TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 01/10 pp653–663
DOI: 10.26599/TST.2021.9010058
Volume 27, Number 4, August 2022

Residual Convolutional Graph Neural Network with Subgraph Attention Pooling

Yutai Duan, Jianming Wang, Haoran Ma, and Yukuan Sun*

Abstract: The pooling operation is used in graph classification tasks to leverage hierarchical structures preserved in data and reduce computational complexity. However, pooling shrinkage discards graph details, and existing pooling methods may lead to the loss of key classification features. In this work, we propose a residual convolutional graph neural network to tackle the problem of key classification features losing. Particularly, our contributions are threefold: (1) Different from existing methods, we propose a new strategy to calculate sorting values and verify their importance for graph classification. Our strategy does not only use features of simple nodes but also their neighbors for the accurate evaluation of its importance. (2) We design a new graph convolutional layer architecture with the residual connection. By feeding discarded features back into the network architecture, we reduce the probability of losing critical features for graph classification. (3) We propose a new method for graph-level representation. The messages for each node are aggregated separately, and then different attention levels are assigned to each node and merged into a graph-level representation to retain structural and critical information for classification. Our experimental results show that our method leads to state-of-the-art results on multiple graph classification benchmarks.

Key words: graph neural network; graph pooling; information loss

1 Introduction

The recent success of deep learning neural networks (e.g., convolutional neural networks (CNNs)[1] and recurrent neural networks (RNNs)[2]) has boosted research on pattern recognition and data mining, which mainly capture hidden patterns of Euclidean data. With an increasing number of applications where data are non-Euclidean and have to be represented in the form of graphs, there is an increasing interest in extending deep learning for graph data and graph neural networks (GNNs)[3, 4].

Wu et al.[5] divided GNNs into four categories: recurrent GNNs, convolutional GNNs, graph autoencoders, and spatial-temporal GNNs. Convolutional GNNs generalize the convolution operation from grid data to graph data. The primary idea is to generate a representation of a node by aggregating its own features and neighbors' features and extract high-level node representations by stacking multiple graph convolutional layers. There are two different convolutional GNN frameworks: node classification and graph classification.

The convolutional GNN frameworks for node classification adopt a nonlinear transformation after each graph convolutional layer, and the final hidden representation of each node aggregates features from

• Yutai Duan is with Information and Communication Engineering Department, Tiangong University, Tianjin 300387, China. E-mail: ytduan398@163.com.
• Jianming Wang is with Computer Science Department, Tiangong University, Tianjin 300387, China. E-mail: wangjianming@tiangong.edu.cn.
• Haoran Ma is with Software Engineering Department, Tiangong University, Tianjin 300387, China. E-mail: mahaoran068@163.com.
• Yukuan Sun is with the Center for Engineering Intership and Training, Tiangong University, Tianjin 300387, China. E-mail: sunyukuan@tiangong.edu.cn.
∗ To whom correspondence should be addressed.
Manuscript received: 2021-04-27; revised: 2021-07-08; accepted: 2021-07-30

a further neighborhood by stacking multiple layers. In graph classification models, a graph convolutional layer is always followed by a pooling layer to coarse a graph into subgraphs, so that node representations on coarsened graphs represent high graph-level representations.

Similar to CNN models, pooling in GNNs is a downsampling strategy to reduce the size of model parameters and solve the computational problem. Nowadays, the mean/max/sum pooling is the most primitive and effective way to implement pooling in GNNs because of the fast calculation of the mean/max/sum values. The values represent the importance of nodes and are used to order all nodes in a graph. Consequently, the nodes at the end of the sorting will be abandoned.

Pooling shrinkage discards graph details and unavoidably loses information. Moreover, existing pooling methods may lead to discarding critical features for classification owing to two reasons: the network may not choose the most representative nodes, and feature extraction methods are lacking.

The pooling layer needs to retain or discard nodes, which demands GNNs to completely learn the topology and features of each node. Global pooling only has one pooling layer, which cannot extract rich substructures. Existing hierarchical methods[6] only use one convolutional layer for learning and then performing a pooling operation. Because of the insufficient expressivity of a graph convolutional layer, the pooling layer may not retain important substructures. An obvious idea is to learn the detailed topology using several graph convolution layers before each pooling layer. However, the stacking of several convolutional layers will lead to over-smoothing problems and GNN degradation. Therefore, we need to design a dedicated GNN to ensure that a network with a pooling layer extracts key substructures and that the GNN is not plagued by the over-smoothing problem and performance degradation[7].

To this end, we propose a novel GNN that contains threefold solutions to the problem: (1) Different from the mean/max/sum pooling, we propose a more sophisticated strategy to calculate sorting values. Our strategy uses not only the features of one node but also those of its neighbors. (2) Inspired by residual neural networks[8], we design a new architecture of the graph convolutional layer with the residual connection. By stacking several convolutional layers, which increase the receptive field of GNNs, we reduce the probability of losing critical features for graph classification to learn the complete structural information and feed abandoned features back into the network architecture multiple times. (3) The global extract approach based readout operation in GNNs is mainly used to generate graph-level representations, which can result in the loss of critical and structural information for graph classification. We propose a new strategy for graph-level representation generation, which separately aggregates each node's information and introduces the attention mechanism to distinguish the different contributions of each node to graph representation and alleviate the loss of critical and structural information.

In this work, we propose a novel GNN called residual convolutional GNN (RCGNN), which alleviates the loss of critical and structural information for graph classification. Our contributions can be summarized as follows:

- We propose a graph residual learning block (GRLB). It contains three graph convolutional network (GCN)[4] layers and a residual connection, which enhances expression and is not plagued by the over-smoothing problem.
- We propose 1-hop subgraph attention pooling (SAPooling), which can retain representative subgraphs.
- We propose a graph representation attention summarizing (GRAS) module, a new-generation strategy of graph-level representation that can retain critical and structural information for classification.

The remainder of the paper is structured as follows: Section 2 introduces related works. Section 3 presents the detailed introduction of the RCGNN. Section 4 describes the implementation of the experiment and the results of the experiments. Section 5 discusses the conclusions of the study.

2 Related Work

2.1 Graph convolution

In recent years, generalizing the convolution operator to graph data has become an interesting topic. The approaches are divided into spectral and non-spectral domains. Spectral approaches utilize spectral filters by redefining the Fourier transform on graphs and applying the semi-supervised node classification task. Non-spectral approaches, such as GraphSAGE[9], aggregate messages through the structure of the graph. Researchers also have introduced attention mechanisms[10] to convolution operations. Noteworthily, the essence of the above methods is message passing between neighbor nodes.

Kipf and Welling[4] proposed a GNN containing two graph convolutional layers and found that performance declines with the increasing depth of the network. This phenomenon is called the over-smoothing phenomenon, where the features of nodes will become difficult to distinguish as the number of graph convolutional layers increases. Some researchers have attempted to find solutions for this problem and found that increasing the network depth through effective structures or approaches would improve the performance of the model[11, 12].

2.2 Graph pooling methods

A graph convolution operation can also be used for graph classification tasks by the addition of a feature matrix. However, the approach fails to aggregate the multi-scale information of a graph, which results in obtaining flat features. The approach also lacks the capability of capturing important substructures in a graph, which is a crucial basis for classification. For example, functional groups in compounds and important small groups in social networks are all important features of identifying a graph.

Graph pooling is another crucial step of a GNN for downsampling in graph classification. Pooling layers can characterize and collect the multi-scale information of graphs. Pooling layers can also rationally change the structures of graphs and learn important local structures and features. Pooling approaches are divided into structure-based or feature-based methods.

Researchers have proposed many pooling methods from different aspects. Structure-based approaches[13] output a coarsened graph via clustering through the convolutional layer. Feature-based approaches[6, 14, 15] leverage the node features to give a score for each node and then remove a part of the nodes based on the scores. Both approaches use the same architecture and approaches of feature computation based on global pooling.

3 Proposed Framework

Problem statement: First, we use $G(V, E)$ to represent a complete graph, which consists of vertex set V with N nodes and set of edges E. The structural information of a graph can be represented by an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ and feature matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$, where d is the dimension of node features in datasets. A set of graphs can be represented as $\{G_i\}$, which are associated with a set of labels $\{y_i\}$. The graph classification task ains to predict the corresponding label of the input graph. We aim to design a novel GNN to break the limitations of previous works, which can alleviate the loss of structural and critical information for graph classification and degradation of GNN.

3.1 Overview of the RCGNN

In this paper, a GCN[4], an effective approach for node-level representation learning, is set as the basis of our work. First, a GCN model was designed for semi-supervised node classification. In the GCN model, the forward model can be written as

$$\boldsymbol{X}^l = \sigma(\tilde{\boldsymbol{D}}^{-\frac{1}{2}} \times \tilde{\boldsymbol{A}} \times \tilde{\boldsymbol{D}}^{-\frac{1}{2}} \times \boldsymbol{X}^{l-1} \times \boldsymbol{W}^{l-1}) \quad (1)$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ for the self-loop, $\tilde{\boldsymbol{D}} \in \mathbb{R}^{N \times N}$ is the degree matrix of the adjacency matrix $\tilde{\boldsymbol{A}}$. $\boldsymbol{X}$ is the feature matrix. σ is the activation function, which represents ReLU. The output of the GCN is a new feature matrix. Each row of the new feature matrix corresponds to the feature of the node.

Figure 1 shows an illustrative example of the RCGNN. When a graph enters, the high-level features are extracted using the GRLB. Then, a new graph is obtained by removing part of the nodes and maintaining links through 1-hop SAPooling layers. The RCGNN contains nine GCN layers, and every graph will be pooled three times. After pooling the layers, three subgraphs will be sent to the GRAS module to generate a graph-level representation for classification. Finally, a multilayer perceptron with two full connection layers will be employed for prediction.

3.2 GRLB

To preserve identifiable subgraphs in pooling layers, GNNs need to completely learn graph information. For learning the topology and features of each node, we replace one graph convolutional layer with three graph convolutional layers. Meanwhile, inspired by He et al.[8], we added residual connections between the convolutional layers to fix the over-smoothing problem caused by the stacking graph convolutional layers. The GRLB is shown in Fig. 2, and the forward model can be written as

$$\boldsymbol{Z}^{l+1} = \sigma(\boldsymbol{X}_3^l + \sigma(\boldsymbol{X}_1^l)),$$
$$\boldsymbol{X}_i^l = \sigma(\tilde{\boldsymbol{D}}^{-\frac{1}{2}} \times \tilde{\boldsymbol{A}} \times \tilde{\boldsymbol{D}}^{-\frac{1}{2}} \times \boldsymbol{X}_{i-1}^l \times \boldsymbol{W}_{i-1}^l) \quad (2)$$

where $\boldsymbol{Z}^{l+1}$ is the output of the $(\boldsymbol{l}+1)$-th GRLB and $\boldsymbol{X}_i^l$ is the output of the i-th GCN layer of the l-th GRLB. $\boldsymbol{W}_i^l \in \mathbb{R}^{F \times F}$ are learnable parameters. F is the hidden dimension of node features. To highlight the features

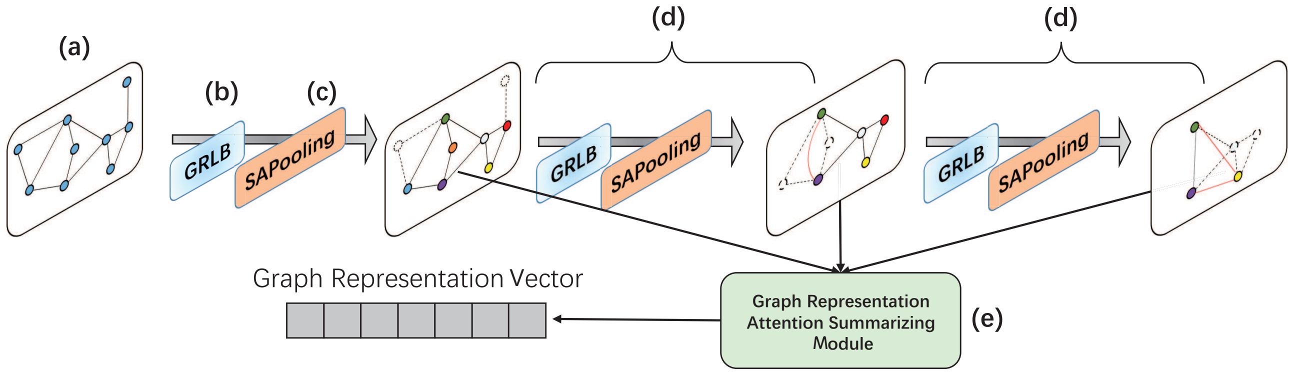
(a)
(b)
(c)
(d)
(d)
GRLB
SAPooling
GRLB
SAPooling
GRLB
SAPooling
Graph Representation Vector
Graph Representation Attention Summarizing Module
(e)

Fig. 1 Overview of the RCGNN: (a) Input a graph to RCGNN. (b) The high-level features of the graph are extracted by using the GRLB module (refer to Section 3.2). (c) Then, the graph is pooled by the SAPooling (Section 3.3) layer to obtain a subgraph. (d) Steps (b) and (c) are repeated three times. Three subgraphs are obtained. (e) Finally, the features of the three subgraphs are summarized by using the GRAS module (Section 3.4) for computing the graph-level representation.

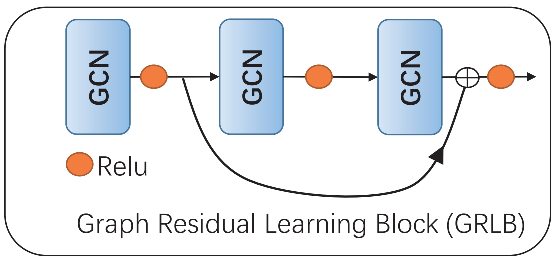

Fig. 2 Illustration of GRLB.

of the current nodes, we set $\tilde{A}$ to $\tilde{A} = A + 2I$, which is a common graph processing technique. The GRLB consists of three graph convolutional layers, and we set the start of the residual connection after the first convolutional layer. The end of the residual connection was installed in the third convolutional layer of the GRLB.

3.3 1-hop SAPooling

Our objective is to make the pooling layer preserve key subgraphs that can represent the whole graph. Therefore, we propose the 1-hop SAPooling layer. We divided each node into a 1-hop subgraph. 1-hop subgraph features are used to replace node features to compute the attention scores (node importance) for each node. Nodes with low attention scores will be discarded.

3.3.1 1-hop subgraph setting

First, we introduce the 1-hop subgraph division strategy. The attention scores of nodes represent the importance of nodes in a graph. If the attention scores of some nodes are very high, then the scores of their neighbors increase after message passing. Thus, the pooling operator selects clusters in the local neighborhood instead of subgraphs that represent the whole graph.

We set the 1-hop subgraph and replaced the features of nodes with the features of the 1-hop subgraphs. For graph $G(V,E)$, the feature matrix is $\boldsymbol{X}$ and its adjacency matrix is $\boldsymbol{A}$. Before sampling by pooling layers, we divided the subgraph $\boldsymbol{v}_i^{sG}$ with each node $\boldsymbol{v}_i$ as the center and 1-hop neighbor as the radius. We can derive the subgraph set $\boldsymbol{V}^{sG}$, where the feature matrix is $\boldsymbol{X}^{sG} \in \mathbb{R}^{N\times 2F}$. $\boldsymbol{x}_i^{sG}$ is defined as

$$\boldsymbol{x}_i^{sG} = \max_{j\in\mathcal{N}(v_i)} (\boldsymbol{x}_i - \boldsymbol{x}_j) || \frac{1}{|j \in \mathcal{N}(v_i)|} \sum_{j\in\mathcal{N}(i)} (\boldsymbol{x}_i - \boldsymbol{x}_j) \tag{3}$$

where $\boldsymbol{x}_i$ represents the i-th node feature vector in $G(V,E)$, $\mathcal{N}(v_i)$ represents the 1-hop neighbor of the node, and $||$ is the concatenation operator, as shown in Fig. 3.

The function of Eq. (3) is to aid pooling layers in measuring the importance of nodes, where the 1-hop subgraph features are extracted for each node. Equation (3) extracts the 1-hop subgraph features with the current node as the center and 1-hop neighbor as the radius through a non-learning way. We used the 1-hop subgraph features instead of single-node features to compute attention scores for node selection. The left half of Eq. (3) represents the relative value of the central node and the first-order neighbor feature. The right half of Eq. (3) represents the mean values of all nodes in the subgraph.

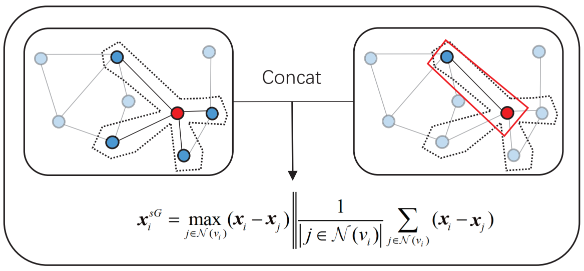
Concat
$x_i^{sG} = \max_{j\in\mathcal{N}(v_i)} (x_i - x_j) \| \frac{1}{|j\in\mathcal{N}(v_i)|} \sum_{j\in\mathcal{N}(v_i)} (x_i - x_j)$

Fig. 3 Illustration of the 1-hop subgraph setting.

Particularly, the feature matrix $\boldsymbol{X}^{sG}$ is only used in the attention computing of pooling layers and classification feature computing (Section 3.4).

3.3.2 Node selection strategy

To verify the effectiveness of the 1-hop subgraph setting, we conducted experiments on two feature-based pooling methods. We chose SAGPool[8] and ASAP[15] as the base models to conduct the experiments. First, we present a new self-attention calculation method based on ASAP and SAGPool:

$$\phi_i = sigmod(\boldsymbol{x}_i^{sG}\boldsymbol{W}_1 + \sum_{j\in\mathcal{N}} \boldsymbol{A}_{i.j}(\boldsymbol{x}_i^{sG}\boldsymbol{W}_2 - \boldsymbol{x}_j^{sG}\boldsymbol{W}_3)) \tag{4}$$

$$\phi_i^s = \sigma(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{X}^{sG}\boldsymbol{W}_{att}) \tag{5}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{2F\times 1}$, $\boldsymbol{W}_2 \in \mathbb{R}^{2F\times 1}$, $\boldsymbol{W}_3 \in \mathbb{R}^{2F\times 1}$, and $\boldsymbol{W}_{att} \in \mathbb{R}^{2F\times 1}$ are learnable. The elements ϕ_i in $\boldsymbol{\phi} \in \mathbb{R}^{N\times 1}$ and ϕ_i^s in $\boldsymbol{\phi}^s \in \mathbb{R}^{N\times 1}$ are scalars that represent the attention scores for nodes.

Equations (4) and (5) use the methods in SAGPool[8] and ASAP[15] to calculate the attention scores of nodes, respectively. Different from ASAP[15] and SAGPool[8], we replaced the information of the single-node feature $\boldsymbol{x}_i$ with the 1-hop subgraph feature $\boldsymbol{x}_i^{sG}$. $\boldsymbol{x}_i^{sG}$ is calculated by Eq. (3) to verify the effectiveness of the 1-hop subgraph setting. We provide the corresponding experiments in Section 4. In this work, Eq. (4) is used to compute attention scores in the 1-hop SAPooling layer.

After obtaining $\boldsymbol{\phi} \in \mathbb{R}^{N\times 1}$, we adopted the strategy of Gao and Ji[14]. $\lceil kN \rceil$ nodes in the graph were preserved based on the pooling ratio $k \in (0,1)$, which is a hyperparameter. The index of the preserved nodes is defined as

$$\hat{i} = TOPK(\boldsymbol{\phi}, \lceil kN \rceil) \tag{6}$$

where $TOPK(\boldsymbol{\phi}, \lceil kN \rceil)$ is a function that ranks the scores and selects the index of the preserved nodes. Following Ying et al.[13], we constructed the distribution matrix $\boldsymbol{S} \in \mathbb{R}^{N\times N}$, which reflects the affiliation between the subgraphs and nodes. The new graph is constructed as

$$\hat{\boldsymbol{S}} = \boldsymbol{S}(:,\hat{i}), \quad \hat{\boldsymbol{X}} = \boldsymbol{X}(\hat{i},:) \tag{7}$$

$$\hat{\boldsymbol{A}} = \hat{\boldsymbol{S}}^{\mathrm{T}}\tilde{\boldsymbol{A}}\hat{\boldsymbol{S}} \tag{8}$$

Equation (7) represents the distribution matrix and feature matrix that perform row or column extractions for reconstructing links between nodes. Then, the new graph is constructed by Eq. (8), which any two nodes, having common neighbors removed by pooling layers, will be connected. The attention scores were multiplied by the feature matrix as a proper scaling for the feature matrix $\boldsymbol{X}_{out}$. Finally, the new feature matrix is defined as

$$\boldsymbol{X}_{out} = \hat{\boldsymbol{X}} \odot \boldsymbol{\phi} \tag{9}$$

where $\odot$ is the broadcasted Hadamard product.

3.4 GRAS module

In graph classification tasks, it is a common operation to extract graph features using max and mean aggregators. The RCGNN has nine GCN layers, which is more than that in other networks with graph pooling layers. However, as the number of GCN layers increases, the features of nodes tend to be the same. Although a GRLB is used to alleviate this problem, the max/mean/sum aggregators may not be able to extract distinguished graph representations, resulting in a decline in the model performance.

Briefly, a GCN is a message-passing model, where node representations become similar as the number of network layers increases[16]. On this basis, we make the following conjecture:

Conjecture 1 *The over-smoothing problem will result in indistinguishable nodes, which will further lead to the degradation of model performance in graph classification tasks.*

We will verify Conjecture 1 on real-world datasets through our experiments (in Section 4).

Accordingly, we need to ensure that the extracted graph representations are distinguishable. To this end, we propose the GRAS module, which is shown in Fig. 4. Different nodes remain in the network for different time because some nodes are discarded when they pass through different pooling layers. GRAS follows the idea that nodes with a long retention time will have a fusion of multiple expressions.

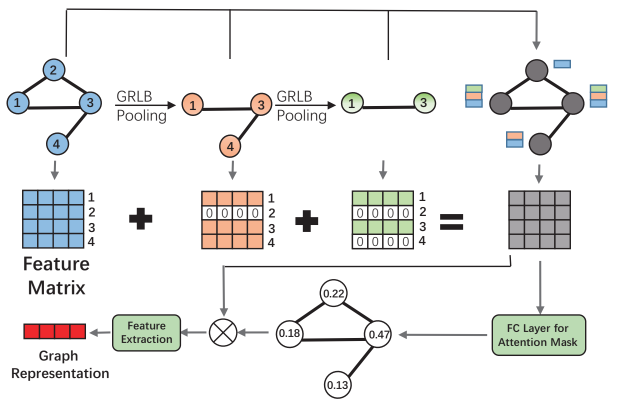

Fig. 4 Illustration of the GRAS module.

For the graph $G(V, E)$, assuming that there are M pooling layers in the network, we denote the m-th pooled graph as G_m^P, where the number of nodes is N_m^P and the feature matrix is $\boldsymbol{X}_m^P \in \mathbb{R}^{N_m^P \times d}$. The 1-hop subgraph feature matrix for G_m^P is $\boldsymbol{X}_m^{sG} \in \mathbb{R}^{N_m^P \times 2d}$. $\boldsymbol{x}_{m,i}^{sG}$ is computed as

$$\boldsymbol{x}_{m,i}^{sG} = \max_{j \in \mathcal{N}(v_i)} \left(\boldsymbol{x}_i - \boldsymbol{x}_j\right) || \frac{1}{|j \in \mathcal{N}(v_i)|} \sum_{j \in \mathcal{N}(i)} (\boldsymbol{x}_i - \boldsymbol{x}_j) \tag{10}$$

Naturally, $N_1^P = \max\{N_m^P \,|m = 1, 2, \ldots\}$. For graph G_m^P, the index of preserved nodes is denoted as idx_m^P. We generated the zero matrix $0 \in \mathbb{R}^{N_1^P \times 2d}$, and constructed medium matrices $\boldsymbol{O}_m^{sG}$ for each pooled graph G_m^P. The matrix $\boldsymbol{O}_m^{sG}$ and hierarchical information aggregation matrix $\boldsymbol{X}^A \in \mathbb{R}^{N_1^P \times 2d}$ are contrasted in the following manner:

$$\begin{gathered}\boldsymbol{O}_m^{sG} = \text{distribute}(0^{N_1^P \times 2d}, \boldsymbol{X}_m^{sG}, idx_m^P),\ i = 1, 2, \ldots, \\ \boldsymbol{X}^A = \sum_{m=1}^{M} \boldsymbol{O}_m^{sG}\end{gathered} \tag{11}$$

where distribute$(0^{N_1^P \times d}, \boldsymbol{X}_m^{sG}, idx_m^P)$ is an operator that distributes row vectors in $\boldsymbol{X}_m^{sG}$ to $0^{N_1^P \times 2d}$. The purpose of Eq. (11) is shown in Fig. 4. If the node is removed by the pooling layer, then the features corresponding to the node will be set to 0 vector. In addition, Eq. (11) adds several features matrices, which makes nodes with longer retention time have more features. $\boldsymbol{x}_i^A = \sum_m \boldsymbol{x}_{m,i}^{sG}$, which represents that each 1-hop subgraph feature is summarized separately. We pay attention to each subgraph through the full connection layer and compute the classification vector as follows:

$$\begin{aligned}\tilde{\boldsymbol{s}} &= \text{softmax}(F_{attn}(\boldsymbol{X}^A)), \\ \boldsymbol{X}^A &= \boldsymbol{X}^A \odot \tilde{\boldsymbol{s}}, \\ \boldsymbol{S} &= \sum_{i=1}^{N_1^P} \boldsymbol{x}_i^A \left\| \max_i^{N_1^P} \boldsymbol{x}_i^A \right.\end{aligned} \tag{12}$$

where the softmax function is evaluated along the column dimension and $F_{attn}(\cdot)$ represents two fully connected layers. The number of neurons in the first layer is F, and the number of neurons in the second layer should be $1/2F$. In Eq. (12), $\tilde{\boldsymbol{s}}$ is the attention value of each node before generating a graph-level representation. If one node is removed after the first pooling layer, then there is no information of that node in the matrix $\boldsymbol{X}^A$. Finally, we send the graph-level representation $\boldsymbol{S} \in \mathbb{R}^{1 \times 4F}$ to two full connection layers for classification to obtain the final prediction.

4 Experiment

We first verified the effectiveness of our proposed 1-hop subgraph setting. We also conducted experiments to verify Conjecture 1. Then we evaluated the performance of the proposed framework RCGNN on a graph classification task. The RCGNN was compared with several global and hierarchical pooling methods. The details of the datasets used for evaluation and the baseline are introduced below. On this basis, we also conducted multiple ablation experiments to verify the effectiveness of each part of the RCGNN. We conducted all experiments based on PyTorch Geometric[17]. We evaluated each dataset using 10-fold cross-validation and calculated the average accuracy of 10 times cross-validation in all experiments. We randomly split each dataset: 80% as the training set, 10% as the validation set, and the remaining 10% as the test set.

4.1 Dataset

We evaluated our approach on five datasets that contained large amounts of graph data. PROTEINS[18, 19] and DD[20] represent proteins, and the nodes are amino acid units. If the distance between two amino acid units is less than a certain threshold, then an edge is produced. The label indicates whether protein is a non-enzyme. NCI1[21] is a biological dataset for screening for anticancer activity classification, and NCI109 is targeted at ovarian cancer cells. Each graph in the dataset is a compound, with nodes representing courtyards and edges representing bonds. FRANKENSTEIN is a series of molecular diagrams with high dimensional node features. The label indicates whether or not it is a mutagen. The dataset statistics are shown in Table 1.

4.2 Baseline

The baseline we chose is as follows: The RCGNN was compared to GCN[4] and GraphSAGE[9]. We compared the RCGNN to global pooling approaches including SET2SET[22], Sortpool[23] and SAGPool[8] (global

Table 1 Statistics of datasets.

Dataset	G_{num}	C_{num}	N_{avg}	E_{avg}
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

Note: G_{num} and C_{num} represent the number of graphs and classes, respectively. N_{avg} and E_{avg} represent the average number of nodes and edges, respectively.

architecture and hierarchical architecture). We compared the RCGNN to hierarchical pooling approaches including Topk[14], ASAP[15], and PANPool[24].

4.3 Experiment implementation

4.3.1 Verification of the 1-Hop subgraph setting

We first conducted experiments to verify the effectiveness of the 1-hop subgraph setting. We used SAGPool and ASAP as the base models. As shown in Eqs. (4) and (5), we replaced the feature matrix $\boldsymbol{X}$ with our subgraph feature matrix $\boldsymbol{X}^{sG}$ when computing attention scores. We followed the network structure in Ref. [8]. They both have the same network structure. We set the batch size to 128, learning rate to 0.01, weight decay to 1×10^{-5}, pooling ratio to $k = 0.5$, and the hidden dimension of node features F to 64. The experimental results are shown in Table 2, with an average increase of 0.77%. Because we only changed the feature matrix, the 1-hop subgraph setting allowed the pooling layer to retain discernible subgraphs. In the remaining experiments, we used Eq. (4) to compute the attention scores in the pooling layers.

4.3.2 Verification of Conjecture 1

In this section, we verify Conjecture 1 by examining how the graph classification accuracy varies with the layer number in the graph classification task on the NCI1 and NCI109 datasets. We set the number of GCN layers to 2, 4, 6, and 8. The hyperparameters are set as follows: batch size = 128, learning rate = 0.01, weight decay = 1×10^{-5}, and feature hidden dimension $F = 64$. The results are shown in Table 3.

From Table 3, we have the following observations: As we extend the network depth, the accuracy of graph classification rapidly drops. We consider that over-smoothing leads to indistinguishable node representation, which further results in indistinguishable graph-level representation extracted from the feature matrix.

4.3.3 Experiments on graph classification dataset

Baselines and the RCGNN uniformly use the learning rate 1×10^{-3} on the DD and FRANKENSTEIN datasets and 1×10^{-2} on the NCI1, NCI109, and PROTEINS datasets. This is because the nodes in DD and FRANKENSTEIN have high dimensional feature vectors. In the experiment, the pooling ratio was selected from $k \in [0.1, 0.9]$ and the hidden dimension of node features in GCN was selected from 16, 32, 64, and 128. Every approach trained 150 epochs in each dataset. Because the authors[24] did not publish the code, we cited the results in the paper, as shown in Table 4.

4.3.4 Ablation study

To verify the effectiveness of the GRLB, 1-hop subgraph setting, and GRAS module in the network, we ablated them separately. For the GRLB, we replaced it with a single layer of the GCN. Furthermore, we ablated

Table 2 Effectiveness verification of the 1-hop subgraph setting in different base models. 1-Hop represents that the 1-hop subgraph setting will be used in base models.

(%)

1-Hop setting	NCI1	NCI109
SAGPool	70.71	69.95
SAGPool + 1-Hop	**71.87**	**70.67**
ASAP	71.14	69.28
ASAP + 1-Hop	**72.33**	**70.28**

Table 3 Verification of Conjecture 1. Results of the graph classification task using different numbers of graph convolutional layers.

(%)

Dataset	Number of GCN layers			
	2	4	6	8
NCI1	70.44	71.12	57.91	50.07
NCI109	70.12	65.56	52.45	50.37

Table 4 Graph classification results on the validation/test sets (best viewed in bold). All results are averaged over ten different runs. "–" represents that the method does not show results on this dataset.

(%)

Dataset	D&D	PROTEINS	NCI1	NCI109	FRANKENSTEIN
GCN	74.89 / 72.25	76.17 / 72.51	73.64 / 71.67	71.93 / 69.56	64.42 / 61.59
GRAPHSAGE	74.30 / 71.19	74.32 / 70.47	77.35 / 74.52	74.63 / 71.74	66.26 / 64.05
SORT-POOL	73.95 / 71.87	75.83 / 72.73	76.75 / 72.29	74.62 / 70.73	69.54 / 65.04
SET2SET	76.48 / 72.41	74.77 / 71.62	75.79 / 71.99	75.90 / 71.90	64.38 / 62.02
SAGPool(G)	78.23 / 76.45	78.34 / 73.91	78.34 / 73.91	75.33 / 73.35	68.44 / 65.80
TopK	76.60 / 73.68	77.04 / 73.86	77.01 / 74.12	72.36 / 68.96	62.89 / 58.84
SAGPool(H)	79.92 / 76.10	78.72 / 72.81	77.25 / 71.63	76.24 / 70.32	67.02 / 62.81
ASAP	79.86 / 76.78	77.09 / 73.79	74.64 / 71.43	73.62 / 70.08	69.03 / 66.36
PANGCN + PANPool	– / –	– / 72.65	– / 68.98	– / –	– / –
RCGNN(Ours)	**81.67 / 77.65**	**80.07 / 75.09**	**82.20 / 80.45**	**81.00 / 78.67**	**70.51 / 68.11**

the residual connection of the GRLB to verify the effectiveness of the residual connection. For the 1-hop subgraph setting, we used the original feature matrix to participate in the calculation. For the GRAS module, we also used the output of the readout layer, similar to Lee et al.[6] The results of the ablation experiments are shown in Tables 5 and 6. We also conducted comparison experiments using a single-layer GCN/3-layer GCN instead of the GRLB. The results of the comparison experiments are shown in Table 7.

In addition, we conducted the ablation experiment of the pooling ratio k. We used different pooling ratios for the experiments and reported the results (graph classification accuracy) of our proposed model on the NCI1 dataset. We also compared the results with ASAP. The experimental results are shown in Fig. 5.

4.4 Result

Table 4 shows that the RCGNN can achieve top performance on a variety of graph classification datasets. Tables 5 and 6 report the results of the ablation experiments. The experimental results show that the RCGNN's performance degrades if it lacks the GRLB, GRAS, or 1-hop subgraph settings, but it still achieves top performance on most datasets. This finding indicates that each part of the RCGNN achieves the purpose of retaining critical or structural information for classification.

The results in Fig. 5 show that the RCGNN

Table 5 Ablation experiments to verify the effectiveness of each module in RCGNN. FRANKEN represents the FRANKENSTEIN dataset. (%)

GRLB	1-HOP	GRAS	NCI1	NCI109	FRANKEN
✓		✓	75.17	75.07	63.61
✓	✓		76.82	75.22	66.60
	✓	✓	79.46	77.62	67.30
✓	✓	✓	**80.45**	**78.67**	**68.11**

Table 6 Ablation study (graph classification results) of the GRLB's residual connection. (%)

GRLB	NCI1	NCI109	FRANKEN
No Residual	79.32	77.95	67.34
Residual	**80.45**	**78.67**	**68.11**

Table 7 Comparison experiments of using a single-layer GCN/3-layer GCN instead of the GRLB. (%)

GRLB	NCI1	NCI109	FRANKEN
Model with a GCN layer	79.46	77.62	67.30
Model with 3-layer GCN	79.32	77.89	67.32
Model with GRLB	**80.45**	**78.67**	**68.11**

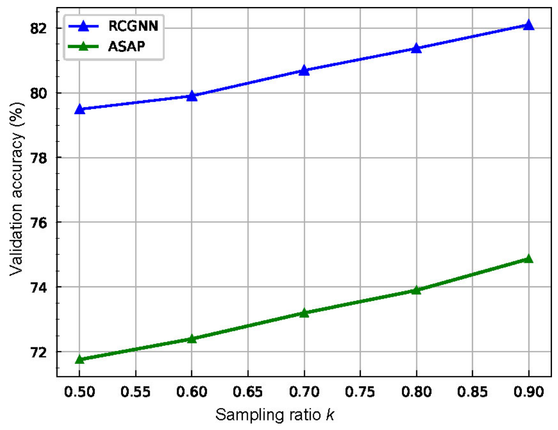

Fig. 5 Validation accuracy vs. sampling ratio k on the NCI1 dataset.

consistently outperforms ASAP. Moreover, a higher k will lead to better performance and more information retention, as shown in Fig. 5. However, for different datasets, the k corresponding to the best performance is different. For example, in the PROTEINS and DD datasets, the performance is the best when $k = 0.8$ and $k = 0.4$.

4.5 Analysis

4.5.1 Effects of the GRLB

Li et al.[11] and Micheli[25] found that the graph convolutional layer can extract complete structural information by stacking local graph convolutional layers. Thus, we replaced a single graph convolutional layer with three graph convolutional layers.

In addition, we set the start of the residual connection after the first convolutional layer because the first convolutional layer has different input and output dimensions. Although the full connection layer is used to change dimensions, as proposed by Chen et al.[26], we consider that it damages the original features of nodes. Furthermore, Li et al.[27] argued that the original information of nodes ultimately determines the node representation. We installed the end of the residual after the third convolution layer. Because stacking the convolutional layer can cause the over-smoothing problem, the early information was fed to the network again, and it was not propagated many times, which alleviated the over-smoothing problem.

In this way, we can expand the receptive field of the network, which can learn complete structural information. The experimental results and the results of the ablation studies indicate that the GRLB is effective. Residual connections are also responsible for model performance. In ablation studies, the average performance was improved by 1%.

The comparison of the experiment results of using the single-layer GCN/3-layer GCN instead of the GRLB is shown in Table 7. The model with the three-layer GCN hardly improved the performance of the model, and there was a performance degradation on the NCI1 dataset. After adding residual connections, the performance was improved. The results reflect the advantages of GRLB over a single-layer GCN and three-layer GCN.

4.5.2 Effects of the 1-hop subgraph setting

In the ablation experiment, we find that using the 1-hop subgraph information will greatly improve the performance of the model.

Hence, it is reasonable to take the subgraph information as a sampling basis. The pooling layer determines whether a node is retained/discarded based on the attention score (node importance). After message passing by the graph convolutional layer, the subgraphs generated by the pooling layers may be composed of nodes with high attention scores and their local neighbors. Thus, the reserved subgraphs are a cluster of high-scoring nodes rather than a representative structure of the graph. To avoid this situation, we chose to use Eq. (3) to evaluate the relative values of a node and its neighbors. Based on the experimental results and ablation studies, this evaluation method significantly improves the performance of the model, which indicates that the 1-hop SAPooling layer can capture more representative graph structures to improve the loss of key information for graph classification.

4.5.3 Effects of graph representation

In the existing method, the features of a pooled graph were extracted by max/mean/sum aggregators. However, this approach does not distinguish nodes removed by different pooling layers. For example, a part of the nodes was removed by the second pooling layer, and nodes were not removed, but these nodes will participate in the final graph representation through global aggregation without any difference.

After passing the GCN layer, node representations tend to be similar. The phenomenon that can lead to graph representations cannot be distinguished by the Weisfeiler-Lehman (WL) algorithm. The reason is that the upper bound of the GCN is the WL algorithm, which was proven by Xu et al.[28] Thus, the essence of distinguishing graphs is to make different nodes have different representations. Essentially, the GRAS module summarizes different amounts of information for different nodes. Then, it learns which nodes have a great impact on the graph representation and attaches attention to them. This process leads to differentiated representations of different types of nodes, which can distinguish the graphs.

In GRAS, $x_i^A = \sum_m x_{m,i}^{sG}$. Thus, a large M indicates that the nodes remain in the graph for a long time, which will be represented by different-level information. In the next step, the GRAS assigns attention to the 1-hop subgraph information of each node, so it is possible to essentially distinguish different contributions of nodes to the graph-level representation. Accordingly, the model can retain complete graph structure information to avoid the loss of critical information for classification. In ablation studies, the average performance was improved by 2.86%.

4.5.4 Complexity analysis

Because GRLB is made up of GCN layers, its complexity is the same as the complexity of GCN. The 1-hop subgraph setting is a non-learning feature extraction strategy, which is calculated by Eq. (3) without introducing any new parameters. Therefore, the time complexity of the 1-hop SAPooling layer depends on which pooling layer is combined with the 1-hop subgraph setting. In this work, the time complexity of a single 1-hop SAPooling layer is $\mathcal{O}(|E|F)$. F is the hidden dimension of node features. Only two fully connected layers in the GRAS module contain learnable weights. The number of neurons in the first layer is F, and the number of neurons in the second layer should be $1/2F$. Thus, the time complexity of GRAS is $\mathcal{O}(NF^2)$.

5 Conclusion

In this paper, we introduce the RCGNN for graph classification. It is intended to alleviate the loss of critical and structural information for graph classification in a previous graph classification framework that was applied to the pooling layer and to alleviate the degradation of GNNs. To this end, we designed a GRLB to learn the complete node representation and introduce different-level information, and then designed the 1-hop SAPooling layer to generate a representative subgraph structure of roots. Finally, a novel graph information summary module was designed to distinguish the different contributions of nodes to the graph-level representation and retained the complete graph structure information to aid in graph classification. The RCGNN achieved state-of-the-art performance on several graph classification datasets, which demonstrated that it retains more effective information and structural information for graph classification. In addition, we conducted ablation experiments on the RCGNN, which proved the effectiveness and interpretability of our modules.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 62072335) and the Tianjin Science and Technology Program (No. 19PTZWHZ00020).

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] T. Mikolov, G. Corrado, K. Chen, and J. Dean, Efficient estimation of word representations in vector space, in *Proc. of the International Conference on Learning Representations*, Scottsdale, AZ, USA, 2013.

[3] J. Atwood and D. Towsley, Diffusion-convolutional neural networks, in *Proc. of the 30th International Conference on Neural Information Processing Systems*, Barcelona, Spain, 2016, pp. 2001–2009.

[4] T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, presented at the 5th International Conference on Learning Representations, Toulon, France, 2017.

[5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, A comprehensive survey on graph neural networks, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[6] J. Lee, I. Lee, and J. Kang, Self-attention graph pooling, in *Proc. the 36th International Conference on Machine Learning*, Long Beach, CA, USA, 2019, pp. 3734–3743.

[7] K. Zhou, Y. Dong, K. Wang, W. S. Lee, B. Hooi, H. Xu, and J. Feng, Understanding and resolving performance degradation in graph convolutional networks, https://arxiv.org/abs/2006.07107, 2021.

[8] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 770–778.

[9] W. L. Hamilton, R. Ying, and J. Leskovec, Inductive representation learning on large graphs, in *Proc. of the 31th NIPS. Conference on Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 1025–1035.

[10] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, Graph attention networks, presented at the 6th International Conference on Learning Representations, Vancouver, Canada, 2018.

[11] G. Li, M. Müller, A. K. Thabet, and B. Ghanem, Inductive DeepGCNs: Can GCNs go as deep as CNNs? in *Proc. of the 2019 IEEE/CVF International Conference on Computer Vision*, Seoul, Republic of Korea, 2017, pp. 9266–9275.

[12] Y. Rong, W. Huang, T. Xu, and J. Huang, DropEdge: Towards deep graph convolutional networks on node classification, presented at the 8th International Conference on Learning Representations, Addis Ababa, Ethiopia, 2020.

[13] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, Hierarchical graph representation learning with differentiable pooling, in *Proc. of the 32th International Conference on Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 4805–4815.

[14] H. Gao and S. Ji, Graph U-nets, in *Proc. of the 36th International Conference on Machine Learning*, Long Beach, CA, USA, 2019, pp. 2083–2092.

[15] E. Ranjan, S. Sanyal, and P. Talukdar, ASAP: Adaptive structure aware pooling for learning hierarchical graph representations, in *Proc. of the 34th AAAI Conference on Artificial Intelligence*, New York, NY, USA, 2020, pp. 5470–5477.

[16] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, Measuring and relieving the over-smoothing problem for graph neural networks from the topological view, in *Proc. 34th AAAI Conference on Artificial Intelligence*, New York, NY, USA, 2020, pp. 3438–3445.

[17] M. Fey and J. E. Lenssen, Fast graph representation learning with PyTorch geometric, presented at the 7th International Conference on Learning Representations, New Orleans, LA, USA, 2019.

[18] P. D. Dobson and A. J. Doig, Distinguishing enzyme structures from non-enzymes without alignments, *Journal of Molecular Biology*, vol. 330, no. 4, pp. 771–783, 2003.

[19] K. M. Abrahams, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H. P. Kriegel, Protein function prediction via graph kernels, *Bioinformatics*, vol. 21, no. 1, pp. 47–56, 2005.

[20] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, Weisfeiler-lehman graph kernels, *The Journal of Machine Learning Research*, vol. 12, no. 3, pp. 2539–2561, 2011.

[21] N. Wale, I. A. Watson, and G. Karypis, Comparison of descriptor spaces for chemical compound retrieval and classification, *Knowledge and Information Systems*, vol. 14, no. 3, pp. 347–375, 2008.

[22] O. Vinyals, S. Bengio, and M. Kudlur, Order matters: Sequence to sequence for sets, presented at the 4th International Conference on Learning Representations, San Juan, Puerto Rico, 2016.

[23] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, An end-to-end deep learning architecture for graph classification, in *Proc. 32th AAAI Conference on Artificial Intelligence*, New Orleans, LA, USA, 2018, pp. 1127–1137.

[24] Z. Ma, J. Xuan, Y. G. Wang, M. Li, and P. Liò, Path integral based convolution and pooling for graph neural networks, presented at the 33th Conference on Neural Information Processing Systems, Vancouver, Canada, 2020.

[25] A. Micheli, Neural network for graphs: A contextual constructive approach, *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[26] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, Simple and deep graph convolutional networks, in *Proc. of the 37th International Conference on Machine Learning*, New Orleans, LA, USA, 2020, pp. 1725–1735.

[27] E. Chien, J. Peng, P. Li, and O. Milenkovic, Adaptive universal generalized PageRank graph neural network, presented at the 8th International Conference on Learning Representations, Addis Ababa, Ethiopia, 2020.

[28] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, How powerful are graph neural networks? presented at the 7th ICLR International Conference on Learning Representations, New Orleans, LA, USA, 2019.

Yutai Duan received the BS degree in electronic information science and technology from Inner Mongolia University of Technology, China in 2019. He is a master student at Information and Communication Engineering Department, Tiangong University. His main research interests are graph machine learning and computer vision.

Jianming Wang received the MS and PhD degrees from Tianjin University, China in 2000 and 2003, respectively. From September 2007 to September 2008, he was a visiting scholar at the Department of Statistics, Carnegie Mellon University, United States. He is now a full professor at Computer Science Department, Tiangong University, China. His research interests are in computer vision, robotics, and electrical tomography.

Haoran Ma received the BE degree in software engineering from Hebei University, China in 2020. He is a master student at Software Engineering Department, Tiangong University. His main research interest is graph machine learning.

Yukuan Sun received the BE and MS degrees from Tiangong University, in 2009 and 2014, respectively. Since 2019, he has been studying for a PhD degree at Ajou University, Republic of Korea. Since 2014, he has been an engineer at Tiangong University. His main research interests are model compression and pruning, distributed deep neural network, Meta-learning, and swarm intelligent robot. He has published more than five papers in journals and international conferences.