Multi-layered Graph Representation Learning with Edge Restructured Pooling

Atsushi Miyamoto[1], Koji Fukuda[1], and Du Yicheng[1]

[1] Center for Exploratory Research, Hitachi, Ltd., Japan
{atsushi.miyamoto.eu,koji.fukuda.jf,yicheng.du.pj}@hitachi.com

Abstract. Recently, there is much research on convolution for graphs, however research on deep learning for graphs (multi-layered network as in CNN on images) has yet to make much progress. To achieve multi-layering for graphs, there is a need for pooling or an operation that more generally reduces the freedom of parameters as they go up the layers. But because graphs have uneven structures, they are difficult to handle explicitly. This paper proposes Edge Restructured Pooling, based on graph topology reconstruction, as a new pooling method for the problem of multi-layering. The key idea of the proposed method is to apply the concept close to the pooling operation on the image to the graph. Graphs cannot be summarized in windows like image pooling but, instead, the proposed method re-redraws the edge in every pooling operation so that the method can explicitly calculate a faraway node every time the layer is superimposed. This enables pseudo-grouping in accordance with pooling size of, such as CNN. Then we redefine the convolution operation or max-pooling for edge restructured pooling to realize multi-layering. The effectiveness of the proposed method is validated by experiments using Cora and Citeseer for node prediction tasks.

Keywords: Graph Convolutional Network, Pooling, Multi-layered.

1 Introduction

A convolutional neural network (CNN) has been a major success in the image processing field and is one of the main factors in the recent deep learning trend. Image processing around 2000 was mainly a combination of a feature vector designed manually, such as SIFT [1], and a classifier, such as SVM [2]. However, a CNN method called AlexNet appeared at the Image Net Large Scale Visual Recognition Challenge (The ILSVRC is an international competition that evaluates the precision of image recognition) held in 2012, and won by a wide margin over conventional approaches [3] [4]. From this point, the methods, such as CNN based on deep learning, have become mainstream in the image processing field. CNN is an extension of the classic multilayer perceptron but it locally limits the connections between neurons based on the knowledge in the brain's visual cortex structure and has a feature that is a loose coupling between layers. Its advantage is that it can easily extract a feature suitable for each task by directly using the pixels of an input image.

CNN mainly consists of convolution layers that extracts local features from the image and pooling layers that summarize the feature locally. The convolution layer has multiple kernels and outputs a feature map by convoluting the kernels for the input. The pooling layer subsamples the feature map output by the convolutional layer; max-pooling is widely used for that method [5] [6]. Each CNN parameter is updated (from a randomly determined initial value) by supervised learning; the error backpropagation method and an optimal parameter can be obtained [7] [8].

However, CNN is a technique developed based on image processing and the input data assumes a regular and a uniform data structure such as a grid. The convolution layer and the pooling layer are a grid-aware aggregation operation and the parameters of convolution filter are also shared and used in all locations. This means they must have a regular and uniform structure and causes problems that make it difficult to handle other data.

In response to this problem, in recent years attention has been focused on CNN for structural data that can be expressed more generally, such as graphs. The graph structure data is diverse and, for example, includes social network data, geospatial data, traffic networks, financial networks, proteins, molecular structure data, etc. The application is expected to be a more generalized method because it can express not only regular structures like grids but also complex relationship information.

A graph is a data structure consisting of a collection of vertices and edges. Early research on graph neural networks (GNN) proposed as recursive neural networks (RNN), which directly handled graphs [9] [10] [11]. However, in recent years, the focus of research has shifted to generalizing the convolution of graphs. They are mainly classified into frequency-based and space-based methods. After Bruna et al. [12] defined the fundamental convolution on a graph in the frequency base, various types of networks have been developed [13]-[27].

However, while there is much research on convolution for graphs, research on deep learning (multi-layered networks as in CNN on images) has only partly begun [26] [27] and has yet to make much progress.

This paper proposes Edge Restructured Pooling, based on graph topology reconstruction, as a new pooling method for the problem of multi-layering, which remains an issue for graph convolutional networks. The key idea of the proposed method is to apply the concept close to the pooling operation on the image to the graph. Graphs cannot be summarized in windows like image pooling but, instead, the proposed method re-redraws the edge in every pooling operation so that the method can explicitly calculate a faraway node every time the layer is superimposed. This enables pseudo-grouping in accordance with pooling size of, such as CNN. Then we redefine the convolution operation or max-pooling for edge restructured pooling and realize a multi-layered graph representation by executing these three operations as a set. The effectiveness of the proposed method is validated by experiments using Cora and Citeseer for node prediction tasks.

2 Previous works on graph convolutional networks

Previous works on graph convolutional networks are mainly divided into frequency-based and spatially-based methods.

Bruna et al. [12] proposed a frequency domain-based graph convolution. In this method, a local convolution filter on the graph is designed by executing spectral clustering after calculating the eigenvalue decomposition of graph Laplacian to learn multiscale features. Defferrard et al. [13] next proposed an approximation method with Chebyshev polynomials to reduce the calculation cost for eigenvalue decomposition of graph Laplacian. Kipf & Welling et al. [14] further simplified the model approximated by the Chebyshev polynomial by limiting the target filter to the nearest (one step) filter. The model for the graph convolutional network (GCN) proposed by Kipf & Welling et al. have the format spatially, where the convolution of adjacent nodes is simply expressed and, as a result, it was merged with the spatially-based method.

A spatially-based method acts on spatially close nodes on a graph and directly defines the convolution. In particular, when it calculates node embedding, it performs a convolution operation that applies a filter (linear conversion) to all adjacent node information, aggregates the information, and then executes a non-linear operation such as ReLU. In addition to GCN [15], which showed the coupling with a frequency base by limiting to adjacency, methods and extensions such as [16], [17], [18], [19] and [20] have been proposed.

Furthermore, in the latest approaches, the graph attention network (GAT) [23], the relational graph convolutional network (RGCN) [24], and edge embedding [17] [25] have been developed. The graph attention network (GAT) [23] makes the relationship have a weight by combining the attention techniques [20] [21] [22] with a method such as GCN. The relational graph convolutional network (RGCN) [24] expanded GCN, enabling to handle edge attributes and directions. The edge embedding can embed the edge attributes. The CNN on these graphs is applied to the graph clustering, semi-supervised node classification [14] [18], the edge prediction [14] and the knowledge graph [24].

While there has been much research on convolution regarding graphs, deep learning (multi-layering) research on graphs is not being done, although some groups [26] [27] have begun. To achieve multi-layering, there is a need for pooling or an operation that more generally reduces the freedom of parameters as they go up the layers. But because graphs, unlike grids, have uneven structures, they are difficult to handle explicitly. If many convolution layers are superimposed without pooling, the number of parameters only increases while the freedom to express (regularization cannot be applied) is not decreasing. As a result, when the method causes over-training when learning the parameters or, in a worst-case scenario, an ill-posed problem occurs, learning may not proceed at all. In addition, depending on the problem to be applied, if it cannot make multilayers, it will be inferred only from only adjacent nodes. If the impact of distant nodes is significant in node prediction, or if layered information is required in class separation, then precision itself cannot be expected. In GCN [14], it is reported that the precision decreases if the layer is deepened. It is believed that a shallow network of about two layers would be good.

Some groups such as Ying et al. [26] have proposed a graph pooling module (DIFFPOOL) that is differentiable and can be used by combining with GCN. DIFFPOOL is a method that learns soft cluster allocation for the node in each layer and accumulates the layers while inputting the data in the next layer based on the cluster allocation and generating the adjacency matrix. This method inputs the data by a matrix (cluster allocation) and degenerates the adjacency matrix. This aspect of the method corresponds to pooling, and has a structure where it can learn another graph neural network to learn the matrix.

Xu et al. [27] has proposed the Jumping Knowledge Network (JK-Net) by focusing, as a problem of multi-layering, on the fact that the reachable range within the same step is different for each node according to the locality (the difference between sparseness and density) in the graph topology. The JK-NET can learn the data by adaptively adjusting the impacted range for each node. By providing a path that can skip the network in each layer, JK-NET determines the final embedding by using an aggregator (Concat/Max-pool/LSTM) that aggregates the information. This is different from the so-called CNN pooling operation, but it can be a significantly critical element if the graph is more uneven with a lot of sparseness and density.

3 Edge Restructure Pooling

The previous chapter described the difficulty of multi-layered graph convolution and introduced DIFFPOOL [26] and JK-Net [27] as an approach to multi-layering that some groups have just started to tackle. To overcome these issues, this paper proposes a pooling method that more directly reconfigures graph topology.

DIFFPOOL learns soft cluster allocation in an additional graph neural network and obtains a matrix that degenerates the number of nodes (number of clusters), but it defines the operation as a differentiable calculation. As a result, it could not input the basic factors that are in CNN. JK-Net cannot consider the pooling operation between the nodes on the graph while it selects Max-pooling between layers in the aggregator. This is because JK-NET inherently has a motivation that is different from pooling operations such as CNN.

This paper discusses a method that more directly artificially enables grouping (in accordance to pooling size, such as CNN) or Max-pooling. The key idea of the proposed method is to adapt the concept (similar to a pooling operation on the image) to the graph. A graph cannot be collected in the window like pooling an image; instead, the proposed method redraws the edge in every pooling operation so that it can explicitly perform calculations as well with distant nodes every time a layer is superimposed.

The following outlines the basics of GCN [14] and RGCN [24] and then considers the key points of pooling in the graph. Then, a new pooling method (a process that redraws an edge) and an extension of GCN, including a process to re-draw an edge, is described. Finally, multi-layering is described.

3.1 Graph Convolutional Network (GCN)

The Graph Convolutional Network (GCN) is expressed by the following equation where the feature vector in the l-th layer is $H^{(l)}$ in the graph $G = (V, E)$ consisting of node $v \in V$ and edge $(v_i, v_j) \in E$.

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

Here, $\tilde{A} = A + I_N$ expresses an adjacency matrix in which the self loop is added. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ and $W^{(l)}$ are the weighted matrix of each layer. $\sigma(\cdot)$ is the activating function such as ReLU$(\cdot)$. $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ shows the regularized target matrix, expressing the weighted sum of the node in the adjacent relationship.

As a result, Equation (1) can be rewritten as follows by substituting ${h_i}^{(l)}$ for the feature vector of node v_i and assuming that the set of nodes adjacent to node v_i is N_i.

$${h_i}^{(l+1)} = \sigma\left(\sum_{j\in\tilde{N}_i}\frac{1}{\sqrt{c_i c_j}}h_j^{(l)}W^{(l)}\right) \tag{2}$$

Assume that the node set N_i adjacent to node v_i is $\{v_j \in V|(v_i, v_j) \in E\}$ and the node set $\tilde{N}_i$ in graph $\tilde{G}$ including the self loop is $\{v_i\} \cup \{v_j \in V|(v_i, v_j) \in E\}$. Here, $\frac{1}{\sqrt{c_i c_j}}$ is the normalized term and c_i expresses the degree of node v_i.

Hamilton et al. [18] also proposed a model very similar to these models by setting the normalized term as the mean aggregator.

$${h_i}^{(l+1)} = \sigma\left(\sum_{j\in\tilde{N}_i}\frac{1}{\tilde{c}_j}h_j^{(l)}W^{(l)}\right) \tag{3}$$

Here, $\tilde{c}_j$ is the degree of node v_i in graph $\tilde{G}$ including the self loop.

The following shows the RGCN, an extended model of GCN, which can handle the digraph and relationship information.

$${h_i}^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{j\in N_i^r}\frac{1}{c_{i,r}}W_r^{(l)}\,h_j^{(l)} + W_0^{(l)}h_0^{(l)}\right) \tag{4}$$

Here, N_i^r is the node set adjacent to Node v_i with the relationship $r \in R$. $c_{i,r}$ is the normalized term expressed by $c_{i,r} = |N_i^r|$. Note that, unlike GCN, weighted matrix $W_r^{(l)}$ is provided not only for 1 channel but also for the number of relationships rs.

Fig. 1 shows a diagram of GCN or RGCN. As shown in the figure, the convolution (GCN or RGCN) on the graph can be defined as a method that assumes the convolution as one hop filter on the graph and calculates the feature vector by taking into account the relationship with the adjacent nodes for each node. For multi-layering, this model is applied repeatedly. However, this method may impose some problems.

For example, it is not easy to interact with a faraway node because only one hop is considered in one layer at a time and the method cannot learn well because only the number of parameters increases because the topology does not change. Actually, in many graph convolutions of the GCN type, the number of layers is about two.

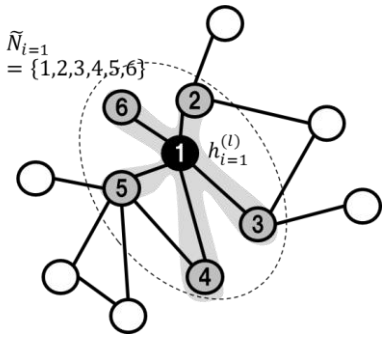


Fig. 1. Diagram for convolution in the GCN of RGCN model.

3.2 Pooling on graph

Fig. 2 shows the pooling operation on a grid (image). This pooling operation defines a small area by grouping the nodes as shown in the figure below and does a subsampling to degenerate the matrix while keeping critical information (features), particularly, in the Max-pooling normally used, the operation to select the maximum area among small areas is done. The most critical effect impacted by this pooling is mainly, (i) selecting the maximum area compresses the information and, (ii) grouping the nodes changes the adjacency relationship in the next layer.

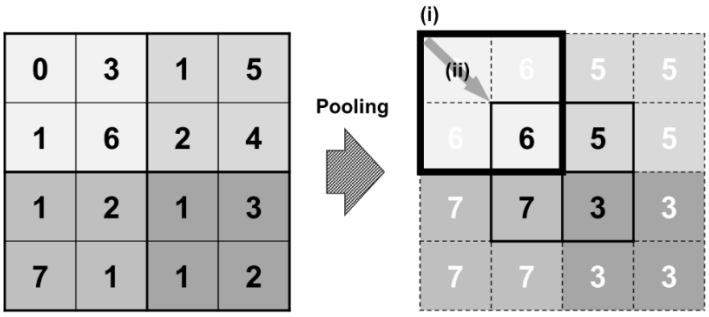
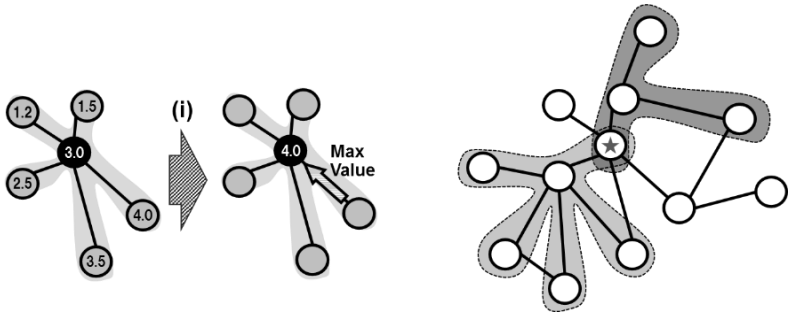


Fig. 2. Example of a pooling operation on an image.

If the pooling operation above needs to be realized on a graph, the (i)-operation only needs to select the maximum area among its own node and adjacent nodes, as shown in Fig. 3 (a), and it can be naturally defined. On the other hand, the (ii)-operation is not easy because it is not an operation on the grid, as shown in Fig. 3 (b). If there is an adjacent node (marked with a star in the figure), which is looked up

commonly by different nodes, how the nodes can be grouped is not yet found, meaning there is no specific method to do so.



(a) (i)-operation (b) (ii)-operation

Fig. 3. Pooling operation on a graph: (a) (i)-operation on a graph, (b) (ii)-operation on a graph.

However, instead of grouping the nodes, if the (ii)-operation can be defined as to change the adjacency relationship in the next layer, then it may be artificially realized by the process redrawing the edges. The proposed method calls this "Edge Restructured Pooling (ERP)". For example, in the *l*-th layer graph where Max-pooling with adjacent node in each node is calculated, the graph topology of *l*+1-th layer is converted to one which connects to the 2 hops ahead, as shown in Fig. 4.

In each layer where the convolution is calculated, executing Max-pooling and Edge Restructured Pooling together can define the direct pooling operation on the graph, aiming to realize multi-layering.

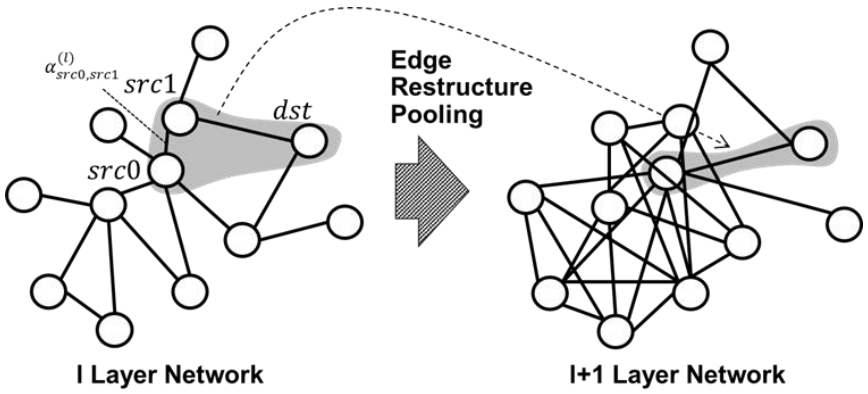


Fig. 4. Overview of Edge Restructured Pooling.

3.3 Proposed method

To realize the idea of proposed Edge Restructured Pooling, the edge coefficient α is newly defined. The edge coefficient α prevents the number of edges from increasing too many when the graph is converted to a 2-hop graph in each layer. For example,

there may be cases where a node that can be a hub is adjacent to the node. If the edge coefficient is not supposed, a tightly-coupled graph may be restructured immediately because the adjacent node of the hub is directly connected in the next layer. So, for the edge coefficient α, the following is taken into consideration: "As the interaction with the faraway node progresses in each layer, its influence decreases". Furthermore, the following is also taken into consideration: "The adjacent node is connected to how many nodes ahead." It is a natural manipulation to take into consideration "decreasing the influence level by superimposing the layers" and "influence the relay node 2-hops ahead" for α.

The flow of 1-layer convolution and pooling by the proposed method consists of the following: (i) Extended GCN, (ii) Graph Max Pooling, and (iii) Edge Restructured Pooling. They are described in the following.

(i) Extended Graph Convolutional Network (GCN)

The extended GCN in the proposed method extends takes the edge coefficient α into consideration. The edge coefficient α is given to each edge, and expressed as $\alpha_{i,j}^{(l)}$ in edge $(v_i, v_j) \in E$ in the *l*-th layer. The extended GCN is expressed as follows by using $\alpha_{i,j}^{(l)}$ where as the feature vector of node v_i is ${h_i}^{(l)}$, node set adjacent to node v_i in $\tilde{G}$ is $\tilde{N}_i$ and the weighted matrix is $W^{(l)}$.

$$ {h_i}^{(l+1)} = \sigma\left(\sum_{j \in \tilde{N}_i} \alpha_{i,j}^{(l)} h_j^{(l)} W^{(l)}\right) \tag{5} $$

This is the equation where the normalized terms in Equation (2) and Equation (3) are replaced by $\alpha_{i,j}^{(l)}$. As the initial value of the edge coefficient α, $\alpha_{i,j}^{(1)} = \frac{1}{\tilde{c}_j}$ is set in the 1st layer. The updated equation of $\alpha_{i,j}^{(l)}$ is described in "(iii) Edge Restructured Pooling". It can be considered that $\alpha_{i,j}^{(l)}$ functions as the attention statistically determined by the topology.

(ii) Graph Max Pooling

Graph Max Pooling executes "Max calculation" between the self node and adjacent node (node set $\tilde{N}_i = \{v_i\} \cup \{v_j \in V | (v_i, v_j) \in E\}$) as shown in the equation below.

$$ h\_pool_i^{(l)} = max\left(\{h_j^{(l)} | j \in \tilde{N}_i\}\right) \tag{6} $$

Instead of taking the maximum value from node set $\tilde{N}_i$, the pooling value may be obtained by sampling the expected value which stochastically has the maximum value by considering $\alpha_{i,j}^{(l)}$.

(iii) Edge Restructured Pooling

As shown in Fig. 5, the proposed method changes the adjacency relationship by redrawing the edges after the convolution process and pooling process. Fig. 5 shows the outline of the process updating the edges. As for the three points - $src0$, $src1$ and dst, if edge (v_{src0}, v_{src1}) and edge (v_{src1}, v_{dst}) exist in the *l*-th layer, as shown in the

figure, edge (v_{src0}, v_{src1}) and edge (v_{src1}, v_{dst}) are deleted in the l+1-th layer, and edge (v_{src1}, v_{dst}) is newly created. The following shows the equation to update the edge coefficient.

$$\alpha_{src0,dst}^{(l+1)} = \begin{cases} \alpha_{src0,src1}^{(l)} \dfrac{\alpha_{src1,dst}^{(l)}}{\sum_{j \in \tilde{N}_{src1}} \alpha_{src1,j}^{(l)}} & (\alpha_{src0,src1}^{(l)} \dfrac{\alpha_{src1,dst}^{(l)}}{\sum_{j \in \tilde{N}_{src1}} \alpha_{src1,j}^{(l)}} \geq \theta) \\ 0 & (\alpha_{src0,src1}^{(l)} \dfrac{\alpha_{src1,dst}^{(l)}}{\sum_{j \in \tilde{N}_{src1}} \alpha_{src1,j}^{(l)}} < \theta) \end{cases} \tag{7}$$

The method compares the edge with the hyper parameter θ, and if the edge has the same value or higher of the hyper parameter θ, the edge coefficient is updated. If it is less than θ, the edge is deleted. In this equation, $src1$ is multiplied by the edge coefficient $\alpha_{src0,src1}^{(l)}$ inflowed from $src0$, and the ratio of $\alpha_{src1,dst}^{(l)}$ in the edge coefficient sum $\sum_{j \in \tilde{N}_{src1}} \alpha_{src1,j}^{(l)}$ is considered. As a result, in the l+1-th layer where the edge as a relay is deleted, the influence of l+1-th layer can be reflected to a certain degree. Additionally, operating the threshold value in the edge coefficient solves the problem of an increased number of edges as the layers are superimposed.

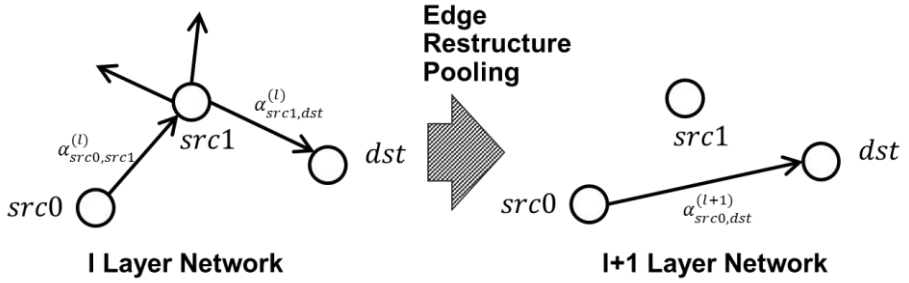


Fig. 5. Update of edge coefficient on Edge Restructured Pooling.

The above described the flow of methods including proposed Edge Restructured Pooling. In this paper, these methods are combined with GCN. In this paper, these methods are combined with GCN, but they can also be combined with RGCN. Although details are not described, in that case, it is necessary to consider edge embedding in convolution.

The proposed method uses Edge Restructured Pooling combining with Graph Max Pooling after GCN or RGCN. Doing so enables direct value degeneration and interactive conversion such as the pooling on the image.

3.4 Multi-layering

These three elements (i) GCN, (ii) Graph Max Pooling, and (iii) Edge Restructured Pooling are superimposed in order and set as one basic layer. These basic layers are set up to multiple layers as shown in Fig. 6. The GCN calculates the convolution.

Then the Graph Max Pooling does the pooling. At the end, the Edge Restructured Pooling converts the interaction. These operations are repeated multiple times. The hyperparameter in each layer is dimension D of $h_i^{(l)}$ and the edge coefficient's threshold value θ. These can be set individually.

In "Learning", a decoder must be separately designed for a particular task such as classification, node prediction, edge prediction, etc., in addition to the encoder shown in Fig. 6. The proposed method includes the pooling process inside the encoder. Therefore in the case of graph classification, it can directly infer the label from the feature vector in the last layer. But, if the method needs to predict a fine-graded object such as node prediction or edge prediction, it needs an "up-sampling" in image processing or a scheme such as JK-Net.

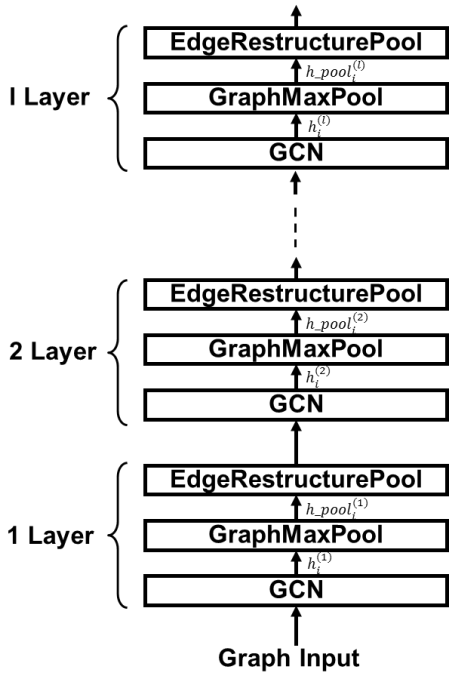


Fig. 6. Proposed deep graph convolutional network with graph pooling.

4 Experiments

4.1 Node prediction

Edge Restructured Pooling is applied to the node prediction task. Fig. 7 shows a 3-layer node prediction model. In this experiment, a decoder similar to JK-Net is considered. This model aggregates the feature vectors in each layer through JK-Net and the Layer Aggregator summarizes them. It linear-converts the information and pre-

dicts the label. "Concat" is used as the Layer Aggregator. "ReLU" is used as the activation function of GCN. In addition, the layer normalization was performed after GCN. This model can be considered to be one in which Graph Max Pooling and Edge Restructured Pooling have been added to the JK-Net model.

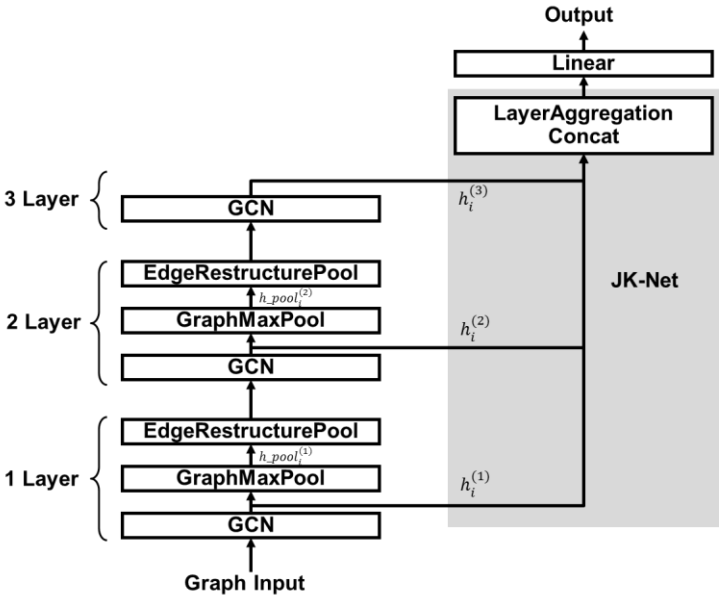


Fig. 7. Model with Edge Restructured Pooling on node prediction.

4.2 Setup

Two benchmark sets (Cora, Citeseer) [28] are used to evaluate Edge Restructured Pooling in the node prediction task. Cora is the network for paper reference relationships; it is a dataset where 2,708 papers about machine learning have been classified into seven categories (e.g., neural-networks, probabilistic-methods, etc.). The input value for each node (paper) is "bag-of-words" and the edge (directed line) indicates the reference relationship of the papers. The label of each node denotes the paper's category and Cora is the task predicting the category. Similar to Cora, Citeseer is also a network for paper reference relationships; it is a dataset where 3,327 papers are classified into six categories. The information that the node, edge or label indicates is the same as Cora. Table 1 describes the outline of the dataset.

In this experiment, the number of layers is changed (2~7) and the result is observed. The edge coefficient's threshold value $\theta^{(l)}$ (i.e., hyper parameter in each layer) is fixed to 0.02 in all layers. For dimension $D^{(l)}$ of the feature vector, 2 types (16 and 32) are tested in all layers. For optimization, Adam is used and its learning rate is 0.005. To prevent over-training, drop out as the model's parameter is applied with probability 0.5 and L2 regularization is applied in 0.0005.

The both dataset uses 80% samples as training data and 20% samples as test data. Because the initial value, etc. of a parameter is given as a random number, the task performs an evaluation 10 times, and evaluates the average classification precision of the node with the standard deviation. For comparison as a baseline, GCN [14] and JK-Net [32] are also re-evaluated with same conditions.

Table 1. Dataset statistics.

Dataset	Nodes	Edges	Classes	Features
Cora	2708	5729	7	1433
Citeseer	3312	4715	6	3703

4.3 Results

Tables 2 and 3 list the experiment results for Cora and Citeseer. The number in the parentheses "( )" next to the model name indicates the dimension number of the feature vector. For the baseline, "GCN" indicates original GCN implementation [14], "JK-Concat" indicates JK-Net implementation [27] which has the Concat aggregator. For proposed method, "GCNERP-Concat" indicates GCN based implementation with Graph Max Pooling and Edge Restructured Pooling, and Concat aggregator in JK-Net decoder is implemented.

In both Cora and Citeseer, it is apparent that GCN has difficulty achieving high precision as the number of layers increases. On the other hand, there is little precision degradation in JK-Net and the proposed method as the number of layers increases. The proposed method superimposes the three elements – (i) GCN, (ii) Graph Max Pooling and (iii) Edge Restructured Pooling, and when it is evaluated with the same conditions, it achieves higher precision compared with the method of JK-Net. This means that "Information compression by Graph Max Pooling for the convolution layer" and "Interaction conversion by Edge Restructured Pooling" have effectively worked for multilayer architecture.

Table 2. Prediction result for Cora dataset.

Model	2 layers	3 layers	4 layers	5 layers
Baseline				
GCN (16)	87.83±0.87	87.75±1.14	87.05±1.50	84.52±2.35
GCN (32)	87.16±1.14	87.66±1.09	87.75±1.24	87.36±1.42
JK-Concat (16)	87.79±0.88	**88.06±1.20**	87.90±1.04	87.86±0.92
JK-Concat (32)	87.21±1.17	87.56±1.21	87.49±1.22	87.51±1.34
Proposed				
GCNERP-Concat (16)	87.78±1.22	87.77±1.12	87.77±1.39	87.82±1.24
GCNERP-Concat (32)	**88.32±1.24**	87.97±1.24	**88.17±1.14**	**87.90±1.25**
Model	**6 layers**	**7 layers**		
Baseline				
GCN (16)	79.08±2.63	75.02±4.42		
GCN (32)	84.89±0.48	83.52±1.69		
JK-Concat (16)	87.84±0.36	87.56±0.47		
JK-Concat (32)	87.71±0.25	87.53±0.31		
Proposed				
GCNERP-Concat (16)	87.90±1.23	87.90±1.23		
GCNERP-Concat (32)	**88.49±1.33**	**88.49±1.50**		

Table 3. Prediction result for Citeseer dataset.

Model	2 layers	3 layers	4 layers	5 layers
Baseline				
GCN (16)	**77.53±1.75**	76.57±1.19	75.04±0.95	71.50±3.31
GCN (32)	77.12±1.88	76.91±1.44	76.64±1.19	75.53±1.51
JK-Concat (16)	77.18±1.83	**77.51±1.84**	77.03±1.69	76.88±1.63
JK-Concat (32)	76.82±1.64	76.94±1.77	77.34±1.40	76.98±1.65
Proposed				
GCNERP-Concat (16)	77.14±1.52	77.12±1.48	**77.39±1.37**	**77.48±1.38**
GCNERP-Concat (32)	77.32±1.50	77.17±1.37	77.22±1.71	77.39±1.51

Model	6 layers	7 layers
Baseline		
GCN (16)	65.61±5.58	61.10±3.92
GCN (32)	73.65±3.19	73.26±1.75
JK-Concat (16)	77.15±1.44	77.38±1.55
JK-Concat (32)	77.07±1.70	77.12±1.58
Proposed		
GCNERP-Concat (16)	77.39±1.48	77.25±1.58
GCNERP-Concat (32)	**77.60±1.84**	**77.50±1.51**

As a result, it has been confirmed that the proposed method can achieve the highest accuracy compared to the conventional method for every number of layers. However, in this experiment, the accuracy do not improve according to the number of layers (the accuracy hardly changes regardless of the layer). This is considered to be because Cora and Citeseer are simple datasets and do not require much interaction with distant nodes. Edge Restructured Pooling is a method for accumulating Max-pooling and convolution operations while updating the interaction with distant nodes, so if it is complex data that needs to be considered also with the influence of distant nodes, the effect is considered to be greater.

In proposed method, the graph in each layer changes while updating the interactive nodes by inserting the Edge Restructured Pooling. Table 2 shows the changes of the number of edges in each layer when the edge coefficient's threshold is set to 0.02.

Table 2. Number of edges on each layer.

Original	2 layers	3 layers	4 layers	5 layers	6 layers	7 layers
5429	8213	6339	2956	1508	1132	841

The number of edges increases once but it significantly decreases as the layers are superimposed. This shows that the influence rate by the edge coefficient is well controlled by manipulating the threshold of $\alpha_{i,j}^{(l)}$. If the edge coefficient's threshold $\theta^{(l)}$ is set high, more edges are generated for the threshold. In this case, another problem may occur that too many numbers of edges significantly increase the calculation amount. In this experiment, the value of the edge coefficient's threshold $\theta^{(l)}$ is set experimentally, but it does not mean that many edges can be left and some level of edge-cut operation may have functioned as regularizing the number of edges.

5 Conclusion

This paper has proposed a new pooling method for graphs, enabling pooling such as CNN in a more direct way, to realize multi-layering of the graph convolutional network.

The proposed Edge Restructured Pooling executes edge redrawing in every layer and restructures the graph topology. Using it with combined Graph Max Pooling, it substituted the interaction conversion function in the CNN's pooling, and solved the problem of pooling in the graph.

In the experiments, the proposed methods were compared and evaluated with the conventional GCN and JK-Net using Cora and Citeseer datasets in nodes prediction tasks. As a result, GCN shows that the precision decreases as the number of layers increases. On the other hand, JK-Net and the proposed method show that precision does not decrease even when the number of layers increases. Furthermore, the proposed method achieves better precision than JK-Net. It is proved that "information compression by Graph Max Pooling for the convolution layer" and "Interaction conversion by Edge Restructured Pooling" works effectively for multilayer architecture.

In the future, we plan to confirm the effect of the proposed method not only on simple graphs but also on complex graphs that require interaction with distant nodes.

References

1. Lowe, D.: Distinctive image features from scale-invariant key-points. International Journal of Computer Vision 60(2), 91-110 (2004).
2. Cortes, C., Vapnik, V.: Support-vector network. Machine Learning 20, 273–297 (1995).
3. Krizhevsky, A., Sutskever, I., Hinton, G. E.: ImageNet classification with deep convolutional neural networks. In: Neural Information Processing Systems, pp. 1106-1114 (2012).
4. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z, Karpathy, A., Khosla, A., Bernstein, M., Berg, B. C., Fei-Fei, F.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 1-42, (2015).
5. Scherer, D., Muller, A., Behnke, S.: Evaluation of pooling operations in convolutional architectures for object recognition. In: International Conference on Artificial Neural Networks (2010).
6. Boureau, Y., Bach, F., LeCun Y., Ponce, J.: Learning mid-level features for recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (2010).
7. Rumelhart, D. E., Hinton, G. E., Williams, R. J.: Learning internal representations by error propagation. In: Parallel Distributed Processing: Explorations in the Microstructures of Cognition 1, 318-362. MIT Press (1986).
8. Duffner, S., Garcia, S.: An online backpropagation algorithm with validation error-based adaptive learning rate. In: International Conference on Artificial Neural Networks 1, 249-258, (2007).
9. Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In: IEEE International Joint Conference on Neural Networks, pp. 729-734 (2005).
10. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks 20(1), 61-80 (2009).

11. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. In: International Conference on Learning Representations (2016).
12. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. In: International Conference on Learning Representations (2014).
13. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems, pp. 3844-3852 (2016).
14. Kipf, T. N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (2017).
15. Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., Adams, R. P.: Convolutional networks on graphs for learning molecular Fingerprints. In: Advances in neural information processing systems, pp. 2224-2232 (2015).
16. Monti, F., Boscaini, D., Masci, J., Rodol`a, E., Svoboda, J., Bronstein, M. M.: Geometric deep learning on graphs and manifolds using mixture model cnns. arXiv preprint arXiv:1611.08402 (2016).
17. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., Dahl, G. E.: Neural message passing for quantum chemistry. In: International Conference on Machine Learning, pp. 1273–1272 (2017).
18. Hamilton, W. L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Neural Information Processing Systems (2017).
19. Chen, J., Zhu, J., Song, L.: Stochastic training of graph convolutional networks with variance reduction. In: Proceedings of International Conference on Machine Learning, pp. 942-950 (2018).
20. Bahdanau, D., Cho K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: International Conference on Learning Representations (2015).
21. Luong, T., Pham H., Manning, C. D.: Effective approaches to attention-based neural machine translation. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, pp. 1412-1421 (2015).
22. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L.,Gomez, A. N., Kaiser L. U., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998-6008 (2017).
23. Veli˘ckovi´c, P., Cucurull, G., Casanova, A., Romero, A., Li`o, P., Bengio, Y.; Graph attention networks. In: International Conference on Learning Representations (2018).
24. Schlichtkrull, M., Kipf, T. N., Bloem, P., vd Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Extended Semantic Web Conference (2018).
25. Kipf, T. N., Fetaya, E., Wang, K., Welling, M., Zemel, R.: Neural relational inference for interacting systems: In: Proceedings of International Conference on Machine Learning (2018).
26. 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.4800-4810 (2018).
27. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: Proceedings of International Conference on Machine Learning, pp. 5449-5458 (2018).
28. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher B., Eliassi-Rad, T.: Collective classification in network data. AI magazine 29(3), 93 (2008).