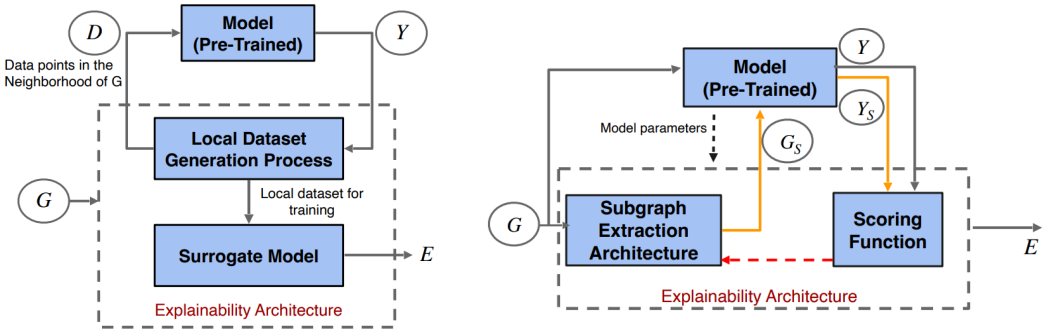
All the above methods depend on this particular intuition that the gradients can be good indicators of importance. However, this might not be useful in many settings. Gradients indicates sensitivity which does not reflect importance accurately. Moreover, saturation regions where the prediction of the model does not change significantly with the input, can be seen as another issue in SA and Guided-BP.

3.1.3 Surrogate Methods

Within a large range of input values, the relationship between input and output can be complex. Hence, we need complex functions to model this relationship and the corresponding model might not be interpretable. However, in a smaller range of input values, the relationship between input and output can be approximated by simpler and interpretable functions. This intuition leads to *surrogate methods* that fit a simple and interpretable surrogate model in the locality of the prediction. Table 3 shows different locality-based data extraction techniques and surrogate models used by surrogate methods. This surrogate model can then be used to generate explanations. As seen in the figure 3a, these methods adopt a two-step approach. Given an instance G, they first generate data from the prediction's neighborhood by utilizing multiple inputs D within the vicinity and recording the model's prediction Y. Subsequently, a surrogate model is employed to train on this data. The explanation E provided by the surrogate model serves as an explanation for the original prediction.

PGMExplainer constructs a Bayesian network to explain the prediction. First, it creates a tabular dataset by random perturbations on node features of multiple nodes of the computational graph and records its influence on the prediction. A grow-shrink algorithm is used to select top influential nodes. Using structure learning, a Bayesian network is learnt that optimizes the Bayesian Information Criterion (BIC) scores and the DAG of conditional probabilities act as an explanation. In **GraphLime** [34], the local explanations are based on Hilbert-Schmidt Independence Criterion Lasso (HSIC Lasso) model, which is a kernel-based nonlinear interpretable feature selection algorithm. This method assumes that the node features in the original graph are easily interpretable. The HSIC model takes a node and its N-hop neighbourhood (for some N), and selects a subset of node features that are the most influential to the prediction. These selected features act as the explanation. To construct a local dataset, **GraphSVX** [17] uses a mask generator to jointly perturb the nodes and the features and observes its effects on the predictions. The mask generator isolates the masked nodes and replaces masked features by its expected values. It then fits a weighted linear regression model (WLR) on the local dataset. The coefficients of WLR act as explanations.

The next two approaches use GNN based models to act as surrogate models. **RelEx** [124] uses a BFS-based sampling strategy to select nodes and then perturb them to create the local dataset. Then, a GCN model with residual connections is used to fit this dataset. In contrast to other methods in this category, the surrogate model of RelEx is not interpretable. Hence, it uses perturbation-based strategy to find a mask that acts as explanation. We note that the surrogate model is more complex compared to other methods and it requires the use of another explanation method to derive explanations from the surrogate model. **DistilnExplain (DnX)** [75] first learns a surrogate GNN



(a) **Generic schema of surrogate methods** (b) **Generic schema of perturbation-based methods**

Figure 3: **a) Surrogate:** These methods follow a two-step process. For any instance G, they generate data from the neighbourhood of the prediction by using multiple inputs D in the locality and recording its model prediction Y. Then a surrogate model is used to fit this data. Explanation E for the surrogate model is the explanation for the prediction, **b) Perturbation-based:** They have two key modules: a subgraph extraction architecture and a scoring function. For an input G, the subgraph extraction module extracts a subgraph G_s. The model prediction Y_s for subgraph G_s are scored against the actual predictions Y using a scoring function. The feedback from the scoring function can be used to train the subgraph extraction module. Sometimes model parameters are also used as the training input to the subgraph extraction module. The optimal subgraph G_s^* acts as the final explanation E.

Table 3: Key highlights of *surrogate methods*

Method	Local Dataset extraction	Surrogate model	Explanation
GraphLime [34]	N-hop neighbor nodes	HSIC Lasso	Weights of the model
PGMExplainer [97]	Random node feature perturbation	Bayesian network	DAG of conditional dependence
ReLex [124]	Random sampling of connected subgraphs	GCN	Perturbation-based method
DistilnExplain [75]	Entire dataset	Knowledge distilled Simple GCN [105]	Convex programming, decomposition
GraphSVX [17]	Input perturbations via mask generator	Weighted Linear Regression (WLR)	Weights of WLR

via knowledge distillation and then provides an explanation by solving a simple convex program. In contrast to RelEx, DnX uses a simpler surrogate model which is a linear architecture termed as Simplified Graph convolution (SGC) [105]. SGC does not have any non-linear activation layers and uses a single parameter matrix across layers. The parameters in SGC are learned via knowledge distillation with an objective to minimize the KL-divergence between the predictions by SGC and the model. Furthermore, explanations can be derived from SGC by solving a simple convex program.

3.1.4 Perturbation-based Methods

These methods find *important subgraphs* as explanations by perturbing the input. Fig. 3b presents two key modules of these methods: the subgraph extraction module and the scoring function module. For an input G, the subgraph extraction module extracts a subgraph G_s. The model predictions Y_s for subgraphs are scored against the actual predictions Y using a scoring function. The feedback from the scoring function can be used to train the subgraph extraction module. In some cases, model parameters are also used as the training input for the subgraph extraction module. These methods provide explanations E in the form of a subgraph structure and some also provide node features as explanations. Table 4 presents a summary of these methods.

GNNExplainer [115] is one of the initial efforts towards the explainability of GNNs. It identifies an explanation in the form of a Subgraph including a subset of node features that have the maximum influence on the prediction. It learns continuous masks for both adjacency matrix and features by optimizing cross entropy between the class label and model prediction on the masked subgraph. In a follow-up work, **PGExplainer** [56] extends the idea in GNNExplainer by assuming the graph to be a random Gilbert graph, where the probability distribution of edges is conditionally independent.

Table 4: Key highlights of the *perturbation-based* methods. Note that MI is mutula information, SV is Shapley value, and Explanation denotes node feature explanation.

Method	Subgraph Extraction Strategy	Scoring function	Constraints	Explanation
GNNExplainer [115]	Continuous relaxation	MI	Size	Yes
SubgraphX [122]	Monte Carlo Tree Search	SV	Size, connectivity	No
GraphMask [82]	Layer-wise parameterized edge selection	L_0 norm	Prediction divergence	No
PGexplainer [56]	Parameterized edge selection	MI	Size and/or connectivity	No
Zorro [23]	Greedy selection	Fidelity	Threshold fidelity	Yes
ReFine [100]	Parameterized edge attribution	MI	Number of edges	No
GstarX [123]	Monte Carlo sampling	HN-value	Size	No

Table 5: Key highlights of the *generation-based* methods

Methods	Explanation Type	Optimization	Constraints	Task
XGNN [119]	Model level	RL-policy gradient	Domain specific rules	Graph classification
RG-Explainer [85]	Instance level	RL-policy gradient	Size, radius, similarity	Node & Graph classification
GFLOW Explainer [46]	Instance level	TD flow matching	Connectivity / cut vertex	Node & Graph classification
GNNinterpreter [101]	Model level	Continuous relaxation	Similarity to mean	Graph Classification
GEM [49]	Instance level	Autoencoder	Graph validity rules	Node & Graph Classification

The distribution of each edge is independently modeled as a Bernoulli distribution, i.e., each edge has a different parametric distribution. These parameters are modeled by a neural network (MLP), and the parameters of this MLP is computed by optimizing the mutual information between the explanation subgraph and the predictions of the underlying GNN model. Another masking-related method, **GraphMask** [82] provides an explanation by learning a parameterized edge mask that predicts the edge to drop at every layer. A single-layer MLP classifier is trained to predict the edges that can be dropped. To keep the topology unaffected, these edges are not dropped but are replaced by a learned baseline vector. The training objective is to minimize the L_0 norm i.e., the total number of edges not masked, such that the prediction output remains within a tolerance level. To make the objective differentiable, it uses sparse relaxations through the reparameterization trick and the hard concrete distribution [59, 36].

Another approach **Zorro** [23] finds explanations in the form of important nodes and features that maximizes *Fidelity* (see Sec. 8). It uses a greedy approach that selects the node and the feature at each step with the highest fidelity score. Fidelity is computed as the expected validity of the perturbed input. The approach uses a discrete mask for selecting a subgraph without any backpropagation. A two-staged approach, **ReFine** [100] consists of the edge attribution or pre-training and the edge selection or fine-tuning steps. During pre-training, a GNN and an MLP are trained to find the edge probabilities for the entire class by maximizing mutual information and contrastive loss between classes. During the fine-tuning step, the edge probabilities from the previous stage are used to sample edges and find an explanation that maximizes mutual information for a specific instance.

The next two approaches use cooperative game theoretic techniques. **SubgraphX** [122] applies the Monte Carlo Tree search technique for subgraph exploration and uses the Shapley value [86] to measure the importance of the subgraphs. For the search algorithm, the child nodes are obtained by pruning the parent graph. In computing the Shapley values, the Monte Carlo sampling helps to find a coalition set and the prediction from the GNN is used as the pay-off in the game. In a subsequent work, **GStarX** [123] uses a different technique from cooperative game theory known as HN value [27], to compute importance scores of a node for both graph and node classification tasks. In contrast to the Shapley value, the HN value is a structure-aware metric. Since computing the HN values is expensive, Monte Carlo sampling is used for large graphs. The nodes with the top-k highest HN values act as an explanation.

3.1.5 Generation-based methods

Generation-based approaches either use generative models or graph generators to derive instance-level or model-level explanations. Furthermore, to ensure the validity of the generated graphs, different approaches have been proposed. Table 5 provides a summary of the generation-based methods.

XGNN [119] provides model-level explanations by generating key subgraph patterns to maximize prediction for a certain class. The subgraph is generated using a Reinforcement learning (RL) based

graph generator which is optimized using policy gradient. In the setup for the RL agent, the previous graph is the state; adding an edge is an action; and the model prediction along with the validity rules acts as the reward. Unsurprisingly, the validity rules are specified based on domain knowledge. Another RL-based method, **RG-Explainer** [85] formulates the underlying problem as combinatorial optimization instead of using continuous relaxation or search methods to find the subgraph. A starting point is selected using an MLP which acts as an input to the graph generator. The graph generator is an RL agent that optimizes for the policy using policy gradient with subgraph as the state, adding neighboring nodes as the action, and the function of the cross entropy loss as the reward.

A non-RL method, **GNNinterpreter** [101] is a generative model-level explanation method for the graph classification task. Its objective is to maximize the likelihood of predicting the explanation graph correctly for a given class. The similarity between the explanation graph embedding and the mean embedding of all graphs act as an optimization constraint. Intuitively, this ensures that the explanation graph stays closer to the domain and is meaningful. Since the adjacency matrix and sometimes even the features can be categorical, GNNinterpreter uses the Grumbel softmax method [36] to enable backpropagation of gradients. Contrary to XGNN with domain-specific hand-crafted rules, GNNinterpreter uses numerical optimization and does not need any domain knowledge.

GFLOW Explainer [46] uses GFLOWNETs as the generative component. The objective is to construct a TD-like flow matching condition [6] to learn a policy to generate a subgraph by sequentially adding neighbors (nodes) such that the probability of the subgraph of a class is proportional to the mutual information between the label and the distribution of possible subgraphs. A *state* consists of several nodes with the initial state as the single most influential node and the end state that satisfies the stopping criteria. *Action* is adding a node and the *reward* is a function of the cross-entropy loss.

GEM [49] uses the principles of Granger causality to generate ground-truth explanations which are used to train the explainer. It quantifies the causal contribution of each edge in the computational graph by the difference in the loss of the model with and without the edge. This distilled ground-truth for the computation graph is used to train the generative auto-encoder based explainer. This explainer provides an explanation for any instance in the form of the subgraph of the computation graph.

3.2 Self-interpretable

In self-interpretable methods, the explainable procedure is intrinsic to the model. Such methods derive explainability by incorporating interpretability constraints. These methods use either information constraints or cardinality (structural) constraints to derive an informative subgraph which is used for both the prediction and the explanation. Based on the design of the explainability, we further classify the self-interpretable methods into two types based on the imposed constraints (Fig. 4).

3.2.1 Methods with information constraints

One of the major challenges in constructing explanations via subgraphs is that the critical subgraphs may have different sizes and can be irregular. Thus, constraining the size of the explanation may not be appropriate for the underlying prediction task. To address this challenge, the methods based on information constraint use the principle of information bottleneck (IB) [94] to impose constraints on the information instead of the size. For a graph G, subgraph G_s and label Y, the graph information bottleneck (GIB) objective is:

$$\max_{G_s} I(Y, G_s) \;\; \text{such that} \;\; I(G, G_s) \leq \gamma$$

where I denotes the mutual information. Using Lagrangian multiplier β, we can write the equation as:

$$\min_{G_s} -I(Y, G_s) \; + \; \beta * I(G, G_s)$$

As seen from the equations, GIB objective-based methods have two parts in the objective function and both are intractable. All methods approximate $I(Y, G_s)$ by calculating the cross-entropy loss. However, all methods vary in their approach in making $I(G, G_s)$ tractable i.e., all have different approaches to compressing the graph and finding the informative subgraph G_s. This subgraph is used for both prediction and interpretation. Table 6 provides the summary of all methods in this category.

GSAT [69] uses a stochastic attention mechanism to calculate the variational upper bound for $I(G, G_s)$. First, it encodes graph G using a GNN to find the representation for each node. Then, for

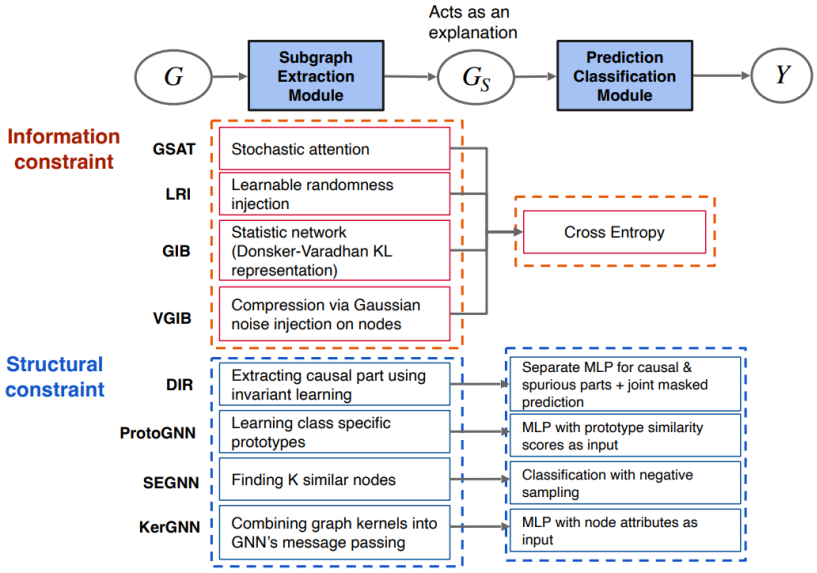


Figure 4: **Self-interpretable methods:** Every self-interpretable method has a *subgraph extraction* and a *prediction* module. The subgraph extraction module (the function g) uses constraints to find an informative subgraph G_s from input graph G. The prediction module uses G_s to predict label Y. This also shows the techniques used by each method to implement these individual modules. Self-interpretable Methods are categorized based on constraints: **(1) Information constraint:** GIB [118], VGIB [116], GSAT [69], LRI [70]; **(2) Structural constraint:** DIR [107], ProtGNN [125], SEGNN [12], KER-GNN [21].

Table 6: Key highlights of the methods with *information constraints*

Method	Process of calculating $I(G, G_s)$	Injection of Randomness	Subgraph Extractor Architecture
GSAT [69]	Stochastic attention	Bernoulli as Prior for KL divergence	GNN + MLP + Reparameterization
LRI [70]	Learnable Randomness injection	Bernoulli and Gaussian as prior	GNN + MLP + Reparameterization
GIB [118]	Donsker-Vardhan KL representation [16]	No randomness injection	Statistic Network: GNN + MLP
VGIB [116]	Compression via Noise injection	Gaussian noise on node features	GNN + MLP + Reparameterization

each node pair (u, v), GSAT uses an MLP to calculate P_{uv}. This is used to sample stochastic attention from Bernoulli distribution $Bern(P_{uv})$ to extract a subgraph G_s. The variational upper bound is the KL divergence between $Bern(P_{uv})$ and $Bern(\alpha)$ where α is a hyper-parameter. Building on similar concepts, **LRI** [70] uses both Bernoulli and Gaussian distribution as the prior distribution. LRI-Bernoulli provides the existence importance of points and LRI-Gaussian provides the location importance of the points i.e., how perturbing the location of the point in different directions affects the prediction. Another method, **GIB** [118] assumes that there is no reasonable prior distribution to solve $I(G, G_s)$ via KL divergence in the graph space. Hence, it uses the Donsker-Vardhan KL representation [16] in the latent space. It employs a bi-level optimization wherein the statistic network of the Donsker-Varadhan representation is used to estimate $I(G, G_s)$ in the inner loop. This estimate with classification and connectivity loss is used to optimize the GIB objective in the outer loop. This bi-level training process is inefficient and unstable; and hence **VGIB** [116] uses a different compression technique. The information in the original graph is dampened by injecting noise into the node representations via a learned probability P_i for each node i. The classification loss will be higher if the informative substructure G_s^* is injected with noise. Hence, G_s^* is less likely to be injected with noise compared to label-irrelevant substructures.

3.2.2 Methods with structural constraints

Imposing structural constraints on the input to derive the most informative subgraph has also been a common approach. The obtained informative subgraph is used for both making predictions and generating explanations. The key difference across the methods is the set up of the structural constraints. In Table 7, we provide the key highlights of these methods.

Table 7: Key highlights of explainability methods with *structural constraints*. Note that NC and GC denote node and graph classification respectively.

Method	Subgraph Extraction	Explanation form	Prediction/ classification module	Task
DIR [107]	Separating pattern that is invariant across interventional distribution	Invariant rationale	Seperate MLP for spurious and invariant parts	GC
ProtoGNN [125]	Computes similarity between Graph embedding and several learned diverse prototypes	Prototypes with high similarity	MLP with similarity scores as input	GC
SEGNN [12]	Finds K nodes that have similar structure and node features via contrastive loss	Similar nodes	classification via negative sampling of nodes	NC
KER-GNN [21]	Kernel filters integrated in message passing of GNNs	learned kernels and output node attributes	MLP with node attributes as input	NC, GC

One of the earlier methods, **DIR** [107] finds explanations in the form of invariant causal rationales by learning to split the input into causal (C) and non-causal (S) parts. The objective is to minimize the classification loss such that Y (the prediction) is independent of S given C. To achieve this, it first creates multiple interventional distributions by conducting interventions on the training distribution. The part that is invariant across these distributions is considered a causal part. Moreover, the implementation has three key stages. First, the architecture consists of a rationale generator (GNN) that splits the input graph into a causal part with top k edges and a non-causal part. Second, a distribution intervener, i.e., a random replacement from a set, creates perturbed distribution to infer the invariant causal parts. Finally, two classifiers are used to generate a joint prediction on causal and non-causal parts.

ProtoGNN [125] combines prototype learning [83] with GNNs. Prototype learning is a form of case-based reasoning which makes predictions for new instances by comparing them with several learned exemplar cases also called prototypes. ProtoGNN computes the similarity scores between the graph embedding and multiple learned prototypes. Moreover, these prototypes are projected onto the nearest latent training subgraph during training using the Monte Carlo tree search [88, 7]. The similarity scores are used for the classification task where the subgraphs with high similarities can be used for explanation. In another work, for a given unlabeled node, **SEGNN** [12] finds k nearest labeled nodes that have structural and feature similarities and can be used for both generating predictions and explanations. It uses contrastive loss on node representations for feature similarity and also on edge representations of local neighborhood nodes for structural similarity. Moreover, the classification loss uses negative sampling with approximate k similar nodes. These k nearest nodes can be used to derive an explanation subgraph with threshold importance.

The method, **KER-GNN** [21] integrates graph kernels into the message-passing process of GNNs to increase the expressivity of GNNs beyond the 1-WL isomorphism test. In each layer, the node embeddings are updated by computing the similarity between the node's subgraph (the node with its ego-net) and trainable filters in the form of hidden graphs. The learned graph filters can provide important structural information about the data. Moreover, the output node attributes can be used to extract important substructures.

4 Counterfactual Explanation

Counterfactual methods provides an explanation by identifying the minimal alteration in the input graph that results in a change in the model's prediction. Recently, there have been several attempts to have explanations of graph neural networks (GNNs) via counterfactual reasoning. We classify these explainer methods that find counterfactuals into three major categories based on the type of methods: **(1) Perturbation-based**, **(2) Neural framework-based**, and **(3) Search-based**. We discuss the works in the individual categories below.

4.1 Perturbation-based methods

An intuitive way to generate counterfactuals for both the graph classification and the node classification task is to *alter the edges*, i.e., add or delete the edges in the graph such that it would change the prediction of the underlying GNN method. This alteration can be achieved by perturbing either

Table 8: Key highlights of *perturbation-based* methods for counterfactuals

Method	Explanation Type	Downstream Task	Perturbation Target	Datasets Evaluated
CF-GNNExplainer [55]	Instance level	Node Classification	Computation graph	Tree-Cycles [115], Tree-Grids [115] BA-Shapes [115]
CF^2 [93]	Instance level	Graph Classification Node Classification	Original graph	BA-Shapes [115], Tree-Cycles [115] Mutag [14], NCI1 [99], CiteSeer [24]
GREASE [9]	Instance level	Node Ranking	Computation graph	LastFM, Yelp

the adjacency matrix or the computational graph of a node. The perturbation-based methods are summarized in Table 8.

One of the initial efforts, **CF-GNNExplainer** [55] aims to perturb the computational graph by using a binary mask matrix. It uses a binary matrix (all values are 0 or 1) P and modifies the computational graph matrix as $\tilde{A_v} = P \odot A_v$, where A_v is the original computational graph matrix and $\tilde{A_v}$ is computational graph matrix after the perturbation. The matrix P is computed by minimizing a combination of two different loss functions: L_{pred}, and L_{dist}. They are combined using a hyper-parameter in the final loss (L) as $L_{pred} + \beta L_{dist}$. The loss function, L_{pred} quantifies the accuracy of the produced counterfactual, and L_{dist} captures the distance (or similarity) between the counterfactual graph and the original graph. In follow-up work, the method **CF^2** [93] extends the method in CF-GNNExplainer [55] by including a contrastive loss that jointly optimizes the quality of both the factual explanation and the counterfactual one. For an input graph, G, it aims to find an optimal subgraph G_s where G_s is a good factual explanation, and $G \backslash G_s$ is a good counterfactual. These objectives are formulated as a single optimization problem with the corresponding loss as $L_{overall} = \alpha L_{factual} + (1 - \alpha) L_{counterfactual}$, where α is a hyperparameter.

Another method, **GREASE** [9] follows the standard technique of using a perturbation matrix to generate a counterfactual, but with two key modifications mainly to accommodate GNNs used for recommendation systems instead of classification tasks. In the recommendation task, GNNs rank the items (nodes) by assigning them a score instead of classifying them. GREASE uses a loss function based on the scores given by the GNN before and after the perturbation. This score helps to rank the items or nodes. The second modification is the perturbation matrix, which acts as the mask, and is used to perturb the computational graph (*l*-hop neighborhood of the node) instead of perturbing the entire graph. Here *l* denotes the number of layers in the GNN. Similar to CF^2 [93], GREASE also optimizes counterfactual and factual explanation losses, but not jointly.

In summary, all these techniques share similarities in computing the counterfactual similarity and constructing the search space. Similarity is measured by the number of edges removed from input instances and the search space is the set of all subgraphs obtained by edge deletions in the original graph. Because of the unrestricted nature of the search space, these methods might not be ideal for graphs such as molecules, where the validity of the subgraphs has valency restrictions. On the other hand, the mentioned methods differ mainly in the loss function formulations and the perturbation operations for the downstream tasks. For instance, CF-GNNExplainer [55] and GREASE [9] perform node classification and regression, they can use perturbations on the computation graph. However, CF^2 [93] considers both graph and node classification tasks, hence it uses perturbations on the entire graph, i.e., the adjacency matrix.

4.2 Neural framework-based methods

The approaches in this section use neural architectures to generate counterfactual graphs as opposed to the perturbation-based methods where the adjacency matrix of the input graph is minimally perturbed to generate counterfactuals. Table 9 summarizes these methods.

The objective of **RCExplainer** [4] is to identify a resilient subset of edges that, when removed, alter the prediction of the remaining graph. This is accomplished by modeling the implicit decision regions using graph embeddings. Even though the counterfactual graph generated by a neural architecture is used in conjunction with the adjacency matrix of the input graph, the counterfactual itself is not generated through perturbations on the adjacency matrix. RCExplainer addresses the issue of fragility where an interpretation is fragile (or non-robust) if systematic perturbations in the input graph can lead to dramatically different interpretations without changing the label. The standard explainers aim to generate good counterfactuals by choosing the closest counterfactual to the input

Table 9: Key highlights of *neural framework-based* methods for counterfactuals

Method	Explanation Type	Downstream Task	Counterfactual Generator	Datasets Evaluated
RCExplainer [4]	Instance level	Graph classification Node classification	Edge prediction with Neural Network	Mutag [14], BA-2motifs [56], NCI1 [99] Tree-Cycles [115], Tree-Grids [115] BA-Shapes [56], BA-Community [115]
CLEAR [57]	Instance level	Graph classification Node classification	Graph generation with Variational Autoencoder	Community [18], Ogbg-molhiv, IMDB-M

instance and it might induce over-fitting. RCExplainer reduces this over-fitting by first clustering input graphs using polytopes, and finding good counterfactuals close to the cluster (polytope) instead of individual instances. Another method, **CLEAR** [57] generates counterfactual graphs by leveraging a graph variational autoencoder. Two major issues often seen in other explainer methods, namely, generalization and causality are addressed in this paper.

Both methods use a generative neural model to find counterfactuals, but the generative model is different across the methods. While **RCExplainer** [4] uses a neural network that takes pairwise node embeddings and predict the existence of an edge between them, **CLEAR** [57] uses a variational autoencoder to generate a complete graph. This shows that while the former method cannot create nodes that are not present in the original graph, the latter can. In terms of the objective, the primary focus in **RCExplainer** [4] is the robustness of the generated counterfactual, but **CLEAR** [57] aims to generate counterfactuals that explain the underlying causality.

4.3 Search-based methods

These methods usually depend on search techniques over the counterfactual space for relevant tasks or applications (see the highlights in Table 10). For example, given an inactive molecule in a chemical reaction, the task is to find a similar but active molecule. Here, generative methods or perturbation methods might not be effective, and the perturbations might not even result in a valid molecule. In such cases, a good search technique through the space of counterfactuals could be more useful. An inherent challenge is that the search space of counterfactuals might be exponential in size. Hence, building efficient search algorithms is required.

The major application is finding counterfactual examples for molecules in related tasks. The method **MMACE** [102] finds counterfactuals for molecules. In the corresponding graph classification problem, it aims to classify a molecule based on a specific property. Examples include whether a molecule will permeate blood brain barrier and molecule's solubility. The search space can be generated by a method called *Superfast Traversal, Optimization, Novelty, Exploration and Discovery (STONED)* [72]. MMACE uses this method to generate the close neighbourhood and searches with a BFS-style algorithm to find an optimal set of counterfactuals.

Similarly, **MEG** [73] also aims to find a counterfactual and the search space consists of molecules. However, instead of searching the space with traditional graph search algorithms, MEG uses a reinforcement learning-based approach to navigate the search space more efficiently. The reward for finding a counterfactual is defined as the inverse of the probability that the candidate molecule found by the agent is not a counterfactual. This method is applied in a classification problem of predicting toxicity of a molecule as well as in a regression problem of predicting solubility of a molecule.

Another approach **GCFExplainer** [35] uses a random walk-based method to search the counterfactual space. The objective is not to find an individual counterfactual for each input sample but to find a small set of counterfactuals that explain all or a subset of the input samples. Hence, this is a global method (see Sec. 5.2). Here the counterfactual search space is obtained by applying graph edit operations on the training data. The method uses a random walk called **Vertex Reinforced Random Walk (VRRW)** [74], which is a modified version of a Markov chain where the state transition probabilities depend on the number of previous visits to that state.

Both **MMACE** [102] and **MEG** [73] are developed for GNNs that predict molecular properties while the objective of **GCFExplainer** [35] is to generate global explanations. However, the search algorithms and the generation mechanisms of the counterfactual space are quite different. For instance, MMACE employs a graph search algorithm to locate the nearest counterfactual instance. In contrast, MEG utilizes reinforcement learning, and GCFExplainer employs random walks to achieve the same.

Table 10: Key highlights of *search-based* methods for counterfactuals

Method	Explanation Type	Downstream Task	Counterfactual Similarity Metric	Datasets Evaluated
MMACE [102]	Instance level	Graph classification Node classification	Tanimoto similarity	Blood brain barrier dataset [63] Solubility data [89] HIV drug dataset [73]
GCFExplainer [35]	Global	Graph classification	Graph edit distance	Mutag [14], NCII [99]
MEG [73]	Instance level	Graph classification Node classification	Cosine Similarity Tanimoto similarity	Tox21 [40], ESOL [108]

5 Others

In this section we describe the explainer methods in three special categories: **explainer for temporal GNNs, global explainers** and **causality-based explainers** in this section.

5.1 Explainers for Temporal GNNs

In temporal or dynamic graphs, the graph topology and node attributes evolve over time. For instance, in social networks the relationships can be dynamic, or in the citation networks co-authorships change over time. There has been effort towards explaining the GNN models that are specifically designed for such structures.

One of the earlier explainer methods on dynamic graphs is GCN-SE [20]. GCN-SE learns the attention weights in the form of linear combination of the representations of the nodes over multiple snapshots (i.e., over time). To quantify the explanatory power of the proposed method, importance of different snapshots are evaluated via these learned weights. Another method [31] designs a two-step process. It uses static explainer such as PGM-explaine [97] to explain the underlying temporal GNN (TGNN) model (such as TGCN [126]) for each time step separately and then it aims to discover the dominant explanations from the explanations identified by the static one. DGExplainer [110] also generates explanations for dynamic GNNs by computing the relevance scores that capture the contributions of each component for a dynamic graph. More specifically, it redistributes the output activation score to the relevance of the neurons of its previous layer in the model. This process iterates until the relevance scores of the input neuron are obtained. Recently, T-GNNExplainer [109] has been proposed for temporal graph explanation where a temporal graph constituted by a sequence of temporal events. T-GNNExplainer solves the problem of finding a small set of previous events that are responsible for the model's prediction of the target event. In [51], the approach involves a smooth parameterization of the GNN predicted distributions using axiomatic attribution. These distributions are assumed to be on a low-dimensional manifold. The approach models the distributional evolution as smooth curves on the manifold and reparameterize families of curves by designing a convex optimization problem. The aim is to find a unique curve that approximates the distributional evolution and will be useful for human interpretation.

The following ones also design explainers of temporal GNNs but with specific objectives or applications. [98] studies the limit of perturbation-based explanation methods. The approach constructs some specific instances of TGNNs and evaluate how reliably node-perturbation, edge-perturbation or both can reliably identify specific graph components carrying out the temporal aggregation in temporal GNNs. In [114], a novel interpretable model on temporal heterogeneous graphs has been proposed. The method constructs temporal heterogeneous graphs which represent the research interests of the target authors. After the detection task, a deep neural network has been used for the generation process of interpretation on the predicted results. This method has been applied to research interest shift detection of researchers. Another related work [29] is on explaining GraphRC which is a special type of GNN and popular because of its training efficiency. The proposed method explores the specific role played by each reservoir node (neuron) of GraphRC by using attention mechanism on the distinct temporal patterns in the reservoir nodes.

5.2 Global Explainers

Majority of the explainers provide explanation for specific instances and can be seen as *local explainers*. However, global explainers aim to explain the overall behavior of the model by finding common input patterns that explain certain predictions [119]. Global explainers provide a high-level and generic explanation compared to local explainers. However, local explainers can be more accurate compared to global explainers [119] especially for individual instances. We categorize

global explainers into following three types.
(1) Generation-based: These post-hoc methods use either a generator or generative modeling to find explanations. For instance, **XGNN** [119] uses a reinforcement learning (RL) based graph generator optimized using policy gradient. In contrast, **GNNInterpreter** [101] is a generative global explainer that maximizes the likelihood of explanation graph being predicted as the target class by the model (the details are in Sec. 3.1.5).
(2) Concept-based: These methods provide concept based explanations. Concepts are small higher level units of information that can be interpreted by humans [25]. The methods differ in the approaches to find concepts. **GCExplainer** [60] adapts an image explanation framework known as automated concept based explanation (ACE) [25] to find global explanation for graphs. It finds concepts by clustering the embeddings from the last layer of the GNN. A concept and its importance are represented by a cluster and the number of nodes in it respectively. Another method **GCneuron** [113], which is inspired by Compositional Explanations of Neurons [71], finds global explanation for GNNs by finding compositional concepts aligned with neurons. A base concept is a function C on Graph G that produces a binary mask over all the input nodes V. A compositional concept is logical combination of base concepts. This method uses beam search to find compositional concept that minimizes the divergence between the concept and the neuron activation. Lastly, **GLGExplainer** [2] uses local explanations from PGExplainer [56] and projects them to a set of learned prototype or concepts (similar to ProtGNN [125]) to derive a concept vector. A concept vector is vector of distances between graph explanation and each prototype. This concept vector is then used to train an Entropy based logic explainable network (E-LEN) [10] to match the prediction of the class. The logic formula from the entropy layer for each class acts as explanations.
(3) Counterfactual: **Global counterfactual explainer** [35] finds a candidate set of counterfactuals using vertex re-inforced random walk. It then uses a greedy strategy to select the top k counterfactuals from the candidate set as global explanations. We explain it in more detail in Sec. 4.3.

5.3 Causality-based Explainers

Most of the GNN classifiers learn all statistical correlation between the label and input features. As a result, these may not distinguish between causal and non-causal features and may make prediction using shortcut features [90]. Shortcut features serve as confounders between the causal features and the prediction. Methods in this category attempt to reduce the confounding effect so that the model exploits causal substructure for prediction and these substructures also act as explanations. They can be categorized into the followings.

Self-interpretable methods. Methods in this category have the explainer architecture inbuilt into the model. One of the methods, **DIR** [107] creates multiple interventional distribution by conducting intervention on the training distribution. The invariant part across these distributions is considered as the causal part (see details in Sec. 3.2.2). **CAL** [90] uses edge and node attention to estimate causal and shortcut features of the graph. Two classifiers are used to make prediction on causal and shortcut features respectively. Loss on causal features is used as classification loss. Moreover, KL divergence is used to push the prediction based on shortcut features to have uniform distribution across classes. Finally, CAL creates an intervention graph in the representation space via random additions. The loss on this intervened graph classification is considered as the causal loss. These three loss terms are used to reduce the confounding effect and find the causal substructure that acts as explanation. **DisC** [19] uses a disentangled GNN framework to separate causal and shortcut substructures. It first learns a edge mask generator that divides the input into causal and shortcut substructures. Two separate GNNs are trained to produce disentangled representation of these substructures. Finally, these representations are used to generate unbiased counterfactual samples by randomly permuting the shortcut representation with the causal representation.

Generation-based methods. These methods use generative modeling to find explanations and are post-hoc. **GEM [49]** trains a generative auto-encoder by finding the causal contribution of each edge in the computation graph (details are in Sec. 3.1.5). While GEM focuses on the graph space, **OrphicX** [50] identifies the causal factors in the embedding space. It trains a variational graph autoencoder (VGAE) that has an encoder and a generator. The encoder outputs latent representations of causal and shortcut substructures of input. Generator uses both of these representations to generate the original graph and the causal representation to produce a causal mask on original graph. The information flow between the latent representation of the causal substructure and the prediction is maximized to train the explainer. The causal substructure also acts as an explanation.

6 Applications

We describe the explainers methods that are relevant for specific applications in different domains such as in social networks, biology, and computer security.

Computer Security. This work [30] focuses on designing an explanation framework for cyber-security applications using GNN models by identifying the important nodes, edges, and attributes that are contributing to the prediction. The applications include code vulnerability detection and smart contract vulnerability detection. Another work [33] proposes CFGExplainer for GNN oriented malware classification and identifies a subgraph of the malware control flow graph that is most important for the classification. Some other work focuses on the problem of botnet detection. The first method BD-GNNExplainer [128] extracts the explainer subgraph by reducing the loss between the classification results generated by the input subgraph and the entire input graph. The XG-BoT detector proposed in [53] detects malicious botnet nodes in botnet communication graphs. The explainer is based on the GNNExplainer and saliency map in the XG-BoT.

Social Networks. A recent work [80] studies the problem of detecting fake news spreaders in social networks. The proposed method SCARLET is a user-centric model that uses a GNN with attention mechanism. The attention scores help in computing the importance of the neighbors. The findings include that a person's decision to spread false information is dependent on its perception (or trust dynamics) of neighbor's credibility. On the other hand, **GCAN** [54] uses sequence models. The aim is to find a fake tweet based on the user profile and the sequence of its retweets. The sequence models and GNNs help to learn representation of retweet propagation and representation of user interactions respectively. A co-attention mechanism is further used to learn the correlation between source tweet and retweet propagation and make prediction. In [58], a GNN model has been proposed along with the explanation of its prediction for the problem on drug abuse in social networks.

Computational Biology. One of the long standing problems in neuroscience is the understanding of Brain networks, especially understanding the Regions of Interests (ROIs) and the connectivity between them. These regions and their connectivity can be modelled as a graph. A recent work on explainability, IBGNN [11] explores the explainable GNN methods to solve the task of identifying ROIs and their connectivity that are indicative of brain disorders. It uses a perturbation matrix to create an edge mask, and extracts important edges and nodes. A few more works also focus on the same task of identifying ROIs, but use different explanation techniques. In [64], the method uses a perturbation matrix with feature masks and optimizes mutual information to find the explanations. This work [127] uses Grad-CAM [77] to find important ROIs. [1] uses a search-based method to extract counterfactuals, which can serve as good candidates for important ROIs. The method uses graph edit operations to navigate from input graph to a counterfactual, but it optimizes this by using a lookup database to select edges that are the most effective in discriminating between different predicted classes. As another interesting application, this work [76] explores is related to the extraction of subgraphs in protein-protein interaction (PPI) network, where the downstream task is to detect the relevance of a protein to cancer.

Chemistry. GNNs are being used to study molecular properties extensively and often requires explanations to better understand the model's predictions. A recent work [32] focuses on improving self-interpretability of GCNs by imposing orthogonality of node features and sparsity of the GCN's weights using Gini regularization. The intuition behind the orthogonality of features is driven by the assumption that atoms in a molecule can be represented by a linear combination of orthonormal basis of wavefunctions. Another method, APRILE [112] aims at finding the parts of a drug molecule responsible for side effects. It uses perturbation techniques to extract an explanation. In drug design, the method in [38] uses integrated gradients [92] to assign importance to atoms (nodes) and the atomic properties (node features) to understand the properties of a drug.

Pathology. In medical diagnosis a challenging task is to understand the reason behind a particular diagnosis, whether it is made by a human or a machine learning system. To this end, explainer frameworks for the machine learning models become useful. In many cases the diagnosis data can be represented by graphs. This work [106] builds a graph using words, entities, clauses and sentences extracted from a patient's electronic medical record (EMR). The objective is to extract the entities most relevant for the diagnosis by training an edge mask, and is achieved by minimizing the sum

Table 11: It shows the datasets for different categories, explanation types and tasks.

Dataset	References	Nature	Explanation Type	Task
BA-Shapes	[115, 97, 124, 56, 49, 55, 4]	Synthetic	Compared to Motif	Node classification
BA-Community	[85, 115, 56, 4, 57]	Synthetic	Compared to Motif	Node classification
Tree Cycle	[49, 56, 124, 4, 55]	Synthetic	Compared to Motif	Node classification
Tree Grids	[115, 56, 124, 49, 4, 55]	Synthetic	Compared to Motif	Node classification
BA-2Motif	[56, 122, 69, 4]	Synthetic	Compared to Motif	Graph classification
Spurious Motifs	[69, 107]	Synthetic	Compared to Motif	Graph classification
Mutagenicity	[115, 56, 49, 125, 122, 4, 119]	Real-World	Compared to Chemical property	Graph classification
NCI1	[49, 4]	Real-World	Compared to Chemical property	Graph classification
BBBP	[125, 93, 108]	Real-World	Compared to Chemical property	Graph classification
Tox21	[73, 108]	Real-World	Compared to Chemical property	Graph classification
MNIST-75sp	[97, 69, 43]	Real-World	Visual	Graph classification
Sentiment Graphs	[122, 69, 125, 121]	Real-World	Visual	Graph classification

of the elements in the mask matrix. Another method [37] focuses on generating explanations for histology (micro-anatomy) images. It first converts the image into a graph of biological entities, where the nodes could be cells, tissues or some other task specific biological features. Afterwards the standard explainer techniques described in gradient 3.1.2 or perturbation 3.1.4 based methods are used to generate the explanations. Another work [117] in this field modifies the objective to optimise for both necessity and sufficiency (Sec. 8). The explanation is generated in such a way that the mutual information between explanation subgraph and the prediction is maximized. Additionally, the mutual information between the remaining graph after removing the explanation subgraph and the prediction is minimized.

7 Datasets

A set of synthetic as well as real-world datasets have been used for evaluating the proposed explainers in several tasks such as node classification and graph classification. Table 11 lists down the set of datasets and the corresponding explanation types and tasks used in the literature.

7.1 Synthetic datasets

Annotating ground truth explanations in graph data is laborious and requires domain expertise. To overcome this challenge, several explainers have been evaluated using synthetic datasets that are created using certain motifs as ground truth values. We highlight *six* popular synthetic datasets:

BA-Shapes [115]: This graph is formed by randomly connecting a base graph to a set of motifs. The base graph is a Barabasi-Albert (BA) graph with 300 nodes. It includes 80 house-structured motifs with five nodes each, formed by a top, a middle, and a bottom node type.

BA-Community [115]: The BA-community graph is a combination of two BA-Shapes graphs. The features of each node are assigned based on two Gaussian distributions. Also, nodes are assigned a class out of eight classes based on the community they belong to.

Tree Cycle [115]: This consists of a 8-level balanced binary tree as a base graph. To this base graph, 80 cycle motifs with six nodes each are randomly connected. It just has two classes; one for the nodes in the base graph and another for nodes in the defined motif.

Tree Grids [115]: This graph uses same base graph but a different motif set compared to the tree cycle graph. It uses 3 by 3 grid motifs instead of the cycle motifs.

BA-2Motifs [56]: This is used for graph classification and has two classes. The base graph is BA graph for both the classes. However, one class has a house-structure motif and another has a 5-node cycle motif.

Spurious Motifs [107]: With 18000 graphs in the dataset, each graph is a combination of one base S (Tree, Ladder or Wheel) and one motif C (Cycle, House, Crane). Ground-truth Y is determined by the motif. A spurious relation between S and Y is manually induced. This spurious correlation can be varied based on a parameter that ranges from 0 to 1.

7.2 Real-world datasets

Due to the known chemical properties of the molecules, molecular graph datasets become a good choice for evaluating the generated explanation structure. We highlight some widely used molecular datasets for evaluating explainers in the *graph classification task*.

Mutag [14]: This consists of 4337 molecules (graphs) with two classes based on the mutagenic effect. Using domain knowledge, specific chemical groups are assigned as ground truth explanations.

NCI1 [99]: It is a graph classification dataset with 4110 instances. Each graph is a chemical compound where a node represents an atom and an edge represents a bond between atoms. Each molecule is screened for activity against non-small cell lung cancer or ovarian cancer cell lines.

BBBP [108]: Similar to Mutag, Blood-brain barrier penetration (BBBP) is also a molecule classification dataset with two classes with 2039 compounds. Classification is based on their permeability properties.

Tox21 [108]: This dataset consists of 7831 molecules with 12 different categories of chemical compounds. The categorization is based on the chemical structures and properties of those compounds.

Visual explanation can be an important component of comparing explainers. Hence, researchers also use datasets that do not have ground truth explanations but can be visually evaluated through generated examples. Below are some of the datasets used for visual analysis:

MNIST-75sp [43]: An MNIST image is converted to a super-pixel graph with at most 75 nodes, where each node denotes a "super pixel". Pixel intensity and coordinates of their centers of masses are used as the node attributes. Edges are formed based on the spatial distance between the super-pixel centers. Each graph is assigned one of the 10 MNIST classes, i.e., numerical digits.

Sentiment Graphs [121]: Graph SST2, Graph SST5, and Graph Twitter are based on text sentiment analysis data of SST2, SST5, and Twitter datasets. A graph is constructed by considering tokens as nodes, relations as edges, and sentence sentiment as its label. The BERT architecture is used to obtain 768-dimensional word embeddings for the dataset. The generated explanation graph can be evaluated for its textual meaning.

8 Evaluation

The evaluation of the explainer methods is based on the quality of the explainer's ability to generate human-intelligible explanations about the model prediction. As this might be subjective depending on the applications in hand, the evaluation measures consider both quantitative and qualitative metrics.

8.1 Quantitative Evaluation

Quantitative evaluation metrics help in having a standardized evaluation that is free of human bias. For this, explainability is posed as a binary classification problem. The explainers assign a score to the *node features*, *edges*, and *motifs*, which are the most responsible for the prediction according to the explainer. We are also provided with the ground-truth binary labels for the features and structures based, denoting whether they are responsible for the prediction or not. The explainer is then evaluated by comparing these scores to the ground-truth explanation labels using different methods:

Accuracy [55, 93]: To find the accuracy, the top-k edges produced by the explainer are set to be positive, and the rest are negative. These top-k edges and the ground-truth labels are compared to compute the accuracy.

Area Under Curve (AUC) [124, 102, 4]: We compare the top-k raw scores directly against the ground-truth labels by computing the area under the ROC curve.

Fidelity [116, 55, 4]: This is used for explainers that generate a subgraph as the explanation. It compares the performances of the base GNN model on the input graph and the explainer subgraph. Let N be the number of samples, y_i is the true label of sample i, $\hat{y}_i$ is the predicted label of sample i, $\hat{y}^k$ is the predicted label after choosing the subgraph formed by nodes with top-$k\%$ nodes, and $\mathbb{1}[\cdot]$ is the indicator function. Fidelity measures how close the predictions of the explanation sub-graph are to the input graph. For factual explainers, the lower this value, the better is the explanation. It is

formally defined as follows:

$$\text{Fidelity} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[y_i = \hat{y}_i] - \mathbb{1}[y_i = \hat{y}_i^k]$$

Sparsity [116, 55]: It measures the conciseness of explanations (e.g., the sub-graphs) that are responsible for the final prediction. Let $|p_i|$ and $|G_i|$ denote the number of edges in the explanation, and the same in the original input graph, respectively. The sparsity is then defined as follows:

$$\text{Sparsity} = 1 - \frac{1}{N}\sum_{i=1}^{N} \frac{|p_i|}{|G_i|}$$

Robustness [4]: It quantifies how resistant an explainer is to perturbations on input graph. Here perturbations are addition or deletion of edges randomly such that it does not change the prediction of the underlying GNN. The robustness is the percentage of graphs for which these perturbations do not change the explanation.

Probability of Sufficiency (PS) [93, 9]: It is the percentage of graphs for which the explanation graph is sufficient to generate the same prediction as the original input graph.

Probability of Necessity (PN) [93, 9]: It is the percentage of graphs for which the explanation graph when removed from the original input graph will alter the prediction made by the GNN.

Generalization [85]: This measures the capability of generalization of the explainer method in an inductive setting. To measure this, the training dataset size is usually varied and the AUC scores are computed for these tests. Generalisation plays an important role in explanability as the good generalizable models are generally sparse in terms of inputs. This metric is highly relevant for the self-interpretable models.

8.2 Qualitative Evaluation

Explanations can also be evaluated qualitatively using expert domain knowledge. This mode of evaluation is crucial especially while working with real-world datasets that do not have ground truth labels.

Domain Knowledge [115]: Generated explanations can be evaluated for their meaning in the application domain. For example, GNNExplainer [115] correctly identifies the carbon ring as well as chemical groups NH2 and NO2, which are known to be mutagenic.

Manual Scoring [97]: Another method of evaluating the explanations is by asking users (e.g., domain experts) to score, say on a scale of 1-10, the explanations generated by various explainers and compare them. One can also use RMSE scores to quantitatively compare these explainers based on the scores.

9 Future Directions

Combinatorial problems: Most of the existing explanation frameworks are for prediction tasks such as node and graph classification. However, graphs are prevalent in various domains, such as in social networks [39, 68], healthcare [103], and infrastructure development [66, 65]. Solving combinatorial optimization problems on graphs is a common requirement in these domains. Several architectures based on Graph Neural Networks (GNNs) [41, 61, 78] have been proposed to tackle these problems that are usually computationally hard. However, the explainability of these methods for such combinatorial problems is largely missing. One potential direction is to build frameworks that can explain the behavior of the solution set in such problems.

Global methods: Most explainers primarily adopt a local perspective by generating examples specific to individual input graphs. From global explanations, we can extract higher-level insights that complement the understanding gained from local explanations (see details on global methods in Sec. 5.2). Moreover, global explanations can be easily understood by humans even for large datasets. Real-world graph datasets often consist of millions of nodes. When generating explanations specific to each instance, the number of explanations increases proportionally with the size of the dataset. As

a result, the sheer volume of explanations becomes overwhelming for human cognitive capabilities to process effectively. Global approaches can immensely help in these scenarios.

Visualization and HCI tools: Graph data, unlike textual and visual data, cannot be perceived by human senses. Thus, qualitative evaluation of explanation becomes a non-trivial problem and often requires expert guidance [115, 97]. This makes crowdsourcing evaluations difficult and not scalable. Other ways to qualitatively assess graph structures for explanation of a certain prediction can be explored. Additionally, since explainability is human-centric, it is crucial that explainers are influenced by human cognition and behavior, particularly those of domain experts [48] while using GNNs in making important decisions [67]. HCI research can help in designing the interface for the experts to assess the generated explanation graphs [55].

Temporal GNNs: Temporal graph models are designed to predict the graph structure and labels in the future by exploiting how the graph has evolved in the past. This increases the complexity of explanations significantly as they now involve combinations of graph structures at different time intervals. Existing methods [20, 110, 31, 44] mostly focus on discrete-time models where graphs are provided at different points in time. Future works can explore ways to explain the prediction of a continuous-time dynamic graph model, where interactions happen in real time [109]. One direction could be to optimize over a parameterized temporal point process [95].

10 Conclusions

In this survey, we have provided a comprehensive overview of explanation methods for Graph Neural Networks (GNNs). Besides outlining some background on GNNs and explainability, we have presented a detailed taxonomy of the papers from the literature. By categorizing and discussing these methods, we have highlighted their strengths, limitations, and applications in understanding GNN predictions. Moreover, we have highlighted some widely used datasets and evaluation metrics in assessing the explainability of GNNs. As GNNs continue to play a significant role in various fields, such as healthcare, recommendation systems, and natural language processing, the need for interpretable and transparent models becomes increasingly important. Overall, we believe this survey serves as a valuable resource for researchers and practitioners interested in the explainability of GNNs and provides a foundation for further advancements in interpretable graph representation learning.

References

[1] Carlo Abrate and Francesco Bonchi. Counterfactual graphs for explainable classification of brain networks. In *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2495–2504, 2021.

[2] 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.

[3] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[4] Mohit Bajaj, Lingyang Chu, Zi Yu Xue, Jian Pei, Lanjun Wang, Peter Cho-Ho Lam, and Yong Zhang. Robust counterfactual explanations on graph neural networks. *Advances in Neural Information Processing Systems*, 34:5644–5655, 2021.

[5] Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*, 2019.

[6] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *arXiv preprint arXiv:2111.09266*, 2021.

[7] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[8] Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[9] Ziheng Chen, Fabrizio Silvestri, Jia Wang, Yongfeng Zhang, Zhenhua Huang, Hongshik Ahn, and Gabriele Tolomei. Grease: Generate factual and counterfactual explanations for gnn-based recommendations. *arXiv preprint arXiv:2208.04222*, 2022.

[10] Gabriele Ciravegna, Pietro Barbiero, Francesco Giannini, Marco Gori, Pietro Lió, Marco Maggini, and Stefano Melacci. Logic explained networks. *Artificial Intelligence*, 314:103822, 2023.

[11] Hejie Cui, Wei Dai, Yanqiao Zhu, Xiaoxiao Li, Lifang He, and Carl Yang. Interpretable graph neural networks for connectome-based brain disorder analysis. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part VIII*, pages 375–385. Springer, 2022.

[12] Enyan Dai and Suhang Wang. Towards self-explainable graph neural network. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 302–311, 2021.

[13] 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.

[14] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. 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, 1991.

[15] John S. Delaney. Esol: Estimating aqueous solubility directly from molecular structure. *Journal of Chemical Information and Computer Sciences*, 44(3):1000–1005, 2004. PMID: 15154768.

[16] Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975.

[17] Alexandre Duval and Fragkiskos D Malliaros. Graphsvx: Shapley value explanations for graph neural networks. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*, pages 302–318. Springer, 2021.

[18] P Erd. os and a. r enyi,\on random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

[19] Shaohua Fan, Xiao Wang, Yanhu Mo, Chuan Shi, and Jian Tang. Debiasing graph neural networks via learning disentangled causal substructure. *arXiv preprint arXiv:2209.14107*, 2022.

[20] Yucai Fan, Yuhang Yao, and Carlee Joe-Wong. Gcn-se: Attention as explainability for node classification in dynamic graphs. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1060–1065. IEEE, 2021.

[21] Aosong Feng, Chenyu You, Shiqiang Wang, and Leandros Tassiulas. Kergnns: Interpretable graph neural networks with graph kernels. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6614–6622, 2022.

[22] Qizhang Feng, Ninghao Liu, Fan Yang, Ruixiang Tang, Mengnan Du, and Xia Hu. Degree: Decomposition based explanation for graph neural networks. In *International Conference on Learning Representations*, 2022.

[23] Thorben Funke, Megha Khosla, Mandeep Rathee, and Avishek Anand. Z orro: Valid, sparse, and stable explanations in graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[24] L Getoor. Advanced methods for knowledge discovery from complex data. *Link-based Classification*, 189:207, 2005.

[25] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. *Advances in Neural Information Processing Systems*, 32, 2019.

[26] Zhimeng Guo, Teng Xiao, Charu Aggarwal, Hui Liu, and Suhang Wang. Counterfactual learning on graphs: A survey. *arXiv preprint arXiv:2304.01391*, 2023.

[27] Gérard Hamiache and Florian Navarro. Associated consistency, value and graphs. *International Journal of Game Theory*, 49:227–249, 2020.

[28] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[29] Xinyu Han and Yi Zhao. Interpretable graph reservoir computing with the temporal pattern attention. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[30] Haoyu He, Yuede Ji, and H Howie Huang. Illuminati: Towards explaining graph neural networks for cybersecurity analysis. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 74–89. IEEE, 2022.

[31] Wenchong He, Minh N Vu, Zhe Jiang, and My T Thai. An explainer for temporal graph neural networks. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 6384–6389. IEEE, 2022.

[32] Ryan Henderson, Djork-Arné Clevert, and Floriane Montanari. Improving molecular graph neural network explainability with orthonormalization and induced sparsity. In *International Conference on Machine Learning*, pages 4203–4213. PMLR, 2021.

[33] Jerome Dinal Herath, Priti Prabhakar Wakodikar, Ping Yang, and Guanhua Yan. Cfgexplainer: Explaining graph neural network-based malware classification from control flow graphs. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 172–184. IEEE, 2022.

[34] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[35] Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, and Ambuj Singh. Global counterfactual explainer for graph neural networks. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 141–149, 2023.

[36] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[37] Guillaume Jaume, Pushpak Pati, Behzad Bozorgtabar, Antonio Foncubierta, Anna Maria Anniciello, Florinda Feroce, Tilman Rau, Jean-Philippe Thiran, Maria Gabrani, and Orcun Goksel. Quantifying explainers of graph neural networks in computational pathology. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8106–8116, 2021.

[38] José Jiménez-Luna, Miha Skalic, Nils Weskamp, and Gisbert Schneider. Coloring molecules with explainable artificial intelligence for preclinical relevance assessment. *Journal of Chemical Information and Modeling*, 61(3):1083–1094, 2021.

[39] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[40] Kristian Kersting, Nils M Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels. 2016.

[41] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

[42] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[43] Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[44] Mert Kosan, Arlei Silva, Sourav Medya, Brian Uzzi, and Ambuj Singh. Event detection on dynamic graphs. *arXiv preprint arXiv:2110.12148*, 2021.

[45] Mert Kosan, Arlei Silva, and Ambuj Singh. Robust ante-hoc graph explainer using bilevel optimization. *arXiv preprint arXiv:2305.15745*, 2023.

[46] 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.

[47] 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.

[48] Q Vera Liao and Kush R Varshney. Human-centered explainable ai (xai): From algorithms to user experiences. *arXiv preprint arXiv:2110.10790*, 2021.

[49] 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.

[50] 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.

[51] Yazheng Liu, Xi Zhang, and Sihong Xie. A differential geometric view and explainability of gnn on evolving graphs. In *The Eleventh International Conference on Learning Representations*, 2023.

[52] Yunchao Liu, Yu Wang, Oanh T Vu, Rocco Moretti, Bobby Bodenheimer, Jens Meiler, and Tyler Derr. Interpretable chirality-aware graph neural network for quantitative structure activity relationship modeling in drug discovery. *bioRxiv*, pages 2022–08, 2022.

[53] Wai Weng Lo, Gayan Kulatilleke, Mohanad Sarhan, Siamak Layeghy, and Marius Portmann. Xg-bot: An explainable deep graph neural network for botnet detection and forensics. *Internet of Things*, 22:100747, 2023.

[54] Yi-Ju Lu and Cheng-Te Li. Gcan: Graph-aware co-attention networks for explainable fake news detection on social media. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 505–514, 2020.

[55] Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 4499–4511. PMLR, 2022.

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

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

[58] Zuanjie Ma, Hongming Gu, and Zhenhua Liu. Understanding drug abuse social network using weighted graph neural networks explainer. In *Computational Science and Its Applications–ICCSA 2021: 21st International Conference, Cagliari, Italy, September 13–16, 2021, Proceedings, Part III 21*, pages 52–61. Springer, 2021.

[59] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[60] Lucie Charlotte Magister, Dmitry Kazhdan, Vikash Singh, and Pietro Liò. Gcexplainer: Human-in-the-loop concept-based explanations for graph neural networks. *arXiv preprint arXiv:2107.11889*, 2021.

[61] Sahil Manchanda, Akash Mittal, Anuj Dhawan, Sourav Medya, Sayan Ranu, and Ambuj Singh. Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs. *Advances in Neural Information Processing Systems*, 33:20000–20011, 2020.

[62] Debmalya Mandal, Sourav Medya, Brian Uzzi, and Charu Aggarwal. Metalearning with graph neural networks: Methods and applications. *SIGKDD Explor. Newsl.*, 23(2):13–22, jan 2022.

[63] Ines Filipa Martins, Ana L Teixeira, Luis Pinheiro, and Andre O Falcao. A bayesian approach to in silico blood-brain barrier penetration modeling. *Journal of chemical information and modeling*, 52(6):1686–1697, 2012.

[64] Chiara Mauri, Stefano Cerri, Oula Puonti, Mark Mühlau, and Koen Van Leemput. Accurate and explainable image-based prediction using a lightweight generative model. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part VIII*, pages 448–458. Springer, 2022.

[65] Sourav Medya, Petko Bogdanov, and Ambuj Singh. Towards scalable network delay minimization. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1083–1088. IEEE, 2016.

[66] Sourav Medya, Sayan Ranu, Jithin Vachery, and Ambuj Singh. Noticeable network delay minimization via node upgrades. *Proceedings of the VLDB Endowment*, 11(9):988–1001, 2018.

[67] Sourav Medya, Mohammad Rasoolinejad, Yang Yang, and Brian Uzzi. An exploratory study of stock price movements from earnings calls. In *Companion Proceedings of the Web Conference 2022*, pages 20–31, 2022.

[68] Sourav Medya, Arlei Silva, and Ambuj Singh. Approximate algorithms for data-driven influence limitation. *IEEE Transactions on Knowledge and Data Engineering*, 34(6):2641–2652, 2020.

[69] 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.

[70] Siqi Miao, Yunan Luo, Mia Liu, and Pan Li. Interpretable geometric deep learning via learnable randomness injection. *arXiv preprint arXiv:2210.16966*, 2022.

[71] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, 33:17153–17163, 2020.

[72] AkshatKumar Nigam, Robert Pollice, Mario Krenn, Gabriel dos Passos Gomes, and Alan Aspuru-Guzik. Beyond generative models: superfast traversal, optimization, novelty, exploration and discovery (stoned) algorithm for molecules using selfies. *Chemical science*, 12(20):7079–7090, 2021.

[73] Danilo Numeroso and Davide Bacciu. Meg: Generating molecular counterfactual explanations for deep graph networks. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[74] Robin Pemantle. Vertex-reinforced random walk. *Probability Theory and Related Fields*, 92(1):117–136, 1992.

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

[76] Bastian Pfeifer, Anna Saranti, and Andreas Holzinger. Gnn-subnet: disease subnetwork detection with explainable graph neural networks. *Bioinformatics*, 38(Supplement_2):ii120–ii126, 2022.

[77] 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.

[78] Rishabh Ranjan, Siddharth Grover, Sourav Medya, Venkatesan Chakaravarthy, Yogish Sabharwal, and Sayan Ranu. Greed: A neural framework for learning graph distance functions. In *Advances in Neural Information Processing Systems*, 2022.

[79] Susie Xi Rao, Shuai Zhang, Zhichao Han, Zitao Zhang, Wei Min, Zhiyao Chen, Yinan Shan, Yang Zhao, and Ce Zhang. xfraud: explainable fraud transaction detection. *Proceedings of the VLDB Endowment*, (3):427–436, 2021.

[80] Bhavtosh Rath, Xavier Morales, and Jaideep Srivastava. Scarlet: explainable attention based graph neural network for fake news spreader prediction. In *Advances in Knowledge Discovery and Data Mining: 25th Pacific-Asia Conference, PAKDD 2021, Virtual Event, May 11–14, 2021, Proceedings, Part I*, pages 714–727. Springer, 2021.

[81] Cynthia Rudin. Please stop explaining black box models for high stakes decisions. *stat*, 1050:26, 2018.

[82] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for nlp with differentiable edge masking. *International Conference on Learning Representations*, 2021.

[83] Rainer Schmidt, Stefania Montani, Riccardo Bellazzi, Luigi Portinale, and Lothar Gierl. Cased-based reasoning for medical knowledge-based systems. *International Journal of Medical Informatics*, 64(2-3):355–367, 2001.

[84] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon. Higher-order explanations of graph neural networks via relevant walks. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):7581–7596, 2021.

[85] Caihua Shan, Yifei Shen, Yao Zhang, Xiang Li, and Dongsheng Li. Reinforcement learning enhanced explainer for graph neural networks. *Advances in Neural Information Processing Systems*, 34:22523–22533, 2021.

[86] Lloyd S Shapley et al. A value for n-person games. 1953.

[87] Ben Shneiderman. Bridging the gap between ethics and practice: guidelines for reliable, safe, and trustworthy human-centered ai systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 10(4):1–31, 2020.

[88] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[89] Murat Cihan Sorkun, Abhishek Khetan, and Süleyman Er. Aqsoldb, a curated reference set of aqueous solubility and 2d descriptors for a diverse set of compounds. *Scientific data*, 6(1):143, 2019.

[90] Yongduo Sui, Xiang Wang, Jiancan Wu, Min Lin, Xiangnan He, and Tat-Seng Chua. Causal attention for interpretable and generalizable graph classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1696–1705, 2022.

[91] Mengying Sun, Sendong Zhao, Coryandar Gilvary, Olivier Elemento, Jiayu Zhou, and Fei Wang. Graph convolutional networks for computational drug development and discovery. *Briefings in bioinformatics*, 21(3):919–935, 2020.

[92] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[93] Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *Proceedings of the ACM Web Conference 2022*, pages 1018–1027, 2022.

[94] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

[95] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*, 2019.

[96] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

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

[98] Minh N Vu and My T Thai. On the limit of explaining black-box temporal graph neural networks. In *AAAI*, 2022.

[99] Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14:347–375, 2008.

[100] Xiang Wang, Yingxin Wu, An Zhang, Xiangnan He, and Tat-Seng Chua. Towards multi-grained explainability for graph neural networks. *Advances in Neural Information Processing Systems*, 34:18446–18458, 2021.

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

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

[103] Bryan Wilder, Han-Ching Ou, Kayla de la Haye, and Milind Tambe. Optimizing network structure for preventative health. In *AAMAS*, pages 841–849, 2018.

[104] 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.

[105] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[106] Haoran Wu, Wei Chen, Shuang Xu, and Bo Xu. Counterfactual supporting facts extraction for explainable medical record based diagnosis with graph network. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1942–1955, 2021.

[107] Ying-Xin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. Discovering invariant rationales for graph neural networks. *International Conference on Learning Representations*, 2022.

[108] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[109] Wenwen Xia, Mincai Lai, Caihua Shan, Yao Zhang, Xinnan Dai, Xiang Li, and Dongsheng Li. Explaining temporal graph models through an explorer-navigator framework. In *The Eleventh International Conference on Learning Representations*, 2023.

[110] Jiaxuan Xie, Yezi Liu, and Yanning Shen. Explaining dynamic graph neural networks via relevance back-propagation. *arXiv preprint arXiv:2207.11175*, 2022.

[111] Jiacheng Xiong, Zhaoping Xiong, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. Graph neural networks for automated de novo drug design. *Drug Discovery Today*, 26(6):1382–1393, 2021.

[112] Hao Xu, Shengqi Sang, Herbert Yao, Alexandra I Herghelegiu, Haiping Lu, James T Yurkovich, and Laurence Yang. Aprile: Exploring the molecular mechanisms of drug side effects with explainable graph neural networks. *bioRxiv*, pages 2021–07, 2021.

[113] Han Xuanyuan, Pietro Barbiero, Dobrik Georgiev, Lucie Charlotte Magister, and Pietro Lió. Global concept-based interpretability for graph neural networks via neuron analysis. *arXiv preprint arXiv:2208.10609*, 2022.

[114] Qiang Yang, Changsheng Ma, Qiannan Zhang, Xin Gao, Chuxu Zhang, and Xiangliang Zhang. Interpretable research interest shift detection with temporal heterogeneous graphs. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 321–329, 2023.

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

[116] Junchi Yu, Jie Cao, and Ran He. Improving subgraph recognition with variational graph information bottleneck. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19396–19405, 2022.

[117] Junchi Yu, Tingyang Xu, and Ran He. Towards the explanation of graph neural networks in digital pathology with information flows. *arXiv preprint arXiv:2112.09895*, 2021.

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

[119] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 430–438, 2020.

[120] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[121] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[122] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International Conference on Machine Learning*, pages 12241–12252. PMLR, 2021.

[123] Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. Gstarx: Explaining graph neural networks with structure-aware cooperative games. In *Advances in Neural Information Processing Systems*, 2022.

[124] Yue Zhang, David Defazio, and Arti Ramesh. Relex: A model-agnostic relational model explainer. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 1042–1049, 2021.

[125] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Cheekong Lee. Protgnn: Towards self-explaining graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9127–9135, 2022.

[126] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE transactions on intelligent transportation systems*, 21(9):3848–3858, 2019.

[127] Houliang Zhou, Lifang He, Yu Zhang, Li Shen, and Brian Chen. Interpretable graph convolutional network of multi-modality brain imaging for alzheimer's disease diagnosis. In *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, pages 1–5. IEEE, 2022.

[128] Xiaolin Zhu, Yong Zhang, Zhao Zhang, Da Guo, Qi Li, and Zhao Li. Interpretability evaluation of botnet detection model based on graph neural network. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2022.

[129] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

[130] Marinka Zitnik, Francis Nguyen, Bo Wang, Jure Leskovec, Anna Goldenberg, and Michael M Hoffman. Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. *Information Fusion*, 50:71–91, 2019.