TOPOLOGY-AWARE POOLING VIA GRAPH ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Pooling operations have shown to be effective on various tasks in computer vision and natural language processing. One challenge of performing pooling operations on graph data is the lack of locality that is not well-defined on graphs. Previous studies used global ranking methods to sample some of the important nodes, but most of them are not able to incorporate graph topology information in computing ranking scores. In this work, we propose the topology-aware pooling (TAP) layer that uses attention operators to generate ranking scores for each node by attending each node to its neighboring nodes. The ranking scores are generated locally while the selection is performed globally, which enables the pooling operation to consider topology information. To encourage better graph connectivity in the sampled graph, we propose to add a graph connectivity term to the computation of ranking scores in the TAP layer. Based on our TAP layer, we develop a network on graph data, known as the topology-aware pooling network. Experimental results on graph classification tasks demonstrate that our methods achieve consistently better performance than previous models.

1 INTRODUCTION

Pooling operations have been widely applied in various fields such as computer vision (He et al., 2016; Huang et al., 2017), and natural language processing (Zhang et al., 2015). Pooling operations can effectively reduce dimensional sizes (Simonyan & Zisserman, 2015) and enlarge receptive fields (Chen et al., 2016). The application of regular pooling operations depends on the well-defined spatial locality in grid-like data such as images and texts. However, it is still challenging to perform pooling operations on graph data. In particular, there is no spatial locality or order information among nodes (Gao et al., 2018). Some works try to overcome this limitation with two kinds of methods; those are node clustering (Ying et al., 2018) and primary nodes sampling (Gao & Ji, 2019; Zhang et al., 2018). The node clustering methods create graphs with super-nodes by learning a nodes assignment matrix. These methods suffer from the over-fitting problem and need auxiliary link prediction tasks to stabilize the training (Ying et al., 2018). The primary nodes sampling methods like top-k pooling (Gao & Ji, 2019) rank the nodes in a graph and sample top-k nodes to form the sampled graph. It uses a small number of additional trainable parameters and is shown to be more powerful (Gao & Ji, 2019). However, the top-k pooling layer does not explicitly incorporate the topology information in a graph when computing ranking scores, which may cause performance loss.

In this work, we propose a novel topology-aware pooling (TAP) layer that explicitly encodes the topology information when computing ranking scores. We use an attention operator to compute similarity scores between each node and its neighboring nodes. The average similarity score of a node is used as its ranking score in the selection process. To avoid isolated nodes problem in our TAP layer, we further propose a graph connectivity term for computing the ranking scores of nodes. The graph connectivity term uses degree information as a bias term to encourage the layer to select highly connected nodes to form the sampled graph. Based on the TAP layer, we develop topology-aware pooling networks for network embedding learning. Experimental results on graph classification tasks demonstrate that our proposed networks with TAP layer consistently outperform previous models. The comparison results between our TAP layer and other pooling layers based on the same network architecture demonstrate the effectiveness of our method compared to other pooling methods.

2 BACKGROUND AND RELATED WORK

In this section, we describe graph pooling operations and attention operators.

2.1 GRAPH POOLING OPERATIONS

The pooling operations on graph data mainly include two categories; those are node clustering and node sampling. DIFFPOOL (Ying et al., 2018) realizes graph pooling operation by clustering nodes into super-nodes. By learning an assignment matrix, DIFFPOOL softly assigns each node to different clusters in the new graph with specified probabilities. The pooling operations under this category retain and encode all nodes information into the new graph. One challenge of methods in this category is that they may increase the risk of over-fitting by training another network to learn the assignment matrix. In addition, the new graph is mostly connected where each edge value represents the strength of connectivity between two nodes. The connectivity pattern in the new graph may greatly differ from that of the original graph. The node sampling methods mainly select a fixed number k of the most important nodes to form a new graph. In SortPool (Zhang et al., 2018), the same feature of each node is used for ranking and k nodes with the largest values in this feature are selected to form the coarsened graph. Top-k pooling (Gao & Ji, 2019) generates the ranking scores by using a trainable projection vector that projects feature vectors of nodes into scalar values. k nodes with the largest scalar values are selected to form the coarsened graph. These methods involve none or a very small number of extra trainable parameters, thereby avoiding the risk of over-fitting. However, these methods suffer from one limitation that they do not explicitly consider the topology information during pooling. Both SortPool and top-k pooling select nodes based on scalar values that do not explicitly incorporate topology information. In this work, we propose a pooling operation that explicitly encodes topology information in ranking scores, thereby leading to an improved operation.

2.2 ATTENTION OPERATORS

Attention operator has shown to be effective in challenging tasks in various fields such as computer vision (Xu et al., 2015b; Lu et al., 2016; Li et al., 2018) and natural language processing (Malinowski et al., 2018; Bahdanau et al., 2015; Vaswani et al., 2017). Attention operator is capable of capturing long-range relationships, thereby leading to better performances (Wang et al., 2018). The inputs to an attention operator consist of three matrices; those are a query matrix $\boldsymbol{Q} \in \mathbb{R}^{d \times m}$, a key matrix $\boldsymbol{K} \in \mathbb{R}^{d \times n}$, and a value matrix $\boldsymbol{V} \in \mathbb{R}^{p \times n}$. The attention operator computes the response of each query vector in $\boldsymbol{Q}$ by attending it to all key vectors in $\boldsymbol{K}$. It uses the resulting coefficient vector to take a weighted sum over value vectors in $\boldsymbol{V}$. The layer-wise operation of an attention operation is defined as $\boldsymbol{O} = \boldsymbol{V}\text{softmax}(\boldsymbol{K}^T\boldsymbol{Q})$. When the attention operator is applied to graph, each node only attends to its neighboring nodes (Veličković et al., 2017). Self-attention can also produce an attention mask to control information flow on selected nodes in pooling operation (Lee et al., 2019). In our proposed pooling operation, we employ an attention operator to compute ranking scores that explicitly encode topology information.

3 TOPOLOGY-AWARE POOLING LAYERS AND NETWORKS

In this work, we propose the topology-aware pooling (TAP) layer that uses attention operators to encode topology information in ranking scores for node selection. We also propose a graph connectivity term in the computation of ranking scores, which encourages better graph connectivity in the coarsened graph. Based on our TAP layer, we propose the topology-aware pooling networks for network representation learning.

3.1 TOPOLOGY-AWARE POOLING LAYER

Pooling layers have shown to be important on grid-like data with regard to reducing feature map sizes and enlarging receptive fields (Yu & Koltun, 2016; Carreira et al., 2012). On graph data, two kinds of pooling layers have been proposed; those are node clustering (Ying et al., 2018) and primary nodes sampling (Gao & Ji, 2019; Zhang et al., 2018). A primary nodes sampling method, known as top-k pooling (Gao & Ji, 2019), uses a projection vector to generate ranking scores for each node in the

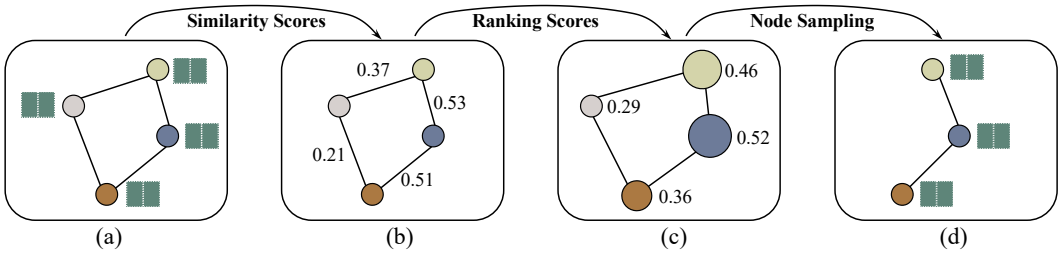


Figure 1: An illustration of the proposed topology-aware pooling layer that selects $k = 3$ nodes. This graph contains four nodes, each of which has 2 features. Given the input graph, we firstly use an attention operator to compute similarity scores between every pair of connected nodes. Here, we use self-attention without linear transformation for notation simplicity. In graph (b), we label each edge by the similarity score between its two ends. Then we compute the ranking score of each node by taking the average of the similarity scores between it and its neighboring nodes. In graph (c), we label each node by its ranking score and bigger node indicates a higher ranking score. By selecting the nodes with the $k = 3$ largest ranking scores, the selected graph is shown in graph (d).

graph. The graph is created by choosing nodes with k-largest scores. However, the sampling process relies on the projection vector and does not explicitly consider the topology information in the graph.

In this section, we propose the topology-aware pooling (TAP) layer that performs primary nodes sampling by considering the graph topology. In this layer, we generate the ranking scores based on local information. To this end, we employ an attention operator to compute the similarity scores between each node and its neighboring nodes. The ranking score for a node i is the mean value of the similarity scores with its neighboring nodes. The resulting ranking score for a node indicates the similarity between this node and its neighboring nodes. If a node has a high ranking score, it can highly represent a local graph that consists of it and its neighboring nodes. By choosing nodes with the highest ranking scores, we can retain the maximum information in the sampled graph.

Suppose there are N nodes in a graph $\mathbb{G}$, each of which contains C features. In layer ℓ, we use two matrices to represent the graph; those are the adjacency matrix $\boldsymbol{A}^{(\ell)} \in \mathbb{R}^{N\times N}$ and the feature matrix $\boldsymbol{X}^{(\ell)} \in \mathbb{R}^{N\times C}$. The non-zero entries in $\boldsymbol{A}^{(\ell)}$ represent edges in the graph. The ith row in $\boldsymbol{X}^{(\ell)}$ denotes the feature vector of node i. The layer-wise forward propagation rule of TAP in layer ℓ is defined as

$$\boldsymbol{K} = \boldsymbol{X}^{(\ell)}\boldsymbol{W}^{(\ell)}, \qquad \in\mathbb{R}^{N\times C} \tag{1}$$

$$\boldsymbol{E} = \boldsymbol{X}^{(\ell)}\boldsymbol{K}^{T}, \qquad \in\mathbb{R}^{N\times N} \tag{2}$$

$$\tilde{\boldsymbol{E}} = \boldsymbol{E} \circ \boldsymbol{A}^{(\ell)}, \qquad \in\mathbb{R}^{N\times N} \tag{3}$$

$$\boldsymbol{d} = \sum_{j=1}^{N} \boldsymbol{A}^{(\ell)}_{:j}, \qquad \in\mathbb{R}^{N} \tag{4}$$

$$\boldsymbol{s} = \text{sigmoid}\left(\frac{\sum_{j=1}^{N}\tilde{\boldsymbol{E}}_{:j}}{\boldsymbol{d}}\right), \qquad \in\mathbb{R}^{N} \tag{5}$$

$$\boldsymbol{idx} = \text{Ranking}_k(\boldsymbol{s}), \qquad \in\mathbb{R}^{k} \tag{6}$$

$$\boldsymbol{A}^{(\ell+1)} = \boldsymbol{A}^{(\ell)}(\boldsymbol{idx}, \boldsymbol{idx}), \qquad \in\mathbb{R}^{k\times k} \tag{7}$$

$$\boldsymbol{X}^{(\ell+1)} = \boldsymbol{X}^{(\ell)}(\boldsymbol{idx}, :)\text{diag}(\boldsymbol{s}(\boldsymbol{idx})), \qquad \in\mathbb{R}^{k\times C} \tag{8}$$

where $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{C\times C}$ is a trainable weight matrix, $\boldsymbol{A}^{(\ell)}_{:j}$ is the jth column of matrix $\boldsymbol{A}^{(\ell)}$, $\circ$ denotes the element-wise matrix multiplication, $\tilde{\boldsymbol{E}}_{:j}$ is the jth column of matrix $\tilde{\boldsymbol{E}}$, k is the number of nodes selected in the sampled graph, and $\text{diag}(\cdot)$ constructs a diagonal matrix using input vector as diagonal elements. Ranking_k operator ranks the scores and return the indices of k-largest values in $\boldsymbol{s}$.

To compute attention scores, we perform a linear transformation on feature matrix $\boldsymbol{X}^{(\ell)}$ in Eq. (1), which results in the key matrix $\boldsymbol{K}$. We use the input feature matrix as the query matrix. The similarity

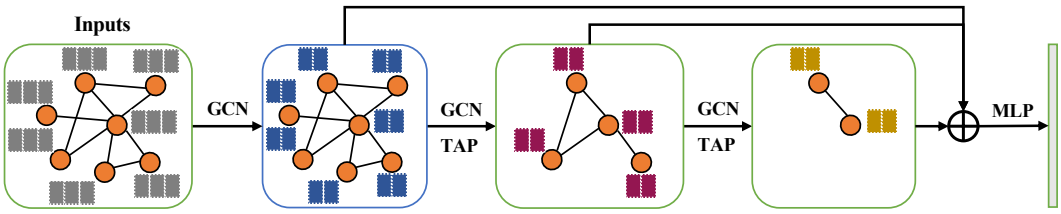


Figure 2: An illustration of the topology-aware pooling network. $\oplus$ denotes the concatenation operation of feature vectors. Each node in the input graph contains three features. We use a GCN layer to transform the feature vectors into low-dimensional representations. We stack two blocks, each of which consists of a GCN layer and a TAP layer. A global reduction operation such as max-pooling is applied to the outputs of the first GCN layer and TAP layers. The resulting feature vectors are concatenated and fed into the final multi-layer perceptron for prediction.

score matrix $\boldsymbol{E}$ is obtained by the matrix multiplication between $\boldsymbol{X}^{(\ell)}$ and $\boldsymbol{K}$ in Eq. (2). Each value e_{ij} in $\boldsymbol{E}$ measures the similarity between node i and node j. Since $\boldsymbol{E}$ contains similarity scores for nodes that are not directly connected, we use the adjacency matrix $\boldsymbol{A}^{(\ell)}$ as a mask to set these entries in $\boldsymbol{E}$ to zeros in Eq. (3), resulting in $\tilde{\boldsymbol{E}}$. We compute the degree of each node in Eq. (4). The ranking score of a node is computed in Eq. (5) by taking the average of similarity scores between this node and its neighboring nodes followed by a sigmoid operation. Here, we perform element-wise division between two vectors. The resulting score vector is $\boldsymbol{s} = [s_1, s_2, \ldots, s_N]^T$ where s_i represents the ranking score of node i. Ranking_k is an operator that selects k-largest values and returns the corresponding indices. In Eq. (6), we use Ranking_k to select the k-most important nodes with indices in $\boldsymbol{idx}$. Using $\boldsymbol{idx}$, we extract new adjacency matrix $\boldsymbol{A}^{(\ell+1)}$ in Eq. (7) and new feature matrix $\boldsymbol{X}^{(\ell+1)}$ in Eq. (8). Here, we use the ranking scores $\boldsymbol{s}(\boldsymbol{idx})$ as gates to control information flow and enable the gradient back-propagation for trainable transformation matrix $\boldsymbol{W}^{(\ell)}$ (Gao & Ji, 2019).

This method can be considered as a local-voting, global-ranking process. In our TAP layer, the ranking scores are derived from the similarity scores of each node with its neighboring nodes, thereby encoding the topology information of each node in its ranking score. This can be considered as a local voting process that each node gets its votes from the local neighborhood. When performing global ranking, the nodes that get the highest votes from local neighborhoods are selected such that maximum information in the graph can be retained. Figure 1 provides an illustration of our proposed TAP layer. Compared to the top-k pooling (Gao & Ji, 2019), our TAP layer considers topology information in the graph, thereby leading to a better coarsened graph.

3.2 Graph Connectivity Term

Our proposed TAP layer computes the ranking scores by using similarity scores between nodes in the graph, thereby regarding topology information in the graph. However, the coarsened graph generated by the TAP layer may suffer from the problem of isolated nodes. In sparsely connected graphs, some nodes have a very small number of neighboring nodes or even only themselves. Suppose node i only connects to itself. The ranking score of node i is the similarity score to itself, which may result in high ranking scores in the graph. The resulting graph can be very sparsely connected.

To overcome the limitation of TAP layer and encourage better connectivity in the selected graph, we propose to add a graph connectivity term to the computation of ranking scores. To this end, we use node degrees as an indicator for graph connectivity and add degree values to their ranking scores such that densely-connected nodes are preferred during nodes selection. By using the node degree as the graph connectivity term, the ranking score of node i is computed as

$$s_i = \text{sigmoid}\left(\frac{\sum_{j=1}^{N} \tilde{E}_{ij}}{d_i}\right) + \lambda \frac{d_i}{N}, \tag{9}$$

where d_i is the degree of node i, and λ is a hyperparameter that sets the importance of the graph connectivity term to the computation of ranking scores. The graph connectivity term can overcome the limitation of the TAP layer. The computation of ranking scores now considers nodes degrees and gives rise to better connectivity in the resulting graph.

3.3 TOPOLOGY-AWARE POOLING NETWORKS

Based on our proposed TAP layer, we build a family of networks known as topology-aware pooling networks (TAPNets) for graph classification tasks. In TAPNets, we firstly apply a graph embedding layer to produce low-dimensional representations of nodes in the graph, which helps to deal with some datasets with very high-dimensional input feature vectors. Here, we use a GCN layer (Kipf & Welling, 2017) for node embedding. After the embedding layer, we stack several blocks, each of which consists of a GCN layer for high-level feature extraction and a TAP layer for graph coarsening. In the ith TAP layer, we use a hyperparameter $k^{(i)}$ to control the number nodes in the sampled graph. We feed the output feature matrices of the graph embedding layer and TAP layers to a classifier. Suppose we stack m blocks and all GCN and TAP layers output h feature maps. Given an input graph with the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ and the feature matrix $\boldsymbol{X} \in \mathbb{R}^{N \times C}$, our TAPNet outputs a list of feature matrices $[\boldsymbol{Y}_0, \boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_m]$. Here, $\boldsymbol{Y}_0 \in \mathbb{R}^{N \times h}$ is the output of the graph embedding layer and $\boldsymbol{Y}_i \in \mathbb{R}^{k^{(i)} \times h}$ is the output of TAP layer in the ith block.

In TAPNets, we use a multi-layer perceptron as the classifier. We first transform network outputs to a one-dimensional vector. Specificity, the resulting vector $\boldsymbol{z} = [\boldsymbol{y}_0^T, \boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_m^T]^T$ where $\boldsymbol{y}_i$ is transformed from $\boldsymbol{Y_i}$. Global max and average pooling operations are two popular ways for the transformation, which reduce the spatial size of feature matrices to 1 using max and average functions, respectively. Recently, Xu et al. (2018) proposed to use the summation function that results in promising performances. In TAPNets, we concatenate transformation output vectors produced by the global pooling operations using max, averaging, and summation, respectively. The resulting feature vector is fed into the classifier. Figure 2 illustrates a sample TAPNet with two blocks.

3.4 AUXILIARY LINK PREDICTION OBJECTIVE

Multi-task learning has shown to be effective across various machine learning tasks (Ruder, 2017). It can leverage useful information in multiple related tasks, thereby leading to better generalization and performance. In this section, we propose to add an auxiliary link prediction objective during training by using a by-product of our TAP layer. In Eq. (2), we compute the similarity scores $\boldsymbol{E}$ between every pair of nodes in the graph. By applying an element-wise sigmoid$(\cdot)$ on $\boldsymbol{E}$, we can obtain a link probability matrix $\hat{\boldsymbol{E}}^{(\ell)} \in \mathbb{R}^{N \times N}$ with each element $\hat{e}_{ij}$ measures the likelihood of a link between node i and node j. With the adjacency matrix $\boldsymbol{A}^{(\ell)}$, we compute the auxiliary link prediction loss as

$$loss_{aux} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} f\left(\hat{\boldsymbol{E}}_{ij}^{(\ell)}, \boldsymbol{A}_{ij}^{(\ell)}\right), \tag{10}$$

where $f(\cdot, \cdot)$ is a loss function. Note that the adjacency matrix used as the link prediction objective is directly derived from the original graph. Since the TAP layer extracts a sub graph from the original one, the connectivity between two nodes in the sampled graph is the same as that in the original graph. Compared to auxiliary link prediction in DiffPool (Ying et al., 2018) that uses the learned adjacency matrix as objective, our method uses the original links, thereby providing more accurate information.

4 EXPERIMENTAL STUDIES

In this section, we evaluate our methods and networks on graph classification tasks using bioinformatics and social network datasets. We conduct ablation experiments to evaluate the contributions of the TAP layer and each term in it to the overall network performance.

4.1 GRAPH CLASSIFICATION RESULTS ON SOCIAL NETWORK DATASETS

We conduct experiments on graph classification tasks to evaluate our proposed methods and TAPNets. We use 6 social network datasets; those are COLLAB, IMDB-BINARY (IMDB-B), IMDB-MULTI (IMDB-M), REDDIT-BINARY (RDT-B), REDDIT-MULTI5K (RDT-M5K) and REDDIT-MULTI12K (RDT-M12K) (Yanardag & Vishwanathan, 2015) datasets. Note that REDDIT datasets are popular large datasets used for network embedding learning in terms of graph size and number of graphs (Ying et al., 2018; Xu et al., 2018). Since there is no feature for nodes in social networks,

Table 1: Comparisons between TAPNets and other models such as WL (Shervashidze et al., 2011) and PSCN (Niepert et al., 2016) in terms of graph classification accuracy (%) on social network datasets including COLLAB, IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, REDDIT-MULTI5K and REDDIT-MULTI12K datasets.

	COLLAB	**IMDB-B**	**IMDB-M**	**RDT-B**	**RDT-M5K**	**RDT-M12K**
# graphs	5000	1000	1500	2000	4999	11929
# nodes	74.5	19.8	13.0	429.6	508.5	391.4
# classes	3	2	3	2	5	11
WL	78.9 ± 1.9	73.8 ± 3.9	50.9 ± 3.8	81.0 ± 3.1	52.5 ± 2.1	44.4 ± 2.1
PSCN	72.6 ± 2.2	71.0 ± 2.2	45.2 ± 2.8	86.3 ± 1.6	49.1 ± 0.7	41.3 ± 0.8
DGCNN	73.8	70.0	47.8	-	-	41.8
DIFFPOOL	75.5	-	-	-	-	47.1
g-U-Net	77.5 ± 2.1	75.4 ± 3.0	51.8 ± 3.7	85.5 ± 1.3	48.2 ± 0.8	44.5 ± 0.6
GIN	80.6 ± 1.9	75.1 ± 5.1	52.3 ± 2.8	92.4 ± 2.5	$\mathbf{57.5 \pm 1.5}$	-
TAPNet (ours)	$\mathbf{84.6 \pm 1.7}$	$\mathbf{79.5 \pm 4.1}$	$\mathbf{55.6 \pm 2.9}$	$\mathbf{94.1 \pm 1.9}$	57.1 ± 1.3	$\mathbf{49.2 \pm 1.6}$

Table 2: Comparisons between TAPNets and other models in terms of graph classification accuracy (%) on bioinformatics datasets including DD, PTC, MUTAG, and PROTEINS datasets.

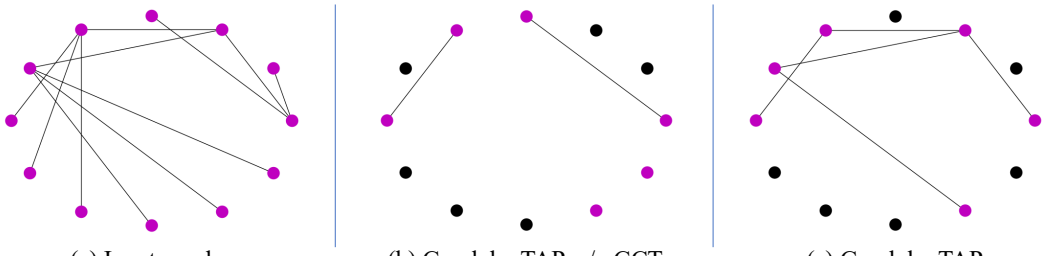
	DD	**PTC**	**MUTAG**	**PROTEINS**
# graphs	1178	344	188	1113
# nodes	284.3	25.5	17.9	39.1
# classes	2	2	2	2
WL	78.3 ± 0.6	59.9 ± 4.3	90.4 ± 5.7	75.0 ± 3.1
PSCN	76.3 ± 2.6	60.0 ± 4.8	92.6 ± 4.2	75.9 ± 2.8
DGCNN	79.4 ± 0.9	58.6 ± 2.4	85.8 ± 1.7	75.5 ± 0.9
SAGPool	76.5	-	-	71.9
DIFFPOOL	80.6	-	-	76.3
g-U-Net	82.4 ± 2.9	64.7 ± 6.8	87.2 ± 7.8	77.6 ± 2.6
GIN	82.0 ± 2.7	64.6 ± 7.0	90.0 ± 8.8	76.2 ± 2.8
TAPNet (ours)	$\mathbf{84.2 \pm 3.7}$	$\mathbf{72.7 \pm 6.0}$	$\mathbf{93.0 \pm 5.8}$	$\mathbf{78.9 \pm 4.2}$

we create node features by following the practices in (Xu et al., 2018). In particular, we use one-hot encodings of node degrees as feature vectors for nodes in social network datasets. To ensure fair comparisons, we do not use the auxiliary link prediction objective in these experiments. We provide more details about the experimental setups in the appendix.

We compare our TAPNets with other state-of-the-art models in terms of graph classification accuracy. The comparison results are summarized in Table 1. We can observe from the results that our TAPNets significantly outperform other models on most social network datasets by margins of 4.0%, 4.4%, 3.3%, 1.7%, and 2.1% on COLLAB, IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, and REDDIT-MULTI12K datasets, respectively. The promising performances, especially on large datasets such as REDDIT, demonstrate the effectiveness of our methods. Note that the superior performances over g-U-Net (Gao & Ji, 2019) show that our TAP layer can produce better-coarsened graph than that using the top-k pooling layer.

4.2 Graph Classification Results on Bioinformatics Datasets

We have shown the promising performances of our TAPNets on social network datasets. To fully evaluate our methods, we conduct experiments on graph classification tasks using 4 bioinformatics datasets; those are DD , PTC , MUTAG , and PROTEINS (Xu et al., 2018) datasets. Notably, nodes in bioinformatics datasets have categorical features. We compare our TAPNets with other state-of-the-art models in terms of graph classification accuracy without using auxiliary link prediction. The comparison results are summarized in Table 2. We can observe from the results that our TAPNets achieve significantly better results than other models by margins of 1.2%, 8.1%, 3.0%, and 2.7% on DD, PTC, MUTAG, and PROTEINS datasets, respectively. Notably, some bioinformatics datasets such as PTC and MUTAG are much smaller than social network datasets in terms of number of graphs and number of nodes in graphs. The promising results on these small datasets demonstrate that our



(a) Input graph (b) Graph by TAP w/o GCT (c) Graph by TAP

Figure 3: Illustrations of coarsened graphs generated by TAP and TAP w/o GCT. Here, GCT denotes the graph connection term. The input graph (a) contains 12 nodes. The pooling layers select 6 nodes to form new graphs. The nodes that are not selected are colored black. The new graph in (b) generated by TAP w/o GCT is sparsely connected. The new graph generated by TAP is illustrated in (c), which is shown to be much better connected.

methods can achieve good generalization and performances without involving the risk of over-fitting. SAGPool (Lee et al., 2019) uses a GCN layer to compute ranking scores and a self-attention operator to generate a mask to control the information flow. Here, our method employs an attention operator to compute ranking scores that better encodes the topology information in the graph. The superior performances over SAGPool on DD and PROTEINS datasets demonstrate that our methods can better capture the topology information, thereby leading to better performances.

4.3 Comparison with Other Graph Pooling Layers

It may be argued that our TAPNets achieve promising results by employing superior networks. In this section, we conduct experiments on the same TAPNet architecture to compare our TAP layer with other graph pooling layers; those are DIFFPOOL, sort pooling, and top-k pooling layers. We denote the networks with the TAPNet architecture while using these pooling layers as Net_{diff}, Net_{sort}, and $\text{Net}_{\text{top-}k}$, respectively. We evaluate them on PTC, IMDB-MULTI, and REDDIT-BINARY datasets and summarize the results in Table 3. Note that these models use the same experimental setups to ensure fair comparisons. The results demonstrate the superior performance of our proposed TAP layer compared with other pooling layers using the same network architecture.

Table 3: Comparisons between different pooling operations based on the same TAPNet architecture.

Model	PTC	IMDB-M	RDT-B
Net_{diff}	70.9	54.9	92.1
Net_{sort}	70.6	54.8	92.3
$\text{Net}_{\text{top-}k}$	71.5	55.2	92.8
TAPNet	**72.7**	**55.6**	**94.1**

4.4 Ablation Studies

In this section, we investigate the contributions of TAP layer and its components in ranking score computation; those are the attention score term (AST) and the graph connectivity term (GCT). We remove TAP layers from TAPNet which we denote as TAPNet w/o TAP. To explore the contributions of terms in ranking scores computation, we separately remove ASTs and GCTs from all TAP layers in TAPNets. We denote the resulting models as TAPNet w/o AST and TAPNet w/o GCT, respectively. In addition, we add the auxiliary link prediction objective as described in Section 3.4 in training. We denote the TAPNet using auxiliary training objective as TAPNet w AUX. We evaluate these models on three datasets; those are PTC, IMDB-MULTI, and REDDIT-BINARY datasets. The comparison results are summarized in Table 4. The results show that TAPNets outperform TAPNets w/o TAP by margins of 2.1%, 3.5%, and 2.4% on PTC, IMDB-MULTI, and REDDIT-BINARY datasets, respectively. The better results of TAPNet over

Table 4: Comparisons among TAPNets with and without TAP layers, TAPNet without attention score term (AST), TAPNet without graph connection term (GCT), and TAPNet with auxiliary link prediction objective (AUX) in terms of graph classification accuracy (%) on PTC, IMDB-MULTI, and REDDIT-BINARY datasets.

Model	PTC	IMDB-M	RDT-B
TAPNet w/o TAP	70.6	52.1	91.0
TAPNet w/o AST	71.2	54.8	91.5
TAPNet w/o GCT	72.0	55.1	93.0
TAPNet	72.7	55.6	94.1
TAPNet w AUX	**73.0**	**55.8**	**94.2**

TAPNet w/o AST and TAPNet w/o GCT show the contributions of ASTs and GCTs to performances. It can be observed that TAPNet w AUX achieves better performances than TAPNet, which shows the effectiveness of the auxiliary link prediction objective. To fully study the impact of GCT on TAP layer, we visualize the coarsened graphs generated by TAP and TAP without GCT (denoted as TAP w/o GCT). We select a graph from PTC dataset and illustrate outcome graphs in Figure 3. We can observe from the figure that TAP produces better-connected graph than that by TAP w/o GCT.

4.5 Parameter Study of GAP

Since TAP layer employs an attention operator to compute ranking scores, which involves extra trainable parameters, we conduct experiments to study the number of parameters in TAPNet. We remove the extra trainable parameters from TAP layers in two ways; those are removing TAP layers from TAPNet and removing attention score terms (AST) from TAP layers. We denote the resulting two networks as TAPNet w/o TAP and TAPNet w/o AST, respectively. The comparison results on REDDIT-BINARY dataset is summarized in Table 5. We can see from the results that TAP layers only need 2.13% additional trainable parameters. We believe the negligible usage of extra parameters will not increase the risk of over-fitting but can bring 1.9% performance improvement over TAPNet w/o TAP and TAPNet w/o AST on REDDIT-BINARY dataset. Also, the promising performances of TAPNets on small datasets like PTC and MUTAG in Table 2 show that TAP layers will not cause the over-fitting problem.

Table 5: Comparisons among TAPNets with and without TAP layers, and TAPNet without attention score term (AST) on REDDIT-BINARY dataset.

Model	Accuracy	#Params	Ratio
TAPNet w/o TAP	91.0	323,666	0.00%
TAPNet w/o AST	91.5	323,666	0.00%
TAPNet	**94.1**	330,578	2.13%

4.6 Performance Study of λ

In Section 3.2, we propose to add the graph connectivity term into the computation of ranking scores to improve the graph connectivity in the coarsened graph. It can be seen that λ is an influential hyperparameter in the TAP layer. In this part, we study the performances of different λ values. We select different λ values from the range of 0.01, 0.1, 1.0, 10.0, and 100.0 to cover a reasonable range of values. We evaluate TAPNets using different λ values on PTC, IMDB-MULTI, and REDDIT-BINARY datasets. The results are shown in Figure 4. We can observe that the best performances are achieved with $\lambda = 0.1$. When λ becomes larger, the performance of TAPNet models decrease. This indicates that the graph connectivity term is a plus for generating reasonable ranking scores but it should not overwhelm the attention score term that encodes the topology information in ranking scores.

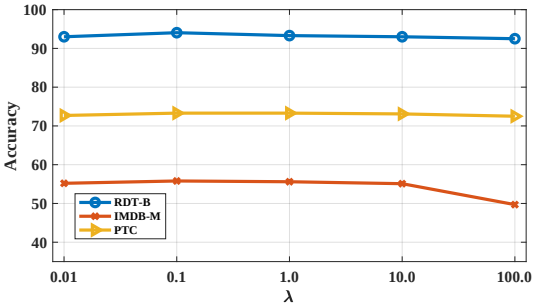


Figure 4: Comparison results of TAPNets using different λ values in TAP layers.

5 Conclusions

In this work, we propose a novel topology-aware pooling (TAP) layer that applies attention mechanism to explicitly encode the topology information in ranking scores. A TAP layer attends each node to its neighboring nodes and uses the average similarity score with its neighboring nodes as its ranking score. The primary nodes sampling based on these ranking scores can incorporate the topology information, thereby leading to a better-coarsened graph. Moreover, we propose to add a graph connectivity term to the computation of ranking scores to overcome the isolated problem which a TAP layer may suffer from. Based on the TAP layer, we develop topology-aware pooling networks (TAPNets) for network representation learning. We add an auxiliary link prediction objective to train our networks by employing the similarity score matrix generated in TAP layers. Experimental results on graph classification tasks using both bioinformatics and social network datasets demonstrate that our TAPNets achieve performance improvements as compared to previous models. Ablation Studies show the contributions of our TAP layers and terms in ranking score computation to network performances.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*, 2015.

Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *European Conference on Computer Vision*, pp. 430–443. Springer, 2012.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *Transactions on Pattern Analysis and Machine Intelligence*, 2016.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Hongyang Gao and Shuiwang Ji. Graph U-nets. In *Proceedings of The 36th International Conference on Machine Learning*, 2019.

Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1416–1424. ACM, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *The International Conference on Learning Representations*, 2015.

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

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *Proceedings of The 36th International Conference on Machine Learning*, 2019.

Guanbin Li, Xiang He, Wei Zhang, Huiyou Chang, Le Dong, and Liang Lin. Non-locally enhanced encoder-decoder network for single image de-raining. In *2018 ACM Multimedia Conference on Multimedia Conference*, pp. 1056–1064. ACM, 2018.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pp. 289–297, 2016.

Mateusz Malinowski, Carl Doersch, Adam Santoro, and Peter Battaglia. Learning visual question answering by bootstrapping hard attention. In *European Conference on Computer Vision*, pp. 3–20. Springer, 2018.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pp. 2014–2023, 2016.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep): 2539–2561, 2011.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Proceedings of the International Conference on Learning Representations*, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 6000–6010, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pp. 4, 2018.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015a.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pp. 2048–2057, 2015b.

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

Pinar Yanardag and SVN Vishwanathan. A structural smoothing framework for robust graph comparison. In *Advances in Neural Information Processing Systems*, pp. 2134–2142, 2015.

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*, pp. 4800–4810, 2018.

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *Proceedings of the International Conference on Learning Representations*, 2016.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pp. 649–657, 2015.

Appendix

1 Experimental Setup for Graph Classification Tasks

We evaluate our methods using social network datasets and bioinformatics datasets. They share the same experimental setups except for minor differences. The node features in social network networks are created using one-hot encodings of node degrees. The nodes in bioinformatics have categorical features. We use the TAPNet proposed in Section 3.3 that consists of one GCN layer and three blocks. The first GCN layer is used to learn low-dimensional representations of nodes in the graph. Each block is composed of one GCN layer and one TAP layer. All GCN and TAP layers output 48 feature maps. We use Leaky ReLU (Xu et al., 2015a) with a slop of 0.01 to activate the outputs of GCN layers. The three TAP layers in the networks select numbers of nodes that are proportional to the nodes in the graph. We use the rates of 0.8, 0.6, and 0.4 in three TAP layers, respectively. We use $\lambda = 0.1$ to control the importance of the graph connectivity term in the computation of ranking scores. Dropout (Srivastava et al., 2014) is applied to the input feature matrices of GCN and TAP layers with keep rate of 0.7. We use a two-layer feed-forward network as the network classifier. Dropout with keep rate of 0.8 is applied to input features of two layers. We use ReLU activation function on the output of the first layer on DD, PTC, MUTAG, COLLAB, REDDIT-MULTI5K, and REDDIT-MULTI12K datasets. We use ELU (Clevert et al., 2015) for other datasets. We train our networks using Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.001. To avoid over-fitting, we use L_2 regularization with $\lambda = 0.0008$. All models are trained using one NVIDIA GeForce RTX 2080 Ti GPU.

We will release our code in the final version.

References

All citations refer to the references in the main paper.