SES: Bridging the Gap Between Explainability and Prediction of Graph Neural Networks

Zhenhua Huang[†], Kunhao Li[†], Shaojie Wang[†], Zhaohong Jia[*†], Wentao Zhu[‡], Sharad Mehrotra[§]
[†]Anhui University, Hefei, China
[‡]Amazon Research, Seattle, USA
[§]University of California Irvine, Irvine, USA
[*]Key Lab of Intelligent Computing and Signal Processing of Ministry of Education, Hefei, China
{zhhuangscut, kunhomlihf, wsj.ahu, wentaozhu91}@gmail.com, zhjia@mail.ustc.edu.cn, sharad@ics.uci.edu

***Abstract*—Despite the Graph Neural Networks' (GNNs) proficiency in analyzing graph data, achieving high-accuracy and interpretable predictions remains challenging. Existing GNN interpreters typically provide post-hoc explanations disjointed from GNNs' predictions, resulting in misrepresentations. Self-explainable GNNs offer built-in explanations during the training process. However, they cannot exploit the explanatory outcomes to augment prediction performance, and they fail to provide high-quality explanations of node features and require additional processes to generate explainable subgraphs, which is costly. To address the aforementioned limitations, we propose a self-explained and self-supervised graph neural network (SES) to bridge the gap between explainability and prediction. SES comprises two processes: explainable training and enhanced predictive learning. During explainable training, SES employs a global mask generator co-trained with a graph encoder and directly produces crucial structure and feature masks, reducing time consumption and providing node feature and subgraph explanations. In the enhanced predictive learning phase, mask-based positive-negative pairs are constructed utilizing the explanations to compute a triplet loss and enhance the node representations by contrastive learning.**

Extensive experiments demonstrate the superiority of SES on multiple datasets and tasks. SES outperforms baselines on real-world node classification datasets by notable margins of up to 2.59% and achieves state-of-the-art (SOTA) performance in explanation tasks on synthetic datasets with improvements of up to 3.0%. Moreover, SES delivers more coherent explanations on real-world datasets, has a fourfold increase in Fidelity+ score for explanation quality, and demonstrates faster training and explanation generating times. To our knowledge, SES is a pioneering GNN to achieve SOTA performance on both explanation and prediction tasks.

***Index Terms*—Graph Neural Networks, Model Explanation, Node Classification, Self-Supervised Learning**

I. Introduction

Graph neural networks (GNNs) have become pivotal in handling graph data, proving essential in a variety of applications including node classification [1], [2], knowledge representation [3], [4], molecular classification [5], [6], traffic prediction [7], [8], recommendation system [9], [10], sentiment analysis [11], [12], pose estimation [13], [14], and text classification [15], [16], *etc.*

One group of existing research on graph neural networks focuses on developing novel architectures to enhance predictive accuracy. Typical GNNs include graph convolution networks (GCN) [17], graph attention networks (GAT) [18], GraphSAGE [19], graph isomorphism network (GIN) [20], ARMA [21], UniMP [22], FusedGAT [23], and ASDGN [24], *etc.* However, these advancements have not adequately addressed the need for explainability in the representations learned by GNNs.

To address this, instance-level or model-level approaches have been proposed to offer explanations of GNNs, which are mostly post-hoc models. A post-hoc model is a statistical or predictive model constructed after data processing, enabling retrospective analysis and interpretability of variable relationships [25]. Noteworthy instance-level post-hoc GNN explainers include GNNExplainer [26], PGExplainer [27], PGMExplainer [28], and GraphLIME [29], *etc.* instance-level models offer individualized explanations of structures or features while model-level post-hoc explanation models offer abstract subgraphs conducive to classification for a specific GNN model, such as XGNN [30], PAGE [31] and GNNInterpreter [32], *etc.* The post-hoc explainers contribute to understanding the inner workings of GNNs. Due to the sequential nature of post-hoc GNN explainers, which explain one node at a time, interpreting a batch of nodes using the aforementioned approaches can be time-consuming. Moreover, they require an auxiliary model to explain a target GNN leading to potential bias and misrepresentations [33].

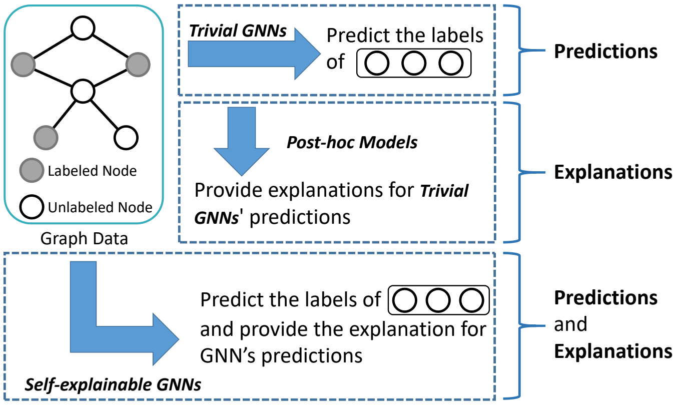


Fig. 1. Categories of GNNs from explanation and prediction perspectives.

TABLE I
CHARACTERISTICS OF SES AND OTHER TYPES OF GRAPH NEURAL NETWORKS.

Method	Prediction	Explainable	Improvement
Trivial graph neural networks (GCN [17], GAT [18], GIN [20], etc.)	✓		
Post-hoc explanation models (GNNExplainer [26], PGExplainer [27], PGMExplainer [28], etc.)		✓	
Self-explainable models (SEGNN [33], ProtGNN [34], PxGNN [35], etc.)	✓	✓	
SES (Ours)	✓	✓	✓

In recent years, self-explainable GNN models have been introduced to address the limitations of post-hoc models, including SEGNN [33], ProtGNN [34], and PxGNN [35]. SEGNN achieves explainable node classification by identifying the K-nearest labeled nodes for each unlabeled node by leveraging node similarity and local structure similarity. However, computing similarity scores between the test node and all labeled nodes is computationally expensive, and finding K-nearest labeled nodes by similarities is unstable. ProtGNN merges prototype learning with GNNs to provide explanations derived from case-based reasoning. However, the inclusion of prototype reflection and Monte Carlo tree search brings additional computational costs. Moreover, the node prototype at the cluster boundary in ProtGNN is not as distinguishable as the subgraph structure, which compromises its node classification efficacy. PxGNN [35] provides class-level explanations and makes classifications by a prototype generator constraining on learnable prototype embeddings. It relies on the prototype generator's capacity and non-representative prototypes can result in subpar interpretative and predictive performance.

Self-explainable GNNs aim to enhance interpretability by jointly modeling representations and explanations. Nevertheless, current self-explainable GNNs exhibit the following issues: (1) Unlike post-hoc methods, they fail to indicate the importance of node features, making them difficult to apply to a wider range of explanation tasks. (2) These methods require additional computations overhead to obtain or search for explanatory subgraphs [33], [34], which is costly. (3) Empirical evidence suggests that these methods exhibit unsatisfactory predictive performance, resulting from ineffective utilization of the interpretation outcomes. Consequently, a chasm persists between the predictive capabilities and the explanations provided by GNNs, which hampers their utility in downstream applications.

As illustrated in Fig. 1, trivial GNNs are typically limited to prediction tasks, offering no interpretive insights. Post-hoc explanation models create explanations for graph data by trained GNNs. In contrast, self-explainable GNNs provide explanations concurrently with the training of node representations. However, they fail to enhance prediction accuracy. We argue that cogent explanations deepen the understanding of graphs and bolster the model's training efficacy and prediction accuracy. To overcome the shortcomings inherent in existing GNNs, we propose a Self-Explained and self-Supervised model (SES) to bridge the gap between the explainability and prediction accuracy of GNN. SES harmonizes the interpretation generation and training of GNNs, instead of treating them as separate processes. The characteristics of GNNs are summarized in TABLE I.

The SES approach consists of two phases: Explainable Training and Enhanced Predictive Learning. During the Explainable Training phase, SES concurrently optimizes the mask generator with the graph encoder to ensure that the feature and structure mask align with the aggregation performed by the backbone GNN. It eliminates the need for computationally expensive searches or constructions of explanatory subgraphs, which significantly reduces the time consumption. The masks are co-trained with the graph encoder with a dedicated loss allowing it to capture the most relevant features and substructures for each node and resulting in accurate explanations. Unlike existing self-explainable GNNs, SES generates masks that provide both feature and structure explanations, making it suitable for a wider range of explanation tasks.

To utilize these explanations and improve prediction performance, inspired by contrastive learning (a self-supervised method), mask-based positive and negative pairs are created to compute a triplet loss that ensures nodes with similar structures are closely situated and discriminates nodes with dissimilar structural attributes. Simultaneously, supervised learning steers the self-supervised training to ensure its alignment with the prediction task. We refer to the second phase as ”Enhanced Predictive Learning” as it enhances the effects of self-supervised training and prevents deviation from the prediction task. Experimental results demonstrate the efficiency and effectiveness of SES across multiple tasks. In summary, our contributions can be summarized as follows:

- We investigate the limitations of current self-explainable GNNs: inadequate feature explanations, costly computations, and plain prediction performance.
- We propose a pioneering self-explained and self-supervised graph neural network that bridges gaps between explainability and prediction of GNNs.
- SES provides more reasonable interpretations of models and significantly reduces the time consumption of generating explanations.
- Extensive quantitative experiments demonstrate that SES achieves SOTA performance in both explanation and prediction tasks on real-world and synthetic datasets.

II. RELATED WORKS

A. Prediction of Graph Neural Networks

Graph neural networks (GNNs) have shown a strong representation ability for dealing with graph format data [36]. The prediction of GNNs is to utilize nodes' or graphs' representation to classify their labels for the downstream tasks. GCN [17] is a typical GNN based on the message-passing among neighbors to aggregate information. GAT [18] utilizes an attention mechanism to enhance the ability of information aggregation based on GCN. GraphSAGE [19] applies a new sampling method to aggregate neighbor nodes to make GNN more scalable and effective. ARMA [21] is a graph convolutional layer by autoregressive and moving average filters to provide a more flexible frequency response and better global graph structure representation. FusedGAT [23] is an optimized version of GAT that fuses message-passing computation for accelerated execution and lower memory footprint. ASDGN [24] is a stable and non-dissipative DGN design framework conceived through ordinary differential equations and preserves remote information between nodes. RAHG [37] is a role-aware GNN considering role features to improve node representation. These GNNs focus on improving the prediction performance without providing explanations.

B. Explanation of Graph Neural Networks

To offer interpretations of GNNs, a considerable amount of methods or GNN explainers are proposed [38]. Many popular explainers are post-hoc models that can be categorized into instance-level and model-level approaches. GNNExplainer [26] is the pioneering instance-level model that provides edge and feature explanations by maximizing the mutual information between the prediction of a GNN and the distribution of possible subgraph structures. PGExplainer [27] adopts a deep neural network to parameterize the explanation generation process, enabling it to provide multi-instance explanations naturally. GraphLIME [29] is a local interpretable model that explains graphs using the Hilbert-Schmidt Independence Criterion (HSIC) Lasso, which focuses on providing feature explanations for graph data. XGNN [30] is the first model-level approach to explain GNNs. It accomplishes this by training a graph generator that maximizes specific predictions made by a model. However, these post-hoc explainers for GNNs rely on trained models and require an additional model or process to support explanations. This dependency leads to a potential misunderstanding between the explainable models and GNNs.

To address the limitations of post-hoc explanations, some self-explainable GNNs that provide explanations during the training process are proposed. SEGNN [33] is the first self-explainable GNN, which utilizes an interpretable similarity module to find the K-nearest labeled nodes for each unlabeled node, enabling explainable node classification. The module considers node and local structure similarity to identify the nearest labeled nodes. ProtGNN [34] combines prototype learning with GNNs, offering a new perspective on GNN explanations. The explanations provided by ProtGNN are derived from the case-based reasoning process. PxGNN [35] is a prototype-based self-explainable GNN that can simultaneously give accurate predictions and prototype-based explanations of predictions.

However, results suggest that SEGNN requires substantial memory and ProtGNN is computationally expensive. Additionally, despite their strengths in elucidating the decision-making processes of GNNs and providing explanations, these models fail to achieve classification accuracy on par with trivial GNNs.

C. Self-Supervised Learning of Graph Neural Networks

Self-supervised learning has emerged as a promising technique for training deep learning models without extensive labeled data [39]. In GNNs, self-supervised learning has gained significant attention due to its ability to learn useful representations from graph data without extra labeling works [40]. GraphCL [41] develops comparative learning with data enhancement for GNN pre-training to address the heterogeneity of graph data. GCC [42] is an unsupervised graph representation learning framework that captures common network topology properties from multiple networks. MERIT [43] combines the advantages of Siamese knowledge distillation and conventional graph contrastive learning. HeCo [44] contrasts heterogeneous graphs using two perspectives (network schema and meta-path) and trains an encoder to maximize the mutual information between node embeddings. MolCLR [45] is a self-supervised GNN framework for molecular contrast learning, proposing enhancement methods to ensure consistency within the same molecule and inconsistency among different molecules. These methods are effective in the pre-training and general training phases of GNNs. However, there are currently no approaches to integrate self-supervised learning into explainable GNNs training effectively.

TABLE II
NOTATIONS IN SES.

Symbols	Definition and description
G	A general graph
V	Node set
A	Adjacency matrix
$A^{(k)}$	The k-hop adjacency matrix of A
$\tilde{A}^{(k)}$	The complement adjacency of $A^{(k)}$
Z	The output of graph encoder
Z_m	The optimized output by masks of graph encoder
$\hat{Z}$	The output of graph encoder in enhanced predictive learning
P_r	Relational neighbor set
P_n	Negative neighbor set
θ_e	The learnable parameters in graph encoder
θ_m	The learnable parameters in mask generator
M_f	The feature mask from SES
M_s	The structure mask from SES
$\hat{M}_s$	The transferred structure mask from SES
E_{feat}	The feature explanation based on M_f
E_{sub}	The substructure explanation based on M_s
Y	The label set of nodes
$\mathcal{L}_{xent}$	The cross-entropy loss
$\mathcal{L}_{sub}$	The subgraph loss
$\mathcal{L}_{triplet}$	The triplet loss

III. PROBLEM DEFINITION

A graph is denoted by $G = (V, A, X)$, where $V = \{v_1, \cdots, v_N\}$ denotes the node set with N nodes, $A \in \mathcal{R}^{N\times N}$ denotes the adjacency matrix of G, $a_{ij} \in A$ and $a_{ij} = 1$ if v_i and v_j are connected. $X \in \mathcal{R}^{N\times F}$ represents the node features with F dimensions.

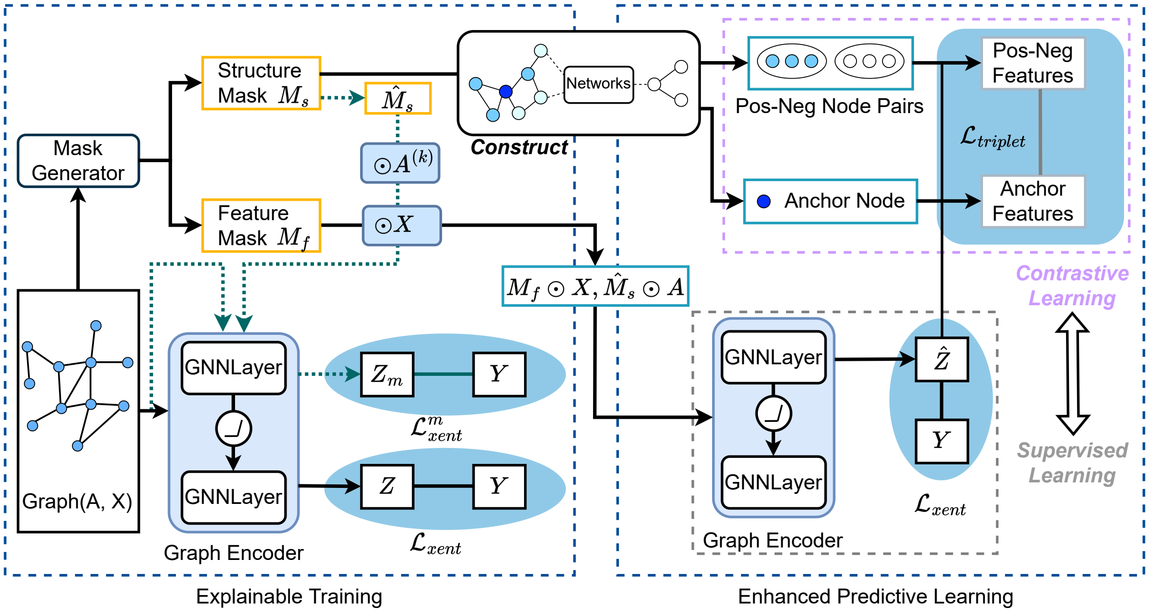


Fig. 2. The framework of SES involves two phases: explainable training and enhanced predictive learning. The parameters of the graph encoder are shared in two phases. The green dashed arrows marked the process where the mask generator and the graph data are fed into the graph encoder.

For predicting the node's label task (i.e., node classification task), the label set of nodes is denoted as Y_L. The labeled and unlabeled node sets from V can be written as $V_L = \{v_{l1}, \cdots, v_{ln}\}$ and $V_U = \{v_{u1}, \cdots, v_{un}\}$. The objective of the node classification task is to utilize the graph G and the set of labeled nodes V_L to train the GNN, enabling the prediction of labels for the unlabeled nodes in V_U.

For the instance-level explanation, a model offers explanations for node features or substructures. A feature explanation in SES represented as $E_{feat} \in \mathcal{R}^{N\times F}$ is provided to clarify the significance of each feature dimension in predicting a node's label. Additionally, a substructure explanation denoted as $E_{sub} \in \mathcal{R}^{N\times N}$ in SES is applied to highlight the importance of a node's local neighbors.

Following GNNExplainer [26], the problem of describing a self-explainable and self-supervised GNN can be written as:

Given a general graph $G(V, A, X)$, with labeled and unlabeled node set V_L and V_U, we learn a self-explainable and discriminative self-supervised GNN f: $v_{ui} \to y$, which provide feature explanation E_{feat} and substructure explanation E_{sub} as the instance-level explanation for each $v_{ui} \in V_U$.

To facilitate comprehension, symbols used in this paper and their definitions are summarized in TABLE II.

IV. Proposed Method

The framework of SES is depicted in Fig. 2. SES consists of two primary phases: explainable training and enhanced predictive learning. SES employs a mask generator co-trained with a graph encoder in the explainable training phase. The mask generator generates feature masks for nodes and structure masks for the subgraphs and is optimized during training. In the enhanced predictive learning phase, SES constructs positive-negative node pairs based on the structure masks. These node pairs are then utilized to optimize the node features, which are subsequently processed by a shared graph encoder. By contrastive learning, SES designs a triplet loss to supervise the representation learning of nodes. Supervised learning is also employed to bolster the contrastive learning. The detailed procedures of SES is delineated in Algorithm 2.

A. Graph Encoder and Mask Generator

1) Graph Encoder: We utilize a GNN backbone as a graph encoder to generate node embeddings. The GNN backbone is flexible and can be GCN [17] and GAT [18], etc. A general process of one convolution layer $Conv$ in GNN based on message passing [46] for a node v is described as follows:

$$h_v^{(l)} = COB^{(l)}\{AGG^{(l)}(h_u^{(l-1)} : u \in \mathcal{N}(v)),\ h_v^{(l-1)}\}, \quad (1)$$

where $h_v^{(l)}$ is the feature vector of v in the $l^{(th)}$ layer and $\mathcal{N}(v)$ is the neighbors of v. $h_v^{(l-1)}$ represents the feature vector of v in the $(l-1)^{(th)}$ layer. AGG is the function to aggregate features of nodes with their neighbors. COB is the combination function to update the representation of nodes.

The process of a graph encoder with two convolution layers in Eq. (1) is summarized as follows:

$$Z = Conv_2(\sigma(Conv_1(A, X)), A), \quad (2)$$

where σ is the activation function, and Z is the output of the graph encoder used for node classification. $H \in \mathcal{R}^{N\times F_{fid}} = Conv_1(A, X))$ is the output from the first convolution layer $Conv_1$ employed for feature mask generation. F_{hid} denotes the hidden size of $Conv_1$. $X = \{h_v^0, v \in V\}$, $H = \{h_v^1, v \in V\}$, $Z = \{h_v^2, v \in V\}$.

2) Mask Generator: The masks are weight matrices to provide explanations of features and structures, emphasizing crucial feature dimensions and neighboring nodes. The framework of the global mask generator is depicted in Fig. 3.

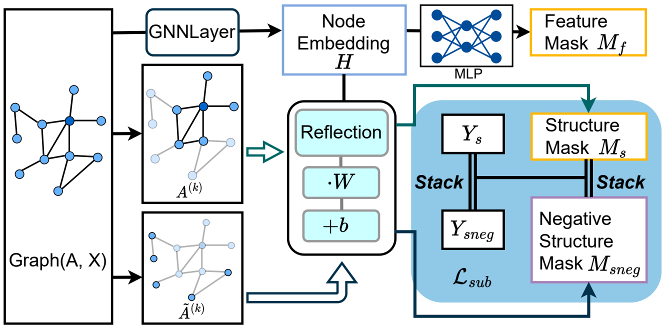


Fig. 3. The framework of the mask generator in SES.

H is passed through a multi-layer perception to generate a node feature mask denoted as M_f:

$$M_f = MLP(H), \tag{3}$$

To generate the structure mask M_s, a k-hop adjacency matrix denoted as $A^{(k)}$ is constructed. This matrix captures the neighbor relations of central nodes within their subgraphs. The structure mask $M_s \in \mathcal{R}^{N_k \times 1}$ is a composite matrix that combines output features of neighboring nodes and central nodes, N_k is the number of edges in $A^{(k)}$.

To enhance the learning of the substructure, we conduct negative sampling on nodes that are not part of the subgraph of the central node and with different labels. The negative structure mask based on negative samples are represented as $M_{sneg} \in \mathcal{R}^{N_k \times 1}$, which is aligned with M_s to compute the substructure loss in Eq. (7). M_s and M_{sneg} are constructed by the following formulations:

$$\begin{aligned}
&P_r(v_i) = \{j \mid a_{ij} \neq 0\}, \quad a_{ij} \in A^{(k)}, \\
&M_s = \sigma[W \cdot (\underset{i=1}{\overset{N}{stk}} \underset{k \in P_r(v_i)}{cat} (h_i, h_k)) + b], \\
&M_{sneg} = \sigma[W \cdot (\underset{i=1}{\overset{N}{stk}} \underset{k \in P_n(v_i)}{cat} (h_i, h_k)) + b],
\end{aligned} \tag{4}$$

where $h_i = h_i^1$ is the output feature vector of node i's from the first convolution layer $Conv_1$, and $h_k = h_k^1$ is the output feature vector of node k but from $P_r(v_i)$ or $P_n(v_i)$. $P_r(v_i)$ represents the neighbor relational set of node v_i, while $P_n(v_i)$ denotes the negative set of node v_i. $P_n(v_i)$ is obtained by randomly selecting an equal number of v_i's k-hop neighbors from the complement of the adjacency matrix $\tilde{A}^{(k)}$, where $\tilde{A}^{(k)} = I - A^{(k)}$. The function cat performs concatenation operations on the h_i and h_k, stk is the column operation for stacking the result of h_i and h_k's concatenation. The concatenation function performs a concatenation operation along the first dimension of h_i and h_k, while stacking is used to stack the concatenated results of h_i and h_k column-wise. Additionally, W and b represent the shared learnable weight and bias applied to M_s and M_{sneg}, respectively. σ is the activation function (sigmoid here). Inspired by the general approach for link predictions, we aim to make the node features within the neighborhood more similar and distinguish them from the features of nodes outside the neighborhood by a loss function. So M_s and M_{sneg} are constructed and both used in calculating the objective function.

To align the weights inside M_s with $A^{(k)}$. A matrix $Idx \in \mathcal{R}^{2 \times N_k}$ containing edge indices of $A^{(k)}$ is utilized to transfer the shape of M_s. $\hat{M}_s \in \mathcal{R}^{N \times N}$ is the transferred mask of structure, $s_{ij} \in \hat{M}_s$ is computed as:

$$s_{ij} = \begin{cases} M_{sk} & \text{if} \quad Idx_{1,k} = i \quad \text{and} \quad Idx_{2,k} = j \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

B. Explanation with Masks

After explainable training, a matrix $M_f \in \mathcal{R}^{N \times F}$ is obtained from a well-trained mask generator and contains important weights for node features. By applying M_f to the feature matrix X, we derive an explanation matrix $E_{feat} = M_f \odot X$ for the nonzero features of the node to provide feature explanations, where $\odot$ is Hadamard product represents the multiplication of the corresponding elements.

Similarly, the subgraph explanation E_{sub}, computed by $E_{sub} = \hat{M}_s \odot A^{(k)}$, provides important weights for the central nodes' k-hop neighbors. By covering these weights over the edges, SES provides explanations of the central node's neighbors in the subgraph.

C. Construction of Positive-Negative Pairs

To enhance the improvement of GNN's training, we extend supervisory information from the generated masks. Specifically, M_s is used to construct positive and negative sample pairs for subsequent contrastive learning. This process involves a k-hop weight matrix, denoted as $\hat{A}^{(k)}$. For each node v_i in G, we sort and sample v_i's neighbors based on their weights in $\hat{A}^{(k)}$, forming the positive node set $S^p(v_i)$. In addition, we sample an equal number of nodes from $P_n(v_i)$ to construct the negative node set $S^n(v_i)$. The complete process is summarized in Algorithm 1, wherein **sorted** returns sorted neighbors of v_i according to their weights in $\hat{A}^{(k)}$, **len** returns the number of neighbors of v_i, and **random_sample** randomly selects num_sample nodes from $P_n(v_i)$ to serve as negative samples.

D. Learning Objective

1) Explainable Training: During the explainable training phase, we simultaneously train the graph encoder and the mask generator.

For accuracy optimization, we minimize the cross-entropy loss for the graph encoder during training. The cross-entropy loss $\mathcal{L}_{xent}$ is defined as follows:

$$\mathcal{L}_{xent} = -\sum_{l \in Y_L} \sum_{c=1}^{C} Y_{lc} \ln Z_{lc}, \tag{6}$$

where Y_L denotes the node labels, C denotes the class number of dataset, Z is the output of graph encoder.

Algorithm 1: Construction of Positive-Negative Pairs.

```
Input: Structure mask $\hat{M}_s$, k-hop adjacency $A^{(k)}$,
       negative node set $P_n$, sample ratio $r$, number
       of nodes $N$.
Output: The positive and negative node sets.
Compute the k-hop adjacency weight matrix
  $\hat{A}^{(k)} = \hat{M}_s \cdot A^{(k)}$ ;
for $i$ to $N$ do
    neighs_i = sorted ($\hat{A}_i^{(k)}$);
    num_sample = r × len (neighs_i);
    $S^p(i)$ = neighs_i[0 : num_sample];
    $S^n(i)$ = random_sample ($P_n(v_i)$, num_sample);
end
return positive and negative node sets $S^p$ and $S^n$;
```

To ensure that the significant neighbors provided by M_s conform to the original structure of the neighborhood, we design a subgraph loss. The loss stacks M_s and M_{sneg} together and minimizes the distance from the corresponding stacked labels Y_s and Y_{sneg}. The formula for the subgraph loss $\mathcal{L}_{sub}$ is as follows:

$$\mathcal{L}_{sub} = \frac{1}{N_k} \sum_{i=1}^{N_k} \left| stk(M_s, M_{sneg}) - stk(Y_s, Y_{sneg}) \right|, \quad (7)$$

where Y_s and Y_{sneg} are neighboring nodes' labels and negative sample labels, respectively.

For reliable interpretations, $\hat{M}_s$ and M_f are applied to features and adjacency to facilitate the training of the mask generator. The new output of the graph encoder is Z_m. Cross-entropy loss for Z_m denotes $\mathcal{L}_{xent}^m$ that optimizes the neighborhood weights and feature weights suitable for the graph encoder. The formulations are computed as follows:

$$\begin{aligned} Z_m &= GE(M_f \odot X, \hat{M}_s \odot A^{(k)}), \\ \mathcal{L}_{xent}^m &= - \sum_{l \in Y_L} \sum_{c=1}^{C} Y_{lc} \log Z_{mc}, \end{aligned} \quad (8)$$

where GE denotes the graph encoder refers to Eq. (2).

To sum up, for the explainable training, the loss is:

$$\min_{\theta_m, \theta_e} [\alpha(\mathcal{L}_{sub} + \mathcal{L}_{xent}^m) + (1-\alpha)\mathcal{L}_{xent}], \quad (9)$$

where α is the hyperparameter to balance the optimization weights of the graph encoder and mask generator. θ_m and θ_e represent the learnable parameters of the mask generator and graph encoder, respectively.

2) Enhanced Predictive Learning: In the enhanced predictive learning phase, the structure mask $\hat{M}_s$ and feature mask M_f generated by the trained mask generator are applied to the adjacency matrix A and the node features X, respectively. This process highlights the critical components of structure and features, enhancing the graph encoder's representation ability. The output of GE is denoted as $\hat{Z}$ and is computed as follows:

$$\hat{Z} = GE(M_f \odot X, \hat{M}_s \odot A), \quad (10)$$

To learn essential neighboring features from positive and negative sample pairs to enhance the node representation, a triplet loss $\mathcal{L}_{triplet}$ is designed. For each node v_i, we consider it as an anchor node and obtain the features of the positive, negative, and anchor samples mapped by $\hat{Z}$:

$$p_i = \overset{\mathcal{N}_i}{\underset{j=1}{stk}} \hat{Z}_{S^p(v_i)_j}, \quad n_i = \overset{\mathcal{N}_i}{\underset{j=1}{stk}} \hat{Z}_{S^n(v_i)_j}, \quad a_i = \overset{\mathcal{N}_i}{\underset{j=1}{stk}} \hat{Z}_i, \quad (11)$$

where $\mathcal{N}_i$ denotes the set number of node v_i in S^p and S^n.

The $\mathcal{L}_{triplet}$ is computed as follows:

$$\mathcal{L}_{triplet} = \frac{1}{N} \sum_{i=1}^{N} \{\max(\|a_i - p_i\|_2 - \|a_i - n_i\|_2 + m, 0)\}, \quad (12)$$

where m denotes the margin parameter in the triplet loss.

For enhanced predictive learning, the learning objective of SES is:

$$\min_{\theta_e} [\beta \mathcal{L}_{triplet} + (1-\beta)\mathcal{L}_{xent}], \quad (13)$$

where β is the hyperparameter.

Algorithm 2: Framework of SES.

```
Input: A graph $G$, hyperparameters $\alpha$ and $\beta$.
Output: The feature explanation $E_{feat}$ and
        substructure explanation $E_{sub}$. The nodes'
        representation $\hat{Z}$.
Initialize the parameters $\theta_e$, $\theta_m$ of graph encoder and
  mask generator using Xavier initialization [47];
while epoch in explainable training do
    Obtain the node representation $Z$ by the graph
      encoder (Eq. (2));
    Calculate the feature and structure mask $M_f$, $M_s$,
      and transferred structure mask $\hat{M}_s$ by the mask
      generator (Eqs. (3, 4, 5));
    Update $\theta_e$, $\theta_m$ by the loss in explainable training
      (Eq. (8, 9));
end
Construct positive-negative node set $S^p$, $S^n$;
while epoch in enhanced predictive learning do
    Obtain the node representation $\hat{Z}$ by the graph
      encoder (Eq. (10));
    Calculate the positive-negative features by $\hat{Z}$ (Eq.
      (11));
    Compute the triplet loss (Eq. (12));
    Update $\theta_e$ by the loss in enhanced predictive
      learning (Eq. (13));
end
Return the output $\hat{Z}$, feature explanation $E_{feat}$ by $M_f$,
  substructure explanation $E_{sub}$ by $\hat{M}_s$;
```

E. Time Complexity

During the training process, the time consumption of the explainable training phase in SES contains the following main components: the backbone GNN, the computation of the mask

TABLE III
PREDICTION ACCURACY (%) ON NODE CLASSIFICATION.

Methods	GCN	GAT	UniMP	FusedGAT	ASDGN	SEGNN	ProtGNN	SES (GCN)	SES (GAT)	Imp.
Cora	86.83±2.98	86.81±1.36	88.18±1.52	80.26±1.47	83.28±0.82	84.35±0.33	81.98±2.13	**90.64±0.65**	90.39±0.56	2.46
CiteSeer	75.50±0.50	72.22±0.85	75.33±1.34	74.22±1.16	75.20±0.37	76.10±0.74	73.42±1.68	78.51±0.86	**78.69±0.84**	2.59
PolBlogs	93.86±5.28	94.72±1.15	95.45±0.91	94.63±0.99	80.45±1.89	–	88.77±10.98	**97.90±0.55**	97.86± 0.47	2.45
CS	90.08±0.11	91.72±0.44	93.65±0.34	91.35±0.05	93.70±0.12	–	84.30±1.52	**94.54±0.45**	94.10±0.31	0.84

generator, and Algorithm 1. In the case of backbone GCN [17], the model complexity is $O(|E| \times F \times F_{hid})$. The Algorithm 1 has a time complexity of $O(Nlog(N))$. For the feature mask M_f in the mask generator, its time complexity is $O(F \times F_{hid})$. For the structure mask M_s and the negative mask M_{sneg} in the mask generator both have time complexities of $O(|V| \times N_k \times F_{hid})$. Similarly, the time complexity for the enhanced predictive learning phase is $O(|E| \times F \times F_{hid}) + O(F_{hid} \times \sum_{i=1}^{|V|} \mathcal{N}_i^2)$. Assuming sparsity in the graphs, we can simplify the complexities by considering $|E| = |V| \times \bar{K}_1/2$, $N_k = |V| \times \bar{K}_2/2$, where $\bar{K}_1$ and $\bar{K}_2$ is the average degree of nodes in V and $A^{(k)}$ respectively. After discarding lower-order terms, the final time complexity for SES is $O(|V|^2 \times \bar{K}_1 \times \bar{K}_2 \times F)$.

The time complexity of SEGNN [33] can be expressed as $O(F \times \sum_{v_t \in \mathcal{V}_t} \sum_{v_l \in \mathcal{S}_t} |E_t| \cdot |E_l|)$, where $\mathcal{V}_t$ is the set of target nodes, $\mathcal{S}_t$ denotes the sampled positive and negative nodes, and E_l are the edges of the union set of $\mathcal{V}_t$ and $\mathcal{S}_t$. The number of nodes in the union set are comparable to $|V|$. The time complexity of SEGNN is $O(|V|^3 \times \bar{K}_3 \times \bar{K}_4 \times F)$, where $\bar{K}_3$ is the average degree of nodes of the union set, and $\bar{K}_4$ is the average number of the sample nodes. Neglecting the constants, the time complexity of SEGNN is higher than SES.

V. EXPERIMENTAL RESULTS

A. Datasets Description

1) Real-World Datasets: Three classic datasets are considered: (1) **Cora** [48] is a citation network with 2,708 nodes and 10,556 edges. Nodes represent documents, and edges represent citation links. Each paper is represented by a 1433-dimensional word vector and labeled into one of seven machine-learning topics. (2) **CiteSeer** [48] is also a citation network with 3,327 nodes and 9,104 edges; it has six classes and the data structure is the same as Cora. (3) **PolBlogs** [49] is a graph with 1,490 nodes and 19,025 edges. A node represents political blogs, and edges represent links between blogs. Each node has a label that indicates its political inclination: liberal or conservative. (4) **CS**: The Coauthor CS network from [50] with 18,333 nodes and 163,788 edges. Nodes represent authors that are connected by an edge if they co-authored a paper.

2) Synthetic Datasets: Following previous works [26], [27] to construct four synthetic datasets to validate the performance of GNNs on explainability tasks: (1) **BAShapes** contains a Barabasi-Albert graph with 300 nodes and a set of 80 “house”-structured graphs connected to it. The nodes are classified into four categories based on their structural roles. (2) **BACommunity** dataset is a union of BAShapes with two “house”-structure graphs. Nodes have normally distributed features and are assigned to one of eight classes based on their structural roles and community membership. (3) **Tree-Grid** contains nodes of the tree structure and grid motifs. The categories of nodes are the grid motif or tree structure they belong to. (4) **Tree-Cycle** is similar to the “Tree-Grid”, consisting of the tree structure and cycle motifs.

B. Baselines

We consider the following strong baselines and verify whether the SES can improve the performance of backbone GNNs and achieve SOTA performance on the node classification tasks. We only report results of SES with GCN and GAT as backbone GNNs following ProtGNN [34] and report the best performance of ProtGNN by GCN and GAT as backbone GNNs. PxGNN [35] has neither been published nor has public codes, making it unsuitable as a baseline.

- GCN [17] is a typical graph neural network that integrates nodes’ information from their neighbors.
- GAT [18] fuses the attention mechanism based on GCN and enhances the ability of node representation.
- UniMP [22] is a novel unified message-passing model incorporating feature and label propagation at training and inference time.
- SEGNN [33] can find K-nearest labeled nodes for each unlabeled node to give explainable node classification.
- FusedGAT [23] is an optimized version of GAT that fuses message-passing computation for accelerated execution and lower memory footprint.
- ProtGNN [34] combines prototype learning with GNNs and provides a new perspective on the explanations of GNNs.
- ASDGN [24] is a framework for stable and non-dissipative DGN design conceived through the lens of ordinary differential equations.

For the explainability tasks of GNNs, we consider the following strong baselines. Note that the SEGNN and ProtGNN are unsuitable for the feature explanation tasks.

- GRAD [26] is a gradient-based method that computes the gradient of the GNN’s loss function for the adjacency matrix and the associated node features.
- ATT [18] is a graph attention network that is used to provide explanations.
- GNNExplainer [26] is a GNN explanation method that maximizes the mutual information between a GNN’s prediction and distribution of possible subgraph structures to provide consistent and concise explanations for an entire class of instances.

- PGExplainer [27] employs deep neural networks to parameterize the interpretation generation process, which makes it a natural way to explain multiple instances collectively.
- PGMExplainer [28] utilizes a generation probability model for graph data. This approach enables the model to learn concise underlying structures from observed graph data.
- GraphLIME [29] is a model-agnostic, local, and nonlinear explanation method for GNN for node classification tasks motivated from LIME [51], which uses Hilbert-Schmit Independence Criterion (HSIC) Lasso, a nonlinear interpretable model.

C. Experimental Settings

For the prediction task on node classification, we randomly divide the datasets as 60% training set, 20% validation set, and 20% test set [52]. For the explanation task, the synthetic datasets are divided into an 80% training set, 10% validation set, and 10% test set corresponding to the settings of [26]. For SES and baselines, the learning rate with the Adam optimizer is set to 0.003. The hidden layer size is set to 128. The sample ratio in Algorithm 1 is set to 0.8 and the margin m in Eq. (12) is set to 1.0.

D. Node Classification

We evaluated the performance of SES and baselines on real-world datasets for node classification. The experimental results are summarized in TABLE III. The second highest performance is highlighted with underline. The Imp. represents improvements by SES compared to the best baseline.

SES (GCN) and SES (GAT) denote SES using GCN and GAT as backbone GNNs, respectively. The time consumption of SES(GCN) on Cora, CiteSeer, PolBlogs, and CS are 10.5s, 12.3s, 13.1s, and 89.7s, respectively. The time consumption of SES(GAT) on Cora, CiteSeer, PolBlogs, and CS are 10.7s, 12.4s, 13.3s, and 92.2s, respectively. Notice that PolBlogs lacks node features. We assign a unit matrix as the node features, ensuring each node has an associated feature representation. SEGNN is not suitable for PolBlogs and CS. The SES framework achieved the SOTA performance and demonstrated superiority on all real-world datasets. SES outperformed the second-best method by a significant margin. SES has improvements of 2.46%, 2.59%, 2.45%, and 0.84% in absolute accuracy over the second-best method on Cora, CiteSeer, PolBlogs, and CS, respectively. While current self-explainable methods SEGNN and ProtGNN perform poorly in the tasks, not as well as the backbone GNNs in most cases.

E. Explanation Qualification

We performed experiments on widely applied synthetic datasets to compare SES with other GNN explanation methods. Following the experimental settings in GNNExplainer [26], we employed ground-truth explanations available for the synthetic datasets to quantify the accuracy of different explanation methods. The AUC scores for explanation tasks are summarized in TABLE IV.

SES performs superior on the BAShape and Tree-grid datasets over all other methods, demonstrating significantly enhanced interpretability. SES showcases the least relative improvement of 2.5% on the BAShape dataset and 3.0% on the Tree-grid dataset. SES also performs outstandingly on the Tree-Cycle dataset, achieving an AUC score close to 100%. On the BACommunity dataset, both SES and PGExplainer achieved comparable performance. SEGNN performs well on the BAShapes while performing much less on the other three datasets. SES showcases its effectiveness in providing accurate and reliable explanations of synthetic datasets.

TABLE IV
EXPLANATION ACCURACY (%) ON SYNTHETIC DATASETS.

Dataset	BAShapes	BACommunity	Tree-Cycle	Tree-Grid
GRAD	88.2	75.0	90.5	61.2
ATT	81.5	73.9	82.4	66.7
GNNExplainer	92.5	83.6	94.8	87.5
PGExplainer	96.3	**94.5**	98.7	90.7
PGMExplainer	96.5	92.6	96.8	89.2
SEGNN	97.3	77.2	62.3	50.5
SES	**99.8** (2.5↑)	**94.5**	**99.4** (0.7↑)	**93.7** (3.0↑)

To evaluate the feature explanation quality on real-world datasets, we consider SES and two classical methods, GNNExplainer [26] and GraphLIME [29], which are suitable for the node feature explanation. However, PGExplainer and PGMExplainer, target for edge and node explanations, respectively, lack the capability to generate the weight set for the features, rendering them unsuitable for the present task. Fidelity+ [53], a widely used metric, is employed. Fidelity+ measures the dissimilarity in accuracy or predicted probability between the original predictions and the predictions obtained after masking out crucial input features. Mathematically, it is expressed as follows:

$$Fidelity+^{acc} = \frac{1}{N}\sum_{i=1}^{N}(\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i^{1-m_i} = y_i)), \quad (14)$$

where y_i represents the original prediction, and $1-m_i$ denotes the complementary mask that eliminates the important input features. The indicator function $\mathbb{1}(\hat{y}_i = y_i)$ evaluates to 1 when y_i and $\hat{y}_i$ are equal, and 0 otherwise.

Due to the sparsity of the citation network, we remove the top-5 important features of each node based on the weights assigned by explainers. We conduct experiments by 2-layer GCN and GAT with identical parameters. The Fidelity+ scores are presented in TABLE V. The importance weights assigned by GraphLIME do not significantly influence node classification's performance, as the accuracy only experiences a minor decline after removing the important features.

SES achieves the highest Fidelity+ scores on Cora, CiteSeer, Polblogs, and CS datasets. The Fidelity+ of SES (GCN) is about four times of GNNExplainer (GCN) on CiteSeer. When GAT is used as the backbone GNN, SES (GAT) and

TABLE V
FIDELITY+ (%) OF MODELS ON REAL-WORLD DATASETS.

Dataset	Cora	CiteSeer	PolBlogs	CS
GNNExplainer (GCN)	8.3	4.3	40.5	0.17
GraphLIME (GCN)	1.6	1.7	2.0	0.09
SES (GCN)$-\{\mathcal{L}^{m}_{xent}\}$	5.27	1.79	48.53	0.6
SES (GCN)	**14.7**	**16.1**	**49.3**	**2.77**
Imp.	6.4	11.8	8.8	2.17
GNNExplainer (GAT)	15.4	9.4	**44.8**	0.15
GraphLIME (GAT)	1.2	1.0	2.8	0.12
SES (GAT)$-\{\mathcal{L}^{m}_{xent}\}$	1.30	2.17	39.13	0.3
SES (GAT)	**17.2**	**11.0**	44.6	**2.96**
Imp.	1.8	1.6	-0.2	2.66

GNNExplainer (GAT) represent comparable performance on the PolBlogs. The gaps in performance are also reduced on Cora and CiteSeer.

The post-hoc methods GNNExplainer and GraphLIME utilize a separate training process to generate explanations, which may result in disjointed explanations. The feature and structure masks in SES directly work on the feature and adjacency matrices, respectively, and they are co-trained with the backbone GNN by Eq. (8). It ensures feature and structure masks are consistent with the backbone GNN's aggregation and decision process. To validate the reason, we removed $\mathcal{L}^{m}_{xent}$ from Eq. (8), which is denoted as $-\{\mathcal{L}^{m}_{xent}\}$. The results demonstrate a significant performance decay for SES after eliminating the impact of the consistency, which indicates the effectiveness of the proposed mask generator.

F. Time Consumption

We evaluate and compare the time consumption post-hoc explainers and self-explainable GNNs to generate explanations on Cora. Traditionally speaking in machine learning, inference time is the duration time from the input to a well-trained model to its producing outputs, which are milliseconds and is not particularly meaningful for comparing GNNs. we define inference time for generating explanations as the period required from a well-trained backbone GNN to produce explanations for all nodes following the way of PGExplainer [27]. For GNNExplainer and GraphLIME, despite backbone GNNs have been trained, they necessitate re-training for each individual node [26], [29], so the inference time counts on the re-training duration. For SES and SEGNN, the inference time incorporates the training time since they train backbone GNN models while giving explanations for the same process. This allows for a fair comparison of the time consumption between post-hoc methods and self-explainable GNNs. However, ProtGNN cannot construct explainable subgraphs for node classification tasks, making it inapplicable for this specific task. The explainable training of SES spans 300 epochs with an additional 15 epochs dedicated to enhanced predictive learning. The training epochs for other explainers are aligned with SES. The experiments were on an NVIDIA RTX 3090 GPU, with GCN as the backbone GNN, and are presented in TABLE VI.

TABLE VI
INFERENCE TIME OF GENERATING EXPLANATIONS FOR ALL NODES ON THE CORA DATASET.

GNNExplainer	GraphLIME	PGExplainer	SEGNN	SES (et)
9 min 50s	4 min 24s	1 min 13s	1 min 32s	4.3s

TABLE VII
TRAINING AND INFERENCE TIME OF SES(GCN).

Dataset	Cora	CiteSeer	PolBlogs	CS
Inference time	4.3s	4.4s	9.1s	34.0s
Training time	10.8s	12.3s	13.1s	89.7s

TABLE VI showcases that SES outpaces SEGNN and other post-hoc explainers in terms of speed. The mask of SES is co-trained with the graph encoder and subsequently provides explanations once the explainable training (SES (et)) is completed, clocking in at a swift 4.3 seconds. The enhanced predictive learning phase of SES (SES (epl)) that involves conducting contrastive learning for each node takes 6.5 seconds. The total training time for the whole SES is 10.8 seconds. Note that SES (epl) does not affect the explainability of SES but refines its prediction accuracy. The training and inference times for SES (GCN) across real-world datasets, as detailed in Table VII, exhibit an increase corresponding to the growing graph sizes and densities.

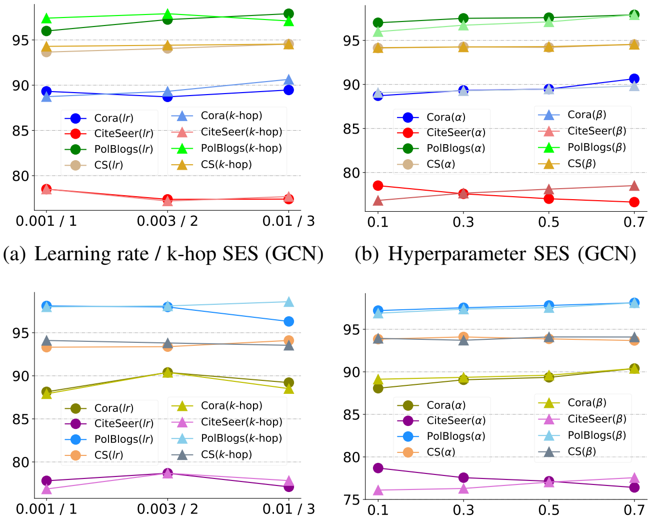
The time complexity of Algorithm 1 is primarily dictated by the sorting and random sampling operations that have $O(NlogN)$ and $O(N)$ complexity, respectively, which only contributes a minor fraction of overall time consumption in SES. We calculate the time consumption of the Algorithm 1 by synthesizing a sparse graph with a fixed number of nodes and twice as many edges, which is reported in TABLE VIII.

In the trade-off analysis, post-hoc methods exhibit competitive explanation accuracy compared to current self-explainable models but fall short or underperform in predicting node labels. Self-explainable methods SEGNN and SES offer lower inference time in comparison to GNNExplainer and GraphLIME that necessitate retraining for each node's explanation generation. SES demonstrates that high prediction accuracy and explanation quality can be achieved simultaneously. With the application of appropriate techniques, time efficiency need not be compromised. But SES and SEGNN come with the trade-off of higher memory demands which will be optimized in future work. In contrast, GNNExplainer and GraphLIME require smaller memory by interpretating on individual instances.

TABLE VIII
TIME CONSUMPTION OF CONSTRUCTING POSITIVE-NEGATIVE NODE PAIRS.

Nodes	0.1k	1k	10k	50k	70k
Time consumption	0.005s	0.045s	2.11s	28.92s	38.53s

G. Parameter Sensitivity



(a) Learning rate / k-hop SES (GCN)

(b) Hyperparameter SES (GCN)

(c) Learning rate / k-hop SES (GAT)

(d) Hyperparameter SES (GAT)

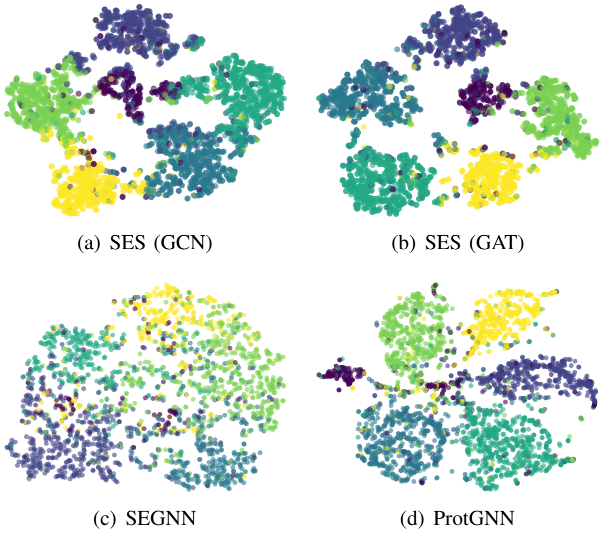
Fig. 4. Parameter sensitivities of SES.

The parameter sensitivity analyses are presented in Fig. 4, the learning rate (lr), k (k-hop), and hyperparameters α and β exhibit varying degrees of influence on SES's performance. In Fig. 4(a), SES (GCN)'s accuracy on CiteSeer improves as the learning rate and k decrease. In the case of a larger neighborhood in Cora, SES (GCN) performs well owing to the wider range of papers cited from different fields. Additionally, the effect on the PolBlogs increases with the learning rate. Fig. 4(b) indicates that assigning higher weights to α and β enhances the performance of SES (GCN) on Cora and PolBlogs. It suggests that mask training and self-supervised learning contribute to characterizing GNNs. However, a lower α weight is required for CiteSeer.

From Fig. 4(c), SES (GAT) consistently achieves the best performance on citation (Cora and CiteSeer) datasets when the learning rate is 0.003. On the PolBlogs, the value k that leads to a larger neighborhood of a node and a small learning rate benefit the performance. Fig. 4(d) shows that the performance of the two hyperparameters exhibits a slow increase trend on Cora and PolBlogs. In the case of CiteSeer, SES (GAT) relies more on triplet loss to achieve optimal performance. However, the performance of SES is stable and notable in most cases. On the CS dataset, the performance of SES (GCN) and SES (GAT) is not sensitive to hyperparameters. Overall, we observe that higher learning rates and smaller neighborhood ranges lead to improved accuracy in SES.

H. Visualization

We visualized the learned node representations of GNNs to validate their representation ability on the CiteSeer dataset. Two self-explainable GNN models (SEGNN and ProtGNN) are applied. The output vectors of these GNNs are originally 128-dimensional and transformed into two dimensions by TSNE [56]. Fig. 5 depicts the visualizations of node representations. We observe that SES (GCN) and SES (GAT) provide more densely connected clusters than baselines. To qualify the clustering effectiveness of different methods, classical clustering evaluation indicators Silhouette score [54] and Calinski Harabasz score [55] are employed, with higher values indicating better outcomes. TABLE IX presents the assessment of cluster effects following the visualizations of Fig. 5. From TABLE IX, SES exhibits tighter clusters among nodes belonging to the same classes, resulting in higher scores.



(a) SES (GCN)

(b) SES (GAT)

(c) SEGNN

(d) ProtGNN

Fig. 5. Visualization of node representations after training on Citeceer.

TABLE IX
STATISTICAL METRICS FOR VISUALIZATION ON CITESEER.

Metric	Silhouette [54]	Calinski harabasz [55]
SES (GCN)	0.316	1694.75
SES (GAT)	**0.375**	**2131.56**
SEGNN	0.131	456.37
ProtGNN	0.277	1090.13

To comprehensively evaluate the effects of different explanation methods, we visualize the explanations of GNNExplainer, PGExplainer, PGMExplainer, and SES on four synthetic datasets [26]. These visualizations highlight the important subgraph structures of the datasets. In the visualizations, the color of the edges in the corresponding graphs represents the importance weight, with darker edges indicating higher importance. The visualizations are presented in Fig. 6. The explanations of SES on BAShape and BACommunity are evident that SES effectively identifies and matches important "house" structures in the datasets. For the Tree-Cycle and Tree-Grid datasets, SES also successfully recognizes the "tree", "circle", and "grid" nodes on the Tree-Cycle and Tree-Grid datasets. As shown in the figures, the interpretation results from baselines include subgraphs with unrelated structures.

I. Ablation Studies and Variants

Ablation studies and variants are conducted to investigate the contributions of components in SES. To verify the functionality of the feature mask on GNN, we remove the M_f on node features and denote it as $-\{M_f\}$. Similarly,

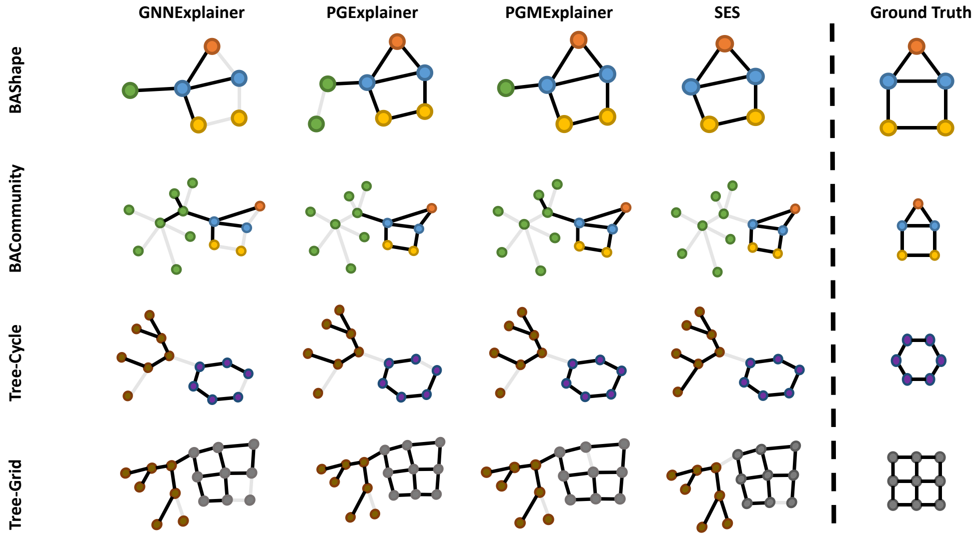


Fig. 6. Visualizations of subgraph explanations on synthetic datasets.

a general adjacency matrix rather than $\hat{M}_s$ on adjacency put into the graph encoder during the enhanced predictive learning phase is denoted as $-\{\hat{M}_s\}$. To verify the guiding effect of cross-entropy on triplet loss, we remove $\mathcal{L}_{xent}$ and denote it as $-\{\mathcal{L}_{xent}\}$ in the enhanced predictive learning phase. To verify the effectiveness of enhanced predictive learning, the triplet loss is removed, denoted as $-\{Triplet\}$. To validate the importance of explainable training, we replace the mask generator with the post-hoc GNNs (GNNExplainer (GEX), PGExplainer (PGE)) and denote them as $+\{epl\}$. The performance of ablation studies is reported in TABLE X. The performance of SES was significantly impacted after removing the contrast learning. It suggests that self-supervised learning is significant in the prediction performance of SES, and the interpretation results play a crucial role in providing feedback and enhancing the predictive learning of models. The cross-entropy also plays an important role in enhancing predictive learning since removing $\mathcal{L}_{xent}$ in the second phase in SES leads to a sharply decreased accuracy, especially for the CS dataset. The absence of M_f and $\hat{M}_s$ both results in a decline in performance. This indicates that interpreting node features and structure importance plays significant roles in SES. Integrating self-supervised learning and the mask generator that considers structural and feature explanations leads to the best performance in SES. The results of $+\{epl\}$ combined with masks using the post-hoc model are worse than SES and most variants of SES. This indicates that the explanations provided by SES are more reliable and better suited for downstream constrastive learning.

TABLE X
ABLATION STUDIES OF SES ON REAL-WORLD DATASETS.

Dataset	Cora	CiteSeer	PolBlogs	CS
SES (GCN)$-\{M_f\}$	90.05	77.29	97.41	93.78
SES (GCN)$-\{\hat{M}_s\}$	89.31	78.05	96.90	94.24
SES (GCN)$-\{\mathcal{L}_{xent}\}$	88.90	77.23	95.89	88.20
SES (GCN)$-\{Triplet\}$	88.31	76.80	95.42	93.26
GEX(GCN)$+\{epl\}$	75.51	74.73	92.16	92.72
PGE (GCN)$+\{epl\}$	87.48	75.64	95.05	93.08
SES (GCN)	**90.64**	**78.51**	**97.90**	**94.54**
SES (GAT)$-\{M_f\}$	89.56	77.29	96.98	92.86
SES (GAT)$-\{\hat{M}_s\}$	88.29	77.41	95.14	92.91
SES (GAT)$-\{\mathcal{L}_{xent}\}$	88.43	77.93	96.24	88.28
SES (GAT)$-\{Triplet\}$	87.81	76.76	94.81	91.60
GEX(GAT)$+\{epl\}$	83.61	71.69	95.63	91.86
PGE (GAT)$+\{epl\}$	87.66	72.60	95.06	92.45
SES (GAT)	**90.39**	**78.69**	**97.86**	**94.10**

J. Case Studies

For the real-world datasets, we visualize the subgraphs by potential neighbor scores produced from the $\hat{M}_s$ of SES and edge masks of baselines to demonstrate their rationality in explanations. Especially, SES ranks the important neighbors with the weights obtained by $\hat{M}_s$, and the other baselines (GNNExplainer, PGExplainer, and PGMExplainer) rank important neighbors by the weights of edge masks for the central node in the neighborhood. Fig. 8 draws the 2-hop subgraphs of node 78 in Cora, node 50 in CiteSeer, node 539 in PolBlogs, and node 212 in CS and lists the ranked node sequence of models.

In Fig. 8(a), SES and PGMExplainer put node 1418 as the most important neighbor of central node 78 since the nodes belong to the same class. In contrast, node 1418 is ranked with much lower importance in GNNExplainer and PGExplainer. In Fig. 8(b), the neighborhood of the central node contains multiple classes, which challenges ranking the important neighbors. SES ranks the nodes with the same class at the top of the sequence, while baseline methods fail to achieve it. In Fig. 8(c), SES assigns the lowest importance to the political pages of the green camp and produces more weight to nodes with the same class, while PGExplainer considers more important pages from the opposing party. In Fig. 8(d), SES and PGExplainer ranked nodes in the same classes as 212 in the top position, while GEX and PGM rated

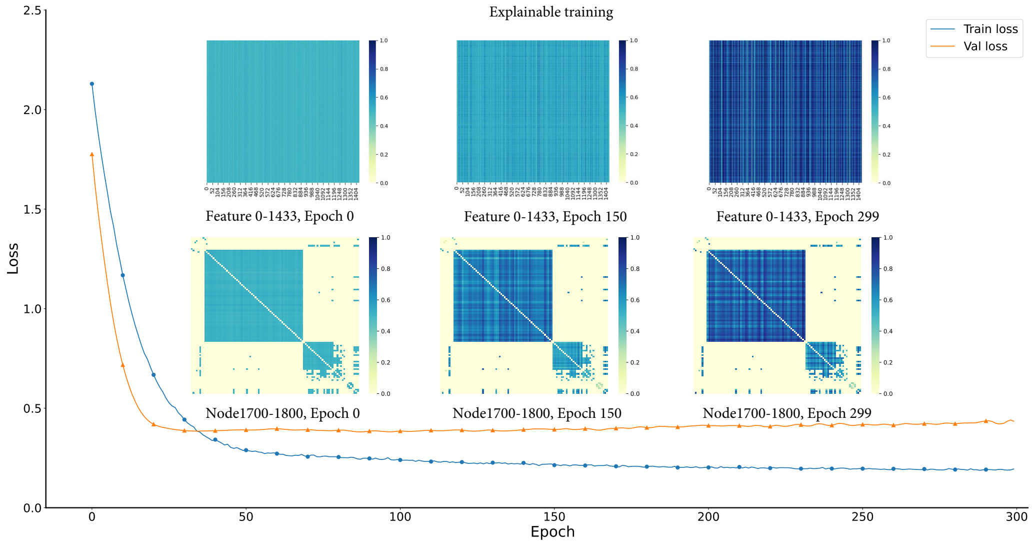
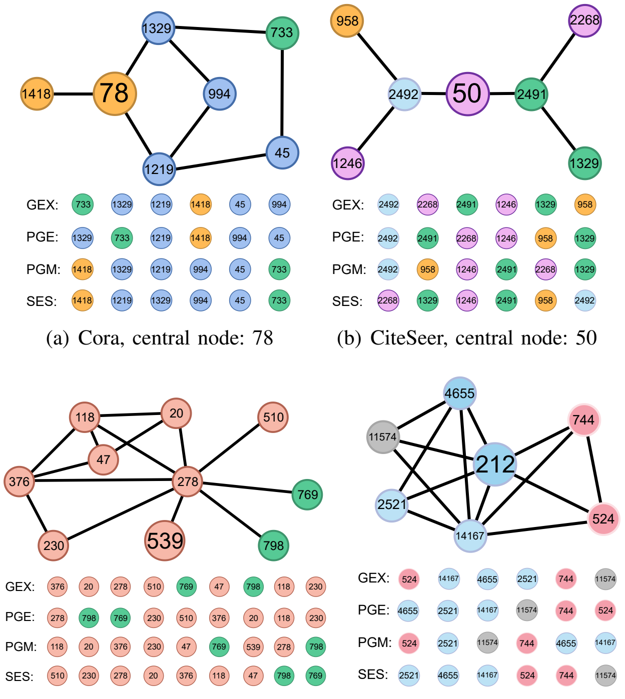


Fig. 7. Optimization of the feature and structural masks during the explainable training on the Cora dataset.

node 524 as more important. The cases demonstrated that SES provides more reasonable explanations than GNNExplainer, PGExplainer, and PGMExplainer.


78
1418
1329
994
1219
733
45
GEX: 733 1329 1219 1418 45 994
PGE: 1329 733 1219 1418 994 45
PGM: 1418 1329 1219 994 45 733
SES: 1418 1219 1329 994 45 733
50
958
2492
1246
2491
2268
1329
GEX: 2492 2268 2491 1246 1329 958
PGE: 2492 2491 2268 1246 958 1329
PGM: 2492 958 1246 2491 2268 1329
SES: 2268 1329 1246 2491 958 2492
539
118
20
47
376
278
510
769
230
798
GEX: 376 20 278 510 769 47 798 118 230
PGE: 278 798 769 230 510 376 20 118 230
PGM: 118 20 376 230 47 769 539 278 798
SES: 510 230 278 20 376 118 47 798 769
212
4655
11574
744
2521
14167
524
GEX: 524 14167 4655 2521 744 11574
PGE: 4655 2521 14167 11574 744 524
PGM: 524 2521 11574 744 4655 14167
SES: 2521 4655 14167 524 744 11574

(a) Cora, central node: 78

(b) CiteSeer, central node: 50

(c) PolBlogs, central node: 539

(d) CS, central node: 212

Fig. 8. Case study of subgraph explanations on real-world datasets. The rank of neighbors is based on the structure mask $\hat{M}_s$ of SES and edge masks of GNNExplainer (GEX), PGExplainer (PGE), and PGMExplainer (PGM). Different colored nodes have different labels.

K. Mask Optimization

We analyze the training and validation loss curves presented in Figure 7, as well as the evolution of the feature and structure masks during the training phase. The feature mask is presented for all dimensions of all nodes at epochs 0, 150, and 299. Moreover, we draw the structural mask of the 2-hop for nodes ranging from 1700 to 1800. In Figure 7, the initial state of the mask weights is depicted with a uniform color palette, reflecting their random starting points. As training proceeds, a clear transition in the mask weights is observed: they begin to diverge, exhibiting a pronounced contrast between darker and lighter shades. Notably, the darker weights tend to stabilize and remain consistent in the latter stages of training. This pattern of change provides visual evidence that the masks are finely optimized with the loss function.

VI. Conclusion

Previous self-explainable GNNs provide built-in explanations while suffering from a subpar performance in prediction. The feedback explanations are ineffectively applied to supervise the training phase in current GNNs. To address the challenges, we introduce a self-explainable and self-supervised graph neural network (SES) that bridges the explainability and prediction of GNNs by two-phase training. A global mask generator in SES is designed to generate reliable instance-level explanations until explainable training is finished, resulting in notable time savings. The parameters of the graph encoder are shared between two phases of SES. The explanations derived in the explainable training phase are utilized as supervisory information with a self-supervised objective loss during the enhanced predictive learning phase. Extensive experiments demonstrate that SES achieves SOTA explanation quality and significantly improves the prediction accuracy of current GNNs. Our work illustrates that the tasks of prediction and explainability can be concurrently enhanced during the training of GNNs.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 71971002).

References

[1] F. Zhou, C. Cao, K. Zhang, G. Trajcevski, T. Zhong, and J. Geng, "Meta-gnn: On few-shot node classification in graph meta-learning," in *CIKM*, 2019, pp. 2357–2360.

[2] A. Awasthi, A. K. Garov, M. Sharma, and M. Sinha, "Gnn model based on node classification forecasting in social network," in *AISC*. IEEE, 2023, pp. 1039–1043.

[3] Y. Zhao, H. Zhou, R. Xie, F. Zhuang, Q. Li, and J. Liu, "Incorporating global information in local attention for knowledge representation learning," in *IJCNLP*, 2021, pp. 1341–1351.

[4] Q. Zhong, L. Ding, J. Liu, B. Du, H. Jin, and D. Tao, "Knowledge graph augmented network towards multiview representation learning for aspect-based sentiment analysis," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–14, 2023.

[5] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *NeurIPS*, vol. 31, 2018.

[6] P. Hawkins, F. Maire, S. Denman, and M. Baktashmotlagh, "Modular construction planning using graph neural network heuristic search," in *AJCAI*. Springer, 2022, pp. 228–239.

[7] M. Li and Z. Zhu, "Spatial-temporal fusion graph neural networks for traffic flow forecasting," in *AAAI*, vol. 35, no. 5, 2021, pp. 4189–4196.

[8] L. Zhong, J. Tang, C. Xu, B. Ren, B. Du, and Z. Huang, "Traffic prediction of converged network for smart gird based on gnn and lstm," in *ICBAIE*. IEEE, 2022, pp. 341–348.

[9] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *KDD*, 2018, pp. 974–983.

[10] W. Wang, Z. Quan, S. Zhao, G. Sun, Y. Li, X. Ben, and J. Zhao, "User-context collaboration and tensor factorization for gnn-based social recommendation," *IEEE Transactions on Network Science and Engineering*, pp. 1–12, 2023.

[11] S. Yang, L. Xing, Y. Li, and Z. Chang, "Implicit sentiment analysis based on graph attention neural network," *Engineering Reports*, vol. 4, no. 1, p. e12452, 2022.

[12] Z. Jin, M. Tao, X. Zhao, and Y. Hu, "Social media sentiment analysis based on dependency graph and co-occurrence graph," *Cognitive Computation*, vol. 14, no. 3, pp. 1039–1054, 2022.

[13] Z. Chen, S. Li, B. Yang, Q. Li, and H. Liu, "Multi-scale spatial temporal graph convolutional network for skeleton-based action recognition," in *AAAI*, vol. 35, no. 2, 2021, pp. 1113–1122.

[14] N. Nejatishahidin, W. Hutchcroft, M. Narayana, I. Boyadzhiev, Y. Li, N. Khosravan, J. Košecká, and S. B. Kang, "Graph-covis: Gnn-based multi-view panorama global pose estimation," in *ICCV*, 2023, pp. 6458–6467.

[15] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *AAAI*, vol. 33, no. 01, 2019, pp. 7370–7377.

[16] K. Wang, S. C. Han, and J. Poon, "Induct-gcn: Inductive graph convolutional networks for text classification," in *ICPR*. IEEE, 2022, pp. 1243–1249.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[18] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[19] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, vol. 30, 2017, p. 1025–1035.

[20] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*, 2018.

[21] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Graph neural networks with convolutional arma filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3496–3507, 2021.

[22] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked label prediction: Unified message passing model for semi-supervised classification," in *IJCAI*, 2021, pp. 1548–1554.

[23] H. Zhang, Z. Yu, G. Dai, G. Huang, Y. Ding, Y. Xie, and Y. Wang, "Understanding gnn computational graph: A coordinated computation, io, and memory perspective," in *MLSys*, vol. 4, 2022, pp. 467–484.

[24] A. Gravina, D. Bacciu, and C. Gallicchio, "Anti-symmetric DGN: a stable architecture for deep graph networks," in *ICLR*, 2023.

[25] G. Shmueli, "To explain or to predict?" *Statistical Science*, vol. 25, no. 3, pp. 289–310, 2010.

[26] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *NeurIPS*, vol. 32, 2019, pp. 9244–9255.

[27] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," in *NeurIPS*, vol. 33, 2020, pp. 19 620–19 631.

[28] M. Vu and M. T. Thai, "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks," in *NeurIPS*, vol. 33, 2020, pp. 12 225–12 235.

[29] Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang, "Graphlime: Local interpretable model explanations for graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 7, pp. 6968–6972, 2022.

[30] H. Yuan, J. Tang, X. Hu, and S. Ji, "Xgnn: Towards model-level explanations of graph neural networks," in *KDD*, 2020, pp. 430–438.

[31] Y.-M. Shin, S.-W. Kim, E.-B. Yoon, and W.-Y. Shin, "Prototype-based explanations for graph neural networks (student abstract)," in *AAAI*, vol. 36, no. 11, 2022, pp. 13 047–13 048.

[32] X. Wang and H. W. Shen, "Gnninterpreter: A probabilistic generative model-level explanation for graph neural networks," in *ICLR*, 2023.

[33] E. Dai and S. Wang, "Towards self-explainable graph neural network," in *CIKM*, 2021, pp. 302–311.

[34] Z. Zhang, Q. Liu, H. Wang, C. Lu, and C. Lee, "Protgnn: Towards self-explaining graph neural networks," in *AAAI*, vol. 36, no. 8, 2022, pp. 9127–9135.

[35] E. Dai and S. Wang, "Towards prototype-based self-explainable graph neural network," *arXiv preprint arXiv:2210.01974*, 2022.

[36] B. M. Oloulade, J. Gao, J. Chen, T. Lyu, and R. Al-Sabri, "Graph neural architecture search: A survey," *Tsinghua Science and Technology*, vol. 27, no. 4, pp. 692–708, 2021.

[37] K. Li, Z. Huang, and Z. Jia, "RAHG: A role-aware hypergraph neural network for node classification in graphs," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 4, pp. 2098–2108, 2023.

[38] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5782–5799, 2022.

[39] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *Technologies*, vol. 9, no. 1, p. 2, 2020.

[40] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and S. Y. Philip, "Graph self-supervised learning: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 5879–5900, 2022.

[41] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *NeurIPS*, vol. 33, 2020, pp. 5812–5823.

[42] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," in *KDD*, 2020, pp. 1150–1160.

[43] M. Jin, Y. Zheng, Y.-F. Li, C. Gong, C. Zhou, and S. Pan, "Multi-scale contrastive siamese networks for self-supervised graph representation learning," in *IJCAI*, 2021, pp. 1477–1483.

[44] X. Wang, N. Liu, H. Han, and C. Shi, "Self-supervised heterogeneous graph neural network with co-contrastive learning," in *KDD*, 2021, pp. 1726–1736.

[45] Y. Wang, J. Wang, Z. Cao, and A. Barati Farimani, "Molecular contrastive learning of representations via graph neural networks," *Nature Machine Intelligence*, vol. 4, no. 3, pp. 279–287, 2022.

[46] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *ICML*. PMLR, 2017, p. 1263–1272.

[47] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[48] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *ICML*. PMLR, 2016, pp. 40–48.

[49] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog," in *LinkKDD*, 2005, pp. 36–43.

[50] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," in *Relational Representation Learning Workshop, NeurIPS 2018*, 2018.

[51] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *KDD*, 2016, pp. 1135–1144.

[52] K. Guo, K. Zhou, X. Hu, Y. Li, Y. Chang, and X. Wang, "Orthogonal graph neural networks," in *AAAI*, vol. 36, no. 4, 2022, pp. 3996–4004.

[53] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in *CVPR*, 2019, pp. 10 772–10 781.

[54] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[55] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[56] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.