A Proof of Theorem 1

Proof. Let $\texttt{SEL}(\mathcal{G}_1^L) = \left\{ \left\{ s_i^j \right\}_{i=1}^N \right\}_{j=1}^K$ and $\texttt{SEL}(\mathcal{G}_2^L) = \left\{ \left\{ t_i^j \right\}_{i=1}^M \right\}_{j=1}^K$ with $\sum_{j=1}^k s_i^j = 1 \; \forall i = 1, \ldots, N$ and $\sum_{j=1}^K t_i^j = 1 \; \forall i = 1, \ldots, M$.

Suppose:

$$\mathcal{X}_{1_P} = \left\{ \sum_{i=1}^N \boldsymbol{x}_i^L \cdot s_i^j \right\}_{j=1}^K = \left\{ \sum_{i=1}^M \mathbf{y}_i^L \cdot t_i^j \right\}_{j=1}^K = \mathcal{X}_{2_P}.$$

This implies that there exists a permutation $\pi : \{1, \ldots, K\} \to \{1, \ldots, K\}$ such that

$$\sum_{i=1}^N \boldsymbol{x}_i^L \cdot s_i^j = \sum_{i=1}^M \mathbf{y}_i^L \cdot t_i^{\pi(j)} \quad \forall j = 1, \ldots, K$$

which implies

$$\sum_{j=1}^K \sum_{i=1}^N \boldsymbol{x}_i^L \cdot s_i^j = \sum_{j=1}^K \sum_{i=1}^M \mathbf{y}_i^L \cdot t_i^{\pi(j)} \Leftrightarrow \sum_{i=1}^N \boldsymbol{x}_i^L \cdot \sum_{j=1}^K s_i^j = \sum_{i=1}^M \mathbf{y}_i^L \cdot \sum_{j=1}^K t_i^{\pi(j)} \overset{2}{\Leftrightarrow} \sum_{i=1}^N \boldsymbol{x}_i^L = \sum_{i=1}^M \mathbf{y}_i^L$$

which contradicts 1.

$\square$

B Experimental details

B.1 Hyperparameters of the GNN architecture

The GNN architecture used in all experiments consists of: [2 GIN layers] – [1 pooling layer with pooling ratio 0.1] – [1 GIN layer] – [`global_sum_pool`] – [dense readout].

Each GIN layer is configured with an MLP with 2 hidden layers of 64 units and ELU activation functions. The readout is a 3-layer MLP with units [64, 64, 32], ELU activations, and dropout 0.5. The GNN is trained with Adam optimizer with an initial learning rate of 1e-4 using batches with size 32. The pooling ratio is set to 0.5 for EdgePool and Cmp-Graclus. For SAGPool or ASAPool we used only one GIN layer before pooling. For PanPool we used 2 PanConv layers with filter size 2 instead of the first 2 GIN layers. The auxiliary losses in DiffPool, MinCutPool, and DMoN are added to the cross-entropy loss with weights [0.1,0.1], [0.5, 1.0], [0.3, 0.3, 0.3], respectively. For k-MIS we used $k = 5$ and we aggregated the features with the sum. For Graclus, we aggregated the node features with the sum.

B.2 Statistics of the datasets

Table 2 reports the information about the datasets used in the experimental evaluation. Since the COLLAB and REDDIT-BINARY datasets lack vertex features, we assigned a constant feature value of 1 to all vertices.

B.3 Detailed performance on the benchmark datasets

The average test accuracy of the GNNs configured with the different pooling operators on the graph classification benchmarks is reported in Table 3, while Table 4 reports the run-time of each model expressed in seconds per epochs. The average accuracy and average run-time computed across all datasets are presented in Table 5. For each dataset we use the same GNN configured as described in B.1, including the pooling ratio of 0.1 (except for Graclus and EdgePool, where is 0.5), as the goal is to validate the architecture used in the first experiment. Clearly, by using less aggressive pooling, carefully configuring the GNN models, and increasing their capacity it is possible to improve the results on several datasets. We refer the reader to the original papers introducing the different pooling operators for such results.

Table 2: Details of the graph classification datasets.

Dataset	#Samples	#Classes	Avg. #vertices	Avg. #edges	Vertex attr.	Vertex labels
EXPWL1	3,000	2	76.96	186.46	–	yes
NCI1	4,110	2	29.87	64.60	–	yes
Proteins	1,113	2	39.06	72.82	1	yes
COLORS-3	10500	11	61.31	91.03	4	no
Mutagenicity	4,337	2	30.32	61.54	–	yes
COLLAB	5,000	3	74.49	4,914.43	–	no
REDDIT-B	2,000	2	429.63	995.51	–	no
B-hard	1,800	3	148.32	572.32	–	yes

Table 3: Graph classification test accuracy on benchmark datasets.

Pooling	NCI1	PROTEINS	COLORS-3	Mutagenity	COLLAB	REDDIT-B	B-hard
DiffPool	$77.8_{\pm 3.9}$	$72.8_{\pm 3.3}$	$87.6_{\pm 1.0}$	$80.0_{\pm 1.9}$	$76.6_{\pm 2.5}$	$89.9_{\pm 2.8}$	$70.2_{\pm 1.5}$
DMoN	$78.5_{\pm 1.4}$	$73.1_{\pm 4.6}$	$88.4_{\pm 1.4}$	$81.3_{\pm 0.3}$	$80.9_{\pm 0.7}$	$91.3_{\pm 1.4}$	$71.1_{\pm 1.0}$
MinCut	$80.1_{\pm 2.6}$	$76.0_{\pm 3.6}$	$88.7_{\pm 1.6}$	$81.2_{\pm 1.9}$	$79.2_{\pm 1.5}$	$91.9_{\pm 1.8}$	$71.2_{\pm 1.1}$
ECPool	$79.8_{\pm 3.3}$	$69.5_{\pm 5.9}$	$81.4_{\pm 3.3}$	$82.0_{\pm 1.6}$	$80.9_{\pm 1.4}$	$90.7_{\pm 1.7}$	$74.5_{\pm 1.6}$
Graclus	$81.2_{\pm 3.4}$	$73.0_{\pm 5.9}$	$77.6_{\pm 1.2}$	$81.9_{\pm 1.6}$	$80.4_{\pm 1.5}$	$92.9_{\pm 1.7}$	$72.3_{\pm 1.3}$
k-MIS	$77.6_{\pm 3.0}$	$75.9_{\pm 2.9}$	$82.9_{\pm 1.7}$	$82.6_{\pm 1.2}$	$73.7_{\pm 1.4}$	$90.6_{\pm 1.4}$	$71.7_{\pm 0.9}$
Top-k	$72.6_{\pm 3.1}$	$73.2_{\pm 2.7}$	$57.4_{\pm 2.5}$	$74.4_{\pm 4.7}$	$77.9_{\pm 2.1}$	$87.4_{\pm 3.5}$	$68.1_{\pm 7.7}$
PanPool	$66.1_{\pm 2.3}$	$75.2_{\pm 6.2}$	$40.7_{\pm 11.5}$	$67.2_{\pm 2.0}$	$78.2_{\pm 1.5}$	$83.6_{\pm 1.9}$	$44.2_{\pm 8.5}$
ASAPool	$73.1_{\pm 2.5}$	$75.5_{\pm 3.2}$	$43.0_{\pm 4.7}$	$76.5_{\pm 2.8}$	$78.4_{\pm 1.6}$	$88.0_{\pm 5.6}$	$67.5_{\pm 6.1}$
SAGPool	$79.1_{\pm 3.0}$	$75.2_{\pm 2.7}$	$43.1_{\pm 11.1}$	$77.9_{\pm 2.8}$	$78.1_{\pm 1.8}$	$84.5_{\pm 4.4}$	$54.0_{\pm 6.6}$

Table 4: Graph classification test run-time in s/epoch.

Pooling	NCI1	PROTEINS	COLORS-3	Mutagenity	COLLAB	REDDIT-B	B-hard
DiffPool	0.83s	0.23s	1.67s	0.90s	1.68s	1.74s	0.29s
DMoN	1.01s	0.28s	1.94s	1.06s	1.83s	1.04s	0.33s
MinCut	0.95s	0.28s	1.82s	1.10s	1.82s	1.78s	0.35s
ECPool	4.39s	1.97s	10.30s	4.22s	44.11s	3.17s	6.90s
Graclus	0.95s	0.27s	2.47s	0.98s	3.01s	0.75s	0.31s
k-MISPool	0.88s	0.25s	2.48s	0.95s	1.38s	0.48s	0.43s
Top-k	1.04s	0.29s	2.78s	1.04s	2.79s	0.47s	0.30s
PanPool	2.81s	0.81s	7.16s	5.48s	7.67s	46.15s	6.27s
ASAPool	1.83s	0.52s	4.48s	1.80s	3.97s	0.79s	0.52s
SAGPool	1.09s	0.30s	2.52s	1.07s	2.81s	0.43s	0.28s

Table 5: Average run-time in seconds per epoch (first row) and average classification accuracy (second row) achieved by the different pooling methods on the benchmark datasets.

DiffPool	DMoN	MinCut	ECPool	Graclus	k-MIS	Top-k	PanPool	ASAPool	SAGPool
1.04s	1.07s	1.15s	10.72s	1.24s	0.97s	1.24s	10.90s	1.98s	1.21s
$79.2_{\pm 2.4}$	$80.6_{\pm 1.5}$	$81.1_{\pm 2.0}$	$79.8_{\pm 2.6}$	$79.9_{\pm 2.3}$	$79.2_{\pm 2.1}$	$73.0_{\pm 3.7}$	$65.0_{\pm 4.8}$	$71.7_{\pm 3.7}$	$70.2_{\pm 4.6}$