GNNExplainer: Generating Explanations for Graph Neural Networks

Rex Ying
rexying@stanford.edu
Stanford University

Dylan Bourgeois
dtsbourg@gmail.com
EPFL

Jiaxuan You
jiaxuan@stanford.edu
Stanford University

Marinka Zitnik
marinka@cs.stanford.edu
Stanford University

Jure Leskovec
jure@cs.stanford.edu
Stanford University

Abstract

Graph Neural Networks (GNNs) are a powerful tool for machine learning on graphs. GNNs combine node feature information with the graph structure by recursively passing neural messages along edges of the input graph. However, incorporating both graph structure and feature information leads to complex models and explaining predictions made by GNNs remains unsolved. Here we propose GNNEXPLAINER, the first general, model-agnostic approach for providing interpretable explanations for predictions of any GNN-based model on any graph-based machine learning task. Given an instance, GNNEXPLAINER identifies a compact subgraph structure and a small subset of node features that have a crucial role in GNN's prediction. Further, GNNEXPLAINER can generate consistent and concise explanations for an entire class of instances. We formulate GNNEXPLAINER as an optimization task that maximizes the mutual information between a GNN's prediction and distribution of possible subgraph structures. Experiments on synthetic and real-world graphs show that our approach can identify important graph structures as well as node features, and outperforms baselines by 17.1% on average. GNNEXPLAINER provides a variety of benefits, from the ability to visualize semantically relevant structures to interpretability, to giving insights into errors of faulty GNNs.

1 Introduction

In many real-world applications, including social, information, chemical, and biological domains, data can be naturally modeled as graphs [9, 48, 40]. Graphs are powerful data representations but are challenging to work with because they require modeling structural information as well as node feature information [45, 44]. To address this challenge, Graph Neural Networks (GNNs) have emerged as state of the art for machine learning on graphs, due to their ability to recursively incorporate information from neighboring nodes in the graph, naturally capturing both graph structure and node features [21, 16, 43, 39].

Despite their strengths, GNNs lack transparency as they do not easily allow for a human-intelligible explanation of their predictions. Yet, the ability to understand GNN's predictions is important and useful for several reasons: (i) it can increase trust in the GNN model, (ii) it improves model's transparency in a growing number of decision-critical applications pertaining to fairness, privacy and other safety challenges [11], and (iii) it allows practitioners to get an understanding of the network characteristics, identify and correct systematic patterns of mistakes made by models before deploying them in the real world.

While currently there are no methods for explaining GNNs, recent approaches for explaining other types of neural networks have taken one of two main routes. One line of work locally approximates models with a simpler surrogate model, which itself can be probed for explanations [28, 29, 24]. The

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

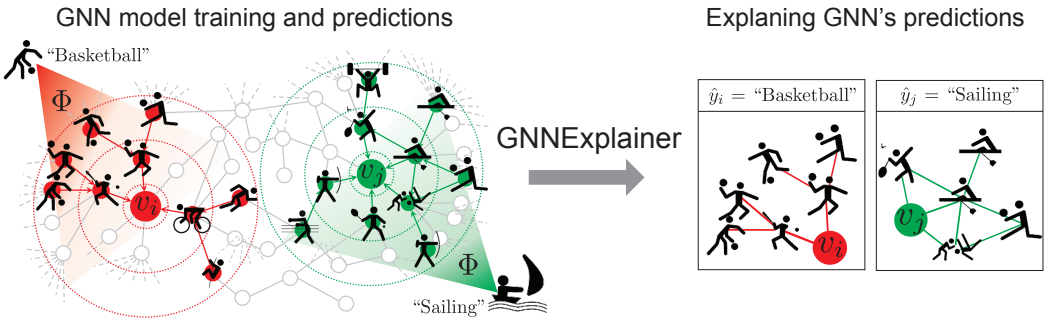


Figure 1: GNNEXPLAINER provides interpretable explanations for predictions made by any GNN model on any graph-based machine learning task. Shown is a hypothetical node classification task where a GNN model Φ is trained on a social interaction graph to predict future sport activities. Given a trained GNN Φ and a prediction $\hat{y}_i$ = "Basketball" for person v_i, GNNEXPLAINER generates an explanation by identifying a small subgraph of the input graph together with a small subset of node features (shown on the right) that are most influential for $\hat{y}_i$. Examining explanation for $\hat{y}_i$, we see that many friends in one part of v_i's social circle enjoy ball games, and so the GNN predicts that v_i will like basketball. Similarly, examining explanation for $\hat{y}_j$, we see that v_j's friends and friends of his friends enjoy water and beach sports, and so the GNN predicts $\hat{y}_j$ = "Sailing."

other line of methods carefully examine models for relevant components and identify, for example, relevant features in the input data [13, 6, 31, 26] and influential input instances [22, 37]. However, these approaches fall short in their ability to incorporate structural information, the essence of graphs. Since this aspect is crucial for the success of machine learning on graphs, any explanation of GNN's predictions should leverage rich relational information provided by the graph as well as node features.

Here, we propose GNNEXPLAINER, an approach to explain predictions from GNNs. GNNEXPLAINER takes a trained GNN model and its prediction(s), and it returns an explanation in the form of a small subgraph of the input graph together with a small subset of node features that are most influential for the prediction(s) (Figure 1). The approach is model-agnostic and can explain predictions of any GNN-based model on any machine learning task for graphs, including node classification, link prediction, and graph classification. It handles single- as well as multi-instance explanations. In the case of single-instance explanations, GNNEXPLAINER explains the GNN's prediction for one particular instance (*i.e.*, a node label, a new link, a graph-level label), and in the case of multi-instance explanations, GNNEXPLAINER provides a consistent explanation for a set of GNN's predictions.

GNNEXPLAINER formalizes an explanation as a rich subgraph of the entire graph the GNN was trained that maximizes the mutual information with the GNN's prediction(s). This is achieved by formulating a mean field variational approximation and learning a real-valued *graph mask* which selects the important subgraph of the GNN's computation graph. Simultaneously, GNNEXPLAINER also learns a *feature mask* that masks out unimportant node features (Figure 1).

We validate GNNEXPLAINER on synthetic and real-world graphs. On synthetic graphs with planted network motifs important for prediction, we show that GNNEXPLAINER can outperform baselines by up to 43.0% in explanation accuracy. Furthermore, we show how GNNEXPLAINER can robustly identify important graph structures and node features that influence a GNN's prediction the most. On two real-world datasets, *i.e.*, molecular graphs and social interaction networks, we show that GNNEXPLAINER can identify graph structures that a GNN learned to use for prediction, such as NO_2 chemical groups or ring structures in molecules, and star structures in Reddit threads.

2 Related work

Although the problem of explaining GNNs is not well-studied, the related problems of interpretability and neural debugging recently received substantial attention in machine learning. At a high level, we can group these interpretability methods for non-graph neural networks into two main families.

Methods in the first family formulate a simple proxy model for a full neural network model. This can be done in a model-agnostic way, usually by learning a locally faithful approximation around the prediction, for example with a linear model [28] or a set of rules, representing sufficient conditions on the prediction [3, 46, 24]. Methods in the second family identify important aspects of the computation, for example, through feature gradients [13, 42], backpropagation of neurons' contributions to the input features [31, 30, 6], and counterfactual reasoning [19]. However, the saliency maps [42] produced by these methods have been shown to be misleading in some instances [2] and prone to issues like gradient saturation [31, 30]. These issues are exacerbated on discrete inputs such as graph adjacency matrices since the gradient values can be very large but only on a very small interval.

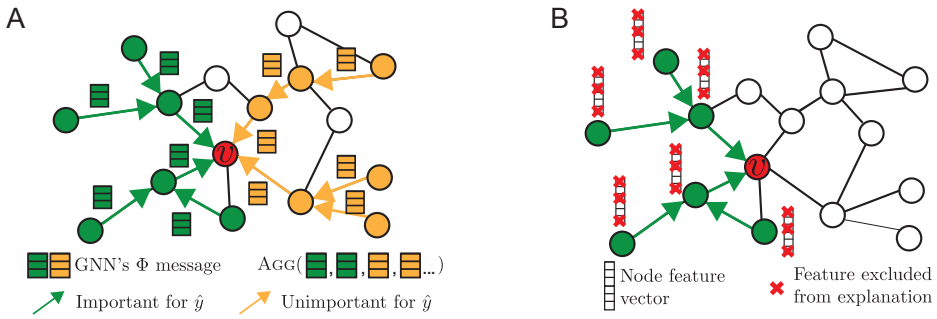


Figure 2: **A.** GNN computation graph G_c for making a prediction $\hat{y}$ at node v. Some edges form important neural message-passing pathways (green), which allow useful node information to be propagated across G_c and aggregated at v_i for prediction, while other edges do not (orange). However, GNN needs to aggregate important as well as unimportant messages to form a prediction at node v, which can dilute the signal accumulated from v_i's neighborhood. The goal of GNNEXPLAINER is to identify a small set of important features and pathways (green) that are crucial for prediction. **B.** In addition to G_S (in green), GNNEXPLAINER identifies what feature dimensions of G_S's nodes are important for prediction by learning a node feature mask.

Because of that, such approaches are not suitable for explaining predictions made by neural networks on graphs.

Instead of creating new, inherently interpretable models, post-hoc interpretability methods [22, 37, 15, 1, 14, 17] consider the model as a black box and then probe it for relevant information. However, no work has been done to leverage relational structures like graphs. The lack of methods for explaining predictions on graph-structured data is problematic, as in many cases, predictions on graphs are induced by a complex composition of nodes and their paths. For example, in some tasks, an edge could be important only when another alternative path exists to form a cycle, which determines the class of the node [10, 12], and thus their joint contribution cannot be modeled well using linear combinations of individual contributions.

Finally, recent GNN models augment the interpretability of GNNs via attention mechanisms [33, 27, 32]. However, although the edge attention values can serve as an indication of graph structure importance, the values are the same for predictions on all nodes. Thus, this contradicts with many applications where an edge is essential for predicting a label of one node but not the label of another node. Furthermore, these approaches are limited to specific GNN architectures and cannot explain predictions using graph structure and node features jointly.

3 Formulating explanations for graph neural networks

Let G denote a graph on edges E and nodes V that are associated with d-dimensional node features $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $x_i \in \mathbb{R}^d$. Without loss of generality, we consider the problem of explaining a node classification task (see Section 4.4 for other tasks). Let f denote a label function on nodes $f : V \mapsto \{1, \ldots, C\}$ that maps every node in V to one of C classes. The GNN model Φ is optimized on all nodes in the training set and can then be used to approximate f on nodes in the test set.

3.1 Background

For each layer, the update of the GNN model Φ involves three key computations [4, 44, 45]. (1) First, the model computes neural messages between every pair of nodes. The message for node pair (v_i, v_j) is a function MSG of v_i's and v_j's representations $\mathbf{h}_i^{l-1}$ and $\mathbf{h}_j^{l-1}$ in the previous layer, and the relation r_{ij} between the nodes: $m_{ij}^l = \text{MSG}(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}, r_{ij})$. (2) Second, for each node v_i, GNN aggregates messages from v_i's neighborhood $\mathcal{N}_{v_i}$ and calculates an aggregated message M_i via an aggregation method AGG [16, 34]: $M_i^l = \text{AGG}(\{m_{ij}^l | v_j \in \mathcal{N}_{v_i}\})$, where $\mathcal{N}_{v_i}$ is neighborhood of node v_i whose definition depends on a particular GNN variant. (3) Finally, GNN takes the aggregated message M_i^l along with v_i's representation $\mathbf{h}_i^{l-1}$ from the previous layer, and it non-linearly transforms them to obtain v_i's representation $\mathbf{h}_i^l$ at layer l: $\mathbf{h}_i^l = \text{UPDATE}(M_i^l, \mathbf{h}_i^{l-1})$. The final embedding for node v_i after L layers of computation is $\mathbf{z}_i = \mathbf{h}_i^L$. Our GNNEXPLAINER provides explanations for any GNN that can be formulated in terms of MSG, AGG, and UPDATE computations.

3.2 GNNEXPLAINER: Problem formulation

Our key insight is the observation that information used by GNN model Φ to compute prediction $\hat{y}$ is completely described by the v_i's computation graph, which is defined by GNN's neighborhood-based aggregation (Fig. 2). Let us denote the computation graph to generate the embedding of node v_i by $G_c(v_i)$, the associated binary adjacency matrix by $A_c(v_i) \in \{0,1\}^{n \times n}$ and the associated feature set by $X_c(v_i) = \{x_j | v_j \in G_c(v_i)\}$. GNN learns a conditional distribution $P_\Phi(Y|G_c, X_c)$, where Y is a random variable representing output class labels $\{1, \ldots, C\}$, indicating the probability of a node belonging to each of C classes.

In effect, this implies that a GNN's prediction is given by $\hat{y} = \Phi(G_c(v_i), X_c(v_i))$, meaning that we only need to consider structural information in $G_c(v_i)$ and node feature information in $X_c(v_i)$ to explain $\hat{y}$ (Figure 2A). Formally, the GNNEXPLAINER provides an explanation for $\hat{y}$ as (G_S, X_S), where G_S is a small subgraph of the computation graph and $X_S = \{x_j | v_j \in G_S\}$ is a small subset of node features that are most important for $\hat{y}$ (Figure 2B).

4 GNNEXPLAINER

Next we describe our approach GNNEXPLAINER. Given a trained GNN model Φ and a prediction (*i.e.*, single-instance explanation, Sections 4.1 and 4.2) or a set of predictions (*i.e.*, multi-instance explanations, Section 4.3), the GNNEXPLAINER will generate an explanation by identifying a subgraph of the computation graph and a subset of node features that are most influential for the model Φ's prediction. In the case of explaining a set of predictions, GNNEXPLAINER will aggregate individual explanations in the set and summarize it with a prototype. We conclude with a discussion on how GNNEXPLAINER can be used for any machine learning task on graphs, including link prediction and graph classification (Section 4.4).

4.1 Single-instance explanations

Our goal is to identify a subgraph $G_S \subseteq G_c(v_i)$ and the associated features $X_S = \{\mathbf{x}_j | v_j \in G_S\}$ that are important for GNN's prediction $\hat{y}$. We formalize the notion of importance using mutual information MI and formulate it in an optimization framework as follows:

$$\max_{G_S} MI\left(Y, (G_S, X_S)\right) = H(Y) - H(Y|G = G_S, X = X_S). \tag{1}$$

For the target node v_i, MI quantifies the change in the probability of prediction $\hat{y} = \Phi(G_c(v_i), X_c(v_i))$ if v_i's computation graph and the associated features are limited to the explanation subgraph $G_S(v_i)$ and $X_S(v_i)$, respectively. For example, consider the situation where $v_j \in G_c(v_i)$, $v_j \neq v_i$. Then, if removing v_j from $G_c(v_i)$ strongly decreases the probability of prediction $\hat{y}$, the node v_j is a good counterfactual explanation of the prediction. Similarly, consider the situation where $(v_j, v_k) \in G_c(v_i)$, $v_j, v_k \neq v_i$. Then, if removing an edge between v_j and v_k strongly decreases the probability of the prediction then it is the absence of a link between v_j and v_k that is a good counterfactual explanation of prediction. Examining Eq. (1), we see that the entropy term $H(Y)$ is constant because Φ is fixed, trained GNN. As a result, maximizing mutual information between predicted label distribution Y and explanation (G_S, X_S) is equivalent to minimizing conditional entropy $H(Y|G = G_S, X = X_S)$, which can be expressed as follows:

$$H(Y|G{=}G_S, X{=}X_S) = -\mathbb{E}_{Y|G_S,X_S}\left[\log P_\Phi(Y|G{=}G_S, X{=}X_S)\right]. \tag{2}$$

Explanation for prediction $\hat{y}$ is thus a subgraph G_S that minimizes uncertainty of Φ when the GNN computation is limited to G_S. In effect, G_S maximizes probability of $\hat{y}$ (Figure 2). To obtain a compact explanation, we impose a constraint on G_S's size as: $|G_S| \leq K_M$, so that G_S has at most K_M nodes. In effect, this implies that GNNEXPLAINER aims to denoise G_c by taking K_M edges that give the highest mutual information with the prediction.

GNNEXPLAINER's optimization framework. Direct optimization of GNNEXPLAINER's objective is not tractable as $G_c(v_i)$ has exponentially many subgraphs G_S that are candidate explanations for $\hat{y}$. We thus consider a fractional adjacency matrix[1] for subgraphs G_S, *i.e.*, $A_S \in [0,1]^{n \times n}$, and enforce the subgraph constraint as: $A_S[j,k] \leq A_c[j,k]$ for all j, k. This continuous relaxation can be interpreted as a variational approximation of distribution of subgraphs of $G_c(v_i)$. In particular, if we treat $G_S \sim \mathcal{G}$ as a random graph variable, the objective in Eq. (2) becomes:

$$\min_{\mathcal{G}} \mathbb{E}_{G_S \sim \mathcal{G}} H(Y|G = G_S, X = X_S), \tag{3}$$

[1]For typed edges, we define $G_S \in [0,1]^{C_e \times n \times n}$ where C_e is the number of edge types.

which, by Jensen's inequality, has the following upper bound:

$$\min_{\mathcal{G}} H(Y|G = \mathbb{E}_{\mathcal{G}}[G_S], X = X_S). \tag{4}$$

To estimate $\mathbb{E}_{\mathcal{G}}$ we use the mean-field variational approximation and decompose $\mathcal{G}$ into a multivariate Bernoulli distribution as: $P_{\mathcal{G}}(G_S) = \prod_{(j,k)\in G_c(v_i)} A_S[j,k]$. This allows us to estimate the expectation with respect to the mean-field approximation and obtain A_S in which (j,k)-th entry represents the expectation on whether edge (v_j, v_k) exists. We observed empirically that this approximation together with a regularizer for promoting discreteness [39] converges to good local minima despite the non-convexity of GNNs. Thus, a computationally efficient version of GNNEXPLAINER's objective, which we optimize using gradient descent, is as follows:

$$\min_{M} -\sum_{c=1}^{C} \mathbb{1}[y = c] \log P_{\Phi}(Y = y|G = A_S \odot \sigma(M), X = X_S), \tag{5}$$

where $M \in \mathbb{R}^{n\times n}$ denotes the mask that we need to learn, $\odot$ denotes element-wise multiplication, and σ denotes the sigmoid that maps the mask to $[0,1]^{n\times n}$. Finally, we remove low values in M through thresholding and compute the element-wise multiplication of $\sigma(M)$ and A_c to arrive at the explanation G_S for GNN's prediction $\hat{y}$ at node v_i.

4.2 Joint consideration of structural and node feature information

To identify what node features are most important for prediction $\hat{y}$, GNNEXPLAINER learns a feature selector for nodes in explanation G_S. Instead of defining X_S to consists of all node features, *i.e.*, $X_S = \{x_j | v_j \in G_S\}$, GNNEXPLAINER considers X_S^T as a subset of features of nodes in G_S, which are defined through a binary feature selector $T \in \{0,1\}^d$ (Figure 2B):

$$X_S^T = \{x_j^T | v_j \in G_S\}, \quad x_j^T = [x_{j,t_1}, \ldots, x_{j,t_k}] \text{ for } T_{t_k} = 1, \tag{6}$$

where x_j^T has node features that are not masked out by T. Explanation (G_S, X_S) is then jointly optimized for maximizing the mutual information objective:

$$\max_{G_S,T} MI\,(Y, (G_S, T)) = H(Y) - H(Y|G = G_S, X = X_S^T), \tag{7}$$

which represents a modified objective function from Eq. (1) that considers structural and node feature information to generate an explanation for prediction $\hat{y}$.

Learning binary feature selector T. We specify X_S^T as $X_S \odot T$, where T acts as a feature mask that we need to learn. Intuitively, if a particular feature is not important, the corresponding weights in GNN's weight matrix take values close to zero. In effect, this implies that masking the feature out does not decrease predicted probability for $\hat{y}$. Conversely, if the feature is important then masking it out would decrease predicted probability. However, the problem with this approach is that it ignores features that are important for prediction but take values close to zero. To address this issue we marginalize over all feature subsets and use a Monte Carlo estimate to sample from empirical marginal distribution for nodes in X_S during training [47]. Further, we use a reparametrization trick [20] to backpropagate gradients in Eq. (7) to the feature mask T. In particular, to backpropagate through a d-dimensional random variable X we reparametrize X as: $X = Z + (X_S - Z) \odot T$ s.t. $\sum_j T_j \leq K_T$, where Z is a d-dimensional random variable sampled from the empirical distribution and K_T is a parameter representing the maximum number of features to be kept in the explanation.

Integrating additional constraints into explanations. To impose further properties on the explanation we can extend GNNEXPLAINER's objective function in Eq. (7) with regularization terms. For example, we use element-wise entropy to encourage structural and node feature masks to be discrete. Further, GNNEXPLAINER can encode domain-specific constraints through techniques like Laplacian regularization, equivalent to the Lagrange multiplier of constraints. Finally, it is important to note that each explanation must be a valid computation graph. In particular, explanation (G_S, X_S) needs to allow GNN's neural messages to flow towards node v_i such that GNN can make prediction $\hat{y}$. Importantly, GNNEXPLAINER automatically provides explanations that represent valid computation graphs because it optimizes structural masks across entire computation graphs. Even if a disconnected edge is important for neural message-passing, it will not be selected for explanation as it cannot influence GNN's prediction. In effect, this implies that G_S tends to be a small connected subgraph.

4.3 Multi-instance explanations through graph prototypes

The output of a single-instance explanation (Sections 4.1 and 4.2) is a small subgraph of the input graph and a small subset of associated node features that are most influential for a single prediction. To answer questions like "How did a GNN predict that a given set of nodes all have label c?", we need to obtain a global explanation of class c. Our goal here is to provide insight into how the identified subgraph for a particular node relates to a graph structure that explains an entire class.

To this end, GNNEXPLAINER provides multi-instance explanations that are based on graph alignments and prototypes. For a given class c (or, any set of predictions that we want to explain), we first choose a reference node v_c, for example, by computing the mean embedding of all nodes assigned to c. We then take explanation $G_S(v_c)$ for reference v_c and align it to explanations of other nodes assigned to class c. Technically, we use relaxed graph matching to find correspondences between nodes in the reference subgraph $G_S(v_c)$ and subgraphs G_S of other nodes [39] (See Appendix for details). Finding optimal matching of large graphs is challenging in practice. However, the single-instance explainer generates small graphs (Section 4.2) and thus near-optimal pairwise graph matchings can be efficiently computed. This procedure gives us adjacency matrices A_S for all nodes predicted to belong to c, where the matrices are aligned with respect to the node ordering in the reference v_c's matrix $A_S(v_c)$. Finally, we can aggregate aligned adjacency matrices into a graph prototype A_{proto} using, for example, a robust median-based approach. Prototype A_{proto} gives insights into graph patterns shared between nodes that belong to the same class. One can then study prediction for a particular node by comparing explanation for that node's prediction (*i.e.*, returned by single-instance explanation approach) to the prototype (see Appendix for more information and experiments).

4.4 GNNEXPLAINER model extensions

Any machine learning task on graphs. In addition to explaining node classification, GNNEXPLAINER provides explanations for link prediction and graph classification with no change to its optimization algorithm. When predicting a link (v_j, v_k), GNNEXPLAINER learns two masks $X_S(v_j)$ and $X_S(v_k)$ for both endpoints of the link. When classifying a graph, the adjacency matrix in Eq. (5) is the union of adjacency matrices for all nodes in the graph whose label we want to explain.

Any GNN model. Modern GNNs are based on message passing architectures on the input graph. The message passing computation graphs can be composed in many different ways and GNNEXPLAINER can account for all of them. Thus, GNNEXPLAINER can be applied to: Graph Convolutional Networks [21], Gated Graph Sequence Neural Networks [25], Jumping Knowledge Networks [35], Attention Networks [32], Graph Networks [4], GNNs with various node aggregation schemes [7, 5, 18, 16, 39, 38, 34], Line-Graph NNs [8], position-aware GNN [41], and many other GNN architectures.

Computational complexity. The number of parameters in GNNEXPLAINER's optimization depends on the size of computation graph $G_c(v_i)$ for node v_i whose prediction we aim to explain. In particular, $G_c(v_i)$'s adjacency matrix $A_c(v_i)$ is equal to the size of the mask M, which needs to be learned by GNNEXPLAINER. However, since computation graphs are typically relatively small, compared to the size of exhaustive L-hop neighborhoods (*e.g.*, 2-3 hop neighborhoods [21], sampling-based neighborhoods [38], neighborhoods with attention [32]), GNNEXPLAINER can effectively generate explanations even when input graphs are large.

5 Experiments

We begin by describing the graphs, baselines, and experimental setup. We then present experiments on explaining GNNs for node classification and graph classification tasks. Our qualitative and quantitative analysis demonstrates that GNNEXPLAINER is accurate and effective in identifying explanations, both in terms of graph structure and node features.

Synthetic datasets. We construct four kinds of node classification datasets (Table 1). (1) In BA-SHAPES, we start with a base Barabási-Albert (BA) graph on 300 nodes and a set of 80 five-node house-structured network motifs, which are attached to randomly selected nodes of the base graph. The resulting graph is further perturbed by adding $0.1N$ random edges. Nodes are assigned to 4 classes based on their structural roles. (2) BA-COMMUNITY dataset is a union of two BA-SHAPES graphs. Nodes have normally distributed feature vectors and are assigned to one of 8 classes based on their structural roles and community memberships. (3) In TREE-CYCLES, we start with a base 8-level balanced binary tree and 80 six-node cycle motifs, which are attached to random nodes of the base graph. (4) TREE-GRID is the same as TREE-CYCLES except that 3-by-3 grid motifs are attached to the base tree graph in place of cycle motifs.

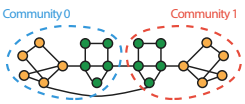
	BA-Shapes	BA-Community	Tree-Cycles	Tree-Grid
Base				
Motif				
Node Features	None	$\mathcal{N}(\mu_l, \sigma_l)$ where l = community ID	None	None
Explanation content	Graph structure	Graph structure Node feature information	Graph structure	Graph structure
Explanation accuracy				
Att	0.815	0.739	0.824	0.612
Grad	0.882	0.750	0.905	0.667
GNNExplainer	**0.925**	**0.836**	**0.948**	**0.875**

Table 1: Illustration of synthetic datasets (see "Synthetic datasets" for details) together with performance evaluation of GNNEXPLAINER and baseline explainability methods.

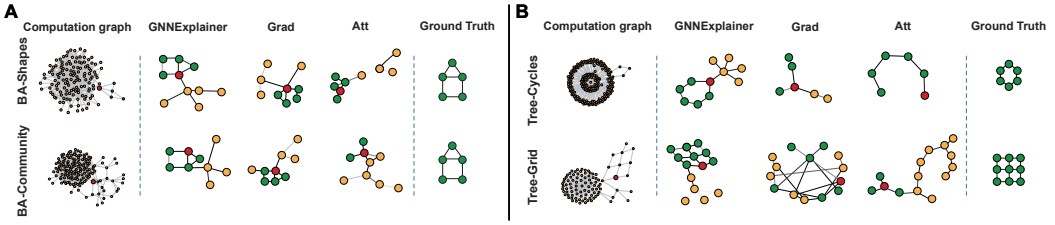


Figure 3: Evaluation of single-instance explanations. **A-B.** Shown are exemplar explanation subgraphs for node classification task on four synthetic datasets. Each method provides explanation for the red node's prediction.

Real-world datasets. We consider two graph classification datasets: (1) MUTAG is a dataset of 4,337 molecule graphs labeled according to their mutagenic effect on the Gram-negative bacterium *S. typhimurium* [10]. (2) REDDIT-BINARY is a dataset of 2,000 graphs, each representing an online discussion thread on Reddit. In each graph, nodes are users participating in a thread, and edges indicate that one user replied to another user's comment. Graphs are labeled according to the type of user interactions in the thread: *r/IAmA* and *r/AskReddit* contain Question-Answer interactions, while *r/TrollXChromosomes* and *r/atheism* contain Online-Discussion interactions [36].

Baselines. Many explainability methods cannot be directly applied to graphs (see Section 2). Nevertheless, we here consider the following baselines that can provide insights into predictions made by GNNs: (1) GRAD is a gradient-based method. We compute gradient of the GNN's loss function with respect to the adjacency matrix and the associated node features, similar to a saliency map approach. (2) ATT is a graph attention GNN (GAT) [32] that learns attention weights for edges in the computation graph, which we use as a proxy measure of edge importance. While ATT does consider graph structure, it does not use node features and can only explain GAT models, not all GNNs. Furthermore, in ATT it is not obvious which attention weights need to be used for edge importance, since a 1-hop neighbor of a node can also be a 2-hop neighbor of the same node due to cycles. Each edge's importance is thus computed as the average attention weight across all layers.

Setup and implementation details. For each dataset, we first train a single GNN for each dataset, and use GRAD and GNNEXPLAINER to explain the predictions made by the GNN. Note that the ATT baseline requires using a graph attention architecture like GAT [32]. We thus train a separate GAT model on the same dataset and use the learned edge attention weights for explanation. Hyperparameters K_M, K_T control the size of subgraph and feature explanations respectively, which is informed by prior knowledge about the dataset. For synthetic datasets, we set K_M to be the size of ground truth. On real-world datasets, we set $K_M = 10$. We set $K_T = 5$ for all datasets. We further fix our weight regularization hyperparameters across all node and graph classification experiments. We refer readers to the Appendix for more training details. Datasets, source code for GNNEXPLAINER, and baselines will be made public at publication time.

Results. We investigate questions: Does GNNEXPLAINER provide sensible explanations? How do explanations compare to the ground-truth knowledge? How does GNNEXPLAINER perform on various graph-based prediction tasks? Can it explain predictions made by different GNNs?

1) Quantitative analyses. Results on node classification datasets are shown in Table 1. We have ground-truth explanations for synthetic datasets and we use them to calculate explanation accuracy for all explanation methods. Specifically, we formalize the explanation problem as a binary classification task, where edges in the ground-truth explanation are treated as labels and importance weights given by explainability method are viewed as prediction scores. A better explainability method predicts

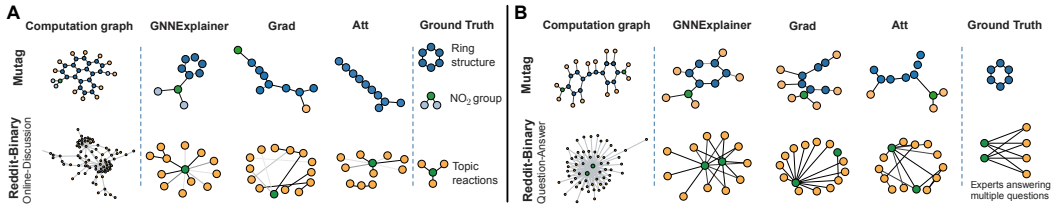


Figure 4: Evaluation of single-instance explanations. **A-B.** Shown are exemplar explanation subgraphs for graph classification task on two datasets, MUTAG and REDDIT-BINARY.

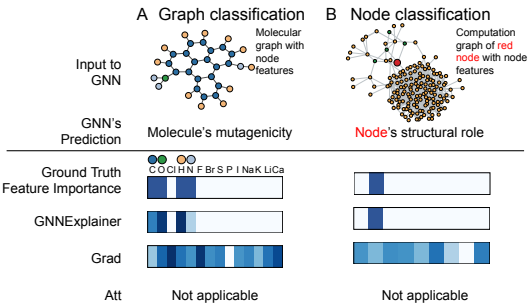


Figure 5: Visualization of features that are important for a GNN's prediction. **A.** Shown is a representative molecular graph from MUTAG dataset (top). Importance of the associated graph features is visualized with a heatmap (bottom). In contrast with baselines, GNNEXPLAINER correctly identifies features that are important for predicting the molecule's mutagenicity, *i.e.* C, O, H, and N atoms. **B.** Shown is a computation graph of a red node from BA-COMMUNITY dataset (top). Again, GNNEXPLAINER successfully identifies the node feature that is important for predicting the structural role of the node but baseline methods fail.

high scores for edges that are in the ground-truth explanation, and thus achieves higher explanation accuracy. Results show that GNNEXPLAINER outperforms baselines by 17.1% on average. Further, GNNEXPLAINER achieves up to 43.0% higher accuracy on the hardest TREE-GRID dataset.

2) Qualitative analyses. Results are shown in Figures 3–5. In a topology-based prediction task with no node features, *e.g.* BA-SHAPES and TREE-CYCLES, GNNEXPLAINER correctly identifies network motifs that explain node labels, *i.e.* structural labels (Figure 3). As illustrated in the figures, house, cycle and tree motifs are identified by GNNEXPLAINER but not by baseline methods. In Figure 4, we investigate explanations for graph classification task. In MUTAG example, colors indicate node features, which represent atoms (hydrogen H, carbon C, *etc*). GNNEXPLAINER correctly identifies carbon ring as well as chemical groups NH_2 and NO_2, which are known to be mutagenic [10].

Further, in REDDIT-BINARY example, we see that Question-Answer graphs (2nd row in Figure 4B) have 2-3 high degree nodes that simultaneously connect to many low degree nodes, which makes sense because in QA threads on Reddit we typically have 2-3 experts who all answer many different questions [23]. Conversely, we observe that discussion patterns commonly exhibit tree-like patterns (2nd row in Figure 4A), since a thread on Reddit is usually a reaction to a single topic [23]. On the other hand, GRAD and ATT methods give incorrect or incomplete explanations. For example, both baseline methods miss cycle motifs in MUTAG dataset and more complex grid motifs in TREE-GRID dataset. Furthermore, although edge attention weights in ATT can be interpreted as importance scores for message passing, the weights are shared across all nodes in input the graph, and as such ATT fails to provide high quality single-instance explanations.

An essential criterion for explanations is that they must be interpretable, *i.e.*, provide a qualitative understanding of the relationship between the input nodes and the prediction. Such a requirement implies that explanations should be easy to understand while remaining exhaustive. This means that a GNN explainer should take into account both the structure of the underlying graph as well as the associated features when they are available. Figure 5 shows results of an experiment in which GNNEXPLAINER jointly considers structural information as well as information from a small number of feature dimensions[2]. While GNNEXPLAINER indeed highlights a compact feature representation in Figure 5, gradient-based approaches struggle to cope with the added noise, giving high importance scores to irrelevant feature dimensions.

Further experiments on multi-instance explanations using graph prototypes are in Appendix.

6 Conclusion

In this paper, we present GNNEXPLAINER, a novel method for explaining predictions of any GNN on any graph-based machine learning task without requiring modification of the underlying GNN architecture or re-training. We show how GNNEXPLAINER can leverage recursive neighborhood-aggregation scheme of graph neural networks to identify important graph pathways as well as

[2]Feature explanations are shown for the two datasets with node features, *i.e.*, MUTAG and BA-COMMUNITY.

highlight relevant node feature information that is passed along edges of the pathways. While the problem of explainability of machine-learning predictions has received substantial attention in recent literature, our work is unique in the sense that it presents an approach that operates on relational structures—graphs with rich node features—and provides a straightforward interface for making sense out of GNN predictions, debugging GNN models and identifying systematic patterns of mistakes.

References

[1] A. Adadi and M. Berrada. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

[2] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. In *NeurIPS*, 2018.

[3] M. Gethsiyal Augasta and T. Kathirvalavakumar. Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems. *Neural Processing Letters*, 35(2):131–150, April 2012.

[4] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

[5] J. Chen, J. Zhu, and L. Song. Stochastic training of graph convolutional networks with variance reduction. In *ICML*, 2018.

[6] Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. Learning to explain: An information-theoretic perspective on model interpretation. *arXiv preprint arXiv:1802.07814*, 2018.

[7] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. In *ICLR*, 2018.

[8] Z. Chen, L. Li, and J. Bruna. Supervised community detection with line graph neural networks. In *ICLR*, 2019.

[9] E. Cho, S. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *KDD*, 2011.

[10] A. Debnath et al. 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.

[11] F. Doshi-Velez and B. Kim. Towards A Rigorous Science of Interpretable Machine Learning. 2017. arXiv: 1702.08608.

[12] D. Duvenaud et al. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, 2015.

[13] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[14] A. Fisher, C. Rudin, and F. Dominici. All Models are Wrong but many are Useful: Variable Importance for Black-Box, Proprietary, or Misspecified Prediction Models, using Model Class Reliance. January 2018. arXiv: 1801.01489.

[15] R. Guidotti et al. A Survey of Methods for Explaining Black Box Models. *ACM Comput. Surv.*, 51(5):93:1–93:42, 2018.

[16] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[17] G. Hooker. Discovering additive structure in black box functions. In *KDD*, 2004.

[18] W.B. Huang, T. Zhang, Y. Rong, and J. Huang. Adaptive sampling towards fast graph representation learning. In *NeurIPS*, 2018.

[19] Bo Kang, Jefrey Lijffijt, and Tijl De Bie. Explaine: An approach for explaining network embedding-based link predictions. *arXiv:1904.12694*, 2019.

[20] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *NeurIPS*, 2013.

[21] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2016.

[22] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.

[23] Srijan Kumar, William L Hamilton, Jure Leskovec, and Dan Jurafsky. Community interaction and conflict on the web. In *WWW*, pages 933–943, 2018.

[24] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec. Interpretable & Explorable Approximations of Black Box Models, 2017.

[25] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv:1511.05493*, 2015.

[26] S. Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In *NIPS*, 2017.

[27] D. Neil et al. Interpretable Graph Convolutional Neural Networks for Inference on Noisy Knowledge Graphs. In *ML4H Workshop at NeurIPS*, 2018.

[28] M. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *KDD*, 2016.

[29] G. J. Schmitz, C. Aldrich, and F. S. Gouws. ANN-DT: an algorithm for extraction of decision trees from artificial neural networks. *IEEE Transactions on Neural Networks*, 1999.

[30] A. Shrikumar, P. Greenside, and A. Kundaje. Learning Important Features Through Propagating Activation Differences. In *ICML*, 2017.

[31] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic Attribution for Deep Networks. In *ICML*, 2017.

[32] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.

[33] T. Xie and J. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. In *Phys. Rev. Lett.*, 2018.

[34] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *ICRL*, 2019.

[35] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, 2018.

[36] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *KDD*, pages 1365–1374. ACM, 2015.

[37] C. Yeh, J. Kim, I. Yen, and P. Ravikumar. Representer point selection for explaining deep neural networks. In *NeurIPS*, 2018.

[38] R. Ying, R. He, K. Chen, P. Eksombatchai, W. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, 2018.

[39] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, 2018.

[40] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. 2018.

[41] J. You, Rex Ying, and J. Leskovec. Position-aware graph neural networks. In *ICML*, 2019.

[42] M. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *ECCV*. 2014.

[43] M. Zhang and Y. Chen. Link prediction based on graph neural networks. In *NIPS*, 2018.

[44] Z. Zhang, Peng C., and W. Zhu. Deep Learning on Graphs: A Survey. *arXiv:1812.04202*, 2018.

[45] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun. Graph Neural Networks: A Review of Methods and Applications. *arXiv:1812.08434*, 2018.

[46] J. Zilke, E. Loza Mencia, and F. Janssen. DeepRED - Rule Extraction from Deep Neural Networks. In *Discovery Science*. Springer International Publishing, 2016.

[47] L. Zintgraf, T. Cohen, T. Adel, and M. Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *ICLR*, 2017.

[48] M. Zitnik, M. Agrawal, and J. Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34, 2018.