Node Information Awareness Pooling for Graph Representation Learning

Chuan Sun, Feihu Huang, and Jian Peng(✉)

College of Computer Science, Sichuan University, Chengdu, China
sunchuan@stu.scu.edu.cn, {huangfh,jianpeng}@scu.edu.cn

Abstract. Graph neural networks (GNNs), which generalize deep neural network models to graph structure data, have attracted increasing attention and achieved state-of-the-art performance in graph-related tasks such as graph classification, link prediction, and node classification. To adapt GNNs to graph classification, existing works aim to define the graph pooling method to learn graph-level representation by downsampling and summarizing the information present in the nodes. However, most existing pooling methods lack a way of obtaining information about the entire graph from both the local and global aspects of the graph. Moreover, in these pooling methods, the difference features between nodes and their neighbors are usually ignored, which is crucial in obtaining graph information in our opinions. In this paper, we propose a novel graph pooling method called Node Information Awareness Pooling (NIAPool), which addresses the limitations of previous graph pooling methods. NIAPool utilizes a novel self-attention framework and a new convolution operation that can better capture the difference features between nodes to obtain node information in the graph from both local and global aspects. Experiments on five public benchmark datasets demonstrate the superior performance of NIAPool for graph classification compared to the state-of-the-art baseline methods.

Keywords: Graph pooling · Graph classification · Graph representation learning · Graph neural network

1 Introduction

Graph neural networks (GNNs) are a class of deep learning models that operate on data represented as graphs with arbitrary topological structures such as body skeletons [22], brain networks [14], molecules [6], and social networks [12]. Unlike some regular grid data (e.g., images and texts), the inputs of GNN are permutation-invariant variable-size graphs consisting of rich information. By passing, transforming, and aggregating node features across the graph, GNNs can capture graph information effectively [9] and demonstrate strong ability on related tasks such as text classification [10], mental illness analysis [32], drug discovery [24], relation extraction [25], and particle physics analysis [23]. Some of the existing methods focus on node-level representation learning to perform tasks such as link prediction [11,18] and node classification [12,27]. Others focus on learning graph-level representations for tasks like graph classification [7,34] and graph regression [21]. In this paper, we focus on graph-level representation learning for the task of graph classification.

In brief, the task of graph classification is to generate the graph-level representation of the entire graph to predict the label of the input graph by utilizing the given graph structure information and node representation. The majority of existing GNNs usually generate the graph-level representation by applying simple global pooling strategies [15,28,35] (i.e., a summation over the final learned node representations). These methods are inherently "flat" [36] and lack the capability of aggregating node information in a hierarchical manner since they treat all the nodes equivalently when generating graph representation using the node representations. Furthermore, the structure information of the entire graph is almost neglected during this process. For example, to prove a molecule is toxic or not, which depends on not only the features of atoms but also the structure information in atoms' interaction networks.

To address this problem, hierarchical pooling architectures have been proposed, which have the ability to coarsen the graph in an adaptive, data-dependent manner within a GNN pipeline, analogous to image downsampling within convolutional neural networks. The first end-to-end learnable hierarchical pooling operator is DiffPool [33]. DiffPool groups nodes into super-nodes by computing soft clustering assignments of nodes. Since the cluster assignment matrix is dense, its ability to scale to large graphs is limited. TopKPool [7] uses a simple scalar projection score for each node to select top-k nodes in a sparse pooling manner, so the computing limitation of DiffPool is overcome. Following that, SAGPool [13], a variant of TopKPool, uses self-attention to incorporate global structure information, and EdgePool [4] learns a localized and sparse hard pooling transform by edge contraction. Nevertheless, a critical limitation of these pooling operations is the lack of a way to combine the local and global information in graphs simultaneously. To address the above limitations, ASAP [17] devises a cluster scoring procedure to select nodes depending on the feature-based fitness scores and self-attention network. [8] uses a two-stage attention voting process that selects more important nodes in a graph. Although [17] and [8] have proved the ability of their methods to capture graph information, we argue that self-attention mechanisms in these methods may render mutual exclusion effect on node importance, which means these methods focus too much on the nodes with high similarity and almost ignore the node information with high differences. In addition, we find that the difference features of nodes are vital for the entire graph. For example, even if two molecules have the same structure and atomic number, as long as there is a pair of different types of atoms, there will be huge differences between the molecules. However, most of the existing methods cannot effectively capture the difference features between nodes or even do not take them into consideration. Therefore, hierarchical graph pooling methods that capture the difference features between nodes effectively while reasonably considering both local and global graph information currently do not exist.

In this work, we propose a new pooling operator called Node Information Awareness Pooling (NIAPool) which overcomes the problems mentioned above. It uses a novel Difference2Token attention framework (D2T), which balances important node selection reasonably, to enhance the node information representation locally, and then utilizes a new convolution operation called Neighbor Feature Awareness Convolution (NFAConv) to capture the difference features between neighbor nodes and perform global node scoring.

Our contributions can be summarized as follows:

– We propose D2T to evaluate each node's information given its neighborhood and then enhance local node information.
– We propose a new convolution operator NFAConv. Compared with state-of-the-art convolution operations, NFAConv is more powerful in extracting difference features between nodes.
– We conduct extensive experiments on five public datasets to demonstrate NIAPool's effectiveness as well as superiority compared to a range of state-of-the-art methods.

2 Preliminaries

2.1 Notations and Problem Formulation

Given a set of graph data $D = \{(G_1, y_1), (G_2, y_2), \ldots, (G_n, y_n)\}$, where the number of nodes and edges in each graph might be different. For an arbitrary graph $G_i = (\mathcal{V}_i, \mathcal{E}_i, X_i)$ with $N = |\mathcal{V}_i|$ nodes and $|\mathcal{E}_i|$ edges. Let $A \in \mathbb{R}^{N \times N}$ be the adjacent matrix describing its edge connection information, and $X \in \mathbb{R}^{N \times d}$ represents the node feature matrix assuming each node has d features. Each graph is also associated with a label y_i indicating the class it belongs to. The goal of graph classification is to learn a mapping function $f : \mathcal{G} \rightarrow \mathcal{Y}$ where $\mathcal{G}$ is the set of graphs and $\mathcal{Y}$ is the set of labels. A pooled graph and its adjacency matrix are denoted by $G^p = (\mathcal{V}^p, \mathcal{E}^p, X^p)$ and A^p, respectively. For each node v_i, we use $\mathcal{N}(v_i)$ to represent its 1-hop neighbors.

2.2 Graph Convolutional Neural Network

Graph convolutional neural network (GCN) [12] is a powerful tool for handling graph-structured data and has shown promising performance in various challenging tasks. Thus, we choose GCN as the building block to design the framework for graph classification. For the l-th layer in GCN, it takes both the adjacent matrix A and hidden representation matrix X^l of the graph as input, and then the new node embedding matrix will be generated as follows:

$$X^{l+1} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X^l W^l\right) \tag{1}$$

Here, $\sigma(\cdot)$ is the non-linear activation function, $\tilde{A} = A + I$ is the adjacency matrix with self-loops. $\tilde{D}$ is the diagonal degree matrix of $\tilde{A}$, and $W^l \in R^{d_l \times d_{l+1}}$ is the trainable weight matrix.

2.3 Self-attention Mechanism

Self-attention is used to discover the dependence of input on itself [26]. The attention coefficient $\alpha_{i,j}$ is computed to map the importance of candidates h_i on target query h_j, where h_i and h_j are obtained from input entities $\boldsymbol{h} = \{h_1, \ldots, h_n\}$. We introduce three variants of self-attention mechanisms.

Token2Token (T2T) [19] explores the dependency between the target and candidates from the input set $\boldsymbol{h}$. The attention coefficient $\alpha_{i,j}$ is computed as:

$$\alpha_{i,j} = \text{softmax}\left(\vec{v}^T \sigma\left(Wh_i \| Wh_j\right)\right) \tag{2}$$

where $\|$ is the concatenation operator.

Source2Token (S2T) [19] drops the target query term to explore the dependency between each candidate and the entire input set $\boldsymbol{h}$.

$$\alpha_{i,j} = \text{softmax}\left(\vec{v}^T \sigma\left(Wh_j\right)\right) \tag{3}$$

Master2Token (M2T) [17] is a self-attention mechanism that works on graph data. M2T utilizes intra-cluster information by using a master function to generate a query vector within the node and its 1-hop neighbors. Compared with other self-attention mechanisms, M2T can capture graph information better. Formally, M2T is defined as:

$$\alpha_{i,j} = \text{softmax}\left(\vec{v}^T \sigma\left(Wm_i \| h_j\right)\right) \tag{4}$$

$$m_i = \max_{h_i \in \mathcal{N}(h_j)} \left(h_i\right) \tag{5}$$

Here, h_j refer to node representation in graph and $\mathcal{N}(h_j)$ are the 1-hop neighbors of h_j.

3 NIAPool (Proposed Method)

In this section, we give an overview of the proposed method NIAPool. As shown in Fig. 1, NIAPool initially focuses on the local structure of the given graph, considering all nodes and their 1-hop neighbors, and utilizes a new self-attention to enhance node information by aggregating neighbor node features. These enhanced nodes are then globally scored using a modified GCN. Further, a fraction of the top-scoring nodes will be saved. Below, we discuss the modules of NIAPool in detail.

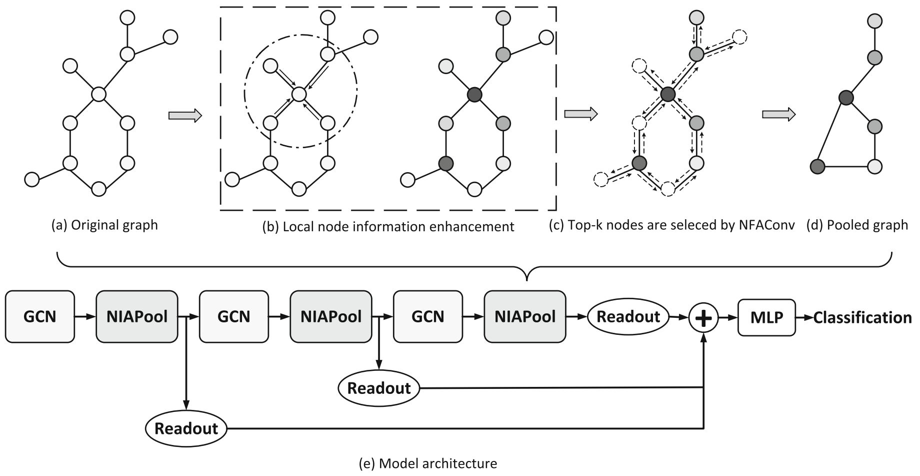

Fig. 1. The illustration of the proposed NIAPool: (a) Input graph to NIAPool. (b) Local node information enhancement based on D2T attention. (c) We utilize NFAConv to capture node difference features and then score nodes. (d) A fraction of top-scoring nodes are kept in the pooled graph. (e) The overview of hierarchical graph classification architecture

3.1 Local Node Information Enhancement

Initially, we consider each node and its 1-hop neighbors. We learn the attention coefficient between each node in the graph and its neighbor nodes through the self-attention mechanism. Further, the learned attention coefficient is used to fuse information from v_i and its neighbors $\mathcal{N}(v_i)$. The task here is how to learn the enhancement representation of nodes effectively by attending to the relevant nodes. We observe that the self-attention mechanisms mentioned in Sect. 2.3 may have a node importance mutual exclusion problem (Please refer to Sect. 4.3 or Sect. 1 for more details). To address this problem, we propose a new variant of self-attention, called **Difference2Token (D2T)** to balance the attention procedure. D2T is defined as:

$$\alpha_{i,j} = \text{softmax}\left(\vec{v}^T \sigma\left(W_1\left(h_i - h_j\right) \| W_2 h_j\right)\right) \tag{6}$$

Here, h_i is node representation and $h_j \in \mathcal{N}(h_i)$.

Our motivation for designing D2T is that if a node's information can be reconstructed or inferred by its neighborhood information, it means this node can probably be deleted in the pooled graph with almost no information loss. In general, the nodes with similar information can be substituted for each other. That is, the more similar the nodes are, the less important they are, which is different from the self-attentions in Sect. 2.3. In Eq. (6), the difference $h_i - h_j$ will show significant differences when a node representation h_i is similar to one representation of its neighbor h_j or not, so D2T can balance the attention procedure reasonably by taking $h_i - h_j$ into consideration.

After obtaining the attention coefficient, node information can be enhanced as follows:

$$h_i^e = h_i + \sum_{j=1}^{|\mathcal{N}(v_i)|} \alpha_{i,j} h_j \tag{7}$$

where h_i^e is the enhanced node representation.

3.2 Global Node Scoring Using NFAConv

The difference feature between nodes is critical for generating graph-level representation. For example, even if the graph has the same topology, different node features can make the whole graph greatly different. How to effectively capture the difference features between nodes becomes a new problem, which is ignored by most graph classification methods. To solve this problem, inspired by [17] and [30], we propose **Neighbor Feature Awareness Convolution (NFAConv)**, a powerful variant of GCN which is aware of node features:

$$\theta_i = h_i \cdot W + \phi\left(\sum_{h_j \in \mathcal{N}(h_i)} (h_i - h_j) \ \| \ (h_i \odot h_j)\right) \tag{8}$$

where W is learnable parameter matrix and ϕ denotes a neural network. $\odot$ is broadcasted hadamard product.

NFAConv utilizes $h_i - h_j$ and $h_i \odot h_j$ to obtain difference features between nodes and their neighbors, and then a neural network ϕ is applied to extract useful information from these features. In this paper, ϕ is a Multilayer Perceptron (MLP) with three linear layers. It is worth noting that ϕ can also be Convolutional Neural Network (CNN) like the one used in [35], or other neural networks. We keep this for future work.

After enhancing local node information, we sample nodes based on the global node fitness score θ_i which calculated by NFAConv. For a given pooling ratio $k \in [0, 1)$, the top $\lceil kN \rceil$ nodes are saved in the pooled graph G^p.

3.3 Graph Coarsening

Following the graph coarsening procedure in [33], we make global node fitness vector $\mathbf{\Theta} = [\theta_1, \theta_2, \ldots, \theta_N]^T$ learnable by multiplying it to the node feature matrix X^l. The indices of selected nodes $\hat{i}$ are obtained by choosing the top $\lceil kN \rceil$ nodes based on X^l:

$$\hat{X}^l = \mathbf{\Theta} \ \odot \ X^l, \quad \hat{i} = \mathrm{TOP}_k\left(\hat{X}^l, \lceil kN \rceil\right) \tag{9}$$

The node feature matrix $X^p \in \mathbb{R}^{\lceil kN \rceil \times d}$ and the pruned cluster assignment matrix $\hat{S} \in \mathbb{R}^{N \times \lceil kN \rceil}$ are given by:

$$X^p = \hat{X}^l(\hat{i}, :), \quad \hat{S} = S(:, \hat{i}) \tag{10}$$

Here, $S \in \mathbb{R}^{N \times N}$ is the cluster assignment matrix, where $S_{i,j}$ represents the membership of node $v_i \in \mathcal{V}$ in $\mathcal{N}(v_j)$. Finally, the pooled new adjacency matrix A^p can be obtained as follows:

$$A^p = \hat{S}^T (A + I) \hat{S} \tag{11}$$

3.4 Model Architecture

The architecture used in SAGPool [13] is adopted in our experiments. We use JK-net [31] as our readout layers to aggregate pooled node features. Figure 1 shows the details of the model architecture.

4 Experiments

In this section, we present our experimental setup and results. Our experiments are designed to answer the following questions:

- **Q1** How does NIAPool perform compared to other state-of-the-art pooling methods at the task of graph classification?
- **Q2** Is local node information enhancement by D2T attention more powerful compared to other self-attentions?
- **Q3** Compared with some state-of-the-art GCN variants, can NFAConv effectively capture the features of nodes in the graph?

4.1 Experimental Setting

Datasets: Five graph datasets are selected from the public benchmark graph data collection. D&D [5,20] and Proteins [3,5] are two datasets containing graphs of protein structures. NCI1 [29] and NCI109 are biological datasets used for anticancer activity classification. Frankenstein [16] is a set of molecular graphs representing the molecules with or without mutagenicity. Table 1 summarizes the statistics of all datasets.

Table 1. Statistics of the datasets.

Dataset	# $\|G\|$	# $\|c\|$	Avg.# $\|\mathcal{V}\|$	Avg.# $\|\mathcal{E}\|$
D&D	1178	2	284.32	715.66
Proteins	1113	2	39.06	72.82
NCI1	4110	2	29.87	32.30
NCI109	4127	2	29.68	32.13
Frankenstein	4337	2	16.90	17.88

Baselines: We compare NIAPool with previous state-of-the-art graph pooling methods, including Set2Set [28], GlobalAttention [15], SortPool [35], DiffPool [33], TopKPool [7], SAGPool [13], ASAP [17], EdgePool [4], GSAPool [34], MinCUT-Pool [2], HGP-SL [36], and GMT [1].

Training Procedures: In our experiments, we use 10-fold cross-validation to evaluate the pooling methods, where each time we split each dataset into three parts: 80% as the training set, 10% as the validation set, and the remaining 10% as the test set. The average performance with standard derivation is reported. For NIAPool, we choose $k = 0.5$ and set the dimension of node representations 128 for all datasets. For baseline algorithms, we use source codes released by authors and follow the experimental setup that is mentioned in their manuscript. Adam optimizer with a learning rate of 0.001 is adopted as our optimizer.

4.2 Q1: Comparison with Baseline Models

We compare the performance of NIAPool with baseline methods on five datasets. The graph classification results are reported in Table 2. We also show the different pooling ratios based on the NIAPool architecture in Table 3.

Table 2. Comparison with the state-of-the-art graph pooling methods.

Methods	D&D	Proteins	NCI1	NCI109	Frankenstein
Set2Set	71.60 ± 0.87	72.16 ± 0.43	66.97 ± 0.74	61.04 ± 2.69	61.46 ± 0.47
GlobalAttention	71.38 ± 0.78	71.87 ± 0.60	69.00 ± 0.49	67.87 ± 0.40	61.31 ± 0.41
SortPool	71.87 ± 0.96	73.91 ± 0.72	68.74 ± 1.07	68.59 ± 0.67	63.44 ± 0.65
DiffPool	66.95 ± 2.41	68.20 ± 2.02	62.32 ± 1.90	61.98 ± 1.98	60.60 ± 1.62
TopKPool	75.01 ± 0.86	71.10 ± 0.90	67.02 ± 2.25	66.12 ± 1.60	61.46 ± 0.84
SAGPool	76.45 ± 0.97	71.86 ± 0.97	67.45 ± 1.11	67.86 ± 1.41	61.73 ± 0.76
ASAP	76.87 ± 0.7	74.19 ± 0.79	71.48 ± 0.42	70.07 ± 0.55	$\mathbf{66.26 \pm 0.47}$
EdgePool	70.37 ± 3.81	73.67 ± 4.18	73.65 ± 2.20	70.66 ± 2.01	65.28 ± 1.88
GSAPool	74.95 ± 4.51	73.22 ± 3.37	72.21 ± 1.35	70.11 ± 2.39	60.99 ± 2.08
MinCUT-Pool	77.50 ± 4.89	74.85 ± 3.72	75.38 ± 1.46	73.73 ± 1.56	63.57 ± 2.26
HGP-SL	76.66 ± 3.07	74.03 ± 3.21	74.59 ± 1.46	72.09 ± 2.06	62.81 ± 1.36
GMT	77.75 ± 3.48	75.01 ± 2.89	74.57 ± 1.59	73.25 ± 2.63	64.35 ± 1.41
NIAPool (ours)	$\mathbf{79.28 \pm 4.30}$	$\mathbf{75.25 \pm 3.71}$	$\mathbf{78.08 \pm 1.92}$	$\mathbf{75.56 \pm 2.32}$	65.35 ± 2.58

Overall, a general observation we can draw from the results is that our model obtains the highest accuracy on most of the datasets compared with baseline models. In particular, NIAPool achieves approximate 2.70% higher accuracy over the best baseline on the NCI1 dataset and 2.31% on the NCI109 dataset, respectively. This superiority of NIAPool may be attributed to its advanced mechanism for effectively capturing both local and global node information in pool operation. Although ASAP exhibit the best performance among all baseline methods and is even better than ours on the Frankenstein dataset, NIAPool has an average improvement of 3.89% over ASAP on the other four datasets.

Table 3. The evaluation of different pooling ratios based on the NIAPool architecture.

Pooling ratio	D&D	Proteins	NCI1	NCI109	Frankenstein
0.25	77.67 ± 4.13	74.03 ± 2.18	76.35 ± 1.81	74.63 ± 2.38	62.39 ± 3.28
0.50	$\mathbf{79.28 \pm 4.30}$	75.25 ± 3.71	$\mathbf{78.08 \pm 1.92}$	75.56 ± 2.32	$\mathbf{65.35 \pm 2.58}$
0.75	76.05 ± 3.40	$\mathbf{75.73 \pm 4.14}$	77.38 ± 1.17	$\mathbf{75.87 \pm 1.95}$	64.46 ± 2.71

4.3 Q2: Effectiveness of D2T Attention

To show the effectiveness of D2T attention, We replace the D2T attention module in NIAPool with previously proposed S2T, T2T, and M2T attention techniques, respectively. The results are shown in Table 4. We observe that D2T attention achieves better performance than other attentions, which indicates that the proposed D2T attention framework can reasonably select important nodes.

T2T models the membership of a node by generating a query based only on the medoid nodes. S2T attention scores each node for a global task. M2T extends T2T by using a master function to utilize intra-cluster information. There is a disadvantage causing the node importance mutual exclusion problem when these methods are used for local node information enhancement. That is, they pay too much attention to the nodes with high similarity, resulting in the loss of other node information. From the results, we can prove that D2T deal with the above problem well by taking $h_i - h_j$ into consideration.

Table 4. Effectiveness of different attention framework.

Attention module	D&D	Proteins	NCI1	NCI109	Frankenstein
T2T	78.18 ± 4.44	74.71 ± 3.12	77.13 ± 3.23	75.07 ± 2.02	65.18 ± 1.53
S2T	78.35 ± 4.43	74.82 ± 4.02	76.55 ± 1.89	74.11 ± 1.95	65.30 ± 2.73
M2T	79.03 ± 4.40	74.93 ± 3.90	77.10 ± 0.85	74.23 ± 2.24	65.29 ± 2.23
D2T	$\mathbf{79.28 \pm 4.30}$	$\mathbf{75.25 \pm 3.71}$	$\mathbf{78.08 \pm 1.92}$	$\mathbf{75.56 \pm 2.32}$	$\mathbf{65.35 \pm 2.58}$

4.4 Q3: Effectiveness of NFAConv

In this section, we analyze the impact of NFAConv as a fitness scoring function in NIAPool. We use three famous graph convolutional operations, including GCN [12], GAT [27], and GraphSAGE [9] as our baselines. In Table 5, we can see that NFAConv performs significantly better than others. In particular, our method has obvious advantages in datasets NCI1 and NCI109, which achieve approximate 3.24% and 2.67% accuracy improvement over the best baseline, respectively.

GCN can be viewed as a procedure that computes a score for each node followed by a weighted average operation over neighbors and a nonlinearity operation. If some of the nodes get a high score, it may increase the scores of its neighbors indirectly, which biases the pooling operator to select nodes. GraphSAGE directly averages the neighbor features of central nodes, ignoring the feature diversity among nodes. GAT is an example of T2T attention in graphs. GAT utilizes attention coefficient to weight and sum node features but lacking a way to capture specific node difference features. NFAConv addresses the above problems by focusing on capturing node difference features between nodes. The results from Table 5 verify the effectiveness of NFAConv.

Table 5. Effectiveness of different graph convolutional operations as fitness scoring function.

Fitness function	D&D	Proteins	NCI1	NCI109	Frankenstein
GCN	78.89 ± 3.54	74.48 ± 3.66	74.14 ± 1.49	72.21 ± 2.25	62.44 ± 3.06
GraphSAGE	78.61 ± 4.03	74.39 ± 3.38	74.84 ± 2.68	72.89 ± 2.10	64.65 ± 3.84
GAT	77.08 ± 2.67	74.11 ± 3.23	74.44 ± 1.50	71.97 ± 2.14	61.24 ± 3.50
NFAConv	$\mathbf{79.28 \pm 4.30}$	$\mathbf{75.25 \pm 3.71}$	$\mathbf{78.08 \pm 1.92}$	$\mathbf{75.56 \pm 2.32}$	$\mathbf{65.35 \pm 2.58}$

5 Conclusion

In this paper, we introduce NIAPool, a novel graph pooling operator for the graph classification task. NIAPool is aware of the node information from both the local and global aspects of the graph. For the local aspect, we propose a Difference2Token self-attention framework to better capture the membership between each node and its 1-hop neighbors. For the global aspect, we propose NFAConv, a novel GCN variant that focuses on capturing node difference features and uses it to score nodes. We validate the effectiveness of the components of NIAPool empirically. Through extensive experiments, we demonstrate that NIAPool achieves state-of-the-art performance on multiple graph classification datasets.

Acknowledgments. This work was supported by the National Key R&D Program of China (2017YFB0202403) and the Key R&D Program of Sichuan Province, China (2017GZDZX0003, 2020YFG0089, 2020YFG0308, 2020YFG0304).

References

1. Baek, J., Kang, M., Hwang, S.J.: Accurate learning of graph representations with graph multiset pooling. In: International Conference on Learning Representations. OpenReview.net (2021)
2. Bianchi, F.M., Grattarola, D., Alippi, C.: MinCUT pooling in graph neural networks. ArXiv abs/1907.00481 (2019)
3. Borgwardt, K.M., Ong, C.S., Schönauer, S., Vishwanathan, S.V.N., Smola, A., Kriegel, H.P.: Protein function prediction via graph kernels. Bioinformatics **21**(Suppl 1), i47–56 (2005)
4. Diehl, F.: Edge contraction pooling for graph neural networks. CoRR abs/1905.10990 (2019)

5. Dobson, P.D., Doig, A.J.: Distinguishing enzyme structures from non-enzymes without alignments. J. Mol. Biol. **330**(4), 771–83 (2003)
6. Duvenaud, D., et al.: Convolutional networks on graphs for learning molecular fingerprints. In: Advances in Neural Information Processing Systems, pp. 2224–2232 (2015)
7. Gao, H., Ji, S.: Graph U-nets. In: International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 2083–2092. PMLR (2019)
8. Gao, H., Liu, Y., Ji, S.: Topology-aware graph pooling networks. IEEE Trans. Pattern Anal. Mach. Intell. **PP** (2021)
9. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, pp. 1024–1034 (2017)
10. Hu, L., Yang, T., Shi, C., Ji, H., Li, X.: Heterogeneous graph attention networks for semi-supervised short text classification. In: Empirical Methods in Natural Language Processing, pp. 4820–4829. Association for Computational Linguistics (2019)
11. Kipf, T.N., Welling, M.: Variational graph auto-encoders. CoRR abs/1611.07308 (2016)
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations. OpenReview.net (2017)
13. Lee, J., Lee, I., Kang, J.: Self-attention graph pooling. In: Chaudhuri, K., Salakhutdinov, R. (eds.) International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 3734–3743. PMLR (2019)
14. Li, X., et al.: BrainGNN: interpretable brain graph neural network for fMRI analysis. Med. Image Anal. **74**, 102233 (2021)
15. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.S.: Gated graph sequence neural networks. In: International Conference on Learning Representations (2016)
16. Orsini, F., Frasconi, P., Raedt, L.D.: Graph invariant kernels. In: Yang, Q., Wooldridge, M.J. (eds.) International Joint Conference on Artificial Intelligence, pp. 3756–3762. AAAI Press (2015)
17. Ranjan, E., Sanyal, S., Talukdar, P.P.: ASAP: adaptive structure aware pooling for learning hierarchical graph representations. In: AAAI Conference on Artificial Intelligence, pp. 5470–5477. AAAI Press (2020)
18. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 593–607. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_38
19. Shen, T., Zhou, T., Long, G., Jiang, J., Pan, S., Zhang, C.: DiSAN: directional self-attention network for RNN/CNN-free language understanding. In: Conference on Artificial Intelligence, pp. 5446–5455. AAAI Press (2018)
20. Shervashidze, N., Schweitzer, P., van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-Lehman graph kernels. J. Mach. Learn. Res. **12**, 2539–2561 (2011)
21. Shi, J.Y., Huang, H., Zhang, Y.N., Long, Y.X., Yiu, S.: Predicting binary, discrete and continued lncRNA-disease associations via a unified framework based on graph regression. BMC Med. Genom. **10**, 55–64 (2017)
22. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12026–12035 (2019)
23. Shlomi, J., Battaglia, P.W., Vlimant, J.: Graph neural networks in particle physics. Mach. Learn. Sci. Technol. **2**(2), 21001 (2021)

24. Stokes, J.M., et al.: A deep learning approach to antibiotic discovery. Cell **180**, 688–702.e13 (2020)
25. Vashishth, S., Joshi, R., Prayaga, S.S., Bhattacharyya, C., Talukdar, P.P.: RESIDE: improving distantly-supervised neural relation extraction using side information. In: Empirical Methods in Natural Language Processing, pp. 1257–1266. Association for Computational Linguistics (2018)
26. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
27. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. CoRR abs/1710.10903 (2017)
28. Vinyals, O., Bengio, S., Kudlur, M.: Order matters: sequence to sequence for sets. In: International Conference on Learning Representations (2016)
29. Wale, N., Watson, I.A., Karypis, G.: Comparison of descriptor spaces for chemical compound retrieval and classification. Knowl. Inf. Syst. **14**, 347–375 (2006)
30. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations. OpenReview.net (2019)
31. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 5449–5458. PMLR (2018)
32. Yang, H., et al.: Interpretable multimodality embedding of cerebral cortex using attention graph network for identifying bipolar disorder. In: Shen, D., et al. (eds.) MICCAI 2019. LNCS, vol. 11766, pp. 799–807. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32248-9_89
33. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W.L., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In: Advances in Neural Information Processing Systems, pp. 4805–4815 (2018)
34. Zhang, L., et al.: Structure-feature based graph self-adaptive pooling. In: International World Wide Web Conference, pp. 3098–3104. ACM/IW3C2 (2020)
35. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: AAAI Conference on Artificial Intelligence, pp. 4438–4445. AAAI Press (2018)
36. Zhang, Z., et al.: Hierarchical graph pooling with structure learning. CoRR abs/1911.05954 (2019)