AttPool: Towards Hierarchical Feature Representation in Graph Convolutional Networks via Attention Mechanism

Jingjia Huang*[1,2], Zhangheng Li* [1,2], Nannan Li[1,2], Shan Liu[3], and Ge Li†[1,2]

[1]School of Electronic and Computer Engineering, Peking University Shenzhen Graduate School, Shenzhen, China
[2]Peng Cheng Laboratory, Shenzhen, China
[3]Tencent Media Lab, Palo Alto, CA94301, USA

Abstract

Graph convolutional networks (GCNs) are potentially insufficient in the ability to learn hierarchical representation for graph embedding, which holds them back in the graph classification task. To address the insufficiency, we propose AttPool, which is a novel graph pooling module based on an attention based mechanism, to remedy the problem. It is able to select nodes that are significant for graph representation adaptively, and generate hierarchical features via aggregating the attention-weighted information in nodes. Additionally, we devise a hierarchical prediction architecture to sufficiently leverage the hierarchical representation and facilitate the model learning. The AttPool module together with the entire training structure can be integrated into existing GCNs, and is trained in an end-to-end fashion conveniently. The experimental results on several graph-classification benchmark datasets with various scales demonstrate the effectiveness of our method.

1. Introduction

Inspired by the success of Convolutional Neural Networks (CNNs) in recent years, many researchers have attempted to extend convolution to graph-structured data, such as social networks [2] and bioinformatics [31]. Unlike grid-structured data, *e.g.,* images and videos, graph-structured data is often represented as the combination of irregularly-arranged nodes and connection-indicated edges, and therefore, the locality of graphs cannot be clearly defined. To handle this challenge, various graph convolutional networks (GCNs) have been proposed, which have achieved remarkable successes for representation learning on graphs [40, 20], especially for node-embedding learning based tasks, such as node classification [18, 37] and link prediction [1, 43]. However, without the pooling strategy, nearly all of existing GCNs are potentially lacking in ability to learn hierarchical representation for graph embedding, which holds them back in the graph classification task. Hierarchical information is important for graph classification. For example, in order to distinguish different organic molecules, it is beneficial to consider both local patch (e.g. individual atoms and their bonds) and coarse-grained structure (e.g. groups of atoms and bonds representing functional units in a molecule) rather than only local patterns [42]. To remedy the problem, a few researchers have tried to further generalize the pooling mechanism from CNNs to GCNs for hierarchical representation learning [42, 16, 6, 11].

Early works perform graph pooling either via applying a global pooling over all the node representations [16] or through coarsening the graph into clusters with deterministic assignment rules in prepossessing [6, 5]. Recently, learnable pooling layers are proposed in [11, 42] to obtain self-adapting cluster assignments, which are data-driven and can be trained in a differentiable way. Inspired by their work, in this paper we present an attention-based novel pooling strategy to generate hierarchical feature representations for input graphs. The attention mechanism allows a model to focus on task-relevant parts of the inputs, helping it to make better decisions. More recently, there has been a growing interest in incorporating the attention mechanism into encoding relationships of neighboring nodes and promoting node-embedding learning solutions [37, 35]. Unlike them, we utilize a global attention mechanism to evaluate the importance for each node in the graph globally. Then, we adaptively select discriminative nodes to form multi-granularity

*The authors have equal contribution to the work and are listed in alphabetical order.
†Corresponding author: Ge Li (email: geli@ece.pku.cn).

graph structures that generate hierarchical graph embedding for the classification. To prevent the attention from being stuck in certain parts of the graph and sampling redundant information, which is caused by the symmetric Laplacian smoothing in GCNs [22], we further propose a novel local-attention mechanism to keep the balance between the importance and the dispersion of nodes during the subgraph selection process. The proposed attention-based pooling, which is referred as AttPool, is also differentiable, and can be conveniently combined with various GCNs in a hierarchical way for an end-to-end training.

The hierarchical representation-learning architecture of the paper is constructed as follows: stack the convolution module over the pooling module, and repeat again. Based on the node representation learned by the convolution module at layer l, AttPool utilizes a trainable projection vector to calculate soft-attention values, and selects the nodes of top-k percent with largest values to build an attention-weighted coarser graph that will be fed in convolution module at layer $l+1$. At each layer, we represent the graph embedding via aggregating node information in a global pooling way. We also add supervision classifiers at each layer to build a hierarchical prediction structure in order to accelerate the convergence of model studying and adequately leverage the learned hierarchical graph representation. We indicate that combing AttPool as well as the hierarchical training strategy with existing GCNs can consistently boost state-of-the-art performance on several benchmark datasets for graph classification tasks.

In a nutshell, the contributions of this paper can be addressed as follows:

1. We introduce an attention-based pooling strategy to generate hierarchical representations for graph-structured data, which is implemented as an independent module that can be integrated into existing GCNs.
2. Apart from the naive global attention, we propose an alternative attention mechanism, the local attention, for the pooling module, which can be calculated conveniently with graph-structured data and prevent the attention being stuck in certain parts of the graph.
3. Combined with existing GCNs, our proposed hierarchical framework achieves state-of-the-art performance on several graph classification datasets with various scales.

2. Related Works

Graph convolution networks Inspired by the great success achieved by CNN on image-based tasks, a great deal on research of generalizing convolution to graph-based data has emerged recently. They broadly can be divided into two categories: spectral- [5, 6, 18, 16, 23] and spatial-based [15, 27, 30, 12] models. Based on spectral graph theory, Bruna et al. [5] define a variant of graph convolution in Fourier domain, which has defects of heavy computation and non-locality property. Since then, a series of works have been proposed to address the challenges. Kipf and Welling [18] acquire a simplified model by introducing 1-st approximation of the Chebyshev expansion in [6], which restricts the convolution to operating locally and is also the backbone network of this paper. Unlike spectral-based approaches that operate on the entire graph concurrently and has difficulty in scaling to large graph, spatial-based models define convolution on the graph directly to generate node presentations via aggregating information from its close neighbors. More recently, some researchers [15, 12] propose a sampling strategy to calculate node embedding in a batch fashion instead of taking the whole graph into consideration, which yields impressive performances on large-scale benchmarks.

Graph classification The goal of graph classification is to predict a label for the entire graph, which dependeds on the representation of the entire graph rather than node embeddings that are often directly obtained from GCNs. Previous works [8, 13, 30, 44] often tackle the problem via aggregating node representations in a flat, non-hierarchical manner. [6, 34, 9] seek to generate a hierarchical structure through running deterministic graph clustering algorithms on node features. For example, [28] obtains cluster assignments by applying k-means algorithm. However, they are two-stage approaches, and cannot be learned in an end-to-end fashion. In [44], Zhang et al. learn a sorting rule to order the nodes whose representations are then concatenated to feed into the final prediction architecture. [42, 11] propose trainable pooling layers that adaptively generate cluster assignments and can be learned in a differentiable way. DiffPool [42] may assign two arbitrary nodes to the same high-level node, which might be far away in the graph. Compared with DiffPool, AttPool naturally incorporates structural constrains and retains local structures in the pooled high-level graphs. Compared with GraphU-net[11], which simply selects the top-k percents of nodes and treats them as high-level nodes, AttPool generates more informative and abstractive high-level nodes and benets gradient back-propagation by the 1-hop aggregation. Therefore, AttPool is easier to train, and achieves better performance. Additionally, we propose a novel local-attention that prevents redundant information sampling in graphs.

Attention models Inspired by the success of attention models in deep learning communities[26, 3, 10, 36], researchers have introduced attention mechanism into graph-based model building, and various approaches have been proposed [37, 24, 1, 35] in recent years. Although they have different definitions of attention and use it for various purposes, they share the same ground in that attention is utilized to focus on most task-relevant parts for decision making. Attention mechanism benefits graph neural networks via aggregating information for nodes [37], integrating multiple models [24], or guiding random-walk with importance [1]. Here, attention module has two merits in our model: 1. it helps select discriminative nodes to form hierarchical graph structure; 2. it generates the graph representation with attention-weighted pooling.

3. Methodology

In this section, we present our attention-based graph convolutional network for the classification task. We propose two attention modules, global attention and local attention, under a common framework, which not only empowers the GCN model with the ability of learning hierarchical representations of a graph, but also prompts it to concentrate on significant task-related regions in graphs and makes better decisions.

3.1. Preliminaries

Let $G = (V, E)$ be a graph where V is the set of nodes and E is the set of edges between the nodes in V. We denote the adjacency matrix of G as $A = [A_{ij}]$. For an unweighted graph, $A_{ij} = 1$ if there exists an edge $(v_i, v_j) \in E$ and $A_{ij} = 0$ otherwise. In the case that G is a weighted graph, given a weight $w \in \mathbf{R}$ for edge $(v_i, v_j) \in E$, $A_{ij} = w$. A graph can be associated with node features F, where $F \in \mathbf{R}^{\mathbf{N}\times\mathbf{D}}$ is a matrix in which each row is a feature vector corresponding to a node in V. Given a graph dataset, graph classification task aims to learn a mapping from graphs to a set of corresponding labels.
Graph Neural Networks (GNNs) are effective tools for handling graph-structured data. We build our model upon a typical spectral-based Graph Convolutional Network introduced by Kipf and Welling [18], the core component of which is the graph convolutional layer (refferd as "GCN layer" for convenience) that operates message propagation and aggregation in the graph, and it is defined as:

$$H^{l+1} = ReLU(\hat{A} H^l W^l) \tag{1}$$

$$\hat{A} = \widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}} \tag{2}$$

Here, $\hat{A}$ is the symmetric normalized adjacency matrix of the graph. $\widetilde{A} = A + I$ is the adjacency matrix with self-loop added to each node, and $\widetilde{D}$ is a diagonal matrix where $\widetilde{D_{ii}} = \sum_j \widetilde{A}_{ij}$. H^l is the input of the l-th GCN layer and H^0 is initialized with the node feature matrix F. $W^l \in \mathbf{R}^{\mathbf{C_l}\times\mathbf{C_{l+1}}}$ is the matrix of learnable filters in layer l, where C_l is the dimension of the input feature and C_{l+1} is the channel number of W^l.

3.2. Graph Convolutional Network with Attention-based Pooling

The GCN layer enables a model to learn node embedding through propagating and aggregating information. However, with only the stacking of GCN layers, it is hard for a model to learn hierarchical features that are crucial for graph representation and classification. To remedy the limitation, generalizing pooling mechanism that has been successfully used in CNN to graph-structured data can be a solution. But, unlike grid-structured data, *e.g.* images, there is no natural notion of spatial locality in graph-structured data, which impedes us in applying pooling operation directly to graphs. Therefore, we propose the Attention-based Pooling (AttPool) layer, which selects the most significant nodes in the graph and operates information aggregation via the attention mechanism. Additionally, it can be trained end-to-end using stochastic gradient descent.
As shown in Fig. 1, AttPool operates down-sampling on graph data in two steps: first, it adaptively selects the top K-percent nodes from the current graph structure according to calculated attention values, and then aggregates the information from close neighbours to the selected nodes. We denote a graph processed by the l-th GCN as $G^l :< H^l, A^l >$, where H^l and A^l are the current hidden states and adjacency matrix, respectively. We define an attention function $f_{att}(H^l)$ to generate a positive weight Att_i for each node v_i in the graph, which can be interpreted as the relative importance given to v_i in the current graph. Then, the index of k-largest nodes are selected, where k is a dynamic scalar determined by down-sampling rate α. The process can be formulated as:

$$Idx = f_{TopK}(Att, k), \tag{3}$$

$$k = max(1, \lfloor \alpha N_l \rfloor), \tag{4}$$

$$Att = f_{att}(H^l), Att \in \mathbf{R}^{\mathbf{N_l}\times\mathbf{1}}, \tag{5}$$

where $f_{TopK}(\cdot)$ is a sorting function and produces indexes of the largest k nodes, and $\lfloor\cdot\rfloor$ is the operation of rounding down.
With the attention vector Att, the hidden state of the graph can be updated as:

$$H^{att^l} = Att \odot H^l, H^{att^l} \in \mathbf{R}^{\mathbf{N_l}\times\mathbf{C_l}} \tag{6}$$

where $\odot$ denotes element-wise product.
With the index, a subgraph that consists of significant parts of the model can be extracted, whose adjacent matrix is noted as:

$$A_i^{'} = \hat{A}^l_{Idx(i)}, A^{'} \in \mathbf{R}^{\mathbf{k}\times\mathbf{N_l}} \tag{7}$$

A softmax function is then applied to $A^{'}$ in a row-wise fashion for the purpose of normalization.
Then, with the selected nodes, we pool the information from H^{att^l} to H^{l+1} as:

$$H^{l+1} = A^{'} H^{att^l}, H^{l+1} \in \mathbf{R}^{\mathbf{k}\times\mathbf{C_l}}. \tag{8}$$

The pooling process is actually an attention-based information aggregation procedure that updates the hidden state of

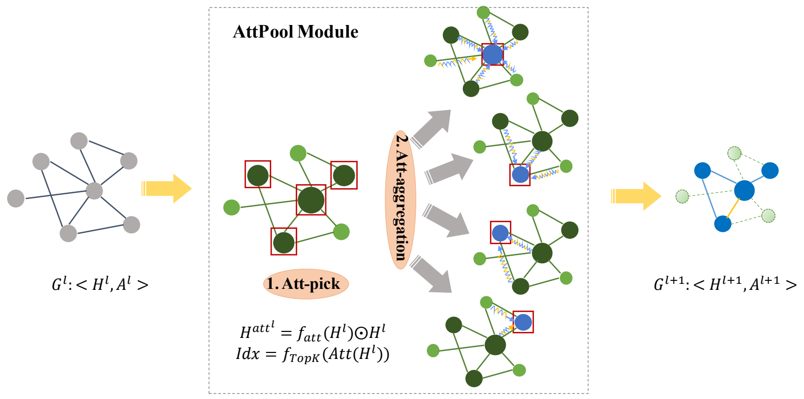
AttPool Module
2. Att-aggregation
$G^l\!:<H^l, A^l>$
1. Att-pick
$H^{att^l} = f_{att}(H^l) \odot H^l$
$Idx = f_{TopK}(Att(H^l))$
$G^{l+1}\!:<H^{l+1}, A^{l+1}>$

Figure 1. An illustration of the AttPool module. The AttPool module takes G^l as input and pick out nodes that are significant for graph presentation according to their attention coefficients calculated by function $f_{att}(\cdot)$. Then, attention-weighted information will be aggregated from the neighbours of each selected nodes to them.

current graph to a higher-level representation. The process retains more gradient back-propagation paths from subsequent layers to the attention function $f_{att}(\cdot)$ via aggregating features from neighbouring nodes that may not be selected, which helps the training of the attention module. Meanwhile, in some cases, if the important nodes are removed, the feature aggregation process would ensure important information is preserved to some extent.

As in [42], we take the distance between clusters, each of which consists of the 1-hop neighbors of a selected node in G^l, as the distance between the nodes in G^{l+1}, and update the adjacency matrix for the subgraph as:

$$\hat{A}^{l+1} = A^{'}\hat{A}^{l}A^{'T}, \hat{A}^{l+1} \in \mathbf{R}^{\mathbf{k}\times\mathbf{k}}, \tag{9}$$

H^{l+1} and $\hat{A}^{l+1}$ are then fed into next GCN module to learn higher-level patterns for the graph.

3.3. Learning Global Attention vs Local Attention

In this section we propose two alternative implementations for the attention function $f_{att}(\cdot)$, *i.e.* global attention and local attention.

3.3.1 Global Attention

The attention mechanism allows a model to focus on task-relevant parts of the inputs, helping it to make better decisions. Naturally, we can leverage an attention model to learn a series of normalized weights that can be interpreted as the relative importance given to each node. When we take all the nodes in a graph into account, that is what we call global attention. In practice, we compute the global attention coefficients as follows:

$$Att_i = softmax_i(H \times W) = \frac{exp(H_i \times W)}{\sum_{j\in N} exp(H_j \times W)}, \tag{10}$$

where $\times$ denote matrix multiplication. $W \in \mathbf{R}^{\mathbf{C}\times\mathbf{1}}$ is a shared linear transformation that is applied to every node. The softmax function is utilized to normalize the attention vector, which makes the attention coefficients comparable across different nodes.

3.3.2 Local Attention

Before the introduction of local attention mechanism, let us take a closer look to equation (1). According to equation (1), for a node v_i, the output of GCN at layer l is computed by:

$$H_i^{l+1} = ReLU(\sum_{j\in 1-hop(i)} \hat{A}_{ij}H_j^l W^l) \tag{11}$$

The equation (11) shows that the GCN layer computes the new features of a vertex as a linear transformation of the weighted-average features from itself as well as its 1-hop neighbours, where

$$\hat{A}H = (\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}})H \tag{12}$$

is actually a special form of Laplacian smoothing - symmetric Laplacian smoothing [22]. It reveals that the message in one node is propagated among itself and the 1-hop connected neighbours, which leads to the features of nodes within each connected component of the graph to be smoothed. That is to say, nodes staying close to each other tend to catch similar attention. As a result, a bunch of nodes sharing redundant information could be selected out, while nodes located at other parts of the graph, which also contain discriminative information, may be suppressed. Fig (4.a) and Fig (4.c) are two visualized samples for the "parochial selection".

Therefore, to prevent the attention from being stuck in a narrow region of the graph, we propose local attention to keep a balance between the importance and the dispersion during the sub-graph selection. As with global attention, we

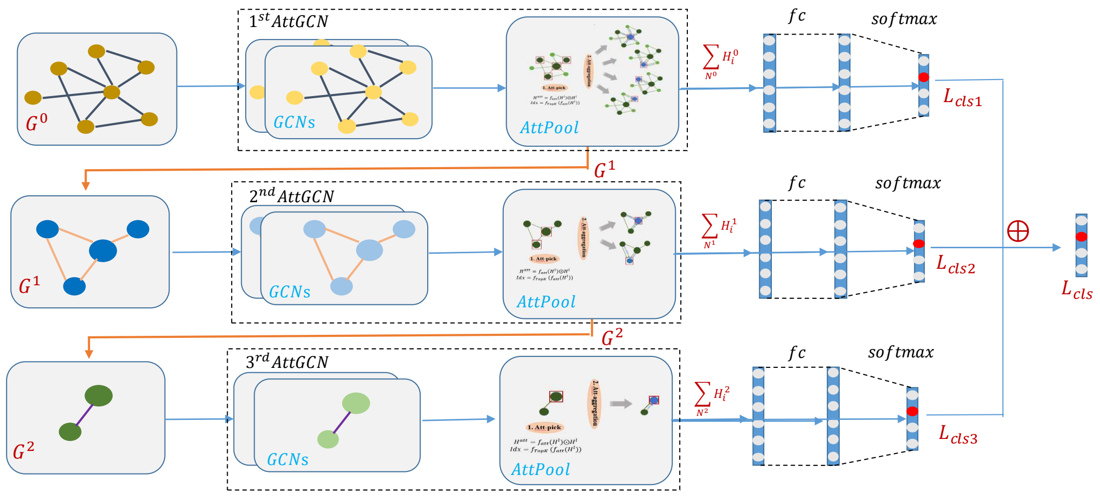

Figure 2. The illustration of a model with 3 AttGCN layers. The first AttGCN takes the original graph G^0 as input, while the second and third AttGCN layers take the sub-graphs generated by first and second modules as input, respectively. Each AttGCN layer is appended with a MLP that takes the graph embedding as input and predicts the label of the graph. $\oplus$ indicates that the loss and the predictions of different stages are added up to get the overall classification loss and final prediction, respectively.

first learn a shared linear transformation $W \in \mathbf{R}^{\mathbf{C}\times \mathbf{1}}$. After applying the linear transformation, we compute 1-hop attention for every node within its 1-hop neighbours rather than all of the nodes, simultaneously and independently:

$$Att_i^{1-hop} = \frac{\hat{A}_{ii} exp(H_i \times W \cdot \tau)}{\sum_{j\in N}(\hat{A}_{ij} exp(H_j \times W \cdot \tau))}, \tag{13}$$

It is actually a weighted softmax function with a learnable scalar τ that adjusts the sensitivity of the function. With equation (13), the attention coefficient of the node is rescaled via taking the importance ($H_j \times W$) as well as the distance ($\hat{A}_{ij}$) of its neighbours into account. Then, for each node, we finally calculate the local attention coefficient by:

$$Att_i = Att_i^{1-hop} \cdot \sum_{j\in N} \hat{A}_{ij} \tag{14}$$

where the attention value of each node is multiplied by its degree in order to eliminate the bias caused by the number of nodes' 1-hop neighbors for the fair comparison globally.

3.4. Hierarchical Prediction Architecture with AttPool

By stacking multiple GCN layers with AttPool layers inserted, we can construct an end-to-end trainable model for graph classification. To further boost the training process and fully utilize the hierarchical features, we adopt the intermediate supervision strategy [39, 29] and propose a hierarchical prediction architecture that is illustrated in Fig.(2). For convenience, we name the AttPool appended GCN layers AttGCN. For each AttGCN layer, it takes a graph as input, and outputs a sub-graph and the hidden state of the input graph. The sub-graph is taken as the input for the analysis in the next stage, *i.e.* another AttGCN layer, while the hidden state is added up across each node as the graph embedding and utilized for graph classification. The advantages of the architecture are two fold: in the training phase, attention-based node selection and graph representation learning for classification are mutually correlated and thus can reinforce each other and accelerate the convergence of the model. In the test phase, we can make a more reliable prediction via taking an comprehensive consideration of results with various receptive fields.

4. Experiments

4.1. Datasets

We conduct comprehensive experiments on several datasets commonly used for graph classification, which include real-world samples in different domains and various scales. Specifically, we apply our model on bio-infomatics datasets that consist of NCI1 [38], D&D [7] and PROTEINS [4], the scientific collaboration dataset COLLAB [21], and the social network datasets, namely REDDIT-BINARY and REDDIT-MULTI-12K [41]. According to the average node numbers for each graph as presented in Table 3, these datasets can be grouped into *small* (NCI1, PROTEINS), *medium* (COLLAB, D&D), and *large* (REDDIT-BINARY, REDDIT-MULTI-12K) scale datasets. To better justify the efficacy of our method, we will discuss the experimental results for each of the different groups.

4.2. Experimental Setups

In our experiments, we set a 3-layer GCN as our baseline, and compare our method with both the state-of-the-art

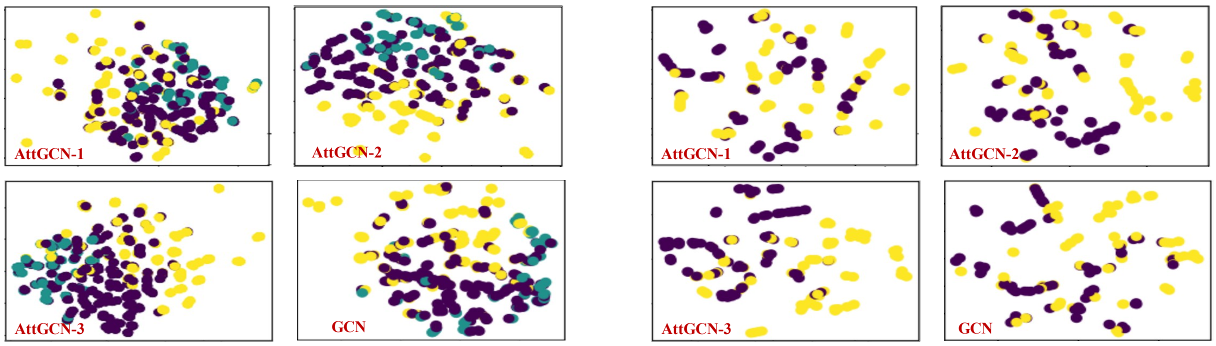

Figure 3. Visualization of graph embedding from GCN baseline and each AttGCN layer of AttPool-L model on COLLAB (left) and REEDIT-BINARY (right). "AttGCN-i" indicates the result of the i th layer in the model. Points of different colors indicate graph samples with different labels. The visualization is done with t-SNE [25].

Table 1. Comparisons of classification accuracy achieved by different AttGCN layers in our models. "AVG" means to fuse the prediction results via averaging the classification scores of different AttGCN layers.

	NCI1		COLLAB		REDDIT-BINARY	
LAYER	G	L	G	L	G	L
GCN	75.74		68.18		85.20	
1ST	76.40	77.06	72.36	70.31	86.60	91.10
2ND	78.93	78.27	75.96	77.31	88.30	91.15
3RD	77.93	78.18	75.52	78.09	87.40	92.30
4TH	77.32	77.37	75.00	77.40	86.90	91.65
AVG	80.58	81.68	77.04	79.66	90.30	93.15

kernel-based approaches [33, 41, 19, 32] and graph neural networks based approaches [15, 30, 44, 42]. For a fair comparison [42, 44], we run every different experiment with 10-fold cross validation, and the model at the epoch with the best cross-validation accuracy averaged over the 10 folds was selected. For the datasets preprocessing, we follow the official code of the SortPool [44]. Our models have 4 AttGCN layers with 64 hidden units for NCI-1, COLLAB and REDDIT datasets. We set the layer number to 3 and halve the hidden units for D&D and PROTEINS, which have fewer training samples, to prevent over-fitting. We use GCN as our backbone network. The first AttGCN layer uses 3 GCN layers while subsequent AttGCN layers use 1 GCN layer for basic reasoning. In our main experiments, the pooling ratio is set to 0.5. We use a 2-layer MLP with 100 hidden units for all datasets. We utilize Xavier normal distribution[14] for weight initialization, and use cross-entropy loss for training. For all datasets, we train the networks for 500 epochs using Adam Optimizer [17] with a learning rate of 0.001 and set the batch size to 20. More implementation details and codes can be accessed in the supplementary materials[1].

[1]https://github.com/hjjpku/Attention_in_Graph

4.3. Hierarchical Representation for Graph Classification

In this subsection, we justify the effectiveness of the hierarchical representation for graph classification tasks via studying the performance of each AttGCN layer in our models. As shown in Table.(1), the first AttGCN layers of both the AttPool-G and AttPool-L models perform slightly better than the GCN baseline on the three datasets, which demonstrates that gradients from deeper AttGCN layers are of benefit to the training of shallow AttGCN layers. Meanwhile, we can observe that accuracy of deeper layers is always higher than that of the first layer, while the location of peak points varies depending on datasets. It reveals that different levels of graph embedding have different representation capabilities, and the most effective representation emerges at different layers according to datasets. Additionally, we find that when we fuse the predictions of different AttGCN layers by averaging the scores, better performance can be achieved than that of any single AttGCN layer. The result suggests that hierarchical information is significant for graph classification tasks. To make the discussion more concrete, we visualize graph embedding from GCN baseline and different AttGCN layers of our model on COLLAB and REDDIT-BINARY datasets. In Fig.(3), points of different colors indicate graph samples with different labels. We can see that embedding from the 3rd layer of AttPool-L models show better discriminability than the other layers, which is consistent with the results in Table.(1).

4.4. Comparisons with Other Pooling Methods

To evaluate the efficacy of the AttPool module, we set the GCN model as baseline, and compare our AttPool against another two differentiable graph pooling mechanisms, *i.e.* Graph U-net pool [11] and DiffPool [42]. Specifically, we replace the AttPool module in our model with the pooling modules of Graph U-net and DiffPool, and keep the backbones and other parts of the networks all the same (we

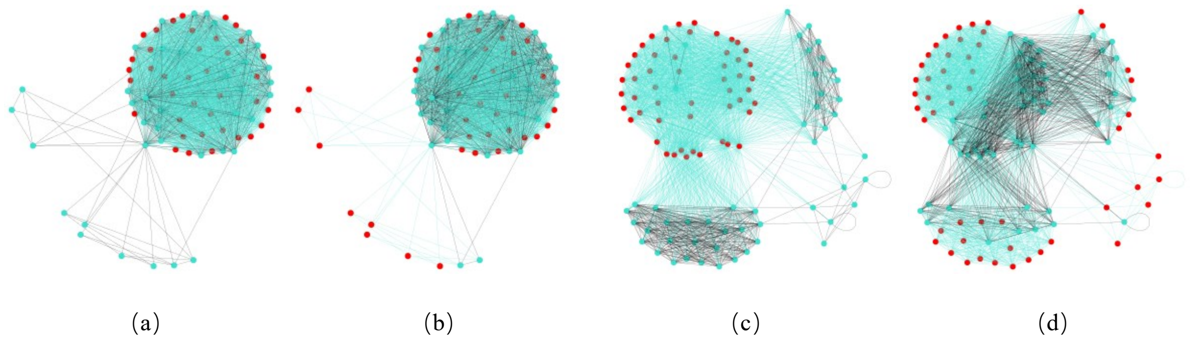

Figure 4. Visualization of attention based node selection in first AttPool layer, using two samples from COLLAB. 50% of nodes are selected for sub-graph construction and labeled in red. The figure (a), (b) and figure (c), (d) are two pairs of samples where (a), (c) are the outputs of the global attention based pooling module while (b), (d) are the outputs of the local attention based pooling module.

denote these two modified networks as Graph U-net* and DiffPool*). For consistency with the original DiffPool approach, we use a 3-layer GCN for assignment matrix generation in each DiffPool layer, and the maximal number of nodes after DiffPool layer is set to 2 times of the average node number of each dataset. For a fair comparison, we do not adopt auxiliary methods (*e.g.* auxiliary link prediction objective [42] and graph connectivity augmentation [11]) for neither our method nor the compared methods.

As shown in Table 2, compared with the baseline, our AttPool achieves superior performances on all of the three datasets. For Graph U-net* and DiffPool*, they bring about performance benefits on NCI1 and COLLAB datasets, but scores relatively moderately on REDDIT-BINARY compared with the GCN model. From the comparison of different pooling mechanisms, we can find that AttPool and DiffPool*, both of which perform information aggregation during the pooling operation, achieve better performance than Graph U-Net*. It reveals that the information aggregation can be helpful for graph representation learning, especially on medium and large scale datasets.

Table 2. Comparisons of classification accuracy of models with different attention modules. * indicates the models with our implementations of Graph U-net pool layer or DiffPool layer.

Dataset / Module	NCI1	COLLAB	REDDIT-BINARY
GCN	75.74	68.18	85.20
Graph U-net*	79.52	69.42	70.25
DiffPool*	78.32	76.18	83.75
AttPool-G	80.58	77.04	90.30
AttPool-L	**81.68**	**79.66**	**93.15**

4.5. Hierarchical Prediction Architecture V.S. Conventional Classification Architecture

In this subsection, we validate the efficiency of the hierarchical prediction architecture that utilizes multi-granularity features to predict the graph label and leverages multi-layer classification losses to train the model. We compare the proposed learning architecture against the conventional classification architecture that only appends an MLP to the head of the model. Additionally, to show the generality of this architecture, we also present contrasting results for the Graph U-Net* and DiffPool*.

The experimental results are presented in Table 4. Models with the hierarchical prediction architecture are denoted by *w/ H*, while *w/o H* denotes the models with conventional classification architecture. As shown in Table 4, when compared with *Graph U-Net* w/o H*, the hierarchical prediction architecture gives Graph U-Net* significant improvement on NCI1 and REDDIT-BINARY. For DiffPool* and AttPool-G, better results can be achieved on NCI1 and COLLAB with the proposed architecture. *AttPool-L w/ H* consistently outperforms *AttPool-L w/o H* on the three datasets and achieves best performance among the methods mentioned in Table 2.

4.6. Global Attention V.S. Local Attention

In this subsection, we discuss the performance of the two alternative attention mechanisms proposed in this paper. We report the results of the AttPool-G and the AttPool-L in Table 3. We find that on small scale and medium scale datasets, global attention and local attention have competitive performance. For the two large scale datasets, local attention performs 2.85% better than global attention on REDDIT-BINARY, and gives 2.9% improvements on REDDIT-MULTI-12K. The results illustrate that local attention can be a better choice for the representation learning on large graphs.

To further conceptualize the difference of global attention and local attention, we visualize the attention-based node selections with two graphs from COLLAB. For each graph, 50% of nodes from the first AttPool module are selected in our models. As shown in Fig 4, we can find that global at-

Table 3. Comparisons of classification accuracy of different graph models. The **bold fonts** denote the best accuracy on each dataset. * indicates our implementation of GCN.

Model	NCI1	PROTEINS	COLLAB	D&D	REDDIT-BINARY	REDDIT-MULTI-12K
NUM. OF GRAPHS	4110	1113	5000	1178	2000	11929
AVG. NUM. OF NODES	30	39	74	284	430	391
AVG. NUM. OF EDGES	32	73	2458	716	498	457
GRAPHLET	62.49	71.39	64.66	74.38	–	21.73
DGK	80.31	75.68	73.09	–	78.04	32.22
WL	**84.46**	73.76	78.61	74.02	80.8	39.03
PSCN	–	75.00	72.6	76.27	–	41.32
GRAPHSAGE	–	70.48	68.25	75.42	–	42.24
SORTPOOL	74.44	75.54	73.36	79.37	–	–
DIFFPOOL	–	76.25	75.48	**80.64**	–	47.08
GCN*	75.74	71.08	68.18	72.07	85.20	44.67
ATTPOOL-G	80.58	**76.50**	77.04	79.20	90.30	46.53
ATTPOOL-L	81.68	75.14	**79.66**	76.07	**93.15**	**49.40**

Table 4. Comparisons of classification accuracy of models with the hierarchical prediction architecture (w/ H) and conventional classification architecture (w/o H). * indicates the models with our implementations of Graph U-net-pool layer or DiffPool layer.

DATASET / MODULE	NCI1	COLLAB	REDDIT-BINARY
GRAPH U-NET* W/O H	68.02	**71.65**	51.8
GRAPH U-NET* W/ H	**79.52**	69.42	**70.25**
DIFFPOOL* W/O H	77.73	75.00	82.44
DIFFPOOL* W/ H	**78.32**	**76.18**	**83.75**
ATTPOOL-G W/O H	76.93	75.70	**90.40**
ATTPOOL-G W/ H	**80.58**	**77.04**	90.30
ATTPOOL-L W/O H	77.80	77.64	91.80
ATTPOOL-L W/ H	**81.68**	**79.66**	**93.15**

tention tends to focus on the main part of the graph while local attention tends to retain the hyper-architecture of the graph with a more sparse node-selection strategy.

4.7. Comparisons with Other Graph Classification Models

We compare the graph classification performance of our model against other representative GNN-based methods, namely GCN [18], GraphSage [15], PSCN [30], SortPool [44], DiffPool [42], as well as some well-known kernel-based methods, such as Graphlet [33], WEISFEILER-LEHMAN subtree kenerl (WL) [32] and Deep Graph Kernel (DGK) [41]. The performance comparison can be seen in Table 3. Our method brings about a significant improvement to the GCN baseline, which arrives an average of 6.91% over all the 6 datasets. Additionally, we achieve the best results on 4 out of 6 benchmarks and improve the state-of-the-art performance, especially on the large scale datasets (*i.e., REDDIT*). It can be observed that previous state-of-the-art models usually have varied performances over different datasets. For example, the performances of WL method is significantly ahead of other previous methods in the NCI1 and COLLAB datasets, but falls behind the DiffPool by a large margin in the PROTEINS, D&D and REDDIT-MULTI-12K datasets. Compared with previous models, the AttPool behaves more consistently over various datasets, which shows our proposed attention mechanism can extract better representation for a wide family of graph-structured data.

5. Conclusion

In this paper, we present learning the hierarchical feature representation for a graph via a novel attention-based pooling mechanism. The devised attention-pooling layer selects discriminative nodes based on calculated attention values to build the coarser graph. Meanwhile, the graph embedding at each layer is generated via aggregating the node representation in an attention-weighted manner. We also design a hierarchical prediction-learning structure for the graph classification task. The attention pooling module together with the hierarchical learning strategy can be combined with existing GCNs and be trained end-to-end. We test the proposed hierarchical learning framework on several graph-classification benchmark datasets, and achieve superior or comparable results compared with other representative methods.

Acknowledgment

This work is supported by Shenzhen Municipal Science and Technology Program (No. JCYJ20170818141146428), National Engineering Laboratory for Video Technology - Shenzhen Division, Shenzhen Key Laboratory for Intelligent Multimedia and Virtual Reality (No. ZDSYS201703031405467), and Shenzhen Fundamental Research Program (JCYJ20170818141120312).

References

[1] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alex Alemi. Watch your step: Learning graph embeddings through attention. *arXiv preprint arXiv:1710.09599*, 2017. 1, 2, 3

[2] Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 635–644. ACM, 2011. 1

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 2

[4] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005. 5

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 1, 2

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016. 1, 2

[7] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003. 5

[8] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015. 2

[9] M. Fey, J. E. Lenssen, F.Weichert, and H. Mller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 2

[10] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4476–4484. IEEE, 2017. 2

[11] Hongyang Gao and Shuiwang Ji. Graph u-net, 2019. 1, 2, 6, 7

[12] H. Gao, Z.Wang, and S. Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1416–1424, 2018. 2

[13] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272, 2017. 2

[14] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. 6

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017. 2, 6, 8

[16] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 1, 2

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[18] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 1, 2, 3, 8

[19] Nils M Kriege, Pierre-Louis Giscard, and Richard Wilson. On valid optimal assignment kernels and applications to graph classification. In *Advances in Neural Information Processing Systems*, pages 1623–1631, 2016. 6

[20] John Boaz Lee, Ryan A Rossi, Sungchul Kim, Nesreen K Ahmed, and Eunyee Koh. Attention models in graphs: A survey. *arXiv preprint arXiv:1807.07984*, 2018. 1

[21] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM, 2005. 5

[22] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *CoRR*, abs/1801.07606, 2018. 2, 4

[23] R. Li, S. Wang, F. Zhu, and J. Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3546–3553, 2018. 2

[24] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. In *Proceedings of the International Conference on Learning Representations*, 2005. 2, 3

[25] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 6

[26] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014. 2

[27] F. Monti, D. Boscaini, J. Masci, E. Rodol, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[28] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B Tenenbaum, and Daniel LK Yamins. Flexible neural representation for physics prediction. In *arXiv preprint arXiv:1806.08047*, 2018. 2

[29] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016. 5

[30] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016. 2, 6, 8

[31] Jian Pei, Daxin Jiang, and Aidong Zhang. On mining cross-graph quasi-cliques. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 228–238. ACM, 2005. 1

[32] Nino Shervashidze, Pascal Schweitzer, Erik Jan, Van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(3):2539–2561, 2011. 6, 8

[33] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pages 488–495, 2009. 6, 8

[34] M. Simonovsky and N. Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 29–38, 2017. 2

[35] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018. 1, 2

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. 2

[37] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 1(2), 2017. 1, 2

[38] Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008. 5

[39] Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. Premise selection for theorem proving by deep graph embedding. In *Advances in Neural Information Processing Systems*, pages 2786–2796, 2017. 5

[40] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019. 1

[41] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM, 2015. 5, 6, 8

[42] 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*, pages 4805–4815, 2018. 1, 2, 4, 6, 7, 8

[43] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *arXiv preprint arXiv:1802.09691*, 2018. 1

[44] 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. 2, 6, 8