Hierarchical Graph Convolutional Networks for Semi-supervised Node Classification

Fenyu Hu[1,2*] , **Yanqiao Zhu**[1,2*] , **Shu Wu**[1,2†] , **Liang Wang**[1,2] and **Tieniu Tan**[1,2]
[1] University of Chinese Academy of Sciences
[2] Center for Research on Intelligent Perception and Computing, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
{fenyu.hu,yanqiao.zhu}@cripac.ia.ac.cn, {shu.wu,wangliang,tnt}@nlpr.ia.ac.cn

IJCAI 2019

2019.08.20

Motivation

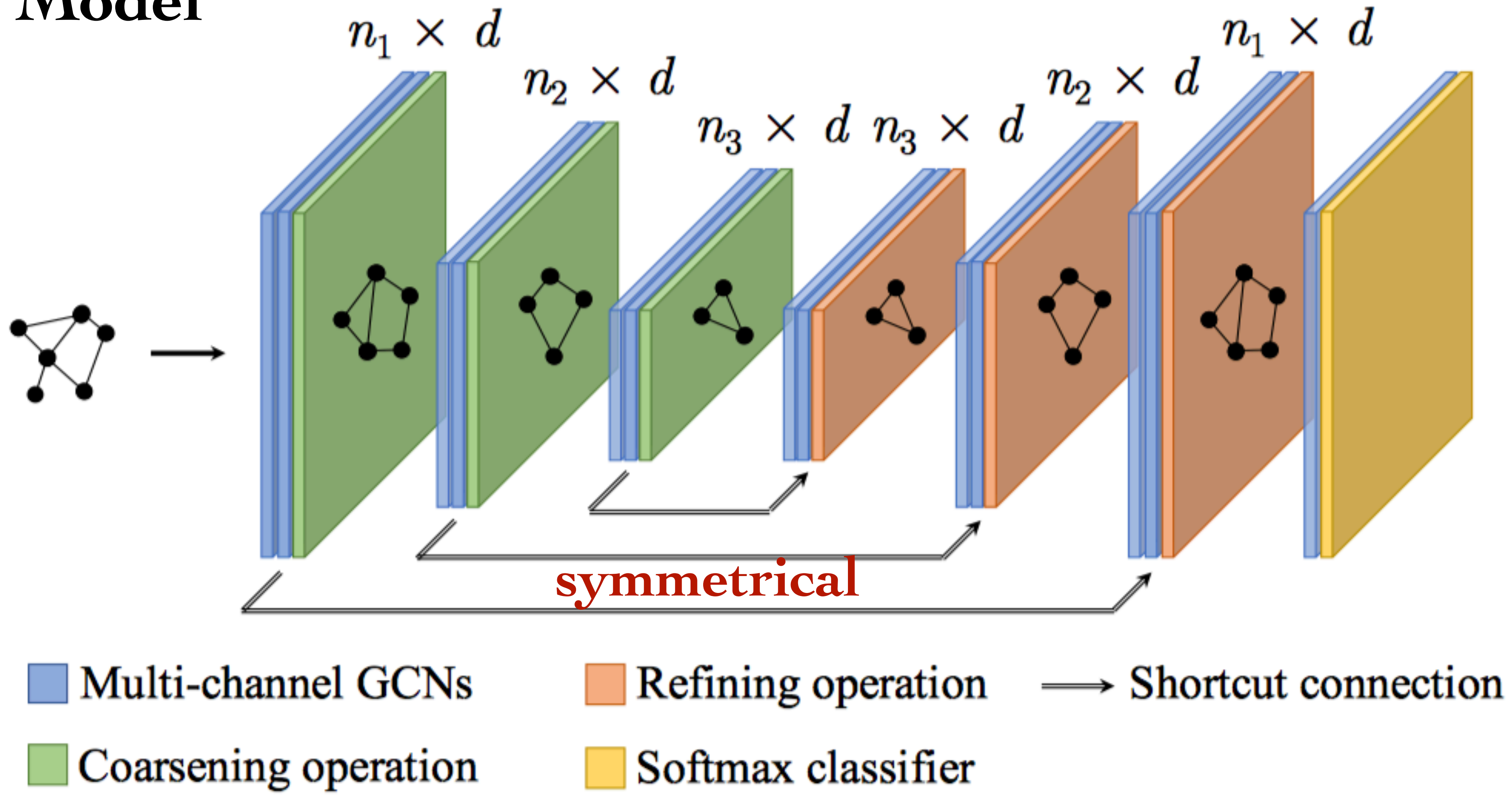
- Most of these models based on neighborhood aggregation are usually **shallow** and **lack the "graph pooling" mechanism**, which prevents the model from obtaining adequate global information.
- In order to **increase the receptive field**, we propose a novel deep Hierarchical Graph Convolutional Network (H-GCN) for semisupervised node classification.

Contributions

The main contributions of this paper are twofold.

- Firstly, to the best of our knowledge, it is the first work to **design a deep hierarchical model** for the semi-supervised node classification task. Compared to previous work, the proposed model consists of **more layers** with **larger receptive fields**, which is able to obtain more global information through the coarsening and refining procedures.
- Secondly, we conduct extensive experiments on a variety of public datasets and show that the proposed method constantly **outperforms other state-of-the-art approaches**. Notably, our model gains a considerable improvement over other approaches with **very few labeled samples** provided for each class.

Model



i^{th} layer:

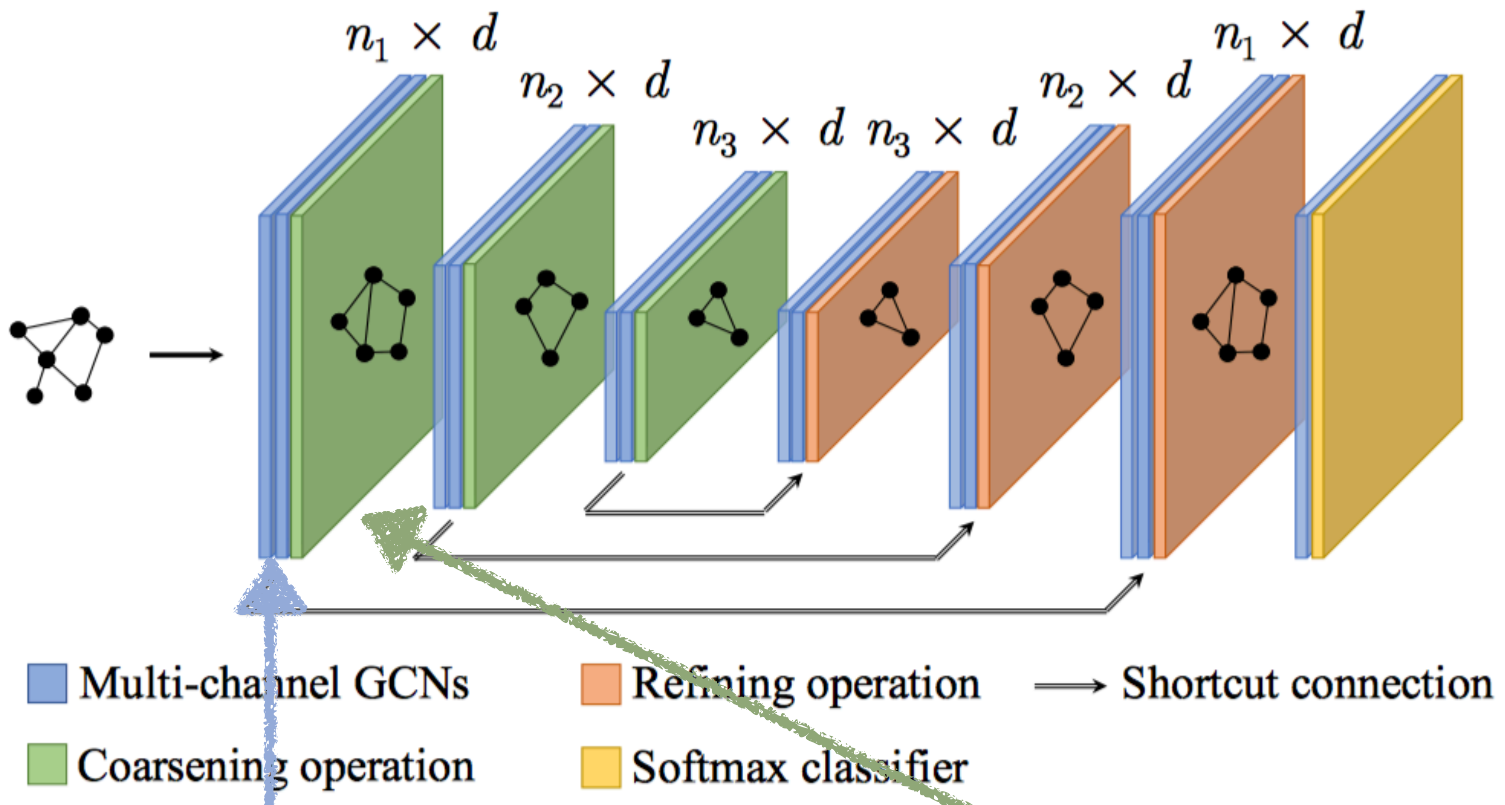
$\mathcal{G}_i$ with n_i nodes

$A_i \in \mathbb{R}^{n_i \times n_i}$: adjacency matrix

$H_i \in \mathbb{R}^{n_i \times d_i}$: hidden representation matrix

$d_i = d_{i+1} = d$

Model



$$G_i = \mathrm{ReLU}\left(\tilde{D}_i^{-\frac{1}{2}}\tilde{A}_i\tilde{D}_i^{-\frac{1}{2}}H_i\theta_i\right), \qquad (1)$$

where $H_1 = X$, $\mathrm{ReLU}(x) = \max(0, x)$, adjacency matrix with self-loop $\tilde{A}_i = A_i + I$, $\tilde{D}_i$ is the degree matrix of $\tilde{A}_i$, and $\theta_i \in \mathbb{R}^{d_i \times d_{i+1}}$ is a trainable weight matrix. For ease of parameter tuning, we set output dimension $d_i = d$ for all coarsening and refining layers throughout this paper.

aggregates structurally similar nodes into hyper-nodes, producing a coarser graph $\mathcal{G}_{i+1}$ and node embedding matrix $H_{i+1}$with fewer nodes.

Model — Coarsening operation

aggregates structurally similar nodes into hyper-nodes, producing a coarser graph $\mathcal{G}_{i+1}$ and node embedding matrix $H_{i+1}$with fewer nodes.

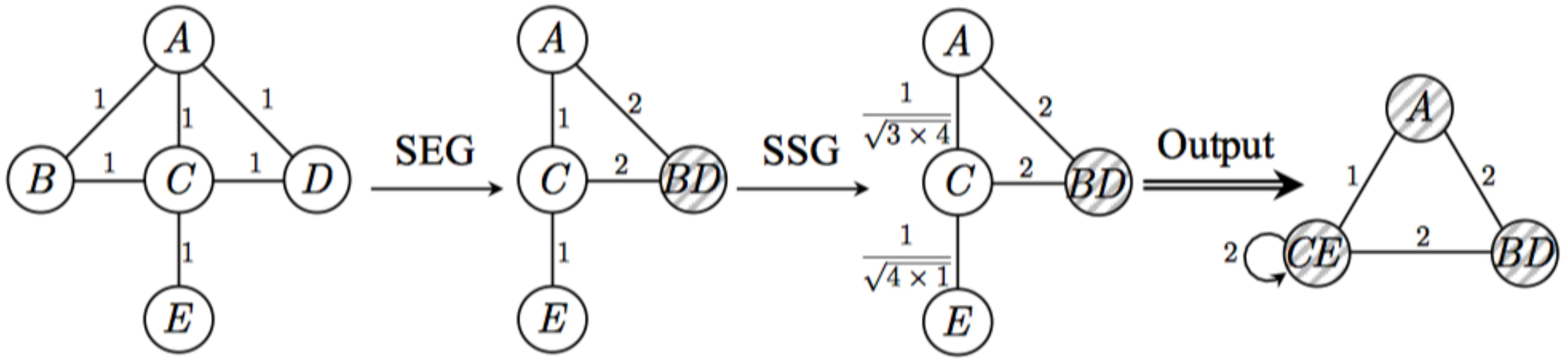
1. **conduct structural equivalence grouping(SEG).**

If two nodes **share the same set of neighbors**, they are considered to be structurally equivalent. Assign these two nodes to **be a hyper-node**. Then **mark**.

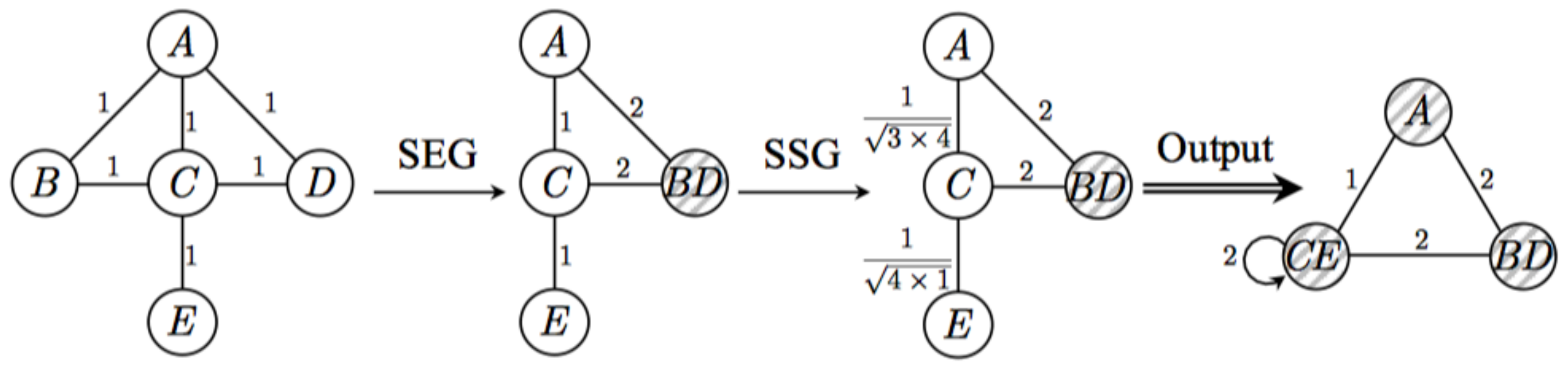
2. **structural similarity grouping(SSG).**

Then, we calculate the structural similarity between the unmarked node pairs (v_j, v_k) as the normalized connection strength $s(v_j, v_k)$:

$$s(v_j, v_k) = \frac{A_{jk}}{\sqrt{D(v_j) \cdot D(v_k)}},$$



Model — Coarsening operation



$M_i \in \mathbb{R}^{n_i \times n_{i+1}}$. Formally, at layer i, entry m_{jk} in the grouping matrix M_i is calculated as:

$$m_{jk} = \begin{cases} 1, & \text{if } v_j \text{ in } \mathcal{G}_i \text{ is grouped into } v_k \text{ in } \mathcal{G}_{i+1}; \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

$$A_1 = \begin{array}{c|c} \begin{array}{ccccc} A & B & C & D & E \end{array} & \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} & \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} \end{array} \qquad M_1 = \begin{array}{cc} \begin{array}{ccc} A & BD & CE \end{array} & \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} \end{array}$$

$$A_2 = M_1^\top A_1 M_1 = \begin{array}{cc} \begin{array}{ccc} A & BD & CE \end{array} & \\ \begin{pmatrix} 0 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 2 \end{pmatrix} & \begin{array}{c} A \\ BD \\ CE \end{array} \end{array}$$

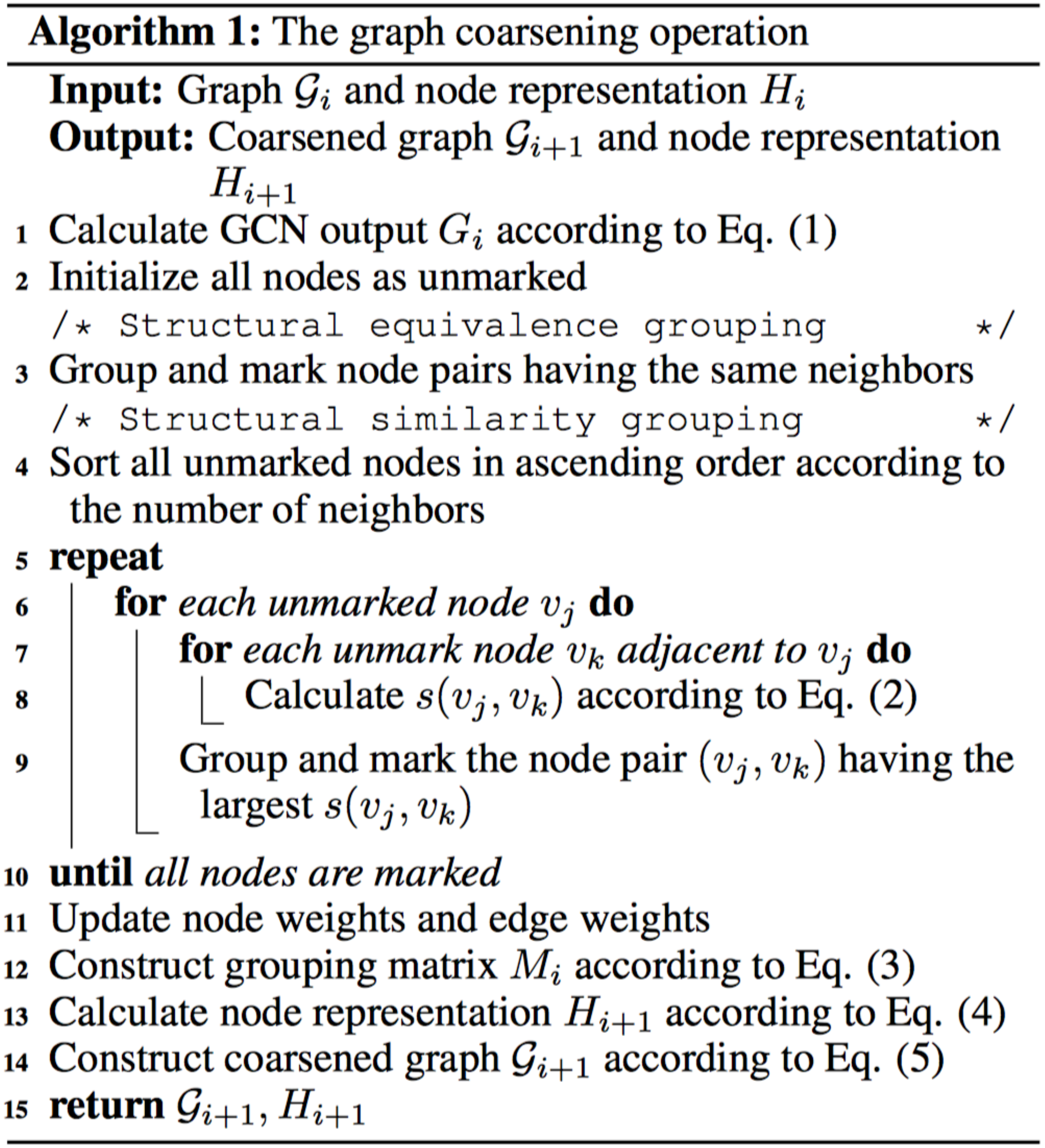
hidden node embedding matrix:

$H_{i+1} = M_i^\top \cdot G_i$

adjacency matrix of $\mathscr{G}_{i+1}$

$A_{i+1} = M_i^\top \cdot A_i \cdot M_i$

Model — Coarsening operation



Algorithm 1: The graph coarsening operation

```
Input: Graph $\mathcal{G}_i$ and node representation $H_i$
Output: Coarsened graph $\mathcal{G}_{i+1}$ and node representation
        $H_{i+1}$
Calculate GCN output $G_i$ according to Eq. (1)
Initialize all nodes as unmarked
/* Structural equivalence grouping                    */
Group and mark node pairs having the same neighbors
/* Structural similarity grouping                     */
Sort all unmarked nodes in ascending order according to
  the number of neighbors
repeat
    for each unmarked node $v_j$ do
        for each unmark node $v_k$ adjacent to $v_j$ do
            Calculate $s(v_j, v_k)$ according to Eq. (2)
        Group and mark the node pair $(v_j, v_k)$ having the
          largest $s(v_j, v_k)$
until all nodes are marked
Update node weights and edge weights
Construct grouping matrix $M_i$ according to Eq. (3)
Calculate node representation $H_{i+1}$ according to Eq. (4)
Construct coarsened graph $\mathcal{G}_{i+1}$ according to Eq. (5)
return $\mathcal{G}_{i+1}$, $H_{i+1}$
```

Model – Refining operation

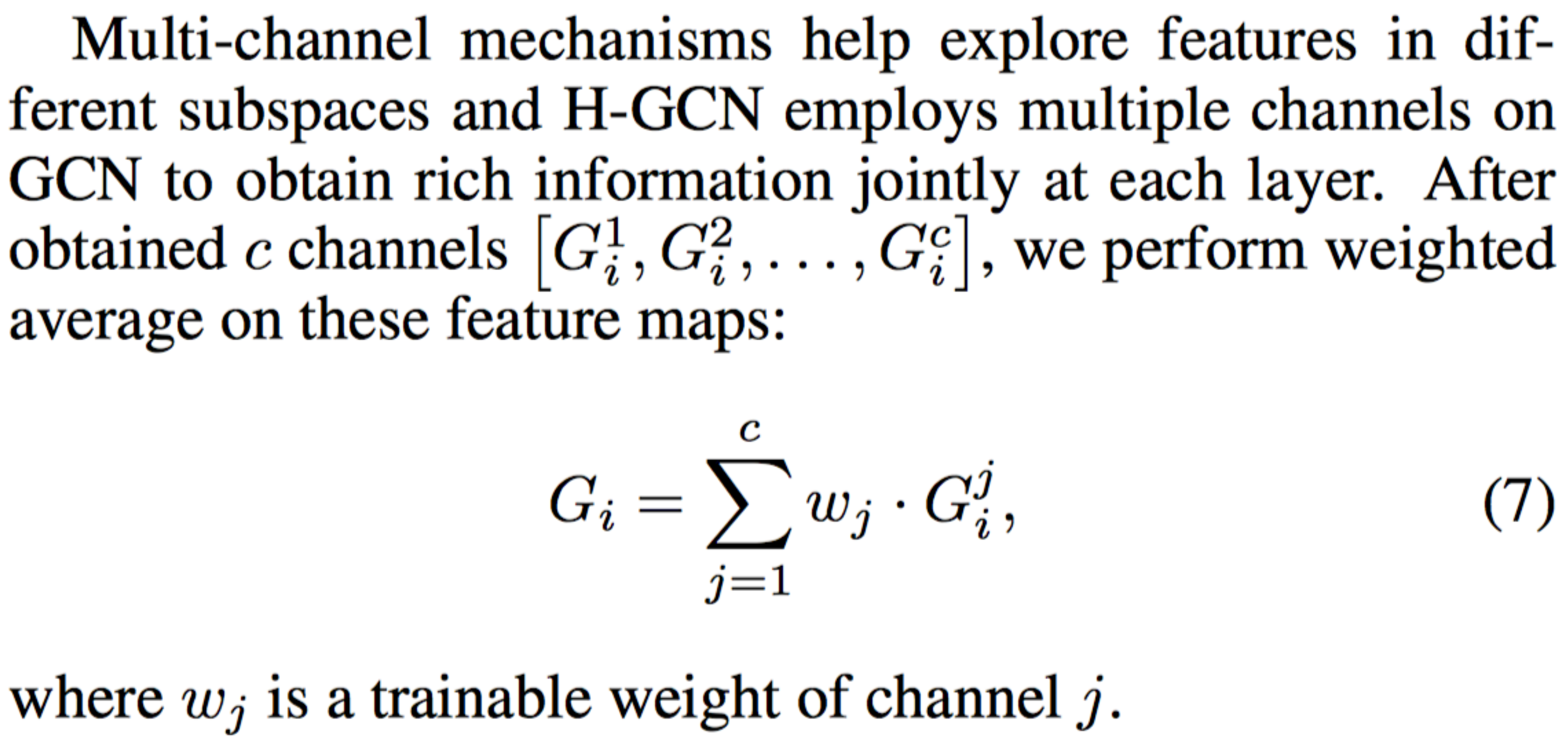
- coarsening layers and refining layers are symmetrical $M_i = M_{l-i}$
- residual connections between the two corresponding coarsening and refining layers.

$$H_i = M_{l-i} \cdot G_i + G_{l-i}.$$

Model — Node Weight Embedding and Multiple Channels

Here we transform the node weight into real-valued vectors by looking up one randomly initialized node weight embedding matrix $V \in \mathscr{R}^{|T| \times p}$, where T is the set of node weights and p is the dimension of the embedding.

We then concatenate H_i and S_i and the resulting (d+p)-dimensional matrix will be fed into the next GCN layer subsequently. $\theta_i \in \mathscr{R}^{(d+p) \times d}$



Multi-channel mechanisms help explore features in different subspaces and H-GCN employs multiple channels on GCN to obtain rich information jointly at each layer. After obtained c channels $\left[G_i^1, G_i^2, \ldots, G_i^c\right]$, we perform weighted average on these feature maps:

$$G_i = \sum_{j=1}^{c} w_j \cdot G_i^j, \qquad (7)$$

where w_j is a trainable weight of channel j.

Model — the output layer

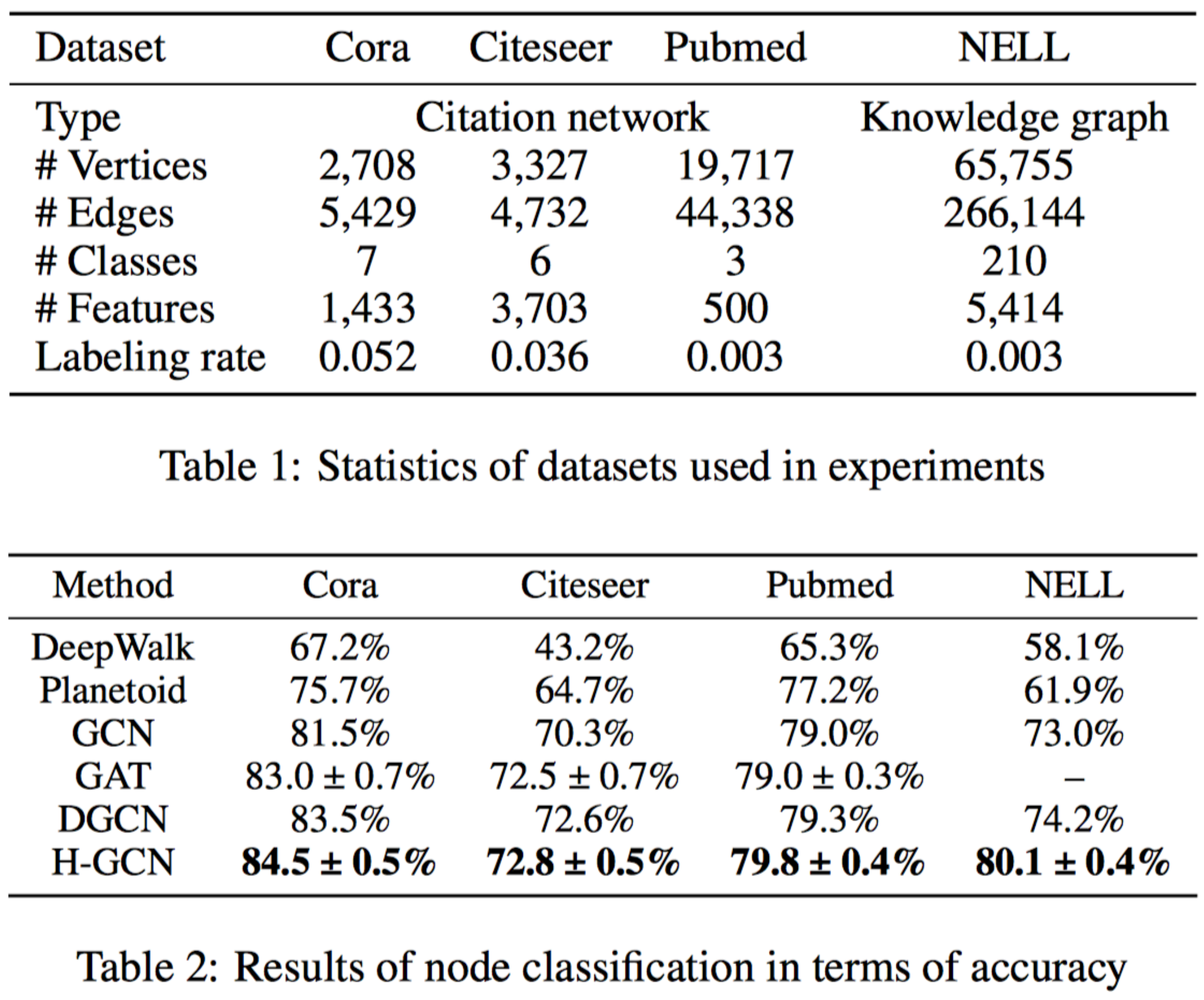
$$H_l = \text{softmax}\left(\text{ReLU}\left(\tilde{D}_l^{-\frac{1}{2}}\tilde{A}_l\tilde{D}_l^{-\frac{1}{2}}H_{l-1}\theta_l\right)\right), \qquad (8)$$

where $\theta_l \in \mathbb{R}^{d\times|\mathcal{Y}|}$ is a trainable weight matrix and $H_l \in \mathbb{R}^{n_1\times|\mathcal{Y}|}$ denotes the probabilities of nodes belonging to each class $y \in \mathcal{Y}$.

The loss function is defined as the cross-entropy of predictions over the labeled nodes:

$$\mathcal{L} = -\sum_{i=1}^{m}\sum_{y=1}^{|\mathcal{Y}|}\mathbb{I}(h_i = y_i)\log P(h_i, y_i), \qquad (9)$$

Experiments

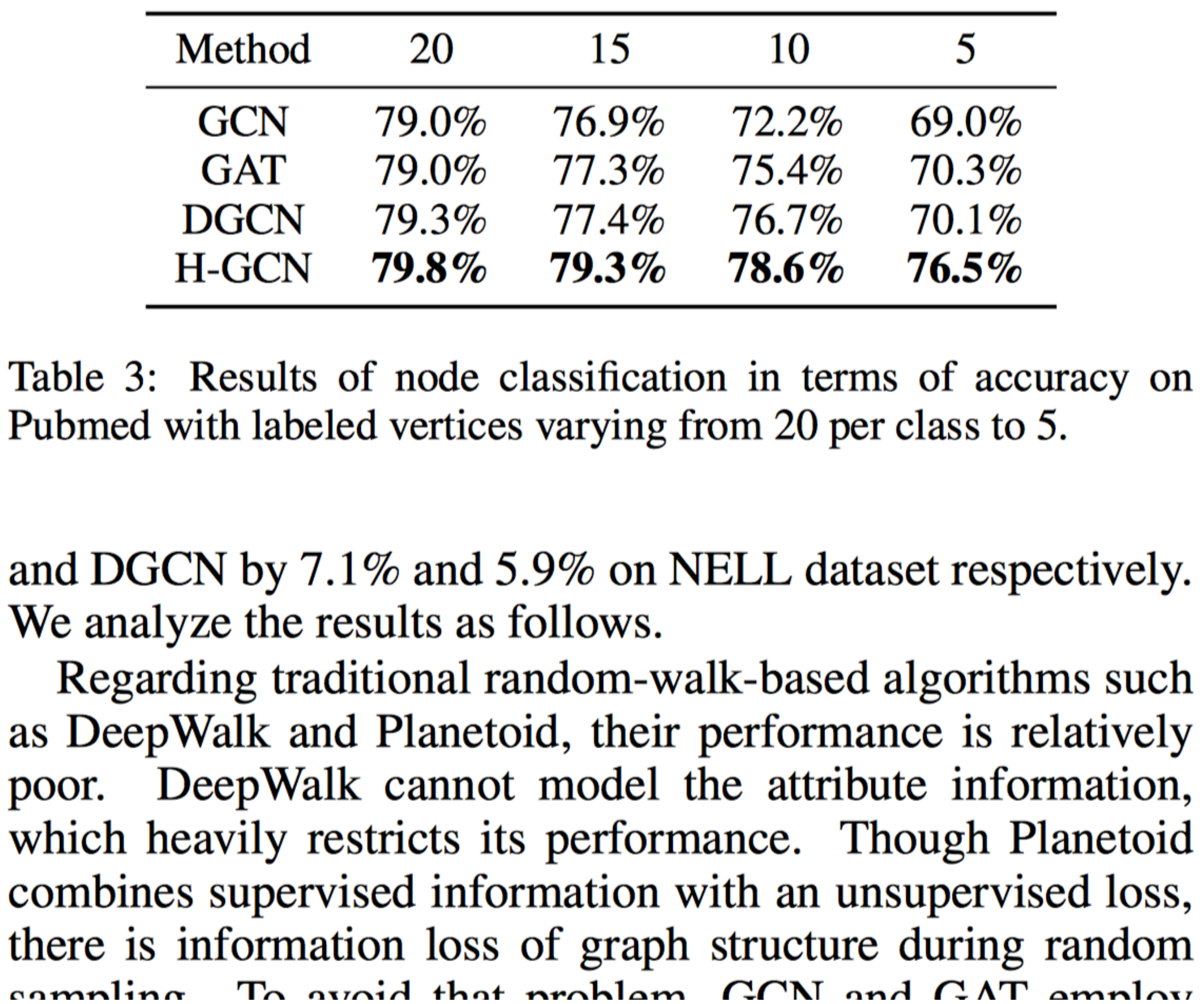


Dataset	Cora	Citeseer	Pubmed	NELL
Type	Citation network			Knowledge graph
# Vertices	2,708	3,327	19,717	65,755
# Edges	5,429	4,732	44,338	266,144
# Classes	7	6	3	210
# Features	1,433	3,703	500	5,414
Labeling rate	0.052	0.036	0.003	0.003

Table 1: Statistics of datasets used in experiments

Method	Cora	Citeseer	Pubmed	NELL
DeepWalk	67.2%	43.2%	65.3%	58.1%
Planetoid	75.7%	64.7%	77.2%	61.9%
GCN	81.5%	70.3%	79.0%	73.0%
GAT	83.0 ± 0.7%	72.5 ± 0.7%	79.0 ± 0.3%	–
DGCN	83.5%	72.6%	79.3%	74.2%
H-GCN	**84.5 ± 0.5%**	**72.8 ± 0.5%**	**79.8 ± 0.4%**	**80.1 ± 0.4%**

Table 2: Results of node classification in terms of accuracy



Method	20	15	10	5
GCN	79.0%	76.9%	72.2%	69.0%
GAT	79.0%	77.3%	75.4%	70.3%
DGCN	79.3%	77.4%	76.7%	70.1%
H-GCN	**79.8%**	**79.3%**	**78.6%**	**76.5%**

Table 3: Results of node classification in terms of accuracy on Pubmed with labeled vertices varying from 20 per class to 5.

and DGCN by 7.1% and 5.9% on NELL dataset respectively. We analyze the results as follows.

Regarding traditional random-walk-based algorithms such as DeepWalk and Planetoid, their performance is relatively poor. DeepWalk cannot model the attribute information, which heavily restricts its performance. Though Planetoid combines supervised information with an unsupervised loss, there is information loss of graph structure during random sampling. To avoid that problem, GCN and GAT employ

Thanks :)