# On Exploring Node-feature and Graph-structure Diversities for Node Drop Graph Pooling

**Anonymous authors**
Paper under double-blind review

## Abstract

Graph pooling is essential in learning effective graph-level representations. One mainstream type of graph pooling is node drop pooling, which preserves the nodes in graphs with top-k calculated significance scores. However, despite being commonly adopted, current node drop pooling methods generally ignore node diversity from the perspectives of node features and graph structures. Therefore, they still obtain graph-level representations suboptimally. To address the issue mentioned above, we propose a novel plug-and-play scheme, termed MID, using a **M**ultidimensional score space with two score operations, *i.e.*, fl**I**pscore and **D**ropscore, to explore the node-feature and graph-structure diversities in graphs. Specifically, the multidimensional score space depicts the significance of nodes through multiple criteria; the flipsscore encourages the maintenance of dissimilar features, thus preserving the node-feature diversity; and the dropscore forces the model to notice diverse graph structures instead of being stuck in significant local structures. What is more, we evaluate our proposed MID by applying it to a variety of popular node drop pooling methods, including TopKPool, SAGPool, GSAPool, and ASAP. Extensive experiments on seventeen real-world graph classification datasets demonstrate that our proposed scheme efficiently and consistently brings over 2.8% improvements in average when using different backbone models and datasets. The datasets include FRANKENSTEIN, IMDB-B, IMDB-M, REDDIT-B, COLLAB from the social domain and D&D, PROTEINS, NCI1, MUTAG, PTC-MR, NCI109, ENZYMES, MUTAGENICITY, HIV, BBBP, TOX-CAST, TOX21 from the biochemical domain.[1]

## 1 Introduction

Graph Neural Networks (GNNs) have achieved excellent performance on a wide range of graph-based tasks, including node classification (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018), link prediction (Zhang & Chen, 2018; Cai & Ji, 2020), and graph classification (Errica et al., 2020; You et al., 2020; Qiu et al., 2020; Rong et al., 2020a; Sun et al., 2021). In node classification and link prediction, GNNs propagate information between nodes through graph convolutions, whereas, in graph classification, information of all nodes is integrated together to generate graph-level representations through graph pooling.

Earlier graph pooling directly adopts global pooling, including average pooling and max pooling, which ignore the node correlation, hampering the overall performance (Duvenaud et al., 2015; Vinyals et al., 2016; Zhang et al., 2018). Later, graph pooling utilizes hierarchical architectures to model the node correlations (Ma et al., 2019; Wang & Ji, 2020; Gao et al., 2021a) and can be roughly classified into two types: node clustering pooling and node drop pooling. Node clustering pooling requires clustering nodes into new nodes, which is time-and space-consuming (Ying et al., 2018; Bianchi et al., 2020; Yuan & Ji, 2020). In contrast, drop node pooling only preserves the representative nodes by calculating the significance of nodes and is more fit for large-scale graphs (Gao & Ji, 2019; Lee et al., 2019; Zhang et al., 2020a; Gao et al., 2021b).

[1]Code will be made publicly available at `http://github.com/xxx/xxx`.

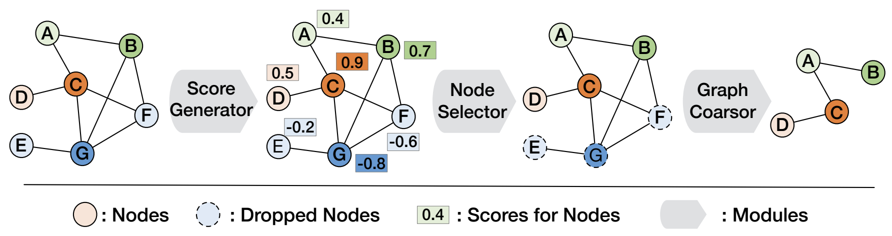


Figure 1: Our modularized scheme for node drop pooling.

In this paper, we focus on node drop pooling. We have reviewed previous node drop pooling methods and conclude that despite efficiency, current node drop pooling methods still obtain suboptimal graph-level representations because they ignore the diversities in graphs, including the node-feature diversity and the graph-structure diversity, when selecting top-k significance nodes. Specifically, current node drop pooling methods monotonically calculate the significance of nodes. **Node-feature Diversity:** Analytically, current drop node pooling methods tend to highlight nodes with similar features, and thus performing less satisfactorily when multiple dissimilar nodes contribute to the graph-level representations. **Graph-structure Diversity:** Furthermore, nodes that are connected would share similar information through GNNs (McPherson et al., 2001; Wu et al., 2019). Current node drop pooling methods are probably stuck into the nodes of local structures, while ignoring the rest of representative graph structures.

To explore the diversity, we propose a novel plug-and-play scheme, termed MID, consisting of **M**ultidimensional score space with fl**I**pscore and **D**ropscore operations. Analytically, multidimensional score space depicts the significance of nodes using vectors instead of scalars, and thus could comprehensively describe the attributes of nodes; flipscore reverses the negative confidence scores as positive ones, encouraging the drop node pooling to highlight nodes with dissimilar features; dropscore randomly drops several nodes only during training when selecting the top-k scores, probably deleting nodes in local structures and forcing the node drop pooling to notice diverse graph structures. Experimentally, we apply our MID to four typical node drop pooling methods. The results on seventeen real-world graph classification datasets demonstrate the ability and generalizability of MID. Specifically, MID consistently brings over 2.8% improvements in average across different backbone models and datasets. Ablation experiments further demonstrate the contribution of individual components of MID to explore the diversity of graphs.

The main contributions of this paper are summarized as follows: **1)** We have reviewed and summarized the previous node drop pooling methods. Accordingly, we design a new modularized scheme for node drop pooling. **2)** We propose a novel play-and-plug scheme, MID, which contains multidimensional score space, flipscore operation, and dropscore operation. MID could be applied in most node drop pooling models and improve their performance with relatively low computational cost. **3)** We conducted extensive experiments for four typical drop node pooling methods with and without MID on seventeen real-world and two synthetic datasets for graph classification. The experimental results comprehensively demonstrate the capability of our MID.

## 2 Related Work of Graph Pooling

Graph pooling could be divided into global pooling and hierarchical pooling. Global pooling performs the mean or sum on all node features to obtain the graph-level representation, which results in a loss of information since they ignore the structures of graphs (Duvenaud et al., 2015). Later, hierarchical pooling models were proposed, which could be roughly divided into node clustering pooling, node drop pooling, and other pooling. **1) Node clustering pooling** considers the graph pooling problem as a node clustering problem to map the nodes into a set of clusters (Ying et al., 2018; Noutahi et al., 2019; Wang et al., 2020; Bianchi et al., 2020; Yuan & Ji, 2020; Baek et al., 2021). One limitation of these methods is their time and storage complexity caused by the computation of the soft clustering. Besides, according to (Mesquita et al., 2020), clustering-enforcing regularization is usually innocuous. **2) Node drop pooling** uses learnable scoring functions to delete nodes with lower significance scores (Zhang et al., 2018; Gao & Ji, 2019; Lee et al., 2019; Huang et al., 2019; Ranjan et al., 2020; Zhang et al., 2020c; Khasahmadi et al., 2020; Li et al., 2020a; Zhang et al., 2020a; Gao et al., 2021b; Tang et al., 2021). Node drop pooling is more efficient and

Table 1: Summary of node drop pooling models in our framework.

| Models | Score Generator | Node Selector | Graph Coarsor |
|---|---|---|---|
| SortPool (Zhang et al., 2018) | $\boldsymbol{S} = \boldsymbol{X}[:, -1]$ | $\text{idx} = \text{top}(\boldsymbol{S}, \lceil k \times n \rceil)$ | $\boldsymbol{X}' = \boldsymbol{X}_{\text{idx}} \odot \boldsymbol{S}_{\text{idx}}; \quad \boldsymbol{A}' = \boldsymbol{A}_{\text{idx,idx}}$ |
| TopKPool (Gao & Ji, 2019) | $\boldsymbol{S} = \boldsymbol{X}\mathbf{p} / \lVert \mathbf{p} \rVert_2$ | $\text{idx} = \text{top}(\boldsymbol{S}, \lceil k \times n \rceil)$ | $\boldsymbol{X}' = \boldsymbol{X}_{\text{idx}} \odot \boldsymbol{S}_{\text{idx}}; \quad \boldsymbol{A}' = \boldsymbol{A}_{\text{idx,idx}}$ |
| SAGPool (Lee et al., 2019) | $\boldsymbol{S} = \sigma\left(\tilde{\boldsymbol{D}}^{-\frac{1}{2}} \tilde{\boldsymbol{A}} \tilde{\boldsymbol{D}}^{-\frac{1}{2}} X \boldsymbol{\Theta}_{att}\right)$ | $\text{idx} = \text{top}(\boldsymbol{S}, \lceil k \times n \rceil)$ | $\boldsymbol{X}' = \boldsymbol{X}_{\text{idx}} \odot \boldsymbol{S}_{\text{idx}}; \quad \boldsymbol{A}' = \boldsymbol{A}_{\text{idx,idx}}$ |
| AttPool (Huang et al., 2019) | $\boldsymbol{S} = f_{att}(\boldsymbol{X})$ | $\text{idx} = \text{top}(\boldsymbol{S}, \lceil k \times n \rceil)$ | $\boldsymbol{X}' = \boldsymbol{X}_{\text{idx}} \odot \boldsymbol{S}_{\text{idx}}; \quad \boldsymbol{A}' = \boldsymbol{A}_{\text{idx,idx}}$ |
| ASAP (Ranjan et al., 2020) | $\boldsymbol{S} = LEconv(\boldsymbol{X}, \boldsymbol{A})$ | $\text{idx} = \text{top}(\boldsymbol{S}, \lceil k \times n \rceil)$ | $\boldsymbol{X}' = \boldsymbol{X}^c_{\text{idx}} \odot \boldsymbol{S}_{\text{idx}}; \quad \boldsymbol{A}' = \boldsymbol{A}^c_{\text{idx,idx}}$ |
| HGP-SL (Zhang et al., 2020c) | $\boldsymbol{S} = \left\lVert \left(\mathbf{I} - (\mathbf{D})^{-1} \mathbf{A}\right) \mathbf{X} \right\rVert_1$ | $\text{idx} = \text{top}(\boldsymbol{S}, \lceil k \times n \rceil)$ | $\boldsymbol{X}' = \boldsymbol{X}^c_{\text{idx}} \odot \boldsymbol{S}_{\text{idx}}; \quad \boldsymbol{A}' = \boldsymbol{A}^c_{\text{idx,idx}};$ <br> $\mathbf{E}_i(p,q) = \sigma\left(\overrightarrow{\mathbf{a}} \left[\mathbf{X}^k_i(p,:) \Vert \mathbf{X}_i(q,:)\right]^\top\right) + \lambda \cdot \mathbf{A}_i(p,q)$ |
| GSAPool (Zhang et al., 2020a) | $\boldsymbol{S}_1 = \sigma(\text{GNN}(\boldsymbol{X}, \boldsymbol{A}))$ <br> $\boldsymbol{S}_2 = \sigma(\text{MLP}(\boldsymbol{X}))$ <br> $\boldsymbol{S} = \alpha \boldsymbol{S}_1 + (1-\alpha)\boldsymbol{S}_2$ | $\text{idx} = \text{top}(\boldsymbol{S}, \lceil k \times n \rceil)$ | $\boldsymbol{X}' = \boldsymbol{A}\boldsymbol{X}\boldsymbol{W} \odot \boldsymbol{S}_{\text{idx}},$ <br> $\boldsymbol{A}' = \boldsymbol{A}_{\text{idx,idx}}$ |
| RepPool (Li et al., 2020a) | $\boldsymbol{S} = \sigma\left(\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{X}\mathbf{p} / \lVert \mathbf{p} \rVert\right)$ | $\text{idx} = \text{top}(\boldsymbol{S}, \lceil k \times n \rceil)$ | $\mathbf{B} = \boldsymbol{X}\mathbf{W}_{\mathbf{b}}(X_{\mathbf{idx}})^T$ <br> $\mathbf{X}' = (\text{softmax}(\mathbf{B} \odot \mathbf{M}))^T \left(\mathbf{X} \odot \left(\mathbf{s}\mathbf{1}^T_d\right)\right)$ <br> $\mathbf{A}' = (\text{softmax}(\mathbf{B} \odot \mathbf{M}))^T \mathbf{A} (\text{softmax}(\mathbf{B} \odot \mathbf{M}))$ |
| Ipool (Gao et al., 2021b) | $\boldsymbol{S} = \left\lvert \left(\boldsymbol{I} - \frac{1}{t} \sum_1^t \boldsymbol{D}^{-1}\boldsymbol{A}\right) \boldsymbol{X} \right\rvert_2$ | $\text{idx} = \text{top}(\boldsymbol{S}, \lceil k \times n \rceil)$ <br> $\text{idx} = \text{top}(\tilde{\boldsymbol{S}}, \lceil k \times n \rceil)$ | $\boldsymbol{X}' = \boldsymbol{X}_{\text{idx}} \odot \boldsymbol{S}_{\text{idx}},$ <br> $(\boldsymbol{A}')_{ij} = \lambda (\boldsymbol{A} + \boldsymbol{I})_{\mathbf{idx}[i], idx[j]}$ |

likely applied to large-scale networks than node clustering pooling. However, one major issue of node drop pooling is the inevitable information loss (including the diversity). **3) Other pooling.** In addition, numerous graph pooling methods have emerged recently (Murphy et al., 2019; Studer et al., 2020; Ma et al., 2020; Li et al., 2020b; Zhang et al., 2021; Pang et al., 2021; Du et al., 2021; Wu et al., 2021; Nadgeri et al., 2021).

## 3 METHOD

This section first reviews current node drop pooling methods. Then, we present the details of our MID and explain how MID improves the node drop pooling.

**Notation.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph with nodeset $\mathcal{V}$ and edgeset $\mathcal{E}$. The node features are denoted as $\boldsymbol{X} \in \mathbb{R}^{n \times c}$ with $n$ nodes and the $c$ dimensional features. The adjacency matrix is defined as $\boldsymbol{A} \in \{0, 1\}^{n \times n}$. More notations are introduced in Table 5 in Appendix.

### 3.1 REVIEW OF NODE DROP POOLING

We first summarize previous node drop pooling methods to give an in-depth analysis. Specifically, we design a new modularized framework to describe the process of node drop pooling, which is shown in Figure 1. We deconstruct node drop graph pooling with three disjoint modules: **1) Score generator.** Given an input graph, the score generator calculates the significance scores for each node. **2) Node selector.** Node selector selects the nodes with the top-k significance scores. **3) Graph coarser.** With the selected nodes, a coarsened graph from the original graph is obtained by learning the feature matrix and the adjacency matrix. The process can be formulated as follows:

$$\begin{aligned} \boldsymbol{S}^{(l)} &= \text{SCORE}(\boldsymbol{X}^{(l)}, \boldsymbol{A}^{(l)}); \quad \text{idx}^{(l+1)} = \text{TOP}_k(\boldsymbol{S}^{(l)}); \\ \boldsymbol{X}^{(l+1)}, \boldsymbol{A}^{(l+1)} &= \text{COARSOR}(\boldsymbol{X}^{(l)}, \boldsymbol{A}^{(l)}, \boldsymbol{S}^{(l)}, \text{idx}^{(l+1)}), \end{aligned} \tag{1}$$

where functions SCORE, TOP$_k$, and COARSOR are specially designed for the score generator, node selector, and graph coarsor by each method, respectively. TOP$_k$ preserves the top-k nodes. $\boldsymbol{S}^{(l)} \in \mathbb{R}^{n \times 1}$ indicates the significance scores, $\text{idx}^{(l+1)}$ indicates the preserving node indexes. $l$ and $l + 1$ indicate the layer numbers.

Accordingly, nine typical node drop pooling models are presented in Table 1. As we can observe, all node drop pooling methods only improve the processes of score generator and graph coarsor. However, a simple node selector tend to highlight nodes with similar features during training, thus ignoring the node-feature diversity and graph-structure diversity in graphs.

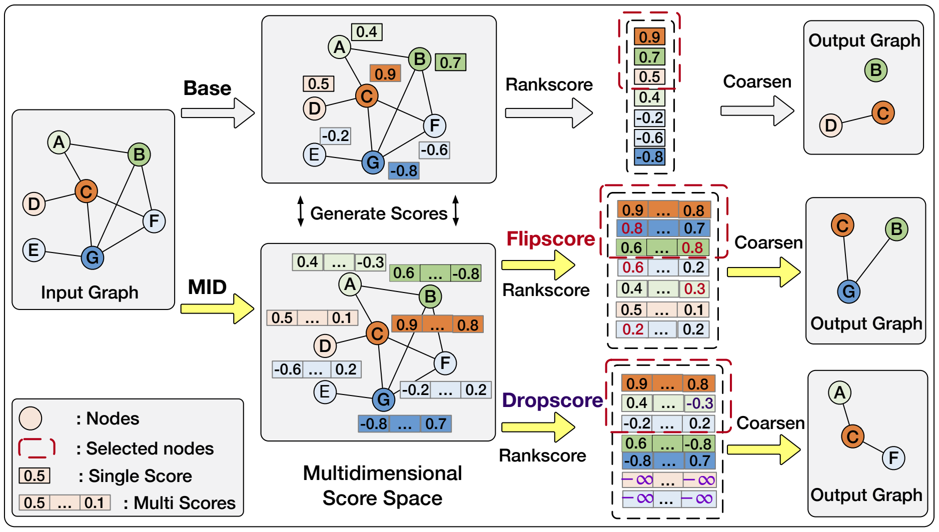


Figure 2: Architecture of our proposed method.

## 3.2 Our Proposed MID

In contrast to the previous models, we mainly focus on the node selector module. We propose MID, which is a play-and-plug scheme for improving node drop pooling. As shown in Figure 2, MID consists of 1) **a multidimensional score space**, 2) **a flipscore operation**, and 3) **a dropscore operation**. When the above three components are appropriately integrated, MID can alleviate the problems of node-feature diversity and graph-structure diversity. In the following subsections, we discuss and justify each component analytically in detail.

### 3.2.1 The Multidimensional Score Space

We first extend the original one-dimensional score space to a multidimensional score space, therefore depicting comprehensive information of nodes:

$$\boldsymbol{S}_{\text{multi}}^{(l)} = \text{SCORE}(\boldsymbol{X}^{(l)}, \boldsymbol{A}^{(l)}), \tag{2}$$

where $\boldsymbol{S}_{\text{multi}}^{(l)} \in \mathbb{R}^{n \times h}$ is the significance score matrix. For each node, there are $h$ different scalars, and the final significance score is the sum of all scalars in one vector. Let $\boldsymbol{S}_{\text{multi}}^{(l)} = \{s_1^{(l)} \| s_2^{(l)} \| ..., \| s_h^{(l)}\}$, then the new features of the nodes is defined as: $\{(\boldsymbol{X}^{(l)} \odot s_1^{(l)}) \| (\boldsymbol{X}^{(l)} \odot s_2^{(l)}) \| ..., \| (\boldsymbol{X}^{(l)} \odot s_h^{(l)})\}$. Here, $\|$ indicates the concatenation.

**Intuition** Previous methods mainly use a scalar quantity to represent the importance of each node. However, due to the complexity of a graph, a scalar is not enough to capture the significance of a node from different views. Therefore, we extend the importance score for each node from a scalar to a multidimensional vector to evaluate the significance of nodes from multi-views.

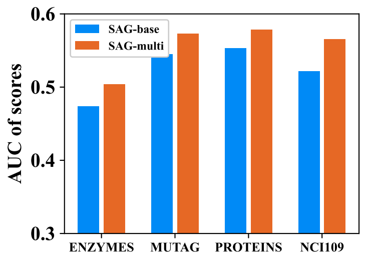


Figure 3: Score Correctness.

**Analytical Verification** Motivated by the previous study (Knyazev et al., 2019), we calculated the score correctness on four benchmark datasets with SAGPool (Lee et al., 2019) model to verify whether our method evaluates the importance of the nodes more accurately. As shown in Figure 3, we import AUC to evaluate the score correctness. The details of metrics and experimental settings are described in Appendix G.1. It is obvious that models with a multidimensional score space consistently generate more accurate scores for nodes, which confirms that using a multidimensional vector to provide the significance of nodes can improve the evaluation.

### 3.2.2 The Flipscore Operation

After getting a multidimensional score space, we entail the scores generated by models to extract as many diverse nodes as possible. Specifically, we apply an $L_1$-norm to the score vector:

$$\boldsymbol{S}_{L_1} = \left\| \boldsymbol{S}_{\text{mutli}} \right\|_1 , \tag{3}$$

where $\boldsymbol{S}_{L_1}$ is the resulting score matrix, and $\left\| \cdot \right\|_1$ is the $L_1$-norm of the argument. Note that the flipscore operation will not influence the node updating of graph coarsor. We adopt this operation in the training, validation, and test phases.

**Intuition** The generated scores of most previous models range from -1 to 1 by using Tanh (Lee et al., 2019; Gao & Ji, 2019; Zhang et al., 2020a). In a monotonous manner, previous methods tend to highlight similar and significant nodes instead of representative and significant nodes. In contrast, our flipscore operation could highlight nodes with extremely different scores in original space, encouraging node drop pooling to capture nodes with dissimilar features. Intuitively, dissimilar features lead to more information gain, which experimentally contributes to better graph classification (Zhang et al., 2020c; Gao et al., 2021b).

**Analytical Verification** To verify *whether the flipscore operation encourages the models to explore diverse features*, we compute information gain after one aggregation and pooling operation by Kullback–Leibler divergence (Kullback & Leibler, 1951) following the previous study (Hou et al., 2020). The details of experiment settings and results are described in Appendix G.2. As shown in Figure 4, we conducted experiments on the MUTAG dataset by using SAGPool (Lee et al., 2019) and GSAPool (Zhang et al., 2020a) models. As the layer number increases, information gain gradually decreases. Adding our flipscore operation significantly improves the information gain, confirming that flipscore operation promotes models to maintain more diverse node features.

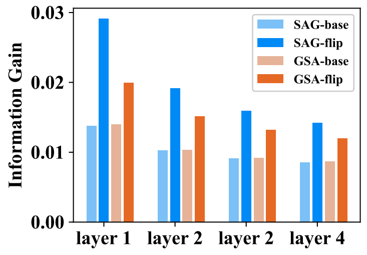


Figure 4: Information gain.

### 3.2.3 The Dropscore Operation

Dropscore operation is proposed to force node drop pooling models to notice as many substructures of graphs as possible. Specifically, dropscore operation randomly drops out several nodes with a certain rate in the graph during training only when selecting top-k scores:

$$\boldsymbol{S}_{\text{drop}} = \boldsymbol{I}_{\lceil p_s \times n \rceil} \boldsymbol{S}_{\text{multi}}, \tag{4}$$

where $\boldsymbol{S}_{\text{drop}}$ is the resulting score matrix, $\boldsymbol{I}_{\lceil p_s \times n \rceil}$ is a matrix generated by randomly dropping $\lceil p_s \times n \rceil$ none-zero elements of a unit matrix with $n$ dimensions, $\lceil \cdot \rceil$ is the operation of rounding up and $p_s$ is the score dropping rate. We adopt this operation in the training phase.

**Intuition** Note that through GNNs, nodes that are directly connected tend to share similar information (McPherson et al., 2001; Wu et al., 2019). Therefore, models generate similar scores for nearby nodes, which is also observed in the previous study (Knyazev et al., 2019). Under this condition, models may be stuck into significant local structures and select redundant nodes, ignoring significant nodes from other substructures and losing structure information. Specifically, we devise dropscore operation, which randomly drops out a certain rate of nodes during training, expecting that models do not focus on one local substructure by deleting several nodes in the local substructure.

**Case Study** To investigate *whether the dropscore tech can assist models to cover more significant substructures*, we conduct experiments on the NCI1 dataset by using the SAGPool model (Lee et al., 2019). The details of experimental settings and results are described in Appendix G.3. We observe that SAGPool is likely to select nodes concentrated in the same area as shown in Figure 5. Therefore, important information of other parts might be

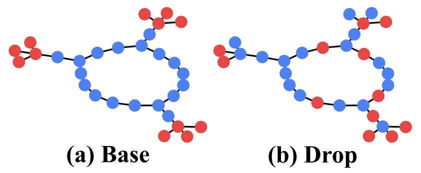


Figure 5: Visualization of node selection results with and without dropscore operation. Selected nodes are red.

neglected. After adopting our drop scores operation, except for the performance improvements, selected nodes are distributed in different substructures covering the whole graph, confirming that our method encourages models to maintain more diverse structure information.

## 4 Theoretical Analysis

In this subsection, we present the theoretical analysis regarding the propositions of the trapped scores and the graph permutation equivariance, and then prove that our proposed MID could promote the backbone pooling models to be more powerful in distinguishing different graphs. Note that all detailed proofs are provided in Appendix F.

### 4.1 Trapped Scores

In Figure 5, we find that scores, which are generated by pooling models, tend to be stuck in the local structure, we refer the scores as *trapped scores*. In the following, we prove that, under some conditions, the selected nodes (correspond to trapped scores) of most node drop pooling models are stuck in the local structure. Moreover, the finding of the proposition could be applied to other cases, *e.g.*, SAGPool (Lee et al., 2019), GASPool (Zhang et al., 2020a), and ASAP (Ranjan et al., 2020).

**Proposition 1.** *Consider a graph $\mathcal{G}$ with the adjacency matrix $\mathbf{A}$, $\mathbf{x}_v$ is the feature for node $v$. Assume that linked nodes in the same local structure have similar features. Let $\mathbf{x}_1, \mathbf{x}_2$ be node features and $\mathbf{s}_1, \mathbf{s}_2$ be the significance scores, and then, the distance between $\mathbf{s}_1$ and $\mathbf{s}_2$ approximates to zero, i.e., $\|\mathbf{s}_1 - \mathbf{s}_2\|_2 \to 0$, as long as $\|\mathbf{x}_1 - \mathbf{x}_2\|_2 \to 0$.*

Note that the assumption that linked nodes in the same local structure have similar features is common in most real-world networks (McPherson et al., 2001), and the scores $\mathbf{s}_1$, $\mathbf{s}_2$ are vectors involving different elements. We use $\|\cdot\|_2$ to denote the $L_2$ norm.

### 4.2 Graph Permutation Equivariance

Graph pooling generates isomorphic pooled graphs after graph permutation, which is defined as graph permutation equivariancce (Ranjan et al., 2020). We show that our proposed MID would not break the graph permutation equivariant property of backbone models, e.g., TopKPool (Gao & Ji, 2019), SAGPool (Lee et al., 2019), GASPool (Zhang et al., 2020a), and ASAP (Ranjan et al., 2020).

**Proposition 2.** *Suppose the backbone graph pooling model is graph permutation equivariant, and the model combined with our MID is still graph permutation equivariant.*

### 4.3 Connection with Weisfeiler-Lehman Graph Isomorphism Test

Weisfeiler-Lehman (WL) test of graph isomorphism (Weisfeiler & Leman, 1968) provides an effective solution to distinguish different graphs. According to previous studies (Xu et al., 2019), a GNN can have as large discriminative power as the WL test if the aggregation scheme is an injective function. The overall model for graph classification consists of two phases: graph convolution and graph pooling. If we make the pooling function injective, then the model can be as powerful as the WL test in distinguishing different graphs. In the following, we demonstrate that the proposed MID could promote the backbone pooling models to be more powerful in distinguishing different graphs. We first introduce the theorem from Xu et al. (2019).

**Theorem 1 (Non-isomorphic Graphs to Different Embeddings).** *Let $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ be a GNN. If the aggregation function and the graph-level readout function are injective, $\mathcal{A}$ can distinguish any two non-isomorphic graphs $G_1$ and $G_2$, which are decided by the Weisfeiler-Lehman test of isomorphism.*

According to the Theorem 1, the key of the expressive power is the injectiveness of function, which means the outputs of the function are more unique. Our next corollary shows that our proposed MID enables the outputs of backbone pooling models to be more powerful.

**Corollary 1 (Expressive power on Node Drop Pooling Models).** *Assume the feature space $\mathcal{X}$ is a countable set. Let $\boldsymbol{X} \in \mathcal{X}$ be the matrix of node features, $A$ be the adjacency matrix, $p_s$ be the score dropping ratio, and $h$ be the score dimension. The output of POOL-MID$(\mathbf{X}, \mathbf{A}, p_s, h)$ with*

Table 2: MID performance across four backbone models and seventeen datasets in graph classification task. The reported results are mean and standard deviation over 10 different runs. Red: the best performance. Blue: the second best performance. Hyphen(-) denotes out-of-resources.

| | **Social Domain in TU Datasets (5)** | | | | | **OGB Datasets (4)** | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **FRAN.** | **IMDB-B** | **IMDB-M** | **REDDIT-B** | **COLLAB** | **HIV** | **BBPB** | **Tox21** | **ToxCast** |
| Set2set | 61.79$_{\pm 0.23}$ | 73.10$_{\pm 0.48}$ | 50.15$_{\pm 0.58}$ | **90.03**$_{\pm 0.45}$ | 79.88$_{\pm 0.50}$ | 73.42$_{\pm 2.34}$ | 64.43$_{\pm 2.16}$ | 73.42$_{\pm 0.67}$ | 59.76$_{\pm 0.65}$ |
| SortPool | 62.42$_{\pm 0.57}$ | 72.49$_{\pm 0.78}$ | 49.62$_{\pm 0.36}$ | 87.00$_{\pm 1.03}$ | 80.21$_{\pm 0.35}$ | 71.88$_{\pm 1.83}$ | 64.33$_{\pm 3.10}$ | 68.90$_{\pm 0.78}$ | 59.28$_{\pm 0.99}$ |
| DiffPool | **63.95**$_{\pm 0.81}$ | 72.99$_{\pm 0.65}$ | **51.03**$_{\pm 0.48}$ | – | 79.24$_{\pm 0.57}$ | 75.05$_{\pm 1.71}$ | 64.77$_{\pm 2.43}$ | 75.82$_{\pm 0.69}$ | **65.79**$_{\pm 0.87}$ |
| EdgePool | 58.09$_{\pm 2.04}$ | 72.13$_{\pm 0.72}$ | **51.05**$_{\pm 0.53}$ | 89.12$_{\pm 1.22}$ | **81.22**$_{\pm 0.94}$ | 72.15$_{\pm 1.56}$ | **68.56**$_{\pm 1.43}$ | 74.54$_{\pm 0.79}$ | 62.57$_{\pm 1.36}$ |
| MinCutPool | 62.32$_{\pm 0.55}$ | 73.05$_{\pm 0.80}$ | 50.22$_{\pm 0.81}$ | 86.69$_{\pm 0.48}$ | 78.78$_{\pm 0.61}$ | 73.91$_{\pm 1.10}$ | 66.47$_{\pm 1.90}$ | **78.78**$_{\pm 0.61}$ | 63.66$_{\pm 1.56}$ |
| HaarPool | – | **73.46**$_{\pm 0.55}$ | 50.37$_{\pm 0.55}$ | – | – | – | – | – | – |
| MemPool | 62.05$_{\pm 0.45}$ | 71.20$_{\pm 0.82}$ | 49.91$_{\pm 0.76}$ | – | – | 73.75$_{\pm 1.90}$ | 66.47$_{\pm 1.90}$ | 72.05$_{\pm 0.93}$ | 61.85$_{\pm 0.36}$ |
| GMT | 63.15$_{\pm 0.25}$ | 73.10$_{\pm 0.44}$ | 50.50$_{\pm 0.54}$ | 88.88$_{\pm 0.44}$ | **80.74**$_{\pm 0.54}$ | **76.41**$_{\pm 2.32}$ | 66.88$_{\pm 1.59}$ | **76.56**$_{\pm 0.90}$ | **64.53**$_{\pm 0.92}$ |
| SAGPool | 60.62$_{\pm 0.66}$ | 71.87$_{\pm 0.59}$ | 50.42$_{\pm 0.45}$ | 87.42$_{\pm 0.62}$ | 79.07$_{\pm 0.28}$ | 70.19$_{\pm 3.66}$ | 64.29$_{\pm 2.96}$ | 69.39$_{\pm 1.88}$ | 59.09$_{\pm 1.38}$ |
| **SAGPool + MID** | **61.33**$_{\pm 0.28}$ | **73.08**$_{\pm 0.30}$ | **51.05**$_{\pm 0.69}$ | **91.62**$_{\pm 0.30}$ | **79.93**$_{\pm 0.65}$ | **74.51**$_{\pm 1.31}$ | **66.54**$_{\pm 2.33}$ | **73.06**$_{\pm 0.66}$ | **60.23**$_{\pm 0.64}$ |
| TopKPool | 64.63$_{\pm 0.67}$ | 71.47$_{\pm 0.71}$ | 49.55$_{\pm 0.58}$ | 85.37$_{\pm 1.04}$ | 77.45$_{\pm 0.56}$ | 71.24$_{\pm 2.97}$ | 65.93$_{\pm 2.60}$ | 68.69$_{\pm 2.02}$ | 58.63$_{\pm 1.56}$ |
| **TopKPool + MID** | **66.30**$_{\pm 0.55}$ | **72.55**$_{\pm 0.73}$ | **50.38**$_{\pm 0.51}$ | 83.20$_{\pm 0.96}$ | **78.86**$_{\pm 0.52}$ | **75.11**$_{\pm 2.42}$ | **66.64**$_{\pm 2.33}$ | **71.24**$_{\pm 2.14}$ | **60.50**$_{\pm 1.47}$ |
| ASAP | 60.57$_{\pm 0.62}$ | 72.25$_{\pm 0.83}$ | 48.55$_{\pm 0.64}$ | – | – | 71.60$_{\pm 1.71}$ | 61.93$_{\pm 3.18}$ | 70.00$_{\pm 1.50}$ | 60.32$_{\pm 1.34}$ |
| **ASAP + MID** | **61.55**$_{\pm 0.95}$ | **73.12**$_{\pm 0.56}$ | **49.47**$_{\pm 0.48}$ | – | – | **72.50**$_{\pm 2.16}$ | **64.03**$_{\pm 1.86}$ | **71.04**$_{\pm 0.92}$ | **61.04**$_{\pm 0.42}$ |
| GSAPool | 60.41$_{\pm 0.53}$ | 72.41$_{\pm 0.57}$ | 50.72$_{\pm 0.57}$ | 87.46$_{\pm 0.77}$ | 78.97$_{\pm 0.33}$ | 71.47$_{\pm 2.43}$ | 64.49$_{\pm 3.31}$ | 69.18$_{\pm 2.05}$ | 59.60$_{\pm 1.17}$ |
| **GSAPool + MID** | **61.26**$_{\pm 0.37}$ | **73.06**$_{\pm 0.20}$ | **50.93**$_{\pm 0.20}$ | **88.88**$_{\pm 0.42}$ | **79.44**$_{\pm 0.30}$ | **74.49**$_{\pm 1.42}$ | **67.27**$_{\pm 1.86}$ | **72.61**$_{\pm 1.14}$ | **61.90**$_{\pm 0.80}$ |

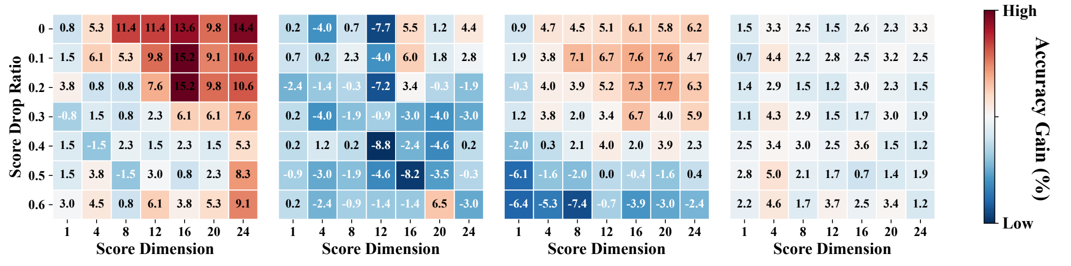
| | **Biochemical Domain in TU Datasets (8)** | | | | | | | | **Avg. rank** |
|---|---|---|---|---|---|---|---|---|---|
| | **D&D** | **PROTEINS** | **NCI1** | **MUTAG** | **PTC-MR** | **NCI109** | **ENZYMES** | **Mutagen.** | |
| Set2set | 72.65$_{\pm 0.47}$ | 73.14$_{\pm 0.97}$ | 71.70$_{\pm 0.73}$ | 70.50$_{\pm 1.99}$ | 54.53$_{\pm 1.69}$ | 69.78$_{\pm 0.43}$ | 42.92$_{\pm 2.05}$ | 79.86$_{\pm 0.43}$ | 7.9 |
| SortPool | 77.50$_{\pm 0.50}$ | 74.16$_{\pm 0.53}$ | 72.88$_{\pm 0.93}$ | 70.56$_{\pm 2.73}$ | 52.62$_{\pm 2.11}$ | 71.77$_{\pm 0.67}$ | 36.17$_{\pm 2.58}$ | 77.03$_{\pm 0.51}$ | 9.6 |
| DiffPool | 67.95$_{\pm 2.44}$ | 72.86$_{\pm 1.00}$ | **77.04**$_{\pm 0.73}$ | 82.50$_{\pm 2.54}$ | 55.26$_{\pm 3.84}$ | 75.38$_{\pm 0.66}$ | **51.27**$_{\pm 2.89}$ | 79.80$_{\pm 0.24}$ | 5.6 |
| EdgePool | 76.97$_{\pm 0.44}$ | 74.82$_{\pm 0.56}$ | 76.53$_{\pm 0.50}$ | 73.00$_{\pm 0.87}$ | **58.09**$_{\pm 2.04}$ | 75.58$_{\pm 0.50}$ | 31.18$_{\pm 1.99}$ | **80.78**$_{\pm 0.17}$ | 5.1 |
| MinCutPool | 74.69$_{\pm 0.52}$ | 74.28$_{\pm 0.76}$ | 71.15$_{\pm 0.98}$ | **85.17**$_{\pm 1.41}$ | 54.68$_{\pm 2.45}$ | 71.68$_{\pm 0.46}$ | 25.33$_{\pm 1.47}$ | 75.39$_{\pm 0.39}$ | 7.9 |
| HaarPool | - | - | - | 74.89$_{\pm 0.69}$ | 55.97$_{\pm 1.26}$ | - | 20.57$_{\pm 1.05}$ | - | 7.6 |
| MemPool | 72.96$_{\pm 0.84}$ | 71.99$_{\pm 0.65}$ | 67.65$_{\pm 0.67}$ | 69.17$_{\pm 2.04}$ | 53.26$_{\pm 2.67}$ | 65.50$_{\pm 1.16}$ | **44.73**$_{\pm 2.21}$ | 74.96$_{\pm 0.22}$ | 10.9 |
| GMT | **78.48**$_{\pm 0.48}$ | **75.08**$_{\pm 0.85}$ | 76.23$_{\pm 0.49}$ | **83.04**$_{\pm 1.01}$ | 55.41$_{\pm 1.30}$ | 74.49$_{\pm 0.52}$ | 37.38$_{\pm 1.52}$ | 79.94$_{\pm 0.35}$ | 3.5 |
| SAGPool | 76.15$_{\pm 0.73}$ | 73.42$_{\pm 0.93}$ | 70.89$_{\pm 0.80}$ | 67.67$_{\pm 3.56}$ | 54.38$_{\pm 1.96}$ | 70.26$_{\pm 0.99}$ | 36.30$_{\pm 2.51}$ | 74.20$_{\pm 0.75}$ | 11.4 |
| **SAGPool + MID** | **76.97**$_{\pm 0.72}$ | **74.52**$_{\pm 0.52}$ | **74.08**$_{\pm 0.40}$ | **72.72**$_{\pm 1.96}$ | **56.24**$_{\pm 1.22}$ | **75.21**$_{\pm 0.66}$ | **39.88**$_{\pm 1.93}$ | **79.75**$_{\pm 0.17}$ | **5.5** |
| TopKPool | 72.12$_{\pm 1.22}$ | 72.72$_{\pm 1.16}$ | 73.82$_{\pm 0.65}$ | 77.06$_{\pm 2.25}$ | 55.59$_{\pm 2.43}$ | 72.94$_{\pm 0.67}$ | 29.77$_{\pm 1.74}$ | 76.51$_{\pm 1.13}$ | 10.9 |
| **TopKPool + MID** | **74.65**$_{\pm 1.68}$ | **73.07**$_{\pm 0.80}$ | **78.50**$_{\pm 0.50}$ | **80.38**$_{\pm 2.26}$ | **56.76**$_{\pm 1.76}$ | **77.55**$_{\pm 0.49}$ | **30.37**$_{\pm 1.94}$ | **81.39**$_{\pm 0.35}$ | **6.1** |
| ASAP | 75.91$_{\pm 1.01}$ | 71.25$_{\pm 0.79}$ | 73.86$_{\pm 0.74}$ | 79.33$_{\pm 4.02}$ | 55.68$_{\pm 1.45}$ | 73.15$_{\pm 0.70}$ | 20.10$_{\pm 1.13}$ | 77.31$_{\pm 0.62}$ | 10.9 |
| **ASAP + MID** | **77.57**$_{\pm 0.79}$ | **72.14**$_{\pm 0.67}$ | **75.30**$_{\pm 0.43}$ | **82.33**$_{\pm 3.40}$ | **56.76**$_{\pm 1.88}$ | **75.60**$_{\pm 0.63}$ | **21.67**$_{\pm 1.25}$ | **78.94**$_{\pm 0.50}$ | **7.9** |
| GSAPool | 75.91$_{\pm 0.77}$ | 73.64$_{\pm 0.89}$ | 71.33$_{\pm 0.92}$ | 68.83$_{\pm 1.54}$ | 53.59$_{\pm 2.69}$ | 70.01$_{\pm 1.60}$ | 34.93$_{\pm 2.04}$ | 76.56$_{\pm 0.77}$ | 10.6 |
| **GSAPool + MID** | **76.70**$_{\pm 0.56}$ | **75.04**$_{\pm 0.42}$ | **75.72**$_{\pm 0.92}$ | **71.72**$_{\pm 1.04}$ | **55.47**$_{\pm 2.25}$ | **75.34**$_{\pm 0.37}$ | **42.48**$_{\pm 2.52}$ | **79.88**$_{\pm 0.20}$ | **5.6** |

*our method MID could be more unique than the output of backbone pooling models POOL$(\mathbf{X}, \mathbf{A})$. Then, the backbone pooling models with MID could approximately be as powerful as the WL test.*

Note that POOL$(\mathbf{X}, \mathbf{A})$ involves TopKPool (Gao & Ji, 2019), SAGPool (Lee et al., 2019), GASPool (Zhang et al., 2020a), and ASAP (Ranjan et al., 2020). And POOL-MID$(\mathbf{X}, \mathbf{A}, p, h)$ refers to the backbone pooling models combined with our proposed MID. Corollary 1 indicates that our proposed MID could promote the backbone pooling models to be more powerful in distinguishing different graphs.

## 5 Experiment

**Datasets.** We choose 13 datasets from TU datasets (Morris et al., 2020), including 8 datasets on the Biochemical domain and 5 datasets on the Social domain. Besides, we select 4 relatively large



(a) MUTAG (b) PTC-MR (c) NCI1 (d) IMDB-BINARY

Figure 7: Parameter sensitivity of score drop ratio $p_s$ and score dimension $h$ on four datasets.

datasets from the OGB datasets (Hu et al., 2020). The above seventeen real-world datasets vary in content domains and dataset sizes. The dataset statistics are summarized in Table 10 in Appendix.

**Models.** Four representative node drop graph pooling methods are selected, including **1) TopKPool**, **2) SAGPool**, **3) ASAP**, and **4) GSAPool**. Besides, we further present eight graph pooling methods for comparision, including **5) Set2set**, **6) SortPool**, **7) DiffPool**, **8) EdgePool**, **9) MinCutPool**, **10) HaarPool**, **11) MemPool**, and **12) GMT**. The detailed descriptions of all models are explained in Appendix I.

**Implementation Details.** We use the same experimental settings on the 17 benchmark datasets following (Xu et al., 2019; Baek et al., 2021). We evaluate the model performance on TU datasets for 10-fold cross validation (Zhang et al., 2018; Xu et al., 2019; Baek et al., 2021), using accuracy for evaluation. We evaluate the performance on OGB datasets with their original data split settings (Hu et al., 2020) with ROC-AUC for evaluation metric. For a fair comparison, we fix the pooling ratio in each pooling layer for node drop pooling with and without MID. In addition, we follow the parameter settings for some comparing models if the settings are provided in the corresponding papers. Experimental details are described in Appendix J.

**Overall Results.** The accuracy results of our methods in Table 2 are averaged over 10 runs with 10 different seeds through a 10-fold cross validation. We highlight the best performance per backbone model and dataset. It is observed that MID consistently improves the accuracy of node drop pooling models on all datasets, except for only a single exception of TopKPool on REDDIT-BINARY. Specifically, MID achieves improvements over all 4 node drop pooling models (averaged across datasets): 3.96% (SAGPool), 3.97% (TopKPool), 0.94% (ASAP) and 3.46% (GSAPool).

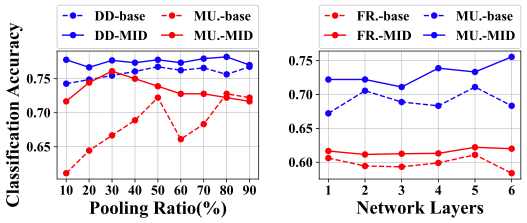
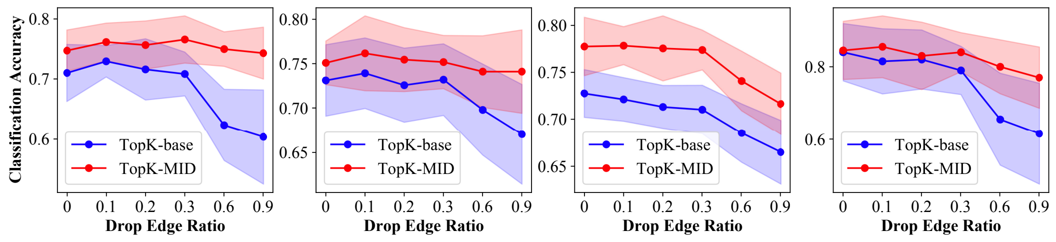


Figure 6: Model performance varying with the pooling ratio and number of model layers.

**Parameter Sensitivity.** We further study how the number of neural network layers $l$ and graph pooling ratio $k$ would affect the graph classification performance on D&D and MUTAG datasets with the SAGPool model. As shown in Figure 6 (a), our method performs better in all cases. In addition, the accuracy range of our proposed MID is relatively smaller, suggesting that our method learns the nodes that are essential for graph-level representation learning regardless of the pooling ratio. Then, in Figure 6 (b), we can observe that the performance of baseline models drops when layers go deeper (especially from 5 layers to 6 layers). In contrast, the performance of our method keeps growing. Afterward, we study the effects of two new parameters, the dimension of the score $h$ and the score dropping rate $p_s$, introduced by our proposed method with the SAGPool model. In this parameter sensitivity study, $h$ was searched within the range of $\{4, 8, 12, 16, 20\}$, while in the experiments, the search space is only $\{5, 9\}$. $p_s$ was searched within the range of $\{0.1, 0.2, ..., 0.9\}$, while in all other experiments, the search space was only $\{0.1, 0.2\}$ for all datasets. As shown in Figure 7, Biochemical datasets (subfigure a, b, and c) prefer a relatively high score dimension and a relatively low score drop ratio, while social datasets (subfigure d) are at the opposite ends.



(a) DD (Avg # edges 715.66) (b) PROTEINS (Avg # edges 72.82) (c) NCI1 (Avg # edges 32.30) (d) MUTAG (Avg # edges 19.79)

Figure 8: The robustness of our method against edge perturbations using graph classification results of different drop edge ratios. Solid lines denote the mean, and shaded areas denote the variance.

**Ablation Study.** We conduct ablation studies to verify that each component of our method contributes to the improvement of performance. For convenience, we name the models without multidimensional score space, flipscore, and dropscore as **w/o multi**, **w/o flip**, and **w/o drop**, respectively. Note that except for the selected component, the rest remains the same as the complete model. We can observe that all variants with some components removed exhibit clear performance drops compared to the complete model in Table 3, indicating that each component contributes to the improvements.

Table 3: Ablation study results.

| | NCI1 | | IMDB-B | |
|---|---|---|---|---|
| | SAG. | TopK. | SAG. | TopK. |
| Base | 70.89 | 73.82 | 71.87 | 71.47 |
| MID | **76.06** | **78.78** | **76.10** | **73.90** |
| w/o flip | 74.76 | 77.22 | 75.90 | 73.80 |
| w/o drop | 74.96 | 78.39 | 74.92 | 72.22 |
| w/o multi | 71.97 | 76.15 | 74.51 | 73.50 |

**Robustness Analysis.** We validate the robustness of our method against edge perturbations, which are perturbing the structure by randomly removing the edges in the pooling layer. As shown in Figure 8, we present the classification accuracies of two backbone models with respect to different perturbation rates on four benchmark datasets. We observe that our method consistently outperforms backbone models across all perturbation rates and all datasets. Especially in D&D and PROTEINS datasets whose edge numbers are large, the performance of our method does not degrade significantly with the increase of the perturbation rate compared to the backbone models.

**Efficiency Analysis.** Our method is a plug-in method. Therefore, it is essential to take the efficiency into account. The efficiency of our method is mainly influenced by the dimension of score $h$. The flipscore and dropscore operations rarely influence the time cost. Table 4 reports the average per-epoch training time on all 13 datasets from TU datasets, including different samples and graph sizes under different values of $h$ and with training epochs fixed to 10 for 10 different random seeds. We name the models with $h$-dimension score as MID$-h$. We observe that the additional time consumption keeps relatively small with the increase of the score dimension $h$, which validates that our MID is practically efficient. In addition, from Figure 7, we observe that increasing $h$ can significantly improve the model's classification accuracy at the cost of its training efficiency. In practice, we can adjust the values of $h$ to balance the trade-off between performance and efficiency.

Table 4: Training time per epoch.

| | SAGPool | TopKPool |
|---|---|---|
| Base | 0.8358 (1x) | 0.7985 (1x) |
| MID-1 | 0.8487 (1.01x) | 0.8005 (1.00x) |
| MID-5 | 0.9881 (1.18x) | 0.9688 (1.21x) |
| MID-9 | 1.2020 (1.44x) | 1.2052 (1.51x) |

## 6 Conclusion

In this work, we propose an efficient scheme, MID, which improves node drop pooling by exploring node-feature diversity and graph-structure diversity. Specifically, we first build a multidimensional score space to depict more comprehensive semantic information. Then, two operations on the multidimensional scores, flipscore and dropscore, are devised to reserve the feature diversity by highlighting dissimilar features and to maintain the structure diversity by covering more substructures. Extensive experiments on seventeen benchmark datasets involving different domains, samples, and graph sizes have verified that MID can generally and consistently improve the performance of current node drop pooling models (*e.g.*, TopKPool, SAGPool, GSAPool, and ASAP).

## REFERENCES

Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate learning of graph representations with graph multiset pooling. In *International Conference on Learning Representations*, 2021.

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 874–883. PMLR, 13–18 Jul 2020.

Lei Cai and Shuiwang Ji. A multi-scale approach for graph link prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:3308–3315, Apr. 2020.

Frederik Diehl. Edge contraction pooling for graph neural networks. *arXiv preprint arXiv:1905.10990*, 2019.

Jinlong Du, Senzhang Wang, Hao Miao, and Jiaqiang Zhang. Multi-channel pooling graph neural networks. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 1442–1448. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/199. Main Track.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pp. 2224–2232, Cambridge, MA, USA, 2015. MIT Press.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2020.

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural network for semi-supervised learning on graphs. In *NeurIPS'20*, 2020.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pp. 2083–2092. PMLR, 2019.

Hongyang Gao, Yi Liu, and Shuiwang Ji. Topology-aware graph pooling networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021a.

Xing Gao, Wenrui Dai, Chenglin Li, Hongkai Xiong, and Pascal Frossard. ipool–information-based pooling in hierarchical graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021b.

Zhangyang Gao, Haitao Lin, Stan Li, et al. Lookhops: light multi-order convolution and pooling for graph classification. *arXiv preprint arXiv:2012.15741*, 2020.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard T. B. Ma, Hongzhi Chen, and Ming-Chang Yang. Measuring and improving the use of graph information in graph neural networks. In *International Conference on Learning Representations*, 2020.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Jingjia Huang, Zhangheng Li, Nannan Li, Shan Liu, and Ge Li. Attpool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6480–6489, 2019.

John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. Zinc: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52 (7):1757–1768, 2012. doi: 10.1021/ci3001277. PMID: 22587354.

Amir Hosein Khasahmadi, Kaveh Hassani, Parsa Moradi, Leo Lee, and Quaid Morris. Memory-based graph networks. In *International Conference on Learning Representations*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International Conference on Machine Learning*, pp. 3734–3743. PMLR, 2019.

Juanhui Li, Yao Ma, Yiqi Wang, Charu Aggarwal, Chang-Dong Wang, and Jiliang Tang. Graph pooling with representativeness. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 302–311. IEEE, 2020a.

Maosen Li, Siheng Chen, Ya Zhang, and Ivor W. Tsang. Graph cross networks with vertex infomax pooling. In *NeurIPS*, 2020b.

Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 723–731, 2019.

Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. Path integral based convolution and pooling for graph neural networks. In *NeurIPS*, 2020.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001. doi: 10.1146/annurev.soc.27.1.415.

Diego Mesquita, Amauri Souza, and Samuel Kaski. Rethinking pooling in graph neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2220–2231. Curran Associates, Inc., 2020.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4663–4673. PMLR, 09–15 Jun 2019.

Abhishek Nadgeri, Anson Bastos, Kuldeep Singh, Isaiah Onando Mulang, Johannes Hoffart, Saeedeh Shekarpour, and Vijay Saraswat. Kgpool: Dynamic knowledge graph context selection for relation extraction. *ACL*, 2021.

Emmanuel Noutahi, Dominique Beaini, Julien Horwood, Sébastien Giguère, and Prudencio Tossou. Towards interpretable sparse graph representation learning with laplacian pooling. *arXiv preprint arXiv:1905.11577*, 2019.

Yunsheng Pang, Yunxiang Zhao, and Dongsheng Li. Graph pooling via coarsened graph infomax. SIGIR '21, pp. 2177–2181. Association for Computing Machinery, 2021. ISBN 9781450380379. doi: 10.1145/3404835.3463074.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=S1e2agrFvS`.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. *GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training*, pp. 1150–1160. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450379984.

Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 5470–5477, 2020.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying WEI, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. In *Advances in Neural Information Processing Systems*, volume 33, pp. 12559–12571. Curran Associates, Inc., 2020a.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020b.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Linda Studer, Jannis Wallau, Rolf Ingold, and Andreas Fischer. Effects of graph pooling layers on classification with graph neural networks. In *2020 7th Swiss Conference on Data Science (SDS)*, pp. 57–58, 2020. doi: 10.1109/SDS49233.2020.00021.

Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Yuanxing Ning, Philip S. Yu, and Lifang He. Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. In *Proceedings of the Web Conference 2021*, WWW '21, pp. 2081–2091. Association for Computing Machinery, 2021. ISBN 9781450383127. doi: 10.1145/3442381.3449822.

Haoteng Tang, Guixiang Ma, Lifang He, Heng Huang, and Liang Zhan. Commpool: An interpretable graph pooling framework for hierarchical graph representation learning. *Neural Networks*, 2021.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pp. 807–816. Association for Computing Machinery, 2009. ISBN 9781605584959.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Clément Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. In *NeurIPS*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/a32d7eeaae19821fd9ce317f3ce952a7-Abstract.html`.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *4th International Conference on Learning Representations, ICLR 2016*, 2016.

Yu Guang Wang, Ming Li, Zheng Ma, Guido Montufar, Xiaosheng Zhuang, and Yanan Fan. Haar graph pooling. In *International conference on machine learning*, pp. 9952–9962. PMLR, 2020.

Zhengyang Wang and Shuiwang Ji. Second-order pooling for graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4):1–38, 2010.

Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

Chuhan Wu, Fangzhao Wu, Yongfeng Huang, and Xing Xie. User-as-graph: User modeling with heterogeneous graph pooling for news recommendation. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 1624–1630. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/224. URL `https://doi.org/10.24963/ijcai.2021/224`. Main Track.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 6861–6871. PMLR, 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5812–5823. Curran Associates, Inc., 2020.

Hao Yuan and Shuiwang Ji. Structpool: Structured graph pooling via conditional random fields. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Liang Zhang, Xudong Wang, Hongsheng Li, Guangming Zhu, Peiyi Shen, Ping Li, Xiaoyuan Lu, Syed Afaq Ali Shah, and Mohammed Bennamoun. Structure-feature based graph self-adaptive pooling. In *Proceedings of The Web Conference 2020*, pp. 3098–3104, 2020a.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 5171–5181. Curran Associates Inc., 2018.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Xikun Zhang, Chang Xu, and Dacheng Tao. On dropping clusters to regularize graph convolutional neural networks. In *Computer Vision – ECCV 2020*, pp. 245–260, Cham, 2020b. Springer International Publishing. ISBN 978-3-030-58589-1.

Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. Hierarchical graph pooling with structure learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020c.

Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Zhao Li, Chengwei Yao, Dai Huifen, Zhi Yu, and Can Wang. Hierarchical multi-view graph pooling with structure learning. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021. doi: 10.1109/TKDE.2021.3090664.

## A NOTATIONS

The notations used in this paper are explained in Table 5.

Table 5: Commonly used notations and their descriptions.

| Notation | Description | Notation | Description |
|---|---|---|---|
| $\mathcal{G}$ | A graph | $\mathcal{E}$ | The set of edges in a graph |
| $\mathcal{V}$ | The set of nodes in a graph | $\boldsymbol{X}/\boldsymbol{X}^l$ | The matrix of node features (in layer $l$) |
| $\boldsymbol{S}/\boldsymbol{S}^l$ | The matrix of scores generated by the pooling models (in layer $l$) | $\mathrm{idx}/\mathrm{idx}^l$ | The preserving node indexes (in layer $l$) |
| $n$ | Number of nodes in a graph | $c$ | The dimension of a node feature |
| $k$ | The pooling ratio | $p_s$ | The score dropping rate |
| $h$ | The dimension of a score for a node | $\odot$ | Dot product |
| $\|\cdot\|_1$ | Manhattan norm ($L_1$ norm) | $\|$ | Vector concatenation |
| $\|\cdot\|_2$ | Euclidean Distance ($L_2$ norm) | $\lceil\cdot\rceil$ | The operation of rounding up |

## B RELATED WORK OF GRAPH NEURAL NETWORKS

There have been numerous researches on Graph Neural Networks (GNNs) to solve graph-based tasks. Typical GNNs include graph convolution networks (Kipf & Welling, 2017; Hamilton et al., 2017) and graph attention networks (Veličković et al., 2018) to pass messages between nodes through convolutions and attentions, respectively. GNNs utilize a neighborhood aggregation scheme to learn a node representation vector $x_v$ for each node $v$. The framework consists of two functions, AGGREGATE and COMBINE:

$$x_v^{(l)} = \mathrm{COMBINE}^{(l)}\left(\left\{x_v^{(l-1)}, \mathrm{AGGREGATE}^{(l)}\left(\left\{x_{v'}^{(l-1)} : v' \in \mathcal{N}_v\right\}\right)\right\}\right), \tag{5}$$

where $l$ is the round of aggregation, and $\mathcal{N}_v$ denotes the set of neighbors of a node $v$.

## C EXPLANATION OF NODE-FEATURE AND GRAPH-STRUCTURE DIVERSITIES

As shown in Figure 9, the input graph contains three types of node features highlighted in different colors and two kinds of graph structures, triangular structure (*e.g.*, nodes E, D and F) and dumbbell-shaped structure (*e.g.*, nodes B and C).

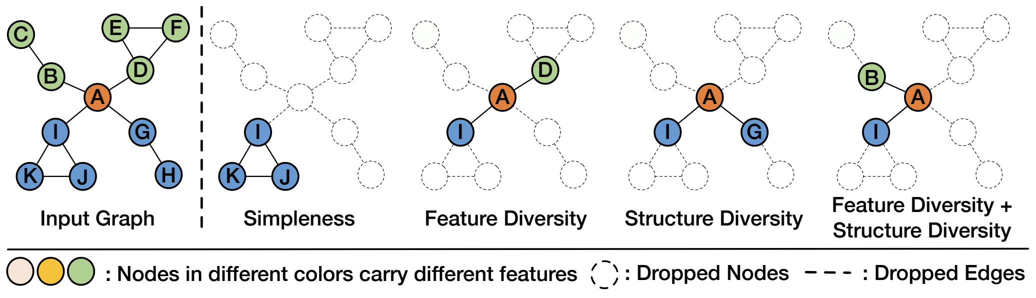


Figure 9: Illustration of node-feature diversity and graph-structure diversity. Nodes highlighted in the same color carry similar feature while nodes in different colors carry dissimilar features.

Suppose we only maintain three nodes, and blue nodes obtain higher significance scores using previous node drop pooling methods. Therefore, after pooling the graph, the reserved nodes may have four possible results shown in the right part: 1) nodes I, J, and K, which contain only one type of feature and one type of structures; 2) nodes I, A, and D, which contain all types of features; 3) nodes I, A, G, which include all kinds of structures; and 4) nodes I, A, and B, which cover all types of features and all kinds of structures, respectively. According to the analysis in Section 3, current node drop pooling models tend to preserve the nearby nodes, *i.e.*, the nodes of Simpleness, resulting in losing the node-feature diversity and graph structure diversity. We expect that our method MID could promote models to capture more diversity information, *i.e.,* preserving the nodes I, A, and B, when pooling the graph.

## D Discussion of Dropscore

**Dropout** (Srivastava et al., 2014) is first proposed to regularize fully connected networks by randomly setting feature dimensions to be zeros. To mitigate over-fitting and over-smoothing, **DropEdge** (Rong et al., 2020b) generates the dropout to graph convolution networks (GCNs) by randomly removing a certain number of edges from an input graph in each training epoch. Then, **DropNode** (Feng et al., 2020) also uses the dropout trick by randomly selecting and dropping the entire nodes to improve the generalization ability of GCNs. To better regularize GCNs, **DropCluster** (Zhang et al., 2020b) consider spatial and depth-wise correlations when dropping entries. **Dropscore** is different from the above-mentioned methods. Specifically, dropscore randomly drops out several nodes with a certain rate during training only when selecting top-k scores, aiming to promote models to cover as many substructures of a graph as possible. Except for the drop nodes, dropscore will not influence the feature represenation of the rest nodes and the process of graph coarsor.

## E Additional Analysis and Experimental Results

### E.1 Graph Reconstruction

We further validate our method on the graph reconstruction task, which reveals how much meaningful information is reserved during pooling.

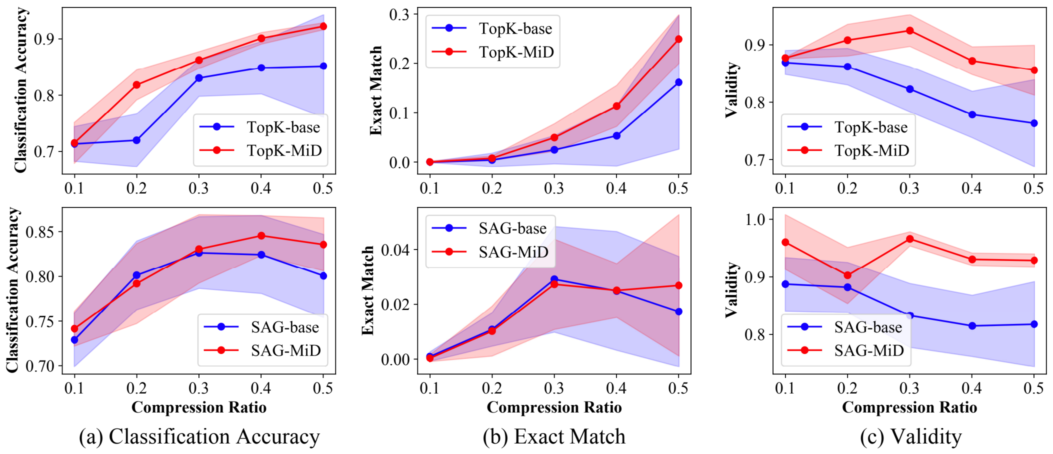


Figure 10: Reconstruction results on the ZINC dataset with different pooling ratios. Solid lines denote the mean, and the shaded areas denote the variance.

**Datasets.** We conduct experiments on a real-world dataset, ZINC (Irwin et al., 2012), which contains 12K molecular graphs.

**Models.** We evaluate our method MID by applying it to TopKPool and SAGPool models. Following MinCutPool and GMT (Bianchi et al., 2020; Baek et al., 2021), we use two graph convolution layers both before the pooling operation and the unpooling operation as illustrated in the Figure.

The pooling operations are TopKPool and SAGPool, and the unpooling operation is proposed in the graph U-Net (Gao & Ji, 2019).

**Experimental Settings.** We perform all experiments 10 times with 10 random seeds from 42 to 51, and then report the average results with the standard deviation. We strictly follow the dataset splitting provided by (Dwivedi et al., 2020) and use the early stopping with the patience as 50 on the validation loss. We set the maximum number of epochs as 500, the batch size as 128, the hidden dimension as 32, and the pooling ratio of all models as 10%, 20%, 30%, 40%, and 50%. The rest hyperparameters remain the same as those of the graph classification task.

**Performance Measure.** We use three metrics suggested by (Baek et al., 2021) as follows: a) Classification accuracy refers to the classification accuracy of atom types of all nodes. b) Exact match indicates the number of reconstructed molecules that are the same as the original molecules. c) Validity means the number of reconstructed molecules that are chemically valid.

**Results.** The detailed results are shown in Figure 10. We can observe that the models using MID significantly improve the performance in the **validity** metric, which means MID enables the models to capture more meaningful nodes in the original molecules. Besides, our method improves the performance of backbone models in the Accuracy and Exact match metrics. In a nutshell, our method MID makes the models more powerful to capture significant semantic information for the reconstruction of the original graph.

## E.2 Node Classification

We also validate our method on the graph classification task by using the architecture proposed by graph U-Net (Gao & Ji, 2019). As our method MID aims to maintain more dissimilar features, we conduct our experiments on four heterophily datasets.

Table 6: Node classification results on four benchmark heterophily datasets.

| | **Texas** | **Wisconsin** | **Actor** | **Cornell** |
|---|---|---|---|---|
| Hom ratio $r_{hom}$ | 0.11 | 0.21 | 0.22 | 0.3 |
| # Nodes $\|\mathcal{V}\|$ | 183 | 251 | 7,600 | 183 |
| # Edges $\|\mathcal{E}\|$ | 295 | 466 | 26,752 | 280 |
| # Classes $\|\mathcal{Y}\|$ | 5 | 5 | 5 | 5 |
| SAGPool-base | 59.81 ± 0.36 | 53.78 ± 1.25 | 29.67 ± 0.16 | 58.51 ± 0.93 |
| SAGPool-MID | **61.16 ± 0.44** | **55.35 ± 1.12** | **29.73 ± 0.14** | **59.70 ± 1.07** |

**Datasets.** We conduct experiments on four benchmark heterophily datasets: Texas, Wisconsin, Actor, and Cornell (Pei et al., 2020; Tang et al., 2009). We summarize the statistics of datasets in Table 6. The edge homophily ratio $r_{hom}$ is defined as the fraction of edges in a graph that connects nodes with the same class label :

$$r_{hom} = \frac{|\{(u, v) : (u, v) \in \mathcal{E} \wedge y_u = y_v\}|}{|\mathcal{E}|}, \tag{6}$$

where $y_u$ and $y_v$ are the class labels for nodes $u$ and $v$, respectively.

**Models.** We evaluate our MID by applying it to the SAGPool model, and the architecture that we use for node classification is the graph U-Net (Gao & Ji, 2019).

**Experimental Settings.** We perform all experiments 10 times with 10 random seeds from 42 to 51, and then report the average results with the standard deviation. We strictly follow the dataset splitting provided by (Pei et al., 2020). We set the maximum number of epochs as 200, the batch size as 128, the hidden dimension as 128, and the pooling ratio as 50%.

**Performance Measure.** We use classification accuracy to evaluate the performance.

**Results.** The detailed results are shown in Table 6. We can observe that the SAGPool model combined with MID outperforms backbone models across all datasets.

### E.3 GENERALIZATION IMPROVEMENT

We are concerned about MID's ability to be generalized to larger and more complex graphs. Therefore, we test MID on graphs whose sizes are larger than the graph sizes during training following the previous study (Knyazev et al., 2019).

Table 7: Generalization study results.

| | **COLOR** | | **TRIANGLES** | |
|---|---|---|---|---|
| | test-origin | test-large | test-origin | test-large |
| # nodes train | 4-25 | 4-25 | 4-25 | 4-25 |
| # nodes test | 4-25 | 25-200 | 4-25 | 25-100 |
| SAG.-base | $58.98_{\pm 3.43}$ | $9.68_{\pm 0.60}$ | $44.73_{\pm 0.63}$ | $19.44_{\pm 1.18}$ |
| SAG.-MID | **72.19**$_{\pm 1.59}$ | **11.24**$_{\pm 2.51}$ | **59.28**$_{\pm 2.05}$ | **22.36**$_{\pm 0.97}$ |
| TopK.-base | $70.38_{\pm 5.22}$ | $30.24_{\pm 22.75}$ | $45.00_{\pm 1.06}$ | $17.88_{\pm 1.75}$ |
| TopK.-MID | **99.96**$_{\pm 0.03}$ | **82.64**$_{\pm 15.21}$ | **47.30**$_{\pm 2.16}$ | **23.85**$_{\pm 1.14}$ |

**Datasets.** We conduct experiments on two synthetic datasets, COLORS and TRIANGLES (Knyazev et al., 2019).

**Models.** We evaluate our method MID by applying it to TopKPool and SAGPool models. The architecture of the models is the same as the one in the graph classification task.

**Experimental Settings.** As shown in Tabel 8, we strictly follow the experimental settings suggested by (Knyazev et al., 2019).

Table 8: Dataset statistics and model hyperparameters for our generalized experiments.

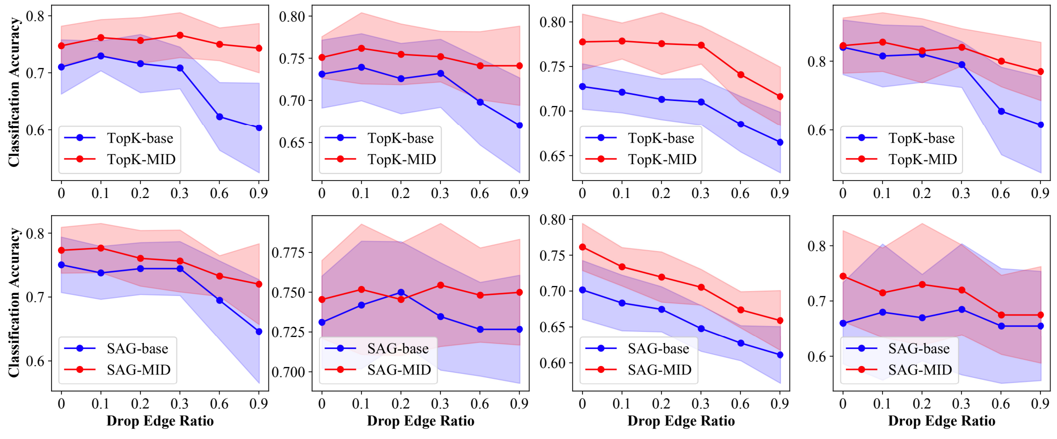
| | **COLORS** | **TRIANGELS** |
|---|---|---|
| # num graphs | 5,500 | 40,000 |
| # classes | 11 | 10 |
| # train graphs | 500 | 30,000 |
| # val graphs | 2,500 | 5,000 |
| # test graphs origin | 2,500 | 5,000 |
| # test graphs large | 2,500 | 5,000 |
| # nodes train/val | 4-25 | 4-25 |
| # nodes test origin | 4-25 | 4-25 |
| # nodes test large | 25-200 | 25-100 |
| Convolution layers | 2 | 3 |
| Convolution layer type | GIN Aggregator<br>2 layer MLP with 64 hid. units | GIN aggregator<br>2 layer MLP with 64 hid. units |
| Pooling layers | 1 | 2 |
| Pooling layer type | SAGPool<br>min score 0.05, 64 hid. units | SAGPool<br>min score 0.001, 64 hid. units |
| READOUT layer | global sum | global max |
| Linear layers | 1 | 1 |
| seeds | 20: [42-61] | 10 : [42-51] |
| Training params | epochs: 100, dropout: 0.5, batch size: 32<br>lr: 1e-3, weigh decay: 1e-4 | epochs: 100, dropout: 0.5, batch size: 32<br>lr: 1e-3, weigh decay: 1e-4 |

**Performance Measure.** We use the regression accuracy between the predicted labels and ground-truth labels as our performance measure.

**Results.** The results in Table 7 demonstrate that our method significantly improves the performance across all cases and datasets. Furthermore, when our method is combined with the TopKPool model on the COLOR dataset, the accuracy does not degrade significantly when being generalized to graphs with larger sizes during the test.

### E.4 Robustness Analysis

We present additional experiment results of robustness analysis in Section 5. As shown in Figure 11, backbone models combined with our method MID consistently perform better. In most cases, the gap gradually gets bigger with the increase of the drop edge ratios.

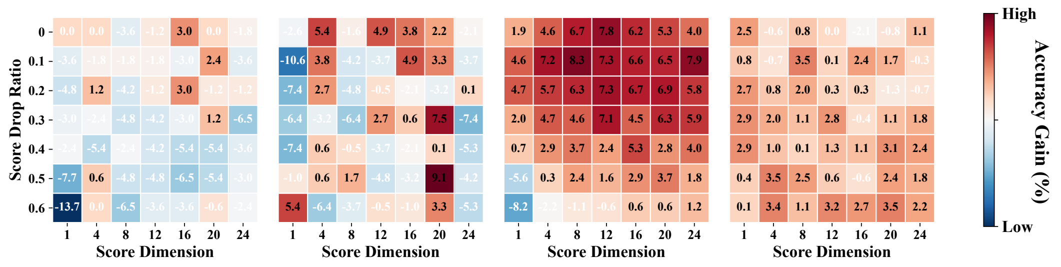


(a) DD (Avg # edges 715.66) (b) PROTEINS (Avg # edges 72.82) (c) NCI1 (Avg # edges 32.30) (d) MUTAG (Avg # edges 19.79)

Figure 11: The robustness of our method against edge perturbations. Graph classification results on four benchmark datasets by varying the drop edge ratio. Solid lines denote the mean, and shaded areas denote the variance.

### E.5 Parameter Analysis

We present more results of parameter analysis.



(a) MUTAG (b) PTC-MR (c) NCI1 (d) IMDB-BINARY

Figure 12: Parameter sensitivity of score drop ratio $p$ and score dimension $h$ on four datasets with the TopKPool model.

## F Proofs

In this section, we present the detailed proofs for all theoretical analysis in Section 4.

### F.1 PROOFS OF TRAPPED SCORES

We prove theoretically that, under some conditions, the selected nodes of most node drop pooling models are stuck in the local structure.

**Proposition.** *1. Consider a graph $\mathcal{G}$ with the adjacency matrix $\mathbf{A}$, $\mathbf{x}_v$ is the feature for node $v$. Assume that linked nodes in the same local structure have similar features. Let $\mathbf{x}_1, \mathbf{x}_2$ be node features and $\mathbf{s}_1, \mathbf{s}_2$ be the significance scores, then, the distance between $\mathbf{s}_1$ and $\mathbf{s}_2$ approximates to zero, i.e., $\|\mathbf{s}_1 - \mathbf{s}_2\| \to 0$, as long as $\|\mathbf{x}_1 - \mathbf{x}_2\| \to 0$.*

*Proof.* Take the TopKPool model as an example. TopKPool model predicts scores through:

$$\mathbf{S} = \mathbf{X}\frac{\mathbf{p}}{\|\mathbf{p}\|}, \tag{7}$$

where $\mathbf{p}$ is a learnable vector. Given two features $\mathbf{x}_1$ and $\mathbf{x}_2$ for nodes $u$ and $v$, the scores of two nodes are:

$$\mathbf{s}_1 = \mathbf{x}_1\frac{\mathbf{p}^T}{\|\mathbf{p}\|}, \quad \mathbf{s}_2 = \mathbf{x}_2\frac{\mathbf{p}^T}{\|\mathbf{p}\|}. \tag{8}$$

Then, we have:

$$\|\mathbf{s}_1 - \mathbf{s}_2\| = \left\|\mathbf{x}_1\frac{\mathbf{p}^T}{\|\mathbf{p}\|} - \mathbf{x}_2\frac{\mathbf{p}^T}{\|\mathbf{p}\|}\right\| \leq \|\mathbf{x}_1 - \mathbf{x}_2\|. \tag{9}$$

Therefore, based on Eq. (9) and $\|\mathbf{x}_1 - \mathbf{x}_2\| \to 0$, we get:

$$\|\mathbf{s}_1 - \mathbf{s}_2\| \to 0. \tag{10}$$

□

### F.2 PROOFS OF GRAPH PERMUTATION EQUIVARIANCE

Graph pooling should generate isomorphic pooled graphs after graph permutation, which is defined as graph permutation equivariance. We first give a definition of graph permutation and graph permutation equivariance.

**Definition 1.** *Given a permutation matrix $\mathbf{P} \in \{0,1\}^{n\times n}$ and it further satisfies $P1 = 1$ and $\mathbf{P}^T\mathbf{1} = 1$ which means there exists exactly one nonzero entry per row and column of $P$, a pooling function $f(\mathbf{X}, \mathbf{A})$, then graph permutation is $f\left(\mathbf{PX}, \mathbf{PAP}^T\right)$, which consists of node-feature permutation $f\left(\mathbf{PX}, \mathbf{A}\right)$ and graph-structure permutation $f\left(\mathbf{X}, \mathbf{PAP}^T\right)$.*

**Definition 2.** *Graph permutation equivariance is $f(\mathbf{PX}, \mathbf{PAP}^\top) = \mathbf{P}^\top f(\mathbf{X}, \mathbf{A})$.*

We first introduce a theorem from Vignac et al. (2020).

**Theorem 2 (Permutation Equivariance of Graph Neural Networks).** *Give a GNN function $\phi$, then we have:*

$$\phi(\mathbf{PX}, \mathbf{PAP}^\top) = \mathbf{P}^\top\phi(\mathbf{X}, \mathbf{A}). \tag{11}$$

*Proof.* The detailed proofs of Theorem 2 can be found in Vignac et al. (2020). □

Then, we show that our MID would not break the graph permutation equivariant property of backbone models. We take SAGPool as an example to prove Proposition 2.

**Proposition.** ***2.** Suppose the backbone graph pooling model is graph permutation equivariant, the model combined with our MID is still graph permutation equivariant.*

*Proof.* The SAGPool model predict scores through a GNNs layer:

$$\mathbf{S} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\boldsymbol{\Theta}\right), \tag{12}$$

where $\mathbf{S}$ is the scores for nodes, $\sigma$ is the activation function (tanh), $\tilde{\mathbf{A}}$ is the adjacency matrix with self-connections ( $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$), $\tilde{\mathbf{D}}$ is the degree matrix, and $\boldsymbol{\Theta}$ is the matrix of parameters.

According to Theorem 2 and Eq. 12, we have:

$$\hat{\mathbf{S}} = \mathbf{P}^{\top}\mathbf{S}, \tag{13}$$

where $\hat{\mathbf{S}} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{P}\tilde{\mathbf{A}}\mathbf{P}^{\top}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{P}\mathbf{X}\boldsymbol{\Theta})$. Then, selecting top $\lceil k \times n \rceil$ nodes according to $\hat{\mathbf{S}}$ denoted by indices idx. And using Eq. (13) and $\mathbf{X}^p = \hat{\mathbf{X}}(\mathrm{idx})$, $\mathbf{A}^p = \hat{\mathbf{A}}(\mathrm{idx})$, we can write:

$$\begin{aligned} &\mathbf{X}^p \rightarrow \mathbf{P}[\mathrm{idx}, \mathrm{idx}]\mathbf{X}^p, \\ &\mathbf{A}^p \rightarrow \mathbf{P}[\mathrm{idx}, \mathrm{idx}]\mathbf{A}^p(\mathbf{P}[\mathrm{idx}, \mathrm{idx}])^T \end{aligned} \tag{14}$$

We see that graph permutation does not change the output features. Our method MID as formulated in Eq. (15) only makes changes in selecting top $\lceil k \times n \rceil$ nodes and is not affected by the input order.

$$\mathbf{S}_{\text{multi}} = \mathrm{SCORE}(\mathbf{X}, \mathbf{A}), \quad \mathbf{S}_{L_1} = \|\mathbf{S}_{\text{mutli}}\|_1, \quad \mathbf{S}_{\text{drop}} = \mathbf{I}_{\lceil p_s \times n \rceil}\mathbf{S}_{\text{multi}}. \tag{15}$$

Therefore, from Eq. (14) and Eq. (15), we conclude that our method MID would not break the graph permutation equivariant property of backbone models.

□

### F.3 Connection with Weisfeiler-Lehman Graph Isomorphism Test

Weisfeiler-Lehman (WL) test of graph isomorphism (Weisfeiler & Leman, 1968) provides an effective solution to distinguish different graphs. According to previous studies (Xu et al., 2019), a GNN can have as large discriminative power as the WL test if the aggregation scheme is an injective function. The overall model for graph classification consists of two phases: graph convolution and graph pooling. If we make the pooling function injective, then the model can be as powerful as the WL test in distinguishing different graphs. In the following, we demonstrate that the proposed MID could promote the backbone pooling models to be more powerful in distinguishing different graphs. We first introduce the theorem from Xu et al. (2019).

**Theorem.** *1. **(Non-isomorphic Graphs to Different Embeddings)** Let $\mathcal{A} : \mathcal{G} \rightarrow \mathbb{R}^d$ be a GNN. If the aggregation function and the graph-level readout function are injective, $\mathcal{A}$ can distinguish any two non-isomorphic graphs $G_1$ and $G_2$, which are decided by the Weisfeiler-Lehman test of isomorphism.*

*Proof.* The detailed proofs of Theorem 1 can be found in Appendix B of (Xu et al., 2019). □

According to the Theorem 1, the key of the expressive power is the injectiveness of function, which means the outputs of the function are more unique. Our next corollary shows that our proposed MID enables the outputs of backbone pooling models to be more powerful.

**Corollary.** *1. **(Expressive power on Node Drop Pooling Models)** Assume the feature space $\mathcal{X}$ is a countable set. Let $\boldsymbol{X} \in \mathcal{X}$ be the matrix of node features, $A$ be the adjacency matrix, $p_s$ be the score dropping ratio, and $h$ be the score dimension. The output of POOL-MID$(\mathbf{X}, \mathbf{A}, p_s, h)$ with our method MID could be more unique than the output of backbone pooling models POOL$(\mathbf{X}, \mathbf{A})$. Then, the backbone pooling models with MID could approximately be as powerful as the WL test.*

*Proof.* According to our analysis in Section 3, scores generated by modern node drop pooling models tend to be stuck in the local structures, which would cause the models failing to distinguish two different graphs. As shown in Figure 13 (a) and (b), if modern pooling models keep nodes A, B, C

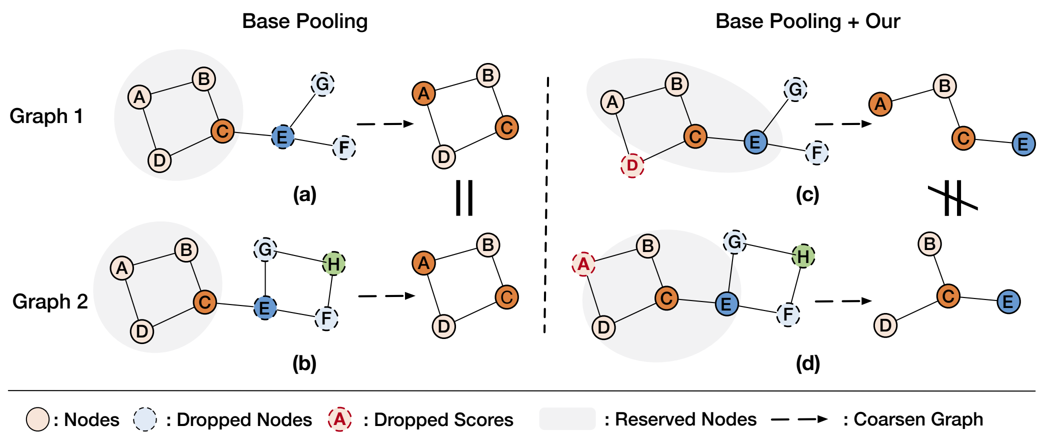


Figure 13: Our method enable the pooled graph of backbone pooling methods to be more unique.

and D in the pooled graph both in Graph 1 and Graph 2, then the outputs of the above two graphs are the same, which indicates that models can not distinguish the two different graphs.

Then, we consider the models combined with our method MID. Specifically, the dropscore operation of MID, $\mathbf{S}_{\text{drop}} = I_{\lceil p_s \times n \rceil} \mathbf{S}_{\text{multi}}$, could help make the outputs of two graphs more distinguishable. As shown in Figure 13 (c) and (d), even if models generate the same top-k scores for nodes in two graphs, our dropscore operation will randomly drop out certain rate of nodes, which increases the probability that the output of two graphs are different. Therefore, it confirms that our method MID makes the outputs of backbone pooling models more unique.

Moverover, we know that outputs of backbone pooling models combined with our MID are more unique, which means that the pooling function of backbone models combined with MID are more convergent to injective. According to the Theorem 1, the key of expressive power is the injectiveness of function. Therefore, our MID prompts models to be more approximately as powerful as the WL test. $\square$

## G Detailed Case Study Description for Each component

### G.1 Multidimension Score Space

**Datasets.** We select five benchmark datasets (MUTAG, ENZYMES,NCI, PROTEINS, and NCI109).

**Models.** We apply SAGPool with or without multidimension score space.

**Experimental Settings.** We use a random splitting with the fixed seed 777. Other experimental settings keep the same with the graph classification task in Section 5.

**Performance Measure.** To evaluate score correctness of models with pooling, we follow the idea from (Knyazev et al., 2019; Zeiler & Fergus, 2014). After training a model, we remove node $i$ and compute an absolute difference from prediction $y$ for the original graph:

$$\alpha_i^{WS} = \frac{|y_i - y|}{\sum_{j=1}^{N} |y_j - y|}, \tag{16}$$

where $\alpha_i^{WS}$ is the ground truth score and $y_i$ is a model's prediction for the graph without node $i$. After obtaining the ground truth scores and calculated scores from the first pooling layer in the test dataset, we calculate the score correctness by Rank-Biased Overlap which is a method to quantify the similarity between two ranked lists (Webber et al., 2010). It returns a lower-bound estimate and an upper bound estimate. The higher the values are, the better they are.

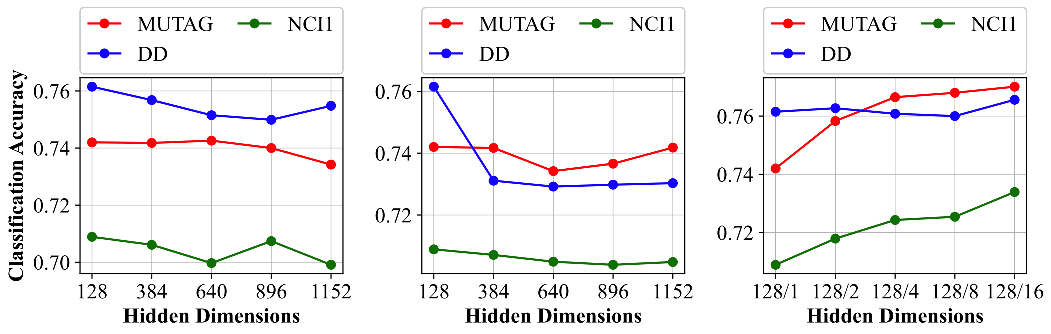
**Results.** The detailed results are shown in Table 9. We can observe that the multidimension score space operation consistently improves the attention correctness, while the flipscore operation only performs better in some situations.

### G.1.1 Sore Dimension Analysis

According to our analysis in Section 3, the multidimension score space operation would enlarge the dimension of embedding vectors. Therefore, we give an in-depth study on how the parameter, embedding dimension, affects the performance in the graph classification task. For a fair comparison, we design three experiments: 1) Repeat vectors. Raising the embedding dimension in base models by repeating the embedding vectors in the pooling layer. 2) MLP. Raising the embedding dimension in base models through adding one Multi-layer perceptron layer between the graph convolution layer and the pooling layer. 3) Multi-head. Fixing the embedding dimension as 128 with multidimension score space. Specifically, we first equalize the embedding vectors $\mathbf{X}^l$ into $h$ parts, $\mathbf{X}^l_1, \mathbf{X}^l_2, ..., \mathbf{X}^l_h$. Then the new features of nodes are calculated by:

$$\{\boldsymbol{X}_1^{(l)} \odot s_1^{(l)} \| \boldsymbol{X}_2^{(l)} \odot s_2^{(l)} \| ..., \| \boldsymbol{X}_h^{(l)} \odot s_h^{(l)}\}, \tag{17}$$

where $\|$ indicates the concatenation.



(a) Repeat Vectors (b) MLP (c) Multi-head

Figure 14: Accuracy results varying with different hidden dimensions. 128/2 means that the dimension of the embedding vector is 128, and the dimension of score, $h$, is 4.

**Datasets.** We select three benchmark datasets (MUTAG, DD, and NCI1), and the samples of these datasets are 188, 1,178, and 4,110, respectively.

**Models.** We choose SAGPool model as a baseline model for *Repeat vectors* and *MLP* experiments. When we conduct the *Multi-head* experiment, we only apply the multidimension score space operation.

**Experimental Settings.** The experimental settings are kept the same as those of graph classification task in Section 5.

**Performance Measure.** We use classification accuracy between the predicted labels and ground-truth labels as our performance measure. The model performance is evaluated on the test split of the 10 random splitting for all datasets and reported as the average and the standard deviation of the 10 scores.

**Results.** The detailed results are shown in the Figure 14. We can observe that raising the hidden dimensions in MUTAG, DD, and NCI1 datasets could not contribute to the improvement of the accuracy. On the contrary, the accuracy decades when the dimension increases. And in Figure 14 (c), results demonstrate that our multidimension score space operation contributes to the accuracy improvement even if we do not raise the hidden dimensions.

## G.2 The Flipscore Operation

**Datasets.** We conduct our experiments on the MUTAG dataset.

Table 9: Score correctness.

| | **MUTAG** | | **ENZYMES** | | **NCI1** | | **PROTEINS** | | **NCI109** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Lower | Upper | Lower | Upper | Lower | Upper | Lower | Upper | Lower | Upper |
| Base | 0.4069 | 0.4667 | 0.3012 | 0.3035 | 0.4198 | 0.4261 | 0.5956 | 0.8284 | 0.4341 | 0.5262 |
| Flip | **0.7405** | **0.8004** | 0.2872 | 0.2895 | 0.4255 | 0.4318 | 0.5773 | 0.8102 | 0.4967 | 0.5888 |
| Multi | 0.6521 | 0.7119 | **0.4120** | **0.4143** | **0.5554** | **0.5605** | **0.6769** | **0.9097** | **0.6097** | **0.7017** |

**Models.** We apply SAGPool and GSAPool models with or without multidimension score space. The architecture of the model is shown in Figure 17.

**Experimental Settings.** We use a random splitting with the fixed seed 777. Other experimental settings keep the same with the graph classification task in Section 5.

**Performance Measure.** According to the previous study (Gao et al., 2020; Hou et al., 2020), the information gain can be calculated by KL-divergence:

$$D_{KL}\left(S^{(k)} \| C^{(k)}\right) = \int_{\mathcal{X}_k} S^{(k)}(\boldsymbol{x}) \cdot \log \frac{S^{(k)}(\boldsymbol{x})}{C^{(k)}(\boldsymbol{x})} d\boldsymbol{x}. \tag{18}$$

According to the proof (The detailed proof can be found in (Gao et al., 2020; Hou et al., 2020)), $D_{KL}$ is positively correlated to :

$$\lambda = \frac{\left\| \sum_{v \in \mathcal{V}} \left( \sum_{v' \in \mathcal{N}_v} \left( x_v - x_{v'} \right) \right)^2 \right\|_1}{|\mathcal{E}| \cdot d}. \tag{19}$$

Then, we calculated the pooling information gain by function 19, but only calculated the value of selected nodes. We refer to it as information gain during one-time pooling operation.

### G.3 The Dropscore Operation

**Datasets.** We select five benchmark datasets (NCI1, ENZYMES, PROTEINS, IMDB-BINARY, COLLAB), including different graph sizes and different domains (social and bio-chemical).

**Models.** We apply SAGPool model with or without our method. The architecture of the model is shown in Figure 17.

**Experimental Settings.** We use a random splitting with the fixed seed 777, and the pooling ratio is set as 40%. We report the selected nodes in the first pooling layer of test set in the last epoch. For convenience, we set the batch size of test set as 1. Other experimental settings keep the same with the graph classification task in Section 5.

**Results.** The detailed results are shown in the Figure 15. The selected nodes are highlighted in red. We can observe our dropscore operation encourages models to cover more substructures in the graph.

## H Dataset Statistics

The detailed statistics of datasets are shown in Table 10.

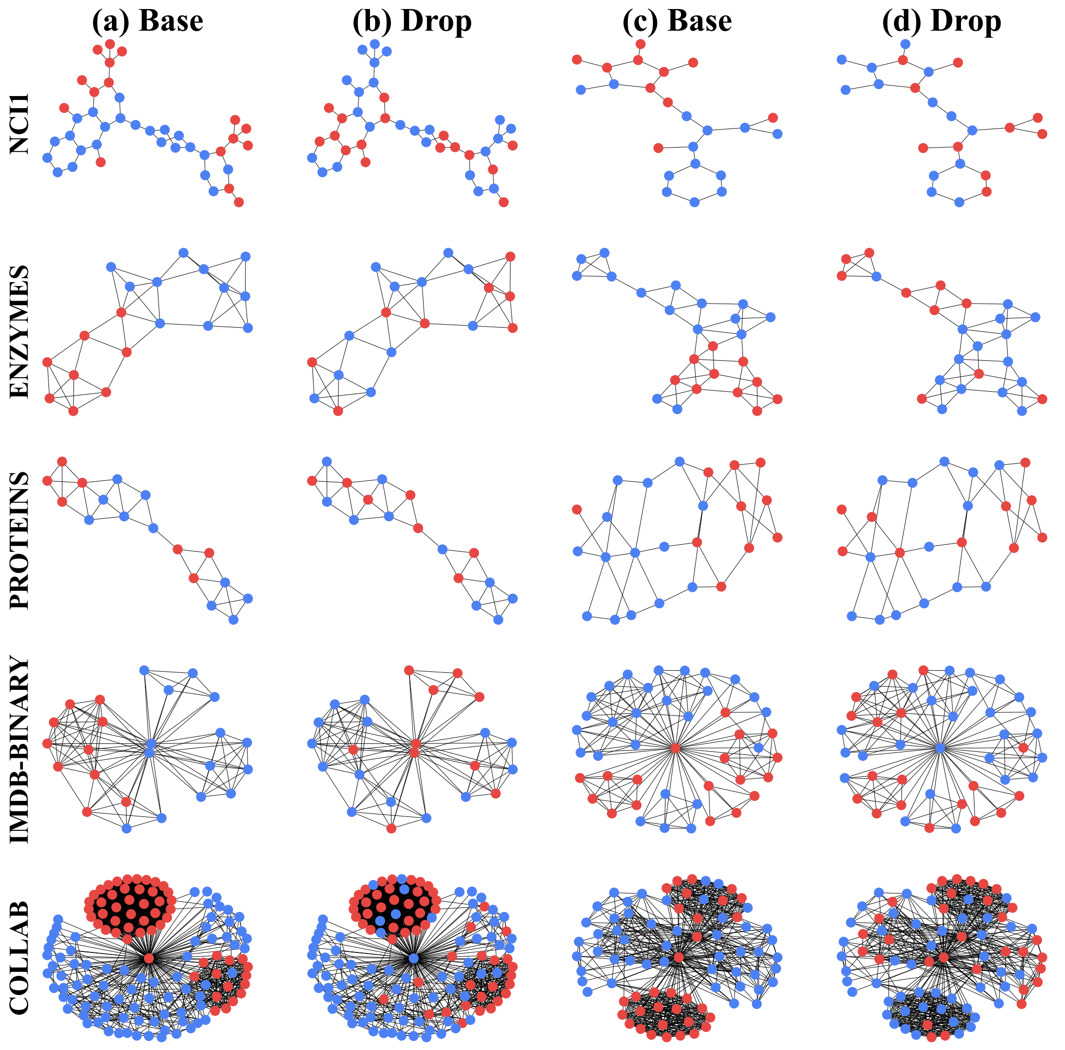


Figure 15: Detailed visualization of node selection results at first layer in SAGPool and SAGPool with dropscore operation. Selected nodes are highlighted in red.

## I Baseline and Backbone Models

### I.1 Backbones

**1) TopKPool (Gao & Ji, 2019).** This method select top-k nodes according to scores generated by a learnable function, which only considers the node features.

**2) SAGPool (Lee et al., 2019).** This method selects the important nodes by dropping unimportant nodes with lower scores that are generated by another graph convolutional layer, which involves the nodes features and graph structures. Particularly, this method has two variants. .1) SAGPool(G) is the global node drop method that drops unimportant nodes one time at the end of their architecture. .2) SAGPool(H) is the hierarchical node drop method that drops unimportant nodes sequentially with multiple graph convolutional layers. We Use SAGPool(H) in this paper.

**3) ASAP Ranjan et al. (2020).** This method clusters the neighboring nodes, and then drops the lower score clusters using a spetial scoring function.

Table 10: Statistics and properties of benchmark datasets

| | | # graphs | # classes | Avg # nodes | Avg # edges |
|---|---|---|---|---|---|
| TU Datasets | DD | 1,178 | 2 | 284.32 | 715.66 |
| | PROTEINS | 1,113 | 2 | 39.06 | 72.82 |
| | NCI1 | 4,110 | 2 | 29.87 | 32.30 |
| | MUTAG | 188 | 2 | 17.93 | 19.79 |
| | PTC-MR | 344 | 2 | 14.30 | 14.69 |
| | NCI109 | 4,127 | 2 | 29.68 | 32.13 |
| | ENZYMES | 600 | 6 | 32.63 | 124.20 |
| | Mutagenicity | 4,337 | 2 | 30.32 | 30.77 |
| | FRANKENSTEIN | 4,337 | 2 | 16.90 | 17.88 |
| | REDDIT-B | 2,000 | 2 | 429.63 | 497.75 |
| | IMDB-B | 1,000 | 2 | 19.77 | 96.53 |
| | IMDB-M | 1,500 | 3 | 13.00 | 65.94 |
| | COLLAB | 5,000 | 3 | 74.49 | 2457.78 |
| OGB Datasets | HIV | 41,127 | 2 | 25.51 | 27.52 |
| | Tox21 | 7,831 | 12 | 18.57 | 19.3 |
| | ToxCast | 8,576 | 617 | 18.78 | 19.3 |
| | BBBP | 2,039 | 2 | 24.06 | 26.0 |

**4) GSAPool (Zhang et al., 2020a).** This method generate scores from two perspectives: 1) Using MLP to capture the significant node features, 2) Using GNN to capture the significant graph structures. Then the model linearly combines the two scores.

### I.2 Baselines

**5) Set2set (Vinyals et al., 2016).** This method uses a recurrent neural network to encode a set of all nodes, with content-based attention over them. And the readout layer is unnecessary in this method.

**6) SortPool (Zhang et al., 2018)** This method drops unimportant nodes by sorting the final channel of their features, which are directly generated from the previous GNN layers. Motivated by WL colors, this method regards the last layer's output of GNN as most refined WL colors.

**7) DiffPool (Ying et al., 2018).** This method clusters similar nodes into the few new nodes through graph convolutional layers to coarsen the graph.

**8) MinCutPool (Bianchi et al., 2020).** This method clusters the node through the spectral clustering with GNNs, to coarsen the nodes and the adjacency matrix of a graph.

**9) HaarPool (Wang et al., 2020).** This method compresseses the node features with a nonlinear transformation in a Haar wavelet domain.

**10) EdgePool (Diehl, 2019).** This method calculates the scores for edges, and then gradually merges the nodes connected by the high score edges.

**11) MemPool (Khasahmadi et al., 2020).** This method proposes an efficient memory layer to jointly learn node representations and coarsen the graph.

**12 GMT (Baek et al., 2021).** Graph Multiset Transformer first condenses all nodes into the important nodes by GMPool, and then considers interactions between nodes in a set.

## J Experimental Settings

**Source Code.** Except for **HaarPool** and **HGP-SL** models, where we use the source codes provided by authors, for the rest of the models, we use the PyTorch Geometric library (Fey & Lenssen, 2019).

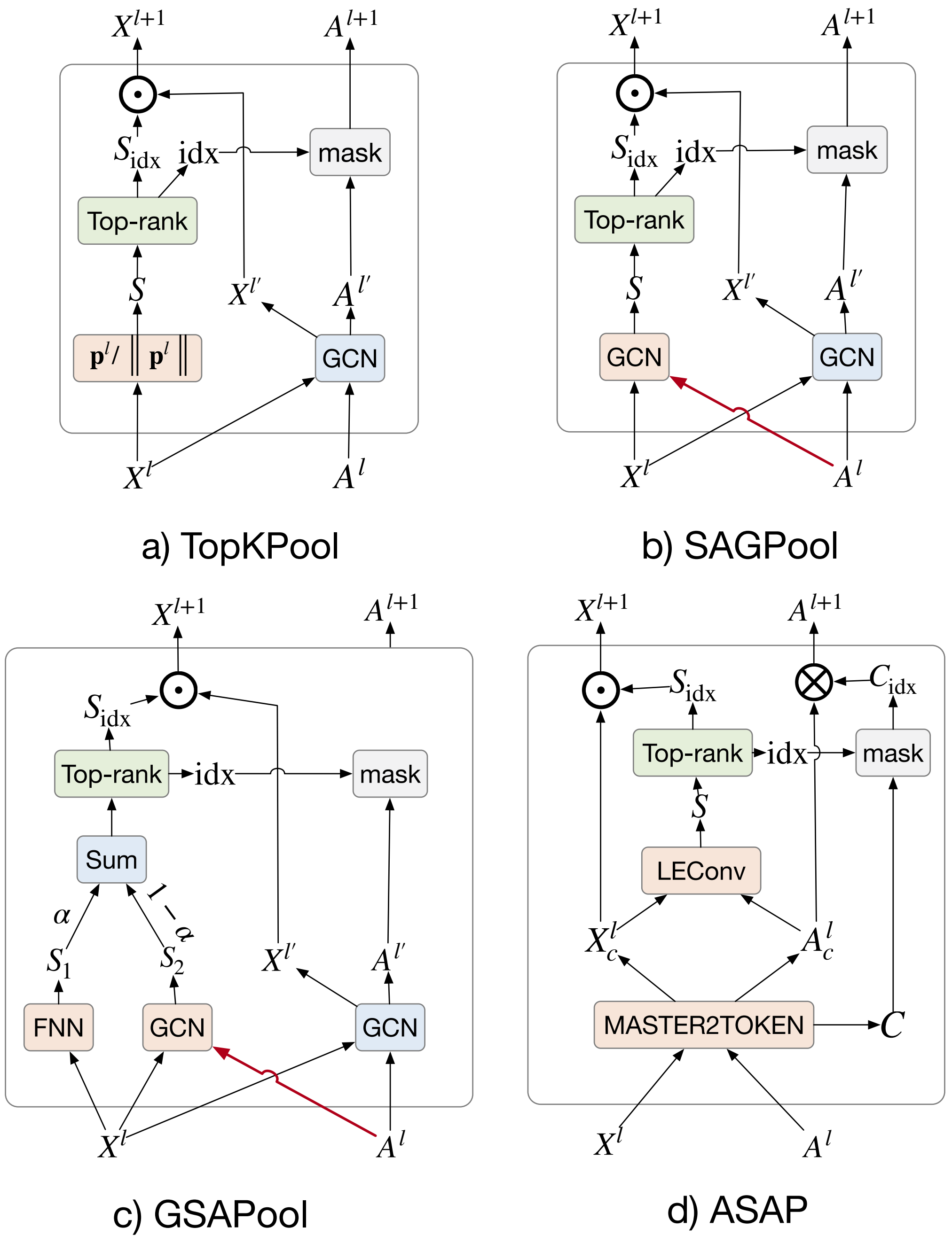


Figure 16: Backbone models

**Hardware Environments.** Each experiment was run on a single GPU (NVIDIA V100 with a 16 GB memory size) and experiments were run on the server at any given time.

**Software Environments.** All models are implemented with Python 3.7. It requires PyTorch 1.9.0 or above (which further requires CUDA 10.2 or above) and PyTorch-Geometric 1.7.3 or above.

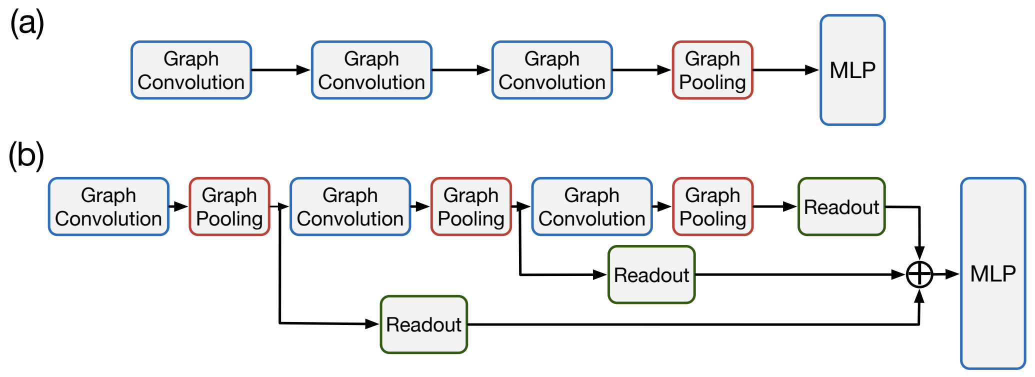


Figure 17: The illustration of model architectures.

### J.1 Graph Classification with TU Datasets

**Datasets.** We select 13 benchmark datasets from TU datasets containing different domains, samples, and graph sizes. D&D, PROTEINS,NCI1, MUTAG, PRC-MR, NCI109, ENZYNMES, and Mutagenicity datasets are on Biochemical domain, and FRANKENSTEIN, REDDIT-BINARY, IMDB-BINARY, IMDB-MULTI, and COLLAB are on Social domain. The detailed statistics and properties are described in Table 10.

**Models.** We apply 4 representative backbone models with or without our method(TopK, SAG-Pool, ASAP, and GSAPool). Following the architecture design of these methods, we adopt the architecture shown in Figure 17 **Bottom** for all 4 models. Also, We run 8 baseline models(Set2set, SortPool, DiffPool, MinCutPool, EdgePool, HaarPool, MemPool, and GMT). Among them, Diff-Pool, MinCutPool, EdgePool, and HaarPool use the hierarchical architecture(Figure 17 **Bottom**), while Set2set, SortPool, GMT use the global architecture(Figure 17 **Top**). The specific of each part in architecture for all models are described in Table 11.

**Implementation Details.** For all experiments on TU datasets, we evaluate the model performance with a 10-fold cross validation setting, where the dataset split is based on the conventionally used training/test splitsZhang et al. (2018); Xu et al. (2019); Baek et al. (2021). We use the early stopping criterion, where we stop the training if there is no further improvement on the validation loss during 50 epochs. We then report the average performances on the test sets, by performing overall experiments 10 times with different seeds from 42 to 51. We optimize the network with Adam optimizer (Kingma & Ba, 2014). Also, we set the pooling ratio as 50% in each pooling layer for both baselines and our models. The detailed hyperparameter settings for each model on each dataset are shown in Table 11.

**Performance Measure.** We use classification accuracy between the predicted labels and ground-truth labels as our performance measure. The model performance is evaluated on the test split of the 10 folds for all TU datasets, and reported as the average and the standard deviation of the 100 scores with 10 different seeds.

### J.2 Graph Classification with OGB Datasets

**Datasets.** We select 4 benchmark datasets from OGB datasets(HIV, Tox21, ToxCast, BBBP). The detailed statistics and properties are described in Table 10.

**Models.** We apply 4 representative backbone models with or without our method(TopK, SAG-Pool, ASAP, and GSAPool). Following the architecture design of these methods, we adopt the architecture shown in Figure 17 **Bottom** for all 4 models. Also, We run 8 baseline models(Set2set, SortPool, DiffPool, MinCutPool, EdgePool, HaarPool, MemPool, and GMT). Among them, Diff-

Table 11: Hyperparameter settings for TU expriments. The layer is the type of graph convolution; batch is the training batch size; drop is the ratio of dropout between linear layers; lr is the learning rate; weight is the weight decay; patience is the training patience.

| Model | Dataset | Hyperparameter | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Layer | batch | hidden | drop | lr | weight | epochs | patience | Other |
| TopKPool | All | GCN | 128 | 128 | 0.5 | 5E-04 | 1E-04 | 5E+05 | 50 | Readout: mean+max |
| SAGPool | All | GCN | 128 | 128 | 0.5 | 5E-04 | 1E-04 | 5E+05 | 50 | Readout: mean+max |
| GSAPool | All | GCN | 128 | 128 | 0.5 | 5E-04 | 1E-04 | 5E+05 | 50 | Readout: mean+max,<br>alpha: 0.4,<br>feature_fusion_type: GAT, |
| Set2Set | All | GCN | 128 | 128 | 0.5 | 5E-04 | 1E-04 | 5E+05 | 50 | 1 layer LSTM |
| SortPool | All | GCN | 128 | 128 | 0.5 | 1E-04 | 1E-04 | 5E+05 | 50 | K=30,<br>Conv1d(self.nhid, 32, 5) |
| DiffPool | All | DenseSAGEConv | 128 | 64 | 0 | 5E-04 | 1E-04 | 5E+05 | 50 | Loss = Total,<br>BatchNorm1d<br>Readout Mean<br>max_nodes =100 |
| EdgePool | All | GCN | 128 | 128 | 0.5 | 5E-04 | 1E-04 | 5E+05 | 50 | edge_drop : 0.2<br>Readout: mean + max |
| MInCutPool | All | GCNConv<br>DenseGraphConv | 128 | 32 | 0.5 | 5E-04 | 1E-04 | 5E+05 | 50 | Readout : mean |
| MemPool | DD<br>PROTEINS<br>ENZYMES<br>COLLAB<br>REDDIT-B<br>Others | GATConv<br>DeepGCNLayer | 60<br>20<br>20<br>64<br>32<br>128 | 120<br>80<br>100<br>100<br>16<br>128 | 0.5 | 5E-04 | 1E-04 | 5E+05 | 50 | BatchNorm1d,<br>heads=5,<br>num_clusters=10,<br>LeakyReLU,<br>2 heads=1 / Readout pooling<br>heads=5 |
| HaarPool | NCI09 | GCN | 100 | 256 | 0.6 | 0.01 | 1E-04 | 150 | 50 | Linear2(nhid,nhid//2)<br>Conv: 3<br>Haarpool:1 |
| | PROTEINS | GCN | 60 | 120 | 0.5 | 1E-03 | 1E-04 | 20 | 20 | Linear2(nhid,nhid//2)<br>Conv: 3<br>Haarpool:3 |
| | MUTAG | GCN | 60 | 60 | 0.5 | 0.01 | 1E-04 | 30 | 15 | Linear1(nhid,3*nhid)<br>Conv: 1<br>Haarpool: 1 |
| | NCI1 | GCN | 100 | 256 | 0.5 | 1E-03 | 1E-04 | 150 | 50 | Linear2(nhid,nhid*4)<br>Linear3(nhid*,nhid*8)<br>Conv: 2<br>Haarpool:1 |
| | DD | GCN | 60 | 120 | 0.5 | 1E-03 | 1E-04 | 20 | 20 | Linear2(nhid,nhid//2)<br>Conv: 3<br>Haarpool:1 |
| | ENZYMES | GCN | 60 | 60 | 0.5 | 0.01 | 1E-04 | 30 | 15 | |
| | PTC-MR | GCN | 60 | 60 | 0.5 | 0.01 | 1E-04 | 30 | 15 | |
| | Mutagenicity | GCN | 100 | 256 | 0.5 | 0.01 | 1E-04 | 50 | 50 | |
| | Others | GCN | 100 | 128 | 0.5 | 0.01 | 1E-04 | 150 | 50 | |
| ASAP | All | GraphConv | 64 | [16,<br>64,<br>128] | 0.5 | 0.01 | 1E-04 | 5E+05 | 50 | Readout: mean+max<br>4 layer |
| HGP-SL | All | GCN | 64 | 128 | 0 | 0.01 | 0.001 | 1000 | 100 | Readout: mean+max<br>Lamb =1.0,<br>sample_neighbor= True<br>sparse_attention = True<br>structure_learning = True |

Pool, MinCutPool, EdgePool, and HaarPool use the hierarchical architecture(Figure 17 **Bottom**), while Set2set, SortPool, GMT use the global architecture(Figure 17 **Top**). The specific of each part in architecture for all models are described in Table 11.

**Implementation Details.** For all experiments on OGB datasets, we evaluate the model performance with a 10-fold cross validation setting, where the dataset split is based on the conventionally used training/validation/test splitsHu et al. (2020). We then report the average performances on the test sets, by performing overall experiments 10 times with different seeds from 42 to 51. We optimize the network with Adam optimizer (Kingma & Ba, 2014). Also, we set the pooling ratio as 25% in each pooling layer for both baselines and our models. For all experiments on OGB datasets except the HIV, batch size is set to 128. Since the HIV dataset contains a large number of graphs compared to others (See Table in the main paper), the batch size is set to 512 for fast training. The detailed hyperparameter settings for each model on each dataset are shown in Table 11.

**Performance Measure.** We use classification accuracy between the predicted labels and ground-truth labels as our performance measure. The model performance is evaluated on the test split of the 10 folds for all OGB datasets, and reported as the average and the standard deviation of the 100 scores with 10 different seeds.