Explainability in Graph Neural Networks: A Taxonomic Survey

Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji

Abstract—Deep learning methods are achieving ever-increasing performance on many artificial intelligence tasks. A major limitation of deep models is that they are not amenable to interpretability. This limitation can be circumvented by developing *post hoc* techniques to explain the predictions, giving rise to the area of explainability. Recently, explainability of deep models on images and texts has achieved significant progress. In the area of graph data, graph neural networks (GNNs) and their explainability are experiencing rapid developments. However, there is neither a unified treatment of GNN explainability methods, nor a standard benchmark and testbed for evaluations. In this survey, we provide a unified and taxonomic view of current GNN explainability methods. Our unified and taxonomic treatments of this subject shed lights on the commonalities and differences of existing methods and set the stage for further methodological developments. To facilitate evaluations, we generate a set of benchmark graph datasets specifically for GNN explainability. We summarize current datasets and metrics for evaluating GNN explainability. Altogether, this work provides a unified methodological treatment of GNN explainability and a standardized testbed for evaluations.

Index Terms—Graph analysis, graph neural networks, explainability, interpretability, taxonomy, evaluation, survey.

✦

1 INTRODUCTION

THe developments of deep neural networks have revolutionized the fields of machine learning and artificial intelligence. Deep neural networks have achieved promising performance in many research tasks, such as computer vision [1], [2], natural language processing [3], [4], and graph data analysis [5], [6]. These facts motivate us to develop deep learning methods for real-world applications in interdisciplinary domains, such as finance, biology, and agriculture [7], [8], [9]. However, since most deep models are developed without interpretability, they are treated as black-boxes. Without reasoning the underlying mechanisms behind the predictions, deep models cannot be fully trusted, which prevents their use in critical applications pertaining to fairness, privacy, and safety. In order to safely and trustfully deploy deep models, it is necessary to provide both accurate predictions and human-intelligible explanations, especially for users in interdisciplinary domains. These facts raise the need of developing explanation techniques to explain deep neural networks.

The explanation techniques of deep models generally study the underlying relationships behind the predictions of deep models. Several techniques are proposed to explain the deep models for image and text data [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21]. These methods can provide input-dependent explanations, such as studying the important scores for input features, or a high-level understanding of the general behaviors of deep models. For example, by studying the gradients or weights [10], [11], [18], we can analyze the sensitivity between the input features and the predictions. Existing approaches [12], [13], [19] map hidden feature maps to the input space and highlight the important input features. In addition,

by occluding different input features, we can observe and monitor the change of predictions to identify important features [14], [15]. Meanwhile, several studies [10], [16] focus on providing input-independent explanations, such as studying the input patterns that maximize the predicted score of a certain class. Furthermore, the meaning of hidden neurons is explored to understand the whole prediction procedures [17], [22]. Recent survey works [23], [24], [25], [26] provides good systematic reviews and taxonomies of these methods. However, these surveys only focus on the explanation methods for image and text domains and ignore the explainability of deep graph models.

Recently, Graph Neural Networks (GNNs) have become increasingly popular since many real-world data are represented as graphs, such as social networks, chemical molecules, and financial data [27], [28]. Several graph-related tasks are widely studied, such as node classification [29], [30], [31], graph classification [6], [32], link predictions [33], [34], [35]. In addition, many advanced GNN operations are proposed to improve the performance, including graph convolution [5], [36], [37], graph attention [38], [39], and graph pooling [40], [41], [42]. However, compared with image and text domains, the explainability of graph models are less explored, which is critical for understanding deep graph neural networks. Recently, several approaches are proposed to explain the predictions of GNNs, such as XGNN [43], GNNExplainer [44], PGExplainer [45], and-SubgraphX [46], etc. These methods are developed from different angles and provide different levels of explanations. In addition, it is still lacking standard datasets and metrics for evaluating the explanation results. Hence, it raises the need of a systematic survey covering the methods and evaluations of GNN explanation techniques.

To this end, this survey provides a systematic study of different GNN explanation techniques. We aim at providing intuitive understanding and high-level insights of different

• H. Yuan, H. Yu, S. Gui, and S. Ji are with Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843, USA, e-mail: {hao.yuan, haiyang, shurui.gui, sji}@tamu.edu.

methods so that researchers can choose proper directions to explore. The contributions of this work are summarized as below:

- This survey provides a systematic and comprehensive review of existing explanation techniques for deep graph models. To the best of our knowledge, this is the first and only survey work on such a topic.
- We propose a novel taxonomy of existing GNN explanation techniques. We conclude the key idea of each category and provide insightful analysis.
- We introduce each GNN explanation method in detail, including its methodology, advantages, drawbacks, and difference compared with other methods.
- We summarize the commonly employed datasets and evaluation metrics for GNN explanation tasks. We discuss their limitations and recommend several convincing metrics.
- We build three human-understandable datasets from the text domain by converting sentences to graphs. These datasets are now publicly available and can be directly used for GNN explanation tasks.
- We develop an open-source library for the GNN explanation research. In this library, we include the implementations of several existing techniques, commonly used datasets, and different evaluation metrics. It can be used to reproduce existing methods and develop new GNN explanation techniques.

Explainability versus Interpretability: Explainable artificial intelligence is an emerging area of research, and the use of terminology is not completely standardized. In particular, the differences of the terms "explainability" and "interpretability" are not completely defined. In some studies, these two terms are used interchangeably [26], while in other studies some subtle differences have been made. In this work, we argue that these two terms should be differentiated. In particular, we follow the existing work [47] to distinguish these two terms. We consider a model to be "interpretable" if the model itself can provide humanly understandable interpretations of its predictions. Note that such a model is no longer a black box to some extent. For example, a decision tree model is an "interpretable" one. Meanwhile, an "explainable" model implies that the model is still a black box whose predictions could potentially be understood by *post hoc* explanation techniques. We articulate such a difference with the hope that the community could eventually develop a standardized use of terminology as this field matures.

2 THE CHALLENGES

Explaining deep graph models is an important but challenging task. In this section, we discuss the common challenges for explaining GNNs.

First, unlike images and texts, graphs are not grid-like data, which means there is no locality information and each node has different numbers of neighbors. Instead, graphs contain important topology information and are represented as feature matrices and adjacency matrices. To explain feature importance, we may directly extend the explanation methods for image data to graph data. However,

the adjacency matrices represent the topology information and only contain discrete values. Then existing methods cannot be directly applied. For example, input optimization methods [10], [16] are popular to explain the general behaviors of image classifiers. It treats the input as trainable variables and optimizes the input via back-propagation to obtain abstract images to explain the model. However, the discrete adjacency matrices cannot be optimized in the same manner. In addition, several methods [14], [48] learn soft masks to capture important image regions. However, applying soft masks to the adjacency matrices will destroy the discreteness property.

In addition, for images and texts, we study the importance of each pixel or word. However, it is more important to study the structural information for graph data. First, the nodes in graphs may be unlabeled and the labels of the whole graphs are determined by graph structures. Then studying each node may be meaningless since those unlabeled nodes contain no semantic meaning. In addition, for graphs in biochemistry, neurobiology, ecology, and engineering, graph substructures are highly related to their functionalities [49]. An example is network motifs, which can be considered as the building blocks of many complex networks [50], [51]. Then such structural information should not be ignored in the explanation tasks. However, existing methods from image domains cannot provide explanations regarding the structures. Next, for node classification tasks, the prediction of each node is determined by different message walks from its neighbors. Investigating such message walks is meaningful but challenging. None of the existing methods in the image domain can consider such walk information, which needs further explorations.

Furthermore, graph data are less intuitive than images and texts. To understand deep models, domain knowledge for the datasets is necessary. For images and texts, understanding the semantic meaning of input data is simple and straightforward. For their explanations, humans can easily understand them even though the explanations are highly abstract. However, since graphs can represent complex data, such as molecules, social networks, and citation networks, it is challenging for humans to understand the meaning of graphs. In addition, in interdisciplinary areas such as chemistry and biology, there are many unsolved mysteries and the domain knowledge is still lacking. Hence, it is non-trivial to obtain human-understandable explanations for graph models, thus raising the need of standard datasets and evaluation metrics for explanation tasks.

3 AN OVERVIEW OF THE TAXONOMY

Recently, several approaches are proposed to explain the predictions of deep graph models. These methods focus on different aspects of the graph models and provide different views to understand these models. They generally answer a few questions; some of those are, which input edges are more important? which input nodes are more important? which node features are more important? what graph patterns will maximize the prediction of a certain class? To better understand these methods, we provide a taxonomy of different explanation techniques for GNNs. The structure of our taxonomy is shown in Figure 1. Based on what
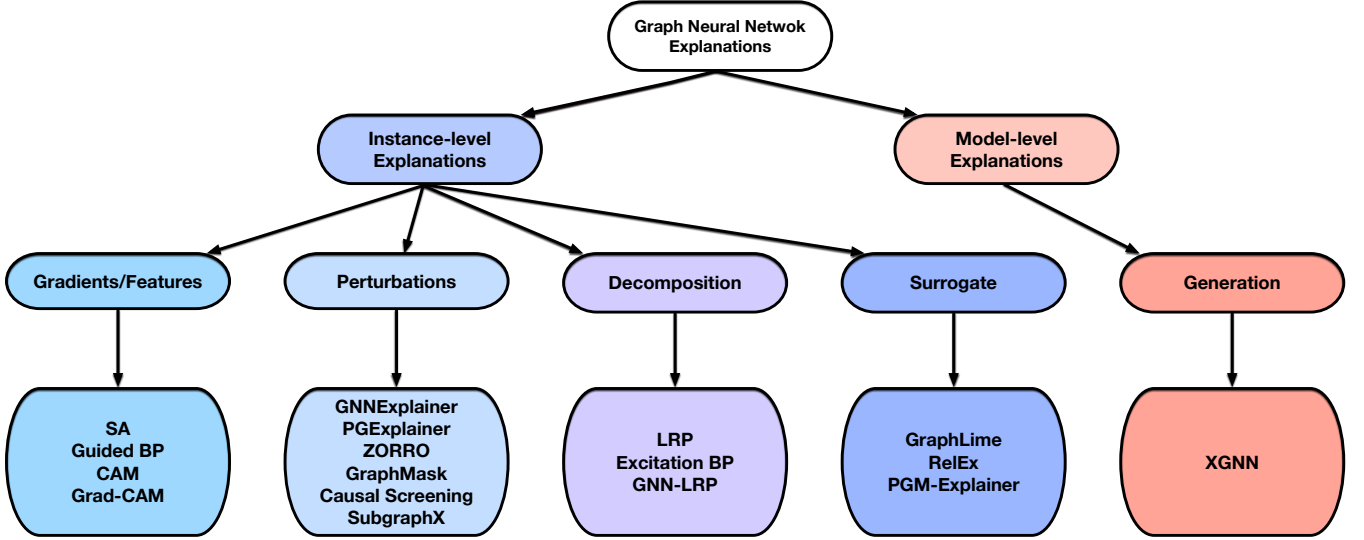
Fig. 1. An overview of our proposed taxonomy. We categorize existing GNN explanation approaches into two branches: instance-level methods and model-level methods. For the instance-level methods, the gradients/features-based methods include SA [52], Guided BP [52], CAM [53], and Grad-CAM [53]; the perturbation-based methods are GNNExplainer [44], PGExplainer [45], ZORRO [54], GraphMask [55], Causal Screening [56], and SubgraphX [46]; the decomposition methods contains LRP [52], [57], Excitation BP [53] and GNN-LRP [58]; the surrogate methods include GraphLime [59], RelEx [60], and PGM-Explainer [61]. For the model-level methods, the only existing approach is XGNN [43].

types of explanations are provided, different techniques are categorized into two main classes: instance-level methods and model-level methods.

First, instance-level methods provide input-dependent explanations for each input graph. Given an input graph, these methods explain deep models by identifying important input features for its prediction. Based on how the importance scores are obtained, we categorize the methods into four different branches: gradients/features-based methods [52], [53], perturbation-based methods [44], [45], [46], [54], [55], [56], decomposition methods [52], [53], [57], [58], and surrogate methods [59], [60], [61]. Specifically, the gradients/features-based methods employ the gradients or the feature values to indicate the importance of different input features. In addition, perturbation-based methods monitor the change of prediction with respect to different input perturbations to study input importance scores. The decomposition methods first decompose the prediction scores, such as predicted probabilities, to the neurons in the last hidden layer. Then they back propagate such scores layer by layer until the input space and use such decomposed scores as the importance scores. Meanwhile, for a given input example, surrogate based methods first sample a dataset from the neighbors of the given example. Next, these methods fit a simple and interpretable model, such as the decision tree, to the sampled dataset. Then the explanations from the surrogate model are used to explain the original predictions. Second, model-level methods explain graph neural networks without respect to any specific input example. The input-independent explanations are high-level and explain general behaviors. Compared with instance-level methods, this direction is still less explored. The only existing model-level method is XGNN [43], which is based on graph generation. It generates graph patterns to maximize the predicted probability for a certain class and uses such graph patterns to explain this class.

Overall, these two categories of methods explain deep graph models from different views. Instance-level methods provide example-specific explanations while model-level methods provide high-level insights and a generic understanding of how deep graph models work. To verify and trust deep graph models, it requires human supervision to check the explanations. For instance-level methods, more human supervisions are needed since experts need to explore the explanations for different input graphs. For model-level methods, since the explanations are high-level, less human supervisions are involved. Furthermore, the explanations of instance-level methods are based on real input instances so that they can be easily understood. However, the explanations for model-level methods may not be human-intelligible since the obtained graph patterns may not even exist in the real-world. Overall, these two types of methods can be combined together to better understand deep graph models, and hence it is necessary to investigate both of them.

4 INSTANCE-LEVEL EXPLANATIONS

We start our explorations by introducing the instance-level methods for GNN explanations, including gradients or features based methods, perturbation-based methods, decomposition methods, and surrogate methods. In this section, we will discuss the motivations, methodologies, advantages, limitations of each category.

4.1 Gradients/Features-Based Methods

First, we introduce the gradients/features-based methods for explaining deep graph models.

4.1.1 A unified view

Employing gradients or features to explain the deep models is the most straightforward solution, which is widely used

in image and text tasks. The key idea is to use the gradients or hidden feature map values as the approximations of input importance. Specifically, gradients-based methods [10], [11] compute the gradients of target prediction with respect to input features by back-propagation. Meanwhile, features-based methods [12], [13] map the hidden features to the input space via interpolation to measure importance scores. Generally, in such methods, larger gradients or feature values indicate higher importance. Note that both gradients and hidden features are highly related to the model parameters, then such explanations can reflect the information contained in the model. Since these methods are simple and general, they can be easily extended to the graph domain. Recently, several methods have been applied to explain graph neural networks, including SA [52], Guided BP [52], CAM [53], and Grad-CAM [53]. The key difference among these methods lies in the procedure of gradient back-propagation and how different hidden feature maps are combined.

4.1.2 Methods

SA [52] directly employs the squared values of gradients as the importance scores of different input features. It can be directly computed by back-propagation, which is the same as network training but the target is input features instead of model parameters. Note that the input features can be graph nodes, edges, or node features. It assumes that higher absolute gradient values indicate the corresponding input features are more important. While it is simple and efficient, it has several limitations. First, SA can only reflect the sensitivity between input and output, which cannot accurately show the importance. In addition, it also suffers from saturation problems [62]. In the saturation regions of the model, where the model output changes minimally with respect to any input change, the gradients can hardly reflect the contributions of inputs.

Guided BP [52] shares a similar idea with SA but modifies the procedure of backs propagating gradients. Since negative gradients are challenging to explain, Guided BP only back propagates positive gradients while clipping negative gradients to zeros. Then only positive gradients are used to measure the importance of different input features. Note that Guided BP shares the same limitations as SA.

CAM [53] maps the node features in the final layer to the input space to identify important nodes. It requires the GNN model to employ a global average pooling (GAP) layer and a fully-connected (FC) layer as the final classifier. Specifically, CAM takes the final node embeddings and combines different feature maps by weighted summations to obtain importance scores for input nodes. Note that the weights are obtained from the final fully-connected (FC) layer connected with the target prediction. This approach is also very simple and efficient but still has several major limitations. First, CAM has special requirements for the GNN structure, which limits its application and generalization. Second, it assumes that the final node embeddings can reflect the input importance, which is heuristic and may not be true. Furthermore, it can only explain graph classification models and cannot be applied to node classification tasks.

Grad-CAM [53] extends the CAM to general graph classification models by removing the constraint of the GAP layer. Similarly, it also maps the final node embeddings to the input space to measure node importance. However, instead of using the weights between the GAP output and FC output, it employs gradients as the weights to combine different feature maps. Specifically, it first computes the gradients of the target prediction with respect to the final node embeddings. Then it averages such gradients to obtain the weight for each feature map. Compared with the CAM, Grad-CAM does not require the GNN model to employ a GAP layer before the final FC layer. However, it is also based on heuristic assumptions and cannot explain node classification models.

4.2 Perturbation-Based Methods

Next, we introduce the perturbation-based methods for explaining deep graph models.

4.2.1 A unified view

The perturbation-based methods are widely employed to explain deep image models [14], [15], [63]. The underlying motivation is to study the output variations with respect to different input perturbations. Intuitively, when important input information is retained, the predictions should be similar to the original predictions. Existing methods [14], [15], [63] learn a generator to generate a mask to select important input pixels to explain deep image models. However, such methods cannot be directly applied to graph models. Unlike images, graphs are represented as nodes and edges, and they cannot be resized to share the same node and edge numbers. In addition, different from images, the structural information is critical for graphs and can determine the functionalities.

To explain deep graph models, several perturbation-based methods are proposed, including GNNExplainer [44], PGExplainer [45], ZORRO [54], GraphMask [55], Causal Screening [56], and SubgraphX [46]. They share a similar high-level pipeline as shown in Figure 2. First, given the input graph, masks are obtained to indicate important input features. Note that depending on the explanation tasks, different masks are generated, such as node masks, edge masks, and node feature masks. Next, the generated masks are combined with the input graph to obtain a new graph containing important input information. Finally, the new graph is fed into the trained GNNs to evaluate the masks and update the mask generation algorithms. Intuitively, the important input features captured by masks should convey the key semantic meaning and hence lead to a similar prediction to the original one. The difference between these methods mainly lies in three aspects: the mask generation algorithm, the type of masks, and the objective function.

It is noteworthy that there are three different types of masks: soft masks, discrete masks, and approximated discrete masks. In Figure 2, we show soft masks for node features, discrete masks for edges, and approximated discrete masks for nodes. The soft masks contain continuous values between $[0, 1]$ and the mask generation algorithm can be directly updated by back-propagation. However, soft masks are suffered from the "introduced evidence" problem [14] that any non-zero or non-one value in the mask may introduce new semantic meaning or new noise to the input
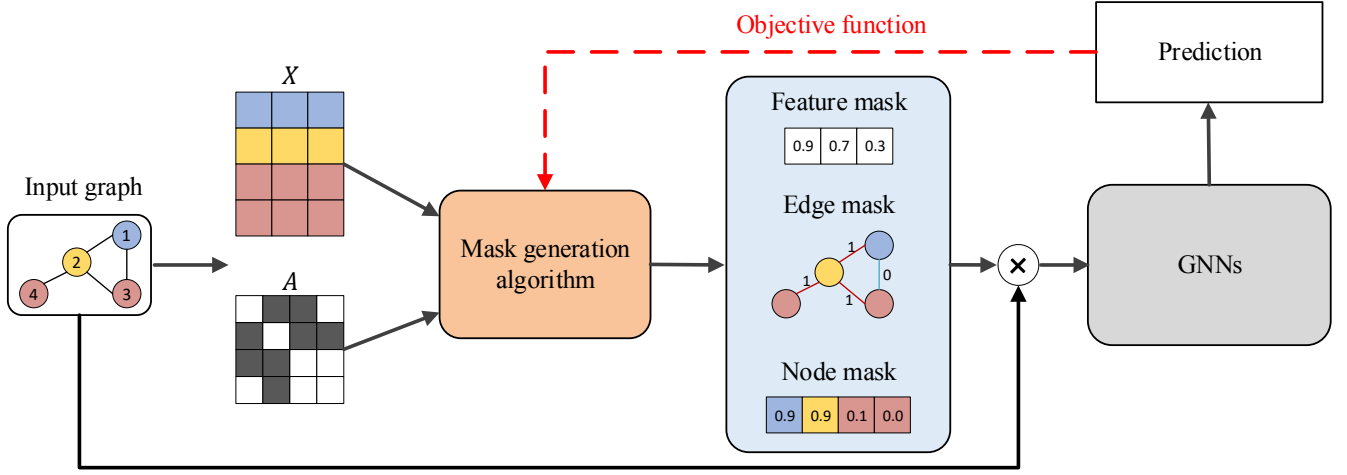
Fig. 2. The general pipeline of the perturbation-based methods. They employ different mask generation algorithms to obtain different types of masks. Note that the mask can correspond to nodes, edges, or node features. In this example, we show a soft mask for node features, a discrete mask for edges, and an approximated discrete mask for nodes. Then the mask is combined with the input graph to capture important input information. Finally, the trained GNNs evaluate whether the new prediction is similar to the original prediction and can provide guidance for improving the mask generation algorithms.

graph, thus affecting the explanation results. Meanwhile, the discrete masks only contain discrete values 0 and 1, which can avoid the "introduced evidence" problem since no new numerical value is introduced. However, discrete masks always involve non-differentiable operations, such as sampling. One popular way to solve it is the policy gradient technique [64]. Furthermore, recent studies [48], [65], [66] propose to employ reparameterization tricks, such as Gumbel-Softmax estimation and sparse relaxations, to approximate the discrete masks. Note that the output mask is not strictly discrete but provides a good approximation, which not only enables the back-propagation but also largely alleviates the "introduced evidence" problem.

4.2.2 Methods

GNNExplainer [44] learns soft masks for edges and node features to explain the predictions via mask optimization. The soft masks are randomly initialized and treated as trainable variables. Then GNNExplainer combines the masks with the original graph via element-wise multiplications. Next, the masks are optimized by maximizing the mutual information between the predictions of the original graph and the predictions of the newly obtained graph. Even though different regularization terms, such as element-wise entropy, are employed to encourage optimized masks to be discrete, the obtained masks are still soft masks so that GNNExplainer cannot avoid the "introduced evidence" problem. In addition, the masks are optimized for each input graph individually and hence the explanations may lack a global view.

PGExplainer [45] learns approximated discrete masks for edges to explain the predictions. It trains a parameterized mask predictor to predict edge masks. Given an input graph, it first obtains the embeddings for each edge by concatenating node embeddings. Then the predictor uses the edge embeddings to predict the probability of each edge being selected, which can be treated as the importance score. Next, the approximated discrete masks are sampled

via the reparameterization trick. Finally, the mask predictor is trained by maximizing the mutual information between the original predictions and new predictions. Note that even though the reparameterization trick is employed, the obtained masks are not strictly discrete but can largely alleviate the "introduced evidence" problem. In addition, since all edges in the dataset share the same predictor, the explanations can provide a global understanding of the trained GNNs.

GraphMask [55] is a post-hoc method for explaining the edge importance in each GNN layer. Similar to the PGExplainer, it trains a classifier to predict whether an edge can be dropped without affecting the original predictions. However, GraphMask obtains an edge mask for each GNN layer while PGExplainer only focuses the input space. In addition, to avoiding changing graph structures, the dropped edges are replaced by learnable baseline connections, which are vectors with the same dimensions as node embeddings. Note that binary Concrete distribution [66] and reparameterization trick is employed to approximate discrete masks. In addition, the classifier is trained using the whole dataset by minimizing a divergence term, which measures the difference between network predictions. Similar to PGExplainer, it can largely alleviate the "introduced evidence" problem and provide a global understanding of the trained GNNs.

ZORRO [54] employ discrete masks to identify important input nodes and node features. Given an input graph, a greedy algorithm is used to select nodes or node features step by step. For each step, ZORRO selects one node or one node feature with the highest fidelity score. Note that the objective function, fidelity score, measure how the new predictions match the original predictions of the model by fixing the selected nodes/features and replacing the others with random noise values. Since there is no training procedure involved, the non-differentiable limitation of discrete masks is avoided. In addition, by using hard masks, ZORRO is not suffered from the "introduced evidence" problem.
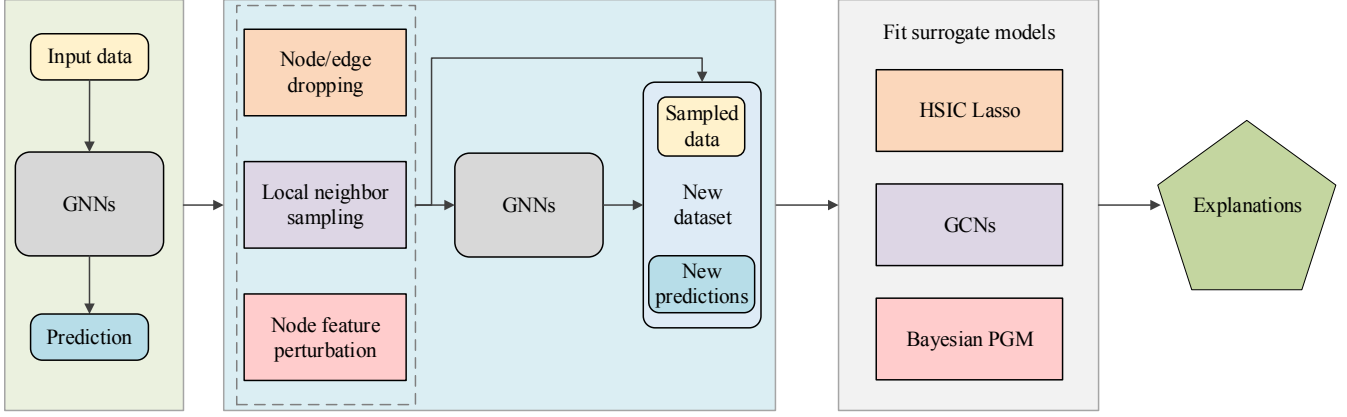
Fig. 3. The general pipeline of the surrogate methods. Given an input graph and its prediction, they first sample a local dataset to represent the relationships around the target data. Then different surrogate methods are applied to fit the local dataset. Note that surrogate models are generally simple and interpretable ML models. Finally, the explanations from the surrogate model can be regarded as the explanations of the original prediction.

However, the greedy mask selection algorithm may lead to local optimal explanations. In addition, the explanations may lack a global understanding since masks are generated for each graph individually.

Causal Screening [56] studies the causal attribution of different edges in the input graph. It identifies an edge mask for the explanatory subgraph. The key idea of causal attribution is to study the change of predictions when adding an edge into the current explanatory subgraph, known as the causal effect. For each step, it studies the causal effects of different edges and selects one edge to add to the subgraph. Specifically, it employs the individual causal effect (ICE) to select edges, which measures mutual information (between the predictions of original graphs and the explanatory subgraphs) difference after adding different edges to the subgraph. Similar to ZORRO, Causal Screening is a greedy algorithm generating discrete masks without any training procedure. Hence, it is not suffered from the "introduced evidence" problem but may lack a global understanding and stuck in local optimal explanations.

SubgraphX [46] explores subgraph-level explanations for deep graph models. It employs the Monte Carlo Tree Search (MCTS) algorithm [67] to efficiently explore different subgraphs via node pruning and select the most important subgraph from the leaves of the search tree as the explanation of the prediction. In addition, it employs Shapley value [68] as the reward of MCTS to measure the importance of subgraphs and proposes an efficient approximation of Shapley value by only considering the interactions within message passing range. While SubgraphX is not directly studying masks, the MCTS algorithm can be understood as the mask generation algorithm and its node pruning actions can be regarded as different masks to obtain subgraphs. In addition, the Shapley values can be treated as the objective function to update the mask generation algorithm. Compared with other perturbation-based methods, its obtained subgraphs are more human-intelligible and suitable for graph data. However, the computational cost is more expensive since it needs to explore different subgraphs with the MCTS algorithm.

4.3 Surrogate Methods

In this section, we introduce the surrogate methods for explaining deep graph models.

4.3.1 A unified view

Deep models are challenging to explain because of the complex and non-linear relationships between the input space and output predictions. A popular way to provide instance-level explanations for image models is known as surrogate method. The underlying idea is to employ a simple and interpretable surrogate model to approximate the predictions of the complex deep model for the neighboring areas of the input example. Note that these methods assume that the relationships in the neighboring areas of the input example is less complex and can be well captured by a simpler surrogate model. Then the explanations from the interpretable surrogate model are used to explain the original prediction. Applying surrogate methods to the graph domain is challenging since graph data are discrete and contain topology information. Then it is not clear how to define the neighboring regions of the input graph and what interpretable surrogate models are suitable.

Recently, several surrogate methods are proposed to explain deep graph models, including GraphLime [59], RelEx [60], and PGM-Explainer [61]. The general pipeline of these methods in shown in Figure 3. To explain the prediction of a given input graph, they first obtain a local dataset containing multiple neighboring data objects and their predictions. Then they fit a interpretable model to learn the local dataset. Finally, the explanations from the interpretable model are regarded as the explanations of the original model for the input graph. While these methods share a similar high-level idea, the key difference lies in two aspects: how to obtain the local dataset and what interpretable surrogate model to use.

4.3.2 Methods

GraphLime [59] extends the LIME algorithm [69] to deep graph models and studies the importance of different node features for node classification tasks. Given a target node in

the input graph, GraphLime considers its N-hop neighboring nodes and their predictions as its local dataset where a reasonable choice of N is the number of layers in the trained GNNs. Then a nonlinear surrogate model, Hilbert-Schmidt Independence Criterion (HSIC) Lasso [70], is employed to fit the local dataset. Note that HSIC Lasso is a kernel-based feature selection algorithm. Finally, based on the weights of different features in HSIC Lasso, it can select important features to explain the HSIC Lasso predictions. Those selected features are regarded as the explanations of the original GNN prediction. However, GraphLime can only provide explanations for node features but ignore graph structures, such as nodes and edges, which are more important for graph data. In addition, GraphLime is proposed to explain node classification predictions but cannot be directly applied to graph classification models.

RelEx [60] also studies the explainability of node classification models by combining the ideas of surrogate methods and perturbation-based methods. Given a target node and its computational graph (N-hop neighbors), it first obtains a local dataset by randomly sampling connected subgraphs from the computational graph and feeding these subgraphs to the trained GNNs to obtain their predictions. Specifically, starting from the target node, it randomly selects neighboring nodes in a BFS manner. Next, it employs a GCN model as the surrogate model to fit the local datasets. Note that different from LIME and GraphLime, the surrogate model in RelEx is not interpretable. After training, it further applies the aforementioned perturbation-based methods, such as generating soft masks or Gumbel-Softmax masks, to explain the predictions. Compared with GraphLime, it can provide explanations regarding important nodes. However, it contains multiple steps of approximations, such as using the surrogate model to approximate local relationships and using masks to approximate the edge importance, thus making the explanations less convincing and trustable. Furthermore, as perturbation-based methods can be directly employed to explain original deep graph models, then it is not necessary to build another non-interpretable deep model as the surrogate model to explain. It is also unknown how it can be applied for graph classification tasks.

PGM-Explainer [61] builds a probabilistic graphical model to provide instance-level explanations for GNNs. The local dataset is obtained by random node feature perturbation. Specifically, given an input graph, each time PGM-Explainer randomly perturbs the node features of several random nodes within the computational graph. Then for any node in the computational graph, PGM-Explainer records a random variable indicating whether its features are perturbed and its influence on the GNN predictions. By repeating such procedures multiple times, a local dataset is obtained. Note that the local dataset of PGM-Explainer contains node variables instead of different neighboring graph samples. Then it selects top dependent variables to reduce the size of the local dataset via the Grow-Shrink (GS) algorithm [71]. Finally, an interpretable Bayesian network is employed to fit the local dataset and to explain the predictions of the original GNN model. PGM-Explainer can provide explanations regarding graph nodes but ignore graph edges, which contain important graph topology information. In addition, different from GraphLime and

RelEx, the PGM-Explainer can be used to explain both node classification and graph classification tasks.

4.4 Decomposition Methods

Finally, we introduce the decomposition methods for explaining deep graph models.

4.4.1 A unified view

Another popular way to explain deep image classifiers is decomposition methods, which measure the importance of input features by decomposing the original model predictions into several terms. Then those terms are regarded as the importance scores of the corresponding input features. These methods directly study the model parameters to reveal the relationships between the features in the input space and the output predictions. Note that the conservative property of these methods requires that the sum of the decomposed terms is equal to the original prediction score. However, it is challenging to directly apply such methods to the graph domain since graphs contain nodes, edges, and node features. It is non-trivial to distribute scores to different edges while graph edges contain important structural information that cannot be ignored.

Recently, several decomposition methods are proposed to explain the deep graph neural networks, including Layer-wise Relevance Propagation (LRP) [52], [57], Excitation BP [53] and GNN-LRP [58]. The intuition of these algorithms is to build score decomposition rules to distribute the prediction scores to the input space. The general pipeline of these methods is shown in Figure 4. These methods distribute the prediction score layer by layer in a back-propagation manner until the input layer. Starting from the output layer, the model's prediction is treated as the initial target score. Then the score is decomposed and distributed to the neurons in the previous layer following the decomposition rules. By repeating such procedures until input space, they can obtain importance scores for node features, which can be combined to represent edge importance, node importance, and walk importance. It is noteworthy that all of these algorithms ignore the activation functions in deep graph models. The main difference among these methods is the score decomposition rules and the targets of explanations.

4.4.2 Methods

LRP [52], [57] extends the original LRP algorithm [72] to deep graph models. It decomposes the output prediction score to different node importance scores. The score decomposition rule is developed based on the hidden features and weights. For a target neuron, its score is represented as a linear approximation of neuron scores from the previous layer. Intuitively, the neuron with a higher contribution of the target neuron activation receives a larger fraction of the target neuron score. Specifically, Baldassarre *et al.* [52] applies the ϵ-stabilized rule from the original LRP, while Schwarzenberg *et al.* [57] employs the z^+-rule to perform decomposition. To satisfy the conservative property, the adjacency matrix is treated as a part of GNN model in the post-hoc explanation phase so that it can be ignored during score distribution; otherwise, the adjacent matrix will also receive decomposed scores, thus making the conservative
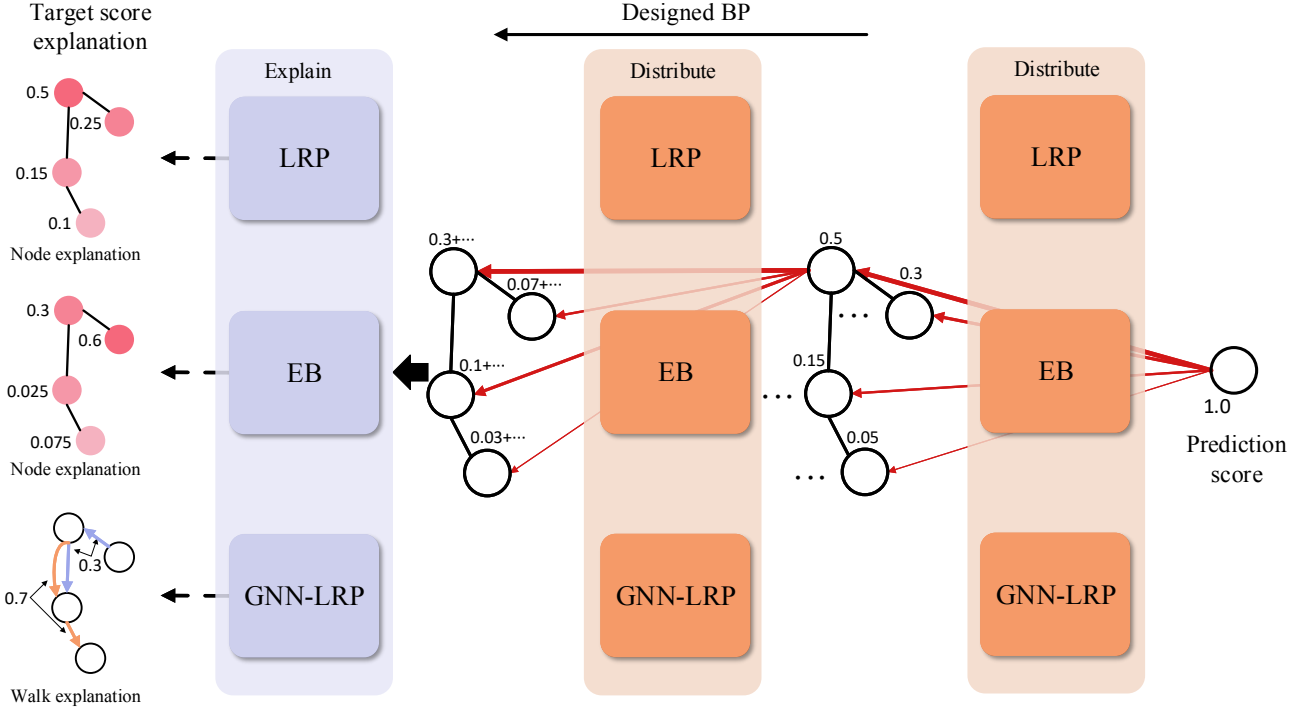
Fig. 4. The general pipeline of the decomposition methods. These methods distribute the prediction score to input space to indicate the input importance. The target score is decomposed layer by layer in a back-propagation manner. Note that the main difference among these methods is the employed score decomposition rule.

property invalid. Since LRP is directly developed based on the model parameters, its explanation results are more trustable. However, it can only study the importance of different nodes and cannot be applied to graph structures, such as subgraphs and graph walks, which is more important for understanding GNNs. In addition, such a algorithm requires a comprehensive understanding of the model structures, which limits its applications for non-expert users, such as interdisciplinary researchers.

Excitation BP [53] shares a similar idea as the LRP algorithm but is developed based on the law of total probability. It defines that the probability of a neuron in the current layer is equal to the total probabilities it outputs to all connected neurons in the next layer. Then the score decomposition rule can be regarded as decomposing the target probability into several conditional probability terms. Note that the computation of Excitation BP is highly similar to the z^+-rule in LRP. Hence, it shares the same advantages and limitations as the LRP algorithm.

GNN-LRP [58] studies the importance of different graph walks. It is more coherent to the deep graph neural networks since graph walks correspond to message flows when performing neighborhood information aggregation. The score decomposition rule is the high-order Taylor decomposition of model predictions. It is shown that the Taylor decomposition (at root zero) only contains T-order terms where T is the number of layers in the trained GNNs. Then each term corresponds to a T-step graph walk and can be regarded as its importance score. Since it is not possible to directly compute the high-order derivatives given by Taylor expansion, GNN-LRP also follows a back propagation procedure to approximate the T-order terms. Note that the back

propagation computation in GNN-LRP is similar to the LRP algorithm. However, instead of distributing scores to nodes or edges, GNN-LRP distributes scores to different graph walks. It records the paths of the distribution processes from layer to layer. Those paths are considered as different walks and the scores are obtained from their corresponding nodes. While GNN-LRP has a solid theoretical background, the approximations in its computations may not be accurate. In addition, the computational complexity is high since each walk is considered separately. Furthermore, it is also challenging for non-experts to use, especially for interdisciplinary domains.

5 MODEL-LEVEL EXPLANATIONS

Different from instance-level methods, model-level methods aim at providing the general insights and high-level understanding to explain deep graph models. Specifically, they study what input graph patterns can lead to a certain GNN behavior, such as maximizing a target prediction. Input optimization [16] is a popular direction to obtain model-level explanations for image classifiers. However, it cannot be directly applied to graph models due to the discrete graph topology information, thus making explaining GNNs at the model-level more challenging. Hence, it is still an important but less studied topic. To the best of our knowledge, the only existing model-level method for explaining graph neural networks is XGNN [43].

XGNN proposes to explain GNNs via graph generation. Instead of directly optimizing the input graph, it trains a graph generator so that the generated graphs can maximize a target graph prediction. Then the generated graphs are

TABLE 1
A comprehensive analysis of different explanation methods. Here "Type" indicates what type of explanations are provided, "Learning" denotes whether learning procedures are involved, "Task" means what tasks each method can be applied to, "Target" indicates the targets of explanations, "Black-box" means if the trained GNNs are treated as a black-box during the explanation stage, "Flow" denotes the computational flow for explanations, and "Design" indicates whether an explanation method has specific designs for graph data. Note that GC denotes graph classification, NC denotes node classification, N means nodes, E means edges, NF represents node features, and Walk indicates graph walks.

Method	TYPE	LEARNING	TASK	TARGET	BLACK-BOX	FLOW	DESIGN
SA [52], [53]	Instance-level	✗	GC/NC	N/E/NF	✗	Backward	✗
Guided BP [52]	Instance-level	✗	GC/NC	N/E/NF	✗	Backward	✗
CAM [53]	Instance-level	✗	GC	N	✗	Backward	✗
Grad-CAM [53]	Instance-level	✗	GC	N	✗	Backward	✗
GNNExplainer [44]	Instance-level	✓	GC/NC	E/NF	✓	Forward	✓
PGExplainer [45]	Instance-level	✓	GC/NC	E	✗	Forward	✓
GraphMask [55]	Instance-level	✓	GC/NC	E	✗	Forward	✓
ZORRO [54]	Instance-level	✗	GC/NC	N/NF	✓	Forward	✓
Causal Screening [56]	Instance-level	✗	GC/NC	E	✓	Forward	✓
SubgraphX [46]	Instance-level	✓	GC/NC	Subgraph	✓	Forward	✓
LRP [52], [57]	Instance-level	✗	GC/NC	N	✗	Backward	✗
Excitation BP [53]	Instance-level	✗	GC/NC	N	✗	Backward	✗
GNN-LRP [58]	Instance-level	✗	GC/NC	Walk	✗	Backward	✓
GraphLime [59]	Instance-level	✓	NC	NF	✓	Forward	✗
RelEx [60]	Instance-level	✓	NC	N/E	✓	Forward	✓
PGM-Explainer [61]	Instance-level	✓	GC/NC	N	✓	Forward	✓
XGNN [43]	Model-level	✓	GC	Subgraph	✓	Forward	✓

regarded as the explanations for the target prediction and are expected to contain discriminative graph patterns. In XGNN, the graph generation is formulated as a reinforcement learning problem. For each step, the generator predicts how to add an edge to the current graph. Then the generated graphs are fed into the trained GNNs to obtain feedback to train the generator via policy gradient. In addition, several graph rules are incorporated to encourage the explanations to be both valid and human-intelligible. Note that XGNN is a general framework for generating model-level explanations so that any suitable graph generation algorithm can be applied. In addition, the explanations are general and provide a global understanding of the trained GNNs. However, the proposed XGNN only demonstrates its effectiveness in explaining graph classification models and it is unknown whether XGNN can be applied to node classification tasks, which is an important direction to explore in future studies.

6 A COMPREHENSIVE COMPARATIVE ANALYSIS

In this section, we provide a comprehensive analysis of different explanation methods. The properties of different models are concluded in Table 1. Specifically, for each technique, we focus on the following aspects.

- **Type:** It indicates the type of explanations provided by each explanation technique, including instance-level explanations and model-level explanations. Instance-level explanations are input-dependent and more specific while model-level explanations are more high-level and general.
- **Learning**: It denotes whether the explanation technique involves any learning procedure. Generally, the explanation method with learning procedures tends to better capture the relationships between inputs and predictions. However, they become less trustable when the learning procedures introduce

additional black-boxes. As shown in Table 1, there is no learning procedure in the feature/gradient-based methods and decomposition methods.
- **Task**: It indicates what tasks each method can be applied to. Here we only consider node classification (NC) and graph classification (GC) tasks. It demonstrates the generalizability of different methods.
- **Target**: It shows the target explanation of each method, such as showing node importance, edge importance, or graph walk importance. In Table 1, N denotes nodes, E means edges, NF denotes node features, and Walk denotes graph walks. For graph models, it is more important to study the structures of input graphs, such as edges, graph walks, and subgraphs.
- **Black-box**: It indicates whether the explanation technique treats the trained GNNs as a black-box. Several methods, including XGNN, GraphLime, PGM-Explainer, GNNExplainer, ZORRO, Causal Screening, and RelEx, only need to access the inputs and outputs to explain the trained GNNs, which means the trained GNNs are still treated as a black-box. For the other methods, they need to access the parameters or hidden representations of the trained GNNs. Note that by treating the trained GNNs as a black-box, the explanation method can be better generalized to applications where the GNN model is encapsulated or highly complex.
- **Flow**: It denotes the computational flow for explaining the trained GNNs. Several methods generate explanations in a backward manner, such as SA, Guided BP, CAM, Grad-CAM, LRP, Excitation BP, and GNN-LRP. The other methods only need the forward computations of the trained GNNs.
- **Design**: It shows whether an explanation technique has specific designs for graph data or is simply ex-

tended from the image domain. Since graph data are special and their topology/structural information is important, it is necessary to explicitly consider such information to provide explanations.

7 EVALUATIONS

In this section, we introduce the evaluations of graph model explanation techniques. Intuitively, good explanation results should faithfully explain the behaviors of GNN models. However, evaluating the explanation results is non-trivial due to the lack of ground truths. Hence, we discuss and analyze several commonly used datasets and metrics.

7.1 Datasets

First, it is critical to select proper datasets to evaluate different explanation techniques. Since human evaluations are necessary, the explanations need to be understandable by humans. Hence, it is desired that the data are intuitive and easy to visualize. In addition, good datasets should contain human-understandable rationales between the data examples and labels so that experts can verify whether such rationales are identified by explanation algorithms. To evaluate different explanation techniques, several types of datasets are commonly employed, including synthetic data, sentiment graph data, and molecular data.

7.1.1 Synthetic data

Recently, several synthetic datasets are built to evaluate explanation techniques [44], [45]. In such datasets, different graph motifs are included and can determine the node labels or graph labels. In addition, the relationships between data examples and data labels are well defined by humans. Even though the trained GNNs may not perfectly capture such relationships, the graph motifs can be employed as reasonable approximations of the ground truths of explanation results. Here we introduce several common synthetic datasets.

1) BA-shapes: It is a node classification dataset with 4 different node labels. For each graph, it contains a base graph (300 nodes) and a house-like five-node motif. Note that the base graph is obtained by the *Barabási-Albert* (BA) model, which can generate random scale-free networks with a preferential attachment mechanism [73]. The motif is attached to the base graph while random edges are added. Each node is labeled based on whether it belongs to the base graph or different spatial locations of the motif.
2) BA-Community: It is a node classification dataset with 8 different labels. For each graph, it is obtained by combining two BA-shapes graphs with randomly added edges. Node labels are determined by the memberships of BA-shapes graphs and their structural locations.
3) Tree-Cycle: It is a node classification dataset with 2 different labels. For each graph, it consists of a base balanced tree graph with the depth equal to 8 and a six-node cycle motif. These two components are randomly connected. The label for the nodes in based graphs is 0 otherwise 1.

TABLE 2
Statistics and properties of our sentiment graph datasets. We report the prediction accuracies of three GNN models on our datasets. Note that "BERT accuracy" denotes the fine-tuning results of a pre-trained twelve-layer base BERT using original sentence data.

Dataset	Graph-SST2	Graph-SST5	Graph-Twitter
# of classes	2	5	3
# of features	768	768	768
Avg. # of nodes	10.199	19.849	21.103
# of train graphs	67,349	8,544	4,998
# of val. graphs	872	1,101	1,250
# of test graphs	1,821	2,210	692
GCN accuracy	0.892	0.443	0.614
GAT accuracy	0.901	0.462	0.601
GIN accuracy	0.873	0.463	0.624
BERT accuracy	0.912	0.532	0.736

4) Tree-Grids: It is a node classification dataset with 2 different labels. It is the same as the Tree-Cycle dataset, except that the Tree-Grids dataset employs nine-node grid motifs instead of cycle motifs.
5) BA-2Motifs: It is a graph classification dataset with 2 different graph labels. There are 800 graphs and each of them is obtained by attaching different motifs, such as the house-like motif and the five-node cycle motif, to the base BA graph. Different graphs are labeled based on the type of motif.

Note that in these datasets, all node features are initialized as vectors containing all 1s. The trained GNNs models are expected to capture graph structures to make predictions. Then based on the rules of building each dataset, we can analyze the explanation results. For example, in the BA-2Motifs dataset, we can study whether the explanations can capture the motif structures. However, synthetic datasets only contain simple relationships between graphs and labels, which may be not enough for comprehensive evaluations.

7.1.2 Sentiment graph data

Traditional graph datasets are challenging to understand since humans only have limited domain knowledge. It raises the need of building human-understandable graph datasets. Text data can be a proper choice for graph explanation tasks since they consist of words and phrases with understandable semantic meanings. Then the explanation results can be easily evaluated by humans. Hence, we build three sentiment graph datasets based on text sentiment analysis data, including SST2 [74], SST5 [74], and Twitter [75] datasets.

First, for each text sequence, we convert it to a graph that each node represents a word while the edges reflect the relationships between different words. Specifically, we employ the Biaffine parser [76] to extract word dependencies. We show an example of our generated sentiment graph in Figure 5. Note that the generated graphs are directed but the edge labels are ignored since most GNNs cannot capture edge label information. Next, we employ BERT [77] to obtain the word embeddings and such embeddings are used as initial embeddings of graph nodes. Specifically, we use a pre-trained twelve-layer base BERT model to extract a 768-dimension feature vector for each word.

We build three sentiment graph datasets, known as Graph-SST2, Graph-SST5, and Graph-Twitter. They are now
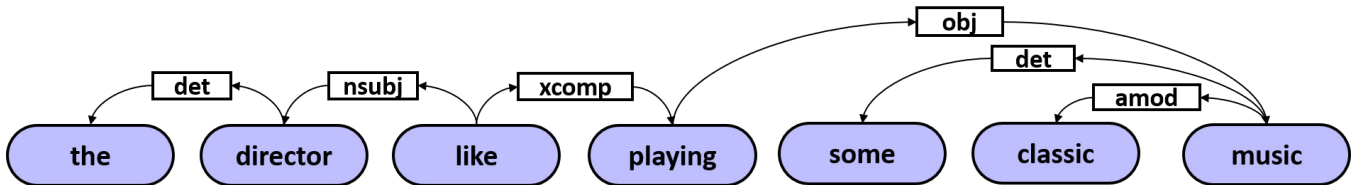
Fig. 5. A data example of our graph sentiment datasets. Each node represents a word while edges indicates the relationship between the words. Note that edges are directed but their labels are ignored.

publicly available and can be directly used to study different explanation techniques[1]. The statistics and properties of these datasets are reported in Table 2. We report the prediction accuracies of two-layer GNNs on these datasets, including GCNs, GATs, and GINs. In addition, we also show the fine-tuned accuracies of a pre-trained twelve-layer base BERT [77] using the original sentence datasets. Since the results of BERT are obtained by finetuning the model and word embeddings, it is expected to perform better than the GNN models without finetuned node embeddings. Notably, we choose to use the word embeddings as the node embeddings without finetuning because finetuning may encode the label information into the word embeddings, which affect the structural information captured by GNNs and the final explanations. Hence, we believe these datasets are reasonable choices for building graph models to explain. Based on the semantic meaning of different words and the sentiment labels, we can study whether the explanations can identify the words with key meanings and the relationships among different words.

7.1.3 Molecule data

Molecular datasets are also widely used in explanation tasks, such as MUTAG [78], BBBP, and Tox21 [79]. Each graph in such datasets corresponds to a molecule where nodes represent atoms and edges are the chemical bonds. The labels of molecular graphs are generally determined by the chemical functionalities or properties of the molecules. Employing such datasets for explanation tasks requires domain knowledge, such as what chemical groups are discriminative for their functionalities. For example, in the dataset MUTAG, different graphs are labeled based on their mutagenic effects on a bacterium. In addition, it is known that carbon rings and NO_2 chemical groups may lead to mutagenic effects [78]. Then we can study whether the explanations can identify such patterns for the corresponding class.

7.2 Evaluation Metrics

Even though visualization results can provide an insightful understanding regarding whether the explanations are reasonable to humans, such evaluations are not fully trustable due to the lack of ground truths. In addition, to compare different explanation methods, humans need to study the results for each input example, which is time-consuming. Furthermore, human evaluations are highly dependent on their subjective understanding, which is not fair enough.

1. https://github.com/divelab/DIG/tree/main/dig/xgraph/datasets

Hence, evaluation metrics are crucial for studying explanation methods. Good metrics should evaluate the results from the model's perspective, such as whether the explanations are faithful to the model [80], [81]. In this section, we introduce several recently proposed evaluation metrics for explanation tasks.

7.2.1 Fidelity

First, the explanations should be faithful to the model. They should identify input features that are important for the model, not our humans. To evaluate this, the Fidelity+ [53] metric is recently proposed. Intuitively, if important input features (nodes/edges/node features) identified by explanation techniques are discriminative to the model, the predictions should change significantly when these features are removed. Hence, Fidelity+ is defined as the difference of accuracy (or predicted probability) between the original predictions and the new predictions after masking out important input features [53], [82].

Formally, let $\mathcal{G}_i$ denote the i-th input graph and $f(\cdot)$ denote the GNN classifier to be explained. The prediction result of this graph is represented as $\hat{y}_i = \mathbf{argmax}\, f(\mathcal{G}_i)$. Then its explanations can be considered as a hard importance map m_i where each element is 0 or 1 to indicate if the corresponding feature is important. Note that for methods like ZORRO [54] and Causal Screening [56], the generated explanations are discrete masks, which can be directly used as the importance map. In addition, for methods like GNNExplainer [44] and GraphLime [59], the importance scores are continuous values, then the importance map m_i can be obtained by normalization and thresholding. Finally, the Fidelity+ score of prediction accuracy can be computed as

$$Fidelity+^{acc} = \frac{1}{N}\sum_{i=1}^{N}(\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i^{1-m_i} = y_i)), \quad (1)$$

where y_i is the original prediction of graph i and N is the number of graphs. Here $1 - m_i$ means the complementary mask that removes the important input features and $\hat{y}_i^{1-m_i}$ is the prediction when feeding the new graph into trained GNN $f(\cdot)$. The indicator function $\mathbb{1}(\hat{y}_i = y_i)$ returns 1 if $\hat{y}_i$ and y_i are equal and returns 0 otherwise. Note that the $Fidelity+^{acc}$ metric studies the change of prediction accuracy. By focusing on the predicted probability, the Fidelity+ of probability can be defined as

$$Fidelity+^{prob} = \frac{1}{N}\sum_{i=1}^{N}(f(\mathcal{G}_i)_{y_i} - f(\mathcal{G}_i^{1-m_i})_{y_i}), \quad (2)$$

where $\mathcal{G}_i^{1-m_i}$ represents the new graph obtained by keeping features of $\mathcal{G}_i$ based on the complementary mask $1 - m_i$.

Note that $Fidelity+^{prob}$ monitors the change of predicted probability, which is more sensitive than $Fidelity+^{acc}$. For both metrics, higher values indicate better explanations results and more discriminative features are identified.

The Fidelity+ metric studies the prediction change by removing important nodes/edges/node features. In contrast, the metric Fidelity- studies prediction change by keeping important input features and removing unimportant features. Intuitively, important features should contain discriminative information so that they should lead to similar predictions as the original predictions even unimportant features are removed. Formally, the metric Fidelity- can be computed as

$$Fidelity-^{acc} = \frac{1}{N}\sum_{i=1}^{N}(\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i^{m_i} = y_i)), \quad (3)$$

$$Fidelity-^{prob} = \frac{1}{N}\sum_{i=1}^{N}(f(\mathcal{G}_i)_{y_i} - f(\mathcal{G}_i^{m_i})_{y_i}), \quad (4)$$

where $\mathcal{G}_i^m$ is the new graph when keeping important features of $\mathcal{G}_i$ based on explanation m_i and $\hat{y}_i^{m_i}$ is the new prediction. Note that for both $Fidelity-^{acc}$ and $Fidelity-^{prob}$, lower values indicate less importance information are removed so that the explanations results are better.

7.2.2 Sparsity

Second, good explanations should be sparse, which means they should capture the most important input features and ignore the irrelevant ones. The metric Sparsity measures such a property. Specifically, it measures the fraction of features selected as important by explanation methods [53]. Formally, give the graph $\mathcal{G}_i$ and its hard importance map m_i, the Sparsity metric can be computed as

$$Sparsity = \frac{1}{N}\sum_{i=1}^{N}(1 - \frac{|m_i|}{|M_i|}), \quad (5)$$

where $|m_i|$ denotes the number of important input features (nodes/edges/node features) identified in m_i and $|M_i|$ means the total number of features in $\mathcal{G}_i$. Note that higher values indicate the explanations are more sparse and tend to only capture the most important input information.

7.2.3 Stability

In addition, good explanations should be stable. Intuitively, when small changes are applied to the input without affecting the predictions, the explanations should remain similar. The recent proposed Stability metric measures whether an explanation method is stable [83]. Given an input graph $\mathcal{G}_i$, its explanations m_i is regarded as the ground truth. Then the input graph $\mathcal{G}_i$ is perturbed by small changes, such as attaching new nodes/edges, to obtain a new graph $\hat{\mathcal{G}}_i$. Note that $\mathcal{G}_i$ and $\hat{\mathcal{G}}_i$ are required to have the same predictions. Then the explanations of $\hat{\mathcal{G}}_i$ is obtained, denoted as $\hat{m}_i$. By comparing the difference between m_i and $\hat{m}_i$, we can compute the Stability score. Note that lower values indicate the explanation technique is more stable and more robust to noisy information. In addition, since graph representations are sensitive, selecting a proper amount of perturbations may be challenging.

7.2.4 Accuracy

Furthermore, the Accuracy metric is proposed for synthesis datasets [44], [83]. In synthesis datasets, even though it is unknown whether the GNNs make predictions in our expected way, the rules of building these datasets, such as graph motifs, can be used as reasonable approximations of the ground truths. Then for any input graph, we can compare its explanations with such ground truths. For example, when studying important edges, we can study the matching rate for important edges in explanations compared with those in the ground truths. The common metrics for such comparisons include general accuracy, F1 score, ROC-AUC score. Note that higher values indicate the explanations are closer to ground truths and can be considered as better results. In addition, the Accuracy metric cannot be applied to real-world datasets due to the lack of ground truths.

7.2.5 Discussions

It is noteworthy that different metrics should be combined to evaluate explanation results. For example, Sparsity and Fidelity+/Fidelity- are highly correlated. When the explanation results are soft values, the Sparsity is determined by a threshold value. Intuitively, larger threshold values tend to identify fewer features as important and hence increase the Sparsity score and decrease the Fidelity+ score. Hence, to fairly compare different explanation methods, we suggest comparing their Fidelity+ scores with the same level of Sparsity scores. One possible way is to select a fixed percentage of input features as important, and then compare their Fidelity+ or Fidelity- scores.

In addition, there are several other metrics to evaluate explanation results, such as Contrastivity [53] and Consistency [83]. The Contrastivity score is based on a strong assumption that the explanations for different classes should be significantly different, which ignores the common patterns among different classes. Meanwhile, the Consistency metric assumes that high-performing model architectures should have consistent explanations. However, different models may capture different relationships even though they achieve competitive performance, especially for large-scale and complex datasets. Hence, we believe such metrics may not be fair enough to evaluate explanation results and they are not listed as recommended metrics.

8 AN OPEN-SOURCE LIBRARY

To facilitate research into graph neural networks, we develop an open-source library, known as DIG: Dive into Graphs. A module in the library is devoted to explainability in GNNs, which is now publicly available[2]. Our library can be directly used to reproduce results of existing GNN explanation methods, develop new algorithms, and conduct evaluations for explanation results. First, in this library, we include the implementations of several existing methods, such as GNNExplainer [44], PGExplainer [45], DeepLIFT [62], GNN-LRP [58], Grad-CAM [53], SubgraphX [46], and XGNN [43]. While several methods provide official implementations, their development environments may vary, thus making fair comparisons difficult. To this end, we implement these methods in the same

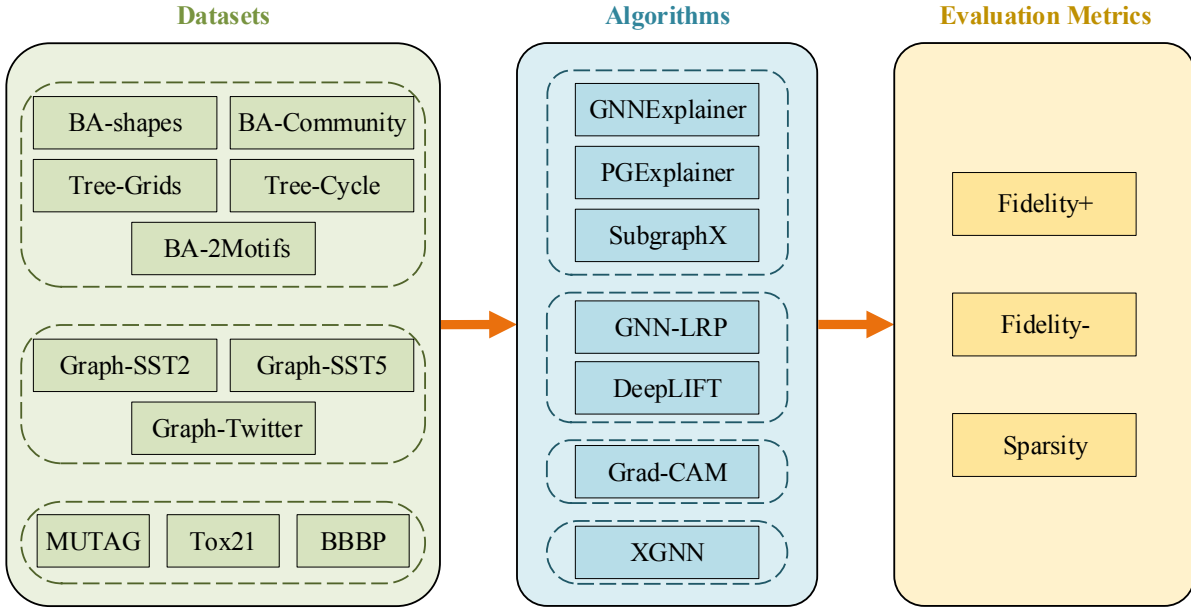2. https://github.com/divelab/DIG/tree/main/dig/xgraph

Fig. 6. A overview of our open-source library, which includes unified implementations of multiple common baseline methods, commonly used datasets, and three evaluation metrics.

development environment that the key dependencies of our library are PyTorch 1.6.0 [84] and PyTorch Geometric 1.6.1 [85]. With such a unified environment, new techniques can be easily and fairly compared with existing methods. Second, we include several datasets for GNN explanation tasks in our library. Note that we not only provide the aforementioned sentiment graph datasets, but also include several commonly used datasets, such as MUTAG, BBBP, BA-shapes, and BA-2Motifs, etc. These datasets include synthetic data, text data, and molecular data, and cover both graph classification tasks and node classification tasks. Hence, we believe that using such datasets can provide a comprehensive study of different GNN explanation methods. In addition, our library provides implementations of different evaluation metrics, including Fidelity+, Fidelity-, and Sparsity. These metrics can be directly applied to evaluate the explanation results obtained from our framework and provide comprehensive evaluations from the model's perspective.

9 CONCLUSION

Graph Neural Networks are widely studied recently but the explainability of graph models is still less explored. To investigate the underlying mechanisms of these black-boxes, several explanation methods are proposed to explain graph models, including XGNN, GNNExplainer, etc. These methods explain graph models from different views and motivations, thus lacking a comprehensive study and analysis of these methods. In this work, we provide a systematic and comprehensive survey for these methods. We first propose a novel taxonomy of existing GNN explanation methods and introduce the key idea behind each category. Then we discuss each explanation method in detail, including the methodology, insights, advantages, and drawbacks. In

addition, we provide a comprehensive analysis of different explanation methods. Furthermore, we introduce and analyze the commonly employed datasets and metrics for evaluating GNN explanation methods. Last but not the least, we build three graph datasets from text data, which are human-understandable and can be directly used for GNN explanation tasks.

REFERENCES

[1] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2013.
[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT (1)*, 2019.
[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.
[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
[6] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km
[7] Z. Wang, M. Liu, Y. Luo, Z. Xu, Y. Xie, L. Wang, L. Cai, and S. Ji, "MoleculeKit: Machine learning methods for molecular property prediction and drug discovery," *arXiv preprint arXiv:2012.01981*, 2020.
[8] H. Mao, X. Liu, N. Duffield, H. Yuan, S. Ji, and B. Mohanty, "Context-aware deep representation learning for geo-spatiotemporal analysis," in *Proceedings of the IEEE International Conference on Data Mining*, 2020.

[9] H. Yuan, L. Cai, Z. Wang, X. Hu, S. Zhang, and S. Ji, "Computational modeling of cellular structures using conditional deep generative networks," *Bioinformatics*, vol. 35, no. 12, pp. 2141–2149, 2019.

[10] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[11] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," *arXiv preprint arXiv:1706.03825*, 2017.

[12] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2921–2929.

[13] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 618–626.

[14] P. Dabkowski and Y. Gal, "Real time image saliency for black box classifiers," in *Advances in Neural Information Processing Systems*, 2017, pp. 6967–6976.

[15] H. Yuan, L. Cai, X. Hu, J. Wang, and S. Ji, "Interpreting image classifiers by generating discrete masks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[16] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017, https://distill.pub/2017/feature-visualization.

[17] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, "The building blocks of interpretability," *Distill*, 2018, https://distill.pub/2018/building-blocks.

[18] F. Yang, S. K. Pentyala, S. Mohseni, M. Du, H. Yuan, R. Linder, E. D. Ragan, S. Ji, and X. Hu, "Xfake: explainable fake news detector with visualizations," in *The World Wide Web Conference*, 2019, pp. 3600–3604.

[19] M. Du, N. Liu, Q. Song, and X. Hu, "Towards explanation of dnn-based prediction with guided feature inversion," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1358–1367.

[20] Z. Wang, X. Hu, and S. Ji, "iCapsNets: Towards interpretable capsule networks for text classification," *arXiv preprint arXiv:2006.00075*, 2020.

[21] M. Du, N. Liu, F. Yang, S. Ji, and X. Hu, "On attribution of recurrent neural network predictions via additive decomposition," in *Proceedings of the Web Conference*, 2019, pp. 383–393.

[22] H. Yuan, Y. Chen, X. Hu, and S. Ji, "Interpreting deep models for text analysis via optimization and regularization methods," in *AAAI-19: Thirty-Third AAAI Conference on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence, 2019.

[23] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Communications of the ACM*, vol. 63, no. 1, pp. 68–77, 2019.

[24] A. Rai, "Explainable ai: From black box to glass box," *Journal of the Academy of Marketing Science*, vol. 48, no. 1, pp. 137–141, 2020.

[25] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE, 2018, pp. 0210–0215.

[26] C. Molnar, *Interpretable Machine Learning*, 2019, https://christophm.github.io/interpretable-ml-book/.

[27] Y. Ma and J. Tang, *Deep Learning on Graphs*. Cambridge University Press, 2020.

[28] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159.

[29] H. Gao and S. Ji, "Graph u-nets," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[30] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[31] M. Liu, H. Gao, and S. Ji, "Towards deeper graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020, pp. 338–348.

[32] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification." in *AAAI*, vol. 18, 2018, pp. 4438–4445.

[33] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 5165–5175.

[34] L. Cai and S. Ji, "A multi-scale approach for graph link prediction," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020.

[35] L. Cai, J. Li, J. Wang, and S. Ji, "Line graph neural networks for link prediction," *arXiv preprint arXiv:2010.10046*, 2020.

[36] M. Liu, Z. Wang, and S. Ji, "Non-local graph neural networks," *arXiv preprint arXiv:2005.14612*, 2020.

[37] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 1416–1424.

[38] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[39] K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li, "Attention-based graph neural network for semi-supervised learning," *arXiv preprint arXiv:1803.03735*, 2018.

[40] H. Yuan and S. Ji, "Structpool: Structured graph pooling via conditional random fields," in *International Conference on Learning Representations*, 2020.

[41] Z. Wang and S. Ji, "Second-order pooling for graph neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[42] H. Gao, Y. Liu, and S. Ji, "Topology-aware graph pooling networks," *arXiv preprint arXiv:2010.09834*, 2020.

[43] H. Yuan, J. Tang, X. Hu, and S. Ji, "XGNN: Towards model-level explanations of graph neural networks," ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 430–438. [Online]. Available: https://doi.org/10.1145/3394486.3403085

[44] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *Advances in neural information processing systems*, 2019, pp. 9244–9255.

[45] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," in *Advances in neural information processing systems*, 2020.

[46] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," *arXiv preprint arXiv:2102.05152*, 2021.

[47] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[48] J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan, "Learning to explain: An information-theoretic perspective on model interpretation," in *International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 882–891.

[49] U. Alon, "Network motifs: theory and experimental approaches," *Nature Reviews Genetics*, vol. 8, no. 6, pp. 450–461, 2007.

[50] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[51] U. Alon, *An introduction to systems biology: design principles of biological circuits*. CRC press, 2019.

[52] F. Baldassarre and H. Azizpour, "Explainability techniques for graph convolutional networks," in *International Conference on Machine Learning (ICML) Workshops, 2019 Workshop on Learning and Reasoning with Graph-Structured Representations*, 2019.

[53] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10772–10781.

[54] Anonymous, "Hard masking for explaining graph neural networks," in *Submitted to International Conference on Learning Representations*, 2021, under review. [Online]. Available: https://openreview.net/forum?id=uDN8pRAdsoC

[55] M. S. Schlichtkrull, N. De Cao, and I. Titov, "Interpreting graph neural networks for nlp with differentiable edge masking," *arXiv preprint arXiv:2010.00577*, 2020.

[56] Anonymous, "Causal screening to interpret graph neural networks," in *Submitted to International Conference on Learning Representations*, 2021, under review. [Online]. Available: https://openreview.net/forum?id=nzKv5vxZfge

[57] R. Schwarzenberg, M. Hübner, D. Harbecke, C. Alt, and L. Hennig, "Layerwise relevance visualization in convolutional text graph classifiers," *arXiv preprint arXiv:1909.10911*, 2019.

[58] T. Schnake, O. Eberle, J. Lederer, S. Nakajima, K. T. Schütt, K.-R. Müller, and G. Montavon, "Higher-order explanations of graph neural networks via relevant walks," 2020.

[59] Q. Huang, M. Yamada, Y. Tian, D. Singh, D. Yin, and Y. Chang, "Graphlime: Local interpretable model explanations for graph neural networks," *arXiv preprint arXiv:2001.06216*, 2020.

[60] Y. Zhang, D. Defazio, and A. Ramesh, "Relex: A model-agnostic relational model explainer," *arXiv preprint arXiv:2006.00305*, 2020.

[61] M. N. Vu and M. T. Thai, "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks," in *Advances in neural information processing systems*, 2020.

[62] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International Conference on Machine Learning*, 2017, pp. 3145–3153.

[63] J. Chen, L. Song, M. Wainwright, and M. Jordan, "Learning to explain: An information-theoretic perspective on model interpretation," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 883–892. [Online]. Available: http://proceedings.mlr.press/v80/chen18j.html

[64] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, pp. 1057–1063, 1999.

[65] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations*, 2016.

[66] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," *arXiv preprint arXiv:1712.01312*, 2017.

[67] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[68] H. W. Kuhn and A. W. Tucker, *Contributions to the Theory of Games*. Princeton University Press, 1953, vol. 2.

[69] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[70] M. Yamada, W. Jitkrittum, L. Sigal, E. P. Xing, and M. Sugiyama, "High-dimensional feature selection by feature-wise kernelized lasso," *Neural computation*, vol. 26, no. 1, pp. 185–207, 2014.

[71] D. Margaritis and S. Thrun, "Bayesian network induction via local neighborhoods," *Advances in neural information processing systems*, vol. 12, pp. 505–511, 1999.

[72] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, p. e0130140, 2015.

[73] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

[74] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[75] L. Dong, F. Wei, C. Tan, D. Tang, M. Zhou, and K. Xu, "Adaptive recursive neural network for target-dependent twitter sentiment classification," in *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: Short papers)*, 2014, pp. 49–54.

[76] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer, "Allennlp: A deep semantic natural language processing platform," *arXiv preprint arXiv:1803.07640*, 2018.

[77] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[78] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of medicinal chemistry*, vol. 34, no. 2, pp. 786–797, 1991.

[79] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "Moleculenet: a benchmark for molecular machine learning," *Chemical science*, vol. 9, no. 2, pp. 513–530, 2018.

[80] A. Jacovi and Y. Goldberg, "Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness?" *arXiv preprint arXiv:2004.03685*, 2020.

[81] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," *arXiv preprint arXiv:1908.04626*, 2019.

[82] S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim, "A benchmark for interpretability methods in deep neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 9737–9748.

[83] B. Sanchez-Lengeling, J. Wei, B. Lee, E. Reif, P. Wang, W. W. Qian, K. McCloskey, L. Colwell, and A. Wiltschko, "Evaluating attribution for graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[84] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[85] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.