SEGA: Structural Entropy Guided Anchor View for Graph Contrastive Learning

Junran Wu [1 *] **Xueyuan Chen** [1 *] **Bowen Shi** [1] **Shangzhe Li** [1] **Ke Xu** [1 2]

Abstract

In contrastive learning, the choice of "view" controls the information that the representation captures and influences the performance of the model. However, leading graph contrastive learning methods generally produce views via random corruption or learning, which could lead to the loss of essential information and alteration of semantic information. An anchor view that maintains the essential information of input graphs for contrastive learning has been hardly investigated. In this paper, based on the theory of graph information bottleneck, we deduce the definition of this anchor view; put differently, *the anchor view with essential information of input graph is supposed to have the minimal structural uncertainty*. Furthermore, guided by structural entropy, we implement the anchor view, termed **SEGA**, for graph contrastive learning. We extensively validate the proposed anchor view on various benchmarks regarding graph classification under unsupervised, semi-supervised, and transfer learning and achieve significant performance boosts compared to the state-of-the-art methods.

1. Introduction

Self-supervised learning has gained popularity recently and achieved great success in deep learning, $e.g.$, BERT (Devlin et al., 2019) and MoCo (He et al., 2020). Compared with supervised learning, self-supervised learning gets equal or even better performance with limited or no-labeled data which saves much annotation time and plenty of resources. As one of the empirical self-supervised learning methods, contrastive learning develops rapidly and recently has been applied to various domains because of the scarcity of datasets with labels. Meanwhile, graph neural networks (GNNs) have become ubiquitous for graphs because of their ability to model structural information (Li et al., 2022b; Zhang et al., 2022). Therefore, graph contrastive learning (You et al., 2020; Suresh et al., 2021; Yang & Hong, 2022), based on the success of contrastive learning in computer vision and natural language processing, has attracted plenty of research interest after its presentation.

In contrastive learning, the choice of view controls the information that the representation captures. Researchers have found that the quality of views influences the performance of contrastive learning models (Tian et al., 2020) and focus on the generation of effective views that lead to better performance for graphs through the data augmentation (Suresh et al., 2021; You et al., 2021). Despite the effectiveness of these graph views on various tasks, the proposed data augmentations via random corruption or learning suffer from structural damage and artificially introduced noise, which could alter the fundamental property of input graphs. Unlike images, data augmentation on graphs is much harder to provide high-quality contrastive samples due to the rich structural information of various contexts in the graph data (Feng et al., 2022). So far, little attention has been paid to the anchor view for graph contrastive learning that maintains the essential information of input graphs regarding graph classification. Therefore, we are eager to provide high-quality contrastive samples by settling the two questions: *(1) What is the anchor view holding essential information? (2) How to generate the anchor view for graph contrastive learning?*

Recently, the information bottleneck theory that encourages model to capture minimal but sufficient information, that is essential information, has been applied to learn graph representation, called Graph Information Bottleneck (GIB) (Wu et al., 2020). In light of GIB, we conclude that *the anchor view with essential information of input graph is supposed to have the minimal structural uncertainty* ($i.e., G^* = \min \mathcal{H}(G)$). Now, with the definition of target anchor view, the last question is its instantiation for graph contrastive learning. Thus, a metric for graph structural uncertainty measurement is needed. Recently, based on the classic uncertainty metric, Shannon entropy (Shannon,

[*]Equal contribution [1]State Key Lab of Software Development Environment, Beihang University, Beijing, 100191, China [2]Zhongguancun Laboratory, Beijing 100094, China. Correspondence to: Shangzhe Li <shangzheli@buaa.edu.cn>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

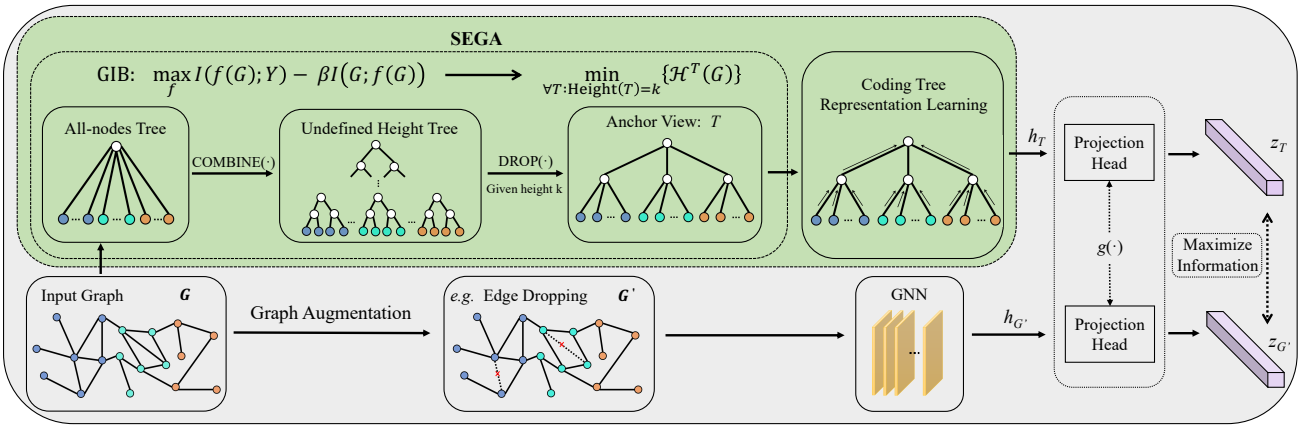


Figure 1. **Graph contrastive learning framework with SEGA.** The anchor view with essential information for graph contrastive learning is realized with the graph information bottleneck and structural entropy. The essential information of given graph is decoded to corresponding coding tree by structural entropy minimization. Under the architecture of contrastive learning, the model is trained to maximize the consensus between the coding tree representation h_T and the augmentation-based graph embedding $h_{G'}$.

1948), researchers proposed the structural entropy to measure the uncertainty of graph structures (Li & Pan, 2016). This theory implies that under intuition, people fear uncertainty and usually follow the choices that minimize such non-determinism. According to the structural information theory, the essential information embedded in the corresponding graph can be decoded by minimizing its structural uncertainty, that is to say, minimizing the structural entropy.

Here, in view of the definition of the anchor view given above, we propose its instantiation, termed **SEGA** (see Figure 1), guided by structural entropy minimization for graph contrastive learning. Specifically, an optimization algorithm is first introduced for structural entropy minimization, in which the coding trees of corresponding graphs are generated for essential information extraction. Then, based on the message-passing scheme in GNNs, an encoder is proposed to obtain the essential information held by the transformed coding trees. Contrasted with the effective views in previous works, extensive experiments, including unsupervised, semi-supervised, and transfer learning, are conducted on various benchmarks regarding graph classification. Superior performance can be observed in comparison with those state-of-the-art (SOTA) methods. The contributions of this work can be summarized as follows:

- Based on the theory of graph information bottleneck, to the best of our knowledge, we are the first to figure out the anchor view with essential information of input graphs for graph contrastive learning.
- Guided by structural entropy minimization, we present an instantiation, termed SEGA, to implement the proposed anchor view for graph contrastive learning.
- We extensively evaluate the proposed anchor view on various benchmarks under the setting of unsupervised, semi-supervised, and transfer learning, and obtain superior performance compared to the SOTA methods.

2. Related Works

Graph Contrastive Learning. Great success has been achieved by graph contrastive learning when facing the label scarcity in real-world network data (You et al., 2020; Suresh et al., 2021; You et al., 2021; Feng et al., 2022; You et al., 2022). However, unlike the data augmentation on images that do not require rich domain knowledge, the graph augmentation is far less intuitive and hard to analyze, which makes it difficult to produce high-quality contrast samples (Feng et al., 2022; You et al., 2022). Hence, the study of the graph contrastive view is a key issue in graph contrastive learning. Recently, based on the data augmentation on images, plenty of efforts have been devoted to exploring various augmentations on graphs (You et al., 2020; 2021; Suresh et al., 2021; You et al., 2022; Li et al., 2022a). While effective, none of them try to identify the essential information from graphs. Furthermore, the proposed data augmentations via random perturbation or learning suffer from structure damage and noisy information (You et al., 2020; Suresh et al., 2021). Recently, besides the view exploration, GraphLoG (Xu et al., 2021) and OEPG (Yang & Hong, 2022) are built upon the generic graph contrastive learning methods to discover the global semantic structure underlying the whole dataset. Although they present excellent performance, in this work, we are devoted to the domain of view generation, which is orthogonal to the works for dataset semantic structure exploration; put differently, ex-

tensive works for contrastive view generation can work with the framework of GraphLoG and OEPG to produce more superior performance.

Structural Entropy. Information entropy, as the basis of Information Theory, stems from the demand for information measuring in communication systems (Shannon, 1948). Considering measuring the information in graphs, lots of metrics were proposed. The entropy of graphs can be measured with $p(G)$ at the global level (Mowshowitz & Dehmer, 2012). For a signal graph, various works aim to measure the structural entropy of nodes. A local measurement of graph entropy was first proposed based on distance (Raychaudhury et al., 1984). Subsequently, extensive researches attempted to measure the structural information of graph from different angles, such as Von Neumann entropy (Braunstein et al., 2006), parametric graph entropy (Dehmer, 2008), Gibbs entropy (Bianconi, 2009). However, these definitions all destructure the graph into an unstructured probability distribution and then apply Shannon entropy to define the information of the graph. Thus, these metrics can not serve as the measurement of structural information that is crucial for graphs and the key to the success of GNNs. Recently, based on coding trees, structural entropy was proposed to evaluate the complexity of the hierarchical structure of a graph (Li & Pan, 2016). Considering the measurement of graph information with fixed hierarchical manners, k-dimensional structural entropy was further defined and can be used to decode the essential information of graphs (Li et al., 2018; Wu et al., 2022a;b).

3. Notations and Preliminaries

Some preliminary concepts and notations are introduced here. In this work, $\mathbb{G} = \{G_1, G_2, \cdots, G_M\}$ refers to a set of graphs, and each graph can be represented as a two tuple $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of nodes and edges.

Graph representation learning. In this work, GNNs with message-passing scheme are adopted as the encoders. A GNN aims to learn an embedding vector $h_v \in \mathbb{R}$ for each node and a vector $h_G \in \mathbb{R}$ for the entire graph G. A node representation h_v is initialized as $h_v^{(0)} = X_v$, and will be iteratively updated by an encoder. For an L-layer GNN, each node presentation will be updated using $L-$hop information from surrounding nodes. The l-layer of a GNN (Gilmer et al., 2017) can be expressed as

$$h_v^{(l)} = f_U^{(l)}(h_v^{(l\text{-}1)}, f_M^{(l)}(\{(h_v^{(l\text{-}1)}, h_u^{(l\text{-}1)})|u \in N(v)\})), \quad (1)$$

where $N(v)$ is the neighborhood node set for v, $h_v^{(l)}$ is the node representation of v in the l-th layer, $f_U^{(l)}$ denotes the update function in the l-th layer, and $f_M^{(l)}$ refers to the trainable message-passing function in the l-th layer. h_v can be referred to a summary of neighborhood nodes, like a subgraph. Thus, after L iterations, the entire graph representation can be formalized as follows:

$$h_G = f_R(\{h_v|v \in \mathcal{V}\}), \quad (2)$$

where f_R is the readout function which pools the final set of node representations.

Graph contrastive learning. In a generic contrastive learning model for graph classification, two corresponding views of the same graph G_i are generally generated by two data augmentation operators and serve as a positive pair. Let $\tilde{G}_i^1$ and $\tilde{G}_i^2$ be the two augmented views; then, a GNN-based encoder is adopted to model the structural information underlying the given graph. In the pre-training phase, a projection head is further employed to map the two views into an embedding space for contrasting. The released feature vectors h_i^1 and h_i^2 are designed to identify themselves from the others. Correspondingly, the NT-Xent loss (Chen et al., 2020) helps to achieve the goal of graph contrastive learning that maximizes the consensus of two correlated views:

$$\mathcal{L}_i = -\log \frac{e^{sim(h_i^1, h_i^2)/\tau}}{\sum_{j=1, j\neq i}^{N} e^{sim(h_i^1, h_j^2)/\tau}}, \quad (3)$$

where N denotes the batch size, τ is the temperature parameter, and $sim(h^1, h^2)$ is generally implemented by a cosine similarity function $\frac{h^{1\top} h^2}{||h^1|| \cdot ||h^2||}$.

4. Methodology

In this section, we first introduce our theoretical motivation and try to give the definition of the anchor view with essential information. Based on the structural information theory, we then present an instantiation of the anchor view for graph contrastive learning.

4.1. The Anchor View Holding Essential Information

Based on the idea of information bottleneck, GIB (Wu et al., 2020) presents a statement that motivates us to think about the anchor view that maintains the essential information of input graphs. Specifically, through maximizing the mutual information (MI) between the output and target (i.e., $\max I(f(G); Y)$) while stinting such information between the input and output (i.e., $\min I(G; f(G))$, models are capable of learning minimal but sufficient information for a given task. Therefore, the formal description of essential information is given by

Definition 4.1. The **essential information** learned from the input graphs is supposed to be the minimal sufficient information required for a downstream prediction task.

In computer vision, researchers empirically gave a similar answer for contrastive learning; put differently, compressing the mutual information between views while maintaining

the integrity of information related to downstream tasks (Tian et al., 2020), which further convinces us to build the target anchor view via the GIB. Formally, the objective of graph information bottleneck can be written as

$$\text{GIB:} \quad \max I(f(G); Y) - \beta I(G; f(G)), \tag{4}$$

where $(G, Y) \sim \mathbb{P}_{\mathcal{G}\times\mathcal{Y}}$ and $\beta > 0$.

Here, we give the definition of the target view:

Definition 4.2. The anchor view for graph contrastive learning is supposed to have minimal but sufficient information for downstream tasks and help the other view learn such information during training.

In the light of GIB, to acquire the essential information, the anchor view for graph contrastive learning should have minimal but sufficient information for downstream tasks. However, unfortunately, the first part of GIB requires the target-relevant information of the given task (i.e., Y), and as we know, it is impossible under the architecture of self-supervised training. [1] In this context, the other part that does not require such target-related information sheds light on the path of the target anchor view exploration. Therefore, minimizing the mutual information between the learned representation and input graph (i.e., $\min I(G; f(G))$) suggests the essential information that graph contrastive learning should conquer. Formally, we have

$$\begin{aligned}\text{GIB:} \ & \max I(f(G); Y) - \beta I(G; f(G)), \\ & \Rightarrow \min I(G; f(G)).\end{aligned} \tag{5}$$

Here, we first give a property that the target anchor view should own:

Definition 4.3. The anchor view with essential information is supposed to be a substructure of the given graph to avoid artificially introduced noise.

In computer vision, to obtain the essential information, data augmentation via random perturbation has been ubiquitously adopted, and the accompanying induced noise is also approved for robust representation learning (Tian et al., 2020). However, unlike the data augmentation on images that does not require rich domain knowledge, the graph augmentation is far less intuitive and hard to analyze, which makes it difficult to produce high-quality contrast samples (Feng et al., 2022; You et al., 2022). Thus, we argue that the anchor view with essential information of given graphs should avoid the artificially introduced noise from random perturbation. Now, let G^* be the target anchor view of a graph G, the mutual information between G and G^* can be formulated as

$$I(G^*; G) = \mathcal{H}(G^*) - \mathcal{H}(G^*|G), \tag{6}$$

where $\mathcal{H}(G^*)$ is the entropy of G^* and $\mathcal{H}(G^*|G)$ is the conditional entropy of G^* conditioned on G. [2]

Theorem 4.4. *According to Definition 4.3, the information in G^* is a subset of information in G (i.e., $\mathcal{H}(G^*) \subseteq \mathcal{H}(G)$); thus, we have:*

$$\mathcal{H}(G^*|G) = 0. \tag{7}$$

The detailed proof of Theorem 4.4 is shown in Appendix A. Here, the mutual information between G and G^* can be rewritten as

$$I(G^*; G) = \mathcal{H}(G^*). \tag{8}$$

Accordingly, to acquire the anchor view with essential information, we need to optimize:

$$\min I(G; f(G)) \Rightarrow \min \mathcal{H}(G^*). \tag{9}$$

Besides information measuring, $\mathcal{H}(G^*)$ also reveal the uncertainty of G^* based on the definition of Shannon entropy (Shannon, 1948). Correspondingly, the definition of the anchor view with essential information for graph contrastive learning is given by:

Definition 4.5. The anchor view with essential information of input graph is supposed to have the minimal structural uncertainty.

Remark. In this paper, base on GIB theory (Wu et al., 2020), we conclude that the anchor view with essential information is supposed to satisfy $\mathcal{H}(G^*)$, and we interpret it from the topological perspective as the view with minimal structural uncertainty. First, this is because topological structures are ubiquitous in complex network systems, while node features may not be always available. For example, in the social network datasets adopted in this study, only topological structures are available and node features are absent. Furthermore, current research on contrastive views mainly focuses on data augmentation methods based on topological structures such as edge perturbation, node dropout and subgraph extraction in GraphCL (You et al., 2020); learning edge dropout in AD-GCL (Suresh et al., 2021); and learning node selection in RGCL (Li et al., 2022a). Therefore, we hope to explore contrastive views with essential information from a more general angle, that is topological structure.

Next, we will elaborate on the instantiation of the defined anchor view by introducing structural information theory.

4.2. Instantiation of Anchor View

In this subsection, we are going to introduce a practical instantiation, a structural entropy guided anchor view (i.e., **SEGA**), for essential information decoding.

[1] Based on the property of structural entropy, the information for downstream tasks within the proposed anchor view G^* is still more than views from augmentations. Details refer to Theorem 4.6.

[2] We omit the graph encoder f for simplicity.

Despite the broader applicability of Shannon entropy, in this work, we need the metric of structural uncertainty for graphs, which has also been asked by Brooks in the "Three great challenges for half-century-old computer science" (Brooks Jr, 2003). The question is how to define the underlying information of a graph so that the essential information of the graph could be decrypted, while Shannon wondered the feasibility of communication graph analysis via a structural theory of information (Shannon, 1953). Recently, structural entropy defined on graphs was proposed to measure the uncertainty of the graph structure (Li & Pan, 2016). According to this structural information theory, a graph is encoded by a coding tree. [3] The structural entropy of graph $G = (\mathcal{V}, \mathcal{E})$ on a coding tree T is defined as

$$\mathcal{H}^T(G) = -\sum_{v_\tau \in T} \frac{g_{v_\tau}}{vol(\mathcal{V})} \log \frac{vol(v_\tau)}{vol(v_\tau^+)}, \tag{10}$$

where v_τ is a nonroot node in T and represents a node subset $\mathcal{V}_\tau \subset \mathcal{V}$ based on its covered leaf nodes, g_{v_τ} is the number of edges with exactly one vertex in $\mathcal{V}_\tau$, v_τ^+ refers to the immediate predecessor of v_τ, and $vol(\mathcal{V})$, $vol(v_\tau)$ and $vol(v_\tau^+)$ are the sums of degrees of vertices in $\mathcal{V}$, v_τ and v_τ^+, respectively. Thus, to decode the essential information of graph G with minimal structural uncertainty, we need to realize the optimal coding tree T with minimum entropy (i.e., $\min_T \mathcal{H}^T(G)$). Besides the optimal coding tree, considering that a real-world network generally has a natural structure with a fixed hierarchy, a coding tree with the corresponding fixed height is preferred. In this context, k-dimensional structural entropy is used to decode the optimal coding tree with a certain height k:

$$\mathcal{H}^k(G) = \min_{\forall T:\text{Height}(T)=k} \{\mathcal{H}^T(G)\}. \tag{11}$$

Now, in light of the structural information theory, we know that the essential information of input graph can be decoded by minimizing its structural entropy. Moreover, the target anchor view for graph contrastive learning with minimum but sufficient information is the coding tree of the graph through k-dimensional structural entropy minimization.

Theorem 4.6. *Given the target anchor view G^* and a general data augmentation function t, we have*

$$I(G^*; Y) \geq I(t(G); Y). \tag{12}$$

Proof. Suppose the encoder f is implemented by a GNN. The optimal encoder f^* is the best model which GNN can find. According to the definition $f^* = argmax_f I(f(G); G)$, f^* should be injective. Given the target anchor view G^*, $G^* \Rightarrow f^*(G^*)$ is an injective deterministic mapping. Thus, for any random variable Q,

$$I(f^*(G^*); Q) = I(G^*; Q). \tag{13}$$

When there is $Q = Y$, we will have,

$$I(f^*(G^*); Y) = I(G^*; Y). \tag{14}$$

In light of the property of structural information theory (Li & Pan, 2016), structural entropy decodes the essential structure of the original system while measuring the structural information to support the semantic analysis of the system. Thus, we have

$$I(f^*(G^*); Y) = I(f^*(G); Y). \tag{15}$$

Now, introducing the data processing inequality (Thomas & Joy, 2006) for data augmentation,

$$\begin{aligned} I(f^*(G); Y) &= I(G; Y) \\ &\geq I(t(G); Y) = I(f^*(t(G)); Y). \end{aligned} \tag{16}$$

Combining above equations, we can have

$$\begin{aligned} I(G^*; Y) &= I(f^*(G^*); Y) \\ &= I(f^*(G); Y) \\ &\geq I(f^*(t(G)); Y) = I(t(G); Y), \end{aligned} \tag{17}$$

which concludes the proof. □

Theorem 4.6 guarantees a lower bound of the mutual information between the learned representations and the labels of the downstream task; put differently, the learned essential information with the anchor view G^* is more than views from augmentations.

For structural entropy minimization, we aim to decrypt the coding tree with fixed height k from a graph. Given a graph $G = (\mathcal{V}, \mathcal{E})$, a coding tree T can be build, in which v_r is the root node of T and $\mathcal{V}$ are the leaf nodes of T. For the coding tree T, there are two function definitions.

Algorithm 1 Structural uncertainty minimization

Input: the given height $k > 1$, and the candidate graph $G = (\mathcal{V}, \mathcal{E})$

Output: a coding tree T that meets the height bar

Build a coding tree T with a root node v_r and all nodes in $\mathcal{V}$ as its children;
// Stage 1: Construct a full-height binary coding tree from bottom to top;
while $|v_r.children| > 2$ **do**
  COMBINE$(v_c^1, v_c^2) \leftarrow argmax_{(v_c^1, v_c^2)}\{\mathcal{H}^T(G) - \mathcal{H}^{T_{\text{COMBINE}(v_c^1, v_c^2)}}(G) | v_c^1, v_c^2 \in v_r.children\}$;
end while
// Stage 2: Squeeze T to meet the height bar;
while Height$(T) > k$ **do**
  DROP$(v_\tau) \leftarrow argmin_{v_\tau}\{\mathcal{H}^{T_{\text{DROP}(v_\tau)}}(G) - \mathcal{H}^T(G) | v_\tau \in T \,\&\, v_\tau \neq v_r \,\&\, v_\tau \notin \mathcal{V}\}$;
end while
return T;

[3] A detailed description and illustration for coding tree T on given graph G can be found in Appendix B.

Definition 4.7. Given any two child nodes of v_r, v_c^1 and v_c^2, there is a function COMBINE(v_c^1, v_c^2) for T to add a new node v_i between v_r and (v_c^1, v_c^2):

$$v_i.children = \{v_c^1, v_c^2\}, \tag{18}$$

$$v_r.children = \{v_i\} \cup v_r.children. \tag{19}$$

Definition 4.8. Given a pair of nodes (v_τ, v_τ^+) in T, there is a function DROP(v_τ) for T to drop node v_τ and fuse the children of v_τ into v_τ^+:

$$v_\tau^+.children = v_\tau^+.children \cup v_\tau.children. \tag{20}$$

Based on the two defined operators, Algorithm 1 shows the realization of structural uncertainty minimization. Specifically, given a coding tree including only root node and leaf nodes, a full-height binary coding tree grows from bottom to top. In this process, two child nodes of root are combined to form a new division in each iteration, which aims to minimize the structural entropy. $T_{\text{COMBINE}(v_c^1, v_c^2)}$ denotes the coding tree that has combined the two children (i.e., v_c^1 and v_c^2) of the root node. Then, considering the height limitation, the well developed coding tree needs to be squeezed. During each iteration, an inner-node from T will be dropped until its height meets the bar. In particular, each dropped node should ensure that T has the minimized structural entropy after each iteration. $T_{\text{DROP}(v_\tau)}$ is the coding tree that has dropped the inner node v_τ. In the end, a fixed height coding tree $T = (\mathcal{V}^T, \mathcal{E}^T)$ will be obtained, in which $\mathcal{V}^T = (\mathcal{V}_0^T, \ldots, \mathcal{V}_k^T)$ and $\mathcal{V}_0^T = \mathcal{V}$. The running process of Algorithm 1 is illustrated in Appendix B.

Anchor view representation learning. Having the algorithm for structural uncertainty minimization, we are capable of producing the anchor view with essential information for graph contrastive learning. To further integrate the coding tree into the architecture of contrastive learning, we give an encoder for the anchor view representation learning. In light of the graph convolution scheme in GNNs, the coding tree encoder is designed to iteratively transfer messages from bottom to top. Specifically, based on the hierarchical structure of the coding tree and the initial node feature of leaves, the non-leaf nodes update their hidden representation by aggregating the hidden features from their children. Formally, the i-th layer of the encoder can be written as, $x_v^i = \text{MLP}^i\left(\sum_{u \in \mathcal{L}(v)} x_u^{(i-1)}\right)$, where x_v^i is the feature of v in the i-th layer of coding tree T, x_v^0 is the input feature of leaf nodes, and $\mathcal{L}(v)$ refers to the children of v.

5. Experiments

In this section, we are devoted to evaluating SEGA with extensive experiments [4]. Note that the proposed anchor view is orthogonal to previous works for graph augmentations, and this also reveals that our method has a superior collaborative capability with previous methods. Therefore, we first validate SEGA via contrasting with the well-known rules for graph augmentations from GraphCL (the first graph contrastive learning method with augmentations) (You et al., 2020). Then, thorough orthogonal experiments are performed to show the superiority of SEGA against SOTA competitors. Further ablation studies are conducted to make an in-depth analysis of SEGA.

[4]The code of SEGA is available at `https://github.com/Wu-Junran/SEGA`.

5.1. Contrastive Learning with Simple Rules

Datasets. For unsupervised and semi-supervised learning, various benchmarks are adopted from TUDataset (Morris et al., 2020), including COLLAB, REDDIT-BINARY, REDDIT-MULTI-5K, IMDB-BINARY, GITHUB, NCI1, MUTAG, PROTEINS and DD. For transfer learning, ZINC15 (Sterling & Irwin, 2015) dataset is adopted for biochemical pre-training. In particular, a subset with two million unlabeled molecular graphs are sampled from the ZINC15. For protein domain, following Hu et al. (2020), 306K unlabeled protein ego-networks are utilized for pre-training. We employ the eight ubiquitous benchmarks from the MoleculeNet dataset (Wu et al., 2018) as the biochemical downstream experiments. The protein downstream task is to predict 40 fine-grained biological functions of 8 species. Further details are shown in Appendix C.

Learning protocol. Following the learning setting in GraphCL (You et al., 2020), the corresponding learning protocols are adopted for a fair comparison. (a) In unsupervised representation learning, all data is used for model pre-training and the learned graph embeddings are then fed into a non-linear SVM classifier to perform 10-fold cross-validation. (b) In transfer learning, we first pre-train the model on ZINC15 and PPI306K. Then, we finetune and evaluate the model on MoleculeNet dataset and PPI using the scaffold split scheme (Chen et al., 2012). (c) In semi-supervised learning, there exist two learning settings. For datasets with a public training/validation/test split, pre-training is performed only on training dataset, finetuning is conducted with 10% of the training data, and final evaluation results are from the validation/test sets. For datasets without such splits, all samples are employed for pre-training while finetuning and evaluation are performed over 10 folds.

Configuration. To keep in line with GraphCL (You et al., 2020), the same GNN architectures are employed with their original hyper-parameters under individual experiment settings. Specifically, in unsupervised learning, GIN (Xu et al., 2019) with 32 hidden units and 3 layers is set up. In addition, the same data augmentations on graphs with the default augmentation strength 0.2 are adopted. In transfer learning, GIN is used with 5 layers and 300 hidden dimensions. In

Table 1. Average accuracies (%) ± Std. of compared methods via unsupervised learning. **Bold** indicates the best performance over all methods. A.A. refers to the average accuracy over eight benchmarks. A.R. implies the abbreviation of average rank. The results of baselines are derived from the published works and - indicates the data missing in the such works.

	NCI1	PROTEINS	DD	MUTAG	COLLAB	RED-B	RED-M5K	IMDB-B	A.A.	A.R.
GL	-	-	-	81.66±2.11	-	77.34±0.18	41.01±0.17	65.87±0.98	-	6.5
WL	80.01±0.50	72.92±0.56	-	80.72±3.00	-	68.82±0.41	46.06±0.21	72.30±3.44	-	5.3
DGK	**80.31±0.46**	73.30±0.82	-	87.44±2.72	-	78.04±0.39	41.27±0.18	66.96±0.56	-	4.3
node2vec	54.89±1.61	57.49±3.57	-	72.63±10.20	-	-	-	-	-	7.7
sub2vec	52.84±1.47	53.03±5.55	-	61.05±15.80	-	71.48±0.41	36.69±0.42	55.26±1.54	-	8.5
graph2vec	73.22±1.81	73.30±2.05	-	83.15±9.25	-	75.78±1.03	47.86±0.26	71.10±0.54	-	5.3
MVGRL	-	-	-	75.40±7.80	-	82.00±1.10	-	63.60±4.20	-	6.7
InfoGraph	76.20±1.06	74.44±0.31	72.85±1.78	89.01±1.13	70.65±1.13	82.50±1.42	53.46±1.03	73.03±0.87	74.02	2.9
GraphCL	77.87±0.41	74.39±0.45	78.62±0.40	86.80±1.34	71.36±1.15	89.53±0.84	55.99±0.28	71.14±0.44	75.71	2.9
SEGA	79.00±0.72	**76.01±0.42**	**78.76±0.57**	**90.21±0.66**	**74.12±0.47**	**90.21±0.65**	**56.13±0.30**	**73.58±0.44**	**77.25**	**1.3**

Table 2. Average test ROC-AUC (%) ± Std. over different 10 runs of SEGA along with all baselines on nine downstream benchmarks. The results of baselines are derived from the corresponding works. **Bold** indicates the best performance among all baselines. Avg. shows the average ROC-AUC over all datasets.

	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	PPI	Avg.
No Pre-Train	65.8±4.5	74.0±0.8	63.4±0.6	57.3±1.6	58.0±4.4	71.8±2.5	75.3±1.9	70.1±5.4	64.8±1.0	66.72
Infomax	68.8±0.8	75.3±0.6	62.7±0.4	58.4±0.8	69.9±3.0	75.3±2.5	76.0±0.7	75.9±1.6	64.1±1.5	69.60
EdgePred	67.3±2.4	76.0±0.6	64.1±0.6	60.4±0.7	64.1±3.7	74.1±2.1	76.3±1.0	**79.9±0.9**	65.7±1.3	69.76
AttrMasking	64.3±2.8	76.7±0.4	64.2±0.5	61.0±0.7	71.8±4.1	74.7±1.4	77.2±1.1	79.3±1.6	65.2±1.6	69.38
ContextPred	68.0±2.0	75.7±0.7	63.9±0.6	60.9±0.6	65.9±3.8	75.8±1.7	77.3±1.0	79.6±1.2	64.4±1.3	70.17
GraphCL	69.68±0.67	73.87±0.66	62.40±0.57	60.53±0.88	75.99±2.65	69.80±2.66	**78.47±1.22**	75.38±1.44	67.88±0.85	70.44
SEGA	**71.86±1.06**	**76.72±0.43**	**65.23±0.91**	**63.68±0.34**	**84.99±0.94**	**76.60±2.45**	77.63±1.37	77.07±0.46	**68.73±0.54**	**73.61**

semi-supervised learning, ResGCN with 128 hidden units and 5 layers is set up for pre-training and finetuning.

As for the anchor view representation learning, the number of tree encoder layer is consistent with the tree height, which ranges from 2 to 5 and the MLP in each iteration has 2 layers. The encoder hidden dimensions are fixed for all layers to keep in line with GraphCL under individual experiment setting. Additional details are shown in the Appendix D.

Unsupervised learning. The compared methods in unsupervised learning have three categories. The published hyper-parameters of these methods are adopted. The first set is three SOTA kernel-based methods that include GL (Shervashidze et al., 2009), WL (Shervashidze et al., 2011), and DGK (Yanardag & Vishwanathan, 2015). The second set is four heuristic self-supervised methods, including node2vec (Grover & Leskovec, 2016), sub2vec (Adhikari et al., 2018), graph2vec (Annamalai Narayanan & Jaiswal, 2017), and InfoGraph (Sun et al., 2020). The final compared methods are MVGRL (Hassani & Khasahmadi, 2020) and GraphCL (You et al., 2020) for unsupervised learning.

The classification accuracies of SEGA contrasted with simple augmentation rules under the setting of unsupervised learning are shown in Table 1, and a significant performance improvement from the appearance of the target anchor view can be witnessed as opposed to the baselines. Specifically, in light of the last column for average rank, SEGA acquires the highest position among the ten methods. Moreover, as can be seen from the column for average accuracy, SEGA outperforms InfoGraph and GraphCL with a 3.22% and 1.54% accuracy gain. In particular, except for the performance on NCI1, SEGA obtains the best performance on the other seven benchmarks, and we still can observe that SEGA obtains the highest accuracy over all eight benchmarks under the scenario without kernel-based methods. Thus, we can conclude that better performance can consistently be achieved when contrasting with the proposed anchor view.

Transfer learning. The baseline methods under the setting of transfer learning include EdgePred, AttrMsking, ContexPred (Hu et al., 2020), Infomax (Velickovic et al., 2019) and GraphCL (You et al., 2020). A model without pre-train, termed 'No Pre-Train', is also adopted for comparison.

The results of SEGA, along with baselines under the setting of transfer learning on nine benchmarks, are shown in Table 2. To summarize, the proposed anchor view, SEGA, obtains superior performance compared to previous works. Specifically, SEGA achieves the best performance on seven out of nine benchmarks, and a 3.17% performance gain is obtained in terms of average ROC-AUC compared to GraphCL. Thus, we can conclude that the proposed anchor view servers as a good contrastive branch to help the graph encoder model essential information of given graphs and improve generalization and performance.

Semi-supervised learning. Five baselines are adopted for semi-supervised learning, including a naive GCN without

Table 3. Average accuracies (%) ± Std. of compared methods via semi-supervised representation learning with 10% labels. **Bold** indicates the best performance over all methods. A.A. is short for average accuracy. The results of baselines are derived from the published works.

	NCI1	PROTEINS	DD	COLLAB	RED-B	RED-M5K	GITHUB	A.A.
No Pre-Train	73.72±0.24	70.40±1.51	73.56±0.41	73.71±0.27	86.63±0.27	51.33±0.44	60.87±0.17	70.03
GAE	74.36±0.24	70.51±0.17	74.54±0.68	75.09±0.19	87.69±0.40	53.58±0.13	63.89±0.52	71.38
ContextPred	73.00±0.30	70.23±0.63	74.66±0.51	73.69±0.37	84.76±0.52	51.23±0.84	62.35±0.73	69.99
Infomax	74.86±0.26	72.27±0.40	75.78±0.34	73.76±0.29	88.66±0.95	53.61±0.31	65.21±0.88	72.02
GraphCL	74.63±0.25	74.17±0.34	76.17±1.37	74.23±0.21	89.11±0.19	52.55±0.45	65.81±0.79	72.38
SEGA	**75.09±0.22**	**74.65±0.54**	**76.33±0.43**	**75.18±0.22**	**89.40±0.23**	**53.73±0.28**	**66.01±0.66**	**72.92**

Table 4. Orthogonal experiment results (%) of SEGA with SOTAs in unsupervised representation learning. **Bold** indicates the best performance within each specific opeartion. A.A. shows the average accuracy over all datasets. The results of baselines are derived from the published works and - indicates the data missing in the such works.

	NCI1	PROTEINS	DD	MUTAG	COLLAB	RED-B	RED-M5K	IMDB-B	IMDB-M	A.A.
AD-GCL-FIX	69.67±0.51	73.59±0.65	74.49±0.52	89.25±1.45	73.32±0.27	85.52±0.79	53.00±0.82	71.57±1.01	49.04±0.53	71.05
SEGA-AD-GCL-FIX	**70.38±0.76**	**74.61±0.81**	**75.84±0.64**	**89.89±0.69**	**75.03±0.36**	**87.74±0.39**	**54.29±0.54**	**72.32±0.49**	**50.83±0.34**	**72.33**(↑1.28)
JOAO	**78.07±0.47**	74.55±0.41	77.32±0.54	87.35±1.02	69.50±0.36	85.29±1.35	55.74±0.63	70.21±3.08	-	74.75
SEGA-JOAO	76.19±0.77	**75.44±0.54**	**78.27±1.32**	**87.70±1.31**	**72.82±0.35**	**86.79±1.31**	**56.17±0.67**	**71.74±1.26**	-	**75.64**(↑0.89)
JOAOv2	**78.36±0.53**	74.07±1.10	77.40±1.15	87.67±0.79	69.33±0.34	86.42±1.45	56.03±0.27	70.83±0.25	-	75.01
SEGA-JOAOv2	78.04±0.19	**75.94±0.88**	**78.37±1.26**	**88.53±2.45**	**72.76±0.27**	**87.98±0.29**	**56.15±0.29**	**72.12±0.79**	-	**76.24**(↑1.23)
AutoGCL	**82.00±0.29**	75.80±0.36	77.57±0.60	88.64±1.08	70.12±0.68	88.58±1.49	56.75±0.18	73.30±0.40	-	76.59
SEGA-AutoGCL	81.84±0.53	**76.43±0.67**	**78.31±1.37**	**89.03±1.01**	**72.68±0.23**	**89.88±1.21**	**57.43±0.37**	**73.95±0.87**	-	**77.44**(↑0.85)
RGCL	78.14±1.08	75.03±0.43	78.86±0.48	87.66±1.01	70.92±0.65	90.34±0.58	56.38±0.40	71.85±0.84	-	76.15
SEGA-RGCL	**79.42±0.82**	**75.87±0.45**	**79.54±1.14**	**88.79±1.87**	**73.14±0.37**	**90.75±0.84**	**57.28±0.42**	**72.75±0.66**	-	**77.19**(↑1.04)

pre-training (You et al., 2020), GAE (Kipf & Welling, 2016), Infomax (Velickovic et al., 2019), ContextPred (Hu et al., 2020) and GraphCL (You et al., 2020).

The classification accuracies of SEGA and compared methods under the setting of semi-supervised learning are shown in Table 3, and SEGA outperforms these compared methods across all benchmarks. Despite the least performance improvement, the effectiveness of our proposed anchor view still has been validated in semi-supervised learning.

5.2. Orthogonal to SOTAs

As mentioned above, the proposed anchor view is orthogonal to previous works for graph augmentations; thus, we further evaluate SEGA in collaboration with these augmented views in unsupervised learning setting, including AD-GCL (Suresh et al., 2021), JOAO (You et al., 2021), AutoGCL (Yin et al., 2022) and RGCL (Li et al., 2022a). Detailed settings for orthogonal experiments are shown in Section D.4.

The orthogonal results are shown in Table 4, and we can see that a general performance improvement is achieved with the SEGA. Despite several specific failures, 0.85%∼1.28% average accuracy gains confirm the effectiveness of SEGA as an anchor view for graph contrastive learning.

5.3. Ablation Study

Here, we make an in-depth analysis about the performance of SEGA under the setting of unsupervised learning.

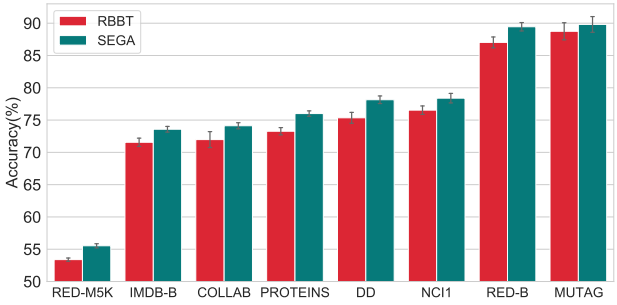


Figure 2. Performance comparison between RBBT and SEGA.

Guidance from structural entropy. Besides the superior performance of SEGA, we further evaluate the effectiveness of Algorithm 1 for structural uncertainty minimization. In unsupervised learning, we produce the anchor view without guidance from structural entropy but adopt a random coding tree, i.e., a randomly balanced binary tree (RBBT) with a height of two. We also fix the height of the guided anchor view to two for fair comparison. The results are shown in Figure 2 and the structural entropy-guided anchor view surpasses the random coding tree on all eight benchmarks.

The height k of graph's natural hierarchy. In experimental setup, the height k of coding tree ranges from 2 to 5. Here, we delve deeper into the optimal height k of graph's natural hierarchy. The specific performance of SEGA under each height k via unsupervised learning is shown in Fig-

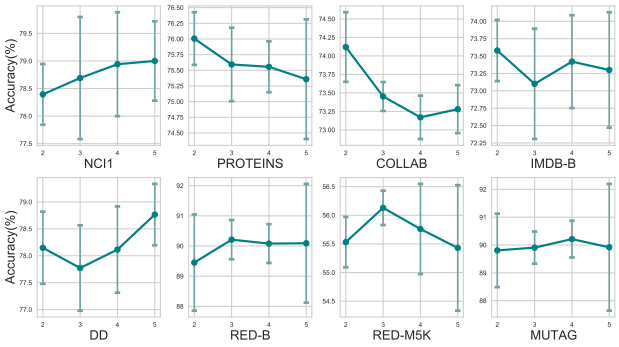


Figure 3. The natural hierarchy of graph.

ure 3. As can be seen, the optimal height k with the highest accuracy varies among datasets. Except for NCI1 and DD, the other six benchmarks achieve the best performance with shallow layers (less than 5), and the rising trend of NCI1 and DD also implies the great potential of SEGA.

6. Conclusion

In this work, we try to explore an anchor view that maintains the essential information of input graphs for graph contrastive learning. In the light of the graph information bottleneck, we attempt to give the definition of the expected anchor view. Moreover, based on the structural information theory, we present a practical instantiation, called SEGA, to implement this anchor view for graph contrastive learning. Contrasted with extensive views in previous works, SEGA shows superior performance on tasks regarding graph classification compared to SOTAs. An anchor view that maintains the essential information for node classification sheds light on our future research direction.

Acknowledgements

This research was supported by NSFC (Grant No. 61932002).

References

Aids antiviral screen data. Accessed: 2017-09-27, `https://wiki.nci.nih.gov/display/NCIDTPdata/AIDS+Antiviral+Screen+Data`.

Tox21 data challenge. Accessed: 2017-09-27, `https://tripod.nih.gov/tox21/challenge`, 2014.

Adhikari, B., Zhang, Y., Ramakrishnan, N., and Prakash, B. A. Sub2vec: Feature learning for subgraphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 170–182. Springer, 2018.

Annamalai Narayanan, Mahinthan Chandramohan, R. V. L. C. Y. L. and Jaiswal, S. graph2vec: Learning distributed representations of graphs. In *Proceedings of the 13th International Workshop on Mining and Learning with Graphs (MLG)*, 2017.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., et al. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

Bemis, G. W. and Murcko, M. A. The properties of known drugs. 1. molecular frameworks. *Journal of Medicinal Chemistry*, 39(15):2887–2893, 1996.

Bianconi, G. Entropy of network ensembles. *Physical Review E*, 79(3):036114, 2009.

Braunstein, S. L., Ghosh, S., and Severini, S. The laplacian of a graph as a density matrix: a basic combinatorial approach to separability of mixed states. *Annals of Combinatorics*, 10(3):291–317, 2006.

Brooks Jr, F. P. Three great challenges for half-century-old computer science. *Journal of the ACM (JACM)*, 50(1): 25–26, 2003.

Chen, B., Sheridan, R. P., Hornak, V., and Voigt, J. H. Comparison of random forest and pipeline pilot naive bayes in prospective qsar predictions. *Journal of Chemical Information and Modeling*, 52(3):792–803, 2012.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *ICML*, pp. 1597–1607. PMLR, 2020.

Consortium, G. O. The gene ontology resource: 20 years and still going strong. *Nucleic Acids Research*, 47(D1): D330–D338, 2019.

Dehmer, M. Information processing in complex networks: Graph entropy and information functionals. *Applied Mathematics and Computation*, 201(1-2):82–94, 2008.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Feng, S., Jing, B., Zhu, Y., and Tong, H. Adversarial graph contrastive learning with information regularization. In *Proceedings of the ACM Web Conference 2022*, pp. 1362–1371, 2022.

Gardiner, E. J., Holliday, J. D., O'Dowd, C., and Willett, P. Effectiveness of 2d fingerprints for scaffold hopping. *Future Medicinal Chemistry*, 3(4):405–414, 2011.

Gayvert, K. M., Madhukar, N. S., and Elemento, O. A data-driven approach to predicting successes and failures of clinical trials. *Cell Chemical Biology*, 23(10):1294–1301, 2016.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, pp. 1263–1272. PMLR, 2017.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. *SIGKDD*, pp. 855–864, 2016.

Hassani, K. and Khasahmadi, A. H. Contrastive multi-view representation learning on graphs. In *ICML*, pp. 4116–4126. PMLR, 2020.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for pre-training graph neural networks. *International Conference on Learning Representations (ICLR)*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

Klopfenstein, D., Zhang, L., Pedersen, B. S., Ramírez, F., Warwick Vesztrocy, A., Naldi, A., Mungall, C. J., Yunes, J. M., Botvinnik, O., Weigel, M., et al. Goatools: A python library for gene ontology analyses. *Scientific Reports*, 8(1):1–17, 2018.

Kuhn, M., Letunic, I., Jensen, L. J., and Bork, P. The sider database of drugs and side effects. *Nucleic Acids Research*, 44(D1):D1075–D1079, 2016.

Landrum, G. Rdkit documentation. *Release*, 1(1-79), 2013.

Li, A. and Pan, Y. Structural information and dynamical complexity of networks. *IEEE Transactions on Information Theory*, 62(6):3290–3339, 2016.

Li, A. L., Yin, X., Xu, B., Wang, D., Han, J., Wei, Y., Deng, Y., Xiong, Y., and Zhang, Z. Decoding topologically associating domains with ultra-low resolution hi-c data by graph structural entropy. *Nature Communications*, 9 (1):3265, 2018.

Li, S., Wang, X., Zhang, A., Wu, Y., He, X., and Chua, T.-S. Let invariant rationale discovery inspire graph contrastive learning. In *International Conference on Machine Learning*, pp. 13052–13065. PMLR, 2022a.

Li, S., Wu, J., Jiang, X., and Xu, K. Chart gcn: Learning chart information with a graph convolutional network for stock movement prediction. *Knowledge-Based Systems*, 248:108842, 2022b.

Martins, I. F., Teixeira, A. L., Pinheiro, L., and Falcao, A. O. A bayesian approach to in silico blood-brain barrier penetration modeling. *Journal of Chemical Information and Modeling*, 52(6):1686–1697, 2012.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020.

Mowshowitz, A. and Dehmer, M. Entropy and the complexity of graphs revisited. *Entropy*, 14(3):559–570, 2012.

Novick, P. A., Ortiz, O. F., Poelman, J., Abdulhay, A. Y., and Pande, V. S. Sweetlead: an in silico database of approved drugs, regulated chemicals, and herbal isolates for computer-aided drug discovery. *PloS One*, 8(11): e79568, 2013.

Ramsundar, B., Eastman, P., Walters, P., and Pande, V. *Deep learning for the life sciences: applying deep learning to genomics, microscopy, drug discovery, and more*. O'Reilly Media, 2019.

Raychaudhury, C., Ray, S., Ghosh, J., Roy, A., and Basak, S. Discrimination of isomeric structures using information theoretic topological indices. *Journal of Computational Chemistry*, 5(6):581–588, 1984.

Richard, A. M., Judson, R. S., Houck, K. A., Grulke, C. M., Volarath, P., Thillainadarajah, I., Yang, C., Rathman, J., Martin, M. T., Wambaugh, J. F., et al. Toxcast chemical landscape: paving the road to 21st century toxicology. *Chemical Research in Toxicology*, 29(8):1225–1251, 2016.

Shannon, C. The lattice theory of information. *Transactions of the IRE professional Group on Information Theory*, 1 (1):105–107, 1953.

Shannon, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

Sheridan, R. P. Time-split cross-validation as a method for estimating the goodness of prospective prediction. *Journal of Chemical Information and Modeling*, 2013.

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., and Borgwardt, K. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pp. 488–495. PMLR, 2009.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Sterling, T. and Irwin, J. J. Zinc 15–ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015.

Subramanian, G., Ramsundar, B., Pande, V., and Denny, R. A. Computational modeling of β-secretase 1 (bace-1) inhibitors using ligand based approaches. *Journal of Chemical Information and Modeling*, 56(10):1936–1949, 2016.

Sun, F.-Y., Hoffman, J., Verma, V., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *ICLR*, 2020.

Suresh, S., Li, P., Hao, C., and Neville, J. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Thomas, M. and Joy, A. T. *Elements of information theory*. Wiley-Interscience, 2006.

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

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

Wu, J., Chen, X., Xu, K., and Li, S. Structural entropy guided graph hierarchical pooling. In *International Conference on Machine Learning*, pp. 24017–24030. PMLR, 2022a.

Wu, J., Li, S., Li, J., Pan, Y., and Xu, K. A simple yet effective method for graph classification. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, July 23-29, 2022*. ijcai.org, 2022b.

Wu, T., Ren, H., Li, P., and Leskovec, J. Graph information bottleneck. *Advances in Neural Information Processing Systems*, 33:20437–20448, 2020.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *ICLR*, 2019.

Xu, M., Wang, H., Ni, B., Guo, H., and Tang, J. Self-supervised graph-level representation learning with local and global structure. In *ICML*, pp. 11548–11558. PMLR, 2021.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. *SIGKDD*, pp. 1365–1374, 2015.

Yang, L. and Hong, S. Omni-granular ego-semantic propagation for self-supervised graph representation learning. In *International Conference on Machine Learning*, pp. 25022–25037. PMLR, 2022.

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

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

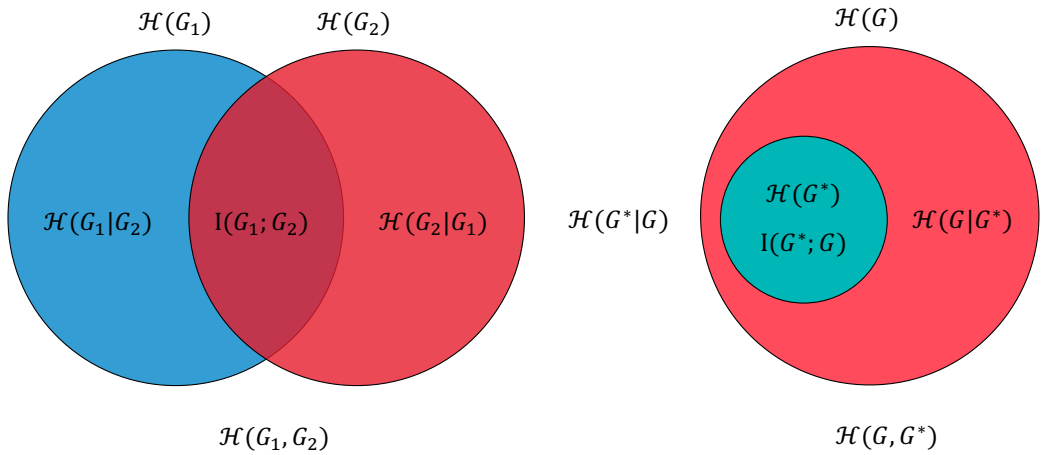
You, Y., Chen, T., Shen, Y., and Wang, Z. Graph contrastive learning automated. In *ICML*, pp. 12121–12132. PMLR, 2021.

You, Y., Chen, T., Wang, Z., and Shen, Y. Bringing your own view: Graph contrastive learning without prefabricated data augmentations. WSDM '22, pp. 1300–1309, New York, NY, USA, 2022. Association for Computing Machinery.

Zhang, C., Zhu, H., Peng, X., Wu, J., and Xu, K. Hierarchical information matters: Text classification via tree based graph neural network. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 950–959, 2022.

Zitnik, M., Sosič, R., Feldman, M. W., and Leskovec, J. Evolution of resilience in protein interactomes across the tree of life. *Proceedings of the National Academy of Sciences*, 116(10):4426–4433, 2019.

A. Proof of $\mathcal{H}(G^*|G) = 0$



(a) MI with data augmentations. (b) MI with the anchor view.

Figure A.1. Mutual information with data augmentations and the target anchor view.

In this section, we present the proof of the statement $\mathcal{H}(G^*|G) = 0$. First, we repeat the property that the target anchor view should own:

Definition A.1. The anchor view with essential information is supposed to be a substructure of the given graph to avoid artificially introduced noise.

Before the proof, Figure A.1 first shows the mutual information of graphs with data augmentations and the target anchor view (i.e., comply with Definition A.1). Figure A.1a suggests that the mutual information with data augmentations is the common part of two views. While Figure A.1b reveals that the mutual information with the target anchor view is the information within the essential part.

Theorem A.2. *Suppose the target anchor view G^* of the corresponding graph G owns the property in Definition A.1, the mutual information between G^* and G is*

$$\begin{aligned} I(G^*;G) &= \mathcal{H}(G^*) - \mathcal{H}(G^*|G) \\ &= \mathcal{H}(G^*). \end{aligned} \tag{21}$$

Proof. According to the definition of Shannon entropy, i.e., $\mathcal{H}(X) = -\sum_{x\in X} P(x)\log P(x)$, we follow the formulation of graph mutual information in Sun et al. (2020) that X is a set of node representations drawn from an empirical probability distribution of graph G, so the conditional entropy can be written as

$$\begin{aligned} \mathcal{H}(G^*|G) &= \mathcal{H}(X^*|X) \\ &= \sum_{x\in X} P(x) H(X^*|X=x) \\ &= -\sum_{x\in X} P(x) \sum_{x^*\in X^*} P(x^*|x)\log P(x^*|x) \\ &= -\sum_{x\in X}\sum_{x^*\in X^*} P(x^*,x)\log P(x^*|x) \\ &= -\sum_{x^*,x} P(x^*,x)\log P(x^*|x). \end{aligned} \tag{22}$$

Considering that G^* complies with the Definition A.1b, the illustration of probability distribution of G^* and G is shown Figure A.1b. Here, let us firstly discuss that

- when $x \in X$ and $x \notin X^*$, we have $P(x^*, x) = 0$.

Therefore, we can transform Equation 22 to

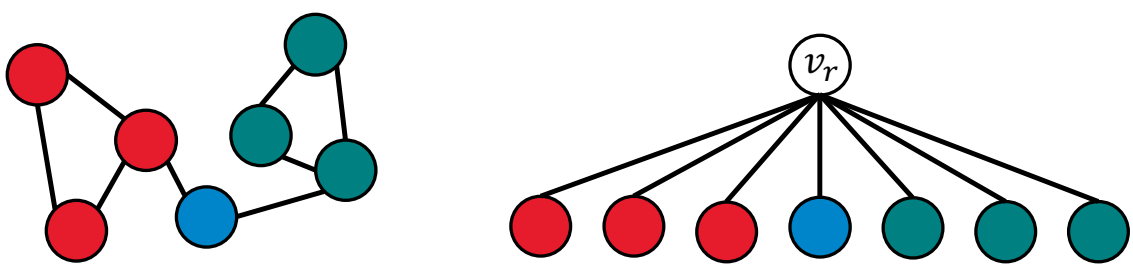
$$\begin{aligned}
\mathcal{H}(X^*|X) = & -\sum_{x^*,x} P(x^*,x)\log P(x^*|x) \\
= & -\sum_{x^*,x^*} P(x^*,x^*)\log P(x^*|x^*) \\
& -\sum_{x^*,x\notin X^*} P(x^*,x)\log P(x^*|x) \\
= & -\sum_{x^*\in X^*}\sum_{x^*\in X^*} P(x^*,x^*)\log P(x^*|x^*) \\
= & -\sum_{x^*\in X^*} P(x^*)\sum_{x^*\in X^*} P(x^*|x^*)\log P(x^*|x^*) \\
= & \sum_{x^*\in X^*} P(x^*)H(X^*|X^*=x^*) \\
= & \mathcal{H}(X^*|X^*) \\
= & 0.
\end{aligned} \tag{23}$$

Therefore, given $\forall x \in X$, we have $\mathcal{H}(G^*|G) = \mathcal{H}(X^*|X) = 0$. Accordingly, we have

$$I(G^*;G) = \mathcal{H}(G^*). \tag{24}$$

□

B. Illustrations for Algorithm 1



(a) Original Graph G. (b) Initial Coding Tree of G.

Figure A.2. The original graph G and the initialized coding tree with only root node and vertices from G as leaves.

Here, we present several figures to clearly reveal the running process of function COMBINE$(\cdot)$ and DROP$(\cdot)$ needed by Algorithm 1. Moreover, the growing process of a coding tree with a fixed height 2 from its original graph is further presented in Figure A.5. First, as shown in Figure A.2, we give a simple undirected graph $G = \{\mathcal{V}, \mathcal{E}\}$ for structural entropy minimization and the corresponding initialized coding tree T from G. Here, we present the definition of coding tree.

Coding Tree. A coding tree of a simple undirected graph $G = \{\mathcal{V}, \mathcal{E}\}$ is defined as a rooted tree T that has the following properties:

- The root node v_r is associated with the vertices set $\mathcal{V}$ of G. v_r is termed as the codeword of $\mathcal{V}$, that is $c(V) = v_r$. $\mathcal{V}$ is termed as the marker of v_r, that is $M(v_r) = \mathcal{V}$.
- Every node $v_\tau \in T$ is a codeword of a subset $\tilde{\mathcal{V}} \subset \mathcal{V}$; put differently, $c(\tilde{\mathcal{V}}) = v_\tau$ and $M(v_\tau) = \tilde{\mathcal{V}}$.

- For every node $v_\tau \in T$, suppose that $v_1^+, v_2^+, \cdots, v_N^+$ are all the immediate successors of v_τ in T; then all $M(v_i^+)$ are disjointed, and $M(v_\tau) = \bigcup_{i=1}^{N} M(v_i^+)$.
- For every leaf node $v_\tau^l \in T$, $M(v_\tau^l)$ is a singleton $\{v\}$ for some vertex $v \in \mathcal{V}$, and for every vertex $v \in \mathcal{V}$, there is a unique leaf node $v_\tau^l \in T$ such that $M(v_\tau^l) = v$ and $c(v) = v_\tau^l$.

B.1. Illustration of COMBINE

The process of COMBINE$(\cdot)$ is illustrated in Figure A.3. Specifically, let v_c^1 and v_c^1 be any two child nodes of root node v_r, then, a virtual node v_i is inserted between the root node and the two children, in which v_c^1 and v_c^1 become the children of v_i and v_i directly dissolves into the children cluster of v_r.

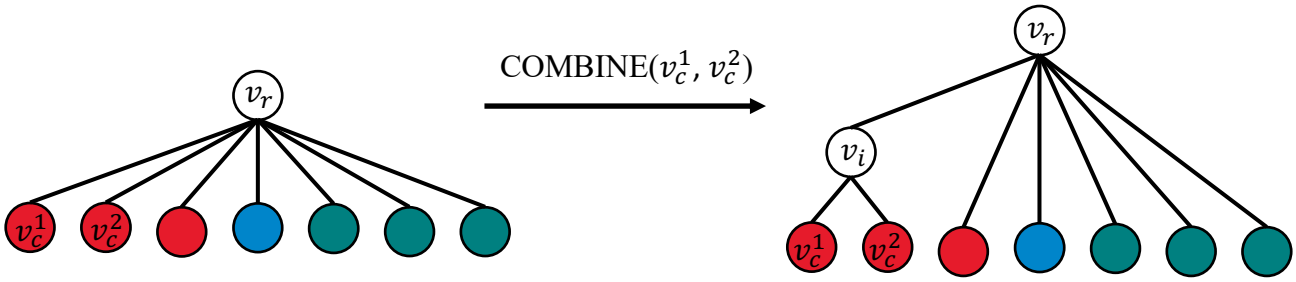


Figure A.3. An illustration of COMBINE$(\cdot)$.

B.2. Illustration of DROP

The process of DROP$(\cdot)$ is illustrated in Figure A.4. Specifically, given an inner node v_τ of the coding tree, then, v_τ is removed from the tree and its children are adopted by its parent node.

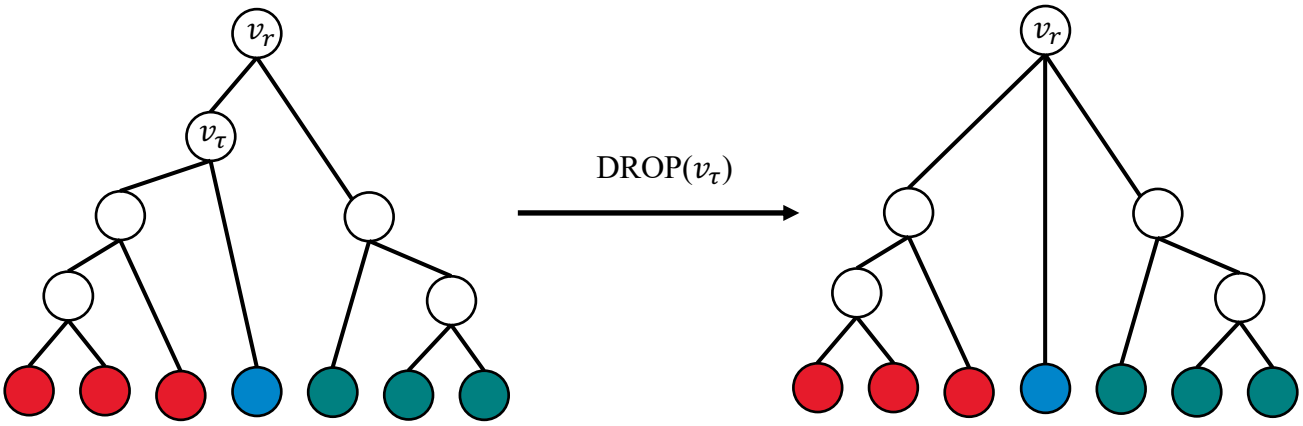
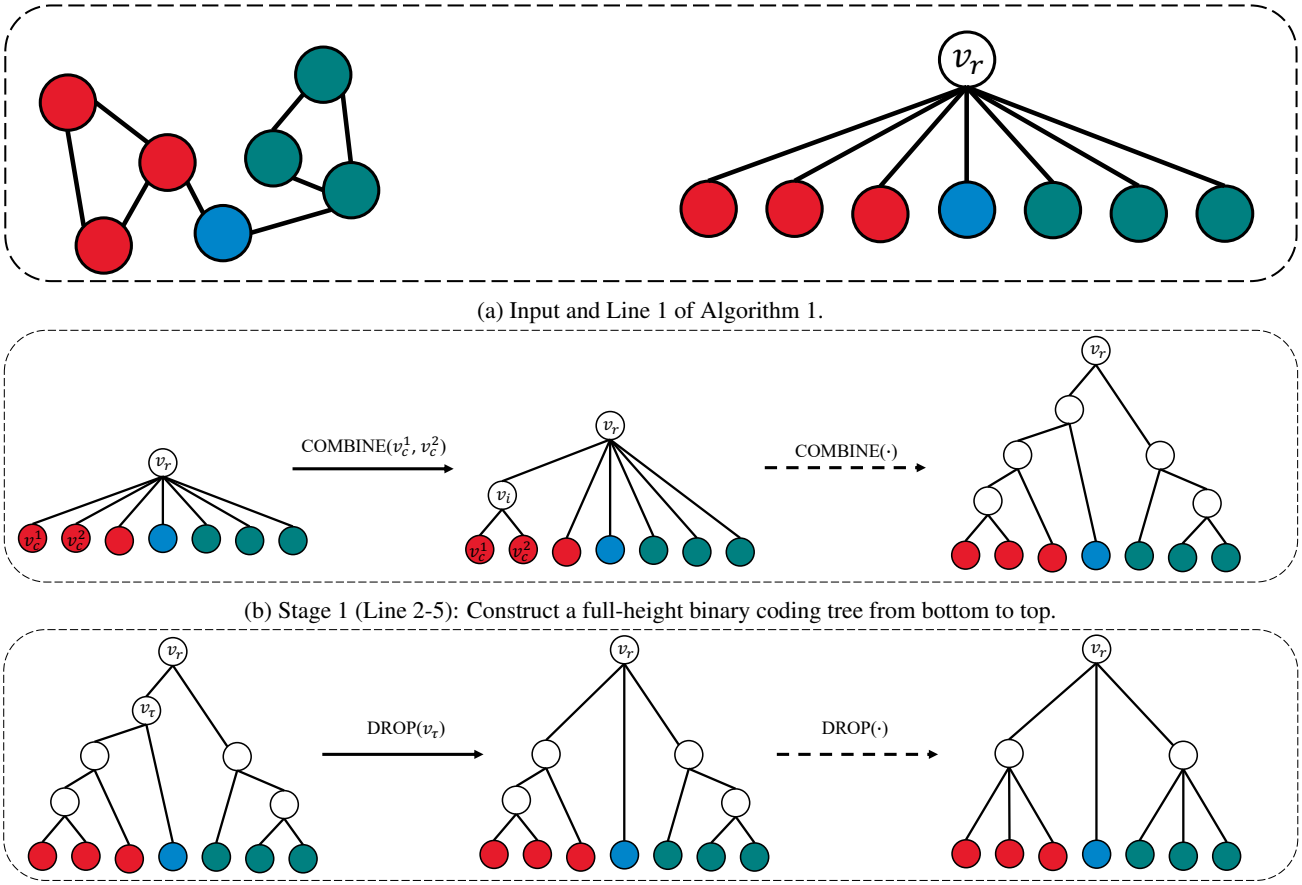


Figure A.4. An illustration of DROP$(\cdot)$.

B.3. Illustration of Algorithm 1

The running process of Algorithm 1 is illustrated in Figure A.5 and we set the target height of coding tree to 2. The input graph and the initialized coding tree are shown in Figure A.5a, in which the coding tree is initialized with a root node v_r and all vertices from input graph as leaves. Figure A.5b shows the process of Stage 1. Through iteratively combining two children of the root node, which can achieve the maximal structural entropy reduction after combination, a full-height binary coding tree is weaved from bottom to top. Figure A.5c reveals the process of State 2. Each time, an inner node that achieves the minimal structural entropy restoration is dropped. Finally, a coding tree with height of 2 can be harvested as a view for graph contrastive learning.



(a) Input and Line 1 of Algorithm 1.

(b) Stage 1 (Line 2-5): Construct a full-height binary coding tree from bottom to top.

(c) Stage 2 (Line 6-9): Squeeze full-height binary coding tree to fixed height 2.

Figure A.5. An illustration of the running process of Algorithm 1.

Complexity analysis. Given a graph $G = (V, E)$, $|V| = n$ and $|E| = m$, the runtime complexity of Algorithm 1 is $O(h_{max}(m \log n + n))$, in which h_{max} is the height of coding tree T after the first stage. In general, the coding tree T tends to be balanced in the process of structural entropy minimization, thus, h_{max} will be around $\log n$. Furthermore, a graph generally has more edges than nodes, i.e., $m \gg n$, thus the runtime of Algorithm 1 almost scales linearly in the number of edges.

C. Summary of Datasets

C.1. Datasets for Unsupervised and Semi-supervised Learning

A wide variety of datasets from different domains for a range of graph property prediction tasks are used for our experiments. Here, we present detailed descriptions of the 10 benchmarks utilized in this paper. Table A.1 shows statistics for datasets.

Social Network Datasets. IMDB-BINARY and IMDB-MULTI are derived from the collaboration of a movie set. In these two datasets, every graph consists of actors or actresses, and each edge between two nodes represents their cooperation in a certain movie. Each graph is derived from a prespecified movie, and its label corresponds to the genre of this movie. Similarly, COLLAB is also a collaboration dataset but from a scientific realm, which includes three public collaboration datasets (i.e., Astro Physics, High Energy Physics and Condensed Matter Physics). Many researchers from each field form various ego networks for the graphs in this benchmark. The label of each graph is the research field to which the nodes belong. REDDIT-BINARY and REDDIT-MULTI-5K are balanced datasets, where each graph corresponds to an online discussion thread and nodes correspond to users. An edge is drawn between two nodes if at least one of them responds to another's comment. The task is to classify each graph into the community or subreddit to which it belongs.

Small Molecules. NCI1 is a dataset made publicly available by the National Cancer Institute (NCI) and is a subset of balanced datasets containing chemical compounds screened for their ability to suppress or inhibit the growth of a panel of human tumor cell lines; this dataset possesses 37 discrete labels. MUTAG has seven kinds of graphs that are derived from 188 mutagenic aromatic and heteroaromatic nitro compounds. PTC includes 19 discrete labels and reports the carcinogenicity of 344 chemical compounds for male and female rats.

Bioinformatic Datasets. DD contains graphs of protein structures. A node represents an amino acid and edges are constructed if the distance of two nodes is less than $6\mathring{A}$. A label denotes whether a protein is an enzyme or non-enzyme. PROTEINS is a dataset where the nodes are secondary structure elements (SSEs), and there is an edge between two nodes if they are neighbors in the given amino acid sequence or in 3D space. The dataset has 3 discrete labels, representing helixes, sheets or turns.

Table A.1. Statistics for datasets of diverse nature from the benchmark TUDataset.

Dataset	#Graphs	#Classes	Avg. #Nodes	Avg. #Edges
Social Networks				
COLLAB	5,000	3	74.49	2457.78
REDDIT-BINARY	2,000	2	429.63	497.75
REDDIT-MULTI-5K	4,999	5	508.52	594.87
IMDB-BINARY	1,000	2	19.77	96.53
IMDB-MULTI	1,500	3	13.00	65.94
GITHUB	12,725	2	113.79	234.64
Small Molecules				
NCI1	4,110	2	29.87	32.30
MUTAG	188	2	17.93	19.79
Bioinformatics				
PROTEINS	1,113	2	39.06	72.82
DD	1,178	2	284.32	715.66

C.2. Details of Molecular Datasets

Input graph representation. For simplicity, we use a minimal set of node and bond features that unambiguously describe the two-dimensional structure of molecules. We use RDKit (Landrum, 2013) to obtain these features.

- Node features:
  - Atom number: [1, 118]
  - Chirality tag: {unspecified, tetrahedral cw, tetrahedral ccw, other}
- Edge features:
  - Bond type: {single, double, triple, aromatic}
  - Bond direction: {–, endupright, enddownright}

Downstream task datasets. 8 binary graph classification datasets from MoleculeNet (Wu et al., 2018) are used to evaluate model performance.

- BBBP (Martins et al., 2012). Blood-brain barrier penetration (membrane permeability), involves records of whether a compound carries the permeability property of penetrating the blood-brain barrier.
- Tox21 (Tox, 2014). Toxicity data on 12 biological targets, which has been used in the 2014 Tox21 Data Challenge and includes nuclear receptors and stress response pathways.
- ToxCast (Richard et al., 2016). Toxicology measurements based on over 600 in vitro high-throughput screenings.

Table A.2. Datasets statistics summary.

Dataset	Category	Utilization	#Tasks	#Graphs	Avg.Node	Avg.Degree
ZINC15	Biochemical Molecules	Pre-Training		2,000,000	26.63	57.72
PPI-306K	Protein-Protein Intersection Networks	Pre-Training		306,925	39.82	729.62
BBBP	Biochemical Molecules	Finetuning	1	2,039	24.06	51.90
Tox21	Biochemical Molecules	Finetuning	12	7,831	18.57	38.58
ToxCast	Biochemical Molecules	Finetuning	617	8,576	18.78	38.52
SIDER	Biochemical Molecules	Finetuning	27	1,427	33.64	70.71
ClinTox	Biochemical Molecules	Finetuning	2	1,477	26.15	55.76
MUV	Biochemical Molecules	Finetuning	17	93,087	24.23	52.55
HIV	Biochemical Molecules	Finetuning	1	41,127	25.51	54.93
BACE	Biochemical Molecules	Finetuning	1	1,513	34.08	73.71
PPI	Protein-Protein Intersection Networks	Finetuning	40	88,000	49.35	890.77

- SIDER (Kuhn et al., 2016). Database of marketed drugs and adverse drug reactions (ADR), grouped into 27 system organ classes and also known as the Side Effect Resource.
- ClinTox (Novick et al., 2013; Gayvert et al., 2016). Qualitative data classifying drugs approved by the FDA and those that have failed clinical trials for toxicity reasons.
- MUV (Gardiner et al., 2011). Subset of PubChem BioAssay by applying a refined nearest neighbor analysis, designed for validation of virtual screening techniques.
- HIV (HIV). Experimentally measured abilities to inhibit HIV replication.
- BACE (Subramanian et al., 2016). Qualitative binding results for a set of inhibitors of human β-secretase 1.

Details of Dataset Splitting For molecular prediction tasks, following Ramsundar et al. (2019), we cluster molecules by scaffold (molecular graph substructure) (Bemis & Murcko, 1996), and recombine the clusters by placing the most common scaffolds in the training set, producing validation and test sets that contain structurally different molecules. Prior work has shown that this scaffold split provides a more realistic estimate of model performance in prospective evaluation compared to random split (Chen et al., 2012; Sheridan, 2013). The split for train/validation/test sets is 80%:10%:10%.

C.3. Details of Protein Datasets

Input graph representation. The protein subgraphs only have edge features.

- Edge features (These edge features indicate whether a particular type of relationship exists between a pair of proteins):
  - Neighborhood: {True, False}, if a pair of genes are consistently observed in each other's genome neighborhood
  - Fusion: {True, False}, if a pair of proteins have their respective orthologs fused into a single protein-coding gene in another organism
  - Co-occurrence: {True, False}, if a pair of proteins tend to be observed either as present or absent in the same subset of organisms
  - Co-expression: {True, False}, if a pair of proteins share similar expression patterns
  - Experiment: {True, False}, if a pair of proteins are experimentally observed to physically interact with each other
  - Database: {True, False}, if a pair of proteins belong to the same pathway, based on assessments by a human curator
  - Text: {True, False}, if a pair of proteins are mentioned together in PubMed abstracts

Datasets. A dataset containing protein subgraphs from 50 species is used (Zitnik et al., 2019). The original PPI networks do not have node attributes, but contain edge attributes that correspond to the degree of confidence for 7 different types of protein-protein relationships. The edge weights range from 0, which indicates no evidence for the specific relationship,

to 1000, which indicates the highest confidence. The weighted edges of the PPI networks are thresholded such that the distribution of edge types across the 50 PPI networks are uniform. Then, for every node in the PPI networks, subgraphs centered on each node were generated by: (1) performing a breadth first search to select the subgraph nodes, with a search depth limit of 2 and a maximum number of 10 neighbors randomly expanded per node, (2) including the selected subgraph nodes and all the edges between those nodes to form the resulting subgraph.

The entire dataset contains 394,925 protein subgraphs derived from 50 species. Out of these 50 species, 8 species (arabidopsis, celegans, ecoli, fly, human, mouse, yeast, zebrafish) have proteins with GO protein annotations. The dataset contains 88,000 protein subgraphs from these 8 species, of which 57,448 proteins have at least one positive coarse-grained GO protein annotation and 22,876 proteins have at least one positive fine-grained GO protein annotation. For the self-supervised pre-training dataset, we use a subset 306,925 protein subgraphs.

Fine-grained protein functions is defined as Gene Ontology (GO) annotations that are leaves in the GO hierarchy, and coarse-grained protein functions is defined as GO annotations that are the immediate parents of leaves (Ashburner et al., 2000; Consortium, 2019). For example, a fine-grained protein function is "Factor XII activation", while a coarse-grained function is "positive regulation of protein". The former is a specific type of the latter, and is much harder to derive experimentally. The GO hierarchy information is obtained using GOATOOLS (Klopfenstein et al., 2018). The supervised pre-training dataset and the downstream evaluation dataset are derived from the 8 labeled species. The 40-th most common fine-grained protein label only has 121 positively annotated proteins, while the 40-th most common coarse-grained protein label has 9386 positively annotated proteins. This illustrates the extreme label scarcity of the downstream tasks.

Dataset splitting. In the PPI network, species split simulates a scenario where we have only high-level coarse-grained knowledge on a subset of proteins (prior set) in a species of interest (human in our experiments), and want to predict fine-grained biological functions for the rest of the proteins in that species (test set). For species split, we use 50% of the protein subgraphs from human as test set, and 50% as a prior set containing only coarse-grained protein annotations. The protein subgraphs from 7 other labelled species (arabidopsis, celegans, ecoli, fly, mouse, yeast, zebrafish) are used as train and validation sets, which are split $85\% : 15\%$. The effective split ratio for the train/validation/prior/test sets is $69\% : 12\% : 9.5\% : 9.5\%$.

D. Detailed Experiment Setup

D.1. Settings for Unsupervised Learning

Following the learning setting in SOTA works, the corresponding learning protocols are adopted for a fair comparison. In unsupervised representation learning (Sun et al., 2020), all data is used for model pre-training and the learned graph embeddings are then fed into a non-linear SVM classifier to perform 10-fold cross-validation. Experiments are performed for 5 times each of which corresponds to a 10-fold evaluation as (Sun et al., 2020), with mean and standard deviation of accuracies (%) reported. As for graph representation learning, models are trained 20 epochs and tested every 10 epochs. Hidden dimension is chosen from $\{32, 64\}$, and batch size is chosen from $\{32, 128\}$. An Adam optimizer (Kingma & Ba, 2015) is employed to minimize the contrastive lose with $\{0.01, 0.005, 0.001\}$ learning rate.

Data Augmentations on Graphs. Follow the data augmentations in GraphCL (You et al., 2020), there are four types of general data augmentations for graph-structured data:

- **Node dropping.** Given the graph G, node dropping will randomly discard certain portion of vertices along with their connections. The underlying prior enforced by it is that missing part of vertices does not affect the semantic meaning of G. Each node's dropping probability follows a default i.i.d. uniform distribution (or any other distribution).
- **Edge perturbation.** It will perturb the connectivities in G through randomly dropping certain ratio of edges. It implies that the semantic meaning of G has certain robustness to the edge connectivity pattern variances. We also follow an i.i.d. uniform distribution to drop each edge.
- **Attribute masking.** Attribute masking prompts models to recover masked vertex attributes using their context information, i.e., the remaining attributes. The underlying assumption is that missing partial vertex attributes does not affect the model predictions much.
- **Subgraph.** This one samples a subgraph from G using random walk. It assumes that the semantics of G can be much

preserved in its (partial) local structure.

D.2. Setting for Semi-supervised Learning

Configuration. ResGCN with 128 hidden units and 5 layers is set up in semi-supervised learning. In addition, the same data augmentations on graphs with the default augmentation strength 0.2 are adopted. For all datasets we perform experiments with 10% label rate for 5 times, each of which corresponds to a 10-fold evaluation as (You et al., 2020), with mean and standard deviation of accuracies (%) reported. For pre-training, learning rate is tuned in $\{0.01, 0.001, 0.0001\}$ and epoch number in $\{20, 40, 60, 80, 100\}$ where grid search is performed. For fine-tuning, we following the default setting in (You et al., 2020), that is, learning rate is 0.001, hidden dimension is 128, bath size is 128, and the pre-trained models are trained 100 epochs.

Learning protocols. Following the learning setting in SOTA works, the corresponding learning protocols are adopted for a fair comparison. In semi-supervised learning (You et al., 2020), there exist two learning settings. For datasets with a public training/validation/test split, pre-training is performed only on training dataset, finetuning is conducted with 10% of the training data, and final evaluation results are from the validation/test sets. For datasets without such splits, all samples are employed for pre-training while finetuning and evaluation are performed over 10 folds.

D.3. Setting for Transfer Learning

Pre-training dataset. ZINC15 (Sterling & Irwin, 2015) dataset is adopted for biochemical pre-training. In particular, a subset with two million unlabeled molecular graphs are sampled from the ZINC15. For protein domain, following Hu et al. (2020), 306K unlabeled protein ego-networks are utilized for pre-training.

Pre-training details. In the graph encoder setting in Hu et al. (2020), GIN (Xu et al., 2019) with five convolutional layers is adopted for message passing. In particular, the hidden dimension is fixed to 300 across all layers and a pooling readout function that averages graph nodes is hired for NT-Xent loss calculation with the scale parameter $\tau = 0.1$. The hidden representations at the last layer are injected into the average pooling function. An Adam optimizer (Kingma & Ba, 2015) is employed to minimize the integrated losses produced by the 5-layer GIN encoder. The batch size is set as 256, and all training processes will run 100 epochs.

Fine-tuning dataset. We employ the eight ubiquitous benchmarks from the MoleculeNet dataset (Wu et al., 2018) as the biochemical downstream experiments. These benchmarks include a variety of molecular tasks like physical chemistry, quantum mechanics, physiology, and biophysics. The protein downstream task is to predict 40 fine-grained biological functions of 8 species. For dataset split, the scaffold split scheme (Chen et al., 2012) is adopted for train/validation/test set generation.

Fine-tuning details. For downstream tasks, a linear layer is stacked after the pre-trained graph encoders for final property prediction. The downstream model still employs the Adam optimizer for 100 epochs fine-tuning. All experiments on each dataset are performed for ten runs with different seeds, and the results are the averaged ROC-AUC scores (%) $\pm$ standard deviations. The learning rate is selected from $\{0.01, 0.001, 0.0001\}$ and is symmetric for both the encoder and augmenter during self-supervision on the pre-train dataset. To be in line with (You et al., 2020), the number of training epochs for pre-training is chosen among $\{20, 40, 60, 80, 100\}$ based on the validation performance on the fine-tune datasets.

D.4. Settings for Orthogonal Experiment

AD-GCL. In cooperation with AD-GCL (Suresh et al., 2021), we faithfully follow the original setting while switching the anchor view from the original graph to the proposed anchor view. Note that, in the evaluation stage, the linear SVM is adopted to keep in line with the results in the main text of AD-GCL, which is different from the setting of GraphCL. In particular, the key hyperparameter λ that prevents AD-GCL from very aggressive perturbation is fixed to 5, that is the AD-GCL-FIX in the original work.

JOAO(v2). In cooperation with JOAO(v2) (You et al., 2021), the same experimental setting is adopted from the published paper while recalling one of the two views and assigning the proposed anchor view to that place. Naturally, JOAO(v2) only needs to search the other view from data augmentations. Similarly, GIN is adopted for graph encoding while non-linear SVM

is employed for evaluation. The hyperparameter γ controlling the trade-off between the contrastive loss and view distance is tuned in the range of $\{0.01, 0.1, 1\}$. In particular, JOAO is pre-trained with 20 epochs, while JOAOv2 is pre-trained with double epochs since multiple projection heads are applied.

AutoGCL. We adopt the naive training strategy proposed in AutoGCL to make a fair comparison. Specifically, we retain one of the two graph generators and assign our proposed anchor view to the blank position. In particular, AutoGCL extends the layer number of graph encoder from 3 to 5 and the hidden size from 32 to 128. Moreover, AutoGCL is pre-trained with 30 epochs rather than 20 epochs.

RGCL. In cooperation with RGCL (Li et al., 2022a), we faithfully follow the experiment settings revealed in their codes while replacing one of the two rationale-augmented views with SEGA. Note that, the tuned hyper-parameters in RGCL includes learning rate, sampling ratio ρ, loss temperature τ, and loss balance λ. In particular, RGCL is pre-trained 40 epochs in total and evaluated every 5 epochs.