Journal Pre-proof


GMNI: Achieve good data augmentation in unsupervised graph contrastive learning

Xin Xiong, Xiangyu Wang, Suorong Yang, Furao Shen, Jian Zhao

PII: S0893-6080(24)00728-7
DOI: https://doi.org/10.1016/j.neunet.2024.106804
Reference: NN 106804

To appear in: *Neural Networks*

Received date : 23 May 2024
Revised date : 22 September 2024
Accepted date : 10 October 2024

Please cite this article as: X. Xiong, X. Wang, S. Yang et al., GMNI: Achieve good data augmentation in unsupervised graph contrastive learning. *Neural Networks* (2024), doi: https://doi.org/10.1016/j.neunet.2024.106804.

# GMNI: Achieve good data augmentation in unsupervised graph contrastive learning

Xin Xiong[a,b], Xiangyu Wang[a,b], Suorong Yang[a,c], Furao Shen[a,b,*], Jian Zhao[d]

[a] *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023, China*
[b] *School of Artificial Intelligence, Nanjing University, Nanjing, 210023, China*
[c] *Department of Computer Science and Technology, Nanjing University, Nanjing, 210023, China*
[d] *School of Electronic Science and Engineering, Nanjing University, Nanjing, 210023, China*

**Abstract**

Graph contrastive learning (GCL) shows excellent potential in unsupervised graph representation learning. Data augmentation (DA), responsible for generating diverse views, plays a vital role in GCL, and its optimal choice heavily depends on the downstream task. However, it is impossible to measure task-relevant information under an unsupervised setting. Therefore, many GCL methods risk insufficient information by failing to preserve essential information necessary for the downstream task or risk encoding redundant information. In this paper, we propose a novel method called Minimal Noteworthy Information for unsupervised Graph contrastive learning (GMNI), featuring automated DA. It achieves good DA by balancing missing and excessive information, approximating the optimal views in contrastive learning. We employ an adversarial training strategy to generate views that share minimal noteworthy information (MNI), reducing nuisance information by minimization optimization and ensuring sufficient information by emphasizing noteworthy information. Besides, we introduce randomness based on MNI to augmentation, thereby enhancing view diversity and stabilizing the model against

*Corresponding author
*Email addresses:* xiongxin@smail.nju.edu.cn (Xin Xiong), xiangyuwang@smail.nju.edu.cn (Xiangyu Wang), sryang@smail.nju.edu.cn (Suorong Yang), frshen@nju.edu.cn (Furao Shen), jianzhao@nju.edu.cn (Jian Zhao)

perturbations. Extensive experiments on unsupervised and semi-supervised learning over 14 datasets demonstrate the superiority of GMNI over GCL methods with automated and manual DA. GMNI achieves up to a 1.64% improvement over the state-of-the-art in unsupervised node classification, up to a 1.97% improvement in unsupervised graph classification, and up to a 3.57% improvement in semi-supervised graph classification.

*Keywords:* Graph neural network, Graph contrastive learning, Data augmentation

---

## 1. Introduction

Computer technology is successfully applied in various real-world scenarios (Devlin, 2018; Zhou et al., 2024; Fan et al., 2024). Most real-world data, such as traffic networks, social networks, and protein structures, are unstructured and can be abstracted as graphs. Consequently, research on graphs has gained significant attention in recent years. However, labeling graphs is a very challenging and laborious task since it generally requires domain knowledge, and graphs usually have numerous nodes with complex relationships. Thus, unsupervised graph representation learning (Ju et al., 2023; Liu et al., 2023; Mo et al., 2022) has gained significant attention recently, which aims to obtain low-dimensional representations of nodes or graphs without label information. These representations can be used for a wide range of downstream tasks, such as node classification (Xiao et al., 2022), graph classification (Sun et al., 2019), and graph clustering (Pan and Kang, 2021). Graph contrastive learning (GCL) (Lin et al., 2023; Chen and Kou, 2023; Gong et al., 2023; You et al., 2021) shows great potential in unsupervised graph representation learning due to its excellent data efficiency and generalization ability. GCL generally includes two sequential modules: view generation and view comparison. View generation generates views by employing data augmentation (DA) on the original graph. View comparison acquires view representations through an encoder and then optimizes the encoder by pulling view representations from the same distribution closer while pushing away view representations from different distributions.

Nevertheless, GCL heavily depends on DA, and an inappropriate augmentation will lead to severe performance loss. Graph data augmentation (GDA) necessitates the consideration of both the complex graph topology and feature information. Various GDA techniques have been proposed, which can be categorized into three categories based on their augmentation modality (Ding et al., 2022): structure-oriented (Mishra et al., 2020; Velickovic et al., 2019; Zhu et al., 2021; Page et al., 1999; Kondor and Lafferty, 2002), feature-oriented (Feng et al., 2019; Yang et al., 2021; You et al., 2020a), and label-oriented (You et al., 2020b; Park et al., 2022). Ensuring the selection of an appropriate GDA technique is critical in GCL (You et al., 2020a), prompting many GCL methods to rely on trial-and-error or empirical approaches when choosing GDAs. GraphCL (You et al., 2020a) introduces four GDA techniques, and demonstrates that good GDA relies on specific characteristics of various graph datasets through pairwise combinations of these GDAs. MERIT (Jin et al., 2021) achieves GDA by superimposing the methods from its GDA pool. GCA (Zhu et al., 2021) conducts GDA by employing three centralities and selecting the optimal one according to its performance on the downstream task. On the one hand, these methods require a prudent design of the GDA pool. On the other hand, selecting the best GDA according to performance on the downstream task of various datasets is entailed, which is both time-consuming and computationally expensive. Thus, automated GDA is of significant importance.

The mutual information maximization principle (InfoMax) (Linsker, 1988) is widely used in GCL (Velickovic et al., 2019; Zhu et al., 2020; Thakoor et al., 2021), which refers to maximizing the agreement (mutual information, MI) between view representations. Through InfoMax, the contrastive model can identify the pairs augmented from the same node or graph, thereby learning the basic topology and feature information on the graph. However, InfoMax may risk the model learning redundant information irrelevant to the downstream task, though this information is beneficial for identifying the pairs. Encoding redundant information results in brittle representations and may severely degrade the encoder's performance on the downstream task (Tschannen et al., 2019). For example, when training an optical character recognition model, the color information is redundant and harmful to recognition, and should be removed beforehand. Thus, before implementing InfoMax, redundant information shared between views should be stripped out. Now the new question is *what kind of views InfoMax needs.*

The InfoMin principle (Tian et al., 2020) in the visual domain indicates that a good set of views should share the minimal information necessary to perform well at the downstream task, and we name such information *minimal necessary information*. It points out that information between views should contain the necessary information needed for the downstream task while excluding nuisance information (Tian et al., 2020). Indeed, GCL methods with manual DA essentially search in the DA pool for augmentations that yield views to best satisfy the InfoMin principle. However, many GCL methods with automated DA fail to follow InfoMin fully. AutoGCL (Yin et al., 2022), AD-GCL (Suresh et al., 2021), and JOAO (You et al., 2021) adhere to the min-max optimization framework, seeking views with minimal similarity or seeking views that are the most challenging. Unfortunately, the views derived by minimizing the similarity between views or by minimizing the agreement between view representations are far from optimal. These approaches reduce nuisance information but do not emphasize the retention of the necessary information, potentially resulting in excessive information removal and risking the model suffering from insufficient information. Therefore, applying InfoMin to acquire optimal views becomes the key: removing redundant information while maintaining necessary information related to the downstream task. However, it is impossible to measure task-relevant information under an unsupervised setting. Thus, it is hard to define minimal necessary information in InfoMin. For instance, InfoGCL (Cheng et al., 2022) and LP-A3 (Yang et al., 2022) rely on task-specific labels $Y$ to guide the acquisition of views. Limited research has been carried out on achieving optimal views in InfoMin under an unsupervised setting. Nevertheless, we find minimal information that is noteworthy for different downstream tasks can approximate minimal necessary information, which avoids requiring task-relevant information. We name such information *minimal noteworthy information* (MNI).

In this paper, we propose a novel method called Minimal Noteworthy Information for unsupervised Graph contrastive learning (GMNI). Specifically, GMNI uses max-min optimization to learn an importance graph. The set of views derived by the importance graph shares MNI, making these views approximations of InfoMin's optimal views. In addition, GMNI introduces randomness based on MNI to views, thereby enhancing view diversity and stabilizing the model against perturbations. Finally, an encoder is optimized by applying InfoMax to view representations. Moreover, GMNI achieves outstanding performances without using corrupted views, showing that negative views are unnecessary for GCL.

The significant contributions of this paper are summarized as follows:

- Manual data augmentation GCL methods are both time-consuming and computationally expensive, while other automated data augmentation

GCL methods risk insufficient information. In this paper, a novel unsupervised GCL method with automated data augmentation, GMNI, is proposed, in which the tedious manual selection of data augmentation is eliminated and avoids the risks of both insufficient and redundant information.

- The InfoMin principle defines optimal views in contrastive learning; however, these views depend on downstream tasks, making them unattainable in unsupervised settings. In this paper, we achieve good DA in unsupervised GCL, presenting a more effective approach to approximating the optimal views of InfoMin under an unsupervised setting. An adversarial training strategy is employed to generate views that share minimal noteworthy information, which avoids using task-relevant information. Applying InfoMax to these views is risk-free, as they reduce nuisance information and emphasize noteworthy information to ensure sufficient information.

- Extensive experiments on node and graph classification over 14 datasets show the effectiveness of our approximately optimal views on different tasks and the superiority of GMNI over GCL methods with automated and manual DA.

## 2. Related work

Matrix-factorization-based (Ahmed et al., 2013; Cao et al., 2015; Ou et al., 2016) and random-walk-based methods (Perozzi et al., 2014; Grover and Leskovec, 2016) are classical approaches for unsupervised graph representation learning. However, these methods may emphasize topology excessively while neglecting feature information. Deep unsupervised graph representation learning (Zhou et al., 2022; Oyallon, 2020) has gradually evolved in recent years, with GCL emerging as a promising method.

InfoMax (Linsker, 1988) is one of the most commonly used principles in GCL. InfoMax learns the encoder via maximizing the mutual information between view representations, i.e., $\max_f I(f(V_1); f(V_2))$, where $V_1$, $V_2$ denote the views, and $f$ denotes the encoder. $I(a; b)$ is the mutual information between $a$ and $b$. Through InfoMax, the contrastive model can discern pairs augmented from the same node or graph, facilitating the learning of fundamental topology and feature information within the graph. DIM (Hjelm et al., 2018) learns by maximizing the mutual information between the input

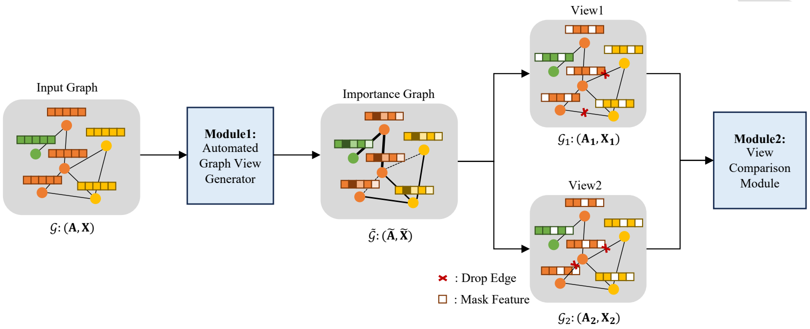



Figure 1: Overview of GMNI, with the automated graph view generator and view comparison modules as the main components.

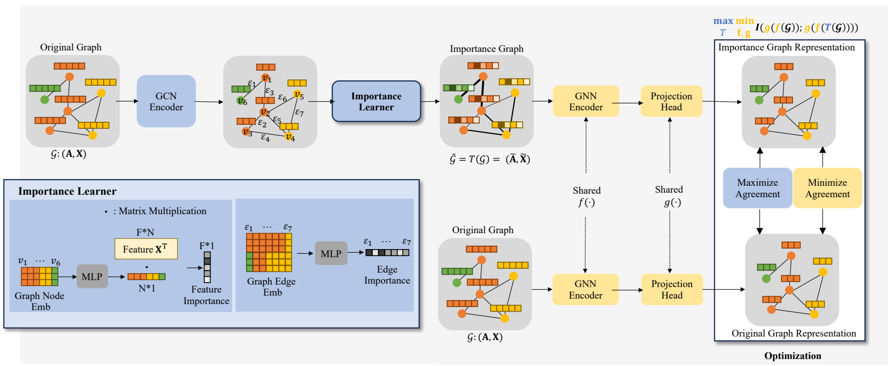



Figure 2: Overview of the automated graph view generator. In the importance graph, darker features and thicker edges are more significant. The original graph is fed into the GCN encoder and importance learner to generate an importance graph. After acquiring representations of the importance and original graph using a shared GNN encoder and projection head, max-min optimization is performed. “Maximize Agreement” optimizes the blue GCN encoder and importance learner, while “Minimize Agreement” optimizes the yellow GNN encoder and projection head.

and the output of an encoder. DGI (Velickovic et al., 2019) follows DIM, applying InfoMax to graph data for the first time. It uses node shuffling to corrupt the original graph to generate negative pairs, and the positive and negative pairs are distinguished through a discriminator optimized by InfoMax. GRACE (Zhu et al., 2020) generates two views through the corruption of removing edges and masking node features and then learns representations by applying InfoMax. BGRL (Thakoor et al., 2021) generates views through stochastic node feature masking and edge masking. It learns by maximizing the cosine similarity of the view representations at different stages, essentially applying InfoMax. Moreover, BGRL asserts that a corruption method is unnecessary for contrastive learning (CL). All the above works involve manually selecting DA or corruption method. GCL with manual DA demands significant time and computational resources to identify an appropriate DA strategy. Therefore, our work focuses on GCL with automated DA.

JOAO (You et al., 2021) searches for the best combination of DAs from a fixed pool, with the combination coefficients determined by min-max optimization. However, designing a proper DA pool still involves human knowledge, and optimizing only the combination coefficient limits DA's flexibility. AD-GCL (Suresh et al., 2021) augments graphs via adversarial training following the information bottleneck (IB) principle (Tishby et al., 2000; Goldfeld and Polyanskiy, 2020; Alemi et al., 2016), hence reducing redundant information shared between views. However, AD-GCL only augments on edges but ignores critical feature information. In InfoGCL (Cheng et al., 2022), DA is decided mathematically by minimizing mutual information between views and maximizing mutual information between views and tasks, which essentially follows IB. InfoGCL relies on task-specific labels $Y$ to guide the generation of optimal view augmentations, the view encoder, and the contrastive mode. A manually designed DA pool is entailed in InfoGCL too, and the best augmentation is acquired by exhaustive search, which is time-consuming and computationally expensive. In fact, IB and InfoMin are closely related. IB aims to simultaneously minimize $I(X;Z)$ and maximize $I(Z;Y)$, where $X$ represents the input, $Z$ is the hidden layer output, and $Y$ denotes the downstream task information. IB states that the encoder should minimize the information in the original data while maximizing information relevant to downstream tasks. AutoGCL (Yin et al., 2022) uses two learnable view generators, each of which learns a probability distribution over the nodes of the input graph. It minimizes the similarity between two views and maximizes the agreement between the representations of the two views.

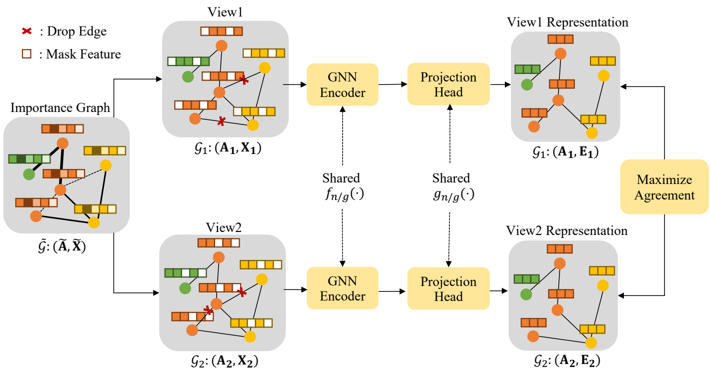



Figure 3: Overview of the view comparison module. Two views are generated by masking features and dropping edges according to the importance graph. Representations of the views are calculated through a shared GNN encoder and projection head. If the downstream task is graph classification, the node representations need to pass through a readout function to get graph representations.

Nevertheless, it merely executes node-level view learning without considering edges.

The above GCL methods with automated DA, including JOAO, AD-GCL, and AutoGCL, adopt a min-max training strategy. They fail to adhere to InfoMin fully, and their information reduction may lead to a potential deficiency in essential information. InfoMin, which seeks the optimal views, states that $(V_1^*, V_2^*) = \text{argmin}_{V_1,V_2} I(V_1; V_2)$, subject to $I(V_1; Y) = I(V_2; Y) = I(X; Y)$. Tian et al. (Tian et al., 2020), the authors of InfoMin, leverage such a min-max training strategy to complete the unsupervised view learning of InfoMin too. Given image $X$, the transformed image $\hat{X} = g(X)$ and two encoders $f_1, f_2$, the objective is: $\min_g \max_{f_1,f_2} I(f_1(g(X)_1); f_2(g(X)_{2:3}))$. $\{g(X)_1, g(X)_{2:3}\}$ represent the split channels of $g(X)$ and thus serve as the two views $V_1, V_2$. The authors mention that this strategy heavily breaks constraint $I(V_1; Y) = I(V_2; Y) = I(X; Y)$, and $I(V_1; V_2)$ is overly reduced. Therefore, a better approximation of InfoMin under an unsupervised setting, is proposed in this paper.

## 3. Methodology

GMNI consists of two sequential modules, the *automated graph view generator* and the *view comparison module*. The automated graph view generator acquires an importance graph used to generate two views. These two views are fed into the view comparison module. Figure 1 illustrates how the two modules cooperate, while Figure 2 and 3 depict their structures.

*Notations.* A graph $\mathcal{G}$ consists of $\mathcal{V} = \{v_1, ..., v_N\}$, a set of nodes and $\mathcal{E} = \{\varepsilon_1, ..., \varepsilon_M\}$, a set of edges. The feature matrix, adjacency matrix and degree matrix of $\mathcal{G}$ are denoted by $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\} \in \mathbb{R}^{N\times F}$, $\boldsymbol{A} \in \{0,1\}^{N\times N} : \boldsymbol{A}_{ij} = \mathbb{I}((v_i, v_j) \in \mathcal{E})$ and $\boldsymbol{D}$: $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$, where $F$ is the feature dimension and $\boldsymbol{x}_i \in \mathbb{R}^F$ is the feature of node $v_i$. We use $\mathcal{G} : (\boldsymbol{A}, \boldsymbol{X})$ to represent the graph. GMNI achieves the approximation of the InfoMin principle without task-relevant information. To verify the effectiveness of such an approximation strategy for different downstream tasks, we discuss two downstream tasks, node classification and graph classification. For node classification, whose input is a single graph $\mathcal{G}$, we aim to learn an encoder $f_n : \mathcal{G} \rightarrow \mathbb{R}^{N\times d}$ to obtain the low-dimensional node embeddings $f_n(\mathcal{G}), d \ll F$. For graph classification, whose input is a set of graphs $\{\mathcal{G}_i\}_{i=1}^{Q}$, we aim to learn an encoder $f_g : \mathcal{G}_i \rightarrow \mathbb{R}^d$ to obtain the low-dimensional graph embeddings $f_g(\mathcal{G}_i), i = 1, ..., Q$.

### *3.1. Automated graph view generator*

In this section, we present the mathematical motivation for the automated graph view generator and then its implementation details.

The InfoMin principle indicates that a good set of views in contrastive learning should share the minimal information necessary to perform well at the downstream task. However, applying the InfoMin principle under an unsupervised setting is challenging due to the inability to quantify task-relevant information. Therefore, we propose focusing on views that share the minimal noteworthy information for different downstream tasks as an alternative to the unavailable minimal necessary information. This approach effectively aligns with the InfoMin principle without requiring task-relevant data. Next, we provide the definitions of minimal noteworthy information and the importance graph, with minimal noteworthy information being derived from the importance graph.

**Definition 1** (Importance Graph)**.** *The importance graph* $\tilde{\mathcal{G}} = T(\mathcal{G}) = (\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{X}})$*, where* $T(\cdot)$ *denotes the function that introduces feature and edge im-*

*portance to the original graph. The edge and feature importance are represented by* $\boldsymbol{P}_e = \{p_{e,1}, ..., p_{e,M}\} \in \mathbb{R}^{M\times 1}$ *and* $\boldsymbol{P}_f = \{p_{f,1}, ..., p_{f,F}\} \in \mathbb{R}^{F\times 1}$*, respectively. The adjacency matrix* $\tilde{\boldsymbol{A}}$ *of* $\tilde{\mathcal{G}}$ *is formed by replacing the edge values in* $\boldsymbol{A}$ *with those from* $\boldsymbol{P}_e$*. Broadcast* $\boldsymbol{P}_f \in \mathbb{R}^{F\times 1}$ *to* $\boldsymbol{P}'_f \in \mathbb{R}^{N\times F}$*. The feature matrix* $\tilde{\boldsymbol{X}}$ *of* $\tilde{\mathcal{G}}$ *is derived as* $\tilde{\boldsymbol{X}} = \boldsymbol{X} \odot \boldsymbol{P}'_f$*, where* $\odot$ *is the Hadamard product.*

**Definition 2** (Minimal Noteworthy Information)**.** *Graph information includes both topology and features. Therefore, minimal noteworthy information (MNI) consists of critical edges and features. For a graph* $\mathcal{G}$ *and its corresponding importance graph* $\tilde{\mathcal{G}}$*, MNI refers to the minimal subset of edges and features from* $\mathcal{G}$ *that exhibit high importance in* $\tilde{\mathcal{G}}$*, precisely edges with high* $p_{e,i}$ *and features with high* $p_{f,j}$*.*

#### *3.1.1. Importance graph formulation*

The importance graph is derived via a max-min optimization,

$$\begin{aligned} &\max_{T} \min_{f,g} I(g(f(\mathcal{G})); g(f(T(\mathcal{G})))) - |T(\mathcal{G})|, \\ &\text{s.t.}\, I(g(f(\mathcal{G})); g(f(T(\mathcal{G})))) \geq \zeta, \end{aligned} \tag{1}$$

where $f$ is a graph encoder for node embeddings, and $g$ is a projection head to increase the ability of expression. It should be mentioned that $f$ and $g$ are not the encoders of the view comparison module. The regularization term $|T(\mathcal{G})|$ is the normalized sum of edge importance and feature importance. $I(g(f(\mathcal{G})); g(f(T(\mathcal{G})))) \geq \zeta$ is introduced to prevent the degeneration of $g(f(\cdot))$ into the corner case, i.e., $I(g(f(\mathcal{G})); g(f(T(\mathcal{G})))) = 0$. Once the optimization reaches the lower bound $\zeta$, we cease the optimization for both $f$ and $g$. However, identifying a corner case of $g(f(\cdot))$ degeneration presents a challenge for the optimizer. Experiments indicate that the degeneration of $g(f(\cdot))$ does not occur.

**Theorem 1.** *Eqn. (1) is equivalent to the reduction of* $I(\mathcal{G}; T(\mathcal{G})|\mathcal{I})$ *and the increase of* $I(T(\mathcal{G}); \mathcal{I})$*, where* $\mathcal{I}$ *represents the noteworthy information.*

**Proof 1.** *Eqn. (1) involves both maximization and minimization; we first introduce the maximization part, i.e.,*

$$\max_{T} I(g(f(\mathcal{G})); g(f(T(\mathcal{G})))) - |T(\mathcal{G})|. \tag{2}$$

$\max_T I(g(f(\mathcal{G})); g(f(T(\mathcal{G}))))$ *is achieved by setting* $T(\mathcal{G}) = \mathcal{G}$*. However, since* $-|T(\mathcal{G})|$ *serves as a regularization term, it effectively constrains the number of emphasized edges and features.*

*Consequently, to maximize the mutual information between the importance graph* $T(\mathcal{G})$ *and the original graph* $\mathcal{G}$*, it becomes imperative for* $T(\mathcal{G})$ *to retain the most vital information within its confined information selectively.*

*This is a scenario of the greedy algorithm. For clarity, we simplify the problem. Given a set* $\mathcal{G}$ *with* $|\mathcal{G}|$ *items, each associated with a value* $g_i$ *for* $i = 1, \ldots, |\mathcal{G}|$*, the objective is to select a subset* $T(\mathcal{G})$ *of size* $|T(\mathcal{G})|$ *that maximizes the total value. The solution is to select the* $|T(\mathcal{G})|$ *items with the highest values* $g_i$*. Suppose we choose a subset* $T(\mathcal{G})'$ *of size* $|T(\mathcal{G})|$ *that does not contain the* $|T(\mathcal{G})|$ *highest values. By replacing items in* $T(\mathcal{G})'$ *with those from the highest values, the total value can always be increased. Therefore, the subset* $T(\mathcal{G})$ *that maximizes the total value must consist of the* $|T(\mathcal{G})|$ *highest-value items from* $\mathcal{G}$*.*

*In other words, optimizing Eqn.* (2) *boosts the proportion of noteworthy information and reduces that of non-noteworthy information within* $T(\mathcal{G})$*. This implies an increase in* $I(T(\mathcal{G}); \mathcal{I})$ *and a decrease in* $H(T(\mathcal{G})|\mathcal{I})$*, where* $H(\cdot)$ *represents the information entropy.*

$$\begin{aligned} I(\mathcal{G}; T(\mathcal{G})|\mathcal{I}) &= H(T(\mathcal{G})|\mathcal{I}) - H(T(\mathcal{G})|\mathcal{I}, \mathcal{G}) \\ &= H(T(\mathcal{G})|\mathcal{I}) - 0 \\ &= H(T(\mathcal{G})|\mathcal{I}). \end{aligned} \tag{3}$$

$I(\mathcal{G}; T(\mathcal{G})|\mathcal{I}) = H(T(\mathcal{G})|\mathcal{I}) - H(T(\mathcal{G})|\mathcal{I}, \mathcal{G})$*as* $I(X;Y) = H(Y) - H(Y|X)$*, and* $H(T(\mathcal{G})|\mathcal{I}, \mathcal{G}) = 0$ *as* $T(\mathcal{G})$ *is a function of* $\mathcal{G}$ *(MacKay, 2003).*

*Therefore, the reduction of* $I(\mathcal{G}; T(\mathcal{G})|\mathcal{I})$ *and the increase of* $I(T(\mathcal{G}); \mathcal{I})$ *is achieved.*

*The aim of* $\min_{f,g} I(g(f(\mathcal{G})); g(f(T(\mathcal{G}))))$ *is to obtain a challenging information encoder* $g(f(\cdot))$*. It is expected that the MNI obtained under a challenging information encoder* $g(f(\cdot))$ *will exhibit enhanced robustness. By treating* $g(f(\cdot))$ *as a subpar channel with limited data capacity,* $T(\mathcal{G})$ *must prioritize transmitting the most crucial information within this limited capacity. This aligns with the greedy algorithm we previously discussed.*

Our max-min optimization differs from the min-max optimization used in methods like AD-GCL. In AD-GCL's min-max optimization, the minimization step reduces shared information between views, while the maximization step optimizes the encoder, risking insufficient information in the views. Conversely, GMNI preserves noteworthy information in views through maximization, enhances its robustness through minimization, and reduces redundancy between views via the regularization term.

After deriving the importance graph $\tilde{\mathcal{G}} = T(\mathcal{G})$, MNI can be identified. MNI represents the minimal subset of edges and features from $\mathcal{G}$ that demonstrate high importance on $\tilde{\mathcal{G}}$. MNI comprises two components: the minimal information for $-\max_T |T(\mathcal{G})| = \min_T |T(\mathcal{G})|$ and the noteworthy information for $\max_T I(g(f(\mathcal{G})); g(f(T(\mathcal{G}))))$. Those critical edges and features of $T(\mathcal{G})$ hold universal significance and is highly likely required by various downstream tasks. For example, the critical edges connecting different clusters within a graph play a crucial role in tasks like graph partition (Karypis and Kumar, 1998; Nazi et al., 2019), graph classification (Kipf and Welling, 2016; Han et al., 2022), and link prediction (Zhang and Chen, 2018; Yang et al., 2021). These edges serve as essential connections that provide valuable insights into the underlying relationships and dependencies within the graph.

*3.1.2. Importance graph construction*

This section details the generation of edge and feature importance, leading to the derivation of the importance graph, as shown in the "importance learner" in Figure 2.

First, node representations are computed using a graph convolutional layer (Welling and Kipf, 2016):

$$\boldsymbol{E} = \sigma(\boldsymbol{AXW}), \tag{4}$$

where $\sigma$ is the activation function, $\boldsymbol{W}$ is the parameter matrix. $\boldsymbol{A}$ is the adjacency matrix without normalization, as the node degrees are also essential for importance computation.

We need to get the node importance first to get the feature importance. The node importance $\boldsymbol{P}_n \in \mathbb{R}^{N\times 1}$ is given by

$$\begin{aligned} \boldsymbol{P}_n &= \mathrm{Gumbel}(h_\phi(\boldsymbol{E})), \\ \mathrm{Gumbel}(t) &= \mathrm{Sigmoid}\left(\frac{\log\eta - \log(1-\eta) + t}{\tau}\right), \ \eta \sim \mathrm{Uniform}(0,1), \end{aligned} \tag{5}$$

where $h_\phi(\cdot)$ is a simple MLP, and $\mathrm{Gumbel}(t)$ is the Gumbel-Max reparametrization function (Maddison et al., 2016). $\tau$ is the temperature parameter, and the closer $\tau$ is to 0, the closer $\boldsymbol{P}_n$ is to binarization. The Gumbel-Max reparametrization trick ensures the process can be backpropagated and makes $\boldsymbol{P}_n$ probabilistically meaningful. Thus, feature importance $\boldsymbol{P}_f \in \mathbb{R}^{F\times 1}$ can be calculated through node importance (Zhu et al., 2021), i.e.,

$$\boldsymbol{P}_f = \boldsymbol{X}^T \boldsymbol{P}_n, \tag{6}$$

where $\boldsymbol{X}^T \in \mathbb{R}^{F\times N}$ is the transpose of feature matrix $\boldsymbol{X}$. Combining node features and node importance yields feature importance. If features are real numbers between 0 and 1, we perform the Gumbel-Max reparametrization trick on $\boldsymbol{P}_f$ rather than on $\boldsymbol{P}_n$ to avoid $\boldsymbol{P}_f$ being too small.

Inspired by AD-GCL (Suresh et al., 2021), the importance of edge $\varepsilon_k = (v_i, v_j) \in \mathcal{E}$ can be expressed by

$$\boldsymbol{p}_{e,k} = \mathrm{Gumbel}(h_\psi([\boldsymbol{E}[v_i]; \boldsymbol{E}[v_j]])), \tag{7}$$

where $[\cdot;\cdot]$ is the concatenation operation, $h_\psi(\cdot)$ is a simple MLP, and $\boldsymbol{E}[v_i]$ is the embedding of node $v_i$. The edge importance $\boldsymbol{P}_e = [\boldsymbol{p}_{e,1}, ..., \boldsymbol{p}_{e,M}]^T \in \mathbb{R}^{M\times 1}$.

The importance graph is $\tilde{\mathcal{G}} = (\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{X}})$. Broadcast $\boldsymbol{P}_f \in \mathbb{R}^{F\times 1}$ to $\boldsymbol{P}'_f \in \mathbb{R}^{N\times F}$. $\tilde{\boldsymbol{X}}$ is given by $\tilde{\boldsymbol{X}} = \boldsymbol{X} \odot \boldsymbol{P}'_f$. Derive $\tilde{\boldsymbol{A}}$ by replacing the edge values in the adjacency matrix $\boldsymbol{A}$ with the edge values of $\boldsymbol{P}_e$.

### *3.2. Importance graph optimization*

This section instantiates Section 3.1.1 and is illustrated in Figure 2. The representation of node $v_i$ in $\mathcal{G}$ is defined by $\boldsymbol{z}_{i,1} = g_\xi\left(f_\theta\left(\mathcal{G}(v_i)\right)\right)$ and the representation of it in $\tilde{\mathcal{G}}$ by $\boldsymbol{z}_{i,2} = g_\xi\left(f_\theta\left(\tilde{\mathcal{G}}(v_i)\right)\right)$. For simplicity, $f_\theta$ is a 1- or 2- layer graph encoder, and $g_\xi$ is a 1- or 2- layer MLP serving as the projection head.

InfoNCE (Oord et al., 2018), a lower bound of mutual information, is applied to estimate mutual information. Thus, the mutual information between $\boldsymbol{z}_{i,1}$ and $\boldsymbol{z}_{i,2}$ can be estimated by

$$\begin{aligned} \hat{I}\left(\boldsymbol{z}_{i,1}; \boldsymbol{z}_{i,2}\right) &= \frac{1}{2}(\hat{I}_0(\boldsymbol{z}_{i,1}; \boldsymbol{z}_{i,2}) + \hat{I}_0(\boldsymbol{z}_{i,2}; \boldsymbol{z}_{i,1})), \\ \hat{I}_0(\boldsymbol{z}_{i,1}; \boldsymbol{z}_{i,2}) &= \log \frac{\exp(s(\boldsymbol{z}_{i,1}, \boldsymbol{z}_{i,2})/\epsilon)}{\sum_{j=1, j\neq i}^{N} \exp(s(\boldsymbol{z}_{i,1}, \boldsymbol{z}_{j,2})/\epsilon)}, \end{aligned} \tag{8}$$

where $s(\cdot,\cdot)$ is the cosine similarity, $\epsilon$ is the temperature parameter, and the symmetrical design considers the same status of $\mathcal{G}$ and $\tilde{\mathcal{G}}$. The mutual information between the representation of $\mathcal{G}$ and $\tilde{\mathcal{G}} = T(\mathcal{G})$ is estimated by

$$\begin{aligned} I(g_\xi(f_\theta(\mathcal{G})); g_\xi(f_\theta(T(\mathcal{G})))) &\rightarrow \hat{I}(g_\xi(f_\theta(\mathcal{G})); g_\xi(f_\theta(T(\mathcal{G})))) \\ &= \frac{1}{N}\sum_{i=1}^{N} \hat{I}(\boldsymbol{z}_{i,1}; \boldsymbol{z}_{i,2}). \end{aligned} \tag{9}$$

The regularization term $|T(\mathcal{G})|$ is defined by

$$|T(\mathcal{G})| = \lambda \left( \sum_{i=1}^{F} \boldsymbol{P}_f^{(i)}/F + \sum_{i=1}^{M} \boldsymbol{P}_e^{(i)}/M \right), \tag{10}$$

where $\lambda$ is the regularization weight, and $\boldsymbol{P}_f^{(i)}$ is the $i$-th dimension of $\boldsymbol{P}_f$. Finally, our optimization strategy to get the importance graph is

$$\begin{aligned} \max_{\boldsymbol{W},\phi,\psi} \min_{\theta,\xi} & \hat{I}(g_\xi(f_\theta(\mathcal{G})); g_\xi(f_\theta(T_{\boldsymbol{W},\phi,\psi}(\mathcal{G})))) \\ & - \lambda \left( \sum_{i=1}^{F} \boldsymbol{P}_f^{(i)}/F + \sum_{i=1}^{M} \boldsymbol{P}_e^{(i)}/M \right), \\ & \text{s.t.} \ \ \hat{I}(g_\xi(f_\theta(\mathcal{G})); g_\xi(f_\theta(T(\mathcal{G})))) \geq \zeta, \end{aligned} \tag{11}$$

which is the instantiation of Eqn. (1).

*3.2.1. View generation*

MNI refers to the minimal subset of edges and features from $\mathcal{G}$ that exhibit high importance in $\tilde{\mathcal{G}}$, specifically edges with high $p_{e,i}$ and features with high $p_{f,j}$. Views for the comparison module will be generated from MNI, which contains essential information from the graph. View generation is depicted in both Figure 1 and Figure 3.

CL aims to encourage the model to find consistency across different views. Employing soft edges and features determined by importance for all views will constrain view diversity, as these views share graph topology and features. Therefore, we introduce randomness based on MNI in views to enhance diversity and stabilize GMNI against perturbations.

Two views are required in CL, $\mathcal{G}_1 : (\boldsymbol{A}_1, \boldsymbol{X}_1)$ and $\mathcal{G}_2 : (\boldsymbol{A}_2, \boldsymbol{X}_2)$. Take the generation of $\mathcal{G}_1$ as an example, and the generation of $\mathcal{G}_2$ is similar. View modifies both the topology and features of the input graph. The edge set $\mathcal{E}_1$

of $\mathcal{G}_1$ is a subset of $\mathcal{E}$. We use a random variable $p_{\varepsilon_k} \sim \text{Bernoulli}(1 - p_{d,\varepsilon_k})$ to select the edge $\varepsilon_k$, i.e., if $p_{\varepsilon_k} = 1$, then $\varepsilon_k \in \mathcal{E}_1$, else $\varepsilon_k \notin \mathcal{E}_1$. $p_{d,\varepsilon_k}$ represents the drop probability of edge $\varepsilon_k$, which is generated based on the edge importance $p_{e,k}$. Therefore, $\boldsymbol{A}_1$ can be derived via $\mathcal{E}_1$. Inspired by GCA (Zhu et al., 2021), $\boldsymbol{P}_{d,\varepsilon} = [p_{d,\varepsilon_1}, ..., p_{d,\varepsilon_M}]$ is given by

$$\boldsymbol{P}_{d,\varepsilon} = \min\left(\frac{\boldsymbol{P}_e^{\max} - \boldsymbol{P}_e}{\boldsymbol{P}_e^{\max} - \boldsymbol{P}_e^{\text{avg}}} \cdot p_{s1}, p_t\right), \tag{12}$$

where $\min(a, b)$ indicates to select the smaller one of $a$ and $b$, $p_{s1}$ and $p_t$ are hyperparameters between 0 and 1, $p_{s1}$ is used to adjust the overall drop rate of edges, and $p_t$ is used to truncate the drop rate such that every edge has a chance of being included. $\boldsymbol{P}_e^{\max}, \boldsymbol{P}_e^{\text{avg}}$ are the maximum and average value of $\boldsymbol{P}_e$ respectively. Thus, edges with higher importance have a greater chance of being included in $\mathcal{E}_1$.

The feature matrix of $\mathcal{G}_1$ can be represented as $\boldsymbol{X}_1 = \boldsymbol{X} \odot \boldsymbol{M}$, where $\boldsymbol{M} \in \mathbb{R}^{N \times F}$ is the mask matrix, broadcasted from $\boldsymbol{m} \in \{0, 1\}^F$. A random variable $p_{f_k} \sim \text{Bernoulli}(1 - p_{d,f_k})$ is used to select the $k$-th dimensional feature, i.e., if $p_{f_k} = 1$, then $\boldsymbol{m}_k = 1$, else $\boldsymbol{m}_k = 0$. $\boldsymbol{m}_k$ represents the $k$-th dimensional of $\boldsymbol{m}$. $p_{d,f_k}$ represents the probability that the $k$-th dimensional feature is masked, which is generated based on the feature importance $p_{f,k}$. $\boldsymbol{P}_{d,f} = [p_{d,f_1}, ..., p_{d,f_F}]$ is given by

$$\boldsymbol{P}_{d,f} = \min\left(\frac{\boldsymbol{P}_f^{\max} - \boldsymbol{P}_f}{\boldsymbol{P}_f^{\max} - \boldsymbol{P}_f^{\text{avg}}} \cdot p_{s2}, p_t\right), \tag{13}$$

where $p_{s2}$ is a hyperparameter between 0 and 1 to control the overall mask rate of features. Thus, features with higher importance are less likely to be masked in $\boldsymbol{X}_1$. The views generated by these random variables effectively preserve MNI while incorporating randomness.

### *3.3. View comparison module*

To validate the effectiveness of the views, we employ the simplest view comparison module, directly pulling view representations closer via InfoMax. When using views generated through MNI, InfoMax can be applied without interference from nuisance information or loss of noteworthy information. The view comparison module is depicted in Figure 3.

The node classification task aims to learn an encoder $f_n : \mathcal{G} \rightarrow \mathbb{R}^{N\times d}$. $\mathcal{G}_1$ and $\mathcal{G}_2$ are the views of $\mathcal{G}$, then our objective is

$$\max_{f_n,g_n} I(g_n(f_n(\mathcal{G}_1)); g_n(f_n(\mathcal{G}_2))), \tag{14}$$

where $f_n$ is a GCN (Welling and Kipf, 2016), $g_n$ is a simple 1- or 2- layer MLP, and we use InfoNCE (Eqn. (8)) as the estimator of mutual information.

The graph classification task aims to learn an encoder $f_g : \mathcal{G}_i \rightarrow \mathbb{R}^d$. Given a minibatch of graphs $\{\mathcal{G}_i\}_{i=1}^K$, $\mathcal{G}_{i,1}$ and $\mathcal{G}_{i,2}$ are the views of $\mathcal{G}_i$, then the objective is

$$\max_{f_g,g_g} \frac{1}{K}\sum_{i=1}^{K} I(g_g(f_g(\mathcal{G}_{i,1})); g_g(f_g(\mathcal{G}_{i,2}))), \tag{15}$$

where $f_g$ is a GIN (Xu et al., 2018), $g_g$ is a simple 1- or 2- layer MLP followed by a readout function which is a simple summation. $f_n$ and $f_g$ can be replaced by any graph encoder. Finally, the node embeddings $f_n(\mathcal{G})$ and graph embeddings $\{f_g(\mathcal{G}_i), i = 1, ..., Q\}$ are used for downstream testing. More details of the algorithm and the computational complexity analysis can be found in Appendix C.

## 4. Experiments

### *4.1. Datasets*

To demonstrate the superiority of GMNI over other GCL methods with automated and manual data augmentations, we conduct experiments on 14 datasets, including 5 node classification datasets and 9 graph classification datasets.

The 5 datasets for node classification are Wiki-CS (Mernyei and Cangea, 2020), Amazon-Computers (Shchur et al., 2018), Amazon-Photo (Shchur et al., 2018), Coauthor-CS (Shchur et al., 2018) and Coauthor-Physics (Shchur et al., 2018). Their detailed statistics are shown in Table 1. Wiki-CS has dense real number features, whereas the other datasets have sparse one-hot features. Following GCA (Zhu et al., 2021), we evaluate models under the public train, test, and validation sets supplied by Wiki-CS. For the other four datasets, we randomly split the dataset into three sets: 80% train, 10% test, and 10% validation.

Table 1: Node classification datasets.

| Dataset | #Nodes | #Edges | #Features | #Classes | Category |
|---|---|---|---|---|---|
| Wiki-CS | 11,701 | 216,123 | 300 | 10 | Reference Networks |
| Amazon-Computers | 13,752 | 245,861 | 767 | 10 | Co-purchase Networks |
| Amazon-Photo | 7,650 | 119,081 | 745 | 8 | Co-purchase Networks |
| Coauthor-CS | 18,333 | 81,894 | 6,805 | 15 | Co-authorship Networks |
| Coauthor-Physics | 34,493 | 247,962 | 8,415 | 5 | Co-authorship Networks |

Table 2: Graph classification datasets.

| Dataset | Avg. #Graphs | Avg. #Nodes | Avg. #Edges | #Classes | Category |
|---|---|---|---|---|---|
| MUTUG | 188 | 17.93 | 19.79 | 2 | Biochemical Molecules |
| PROTEINS | 1,113 | 39.06 | 72.82 | 2 | Biochemical Molecules |
| DD | 1,178 | 284.32 | 715.66 | 2 | Biochemical Molecules |
| NCI1 | 4,110 | 29.87 | 32.30 | 2 | Biochemical Molecules |
| COLLAB | 5,000 | 74.49 | 2457.78 | 3 | Social Networks |
| GITHUB | 12,725 | 113.79 | 234.64 | 2 | Social Networks |
| IMDB-BINARY | 1,000 | 19.77 | 96.53 | 2 | Social Networks |
| REDDIT-BINARY | 2,000 | 429.63 | 497.75 | 2 | Social Networks |
| REDDIT-MULTI-5K | 4,999 | 508.52 | 594.87 | 5 | Social Networks |

We use 9 datasets from TUDataset (Morris et al., 2020) for graph classification, including MUTUG, PROTEINS, DD, NCI1, COLLAB, GITHUB, IMDB-BINARY, REDDIT-BINARY, and REDDIT-MULTI-5K. Their detailed statistics are shown in Table 2. Following JOAO (You et al., 2021), we use the entire dataset to learn graph representations and feed them into the downstream classifier using 10-fold cross-validation.

### *4.2. Baselines*

For node classification, GMNI is compared to (1) GCL methods without data augmentation, including DGI (Velickovic et al., 2019) and GMI (Peng et al., 2020). (2) Manual data augmentation GCL methods, including MV-GRL (Hassani and Khasahmadi, 2020) and GCA (Zhu et al., 2021). (3) Automated data augmentation GCL methods, including AD-GCL (Suresh et al., 2021), JOAOv2 (You et al., 2021), and AutoGCL (Yin et al., 2022).

For graph classification, GMNI is compared to (1) GCL method without data augmentation, including InfoGraph (Sun et al., 2019). (2) Manual data augmentation GCL methods, including GraphCL (You et al., 2020a), MV-GRL, GCA, and HGCL (Ju et al., 2023). (3) Automated data augmentation GCL methods, including AD-GCL, JOAOv2, and AutoGCL. (4) A special baseline, SEGA (Wu et al., 2023), is orthogonal to previous works on graph augmentations. It implements the anchor view for GCL.

The brief introductions of these baselines are as follows.

- DGI (Velickovic et al., 2019) learns graph representations by maximizing mutual information between patch representations and corresponding high-level graph summaries.
- GMI (Peng et al., 2020) learns graph representations by maximizing mutual information between the input and output of a graph neural encoder.
- InfoGraph (Sun et al., 2019) learns graph representations by maximizing mutual information between the graph-level and substructure representations at various scales.
- GraphCL (You et al., 2020a) designs four types of graph augmentations for view generation and learns graph representations by maximizing mutual information between the view representations.
- MVGRL (Hassani and Khasahmadi, 2020) learns graph representations through contrasting encodings derived from first-order neighbors and graph diffusion.
- GCA (Zhu et al., 2021) introduces an adaptive augmentation method for GCL, which integrates diverse priors related to topological and semantic aspects of the graph.
- HGCL (Ju et al., 2023) employs node-level contrastive learning, graph-level contrastive learning, and mutual contrastive learning to capture graph semantics hierarchically. Additionally, the model incorporates a Siamese network and momentum update mechanism.
- AD-GCL (Suresh et al., 2021) learns by optimizing an adversarial graph augmentation strategy to avoid capturing redundant information.
- JOAOv2 (You et al., 2021) proposes a principled bi-level optimization framework to learn the selection of DA methods for GCL.
- AutoGCL (Yin et al., 2022) employs an automated data augmentation strategy to integrate a collection of trainable graph view generators, each learning a probability distribution over the input graph.

Table 3: Unsupervised learning performance on 5 node classification datasets (average accuracy (%) ± std. over 5 runs). OOM indicates Out-Of-Memory on a 32GB GPU. The highest and second performances are in **bold** and underlined, respectively.

| Type | Model | Wiki-CS | Amaz-Comp | Amaz-Photo | Coauthor-CS | Coauthor-Phy |
|---|---|---|---|---|---|---|
| w/o GDA | DGI | 75.35 ± 0.14 | 83.95 ± 0.47 | 91.61 ± 0.22 | 92.15 ± 0.63 | 94.51 ± 0.52 |
| | GMI | 74.85 ± 0.08 | 82.21 ± 0.31 | 90.68 ± 0.17 | OOM | OOM |
| w/ Manual GDA | MVGRL | 77.52 ± 0.08 | 87.52 ± 0.11 | 91.74 ± 0.07 | 92.11 ± 0.12 | 95.33 ± 0.03 |
| | GCA | 78.35 ± 0.05 | 87.85 ± 0.31 | 92.53 ± 0.16 | 93.10 ± 0.01 | 95.73 ± 0.03 |
| w/ Automated GDA | AD-GCL | 73.46 ± 0.36 | 81.32 ± 0.93 | 88.75 ± 0.92 | 92.16 ± 0.36 | 94.57 ± 0.09 |
| | JOAO-v2 | 75.36 ± 0.47 | 85.96 ± 0.98 | 91.15 ± 0.55 | 91.33 ± 0.27 | OOM |
| | AutoGCL | 73.66 ± 0.59 | 86.44 ± 1.24 | 91.98 ± 0.58 | 92.26 ± 0.32 | OOM |
| | **GMNI** | **79.19 ± 0.13** | **89.29 ± 0.05** | **93.52 ± 0.33** | **93.61 ± 0.15** | **95.91 ± 0.10** |

- SEGA (Wu et al., 2023) achieves the anchor view for graph contrastive learning by minimizing structural entropy, which preserves essential information from the input graph.

### *4.3. Comparison with baselines*

This section compares GMNI with baselines in unsupervised and semi-supervised settings for node and graph classification. The details of the experimental setup are provided in Appendix A.

For the baselines, the experimental results are from the original papers. The original papers do not provide results for AD-GCL, JOAOv2, and Auto-GCL in unsupervised node classification, nor for GCA in unsupervised graph classification. Therefore, we either remove or add a pooling layer to obtain their node and graph representations, which are subsequently fed into the downstream classifiers. We report the best results for GCA's 3 variations, GCA-DE, GCA-EVC, and GCA-PR.

*Unsupervised learning on node classification.*

**Settings.** We train GMNI and other baselines using unlabeled data to generate representations, then use these representations to train a simple $\ell_2$-regularized logistic regression classifier with a learning rate of 0.01.

**Result analysis.** Table 3 shows that GMNI achieves outstanding performance on the node classification task. Firstly, it significantly outperforms baselines with automated DA. The reason for the improvement is that the views, defined by retaining minimal noteworthy information, approximate

Table 4: Unsupervised learning performance on graph classification on the benchmark TUDataset (Morris et al., 2020) (average accuracy (%) ± std. over 5 runs). '-' indicates that results are unavailable in papers. The highest and second performances are in **bold** and underlined, respectively.

| Type | Model | MUTAG | PROTEINS | DD | NCI1 |
|---|---|---|---|---|---|
| w/o GDA | InfoGraph | 89.01 ± 1.13 | 74.44 ± 0.31 | 72.85 ± 1.78 | 76.20 ± 1.06 |
| w/ Manual GDA | GraphCL | 86.80 ± 1.34 | 74.39 ± 0.45 | 78.62 ± 0.40 | 77.87 ± 0.41 |
| | MVGRL | 89.70 ± 1.10 | - | - | - |
| | GCA | 90.60 ± 0.76 | 75.53 ± 0.22 | 79.17 ± 0.39 | 75.87 ± 0.96 |
| | HGCL | 90.10 ± 0.80 | 75.50 ± 0.50 | 79.20 ± 0.60 | - |
| w/ Automated GDA | AD-GCL | 89.30 ± 1.50 | 73.59 ± 0.65 | 74.49 ± 0.52 | 69.67 ± 0.51 |
| | JOAOv2 | 87.30 ± 1.00 | 71.25 ± 0.85 | 66.91 ± 1.75 | 72.99 ± 0.75 |
| | AutoGCL | 88.64 ± 1.08 | 75.80 ± 0.36 | 77.57 ± 0.60 | 82.00 ± 0.29 |
| | **GMNI** | **91.57 ± 0.57** | **76.58 ± 0.29** | **79.32 ± 0.26** | **83.12 ± 0.20** |
| Type | Model | COLLAB | IMDB-B | REDDIT-B | REDDIT-M-5K |
| w/o GDA | InfoGraph | 70.65 ± 1.13 | 73.03 ± 0.87 | 82.50 ± 1.42 | 53.46 ± 1.03 |
| w/ Manual GDA | GraphCL | 71.36 ± 1.15 | 71.14 ± 0.44 | 89.53 ± 0.84 | 55.99 ± 0.28 |
| | MVGRL | - | 74.20 ± 0.70 | 84.50 ± 0.60 | - |
| | GCA | **76.67 ± 0.43** | 74.97 ± 0.40 | 87.03 ± 0.90 | 56.04 ± 0.28 |
| | HGCL | 75.80 ± 0.40 | 73.90 ± 0.70 | - | - |
| w/ Automated GDA | AD-GCL | 73.32 ± 0.61 | 71.57 ± 1.01 | 85.52 ± 0.79 | 53.00 ± 0.82 |
| | JOAOv2 | 70.40 ± 2.21 | 71.60 ± 0.86 | 78.35 ± 1.38 | 45.57 ± 2.86 |
| | AutoGCL | 70.12 ± 0.68 | 73.30 ± 0.40 | 88.58 ± 1.49 | 56.75 ± 0.18 |
| | **GMNI** | 76.19 ± 0.71 | **75.54 ± 0.29** | **91.29 ± 0.41** | **57.02 ± 0.14** |

the optimal views in InfoMin while other baselines fail to. Utilizing such views can prevent the contrastive module from learning nuisance information and make it focuses on noteworthy information, thus achieving better performance. Secondly, GMNI outperforms manual augmentation baselines on all datasets. GCL methods with manual augmentation heavily rely on the predefined DA pool, resulting in a limited approximation of the InfoMin principle. Considering factors such as time cost, computational cost, and the ability to approximate optimal views in InfoMin, GMNI emerges as a preferable alternative to GCL with manual GDA. Thus, GMNI liberates GCL from the tedious manual selection of DA.

*Unsupervised learning on graph classification.*
**Settings.** The downstream classifier for graph classification is an SVM with parameter $C$ grid searching in [0.001, 0.01, 0.1, 1, 10, 100, 1000].
**Result analysis.** Table 4 shows that GMNI achieves the best performance on 7 out of 8 datasets and exhibits sub-optimal performance on the remain-

Table 5: Unsupervised learning performance on graph classification (average accuracy (%) ± std) of SEGA, SEGA in collaboration with SOTAs, and GMNI. The highest and second performances are in **bold** and underlined, respectively.

| Model | MUTAG | PROTEINS | DD | NCI1 |
|---|---|---|---|---|
| SEGA | 90.21 ± 0.66 | 76.01 ± 0.42 | 78.76 ± 0.57 | 79.00 ± 0.72 |
| SEGA+AD-GCL | 89.89 ± 0.69 | 74.61 ± 0.81 | 75.84 ± 0.64 | 70.38 ± 0.76 |
| SEGA+JOAOv2 | 88.53 ± 2.45 | 75.94 ± 0.88 | 78.37 ± 1.26 | 78.04 ± 0.19 |
| SEGA+AutoGCL | 89.03 ± 1.01 | 76.43 ± 0.67 | 78.31 ± 1.37 | 81.84 ± 0.53 |
| **GMNI** | **91.57 ± 0.57** | **76.58 ± 0.29** | **79.32 ± 0.26** | **83.12 ± 0.20** |

| | COLLAB | IMDB-B | REDDIT-B | REDDIT-M-5K |
|---|---|---|---|---|
| SEGA | 74.12 ± 0.47 | 73.58 ± 0.44 | 90.21 ± 0.65 | 56.13 ± 0.30 |
| SEGA+AD-GCL | 75.03 ± 0.36 | 72.32 ± 0.49 | 87.74 ± 0.39 | 54.29 ± 0.54 |
| SEGA+JOAOv2 | 72.76 ± 0.27 | 72.12 ± 0.79 | 87.98 ± 0.29 | 56.15 ± 0.29 |
| SEGA+AutoGCL | 72.68 ± 0.23 | 73.95 ± 0.87 | 89.88 ± 1.21 | **57.43 ± 0.37** |
| **GMNI** | **76.19 ± 0.71** | **75.54 ± 0.29** | **91.29 ± 0.41** | 57.02 ± 0.14 |

ing 1. Furthermore, GMNI outperforms automated DA methods across all datasets. The experimental results presented in Table 5 demonstrate that GMNI outperforms both SEGA and SEGA with collaborative models on 7 out of 8 datasets. The results of the graph and node classification tasks prove that our strategy of using minimal noteworthy information to approximate minimal necessary information is effective on different downstream tasks. GMNI approximates the optimal views of InfoMin, without requiring task-relevant information. Lastly, Table 3 and Table 4 illustrate that methods with DA generally outperform methods without DA, highlighting the significance of DA in GCL.

*Semi-supervised learning on graph classification.*
**Settings.** Following AutoGCL (Yin et al., 2022), we perform semi-supervised learning on TUDataset using 10-fold cross-validation.

We apply three methods for GMNI on semi-supervised training: GMNI-Label, GMNI-Alter and GMNI-Unlabel-Fint. GMNI-Label is trained and tested using 10% labeled data. GMNI-Alter involves several alternate training steps, where each step comprises contrastive learning (CL) with 80% unlabeled data and fine-tuning (FT) with 10% labeled data. Finally, it is tested on 10% labeled data. GMNI-Unlabel-Fint is trained on 80% unlabeled data, fine-tuned on 10% labeled data, and tested on 10% labeled data. GMNI-Alter alternates between CL and FT in a sequential manner, i.e., (CL1-FT1-CL2-FT2-....), whereas GMNI-Unlabel-Fint performs FT after completing CL.
**Result analysis.** Table 6 shows that GMNI outperforms baselines on 7 out

Table 6: Semi-supervised learning performance on graph classification (average accuracy (%) ± std. over 5 runs). The highest and second performances are in **bold** and underlined, respectively.

| Dataset | GCA | GraphCL | JOAOv2 | AD-GCL | AutoGCL |
|---|---|---|---|---|---|
| PROTEINS | 73.85 ± 5.56 | 74.21 ± 4.50 | 73.31 ± 0.48 | 73.96 ± 0.47 | 75.65 ± 2.40 |
| DD | 76.74 ± 4.09 | 76.65 ± 5.12 | 75.81 ± 0.73 | 77.91 ± 0.73 | 77.50 ± 4.41 |
| NCI1 | 68.73 ± 2.36 | 73.16 ± 2.90 | 74.86 ± 0.39 | 75.18 ± 0.31 | 73.75 ± 2.25 |
| COLLAB | 74.32 ± 2.30 | 75.50 ± 2.15 | 75.53 ± 0.18 | 75.82 ± 0.26 | 77.16 ± 1.48 |
| GITHUB | 59.24 ± 3.21 | 63.51 ± 1.02 | 66.66 ± 0.60 | - | 62.46 ± 1.51 |
| IMDB-B | 73.70 ± 4.88 | 68.10 ± 5.15 | - | - | 71.90 ± 4.79 |
| REDDIT-B | 77.15 ± 6.96 | 78.05 ± 2.65 | 88.79 ± 0.65 | **90.10 ± 0.15** | 79.80 ± 3.47 |
| REDDIT-M-5K | 32.95 ± 10.89 | 48.09 ± 1.74 | 52.71 ± 0.28 | 53.49 ± 0.28 | 49.91 ± 2.70 |

| Dataset | SEGA | **GMNI-Label** | **GMNI-Alter** | **GMNI-Unlabel-Fint** | Ranks |
|---|---|---|---|---|---|
| PROTEINS | 74.65 ± 0.54 | **78.31 ± 1.31** | 76.46 ± 0.67 | 74.71 ± 0.59 | 1 |
| DD | 76.33 ± 0.43 | **80.69 ± 0.47** | 79.30 ± 1.04 | 76.38 ± 0.91 | 1 |
| NCI1 | 75.09 ± 0.22 | **76.43 ± 0.50** | 76.13 ± 0.59 | 74.32 ± 0.19 | 1 |
| COLLAB | 75.18 ± 0.22 | 75.56 ± 0.30 | **78.50 ± 0.48** | 77.21 ± 0.47 | 1 |
| GITHUB | 66.01 ± 0.66 | 65.62 ± 0.15 | **68.46 ± 0.23** | 66.32 ± 0.41 | 1 |
| IMDB-B | - | **74.16 ± 0.57** | 71.96 ± 0.84 | 69.66 ± 1.00 | 1 |
| REDDIT-B | 89.40 ± 0.23 | 82.68 ± 0.57 | 87.96 ± 0.35 | 85.91 ± 1.94 | 4 |
| REDDIT-M-5K | 53.73 ± 0.28 | 50.16 ± 0.77 | **53.96 ± 0.51** | 52.77 ± 0.57 | 1 |

of 8 datasets and gets an average rank of 1.38. Intriguingly, unlabeled data is not always effective, as the performance of GMNI-Unlabel-Fint is sometimes worse than that of GMNI-Label, which is trained with only 10% labeled data. Using labels directly to fine-tune the model, which was initially trained with unlabeled data, can interfere with the already learned distribution, resulting in a performance loss. GMNI-Alter outperforms GMNI-Unlabel-Fint by using unlabeled and labeled data alternately. The alternating training of CL and FT helps alleviate the divergence of learned knowledge between unlabeled and labeled data, promoting mutual learning between the two sources. GMNI's performance on REDDIT-BINARY is inferior to the two baseline methods due to the limited generalization ability of the shared view generator across all graphs. Thus, training a dedicated view generator for each batch can significantly boost performance, with scores of GMNI-Label at 91.10±0.36 (Rank 1), GMNI-Alter at 89.28±1.83, and GMNI-Unlabel-Fint at 88.35±0.23. However, due to computational constraints, we do not implement this training strategy in practice.

### *4.4. Ablation studies*

Ablation studies on 10 datasets, conducted with identical hyperparameters, are presented in Table 7 and Figure 4 to validate the rationale behind

Table 7: Ablation studies on GMNI (average accuracy (%) ± std. over 3 runs). GMNI-Uni, GMNI-UniE-F, GMNI-E-UniF, and GMNI are one group of ablation experiments, while GMNI-Simp, GMNI-ViewM, GMNI-Simult, and GMNI are another group of ablation experiments. The highest performance on the first and the second ablation group are underlined and in **bold**, respectively.

| Dataset | GMNI-Uni | GMNI-UniE-F | GMNI-E-UniF | GMNI |
|---|---|---|---|---|
| Wiki-CS | 79.30 ± 0.00 | 76.43 ± 0.15 | 79.16 ± 0.07 | 79.29 ± 0.05 |
| Amazon-Computers | 88.12 ± 0.25 | 86.74 ± 0.28 | 88.87 ± 0.17 | 89.29 ± 0.08 |
| Amazon-Photo | 93.27 ± 0.47 | 91.21 ± 0.49 | 93.57 ± 0.26 | 93.68 ± 0.35 |
| Coauthor-CS | 93.32 ± 0.05 | 93.41 ± 0.10 | 93.33 ± 0.02 | 93.62 ± 0.10 |
| MUTUG | 90.79 ± 0.28 | 91.13 ± 0.77 | 90.24 ± 0.42 | 91.83 ± 0.62 |
| PROTEINS | 76.28 ± 0.09 | 75.17 ± 0.37 | 76.25 ± 0.51 | 76.61 ± 0.37 |
| DD | 79.37 ± 0.59 | 76.62 ± 0.48 | 79.62 ± 0.09 | 79.43 ± 0.34 |
| COLLAB | 72.30 ± 1.06 | 74.00 ± 0.28 | 77.02 ± 1.46 | 76.08 ± 0.86 |
| IMDB-B | 75.20 ± 0.70 | 72.67 ± 0.06 | 75.07 ± 0.76 | 75.70 ± 0.26 |
| REDDIT-B | 90.05 ± 0.61 | 92.00 ± 0.48 | 90.05 ± 0.30 | 91.25 ± 0.75 |
| Dataset | GMNI-Simp | GMNI-ViewM | GMNI-Simult | GMNI |
| Wiki-CS | 79.10 ± 0.15 | **79.34 ± 0.05** | 79.13 ± 0.12 | 79.29 ± 0.05 |
| Amazon-Computers | 89.12 ± 0.16 | 89.04 ± 0.08 | 89.00 ± 0.32 | **89.29 ± 0.08** |
| Amazon-Photo | 93.65 ± 0.32 | 93.44 ± 0.39 | **93.80 ± 0.27** | 93.68 ± 0.35 |
| Coauthor-CS | 93.44 ± 0.08 | 93.48 ± 0.02 | 93.39 ± 0.12 | **93.62 ± 0.10** |
| MUTUG | 91.53 ± 1.02 | 90.45 ± 0.54 | 91.35 ± 0.61 | **91.83 ± 0.62** |
| PROTEINS | 76.49 ± 0.19 | 76.53 ± 0.28 | **77.30 ± 0.21** | 76.61 ± 0.37 |
| DD | 79.20 ± 0.22 | 79.00 ± 0.38 | 78.83 ± 0.05 | **79.43 ± 0.34** |
| COLLAB | 70.39 ± 0.95 | 75.91 ± 0.10 | 74.12 ± 0.63 | **76.08 ± 0.86** |
| IMDB-B | 70.23 ± 1.50 | 75.33 ± 1.07 | 75.60 ± 0.66 | **75.70 ± 0.26** |
| REDDIT-B | **91.63 ± 0.25** | 90.90 ± 0.38 | 89.92 ± 0.40 | 91.25 ± 0.75 |

the components of GMNI. Table 8 provides a comparison to clearly illustrate the differences between the variants of GMNI.

*Effectiveness of automated view generator.*

**Settings.** We use 6 variants to verify the effectiveness of the generator. GMNI-Uni employs uniformly distributed edge and feature importance. GMNI-UniE-F employs uniformly distributed edge importance but retains the automated generation of feature importance. GMNI-E-UniF employs uniformly distributed feature importance but retains the automated generation of edge importance. The previously mentioned variants perform DA on both topology and features simultaneously. GMNI-F masks only features based on automatically generated feature importance, while GMNI-E drops only edges based on automatically generated edge importance. GMNI-EF applies masks separately to edges and features in two views according to automatically generated importance.

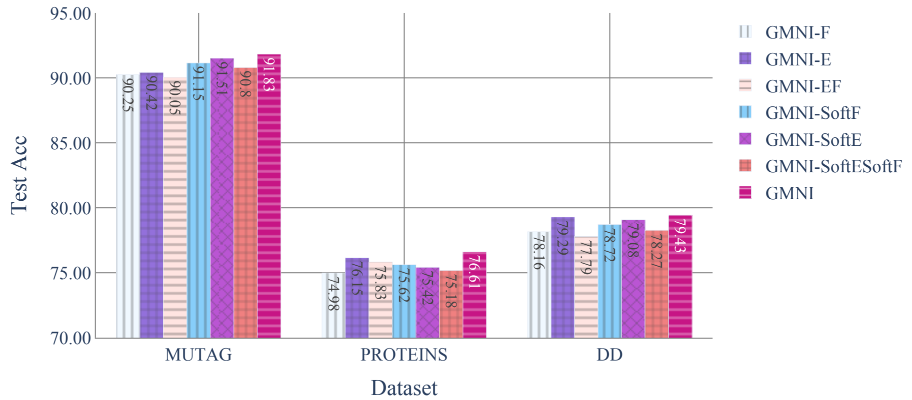



Figure 4: Ablation studies on GMNI (average accuracy (%) over 3 runs). GMNI-F, GMNI-E, GMNI-EF, GMNI-SoftF, GMNI-SoftE, and GMNI-SoftESoftF are different variations of GMNI.

Table 8: Different variations of GMNI for ablation studies.

| | GMNI-UniE-F | GMNI-F | GMNI-E-UniF | GMNI-E | GMNI-EF |
|---|---|---|---|---|---|
| View1 Edge | Uniform Distribution | None | Edge Importance | Edge Importance | None |
| View1 Feat | Feature Importance | Feature Importance | Uniform Distribution | None | Feature Importance |
| View2 Edge | Uniform Distribution | None | Edge Importance | Edge Importance | Edge Importance |
| View2 Feat | Feature Importance | Feature Importance | Uniform Distribution | None | None |
| | GMNI-SoftF | GMNI-SoftE | GMNI-SoftESoftF | GMNI | |
| View1 Edge | Edge Importance | Soft Edges | Soft Edges | Edge Importance | |
| View1 Feat | Soft Features | Feature Importance | Soft Features | Feature Importance | |
| View2 Edge | Edge Importance | Soft Edges | Soft Edges | Edge Importance | |
| View2 Feat | Soft Features | Feature Importance | Soft Features | Feature Importance | |

**Result analysis.** According to Table 7, GMNI outperforms GMNI-Uni on all datasets except Wiki-CS, and performs similarly on Wiki-CS, demonstrating the effectiveness of the automated view generator. GMNI outperforms GMNI-UniE-F on 9 datasets and outperforms GMNI-E-UniF on 8 datasets. The results indicate that combining edge and feature importance is more effective than using only one. According to Figure 4, the performance of GMNI-F, GMNI-E, and GMNI-EF is inferior to that of GMNI. Therefore, GDA should simultaneously incorporate topological and feature augmentation on both views to achieve optimal performance.

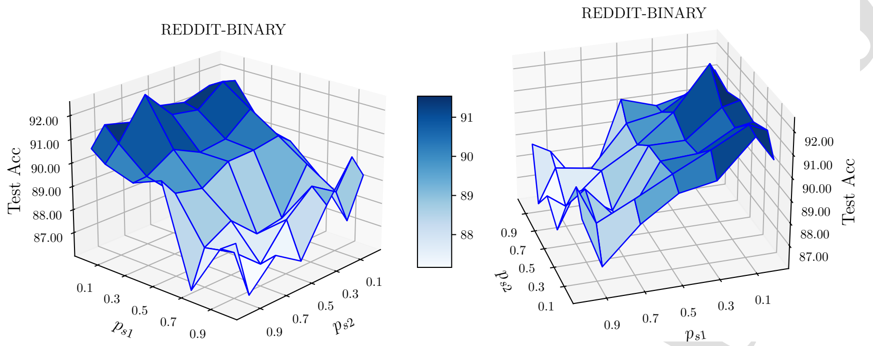



Figure 5: Graph classification performance of GMNI with different $p_{s1}$ and $p_{s2}$ on REDDIT-BINARY.

*Effectiveness of randomness.*

**Settings.** We employ 3 variations to validate the rationality of introducing randomness in view generation. GMNI-SoftE denotes a GMNI variant that uses edge importance as soft edges, while GMNI-SoftF signifies a variant that uses feature importance as soft features. GMNI-SoftE and GMNI-SoftF eliminate randomness from topology and feature data augmentation, respectively. GMNI-SoftESoftF combines both soft edges and soft features, removing randomness from both topology and feature data augmentation.

**Result analysis.** According to Figure 4, the experimental results indicate that GMNI outperforms both GMNI-SoftE and GMNI-SoftF, while GMNI-SoftE and GMNI-SoftF outperform GMNI-SoftESoftF. It indicates that GMNI, which incorporates randomness in both feature and topology DA, performs the best, while GMNI-SoftESoftF, which removes all randomness, performs the worst.

*Rationality of max-min optimization.*

**Settings.** We employ 3 variations to validate the rationality of adversarial optimization in the automated view generator. GMNI-Simp replaces the learnable GCN encoder in the view generator with a parameterless GCN, i.e., $\mathbf{W} = \mathbf{I}$. GMNI-ViewM optimizes the view generator through max-max optimization. GMNI-Simult trains the view generator and comparison module simultaneously, using the GNN encoder from the view comparison module as the GNN encoder for the view generator.

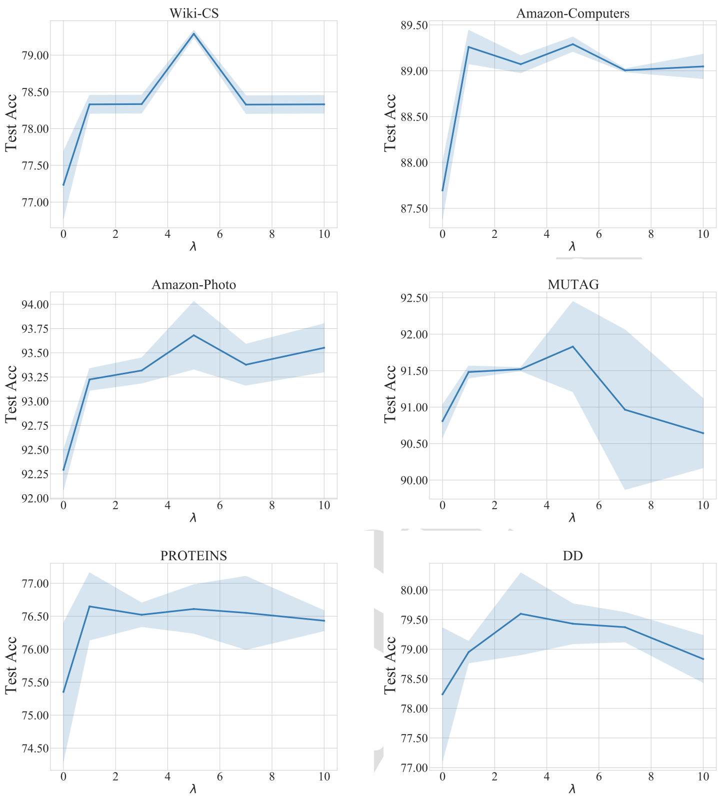



Figure 6: Node and graph classification performance of GMNI with different $\lambda$.

**Result analysis.** According to Table 7, the following conclusions can be drawn. Firstly, GMNI performs better than GMNI-Simp on 9 datasets, proving that the learnable GCN encoder in the generator is effective. Secondly, GMNI outperforms GMNI-ViewM on 9 datasets, showing that the GNN encoder and the projection head trained by minimizing are better than maximizing. Minimizing the objective results in an aggressive graph encoder and projection head, which contribute to producing a more generalized importance graph and a more precise approximation of InfoMin's optimal views. Finally, GMNI beats GMNI-Simult on 8 datasets, as training the view generator and the comparison module simultaneously prevents the generation of

Table 9: Ablation studies on $\ell_2$ distance minimization, cosine similarity maximization, and InfoMax across 13 datasets (average accuracy (%) $\pm$ std. over 3 runs). The highest performance is indicated in **bold**.

| Dataset | $\ell_2$ Distance | Cosine Similarity | InfoMax |
|---|---|---|---|
| Wiki-CS | 75.69 $\pm$ 0.21 | 77.28 $\pm$ 0.11 | **79.19 $\pm$ 0.13** |
| Amazon-Computers | 66.85 $\pm$ 0.89 | 85.87 $\pm$ 0.57 | **89.29 $\pm$ 0.05** |
| Amazon-Photo | 84.07 $\pm$ 0.68 | 91.66 $\pm$ 0.24 | **93.52 $\pm$ 0.33** |
| Coauthor-CS | 89.22 $\pm$ 0.58 | 93.42 $\pm$ 0.05 | **93.61 $\pm$ 0.15** |
| Coauthor-Physics | 85.75 $\pm$ 0.93 | 95.79 $\pm$ 0.03 | **95.91 $\pm$ 0.10** |
| MUTAG | 90.62 $\pm$ 0.26 | 91.51 $\pm$ 0.39 | **91.57 $\pm$ 0.57** |
| PROTEINS | 76.01 $\pm$ 0.26 | 76.28 $\pm$ 0.25 | **76.58 $\pm$ 0.29** |
| DD | 77.70 $\pm$ 0.21 | 77.54 $\pm$ 0.20 | **79.32 $\pm$ 0.26** |
| NCI1 | 79.91 $\pm$ 0.36 | 81.33 $\pm$ 0.10 | **83.12 $\pm$ 0.20** |
| COLLAB | 69.75 $\pm$ 0.27 | 73.99 $\pm$ 0.44 | **76.19 $\pm$ 0.71** |
| IMDB-B | 72.77 $\pm$ 0.17 | 73.17 $\pm$ 0.33 | **75.54 $\pm$ 0.29** |
| REDDIT-B | 87.65 $\pm$ 0.50 | 88.37 $\pm$ 0.49 | **91.29 $\pm$ 0.41** |
| REDDIT-M-5K | 54.05 $\pm$ 0.34 | 53.71 $\pm$ 0.53 | **57.02 $\pm$ 0.14** |

superior views and representations. The view generator emphasizes remaining noteworthy information and discarding nuisance information, whereas the comparison module highlights the quality of the graph encoder. They impede each other when training together.

*Effectiveness of InfoMax.*

**Settings.** First, we use two variants to validate the rationale of InfoMax by modifying the optimization objectives in the GMNI view comparison module to $\ell_2$ distance minimization and cosine similarity maximization. Second, InfoMax consists of two parts: pulling the representations of the same node closer and pushing those of different nodes further apart. Thus, we decompose this objective into two distinct optimization goals: one for solely maximizing the similarity between representations of the same node, and another for solely minimizing the similarity between representations of different nodes.

**Result analysis.** First, the experimental results in Table 9 demonstrate that InfoMax consistently outperforms both $\ell_2$ distance and cosine similarity. Second, the experimental results in Table 10 indicate that InfoMax is optimal. Focusing solely on bringing representations of the same node closer may cause all representations to converge into a uniform state, while solely pushing representations of different nodes apart may result in overly dispersed representations, failing to achieve the classification goal.

Table 10: Ablation studies on the decomposition of the InfoMax over 13 datasets (average accuracy (%) ± std. over 3 runs). The highest performance is indicated in **bold**.

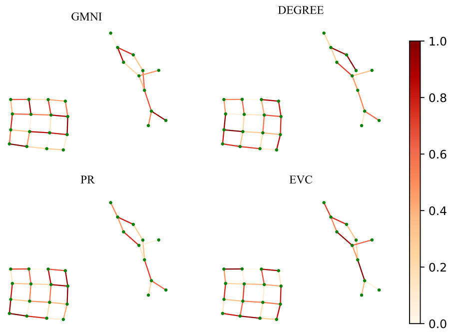
| Dataset | OnlyClose | OnlyFar | InfoMax |
|---|---|---|---|
| Wiki-CS | 75.47 ± 0.03 | 77.28 ± 0.11 | **79.19 ± 0.13** |
| Amazon-Computers | 66.84 ± 0.88 | 85.87 ± 0.57 | **89.29 ± 0.05** |
| Amazon-Photo | 84.07 ± 0.69 | 91.66 ± 0.24 | **93.52 ± 0.33** |
| Coauthor-CS | 89.30 ± 0.60 | 93.42 ± 0.05 | **93.61 ± 0.15** |
| Coauthor-Physics | 85.89 ± 0.91 | 95.79 ± 0.03 | **95.91 ± 0.10** |
| MUTAG | 90.61 ± 0.27 | 91.51 ± 0.39 | **91.57 ± 0.57** |
| PROTEINS | 75.35 ± 0.79 | 76.28 ± 0.25 | **76.58 ± 0.29** |
| DD | 77.14 ± 0.63 | 77.54 ± 0.20 | **79.32 ± 0.26** |
| NCI1 | 80.84 ± 0.24 | 81.33 ± 0.10 | **83.12 ± 0.20** |
| COLLAB | 69.45 ± 0.23 | 73.99 ± 0.44 | **76.19 ± 0.71** |
| IMDB-B | 73.10 ± 0.24 | 73.17 ± 0.33 | **75.54 ± 0.29** |
| REDDIT-B | 86.45 ± 0.57 | 88.37 ± 0.49 | **91.29 ± 0.41** |
| REDDIT-M-5K | 52.32 ± 0.75 | 53.71 ± 0.53 | **57.02 ± 0.14** |

### *4.5. Additional experiments*

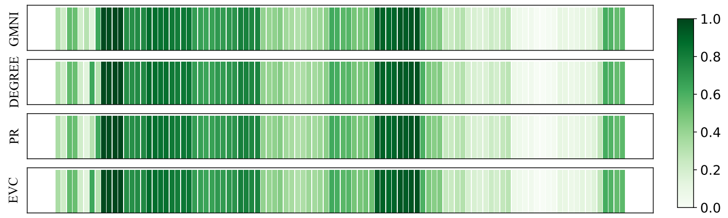
*Hyperparameter analysis.* We analyze critical hyperparameters $p_{s1}, p_{s2}$ of GMNI. To simplify the analysis, $p_{s1}$ and $p_{s2}$ of the two views are set to be identical, and they are utilized to adjust the overall drop rate of edges and the overall mask rate of features. $p_{s1}$ and $p_{s2}$ are selected from [0.0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0]. The results are shown in Figure 5.

Firstly, it can be observed that GMNI is more sensitive to $p_{s1}$ than $p_{s2}$; altering $p_{s1}$ results in a more noticeable performance change. This observation suggests that topology information plays a more crucial role in GMNI than feature information. Secondly, it is worth noting that excessively large values of $p_{s1}$ or $p_{s2}$ have a detrimental effect on performance. When $p_{s1}$ becomes excessively large, the topology information of the graph is almost completely destroyed, resulting in isolated nodes that lack connections. When $p_{s2}$ becomes excessively large, the graph almost degenerates into a featureless graph. The performance tends to be extremely poor when both are large. Thirdly, points in Figure 5 surrounding (0,0) perform better than (0,0). This observation suggests that utilizing importance to generate views, even with small drop or mask rates, is more advantageous than solely employing the original graph for comparison.

Subsequently, we analyze the hyperparameter $\lambda$ used for regularization. The results are shown in Figure 6. Firstly, the model's performance deteriorates when the regularization weight $\lambda$ is set to zero. This occurs because




(a) Edge importance visualization.




(b) Feature importance visualization. 100 dimensions from the features are selected at random for display.

Figure 7: DEGREE, PR, and EVC are degree, PageRank and eigenvector centrality, respectively.

of the absence of the 'minimal condition' required for 'minimal noteworthy information,' resulting in excessive redundant information in the views. Secondly, on the MUTAG and DD datasets, we observe a decrease in model performance with a large regularization weight $\lambda$. This is attributed to the overly restrictive constraints on $\mathbf{P}_f$ and $\mathbf{P}_e$, which hinder the inclusion of noteworthy information in the importance graph.

*Importance visualization.* Centrality reflects the significance of a node in a graph. In GCA, edge and feature importance are defined by degree, PageRank, and eigenvector centrality. To demonstrate the reasonableness of the importance gained by GMNI, we visualize the importance defined by those centralities and obtained by GMNI on a graph of MUTAG. MUTAG is a collection of nitroaromatic compounds, where the graph's vertices stand for atoms, and the edges between vertices represent bonds between the corre-

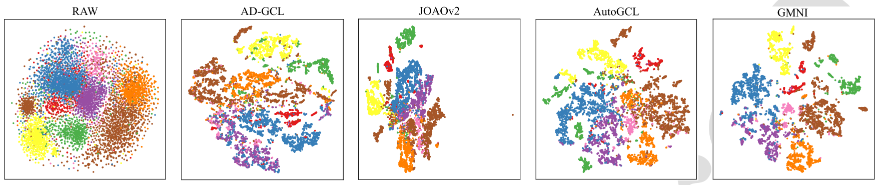



Figure 8: t-SNE visualization of raw features and node representations on Amazon-Photo. Different colors represent different node classes.

sponding atoms. In Figure 7(b), the feature importance gained by GMNI aligns with those acquired by centrality, demonstrating the rationality of the automatically generated importance.

*Embedding visualization.* To demonstrate the quality of the representations, we employ two components PCA (Abdi and Williams, 2010) and t-SNE (Van der Maaten and Hinton, 2008) to visualize the raw features and representations of AD-GCL, JOAOv2, AutoGCL, and GMNI on Amazon-Photo. Figure 8 demonstrates that GMNI achieves a more distinct classification boundary, showcasing the superior representations learned by GMNI. These representations prove to be effective for the downstream classification task.

## 5. Conclusion

In this paper, we propose GMNI, a novel GCL method featuring automated DA. GMNI approximates InfoMin's optimal views by replacing minimal necessary information with minimal noteworthy information, without requiring task-relevant information. Applying InfoMax to these views can avoid the risk of redundant information and insufficient information. In addition, GMNI introduces randomness to augmentation, thus enhancing view diversity and stabilizing the model against perturbations. Extensive experiments on node and graph classification tasks demonstrate GMNI's superiority, which outperforms both automated and manual DA GCL methods.

In the future, we will further validate the effectiveness of our approximate optimal views on more downstream tasks and the strong correlation between noteworthy and necessary information. Furthermore, approximating InfoMin's optimal views in the spectral domain is another promising research approach.

## Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Furao Shen reports financial support was provided by the STI 2030-Major Projects of China under Grant 2021ZD0201300, the National Science Foundation of China under Grant 62276127.

## Data availability

Data will be made available on request.

## Acknowledgments

This work was supported in part by the STI 2030-Major Projects of China under Grant 2021ZD0201300, and by the National Science Foundation of China under Grant 62276127.

## Appendix A. Experimental settings

Table A.11 details the hyperparameters of GMNI for unsupervised node classification. The GNN encoder and projection head of the view generator have the same hidden dimensions as the GCN, and the projection head is 1-layer. The GNN encoder of the view comparison module is 2-layer, and the projection head is 1-layer.

Table A.12 details the hyperparameters of GMNI for unsupervised graph classification. The projection head of the view generator is 2-layer, whereas the view comparison uses a single layer.

For semi-supervised learning, we follow AutoGCL (Yin et al., 2022) and use ResGCN (Chen et al., 2019) with a hidden size of 128 as the GNN encoder for view comparison. All other hyperparameters remain the same as those used for unsupervised graph classification. For GITHUB, the parameters are set as follows: (view 1 $p_{s1}$, view 2 $p_{s1}$, view 1 $p_{s2}$, view 2 $p_{s2}$) = (0.3, 0.2, 0.3, 0.2). The GNN encoder in the view comparison module has a hidden dimension of 64 and is 3-layer. The learning rate for both the view generator and the view comparison module is set to 0.01.

In GMNI, the sum of node representations of the two views and the original graph: $f_n(\mathcal{G}_1) + f_n(\mathcal{G}_2) + 2 * f_n(\mathcal{G})$ or the sum of node representations of the two views: $f_n(\mathcal{G}_1) + f_n(\mathcal{G}_2)$ is used for node classification, considering

Table A.11: Hyperparameters of GMNI for unsupervised node classification. We substitute 'dim' for 'dimension' and 'proj' for 'projection head.' The symbol " is the ditto mark.

| GMNI View Generator | GCN hidden dim | $h_\psi$ structure | $h_\phi$ structure | GNN #layers | $\tau$ | $\lambda$ | activation | learning rate |
|---|---|---|---|---|---|---|---|---|
| Wiki-CS | 128 | 1 | 1 | 1 | 1 | 5 | PReLU | 0.001 |
| Amazon-Computers | 128 | 64-1 | 32-1 | 2 | " | 5 | RReLU | 0.001 |
| Amazon-Photo | 256 | 128-1 | 64-1 | 2 | " | 5 | ReLU | 0.001 |
| Coauthor-CS | 512 | 256-1 | 64-1 | 1 | " | 5 | RReLU | 0.001 |
| Coauthor-Physics | 128 | 64-1 | 32-1 | 2 | " | 0.5 | RReLU | 0.01 |

| GMNI View Comparison | view 1 $p_{s1}$ | view 2 $p_{s1}$ | view 1 $p_{s2}$ | view 2 $p_{s2}$ | $p_t$ | GNN hidden dim | GNN #layers | proj hidden dim | $\epsilon$ | learning rate |
|---|---|---|---|---|---|---|---|---|---|---|
| Wiki-CS | 0.2 | 0.4 | 0.1 | 0.1 | 0.6 | 128 | 2 | 128 | 0.4 | 0.01 |
| Amazon-Computers | 0.6 | 0.3 | 0.2 | 0.1 | 0.7 | 128 | " | 128 | 0.2 | 0.001 |
| Amazon-Photo | 0.3 | 0.5 | 0.1 | 0.1 | 0.7 | 256 | " | 64 | 0.3 | 0.001 |
| Coauthor-CS | 0.1 | 0.1 | 0.3 | 0.2 | 0.7 | 256 | " | 256 | 0.4 | 0.0001 |
| Coauthor-Physics | 0.4 | 0.4 | 0.1 | 0.4 | 0.7 | 128 | " | 64 | 0.5 | 0.001 |

Table A.12: Hyperparameters of GMNI for unsupervised graph classification. We substitute 'VG' for 'View Generator', and 'VC' for 'View Comparison.'

| GMNI | VG's GCN hidden dim | VG's $h_\psi$ structure | VG's $h_\phi$ structure | VG's GNN #layers | $\tau$ | $\lambda$ | activation | VG's learning rate |
|---|---|---|---|---|---|---|---|---|
| All Datasets | 64 | 32-1 | 16-1 | 1 | 1 | 5 | ReLU | 0.001 |

| | view 1 $p_{s1}$ | view 2 $p_{s1}$ | view 1 $p_{s2}$ | view 2 $p_{s2}$ | $p_t$ | VC's GNN hidden dim | VC's GNN #layers | $\epsilon$ | VC's learning rate |
|---|---|---|---|---|---|---|---|---|---|
| MUTAG | 0.3 | 0.2 | 0.3 | 0.2 | 0.7 | 64 | 5 | 0.2 | 0.001 |
| PROTEINS | 0.3 | 0.2 | 0.3 | 0.2 | " | 128 | 5 | " | " |
| DD | 0.3 | 0.2 | 0.3 | 0.2 | " | 128 | 5 | " | " |
| NCI1 | 0.1 | 0.2 | 0.4 | 0.2 | " | 128 | 5 | " | " |
| COLLAB | 0.3 | 0.2 | 0.3 | 0.2 | " | 64 | 2 | " | " |
| IMDB-B | 0.3 | 0.2 | 0.3 | 0.2 | " | 128 | 5 | " | " |
| REDDIT-B | 0.3 | 0.2 | 0.3 | 0.2 | " | 64 | 2 | " | " |
| REDDIT-M-5K | 0.4 | 0.2 | 0.3 | 0.2 | " | 128 | 5 | " | " |

Table A.13: Average training time per epoch of GCL methods with automated DA.

| Dataset | AD-GCL | JOAOv2 | AutoGCL | GMNI VG | GMNI VC |
|---|---|---|---|---|---|
| Wiki-CS | 0.37 ± 0.09 | 1.53 ± 0.15 | 0.29 ± 0.09 | 0.26 ± 0.03 | 0.10 ± 0.00 |
| Amaz-Comp | 0.42 ± 0.09 | 2.28 ± 0.12 | 0.40 ± 0.08 | 0.42 ± 0.05 | 0.13 ± 0.00 |
| Amaz-Photo | 0.29 ± 0.10 | 0.94 ± 0.10 | 0.27 ± 0.08 | 0.33 ± 0.06 | 0.06 ± 0.00 |
| Coauthor-CS | 0.76 ± 0.12 | 7.25 ± 0.28 | 2.01 ± 0.84 | 1.53 ± 0.05 | 0.48 ± 0.01 |
| Coauthor-Phy | 0.40 ± 0.16 | - | - | 2.53 ± 0.20 | 0.93 ± 0.04 |
| MUTAG | 0.27 ± 0.00 | 7.83 ± 0.17 | 0.06 ± 0.00 | 0.03 ± 0.00 | 0.03 ± 0.00 |
| PROTEINS | 1.76 ± 0.15 | 15.99 ± 0.96 | 0.54 ± 0.17 | 0.24 ± 0.02 | 0.15 ± 0.02 |
| DD | 2.52 ± 0.22 | 171.16 ± 2.73 | 0.71 ± 0.15 | 4.33 ± 0.13 | 3.96 ± 0.05 |
| NCII | 6.82 ± 0.54 | 106.52 ± 1.98 | 1.10 ± 0.04 | 0.66 ± 0.04 | 0.49 ± 0.02 |
| COLLAB | 13.78 ± 1.47 | 419.96 ± 10.61 | 2.58 ± 0.41 | 8.01 ± 0.07 | 3.66 ± 0.05 |
| IMDB-B | 1.48 ± 0.01 | 11.95 ± 0.53 | 0.26 ± 0.03 | 0.19 ± 0.02 | 0.10 ± 0.00 |
| RDT-B | 4.79 ± 0.41 | 153.11 ± 6.83 | 1.09 ± 0.09 | 1.55 ± 0.05 | 0.44 ± 0.01 |
| RDT-M-5K | 12.85 ± 1.73 | 520.64 ± 27.55 | 3.37 ± 0.46 | 2.47 ± 0.07 | 2.14 ± 0.02 |

that incorporating view representations can lead to more generalized representations. Graph representations $\{f_g(\mathcal{G}_i), i = 1, ..., Q\}$ are used for graph classification. On all datasets, the Xavier initialization (Glorot and Bengio, 2010) and Adam optimizer (Kingma and Ba, 2014) are utilized.

All our experiments are conducted on four Tesla V100-SXM2 (32GB) with Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz, four NVIDIA GeForce RTX 2080 Ti (11GB) with Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz, and two TITAN Xp (12GB) GPUs with Intel(R) Xeon(R) CPU E5-2643 v4 @ 3.40GHz. Python version is 3.8.8, and PyTorch version is 1.12.0+cu102.

## Appendix B. Notations and abbreviations

| Notation | Meaning |
|---|---|
| $\mathcal{G}$ | The graph. |
| $\mathcal{V} = \{v_1, ..., v_N\}$ | The set of nodes. |
| $\mathcal{E} = \{\varepsilon_1, ..., \varepsilon_M\}$ | The set of edges. |
| $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\} \in \mathbb{R}^{N\times F}$ | The feature matrix. $F$ is the feature dimension. |
| $\boldsymbol{A} \in \{0,1\}^{N\times N} : \boldsymbol{A}_{ij} = \mathbb{I}((v_i, v_j) \in \mathcal{E})$ | The adjacency matrix. |
| $\boldsymbol{D}$: $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$ | The degree matrix. |
| $f_n : \mathcal{G} \to \mathbb{R}^{N\times d}$ | The encoder for node classification. $d$ is the output dimension. |
| $f_g : \mathcal{G}_i \to \mathbb{R}^d$ | The encoder for graph classification. |
| $\boldsymbol{P}_e \in \mathbb{R}^{M\times 1}$ | The edge importance. |
| $\boldsymbol{P}_f \in \mathbb{R}^{F\times 1}$ | The feature importance. |
| $\boldsymbol{P}_n \in \mathbb{R}^{N\times 1}$ | The node importance. |
| $\tilde{\boldsymbol{A}}$ | The adjacency matrix with edge importance. |
| $\tilde{\boldsymbol{X}}$ | The feature matrix with feature importance. |
| $\tilde{\mathcal{G}} = T(\mathcal{G}) = (\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{X}})$ | The importance graph. $T$ is the transformation function that maps $\mathcal{G}$ to $\tilde{\mathcal{G}}$. |
| $\lvert T(\mathcal{G})\rvert$ | The normalized sum of edge importance and feature importance. |
| $\zeta$ | The mutual information lower bound of our objective. |
| $\mathcal{I}$ | The noteworthy information. |
| $\boldsymbol{W}$ | The parameter matrix of the GCN. |
| $\boldsymbol{E}$ | The node representation. |
| $\sigma(\cdot)$ | The activation function. |
| $h_\phi$,$h_\psi$ | Simple MLPs. |
| $\tau, \epsilon$ | The temperature parameter. |
| Gumbel$(\cdot)$ | The Gumbel-Max reparametrization function. |
| $\eta$ | A random variable following the uniform distribution $(0, 1)$. |
| $g_\xi(\cdot)$ | A 1- or 2- layer MLP as the projection head. |

| | |
|---|---|
| $f_\theta(\cdot)$ | A 1- or 2- layer graph encoder. |
| $\lambda$ | The regularization weight. |
| $\hat{I}(\cdot;\cdot)$ | The mutual information estimator. |
| $\boldsymbol{P}_{d,\varepsilon}$ | The edge drop rate. |
| $\boldsymbol{P}_{d,f}$ | The feature mask rate. |
| $\boldsymbol{M} \in \mathbb{R}^{N\times F}$ | The mask matrix. |
| $\mathcal{G}_1 : (\boldsymbol{A}_1, \boldsymbol{X}_1), \mathcal{G}_2 : (\boldsymbol{A}_2, \boldsymbol{X}_2)$ | Two views. |
| $p_{s1}$ | The hyperparameter used to adjust the overall drop rate of edges. |
| $p_{s2}$ | The hyperparameter used to adjust the overall mask rate of features. |
| $p_t$ | The hyperparameter used to truncate the drop rate of edges and features. |
| $g_n, g_g$ | Projection heads. |

| Abbreviation | Expanded Version |
|---|---|
| CL | Contrastive Learning |
| GCL | Graph Contrastive Learning |
| DA | Data Augmentation |
| GDA | Graph Data Augmentation |
| MI | Mutual Information |
| MNI | Minimal Noteworthy Information |

## Appendix C. Algorithm

Algorithm 1 describes the details of the automated graph view generator for GMNI. Algorithm 2 and 3 describe the details of the view comparison module for GMNI on node and graph classification, respectively.

For a graph with $N$ nodes, $F$ features, and $M$ edges, the view generator of GMNI requires $O(N^2F + (M + N)F^2)$ floating-point operations (FLOPs), while the view comparison requires $O(N^2F + NF^2)$ FLOPs. The FLOPs of AD-GCL, JOAOv2, and AutoGCL are $O(N^2F + (M + N)F^2)$, $O(N^2F + NF^2)$, and $O(N^2F + NF^2)$, respectively. Thus, GMNI's view generator has an asymptotic complexity comparable to that of AD-GCL, and GMNI's view comparison has an asymptotic complexity comparable to that of JOAO and AutoGCL.

Table A.13 illustrates the average training time per epoch for various GCL methods with automated DA on a Tesla V100-SXM2 (32GB). GMNI is efficient in terms of computing resources and time. Firstly, the view generator training can be completed in a relatively small number of epochs. Secondly, the view comparison module in GMNI focuses solely on maximization, whereas the other three models require both maximization and minimization processes.

**Algorithm 1** Automated graph view generator

**Require:** graph $\mathcal{G} = (\mathbf{A}, \mathbf{X})$; edge set $\mathcal{E}$, $|\mathcal{E}| = M$; feature dimension $F$; GCN encoder $f_{\mathbf{W}}(\cdot)$; feature importance MLP $h_{\phi}(\cdot)$; edge importance MLP $h_{\psi}(\cdot)$; Gumbel-Max function Gumbel$(\cdot)$; GNN encoder $f_{\theta}(\cdot)$; projection head $g_{\xi}(\cdot)$; mutual information estimator $\hat{I}(\cdot;\cdot)$.

**HyperParas:** regularization weight $\lambda$; scale parameters for view 1 and view 2 $p_{1,s1}, p_{2,s1}, p_{1,s2}, p_{2,s2}$; truncate parameter $p_t$; learning rate $\alpha$.

**Ensure:** view 1 $\mathcal{G}_1 = (\mathbf{A}_1, \mathbf{X}_1)$; view 2 $\mathcal{G}_2 = (\mathbf{A}_2, \mathbf{X}_2)$.

**for** ep in #epochs **do**
  $\mathbf{E} = f_{\mathbf{W}}(\mathbf{A}, \mathbf{X})$;
  $\mathbf{P}_n = \text{Gumbel}(h_{\phi}(\mathbf{E}))$;
  $\mathbf{P}_f = \mathbf{X}^T \mathbf{P}_n$;
  **for** $\forall \varepsilon_k \in \mathcal{E}$ **do**
    $\mathbf{p}_{e,k} = \text{Gumbel}(h_{\psi}([\mathbf{E}[v_i]; \mathbf{E}[v_j]])), \varepsilon_k = (v_i, v_j)$;
  **end for**
  $\mathbf{P}_e = [\mathbf{p}_{e,1}, ..., \mathbf{p}_{e,M}]^T$;
  Broadcast $\mathbf{P}_f \in \mathbb{R}^{F\times 1}$ to $\mathbf{P}'_f \in \mathbb{R}^{N\times F}$;
  $\tilde{\mathbf{X}} = \mathbf{X} \odot \mathbf{P}'_f$;
  Derive $\tilde{\mathbf{A}}$ via $\mathbf{P}_e$;
  $\tilde{\mathcal{G}} = (\tilde{\mathbf{A}}, \tilde{\mathbf{X}})$;
  $\mathcal{L} = -\hat{I}(g_{\xi}(f_{\theta}(\mathcal{G})); g_{\xi}(f_{\theta}(\tilde{\mathcal{G}})))$;
  $\mathcal{R} = \frac{\sum_{i=1}^{F} \mathbf{P}_f^{(i)}}{F} + \frac{\sum_{i=1}^{M} \mathbf{P}_e^{(i)}}{M}$;
  /* Update the importance learner */
  $\mathbf{W} \leftarrow \mathbf{W} - \alpha \nabla_{\mathbf{W}}(\mathcal{L} + \lambda * \mathcal{R})$;
  $\phi \leftarrow \phi - \alpha \nabla_{\phi}(\mathcal{L} + \lambda * \mathcal{R})$;
  $\psi \leftarrow \psi - \alpha \nabla_{\psi}(\mathcal{L} + \lambda * \mathcal{R})$;
  /* Update the GNN encoder and projection head */
  $\theta \leftarrow \theta + \alpha \nabla_{\theta}(\mathcal{L})$;
  $\xi \leftarrow \xi + \alpha \nabla_{\xi}(\mathcal{L})$;
**end for**

$$\mathbf{P}_{d,\varepsilon} = [p_{d,\varepsilon_1}, ..., p_{d,\varepsilon_M}] = \min\left(\frac{\mathbf{P}_e^{\max} - \mathbf{P}_e}{\mathbf{P}_e^{\max} - \mathbf{P}_e^{\text{avg}}} \cdot p_{1,s1}, p_t\right);$$

Sample $p_{\varepsilon_k}$ from Bernoulli$(1 - p_{d,\varepsilon_k}), k = 1, ..., M$;
Use $p_{\varepsilon_k}, k = 1, ..., M$ to generate $\mathbf{A}_1$;

$$\mathbf{P}_{d,f} = [p_{d,f_1}, ..., p_{d,f_F}] = \min\left(\frac{\mathbf{P}_f^{\max} - \mathbf{P}_f}{\mathbf{P}_f^{\max} - \mathbf{P}_f^{\text{avg}}} \cdot p_{1,s2}, p_t\right);$$

Sample $p_{f_k}$ from Bernoulli$(1 - p_{d,f_k}), k = 1, ..., F$;
Use $p_{f_k}, k = 1, ..., F$ to generate $\mathbf{X}_1$;
Generate $\mathbf{A}_2, \mathbf{X}_2$ similarly;
**return** $\mathcal{G}_1 = (\mathbf{A}_1, \mathbf{X}_1)$; $\mathcal{G}_2 = (\mathbf{A}_2, \mathbf{X}_2)$;

**Algorithm 2** View comparison module for node classification

**Input:** graph $\mathcal{G}$; GNN encoder $f_\omega(\cdot)$; projection head $g_\rho(\cdot)$; mutual information estimator $\hat{I}(\cdot;\cdot)$.
**HyperParas:** learning rate $\beta$.
**Ensure:** trained $f_\omega(\cdot)$.

```
(𝒢_1, 𝒢_2) = Algorithm 1 (𝒢);
for ep in #epochs do
    ℒ = −Î(g_ρ(f_ω(𝒢_1)); g_ρ(f_ω(𝒢_2)));
    ω ← ω − β∇_ω ℒ; ρ ← ρ − β∇_ρ ℒ;
end for
return trained f_ω(·);
```

- $(\mathcal{G}_1, \mathcal{G}_2) = \text{Algorithm 1 } (\mathcal{G})$;
- **for** ep in #epochs **do**
  - $\mathcal{L} = -\hat{I}(g_\rho(f_\omega(\mathcal{G}_1)); g_\rho(f_\omega(\mathcal{G}_2)))$;
  - $\omega \leftarrow \omega - \beta\nabla_\omega\mathcal{L}$; $\rho \leftarrow \rho - \beta\nabla_\rho\mathcal{L}$;
- **end for**
- **return** trained $f_\omega(\cdot)$;

**Algorithm 3** View comparison module for graph classification

**Input:** graph set $\{\mathcal{G}_i\}, i = 1, ..., Q$; GNN encoder $f_\omega(\cdot)$; projection head $g_\rho(\cdot)$; mutual information estimator $\hat{I}(\cdot;\cdot)$.
**HyperParas:** learning rate $\beta$; batch size $B$; number of batches $BN$.
**Ensure:** trained $f_\omega(\cdot)$.

- **for** $i = 1$ to $BN$ **do**
  - **for** $\mathcal{G}_j$ in sampled minibatch $\{\mathcal{G}_j\}_{j=1}^{B}$ **do**
    - $(\mathcal{G}_{i,(j,1)}, \mathcal{G}_{i,(j,2)}) = \text{Algorithm 1 } (\mathcal{G}_{i,j})$;
  - **end for**
- **end for**
- **for** ep in #epochs **do**
  - **for** $i = 1$ to $BN$ **do**
    - $\mathcal{L} = -\frac{1}{B}\sum_{j=1}^{B}\hat{I}(g_\rho(f_\omega(\mathcal{G}_{i,(j,1)})); g_\rho(f_\omega(\mathcal{G}_{i,(j,2)})))$;
    - $\omega \leftarrow \omega - \beta\nabla_\omega\mathcal{L}$; $\rho \leftarrow \rho - \beta\nabla_\rho\mathcal{L}$;
  - **end for**
- **end for**
- **return** trained $f_\omega(\cdot)$;

## References

J. Devlin, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

G. Zhou, S. Bu, T. Kirubarajan, Simultaneous spatiotemporal bias compensation and data fusion for asynchronous multisensor systems, Chinese Journal of Information Fusion 1 (2024) 16–32.

H. Fan, D. Lu, Y. Jiang, A. J. Lilienthal, Extraction of motion information from occupancy grid map using keystone transform, Chinese Journal of Information Fusion 1 (2024) 63–78.

W. Ju, Y. Gu, X. Luo, Y. Wang, H. Yuan, H. Zhong, M. Zhang, Unsupervised graph-level representation learning with hierarchical contrasts, Neural Networks 158 (2023) 359–368.

Y. Liu, Y. Zheng, D. Zhang, V. C. Lee, S. Pan, Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating, in: Proceedings of the AAAI conference on artificial intelligence, volume 37, 2023, pp. 4516–4524.

Y. Mo, L. Peng, J. Xu, X. Shi, X. Zhu, Simple unsupervised graph representation learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, 2022, pp. 7797–7805.

S. Xiao, S. Wang, Y. Dai, W. Guo, Graph neural networks in node classification: survey and evaluation, Machine Vision and Applications 33 (2022) 1–19.

F.-Y. Sun, J. Hoffmann, V. Verma, J. Tang, Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization, arXiv preprint arXiv:1908.01000 (2019).

E. Pan, Z. Kang, Multi-view contrastive graph clustering, Advances in neural information processing systems 34 (2021) 2148–2159.

M. Lin, T. Xiao, E. Dai, X. Zhang, S. Wang, Certifiably robust graph contrastive learning, in: A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (Eds.), Advances in Neural Information Processing Systems, volume 36, Curran Associates, Inc., 2023, pp. 17008–17037.

J. Chen, G. Kou, Attribute and structure preserving graph contrastive learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, 2023, pp. 7024–7032.

X. Gong, C. Yang, C. Shi, Ma-gcl: Model augmentation tricks for graph contrastive learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, 2023, pp. 4284–4292.

Y. You, T. Chen, Y. Shen, Z. Wang, Graph contrastive learning automated, in: International Conference on Machine Learning, PMLR, 2021, pp. 12121–12132.

K. Ding, Z. Xu, H. Tong, H. Liu, Data augmentation for deep graph learning: A survey, ACM SIGKDD Explorations Newsletter 24 (2022) 61–77.

P. Mishra, A. Piktus, G. Goossen, F. Silvestri, Node masking: Making graph neural networks generalize and scale better, CoRR abs/2001.07524 (2020). arXiv:2001.07524.

P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, R. D. Hjelm, Deep graph infomax., ICLR (Poster) 2 (2019) 4.

Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Graph contrastive learning with adaptive augmentation, in: Proceedings of the Web Conference 2021, 2021, pp. 2069–2080.

L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: Bringing order to the web., Technical Report, Stanford InfoLab, 1999.

R. I. Kondor, J. Lafferty, Diffusion kernels on graphs and other discrete structures, in: Proceedings of the 19th international conference on machine learning, volume 2002, 2002, pp. 315–322.

F. Feng, X. He, J. Tang, T.-S. Chua, Graph adversarial training: Dynamically regularizing based on graph structure, IEEE Transactions on Knowledge and Data Engineering 33 (2019) 2493–2504.

L. Yang, L. Zhang, W. Yang, Graph adversarial self-supervised learning, Advances in Neural Information Processing Systems 34 (2021) 14887–14899.

Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, Y. Shen, Graph contrastive learning with augmentations, Advances in Neural Information Processing Systems 33 (2020a) 5812–5823.

Y. You, T. Chen, Z. Wang, Y. Shen, When does self-supervision help graph convolutional networks?, in: international conference on machine learning, PMLR, 2020b, pp. 10871–10880.

J. Park, H. Shim, E. Yang, Graph transplant: Node saliency-guided graph mixup with local structure preservation, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, 2022, pp. 7966–7974.

M. Jin, Y. Zheng, Y.-F. Li, C. Gong, C. Zhou, S. Pan, Multi-scale contrastive siamese networks for self-supervised graph representation learning, arXiv preprint arXiv:2105.05682 (2021).

R. Linsker, Self-organization in a perceptual network, Computer 21 (1988) 105–117.

Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Deep graph contrastive representation learning, arXiv preprint arXiv:2006.04131 (2020).

S. Thakoor, C. Tallec, M. G. Azar, R. Munos, P. Veličković, M. Valko, Bootstrapped representation learning on graphs, in: ICLR 2021 Workshop on Geometrical and Topological Representation Learning, 2021.

M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, M. Lucic, On mutual information maximization for representation learning, arXiv preprint arXiv:1907.13625 (2019).

Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, P. Isola, What makes for good views for contrastive learning?, Advances in Neural Information Processing Systems 33 (2020) 6827–6839.

Y. Yin, Q. Wang, S. Huang, H. Xiong, X. Zhang, Autogcl: Automated graph contrastive learning via learnable view generators, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, 2022, pp. 8892–8900.

S. Suresh, P. Li, C. Hao, J. Neville, Adversarial graph augmentation to improve graph contrastive learning, Advances in Neural Information Processing Systems 34 (2021) 15920–15933.

W. Cheng, D. Xu, H. Chen, Information-aware graph contrastive learning, 2022. US Patent App. 17/728,071.

K. Yang, Y. Sun, J. Su, F. He, X. Tian, F. Huang, T. Zhou, D. Tao, Adversarial auto-augment with label preservation: A representation learning principle guided approach, Advances in Neural Information Processing Systems 35 (2022) 22035–22048.

A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, A. J. Smola, Distributed large-scale natural graph factorization, in: Proceedings of the 22nd international conference on World Wide Web, 2013, pp. 37–48.

S. Cao, W. Lu, Q. Xu, Grarep: Learning graph representations with global structural information, in: Proceedings of the 24th ACM international on conference on information and knowledge management, 2015, pp. 891–900.

M. Ou, P. Cui, J. Pei, Z. Zhang, W. Zhu, Asymmetric transitivity preserving graph embedding, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 1105–1114.

B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

Z. Zhou, Y. Hu, Y. Zhang, J. Chen, H. Cai, Multiview deep graph infomax to achieve unsupervised graph embedding, IEEE Transactions on Cybernetics (2022).

E. Oyallon, Interferometric graph transform: a deep unsupervised graph representation, in: International Conference on Machine Learning, PMLR, 2020, pp. 7434–7444.

R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, Y. Bengio, Learning deep representations by mutual information estimation and maximization, arXiv preprint arXiv:1808.06670 (2018).

N. Tishby, F. C. Pereira, W. Bialek, The information bottleneck method, arXiv preprint physics/0004057 (2000).

Z. Goldfeld, Y. Polyanskiy, The information bottleneck problem and its applications in machine learning, IEEE Journal on Selected Areas in Information Theory 1 (2020) 19–38.

A. A. Alemi, I. Fischer, J. V. Dillon, K. Murphy, Deep variational information bottleneck, arXiv preprint arXiv:1612.00410 (2016).

D. J. MacKay, Information theory, inference and learning algorithms, Cambridge university press, 2003.

G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM Journal on scientific Computing 20 (1998) 359–392.

A. Nazi, W. Hang, A. Goldie, S. Ravi, A. Mirhoseini, Gap: Generalizable approximate graph partitioning framework, arXiv preprint arXiv:1903.00614 (2019).

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

X. Han, Z. Jiang, N. Liu, X. Hu, G-mixup: Graph data augmentation for graph classification, in: International Conference on Machine Learning, PMLR, 2022, pp. 8230–8248.

M. Zhang, Y. Chen, Link prediction based on graph neural networks, Advances in neural information processing systems 31 (2018).

S. Yang, B. Hu, Z. Zhang, W. Sun, Y. Wang, J. Zhou, H. Shan, Y. Cao, B. Ye, Y. Fang, et al., Inductive link prediction with interactive structure learning on attributed graph, in: Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21, Springer, 2021, pp. 383–398.

M. Welling, T. N. Kipf, Semi-supervised classification with graph convolutional networks, in: J. International Conference on Learning Representations (ICLR 2017), 2016.

C. J. Maddison, A. Mnih, Y. W. Teh, The concrete distribution: A continuous relaxation of discrete random variables, arXiv preprint arXiv:1611.00712 (2016).

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

K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, arXiv preprint arXiv:1810.00826 (2018).

P. Mernyei, C. Cangea, Wiki-cs: A wikipedia-based benchmark for graph neural networks, arXiv preprint arXiv:2007.02901 (2020).

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

C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, M. Neumann, Tudataset: A collection of benchmark datasets for learning with graphs, arXiv preprint arXiv:2007.08663 (2020).

Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, J. Huang, Graph representation learning via graphical mutual information maximization, in: Proceedings of The Web Conference 2020, 2020, pp. 259–270.

K. Hassani, A. H. Khasahmadi, Contrastive multi-view representation learning on graphs, in: International Conference on Machine Learning, PMLR, 2020, pp. 4116–4126.

W. Ju, Y. Gu, X. Luo, Y. Wang, H. Yuan, H. Zhong, M. Zhang, Unsupervised graph-level representation learning with hierarchical contrasts, Neural Networks 158 (2023) 359–368.

J. Wu, X. Chen, B. Shi, S. Li, K. Xu, Sega: Structural entropy guided anchor view for graph contrastive learning, in: International Conference on Machine Learning, PMLR, 2023, pp. 37293–37312.

H. Abdi, L. J. Williams, Principal component analysis, Wiley interdisciplinary reviews: computational statistics 2 (2010) 433–459.

L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of machine learning research 9 (2008).

T. Chen, S. Bian, Y. Sun, Are powerful graph neural nets necessary? a dissection on graph classification, arXiv preprint arXiv:1905.04579 (2019).

X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

**Declaration of interests**

☐ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The author is an Editorial Board Member/Editor-in-Chief/Associate Editor/Guest Editor for *[Journal name]* and was not involved in the editorial review or the decision to publish this article.

☒ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Furao Shen reports financial support was provided by the STI 2030-Major Projects of China under Grant 2021ZD0201300, the National Science Foundation of China under Grant 62276127.