Compact Graph Structure Learning via Mutual Information Compression

Nian Liu
nianliu@bupt.edu.cn
Beijing University of Posts and Telecommunications
China

Xiao Wang
xiaowang@bupt.edu.cn
Beijing University of Posts and Telecommunications
Peng Cheng Laboratory
China

Lingfei Wu
lwu@email.wm.edu
JD.COM Silicon Valley Research Center
United States

Yu Chen
hugochen@fb.com
Meta AI
United States

Xiaojie Guo
xguo7@gmu.edu
JD.COM Silicon Valley Research Center
United States

Chuan Shi*
shichuan@bupt.edu.cn
Beijing University of Posts and Telecommunications
Peng Cheng Laboratory
China

ABSTRACT

Graph Structure Learning (GSL) recently has attracted considerable attentions in its capacity of optimizing graph structure as well as learning suitable parameters of Graph Neural Networks (GNNs) simultaneously. Current GSL methods mainly learn an optimal graph structure (final view) from single or multiple information sources (basic views), however the theoretical guidance on what is the optimal graph structure is still unexplored. In essence, an optimal graph structure should only contain the information about tasks while compress redundant noise as much as possible, which is defined as "minimal sufficient structure", so as to maintain the accuracy and robustness. How to obtain such structure in a principled way? In this paper, we theoretically prove that if we optimize basic views and final view based on mutual information, and keep their performance on labels simultaneously, the final view will be a minimal sufficient structure. With this guidance, we propose a ***Co**mpact **GSL*** architecture by MI compression, named ***CoGSL***. Specifically, two basic views are extracted from original graph as two inputs of the model, which are refinedly reestimated by a view estimator. Then, we propose an adaptive technique to fuse estimated views into the final view. Furthermore, we maintain the performance of estimated views and the final view and reduce the mutual information of every two views. To comprehensively evaluate the performance of CoGSL, we conduct extensive experiments on several datasets under clean and attacked conditions, which demonstrate the effectiveness and robustness of CoGSL.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Networks** → **Network algorithms**.

KEYWORDS

Graph Neural Networks, Graph Structure Learning, Mutual Information

ACM Reference Format:
Nian Liu, Xiao Wang, Lingfei Wu, Yu Chen, Xiaojie Guo, and Chuan Shi. 2022. Compact Graph Structure Learning via Mutual Information Compression. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3485447.3512206

Proposition 1. *Given two basic views $\boldsymbol{V_1}$ and $\boldsymbol{V_2}$, final view $\boldsymbol{V}^\star$, and labels $\mathcal{Y}_L$ related to downstream task, $\boldsymbol{V}^\star$ is a minimal sufficient structure to $\mathcal{Y}_L$ if the following two principles are satisfied:*

1. $I(\boldsymbol{V_1};\mathcal{Y}_L) = I(\boldsymbol{V_2};\mathcal{Y}_L) = I(\boldsymbol{V}^\star;\mathcal{Y}_L) = H(\mathcal{Y}_L)$

2. *minimize* $I(\boldsymbol{V_1};\boldsymbol{V_2}) + I(\boldsymbol{V_1};\boldsymbol{V}^\star) + I(\boldsymbol{V_2};\boldsymbol{V}^\star)$

1 INTRODUCTION

Graph is capable of modeling real systems in diverse domains varying from natural language and images to network analysis. Nowadays, as an emerging technique, Graph Neural Networks (GNNs) [12, 19, 32] have achieved great success with their characteristic message passing scheme [10] that aims to aggregate information from neighbors continually. So far, GNNs have shown superior performance in a wide range of applications, such as node classification [37, 38] and link prediction [39, 40].

It is well known that the performance of GNNs is closely related to the quality of given graphs [9]. However, due to the complexity of real information sources, the quality of graphs is often unreliable [23]. On one hand, we are not always provided with graph structures, such as in natural language processing [5, 21] or computer vision [29, 30]. In these cases, graphs are constructed by involving prior knowledge, which is sometimes error-prone. On the other hand, even though interactions between objects are extracted, spurious edges are usually inevitably existed in graphs. For example, it

[*] Corresponding author.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512206

is very hard to analyze the molecular structure of unknown proteins [1], so they are prone to be modeled with wrong or useless connections. Furthermore, graphs sometimes suffer from malicious attacks, such that original structures are fatally destroyed. With attacked graphs, GNNs will be very vulnerable. As a result, various drawbacks are prevalent in real graphs, which prohibits original structure from being the optimal one for downstream tasks.

Recently, graph structure learning (GSL) has aroused considerable attentions, which aims to learn optimal graph structure and parameters of GNNs simultaneously [43]. Current GSL methods can be roughly divided into two categories, single-view [9, 16, 17, 42] based and multi-view based [4, 28, 34]. For the former, they usually estimate the optimal structure from one view, i.e., the given adjacency matrix, by forcing the learned structure to accord with some properties. For instance, Pro-GNN [17] learns the graph structure with low rank, sparsity and feature smoothness constraints. For the later, considering that the measurement of an edge based on only one view may be biased, they aim to extract multiple basic views from original structure, and then comprehensively estimate the final optimal graph structure based on these views. As an example, IDGL [4] constructs the structure by two type of views: normalized adjacency matrix and similarity matrix calculated with node embeddings. Multi-view based methods are able to utilize multifaceted knowledge to make the final decision on GSL.

Here, we focus on in-depth analysis of multi-view based GSL methods, and aim to answer one fundamental question: *how can we estimate the optimal graph structure from multiple views in a principled way?* Despite that multiple views based GSL methods are considered as a promising solution, there is still a lack of theoretical guidance to determine what is "optimal" in principle. In essence, an optimal graph structure should only contain the most concise information about downstream tasks (e.g., node labels), no more and no less, so that it can conduct the most precise prediction on labels. If the learned structure absorbs the information of labels as well as additional irrelevance from basic views, this structure is more prone to adversarial attacks when small perturbations are deployed on these irrelevant parts. While if the learned structure only holds limited information about labels, the model probably fails to support downstream tasks. In summary, the optimal structure should contain minimal but sufficient information of labels, and we call it *minimal sufficient structure*, which makes a well balance between effectiveness and robustness. Other GSL methods mainly focus on the performance, while neglecting the compactness of structure. Hence, the structures learnt from them inevitably contain redundant noise, and are also vulnerable to perturbations.

However, it is technically challenging to obtain a minimal sufficient graph structure. Particularly, two obstacles need to be addressed. (1) How to ensure the minimum and sufficiency of the final view? To achieve the sufficiency, the final view should be fully guided by labels, which makes it contain the information about labels as much as possible. And for the minimum, considering that the final view extracts information from basic views, we need to constrain the information flow from basic views to final view, which avoids irrelevant information and contributes to the conciseness of the final view. Therefore, to be minimal and sufficient, we we need to rethink on how to formulate the relations among basic views, final view and labels. (2) How to ensure the effectiveness of basic views? Considering that basic views are the information source of final view, it is vital to guarantee the quality of basic views. On one hand, basic views are also needed to contain the information about labels, which can fundamentally guarantee the performance of final view. On the other hand, these views also should be independent of each other, so that they can eliminate the redundancy and provide diverse knowledge about labels for final view. However, it is hard to guarantee the raw basic views satisfy these requirements, implying that we need to reestimate them..

In this paper, we study the problem of GSL with information theory, and propose CoGSL, a framework to learn compact graph structure with mutual information compression. Specifically, we first carefully extract two basic views from original structure as inputs, and design a view estimator to properly adjust basic views. With the estimated basic views, we propose a novel adaptive non-parameter fusion mechanism to get the final view. In this mechanism, the model will assign weights to basic views according to its predictions on nodes. If it gives a more confident prediction on one view, this view will be assigned with a larger weight. Then, we propose a formal definition *minimal sufficient structure*. And we theoretically prove that if the performances of basic views and final view are guaranteed, we need to minimize the mutual information (MI) between every two views simultaneously. To effectively evaluate the MI between different views, we deploy a MI estimator implemented based on InfoNCE loss [25]. In the end, we adopt a three-fold optimization to practically initialize the principles. Our contributions are summarized as follows:

- To our best knowledge, we are the first to utilize information theory to study the optimal structure in GSL. We propose the concept of "minimal sufficient structure", which aims to learn the most compact structure relevant to downstream tasks in principle, no more and no less, so as to provide a better balance between accuracy and robustness.
- We theoretically prove that the minimal sufficient graph structure heavily depends on modeling the relationships among different views and labels. Based on this, we propose CoGSL, a novel framework to learn compact graph structure via mutual information compression.
- We validate the effectiveness of CoGSL compared with state-of-the-art methods on seven datasets. Additionally, CoGSL also outperforms other GSL methods on attacked datasets, which further demonstrates the robustness of CoGSL.

2 RELATED WORK

Graph Neural Network. Graph neural networks (GNNs) have attracted considerable attentions recently, which can be broadly divided into two categories, spectral-based and spatial-based. Spectral-based GNNs are inheritance of graph signal processing, and define graph convolution operation in spectral domain. For example, [2] utilizes Fourier bias to decompose graph signals; [7] employs the Chebyshev expansion of the graph Laplacian to improve the efficiency. For another line, spatial-based GNNs greatly simplify above convolution by only focusing on neighbors. For example, GCN [19] simply averages information of one-hop neighbors. GraphSAGE [12] only randomly fuses a part of neighbors with various poolings.

GAT [32] assigns different weights to different neighbors. More detailed surveys can be found in [36].

Graph Structure Learning. Graph structure learning aims to estimate a better structure for original graph, which can date back to previous works in network science [13, 22]. In this paper, we mainly focus on GNN based graph structure learning models. LDS [9] jointly optimizes the probability for each node pair and GNN in a bilevel way. Pro-GNN [17] aims to obtain a clear graph by deploying some regularizers, such as low-rank, sparsity and feature smoothness. IDGL [4] casts the GSL as a similarity metric learning problem. GEN [34] presents an iterative framework based on Bayesian inference. However, these methods do not provide a theoretical view to show what the optimal structure is.

3 THE PROPOSED MODEL

In this section, we elaborate the proposed model CoGSL for GSL, and the overall architecture is shown in Fig. 1(a). Our model begins with two basic views. Then, we design a view estimator to optimize two basic views separately. With two estimated views, we propose an adaptive fusion technique to generate final view based on the confidence of predictions. Next, we formally propose the concept "minimal sufficient structure", and make a proposition to guarantee the final view to be minimal and sufficient.

3.1 Problem definition

Let $\mathcal{G} = (\mathcal{V}, \xi)$ represent a graph, where $\mathcal{V}$ is the set of N nodes and ξ is the set of edges. All edges formulate an original adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, where A_{ij} denotes the relation between nodes v_i and v_j. Graph $\mathcal{G}$ is often assigned with node feature matrix $\boldsymbol{X} = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^{N \times D}$, where x_i means the D dimensional feature vector of node i. In semi-supervised classification, we only have a small part of nodes with labels $\mathcal{Y}_L$. The traditional goal of graph structure learning for GNNs is to simultaneously learn an optimal structure and GNN parameters to boost downstream tasks.

As one typical architecture, GCN [19] is usually chosen as the backbone, which iteratively aggregates neighbors' information. Formally, the k^{th} GCN layer can be written as:

$$GCN(\boldsymbol{A}, \boldsymbol{H}^{(k)}) = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{k}, \quad (1)$$

where $\boldsymbol{D}$ is the degree matrix of $\boldsymbol{A}$, and $\boldsymbol{W}^k$ is weight matrix. $\boldsymbol{H}^{(k)}$ represents node embeddings in the k^{th} layer, and $\boldsymbol{H}^{(0)} = \boldsymbol{X}$. In this paper, we simply utilize $GCN(\boldsymbol{V}, \boldsymbol{H})$ to represent this formula, where $\boldsymbol{V}$ is some view and $\boldsymbol{H}$ is the node features or embeddings.

3.2 The selection of basic views

Given a graph G, CoGSL starts from extracting different structures. In this paper, we mainly investigate four widely-studied structures: (1) Adjacency matrix, which reflects the local structure; (2) Diffusion matrix, which represents the stationary transition probability from one node to other nodes and provides a global view of graph. Here, we choose Personal PageRank (PPR), whose closed-form solution [14] is $\boldsymbol{S} = \alpha(\boldsymbol{I} - (1-\alpha)\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2})^{-1}$, where $\alpha \in (0, 1]$ denotes teleport probability in a random walk, $\boldsymbol{I}$ is a identity matrix, and $\boldsymbol{D}$ is the degree matrix of $\boldsymbol{A}$; (3) Subgraph, which is special for large graph. We randomly keep a certain number of edges to generate a subgraph; (4) KNN graph, which reflects the similarity in feature space. We utilize original features to calculate cosine similarity between each node pair, and retain top-k similar nodes for each node to construct KNN graph.

These four views contain the different properties from various angles, and we carefully select two of them as two basic views $\boldsymbol{V_1}$ and $\boldsymbol{V_2}$, which are the inputs of CoGSL.

3.3 View Estimator

Given two basic views $\boldsymbol{V_1}$ and $\boldsymbol{V_2}$, we need to further polish them so that they are more flexible to generate the final view. Here, we devise a view estimator for each basic view, shown in Fig. 1(b). Specifically, for basic view $\boldsymbol{V_1}$, we first conduct a GCN [19] layer to get embeddings $\boldsymbol{Z^1} \in \mathbb{R}^{N \times d_{es}}$:

$$\boldsymbol{Z^1} = \sigma(GCN(\boldsymbol{V_1}, \boldsymbol{X})), \quad (2)$$

where σ is non-linear activation. With embedding $\boldsymbol{Z^1}$, probability of an edge between each node pair in $\boldsymbol{V_1}$ can be reappraised. For target node i, we concatenate its embedding $\boldsymbol{z_i^1}$ with embedding $\boldsymbol{z_j^1}$ of another node j, which is followed by a MLP layer:

$$w_{ij}^1 = \boldsymbol{W_1} \cdot [\boldsymbol{z_i^1} || \boldsymbol{z_j^1}] + b_1, \quad (3)$$

where w_{ij}^1 denotes the weight between i and j, $\boldsymbol{W_1} \in \mathbb{R}^{2d_{es} \times 1}$ is mapping vector, and $b_1 \in \mathbb{R}^{2d_{es} \times 1}$ is the bias vector. Then, we normalize the weights for node i to get the probability p_{ij}^1 between node i and other node j. Moreover, to alleviate space and time expenditure, we only estimate limited scope S^1. For example, for adjacency matrix, KNN or subgraph, we only inspect their k-hop neighbors, and for diffusion matirx, we only reestimate top-h neighbors for each node according to PPR values. Here, h and k are hyper-parameters. So, p_{ij}^1 is calculated as:

$$p_{ij}^1 = \frac{\exp(w_{ij}^1)}{\sum_{k \in S^1} \exp(w_{ik}^1)}. \quad (4)$$

In this way, we construct a probability matrix $\boldsymbol{P^1}$, where each entry is calculated by eq. (4). Combined with original structure, the estimated view is as follows:

$$\boldsymbol{V_{es}^1} = \boldsymbol{V_1} + \mu^1 \cdot \boldsymbol{P^1}, \quad (5)$$

where $\mu^1 \in (0, 1)$ is a combination coefficient, and the ith row of $\boldsymbol{V_{es}^1}$, denoted as $\boldsymbol{V_{es\_i}^1}$, shows new neighbors of node i in the estimated view. Estimating $\boldsymbol{V_2}$ is similar to $\boldsymbol{V_1}$ but with a different set of parameters, and we can get the estimated view $\boldsymbol{V_{es}^2}$ finally.

3.4 View Fusion

Then, the question we would like to answer is: given two estimated views, how can we effectively fuse them in an adaptive way for each node? We utilize the confidence of predictions as the evidence to fuse estimated views, and assign a larger weight to the more confident view. In this way, the final view can make a more confident prediction and get more effectively trained. Specifically, we first utilize two-layer GCNs to obtain predictions of each view:

$$\begin{aligned} \boldsymbol{O^1} &= softmax(GCN(\boldsymbol{V_{es}^1},\ \sigma(GCN(\boldsymbol{V_{es}^1}, \boldsymbol{X})))), \\ \boldsymbol{O^2} &= softmax(GCN(\boldsymbol{V_{es}^2},\ \sigma(GCN(\boldsymbol{V_{es}^2}, \boldsymbol{X})))), \end{aligned} \quad (6)$$

where σ is activation function, and for node i, its predictions on these two views are $\boldsymbol{o_i^1}$ and $\boldsymbol{o_i^2}$. Next, we show two cases to analyze

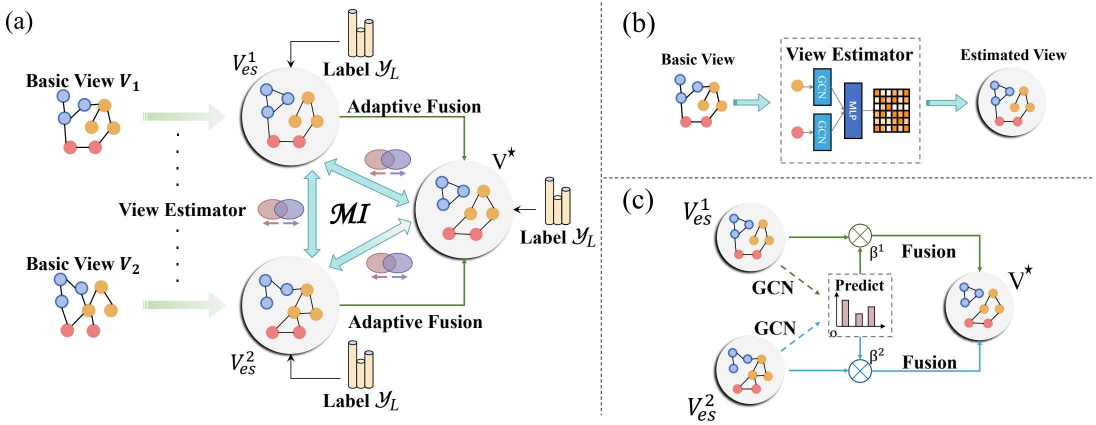


Figure 1: The overview of our proposed CoGSL. (a) Model framework. (b) View estimator. (c) Adaptive fusion.

how to assign weights to estimated views based on node predictions. In the first case, $\boldsymbol{o}_i^1$ presents a sharp distribution (e.g. [0.8, 0.1, 0.1] for three-way classification), while $\boldsymbol{o}_i^2$ is a smoother distribution (e.g. [0.4, 0.3, 0.3]). During the fusion, if we assign larger weight to $\boldsymbol{V}^2_{es\_i}$, the final view still give an uncertain result, and the model cannot be trained effectively. So in this case, we suggest to emphasize $\boldsymbol{V}^1_{es\_i}$. For the second case, predictions $\boldsymbol{o}_j^1$ and $\boldsymbol{o}_j^2$ of node j are [0.5, 0.4, 0.1] and [0.5, 0.25, 0.25], respectively. Although they have the same maximal value, there is a larger margin between maximal value and submaximal value in $\boldsymbol{o}_j^2$, so $\boldsymbol{V}^2_{es\_j}$ is a more confident view. In conclusion, if one estimated view has a higher maximal value and a larger margin between maximal value and submaximal value, it is a more confident view, which should be assigned with larger weight. With above analysis, we propose an adaptive fusion for each node, shown in Fig. 1(c). Specifically, we focus on node i to explain our fusion mechanism. First, we calculate the importance π^1 of $\boldsymbol{V}^1_{es\_i}$:

$$\pi_i^1 = e^{\epsilon\left(\lambda \log o^1_{i,m} + (1-\lambda)\log(o^1_{i,m} - o^1_{i,sm})\right)}, \tag{7}$$

where $o^1_{i,m}$ and $o^1_{i,sm}$ denote the maximal and submaximal values of prediction $\boldsymbol{o}_i^1$, ϵ and λ are hyper-parameters. The eq. (7) has three advantages: (1) If the prediction of one view has a higher maximal value and a larger margin between maximal and submaximal values, this view is prone to make confident decision, and it will lead the fusion. (2) This mechanism fully takes account of each node, so it achieves the adaptive fusion. (3) This mechanism of calculating importance does not introduce new parameters, so it alleviates over-fitting to some extent. Similarly, we can get the importance π_i^2 of $\boldsymbol{V}^2_{es\_i}$. Next, we normalize the importance and get the weights:

$$\beta_i^1 = \frac{\pi_i^1}{\pi_i^1 + \pi_i^2} \quad \textit{and} \quad \beta_i^2 = \frac{\pi_i^2}{\pi_i^1 + \pi_i^2}. \tag{8}$$

Finally, we generate the final view for node i based on weights:

$$\boldsymbol{V}_i^\star = \beta_i^1 \cdot \boldsymbol{V}^1_{es\_i} + \beta_i^2 \cdot \boldsymbol{V}^2_{es\_i}. \tag{9}$$

We likewise copy above operations to get the fusion for other nodes, and the final view $\boldsymbol{V}^\star$ is the combination of these fusion results.

3.5 Learning a minimal sufficient structure $V^\star$

3.5.1 Theoretical Motivation and Formulation. Up to now, we have discussed how to adaptively fuse basic views to generate final view $\boldsymbol{V}^\star$, which will be used for downstream tasks. The next issue is *how to guide the training of* $\boldsymbol{V}^\star$*, and what the principle should be obeyed to deal with the relations between basic views and final view?* Please review that we expect the learnt $\boldsymbol{V}^\star$ only contains the message about labels and filters superfluous noise. In other words, we seek a $\boldsymbol{V}^\star$ that is the minimal sufficient statistics [31] to label $\mathcal{Y}_L$ in information theory. The formal definition is given as follows:

Definition 1. *(Minimal Sufficient Structure) Given two variables U and V, I(U; V) means mutual information (MI), H(U) is entropy, and H(U| V) is conditional entropy. A structure* $\boldsymbol{V}^\star$ *is the minimal sufficient structure if and only if* $I(\boldsymbol{V}^\star; \mathcal{Y}_L) = H(\mathcal{Y}_L)$ *and* $H(\boldsymbol{V}^\star | \mathcal{Y}_L) = 0$.

In this definition, $I(\boldsymbol{V}^\star; \mathcal{Y}_L) = H(\mathcal{Y}_L)$ means $\boldsymbol{V}^\star$ shares all the information about $H(\mathcal{Y}_L)$, and $H(\boldsymbol{V}^\star | \mathcal{Y}_L) = 0$ guarantees that $\boldsymbol{V}^\star$ does not contain any other information except $H(\mathcal{Y}_L)$. To gain such minimal sufficient structure, we have the following proposition.

Proposition 2. *Given the estimated basic views* $\boldsymbol{V}^1_{es}$ *and* $\boldsymbol{V}^2_{es}$*, final view* $\boldsymbol{V}^\star$*, and labels* $\mathcal{Y}_L$ *related to downstream task,* $\boldsymbol{V}^\star$ *is a minimal sufficient structure to* $\mathcal{Y}_L$ *if the following two principles are satisfied:*

1. $I(\boldsymbol{V}^1_{es}; \mathcal{Y}_L) = I(\boldsymbol{V}^2_{es}; \mathcal{Y}_L) = I(\boldsymbol{V}^\star; \mathcal{Y}_L) = H(\mathcal{Y}_L)$
2. *minimize* $I(\boldsymbol{V}^1_{es}; \boldsymbol{V}^2_{es}) + I(\boldsymbol{V}^1_{es}; \boldsymbol{V}^\star) + I(\boldsymbol{V}^2_{es}; \boldsymbol{V}^\star)$

For node classification task, the first principle will build the relationships between $\boldsymbol{V}^1_{es}$, $\boldsymbol{V}^2_{es}$, $\boldsymbol{V}^\star$ and $\mathcal{Y}_L$ based on MI. In this way, the information of $\mathcal{Y}_L$ will be totally contained in $\boldsymbol{V}^1_{es}$, $\boldsymbol{V}^2_{es}$ and $\boldsymbol{V}^\star$, which makes them hold sufficient information about $\mathcal{Y}_L$. Meanwhile, we perform the second principle to constrain the shared information among views, which finally realizes a minimal $\boldsymbol{V}^\star$. Now, we prove the effect of the second principle:

Proof. At the beginning, we introduce some basic properties in information theory [6], which describe entropy $H(X)$, conditional entropy $H(Y|X)$, joined entropy $H(X, Y)$, mutual information $I(X;Y)$ and conditional mutual information $I(X;Z|Y)$.

(1) Nonnegativity:

$$H(X|Y) \geq 0; I(X;Y|Z) \geq 0$$

(2) Chain rule of entropy and MI:

$$H(X,Y) = H(X) + H(Y|X)$$

$$I(X;Y,Z) = I(X;Y) + I(X;Z|Y)$$

(3) Multivariate mutual information:

$$I(X_1;X_2;...;X_{n+1}) = I(X_1;...;X_n) - I(X_1;...;X_n|X_{n+1})$$

Then, we have the following proof:

First, we have $I(\boldsymbol{V}^{\star};\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}) > 0$, because these three views share the information of $\mathcal{Y}_L$ at least, which is guaranteed by the first principle. So, we have:

$$\begin{aligned}
& I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^{\star}) + I(\boldsymbol{V}^2_{es};\boldsymbol{V}^{\star}) \\
> & I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^{\star}) + I(\boldsymbol{V}^2_{es};\boldsymbol{V}^{\star}) - 2I(\boldsymbol{V}^{\star};\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}) \\
= & I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}) + I(\boldsymbol{V}^{\star};\boldsymbol{V}^1_{es}|\boldsymbol{V}^2_{es}) + I(\boldsymbol{V}^{\star};\boldsymbol{V}^2_{es}|\boldsymbol{V}^1_{es}) \\
= & I(\boldsymbol{V}^{\star};\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}|\boldsymbol{V}^{\star}) + I(\boldsymbol{V}^{\star};\boldsymbol{V}^1_{es}|\boldsymbol{V}^2_{es}) \\
& + I(\boldsymbol{V}^{\star};\boldsymbol{V}^2_{es}|\boldsymbol{V}^1_{es}) \\
= & I(\boldsymbol{V}^{\star};\boldsymbol{V}^1_{es}) + I(\boldsymbol{V}^{\star};\boldsymbol{V}^2_{es}|\boldsymbol{V}^1_{es}) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}|\boldsymbol{V}^{\star}) \\
= & I(\boldsymbol{V}^{\star};\boldsymbol{V}^1_{es},\boldsymbol{V}^2_{es}) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}|\boldsymbol{V}^{\star}) \\
= & H(\boldsymbol{V}^{\star}) - H(\boldsymbol{V}^{\star}|\boldsymbol{V}^1_{es}\boldsymbol{V}^2_{es}) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}|\boldsymbol{V}^{\star})
\end{aligned}$$

In the last step, $H(\boldsymbol{V}^{\star}|\boldsymbol{V}^1_{es}\boldsymbol{V}^2_{es}) = 0$. This is because $\boldsymbol{V}^{\star}$ is an adaptive combination of $\boldsymbol{V}^1_{es}$ and $\boldsymbol{V}^2_{es}$, and if $\boldsymbol{V}^1_{es}$ and $\boldsymbol{V}^2_{es}$ are known, there is no uncertainty in $\boldsymbol{V}^{\star}$. Thus, we have:

$$I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^{\star}) + I(\boldsymbol{V}^2_{es};\boldsymbol{V}^{\star}) > H(\boldsymbol{V}^{\star}) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}|\boldsymbol{V}^{\star}). \tag{10}$$

Furthermore, we can expand $H(\boldsymbol{V}^{\star})$ to $H(\boldsymbol{V}^{\star},\mathcal{Y}_L)$, because the information of $\mathcal{Y}_L$ is totally contained in $\boldsymbol{V}^{\star}$, according to the first principle. Next, we have the following derivation:

$$\begin{aligned}
& H(\boldsymbol{V}^{\star}) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}|\boldsymbol{V}^{\star}) \\
= & H(\boldsymbol{V}^{\star},\mathcal{Y}_L) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}|\boldsymbol{V}^{\star}) \\
= & H(\mathcal{Y}_L) + H(\boldsymbol{V}^{\star}|\mathcal{Y}_L) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}|\boldsymbol{V}^{\star}).
\end{aligned} \tag{11}$$

According to eq. (10) and eq. (11), we have:

$$\begin{aligned}
& I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^{\star}) + I(\boldsymbol{V}^2_{es};\boldsymbol{V}^{\star}) \\
> & H(\mathcal{Y}_L) + H(\boldsymbol{V}^{\star}|\mathcal{Y}_L) + I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}|\boldsymbol{V}^{\star}),
\end{aligned} \tag{12}$$

In inequation 12, $H(\boldsymbol{V}^{\star}|\mathcal{Y}_L) \geq 0$ and $I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}|\boldsymbol{V}^{\star}) \geq 0$ according to nonnegativity shown above. $H(\mathcal{Y}_L)$ is a constant, because the information of $\mathcal{Y}_L$ is fixed. Ideally, both of $H(\boldsymbol{V}^{\star}|\mathcal{Y}_L)$ and $I(\boldsymbol{V}^1_{es};\boldsymbol{V}^2_{es}|\boldsymbol{V}^{\star})$ equal to 0 by continuously minimizing the original formula. This means given labels $\mathcal{Y}_L$, $\boldsymbol{V}^{\star}$ does not include other information any more, and become a minimal sufficient structure. Meanwhile, $\boldsymbol{V}^1_{es}$ and $\boldsymbol{V}^2_{es}$ only share the information of $\boldsymbol{V}^{\star}$. So, $\boldsymbol{V}^1_{es}$ and $\boldsymbol{V}^2_{es}$ only share the message about $\mathcal{Y}_L$, and they will provide the most diverse knowledge for $V^{\star}$. □

3.5.2 Iterative Optimization. Based on Proposition 2, we design a three-fold optimization objective: (1) Optimize parameters Θ of classifiers for each view to improve the accuracy on $\mathcal{Y}_L$; (2) Optimize parameters Φ of MI estimator to approach the real MI value; (3) Optimize parameters Ω of view estimator to maintain classification accuracy and minimize the MI between every two views.

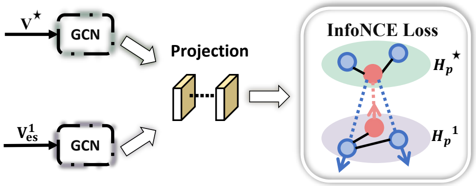


Figure 2: An illustration to show the process of MI estimator. (Take $V^{\star}$ and V^1_{es} for example)

Optimize Θ. Please recall that predictions of $\boldsymbol{V}^1_{es}$ and $\boldsymbol{V}^2_{es}$ have been obtained according to eq. (6), denoted as $\boldsymbol{O}^1$ and $\boldsymbol{O}^2$. Similarly, we also can get the predictions of $\boldsymbol{V}^{\star}$:

$$\boldsymbol{O}^{\star} = softmax(GCN(\boldsymbol{V}^{\star},\ \sigma(GCN(\boldsymbol{V}^{\star},\boldsymbol{X})))). \tag{13}$$

The parameters of GCNs involved in eq (6) and eq. (13) are regarded as the parameters Θ of classifiers together. Θ can be optimized by evaluating the cross-entropy error over $\mathcal{Y}_L$

$$\min_{\Theta} \mathcal{L}_{cls} = -\sum_{O\in\{\boldsymbol{O}^1,\boldsymbol{O}^2,\boldsymbol{O}^{\star}\}} \sum_{v_i\in\mathcal{Y}_L} y_i \ln \boldsymbol{o_i}, \tag{14}$$

where y_i is the label of node v_i, and $\boldsymbol{o_i}$ is its prediction.

Optimize Φ. In view of the second principle, we need to minimize MI values of every two views. However, estimating precise MI is hard [26]. Recently, InfoNCE [3, 15, 25] has been proved as a lower bound of real MI. If InfoNCE loss is minimized, we can approximately approach the real MI. Here, we design a corresponding MI estimator. The whole process of MI estimator is shown in Fig. 2. Specifically, for $\boldsymbol{V}^{\star}$, we first conduct one-layer GCN to get node embeddings based on $\boldsymbol{V}^{\star}$:

$$\boldsymbol{H}^{\star} = \sigma(GCN(\boldsymbol{V}^{\star},\boldsymbol{X})), \tag{15}$$

where σ is PReLU activation, X is the feature matrix.The embeddings $\boldsymbol{H}^1$ and $\boldsymbol{H}^2$ based on $\boldsymbol{V}^1_{es}$ and $\boldsymbol{V}^2_{es}$ can be obtained in a similar way. The parameters of above three GCNs are different, but $\{\boldsymbol{H}^{\star},\boldsymbol{H}^1,\boldsymbol{H}^2\}$ have the same embedding dimension. Then we deploy a shared two-layer MLP to project these embeddings into the same space where MI estimation is employed, and get the projected embeddings $\boldsymbol{H}^{\star}_p$, $\boldsymbol{H}^1_p$ and $\boldsymbol{H}^2_p$, respectively. For example, the projected embeddings $\boldsymbol{H}^{\star}_p$ is got as follows:

$$\boldsymbol{H}^{\star}_p = \boldsymbol{W}^1 \cdot \sigma(\boldsymbol{W}^0 \cdot \boldsymbol{H}^{\star} + b^0) + b^1, \tag{16}$$

where σ is non-linear activation, and $\{\boldsymbol{W}^0,\boldsymbol{W}^1,b^0,b^1\}$ are shared parameters. Then, inspired by GCA [44], we take $\boldsymbol{H}^{\star}_p$ and $\boldsymbol{H}^1_p$ for example to give the InfoNCE loss as follows:

$$\begin{aligned}
& L(\boldsymbol{V}^{\star},\boldsymbol{V}^1_{es}) \\
= & -\frac{1}{2|B|}\sum_{i=1}^{|B|}\Bigg[\log\frac{e^{sim(\boldsymbol{h}^{\star}_{\boldsymbol{p}_i},\boldsymbol{h}^1_{\boldsymbol{p}_i})/\tau}}{e^{sim(\boldsymbol{h}^{\star}_{\boldsymbol{p}_i},\boldsymbol{h}^1_{\boldsymbol{p}_i})/\tau} + \sum_{k\neq i} e^{sim(\boldsymbol{h}^{\star}_{\boldsymbol{p}_i},\boldsymbol{h}^1_{\boldsymbol{p}_k})/\tau}} \\
& + \log\frac{e^{sim(\boldsymbol{h}^1_{\boldsymbol{p}_i},\boldsymbol{h}^{\star}_{\boldsymbol{p}_i})/\tau}}{e^{sim(\boldsymbol{h}^1_{\boldsymbol{p}_i},\boldsymbol{h}^{\star}_{\boldsymbol{p}_i})/\tau} + \sum_{j\neq i} e^{sim(\boldsymbol{h}^1_{\boldsymbol{p}_i},\boldsymbol{h}^{\star}_{\boldsymbol{p}_j})/\tau}}\Bigg],
\end{aligned} \tag{17}$$

where $sim(u,v)$ is cosine similarity of vector u and v, and τ is temperature coefficient. $\boldsymbol{h}^{\star}_{\boldsymbol{p}_i}$ and $\boldsymbol{h}^1_{\boldsymbol{p}_i}$ are the projected embeddings of node i based on V_{es}^1 and V_{es}^2, respectively. B is a batch of nodes that randomly sampled. This formula means if we maximize the similarity of embeddings of the same node but from different views, while minimize the similarity with other nodes in the same batch, we can approximatively approach real MI between $V^\star$ and V_{es}^1. Similarly, we can calculate $L(V^\star, V_{es}^2)$ and $L(V_{es}^1, V_{es}^2)$, and the objective to optimize MI estimator is shown here:

$$\mathcal{L}_{MI} = L(V^\star, V_{es}^1) + L(V^\star, V_{es}^2) + L(V_{es}^1, V_{es}^2). \tag{18}$$

By minimizing the above equation, the MI estimator Φ, including parameters in eq. (15) and the following shared MLP, is well trained.

Optimize Ω**.** Given trained classifiers and MI estimator, we continuously optimize parameters Ω of view estimator. Under the guidance of proposition 2, we have the following loss:

$$\min_{\Omega} \mathcal{L}_{cls} - \eta \cdot \mathcal{L}_{MI}, \tag{19}$$

where η is a balance parameter. With this optimization, V_{es}^1 and V_{es}^2 only share the information of $V^\star$, and $V^\star$ only reserves useful information while filters noise as far as possible.

To effectively train the CoGSL, we alternatively and iteratively perform the above three-fold optimization, where a profile of the whole process is shown in appendix B. We can optimize the proposed CoGSL via back propagation with stochastic gradient descent.

4 EXPERIMENTS

4.1 Experimental Setup

Datasets We employ seven open datasets, including three academic networks (i.e., Citeseer [19], Wiki-CS [24] and MS Academic [20]), three non-graph datasets (i.e., Wine, Breast Cancer (Cancer) and Digits) available in scikit-learn [27] and a blog graph Polblogs [17]. The basic information about datasets is summarized in appendix A.1. Notice that for non-graph datasets, we construct a KNN graph as an initial adjacency matrix as in [4].

Baselines We compare the proposed CoGSL with three categories of baselines: MI based unsupervised methods {DGI [33], GCA [44]}, classical GNN models {GCN [19], GAT [32], GraphSAGE [12]} and three graph structure learning based methods {Pro-GNN [17], IDGL [4], GEN [34]}.

Implementation Details For DGI and GCA, we firstly generate node embeddings, and then evaluate embeddings following the way stated in original papers. For three classical GNN models (i.e. GCN, GAT, GraphSAGE), we adopt the implementations from PyTorch Geometric library [8]. For Pro-GNN, IDGL and GEN, we use the source codes provided by authors, and follow the settings in their original papers with carefully tune. For the proposed CoGSL, we use Glorot initialization [11] and Adam [18] optimizer. We carefully select two basic views for different datasets as two inputs, which are summarized in appendix A.3. We set the learning rate for classifiers Θ and MI estimator Φ as 0.01, and tune it for view estimator Ω from {0.1, 0.01, 0.001}. For combination coefficient μ, we test ranging from {0.1, 0.5, 1.0}. We set ϵ as 0.1 and search on λ from 0.1 to 0.9. Finally, we carefully select total iterations T from {100, 150, 200}, and tune training epochs for each fold $\{\rho_\Theta, \rho_\Phi, \rho_\Omega\}$ from {1, 5, 10}.

For fair comparisons, we set the hidden dimension as 16 and randomly run 10 times and report the average results for all methods. For Pro-GNN, IDGL, GEN and our CoGSL, we uniformly choose two-layer GCN as backbone to valuate the learnt structure. For the reproducibility, we report the related parameters in appendix A.4.

4.2 Node Classification

In this section, we evaluate the proposed CoGSL on semi-supervised node classification. For different datasets, we follow the original splits on training set, validation set and test set. To more comprehensively evaluate our model, we use three common evaluation metrics, including F1-macro, F1-micro and AUC. The results are reported in Table 1, where we randomly run 10 times and report the average results. The "-" symbol in Table 1 indicates that experiments could not be conducted due to memory issue. As can be seen, the proposed CoGSL generally outperforms all the other baselines on all datasets, which demonstrates that CoGSL can boost node classification in an effective way. The huge performance superiority of CoGSL over backbone GCN implies that view estimator and classifier are collaboratively optimized, and promote each other. In comparison with other GSL frameworks, our performance improvement illustrates that proposed principles are valid, and the learnt minimal sufficient structure with more effective information and less noise can offer a better solution.

4.3 Defense Performance

Here, we aim to evaluate the robustness of various methods. Cancer, Citeseer and Wiki-CS are adopted. We focus on comparing GSL models, because these models can adjust original structure, which makes them more robust than other GNNs. Specifically, we choose Pro-GNN from single-view based methods. And for multi-view based methods, IDGL and GEN are both selected.

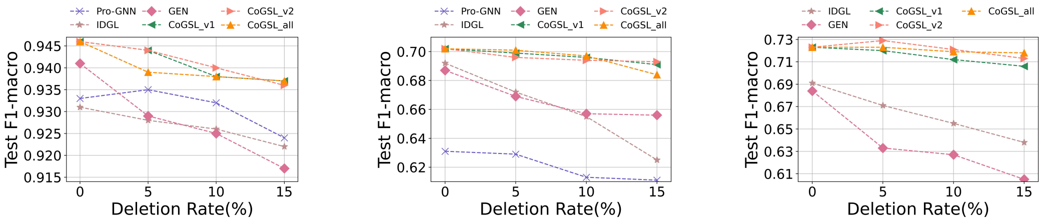
4.3.1 Attacks on edges. To attack edges, we adopt random edge deletions or additions following [4, 9]. Specifically, for edge deletions, we randomly remove 5%, 10%, 15% of original edges, which retains the connectivity of attacked graph. For edge addition, we randomly inject fake edges into the graph by a small percentages of the number of original edges, i.e. 25%, 50%, 75%. In view of that our CoGSL needs two inputs while other methods need one input, for a fair comparison, we deploy attacks on each of two inputs separately and on both of them together with the same percentages. We choose poisoning attack [35], where we firstly generate attacked graphs and then use them to train models. All the experiments are conducted 10 times and we report the average accuracy. The results are plotted in Fig. 3 and 4. Notice that we do not conduct Pro-GNN on Wiki-CS because of time consuming (more than two weeks for a result). Besides, the curves of "CoGSL_v1", "CoGSL_v2" and "CoGSL_all" mean the results that one of inputs of CoGSL is attacked and both of them are attacked, respectively.

From the figures, CoGSL consistently outperforms all other baselines under different perturbation rates by a margin for three cases. We also find that as the perturbation rate increases, the margin becomes larger, which indicates that our model is more effective with violent attack. Besides, "CoGSL_all" also performs competitive. Although both of its two inputs are attacked, "CoGSL_all" still outperforms other baselines.

4.3.2 Attacks on features. To attack feature, we add independent Gaussian noise to features as in [35]. Specifically, we firstly sample a noise matrix $M_{noise} \in \mathbf{R}^{N \times D}$, where each entry is sampled from

Table 1: Quantitative results (%±σ) on node classification.(bold: best; underline: runner-up)

Datasets	Metric	DGI	GCA	GCN	GAT	GraphSAGE	LDS	Pro-GNN	IDGL	GEN	**CoGSL**
Wine	F1-macro	93.6±0.8	94.5±2.7	94.1±0.6	93.6±0.4	96.3±0.8	93.4±1.0	97.3±0.3	96.3±1.1	96.4±1.0	**97.9±0.3**
	F1-micro	93.6±0.8	94.6±2.4	93.9±0.6	93.7±0.3	96.2±0.8	93.4±0.9	97.2±0.3	96.2±1.1	96.3±1.0	**97.8±0.3**
	AUC	99.5±0.1	97.8±1.4	99.6±0.2	97.8±0.2	99.4±0.4	99.0±0.1	99.5±0.1	99.6±0.1	99.3±0.2	**99.7±0.1**
Cancer	F1-macro	85.7±1.9	93.4±1.2	93.0±0.6	92.2±0.2	92.0±0.5	83.1±1.5	93.3±0.5	93.1±0.9	94.1±0.8	**94.6±0.3**
	F1-micro	87.6±1.4	93.8±1.2	93.3±0.5	92.9±0.1	92.5±0.5	84.8±0.8	93.8±0.5	93.6±0.9	94.3±1.0	**95.0±0.3**
	AUC	95.2±2.4	97.9±0.6	**98.9±0.1**	96.9±0.3	96.9±0.5	90.6±0.9	97.8±0.2	98.1±0.3	98.3±0.3	98.5±0.1
Digits	F1-macro	88.9±0.8	89.5±1.4	89.0±1.3	89.9±0.2	87.5±0.2	79.7±1.0	89.7±0.3	92.5±0.5	91.3±1.3	**93.3±0.3**
	F1-micro	89.0±0.8	89.6±1.5	89.1±1.3	90.0±0.2	87.7±0.2	80.2±0.9	89.8±0.3	92.6±0.5	91.4±1.2	**93.3±0.3**
	AUC	99.0±0.1	97.6±0.3	98.9±0.2	98.3±0.4	98.7±0.1	95.1±0.1	98.1±0.2	99.4±0.1	98.4±0.9	**99.6±0.0**
Polblogs	F1-macro	90.9±0.4	95.0±0.2	95.1±0.4	94.1±0.1	93.3±2.5	94.9±0.3	94.6±0.6	94.6±0.7	95.2±0.6	**95.5±0.1**
	F1-micro	90.9±0.4	95.0±0.2	95.1±0.4	94.1±0.1	93.4±2.5	94.9±0.3	94.6±0.6	94.6±0.7	95.2±0.6	**95.5±0.1**
	AUC	96.4±0.3	98.2±0.2	**98.5±0.0**	97.4±0.1	98.1±0.1	98.1±0.4	98.3±0.2	98.2±0.2	98.0±0.6	98.3±0.1
Citeseer	F1-macro	68.1±0.6	60.9±0.9	67.4±0.3	68.4±0.2	67.1±0.8	69.4±0.7	63.1±0.7	69.2±0.9	68.7±0.5	**70.2±0.6**
	F1-micro	72.1±0.6	64.5±1.1	70.1±0.2	72.2±0.2	70.1±0.7	72.2±0.7	65.6±0.8	72.6±0.6	72.5±0.8	**73.4±0.8**
	AUC	90.8±0.1	88.5±0.7	89.9±0.2	90.2±0.1	90.5±0.3	91.3±0.3	88.2±0.3	91.1±0.4	88.4±0.5	**91.4±0.5**
Wiki-CS	F1-macro	56.4±0.1	67.1±1.3	68.8±1.7	70.1±0.1	69.2±0.9	54.6±0.5	63.8±2.0	69.1±1.1	68.4±0.3	**72.3±0.6**
	F1-micro	61.2±0.2	71.3±1.3	70.8±1.8	73.8±0.3	72.2±0.7	53.7±0.5	68.3±1.2	72.7±0.8	71.1±0.9	**75.0±0.3**
	AUC	91.8±0.1	93.2±0.4	95.2±0.3	95.6±0.1	95.0±0.3	88.8±2.1	93.3±0.3	92.0±0.2	91.6±1.2	**96.4±0.2**
MS Academic	F1-macro	88.6±0.2	87.0±1.6	89.4±0.6	86.7±0.6	88.9±0.4	-	-	89.6±0.6	89.8±0.8	**90.5±0.4**
	F1-micro	91.4±0.2	89.8±1.2	91.9±0.5	89.0±0.4	91.1±0.2	-	-	91.9±0.5	92.0±0.5	**92.4±0.5**
	AUC	99.1±0.1	99.3±0.2	99.4±0.1	99.2±0.1	99.4±0.0	-	-	**99.6±0.1**	98.8±0.3	99.4±0.1

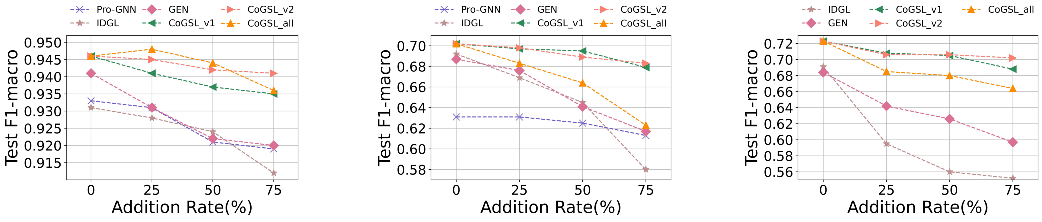


(a) Cancer

(b) Citeseer

(c) Wiki-CS

Figure 3: Results of different models under random edge deletion.



(a) Cancer

(b) Citeseer

(c) Wiki-CS

Figure 4: Results of different models under random edge addition.

$N(0,1)$. Then, we calculate reference amplitude r, which is the mean of maximal value of each node's feature. We add $\aleph \cdot r \cdot M_{noise}$ to original feature matrix X, and get the attacked feature matrix X_{noise}, where $\aleph \in \{0.1, 0.3, 0.5\}$ is the noise ratio. We also conduct poisoning settings and report the results in Table 2, where the results of Pro-GNN on Wiki-CS are not reported for the same reason in section 4.3.1. Again, CoGSL consistently outperforms all other baselines. Together with observations from 4.3.1, we can conclude that CoGSL can approach the minimal sufficient structure, so it is able to defend attacks from edges and features.

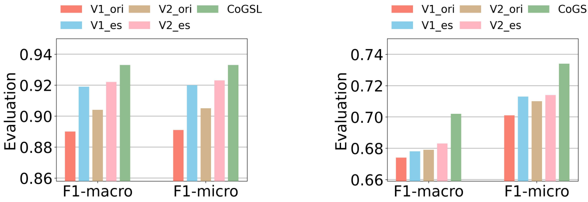
4.4 Model Analysis

4.4.1 Analysis of view estimator. Our model involves two basic views as inputs, each of which will be reestimated with the view estimator. To evaluate the effectiveness of view estimator, we firstly train the model, and pick two final estimated views. After that, we compare the performance of two original views, two final estimated views and the final view. We conduct comparison on Citeseer and Digits, and the results are given in Fig. 5, where $V1\_ori$ and $V2\_ori$ mean two original views, and $V1\_es$ and $V2\_es$ are two estimated views. We can see that all estimated views gain an improvement

Table 2: Quantitative results under feature attack.

Datasets	F1-macro	Pro-GNN	IDGL	GEN	CoGSLL
Cancer	0.0	93.3	93.1	94.1	**94.6**
	0.1	92.9	91.5	92.9	**94.2**
	0.3	92.6	90.5	91.9	**93.6**
	0.5	92.2	90.2	90.9	**93.4**
Citeseer	0.0	63.1	69.2	68.7	**70.2**
	0.1	55.5	64.1	65.3	**67.8**
	0.3	44.1	22.6	36.1	**49.1**
	0.5	36.8	23.3	29.4	**43.5**
Wiki-CS	0.0	-	69.1	68.4	**72.3**
	0.1	-	63.6	46.8	**70.4**
	0.3	-	41.6	24.2	**46.2**
	0.5	-	12.5	18.5	**24.2**

over corresponding original views, which indicates the effectiveness of view estimator. Moreover, CoGSL always outperforms the estimated views, and this proves the reliability of adaptive fusion and following optimization principles.



(a) Digits

(b) Citeseer

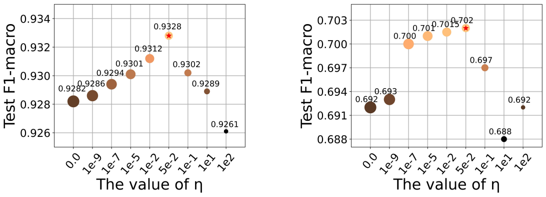
Figure 5: Test on the effectiveness of view estimator.

4.4.2 Analysis of adaptive fusion. We propose an adaptive fusion mechanism, which assigns weights to two estimated views based on the confidence on them for each node as eq. (6)-(9) in section 3.4. To verify the validation of this part, we design two more baselines. One is to simply average two estimated views as the final view. The other is to use attention mechanism to fuse them, where we adopt a channel attention layer in [41]. We test on Citeseer and Digits and show the results in Table 3, where "Adaption" refers to adaptive fusion we introduce. We can see that our newly proposed adaptive fusion is the best behaved of three ways. Also, we notice that "Average" behaves better than "Attention", and we think this is because "Attention" fusion involves some new parameters, which increases the complexity of model and brings the risk of over-fitting.

Table 3: Quantitative results on different fusions.

	Digits			Citeseer		
Fusion	F1-ma	F1-mi	AUC	F1-ma	F1-mi	AUC
Average	93.0	93.0	99.5	69.6	72.8	90.8
Attention	92.9	93.0	99.6	69.4	72.7	91.2
Adaption	**93.3**	**93.3**	**99.6**	**70.2**	**73.4**	**91.4**

4.4.3 Analysis of MI. We need to constrain the MI between views are neither too weak or too strong, so that the final view contain concise information, no more and no less. We notice that as a balance parameter, η in eq. (19) well controls the effect of MI loss. If η increases, MI between views is heavily constrained, and vice versa. So, we investigate the change of η to substitute the change of real MI between views, and the results are shown in Fig. 6, where we report the results on Citeseer and Digits. In this figure, the area of each point means relative size of MI between views. The shallower the color of point is, the better the performance is. And the best point is marked with a red star. We observe that the optimal point is a medium value, neither a too strong or a too weak constraint. Especially, when η equals to zero, we mimic the situation of general GSL methods, and we can see that the results are not very good in this case. It implies that restricting MI between views is necessary.

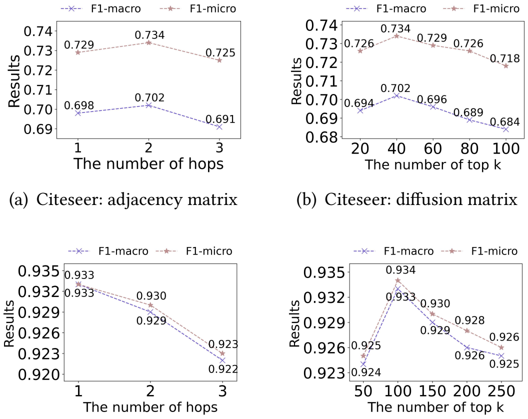


(a) Digits

(b) Citeseer

Figure 6: The investigation of change of MI.

4.4.4 Analysis of hyper-parameter. In this section, we explore the sensitivity of h and k on Citeseer and Digits, introduced in section 3.3. As shown in appendix A.3, the input views of Citeseer and Digits are both adjacency matrix and diffusion matrix, plotted in Fig. 7. For Digits, the optimal h of adjacency matrix is 1-hop, and the optimal k of diffusion matrix is top 100. Similarly, for Citeseer, the optimal points are 2-hop and top 40. We can see that a proper estimation scope is indispensable. If the scope is too small, some important structures are neglected. However if the scope is too large, we can not distinguish the right connections effectively.



(a) Citeseer: adjacency matrix

(b) Citeseer: diffusion matrix

(c) Digits: adjacency matrix

(d) Digits: diffusion matrix

Figure 7: Impact of hyper-parameter scope.

5 CONCLUSION

In this paper, we theoretically study how to estimate a minimal sufficient structure in GSL problem. We prove that if the performances of basic views and final view are maintained, the mutual information between every two views should be minimized simultaneously, so that the learnt final view tends to be minimal sufficient structure. With this theory, we propose CoGSL, a framework to learn a compact graph structure by compressing mutual information. Extensive experimental results, under clean and attacked conditions, are conducted to verify the effectiveness and robustness of CoGSL.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, 62192784, 62172052, 61772082, 62002029, U1936104). It is also supported in part by The Fundamental Research Funds for the Central Universities 2021RC28.

REFERENCES

[1] Mohammed AlQuraishi. 2019. AlphaFold at CASP13. *Bioinformatics* (2019), 4862–4865.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *2nd International Conference on Learning Representations*.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*. 1597–1607.

[4] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *NeurIPS* (2020).

[5] Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2019. Reinforcement learning based graph-to-sequence model for natural question generation. *arXiv preprint arXiv:1908.04942* (2019).

[6] Thomas M Cover, Joy A Thomas, et al. 1991. Entropy, relative entropy and mutual information. *Elements of information theory* (1991), 12–13.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*. 3837–3845.

[8] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

[9] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *ICML*. 1972–1982.

[10] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*. 1263–1272.

[11] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.

[12] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[13] Mark S Handcock and Krista J Gile. 2010. Modeling social networks from sampled data. *The Annals of Applied Statistics* (2010), 5.

[14] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *ICML*. 4116–4126.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*. 9729–9738.

[16] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. 2019. Semi-supervised learning with graph learning-convolutional networks. In *CVPR*. 11313–11320.

[17] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *SIGKDD*. 66–74.

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[20] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.

[21] Hu Linmei, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous graph attention networks for semi-supervised short text classification. In *EMNLP-IJCNLP*. 4821–4830.

[22] Dean Lusher, Johan Koskinen, and Garry Robins. 2013. *Exponential random graph models for social networks: Theory, methods, and applications*. Vol. 35. Cambridge University Press.

[23] Peter V Marsden. 1990. Network data and measurement. *Annual review of sociology* (1990), 435–463.

[24] Péter Mernyei and Cătălina Cangea. 2020. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901* (2020).

[25] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[26] Liam Paninski. 2003. Estimation of entropy and mutual information. *Neural computation* (2003), 1191–1253.

[27] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* (2011), 2825–2830.

[28] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*.

[29] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 2017. 3d graph neural networks for rgbd semantic segmentation. In *ICCV*. 5199–5208.

[30] Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427* (2017).

[31] Stefano Soatto and Alessandro Chiuso. 2016. Modeling Visual Representations: Defining Properties and Deep Approximations. In *ICLR*.

[32] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[33] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341* (2018).

[34] Ruijia Wang, Shuai Mou, Xiao Wang, Wanpeng Xiao, Qi Ju, Chuan Shi, and Xing Xie. 2021. Graph Structure Estimation Neural Networks. In *WWW*. 342–353.

[35] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. 2020. Graph information bottleneck. *arXiv preprint arXiv:2010.12811* (2020).

[36] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* (2021), 4–24.

[37] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.

[38] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*. 5449–5458.

[39] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware graph neural networks. In *ICML*. 7134–7143.

[40] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *NeurIPS* (2018), 5165–5175.

[41] Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. 2021. Heterogeneous Graph Structure Learning for Graph Neural Networks. In *AAAI*.

[42] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust graph representation learning via neural sparsification. In *ICML*. 11458–11468.

[43] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. 2021. Deep Graph Structure Learning for Robust Representations: A Survey. *arXiv preprint arXiv:2103.03036* (2021).

[44] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *WWW*. 2069–2080.

A DETAILS ON EXPERIMENTAL SETUP

In this section, for the reproducibility, we provide some basic information about baselines and datasets. The implementation details, including the detailed hyper-parameter values, are also provided.

A.1 Datasets

Table 4 shows the statistics of seven datasets used in our experiments.

Table 4: The statistics of the datasets

Dataset	Nodes	Edges	Classes	Features	Train/Val/Test
Wine	178	3560	3	13	10/20/148
Cancer	569	22760	2	30	10/20/539
Digits	1797	43128	10	64	50/100/1647
Polblogs	1222	33428	2	1490	121/123/978
Citeseer	3327	9228	6	3703	120/500/1000
Wiki-CS	11701	291039	10	300	200/500/1000
MS Academic	18333	163788	15	6805	300/500/1000

These seven datasets used in experiments can be found in these URLs:

- Wine, Breast Cancer and Digits: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets
- Polblogs: https://github.com/ChandlerBang/Pro-GNN
- Citeseer: https://github.com/tkipf/gcn
- Wiki-CS: https://github.com/pmernyei/wiki-cs-dataset
- MS Academic: https://github.com/klicperajo/ppnp

A.2 Baselines

The publicly available implementations of baselines can be found at the following URLs:

- DGI: https://github.com/PetarV-/DGI
- GCA: https://github.com/CRIPAC-DIG/GCA
- GCN, GAT and GraphSAGE: https://pytorch-geometric.readthedocs.io/en/latest/
- LDS: https://github.com/lucfra/LDS-GNN
- Pro-GNN: https://github.com/ChandlerBang/Pro-GNN
- IDGL: https://github.com/hugochan/IDGL
- GEN: https://github.com/BUPT-GAMMA/Graph-Structure-Estimation-Neural-Networks

A.3 The selected input views

Table 5 shows the basic views we select for different datasets.

Table 5: The selected views for different datasets

Candidate	Wine	Cancer	Digits	Polblogs	Citeseer	Wiki-CS	MS Academic
Adjacency matrix (A)			√	√	√	√	√
Diffusion matrix (S)	√	√	√	√	√		
KNN graph (K)	√	√					
Subgraph (A_{sub})						√	√

A.4 Hyperparameters Settings

We implement CoGSL in PyTorch, and list some important parameter values used in our model in Table 6. In this table, *ve_lr* is the learning rate of view estimator, and *ve_drop* is the dropout used in estimating basic views. Notice that "-" of B indicates that we use all of nodes to calculate InfoNCE loss.

Table 6: The values of parameter used in CoGSL.

Dataset	*ve_lr*	*ve_drop*	T	ρ_Θ	ρ_Φ	ρ_Ω	B	ϵ	λ
Wine	0.001	0.8	100	1	5	1	-	0.1	0.5
Cancer	0.1	0.5	150	1	5	1	-	0.1	0.9
Digits	0.01	0.5	200	10	10	1	-	0.1	0.5
Polblogs	0.1	0.8	150	5	5	1	-	0.1	0.1
Citeseer	0.001	0.2	200	5	10	5	-	0.1	0.5
Wiki-CS	0.01	0.2	200	1	5	1	1000	0.1	0.1
MS Academic	0.0001	0.8	200	15	10	1	1000	1.0	0.2

B THREE-FOLD OPTIMIZATION

In this section, we detail the process of three-fold optimization, shown in Algorithm 1.

Algorithm 1: The CoGSL Algorithm

```
Input  : Basic views $\{V_1, V_2\}$, feature matrix $\mathbf{X}$, labels $\mathcal{Y}_L$
Params : $B$, total iterations $T$,
         training epochs for each fold $\{\rho_\Theta, \rho_\Phi, \rho_\Omega\}$
Output : final view $V^\star$, GCN parameters $\Theta$
Initialize $\Theta$, $\Phi$ and $\Omega$;
for $i = 1$ to $T$ do
    for $j = 1$ to $\rho_\Omega$ do
        % View estimator Training
        Estimate $\{V_1, V_2\}$ as $\{V_{es}^1, V_{es}^2\}$ with eq (2)- (5);
        Adaptively fuse $V_1$ and $V_2$ into $V^\star$;
        Update $\Omega$ with eq. (19);
    end
    Get $\{V_{es}^1, V_{es}^2, V^\star\}$ after view estimating and fusion;
    for $k = 1$ to $\rho_\Theta$ do
        % Classifiers Training
        Calculating predictions with eq. (6) and (13);
        Update $\Theta$ with eq. (14);
    end
    for $l = 1$ to $\rho_\Phi$ do
        % MI estimator Training
        Randomly sample B nodes to calculate eq. (17);
        Update $\Phi$ by minimizing eq. (18);
    end
end
return $V^\star$ and $\Theta$;
```