Structural Entropy Guided Graph Hierarchical Pooling

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

Abstract

Following the success of convolution on non-Euclidean space, the corresponding pooling approaches have also been validated on various tasks regarding graphs. However, because of the fixed compression quota and stepwise pooling design, these hierarchical pooling methods still suffer from local structure damage and suboptimal problem. In this work, inspired by structural entropy, we propose a hierarchical pooling approach, SEP, to tackle the two issues. Specifically, without assigning the layer-specific compression quota, a global optimization algorithm is designed to generate the cluster assignment matrices for pooling at once. Then, we present an illustration of the local structure damage from previous methods in the reconstruction of ring and grid synthetic graphs. In addition to SEP, we further design two classification models, SEP-G and SEP-N for graph classification and node classification, respectively. The results show that SEP outperforms state-of-the-art graph pooling methods on graph classification benchmarks and obtains superior performance on node classifications.

1. Introduction

Chasing the great success of deep learning in natural language processing and images, plenty of research efforts have been devoted to the adoption of neural networks in tasks without data on the Euclidean domain, i.e., in graphs (Kipf & Welling, 2017; Veličković et al., 2018). Thus, recent years, graph neural networks (GNNs) become ubiquitous within deep learning for graphs, and have obtained great accomplishments in various domains, such as node classification (Kipf & Welling, 2017), link prediction (Zhang & Chen, 2018) and graph classification (Xu et al., 2019). In these works, a key direction is the convolutional mechanism of GNNs (Wu et al., 2019; Zhu & Koniusz, 2020; Wu et al., 2022), which aims to learn the structural information in graphs. Meanwhile, another research direction in GNNs is the pooling mechanism, which follows the customs in CNN models that compress a set of nodes into a compact representation (Lee et al., 2019; Bianchi et al., 2020).

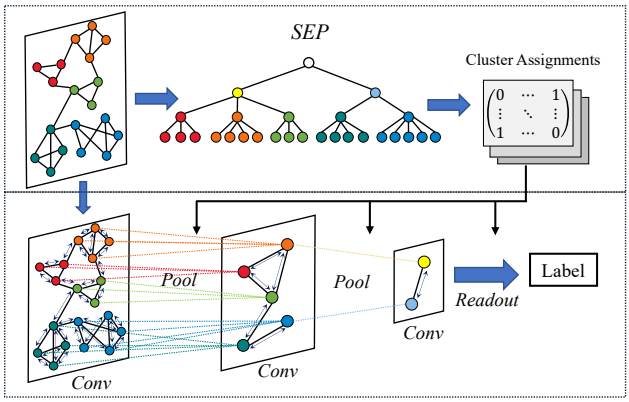


Figure 1: **Architecture of proposed SEP operator combined with graph neural network.** The graph neural network consists of message passing layers and hierarchical pooling layers. A separate algorithm is proposed for cluster assignments generation.

Besides the simplest pooling methods, that sum or average all nodes, various well-designed pooling approaches have been proposed to aggregate the node information in a hierarchical manner. However, despite the effectiveness of these methods on various tasks, there are still many issues that hinder the development of GNNs. First, pooling methods based on node drop, like TopKPool (Gao & Ji, 2019), SAGPool (Lee et al., 2019) and ASAP (Ranjan et al., 2020), unnecessarily cut nodes based on designed ranking strategy in each pooling layer, resulting in information loss (Baek et al., 2021). Although other methods based on node clustering avoid this issue (Ying et al., 2018; Bianchi et al., 2020), the damage on the graph local structure still can not be prevented due to the artificially specified node compression quota (Gao & Ji, 2019; Ranjan et al., 2020), which also

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

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

exists in the node drop methods [1]. Furthermore, the cluster assignments produced by these works only rely on the topology of graph in the current layer without any consideration of the relationships among pooling layers. This would probably lead to suboptimal results in their tasks. Thus, a pooling operation that is globally optimized and has natural node partition is preferred.

In this paper, we present a hierarchical pooling method, termed *SEP*, to address the above two issues that hinder the development of GNNs (Figure 1). Specifically, inspired by structural entropy (Li & Pan, 2016), a metric designed to assess the graph structural information, the essential structure of a graph can be decoded by this metric as a measure of the complexity of its hierarchical structure. In particular, the cluster assignments for hierarchical pooling can be directly obtained from the proposed algorithm for structural entropy minimization. Note that, the proposed algorithm is globally optimized and free from learning, which means the cluster assignments will be generated together to avoid the suboptimal problem. Moreover, the algorithm does not rely on a fixed layer-specific compression quota but the number of compression layers, which would help retain the local structure of graphs.

Before the validation of SEP on classification benchmarks, we first present an illustration of the damage caused by previous hierarchical pooling methods on local structure of graphs. With seven benchmarks from bioinformatics and social networks, we then experimentally validate the effectiveness of SEP on tasks regarding graph classification, and conclude that SEP surpasses the state-of-the-art (SOTA) hierarchical pooling methods on most benchmarks especially on the social network datasets. Besides the superior performance on global attribute discerning, we further evaluate SEP on node classification tasks to better validate its capability of information retaining within the process of pooling. The results show that SEP outperforms the model with the same architecture (i.e., g-U-Nets) and most baselines. To sum up, our contributions are listed as follows:

- We uncover two crucial issues in previous hierarchical pooling works that hinder the development of GNNs, including the local structure damage and suboptimal problem because of the fixed compression quota and stepwise pooling design.
- Through the introduction of the structural information theory, we present a novel hierarchical pooling approach, termed SEP, to address the unveiled issues.
- We extensively validate SEP on graph reconstruction, graph classification, and node classification tasks, in which outperformances are observed in comparison the SOTA hierarchical pooling methods.

[1]The clustering-based pooling methods require the fixed number of clusters and the node drop methods require the fixed node compression ratio.

2. Related Work

Hierarchical pooling. To pursue better generalization and performance, pooling operations have been adopted to amplify the receptive fields and reduce the input sizes. Several designs are proposed from the angle of selecting the most important k nodes from the original graph to organize a new one, such as TopKPool (Gao & Ji, 2019), SAGPool (Lee et al., 2019) and ASAP (Ranjan et al., 2020). Though efficient, this node-drop design would result in information loss and isolated subgraphs, which will deteriorate the performance of GNNs. Thus, another design based on node clustering emerges and avoids this issue, including DiffPool (Ying et al., 2018) and MinCutPool (Bianchi et al., 2020), in which the nodes of original graph are merged into a bunch of clusters. Although preventing information loss, the damage on graph local structures still exists because of the fixed node compression quota.

Structural entropy. Information entropy stems from the demand for information measurement in communication systems (Shannon, 1948). Correspondingly, to measure the information in graphs, structural entropy was proposed and used to evaluate the complexity of the hierarchical structure of a graph (Li & Pan, 2016), which is also a natural node clustering method for graphs. Furthermore, two- and three-dimensional structural entropy, which measure the complexity of two- and three-level hierarchical structures, respectively, have been applied in medicine (Li et al., 2016b), bioinformatics (Li et al., 2018), and the security of networks (Li et al., 2016a). In the light of this global measurement of graph information, structural entropy can be used to decode the essential structure of graphs, which further sparks us the yielding of SEP to address the two issues that impede the development of GNNs.

3. Proposed Method

In this section, under the guidance of structural entropy, we present our key idea and an algorithm for the cluster assignments construction. Then, we design a GNN model, which has several convolutional layers and pooling layers, to learn global representations for graph classification. Furthermore, we develop another model, which is made up of additional convolutional layers and unpooling layers, to obtain local representations for node classification. Before elaborating on them, we first show some notations.

3.1. Preliminaries

A graph G can be represented as a multi-tuple $(\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $|\mathcal{V}| = n$ is the node set, $|\mathcal{E}| = m$ is the edge set,

and $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the feature matrix for n nodes with d-dimensional feature vector. The topology structure of graph G can be found in its adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$.

Graph neural networks. In this work, we select Graph Convolutional Network (i.e., GCN (Kipf & Welling, 2017)) as the convolutional layers of our models. GCN and its variants have achieved excellent performance in different kinds of tasks regarding graphs. There is no doubt that our proposed pooling operator can also work with other GNNs like GAT (Veličković et al., 2018) and GIN (Xu et al., 2019). This will be discussed in the experimental section. For a stacked graph neural networks, the i-th convolutional layer in the form of GCN can be formally written as:

$$H_{i+1} = ReLU(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}H_iW_i), \tag{1}$$

where $ReLU$ is a non-linear activation function, $W_i \in \mathbb{R}^{h\times h}$ is the trainable matrix for this layer, $H_{i+1} \in \mathbb{R}^{n\times h}$ is the output of this layer and $H_0 = \mathbf{X}$. In particular, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ denotes the adjacency matrix with self-loops, and $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$. In the process of hyper-parameter tuning, we fix the same hidden dimension for all layers.

Hierarchical pooling. In general, hierarchical pooling is a graph coarsening process to dig out a subset of representative nodes and form a new graph. Let $\mathbf{S}_i \in \mathbb{R}^{n_{i+1}\times n_i}$ denote the cluster assignment matrix at i-th pooling layer, where n_i and n_{i+i} are the number of nodes before and after graph coarsening. The new adjacency matrix and node feature matrix after pooling are calculated by the next equations:

$$\mathbf{A}_{i+1} = \mathbf{S}_i\mathbf{A}_i\mathbf{S}_i^\top; \quad \mathbf{P}_{i+1} = \mathbf{S}_iH_i, \tag{2}$$

where $\mathbf{A}_i \in \mathbb{R}^{n_i\times n_i}$ is the adjacency matrix at the i-th layer and $H_i \in \mathbb{R}^{n_i\times h}$ refers to the node feature matrix produced by the i-th graph convolutional layer. Specifically, according to the cluster assignments $\mathbf{S}_i$, $\mathbf{P}_{i+1}$ receives the node hidden features H_i and merges these features to refine the initial representations for the n_{i+1} clusters in the novel graph. Correspondingly, $\mathbf{A}_{i+1}$ employs the node connectivities $\mathbf{A}_i$ to weave a more delicate origination among n_{i+1} clusters.

3.2. Cluster Assignments via Structural Entropy Minimization

Here, we present our methodology for hierarchical pooling. As claimed above, although plenty of works have been presented for graph coarsening based on heuristics or theories, little attention has been paid to the relationships among pooling layers. These studies only produce the cluster assignments based on the graph in the current layer. Meanwhile, in various tasks regarding graphs, there are graphs constructed from specific domain (e.g., social networks) or weaved by human (e.g., synthetic graphs), which are usually not optimal for GNNs because of noisy information. It is thus vital to eliminate such noisy structure from graphs in the process of cluster assignments generation. In this paper, inspired by structural entropy (Li & Pan, 2016), we propose a novel hierarchical pooling approach, denoted as SEP, to address the aforementioned issues in previous works. Besides the measurement of graph information, structural entropy can also be used to decode the hierarchical structure of a given graph as a metric of the complexity of its underlying essential structure. Thus, through structural entropy minimization, the hierarchical structure of a graph can be decoded into a corresponding coding tree, in which disturbance derived from noise or stochastic variation can be minimized (Li et al., 2018). We believe an effective structural entropy minimization algorithm could uncover the connections among hierarchical pooling layers and eliminate the noisy structure in graphs.

Based on the definition in (Li & Pan, 2016), let a two-tuple $(\mathcal{V}, \mathcal{E})$ be a graph G, the formal equation of the structural entropy for G on coding tree T can be written as:

$$\mathcal{H}^T(G) = -\sum_{v_t\in T}\frac{g_{v_t}}{vol(\mathcal{V})}\log\frac{vol(v_t)}{vol(v_t^+)}, \tag{3}$$

where v_t is a nonroot node in T that can also be viewed as a node subset $\subset \mathcal{V}$ according to its leaf node partition in T, v_t^+ is the parent of v_t, g_{v_t} refers to the number of edges with an endpoint in the leaf node partition of v_t, and $vol(\mathcal{V})$ and $vol(v_t)$ are the sums of degrees of leaf nodes in $\mathcal{V}$ and v_t, respectively. The minimum entropy realized by the optimal coding tree T is the structural entropy of G, which follows this target equation: $\mathcal{H}(G) = \min_{\forall T}\{\mathcal{H}^T(G)\}$. According to the definition of structural entropy, we know that the coding tree is a natural hierarchical division for graphs, and the connections among different layers are established for the purpose of structural entropy minimization. Furthermore, the local structure in graphs will be retained because we do not need to allocate layer-specific node compression quotas.

Besides the optimal coding tree that realizes the minimized structural entropy, in most cases, a natural coding tree with a certain height is preferred, because we only need a few fixed times of graph coarsening for most tasks (Gao & Ji, 2019; Baek et al., 2021). In this context, the k-dimensional structural entropy of G is proposed to decode the optimal coding tree with a fixed height k:

$$\mathcal{H}^{(k)}(G) = \min_{\forall T:\text{Height}(T)=k}\{\mathcal{H}^T(G)\}. \tag{4}$$

In this paper, under the guidance of k-dimensional structural entropy, we aim to investigate the solution for decrypting the coding tree with a certain height k. Firstly, we define three functions.

Definition 3.1. Let T be any coding tree for graph $G = (\mathcal{V}, \mathcal{E})$, v_r is the root node of T and $\mathcal{V}$ are the leaf nodes

of T. Given any two nodes (v_i, v_j) in T, in which $v_i \in v_r.children$ and $v_j \in v_r.children$.

Define a function $\text{MERGE}_T(v_i, v_j)$ for T to insert a new node v_ε between v_r and (v_i, v_j):

$$v_\varepsilon.children \leftarrow v_i; \tag{5}$$

$$v_\varepsilon.children \leftarrow v_j; \tag{6}$$

$$v_r.children \leftarrow v_\varepsilon; \tag{7}$$

Definition 3.2. Following the setting in Definition 3.1, given any two nodes (v_i, v_j), in which $v_i \in v_j.children$.

Define a function $\text{REMOVE}_T(v_i)$ for T to remove v_i from T and merge $v_i.children$ to $v_j.chileren$:

$$v_j.children \leftarrow v_i.children; \tag{8}$$

Definition 3.3. Following the setting in Definition 3.1, given any two nodes (v_i, v_j), in which $v_i \in v_j.children$ and $|Heigth(v_j) - Height(v_i)| > 1$.

Define a function $\text{FILL}(v_i, v_j)$ for T to insert a new node v_ε between v_i and v_j:

$$v_\varepsilon.children \leftarrow v_i; \tag{9}$$

$$v_j.children \leftarrow v_\varepsilon; \tag{10}$$

Based on the three defined functions, a greedy algorithm to compute the coding tree with a certain height k via structural entropy minimization can be found in Algorithm 1. Specifically, a full-height binary coding tree is first generated from bottom to top. In this stage, two child nodes of root are merged to form a new division in each iteration, which aims to maximize the structural entropy reduction. In the second stage, in order to satisfy a fixed number of graph coarsening, we need squeeze the previous full-height binary coding tree by dropping nodes. Each time, we select an inner-node from T, which makes T have the minimized structural entropy after removing this node. At the end of the second stage, we have already obtained a coding tree with a certain height k under the guidance of structural entropy. However, there may be nodes that do not have immediate successor in the next layer because of cross-layer links, which will cause node missing when realizing hierarchical pooling based on such a coding tree. Therefore, we need perform the third stage to ensure the integrity of information transmission between layers, and also need not interfere with the structural entropy of G on coding tree T (see Proposition 3.4). Finally, a coding tree T for the given graph G can be obtained, where $T = (\mathcal{V}^T, \mathcal{E}^T)$, $\mathcal{V}^T = (\mathcal{V}_0^T, \ldots, \mathcal{V}_k^T)$ and $\mathcal{V}_0^T = \mathcal{V}$. In addition, the cluster assignment matrices can also be obtained from $\mathcal{E}^T$, that is, $\mathbb{S} = (\mathbf{S}_1, \ldots, \mathbf{S}_k)$.

Proposition 3.4. *Let T be a coding tree after the second stage of Algorithm 1, and given two adjacent nodes (v_i, v_j) in T, in which $v_i \in v_j.children$ and $|Heigth(v_j) - Height(v_i)| > 1$. Then, $\mathcal{H}^T(G) = \mathcal{H}^{T_{FILL(v_i, v_j)}}(G)$.*

Algorithm 1 Coding tree with height k via structural entropy minimization

Input: a graph $G = (\mathcal{V}, \mathcal{E})$, a positive integer $k > 1$
Output: a coding tree T with height k

```
Generate a coding tree T with a root node v_r and all nodes in V as leaf nodes;
// Stage 1: Bottom to top construction;
while |v_r.children| > 2 do
    Select v_i and v_j from v_r.children, conditioned on
    argmax_{(v_i,v_j)} {H^T(G) − H^{T_MERGE(v_i,v_j)}(G)};
    MERGE(v_i, v_j);
end while
// Stage 2: Compress T to the certain height k;
while Height(T) > k do
    Select v_i from T, conditioned on
    argmin_{v_i} {H^{T_REMOVE(v_i)}(G) − H^T(G)|
                  v_i ≠ v_r & v_i ∉ V};
    REMOVE(v_i);
end while
// Stage 3: Fill T to avoid cross-layer links;
for v_i ∈ T do
    if |Height(v_i.parent) − Height(v_i)| > 1 then
        FILL(v_i, v_i.parent);
    end if
end for
return T;
```

Proof. Equation 3 shows that the structural entropy of graph G on T is the summation of $\mathcal{H}_v^T = -\frac{g_v}{vol(\mathcal{V})} \log \frac{vol(v)}{vol(v^+)}$ for all nonroot nodes in T. That is, $\mathcal{H}^T(G) = \mathcal{H}_{v_i}^T + \mathcal{H}_{v_j}^T + \ldots$ and $\mathcal{H}^{T_F}(G) = \mathcal{H}_{v_i'}^{T_F} + \mathcal{H}_{v_\varepsilon}^{T_F} + \mathcal{H}_{v_j'}^{T_F} + \ldots$, denote $T_F = T_{\text{FILL}(v_i, v_j)}$ for simplicity and (v_i', v_j') corresponds to (v_i, v_j) after FILL. According to Equation 3, we have:

$$\begin{aligned}
\mathcal{H}_{v_i'}^{T_F} &= -\frac{g_{v_i'}}{vol(\mathcal{V})} \log \frac{vol(v_i')}{vol(v_i'^+)} \\
&= -\frac{g_{v_i'}}{vol(\mathcal{V})} \log \frac{vol(v_i')}{vol(v_\varepsilon)} \\
&= 0 \quad with \quad vol(v_i') = vol(v_\varepsilon),
\end{aligned} \tag{11}$$

$$\begin{aligned}
\mathcal{H}_{v_\varepsilon}^{T_F} &= -\frac{g_{v_\varepsilon}}{vol(\mathcal{V})} \log \frac{vol(v_\varepsilon)}{vol(v_\varepsilon^+)} \\
&= -\frac{g_{v_i}}{vol(\mathcal{V})} \log \frac{vol(v_i)}{vol(v_j)} \\
&= \mathcal{H}_{v_i}^T,
\end{aligned} \tag{12}$$

$$\begin{aligned}
\mathcal{H}_{v_j'}^{T_F} &= -\frac{g_{v_j'}}{vol(\mathcal{V})} \log \frac{vol(v_j')}{vol(v_j'^+)} \\
&= -\frac{g_{v_j}}{vol(\mathcal{V})} \log \frac{vol(v_j)}{vol(v_j^+)} \\
&= \mathcal{H}_{v_j}^T.
\end{aligned} \tag{13}$$

Thus, we have $\mathcal{H}^{T}(G) = \mathcal{H}^{T_{\text{FILL}(v_i,v_j)}}(G)$. □

Complexity analysis. The runtime complexity of Algorithm 1 is $O(2n + h_{max}(m \log n + n))$, and h_{max} is the height of coding tree T after the first stage. Meanwhile, since coding tree T tends to be balanced in the process of structural entropy minimization, h_{max} will be around $\log n$. Furthermore, graph generally has more edges than nodes, i.e., $m \gg n$, thus the runtime of Algorithm 1 almost scales linearly in the number of edges.

3.3. Graph Neural Network for Graph Classification

In this subsection, we present the architecture based on SEP for graph classification, and name it *SEP-G*. As shown in Figure 2, the hierarchical pooling architecture follows the setting in previous pooling studies (Lee et al., 2019; Bianchi et al., 2020; Baek et al., 2021), which consists of three blocks and each block has a GCN layer and a SEP layer. For the graph-level representation, the outputs after each block are summarized and then fed to a prediction layer for classification. Based on Equations 1 and 2, the graph representation with SET-G can be formally written as:

$$h_G = Concat(Readout(SEP_i(GCN_i(H_i, \mathbf{A}_i), \mathbf{S}_i)) \quad |\forall i \in \{1, 2, 3\}). \tag{14}$$

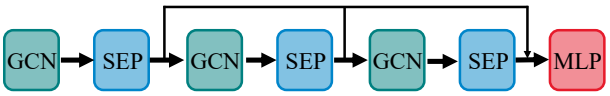


Figure 2: **The SEP-G architecture for graph classification.** Following the design of previous works in hierarchical pooling, the architecture is comprised of three GCN layers and each is followed by corresponding SEP layer.

Permutation invariance. In graph classification, it is important to ensure the permutation invariance of designed graph neural network. In our proposed model for graph classification, there are two main components, that is, GCN layer and SEP layer. The permutation invariance of GCN layer has been confirmed by previous works (Ying et al., 2018; Ma et al., 2019). Thus, the SEP layer should be invariant with permutations.

Proposition 3.5. *Given a permutation matrix* $\mathcal{P} \in \{0, 1\}^{n \times n}$*, then* $SEP(A, H) = SEP(\mathcal{P}A\mathcal{P}^{\top}, \mathcal{P}H)$ *(i.e., SEP is permutation invariant).*

Proof. The cluster assignments are derived from the coding tree via Algorithm 1, which is a traversal algorithm that does not depend on the order of nodes. Therefore, the generated assignments $\mathbb{S}$ will not change with any permutation. In addition, we know that the permutation matrix is orthogonal, thus $\mathcal{P}\mathcal{P}^{\top} = I$ with Equation 2 finishes the proof. □

3.4. Graph Neural Network for Node Classification

Beyond the functionality of cluster assignments to convert graphs into high-level representations, we can also adopt the same matrix $\mathbf{S}_i$ to unpool the compressed graph representation H_i and structure A_i to the original space by:

$$\mathbf{A}_{i+1} = \mathbf{S}_i^{\top} \mathbf{A}_i \mathbf{S}_i; \quad \mathbf{P}_{i+1} = \mathbf{S}_i^{\top} H_i. \tag{15}$$

In this context, we present an illustration of the architecture for node classification in Figure 3, and we call it *SEP-N*. SEP-N is an encoder-decoder network analogous to the design of g-U-nets (Gao & Ji, 2019). In the encoder, several down-sampling blocks are applied to encode higher-order features. Each block consists of a GCN layer and a SEP layer like our SEP-G. In the decoder, we employ the consistent number of decoding components. Thus, we can see the same GCN layer but with a SEP-U layer for unpooling in the decoder block. The skip-connections linking corresponding encoders and decoders are also adopted to enable spatial information transmission for better performance. Finally, a GCN layer is used to perform final predictions.

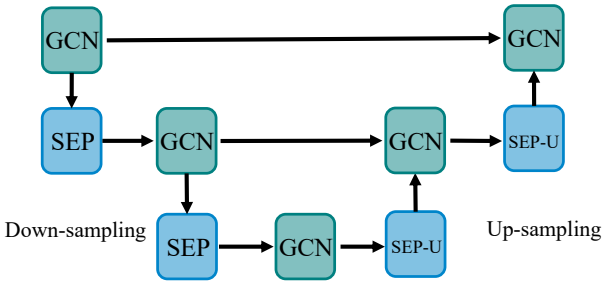


Figure 3: **The SEP-N architecture for node classification.** There are two encoder and two decoder blocks and each block is composed of a GCN layer and a pooling (unpooling) layer. Skip connection links the same-level encoder and decoder to enhance spatial feature transmission.

4. Experiments

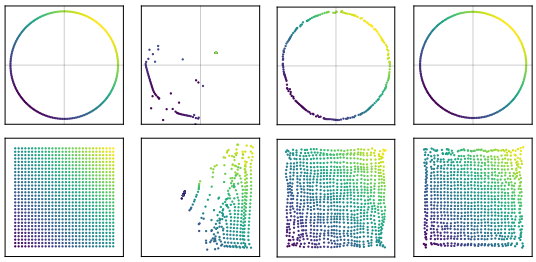
In this section, we describe the experiment setup for graph classification and node classification in detail, and validate the effectiveness of our proposed methods, SEP-G and SEP-N, in several corresponding benchmarks [2].

[2]The implementation of Algorithm 1, SEP-G and SEP-N can be found at https://github.com/Wu-Junran/SEP.

4.1. Graph Reconstruction

Before the presentation of experiments regarding two major classification tasks in GNNs, we first employ a graph reconstruction experiment, which quantifies the structural information retained by pooling layer, to directly reveal the damage caused by previous hierarchical pooling methods to graph's local structures.

Configuration. An autoencoder is trained to reconstruct the input graph with pooling and unpooling layers. The learning objective is to minimize the mean square error (MSE) between input features $\mathbf{X}$ and output features $\mathbf{X}^r$, i.e., $min\|\mathbf{X} - \mathbf{X}^r\|^2$. For configuration, we employ the Synthetic graphs (Bianchi et al., 2020), including a ring and a grid that the input features are the coordinates of nodes in a 2-D Euclidean space. Detailed experiment configuration and model description can be found in Appendix A.1.



(a) Original (b) TopKPool (c) MinCutPool (d) SEP

Figure 4: Illustration of local structure damage from node drop and clustering methods in reconstruction of ring and grid synthetic graphs.

Reconstruction results. Figure 4 shows the original graphs and the reconstructed graphs by model with various pooling methods. We select TopKPool to represent methods with node drop design and MinCutPool to represent methods with node clustering design [3]. We first notice the reconstructed results of TopKPool, where the basic shape of original graphs can not even be identified. This confirms that node drop methods will lead to severe information missing and implies the poor performance of node drop methods in graph classification. On the other hand, MinCutPool indeed retains the basic shape of original graphs. However, we can still see the significant distortion in the edge of ring and the center of grid, which represent the key structures of ring and grid, and this validates the assumption that the local structure in graphs would be ruined by the artificially specified node compression quota. On the contrary, SEP almost reconstructs the ring and retains the essential structure in grid center which suggests that our pooling method obtains the key structural information of original graphs.

[3]Reconstruction results of all hierarchical pooling baselines can be found in Appendix A.1.

4.2. Graph Classification

Graph classification aims to label a given graph G with the maximum probability among several seed categories. To this end, we need learn the high-level representation from its component nodes, which enables the final classifier evaluate the likelihood of category that the graph belongs to.

Datasets. Seven benchmarks for graph classification are selected from TU datasets (Morris et al., 2020). Specifically, we employ three social network datasets, including IMDB-BINARY, IMDB-MULTI, and COLLAB; and four bioinformatics datasets, including MUTAG, PROTEINS, D&D and NCI1. Table 1 summarizes the characteristics of the seven employed datasets, and more detailed descriptions can be found in the Appendix A.2.

Baselines. We first employ two popular backbones in GNNs for comparison, that is, GCN (Kipf & Welling, 2017) and GIN (Xu et al., 2019). Then, we adopt the next five hierarchical pooling methods as baselines: DiffPool (Ying et al., 2018), SAGPool(H) (Lee et al., 2019), TopKPool (Gao & Ji, 2019), ASAP (Ranjan et al., 2020), and MinCutPool (Bianchi et al., 2020). Besides various hierarchical pooling methods, plenty of efforts have also been devoted to global pooling for graph classification. Thus, we also select the following five global pooling techniques for comparison: Set2Set (Vinyals et al., 2016), SortPool (Zhang et al., 2018), SAGPool(G) (Lee et al., 2019), StructPool (Yuan & Ji, 2020), and GMT (Baek et al., 2021).

Configurations. Following (Xu et al., 2019; Lee et al., 2019), 10-fold cross-validation is conducted, and we present the average accuracies achieved to validate the performance of SEP-G in graph classification. In addition, the initial feature inputs is in line with the fair comparison setting in (Errica et al., 2020). Additional details about experiment setup can be found in the Appendix A.2.

Classification results. The classification accuracies of SEP-G and other baselines are shown in Table 1, and we can see that our method consistently achieves better or competitive performance as compared to these SOTA methods. In particular, SEP-G obtains a unified improvement in social network datasets, which differs from the performance in bioinformatics datasets. This performance divergence may be because SEP only relies on the network structure for hierarchical pooling, while the structural information in social network datasets is more redundant than that in bioinformatics datasets (Centola, 2010). It is worth noting

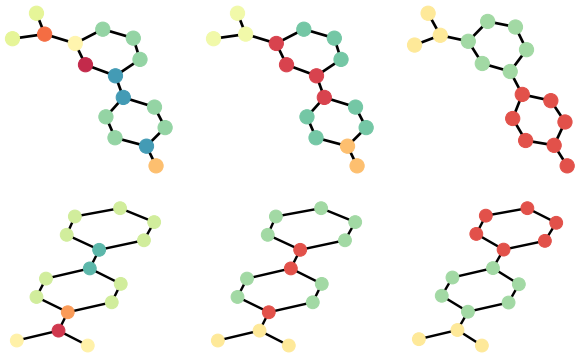
Table 1: **Graph classification accuracies on seven benchmarks (%).** The shown accuracies are mean and standard deviation over 10 different runs. We highlight the best results.

		Social Network			Bioinformatics			
		IMDB-BINARY	IMDB-MULTI	COLLAB	MUTAG	PROTEINS	D&D	NCI1
# Graphs		1,000	1,500	5,000	188	1,113	1,178	4,110
# Classes		2	3	3	2	2	2	2
Avg. # Nodes		19.8	13.0	74.5	17.9	39.1	284.3	29.8
Backbones	GCN	73.26±0.46	50.39±0.41	80.59±0.27	69.50±1.78	73.24±0.73	72.05±0.55	76.29±1.79
	GIN	72.78±0.86	48.13±1.36	78.19±0.63	81.39±1.53	71.46±1.66	70.79±1.17	**80.00±1.40**
Global Pooling	Set2Set	72.90±0.75	50.19±0.39	79.55±0.39	69.89±1.94	73.27±0.85	71.94±0.56	68.55±1.92
	SortPool	72.12±1.12	48.18±0.83	77.87±0.47	71.94±3.55	73.17±0.88	75.58±0.72	73.82±1.96
	SAGPool(G)	72.16±0.88	49.47±0.56	78.85±0.56	76.78±2.12	72.02±1.08	71.54±0.91	74.18±1.20
	StructPool	72.06±0.64	50.23±0.53	77.27±0.51	79.50±0.75	75.16±0.86	78.45±0.40	78.64±1.53
	GMT	73.48±0.76	50.66±0.82	80.74±0.54	83.44±1.33	75.09±0.59	**78.72±0.59**	76.35±2.62
Hierarchical Pooling	DiffPool	73.14±0.70	51.31±0.72	78.68±0.43	79.22±1.02	73.03±1.00	77.56±0.41	62.32±1.90
	SAGPool(H)	72.55±1.28	50.23±0.44	78.03±0.31	73.67±4.28	71.56±1.49	74.72±0.82	67.45±1.11
	TopKPool	71.58±0.95	48.59±0.72	77.58±0.85	67.61±3.36	70.48±1.01	73.63±0.55	67.02±2.25
	ASAP	72.81±0.50	50.78±0.75	78.64±0.5	77.83±1.49	73.92±0.63	76.58±1.04	71.48±0.42
	MinCutPool	72.65±0.75	51.04±0.70	80.87±0.34	79.17±1.64	74.72±0.48	78.22±0.54	74.25±0.86
	SEP-G	**74.12±0.56**	**51.53±0.65**	**81.28±0.15**	**85.56±1.09**	**76.42±0.39**	77.98±0.57	78.35±0.33

that there are not any pooling methods suppress GIN in NCI1, or, put differently, pooling methods also do not show unified promotion in comparison with backbones. Considering the recent finding that message-passing is the crucial mechanism in graph classification (Mesquita et al., 2020), this phenomenon may not be so disappointing.

Variants of SEP-G. As discussed in Section 3.1, besides GCN, our proposed pooling operator can also work with other GNNs like GAT and GIN. Here, we delve deeper into the collaboration ability of our pooling method with other GNNs. Specifically, we employ the ChebNet (Defferrard et al., 2016), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018) and GIN (Xu et al., 2019). The classification results are shown in Table 2. As can be seen, the overall superior performance is obtained by these variants, which suggests the effectiveness and kindness of SEP. Notably, a better performance on IMDB-MULTI, PROTEINS, DD and NCI1 is obtained by SEP with GAT, which further indicates the huge potential of SEP in collaboration with other SOTA backbones.

Visualization case study. To better demonstrate the effectiveness of SEP on essential structure preserving, we present the clustering results of DiffPool, MinCutPool, and SEP on samples from the MUTAG dataset after the first graph coarsening. As shown in Figure 5, DiffPool and MinCutPool severely damage the essential structures of the two compounds, in which the ring structures of two molecular formulas are arbitrarily torn into several pieces. Fortunately, SEP manages to take good care of these vital structures in the process of cluster assignment generation, and this shows the effectiveness of SEP in issues addressing.



(a) DiffPool (b) MinCutPool (c) SEP

Figure 5: Essential structure preserving on MUTAG.

4.3. Node Classification

Node classification is another important task regarding graphs in GNNs, which aims to label each node in a given graph G. Here, we conduct corresponding experiments on ubiquitous benchmarks to validate the effectiveness of our proposed SEP-N.

Datasets. We evaluate SEP-N under the transductive learning setting, which includes three datasets Cora, Citeseer and Pubmed (Sen et al., 2008). The three benchmarks are constructed on the connections of document citations, which means nodes are documents and edges are citation links. The node features are different among three datasets. Specifically, the input features of Cora and Citeseer are one-

Table 2: **Graph classification accuracies of SEP-G with various backbones.** The default backbone is GCN, and we denote it as SEP-G-GCN for a better illustration.

Variants	Social Network			Bioinformatics			
	IMDB-BINARY	IMDB-MULTI	COLLAB	MUTAG	PROTEINS	DD	NCI1
SEP-G-GCN	**74.12±0.56**	51.53±0.65	**81.28±0.15**	**85.56±1.09**	76.42±0.39	77.98±0.57	78.35±0.33
SEP-G-GIN	73.37±0.95	51.81±0.98	79.18±0.60	83.22±1.28	74.77±1.42	75.98±1.15	76.59±1.65
SEP-G-GAT	73.24±0.81	**51.87±0.45**	79.26±0.39	84.45±1.81	**76.72±0.92**	**78.07±0.74**	**78.43±1.07**
SEP-G-ChebNet	73.72±0.42	50.84±0.68	80.73±0.43	83.25±1.13	74.67±0.75	76.69±0.71	77.68±0.97
SEP-G-GraphSAGE	73.14±0.87	50.43±1.31	79.88±0.58	83.75±1.43	75.26±0.86	77.95±0.55	77.65±1.21

hot embedding of words in each document, while the node features of Pubmed are the TF-IDF weighted word vectors. Following the experiment setup in previous works (Kipf & Welling, 2017; Gao & Ji, 2019), the designated training/validation/testing splits on Cora, Citeseer and Pubmed are adopted, that is, training set has 20 nodes for each class, validation set has 500 nodes and testing set has 1,000 nodes. Table 3 shows the dataset statistics.

Baselines. In addition to g-U-nets (Gao & Ji, 2019) that has the same encoder-decoder design, we also include three SOTA backbones in GNNs: GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GIN (Xu et al., 2019). Moreover, other following works based on the three models are also employed for a comprehensive comparison, such as FastGCN (Chen et al., 2018), APPNP (Klicpera et al., 2019), MixHop (Abu-El-Haija et al., 2019), SGC (Wu et al., 2019), DGI (Velickovic et al., 2019), S^2GC (Zhu & Koniusz, 2020), and GCNII (Chen et al., 2020).

Configurations. For node classification tasks, we fix our SEP-N with two blocks of encoders and decoders as presented in Figure 3, and plan to obtain the coding tree for each dataset under the guidance of three-dimensional structural entropy. In particular, each block has a GCN layer followed by a SEP (or SEP-U) layer. Finally, a GCN layer is adopted to make final prediction. There is no doubt that skip connections between layers are also established between encoder and decoder analogous to g-U-nets. Note that we add an additional linear layer after each SEP or SEP-U layer to learn more task-specific node representations. Dropout (Srivastava et al., 2014) with ReLU on feature matrices is applied to all layers in SEP-N, and L2 regularization is also adopted to avoid over-fitting. Detailed descriptions for experimental setup are shown in Appendix A.3.

Classification results. The accuracies of our proposed SEP-N and baselines on three benchmarks are shown in Table 3. In general, we can observe that the SEP-N does not achieve the best performance on any datasets, which are shown in S^2GC on Citeseer and GCNII on Cora and Pubmed. This problem may be attributed to the cluster assignments generation, which only relies on structural information, while the tasks regarding node classification are more dependent on the input features of nodes. However, SEP-N still obtains competitive results with only 5 GCN layers, which is much less than the design of S^2GC (16) and GCNII (64 for Cora, 32 for Citeseer, 16 for Pubmed). In particular, the accuracies of SEP-N on three datasets are consistently better than the three backbones (i.e., GCN, GAT and GIN). Furthermore, compared with g-U-nets, which is also designed with GCN and hierarchical pooling mechanism, our method also surpasses it on two out of three datasets (Cora and Pubmed) even with fewer learning blocks. Note that superior performance of SEP-N on Citeseer has also been achieved with different numbers of blocks, which we will describe it in the next experiments. In summary, we can confirm the effectiveness of the proposed hierarchical pooling operation in node representation learning.

Table 3: **Node classification accuracies on Cora, Citeseer, and Pubmed (%).** We highlight our results and those that are significantly higher than all other methods.

	Cora	Citeseer	Pubmed
# Nodes	2,708	3,327	19,717
# Edges	5,429	4,732	44,338
# Features	1,433	3,703	4,500
# Classes	7	6	3
GCN	81.4±0.4	70.9±0.5	79.0±0.4
GAT	83.3±0.7	72.6±0.6	78.5±0.3
GIN	77.6±1.1	66.1±0.9	77.0±1.2
FastGCN	79.8±0.3	68.8±0.6	77.4±0.3
APPNP	83.3±0.5	71.7±0.6	80.1±0.2
MixHop	81.8±0.6	71.4±0.8	80.0±1.1
DGI	82.5±0.7	71.6±0.7	78.4±0.7
SGC	81.0±0.03	71.9±0.11	78.9±0.01
S^2GC	83.5±0.02	**73.6±0.09**	80.2±0.02
GCNII	**85.5±0.5**	73.4±0.6	**80.3**±0.4
g-U-Nets	84.4±0.6	73.2±0.5	79.6±0.2
SEP-N	**84.8±0.4**	**72.9±0.7**	**80.2±0.8**

Ablation Study of Network Depth. Besides the other bunch of hyper-parameters in GNNs, the network depth, corresponding to the number of blocks, is also another cru-

Table 4: **Node classification accuracies with different network depths (%).** We highlight the best results with different depths for SEP-N and g-U-Nets.

	Cora		Citeseer		Pubmed	
Depth	g-U-Nets	SEP-N	g-U-Nets	SEP-N	g-U-Nets	SEP-N
1	–	84.3±0.6	–	**73.3±0.6**	–	78.9±0.6
2	82.6±0.6	**84.8±0.4**	71.8±0.5	72.9±0.7	79.1±0.3	**80.2±0.8**
3	83.8±0.7	84.5±0.3	72.7±0.7	72.1±0.6	79.4±0.4	79.5±0.5
4	**84.4±0.6**	83.6±0.6	**73.2±0.5**	72.1±0.2	**79.6±0.2**	78.5±0.3
5	84.1±0.5	83.9±0.5	72.8±0.6	72.4±0.6	79.5±0.3	79.8±0.7

cial setting in model construction. We, thus, delve deeper into the effect of model depth on node classification performance. We iteratively test SEP-N with various numbers of blocks $\in \{1, 2, 3, 4, 5\}$, and the results are shown in Table 4. As we mentioned above, SEP-N achieves superior performance on Citeseer with only one encoder and decoder. In particular, compared with g-U-Nets, we can see that the best performance of SEP-N on three datasets is achieved with at most 2 blocks, which once again proves the capacity of shallow networks in high-level feature encoding (Gao & Ji, 2019). Moreover, this scene also reveals that additional connection information among layers for cluster assignments generation is helpful for representation learning, as well as benefits model optimization that better performance with fewer parameters and computations.

5. Conclusions

In this paper, we develop an optimization algorithm to address several limitations of existing hierarchical pooling approaches. In particular, under the guidance of structural entropy minimization, our pooling method, SEP, can not only capture the connectivities among pooling layers but also fix the problem of destroying local structure due to the hyper-parameter for node compression. Based on the proposed SEP, we introduce two learning models, SEP-G and SEP-N, for graph classification and node classification, respectively. Experimental results suggest that SEP-G achieves significant improvements on graph classification, and SEP-N obtains superior performance as compared to other GNNs on node classification tasks. An interesting direction for future work is shown in the combination of structural entropy and node features.

Acknowledgements

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

References

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., and Galstyan, A. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*, pp. 21–29. PMLR, 2019.

Baek, J., Kang, M., and Hwang, S. J. Accurate learning of graph representations with graph multiset pooling. In *ICLR*, 2021.

Bianchi, F. M., Grattarola, D., and Alippi, C. Spectral clustering with graph neural networks for graph pooling. In *ICML*, pp. 874–883. PMLR, 2020.

Centola, D. The spread of behavior in an online social network experiment. *Science*, 329(5996):1194–1197, 2010.

Chen, J., Ma, T., and Xiao, C. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *ICLR*, 01 2018.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *ICML*, pp. 1725–1735. PMLR, 2020.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeruIPS*, volume 29, pp. 3844–3852, 2016.

Errica, F., Podda, M., Bacciu, D., and Micheli, A. A fair comparison of graph neural networks for graph classification. In *ICLR*, 2020.

Gao, H. and Ji, S. Graph u-nets. In *ICML*, pp. 2083–2092. PMLR, 2019.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *NeruIPS*, pp. 1025–1035, 2017.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 02 2019.

Lee, J., Lee, I., and Kang, J. Self-attention graph pooling. In *ICML*, pp. 3734–3743. PMLR, 2019.

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

Li, A., Hu, Q., Liu, J., and Pan, Y. Resistance and security index of networks: structural information perspective of network security. *Scientific Reports*, 6:26810, 2016a.

Li, A., Yin, X., and Pan, Y. Three-dimensional gene maps of cancer cell types: Structural entropy minimisation principle for defning tumour subtypes. *Scientific Reports*, 6:20412, 2016b.

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

Ma, Y., Wang, S., Aggarwal, C. C., and Tang, J. Graph convolutional networks with eigenpooling. In *SIGKDD*, pp. 723–731, 2019.

Mesquita, D., Souza, A., and Kaski, S. Rethinking pooling in graph neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *NeurIPs*, volume 33, pp. 2220–2231. Curran Associates, Inc., 2020.

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

Ranjan, E., Sanyal, S., and Talukdar, P. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *AAAI*, volume 34, pp. 5470–5477, 2020.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Shannon, C. E. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. In *ICLR*, volume 2, pp. 4, 2019.

Vinyals, O., Bengio, S., and Kudlur, M. Order matters: Sequence to sequence for sets. In *ICLR*, 2016.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *ICML*, pp. 6861–6871. PMLR, 2019.

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

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

Ying, R., Morris, C., Hamilton, W. L., You, J., Ren, X., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In *NeurIPs*, volume 2018-Decem, pp. 4800–4810, 2018.

Yuan, H. and Ji, S. Structpool: Structured graph pooling via conditional random fields. In *ICLR*, 2020.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *NeurIPS*, volume 31, pp. 5165–5175, 2018.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.

Zhu, H. and Koniusz, P. Simple spectral graph convolution. In *ICLR*, 2020.

A. Experiment Setup

In this section, we introduce the experimental details about graph reconstruction, graph classification, and node classification tasks respectively.

A.1. Graph Reconstruction

Dataset. Graph reconstruction experiments with synthetic graphs represented in a 2-D Euclidean space, such as ring and grid structures. The node features of a graph consist of their location in a 2-D coordinate space, and the adjacency matrix indicates the connectivity pattern of nodes. The goal here is to restore all node locations from compressed features after pooling, with the intact adjacency matrix.

Figure A.1: The architecture for graph reconstruction.

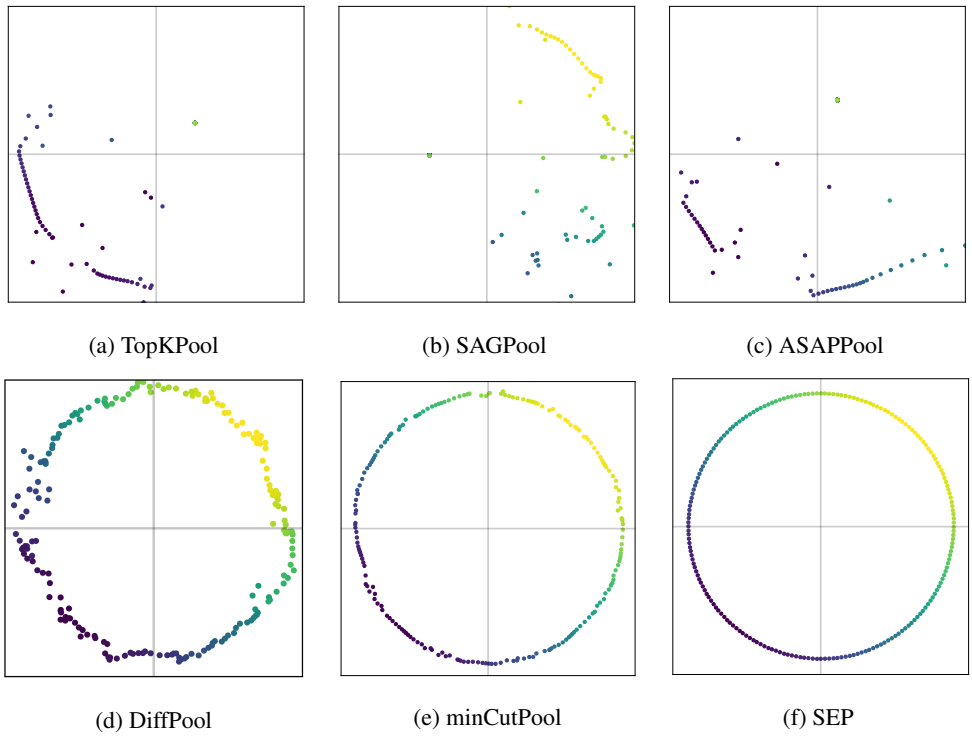
Implementation details. Following (Bianchi et al., 2020), we use the two message passing layers both right before the pooling operation and right after the unpooling operation. Also, both pooling and unpooling operations are performed once and sequentially connected, as illustrated in the Figure A.1. We compare our methods against both the node drop methods, including TopKPool (Gao & Ji, 2019), SAGPool (Lee et al., 2019) and ASAP (Ranjan et al., 2020), and node clustering methods, including DiffPool (Ying et al., 2018) and MinCutPool (Bianchi et al., 2020). For the node drop methods, we use the unpooling operation proposed in the graph U-net (Gao & Ji, 2019). For the node clustering methods, we use the graph coarsening schemes described in the Equation 15, with their specific implementations on generating an assignment matrix. For our proposed method, we follow the setting of node clustering methods. In particular, we finely tuned the height of coding tree produced by Algorithm 1 to make sure the number of nodes compressed in the first layer close to the setting of baselines. For model configuration, the pooling ratio of all models is set to 25%, the learning rate is set to 5×10^{-3}, and hidden size is set to 32. For the loss function, we use the Mean Squared Error (MSE) to train models. We then optimize the network with Adam optimizer. We use the early stopping criterion, where we stop the training if there is no further improvement on the training loss during 1,000 epochs. Further, the maximum number of epochs is set to 10,000. Note that, there is no other available graphs for validation of the synthetic graph, such that we train and test the models only with the given graph in the Figure 3(a).

Reconstruction results on all hierarchical pooling methods. Figure A.2 and A.3 show the results of reconstructed graphs by model with various pooling methods. As can be seen from the first row of Figure A.2 and A.3, the node drop methods suffer from the issue of information loss, resulting in the basic shape of original graphs can not be identified. For node clustering methods, DiffPool and MinCutPool, the basic shape of original graphs is basically retained. However, we can still see the significant distortion in the edge of ring and the center of grid, which are almost prevented in SEP.

A.2. Graph Classification

Social network datasets. IMDB-BINARY and IMDB-MULTI are derived from the collaboration of a movie set. In these two datasets, every graph consists of actors or actresses, and each edge between two nodes represents their cooperation in a certain movie. Each graph is derived from a prespecified movie, and its label corresponds to the genre of this movie. Similarly, COLLAB is also a collaboration dataset but from a scientific realm, which includes three public collaboration datasets (i.e., Astro Physics, High Energy Physics and Condensed Matter Physics). Many researchers from each field form various ego networks for the graphs in this benchmark. The label of each graph is the research field to which the nodes belong.

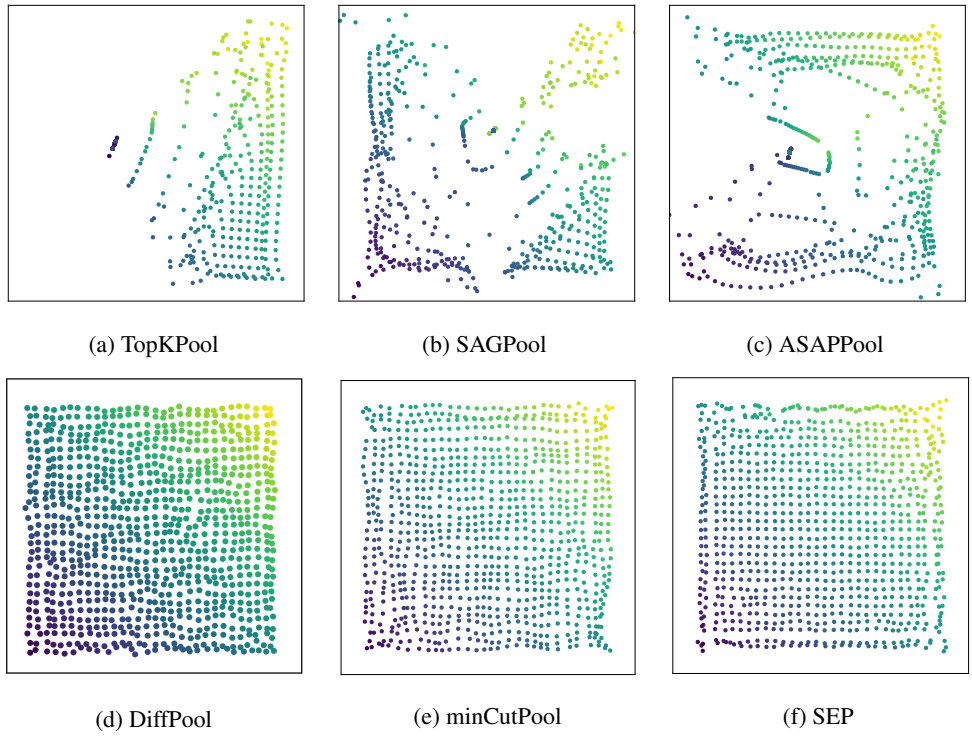
Bioinformatic datasets. D&D contains graphs of protein structures. A node represents an amino acid and edges are constructed if the distance of two nodes is less than $6\mathring{A}$. A label denotes whether a protein is an enzyme or non-enzyme. PROTEINS is a dataset where the nodes are secondary structure elements (SSEs), and there is an edge between two nodes if they are neighbors in the given amino acid sequence or in 3D space. The dataset has 3 discrete labels, representing helixes, sheets or turns. PTC is a dataset containing 344 chemical compounds that reports the carcinogenicity of male and female



(a) TopKPool (b) SAGPool (c) ASAPPool

(d) DiffPool (e) minCutPool (f) SEP

Figure A.2: Reconstruction results of ring synthetic graphs, compared to node drop and clustering methods.



(a) TopKPool (b) SAGPool (c) ASAPPool

(d) DiffPool (e) minCutPool (f) SEP

Figure A.3: Reconstruction results of grid synthetic graphs, compared to node drop and clustering methods.

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

Initial inputs. The data of the bioinformatic datasets and social network datasets differ in that the nodes in bioinformatics graphs have categorical labels that do not exist in social networks. Thus, the initial node features of the HRN inputs are set to one-hot encodings of the node degrees for social networks and a combination of the one-hot encodings of the degrees and categorical labels for bioinformatic graphs.

Implementation details. We evaluate the model performance with a 10-fold cross validation setting, where the dataset split is based on the conventionally used training/test splits (Zhang et al., 2018; Bianchi et al., 2020; Baek et al., 2021). In addition, we use the 10 percent of the training data as a validation data following the fair comparison setup (Errica et al., 2020). We use the early stopping criterion, where we stop the training if there is no further improvement on the validation loss during 50 epochs. Furthermore, the maximum number of epochs is set to 500. We then report the average performances on test sets, by performing overall experiments 10 times. In particular, following the implementation of (Xu et al., 2019), we train each epoch with a fixed number of iterations (i.e., 50) for small datasets. We set the pooling ratio as 25% in each pooling layer for baselines as previous works (Baek et al., 2021; Bianchi et al., 2020), while our model follows the natural cluster assignments produced by Algorithm 1 with given height 3. For model configuration, the learning rate is set to 5×10^{-4}, the hidden size is set $\in \{64, 128\}$, the batch size is set $\in \{32, 128\}$, weight decay is set to 1×10^{-4}, and dropout rate is set $\in \{0, 0.5\}$. Then we optimize the network with Adam optimizer. For a fair comparison of baselines (Lee et al., 2019), we use the three GCN layers (Kipf & Welling, 2017) as a message passing function for all models with skip connections, and only change the pooling architecture throughout all models. Because GMT is the most recent work that replicates these popular pooling approaches in previous studies, and we implement SEP-G based on the code of GMT [4], thus the accuracies of baselines are derived from (Baek et al., 2021).

A.3. Node Classification

Citation datasets. We utilize three standard citation network benchmark datasets: Cora, Citeseer and Pubmed (Sen et al., 2008). In all of these datasets, nodes correspond to documents and edges to (undirected) citations. Node features correspond to elements of a bag-of-words representation of a document. Each node has a class label. In particular, the node features are different among three datasets. Specifically, the input features of Cora and Citeseer are one-hot embedding of words in each document, while the node features of Pubmed are the TF-IDF weighted word vectors. The Cora dataset contains 2708 nodes, 5429 edges, 7 classes and 1433 features per node. The Citeseer dataset contains 3327 nodes, 4732 edges, 6 classes and 3703 features per node. The Pubmed dataset contains 19717 nodes, 44338 edges, 3 classes and 500 features per node.

Implementation details. We closely follow the transductive experimental setup in (Kipf & Welling, 2017). Each class is only allowed 20 nodes for training, which honors the transductive setup, and the training algorithm has access to all of the nodes' feature vectors. The predictive power of the trained models is evaluated on 1000 test nodes, and we use 500 additional nodes for validation purposes. We also use the early stopping criterion, where we stop the training if there is no further improvement on the validation loss during 50 epochs. Furthermore, the maximum number of epochs is set to 1000. We obtain cluster assignment matrices of each dataset under the guidance of three-dimensional structural entropy, and adopt the first two layers for hierarchical pooling. For model configuration, the learning rate is set to 0.01, the hidden size is set $\in \{16, 32, 128, 256\}$, weight decay is set $\in \{0.02, 5 \times 10^{-4}\}$, and dropout rate for each layer is set $\in \{0, \ldots, 0.9\}$. Finally, we optimize the network with Adam optimizer.

Fair comparison with SOTA methods. To make a fair comparison with S^2GC and GCNII, we present the results of the two methods with similar convolutional layers. As shown in Table A.1, SEP-N obtains competitive results with only 5 GCN layers, which is much less than the requirement of S^2GC (16) and GCNII (64 for Cora, 32 for Citeseer, 16 for Pubmed). Furthermore, SEP-N outperforms S^2GC and GCNII when putting similar number of convolution layers, which reveals the effectiveness of hierarchical graph pooling in node classification tasks when facing limited computing resources.

[4] https://github.com/JinheonBaek/GMT

Table A.1: S^2GC and GCNII with similar #Convs of SEP-N.

	Model (#Convs)				
	S^2GC(4)	S^2GC(8)	GCNII(4)	GCNII(8)	SEP-N(5)
Cora	79.8	82.2	82.6	84.2	**84.8**
Citeseer	72.6	72.7	68.9	70.6	**72.9**
Pubmed	79.2	79.7	78.8	79.3	**80.2**