Unsupervised Hierarchical Graph Pooling via Substructure-Sensitive Mutual Information Maximization

Ning Liu
liuning17a@nudt.edu.cn
National University of Defense Technology
Changsha, China

Songlei Jian*
jiansonglei@nudt.edu.cn
National University of Defense Technology
Changsha, China

Dongsheng Li*
dsli@nudt.edu.cn
National University of Defense Technology
Changsha, China

Hongzuo Xu
xuhongzuo13@nudt.edu.cn
National University of Defense Technology
Changsha, China

ABSTRACT

Graph pooling plays a vital role in learning graph embeddings. Due to the lack of label information, unsupervised graph pooling has received much attention, primarily via mutual information (MI). However, most existing MI-based pooling methods only preserve node features while overlooking the hierarchical substructural information. In this paper, we propose SMIP, a novel unsupervised hierarchical graph pooling method based on substructure-sensitive MI maximization. SMIP reconstructs a hard-style substructure encoder based on cluster-based pooling paradigm, and trains it with two substructure-sensitive MI-based objectives, i.e., node-substructure MI and node-node MI. The node-substructure MI guides to transfer maximum node feature information into corresponded substructures and the node-node MI guarantees a more accurate node allocation. Moreover, to avoid extra computation of augmented graphs and prevent noise information during MI estimation, we propose a local-scope contrastive MI estimation method, making SMIP more potent in capturing intrinsic features of the input graph. Experiments on six benchmark graph classification datasets demonstrate that our hierarchical deep learning approach outperforms all state-of-the-art unsupervised GNN-based methods and even surpasses the performance of nine supervised ones. Generalization study shows that the proposed substructure-sensitive MI objective can be successfully embedded into other cluster-based pooling methods to improve their performance.

CCS CONCEPTS

• **Information systems** → *Data mining*.

KEYWORDS

unsupervised graph representation learning, graph pooling

*Corresponding authors.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557485

ACM Reference Format:
Ning Liu, Songlei Jian, Dongsheng Li, and Hongzuo Xu. 2022. Unsupervised Hierarchical Graph Pooling via Substructure-Sensitive Mutual Information Maximization. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557485

1 INTRODUCTION

Graphs, providing explicit dependencies among nodes, have been proven to be an effective way to achieve reliable analysis for diverse data types. Currently, Graph Neural Networks (GNNs) [21] have made remarkable strides in learning node embeddings for graph-structured data to perform tasks such as node classification [39] and link prediction [33]. However, it remains a challenging research problem to combine GNNs with deterministic graph pooling strategy to learn graph embeddings for graph-level tasks, such as graph classification [44] and graph isomorphism [42]. Although there have been a myriad of successful supervised graph pooling methods, an obvious obstacle is the heavy reliance on additional graph labels, which is expensive to acquire by manual annotation. Hence, unsupervised graph pooling, which aims to learn effective graph embeddings based on a large amount of unlabeled graph-structured data, is of great practical value.

Recently, inspired by the InfoMax principle [25], there have been explosive interests in explorations of unsupervised graph pooling based on mutual information (MI). In general, most existing MI-based graph pooling methods [10, 15, 36, 40] design an MI-maximization objective between node embeddings obtained from GNNs and a graph embedding summarized from a readout function (e.g., element-wise sum), where MI measures the dependency between the two variables. Generally speaking, the higher the MI value, the lower the uncertainty of graph embedding given node embeddings, the better the graph embedding quality. Although achieving competitive results compared with other unsupervised approaches, current MI-based graph pooling methods have an obvious drawback, they lose the completeness of graph hierarchical substructural information. Maximizing the node-graph MI has been proved in [36] to be equivalent to distill information from the input node features into hidden vectors. As a result, these methods over-emphasize individual feature information of nodes within a graph but completely neglect hierarchical substructural information. However, hierarchical substructures are widespread and

contain abundant information in real-world graphs. For example, polymerized high molecular compound is a typically hierarchical graph in material science, in which a monomer is formed by bonding atoms, and then large molecules result from the simultaneous polymerisation of two or more monomer units.

To overcome the above drawback, in this work, we propose a novel unsupervised hierarchical graph pooling method based on substructure-sensitive MI maximization to obtain more accurate embeddings of graphs, named SMIP. To this end, we consider graph pooling as a node clustering problem and train a substructure encoder reconstructed from cluster-based pooling paradigm. Specifically, we reconstruct the learnable soft node assign matrix in a hard style to prevent blending less important nodes which are actually noise into substructures. Moreover, we interpret the node assign matrix which is commonly recognized as an assignment probability matrix from a new perspective as a bridge matrix which offers direct feature connection between nodes and substructures. To optimize the substructure encoder for high-quality coarse-grained graph embeddings, we propose node-substructure MI that helps to inherit individual node features into substructures as much as possible based on the assumption that nodes are already allocated to proper substructures. Further, for a more accurate node allocation relationship, we propose node-node MI based on node features getting from the bridge feature matrix to guarantee that nodes allocated to the same cluster be highly correlated with similar features. To achieve MI estimation, we apply contrastive learning. However, classical contrastive method usually requires extra graph augmentation to provide negative samples, which wastes computing resources and introduces noise information from augmented graphs. To better distill the intrinsic features of graphs, we propose a local-scope contrastive learning strategy that generates negative samples within the local scope of the input graph, thus avoiding data augmentation.

The main contributions can be summarized as follows:

- We propose SMIP, an unsupervised MI-based hierarchical graph pooling method that can preserve both individual node features and hierarchical substructural information to obtain high-quality graph embeddings.
- We propose node-substructure MI ad node-node MI, two substructure-sensitive MI-based unsupervised objective that can be incorporated into any pooling methods that follow the cluster-based pooling paradigm.
- We propose a local-scope contrastive learning strategy for MI estimation, which helps to save computing resources and discard external disturbance from augmented graphs, making our model more powerful in capturing intrinsic features of the input graph.

Experiments on six graph classification benchmarks demonstrate that SMIP is superior to nine state-of-the-art unsupervised competitors with an average improvement of 6.83%, and even exceeds nine supervised baselines. The generalization study shows that our proposed substructure-sensitive MI-based objective can be seamlessly embedded into existing cluster-based pooling methods and significantly improve their performance.

2 RELATED WORK

2.1 Unsupervised Graph Pooling

Unsupervised graph pooling is an important research field with a long history. Graph kernels [22, 31, 34, 35, 43] is a form of classic unsupervised methods specifically established for solving graph classification tasks using particular similarity measures—kernels. However, the theoretical properties of graph kernels lead to their suitability and specialization to a particular classification domain. Besides, implicit graph vectors in fixed feature space render an unavoidable barrier for the utilization of ML algorithms. Another typical approach is shallow learning-based methods built upon fixed-length random walks, such as node2vec [13], sub2vec [1] and graph2vec [28]. However, these methods only focus on refining the lowest-level node information reflected in the neighborhood but ignore the optimization of graph embedding correlated with coarse-grained structural information. Recently, with the great success of deep learning and GNNs, significant effort has been devoted to designing GNN-based graph representation learning methods in an unsupervised fashion. UGRAPHEMB [2] learns graph embeddings under existing graph proximity metric, e.g., Graph Edit Distance (GED) [11], which is expensive to acquire through high computational complexity algorithms. StructPool [45], MinCutPool [4] and OTCOARSENING [27] formulate graph pooling as a node clustering problem by aggregating nodes into multiple clusters as new nodes constrained by mincut loss, conditional random fields (CRFs), and optimal transport, respectively. They do not require extra prior knowledge but might not obtain sufficient performance when hierarchically coarsen the graph representations. This is because graphs after one coarsening operation become fully connected, then the mincut loss in MinCutPool and pairwise energy loss defined by l-hop connectivity in StrucPool both fail. More recently, InfoGraph [36], MVGRL [15], LGR [41], CuCo [8] and UHGR [10] are proposed to produce some new forms of formulation by generalizing the conventional mutual information (MI) estimation to the graph domain. Although achieving state-of-the-art graph classification results, they neglect the hierarchical substructural information.

2.2 Mutual Information Neural Estimation

MI estimation is intensively studied by the community in unsupervised learning of features since the InfoMax [25] principle is shown to be effective. MI is historically difficult to compute, especially for general-purpose continuous variables in deep neural networks. Fortunately, Mutual Information Neural Estimation (MINE) [3] allows for MI lower-bound estimation between high dimensional continuous random variables by training a statistics network through contrastive learning. The presentation of MINE has encouraged the research of a wide variety of deep models to resolve tasks based on MI maximization. Deep InfoMax (DIM) [16] employs MI maximization between local patches (i.e., hidden vectors) and high-level image representation. Deep Graph Infomax (DGI) [40] and InfoGraph [36] follow the intuition from DIM to graph domain with MI maximization between node representations and the graph-level representation summarized by a readout function. Specifically, DGI requires generating a corrupted graph to obtain negative samples. GMI [30] abandons the potentially harmful readout and corrupted function. Instead, it defines the MI maximization in a node-wise

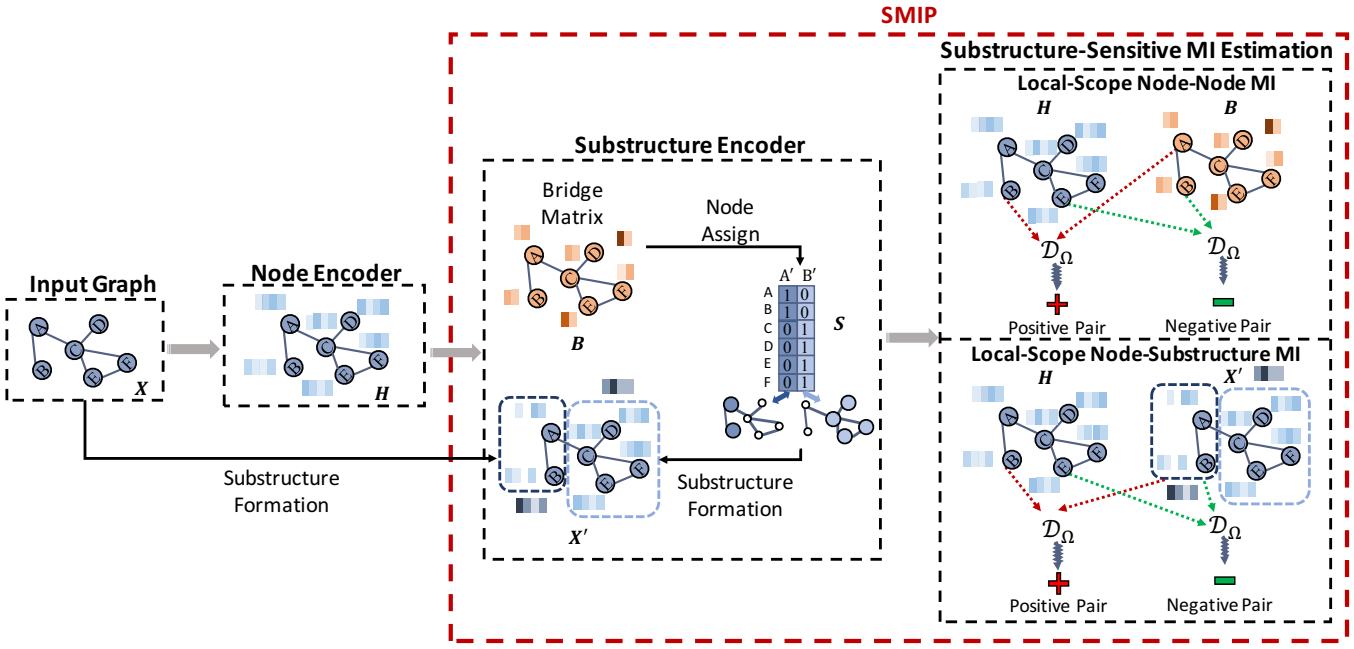

Figure 1: An illustration of the SMIP layer.

form between nodes and the selected l-hop neighbors. Apart from MINE, InfoNCE [38] is another MI lower bound neural estimation method based on Noise Contrastive Estimation [14]. Besides, researchers have recently proposed an upper-bound neural estimation method CLUB [7] for MI minimization tasks.

3 THE PROPOSED METHOD: SMIP

In this section, we first introduce preliminaries of unsupervised graph pooling. Then we formalize a substructure encoder reconstructed from cluster-based pooling paradigm. Subsequently, we introduce the substructure-sensitive MI maximization objectives used for training the substructure encoder and provide theoretical analysis. After that, we will describe the local-scope contrastive learning strategy for MI estimation. Finally, we provide a computational complexity analysis for the training process of the substructure encoder. An illustration of an integrated SMIP layer is given in Figure1.

3.1 Preliminaries

3.1.1 Input Graph. An undirected graph can be represented as a tuple $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with $\mathcal{V} = \{v_i\}_{i=\{1,\cdots,N\}}$ denoting the set of nodes, and $e_{ij} \in \mathcal{E}$ indicating the edge link between node v_i and v_j. Suppose that each node is attached with a D-dimensional feature vector which is initialized as the one-hot encoding of node types or node degrees, then the node feature matrix can be given by $\mathbf{X} \in \mathbb{R}^{N\times D} = \{x_1, \cdots, x_N\}$, which is full column rank. The adjacent matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ is a symmetric matrix which reveals edge connections. $\mathbf{A}_{i,j} = 1$ if there is an edge between node i and node j. It may also be arbitrary real numbers report both structural information and edge weights.

3.1.2 Node Encoder. we adopt l-layer GNNs as node encoder and employ a multi-layer node embedding concatenation as the output node embedding matrix. Assume that the output dimension of a node encoder is D', then the final output of a l-layer node encoder will be given by $\mathbf{H} \in \mathbb{R}^{N\times D'} = \{h_1, \cdots, h_N\}$, where $D' = lD$. For simplicity, we overlook the intermediate stage and use $\mathbf{H} = \Theta(\mathbf{A}, \mathbf{X})$ as the denotation of arbitrary l-layer node encoder function $\Theta(\cdot, \cdot)$ fed with an adjacent matrix $\mathbf{A}$ and input node feature matrix $\mathbf{X}$.

3.1.3 Unsupervised Graph Pooling. Given a set of graphs $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \cdots\}$ accompanied with node features $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots\}$, a node encoder $\Theta(\mathbf{A}, \mathbf{X}) \rightarrow \mathbf{H}$ takes the node feature matrix $\mathbf{X}$ and adjacent matrix $\mathbf{A}$ as input, and outputs a node embedding matrix $\mathbf{H} \in \mathbb{R}^{N\times D'}$. Then, unsupervised graph pooling method performs as a downsampling strategy to find a compressed representation of a graph as a single point in representation space with no predetermined graph label information. In this paper, we perform graph pooling as a substructure encoder and train this encoder via substructure-sensitive MI maximization objectives.

3.2 Substructure Encoder

Cluster-based pooling that captures the local substructures of a graph has received intensive study since the presentation of DiffPool [44]. Recently, cluster-based method which can usually be separated into node assign stage and substructure formation stage becomes a research branch of graph pooling and has achieved a large improvement [4, 26, 32, 45]. Here we attempt to reconstruct a substructure encoder based on commonly used cluster-based pooling paradigm and interpret it from a new perspective.

3.2.1 Node Assign Stage. Node assign stage aims to allocate N nodes in the original graph into N' coarse-grained substructures

where $N > N'$, thus forming a allocation relationship between nodes and substructures. It takes the node embedding matrix $\mathbf{H} \in \mathbb{R}^{N\times D'}$ as input and learns a node assign matrix $\mathbf{B} \in \mathbb{R}^{N\times N'}$ through an assign function $f(\cdot)$ with learnable parameters $\mathbf{W} \in \mathbb{R}^{D'\times N'}$. There are a number of options of assign functions, such as GNNs [44, 45], attention mechanism [26] or MLP [4]:

$$\mathbf{B} = f(\mathbf{H}) \tag{1}$$

3.2.2 Substructure Formation Stage. Substructure Formation stage aims to aggregate node features into substructures according to the allocation relationships obtained through node assign stage. It takes the node feature matrix $\mathbf{X}$ and the node assign matrix $\mathbf{B}$ as input, then generates a substructure embedding matrix $\mathbf{X'} \in \mathbb{R}^{N'\times D'}$ and a new adjacent matrix $\mathbf{A'} \in \mathbb{R}^{N'\times N'}$ for the coarse-grained graph. Substructures will be regarded as nodes in the coarse-grained graph:

$$\mathbf{X'} = \mathbf{B}^\top \mathbf{X}, \tag{2}$$

$$\mathbf{A'} = \mathbf{B}^\top \mathbf{A}\mathbf{B}. \tag{3}$$

3.2.3 Reconstruction-(i): Bridge Matrix. The trainable node assign matrix $\mathbf{B}$ is usually recognized as an assignment probability matrix, where each element $\mathbf{B}_{i,j} \in \mathbb{R}$ indicates the probability that node i is allocated to substructure j. In this paper, we interpret it from a new perspective and view it as a feature matrix, where each row $\mathbf{B}_{(\mathbf{i},\cdot)} \in \mathbb{R}^{N'}$ can be regarded as node feature with dimension N' in the original graph, and each column $\mathbf{B}_{(\cdot,\mathbf{j})} \in \mathbb{R}^{N}$ can be regarded as substructure feature with dimension N in the coarse-grained graph. We term it as a bridge matrix which can offer learnable feature connection between the original graph and the coarse-grained graph. In this sense, it converges both the original graph and the coarse-grained graph on a united feature matrix, thus achieving relative stability between information from the original graph and the coarsened graph.

3.2.4 Reconstruction-(ii): Hard Assign. It should be noted that the node assign stage is usually in a soft style, where nodes and substructures are mapped in an all-to-all manner with different assignment probabilities. However, less critical nodes with lower assignment probabilities are actually "fake" members introducing noise to the coarsened substructure. To address this, we reconstruct the soft bridge matrix $\mathbf{B}$ into a hard-assign matrix $\mathbf{S} \in \mathbb{R}^{N\times N'}$ through Gumbel-SoftMax [19], thus maintaining the more important "real" member in substuctures.

$$\mathbf{S}_{i,j} = \frac{\exp\left(\left(\log \mathbf{B}_{i,j} + \mathbf{g}_{i,j}\right)/\tau\right)}{\sum_{k=1}^{N'} \exp\left(\left(\log \mathbf{B}_{i,k} + \mathbf{g}_{i,k}\right)/\tau\right)} \tag{4}$$

where $\mathbf{g} = -\log\left(-\log\left(\mathbf{u}\right)\right)$ and $\mathbf{u} \sim \mathbf{Uniform}\left(0, 1\right)$. We set the softmax temperature parameter $\tau = 0.1$ to make the cluster assignment close to one-hot.

After being processed by bridge matrix and the hard assign operation, the reconstructed substructure encoder can be defined as follows:

Definition 3.1 (Substructure Encoder). *Based on the original cluster-based pooling paradigm, reconstruct the node assign stage with a hard sampling function. The reconstructed node assign stage takes node embedding matrix* $\mathbf{H} \in \mathbb{R}^{N\times D'}$ *as input and learns a soft bridge matrix* $B \in \mathbb{R}^{N\times N'}$*. Then process the bridge matrix through Gumbel-Softmax function to obtain a hard-assign matrix* $\mathbf{S} \in \mathbb{R}^{N\times N'}$*, s.t.* $\mathbf{S}_{ij} = 1$*, iff node i in the original graph is allocated to substructure j in the coarse-grained graph. Then substructure formation stage aggregates node features into substructures based on the hard-assign matrix* $\mathbf{S}$*:*

$$\mathbf{B} = f(\mathbf{H}), \tag{5}$$

$$\mathbf{S} = g(\mathbf{B}), \mathbf{S}_{ij} = \begin{cases} 1, & i \in substructure(j) \\ 0, & i \notin substructure(j) \end{cases} \tag{6}$$

$$\mathbf{X'} = \mathbf{S}^\top \mathbf{X}, \tag{7}$$

$$\mathbf{A'} = \mathbf{S}^\top \mathbf{A}\mathbf{S}. \tag{8}$$

For simplicity, we overlook the intermediate stage and use $\mathbf{X'}, \mathbf{A'} = \Phi(\mathbf{H})$ as the denotation of the reconstructed substructure encoder function $\Phi(\cdot)$ fed with an the node embedding matrix matrix $\mathbf{H}$.

3.3 Substructure-Sensitive MI

In this section, we propose two MI-basd substructure-sensitive objectives to train the substructure encoder.

3.3.1 Node-Substructure MI. Inspired by InfoMax principle [25], which states that optimizing MI between the input and output variables helps to preserve maximum information about the input, we propose node-substructure MI that maximizes the information preservation between the input and output (i.e., node embeddings and substructure embeddings) of the substructure encoder:

Definition 3.2 (Node-substructure MI). *Given node encoder* $\Theta(\mathbf{A}, \mathbf{X})$*, learn a proper set of parameters* $\mathbf{W}$ *for the substructure encoder* $\Phi(\mathbf{H})$*, maximizing the mutual information* $\mathcal{I}\left(\mathbf{H}; \mathbf{X'}\right)$ *between node embeddings acquired from the node embedding matrix* $\mathbf{H}$ *and a substructure embedding acquired from the substructure embedding matrix* $\mathbf{X'}$ *only when nodes are allocated to this substructure according to the hard-assign matrix* $\mathbf{S}$*.*

It seems that node-substructure MI and the previously proposed node-graph MI are different only in the comparison granularity, where the former is calculated between node and substructure embeddings while the latter between node and graph embeddings. However, there are other several important differences between them. First, node-substructure MI is substructure-sensitive which seeks to obtain substructure embeddings that reserve comprehensive information from nodes. While node-graph MI is node-sensitive which seeks to obtain node embeddings that capture the global information of the entire graph [40]. Second, node-substructure MI can capture high-order dependency between nodes based on global topology while node-graph MI only maintains neighborhood information based on local topology.

3.3.2 Node-Node MI. The node-substructure MI objective can be used to maximize the mutual information between node embeddings and substructure embeddings, thus transferring maximum node information into substructures. In this sense, node-substructure MI is benefit for the substructure formation stage which is actually a node information aggregator. However, according to the processing sequence of the reconstructed substructure encoder, the substructure formation stage follows after the node assign stage. As a result,

the effectiveness of node-substructure MI is based on the assumption that nodes are already allocated to proper substructures, which is actually not guaranteed due to the lack of specific constraint. Hence, we propose node-node MI to provide such a constraint:

Definition 3.3 (node-node MI). *Given node encoder* $\Theta(\mathbf{A}, \mathbf{X})$*, learn a proper set of parameters* $\mathbf{W}$ *for the substructure encoder* $\Phi(\mathbf{H})$*, maximizing the mutual information* $\mathcal{I}(\mathbf{H}; \mathbf{B})$ *between node embeddings acquired from the node embedding matrix* $\mathbf{H}$ *and node features acquired from the bridge matrix* $\mathbf{B}$ *only when these nodes are allocated to the same substructure according to the hard-assign matrix* $\mathbf{S}$.

Node-node MI is substructure-sensitive both theoretically and practically. Theoretically, node-node MI maximization can maximize the mutual dependency between node embeddings within the scope of the same substructure, which is equivalent to maximizing the shared information between nodes. This is consistent with the findings in [45] and [4] that nodes belong to the same substructure are strongly connected with similar features. Practically, instead of calculating MI between pairwise node embeddings both obtained from the node embedding matrix $\mathbf{H}$, the pairwise node-node MI is defined between node embeddings obtained from $\mathbf{H}$ and node features obtained from the learnable bridge matrix $\mathbf{B}$. This implementation is based on our novel interpretation of the trainable bridge matrix $\mathbf{B}$ that rows of $\mathbf{B}$ can be regarded as node features. Ideally, when two nodes are allocated to the same substructure, corresponding rows in bridge matrix $\mathbf{B}$ are totally the same, which also means that the two nodes have the same node features. This implementation helps to transfer the focus of optimization from the quality of individual node embeddings to the quality of node-substructure allocation, where the latter is substructure-sensitive.

3.3.3 Theoretical Analysis. According to InfoMax principle [25], the effectiveness of node-substructure MI $\mathcal{I}(\mathbf{H}; \mathbf{X}')$ is obvious. Hence, we only prove the effectiveness of node-node MI $\mathcal{I}(\mathbf{H}; \mathbf{B})$:

Proof. According to Equation(7), Mutual information $\mathcal{I}(\mathbf{H}; \mathbf{X}')$ can be denoted as follows:

$$\mathcal{I}\left(\mathbf{H}; \mathbf{X}'\right) = \mathcal{I}\left(\mathbf{H}; \mathbf{S}^\top \mathbf{X}\right) = H(\mathbf{H}) - H(\mathbf{H}|\mathbf{S}^\top \mathbf{X}) \tag{9}$$

Since $\mathbf{X}$ is a non-zero constant matrix with full column rank, then we have:

$$H(\mathbf{H}|\mathbf{S}^\top \mathbf{X}) = H(\mathbf{H}|\mathbf{S}^\top) = H(\mathbf{H}|\mathbf{S}) \tag{10}$$

As a result, we get:

$$\mathcal{I}\left(\mathbf{H}; \mathbf{X}'\right) = H(\mathbf{H}) - H(\mathbf{H}|\mathbf{S}) = \mathcal{I}(\mathbf{H}; \mathbf{S}) \tag{11}$$

This proves that maximizing $\mathcal{I}(\mathbf{H}; \mathbf{X}')$ is equivalent to maximizing $\mathcal{I}(\mathbf{H}; \mathbf{S})$. In our feedforward network structure, $\mathbf{H} \to \mathbf{B} \to \mathbf{S}$ is a Markov chain. According to Data Processing Inequality [9], $\mathcal{I}(\mathbf{H}; \mathbf{B}) \geq \mathcal{I}(\mathbf{H}; \mathbf{S})$, which means that the amount of information maintained from $\mathbf{H}$ decreases during the feedforward process. However, we hope that each feedfordward layer can preserve as much information about $\mathbf{H}$ as possible, and we can maximize $\mathcal{I}(\mathbf{H}; \mathbf{B})$ to achieve this. As a result, the node-node MI $\mathcal{I}(\mathbf{H}; \mathbf{B})$ is effective. □

3.4 Local-Scope Contrastive MI Estimation

Given a input graph dataset $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \cdots\}$, our training objective is to maximize the average node-substructure MI $\mathcal{I}(\mathbf{H}; \mathbf{X}')$ and node-node MI $\mathcal{I}(\mathbf{H}; \mathbf{B})$ of all graphs to optimize the parameter matrix $\mathbf{W}$ for substructure encoder:

$$(\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2) = \arg\max_{\mathbf{W}_1, \mathbf{W}_2} \frac{1}{|\mathcal{G}|} \sum_{i=1}^{|\mathcal{G}|} \left(\alpha \hat{\mathcal{I}}_{\mathbf{W}_1}(\mathbf{H}_i; \mathbf{X'}_i) + \beta \hat{\mathcal{I}}_{\mathbf{W}_2}(\mathbf{H}_i; \mathbf{B}_i) \right) \tag{12}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are substructure encoder parameters for the node-substructure MI and node-node MI objectives, respectively. α and β are hyper parameters.

We combine MINE [3] with infoNCE [38] to estimate MI through Jensen-Shannon divergence (JSD) [29] based on positive and negative sample pairs in a self-supervised contrastive learning method. Take the estimation of node-substructure MI $\mathcal{I}(\mathbf{H}_i; \mathbf{X'}_i)$ for graph $\mathcal{G}_i$ as an example:

$$\hat{\mathcal{I}}_{\mathbf{W}_1, \Omega}(\mathbf{H}_i; \mathbf{X'}_i) = \frac{1}{\mathbf{N}_{pos} + \mathbf{N}_{neg}} \left(\sum_{j=1}^{\mathbf{N}_{pos}} \mathbb{E}_{(\mathbf{X}_i, \mathbf{A}_i)} [-sp(-\mathcal{D}_\Omega(h, x')_j)] - \sum_{k=1}^{\mathbf{N}_{neg}} \mathbb{E}_{(\mathbf{X}_i, \mathbf{A}_i)} [sp(\mathcal{D}_\Omega(\tilde{h}, \tilde{x'})_k)] \right) \tag{13}$$

where $\mathcal{D}_\Omega$ with parameters Ω is a contrastive statistics network takes (node, substructure) pairs as input to determine whether the node belongs to the substructure. $sp(z) = \log(1 + e^z)$ is the softplus function. $\mathbf{N}_{pos}$ and $\mathbf{N}_{neg}$ correspond to the number of positive and negative (node, substructure) pairs, respectively.

According to the hard-assign matrix $\mathbf{S}$, we generate negative (node, substructure) pairs by combining nodes with those substructures that they are not allocated to. Similarly, we generate negative (node, node) samples by pairing nodes that are not allocated to the same substructure. In this way, no graph argumentation is needed, all negative pairs are sampled within the local scope of the input graph, we call it a local-scope contrastive learning strategy.

This straightforward but effective local-scope contrastive learning strategy has three advantages: (1) It is free from extra computation of graph augmentation. This promotes the accuracy of self-concerned graph embedding and makes it more potent in capturing intrinsic features of the input graph; (2) It provides sufficient negative samples for contrastive learning, making $\mathbb{P}_{\mathbf{B}}$ and $\mathbb{P}_{\mathbf{X}'}$ to be more competitive to approximate the true distribution $\mathbb{P}_{\mathbf{H}}$, thus maintaining the feature consistency during pooling; and (3) It strictly avoids overlapping between positive and negative pairs by the constraint of the hard-assign matrix $\mathbf{S}$, thus eliminating noisy pairs that influence the discriminate of the statistics network.

3.5 Computational Complexity Analysis

Here, we analyze the time complexity of SMIP to train a substructure encoder for graph $\mathcal{G}$. Section 3.4 mentions that MI estimation of SMIP relies on abundant negative samples far more than positive samples. Accordingly, the time cost for MI estimation mainly depends on the discrimination of negative (node, substructure) and (node, node) pairs. As we have discussed in Section 3.4, SMIP uses a local-scope contrastive fashion to generate all possible negative samples, so that negative (node, substructure) and (node, node) pairs for graph $\mathcal{G}$ are of order of magnitude of NN' and N^2, respectively. Based on node-node and node-substructure MI, the total time complexity for training cluster assignment encoder is $O(N^2)$.

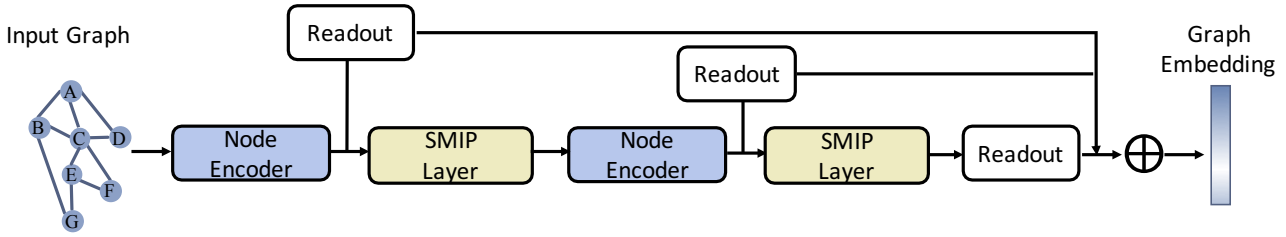

Figure 2: The hierarchical graph representation learning architecture based on SMIP.

Specifically, when properly reduce the number of negative samples, the time complexity could even be decreased to $O(N)$.

4 EXPERIMENTS

In this section, we attempt to answer the following questions:

- **Q1:** How does SMIP perform in graph classification compared with unsupervised and supervised competitors? (Section 4.2)
- **Q2:** How does the substructure-sensitive MI objective of SMIP help to improve other cluster-based pooling methods? (Section 4.3)
- **Q3:** Does SMIP lead to better clustering interpretability than other cluster-based pooling methods? (Section 4.4)
- **Q4**: How do the hyperparameters influence the performance of SMIP? (Section 4.5)

4.1 Experimental setup

4.1.1 Datasets. We evaluate our proposed SMIP on six real-world benchmark datasets, including two social network datasets: IMDB-B and IMDB-M [43], three molecules datasets: MUTAG, PTC [5] and HIV [17], and one protein graph dataset: PROTEINS [6]. The largest dataset, HIV, has 41127 graphs. Complete statistics and properties of them are summarized in Table 1.

Bioinformatic datasets. MUTAG is a dataset of 188 mutagen graphs with 7 discrete node labels. Graph label denotes whether a molecule is a mutagen or non-mutagen. PTC is a dataset of 344 chemical compounds with 19 discrete node labels. Graph label denotes whether it can cause cancerous changes for rats. PROTEINS is a dataset of 1113 protein graphs where nodes are secondary structure elements and two nodes are connected by an edge if they are neighbors in the amino-acid sequence or in 3D space. It has 3 discrete node labels, and graph labels denotes whether or not a protein is a non-enzyme. HIV is a dataset of 41127 molecular graphs.

Social network datasets. IMDB-B is a movie collaboration dataset of 1000 ego-networks where unlabeled nodes are actors/actresses and there is an edge between them if they have a cooperation in the same movie. Graph labels denotes whether the genre for each graph is Action or Romance. IMDB-M is multi-class version of IMDB-B with 1500 movie ego-networks derived from Comedy, Romance and Sci-Fi genres.

For bioinformatic datasets (i.e., MUTAG, PTC, PROTEINS, and HIV), nodes have categorical input features. However, for social network datasets (i.e., IMDB-B and IMDB-M), no node features are attached. We use the one-hot encoding of degrees as the input node feature.

Table 1: Statistics and properties of six benchmark datasets used in experiments.

	Dataset					
	IMDB-B	IMDB-M	MUTAG	PTC	PROTEINS	HIV
#Graphs	1000	1500	188	344	1113	41127
#Classes	2	3	2	2	2	2
#Max Nodes	136	89	28	109	620	222
#Min Nodes	12	7	10	2	4	2
#Avg Nodes	20	13	18	14	39	26

4.1.2 Baselines. SMIP achieves unsupervised graph pooling via maximizing node-node MI and node-substructure MI. We term the variants with node-node, node-substructure, and combined objectives as SMIP-NN, SMIP-NS, and SMIP, respectively. All of the three variants are included in the scope of comparison with two kinds of state-of-the-art GNN-based baselines:

- Supervised GNN-based Pooling Methods: The base method uses the concatenation of GCN-based node-level representations as the final graph representations. MeanPool is an element-wise mean node representation aggregator. GIN [42] is a graph isomorphic network. SortPooling [46], AttPool [18], VIPool, and SAGP [23] are selection-based pooling methods that only select the top k important nodes from the original graph to form a new coarsened one. DiffPool [44] is a cluster-based pooling method. ASAP [32] combines both selection and clustering strategy by selecting top k clusters to form a pooled graph.
- Unsupervised GNN-based Pooling Methods: UGRAPHEMB [2] achieves unsupervised pooling via graph-graph proximity. MinCutPool [4], StructPool [45], and OTCOARSENING [27] are cluster-based unsupervised methods via mincut loss, conditional random fields, and optimal transport. InfoGraph [36], MVGRL [15], LGR [41], CuCo [8] and UHGR [10] are five MI-based unsupervised pooling methods.

4.1.3 Model Structure and Parameter Settings. We conduct a hierarchical graph representation learning model by alternately stacking node encoder and substructure encoder as shown in Figure2. For

Table 2: Graph classification accuracy of a fixed 10-fold cross-validation in percent on six benchmark datasets. SMIP-NN, SMIP-NS, and SMIP represent using node-node-only, node-substructure-only and combined objectives, respectively. The best performance per dataset is boldfaced.

Method		Dataset					
		IMDB-B	IMDB-M	MUTAG	PTC	PROTEINS	HIV
Supervised Methods	Base	74.67±4.43	48.49±3.34	72.89±2.53	58.82±3.25	72.16±5.69	71.25±1.68
	MeanPool	72.24±4.34	48.10±2.86	85.89±5.66	70.58±3.94	79.81±2.91	63.69±0.73
	GIN [42]	72.78±0.86	48.13±1.36	81.39±1.53	-	71.76±1.66	-
	SortPooling [46]	66.79±3.57	46.99±5.01	83.33±5.55	56.47±7.53	74.05±4.40	69.61±3.02
	AttPool [18]	73.80±3.21	51.80±3.43	93.33±6.47	74.11±4.70	79.81±3.78	73.50±1.71
	VIPool [24]	74.86±2.10	52.63±2.71	92.24±5.61	71.18±4.82	78.20±3.91	71.93±3.52
	SAGP [23]	72.80±1.77	49.21±2.22	74.44±3.65	62.05±9.79	71.87±2.73	73.31±2.20
	DiffPool [44]	74.39±3.57	50.33±2.80	92.22±5.66	70.88±6.76	74.23±3.14	74.12±0.53
	ASAP [32]	74.30±2.83	49.02±3.17	79.82±7.64	55.30±6.18	71.43±4.30	70.15±3.62
Unsupervised Methods	UGRAPHEMB [2]	73.26±2.19	51.04±3.03	82.35±4.89	63.89±3.65	73.21±4.25	-
	MinCutPool [4]	74.30±4.05	48.06±4.05	88.33±7.22	70.87±4.04	79.01±4.35	72.74±2.71
	StructPool [45]	73.80±5.36	50.86±2.93	88.88±6.57	70.58±6.41	79.45±3.75	73.10±3.12
	OTCOARSENING [27]	74.62±4.90	50.91±3.31	85.64±6.20	-	74.90±3.91	-
	InfoGraph [36]	73.03±0.94	49.70±0.57	89.50±1.17	61.75±1.33	71.69±4.37	70.21±2.13
	MVGRL [15]	74.27±0.73	51.23±0.56	89.71±1.10	62.52±1.73	-	-
	LGR [41]	73.33±0.52	50.54±0.31	91.80±0.57	65.86±1.34	-	-
	CuCo [8]	-	-	90.52±0.90	57.11±3.06	75.96±0.53	-
	UHGR [10]	68.37±4.06	44.91±2.26	86.73±7.52	69.42±5.71	75.93±1.32	68.96±2.13
	SMIP-NN (Ours)	71.70±2.83	49.53±3.09	96.10±4.33	79.04±3.34	80.26±3.53	72.29±2.41
	SMIP-NS (Ours)	75.00±4.35	53.14±2.08	96.11±3.56	79.12±2.06	81.44±3.67	72.54±2.36
	SMIP (Ours)	**76.50±3.44**	**53.33±1.95**	**97.78±3.68**	**80.29±2.95**	**81.80±3.29**	**74.46±1.82**

each node encoder, we resort to a standard two-layer Graph Convolution Network (GCN) [20] with latent dimensions set as 64 and output dimension set as 32. Batch normalization is applied after each GCN layer. The final output node embedding of each node encoder is the concatenation of the intermediate hidden representations. Hence the node embedding dimension will be 96. Moreover, a readout operation will be used to generate a single-scale graph embedding attached with each node encoder. What follows a node encoder is a SMIP layer that takes node embeddings as input. As shown in Table 1, the graph size distribution of a given dataset may be highly unbalanced. Thus the number of substructures for the first SMIP layer is set as 50% of the average rather than the maximum graph size of the dataset to guarantee that a large proportion of graphs will be coarsened rather than expanded in the current dataset. For the rest SMIP layers, coarsen ratios are all set as 0.5. The new coarsened graph will then be used as the input to another node encoder. A multi-scale concatenation among the output of all node encoders will be operated to generate the final graph embedding. Hyperparameters α and β are set as $(\alpha = 1, \beta = 0)$ for SMIP-NS, $(\alpha = 0, \beta = 1)$ for SMIP-NN, and $(\alpha = 1, \beta = 1)$ for SMIP.

All experiments are implemented in Pytorch and conducted on one Geforce RTX 3090 GPU. The batch size is set to 50 for all the datasets. The model is trained for fixed epochs (100 on MUTAG, PTC, IMDB-M and HIV; 150 on IMDB-B; 80 on PROTEINS) using Adam optimizer with an initial learning rate of 0.0001 for MUTAG, PTC, IMDB-B, HIV and PROTEINS, and 0.0005 for IMDB-M. We use an early stopping strategy with a patience of 20 epochs.

4.1.4 Evaluation metrics. For the graph classification task, we feed the learned training and validation graph embeddings into a logistic regression classifier trained for 20000 epochs with early stopping applied when the validation loss does not decrease for 20 consecutive epochs. We evaluate model performance with a same-split 10-fold cross validation as SortPooling [46].

4.2 Graph Classification

Table 2 reports the fixed 10-fold cross-validation results of the graph classification task on six benchmark datasets. The best performance for each dataset is highlighted in bold. We find that our SMIP is superior to all unsupervised competitors, and even outperforms nine supervised methods under the circumstance of avoiding extensive ground-truth labels. Detailed analysis for the experimental results are as follows.

Compared with unsupervised competitors, we can observe that our proposed SMIP achieves the best classification accuracy across all six datasets with an average improvement of 6.83%. Interestingly, SMIP outperforms UGRAPHEMB, which utilizes existing graph proximity metric for graph embedding. This indicates that intrinsic features is much more important than inter-graph information for a high-quality graph embedding. Besides, it can be noticed that SMIP exceeds all cluster-based unsupervised pooling methods, i.e., MinCutPool, StructPool and OTCOARSING. This result, in one aspect, reflecting that mutual information has natural advantage for the training of substructure encoder compared with others. In another aspect, it demonstrates that hierarchical model structure with

Table 3: Generalization study results of three cluster-based pooling methods with substructure-sensitive MI-based objectives. Average performance improvements are also reported.

Dataset	MUTAG	IMDB-M	PTC	Avg.Imp
DiffPool	92.22±5.66	50.33±2.80	70.88±6.76	-
DiffPool+	**93.89±4.61**	**53.73±3.36**	**74.71±5.91**	**2.97%**
StructPool	88.88±6.57	50.86±2.93	70.58±6.41	-
StructPool+	**91.67±7.14**	**51.27±3.01**	**71.77±4.40**	**1.46%**
MinCutPool	88.33±7.22	48.06±4.05	70.87±4.04	-
MinCutPool+	**94.44±4.30**	**51.07±2.86**	**72.65±3.23**	**3.63%**

several pooling layers can improve the graph embedding quality, while MinCutPool, StructPool and OTCOARSING can only capture graph substructure with one level. More notable is that the performance of other MI-based pooling methods are all inferior to SMIP, which highlights the benefit of our substructure-sensitive MI objective. This result also proves that introduces noise information from other graphs into the final graph representations during contrastive MI estimation is directly harmful to the precision of graph classification tasks. For dataset MUTAG and PTC, which are polymerized high molecules with typical hierarchical graphs with multi-level substructures, we observe that SMIP incredibly outperforms all unsupervised competitors with an average gain of 9.62% and 15.04%, respectively. This significant improvement demonstrates that SMIP fits the characteristics of MUTAG and PTC well, where the substructure-sensitive MI objective in SMIP can extract substructure properly.

Compared with supervised GNN-base methods, we find that the classification results of our SMIP performs better than nine competitors. The result gives evidence that the implicit information lies in graphs can be as powerful as that provided by labels to some extent. Thus, it could be claimed that unsupervised pooling methods are capable of sustaining the performance of downstream graph-level tasks as long as sufficient information is extracted from the original graph.

To better understand the effects of node-substructure MI and node-node MI, we perform ablation studies, where SMIP-NN, SMIP-NS, and SMIP represent using node-node, node-substructure, and combined objectives, respectively. We observe that the SMIP outperforms the other two ablated variants. This result provides positive feedback to our intuition that node-node MI and node-substructure MI are responsible for different stages of substructure encoder. Jointly optimizing them provides more comprehensive constraints for the training of substructure encoder. Besides, it is notable that the performance of SMIP-NN is slightly inferior to SMIP-NS. This phenomenon demonstrates that transforming maximum information of nodes into substructures plays a decisive role for a high-quality graph embedding. Meanwhile, substructure formation stage is tolerant to wrong node allocation.

4.3 Generalization Study

According to the Section 3.2, any cluster-based graph pooling method following the substructure encoder can be jointly optimized by maximizing both task-specific supervised loss and an auxiliary substructure-sensitive MI loss. In this section, we validate the generalization ability of the proposed substructure-sensitive objective and conclude that it can be successfully applied to no matter supervised or unsupervised cluster-based graph pooling models delivering a better version with improved performance. We adopt a baseline model architecture implemented with official codes. This kind of variant is optimized with their original supervised loss, and we term this variant as DiffPool, StructPool and MinCutPool, respectively. For comparison, substructure-sensitive MI-based loss is added to the original supervised loss, and we term this variant as DiffPool+, StructPool+, and MinCutPool+, respectively. This kind of variant is used to investigate the contribution of our substructure-sensitive MI objective.

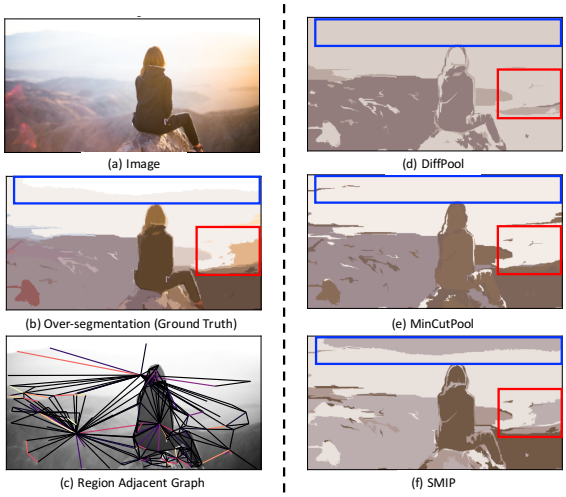

Figure 3: Graph-based Image segmentation results. Figure (a) to (c) illustrates the preprocessing steps, where figure (b) is considered as the ground truth. Figure (d) to (f) shows the image segmentation results of three cluster-based graph pooling methods, in which SMIP yields a more precise segmentation sensitive to similar color patches. Blue and red rectangles mark the detailed differences.

Table 3 provides the comparison results between variants with and without substructure-sensitive MI objective, and we report the average improvement rate on three benchmark datasets. As can be observed, DiffPool+, StructPool+ and MinCutPool+ outperform the original variants on all three datasets with noteworthy performance improvement for an average of 2.97%, 1.46%, and 3.63%, respectively. This reflects the guiding role of the intrinsic characteristic of graph data for the formation of graph embeddings. The results verify our belief that the proposed substructure-sensitive MI objective can serve as a regularization term in cluster-based pooling methods to smooth graph embeddings.

4.4 Interpretability Study

To demonstrate the node assignment interpretability and verify the effectiveness of node-node MI objective in SMIP, we employ image segmentation as a typical case for interpretability study. Graph-based image segmentation is implemented as a node clustering task

Table 4: Parameter test results w.r.t polling depth.

Dataset	Loss	Pooling=1	Pooling=2	Pooling=3
IMDB-B	NN	71.10±2.83	70.40±3.56	67.99±1.84
	NS	72.19±3.99	74.90±3.53	75.00±4.36
	NN+NS	73.50±4.45	74.30±3.76	**76.50±3.44**
MUTAG	NN	94.44±4.30	96.11±4.33	96.10±2.54
	NS	92.77±4.99	96.10±3.55	94.99±3.88
	NN+NS	93.88±5.80	96.21±3.34	**97.78±3.68**
PROTEINS	NN	79.09±3.21	81.26±3.53	80.99±3.97
	NS	79.90±2.34	80.72±3.39	81.44±3.67
	NN+NS	79.09±3.62	80.90±3.50	**81.80±3.29**

on a region adjacent graph [37], where each node corresponds to a region with the same color generated by an over-segmentation procedure [12], and an edge connects the adjacent regions with edge weights to show color similarity. In this task, nodes with similar colours are encouraged to cluster together. The results for three cluster-based graph pooling methods are given in Figure 3 with the desired number of clusters $K = 4$. According to the ground truth (Figure3 (b)) , results for DiffPool, MinCutPool and our proposed SMIP indicate that SMIP yields a more precise segmentation. Note that the segmentation through SMIP preserves detailed regions, such as the small colour patch of the mountains (shown in red rectangles). Meanwhile, SMIP creates more explicit clustering bounds among approximate colour regions (shown in blue rectangles). However, DiffPool and MinCutPool are less sensitive to adjacent regions with similar colours. As marked in the blue and red rectangles, they split different regions into the same cluster.

4.5 Parameter Study

4.5.1 Pooling depth. In this part, we adjust the number of substructure encoders in the model architecture to investigate the influence of pooling depth on classification accuracy. Other involved hyperparameters remain unchanged with a fixed pooling ratio $r = 0.5$. The results are listed in Table 4.

It can be observed that all three datasets achieve the best result when there are three pooling encoders in the model. Generally, graph classification performance improves with pooling depth increasing in a proper pooling depth range. This indicates that enabling hierarchical learning in cluster-based pooling methods is reasonable, which is crucial for capturing multi-level substructural information of graphs. Meanwhile, we observe that the node-substructure MI objective can alleviate the performance deterioration brought from the node-node-only objective to some extent. This demonstrates that the aggregation of node feature information plays a dominating role in cluster-based graph pooling because it imposes a direct constraint on the generation of substructure embeddings, and therefore should always take the lead in the pooling method. Based on the above observations, appropriate pooling depth with constraints of both node-node and node-substructure MI could be a good practice.

4.5.2 Pooling ratios. Apart from the pooling depth, what also influences the coarsening power is the pooling ratio, which determines

Table 5: Parameter test result w.r.t different pooling ratios.

	MUTAG	IMDB-M	PTC
r=0.1	94.99±5.24	51.93±1.80	78.82±3.90
r=0.3	97.22±2.77	50.93±2.49	79.70±4.45
r=0.5	**97.78±3.68**	**53.33±1.95**	**80.29±2.95**
r=0.7	96.10±3.55	52.73±2.92	78.82±2.20
r=0.9	94.44±3.51	52.86±3.40	78.23±3.52

to what extent the original graph will be coarsened in each pooling layer. To further demonstrate how the pooling ratio affects the classification performance and provide a relatively fair comparison, we conduct experiments to evaluate different pooling ratios while keeping other experimental settings unchanged with a fixed pooling depth $d = 3$. The results are reported in Table 5.

As can be observed, all three datasets achieve the best performance with a pooling ratio $r = 0.5$ and the accuracy drops when the pooling ratio is relatively large or small. In addition, the model can obtain competitive performance when the pooling ratio is set within a reasonable range, e.g., $r \in [0.3, 0.7]$. We attribute this discovery to the fact that too large pooling ratio makes graphs compressed in high density. Therefore, clusters are merged losing certain elaborate information. On the other hand, too small pooling ratio makes small graphs expand rather than coarsen, and we assume the further information brought by the expansion may be noise for the following pooling, although it may exhibit good performance on giant graphs.

5 CONCLUSION

To overcome the dilemma of unpractical supervision and the drawbacks of existing unsupervised MI-based graph pooling methods, we propose a unsupervised hierarchical graph pooling method based on substructure-sensitive MI maximization, called SMIP. We consider graph pooling as a node clustering problem and formalize a substructure encoder. We train the substructure encoder through the optimization of node-node and node-substructure MI, making strongly connected nodes with similar features allocated to the same substructure and inheriting individual node features into substructures as much as possible. The local-scope contrastive learning makes SMIP more potent in capturing intrinsic features of the input graph. Graph classification results on six benchmark datasets show that SMIP outperforms state-of-the-art unsupervised methods and is even superior to nine supervised methods. The generalization study presents significant applicability of the proposed substructure-sensitive MI-based objectives to other cluster-based pooling methods.

6 ACKNOWLEDGMENTS

This document is supported by National Natural Science Foundation of China (No. 62025208, No. 61932001 and No. 62002371), State Administration of Science Technology and Industry for National Defense Foundation (No. WDZC20205250104) and the National University of Defense Technology Foundation (No. ZK21-17).

REFERENCES

[1] Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. 2018. Sub2vec: Feature learning for subgraphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 170–182.

[2] Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. 2019. Unsupervised inductive graph-level representation learning via graph-graph proximity. *arXiv preprint arXiv:1904.01098* (2019).

[3] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. 2018. Mutual information neural estimation. In *ICML*. PMLR, 531–540.

[4] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. 2020. Spectral clustering with graph neural networks for graph pooling. In *ICML*. PMLR, 874–883.

[5] Karsten M Borgwardt and Hans-Peter Kriegel. 2005. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*. IEEE, 8–pp.

[6] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21, suppl_1 (2005), i47–i56.

[7] Pengyu Cheng, Weituo Hao, Shuyang Dai, Jiachang Liu, Zhe Gan, and Lawrence Carin. 2020. Club: A contrastive log-ratio upper bound of mutual information. In *International conference on machine learning*. PMLR, 1779–1788.

[8] Guanyi Chu, Xiao Wang, Chuan Shi, and Xunqiang Jiang. 2021. CuCo: Graph Representation with Curriculum Contrastive Learning. In *IJCAI*. 2300–2306.

[9] Thomas M Cover. 1999. *Elements of information theory*. John Wiley & Sons.

[10] Fei Ding, Xiaohong Zhang, Justin Sybrandt, and Ilya Safro. 2020. Unsupervised Hierarchical Graph Representation Learning by Mutual Information Maximization. *arXiv preprint arXiv:2003.08420* (2020).

[11] Roger C Entringer, Douglas E Jackson, and DA Snyder. 1976. Distance in graphs. *Czechoslovak Mathematical Journal* 26, 2 (1976), 283–296.

[12] Pedro F Felzenszwalb and Daniel P Huttenlocher. 2004. Efficient graph-based image segmentation. *International journal of computer vision* 59, 2 (2004), 167–181.

[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[14] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 297–304.

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

[16] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670* (2018).

[17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.

[18] Jingjia Huang, Zhangheng Li, Nannan Li, Shan Liu, and Ge Li. 2019. Attpool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism. In *ICCV*. 6480–6489.

[19] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[20] Thomas Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv: Learning* (2016).

[21] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[22] Risi Kondor and Horace Pan. 2016. The multiscale laplacian graph kernel. *arXiv preprint arXiv:1603.06186* (2016).

[23] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. *arXiv preprint arXiv:1904.08082* (2019).

[24] Maosen Li, Siheng Chen, Ya Zhang, and Ivor W Tsang. 2020. Graph Cross Networks with Vertex Infomax Pooling. *arXiv preprint arXiv:2010.01804* (2020).

[25] Ralph Linsker. 1988. Self-organization in a perceptual network. *Computer* 21, 3 (1988), 105–117.

[26] Ning Liu, Songlei Jian, Dongsheng Li, Yiming Zhang, Zhiquan Lai, and Hongzuo Xu. 2021. Hierarchical adaptive pooling by capturing high-order dependency for graph representation learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[27] Tengfei Ma and Jie Chen. 2019. Unsupervised learning of graph hierarchical abstractions with differentiable coarsening and optimal transport. *arXiv preprint arXiv:1912.11176* (2019).

[28] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005* (2017).

[29] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-gan: Training generative neural samplers using variational divergence minimization. In *NIPS*. 271–279.

[30] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*. 259–270.

[31] Nataša Pržulj. 2007. Biological network comparison using graphlet degree distribution. *Bioinformatics* 23, 2 (2007), e177–e183.

[32] Ekagra Ranjan, Soumya Sanyal, and Partha P Talukdar. 2020. ASAP: Adaptive Structure Aware Pooling for Learning Hierarchical Graph Representations.. In *AAAI*. 5470–5477.

[33] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*. 593–607.

[34] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12, 9 (2011).

[35] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. 2009. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*. PMLR, 488–495.

[36] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2019. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000* (2019).

[37] Alain Trémeau and Philippe Colantoni. 2000. Regions adjacency graph applied to color image segmentation. *IEEE T IMAGE PROCESS* 9, 4 (2000), 735–744.

[38] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv e-prints* (2018), arXiv–1807.

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

[40] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax.. In *ICLR*.

[41] Chenguang Wang and Ziwen Liu. 2021. Learning graph representation by aggregating subgraphs via mutual information maximization. *arXiv preprint arXiv:2103.13125* (2021).

[42] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[43] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1365–1374.

[44] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. *arXiv preprint arXiv:1806.08804* (2018).

[45] Hao Yuan and Shuiwang Ji. 2019. StructPool: Structured graph pooling via conditional random fields. In *ICLR*.

[46] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An End-to-End Deep Learning Architecture for Graph Classification.. In *AAAI*, Vol. 18. 4438–4445.