Self-Interpretable Graph Learning with Sufficient and Necessary Explanations

Jiale Deng, Yanyan Shen[*]

Department of Computer Science and Engineering, Shanghai Jiao Tong University
{jialedeng, shenyy}@sjtu.edu.cn

Abstract

Self-interpretable graph learning methods provide insights to unveil the black-box nature of GNNs by providing predictions with built-in explanations. However, current works suffer from performance degradation compared to GNNs trained without built-in explanations. We argue the main reason is that they fail to generate explanations satisfying both sufficiency and necessity, and the biased explanations further hurt GNNs' performance. In this work, we propose a novel framework for generating SUfficient aNd NecessarY explanations (SUNNY-GNN for short) that benefit GNNs' predictions. The key idea is to conduct augmentations by structurally perturbing given explanations and employ a contrastive loss to guide the learning of explanations toward sufficiency and necessity directions. SUNNY-GNN introduces two coefficients to generate hard and reliable contrastive samples. We further extend SUNNY-GNN to heterogeneous graphs. Empirical results on various GNNs and real-world graphs show that SUNNY-GNN yields accurate predictions and faithful explanations, outperforming the state-of-the-art methods by improving 3.5% prediction accuracy and 13.1% explainability fidelity on average. Our code and data are available at https://github.com/SJTU-Quant/SUNNY-GNN.

Introduction

Graph Neural Networks (GNNs) are widely employed for learning representations of graph-structured data, such as social networks (Xiao et al. 2023), co-purchase graphs (Li et al. 2020), and paper citation networks (Luo et al. 2020a). Despite their remarkable expressive power to model complex graph data, the black-box nature of GNNs poses a significant obstacle that hinders their usage in many application domains that require trust and accountability. Therefore, exploring the explainability of GNNs is crucial since good explanations can provide insights into the decision logic inside GNNs. There have been numerous works proposed for explaining trained GNNs by extracting salient substructures of input graphs as explanations in a post-hoc manner (Ying et al. 2019; Luo et al. 2020b; Wang et al. 2021b; Li et al. 2023). However, post-hoc explanations are not directly generated from GNNs and they are recognized to be biased or inconsistent (Dai and Wang 2021; Miao, Liu, and Li 2022).

[*]corresponding author.

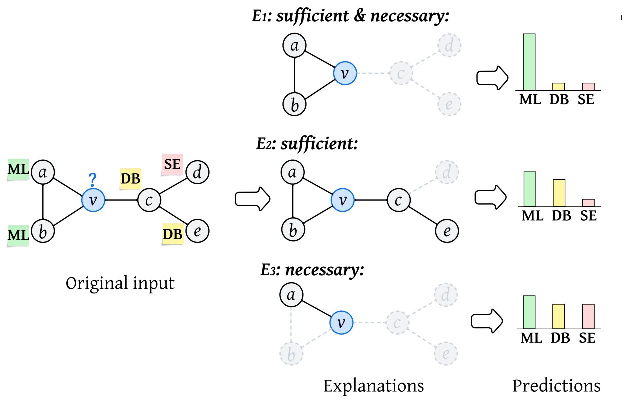

Figure 1: An example of predicting the label (research area) of the author node v in a coauthor network. A sufficient and necessary explanation needs to preserve crucial information such that a GNN can make the correct prediction confidently.

To overcome the shortcomings of post-hoc GNN explainers, self-interpretable graph learning methods that simultaneously provide predictions and built-in explanations has become a prominent solution. A self-interpretable graph learning framework typically consists of (1) an *explanation generator* that extracts substructure from the input graph as the explanation, and (2) a *GNN encoder* that learns the representation of the generated explanation for the final prediction. While such a framework produces a distinct explanation for each prediction, the *quality* of the generated explanations is a crucial factor that determines the GNN performance. One line of works (Yu et al. 2020; Dai and Wang 2021; Miao, Liu, and Li 2022) seek explanations that contain **sufficient** information for the GNN to make correct predictions. Another line of works (Sui et al. 2022; Fan et al. 2022; Wu et al. 2022) focus on discovering **necessary** substructures, expecting that any corruption to the substructures would prevent the GNN from predicting correctly. Nevertheless, a sufficient explanation may introduce noisy information and a necessary explanation may miss crucial patterns, both hurting the GNN performance. In fact, existing self-interpretable graph learning methods suffer from the degradation of the GNN performance, i.e., up to 15.2% lower accuracy than the original GNNs (see Table 2 for details).

Example 1 *Fig. 1 shows an example of a coauthor network, where nodes represent authors and edges denote coauthor relationships. Consider a GNN model that aims to predict the research area of a target node v from ML, DB and SE. Assume the truth label of v is ML. E_1 is an ideal explanation that enables the GNN to infer the truth label of v confidently according to the rationale that v, a, b form an ML research group. E_2 is a sufficient explanation which contains the crucial clique among v, a, b for the prediction. E_3 is a necessary explanation since corrupting (a, v) causes damage to the complete ML research group information (the clique) and hurts the prediction. However, E_2 involves two noisy edges $(v, c), (c, e)$. As c, e are labeled as DB, E_2 may mislead the GNN to classify v as DB. Meanwhile, E_3 fails to disclose the crucial clique to the GNN and may reduce its confidence on ML. In either case, the GNN's performance tends to be compromised due to the biased explanations.*

In this paper, we propose SUNNY-GNN, a novel self-interpretable graph learning framework with sufficient and necessary explanations. Our objective is to promote the quality of the generated explanations toward both sufficiency and necessity directions, encouraging the explanations to reach a sweet spot that improves GNN's performance.

To achieve this, for each generated explanation E, we perform augmentations on E to get its positive and negative samples by adding and corrupting edges, respectively. Our novel attempt is to utilize a contrastive loss $\mathcal{L}_{cts}$ that supervises the explanation generator to produce sufficient and necessary explanations, enabling the GNN to make correct predictions confidently. By analogy to contrastive learning (Oord, Li, and Vinyals 2018), minimizing $\mathcal{L}_{cts}$ means pulling positive samples closer to the anchor while pushing negative samples distant from the anchor. In our context, if E is not a sufficient explanation, it fails to provide enough crucial information for the prediction. Adding edges to E judiciously is able to fill in the missing information that benefits the prediction (e.g., E_3 versus E_1 in Example 1). Hence, we minimize $\mathcal{L}_{cts}$ to pull closer E and its positive samples in their representations produced by the GNN encoder, aiming to turn the explanation to be more sufficient. If E is not a necessary explanation, corrupting E slightly will not change the prediction significantly (e.g., E_2 versus E_1 in Example 1). This deduces that the representations of E and its negative samples are closer after the GNN encoder. Hence, pushing them apart by minimizing $\mathcal{L}_{cts}$ is a supervision signal that promotes the explanation towards the necessity direction. To these ends, minimizing $\mathcal{L}_{cts}$ guides the explanation generator to improve explanations from both sufficiency and necessity perspectives. Finally, our SUNNY-GNN framework combines $\mathcal{L}_{cts}$ with the classification loss, facilitating the GNN's predictions to benefit from the generated faithful explanations in an end-to-end manner.

It is noteworthy that how to augment explanations for generating effective contrastive signals remains challenging. On one hand, we prefer hard positive and negative samples that strengthen the contrastive signals. To build them, we introduce *distance coefficients* to control the strength of perturbations on E and emphasize hard contrastive samples. On the other hand, the positive and negative samples should be reliable. We thus introduce *confidence coefficients* to help filter out positive samples that introduce noisy edges into explanations and negative samples that trivially remove irrelevant edges from explanations (w.r.t. the prediction).

The contribution of this paper can be summarized as:

- We demonstrate existing self-interpretable graph learning suffers from the GNN performance degradation and describe the importance of generating sufficient and necessary explanations that benefit the GNN performance.
- We propose a novel self-interpretable graph learning framework named SUNNY-GNN. It augments explanations to derive contrastive samples and employs a contrastive loss to supervise the explanation generator toward distilling sufficient and necessary explanations.
- We introduce the distance and confidence coefficients to generate hard and reliable contrastive samples that lead to effective contrastive signals.
- SUNNY-GNN achieves state-of-the-art performance in terms of prediction accuracy and explanation quality on various real-world graph datasets. Furthermore, we extend our approach to self-interpretable heterogeneous graph learning and observe a remarkable improvement.

Preliminaries

Graph. Let $G = (\mathcal{E}, \mathcal{V})$ denote a graph where $\mathcal{V}$ is the node set and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ is the edge set. The graph structure can be described by an adjacency matrix $A \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$, with $A_{uv} = 1$ indicating the existence of a directed edge (u, v) from node u to node v and $A_{uv} = 0$ otherwise. We denote the node feature matrix as $X \in \mathbb{R}^{|\mathcal{V}| \times d^{(0)}}$ where $d^{(0)}$ is the input feature dimension of each node.

Graph Neural Network. We consider a GNN model that takes a graph G as input and learns node representations for node classification tasks. Typically, GNN employs the message-passing mechanism to propagate and aggregate information from immediate neighbors along edges. For the l-th layer of an L-layer GNN g, we can compute the representation of node v as: $z_v^{(l)} = g^{(l)}(z_v^{(l-1)}, \{z_u^{(l-1)} | u, (u, v) \in \mathcal{E}\})$, where $z_v^{(0)} = X_v$. The last-layer node representations will be used to make predictions. Each node in the training set $\mathcal{V}_{train}$ is labeled with $y \in \mathcal{Y}$. Label $y \in \mathbb{R}^{|\mathcal{T}|}$ is a one-hot vector and $\mathcal{T}$ is the set of classes. Commonly $|\mathcal{V}_{train}|$ is much smaller than $|\mathcal{V}|$ in real-world tasks.

For an L-layer GNN, the representation of target node v is determined by its L-hop neighbors (Ying et al. 2019). Generally, its local computation graph $G_C = (\mathcal{E}_C, \mathcal{V}_C)$[1] can be extracted from G using these neighbors. As shown in Fig. 3, G_C indicates how messages are propagated, aggregated, and updated layer by layer inside a GNN.

Problem Formulation. We follow previous works (Miao, Liu, and Li 2022; Sui et al. 2022) and employ salient

[1]Subscripts for target node v are omitted in the rest of this paper for the simplification of notations.

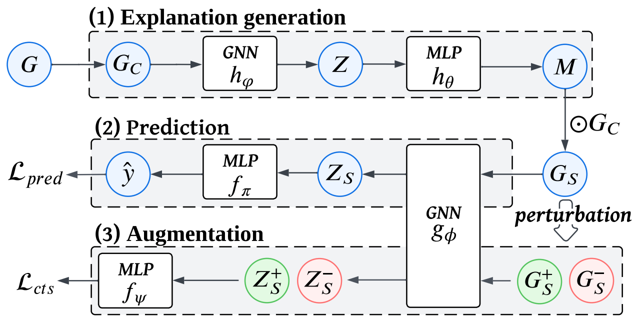

Figure 2: The framework of SUNNY-GNN.

edges as explanations. This paper aims to develop a self-interpretable graph learning framework containing an *explanation generator* h, a *GNN encoder* g, and a *classifier* f. Given a node $v \in \mathcal{V}$ and a graph G, the framework is able to produce faithful explanations using $G_S = h(G, v)$ and accurate predictions with $\hat{y} = f(g(G_S))$.

Methodology

In this section, we first introduce the framework of SUNNY-GNN, including explanation generation, explanation augmentation, model prediction and the optimization objective. We then extend SUNNY-GNN to heterogeneous graphs.

Framework Overview

Fig. 2 illustrates the overall workflow. (1) ***Explanation generation***: We first extract the computation graph G_C of target node v from G, and encode it into node representations Z by GNN encoder h_φ. With Z as input, the decoder MLP h_θ outputs a saliency map $M \in \mathbb{R}^{|\mathcal{E}_C|}$, where m_{ij} denotes the importance score of edge (i, j). The explanation G_S is then generated by combining G_C and M. (2) ***Prediction***: We encode G_S into representations Z_S using another GNN encoder g_ϕ, and produce prediction $\hat{y}$ with MLP-based classifier f_π. The prediction loss $\mathcal{L}_{pred}$ is the cross-entropy between $\hat{y}$ and the label y. (3) ***Augmentation***: we augment G_S into a set of positive and negative samples $\{G_S^+, G_S^-\}$. The encoded representations of augmented samples are fed into a projection head f_ψ to compute the contrastive loss $\mathcal{L}_{cts}$. The whole framework is trained in an end-to-end manner by iteratively updating model parameters $\Theta = \{\varphi, \theta, \phi, \pi, \psi\}$ supervised by $\mathcal{L}_{pred}$ and $\mathcal{L}_{cts}$. Without loss of generality, we employ 2-layer GNN encoders, i.e., $L = 2$ for h_φ and g_ϕ.

Explanation Generation

For any target node v, the explanation generator h assigns a saliency map M to the edges of G_C, and generates an attentive explainable graph $G_{S_{att}} = G_C \odot M$. To ensure the differentiability of backpropagation during training, the explanation G_S is in the form of $G_{S_{att}}$, where the weighted edge importance score is used to control the message aggregation, similar to the graph attention mechanism (Hamilton, Ying, and Leskovec 2017). To provide human-understandable explanations after training, we sample important edges from $G_{S_{att}}$ to form G_S, for example, $G_S \sim \text{Bern}(G_{S_{att}})$, where $\text{Bern}(\cdot)$ is the Bernoulli distribution parameterized by M (Miao, Liu, and Li 2022). The number of sampled edges is $|\mathcal{E}_S| = \lfloor k|\mathcal{E}_C| \rfloor$, where $k \in (0, 1]$ is the sample ratrio.

We now introduce how to produce the saliency map M with h. We first extract the 2-hop computation graph G_C of v, as illustrated in the first step of Fig. 3, and then compute the importance scores for edges in G_C. To be more specific, for a directed edge (i, j) from node i to node j in G_C, we compute its importance score by:

- $m_{ij} = h_\theta(z_i^{(0)} || z_j^{(1)} || z_v^{(2)})$, if (i, j) connects nodes in layer 0 and layer 1, e.g., edge (e, c) in Fig. 3;
- $m_{ij} = h_\theta(z_i^{(1)} || z_j^{(2)} || z_v^{(2)})$, if (i, j) connects nodes in layer 1 and layer 2, e.g., edge (c, v) in Fig. 3;

where h_θ is an MLP parameterized with θ and $||$ means the concatenation operation. We concatenate the target node representation $z_v^{(2)}$ into the edge embedding, because the explanations of different nodes in a graph may have diverse structures (Luo et al. 2020b). For GNN encoders with different feature dimensions in each layer (e.g., $d^{(0)} \neq d^{(1)} \neq d^{(2)}$), we train MLP-based mapping functions p_μ parameterized with μ that project the features of each layer into a common dimension $d^{(2)}$, i.e., $\tilde{z}_i^{(0)} = p_{\mu_0}(z_i^{(0)})$ and $\tilde{z}_i^{(1)} = p_{\mu_1}(z_i^{(1)})$. The mapped representation $\tilde{z}$ is then used to compute m_{ij}.

Augmentation

For a given explanation G_S, we respectively augment G_S toward sufficiency and necessity directions and treat the augmented explanation as positive and negative samples in contrastive learning. We perform augmentations by structurally perturbing the generated explanatory subgraph G_S. To be more specific, considering G_S as the anchor:

- Positive samples $G_S^+ = \{G_{S_1}^+, \cdots, G_{S_{n^+}}^+\}$ are created by adding edges to G_S, where n^+ is the number of positive samples. If G_S is an insufficient explanation and the samples in G_S^+ lead to correct prediction as G_S, pulling G and the samples in G_S^+ closer in their representations is essentially guiding the explanation toward the sufficiency direction. To construct each $G_{S_i}^+, i \in \{1, \cdots, n^+\}$, we sample r^+ edges from $G_C \backslash G_S$ and add them to G_S. Fig. 3 provides an example of a positive sample $G_{S_i}^+ = G_S \oplus (c, v)$, where we add one edge (c, v) to G_S;
- Negative samples $G_S^- = \{G_{S_1}^-, \cdots, G_{S_{n^-}}^-\}$ are created by removing edges from G_S, where n^- is the number of negative samples. If G_S lacks necessity, corrupting its edges into the samples in G_S^- may not change the prediction. Hence, by pushing G_S distant from the samples in G_S^- in their representations, we expect G_S and the samples in G_S^- leading to different predictions, thus guiding the explanation toward the necessity direction. To construct $G_{S_j}^-, j \in \{1, \cdots, n^-\}$, we sample r^- edges from G_S and remove them from G_S. An example for negative samples is shown as $G_{S_j}^- = G_S \ominus (a, b)$ in Fig. 3, where we remove one edge (a, b) from G_S.

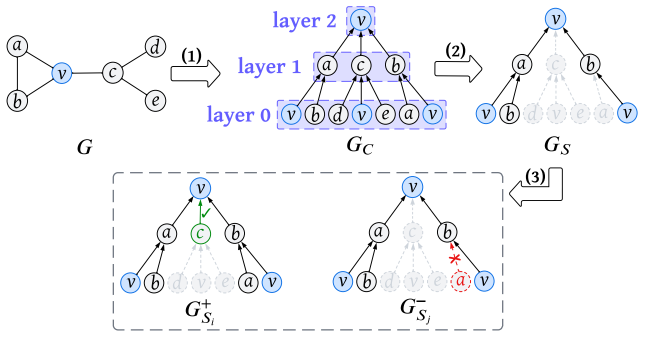

Figure 3: An example of getting augmented samples. For a target node v on the input graph G, we (1) extract its computation graph G_C from G, and (2) sample the explanation G_S from G_C, then (3) get the augmented positive and negative samples $\{G^+_S, G^-_S\}$ by adding or corrupting edges.

Note that in the training phase, G_S is in the form of $G_{S_{att}}$ and we actually perform *soft* augmentations on $G_{S_{att}}$. Specifically, we increase the edge weights of sampled positive edges, equivalent to adding them into G_S, and reduce the edge weights of sampled negative edges, equivalent to removing them from G_S. Due to the space limit, we provide the detailed algorithm in the supplementary material.

A simple way to generate positive and negative samples is to sample edges based on their importance scores in M. However, this way may result in trivial samples that impair the self-supervision signals or unreliable samples that mislead the GNN training. To address the problem, we introduce two coefficients, aiming to get hard and reliable positive and negative samples that ensure effective self-supervision.

Distance Coefficients. As contrastive learning benefits from hard samples (Robinson et al. 2020), we introduce the distance coefficients to control the perturbation strength and create hard positive and negative samples. Recall that the addition and removal operations of edges are implemented by increasing or reducing the corresponding edge weights. Due to the hierarchical tree structure of G_C, intuitively, *perturbations on edges closer to the target node v tend to have a greater impact to v than those on farther edges.* For the example in Fig. 3, increasing the weight of edge (c, v) will amplify not only the information flowing from c to v, but also the information from d, v, and e through c to v. This principle also applies to the removal operation.

Hence, for positive samples, adding an edge that is closer and related to the target node will introduce more information than adding a farther, unrelated one, which is harder to be pulled closer to G_S. For example, $G_S \oplus (c, v)$ is a harder positive sample than $G_S \oplus (d, c)$. Similarly, for negative samples, removing a farther edge leads to less information decrease, which is more difficult to be pushed away from G_S. For example, $G_S \ominus (a, b)$ is a harder negative sample than $G_S \ominus (b, v)$. Formally, we define the **distance coefficients** δ^+ and δ^- as additional weights to the edges in G_C:

$$\delta^+ = 1 - \alpha \cdot \exp(\mathrm{d}) \in \mathbb{R}^{|\mathcal{E}_C|}, \tag{1}$$

$$\delta^- = \alpha \cdot \exp(\mathrm{d}) \in \mathbb{R}^{|\mathcal{E}_C|}, \tag{2}$$

where $\alpha \in \mathbb{R}^+$ is a positive constant and $\mathrm{d} \in \{1, ..., L\}^{|\mathcal{E}_C|}$. $\mathrm{d}_{(i,j)} = l$ means there are l steps starting from the source node i of edge (i, j) to reach v. For example, $\mathrm{d}_{(e,c)} = 2$ since it takes 2 steps from e to v. The positive samples are obtained by adding edges (i.e., increasing edge weights) randomly sampled by $M \odot \delta^+$, while the negative samples are obtained by removing edges (i.e., reducing edge weights) sampled by $M \odot \delta^-$.

Confidence Coefficient. We leverage label information and model output to filter out unreliable augmented samples. Specifically, when constructing positive samples, introducing noise into G_S may result in a decrease in prediction confidence, making these samples far away from the sufficiency direction. Similarly, when constructing negative samples, removing irrelevant edges from G_S may not affect the prediction, i.e., the negative samples tend to have the same prediction as G_S. In either case, the augmented explanations are *untrustworthy* and will mislead the GNN training. Therefore, we want to mitigate the effects of untrustworthy positive and negative samples on the training process. Formally, we define the **confidence coefficients** η^+ and η^- to reweight the augmented samples:

$$\eta^+ = \mathrm{SoftMax}(f_\pi(g_\phi(G^+_S))_{\mathcal{Y}_{vt}}) \in \mathbb{R}^{n^+}, \tag{3}$$

$$\eta^- = (1 - \mathrm{SoftMax}(f_\pi(g_\phi(G^-_S))_{\mathcal{Y}_{vt}})) \in \mathbb{R}^{n^-}, \tag{4}$$

where $f_\pi(g_\phi(G^+_S))_{\mathcal{Y}_{vt}}$ is the prediction probability of augmented samples in the truth label $t \in \mathcal{T}$ of target node v.

Prediction and Optimization

Given explanation G_S, the model outputs the prediction by:

$$z_S = g_\phi(G_S),\ \hat{y} = f_\pi(z_S). \tag{5}$$

We devise the following objective to train SUNNY-GNN:

$$\min_{\Theta} \mathcal{L}_{pred} + \gamma \mathcal{L}_{cts}, \tag{6}$$

where γ is a trade-off hyperparameter, $\mathcal{L}_{pred}$ is the supervised prediction loss term and $\mathcal{L}_{cts}$ is the contrastive loss term for optimizing the explanations.

Prediction Loss. We compute $\mathcal{L}_{pred}$ by the average cross-entropy loss on all labeled nodes, which is defined as:

$$\mathcal{L}_{pred} = -\frac{1}{|\mathcal{V}_{train}|} \sum_{v \in \mathcal{V}_{train}} \sum_{t=1}^{\mathcal{T}} \mathcal{Y}_{vt} \log \hat{\mathcal{Y}}_{vt}, \tag{7}$$

where $\hat{\mathcal{Y}}_{vt}$ is the t-th entry of the model output for labeled node v and $\mathcal{Y}_{vt}$ is the corresponding truth label.

Contrastive Loss. We compute $\mathcal{L}_{cts}(v)$ for node v by:

$$\mathcal{L}_{cts}(v) = \mathbb{E}\left[-\log \frac{\eta^+_i e^{z_S^\top z^+_{S_i}/\tau}}{\sum_j \eta^+_j e^{z_S^\top z^+_{S_j}/\tau} + \sum_k \eta^-_k e^{z_S^\top z^-_{S_k}/\tau}}\right], \tag{8}$$

where $z_S = f_\psi(g_\phi(G_S))$, $z^+_S = f_\psi(g_\phi(G^+_S))$ and $z^-_S = f_\psi(g_\phi(G^-_S))$ are the projected representations of G_S, G^+_S and G^-_S, respectively. $\tau > 0$ is a scalar temperature hyperparameter. Then $\mathcal{L}_{cts}$ over all labeled nodes is computed by:

$$\mathcal{L}_{cts} = \frac{1}{|\mathcal{V}_{train}|} \sum\nolimits_{v \in \mathcal{V}_{train}} \mathcal{L}_{cts}(v). \tag{9}$$

	Citeseer	Cora	Pubmed	Amazon	Coauthor-CS	Coauthor-Physics
$\|\mathcal{V}\|$	3327	2708	19717	7650	18333	34493
$\|\mathcal{E}_C\|$ [2]	68	162	318	31443	861	3413
$\|\mathcal{E}\|$	9228	10556	88651	238163	163788	495924
$\|\mathcal{T}\|$	6	7	3	8	15	5
$\|\mathcal{V}_{train}\|$	120	140	60	100	100	100

Table 1: The statistics of the datasets.

Adapting SUNNY-GNN to Heterogeneous Graphs

Real-world graph learning is often more complex due to the heterogeneity of node and edge types. For example, in a citation network, nodes may have different types such as $paper$, $author$, and $conference$. To generalize our method to such scenarios, we make the following adjustments. The detailed algorithm is provided in the supplementary material.

- We train different mapping functions to map representations of different node types into the same dimension for calculating the saliency map with h.
- We take edge types and meta-path information into consideration when performing augmentations. Meta-path is a predefined semantic pattern. For example, the $author \leftrightarrow paper \leftrightarrow author$ meta-path defines the "coauthor" relationship. Nodes/edges in a path that instantiates the meta-path are semantically related. To construct positive samples, we prioritize the edges that, when added to G_S, will instantiate a pre-defined meta-path. These samples are considered hard positive samples since they introduce more semantic information and make it harder to optimize h toward the sufficiency direction. Similarly, to construct negative samples, we prioritize the edges that are not present in the meta-path instances. They are considered hard negative samples since the corrupted edges are not informative and make it harder to optimize h toward the necessity direction.

Time Complexity Analysis

In the training phase, for each node $v \in \mathcal{V}_{train}$, the time cost is based on its local computation graph size $|\mathcal{E}_C|$. The cost of generating an explanation is $\mathcal{O}(|\mathcal{E}_C|)$. Given the explanation G_S, the cost of computing $\mathcal{L}_{pred}$ is $\mathcal{O}(|\mathcal{E}_S|)$ and the cost of computing $\mathcal{L}_{cts}$ is $\mathcal{O}((n^+ + n^-)|\mathcal{E}_S|)$. $|\mathcal{E}_S| = k|\mathcal{E}_C|$ with $k \in (0, 1]$. Hence, the overall training time complexity is $\mathcal{O}(T \sum_v ((1 + n^+ + n^-)k|\mathcal{E}_C|))$, where T is the number of training iterations. In the inference phase, the time cost for generating the explanation and prediction is $\mathcal{O}((1 + k)|\mathcal{E}_C|)$.

Experiments

Experimental Settings

Datasets. We run experiments on five real-world datasets for node classification tasks. Their statistics are presented in Table 1. We select three widely-used benchmark datasets in citation networks: **Citeseer**, **Cora**, and **Pubmed** (Yang, Cohen, and Salakhudinov 2016), where nodes represent scientific publications and edges represent citation relationships. To further validate the effectiveness of our method on large-scale graphs, we introduce another three datasets. Their local connectivity of nodes is more complex. **Amazon** (Shchur et al. 2018) is the Amazon Photo subset of the Amazon co-purchase graph, whose nodes represent goods and edges represent that two goods are frequently bought together. **Coauthor-CS** and **Coauthor-Physics** (Shchur et al. 2018) are coauthorship networks based on the Microsoft Academic Graph, where nodes are authors, that are connected by an edge if they co-authored a paper. For further tasks in heterogeneous scenarios, we use **IMDB**, **DBLP** and **ACM** datasets. Their detailed statistics can be found in previous works (Wang et al. 2021a; Lv et al. 2021).

Self-Interpretability Baselines. The backbone GNN encoders are implemented with the widely-employed **GCN** (Kipf and Welling 2016) and **GAT** (Hamilton, Ying, and Leskovec 2017) architectures. We use the original graph as input to train the base GNNs, and regard their prediction performance as the **base performance**. We use **SimpleHGN** (Lv et al. 2021) as the backbone GNN in heterogeneous datasets. We also compare prediction performance with state-of-the-art self-interpretable graph learning methods. (1) **GSAT** (Miao, Liu, and Li 2022) injects stochasticity to edges and leverages the reduction of stochasticity to select important edges under the guidance of information bottleneck principle. (2) **CAL** (Sui et al. 2022) proposes the causal attention graph learning strategy that encourages the GNNs to exploit the causal rationales necessary for making correct predictions. (3) **SE-GNN** (Dai and Wang 2021) identifies a set of K-nearest labeled nodes in the holistic graph as explanations, and utilizes them to make predictions. (4) **ProtGNN** (Zhang et al. 2022) learns a set of abstract prototypes as explanations and makes predictions by comparing the input graph with prototypes.

Post-hoc Explainability Baselines. We compare the explainability performance with advanced post-hoc explainers with edge saliency masks as explanations. (1) **GNNExplainer** (Ying et al. 2019) individually learns soft masks for target nodes. (2) **PGExplainer** (Luo et al. 2020b) trains a global parameterized mask predictor to generate edge masks. (3)**ReFine** (Wang et al. 2021b) employs a pre-training and fine-tuning framework to train the mask predictor. Additionally, self-interpretable methods, e.g., GSAT, CAL, and SUNNY-GNN are able to give post-hoc explanations by replacing the GNN encoder with the target GNN.

Metrics. We report classification accuracy (Acc) to measure prediction performance. For explainability, we use fidelity metrics (Yuan et al. 2022) measuring the faithfulness of explanations to the model. Fidelity includes negative fidelity (fid_-) and positive fidelity (fid_+). fid_- **measures sufficiency** by evaluating the prediction change when only explanations are retained. fid_+ **measures necessity** by comparing the original predictions with new predictions obtained without explanations. Lower fid_- indicates better sufficiency, while higher fid_+ indicates better necessity.

[2] $|\mathcal{E}_C|$ is average number of edges in 2-hop computation graphs, which describes the local structural complexity of a graph dataset.

	Citeseer	Cora	Pubmed	Amazon	Coauthor-CS	Coauthor-Physics
GCN	69.84±0.7	81.20±0.7	<u>77.68±0.7</u>	<u>90.18±0.3</u>	83.52±0.4	<u>92.46±0.2</u>
+ GSAT	**70.90±1.1**	<u>81.48±0.7</u>	77.44±0.3	88.36±1.3	<u>83.76±0.6</u>	92.14±0.5
+ CAL	65.60±1.1	75.72±1.2	73.66±0.8	84.32±1.7	82.12±1.2	91.26±0.7
+ SE-GNN	68.90±0.9	80.72±0.1	77.56±0.3	-	83.14±0.8	-
+ ProtGNN	66.30±2.1	77.48±8.7	74.18±3.3	82.46±1.4	79.50±3.7	88.80±3.3
+ SUNNY-GNN	<u>70.72±0.8</u>	**81.68±0.9**	**78.68±0.2**	**90.43±0.4**	**85.03±1.1**	**93.10±0.8**
Average impro. (%)	3.6 ↑	3.1 ↑	3.4 ↑	4.8 ↑	3.2 ↑	3.0 ↑
GAT	69.68±1.2	<u>81.22±0.7</u>	<u>77.50±0.4</u>	89.08±1.8	<u>84.42±0.8</u>	<u>92.30±0.5</u>
+ GSAT	69.42±0.8	81.20±0.7	77.04±0.3	<u>89.73±0.4</u>	84.37±0.7	91.90±0.8
+ CAL	67.64±1.5	76.64±1.1	74.74±0.7	84.86±11.5	78.69±3.8	78.24±5.1
+ SE-GNN	68.18±1.1	79.46±0.4	75.88±0.4	-	83.71±0.5	-
+ ProtGNN	<u>69.90±1.5</u>	80.40±0.9	76.84±0.8	86.52±0.3	80.95±1.2	90.42±2.3
+ SUNNY-GNN	**71.30±0.7**	**82.18±1.3**	**78.14±0.3**	**90.78±0.4**	**85.13±0.5**	**93.06±0.6**
Average impro. (%)	3.2 ↑	3.1 ↑	2.3 ↑	3.0 ↑	3.4 ↑	6.0 ↑

Table 2: Classification Acc(%). The best and second-best results are bolded and underlined, respectively.

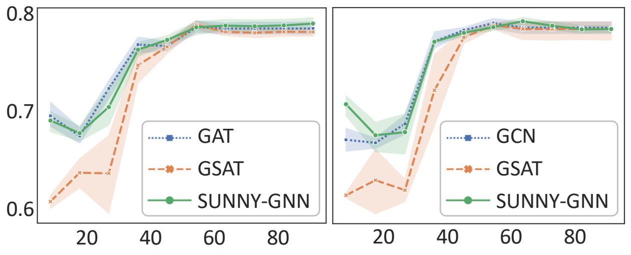

Figure 4: The classification Acc (y-axis) in Pubmed dataset w.r.t. the number of labeled nodes (x-axis).

Formally, they are computed by:

$$fid_{-} = \frac{1}{N}\sum_{i=1}^{N} |\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i{}^{G_S} = y_i)|, \tag{10}$$

$$fid_{+} = \frac{1}{N}\sum_{i=1}^{N} |\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i{}^{G_C \setminus G_S} = y_i)|, \tag{11}$$

where the indicator function $\mathbb{1}(\hat{y}_i = y_i)$ returns 1 if $\hat{y}_i = y_i$ and returns 0 otherwise. N is the number of test samples. $\hat{y}_i{}^{G_S}$ and $\hat{y}_i{}^{G_C \setminus G_S}$ are the model output when inputting the explanation G_S and its complement $G_C \setminus G_S$, respectively.

Implementation Details. We use PyTorch to implement SUNNY-GNN. For baselines, we utilize their original codes with minor adaptations to fit them to tasks in this paper. For SUNNY-GNN, we set coefficient of contrastive loss $\gamma = 0.01$ and the temperature hyperparameter $\tau = 0.1$. All the experiments are conducted 5 times with different random seeds and average results with standard deviations are reported. More details can be found in the supplementary material.

Evaluations of Prediction Performance

Table 2 shows the prediction performance of SUNNY-GNN and all baseline methods. Our method outperforms the baselines by 3.5% on average and up to 6.0%. We have the following observations:

	IMDB	DBLP	ACM
Simple-HGN	62.24±0.7	95.70±0.8	91.44±0.5
+ GSAT	62.18±1.3	95.74±0.8	91.06±0.7
+ SUNNY-GNN	63.10±0.9	95.73±0.5	92.10±0.6

Table 3: Classification Acc(%) in heterogeneous datasets.

(1) **SUNNY-GNN surpasses the base performance while most baselines fail to.** SUNNY-GNN improves the base performance by 1.1% on average and up to 2.3%, as it is able to preserve sufficient and necessary explanations for capturing useful information. GSAT is the second-best baseline, as it achieves an average approximation of -0.1% over the base performance, with a maximum improvement of 1.5%. Its graph information bottleneck principle is loosed to preserve a sufficient amount of salient information while lacking the ability to filter out noise. CAL uses causal attention to select important causal substructures. However, in most cases, the discovered causal structures are sparse. The lack of important information leads to a significant decline in prediction performance compared to the base performance. We are not able to reproduce the results of SE-GNN in Coauthor-Physics and Amazon, due to the extreme memory cost (more than 90 GB) for storing pairwise edge similarity to derive the K-nearest neighbors as explanations.

(2) **SUNNY-GNN is more robust when the labeled nodes get fewer.** Based on Table 2, we observe that SUNNY-GNN exhibits a higher average improvement (3.7%) in datasets with fewer labels (e.g., Pubmed, Amazon, Coauthor-CS and Coauthor-Physics) compared to the improvement (3.2%) in other datasets with more labels. We conduct an extensive experiment to verify this observation. We reduce the number of labeled nodes in Pubmed dataset and record the change of prediction accuracy, as shown in Fig. 4. We compare SUNNY-GNN with base GNNs and the second-best baseline, GSAT. We find that SUNNY-GNN could steadily approximate the base performance as the number of labels decreased, while GSAT fails to. As the

	Citeseer		Cora	
	$fid_+ \uparrow$	$fid_- \downarrow$	$fid_+ \uparrow$	$fid_- \downarrow$
GCN				
+ GNNExplainer	72.27±4.2	9.31±3.4	38.29±4.1	1.08±0.4
+ PGExplainer	82.09±7.2	0.92±2.6	87.47±0.9	1.42±0.3
+ ReFine	83.01±7.1	0.78±0.5	88.19±0.6	**0.00**±**0.0**
+ GSAT	86.75±5.7	2.72±1.1	76.11±16.0	1.82±1.0
+ CAL	86.44±4.3	12.25±3.9	82.15±9.1	5.78±0.8
+ SUNNY-GNN	**87.29**±**5.3**	**0.25**±**0.4**	**90.24**±**0.3**	**0.00**±**0.0**
GAT				
+ GNNExplainer	52.95±16.9	8.61±10.1	36.44±21.0	1.43±2.2
+ PGExplainer	76.80±0.3	1.71±2.8	88.25±6.1	0.17±0.2
+ ReFine	77.78±2.8	**0.32**±**1.3**	88.79±3.2	**0.00**±**0.0**
+ GSAT	72.52±8.1	1.55±1.3	77.43±8.7	1.03±0.5
+ CAL	77.78±6.5	11.46±1.4	85.23±9.6	5.34±1.5
+ SUNNY-GNN	**79.25**±**2.4**	0.46±0.5	**91.79**±**3.2**	**0.00**±**0.0**

Table 4: Explainability performance (%). The best and second-best results are bolded and underlined, respectively.

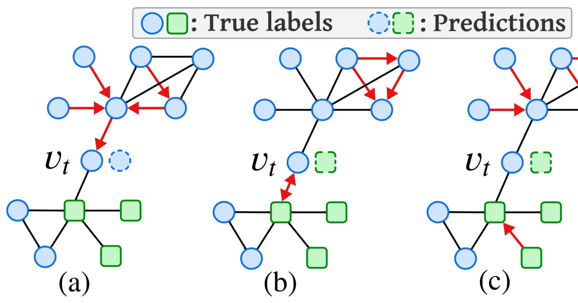

Figure 5: The visualization of explanations generated by (a) SUNNY-GNN, (b) GSAT and (c) CAL in Coauthor-CS.

number of labels decreases, the supervision signal also decreases sharply, making GSAT fail to discover salient explanations. SUNNY-GNN introduces a contrastive loss and constructs more samples through data augmentation, making it more robust to the few-label issue.

(3) **SUNNY-GNN is more effective as the graph structure grows more complex.** In datasets with higher local structural complexity $|\mathcal{E}_C|$ (e.g., Amazon and Coauthor-Physics), SUNNY-GNN shows a higher improvement (4.2%) compared to the improvement (3.2%) in other simpler datasets. To further investigate the performance of self-interpretable methods when it comes to more complex tasks, we conduct extensive experiments in heterogeneous graphs. As shown in Table 3, SUNNY-GNN outperforms heterogeneous baselines 0.8% on average and up to 1.5%.

Evaluations of Explainability Performance

Quantitative Evaluations. Table 4 presents a comparison of the explainability performance of SUNNY-GNN with post-hoc explainable and self-interpretable baselines. Our method outperforms the baselines by 13.1% on average and up to 33.5%. We observe that **SUNNY-GNN generates explanations satisfying both good sufficiency and necessity**, while other baselines fail to. In most cases, SUNNY-GNN achieves low fid_- scores approaching 0 and high fid_+ scores, indicating its capability in generating sufficient and necessary explanations. However, GSAT consistently retains sufficient explanations and CAL tends to generate necessary explanations. For instance, in Citeseer dataset with GAT as backbone, GSAT achieves lower fid_- than CAL, suggesting that it provides more sufficient explanations. Analogously, higher fid_+ implies CAL generates more necessary explanations than GSAT. Experiments in other datasets can be found in the supplementary material.

	Acc ↑	$fid_+ \uparrow$	$fid_- \downarrow$
SUNNY-GNN	71.26±0.7	79.25±2.4	0.46±0.5
w/o $\mathcal{L}_{cts}$	69.01±1.2	76.50±1.4	1.78±0.4
w/o δ	70.78±0.7	79.13±1.8	0.47±0.5
w/o η	70.48±0.8	77.83±1.9	1.28±1.0

Table 5: Individual contributions of proposed modules.

Case Studies. We visualize the generated explanations for author-node *No.15155* of Coauthor-CS, where edges marked in red are considered explanations. As presented in Fig. 5, SUNNY-GNN highlights the most salient input information and prevents harmful information from flowing into v_t, while other baselines (e.g., GSAT and CAL) may introduce noisy information into explanations.

Ablation Studies

As illustrated in Table 5, we evaluate the individual contributions of the contrastive loss $\mathcal{L}_{cts}$, distance coefficients δ, and confidence coefficients η in Citeseer dataset with GAT as backbone. Without $\mathcal{L}_{cts}$, the overall performance has a decrease of 3.4%, suggesting that solely relying on the self-interpretation principle can have a detrimental effect on the model's performance as it fails to learn effective explanations. In contrastive learning, δ is used to prioritize hard samples, and η is used to filter out unreliable samples. The absence of the two coefficients also harms the performance (0.3%↓ and 1.2%↓ respectively).

Conlusion

In this paper, we illustrate the importance of generating sufficient and necessary explanations for improving the performance of self-interpretable graph learning methods. We propose a novel framework, SUNNY-GNN, to generate such explanations while improving prediction performance. It augments an explanation by adding or corrupting edges to obtain positive or negative samples. By empowering the prediction loss with contrastive loss computed from the augmented samples, SUNNY-GNN enbables the explanation generator to produce beneficial explanations that are both sufficient and necessary. Extensive experimental results in real-world datasets show that SUNNY-GNN achieves remarkable improvement in GNNs' performance by providing sufficient and necessary explanations. Furthermore, we extend SUNNY-GNN to heterogeneous graph learning tasks and achieve preliminary results. We leave its applications in more complex graph learning scenarios for future work.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (2022YFE0200500), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), and SJTU Global Strategic Partnership Fund (2021SJTU-HKUST).

References

Dai, E.; and Wang, S. 2021. Towards self-explainable graph neural network. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 302–311.

Fan, S.; Wang, X.; Mo, Y.; Shi, C.; and Tang, J. 2022. Debiasing graph neural networks via learning disentangled causal substructure. *Advances in Neural Information Processing Systems*, 35: 24934–24946.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Li, T.; Deng, J.; Shen, Y.; Qiu, L.; Yongxiang, H.; and Cao, C. C. 2023. Towards Fine-Grained Explainability for Heterogeneous Graph Neural Network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 8640–8647.

Li, Z.; Shen, X.; Jiao, Y.; Pan, X.; Zou, P.; Meng, X.; Yao, C.; and Bu, J. 2020. Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, 1677–1688. IEEE.

Luo, D.; Bian, Y.; Yan, Y.; Liu, X.; Huan, J.; and Zhang, X. 2020a. Local community detection in multiple networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 266–274.

Luo, D.; Cheng, W.; Xu, D.; Yu, W.; Zong, B.; Chen, H.; and Zhang, X. 2020b. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33: 19620–19631.

Lv, Q.; Ding, M.; Liu, Q.; Chen, Y.; Feng, W.; He, S.; Zhou, C.; Jiang, J.; Dong, Y.; and Tang, J. 2021. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 1150–1160.

Miao, S.; Liu, M.; and Li, P. 2022. Interpretable and generalizable graph learning via stochastic attention mechanism. In *International Conference on Machine Learning*, 15524–15543. PMLR.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Robinson, J. D.; Chuang, C.-Y.; Sra, S.; and Jegelka, S. 2020. Contrastive Learning with Hard Negative Samples. In *International Conference on Learning Representations*.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Sui, Y.; Wang, X.; Wu, J.; Lin, M.; He, X.; and Chua, T.-S. 2022. Causal attention for interpretable and generalizable graph classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1696–1705.

Wang, X.; Liu, N.; Han, H.; and Shi, C. 2021a. Self-supervised heterogeneous graph neural network with co-contrastive learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 1726–1736.

Wang, X.; Wu, Y.; Zhang, A.; He, X.; and Chua, T.-S. 2021b. Towards multi-grained explainability for graph neural networks. *Advances in Neural Information Processing Systems*, 34: 18446–18458.

Wu, Y.-X.; Wang, X.; Zhang, A.; He, X.; and Chua, T.-S. 2022. Discovering invariant rationales for graph neural networks. *arXiv preprint arXiv:2201.12872*.

Xiao, S.; Zhu, D.; Tang, C.; and Huang, Z. 2023. Combining Graph Contrastive Embedding and Multi-head Cross-Attention Transfer for Cross-Domain Recommendation. *Data Science and Engineering*, 8(3): 247–262.

Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 40–48. PMLR.

Ying, Z.; Bourgeois, D.; You, J.; Zitnik, M.; and Leskovec, J. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32.

Yu, J.; Xu, T.; Rong, Y.; Bian, Y.; Huang, J.; and He, R. 2020. Graph Information Bottleneck for Subgraph Recognition. In *International Conference on Learning Representations*.

Yuan, H.; Yu, H.; Gui, S.; and Ji, S. 2022. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5): 5782–5799.

Zhang, Z.; Liu, Q.; Wang, H.; Lu, C.; and Lee, C. 2022. Protgnn: Towards self-explaining graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 9127–9135.