Task-Agnostic Graph Explanations

Yaochen Xie*
Texas A&M University
College Station, TX
ethanycx@tamu.edu

Sumeet Katariya
Amazon Search
Palo Alto, CA
katsumee@amazon.com

Xianfeng Tang
Amazon Search
Palo Alto, CA
xianft@amazon.com

Edward Huang
Amazon Search
Palo Alto, CA
ewhuang@amazon.com

Nikhil Rao
Amazon Search
Palo Alto, CA
nikhilsr@amazon.com

Karthik Subbian
Amazon Search
Palo Alto, CA
ksubbian@amazon.com

Shuiwang Ji
Texas A&M University
College Station, TX
sji@tamu.edu

Abstract

Graph Neural Networks (GNNs) have emerged as powerful tools to encode graph-structured data. Due to their broad applications, there is an increasing need to develop tools to explain how GNNs make decisions given graph-structured data. Existing learning-based GNN explanation approaches are task-specific in training and hence suffer from crucial drawbacks. Specifically, they are incapable of producing explanations for a multitask prediction model with a single explainer. They are also unable to provide explanations in cases where the GNN is trained in a self-supervised manner, and the resulting representations are used in future downstream tasks. To address these limitations, we propose a Task-Agnostic GNN Explainer (TAGE) that is independent of downstream models and trained under self-supervision with no knowledge of downstream tasks. TAGE enables the explanation of GNN embedding models with unseen downstream tasks and allows efficient explanation of multitask models. Our extensive experiments show that TAGE can significantly speed up the explanation efficiency by using the same model to explain predictions for multiple downstream tasks while achieving explanation quality as good as or even better than current state-of-the-art GNN explanation approaches. Our code is publicly available as part of the DIG library[2].

1 Introduction

Graph neural networks (GNNs) [11, 25, 32] have achieved remarkable success in learning from real-world graph-structured data due to their unique ability to capture both feature-wise and topological information. Extending their success, GNNs are widely applied in various research fields and industrial applications including quantum chemistry [3, 26], drug discovery [16, 28, 29], large-scale social networks [30, 35], and recommender systems [1, 33]. While multiple approaches have been proposed and studied to improve GNN performance, GNN explainability is an emerging area and has a smaller body of research behind it. Recently, explainability has gained more attention due to

*This work was performed during an internship at Amazon.com Services LLC.
[2] https://github.com/divelab/DIG/tree/main/dig/xgraph/TAGE/

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

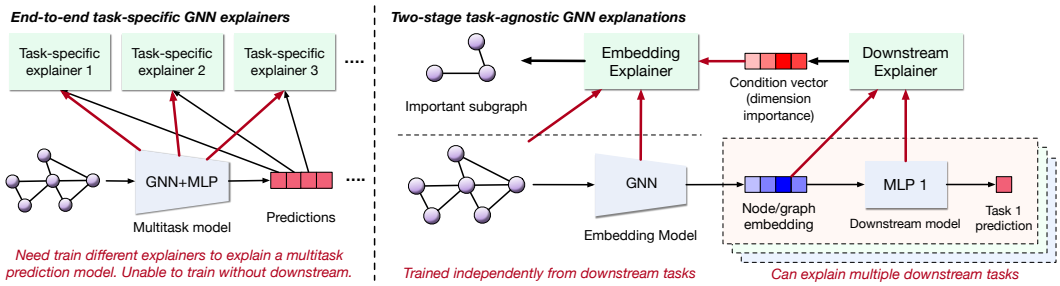

Figure 1: A comparison between typical end-to-end task-specific GNN explainers and the proposed task-agnostic explanation pipeline. To explain a multitask model, typical explanation pipelines need to optimize multiple explainers, whereas the two-stage explanation pipeline only learns one embedding explainer that can cooperate with multiple lightweight downstream explainers.

an increasing desire for GNNs with more security, fairness, and reliability. Being able to provide a good explanation to a GNN prediction increases model reliability and reduces the risk of incorrect predictions, which is crucial in fields such as molecular biology, chemistry, fraud detection, etc.

Existing methods adapting the explanation methods for convolutional neural networks (CNNs) or specifically designed for GNNs have shown promising explanations on multiple types of graph data. A recent survey [37] categorizes existing explanation methods into gradient-based, perturbation, decomposition, and surrogate methods. In particular, perturbation methods involve learning or optimization [12, 13, 15, 34, 36] and, while bearing higher computational costs, generally achieve state-of-the-art performance in terms of explanation quality. These methods train *post-hoc* explanation models on top of the prediction model to be explained. Earlier approaches like GNNExplainer [34] require training or optimizing an individual explainer for each data instance, i.e., a graph or a node to be explained. In contrast, PGExplainer [15] performs inductive learning, i.e., it only requires a one-time training, and the explainer can be generalized to explain all data instances without individual optimization. Compared to other optimization-based explanation methods, PGExplainer significantly improves efficiency in terms of time cost without performance loss by learning. Following a similar inductive learning paradigm, more recent work ReFine [27] and GSAT [17] aim to provide multi-grained explanations and jointly learned explanations with GNNs, respectively.

However, even state-of-the-art explanation methods like PGExplainer are still task-specific at training and hence suffer from two crucial drawbacks. First, current methods are inefficient in explaining multitask prediction for graph-structured data. For example, one may need to predict multiple chemical properties in drug discovery for a molecular graph. In particular, ToxCast from MoleculeNet has 167 prediction tasks. In these cases, it is common to apply a single GNN model with multiple output dimensions to make predictions for all tasks. However, one is unable to employ a single explainer to explain the above model, since current explainers are trained specifically to explain one prediction task. As a result, in the case of ToxCast, one must train 167 explainers to explain the GNN model. Second, in industry settings, it is common to train GNN models in a two-stage fashion due to scaling, latency, and label sparsity issues. The first stage trains a GNN-based embedding model with a massive amount of unlabeled data in an unsupervised manner to learn embeddings for nodes or graphs. The second stage trains lightweight models such as multilayer perceptrons (MLPs) using the frozen embeddings as input to predict the downstream tasks. In the first stage, the downstream tasks are usually unknown or undefined, and existing task-specific explainers cannot be applied. Also, there can be tens to hundreds of downstream tasks trained on these GNN embeddings, and training a separate explainer for each task is undesirable and downright impossible.

To address the above limitations, we present a new task-agnostic explanation pipeline, where the learned explainer is independent from downstream tasks and can take downstream models as input conditions, as shown in Figure 1. Specifically, we decompose a prediction model into a GNN embedding model and a downstream model, designing separate explainers for each component. We design the downstream explainers to cooperate with the embedding explainer. The embedding explainer is trained using a self-supervised training framework, which we dub Task-Agnostic GNN Explainer (TAGE), with no knowledge of downstream tasks, models, or labels. In contrast to existing explainers, the learning objective for TAGE is computed at the graph or node embeddings without involving task-related predictions. In addition to eliminating the need for downstream tasks in

Table 1: Comparisons on properties of common GNN explainers. Inductivity and task-agnosticism are inapplicable for gradient/rule-based methods as they do not require learning. In the last column, we show the number of required explainers for a dataset with N samples and M tasks.

	Learning	Inductive	Task-agnostic	# explainers required
Gradient- & Rule-based	No	-	-	1
GNNExplainer [34]	Yes	No	No	$M * N$
SubgraphX [36]	Yes	No	No	$M * N$
PGExplainer [15]	Yes	Yes	No	M
Task-agnostic explainers	Yes	Yes	Yes	1

TAGE, we argue that the self-supervision performed on the embeddings can bring an additional performance boost in terms of the explanation quality compared to existing task-specific baselines such as GNNExplainer and PGExplainer.

We summarize our contributions as follows: 1) We introduce the task-agnostic explanation problem and propose a two-stage explanation pipeline involving an embedding explainer and a downstream explainer. This enables the explanation of multiple downstream tasks with a single embedding explainer. 2) We propose a self-supervised training framework TAGE, which is based on conditioned contrastive learning to train the embedding explainer. The training of TAGE requires no knowledge of downstream tasks. 3) We perform experiments on real-world datasets and observe that TAGE outperforms existing learning-based explanation baselines in terms of explanation quality, universal explanation ability, and the time required for training and inference.

Relations with Prior Work Our work studies a novel explanation problem under the two-stage and multi-task settings. The settings are important in both industrial and academic scenarios but have not been studied by prior work. Whereas existing studies focus on designing optimization approaches [34, 36] and explainer architectures [15] under the typical task-specific setting, our work focuses on an orthogonal problem to enable task-agnostic explanations with the proposed framework including the universal embedding explainer and conditioned learning objectives.

2 Task-Agnostic Explanations

2.1 Notations and Learning-based GNN explanation

Our study considers the attributed graph G with node set V and edge set E. We formulate the attributed graph as a tuple of matrices $(\boldsymbol{A}, \boldsymbol{X})$, where $\boldsymbol{A} \in \{0,1\}^{|V|\times|V|}$ denotes the adjacency matrix and $\boldsymbol{X} \in \mathbb{R}^{|V|\times d_f}$ denotes the feature matrix with feature dimension of d_f. We assume that the prediction model F that is to be explained operates on graph-structured data through two components: a GNN-based embedding model and lighter downstream models. Denoting the input space by $\mathcal{G}$, a node-level embedding model $\mathcal{E}_n : \mathcal{G} \to \mathbb{R}^{|V|\times d}$ takes a graph as input and computes embeddings of dimension d for all nodes in the graph, whereas a graph-level embedding model $\mathcal{E}_g : \mathcal{G} \to \mathbb{R}^{1\times d}$ computes an embedding for the input graph. Subsequently, the downstream model $\mathcal{D} : \mathbb{R}^d \to \mathbb{R}$ computes predictions for the downstream task based on the embeddings.

Typical GNN explainers consider a task-specific GNN-based model as a complete unit, *i.e.*, $F := \mathcal{D} \circ \mathcal{E}$. Given a graph G and the GNN-based model F to be explained , our goal is to identify the subgraph G_{sub} that contributes the most to the final prediction made by F. In other words, we claim that a given prediction is made because F captures crucial information provided by some subgraph G_{sub}. The learning-based (or optimization-based) GNN explanation employs a parametric explainer $\mathcal{T}_\theta$ associated with the GNN model F to compute the subgraph G_{sub} of the given graph data. Concretely, the explainer $\mathcal{T}_\theta$ computes the importance score for each node or edge, denoted as w_i or w_{ij}, or masks for node attributes denoted as m. It then selects the subgraph G_{sub} induced by important nodes and edges, *i.e.*, whose scores exceed a threshold t, and by masking the unimportant attributes. In our study, we follow [15], focusing on the importance of edges to provide explanations to GNNs. Formally, we have $G_{sub} := (V, E_{sub}) = \mathcal{T}_\theta(G)$, where $E_{sub} = \{(v_i, v_j) : (v_i, v_j) \in E, w_{ij} \geq t\}$.

2.2 Task-Agnostic Explanations

As introduced in Section 1, all existing learning-based or optimization-based explanation approaches are task-specific and hence suffer from infeasibility or inefficiency in many real-application scenarios. In particular, they are of limited use when downstream tasks are unknown or undefined, and fail to employ a single explainer to explain a multitask prediction model.

To enable the explanation of GNNs in two-stage training and multitask scenarios, we introduce a new explanation paradigm called the task-agnostic explanation. The task-agnostic explanation considers a whole prediction model as an embedding model followed by any number of downstream models. It focuses on explaining the embedding model regardless of the number or the existence of downstream models. In particular, the task-agnostic explanation trains only one explainer $\mathcal{T}_\theta^{(tag)}$ to explain the embedding model $\mathcal{E}$, which should satisfy the following features. First, given an input graph G, the explainer $\mathcal{T}_\theta^{(tag)}$ should be able to provide different explanations according to specific downstream tasks being studied. Table 1 compares the properties of common GNN explanation methods and the desired task-agnostic explainers in multitask scenarios. Second, the explainer $\mathcal{T}_\theta^{(tag)}$ can be trained when only the embedding model is available, *e.g.*, at the first stage of a two-stage training paradigm, regardless of the presence of downstream tasks. When downstream tasks and models are unknown, $\mathcal{T}_\theta^{(tag)}$ can still identify which components of the input graph are important for certain embedding dimensions of interest.

3 The TAGE Framework

Our explanation framework TAGE follows the typical scheme of GNN explanation introduced in the previous section. It provides explanations by identifying important edges in a given graph and removing the edges that lead to significant changes in the final prediction. Specifically, the goal of the TAGE is to predict the importance score for each edge in a given graph. Different from existing methods, the proposed TAGE breaks down typical end-to-end GNN explainers into two components. We now provide general descriptions and detailed formulations to the proposed framework.

3.1 Task-Agnostic Explanation Pipeline

Following the principle of the desired task-agnostic explanations, we introduce the task-agnostic explanation pipeline, where a typical explanation procedure is performed in two steps. In particular, we decompose the typical end-to-end learning-based GNN explainer into two parts: the embedding explainer $\mathcal{T}_\mathcal{E}$ and the downstream explainer $\mathcal{T}_{down}$, corresponding to the two components in the two-stage training and prediction procedure. We compare the typical explanation pipeline and the two-stage explanation pipeline in Figure 1. The embedding explainer and downstream explainers can be trained or constructed independently from each other. In addition, the embedding explainer can cooperate with any downstream explainers to perform end-to-end explanations on input graphs.

The downstream explainer aims to explain task-specific downstream models. As downstream models are usually lightweight MLPs, we simply adopt gradient-based explainers for downstream explainers without training. The downstream explainer takes a downstream model and the graph or node embedding vector as inputs and computes the importance score of each dimension on the embedding vector. The importance scores then serve as a condition vector input to the embedding explainer. Given the condition vector, the embedding explainer explains the GNN-based embedding model by identifying an important subgraph from the input graph data. In other words, given different condition vectors associated with different downstream tasks or models, the embedding explainer can provide corresponding explanations for the same embedding model. Formally, we denote the downstream explainer for models from $\mathscr{D}$ by $\mathcal{T}_{down} : \mathscr{D} \times \mathbb{R}^d \to \mathbb{R}^d$, which maps input models and embeddings into importance scores $\boldsymbol{m}$ for all embedding dimensions. We denote the embedding explainer associated with the embedding model $\mathcal{E}$ by $\mathcal{T}_\mathcal{E} : \mathbb{R}^d \times \mathcal{G} \to \mathcal{G}$, which maps a given graph into a subgraph of higher importance, conditioned on the embedding dimension importance $\boldsymbol{m} \in \mathbb{R}^d$.

The training procedures of the embedding explainer are independent of downstream tasks or downstream explainers. In particular, the downstream explainer is obtained from the downstream model only, and the training of the embedding explainer only requires the embedding model and the input graphs. As downstream models are usually constructed as stacked fully connected (FC) layers and the

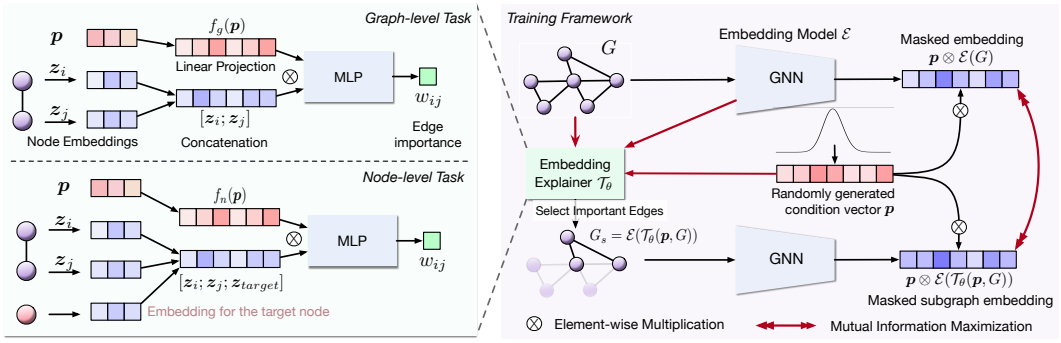

Figure 2: Overviews of the self-supervised training framework for the embedding model (right) and the architecture of the parametric explainers (left). During training, we generate random condition vectors $\boldsymbol{p}$ as an input to the embedding explainer and mask the embeddings. The learning objective seeks to maximize the mutual information between two embeddings on certain dimensions.

explanation of FC layers has been well studied, our study mainly focuses on the non-trivial training procedure and design of the embedding explainer.

3.2 Training embedding explainer under Self-supervision

A straightforward idea of explaining an embedding model with no knowledge of downstream tasks is to employ existing explainers and perform explanation on the pretext task, such as graph reconstruction [10] or context prediction [6], used during the pre-training of GNNs. However, such explanations cannot be generalized to future downstream tasks as there are limited dependencies between the pretext task and downstream tasks. Therefore, training an embedding explainer without downstream models or labels is challenging, and it is desirable to develop a generalizable training approach for the embedding explainer. To this end, we propose a self-supervised learning framework for the embedding explainer.

The learning objective of the proposed framework seeks to maximize restricted mutual information (MI) between two embeddings, i.e., one of the given graph and one of the corresponding subgraph of high importance induced by the explainer, in a conditioned subspace. We introduce a masking vector $\boldsymbol{p} \in \mathbb{R}^d$ as the condition to indicate specific dimensions of embeddings on which to maximize the MI. During the explanation, we obtain the masking vector from the importance vector computed by any downstream explainer $\mathcal{T}_{down}$. As no downstream importance vector is available at training, we sample the masking vector $\boldsymbol{p}$ from a multivariate Laplace distribution due to the sparse gradient, *i.e.*, only a few dimensions are of high importance, assuming embeddings from well-trained models are informative with low dimension redundancy. Empirically, the Laplacian assumption holds on all datasets we work with as we observe that gradients follow a Laplace distribution, as shown in Appendix B. Formally, the learning objective based on the restricted MI is

$$\max_{\theta} \mathrm{E}_{\boldsymbol{p}}[\mathbf{MI}(\boldsymbol{p} \otimes \mathcal{E}(G), \boldsymbol{p} \otimes \mathcal{E}(\mathcal{T}_\theta(\boldsymbol{p}, G)))], \tag{1}$$

where $\mathbf{MI}(\cdot, \cdot)$ computes the mutual information between two random vectors, $\boldsymbol{p}$ denotes the random masking vector sampled from a certain distribution, $\mathcal{T}_\theta(\boldsymbol{p}, G)$ computes the subgraph of high importance, and $\otimes$ denotes the element-wise multiplication, which applies masking to the embeddings $\mathcal{E}(\cdot)$. Figure 2 outlines the training framework and objective. Intuitively, given an input graph and the desired embedding dimensions to be explained, the explainer $\mathcal{T}_\theta$ predicts the subgraph whose embedding shares the maximum mutual information with the original embedding on the desired dimensions.

Practically, the mutual information is intractable and is hence hard to directly compute. A common approach to achieving efficient computation and optimization is to adopt the upper bound estimations of mutual information [31], namely, the Jenson-Shannon Estimator (JSE) [18] and the InfoNCE [4]. These upper bound estimations are also referred to as contrastive loss and are widely applied in self-supervised representation learning [5, 23, 24] for both images and graphs. Adopting these

estimators, the objectives are efficiently computed as

$$\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} \log\left[\sigma\big((\boldsymbol{p}\otimes\boldsymbol{z}_i)^T(\boldsymbol{p}\otimes\boldsymbol{z}_{i,\theta})\big)\right] + \frac{1}{N^2-N}\sum_{i\neq j}\log\left[1-\sigma\left((\boldsymbol{p}\otimes\boldsymbol{z}_i)^T(\boldsymbol{p}\otimes\boldsymbol{z}_{j,\theta})\right)\right], \tag{2}$$

$$\min_{\theta} -\frac{1}{N}\sum_{i=1}^{N}\left[\log\frac{\exp\{(\boldsymbol{p}\otimes\boldsymbol{z}_i)^T(\boldsymbol{p}\otimes\boldsymbol{z}_{i,\theta})\}}{\sum_{j\neq i}\exp\{(\boldsymbol{p}\otimes\boldsymbol{z}_i)^T(\boldsymbol{p}\otimes\boldsymbol{z}_{j,\theta})\}}\right], \tag{3}$$

for JSE and InfoNCE, respectively, where N denotes the number of samples in a mini-batch, σ denotes Sigmoid function, $\boldsymbol{z}_i$ and $\boldsymbol{z}_{i,\theta}$ are embeddings of the original graph G_i and its subgraph $\mathcal{T}_\theta(G_i)$, or target nodes of the two graphs. Our objective involves condition vectors as masks on the embeddings, which differs from typical contrastive loss used in self-supervised representation learning. We hence call the proposed objective the conditioned contrastive loss.

To restrict the size of subgraphs given by the explainer, we follow previous studies [15] to add a size regularization term R, computed as the averaged importance score, to the above objectives. In the case where edge importance scores $w_{ij} \in [0,1]$ are computed, the regularization term is computed as

$$R(G) = \sum_{(v_i,v_j)\in E} \lambda_s|w_{ij}| - \lambda_e\left[w_{ij}\log w_{ij} - (1-w_{ij})\log(1-w_{ij})\right], \tag{4}$$

where λ_s and λ_e are hyper-parameters controlling the size and the entropy of edge importance scores, respectively. We provide detailed descriptions of the objectives in Appendix B.

3.3 Explainer Architectures

Embedding explainers. To be consistent with PGExplainer, we adopt the multilayer perceptron (MLP) as the base architecture to predict the importance score w_{ij} for each edge $(u_i, u_j) \in E$, on top of learned embeddings $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ of the two nodes connected by the edge. Edges with scores higher than a threshold are considered important edges that remain in the selected subgraph. In order for the embedding explainer to cooperate with different downstream explainers and provide diverse explanations for different tasks, it additionally requires a condition vector as input indicating the specific downstream task to be explained. Formally, the graph-level embedding explainer takes the embeddings, $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, and the condition vector $\boldsymbol{p}$ as inputs and computes the importance score by

$$w_{ij} = \text{MLP}_g\big([\boldsymbol{z}_i;\boldsymbol{z}_j]\otimes\sigma(f_g(\boldsymbol{p}))\big), \tag{5}$$

where $[\cdot;\cdot]$ denotes the concatenation along the feature dimension, $\otimes$ denotes the element-wise multiplication, σ denotes the activation function, and $f_g : \mathbb{R}^d \to \mathbb{R}^{2d}$ is a linear projection. The node-level embedding explainer takes an additional node embedding as its input, as the explainers are expected to predict different scores for the same edge when explaining different target nodes. The formulation of computing the importance score is as follows,

$$w_{ij} = \text{MLP}_n\big([\boldsymbol{z}_i;\boldsymbol{z}_j;\boldsymbol{z}_{target}]\otimes\sigma(f_n(\boldsymbol{p}))\big), \tag{6}$$

where $f_g : \mathbb{R}^d \to \mathbb{R}^{3d}$ is a linear projection, and z_{target} denotes the embedding of the target node whose prediction is to be explained.

Downstream explainers. We use the following gradient-based explainer to compute condition vectors for different downstream models. Formally, given an input embedding $\boldsymbol{z}$ and its prediction probabilities $\mathcal{D}(\boldsymbol{z}) \in [0,1]^C$ among all C classes, we compute the gradient of the maximal probability w.r.t. the input embedding:

$$\boldsymbol{g} = \frac{\partial \max_{c\leq C}\mathcal{D}(\boldsymbol{z})[c]}{\partial \boldsymbol{z}} \in \mathbb{R}^{1\times d},$$

where $\mathcal{D}(\boldsymbol{z})[c]$ denotes the probability for class c. To convert the gradient into the condition vector, we further perform normalization and only take positive values reflecting only positive influence to the predicted class probability, *i.e.*, $\boldsymbol{p} = \text{ReLU}(\text{norm}(\boldsymbol{g}^T))$.

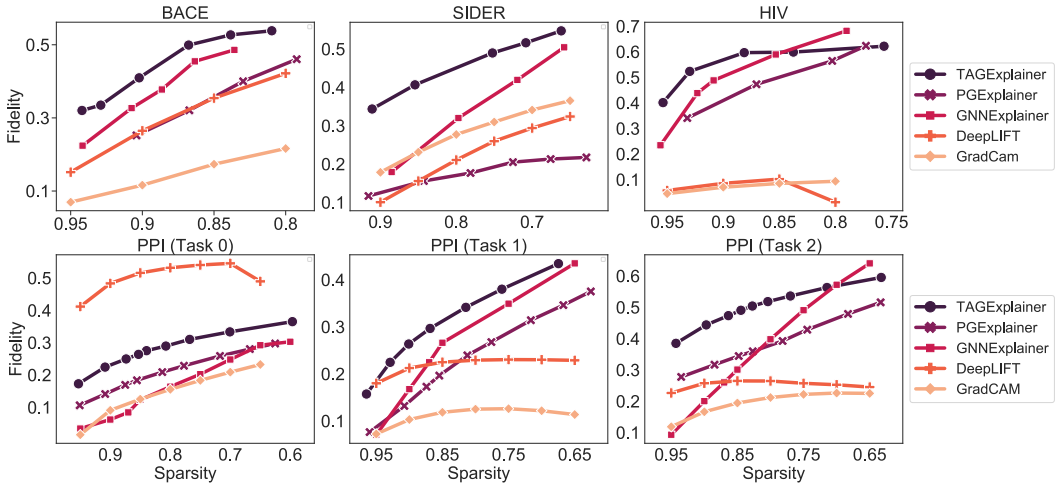

Figure 3: Quantitative performance comparisons on six tasks from MoleculeNet (top row) and PPI (bottom row). The curves are obtained by varying the threshold for selecting important edges.

4 Experimental Studies

We conduct two groups of quantitative studies evaluating the explanation quality and the universal explanation ability, *i.e.*, training a single explainer to explain all downstream tasks, of TAGE. We then compare the efficiency of multiple learning-based GNN explainers in terms of training and explanation time costs. We further provide visualizations to demonstrate the explanation quality as well as the ability to explain GNN models without downstream tasks.

4.1 Datasets

To demonstrate the effectiveness of the proposed TAGE on both node-level and graph-level tasks, we evaluate TAGE on three groups of real-world datasets that contain potentially multiple tasks. The datasets are described as follows and their statistics are summarized in Appendix A.

MoleculeNet. The MoleculeNet [29] library provides a collection of molecular graph datasets for the prediction of different molecule properties. In a molecular graph, each atom in the molecule is considered a node, and each bond is considered an edge. The prediction of molecule properties is a graph-level task. We include three graph classification tasks from MoleculeNet to evaluate the explanation of graph-level tasks: HIV, SIDER, and BACE.

Protein-Protein Interaction. The Protein-Protein Interaction (PPI) [39] dataset documents the physical interactions between proteins in 24 different human tissues. In PPI graphs, each protein is considered as a node with its motif and immunological features, and there is an edge between two proteins if they interact with each other. Each node in the graphs has 121 binary labels associated with different protein functions. As different protein functions are not exclusive to each other, the prediction of each protein function is considered an individual task instead of a multi-class classification. And hence typical approaches require individual explainers for the 121 tasks. We utilize the first five out of 121 tasks to evaluate the explanation of node-level tasks.

E-commerce Product Network. The E-commerce Product Network (EPN)[3] is constructed with subsampled, anonymized logs from an e-commerce store, where entities including buyers, products, merchants, and reviews are considered as nodes, and interactions between entities are considered as edges. We subsample the data for the sake of experimental evaluations and the dataset characteristics do not mirror actual production traffic. We study the explanation of the classification of fraudulent entities (nodes), where the predictions for different types of entities are considered individual tasks. We evaluate our framework specifically on classifications of the buyer, merchant, and review nodes.

[3]Proprietary dataset

4.2 Experiment Settings and Evaluation Metrics

For each real-world dataset, we evaluate explainers on multiple downstream tasks that share a single embedding model. For consistency with industrial use cases, we perform the two-stage training paradigm to obtain GNN models to be explained. In particular, we first use unlabeled graphs to train the GNN-based embedding model in an unsupervised fashion. We then freeze the embedding model and use the learned embeddings to train individual downstream models structured as 2-layer MLPs. Specifically, for graph-level classification tasks in MoleculeNet, we employ the GNN pretraining strategy context prediction [6] to train a 5-layer GIN [32] as the embedding model on ZINC-2M [22] containing 2 million unlabeled molecules. For the node-level classification on PPI, we employ the self-supervised training method GRACE [38] to train a 2-layer GCN [11] on all 21 graphs from PPI without using labels. For the larger-scale node-level classification on EPN, we use graph autoencoder (GAE) [10] to train the embedding model on sampled subgraphs of EPN. More implementation details are provided in Appendix B.

As the involved real-world datasets do not have ground truth for explanations, we follow previous studies [20, 37, 36] to adopt a fidelity score and a sparsity score to quantitatively evaluate the explanations. Intuitively, the fidelity score measures the level of change in the probability of the predicted class when removing important nodes or edges, whereas the sparsity score measures the relative amount of important nodes or nodes associated with important edges. A formulation of the scores is provided in Appendix B. Note that compared to explanation evaluation with ground truths, the fidelity score is considered more faithful to the model, especially when the model makes incorrect predictions, in which case the explanation ground truths become inconsistent with the evidence of making the wrong predictions. In practice, one needs to trade-off between the fidelity score and the sparsity score by selecting the proper threshold for the importance.

Table 2: Fidelity scores with controlled sparsity on graph-level molecule property prediction tasks. Each column corresponds to an explainer model trained on (or without) a specific downstream task. Underlines highlight the best explanation quality in terms of fidelity, on the same level of sparsity.

	PGExplainer (trained on)				TAGE
Eval on	BACE	HIV	BBBP	SIDER	w/o downstream
BACE	**0.252 ±0.340**	0.007 ±0.251	0.026 ±0.022	-0.151 ±0.330	**0.378 ±0.293**
HIV	-0.001 ±0.197	**0.473 ±0.404**	0.013 ±0.029	-0.060 ±0.356	**0.595 ±0.321**
BBBP	0.001 ±0.237	-0.056 ±0.226	**0.182 ±0.169**	-0.252 ±0.440	**0.193 ±0.161**
SIDER	0.012 ±0.219	-0.009 ±0.212	0.003 ±0.029	**0.444 ±0.391**	**0.521 ±0.278**

4.3 Quantitative Studies

We conduct two groups of quantitatively experimental comparisons. We first demonstrate the explanation quality of individual tasks in terms of the fidelity score and the sparsity score. We do this by comparing TAGE with multiple baseline methods including non-learning-based methods GradCAM [20] and DeepLIFT [21], as well as learning-based methods GNNExplainer [34] and PGExplainer [15]. We do not include other optimization or search-based methods such as Monte-Carlo tree search [9] due to the significant time cost on real-world datasets. Note that to show the effectiveness of universal explanations over different downstream tasks, we only train one embedding explainer for all tasks in a dataset, on top of which a gradient-based downstream explainer is applied to explain multiple downstream tasks. In contrast, for existing learning-based methods, we need to train multiple explainers to explain downstream tasks individually. For all methods, we vary the

Table 3: Fidelity scores with controlled sparsity on the node-level classification dataset PPI. Each column corresponds to an explainer model trained on (or without) a specific downstream task. Underlines highlight the best explanation quality in terms of fidelity, on the same level of sparsity.

	PGExplainer (trained on)					TAGE
Eval on	Task 0	Task 1	Task 2	Task 3	Task 4	w/o downstream
Task 0	**0.184 ±0.3443**	-0.005 ±0.268	0.033 ±0.335	0.034 ±0.310	0.018 ±0.194	**0.271 ±0.385**
Task 1	0.046 ±0.447	**0.197 ±0.380**	0.043 ±0.314	0.008 ±0.297	0.021 ±0.183	**0.300 ±0.415**
Task 2	0.028 ±0.434	0.001 ±0.283	**0.345 ±0.458**	0.024 ±0.320	0.097 ±0.320	**0.499 ±0.480**
Task 3	0.075 ±0.364	-0.015 ±0.219	0.036 ±0.317	**0.262 ±0.418**	0.040 ±0.221	**0.289 ±0.427**
Task 4	0.035 ±0.413	-0.021 ±0.238	0.223 ±0.438	0.075 ±0.374	**0.242 ±0.373**	**0.330 ±0.442**

Table 4: Fidelity scores with controlled sparsity on the E-commerce product dataset. Each column corresponds to one explainer model trained on different tasks or without downstream tasks. Underlines highlight the best explanation quality in terms of fidelity, on the same level of sparsity.

	PGExplainer (trained on)			TAGE
Eval on	Buyers	Sellers	Reviews	w/o downstream
Buyers	**0.2009 ±0.2233**	0.1731 ±0.3774	0.1740 ±0.4463	**0.2713 ±0.1834**
Sellers	0.5465 ±0.4773	**0.3246 ±0.4026**	0.1128 ±0.3019	**0.6515 ±0.3426**
Reviews	0.4178 ±0.3683	0.1258 ±0.3492	**0.2310 ±0.4178**	**0.5692 ±0.4214**

Table 5: Comparison of computational time cost among three learning-based GNN explainers on the PPI dataset. The left two columns record time cost breakdown for T downstream tasks. The fourth column estimates the total time cost for explaining all 121 tasks of PPI. The last row shows the speedup times compared to GNNExplainer and PGExplainer, respectively.

Time cost	Training (s)	Inference (s)	Total time (T=1) (s)	Est. total for 121 tasks
GNNExplainer	20040.1*T	–	20040.1	28 d
PGExplainer	7117.0*T	**427.2*T**	7604.2	10.7 d
TAGE	**1405.3**	582.7*T	**1988.0**	**0.83 d**
Speedup	**14.3*T× / 5.1*T×**	– / 0.73×	**10.1× / 3.8×**	**33.7× / 12.9×**

threshold for selecting important nodes or edges and compare how fidelity scores change over sparsity scores on each task and dataset. The results are shown in Figure 3. In particular, TAGE outperforms other learning-based explainers on BACE, SIDER, and PPI (tasks 0 and 1). For HIV and PPI (task 2), TAGE is more effective at higher sparsity levels, *i.e.*, when fewer nodes are considered important and masked.

To justify the necessity of task-agnostic explanation and demonstrate the universal explanation ability of TAGE, we include PGExplainer as our baseline and compare the explanation quality when adopting a single explainer to explain multiple downstream tasks. For PGExplainer, we train multiple explainers on different downstream tasks and evaluate each explainer on different downstream tasks. For TAGE, we train one explainer without downstream tasks and evaluate it on different downstream tasks. Results shown in Table 2 (MoleculeNet), Table 3 (PPI), and Table 4 (EPN) indicate that task-specific explainers fail to generalize to different downstream tasks and hence are unable to provide universal explanations. On the other hand, the task-agnostic explainer, although trained without downstream tasks, can provide explanations with even higher quality for downstream tasks.

GNNExplainer and PGExplainer should generally outperform task-agnostic explainers, as they are specific to data examples or tasks. This should especially be true when TAGE and PGExplainer have the same level of parameters. However, we find that TAGE outperforms the learning-based baselines. We believe that the underperformance of baselines is due to the non-injective characteristic of the downstream MLPs, where different embeddings can produce similar downstream prediction results. In other words, a similar downstream prediction are not necessarily produced by embeddings that share high mutual information. Due to this characteristic, the learning objective of TAGE computed between embeddings brings stronger supervision than the objective computed between final predictions, as the latter objective does not guarantee consistency between embeddings or between input graphs and subgraphs.

Multitask Explanation Efficiency A major advantage of the task-agnostic explanation is that it removes the need for training individual explainers, which consumes the majority of the total time cost to explain a model on a dataset. We hence evaluate the efficiency of TAGE in terms of time cost for explanation and compare it to the two learning-based explainer baselines. We record the time cost for the training and inference of different explanation methods on the same dataset and device, shown in Table 5. All results are obtained from running the explanation on the PPI dataset with 121 node classification tasks with a single Nvidia Tesla V100 GPU. Although the inference time cost of TAGE is slightly higher than that of PGExplainer, the results show TAGE costs significantly less time than GNNExplainer and PGExplainer, especially in the multitask cases ($T > 1$). TAGE allows the explanation of many downstream tasks within a reasonable time duration.

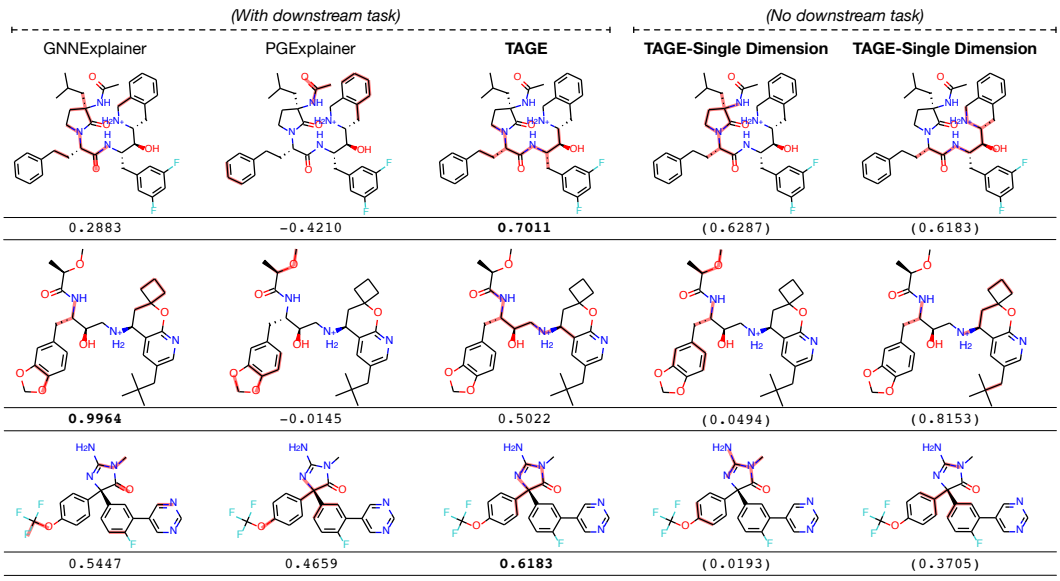

Figure 4: Visualizations on explanations to the GNN model for the BACE task. The top 10% important edges are highlighted with red shadow. The numbers below molecules are fidelity scores when masking out the top 10% important edges. The right two columns are explanations for two certain embedding dimensions without downstream tasks. Fidelity scores in the right two columns explaining two embedding dimensions are still computed for the BACE task but are just for reference.

4.4 Visualization of Explanations

We visualize the explanations of the three learning-based explanation methods on the BACE task, which aims to predict the inhibitor effect of molecules on human β-secretase 1 (BACE-1) [29]. Additional visualizations on HIV and SIDER are also provided in Appendix E. The visualization results are shown in Figure 4. Each molecule visualization shows the top 10% important edges (bonds) predicted by an explainer marked in red, together with the fidelity score on the molecule. The left three columns are explanation results with the BACE downstream task. The right columns are explanations by TAGE to two specific graph embedding dimensions, without downstream models. Embedding dimensions with greater values among all are selected in the visualizations. To obtain explanations of certain embedding dimensions, we input the one-hot vectors to the embedding explainer as condition vectors. The visualization results indicate that while baseline methods select scattered edges as important, TAGE tends to select edges that form a connected substructure, which is more reasonable when explaining molecule property predictions where a certain functional group is important for the property. While there are no ground-truth explanations for the BACE, the validity of results produced by TAGE can be evidenced by multiple domain researches [7, 8]. We discuss them in Appendix E. In addition, the right three columns indicate that dimensions in the embedding correspond to different substructures and TAGE is able to provide explanations to the dimensions without downstream tasks.

5 Conclusions

Existing task-specific learning-based explainers become inapplicable under real scenarios when downstream tasks or models are unavailable and suffer from inefficiency when explaining real-world graph datasets with multiple downstream tasks. We introduced TAGE, including the task-agnostic GNN explanation pipeline and the self-supervised training framework to train the embedding explainer without knowing downstream tasks or models. Our experiments demonstrate that the TAGE generally achieves higher explanation quality in terms of fidelity and sparsity with a significantly reduced explanation time cost. We discuss potential limitations and their solutions in Appendix G.

References

[1] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference*, pages 417–426, 2019.

[2] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[3] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017.

[4] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[5] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

[6] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020.

[7] Dandan Huang, Yonglan Liu, Bozhi Shi, Yueting Li, Guixue Wang, and Guizhao Liang. Comprehensive 3D-QSAR and binding mode of BACE-1 inhibitors using R-group search and molecular docking. *Journal of Molecular Graphics and Modelling*, 45:65–83, 2013.

[8] Priti Jain and Hemant R. Jadhav. Quantitative structure activity relationship analysis of aminoimidazoles as bace-i inhibitors. *Medicinal Chemistry Research*, 22(4):1740–1746, 2013.

[9] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4849–4859. PMLR, 2020.

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

[11] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[12] Wanyu Lin, Hao Lan, and Baochun Li. Generative causal explanations for graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

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

[14] Meng Liu, Youzhi Luo, Limei Wang, Yaochen Xie, Hao Yuan, Shurui Gui, Haiyang Yu, Zhao Xu, Jingtun Zhang, Yi Liu, Keqiang Yan, Haoran Liu, Cong Fu, Bora M Oztekin, Xuan Zhang, and Shuiwang Ji. DIG: A turnkey library for diving into graph deep learning research. *Journal of Machine Learning Research*, 22(240):1–9, 2021.

[15] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. In *Advances in Neural Information Processing Systems*, pages 19620–19631, 2020.

[16] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. GraphDF: A discrete flow model for molecular graph generation. In *Proceedings of the 38th International Conference on Machine Learning*, pages 7192–7203. PMLR, 2021.

[17] Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *Proceedings of the 39th International Conference on Machine Learning*, pages 15524–15543. PMLR, 2022.

[18] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035. 2019.

[20] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10772–10781, 2019.

[21] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3145–3153, 2017.

[22] Teague Sterling and John J Irwin. ZINC 15 – Ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015.

[23] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2019.

[24] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, 2019.

[25] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[26] Limei Wang, Yi Liu, Yuchao Lin, Haoran Liu, and Shuiwang Ji. ComENet: Towards complete and efficient message passing for 3D molecular graphs. In *Advances in Neural Information Processing Systems*, 2022.

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

[28] Zhengyang Wang, Meng Liu, Youzhi Luo, Zhao Xu, Yaochen Xie, Limei Wang, Lei Cai, Qi Qi, Zhuoning Yuan, Tianbao Yang, and Shuiwang Ji. Advanced graph and sequence neural networks for molecular property prediction and drug discovery. *Bioinformatics*, 38(9):2579–2586, 2022.

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

[30] Yaochen Xie, Zhao Xu, and Shuiwang Ji. Self-supervised representation learning via latent graph prediction. In *Proceedings of the 39th International Conference on Machine Learning*, pages 24460–24477. PMLR, 2022.

[31] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[32] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[33] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[34] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems*, pages 9244–9255, 2019.

[35] Haiyang Yu, Limei Wang, Bokun Wang, Meng Liu, Tianbao Yang, and Shuiwang Ji. GraphFM: Improving large-scale GNN training via feature momentum. In *Proceedings of the 39th International Conference on Machine Learning*, pages 25684–25701. PMLR, 2022.

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

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

[38] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020.

[39] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Appendix G.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In Appendix B.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In Section 4.3 - Multitask Explanation Efficiency.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The EPN dataset is anonymized.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]