Pre-training Graph Neural Networks on Molecules by Using Subgraph-Conditioned Graph Information Bottleneck

Van Thuy Hoang, O-Joun Lee*

Department of Artificial Intelligence, The Catholic University of Korea
{hoangvanthuy90, ojlee}@catholic.ac.kr

Abstract

This study aims to build a pre-trained Graph Neural Network (GNN) model on molecules without human annotations or prior knowledge. Although various attempts have been proposed to overcome limitations in acquiring labeled molecules, the previous pre-training methods still rely on semantic subgraphs, i.e., functional groups. Only focusing on the functional groups could overlook the graph-level distinctions. The key challenge to build a pre-trained GNN on molecules is how to (1) generate well-distinguished graph-level representations and (2) automatically discover the functional groups without prior knowledge. To solve it, we propose a novel Subgraph-conditioned Graph Information Bottleneck, named S-CGIB, for pre-training GNNs to recognize core subgraphs (graph cores) and significant subgraphs. The main idea is that the graph cores contain compressed and sufficient information that could generate well-distinguished graph-level representations and reconstruct the input graph conditioned on significant subgraphs across molecules under the S-CGIB principle. To discover significant subgraphs without prior knowledge about functional groups, we propose generating a set of functional group candidates, i.e., ego networks, and using an attention-based interaction between the graph core and the candidates. Despite being identified from self-supervised learning, our learned subgraphs match the real-world functional groups. Extensive experiments on molecule datasets across various domains demonstrate the superiority of S-CGIB.

Introduction

Graph Neural Networks (GNNs) have recently emerged in computational chemistry, offering powerful tools for predicting molecular properties (Gilmer et al. 2017; Kearnes et al. 2016). While GNNs have shown remarkable performance in molecular property prediction, their effectiveness depends on the availability of abundant labeled molecules for model training (Hao et al. 2020; Hoang et al. 2023).

Recently, pre-training strategies have offered considerable potential in overcoming the challenges of the scarcity of labeled molecular data (Rong et al. 2020; Luong and Singh 2023). The existing pre-training strategies can be categorized into three primary groups: node-level pre-training, contrastive learning, and subgraph-level pre-training. Node-level pre-training mainly focuses on node-level prediction, e.g., node attribute reconstruction or edge prediction, which may not fully leverage the high-order structure of molecules, i.e., functional groups (Rong et al. 2020). The second strategy is contrastive learning, which focuses on learning representations by contrasting multiple views of molecules based on random or heuristic augmentations (You et al. 2021; Xu et al. 2021). More recently, subgraph-level strategies focus on semantic subgraphs, which can capture both local and global structural patterns by identifying functional groups (Subramonian 2021; Zhang et al. 2021; Rong et al. 2020; Liu et al. 2023; Inae, Liu, and Jiang 2023). The main idea is to use human annotations or prior knowledge to extract the semantic subgraphs, e.g., frequent subgraphs across molecules, to enhance recognizing significant substructures and molecular property prediction (Luong and Singh 2023; Degen et al. 2008). To sum up, most recent pre-training strategies aggregate information from node-level or subgraph-level to generate graph-level representations.

However, two challenges limit the existing pre-training strategies on molecules. ***First***, the existing strategies lack the ability to generate well-distinguished graph-level representations. Most node-level strategies mainly focus on the local structure and then adopt a pooling function, e.g., mean, max, or sum, to aggregate information, resulting in poor-distinguished graph-level representations as information from noisy and redundant nodes can be aggregated to form graph-level representations (Hu et al. 2020a). Besides, while subgraph-level strategies (Subramonian 2021; Zhang et al. 2021) can capture specific subgraphs at multiple scales, they could overlook the entire graph-level distinctions (Rong et al. 2020; Inae, Liu, and Jiang 2023). That is, subgraph-level representations based on discrete pre-defined patterns, e.g., frequent subgraphs, ignore global interactions between important nodes that derive the molecule's entire structure. For contrastive learning strategies, applying augmentation schemes, e.g., edge perturbation, could potentially disrupt the structures and properties of molecules (Lee, Lee, and Park 2022). ***Second***, it is challenging for subgraph-based strategies to cover all possible functional groups given in diverse molecule datasets. To capture functional groups, recent subgraph-level strategies create dictionaries based on pre-defined rules, e.g., counting the discrete occurrences of

*Corresponding author: O-Joun Lee (Tel.: +82-2-2164-5516)

subgraphs across molecules to decompose molecules into a bag of subgraphs, commonly prioritizing subgraphs with larger sizes and frequent occurrences (Luong and Singh 2023; Kong et al. 2022). The prioritization and static dictionaries could limit the model's ability to capture new and uncommon functional groups.

In this paper, we overcome the above challenge of pre-training strategies by considering how to (1) generate well-separated graph-level representations and (2) automatically capture significant subgraphs without explicit annotations or prior knowledge. The fundamental idea behind our strategy is that, across the chemical domain, molecules share universal core subgraphs that can combine with specific significant subgraphs to robust representations of molecules.

To generate well-distinguished graph-level representations, this initiates a problem in recognizing a set of important nodes (graph cores) that can allow robust and well-separated representations (Hu et al. 2020a). We interpret this problem as the Graph Information Bottleneck (GIB) principle, which aims to compress an input graph into a core subgraph that keeps sufficient and compressed information about the input graph (Yu et al. 2021). However, current GIB methods learn compressed subgraphs by minimizing information loss in predicting the graph labels, which still require the label information. To solve it, we propose a Subgraph-conditioned Graph Information Bottleneck (S-CGIB) for self-supervised pre-training to compress an input graph into a graph core conditioned on specific significant subgraphs without using label information. First, the graph cores contain important nodes, which could generate robust representations under the S-CGIB principle. Second, toward capturing functional groups without prior knowledge, we suppose that graph cores then could reconstruct input graphs conditioned on specific significant subgraphs across molecules.

To discover the significant subgraphs, as we intentionally ignore using prior knowledge about functional groups, we propose to generate a set of functional group candidates, i.e., ego networks rooted at each node. The reason is that the graph core of molecules typically consists of central substructures composing important nodes only for generating well-separated representations. In other words, the compression process ignored unimportant nodes, which can be a part of functional groups in terms of molecular properties. Then, we propose an attention-based interaction between the graph core and candidates to recognize significant subgraphs as functional groups. The attention coefficients can highlight significant subgraphs, which benefit from recognizing functional groups across molecules.

Related work

We now discuss how the existing pre-training strategies can learn the molecular structure and chemical properties in the context of Self-Supervised Learning compared to our method. Early pre-training strategies focus on learning node representations, which are then aggregated into a single graph-level representation through pooling mechanisms, e.g., min, max, or sum, (Hu et al. 2020b,a). The node-level strategies can be mainly grouped into node-level structure reconstruction, e.g., neighborhood prediction (Hu et al. 2020b; Hoang and Lee 2024), or node feature reconstruction (Devlin et al. 2019). However, these methods focus on learning to distinguish individual node representations, which do not directly handle the challenge of incorporating node embeddings into a single graph-level representation. Furthermore, node-level pre-training strategies could be limited to capture the high-order structures, i.e., functional groups.

Another line is contrastive learning, which is generally grouped into two categories: node-level contrastive methods and graph-level contrastive methods (Liu et al. 2022; Stärk et al. 2022; Qiu et al. 2020). The node-level contrastive learning strategies seek to generate multiple node views by applying augmentation schemes, e.g., edge perturbation, which modifies the graph connectivity while preserving the node's identity. For example, GraphCL (You et al. 2020) adopts multiple augmentation schemes to generate multiple views of the input graph and uses contrastive loss to obtain embeddings of these views closer. The idea of JOAO (You et al. 2021) is to automatically search schemes to find the most effective augmentations. In contrast, graph-level methods generate multi-views of an input graph and guarantee similar representations while discriminating them from the other graph-level representations (Xu et al. 2021). Such augmentations often change the molecular connectivity and structure, failing to preserve the molecule's properties.

Recent pre-training approaches on molecules emphasize recognizing and learning semantic patterns, such as functional groups, across molecule data (Zhang et al. 2021; Liu et al. 2023; Inae, Liu, and Jiang 2023). The semantic subgraphs can be discovered under pre-defined rules with the use of prior knowledge or human annotation. For example, GROVER (Rong et al. 2020) extracts 85 frequent functional groups and employs a Self-supervised Learning (SSL) task to predict the presence of the functional groups. MGSSL (Zhang et al. 2021) employs depth-first or breadth-first search to discover the semantic subgraphs at multiple scales. Recently, GraphFP (Luong and Singh 2023) decomposes molecules into bags of functional groups by composing a dictionary via frequent subgraph mining, i.e., common and large frequent subgraphs across molecules. Such strategies rely on pre-defined vocabulary with discrete counting frequent subgraphs in molecules, making it challenging to build a complete dictionary to cover new or less common functional groups. In contrast, we automatically discover functional groups via attention-based interaction between the molecular core and a set of functional group candidates.

Problem Descriptions

We study the problem of pre-trained GNNs on molecules, which recognize graph cores conditioned on significant subgraphs (functional group candidates) under the S-CGIB principle. Thus, we first present notations and then the definition of S-CGIB, which is a modification of GIB and CGIB.

A molecule can be represented as a graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_N\}$ represents the set of atoms and E denotes the set of bonds. G is associated with its adjacency matrix A and feature matrix X. Let $N_k(v)$ be a set of neighboring nodes within a k-hop distance from the root

node v. The set of functional group candidates in G is defined as: $S = \{G[N_k(v)] \mid v \in V\}$, where $G[N_k(v)]$ is the k-hop ego network rooted at node v.

Recently, the Information Bottleneck (IB) principle (Tishby, Pereira, and Bialek 2000) has been used on graphs, called GIB, to discover a core subgraph from an input graph.

Definition 1 (GIB) *The GIB principle was originally introduced by Yu et al. (2021) to recognize a compressed and informative subgraph from an input graph. Given an input graph G and its label Y, the compressed graph, as graph core G_c, is discovered as:*

$$\min_{G_c} -I(Y; G_c) + \beta I(G; G_c), \tag{1}$$

where β is a Lagrange multiplier used to balance the two terms. The first term encourages G_c to be informative to the graph label Y, and the second term is the compression term, which minimizes the mutual information of G and G_c.

In the context of the presence of side information T, several studies have accounted for the conditional information T, named CIB (Chechik and Tishby 2002; Gondek and Hofmann 2003). We then apply CIB on graphs as Conditional Graph Information Bottleneck (CGIB).

Definition 2 (CGIB) *Given an input graph G and its label Y, the graph core G_c is discovered given the side information T, as:*

$$\min_{G_c} -I(Y; G_c|T) + \beta I(G; G_c). \tag{2}$$

The first term quantifies how much information the graph core G_c retains about Y, given the side information T. The variable T is supposed to be known as prior knowledge.

To employ the CGIB principle under an SSL task without prior knowledge, we propose to minimize the first term of Eq. 2 by replacing the prediction with a reconstruction task. As the compression process ignored unimportant nodes, which can be a part of functional groups, we suppose they can be side information for a graph reconstruction task. The key idea is that molecules share universal graph cores, which can reconstruct the molecule structure conditioned on specific significant subgraphs. Specifically, we suppose that G is formed by combining a graph core G_c and a set of functional group candidates S, such that $G = G_c \cup S$. Then, the Subgraph-conditioned Graph Information Bottleneck can be defined below:

Definition 3 (S-CGIB) *Given an input graph G and a set of functional group candidates S, we define the S-CGIB principle conditioned on the subgraph S as:*

$$\min_{G_c} - \underbrace{I\left(G; G_c|S\right)}_{\textit{Conditional Reconstruction}} + \underbrace{\beta I\left(G; G_c\right)}_{\textit{Compression}}, \tag{3}$$

By conditioning on S, the first term encourages the graph core G_c to capture sufficient information for reconstructing input graph G (*conditional reconstruction*), while the graph core also needs to be compressed from the input graph (*compression*). Overall, jointly optimizing the two terms allows G_c to be compressed and preserve the input graph structure conditioned on S. Since S consists of all ego networks rooted at each node, to discover important ego networks, i.e., functional groups, we propose an attention-based strategy detailed in the following Section.

Methodology

Model Architecture

Graph Compression The overall architecture of S-CGIB is shown in Figure 1. Given an input graph G with adjacency matrix A and node feature matrix X, we learn node representations in G through a GNN encoder as:

$$H = f_\phi\left(X, A\right), \tag{4}$$

where $H \in R^{N\times d}$ refers to node embeddings, $f_\phi(\cdot,\cdot)$ is a GNN encoder, e.g., GIN (Xu et al. 2019). Inspired by recent VGIB principle (Yu, Cao, and He 2022) that injects noise information into an input graph to obtain important nodes, we compress G by injecting noise into H to obtain a graph core G_c with new node representations Z, which is a bottleneck to distill the important nodes, thereby generating well-distinguished graph-level representations. The key idea is that the important nodes will be injected with less noise information compared to unimportant nodes. Specifically, given the embedding matrix H, for each node v_i, S-CGIB learns a probability p_i with a multi-layer perceptron (MLP) followed by a $\mathrm{Sigmoid}(\cdot)$ function. We then replace the node representation h_i by ϵ with probability p_i, as:

$$p_i = \mathrm{Sigmoid}\left(\mathrm{MLP}(H_i)\right), \tag{5}$$

$$\varepsilon \sim N\left(\mu_H, \sigma_H^2\right),\ \lambda_i \sim \mathrm{Bernoulli}\left(p_i\right), \tag{6}$$

$$z_i = \lambda_i h_i + (1-\lambda_i)\varepsilon, \tag{7}$$

where λ is obtained by sampling from Bernoulli distribution parameterized with the probability p_i, μ_H and σ_H are the mean and variance of H, ϵ is sampled from H based on Gaussian distribution. Therefore, the information of the input graph is compressed into Z with the probability of p_i by masking unimportant nodes with noisy information. That is, the compression process ensures that Z focuses on the important nodes (graph core) while discarding irrelevant and unimportant nodes. To allow the differentiable sampling, we adopt the Gumbel sigmoid method (Jang, Gu, and Poole 2017; Maddison, Mnih, and Teh 2017), i.e., $\lambda_i = \mathrm{Sigmoid}(1/\tau \log[p_i/(1-p_i)]) + \log\left[q/(1-q)\right]$ where $q \sim \mathrm{Uniform}(0,1)$ and τ is the temperature parameter. For the detailed compression optimization process, we refer readers to the Model Optimization Section.

Subgraph Learning The next problem is to extract functional group candidates and encode them to obtain vector representations. Let $G[N_k(v)]$ denotes the k-hop ego network rooted at the node v. We first apply a GNN encoder on nodes within the $G[N_k(v)]$. Then, the subgraph-level representation rooted at node v is obtained via a pooling function $\mathrm{POOL}(\cdot)$. Formally, the representations of ego networks in the input graph G can be computed as:

$$h_v^{(l+1)} = g_\theta^{(l)}\left(G\left[N_k\left(v\right)\right]\right), l = 0, \cdots, L-1, \tag{8}$$

$$h_v = \mathrm{POOL}\left(h_u^{(L)} | u \in N_k\left(v\right)\right), \tag{9}$$

$$H_S = [h_v | v \in V], \tag{10}$$

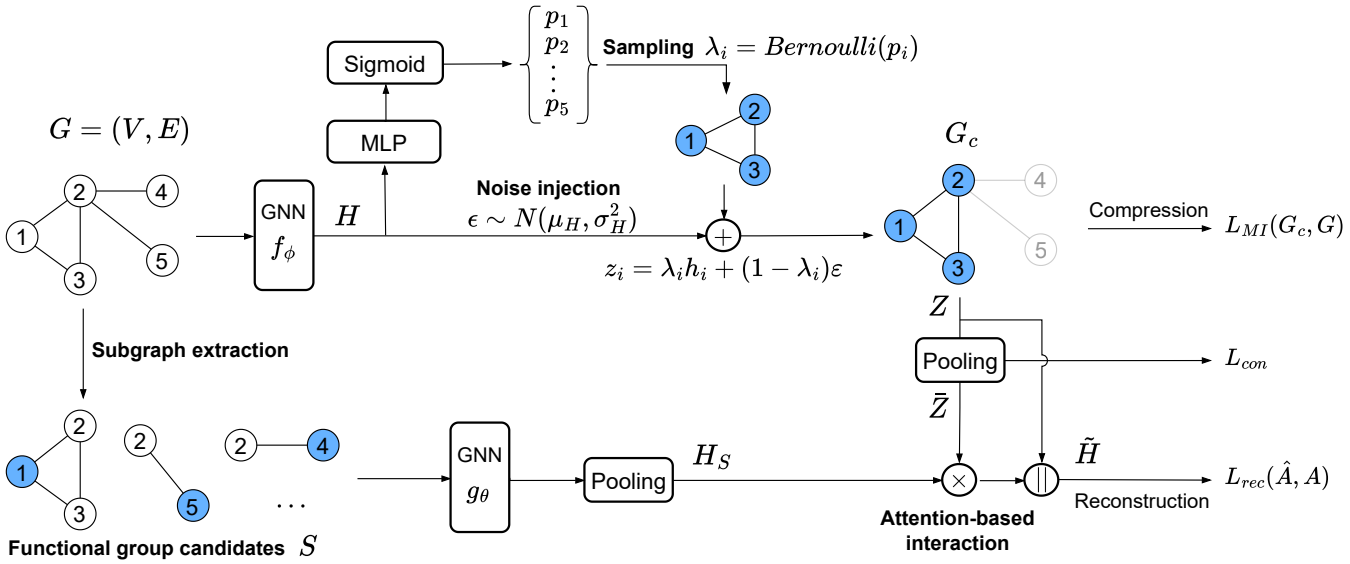

Figure 1: The overall architecture of S-CGIB.

where $g_\theta^{(l)}(\cdot)$ denotes a GNN encoder, e.g., GIN, for the subgraph $G[N_k(v)]$, POOL$(\cdot)$ refers to a subgraph pooling, e.g., sum. Therefore, the local surrounding structures rooted at each node are captured via GNNs.

Graph Core and Subgraph Interaction Note that each functional group has distinct chemical characteristics that contribute differently to the overall molecule behavior. To capture significant subgraphs across molecules, we then propose an attention-based interaction between the graph core and subgraph candidates to highlight specific significant subgraphs. Specifically, we first employ a pooling function to aggregate important node features Z into $\bar{Z}$, i.e., $\bar{Z} = \text{POOL}(Z)$, to obtain a single vector representation of the graph core. We then calculate the normalized attention coefficients for each functional group candidate. Formally, the coefficient of a subgraph candidate rooted at a node v_i can be computed as:

$$\alpha_i = \frac{\exp\left(\left(\bar{Z}||H_{S_i}\right) W_a^T\right)}{\sum_{k=1}^{N} \exp\left(\left(\bar{Z}||H_{S_k}\right) W_a^T\right)}, \tag{11}$$

where $W_a \in R^{1\times 2d}$ refers to learnable projection and H_{S_i} is the representations of the ego network rooted at node v_i. Thus, the interaction could capture the correlation between the graph core and functional group candidates. The final representations are then concatenated along with the coefficients as:

$$\tilde{H}_i = Z_i || \left(\alpha_i H_{S_i}\right). \tag{12}$$

In a nutshell, given an input graph G, our model jointly learns the graph core and the significant subgraphs under the S-CGIB principle. We first inject noisy information into H to obtain the graph core Z under the compression process. Meanwhile, we capture the significant specific subgraphs from the subgraph candidates (ego networks rooted at each node) through the attention-based interaction between the graph core and the ego networks under the graph reconstruction.

Model Optimization

To train the model while optimizing the graph core conditioned on S, we optimize the objective function:

$$G_c^* = \arg\min_{G_c} \; -I(G; G_c|S) + \beta I(G; G_c), \tag{13}$$

where each term denotes the conditional graph reconstruction and compression, respectively. Then, we present upper bounds for each term to guide the optimization.

Minimizing $-I(G; G_c|S)$ The first term of Eq. 13 denotes a reconstruction of the input graph G, given G_c conditioned on S. Thus, we utilize the chain rule for mutual information on the Conditional Graph Reconstruction term as follows:

$$\min -I(G; G_c|S) = \min -I(G; G_c, S) + I(G; S). \tag{14}$$

We observed that including the second term, i.e., $I(G; S)$, into our objectives severely degrades the overall model performance (Appendix D.1). That is, the model performs worse as we push the input graph G and its functional group candidates S far apart. Therefore, we only minimize the first term of Eq. 14.

The first term of Eq. 14 can be bounded as follows:

$$-I(G; G_c, S) \leq \mathbb{E}_{G;G_c,S}[-\log p_\varsigma(G|G_c, S)], \tag{15}$$

where $p_\varsigma(G|G_c, S)$ is a variational approximation of $p(G|G_c, S)$, which outputs the input graph G (see Appendix A.1). Thus, we model $p_\varsigma(G|G_c, S)$ as a graph structure reconstruction parametrized by ς, which outputs the graph G based on the G_c and S. Therefore, we can minimize the upper bound of $-I(G; G_c, S)$ by minimizing the graph reconstruction loss $L_{rec}(G; G_c, S)$, which can be modeled as

graph structure recovery. Specifically, given the output representation $\tilde{H}$, to minimize the graph structure loss, we first capture the similarity between any two nodes v_i and v_j in $\tilde{H}$ by employing cosine similarity, i.e., $\hat{A}_{i,j} = \frac{\tilde{H}_i^\top \tilde{H}_j}{\|\tilde{H}_i\| \|\tilde{H}_j\|}$, (Zhang et al. 2020). The reconstruction loss then can be defined as follows:

$$L_{rec} = \frac{1}{|V|^2} \|A - \hat{A}\|_F^2 \,, \tag{16}$$

where A refers to the original adjacency matrix of G and $||\cdot||_F$ is the Frobenius norm.

Minimizing $I(G; G_c)$ To minimize the second term of Eq. 13, we employ the sufficient encoder assumption that the latent representation Z is lossless in the encoding process, i.e., $I(Z|H) \approx I(G_c|G)$. To optimize the graph core G_c, we can employ a variational upper bound that is tractable and can be minimized during training (Yu, Cao, and He 2022). Formally, given the mutual information $I(G; G_c)$, the variational upper bound of $I(G; G_c)$ is:

$$L_{MI}\left(G, G_c\right) \leq \mathbb{E}\left(-\tfrac{1}{2}\log P + \tfrac{1}{2N}\log P + \tfrac{1}{2N}\log Q^2\right), \tag{17}$$

where $P = \sum_{j=1}^{N} \left(1 - \lambda_j\right)^2$, $Q = \frac{\sum_{j=1}^{N} \lambda_j (H_j - \mu_H)}{\sigma_H}$, and λ is computed from Eq. 6. For the proof of the upper bound, we refer readers to Appendix A.2.

Contrastive Learning We note that minimizing the upper bound of $I(G, G_c)$ from Eq. 17 could lead to over-compression with sufficient information on S. That is, the graph core G_c can be too distinguished from its input graph G compared to other graphs. We argue that the ideal core should at least satisfy the high mutual information with its input graph compared to others during the compression process, as $I\left(G_c, G\right) \geq I\left(G_c, \backslash G\right)$, where $\backslash G$ refers to remaining graphs, excluding G. Thus, we propose to use a contrastive learning-based method to maximize the agreement between the graph core and its input graph. Specifically, the contrastive objective is computed as:

$$L_{con} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp\left(s\left(\bar{Z}^i, \bar{H}^i\right)\right)}{\sum_{j=1, j\neq i}^{B} \exp\left(s\left(\bar{Z}^i, \bar{H}^j\right)\right)}, \tag{18}$$

where B denotes the number of graphs in a mini-batch, $s(\cdot, \cdot)$ is the cosine similarity between graph core and input graph, $\bar{Z} = \text{POOL}(Z)$, and $\bar{H} = \text{POOL}(H)$.

The overall loss for the pre-training task is as follows:

$$L_{total} = L_{con} + L_{rec} + \beta L_{MI}, \tag{19}$$

where β is a hyperparameter to balance the compression and structure preservation trade-off.

Domain Adaptation We note that our pre-trained model can learn significant subgraphs only on the domains of pre-training datasets. However, generalizing toward varied downstream tasks can be challenging due to the node attribute distinctions in specific downstream datasets. That is, while pre-training primarily focuses on structural molecular features, node feature reconstruction is also essential and helps our model adapt to learn node feature characteristics in downstream datasets. Therefore, we employ an unsupervised domain adaptation after pre-training, which acts as a domain-oriented generalization for downstream datasets. Then, we utilize a loss function to reconstruct node feature information for each graph as:

$$L_{att} = \frac{1}{|V|}\sum_{v_i \in V} \|x_i - \hat{x}_i\|_2, \tag{20}$$

where x_i is the initial feature of node v_i and $\hat{x}_i = \text{MLP}(\tilde{H}_i)$.

Evaluation

Experimental Settings

Datasets We conducted experiments across various molecular domains to evaluate the effectiveness of our proposed model, including Biophysics (mol-HIV, mol-PCBA, and BACE (Wu et al. 2018)), Physiology (BBBP, Tox21, ToxCast, SIDER, ClinTox, and MUV), Physical Chemistry (ESOL, FreeSolv, and Lipophilicity) (Wu et al. 2018), Bioinformatics (Mutagenicity, NCI1, and NCI109 (Morris et al. 2020)), Quantum Mechanics (PCQ4Mv2 and QM9 (Hu et al. 2021)). For pre-training datasets, we considered 300k unlabeled molecules sampled from three datasets, i.e., PCQ4Mv2, QM9, and mol-PCBA. The remaining datasets are used as fine-tuning datasets. Furthermore, we also evaluated the model's performance on large molecular graphs using two peptide molecules: peptide-func and peptide-struct (Dwivedi et al. 2022). For the model explainability, we utilized four datasets with the availability of ground truth, i.e., Mutagenicity, Benzene, Alkane Carbonyl, and Fluoride Carbonyl (Agarwal et al. 2023). We randomly split the datasets into training/validation/test sets with a ratio of 6:2:2. The datasets are given in Appendix B.

Baselines and Implementation Details We considered three groups of baselines. (i) The node-level pretraining methods are ContextPred and AttrMasking (Hu et al. 2020a), and EdgePred (Hamilton, Ying, and Leskovec 2017a). (ii) The contrastive learning methods are Infomax (Velickovic et al. 2019), JOAO and JOAOv2 (You et al. 2021), GraphCL (You et al. 2020), and GraphLoG (Xu et al. 2021). (iii) The subgraph-based methods are MICRO-Graph (Subramonian 2021), MGSSL (Zhang et al. 2021), GraphFP (Luong and Singh 2023), GROVE (Rong et al. 2020), SimSGT (Liu et al. 2023), and MoAMa (Inae, Liu, and Jiang 2023). We adopted a 5-layer GIN (Xu et al. 2019) as graph encoders. The hyperparameters and experimental setups are given in Appendix C. The open-source implementation of S-CGIB is available for reproducibility[1].

Performance Analysis

Tables 1 and 2 present the performance of our proposed model and the baselines on the graph classification task. We observed that: (i) S-CGIB consistently outperformed other baselines, obtaining the best performance in 10 out of 11 downstream datasets in the graph classification tasks. For the graph regression tasks, we can observe that S-CGIB

[1]https://github.com/NSLab-CUK/S-CGIB

Table 1: A performance comparison on graph classification tasks in terms of ROC-AUC. (D.A.: Domain Adaptation).

Methods	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE
ContextPred	69.10±0.29	73.26±0.59	63.28±0.68	61.83±0.60	55.63±1.35	71.43±0.79	72.04±0.48	78.39±0.58
AttrMasking	67.12±0.45	73.37±0.55	61.66±1.20	61.21±0.65	60.11±1.19	67.93±0.56	72.71±0.70	75.95±0.50
EdgePred	64.73±1.10	70.32±1.62	60.04±0.81	60.18±0.76	61.62±1.25	70.81±1.58	70.55±1.68	74.29±1.37
Infomax	68.39±0.64	72.66±0.16	62.76±0.54	59.02±0.56	58.62±0.83	72.14±1.25	73.55±0.47	77.80±0.46
JOAO	71.63±1.11	73.67±1.06	63.30±0.27	63.55±0.81	77.02±1.64	69.81±0.26	77.55±1.94	74.94±1.35
JOAOv2	71.98±0.18	73.95±1.88	63.12±1.90	59.88±1.72	65.22±0.75	68.50±1.58	77.13±1.51	74.38±1.71
GraphCL	68.39±0.64	73.26±0.59	62.76±0.54	61.83±0.60	61.62±1.25	72.14±1.25	73.55±0.47	77.80±0.46
GraphLoG	66.75±0.32	71.64±0.49	61.53±0.35	59.09±0.53	53.76±0.95	72.52±2.02	73.76±0.29	76.60±1.04
GraphFP	72.05±1.17	77.35±1.40	69.15±1.92	**65.93±3.09**	76.80±1.83	71.82±1.33	75.71±1.39	80.28±3.06
MICRO-Graph	67.21±1.85	71.79±1.70	60.80±1.15	60.34±0.96	77.56±1.56	70.46±1.62	76.73±1.07	63.57±1.55
MGSSL	79.52±1.98	74.82±1.60	63.86±1.57	57.46±1.45	75.84±1.82	73.44±3.47	77.45±2.94	82.03±3.79
GROVE	87.15±0.06	68.59±0.24	64.45±0.14	57.53±0.23	72.53±0.14	67.67±0.12	75.04±0.13	81.13±0.14
SimSGT	71.51±1.75	76.23±1.27	65.83±0.79	59.74±1.32	74.11±1.05	72.79±1.52	78.13±1.07	79.75±1.28
MoAMa	85.89±0.61	78.29±0.55	68.01±1.07	62.69±0.37	77.11±1.67	72.41±1.76	78.11±0.64	81.32±1.06
S-CGIB w/o D.A.	86.71±0.74	79.52±0.71	68.93±0.45	62.76±1.83	74.69±1.28	74.12±1.85	77.41±1.63	86.51±1.49
S-CGIB	**88.75±0.49**	**80.94±0.17**	**70.95±0.27**	64.03±1.04	**78.58±2.01**	**77.71±1.19**	**78.33±1.34**	**86.46±0.81**

Table 2: A performance comparison on graph classification tasks in terms of accuracy.

Methods	Mutagenicity	NCI1	NCI109
ContextPred	57.95±1.42	49.47±1.12	50.32±1.05
AttrMasking	58.03±1.16	49.51±1.21	46.25±1.73
EdgePred	48.58±1.02	49.88±1.12	49.19±1.36
Infomax	56.64±1.77	49.55±1.14	53.03±1.48
JOAO	62.33±1.13	49.03±1.21	58.23±1.49
JOAOv2	63.36±1.74	50.73±1.74	53.75±1.32
GraphCL	66.32±3.62	49.11±1.31	56.62±1.59
GraphLoG	66.47±1.47	60.94±1.93	57.52±1.61
GraphFP	68.43±1.32	53.77±1.13	58.14±1.45
MICRO-Graph	80.64±1.28	74.45±1.51	76.15±3.53
MGSSL	66.47±1.47	60.94±1.93	57.52±1.61
GROVE	80.49±0.94	75.79±0.91	76.01±0.73
SimSGT	68.27±0.53	56.93±0.43	60.48±0.31
MoAMa	80.37±0.87	78.59±0.81	76.82±1.05
S-CGIB w/o D.A.	80.26±0.71	78.51±1.06	77.08±1.55
S-CGIB (Ours)	**81.12±0.90**	**79.75±0.82**	**77.54±1.51**

Table 3: A performance comparison on regression tasks in terms of RMSE.

Methods	FreeSolv	ESOL	Lipophilicity
ContextPred	3.195±0.058	2.190±0.026	1.053±0.048
AttrMasking	4.023±0.039	2.954±0.087	0.982±0.052
EdgePred	3.192±0.023	2.368±0.070	1.085±0.061
Infomax	3.033±0.026	2.953±0.049	0.970±0.023
JOAO	3.282±0.002	1.978±0.029	1.093±0.097
JOAOv2	3.842±0.012	2.144±0.009	1.116±0.024
GraphCL	3.166±0.027	1.390±0.363	1.014±0.018
GraphLoG	2.335±0.052	1.542±0.026	0.932±0.052
GraphFP	2.528±0.016	2.136±0.096	1.371±0.058
MICRO-Graph	1.865±0.061	0.842±0.055	0.851±0.073
MGSSL	2.940±0.051	2.936±0.071	1.106±0.077
GROVE	2.712±0.327	1.237±0.403	0.823±0.027
SimSGT	1.953±0.038	0.932±0.026	0.771±0.041
MoAMa	2.072±0.053	1.125±0.029	1.085±0.024
S-CGIB w/o D.A.	1.832±0.095	0.894±0.052	0.803±0.067
S-CGIB (Ours)	**1.648±0.074**	**0.816±0.019**	**0.762±0.042**

consistently outperformed other methods on three regression benchmarks, as shown in Table 3. We attribute the superior performance of S-CGIB to its ability to generate well-distinguished representations and effectively capture significant substructures, i.e., functional groups. For example, in the BBBP dataset, the task is to predict the barrier permeability, where molecular structures with different sizes, skeletal ring structures, and functional groups will decide the penetrating properties. Capturing such structural information to generate graph-level molecular representations can enhance the prediction of molecular properties. (ii) Subgraph-level strategies, e.g., GraphFP and MGSSL, outperformed node-level and contrastive learning methods. This implies that subgraph-level methods could capture well global molecular structure and functional groups, which benefits molecular property prediction in downstream tasks. In contrast, S-CGIB not only learns well-separated representations but also automatically captures functional groups.

Performance on Large Molecular Graphs. To validate the model's ability on large molecules, we conducted experiments on two large molecular graphs, i.e., Peptides-func and Peptides-struct, as shown in Table 5. The results demonstrated that S-CGIB outperformed other baselines, including subgraph-based strategies. For example, on the Peptides-func dataset, our proposed model gained a 12.3% improvement compared to the GraphFP method. It indicates that S-CGIB could capture long-range dependencies by generating well-separated representations and then capturing significant subgraphs, thanks to the attention-based interaction between graph core and significant subgraphs. The results verified the effectiveness of our strategy for capturing well-separated representations and significant subgraphs.

Efficiency Analysis. Beyond performance improvement, we also validated the impacts of the pre-trained model's convergence and generalization. Figure 2 shows that S-CGIB

Table 4: An interpretability comparison on functional group detection tasks in terms of $Fidelity-/+$.

Methods	Mutagenicity		BENZENE		Alkane Carbonyl		Fluoride Carbonyl	
	$Fidelity-\downarrow$	$Fidelity+\uparrow$	$Fidelity-\downarrow$	$Fidelity+\uparrow$	$Fidelity-\downarrow$	$Fidelity+\uparrow$	$Fidelity-\downarrow$	$Fidelity+\uparrow$
ContextPred	0.061±0.002	0.223±0.004	0.419±0.008	0.483±0.005	0.261±0.001	0.293±0.007	0.363±0.018	0.413±0.024
AttrMasking	0.078±0.005	0.230±0.004	0.448±0.002	0.543±0.016	0.260±0.011	0.310±0.009	0.276±0.007	0.384±0.005
EdgePred	0.081±0.003	0.451±0.013	0.386±0.068	0.457±0.061	0.581±0.074	0.603±0.069	0.342±0.073	0.389±0.072
Infomax	0.064±0.004	0.240±0.008	0.363±0.041	0.410±0.082	0.353±0.021	0.376±0.054	0.331±0.032	0.453±0.012
JOAO	0.103±0.004	0.424±0.013	0.047±0.005	0.481±0.007	0.263±0.005	0.568±0.008	0.183±0.007	0.295±0.004
JOAOv2	0.152±0.005	0.431±0.008	0.062±0.006	0.481±0.009	0.387±0.004	0.586±0.007	0.184±0.007	0.207±0.008
GraphCL	0.283±0.008	0.476±0.002	0.120±0.005	0.469±0.008	0.430±0.027	0.578±0.016	0.284±0.002	0.570±0.001
GraphLoG	0.117±0.001	0.439±0.003	0.137±0.000	0.459±0.007	0.355±0.002	0.695±0.007	0.358±0.004	0.475±0.006
GraphFP	0.213±0.025	0.603±0.003	0.093±0.006	0.494±0.012	0.276±0.022	0.529±0.005	0.263±0.026	0.595±0.034
MICRO-Graph	0.235±0.008	0.466±0.016	0.140±0.012	0.483±0.003	0.155±0.037	0.524±0.015	0.281±0.007	0.595±0.002
MGSSL	0.150±0.005	0.489±0.006	**0.019±0.001**	0.480±0.006	**0.049±0.001**	0.205±0.002	0.231±0.001	0.550±0.006
GROVE	0.218±0.005	0.522±0.014	0.064±0.029	0.489±0.010	0.183±0.041	0.518±0.007	0.321±0.067	0.628±0.038
MoAMa	0.228±0.020	0.585±0.012	0.076±0.006	0.483±0.002	0.274±0.021	0.514±0.005	0.220±0.007	0.451±0.008
S-CGIB (Ours)	**0.008±0.001**	**0.638±0.003**	0.049±0.001	**0.720±0.003**	0.134±0.001	**0.727±0.003**	**0.133±0.002**	**0.672±0.004**

Table 5: A performance comparison on the two large molecular graph datasets.

Methods	Peptides-func (AP↑)	Peptides-struct (MAE↓)
ContextPred	0.311±0.013	0.587±0.001
AttrMasking	0.318±0.002	0.580±0.002
EdgePred	0.310±0.012	0.546±0.001
Infomax	0.335±0.013	0.574±0.001
JOAO	0.386±0.009	0.463±0.008
JOAOv2	0.398±0.009	0.541±0.008
GraphCL	0.380±0.002	0.973±0.014
GraphLoG	0.313±0.034	0.419±0.006
GraphFP	0.618±0.014	0.327±0.026
MICRO-Graph	0.505±0.014	0.332±0.002
MGSSL	0.541±0.006	0.322±0.008
GROVE	0.587±0.023	0.376±0.005
SimSGT	0.612±0.005	0.358±0.003
MoAMa	0.584±0.019	0.365±0.005
S-CGIB w/o D.A.	0.658±0.013	0.306±0.007
S-CGIB (Ours)	**0.694±0.002**	**0.269±0.004**

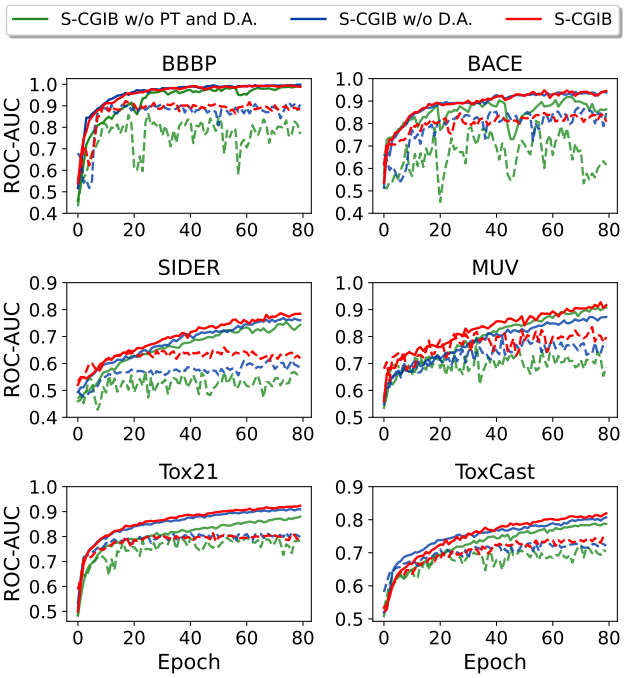

Figure 2: An efficiency analysis for variants of S-CGIB. The solid lines are training curves, and the dashed lines are validation curves (PT: Pre-training, D.A.: Domain Adaptation).

almost converged faster than that without pre-training and domain adaptation. This is because S-CGIB has already captured the important patterns from the pre-training dataset well. Besides, our pre-trained model is considered a one-time effort, which can significantly reduce the training and validation time on specific downstream datasets. Moreover, S-CGIB with pre-training and domain adaptation typically exhibits stable training and validation curves.

Analysis on Distinguishability of Representations To validate the model's ability to generate well-distinguished representations, we further conducted experiments to validate the well-distinguished representations between different molecular structures. We utilized Jensen-Shannon Divergence (JSD) measurement to validate the distinction between learned embeddings, as shown in Table 6. We adopted three datasets: BENZENE, Alkane Carbonyl, and Fluoride Carbonyl datasets, with ground-truth explanations, whose classes are labeled based on structures. We observed that our proposed model learned well-distinguished and robust representations. This robustness enhances the discriminability of molecular representations, confirming the effectiveness of our proposed model in capturing different molecular structures across molecules.

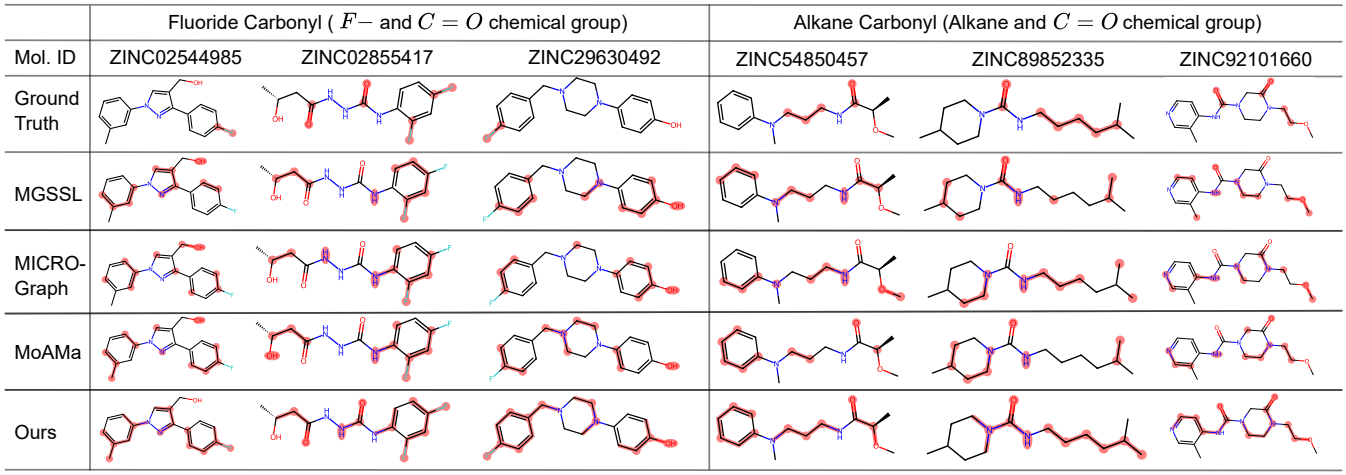

Figure 3: Visualizations of model interpretability in functional group detection tasks.

Table 6: A distinguishability comparison in terms of JSD.

Methods	BENZENE	Alkane Carbonyl	Fluoride Carbonyl
MGSSL	0.201±0.008	0.253±0.012	0.218±0.011
SimSGT	0.173±0.010	0.124±0.014	0.137±0.007
MoAMa	0.215±0.016	0.209±0.013	0.153±0.015
S-CGIB	**0.381±0.009**	**0.362±0.005**	**0.295±0.006**

Table 7: An ablation analysis on graph core (G_c) and attention coefficients (α).

G_c	α	BBBP	Tox21	ToxCast
–	–	81.15±0.75	78.68±0.63	68.39±0.48
✓	–	86.26±0.63	79.48±0.49	69.04±0.38
–	✓	83.41±0.71	78.85±0.16	69.77±0.67
✓	✓	**88.75±0.49**	**80.94±0.17**	**70.95±0.27**

Interpretability Analysis

There have been increasing concerns about the explainability of pre-trained GNNs, as GNNs can be seen as black boxes. Thus, we employed S-CGIB to explain the molecule prediction compared to ground-truth explanations on four downstream datasets, i.e., Mutagenicity, BENZENE, Alkane Carbonyl, and Fluoride Carbony (Agarwal et al. 2023). We used the *Fidelity* score (Amara et al. 2022) to assess how well the explanation aligns with S-CGIB, as shown in Table 4. The lower value of the $Fidelity-$ score shows a more reliable explanation, while the higher $Fidelity+$ score implies that more important nodes are recognized. For S-CGIB, we considered the top 50% of nodes with the highest attention score as explainable nodes. For the baseline methods, we identified the top 50% of nodes with the highest positive saliency values as explainable nodes, following the work Pope et al. (2019). We observed that S-CGIB achieved the highest fidelity scores on almost datasets in terms of the two metrics. This implies that S-CGIB generates a reliable explanation that is aligned with the model prediction. Moreover, we conducted the qualitative validation on graph interpretation via visualization on two datasets, i.e., Alkane Carbonyl and Fluoride Carbonyl datasets, as shown in Figure 3. We observed that S-CGIB provided a more accurate interpretation of molecules than the baselines.

Model Analysis

Ablation Analysis We further validate the contribution of graph core G_c and subgraph learning in S-CGIB, as shown in Table 7. To validate the importance of G_c, we considered the presence when using compression or not. For the subgraph learning, we evaluated the presence of attention coefficients to explore the significant subgraphs. We observed that: (i) S-CGIB with both modules performs best in all the downstream datasets. This result indicates that exploring graph core and significant subgraphs is crucial to fully generating a pre-trained model as well as the property prediction in downstream datasets. (ii) While considering the graph core is important, exploring significant subgraphs is more beneficial for molecular property prediction. For example, the use of graph core can achieve a second ahead in the BBBP dataset, while the model performance for only exploring significant subgraphs remains a close second ahead of other datasets. We argue that for molecular property prediction, while both graph core and significant subgraphs are important, capturing only significant subgraphs is slightly more beneficial than considering graph core for molecular prediction in specific downstream tasks.

Sensitivity Analysis We further conducted sensitivity analyses on the choice of graph encoders (Appendix D.3), the subgraph sizes (Appendix D.4), and the number of GIN layers (Appendix D.5). We observed that the GIN encoder showed the best performance among graph encoders, e.g., GCN, GraphSage, and GT (Dwivedi and Bresson 2021), which matches the previous findings (Luong and Singh 2023). For the subgraph size, S-CGIB gained the best performance when the subgraph size was small ($k \leq 3$). This indi-

cates that the subgraph size should be large enough to capture sufficient information but not larger, which can obtain noisy information. For the number of GIN layers, the model performance remained stable at $l \geq 3$ in most datasets, which helps S-CGIB capture the global graph structures.

Conclusion

In this paper, we present a novel pre-training strategy for molecules, named S-CGIB, which can discover graph core and significant subgraphs to generate well-distinguished representations. The main idea is to explore the graph cores of molecules that contain compressed and sufficient information regarding the reconstruction task conditioned on significant subgraphs under the S-CGIB principle. By doing so, S-CGIB can generate robust representations, improving performance in molecular property prediction tasks. The experiments over numerous domains showed that S-CGIB consistently outperforms baselines in various downstream tasks. Furthermore, S-CGIB also delivers model interpretability regarding the functional group detection task despite being learned from self-supervised learning.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1F1A1065516 and No. 2022K1A3A1A79089461) (O.-J.L.).

References

Agarwal, C.; Queen, O.; Lakkaraju, H.; and Zitnik, M. 2023. Evaluating explainability for graph neural networks. *Scientific Data*, 10(1): 144.

Alemi, A. A.; Fischer, I.; Dillon, J. V.; and Murphy, K. 2017. Deep Variational Information Bottleneck. In *5th International Conference on Learning Representations (ICLR 2017)*. Toulon, France: OpenReview.net.

Amara, K.; Ying, Z.; Zhang, Z.; Han, Z.; Zhao, Y.; Shan, Y.; Brandes, U.; Schemm, S.; and Zhang, C. 2022. GraphFramEx: Towards Systematic Evaluation of Explainability Methods for Graph Neural Networks. In *Proceedings of the 1st Learning on Graphs Conference (LoG 2022)*. Virtual Event.

Cao, S.; Lu, W.; and Xu, Q. 2015. GraRep: Learning Graph Representations with Global Structural Information. In *24th International Conference on Information and Knowledge Management (CIKM 2015)*, 891–900. Melbourne, VIC, Australia: ACM.

Chechik, G.; and Tishby, N. 2002. Extracting Relevant Structures with Side Information. In *Proceedings of the 15th Advances in Neural Information Information Processing Systems (NIPS 2002)*, 857–864. Vancouver, British Columbia, Canada: MIT Press.

Degen, J.; Wegscheid-Gerlach, C.; Zaliani, A.; and Rarey, M. 2008. On the art of compiling and using'drug-like'chemical fragment spaces. *ChemMedChem*, 3(10): 1503.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, 4171–4186. Minneapolis, MN, USA: ACL.

Dwivedi, V. P.; and Bresson, X. 2021. A Generalization of Transformer Networks to Graphs. In *Proceedings of the AAAI Workshop on Deep Learning on Graphs: Methods and Applications (AAAIW 2021)*.

Dwivedi, V. P.; Rampásek, L.; Galkin, M.; Parviz, A.; Wolf, G.; Luu, A. T.; and Beaini, D. 2022. Long Range Graph Benchmark. In *Proceedings of the 35th Advances in Neural Information Processing System (NeurIPS 2022)*. New Orleans, LA, USA.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *Proceedings of the ICLR Workshop on Representation Learning on Graphs and Manifolds (ICLRW 2019)*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, volume 70 of *PMLR*, 1263–1272. Sydney, NSW, Australia: PMLR.

Gondek, D.; and Hofmann, T. 2003. Conditional information bottleneck clustering. In *Proceedings of the 3rd IEEE international conference on data mining, workshop on clustering large data sets (ICMDW 2003)*, 36–42.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017a. Inductive Representation Learning on Large Graphs. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NeurIPS 2017)*, 1024–1034. Long Beach, CA, USA.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017b. Inductive Representation Learning on Large Graphs. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NeurIPS 2017)*, 1024–1034. Long Beach, CA, USA.

Hao, Z.; Lu, C.; Huang, Z.; Wang, H.; Hu, Z.; Liu, Q.; Chen, E.; and Lee, C. 2020. ASGN: An Active Semi-supervised Graph Neural Network for Molecular Property Prediction. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2020)*, 731–752. Virtual Event: ACM.

Hoang, V. T.; Jeon, H.-J.; You, E.-S.; Yoon, Y.; Jung, S.; and Lee, O.-J. 2023. Graph Representation Learning and Its Applications: A Survey. *Sensors*, 23(8).

Hoang, V. T.; and Lee, O. 2024. Transitivity-Preserving Graph Representation Learning for Bridging Local Connectivity and Role-Based Similarity. In *Proceedings of the 38th Conference on Artificial Intelligence (AAAI 2024)*, 12456–12465. Vancouver,Canada: AAAI Press.

Hu, W.; Fey, M.; Ren, H.; Nakata, M.; Dong, Y.; and Leskovec, J. 2021. OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs. In *Proceedings of the 1st Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS 2021)*. Virtual Event.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V. S.; and Leskovec, J. 2020a. Strategies for Pre-training Graph Neural Networks. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*. Addis Ababa, Ethiopia: OpenReview.net.

Hu, Z.; Dong, Y.; Wang, K.; Chang, K.; and Sun, Y. 2020b. GPT-GNN: Generative Pre-Training of Graph Neural Networks. In *Proceedings of the 26th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2020)*, 1857–1867. Virtual Event: ACM.

Inae, E.; Liu, G.; and Jiang, M. 2023. Motif-aware Attribute Masking for Molecular Graph Pre-training. *arXiv preprint*, arXiv:2309.04589.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*. Toulon, France: OpenReview.net.

Kazius, J.; McGuire, R.; and Bursi, R. 2005. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of medicinal chemistry*, 48(1): 312–320.

Kearnes, S.; McCloskey, K.; Berndl, M.; Pande, V. S.; and Riley, P. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8): 595–608.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*. Toulon, France: OpenReview.net.

Kong, X.; Huang, W.; Tan, Z.; and Liu, Y. 2022. Molecule Generation by Principal Subgraph Mining and Assembling. In *Proceedings of the 35th Advances in Neural Information Processing Systems (NeurIPS 2022)*. New Orleans, LA, USA.

Lee, N.; Lee, J.; and Park, C. 2022. Augmentation-Free Self-Supervised Learning on Graphs. In *Proceedings of the 36th Conference on Artificial Intelligence (AAAI 2022)*, 7372–7380. Virtual Event: AAAI Press.

Liu, S.; Wang, H.; Liu, W.; Lasenby, J.; Guo, H.; and Tang, J. 2022. Pre-training Molecular Graph Representation with 3D Geometry. In *Proceedings of the 10th International Conference on Learning Representations (ICLR 2022)*. Virtual Event: OpenReview.net.

Liu, Z.; Shi, Y.; Zhang, A.; Zhang, E.; Kawaguchi, K.; Wang, X.; and Chua, T. 2023. Rethinking Tokenizer and Decoder in Masked Graph Modeling for Molecules. In *Proceedings of the 36th Advances in Neural Information Processing Systems (NeurIPS 2023)*. New Orleans, LA, USA.

Luong, K.; and Singh, A. K. 2023. Fragment-based Pretraining and Finetuning on Molecular Graphs. In *Proceedings of the 36th Advances in Neural Information Processing Systems (NeurIPS 2023)*. New Orleans, LA, USA.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*. Toulon, France.

Morris, C.; Kriege, N. M.; Bause, F.; Kersting, K.; Mutzel, P.; and Neumann, M. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. In *Proceedings of the ICML 2020 Workshop on Graph Representation Learning and Beyond (ICMLW 2020)*.

Pope, P. E.; Kolouri, S.; Rostami, M.; Martin, C. E.; and Hoffmann, H. 2019. Explainability Methods for Graph Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019)*, 10772–10781. Long Beach, CA, USA: Computer Vision Foundation / IEEE.

Qiu, J.; Chen, Q.; Dong, Y.; Zhang, J.; Yang, H.; Ding, M.; Wang, K.; and Tang, J. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *Proceedings of the Conference on Knowledge Discovery and Data Mining (KDD 2020)*, 1150–1160. Virtual Event: ACM.

Rong, Y.; Bian, Y.; Xu, T.; Xie, W.; Wei, Y.; Huang, W.; and Huang, J. 2020. Self-Supervised Graph Transformer on Large-Scale Molecular Data. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems (NeurIPS 2020)*. Virtual Event.

Stärk, H.; Beaini, D.; Corso, G.; Tossou, P.; Dallago, C.; Günnemann, S.; and Lió, P. 2022. 3D Infomax improves GNNs for Molecular Property Prediction. In *Proceedings of the International Conference on Machine Learning (ICML 2022)*, volume 162 of *PMLR*, 20479–20502. Baltimore, Maryland, USA: PMLR.

Sterling, T.; and Irwin, J. J. 2015. ZINC 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11): 2324–2337.

Subramonian, A. 2021. MOTIF-Driven Contrastive Learning of Graph Representations. In *Proceedings of the 35th Conference on Artificial Intelligence (AAAI 2021)*, 15980–15981. Virtual Event: AAAI Press.

Tishby, N.; Pereira, F. C. N.; and Bialek, W. 2000. The information bottleneck method. *arXiv preprint*, arXiv:physics-0004057.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*. Vancouver, BC, Canada: OpenReview.net.

Velickovic, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*. New Orleans, LA, USA: OpenReview.net.

Wang, M.; Zheng, D.; Ye, Z.; Gan, Q.; Li, M.; Song, X.; Zhou, J.; Ma, C.; Yu, L.; Gai, Y.; Xiao, T.; He, T.; Karypis, G.; Li, J.; and Zhang, Z. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315*.

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

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*. New Orleans, LA, USA: OpenReview.net.

Xu, M.; Wang, H.; Ni, B.; Guo, H.; and Tang, J. 2021. Self-supervised Graph-level Representation Learning with Local and Global Structure. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, volume 139 of *PMLR*, 11548–11558. Virtual Event: PMLR.

You, Y.; Chen, T.; Shen, Y.; and Wang, Z. 2021. Graph Contrastive Learning Automated. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, volume 139 of *PMLR*, 12121–12132. Virtual Event: PMLR.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph Contrastive Learning with Augmentations. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS 2020)*. Virtual Event.

Yu, J.; Cao, J.; and He, R. 2022. Improving Subgraph Recognition with Variational Graph Information Bottleneck. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2022)*, 19374–19383. New Orleans, LA, USA: IEEE.

Yu, J.; Xu, T.; Rong, Y.; Bian, Y.; Huang, J.; and He, R. 2021. Graph Information Bottleneck for Subgraph Recognition. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*. Virtual Event: OpenReview.net.

Zhang, J.; Zhang, H.; Xia, C.; and Sun, L. 2020. Graph-Bert: Only Attention is Needed for Learning Graph Representations. *CoRR*, abs/2001.05140.

Zhang, Z.; Liu, Q.; Wang, H.; Lu, C.; and Lee, C. 2021. Motif-based Graph Self-Supervised Learning for Molecular Property Prediction. In *Proceedings of the 34th Advances in Neural Information Processing Systems (NeurIPS 2021)*, 15870–15882. Virtual Event.

Appendix

A. Proofs

A.1. Proof of Equation 15 in the main text

Recall that Eq. 15 in the main text is given as:

$$-I(G;G_c,S) \leq \mathbb{E}_{G;G_c,S}[-\log p_\zeta(G|G_c,S)]. \quad (21)$$

By employing variational approximation $p_\zeta(G|G_c,S)$ to approximate intractable distribution $p(G|G_c,S)$, we can have:

$$\begin{aligned} I\left(G;G_c,S\right) &= \mathbb{E}_{G,G_c,S}\left[\log \frac{p\left(G|G_c,S\right)}{p\left(G\right)}\right] \\ &= \mathbb{E}_{G,G_c,S}\left[\log \frac{p_\zeta\left(G|G_c,S\right)}{p\left(G\right)}\right] \\ &+ \mathbb{E}_{G_c,S}\left[KL\left(p\left(G|G_c,S\right)||p_\zeta\left(G|G_c,S\right)\right)\right]. \end{aligned} \quad (22)$$

Based on the Non-negativity of Kullback–Leibler Divergence, we have:

$$\begin{aligned} I\left(G;G_c,S\right) &\geq \mathbb{E}_{G,G_c,S}\left[\log \frac{p_\zeta\left(G|G_c,S\right)}{p\left(G\right)}\right] \\ &= \mathbb{E}_{G,G_c,S}\left[\log p_\zeta\left(G|G_c,S\right)\right] + H\left(G\right). \end{aligned} \quad (23)$$

A.2. Proof of Equation 17 in the main text

We first employ a pooling function to get the representation of the graph core as $\bar{Z} = \text{POOL}(Z)$. We employ the sufficient encoder assumption that the latent representation $\bar{Z}$ is lossless in the encoding process, meaning that $\bar{Z}$ captures all the information from the graph core, i.e., $I(\bar{Z}|G) \approx I(G_c|G)$. We then introduce the upper bound of $I(\bar{Z};G)$ by employing a variation approximation, as:

$$\begin{aligned} I\left(\bar{Z};G\right) &= \mathbb{E}_{\bar{Z},G}\left[\log \frac{p\left(\bar{Z}|G\right)}{p\left(\bar{Z}\right)}\right] \\ &= \mathbb{E}_{G}\left[KL\left(p\left(\bar{Z}|G\right)||q\left(\bar{Z}\right)\right)\right] \\ &- \mathbb{E}_{\bar{Z},G}\left[KL\left(p\left(\bar{Z}\right)||q\left(\bar{Z}\right)\right)\right]. \end{aligned} \quad (24)$$

Based on the Non-negativity of Kullback–Leibler Divergence, we have:

$$I\left(\bar{Z};G\right) \leq \mathbb{E}_{G}\left[KL\left(p_\phi\left(\bar{Z}|G\right)||q\left(\bar{Z}\right)\right)\right]. \quad (25)$$

Based on the VIB principle (Alemi et al. 2017), we assume that $q(\bar{Z})$ can be obtained by aggregating all the node embeddings from G_c in fully perturbed data. The noise $\epsilon \sim \mathcal{N}(\mu_H, \sigma_H)$ be sampled from the Gaussian distribution, where μ_H and σ_H refer to mean and variance of H. As the summation of a Gaussian distribution results in a Gaussian distribution, we choose sum pooling as the POOLING function; we have:

$$q\left(\bar{Z}\right) = \mathcal{N}\left(N\mu_H, N\sigma_H^2\right), \quad (26)$$

where N is the number of nodes in the input graph. Then, for the $p_\phi\left(\bar{Z}|G\right)$, we have:

$$\begin{aligned} p\left(\bar{Z}|G\right) = &\mathcal{N}\left(N\mu_H + \sum_{j=1}^{N}\lambda_j H_j - \sum_{j=1}^{N}\lambda_j\mu_H,\right. \\ &\left.\sum_{j=1}^{N}\left(1-\lambda_j\right)^2\sigma_H^2\right). \end{aligned} \quad (27)$$

We then plug Eq. 27 and Eq. 26 into Eq. 25, we have:

$$I\left(\bar{Z};G\right) \leq \mathbb{E}_{G}\left[-\frac{1}{2}\log P + \frac{1}{2N}P + \frac{1}{2N}Q^2\right] + r, \quad (28)$$

where $P = \sum_{j=1}^{N}\left(1-\lambda_j\right)^2$, $Q = \frac{\sum_{j=1}^{N}\lambda_j(H_j-\mu_H)}{\sigma_H}$, r denotes a constant value that can be overlooked during the mode training.

B. Statistics of Datasets

We provide details on the datasets used in our experiments. For pre-training datasets, we sampled 300k molecules from three datasets, i.e., PCQ4Mv2, QM9 (Hu et al. 2021), and mol-PCBA (Wu et al. 2018)). Specifically, we sampled 100k molecules from each dataset. The detailed statistics for fine-tuning datasets are summarized in Table 8. For the datasets used in the interpretability analysis task, we used 4 datasets, e.g., Mutagenicity, BENZENE, Alkane Carbonyl, and Fluoride Carbony, with their ground-truth explanations (Agarwal et al. 2023), as:

- The Mutagenicity (mutag) dataset consists of 1,768 molecules, each labeled according to its mutagenic properties. The dataset is pruned from the original mutagenicity dataset (Kazius, McGuire, and Bursi 2005) (4,337 molecules) for the explainability task.
- The BENZENE dataset consists of 12,000 molecules extracted from the ZINC dataset (Sterling and Irwin 2015), where each molecule is labeled into one of two classes based on the presence or absence of a benzene zing.
- The Alkane Carbonyl dataset consists of 1,125 molecules, each labeled based on the presence of an unbranched alkane and a carbonyl ($C = O$).
- The Fluoride Carbonyl consists of 8,671 molecules, labeled according to the presence of fluoride (F-) and a carbonyl ($C = O$).

C. Implementation Details

For the pre-training, we used 5-layer GIN as our GNN encoders for both graph encoding and subgraph learning to learn the representations. The GIN architecture, which is an expressive power, can distinguish molecular representations with different molecular structures.

Model training Detailed hyperparameter specifications are given in Table 10. We conducted a search on the embedding dimensions $\{32, 64, 128, 256\}$. The β is determined with a grid search among $\{0.01, 0.1, 1.0, 10\}$. For the pre-training phase, we train the model for 600 epochs using Adam optimizer with 1×10^{-4} learning rate and 1×10^{-5} weight decay. We also tune the temperature parameter τ among $\{1.0, 0.5, 0.1\}$.

Baselines We compared S-CGIB to three groups of baselines, including node-level pre-training, contrastive learning, and subgraph-based pre-training strategies. These strategies are recent self-supervised pre-training methods on molecular graphs. We follow closely the settings from these studies for a fair comparison. For node-level pre-training strategies,

Table 8: The summary of statistics of datasets.

Category	Dataset	# Tasks	Task Type	# Graphs	Dimension	Metric	Avg. # nodes
Biophysics	mol-HIV	1	Classification	41,127	9	ROC-AUC	25.5
	BACE	1	Classification	1,513	9	ROC-AUC	34.1
Physiology	BBBP	1	Classification	2,039	9	ROC-AUC	23.9
	Tox21	12	Classification	7,831	9	ROC-AUC	18.6
	ToxCast	617	Classification	8,575	9	ROC-AUC	18.7
	SIDER	27	Classification	1,427	9	ROC-AUC	33.6
	ClinTox	2	Classification	1,478	9	ROC-AUC	26.1
	MUV	17	Classification	93,087	9	ROC-AUC	24.2
Physical Chemistry	Lipophilicity	1	Regression	4,200	9	RMSE	27.0
	ESOL	1	Regression	1,128	9	RMSE	13.3
	FreeSolv	1	Regression	642	9	RMSE	8.7
Bioinformatics	Mutagenicity	2	Classification	4,337	21	ACC	30.32
	NCI1	2	Classification	4,110	37	ACC	29.84
	NCI109	2	Classification	4,127	38	ACC	29.66
LRGB	Peptides-func	10	Classification	15,535	9	AP	150.94
	Peptides-struct	11	Regression	15,535	9	MAE	150.94

Table 9: Performance according to graph encoders.

Encoders	BBBP	Tox21	ToxCast	SIDER	MUV	BACE
GCN	76.53±0.48	68.61±0.55	64.74±0.35	55.54±0.17	70.14±0.54	71.20±0.02
GraphSage	83.04±0.38	77.44±0.38	67.75±0.41	57.52±0.06	67.93±0.46	70.47±0.36
GAT	53.31±0.61	58.50±1.18	57.90±0.03	53.41±0.50	63.92±0.86	57.06±2.65
GT	82.55±1.34	77.75±0.24	69.99±0.37	57.15±0.26	68.27±0.92	68.03±1.81
GIN	**88.75±0.49**	**80.94±0.17**	**70.95±0.27**	**64.03±1.04**	**77.71±1.19**	**86.46±0.81**

we evaluated the S-CGIB performance against three node-level strategies:

- ContextPred (Hu et al. 2020a) strategy aims at predicting the k-hop surrounding structures given a target node.
- AttrMasking (Hu et al. 2020a) strategy masks the initial feature of node/edge and predicts these features as a node feature recovery task.
- EdgePred (Hamilton, Ying, and Leskovec 2017a) strategy predicts the presence of an edge between two nodes in graphs.

For contrastive learning, we evaluated the S-CGIB performance against five methods:

- Infomax (Velickovic et al. 2019) strategy aims to maximize the agreement between the graph-level representations and its sampled subgraphs.
- JOAO (You et al. 2021) strategy generates a set of augmentation schemes, i.e., node dropping, subgraph augmentation, edge perturbation, feature masking, and identical, and try to automatically find the useful augmentations.
- JOAOv2 (You et al. 2021) strategy is an improved version of JOAO that can estimate the distributions of initial node features and augmented features to generate more robust augmentation by modifying the projection head.
- GraphCL (You et al. 2020) aims at generating multiple views of graphs based on four graph augmentations, i.e., node dropping, edge perturbation, node feature masking, and sampled subgraph, and then maximizes the agreement between these views based on contrastive objectives.
- GraphLoG (Xu et al. 2021) discover the global graph structures by using hierarchical prototypes by contrasting graph pairs in a sampled batch.

For subgraph-level pre-training strategies, we considered six methods:

- GraphFP (Luong and Singh 2023) strategy aims to decompose input molecules into a set of subgraphs based on a dictionary (a bag of subgraphs based on subgraph frequent mining), which benefits the model capture semantic subgraphs.
- MICRO-Graph (Subramonian 2021) strategy aims to generate a bag of prototypical motifs and automatically learn the important functional group-like motifs.
- MGSSL (Zhang et al. 2021) strategy fragments the input molecules into a bag of functional groups based on an improved algorithm of BRICS by considering ring structures.
- GROVE (Rong et al. 2020) extract 85 functional groups

Table 10: Hyperparameters of S-CGIB used in experiments.

Hyperparameters	Values
Batch size	128
Number of GIN layers	5
Initial feature dimension	32
Embedding dimension	64
Number of pre-training epochs	600
Number of domain adaptation epochs	50
Adam: initial learning rate	1×10^{-4}
Adam: weight decay	1×10^{-5}
POOLING function	SUM
β	1.0
τ	1.0

based on discovering frequent subgraphs based on RDKit and utilize a graph transformer architecture.

- SimSGT (Liu et al. 2023) employs a set of strategies: breaking the input molecule into smaller subgraphs based on frequent subgraph mining, masking node features, and then recovering these features.
- MoAMa (Inae, Liu, and Jiang 2023) masks the whole sampled motifs and then predicts these features based on a reconstruction task.

Training Resources The experiments were conducted in two servers with four NVIDIA RTX A5000 GPUs (24GB RAM/GPU). Our model was developed and tested in Python 3.8.8 using Torch-geometric (Fey and Lenssen 2019) and DGL Library (Wang et al. 2019). For the environmental settings, we ran the experiments on Ubuntu 20.04 LTS server.

D. Additional Experiments

D.1. Performance according to the presence of $I(G, S)$ Recall that the conditional graph construction term in Eq. 14 is $\min -I(G; G_c|S) = \min -I(G; G_c, S) + I(G; S)$, and our objective is to minimize this term. Here, we investigate the effect of minimizing the second term, i.e., $I(G; S)$, by conducting experiments on different weight coefficients for this term in the total loss function. That is, we assign a weight coefficient for the $I(G; S)$ as $\zeta I(G; S)$, along with the total pre-training loss. Note that when $\zeta = 0$, it indicates that we do not minimize this term. Formally, we can define the objective as follows:

$$\min_{f_\phi, g_\theta} \frac{1}{B} \sum_{i=1}^{B} \frac{1}{N} \sum_{j=1}^{N} I\left(H_S^j, \bar{H}^i\right), \tag{29}$$

where B refers to the number of graphs in a batch, N is the number of nodes in the input graph G, $\bar{H} = \text{POOL}(H)$, and H_S^j is the embeddings of a subgraph rooted at node j. We then can simply model $I(\cdot, \cdot)$ as the dot product, i.e., $I(H_S^j, \bar{H}^i) \approx H_S^j \cdot \bar{H}^i$. As shown in Fig. 4, we observed that the model performance decreased when the weight coefficient of $I(G; S)$ increased. This is because minimizing $-I(G; G_c|S)$ is interpreted in terms of graph structure recovery given G_c conditioned on S, i.e., the pre-trained model focuses on the G_c as well as S. We argue that in the learning functional groups, the S are relevant to the graph core G_c to reconstruct the original graph and benefit the model by capturing significant subgraph candidates.

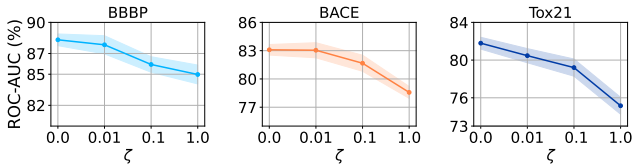

Figure 4: Performance according to weighting factor ζ for the term $I(G; S)$ in Eq. 14.

Table 11: An ablation analysis on different decoders: Fully Connected Layers (FC), Adjacency matrix reconstruction (Adj.), and k-step transition matrix reconstruction (k-step).

FC	Adj.	k-step	BBBP	Tox21	ToxCast
✓	✓	–	88.06±0.37	**81.15±0.44**	70.81±0.63
✓	–	✓	87.53±0.42	78.63±0.39	68.02±0.81
–	–	✓	86.14±0.31	78.12±0.25	69.04±0.19
–	✓	–	**88.75±0.49**	80.94±0.17	**70.95±0.27**

D.2. Performance according to the use of decoders To validate the use of a decoder and global graph structure preservation, we further conducted experiments with Fully Connected layers as a decoder to reconstruct the input adjacency matrix (Zhang et al. 2021). Furthermore, we also investigated the reconstruction of a k-step transition probability matrix to preserve global graph structures (Cao, Lu, and Xu 2015), as shown in Table 11 (k = 2). We observed that S-CGIB with two FC layers achieves accuracy comparable to S-CGIB, which indicates that adding FC layers does not necessarily lead to better performance. This implies that adjacency matrix reconstruction is sufficient to help S-CGIB generate well-distinguished representations. Moreover, reconstructing a k-step transition probability matrix does not improve the model performance compared to adjacency matrix reconstruction. This is because the local neighborhood information is sufficient for the model to learn good representations, achieving a more adequate balance due to the trade-off between compression and reconstruction.

D.3. Performance according to graph encoders Table 9 shows the performances of representative graph encoders, e.g., GCN (Kipf and Welling 2017), GraphSage (Hamilton, Ying, and Leskovec 2017b), GAT (Velickovic et al. 2018), and GT (Dwivedi and Bresson 2021), besides GIN (Xu et al. 2019) encoders. We observed that the GIN encoder is the most expressive GNN among other graph encoders. This implies that using an expressive model is crucial to fully utilize pre-training and that pre-training can even hurt performance when used on models with limited expressive power, e.g., GCN, GraphSAGE, and GAT. This observation matches with the findings from previous studies (Hu et al. 2020a; Luong and Singh 2023; Xu et al. 2019).

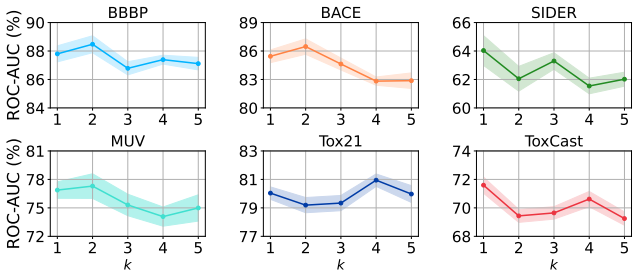

Figure 5: Performance according to subgraph sizes (k).

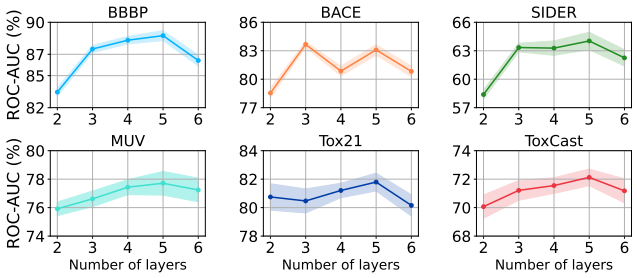

Figure 6: Performance according to the number of GIN layers.

D.4. Sensitivity analysis on the subgraph sizes We verify the performance of S-CGIB according to the choice of subgraph size k, as shown in Figure 5. We observed that when k is small, i.e., 1, 2, 3, the pre-trained model gains the top performance in almost all the datasets. For example, in the BBBP dataset, the model achieves the highest performance when the size $k = 2$ is a 2-hop subgraph rooted at each node. This implies that the subgraph size should be large enough to capture sufficient and significant subgraphs but not larger, which can obtain noisy nodes.

D.5. Sensitivity analysis on the number of layers We observed that as the number of layers increases, i.e., $l = 3$ and $l = 5$, the model performance also slightly improved, showing the ability to capture high-order structures of graph core and functional subgraph candidates.