Graph Reconfigurable Pooling for Graph Representation Learning

Xiaolin Li, Qikui Xu, Zhenyu Xu, Hongyan Zhang, and Li Xu

***Abstract*—In recent years, graph neural networks have been widely used for tasks such as graph classification, link prediction, and node classification, and have achieved excellent results. In order to apply GNNs to graph classification tasks, recent works generate graph-level representations using node representations through a hierarchical pooling approach. Existing graph pooling methods such as DiffPool and EigenPool encourage adjacent nodes to be assigned to the same cluster, making the node assignment process similar to the graph partitioning process that ignores the role of nodes or some substructures (e.g., amino acids) in the process of composing a graph (e.g., proteins). In this article, we propose a new pooling operator RecPool to capture the role played by nodes in the process of composing a graph. Specifically, we probabilistically model the feature distribution of the coarsened graph, construct the feature distribution of each cluster, resample the features of the coarsened graph into the original nodes according to the soft assignment matrix, reconstruct the original graph, and optimize the soft assignment matrix to divide the nodes that play the same role in the reconstruction process into the same cluster. The excellent performance of Recpool is demonstrated through experiments on four public benchmark dataset.**

***Index Terms*—Graph classification, graph pooling, graph neural networks, hierarchical graph representation learning, graph reconfigurable.**

I. Introduction

In RECENT years, the study of extending neural networks to graph-structured data has gradually become a new trend in learning on graphs. Research on this topic is often referred to as Graph neural networks (GNNs) [1], which efficiently obtain a representation of nodes by aggregating network structures and features on top of the underlying computational graph. In this way, the feature information of nodes is propagated through the network topology, and nodes generate embeddings by aggregating the information obtained, which are then used for tasks like node and graph classification [2], [3], [4], [5], link prediction [6], and recommendation [7]. Graph neural networks have achieved excellent results in a variety of graph-based tasks. In this article, we focus on the graph classification task to investigate the process of learning graph-level representations.

Graph classification is a fundamental task on graph data that aims to predict the classification labels of the whole graph. One usually uses graph neural networks to obtain feature as well as structural information about the graph, extract the graph representation, and thus predict the label of a given graph. However, existing GNN methods generate good node representations and then globally aggregate the node representations into a graph representation, which is inherently flat, and its equivalent treatment of all nodes fails to capture the substructure in the graph. For networks like biological networks, which are composed of various functional groups as well as chemical bonds, the use of flattening methods can have a greater impact [8], [9]. This local structure is not captured in the process of global pooling. Therefore, in order to make the pooling method more interpretable and to ensure that it satisfies our intuition of the pooling process, it is necessary to generate graph representations that preserve both local and global structure, and therefore, we need a hierarchical pooling process similar to that of conventional convolutional neural (CNN) networks [10].

There are many recent works that study the pooling process of graph neural networks [2], [3], [11]. These methods reduce the original graph to a coarsened graph by partitioning the nodes in the graph into clusters, treating a cluster as a supernode, and using the feature information of the nodes in the original graph to generate the features of the supernodes. However, in performing pooling and clustering, we find that most approaches follow an underlying rule that neighboring nodes should be encouraged to be divided in a cluster, which seems intuitive, but for biological or chemical networks, a graph is composed of many different combinations of nodes, and two nodes are connected not necessarily because they are the same node, but react because they are different nodes.

For example, in Fig. 1, each endpoint (i.e., node) of compound $Ethyl2-iodooxy-1,3-thiazole-5-carboxylate$ represents an atom, and the edges represent the chemical bonds between atoms. It can be observed that the atoms that are connected (i.e., bonded) with each other in the compound do not always belong to the same class (i.e., with the same color). In other words, atoms that fall into the same class are not necessarily connected with each other. Similarly, in Fig. 2, it can be observed that similar proteins (i.e., connected by black

Manuscript received 4 July 2022; revised 7 January 2023; accepted 13 April 2023. Date of publication 21 April 2023; date of current version 15 March 2024. This work was supported by National Natural Science Foundation of China under Grants U1905211, 61771140, and 62171132. *(Corresponding author: Li Xu.)*

Xiaolin Li, Zhenyu Xu, Hongyan Zhang, and Li Xu are with the College of Computer and Cyber Security and Fujian Provincial Key Lab of Network Security and Cryptology, Fujian Normal University, Fuzhou 350007, China (e-mail: zimuxiaolin@163.com; 1246233565@qq.com; 154366449@qq.com; xuli@fjnu.edu.cn).

Qikui Xu is with the School of Life Sciences, Fudan University, Shanghai 310024, China (e-mail: xuqikui@westlake.edu.cn).

Digital Object Identifier 10.1109/TETC.2023.3268098

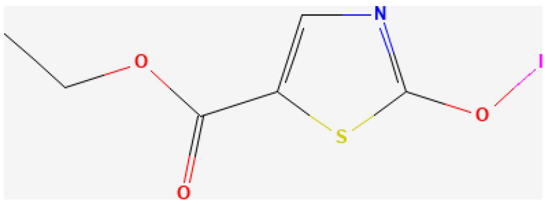


Fig. 1. Chemical structure of Ethyl 2-iodooxy-1,3-thiazole-5-carboxylate, in which atoms with the same color (black/red/blue/yellow) belong to the same class, and bonded atoms are connected with lines. Please note that the carbon atoms are not specifically labelled with 'C' due to the depiction convention of organic compounds.

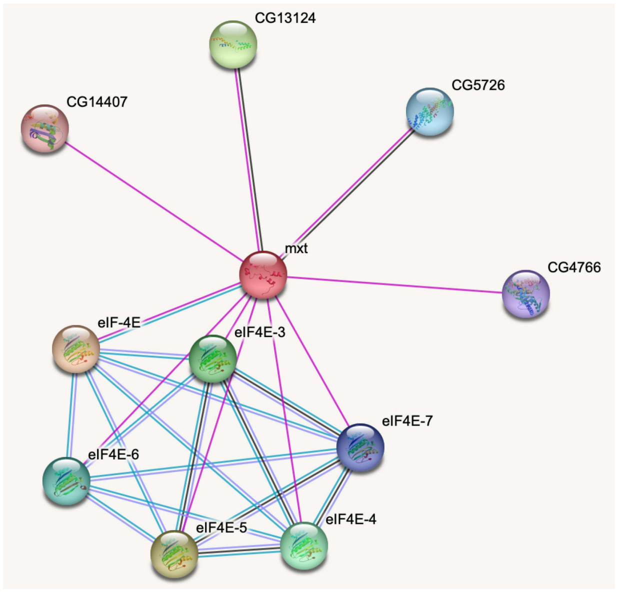


Fig. 2. Protein interaction networks, with purple edge representing experimentally proven interactions, blue edge representing interactions proven in supporting databases, green edge representing gene similarity, and black edge representing genes co-expressed in the same or different species. In essence, proteins connected by purple and blue edges are considered interacting with each other, and those connected by green and black edges are considered similar.

and green edges) do not necessarily interact with each other (i.e., connected by purple and blue edges). Therefore, it can be implied from the two examples that the traditional idea of encouraging inter-connected nodes to be classified into the same class when using GNNs becomes inaccurate for biological/chemical networks. It calls for an urgent need to design a pooling method that captures the role of each node.

Therefore, based on the above assumptions, we no longer encourage the assignment of neighboring nodes to the same cluster, and we add an auxiliary process to the cluster assignment process. By using the features of the coarsened graph obtained from cluster assignment to reconstruct the original graph, we want to capture the role played by each node in the reconstruction process and assign nodes that play similar roles in the composition of the graph to the same cluster. Our main contributions are summarized as follows:

- We propose a novel graph pooling operator named RecPool, which for the first time incorporates the role played by nodes in composing a graph into the hierarchical pooling method and provides a new perspective on graph pooling.
- We model the prior distribution of variational inference using the Laplace distribution, which improves the performance of RecPool on graph classification. And we prove the computational upper bound of Kullback-Leibler Divergence for variational and prior distributions.
- We compare the accuracy of graph classification to state-of-the-art methods on multiple public datasets, which demonstrates RecPool's superior ability to learn graph representations.

The remainder of this article is organized as follows. Section II illustrates the related work. Section III describes the preliminaries. In Section IV, the proposed scheme is introduced in detail. Section V gives the experimental evaluation. The conclusions are summarized finally.

II. Related Work

A. Graph Pooling

GNNs can solve graph classification tasks by generating a graph-level representation. Pooling operations can downsize inputs thus reducing the number of parameters, expanding the receptive fields and suggesting the generalization ability of the model. Therefore combining graph neural networks with pooling is known as a mainstream trend in solving graph classification problems recently, and we can divide the recent graph pooling methods into two major branches: global pooling and hierarchical pooling.

Global graph pooling combines the representation of nodes into a vector as a graph representation. The graph readout operation [12] generates the graph embedding by summing or aggregating the embeddings of all nodes using a neural network. Set2set [13] obtains a representation of the graph by using the LSTM model. DGCNN [14] obtains a fixed size graph representation by sorting the order of the nodes. Global graph pooling generates graph representations by node features only, which may lose important information.

Hierarchical graph pooling methods captures the hierarchical representation of the graph during pooling by constructing a hierarchical GNNs. DiffPool [2] maps nodes to a set of clusters by learning a soft assignment matrix. AttPool [15] and SAGPool [12] learn the graph representation by designing a top-k node selection strategy to select the most important nodes to form a coarsening graph. RepPool [11] reduces information loss by a learnable method to integrate non-selected nodes. NIAPool [16] captures the discrepancies between nodes through an attention mechanism to obtain information about the nodes in the graph from both local and global aspects. EigenPool [3] performs pooling operations by introducing the graph Fourier transform. However, most of the above methods do not consider the role played by the nodes in the composition of the graph, but rather cluster the nodes based on the features of the nodes themselves or considering the structural characteristics of the graph, and most of the adjacent nodes will be divided into the same cluster, which is inappropriate from the perspective of network composition, for example, in a graph (protein), connected nodes (amino acids) should not necessarily be in the same cluster (the same or chemically similar amino acids), and similarly, nodes that are far apart, may belong to the same cluster.

B. Information Bottleneck

Information bottlenecks (IB) [17], originally applied in the field of signal processing, are used to optimize the short code for retaining the maximum amount of information in the input signal. Variational information bottleneck (VIB) [18] introduces information bottleneck theory to deep learning for the first time, allowing the model to learn compressed and meaningful representations. Today, a large amount of research has applied IB and VIB to graph neural networks [19], [20], recommender systems [21], computer vision [22], natural language processing [23], and other fields. In some recent work concerning information theory and graph learning [24], [25], [26], [27], researches have applied information bottleneck theory to graph structure learning tasks with the aim of removing noise and redundant information from graph data and extracting potentially task-relevant key graph structures. However, in this article, we are more concerned with our model's ability to capture the role played by nodes in the composition of the graph, which is crucial in some biochemical networks. Therefore, we utilized a different approach from these works mentioned above by optimizing the hierarchical graph pooling process using information bottleneck theory to generate a coarsened graph that may not be present in the original graph, and on top of that, we added a reconfiguration process to capture the role played by nodes in the composition of the graph.

C. Graph Generation

Graph generation models are used to synthesize graph data that retain certain statistical features of the original data and thus facilitate graph data mining tasks. Early graph generation models mainly used probabilistic models to manually design graphs with certain statistical features based on publishers' observations and empirical knowledge [28], [29].

Due to the rapid development of deep learning, many advanced graph generation models have been developed to take advantage of the more powerful data abstraction capabilities of neural networks to generate graphs that contain more information [30], [31], [32], [33]. For Example, GraphRNN [31] decomposes the process of graph generation into the process of node generation and edge addition, and completes the generation of the graph through two levels of recurrent neural networks; MolGAN [33] uses generative adversarial networks for graph generation, while using reinforcement learning to encourage the generated graphs to have certain properties based on the reward function. These graph generation models are widely used in molecular graph generation [33], social network graph publishing [34] and other fields with state-of-the-art results.

D. Variational Graph Auto-Encoder

Variational Graph Auto-Encoder (VGAE) [30] is an extension of Variational Auto-Encoder (VAE) [35], [36] to graph structure data. Several graph representation learning methods generate embeddings of nodes by obtaining a deterministic hidden representation [37], [38].

Problem 1. Given an attributed graph $G = (A, X)$ with the adjacency matrix $A \in \mathbb{R}^{n\times n}$ and node feature matrix $X \in \mathbb{R}^{n\times d}$, find the probability distribution of the latent representation of nodes $Z \in \mathbb{R}^{n\times m}$, i.e., $p(Z|X, A)$.

To solve this problem, Kipf et al. [30] obtain interpretable latent representations by using a Gaussian distribution $q(Z|X, A) = \prod_{i=1}^{n} q_i(z_i|X, A)$ with $q_i(z_i|X, A) = \mathcal{N}(h_i|\mu_i, \mathrm{diag}(\sigma_i^2))$ to approximate the true posterior distribution of the embedding using a Graph convolutional network(GCN) as an encoder and using a simple inner product decoding, while learning the mean and standard deviation matrices of the approximate distribution using two GCNs, i.e., $\mu = GCN_\mu(X, A), log\sigma = GCN_\sigma(X, A)$.

III. Preliminaries

A. Problem Formulation

We focus on semi-supervised graph classification in an attributed graph $G = (A, X)$, where $A \in \mathbb{R}^{n\times n}$ is the symmetric adjacency matrix with n nodes and $X \in \mathbb{R}^{n\times d}$ is the node feature matrix, where d is the dimension of node features. Specifically, $A_{ij} = 1$ represents there is an edge between node i and j, otherwise, $A_{ij} = 0$. Graph classification maps a set of labeled graphs $\mathcal{D} = \{(G_1, y_1), (G_2, y_2), \ldots\}$, where $y_i \in \mathcal{Y}$ is the label corresponding to graph $G_i \in \mathcal{G}$, to the set of labels by learning a mapping $f : \mathcal{G} \rightarrow \mathcal{Y}$.

B. Graph Neural Networks

Graph neural networks efficiently obtain a representation of nodes by aggregating network structures as well as features [37], [39]. In this way, the feature information of nodes is propagated through the network topology, and the information obtained by node aggregation generates node embeddings. Specifically, we use the "message passing" form to define the GNN:

$$H^{(l)} = ReLU\left(\tilde{A}H^{(l-1)}W^{(l)}\right) \tag{1}$$

where $H^{(l)} \in \mathbb{R}^{n\times d}$ are the embeddings computed after l-step propagation of the GNN, $\tilde{A} = \tilde{D}^{-\frac{1}{2}}(A + I)\tilde{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix, and $W^{(l)}$ is a weight matrix. In particular, $H^{(0)} = X \in \mathbb{R}^{n\times d}$.

C. Differentiable Pooling

Diffpool [2] uses the output of the GNN model to learn the cluster assignment matrix over the nodes. By stacking L GNN modules and learning to use the embeddings generated from GNNs at layers $l - 1$, Diffpool assigns nodes to clusters at layer l in an end-to-end fashion. In layer l, given the feature matrix $X^{(l)}$ of the coarsened nodes, and the coarsened adjacency matrix $A^{(l)}$, two GNNs are used to generate the assignment matrix $S^{(l)}$ and embedding matrix $Z^{(l)}$, respectively:

$$Z^{(l)} = GNN_{l,\text{embed}}(A^{(l)}, X^{(l)}) \tag{2}$$

$$S^{(l)} = \text{softmax}\left(GNN_{l,\text{pool}}(A^{(l)}, X^{(l)})\right) \tag{3}$$

where the softmax function is applied in a row-wise fashion. At layer $l = 0$, the adjacency matrix A and the node feature matrix X are used as inputs to the model. Each row of the cluster assignment matrix $S^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}}$ corresponds to a cluster at layer l, and each column corresponds to a cluster at layer $l+1$ after coarsening, which indicates the soft assignment of each cluster at the current layer to a cluster at the next layer.

After giving the cluster assignment matrix $S^{(l)}$, in order to perform the hierarchical pooling process, it is necessary to give the topological information of the coarsened graph as well as the feature information. A new coarsened adjacency matrix $A^{(l+1)}$ and embedding matrix $X^{(l+1)}$ can be generated by the following two equations:

$$X^{(l+1)} = {S^{(l)}}^T Z^{(l)} \in \mathbb{R}^{n_{l+1} \times d} \tag{4}$$

$$A^{(l+1)} = {S^{(l)}}^T A^{(l)} S^{(l)} \in \mathbb{R}^{n_{l+1} \times n_{l+1}} \tag{5}$$

With the cluster assignment matrix $S^{(l)}$, it is possible to assign n_l nodes at layer l to n_{l+1} clusters at layer $l+1$, where $n_l > n_{l+1}$, so that the clusters at layer $l+1$ correspond to a number of nodes at layer l. $A^{(l+1)}$ represents the adjacency matrix of a fully connected graph, and the elements of the matrix, $A_{ij}^{(l+1)}$, represent the connection strength of node i and node j in the coarsened graph, and similarly, $X^{(l+1)}$ represents the embedding of nodes in the coarsened graph.

D. Graph Information Bottleneck

Given the input graph $\mathcal{G}$ and the label Y, the objective of GIB is maximized to find the internal code Z : $\max_Z I(Z,Y) - \beta I(\mathcal{G}, Z)$, where the value of the hyperparameter β can determine the balance between informativeness and compressibility, and $I(X,Y)$ refers to the Shannon mutual information of two random variables. Optimizing this objective will lead to a compact but informative Z. For the process of graph pooling, we expect to find the most informative coarsened graph G_{coar}. Yu et al. [20] found the maximally informative but compressed subgraph of graph G to enhance tasks such as graph classification, graph interpretation and graph denoising, called IB-subgraph, which are obtained by optimizing the following objectives:

$$\max_{\mathcal{G}_{\text{sub}} \in \mathbb{G}_{\text{sub}}} I(Y, \mathcal{G}_{\text{sub}}) - \beta I(\mathcal{G}, \mathcal{G}_{\text{sub}}) \tag{6}$$

where $\mathbb{G}_{\text{sub}}$ indicates the set of all subgraphs of $\mathcal{G}$.

IV. The Proposed Model

The key idea of Recpool is that we construct the pooling reconstruction module to sample the features from the coarsened graph, map the sampled feature distribution to the orginal nodes according to the assignment matrix, and use the mapped features to reconstruct the graph. At the same time, we use graph information bottleneck theory to optimize the graph pooling process so that the coarsened graph retains as much information as possible, but retains less noise and redundant structural information. The framework of RecPool is shown in Fig. 3. In this section, we will describe in detail the RecPool.

A. Graph Reconfiguration

The extant graph pooling methods, both in terms of model design and experimental results, exhibit a phenomenon where neighboring nodes are assigned to the same cluster. In Diffpool [2], the authors use Auxiliary Link Prediction Objective as the regularization term, which makes it easier for two nodes with higher connection strength to be mapped to the same cluster. Similarly, in EigenPool [3], the authors directly use Spectral Clustering for subgraph partitioning, which maps nodes of the same subgraph to the same cluster. This seems intuitively correct, but in some graph structures, especially biological networks, this causes the model to ignore the role of individual nodes in the graph.

To capture the role played by each node in the configuration of the graph, we use the VGAE [30] to capture the probability distribution corresponding to each cluster after coarsening, use the assignment matrix S to sample features for each node from the corresponding cluster, and finally perform link prediction based on the sampled node features to simulate the process of graph configuration. In the following, we illustrate how to use VGAE to capture the feature distribution of each cluster and how to use the cluster assignment matrix to sample features for each node, and we try to use Laplace prior as the prior distribution of the hidden variables and verify the results from different prior distributions.

In order to obtain the hidden variable H, we parameterize a simple inference model using a two-layer GCN:

$$q(H_{\text{coar}}|Z_{\text{coar}}, A_{\text{coar}}) = \prod_{i=1}^{N} q(h_i|Z_{\text{coar}}, A_{\text{coar}}),$$

$$\text{with} \quad q(h_i|X_{\text{coar}}, A_{\text{coar}}) = \mathcal{N}(h_i|\mu_i, \text{diag}(\sigma_i^2)) \tag{7}$$

where, $\mu = GCN_\mu(Z_{\text{coar}}, A_{\text{coar}})$ is the matrix of mean vectors μ_i; similarly $log\sigma = GCN_\sigma(Z_{\text{coar}}, A_{\text{coar}})$.

Since the number of nodes after pooling is $m < n$, after using the coarsened embedding matrix $Z_{\text{coar}} \in \mathbb{R}^{m \times d_1}$ to obtain the hidden variable $H_{\text{coar}} \in \mathbb{R}^{m \times d_2}$, where d_1, d_2 are constants, we need to construct a method to map the hidden variable H_{coar} of the coarsened graph back into the original nodes according to the cluster assignment. Since the cluster assignment matrix S in (4) represents the mapping relationship between nodes and clusters, we use the following equation to assign the extracted cluster features to the corresponding nodes:

$$H_{\text{extend}} = S H_{\text{coar}} \in \mathbb{R}^{n \times d_2} \tag{8}$$

Equation (8) constructs the feature matrix H_{extend} of the node before pooling by upsampling the feature matrix H_{coar} of the cluster through the cluster assignment matrix S.

After obtaining the approximate joint posterior of H_{coar} parameterized by μ and σ, we use the cluster assignment matrix to extend the resampled H_{coar} to all nodes as shown in (8). Now that the clustering information of each node is captured in H_{extend}, we use the Multi-layer Perceptron (MLP) as a decoder aiming to

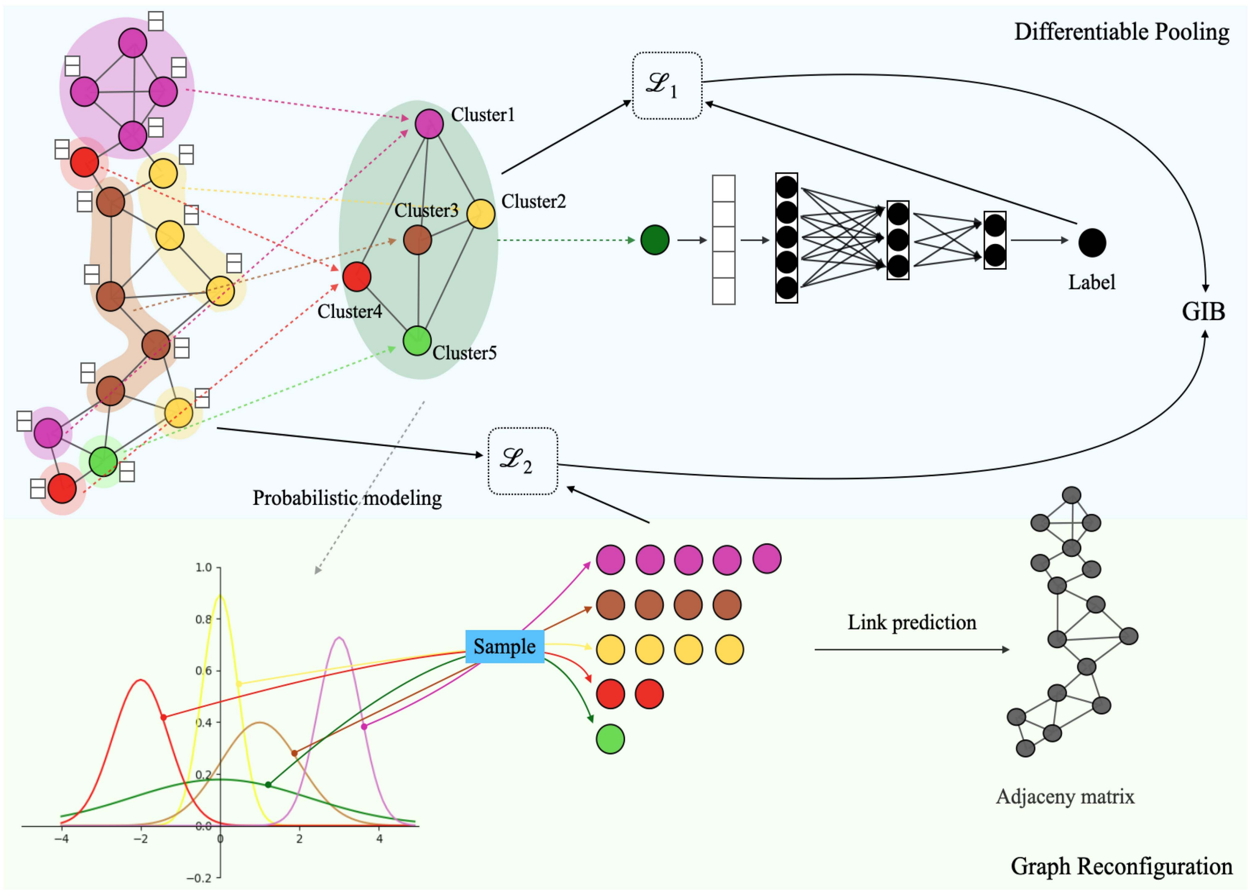


Fig. 3. Illustration of the proposed RecPool framework. In the Differential Pooling process, we use the cluster assignment matrix to hierarchically pool the original graph and predict the graph labels; in the Graph Reconfiguration process, we use the cluster assignment matrix to map the features of the coarsened graph obtained from the Differential Polling process back to the original graph for graph reconstruction, so that the cluster assignment matrix captures the role played by the nodes in the reconfiguration process.

reconstruct the adjacency matrix A so that the cluster assignment matrix can optimize the process of cluster assignment based on the reconfiguration, i.e., each node determines the cluster assignment of a node according to the role it assumes in the process of configuration the graph or some kind of substructure in the graph. Here, MLP is used to decode the adjacency matrix:

$$p(A|H_{\text{extend}}) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(A_{ij}|H_{\text{extend}})$$

$$\text{with} \quad p(A_{ij}=1|H_{\text{extned}}) = \sigma(W \times H_{\text{extend}} + b) \tag{9}$$

where A_{ij} are the elements of A, W is the weight matrix, b is the bias term, and $\sigma(\cdot)$ is the logistic sigmoid function.

We use the distance metric between the generated and original graphs, and the scatter between the vector representation of the nodes and the normal distribution to form the loss function of the reconstruction part and optimize its variational lower bound:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(H_{\text{coar}}|Z_{\text{coar}},A_{\text{coar}})}\left[\log p(A|H_{\text{extend}})\right] \\ &\quad - \text{KL}\left[q(H_{\text{coar}}|Z_{\text{coar}},A_{\text{coar}})||p(H_{\text{coar}})\right]\end{aligned} \tag{10}$$

where $\text{KL}[q(\cdot)||p(\cdot)]$ is the Kullback-Leibler divergence between $q(\cdot)$ and $p(\cdot)$. For the choice of prior distribution, unlike the Gaussian distribution used in the original VGAE, we use the Laplace distribution as the prior distribution $p(H_{\text{coar}}) = \prod_i p(h_i) = \prod_i \mathcal{L}a(h_i|0,\text{I})$.

Lemma 1. Upper bound on the mathematical expectation of the absolute value of a random variable obeying the Laplace distribution with parameters $\lambda,\ \mu$:

$$\mathbb{E}[|X|] \leq \lambda + |\mu| \tag{11}$$

Proof 1:

$$\begin{aligned}\mathbb{E}[|X|] &= \frac{1}{2\lambda}\int_{-\infty}^{+\infty} |x| e^{-\frac{|x-\mu|}{\lambda}} dx \\ &= \frac{1}{2}\int_{-\infty}^{+\infty} |\lambda y + u| e^{-|y|} dy \\ &\leq \frac{1}{2}\int_{-\infty}^{+\infty} (|\lambda y| + |u|)\, e^{-|y|} dy \\ &= \lambda \int_{0}^{+\infty} y e^{-y} dy + |u| \int_{0}^{+\infty} e^{-y} dy \\ &= \lambda + |u|\end{aligned} \tag{12}$$

Theorem 1. The lower bound for the Kullback-Leibler scatter calculation using the Laplace distribution as the prior distribution is:

$$\begin{aligned} &-\mathrm{KL}\left[q(H_{\text{coar}}|Z_{\text{coar}}, A_{\text{coar}})||p(H_{\text{coar}})\right] \\ &\geq log(\lambda_q) - \lambda_q - |\mu_q| + 1 \end{aligned} \tag{13}$$

where, $\mu_q = GCN_\mu(Z_{\text{coar}}, A_{\text{coar}})$ is the matrix of location vectors μ_i; similarly $\lambda_q = GCN_\sigma(Z_{\text{coar}}, A_{\text{coar}})$ is the matrix of scale parameters λ_i.

Proof 2: Suppose $p(H_{\text{coar}}) = \frac{1}{2\lambda_p} e^{-\frac{|x-\mu_p|}{\lambda_p}}$ and $q(H_{\text{coar}}|Z_{\text{coar}}, A_{\text{coar}}) = \frac{1}{2\lambda_q} e^{-\frac{|x-\mu_q|}{\lambda_q}}$. Then

$$\begin{aligned} &-\mathrm{KL}\left[q(H_{\text{coar}}|Z_{\text{coar}}, A_{\text{coar}})||p(H_{\text{coar}})\right] \\ &= \int \frac{1}{2\lambda_q} e^{-\frac{|x-\mu_q|}{\lambda_q}} \log\left(\frac{\frac{1}{2\lambda_p} e^{-\frac{|x-\mu_p|}{\lambda_p}}}{\frac{1}{2\lambda_q} e^{-\frac{|x-\mu_q|}{\lambda_q}}}\right) dh \\ &= \frac{1}{2\lambda_q} \int e^{-\frac{|x-\mu_q|}{\lambda_q}} \left(\log\left(\frac{\lambda_q}{\lambda_p}\right) - \frac{|x-\mu_p|}{\lambda_p} + \frac{|x-\mu_q|}{\lambda_q}\right) dh \\ &= \mathbb{E}_q\left[\log\left(\frac{\lambda_q}{\lambda_p}\right) - \frac{|x-\mu_p|}{\lambda_p} + \frac{|x-\mu_q|}{\lambda_q}\right] \\ &= \log\left(\frac{\lambda_q}{\lambda_q}\right) - \frac{1}{\lambda_p}\mathbb{E}_q\left[|x-\mu_p|\right] + \frac{1}{\lambda_q}\mathbb{E}_q\left[|x-\mu_q|\right] \\ &= \log\left(\frac{\lambda_q}{\lambda_p}\right) - \frac{1}{\lambda_p}\mathbb{E}_q\left[|x-\mu_p|\right] + 1 \\ &\text{Let } \lambda_p = 1, \mu_p = 0, \text{ and apply } Lemma\ 1: \\ &= \log(\lambda_q) - \mathbb{E}_q\left[|x|\right] + 1 \\ &\geq \log(\lambda_q) - \lambda_q - |\mu_q| + 1 \end{aligned}$$

B. Graph Information Bottleneck for Graph Pooling

To find the most informative but compressed subgraph $\mathcal{G}_{\text{sub}}$, Yu et al. [20] achieve this by optimizing the objective in (2). However, for the graph pooling process, we are not limited to finding a representative subgraph in the original graph, but wish to construct a new coarsened graph $\mathcal{G}_{\text{coar}}$ from the original graph by learning a mapping relationship $S \in \mathbb{R}^{n\times m}$ between nodes and clusters. We want to find a most representative coarsening graph $\mathcal{G}_{\text{coar}}$ that contains the maximum information for graph classification while having the maximum compression within a specified range (determined by the hyperparameters). We hope to optimize the cluster assignment process by exploiting the graph information bottleneck so that the coarsened graph has very little redundant information and contains the most information favorable to graph classification. According to (6) we can derive the graph information bottleneck (GIB) objective for the pooling process:

$$\max_{\mathcal{G}_{\text{coar}}\in\mathbb{G}_{\text{coar}}} I(Y, \mathcal{G}_{\text{coar}}) - \beta I(\mathcal{G}, \mathcal{G}_{\text{coar}}) \tag{14}$$

where $\mathbb{G}_{\text{coar}}$ denotes the set of all coarsening graphs of G. Different coarsening graphs $\mathcal{G}_{\text{coar}}$ are determined by different cluster assignment matrices S.

The GIB objective of the pooling process as shown in (14) contains two components. The first component $I(Y, \mathcal{G}_{\text{coar}})$ measure the relevance between Y and $\mathcal{G}_{\text{coar}}$. Give its expanded form:

$$\begin{aligned} I(Y, \mathcal{G}_{\text{coar}}) = &\int p(y, \mathcal{G}_{\text{coar}}) \log p(y|\mathcal{G}_{\text{coar}}) dy\, d\mathcal{G}_{\text{coar}} \\ &+ H(Y) \end{aligned} \tag{15}$$

where the entropy $H(Y)$ of the graph labels can be ignored in the optimization process. We can use the empirical distribution $p(y, \mathcal{G}_{\text{coar}}) = \frac{1}{N}\sum_{i=1}^{N} \delta_y(y_i)\delta_{\mathcal{G}_{\text{coar}}}(\mathcal{G}_{\text{coar}_i})$ to approximate $p(y, \mathcal{G}_{\text{coar}})$, where $\delta()$ is the Dirac function of the sample training data, y_i and $\mathcal{G}_{\text{coar}_i}$ are the coarsened graph and label corresponding to i-th training data. Since obtaining $p(y|\mathcal{G}_{\text{coar}})$ directly is intractable, let $q_{\phi_1}(y|\mathcal{G}_{\text{coar}})$ be a variational approximation to $p(y|\mathcal{G}_{\text{coar}})$. By using the variational approximation $q_{\phi_1}(y|\mathcal{G}_{\text{coar}})$ instead of the true posterior, we can obtain a tractable lower bound for (15):

$$\begin{aligned} I(Y, G_{\text{coar}}) \geq &\int p(y, G_{\text{coar}}) \log q_{\phi_1}(y|G_{\text{coar}}) dy\, dG_{\text{coar}} \\ &\approx \frac{1}{N}\sum_{i=1}^{N} \log q_{\phi_1}(y_i|G_{\text{coar}_i}) \\ &=: -\mathcal{L}_1(q_{\phi_1}(y|G_{\text{coar}}), y_t) \end{aligned} \tag{16}$$

where y_t is the true classification label corresponding to the graph. From (16), we maximize $I(Y, \mathcal{G}_{\text{coar}})$ by minimizing the classification loss $\mathcal{L}_1$ between $\mathcal{G}_{\text{coar}}$ and Y. This encourages the model to use the coarsened graph to predict the labels of the original graph.

For the second part of the GIB objective in (14), a reasonable prior distribution of $p(\mathcal{G}_{\text{coar}})$ is difficult to find because it represents the distribution of the coarsened graph rather than a latent representation, so, analogous to the operation of Yu et al. [20] to approximate the mutual information between $\mathcal{G}$ and $\mathcal{G}_{\text{sub}}$ using the DONSKER-VARADHAN representation [40] of KL-divergence directly, we apply it to the process of graph pooling to obtain the approximate form of the mutual information between $\mathcal{G}$ and $\mathcal{G}_{\text{coar}}$ as follows:

$$\begin{aligned} I(\mathcal{G}, \mathcal{G}_{\text{coar}}) = &\sup_{f_{\phi_2}:\mathbb{G}\times\mathbb{G}\rightarrow\mathbb{R}} \mathbb{E}_{\mathcal{G},\mathcal{G}_{\text{coar}}\in p(\mathcal{G},\mathcal{G}_{\text{coar}})} f_{\phi_2}(\mathcal{G}, \mathcal{G}_{\text{coar}}) \\ &- \log \mathbb{E}_{\mathcal{G}\in p(\mathcal{G}),\mathcal{G}_{\text{coar}}\in p(\mathcal{G}_{\text{coar}})} e^{f_{\phi_2}(\mathcal{G},\mathcal{G}_{\text{coar}})} \end{aligned} \tag{17}$$

We use GNN to design a statistical network f_{ϕ_2}. This network first uses GNN to obtain the embedding of the original graph $\mathcal{G}$ and the coarsened graph $\mathcal{G}_{\text{coar}}$, and then concatenates the embeddings of $\mathcal{G}$ and $\mathcal{G}_{\text{coar}}$ into an MLP network to map the set of graphs into a set of real numbers. In conjunction with the approximation method using the empirical distribution in $\mathcal{L}_1$, we can obtain the mutual information minimization proxy objective for $I(\mathcal{G}, \mathcal{G}_{\text{coar}})$:

$$\begin{aligned} \max_{\phi_2} \mathcal{L}_2(\phi_2, \mathcal{G}_{\text{coar}}) = &\frac{1}{N}\sum_{i=1}^{N} f_{\phi_2}(\mathcal{G}, \mathcal{G}_{\text{coar}_i}) \\ &- \log \frac{1}{N} \sum_{i=1,j\neq i}^{N} e^{f_{\phi_2}(\mathcal{G}_i, \mathcal{G}_{coar_j})} \end{aligned} \tag{18}$$

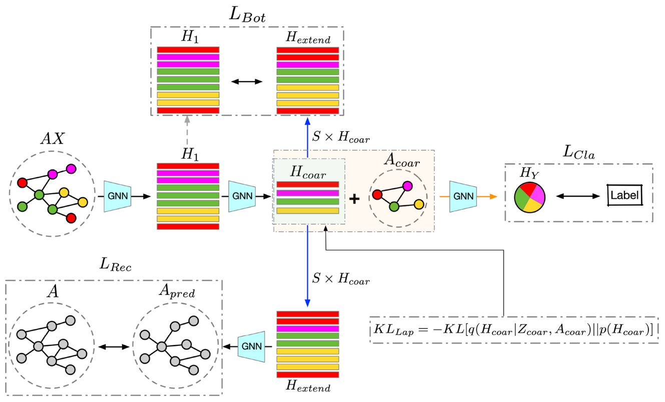


Fig. 4. RecPool's training framework, where L_{Rec} and KL_{Lap} together form (10), L_{Cla} represents (16), and L_{Bot} represents $\mathcal{L}_2$ in (19).

TABLE I
STATISTICS OF THE DATASETS

Datasets	Graphs	Nodes(avg)	Edges(avg)	#Class
ENZYMES	600	32.63	62.14	6
PROTEINS	1113	39.06	72.82	2
D&D	1178	284.32	715.66	2
NCI1	4110	29.87	32.30	2

Combining (14), (16), and (18), we use a bi-level optimization process to optimize the GIB objective:

$$\min_{\mathcal{G}_{\text{coar}},\phi_1} \mathcal{L}(\mathcal{G}_{\text{coar}}, \phi_1, \phi_2^*) = \mathcal{L}_1(q_{\phi_1}(y|G_{\text{coar}}), y_t) + \beta\mathcal{L}_2(\phi_2^*, \mathcal{G}_{\text{coar}}) \quad (19)$$

$$\text{s.t.} \quad \phi_2^* = \arg\max \mathcal{L}_2(\phi_2, \mathcal{G}_{\text{coar}}) \quad (20)$$

In the inner loop, we first optimize (18) to obtain a sub-optimal mutual information estimator ϕ_2^*, and then, in the outer loop, use the estimator ϕ_2^* obtained in the inner loop for global optimization to obtain the parameters ϕ_1 and the coarsening graph $\mathcal{G}_{\text{coar}}$. The overall training framework of RecPool is shown in Fig. 4.

V. EXPERIMENT

In this section, we will experimentally measure the effectiveness of the RecPool. We will first present the dataset used and the state-of-the-art methods we compared, then we will detail the experimental settings of the model and give the results of graph classification and the corresponding analysis. Finally, we will compare and analyze the cluster assignments of the different methods on the real graph dataset.

A. Experimental Settings

1) Datasets: To evaluate the graph classification performance of our proposed model, we conduct experiments on four public graph classification benchmark datasets, which includes three protein graph datasets, i.e., D&D [41], PROTEINS [8] and ENZYMES [42]; and a dataset of anticancer activity of compounds NCI1 [43]. Statistics of the datasets are shown in Table I.

2) Baselines: We take three kinds of methods as baselines: (1) Curriculum Learning method including *CurGraph* [44]; (2) GNN-based methods including *GCN* [45], *GraphSage* [37], *GIN* [39], *SET2SET* [13]; (3) hierarchical graph pooling methods including *Diffpool* [2], *EigenPool* [3], *RepPool* [11], *NIAPool* [16] and *GIB* [20].

3) Experimental Settings: We have implemented Recpool using the pytorch framework. For all GNN baselines as well as Curriculum learning methods, we used 10-fold cross-validation figures reported by the original authors when possible. For all hierarchical graph pooling methods, we use only 1 pooling layer for comparison based on performance as well as fairness considerations. Ying et al. [2] argue that the 1 pooling layer approach can achieve similar performance in small datasets such as ENZYMES, PROTEINS, etc. Therefore, we uniformly use a 1 pooling layer architecture, and the pooling ratio r is set to be in the range of [0.1, 0.4]. The Differentiable Pooling part of our model follows the Diffpool [2] implementation. In the part of RecPool, we use a GCN layer to transform the embedding of the clusters to obtain the hidden variables, and then use two GCNs to construct the mean and variance vectors of the cluster distribution. The expanded feature vector is finally decoded by an MLP composed of 3 fully connected layers to obtain the predicted adjacency matrix. More details of the experimental settings can be found in the experimental source code.[1]

TABLE II
COMPARISON OF GRAPH CLASSIFICATION PERFORMANCE WITH THE STATE-OF-ART METHODS

Method	Data set			
	ENZYMES	D&D	PROTEINS	NCI1
GCN	44.03 ± 3.59	75.90 ± 1.41	74.00 ± 3.59	72.53 ± 2.29
GraphSage	55.42 ± 3.42	76.10 ± 3.20	73.60 ± 3.53	73.22 ± 1.79
GIN	31.11 ± 1.92	65.94 ± 1.87	68.17 ± 2.39	72.18 ± 1.93
SET2SET	40.15 ± 7.81	72.12 ± 6.28	70.13 ± 3.79	71.51 ± 2.31
DiffPool	59.74 ± 2.53	77.27 ± 2.38	75.58 ± 2.36	76.89 ± 1.90
EigenPool	62.44 ± 3.81	75.98 ± 4.03	74.17 ± 3.15	78.27 ± 1.95
CurGraph	$\mathbf{64.80 \pm 3.39}$	78.60 ± 3.04	75.40 ± 3.10	80.61 ± 1.73
RepPool	63.14 ± 1.36	78.06 ± 4.85	77.50 ± 2.15	79.24 ± 2.34
GIB	62.35 ± 1.37	78.16 ± 1.42	74.90 ± 1.51	77.68 ± 1.41
NIAPool	62.04 ± 2.32	79.28 ± 4.30	75.25 ± 3.71	78.08 ± 1.92
RecPool	62.83 ± 3.40	$\mathbf{80.31 \pm 2.70}$	$\mathbf{78.29 \pm 2.49}$	$\mathbf{80.73 \pm 1.85}$

B. Performance on Graph Classification

We compared the performance of RecPool with the baseline algorithm for graph classification on 4 datasets, the accuracy and standard deviation are reported in Table II. First, one can observe that the performance of RecPool is superior to its counterparts on three out of four benchmarks. In particular, RecPool improves over the best baseline by 1.03% on the DD dataset and by 0.79% on the PROTEINS dataset. Next, RecPool consistently outperforms the GNN-based method significantly on all datasets, demonstrating the importance of adding a pooling module to the model. This is because RecPool can extract more useful graph structure information than the global pooling used in the GNN-based approach. By comparing with existing methods of hierarchical pooling methods, RecPool achieves better performance on most datasets, which shows that we can improve the model's performance on graph classification by adding a graph reconfiguration part to the pooling process to optimize the cluster

[1] https://github.com/Marvin-huoshan/RecPool

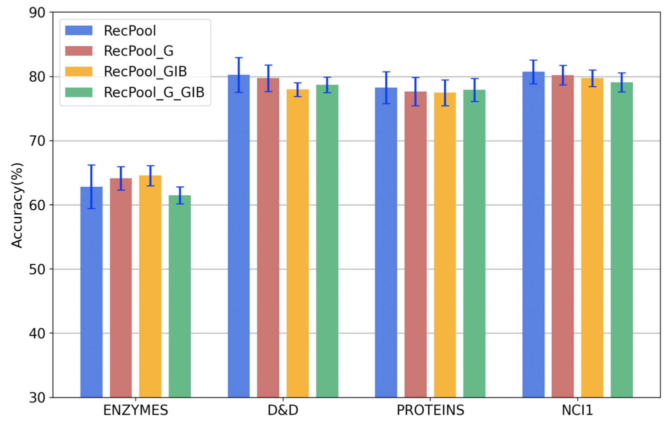


Fig. 5. Accuracy and standard deviation of graph classification between RecPool and its variants.

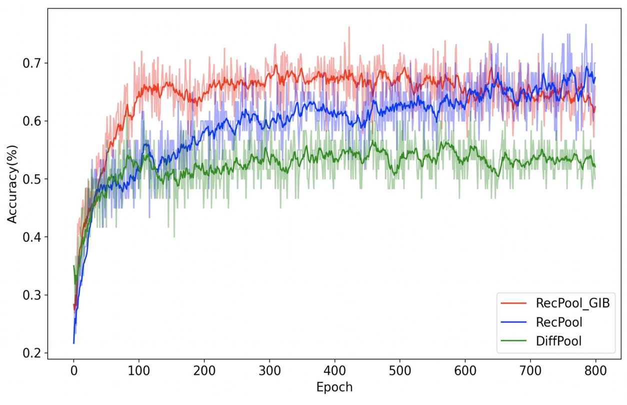
assignment matrix. In particular, for biological networks, each node or substructure assumes a different role in the graph and has multiple combinations to form different proteins or compounds, so building a method to learn how nodes are composed in the graph and assigning the same type of nodes to the same clusters can help optimize our cluster assignment process and ultimately achieve more accurate and more interpretative graph classification performance.

C. Ablation Study

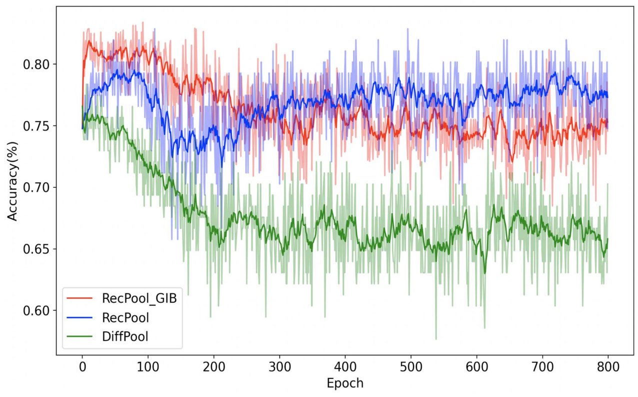
In this section, we verify the effectiveness of the proposed component on helper graph classification. In particular, we investigated the effect of using the Laplace distribution as a prior distribution for VGAE on graph classification accuracy. We also investigate whether the optimization of the graph pooling process using GIB is effective in improving graph classification accuracy. To achieve the goal, we define the following variants of the proposed framework:

- RecPool_G: It denotes the use of Gaussian distribution as the prior distribution of VGAE in the graph reconfiguration part, where RecPool uses the Laplace distribution.
- RecPool_GIB: It denotes the process of optimizing cluster assignment using GIB on top of RecPool.
- RecPool_G_GIB: It denotes the process of optimizing cluster assignment using GIB on top of RecPool_G.

As shown in Fig. 5, RecPool outperforms the other variants in three of the four datasets ENZYMES, D&D, PROTEINS, and NCI1. Among them, by comparing RecPool with RecPool_G we can know that using Laplace distribution as the prior distribution of VGAE can achieve better results in most cases. Meanwhile, RecPool performs poorly on the ENZYMES dataset due to the fact that the Laplace distribution has a higher probability density near and away from the mean compared to the Gaussian distribution. Our modeling of the hidden representation of clusters using the Laplace distribution will cause the features of nodes sampled in the same cluster to be concentrated in the mean part of the cluster and away from the mean, reflecting the most salient features of the current cluster, which is beneficial for binary classification tasks, but in multi-classification tasks. This property



(a) ENZYMES



(b) PROTEINS

Fig. 6. Classification accuracy of RecPool_GIB, RecPool and DiffPool on the validation set during training. The solid line is the trend of the smoothed curve, and the shaded part is the change in classification accuracy on the validation set.

of the Laplace distribution will make the capture of features inadequate. Therefore, we can adjust the prior distribution of the model to get the best results according to the problem the model needs to solve. Also, by comparing the performance of graph classification accuracy and standard deviation before and after optimization with GIB, we can observe that using GIB does not always result in good graph classification accuracy, but has a better effect on reducing the standard deviation of graph classification, which can make the model's performance in graph classification task more stable.

As shown in Fig. 6, we plot the graph classification performance of RecPool, RecPool_GIB and DiffPool on the validation set for the model during the training process, and it can be observed that our proposed RecPool and its variants are superior to DiffPool in terms of performance in graph classification. At the same time, our proposed scheme achieves better performance with fewer Epoch than DiffPool, which indicates that using graph reconstruction for auxiliary graph classification is effective, and by comparing RecPool with RecPool_GIB, we can see that although adding GIB does not improve the accuracy of graph classification more significantly, but using GIB allows the model to obtain a faster convergence rate and avoid oscillations, which increases the stability of the model.

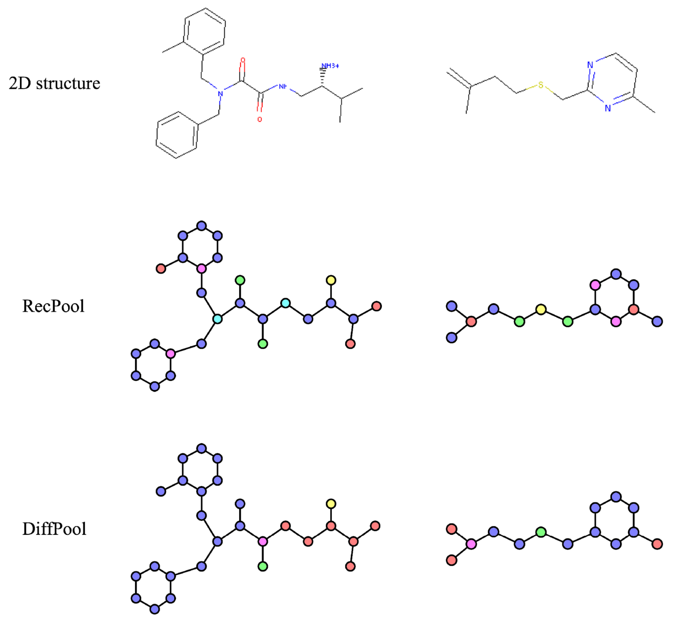


Fig. 7. Visualization of cluster assignment of RecPool and Diffpool on two real compounds.

D. Case Study

Unlike traditional hierarchical graph pooling methods such as DiffPool, our proposed RecPool uses graph reconstruction loss to optimize the cluster assignment process and does not encourage neighboring nodes to be assigned to a cluster, making the clustering of nodes constrained by the role played by each node in the composition of the graph. In this subsection, we illustrate how the cluster assignment in this scheme differs from DiffPool. Specifically, we observe the results of DiffPool and our scheme for node clustering on two real compounds.

As shown in Fig. 7, if nodes are assigned to the same cluster, they are colored the same. It can be observed that nodes in the same cluster of RecPool can appear in different positions in the graph. On the contrary, most of the nodes in the same cluster of DiffPool are close to each other in terms of distance. This is consistent with what we expected before. At the same time, by observing the clustering results with the atomic composition in the compound, we can see that RecPool captures a richer and more accurate function of nodes (i.e., atoms) by simulating the composition process of the graph (i.e., compound) and by avoiding adjacent nodes to be classified in the same class. Therefore, combined with the good graph classification performance of RecPool in Table II, we can conclude that RecPool provides a new perspective for hierarchical pooling of graphs, that is, a more accurate and more explanatory hierarchical pooling method can be obtained by considering the role played by each node.

E. Parameter Analysis

We further investigated the effect of hyper-parameters on RecPool. In detail, we investigate the effect of pooling ratio r and the number of pooling layers k on the classification accuracy of the model. As shown in Fig. 8, we investigate how these parameters affect classification accuracy through changes in values on the data sets: ENZYMES, PROTEINS, D&D and NCI1.

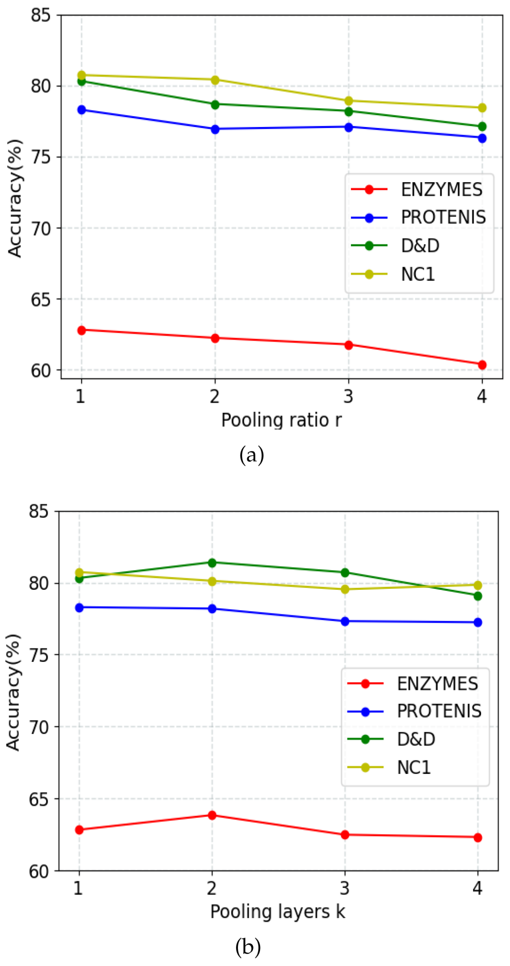


Fig. 8. Graph classification results by varying two parameters: pooling ratio r, pooling layers k on four datasets: ENZYMES, D&D, PROTEINS, NCI1.

We can find a similar trend on different datasets, regardless of the pooling rate or the number of layers pooled, the best graph classification performance of RecPool tends to be achieved with smaller parameters. By observing Fig. 8(a), we can see that the accuracy of graph classification shows an overall decreasing trend as the pooling rate increases, which indicates that when the pooling rate is larger, it may provide more redundant information to the model and affect the classification performance of the model. Observing Fig. 8(b), it can be seen that the classification performance of the model is affected when the number of pooling layers is larger, so fewer pooling layers should be selected, and at the same time, the selection of one pooling layer can already obtain a fairly good graph classification performance on most of the datasets.

VI. Conclusion

This article introduces a new graph pooling operator, which focuses on the role played by each node in the composition of the graph when pooling is considered. Specifically, the pooling operator we design consists of three components, i.e., Differentiable Pooling, Graph Reconfiguration and Graph Information Bottleneck. In the first component, we pool the graph using the framework proposed by Ying et al. [2] In the second component, we first construct the feature distribution of the pooled graph, then map the attributes of the pooled graph back to the original graph and use the mapped graph to perform graph reconstruction to optimize the attributes of the pooled graph and the pooling strategy. In the third component, we use the information bottleneck theory to optimize the generation of coarsened graphs so that the coarsened graphs we generate retain the maximum information beneficial to graph classification while containing the least redundant information. Based on RecPool, we construct a hierarchical graph pooling model to learn graph representations for graph classification through an end-to-end fashion. The excellent performance of RecPool is illustrated by extensive comparison experiments with state-of-the-art methods on several graph benchmark datasets.

References

[1] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[2] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 4805–4815.

[3] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, “Graph convolutional networks with EigenPooling,” in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 723–731.

[4] X. Liu, Z. Tang, P. Li, S. Guo, X. Fan, and J. Zhang, “A graph learning based approach for identity inference in DApp platform blockchain,” *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 1, pp. 438–449, First Quarter 2022.

[5] T. Mortlock, D. Muthirayan, S.-Y. Yu, P. P. Khargonekar, and M. A. Al Faruque, “Graph learning for cognitive digital twins in manufacturing systems,” *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 1, pp. 34–45, First Quarter 2022.

[6] M. Zhang and Y. Chen, “Link prediction based on graph neural networks,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5171–5181.

[7] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for web-scale recommender systems,” in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 974–983.

[8] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H. Kriegel, “Protein function prediction via graph kernels,” in *Proc. 13th Int. Conf. Intell. Syst. Mol. Biol.*, 2005, pp. 47–56.

[9] D. Duvenaud et al., “Convolutional networks on graphs for learning molecular fingerprints,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 2224–2232.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.

[11] J. Li, Y. Ma, Y. Wang, C. C. Aggarwal, C. Wang, and J. Tang, “Graph pooling with representativeness,” in *Proc. IEEE 20th Int. Conf. Data Mining*, 2020, pp. 302–311.

[12] J. Lee, I. Lee, and J. Kang, “Self-attention graph pooling,” in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3734–3743.

[13] O. Vinyals, S. Bengio, and M. Kudlur, “Order matters: Sequence to sequence for sets,” 2015, *arXiv:1511.06391*.

[14] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, “An end-to-end deep learning architecture for graph classification,” in *Proc. 32nd AAAI Conf. Artif. Intell. 30th Innov. Appl. Artif. Intell. 8th AAAI Symp. Educ. Adv. Artif. Intell.*, 2018, pp. 4438–4445.

[15] J. Huang, Z. Li, N. Li, S. Liu, and G. Li, “AttPool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6479–6488.

[16] C. Sun, F. Huang, and J. Peng, “Node information awareness pooling for graph representation learning,” in *Proc. 26th Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2022, pp. 182–193.

[17] N. Tishby, F. C. N. Pereira, and W. Bialek, “The information bottleneck method,” 2000, *arXiv:physics/0004057*.

[18] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, “Deep variational information bottleneck,” 2017, *arXiv:1612.00410*.

[19] T. Wu, H. Ren, P. Li, and J. Leskovec, “Graph information bottleneck,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 20 437–20 448.

[20] J. Yu, T. Xu, Y. Rong, Y. Bian, J. Huang, and R. He, “Graph information bottleneck for subgraph recognition,” 2020, *arXiv:2010.05563*.

[21] Y. Du, H. Liu, and Z. Wu, “M^3-IB: A memory-augment multi-modal information bottleneck model for next-item recommendation,” in *Proc. 27th Int. Conf. Database Syst. Adv. Appl.*, 2022, pp. 19–35.

[22] A. Zhang, Y. Gao, Y. Niu, W. Liu, and Y. Zhou, “Coarse-to-fine person re-identification with auxiliary-domain classification and second-order information bottleneck,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 598–607.

[23] C. Zhang, X. Zhou, Y. Wan, X. Zheng, K. Chang, and C. Hsieh, “Improving the adversarial robustness of NLP models by information bottleneck,” in *Findings of the Association for Computational Linguistics*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Cedarville, OH, USA: Assoc. Comput. Linguistics, 2022, pp. 3588–3598.

[24] S. Miao, M. Liu, and P. Li, “Interpretable and generalizable graph learning via stochastic attention mechanism,” in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 15 524–15 543.

[25] J. Yu, J. Cao, and R. He, “Improving subgraph recognition with variational graph information bottleneck,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 19 374–19 383.

[26] J. Yu, T. Xu, Y. Rong, Y. Bian, J. Huang, and R. He, “Recognizing predictive substructures with subgraph information bottleneck,” *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Sep. 14, 2021, doi: 10.1109/TPAMI.2021.3112205.

[27] Q. Sun et al., “Graph structure learning with variational information bottleneck,” in *Proc. 36th AAAI Conf. Artif. Intell.*, 2022, pp. 4165–4174.

[28] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[29] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks,” *Nature*, vol. 393, pp. 440–442, 1998.

[30] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” 2016, *arXiv:1611.07308*.

[31] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, “GraphRNN: Generating realistic graphs with deep auto-regressive models,” in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 5694–5703.

[32] N. D. Cao and T. Kipf, “MolGAN: An implicit generative model for small molecular graphs,” 2018, *arXiv: 1805.11973*.

[33] C. Zang and F. Wang, “MoFlow: An invertible flow model for generating molecular graphs,” in *Proc. 26th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2020, pp. 617–626.

[34] X. Li, L. Xu, H. Zhang, and Q. Xu, “Differential privacy preservation for graph auto-encoders: A novel anonymous graph publishing model,” *Neurocomputing*, vol. 521, pp. 113–125, 2023.

[35] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” 2014, *arXiv:1312.6114*.

[36] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” in *Proc. 31th Int. Conf. Mach. Learn.*, 2014, pp. 1278–1286.

[37] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[38] S. Zhao, Z. Du, J. Chen, Y. Zhang, J. Tang, and P. S. Yu, “Hierarchical representation learning for attributed networks,” in *Proc. IEEE 38th Int. Conf. Data Eng.*, 2022, pp. 1497–1498.

[39] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?,” 2018, *arXiv:1810.00826*.

[40] M. D. Donsker and S. R. S. Varadhan, “Asymptotic evaluation of certain Markov process expectations for large time,” *Commun. Pure Appl. Math.*, vol. 36, no. 2, pp. 183–212, 1975.

[41] P. D. Dobson and A. J. Doig, “Distinguishing enzyme structures from non-enzymes without alignments,” *J. Mol. Biol.*, vol. 330, no. 4, pp. 771–783, 2003.

[42] I. Schomburg et al., “Brenda, the enzyme database: Updates and major new developments,” *Nucleic Acids Res.*, vol. 32, no. Database-Issue, pp. 431–433, 2004.

[43] N. Wale, I. A. Watson, and G. Karypis, “Comparison of descriptor spaces for chemical compound retrieval and classification,” *Knowl. Inf. Syst.*, vol. 14, no. 3, pp. 347–375, 2008.

[44] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi, “CurGraph: Curriculum learning for graph classification,” in *Proc. Web Conf.*, 2021, pp. 1238–1248.

[45] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2016, *arXiv:1609.02907*.

Xiaolin Li received the BS degree from Shandong Agricultural University, in 2016. He is currently working toward the MS degree with Fujian Normal University, China. His research interests include network security, wireless networks and communication, deep learning.

Qikui Xu received the bachelor’s degrees in computer science and biochemistry from the University of Sydney, Australia, in 2019. He is currently working toward the graduate degree with Fudan University and West Lake University, and his research interests include complex network modeling, bioinformatics, and biochemistry.

Zhenyu Xu received the BE degree in network engineering from the Anhui University of Technology, China, in 2019. He is currently working toward the master’s degree. His research interest include network and information security.

Hongyan Zhang received the BS degree from the Shandong University of Science and Technology, in 2003, and the MS degree from Fujian Normal University, in 2008. She is currently working toward the PhD degree with Fujian Normal University, China. Her research interests include network security and wireless networks and communication.

Li Xu received the PhD degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2004. He is currently a professor and doctoral supervisor with the College of Computer and Cyber Security, Fujian Normal University, Fuzhou, China. He is also the dean with the College of Computer and Cyber Security, Fujian Normal University. His research interests include network and information security, wireless networks and communication, intelligent information in communication networks. He has authored or coauthored more than 150 papers in international journals and conferences, including the *IEEE Transactions on Computer*, *IEEE Transactions on Reliability*, *IEEE Transactions on Parallel and Distributed Systems*, *Information Science*.