Structural Entropy Based Graph Structure Learning for Node Classification

Liang Duan[1,2], **Xiang Chen**[1,2], **Wenjie Liu**[1,2], **Daliang Liu**[1,2], **Kun Yue**[1,2*], **Angsheng Li**[2,3]

[1]Yunnan Key Laboratory of Intelligent Systems and Computing, Yunnan University, Kunming, China
[2]School of Information Science and Engineering, Yunnan University, Kunming, China
[3]School of Computer Science and Engineering, Beihang University, Beijing, China
{duanl, kyue}@ynu.edu.cn, {chenx, liudl}@mail.ynu.edu.cn, liuwenjie@stu.ynu.edu.cn, angsheng@buaa.edu.cn

Abstract

As one of the most common tasks in graph data analysis, node classification is frequently solved by using graph structure learning (GSL) techniques to optimize graph structures and learn suitable graph neural networks. Most of the existing GSL methods focus on fusing different structural features (basic views) extracted from the graph, but very little graph semantics, like hierarchical communities, has been incorporated. Thus, they might be insufficient when dealing with the graphs containing noises from real-world complex systems. To address this issue, we propose a novel and effective GSL framework for node classification based on the structural information theory. Specifically, we first prove that an encoding tree with the minimal structural entropy could contain sufficient information for node classification and eliminate redundant noise via the graph's hierarchical abstraction. Then, we provide an efficient algorithm for constructing the encoding tree to enhance the basic views. Combining the community influence deduced from the encoding tree and the prediction confidence of each view, we further fuse the enhanced views to generate the optimal structure. Finally, we conduct extensive experiments on a variety of datasets. The results demonstrate that our method outperforms the state-of-the-art competitors on effectiveness and robustness.

Introduction

Node classification aims to classify the nodes in a graph with limited labels, which is a fundamental problem in graph analysis and widely used in many applications (Song, Zhang, and King 2022). The mainstream solution is training graph neural networks (GNNs) to generate node embeddings for classification (Kipf and Welling 2017). Since the performance of GNNs is highly dependent on the quality of the input graph structure, various techniques of graph structure learning (GSL) have been proposed to enhance the graph structure and fine-tune the GNN parameters for better classification (Sun et al. 2022).

Existing GSL methods mainly extract multiple graph structures (basic views) from the given graph to generate an optimal structure (final view), which should contain the information for classification while reduce redundant noise as much as possible (Sun et al. 2023). Despite the success of GSL, most of these methods aim to learn the optimal structure based on the mutual information over different views. However, traditional mutual information is unsuitable for quantifying the structural information and theoretical analysis on what is the optimal graph structure for node classification is still unexplored (Li and Pan 2016; Li et al. 2018). Many GSL methods focus on combining different but simple structural features of the original graph to improve the performance of GNNs, and rarely consider the graph semantics like hierarchical communities (Zhu et al. 2021). As a result, these methods might be insufficient when tackling the complex and noisy graphs from real-world systems (Zou et al. 2023). To address these issues, *it is essential to develop a novel theoretic principle for measuring the optimal graph structure and make full use of the structural information to improve the GSL for node classification.*

Recently, graph information bottleneck (GIB) has been proposed to optimize node embeddings by extracting the information from both graph structure and node features (Wu et al. 2020). GIB provides an interesting theoretic principle for GSL that an optimal graph structure should contain the minimal sufficient information for downstream tasks (Liu et al. 2022). Furthermore, GIB relies on the local-dependence assumption for graph data and enhances the embedding of each node by its neighborhood information. Actually, real-world graphs usually contain hierarchical communities. This structural information is useful for node classification, since the nodes within the same community are more likely to have the same class label. How to incorporate the hierarchical community information with GSL to generate the optimal graph structure for node classification is still an underexplored problem.

In this paper, we propose a structural information theory based GSL framework for node classification. Based on the structural entropy (Li and Pan 2016) and GIB, we provide a theoretic principle for GSL to find the optimal graph structure for node classification. We then prove that an encoding tree, as a hierarchical abstraction of a graph, could contain the information for classifying nodes and remove redundant noise by minimizing its structural entropy. We next design an efficient algorithm to construct the encoding tree from each basic view, such that the GSL could be guided for optimizing the graph structure. To fully use the information in the basic views, we also enhance each basic view by a sim-

*Corresponding author.

ilarity graph with minimal structural entropy. With the enhanced views, we propose a fusion mechanism to generate the final view based on the community influence from the encoding trees and the prediction confidence from the enhanced views (Liu et al. 2022). We finally prove that an optimal structure could be obtained by maximizing the mutual information between every two encoding trees, and provide a two-fold objective to train our model effectively.

To summarize, our main contributions are:

- We propose a novel framework of graph structure learning based on the structural information theory for node classification tasks.
- We design an efficient algorithm for constructing encoding tree to extract the hierarchical community information and enhance the basic views.
- We provide a community influence based fusion mechanism to generate the optimal graph structure.
- We conduct extensive experiments on a variety of datasets and the results show the superiority of our proposed method.

Preliminary

In this section, we present the basic concepts of node classification, graph structure learning and structural entropy.

Node Classification

Let $G = (V, E)$ represent a graph, where $V = \{v_1, ..., v_n\}$ is the set of nodes and $E \subseteq V \times V$ is the set of edges. The original graph structure is represented in an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, where $a_{ij} \in \boldsymbol{A}$ denotes the weight between nodes v_i and v_j. All nodes are assigned with node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ and each $\boldsymbol{x}_i \in \boldsymbol{X}$ is a d dimensional feature vector of node v_i. Given a small portion of nodes $V_L = \{v_1, ..., v_q\}$ with labels $Y_L = \{y_1, ..., y_q | y_i \in \{1, ..., C\}\}$, the node classification task is to predict the labels $\hat{Y}_U = \{\hat{y}_{q+1}, ..., \hat{y}_n\}$ for the unlabeled nodes $V_U = V \setminus V_L$. At present, the mainstream solution is to build a GNN encoder $f(\boldsymbol{X}, \boldsymbol{A})$ on the graph structure and node features, and produce low dimensional node embeddings $\boldsymbol{Z} \in \mathbb{R}^{n\times d_z} (d_z \ll d)$ for classification (Song, Zhang, and King 2022).

Graph Structure Learning

Given an input graph G, the traditional goal of GSL is to simultaneously learn an optimal graph structure $\boldsymbol{A}^\star$ and corresponding node embeddings $\boldsymbol{Z}^\star = f(\boldsymbol{X}, \boldsymbol{A}^\star)$ (Zhu et al. 2021). In this work, we focus on the graph structure learning technique for node classification tasks, in which the objective can be formulated as

$$\mathcal{L}_{gsl} = \mathcal{L}_{cls}(\boldsymbol{Z}^\star, Y_L) + \mu \mathcal{L}_{reg}(\boldsymbol{A}^\star, \boldsymbol{Z}^\star, \boldsymbol{A}) \tag{1}$$

where the first term $\mathcal{L}_{cls}$ refers to the node classification objective with respect to the given labels Y_L, the second term $\mathcal{L}_{reg}$ imposes constraints on the learned graph structure and node embeddings, and $\mu \in \mathbb{R}$ is a hyperparameter that balances the two terms.

Structural Entropy

Structural entropy is an extension of Shannon entropy for measuring the uncertainty of a graph under a strategy of hierarchical partitioning (Li and Pan 2016). The optimal hierarchical structure of a graph, also called encoding tree, could be generated by minimizing the K-dimensional structural entropy (Zeng, Peng, and Li 2023).

Encoding Tree The encoding tree $\mathcal{T}$ of a graph $G(V, E)$ is defined with the following properties: (1) Each node $\alpha \in \mathcal{T}$ is associated with a nonempty subset of nodes $T_\alpha \subseteq V$. Intuitively, for the root node λ, T_λ contains all nodes in G, i.e., $T_\lambda = V$. For each leaf node α, T_α contains a single node $v \in V$. (2) For each nonleaf node $\alpha \in \mathcal{T}$, its ith child node is denoted as $\alpha^{<i>}$, and its all child nodes' subsets $T_{\alpha^{<i>}}$ are disjointed, i.e., $T_\alpha = \cup_{i=1}^{m} T_{\alpha^{<i>}}$, where m is the number of α's children. An encoding tree is a hierarchical abstraction of G, and widely used to extract the hierarchical community information from G.

One-dimensional Structural Entropy The one-dimensional structural entropy of G reflects the dynamical complexity of G based on random walk, defined as:

$$H^1(G) = -\sum_{v\in V} \frac{d_v}{vol(G)} \log_2 \frac{d_v}{vol(G)} \tag{2}$$

where d_v is the sum of the weights of v's connected edges, and $vol(G) = \sum_{v\in V} d_v$ is the volume of G.

K-dimensional Structural Entropy Given an encoding tree $\mathcal{T}$ with height no more than K, the K-dimensional structural entropy of G is defined as follows:

$$H^K(G) = \min_{\mathcal{T}} \sum_{\alpha\in\mathcal{T}, \alpha\neq\lambda} H^{\mathcal{T}}(G; \alpha) \tag{3}$$

$$H^{\mathcal{T}}(G; \alpha) = -\frac{g_\alpha}{vol(G)} \log_2 \frac{\mathcal{V}_\alpha}{\mathcal{V}_{\alpha^-}} \tag{4}$$

where g_α is the sum of weights of edges from the nodes in T_α to those outside of T_α, $\mathcal{V}_\alpha = \sum_{v\in T_\alpha} d_v$ is the volume of T_α, and α^- is the parent node of α.

Methodology

In this section, we introduce the framework of structural entropy based graph structure learning for node classification and the technical details of each component.

Overview

The framework of our GSL method is shown in Figure 1. We first extract two basic views from the given graph as the input of our model. Then, we build an encoding tree for each basic view. One advantage of the encoding tree is that it could retain the information for classifying nodes but eliminate the noise as much as possible by minimizing the structural entropy. We use the encoding tree to train a graph convolutional network (GCN) (Kipf and Welling 2017) on each basic view and generate node embeddings, from which we construct a kNN similarity graph to enhance the basic

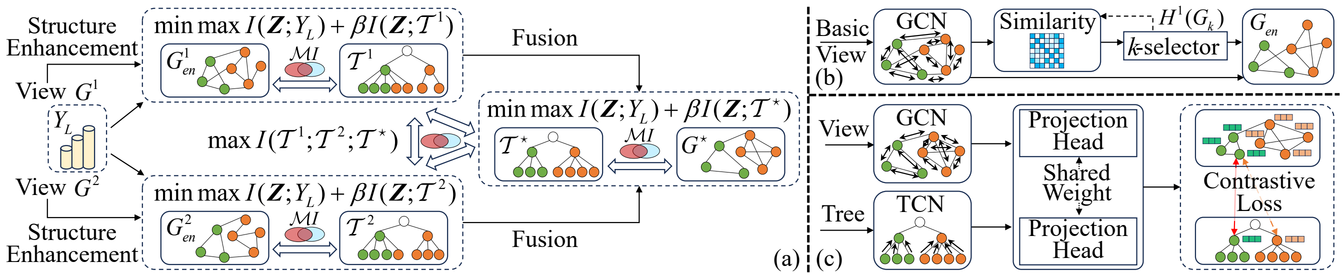

Structure Enhancement
View G^1
View G^2
Y_L
$\min\max I(\boldsymbol{Z};Y_L)+\beta I(\boldsymbol{Z};\mathcal{T}^1)$
$\min\max I(\boldsymbol{Z};Y_L)+\beta I(\boldsymbol{Z};\mathcal{T}^2)$
$\min\max I(\boldsymbol{Z};Y_L)+\beta I(\boldsymbol{Z};\mathcal{T}^\star)$
$\max I(\mathcal{T}^1;\mathcal{T}^2;\mathcal{T}^\star)$
$\mathcal{MI}$
Fusion
(a)
Basic View (b)
GCN
Similarity
$H^1(G_k)$
k-selector
G_{en}
View
Tree
TCN
Projection Head
Shared Weight
Contrastive Loss
(c)

Figure 1: (a) The framework of our method. (b) Basic view enhancement. (c) Maximization of the mutual information between graph and encoding tree.

view. Another advantage of the encoding tree is that the hierarchical community information extracted from the graph could be used for fusing the basic views. Thus, we combine the community influence and prediction confidence of each enhanced view to obtain the final view. Moreover, we also build an encoding tree for the final view to guide the training of our proposed model, which guarantees that the learned graph structure is minimal and sufficient for node classification.

Basic View Selection

Leveraging the structural information theory (Li and Pan 2016) to measure the evolution of graph in GSL, we find that GSL for node classification is to reduce the uncertainty of the original graph structure. This means that adopting more useful basic views could reduce more uncertainty in the graph. Thus, we carefully choose two basic views G^1 and G^2 as the input of our method, following the basic view selection in CoGSL.

Information Flow Constraint

A key challenge of GSL is how to constrain the information flow from basic views to the final view as to learn an optimal graph structure for downstream tasks (Zhu et al. 2021). According to GIB, the optimal structure should contain sufficient information for classification yet eliminates the noise, also called minimal sufficient structure. We adopt GIB to constrain the information flow by maximizing the mutual information between the node embeddings and labels, while minimizing the mutual information between the node embeddings and the original graph:

$$GIB(G,Y;\boldsymbol{Z}^\star)=\max_{\boldsymbol{Z}}\left[I(\boldsymbol{Z};Y)-\beta I(\boldsymbol{Z};G)\right] \quad (5)$$

where Y is the label set and $\beta>0$ is a hyperparameter. The first term $I(\boldsymbol{Z};Y)$ can be optimized by classification loss, but the second term $I(\boldsymbol{Z};G)$ is intractable to minimize.

The traditional solution for Eq. 5 is to sample a subgraph G_s from the input graph to minimize $I(\boldsymbol{Z};G)$, since G_s has less information than G. Suppose that G_s retains the information of node labels, and then we have

$$\min_{\boldsymbol{Z}} I(\boldsymbol{Z};G)\Leftrightarrow\min_{G_s} H^1(G_s) \quad (6)$$

where $H^1(\cdot)$ is the one-dimensional structural entropy.

Therefore, the goal of $\min I(\boldsymbol{Z};G)$ is to generate an enhanced graph that contains sufficient information for node classification while reducing its uncertainty (i.e., redundant information or noise) as much as possible. For this purpose, we give the following proposition.

Proposition 1. *Given a basic view G and a label set Y_L, the enhanced graph could retain the information for node classification and minimize its uncertainty, if the information flow from G to the final view satisfies:*

$$\min_{G_s}\max_{\boldsymbol{Z}} I(\boldsymbol{Z};Y_L)+\beta I(\boldsymbol{Z};G_s) \quad (7)$$

$$s.t., H^1(G)>H^1(G_s), I(G;Y_L)=I(G_s;Y_L) \quad (8)$$

The above min-max principle aims to train an encoder such that the mutual information among node embeddings $\boldsymbol{Z}$, labels Y_L and G_s can be maximized, while Eq. 8 guarantees that G_s can capture the minimal and sufficient information for node classification.

Encoding Tree Construction

In structural information theory, structural entropy is used to measure the uncertainty embedded in a graph (Li et al. 2018). Moreover, an encoding tree is a hierarchical abstraction of a graph, which could be used to represent the hierarchical community partition of the graph. Minimizing the uncertainty of a graph could be implemented by an encoding tree with the minimal structural entropy (Zou et al. 2023), stated as follows.

Proposition 2. *The encoding tree $\mathcal{T}^\star$ of G with the minimal structural entropy could retain the information for node classification and eliminate redundant noise in G.*

According to Proposition 2, we incorporate the encoding tree into the min-max principle to train GNNs for node classification. Different from previous GSL methods that generate the enhanced graph by graph sampling, we adopt the encoding tree to enhance the graph with theoretical guarantee. According to Eq. 3, the encoding tree with minimum K-dimensional structural entropy could be found by

$$\mathcal{T}^\star=\mathop{\arg\min}_{\forall\mathcal{T}:height(\mathcal{T})\le K}\left(H^{\mathcal{T}}(G)\right) \quad (9)$$

However, building an optimal encoding tree is intractable (Zou et al. 2023). For this, we design an efficient algorithm for encoding tree construction. Specifically, given a graph

$G(V, E)$, let $\mathcal{P} = \{P_1, ..., P_c\}$ be a partition of V, where each $P_i \subset V$ is called a community. We define three basic operators as follows:

Definition 1. *(Merging operator) Given any two communities P_i and P_j ($1 \leq i < j \leq c$), merging operator $op_m(P_i, P_j)$ merges P_i and P_j into a new community P_x, i.e., $P_x = P_i \cup P_j$, and then removes P_i and P_j from $\mathcal{P}$. After merging, $\mathcal{P} = \{P_1, ..., P_{i-1}, P_{i+1}, ..., P_{j-1}, P_{j+1}, P_c, P_x\}$.*

According to Eq. 4, the difference of K-dimensional structural entropy $\Delta SE^{\mathcal{P}}_{i,j}(G)$ before and after merging could be calculated by

$$\Delta SE^{\mathcal{P}}_{i,j}(G) = \frac{1}{vol(G)}[(\mathcal{V}_i - g_i)\log_2 \mathcal{V}_i + (\mathcal{V}_j - g_j)\log_2 \mathcal{V}_j - (\mathcal{V}_x - g_x)\log_2 \mathcal{V}_x + (g_i + g_j - g_x)\log_2 vol(G)] \tag{10}$$

Definition 2. *(Compressing operator) Given a graph G and a corresponding partition $\mathcal{P}$, compressing operator $op_c(\mathcal{P})$ compresses G into a smaller graph by transferring each community $P_i \in \mathcal{P}$ to a node v'_i, and assigning the weight of edge between v'_i and v'_j to the sum of the weights of the edges from P_i to P_j.*

Definition 3. *(Updating operator) Given an encoding tree $\mathcal{T}$ and a graph G with partition $\mathcal{P}$, updating operator $op_u(\mathcal{T}, \mathcal{P})$ is to update the encoding tree by taking all communities in $\mathcal{P}$ as the leaf nodes of $\mathcal{T}$, i.e., inserting $\mathcal{P}$ into $\mathcal{T}$ and increasing the height of $\mathcal{T}$.*

Initially, we adopt each node in the graph as a single community, and then iteratively execute the merging and compressing operators until the updating operator could construct a K-dimensional encoding tree. Actually, in the merging operation, we merge the communities with the maximal $\Delta SE^{\mathcal{P}}_{i,j}(G)$ greedily until there are no communities satisfying $\Delta SE^{\mathcal{P}}_{i,j}(G) > 0$, which can achieve the minimal structural entropy. The complete procedure is shown in Algorithm 1.

Algorithm 1: Encoding Tree Construction

```
Input: a graph $G$, an integer $K > 1$
Output: an encoding tree $\mathcal{T}$
 1: $G_1 \leftarrow G$, $\mathcal{T} \leftarrow$ an encoding tree with height 1
 2: for $h = 1$ to $K$ do
 3:     $\mathcal{P}_h \leftarrow$ initialize each node in $G_h$ as a community
 4:     while $True$ do
 5:         $P'_i, P'_j \leftarrow \arg\max \Delta SE^{\mathcal{P}_h}_{i,j}(G_h)$ by Eq. 10
 6:         if $\Delta SE^{\mathcal{P}_h}_{i,j}(G_h) > 0$ then
 7:             $\mathcal{P}_h \leftarrow op_m(P'_i, P'_j)$, continue // Definition 1
 8:         else
 9:             $G_h \leftarrow op_c(\mathcal{P}_h)$, break // Definition 2
10:         end if
11:     end while
12: end for
13: for $h = K - 1$ down to 0 do
14:     $\mathcal{T} \leftarrow op_u(\mathcal{T}, \mathcal{P}_h)$ // Definition 3
15: end for
16: return $\mathcal{T}$
```

Graph Structure Enhancement

To fully use the information contained in the basic views, we have to enhance the basic views before generating the final view. For the basic view G^1, we train a GCN encoder $f(\boldsymbol{X}, \boldsymbol{A}^1)$ to generate the node embeddings $\boldsymbol{Z}^1$. When the embeddings are available, we use the cosine similarity $s^1_{ij} = (\boldsymbol{z}^1_i \cdot \boldsymbol{z}^1_j)/(||\boldsymbol{z}^1_i|| \times ||\boldsymbol{z}^1_j||)$ to measure the relation between node v^1_i and v^1_j. Intuitively, larger s^1_{ij} means higher probability that v^1_i and v^1_j are in the same class. It is reasonable to construct a cosine similarity graph to represent the relations among all nodes, but this similarity graph is unsuitable for node classification since it might be fully connected (i.e., all nodes belong to one class).

Consequently, we build a k-nearest neighbor (kNN) graph G^1_k to enhance the basic view. Based on the structural information theory, we could find an optimal value for k by minimizing the one-dimensional structural entropy of G^1_k, i.e., finding the value of k satisfying $H^1(G^1_{k-1}) \geq H^1(G^1_k) \leq H^1(G^1_{k+1})$. Note that the optimal k guarantees the kNN graph could retain the most useful information in the corresponding node embeddings. Thus, the value of k is selected based on the structural entropy and does not require users to provide. Combining G^1_k with the basic view G^1, we obtain the following enhanced view

$$G^1_{en} = G^1 + \xi G^1_k \tag{11}$$

where $\xi \in [0, 1]$ is a combination coefficient. Similarly, we generate the enhanced view G^2_{en} from G^2.

Final View Generation

An important step in GSL is fusing all basic views to generate the final view (Zhu et al. 2021). Different from previous methods that use the average or attention mechanism (Zhao et al. 2021), we combine the community influence and prediction confidence to fuse the basic views.

First, we define the community influence.

Definition 4. *(Community influence) Given an optimal encoding tree $\mathcal{T}$ of a graph G, the community influence of a leaf node α in $\mathcal{T}$ is*

$$\varepsilon_\alpha = \frac{H^{\mathcal{T}}(G; \alpha)}{\sum_{\delta \in \mathcal{T}} H^{\mathcal{T}}(G; \delta)} \tag{12}$$

where δ is the node in the path from λ to α.

Intuitively, $H^{\mathcal{T}}(G; \alpha)$ reflects the activity of the node α in its community T_{α^-}, and larger ε_α indicates larger influence of α in T_{α^-}.

Then, for each node v_i, we combine the community influence and prediction confidence to measure the importance a^1_i of v_i in the basic view G^1, defined as

$$a^1_i = \frac{\sigma(\pi^1_i) \cdot \pi^1_i + \sigma(\varepsilon^1_i) \cdot \varepsilon^1_i}{\sigma(\pi^1_i) + \sigma(\varepsilon^1_i)} \tag{13}$$

where $\sigma(\cdot)$ is an activation function, and π^1_i is v_i's prediction confidence in G^1. We use the same prediction confidence as CoGSL, and obtain a^2_i of v_i in G^2 analogously.

Next, we normalize the importance and obtain the weights of v_i as follows:

$$w_i^1 = a_i^1/(a_i^1 + a_i^2), \quad w_i^2 = a_i^2/(a_i^1 + a_i^2) \tag{14}$$

Finally, we generate v_i's final view:

$$G_i^\star = w_i^1 \cdot G_{en,i}^1 + w_i^2 \cdot G_{en,i}^2 \tag{15}$$

The above operations are repeatedly executed to fuse all nodes and generate the final view $G^\star$.

Model Training

We now discuss how to instantiate the min-max principle and use the community information to guide the training of our GSL model, as well as the generation of the minimal sufficient structure for node classification.

Minimal Sufficient Structure We aim to make full use of the hierarchical community information to guide the training of our GSL model, since the information could be decoded from the encoding tree. Based on the definitions of minimal sufficient structure (Liu et al. 2022) and structural entropy (Li and Pan 2016), we have the following proposition.

Proposition 3. *Given the enhanced views G_{en}^1 and G_{en}^2, final view $G^\star$ and their encoding trees $\mathcal{T}^1$, $\mathcal{T}^2$ and $\mathcal{T}^\star$, and label set Y_L, $G^\star$ is a minimal sufficient structure guided by community information if the following two principles are satisfied:*

$$I(G_{en}^1; Y_L) = I(G_{en}^2; Y_L) = I(G^\star; Y_L) = H(Y_L) \tag{16}$$

$$\max\{I(\mathcal{T}^1; \mathcal{T}^2) + I(\mathcal{T}^1; \mathcal{T}^\star) + I(\mathcal{T}^2; \mathcal{T}^\star)\} \tag{17}$$

where $H(\cdot)$ denotes the Shannon entropy.

Eq. 16 guarantees that the information of Y_L for node classification is totally contained in G_{en}^1, G_{en}^2 and $G^\star$. Meanwhile, Eq. 17 connects the encoding trees of the basic and the final views. Maximizing the mutual information among these encoding trees could make these trees share their community information for generating the minimal sufficient structure.

Training Objective We design a two-fold objective to optimize the parameters in our model.
(1) Optimizing the parameters of GCNs over the enhanced and final views to improve the classification accuracy by the following cross-entropy loss:

$$\mathcal{L}_{cls} = \sum_{i=1}^{2} \mathcal{L}_{cross}(\Pi^i, Y_L) + \mathcal{L}_{cross}(\Pi^\star, Y_L) \tag{18}$$

where Π^1, Π^2 and $\Pi^\star$ is the prediction confidence of G_{en}^1, G_{en}^2 and $G^\star$, respectively.
(2) Optimizing the parameters of basic view enhancers to constraint the information flow by the min-max principle and maximization of mutual information of encoding trees.

For the min-max principle in Eq. 7, we adopt the cross-entropy loss to maximize the first term $I(\boldsymbol{Z}; Y_L)$, and design a hierarchical contrastive loss for the second term $I(\boldsymbol{Z}; G_s)$. According to Proposition 2, G_s is replaced by an encoding tree $\mathcal{T}$ in our method. Thus, we first provide a tree convolutional network (TCN) to generate the community embedding from $\mathcal{T}$. Actually, the community embedding of node α in $\mathcal{T}$ is the weighted sum of the embeddings of α's children, formally defined as

$$\boldsymbol{h}_\alpha = \sum_{i=1}^{m} \left[\frac{h^{\mathcal{T}}(G; \alpha^{<i>})}{\sum_{j=1}^{m} h^{\mathcal{T}}(G; \alpha^{<j>})} \boldsymbol{h}_{\alpha^{<i>}} \right] \tag{19}$$

where m is the number of α's children. The community embeddings of leaf nodes in $\mathcal{T}$ are the corresponding node embeddings $\boldsymbol{Z}$.

Inspired by InfoNCE (Oord, Li, and Vinyals 2018), we compare the node embeddings with different levels of community embeddings to make the nodes in the same community have similar embeddings. The corresponding hierarchical contrastive loss is

$$\mathcal{L}_{hc}(\boldsymbol{Z}; \mathcal{T}) = -\sum_{l=2}^{K} \theta_l \log_2 \sum_{i=1}^{n} \frac{sim(\boldsymbol{z}_i, \boldsymbol{h}_{(i,l)})}{\sum_{j=1, j\neq i}^{n} sim(\boldsymbol{z}_j, \boldsymbol{h}_{(j,l)})} \tag{20}$$

where $\theta_l = \gamma(1-\gamma)^l$ is a coefficient related to the level number l, $sim(\cdot)$ is the cosine similarity, and $\boldsymbol{h}_{(i,l)}$ is the embedding of the l-th level's community of node v_i in $\mathcal{T}$.

For the enhanced view G_{en}^1, its min-max principle loss is

$$\mathcal{L}_{mmp}^1 = \mathcal{L}_{cross}^1(\Pi^1, Y_L) + \mathcal{L}_{hc}(\boldsymbol{Z}^1; \mathcal{T}^1) \tag{21}$$

Similarly, we can get the loss $\mathcal{L}_{mmp}^2$ and $\mathcal{L}_{mmp}^\star$ for G_{en}^2 and $G^\star$ respectively, as well as the total min-max principle loss $\mathcal{L}_{mmp} = \mathcal{L}_{mmp}^1 + \mathcal{L}_{mmp}^2 + \mathcal{L}_{mmp}^\star$.

To maximize the mutual information between the two encoding trees $\mathcal{T}^1$ and $\mathcal{T}^2$, we extend Eq. 20 as

$$\mathcal{L}_{miet}(\mathcal{T}^1, \mathcal{T}^2) = \frac{1}{2}\left[\mathcal{L}_{hc}(\boldsymbol{Z}^1; \mathcal{T}^2) + \mathcal{L}_{hc}(\boldsymbol{Z}^2; \mathcal{T}^1)\right] \tag{22}$$

Consequently, the loss for the basic view enhancers is

$$\begin{aligned}\mathcal{L}_{ve} =& \mathcal{L}_{mmp} + (\mathcal{L}_{miet}(\mathcal{T}^1, \mathcal{T}^2) + \mathcal{L}_{miet}(\mathcal{T}^1, \mathcal{T}^\star) \\ &+ \mathcal{L}_{miet}(\mathcal{T}^2, \mathcal{T}^\star))\end{aligned} \tag{23}$$

To effectively train our model, we alternatively and iteratively perform the above two-fold objective.

Experimental Study

In this section, we conduct extensive experiments to evaluate the effectiveness and robustness of our method.

Experimental Setup

Datasets We choose eight open benchmark datasets for experiments, including (1) blog graph Polblogs (Pedregosa et al. 2011), (2) website networks from WebKB, Texas and Wisconsin (Bandyopadhyay et al. 2005), (3) citation networks, Citeseer (Kipf and Welling 2017), Wiki-CS (Mernyei and Cangea 2020) and MS Academic (Klicpera, Bojchevski, and Gunnemann 2019), and (4) non-graph datasets, Breast Cancer (Cancer) and Digits (Pedregosa et al. 2011). We construct a kNN graph as an initial adjacency matrix for each non-graph dataset, and adopt the original splits on training, validation and test sets.

Method	Texas	Wisconsin	Cancer	Digits	Polblogs	Citeseer	Wiki-CS	MS Academic
GCN	54.4±5.6	50.7±1.6	93.5±0.4	90.3±0.5	95.2±0.2	69.7±0.6	71.0±0.8	91.3±0.5
GAT	56.0±1.4	52.1±0.9	93.8±0.9	90.8±1.1	94.3±0.3	71.9±0.1	72.8±0.3	89.3±0.2
GATv2	52.1±0.6	49.4±1.2	95.0±0.2	90.6±0.7	94.7±0.6	71.6±0.1	69.2±1.1	89.4±0.8
GraphSAGE	55.0±5.9	51.3±3.9	92.7±0.4	88.2±0.2	93.7±2.1	70.4±0.8	72.5±0.4	91.2±0.2
DGI	43.8±3.6	49.4±3.1	86.9±1.5	89.1±0.4	91.7±0.8	72.0±0.5	61.0±0.3	90.5±1.2
gCooL	59.0±3.8	52.4±2.8	95.1±0.5	91.3±0.9	95.4±0.3	67.9±1.9	74.1±0.5	91.2±0.4
MGEDE	57.1±1.1	50.0±0.4	94.8±0.1	91.5±0.2	94.9±0.1	68.7±0.2	68.9±0.6	90.1±0.2
IDGL	49.5±9.1	54.3±1.9	94.5±0.6	92.8±0.1	94.4±0.6	72.6±0.4	72.7±0.8	-
Pro-GNN	55.1±3.5	58.0±3.1	93.4±0.6	90.5±0.8	95.0±0.1	67.6±0.4	68.9±0.8	-
GEN	51.1±6.7	54.5±4.1	94.1±0.8	91.8±0.9	95.3±0.4	72.5±0.8	71.5±0.7	91.8±0.5
CoGSL	57.7±3.5	55.7±1.2	94.8±0.2	92.5±0.5	95.5±0.1	72.7±0.3	74.7±0.4	92.1±0.2
SE-GSL	59.2±7.6	58.0±4.0	92.6±0.3	82.7±0.5	95.1±0.5	71.4±1.3	53.2±1.6	90.8±0.1
PROSE	58.4±2.1	58.2±1.9	95.4±0.4	92.8±0.5	95.5±0.4	73.3±0.1	75.0±0.7	91.8±0.2
Ours	**61.3±3.0**	**58.6±1.2**	**95.8±0.5**	**93.7±0.3**	**95.9±0.1**	**74.1±0.6**	**75.6±0.3**	**92.7±0.1**

Table 1: Effectiveness comparison on F1-micro ($\% \pm \sigma$). (bold: best, '-': exceeding GPU memory or cannot finish in 12 hours)

Comparison Methods We compare our method with three categories of node classification methods.

(1) Classical GNN methods: GCN (Kipf and Welling 2017), GAT (Velickovic et al. 2018), GATv2 (Brody, Alon, and Yahav 2022) and GraphSAGE (Hamilton, Ying, and Leskovec 2017).

(2) Information theory (IT) based methods: DGI (Velickovic et al. 2019), gCooL (Li, Jing, and Tong 2022) and MGEDE (Yang et al. 2023).

(3) Graph structural learning (GSL) based methods: IDGL (Chen, Wu, and Zaki 2020), Pro-GNN (Jin et al. 2020), GEN (Wang et al. 2021), CoGSL (Liu et al. 2022), SE-GSL (Zou et al. 2023) and PROSE (Wang et al. 2023).

Metrics We evaluate the accuracy of a node classification algorithm by a standard metric F1-micro ranging from 0 - 100%, and a higher score means a more accurate classifier.

Implementation We implement our method in PyTorch. For fair comparison, we use the same dimensionality of node embeddings and optimizer for all methods (except MGEDE), and set other parameters to the values recommended in the original papers. All experiments are conducted on a machine with Intel 13900KF CPU, 128GB RAM and RTX4090 GPU, running Windows 11. Each test is repeated for 10 times, and the average is reported here.

Experimental Results

Exp-1: Effectiveness Evaluation In the first set of tests, we evaluate the effectiveness of our method by comparing with other node classification methods. The encoding tree height K is fixed to 3. The results are reported in Table 1.

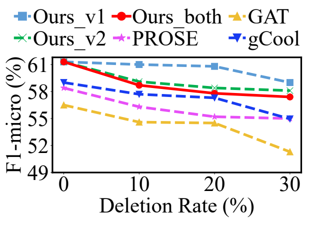
The results tell us that: (a) our method outperforms the other comparison methods on all datasets. (b) PROSE is the second best method and GSL based methods generally perform better than the other two categories of methods, since GSL could enhance the graph structure for classification. (c) gCooL is the best IT based method and GAT is the best classical GNN method. SE-GSL performs worse on Digits and Wiki-CS, since it requires large amount of labeled training data and is unsuitable for node classification with a small set of labels. By adopting the encoding tree to extract the hierarchical community information for the model training, our method improves F1-micro over the second best method PROSE. These verify the effectiveness of our method.

Exp-2: Attacks on Edges In the second set of tests, we follow (Liu et al. 2022) to evaluate the robustness of our method by random edge deletions and additions. We choose PROSE (the second best method), gCooL (the best IT based method) and GAT (the best classical GNN method) for comparison. The curves of 'Ours_v1', 'Ours_v2' and 'Ours_both' are the results of the first, second and all basic views of our method are attacked, respectively. The results on Texas and MS Academic are reported in Figure 2.

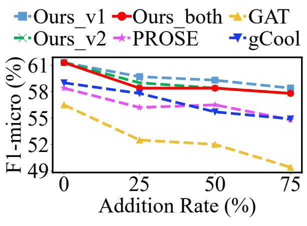
The results tell us that: (a) our method outperforms other methods under different perturbations. (b) GSL methods are better than other methods. (c) all methods become worse with the increase of perturbations. (d) 'Our_both' is competitive with 'Ours_v1' and 'Ours_v2' that only one basic view is attacked. These verify the robustness of our method.

Exp-3: Impacts of Encoding Tree In the third set of tests, we analyze the impacts of encoding tree by comparing with other two community structure extraction methods kNN and Louvain (Blondel et al. 2008) and varying the tree height K from 2 to 5. The results are reported in Figure 3.

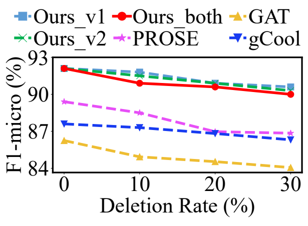
The results tell us that: (a) encoding tree is more effective than the other methods to extract the community information. (b) kNN performs as well as Louvain, which means that community information is indeed useful for GSL. (c) the F1-micro scores are stable with the increase of K, which verifies the insensitivity of our method to the tree height. (d) we fix $K = 3$ by default for better efficiency, since constructing a high encoding tree is time-consuming.



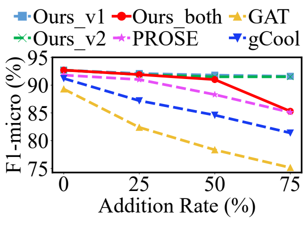
(a) Texas



(b) Texas

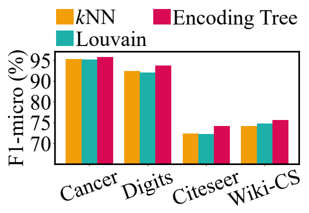


(c) MS Academic

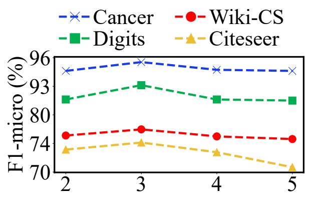


(d) MS Academic

Figure 2: F1-micro results of different methods under edge attacks.



(a) Different community structure extraction methods



(b) Impacts of K

Figure 3: Impacts of encoding tree.

Fusion	Cancer	Polblogs	Citeseer	Wiki-CS
Average	95.0	95.6	73.4	74.9
Attention	95.2	95.3	71.8	70.7
Confidence	95.3	94.7	73.3	75.2
Ours	**95.8**	**95.9**	**74.1**	**75.6**

Table 2: F1-micro results of different fusion mechanisms.

Exp-4: Impacts of Fusion Mechanism In the last set of tests, we evaluate our fusion mechanism by comparing with other three mechanisms: average, attention and prediction confidence. The results are reported in Table 2.

The results tell us that: (a) our fusion mechanism performs better than others on all datasets. (b) 'Average' performs as well as 'Confidence', but 'Attention' performs worse, since it requires large training data to fine-tune parameters. By combining the community influence and prediction confidence, our fusion mechanism outperforms the competitors, which verifies the effectiveness of our method.

Related Work

Node Classification Node classification is a primary task in graph analysis. The mainstream solution is training GNNs to aggregate neighborhood information for better node embeddings, e.g., GCN (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017), GAT (Velickovic et al. 2018) and GATv2 (Brody, Alon, and Yahav 2022). Some methods incorporate the information theory with GNNs, e.g., DGI (Velickovic et al. 2019), gCool (Li, Jing, and Tong 2022) and MGEDE (Yang et al. 2023). Recently, GSL techniques are used to enhance the node embeddings and become the dominant solution (Zhu et al. 2021), which motivates our study of GSL based node classification.

Graph Structure Learning GSL is to simultaneously optimize the graph structure and node embeddings (Song, Zhang, and King 2022). IDGL (Chen, Wu, and Zaki 2020) recalibrates edge weights by node embedding similarities, Pro-GNN (Jin et al. 2020) treats the graph structure as a trainable parameter in GNN training, and GEN (Wang et al. 2021) optimizes the graph structure by stochastic block model. Moreover, information theory has been adopted in GSL, e.g., SE-GSL (Zou et al. 2023) adopts structural entropy to enhance connectivity among uncertain nodes, CoGSL (Liu et al. 2022) uses mutual information to learn the minimal sufficient structure, and PROSE (Wang et al. 2023) uses a progressive strategy to learn graph structures. Most of these methods focus on extracting multiple and simple structural features, and neglect to use the graph semantics, such as hierarchical community information. Different from these methods, we adopt the encoding tree to hierarchically abstract the graph and enhance the basic views in GSL.

Structural Entropy Guided Neural Network As an advanced theory in graph analysis, structural entropy has gained substantial traction and been widely used in bioinformatics (Li, Yin, and Pan 2016) and community detection (Liu et al. 2019). Recent works combine structural entropy with neural networks, e.g., HRN (Wu et al. 2022), SR-MARL (Zeng, Peng, and Li 2023) and SEGA (Wu et al. 2023). Although these methods successfully exploit the structural entropy to optimize neural networks, how to incorporate this theory with GSL to measure and find the optimal structure for node classification is still understudied, and we are among the first attempts.

Conclusion

In this work, we propose a structural entropy based approach to improving GSL for node classification. We first prove that an encoding tree with minimal structural entropy could extract hierarchical community information for classification while removing redundant noise in the graph. We then provide a community influence based fusion mechanism to generate the final view. Finally, we efficiently construct encoding trees for all views and apply them to guide the training of our GSL model. Extensive experiment results show the effectiveness and robustness of our method.

Acknowledgments

This work was supported by the Key Program of National Natural Science Joint Foundation of China (U23A20298), National Natural Science Foundation of China (61932002), Program of Yunnan Key Laboratory (202205AG070003) and Yunnan Fundamental Research Project (202301AT070193).

References

Bandyopadhyay, S.; Maulik, U.; Holder, L. B.; Cook, D. J.; and Getoor, L. 2005. Link-based Classification. *Advanced Methods for Knowledge Discovery From Complex Data*, 189–207.

Blondel, V. D.; Guillaume, J.-L.; Lambiotte, R.; and Lefebvre, E. 2008. Fast Unfolding of Communities in Large Networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10): P10008.

Brody, S.; Alon, U.; and Yahav, E. 2022. How Attentive are Graph Attention Networks? In *ICLR*.

Chen, Y.; Wu, L.; and Zaki, M. 2020. Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings. In *NeurIPS*, 19314–19326.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.

Jin, W.; Ma, Y.; Liu, X.; Tang, X.; Wang, S.; and Tang, J. 2020. Graph Structure Learning for Robust Graph Neural Networks. In *SIGKDD*, 66–74.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Klicpera, J.; Bojchevski, A.; and Gunnemann, S. 2019. Predict then Propagate: Graph Neural Networks Meet Personalized PageRank. In *ICLR*.

Li, A.; and Pan, Y. 2016. Structural Information and Dynamical Complexity of Networks. *IEEE Transactions on Information Theory*, 62(6): 3290–3339.

Li, A.; Yin, X.; and Pan, Y. 2016. Three-dimensional Gene Map of Cancer Cell Types: Structural Entropy Minimisation Principle for Defining Tumour Subtypes. *Scientific Reports*, 6(1): 20412.

Li, A.; Yin, X.; Xu, B.; Wang, D.; Han, J.; Wei, Y.; Deng, Y.; Xiong, Y.; and Zhang, Z. 2018. Decoding Topologically Associating Domains with Ultra-low Resolution Hi-C Data by Graph Structural Entropy. *Nature Communications*, 9(1): 3265.

Li, B.; Jing, B.; and Tong, H. 2022. Graph Communal Contrastive Learning. In *WWW*, 1203–1213.

Liu, N.; Wang, X.; Wu, L.; Chen, Y.; Guo, X.; and Shi, C. 2022. Compact Graph Structure Learning via Mutual Information Compression. In *WWW*, 1601–1610.

Liu, Y.; Liu, J.; Zhang, Z.; Zhu, L.; and Li, A. 2019. REM: From Structural Entropy to Community Structure Deception. In *NeurIPS*, volume 32.

Mernyei, P.; and Cangea, C. 2020. Wiki-CS: A Wikipedia-based Benchmark for Graph Neural Networks. *arXiv preprint arXiv:2007.02901*.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Song, Z.; Zhang, Y.; and King, I. 2022. Towards an Optimal Asymmetric Graph Structure for Robust Semi-supervised Node Classification. In *SIGKDD*, 1656–1665.

Sun, Q.; Li, J.; Peng, H.; Wu, J.; Fu, X.; Ji, C.; and Philip, S. Y. 2022. Graph Structure Learning with Variational Information Bottleneck. In *AAAI*, volume 36, 4165–4174.

Sun, Q.; Li, J.; Yang, B.; Fu, X.; Peng, H.; and Philip, S. Y. 2023. Self-organization Preserved Graph Structure Learning with Principle of Relevant Information. In *AAAI*, volume 37, 4643–4651.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Velickovic, P.; Fedus, W.; Hamilton, W. L.; Lio, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *ICLR*.

Wang, H.; Fu, Y.; Yu, T.; Hu, L.; Jiang, W.; and Pu, S. 2023. PROSE: Graph Structure Learning via Progressive Strategy. In *SIGKDD*, 2337–2348.

Wang, R.; Mou, S.; Wang, X.; Xiao, W.; Ju, Q.; Shi, C.; and Xie, X. 2021. Graph Structure Estimation Neural Networks. In *WWW*, 342–353.

Wu, J.; Chen, X.; Shi, B.; Li, S.; and Xu, K. 2023. SEGA: Structural Entropy Guided Anchor View for Graph Contrastive Learning. *arXiv preprint arXiv:2305.04501*.

Wu, J.; Li, S.; Li, J.; Pan, Y.; and Xu, K. 2022. A Simple Yet Effective Method for Graph Classification. In *IJCAI*, 3580–3586.

Wu, T.; Ren, H.; Li, P.; and Leskovec, J. 2020. Graph Information Bottleneck. In *NeurIPS*, 20437–20448.

Yang, Z.; Zhang, G.; Wu, J.; Yang, J.; Sheng, Q. Z.; Peng, H.; Li, A.; Xue, S.; and Su, J. 2023. Minimum Entropy Principle Guided Graph Neural Networks. In *WSDM*, 114–122.

Zeng, X.; Peng, H.; and Li, A. 2023. Effective and Stable Role-based Multi-Agent Collaboration by Structural Information Principles. In *AAAI*.

Zhao, J.; Wang, X.; Shi, C.; Hu, B.; Song, G.; and Ye, Y. 2021. Heterogeneous Graph Structure Learning for Graph Neural Networks. In *AAAI*, volume 35, 4697–4705.

Zhu, Y.; Xu, W.; Zhang, J.; Liu, Q.; Wu, S.; and Wang, L. 2021. Deep Graph Structure Learning for Robust Representations: A Survey. *arXiv preprint arXiv:2103.03036*, 14.

Zou, D.; Peng, H.; Huang, X.; Yang, R.; Li, J.; Wu, J.; Liu, C.; and Yu, P. S. 2023. SE-GSL: A General and Effective Graph Structure Learning Framework through Structural Entropy Optimization. In *WWW*.