Lasagne: A Multi-Layer Graph Convolutional Network Framework via Node-aware Deep Architecture

(Extended Abstract)

Xupeng Miao[†$] Wentao Zhang[†$] Yingxia Shao[‡] Bin Cui[†¶] Lei Chen[§] Ce Zhang[♯] Jiawei Jiang[♯]
[†] School of Computer Science & Key Lab of High Confidence Software Technologies (MOE), Peking University
[¶]Institute of Computational Social Science, Peking University (Qingdao)
[‡]School of Computer Science, BUPT, [§]Department of CSE, HKUST, [♯]ETH Zurich, [$]Tencent Inc.
[†]{xupeng.miao, wentao_zhang, bin.cui}@pku.edu.cn
[‡]shaoyx@bupt.edu.cn [§]leichen@cse.ust.hk [♯]{ce.zhang, jiawei.jiang}@inf.ethz.ch

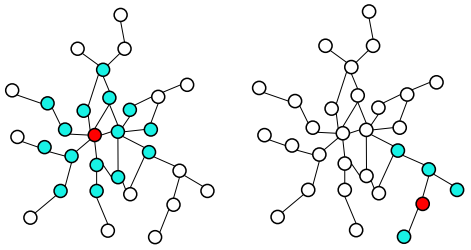
2022 IEEE 38th International Conference on Data Engineering (ICDE) | 978-1-6654-0883-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDE53745.2022.00157

***Abstract*—In this paper, we propose Lasagne, a novel multi-layer graph convolutional network (GCN) framework to overcome the over-smoothing problem and realize the full potentials of deep GCNs. We analyze how node localities affect the information propagation in GCN, propose an adaptive novel node aggregation mechanism and further demystify from a mutual information view. Evaluation results on both real-world benchmark data sets and large-scale industrial production data sets show Lasagne significantly outperforms the state-of-the-art methods without considering the node locality.**

I. Introduction

GCNs [1], [2] are becoming more and more attractive in the graph data management community. They generalize CNNs by applying the "graph convolution" operation on the neighbors of a node to obtain the node embedding layer by layer. With the rapid growth of graph data, stacking multiple GCN layers with a deep architecture becomes promising to learn complex node representations at different levels of abstraction.

However, deep GCNs often achieve the best performance with limited depth even on large graphs [3]. The degradation of learning occurs due to the over-smoothing problem, in which the output features may be over-smoothed and nodes may become indistinguishable.



(a) "central" node (b) "non-cnetral" node

Fig. 1. Illustration of the nodes (red) with different localities and their 2-hop neighborhoods (blue).

In this work, we surprisingly notice that the *node locality* on the graph can be used to reduce the information loss caused by the over-smoothing problem. As shown in Figure 1, central nodes and non-central nodes lead to different number of neighborhoods within the same hops. Therefore, we propose Lasagne, the node-aware architecture to adaptively aggregate the node embeddings for different nodes in a layer level, which makes the architecture fundamentally different from previous studies forcing all nodes using the same network depth. We further demystify the model from a mutual information view and confirm that our approach preserves more useful information across layers than vanilla GCN and its variants.

II. Problem Definition

For the input undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes $v_i \in \mathcal{V}$, edges $(v_i, v_j) \in \mathcal{E}$, let $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ be the adjacency matrix and $\boldsymbol{X} \in \mathbb{R}^{N \times M}$ be the node feature matrix, where M is the dimension of the attributive features. GCN follows the layer-wise propagation rule. At layer l, the output is the hidden representation $\boldsymbol{H}^{(l)}$:

$$\boldsymbol{H}^{(l)} = \delta(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)}), \tag{1}$$

where $\hat{\boldsymbol{A}} = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$ and $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$. $\boldsymbol{I}_N$ is the identity, $\widetilde{\boldsymbol{D}}_{ii} = \sum_j \widetilde{\boldsymbol{A}}_{ij}$ and $\boldsymbol{W}^{(l)}$ is a layer-specific trainable weight matrix. $\delta(\cdot)$ denotes an activation function, such as $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$. $\boldsymbol{H}^{(l-1)}$ is the input of lth layer and $\boldsymbol{H}^{(0)} = \boldsymbol{X}$. An L-layer GCN makes predictions with $\boldsymbol{H}^{(L)}$.

III. Methods

Overview. We illustrate the architecture of Lasagne in Figure 2. The dense connection makes each layer collects information from different hops of neighbors and becomes a unique convolutional filter based on the previous layer, which is similar to the scheme in CNNs. It not only helps to reduce the gradients vanishment but also preserves the node representations at different levels of abstraction and prevents the information loss. Considering the different node localities, we propose a layer aggregator structure and design three different node-aware aggregators. They can automatically learn the hidden representations from

2375-026X/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDE53745.2022.00157

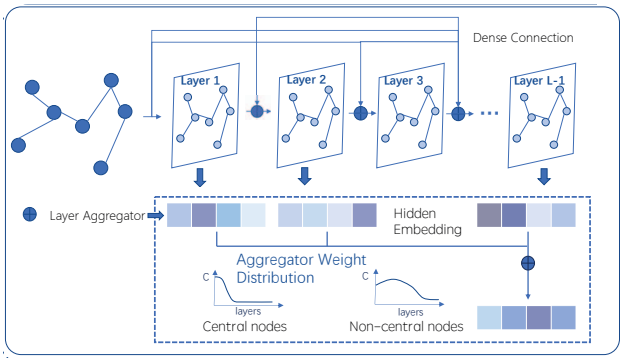


Fig. 2. The main architecture of Lasagne.

the needed layers respectively for each node, which helps to alleviate the over-smoothing problem to some extent.

Layer Aggregator. Figure 2 shows the layer aggregator after each layer's output. It aggregates all previous layers' hidden representations and makes up of a dense connection structure. DenseGCN uses a straightforward vertex-wise concatenation to densely fuses the input graph with all the intermediate GCN layer outputs. However, as the concatenation just treats the node hidden representations from different layers in the same way, it cannot capture the node locality. To better utilize different levels of node abstraction, we propose the node-aware layer aggregators:

$$\boldsymbol{H}^{(l)} = \text{Aggregator}(\boldsymbol{C}^{(l)}, \boldsymbol{H}^{(1)}, \boldsymbol{H}^{(2)}, ..., \boldsymbol{H}^{(l)}), \tag{2}$$

where $1 < l < L$, $\boldsymbol{C}^{(l)} \in \mathbb{R}^{N \times l}$ is the weight matrix, *enabling different nodes using a different weighted aggregation for the previous layers*. For the l-th layer's output of a specific node v_i, the trainable parameter $\mathbf{C}_{i,j}^{(l)}$ represents for the contribution from the j-th layer's output. In our approach, we explore three special approaches below.

1) Weighted aggregator: To learn the node preference of previous layers' information, we use the trainable weight matrix $\boldsymbol{C}^{(l)}$ to capture the different contribution of previous layers for each node. The operation is a kind of weighted-sum on the hidden representations.

2) Max Pooling aggregator: This operation adaptively captures the most informative layer for each feature coordinate without any additional parameters to learn. It can be viewed as a special case of weighted aggregator but scale $\boldsymbol{C}^{(l)}$ to the hidden dimension where each column only has one 1 item and the rest of them are 0.

3) Stochastic aggregator.: We adopt a layer-wise dropout manner that using the Bernoulli sampling procedure to aggregate embeddings. Shortening the depth during training reduces the chain of forward propagation steps and gradient computations, which strengthens the information propagation. Specifically, we propose a learnable activation function for each layer of each node and each item of $\boldsymbol{C}^{(l)}$ becomes an independent Bernoulli random variable.

TABLE I
THE TEST ACCURACY (IN %) ON THE CITATION DATASET. * INDICATES THAT WE RAN OUR OWN IMPLEMENTATION.

Models	Cora	Citeseer	Pubmed
GPNN	81.8	69.7	79.3
NGCN	83.0	72.2	79.5
DGCN	83.5	72.6	80
DropEdge	82.8	72.3	79.6
STGCN	83.6	72.6	79.5
DGI	82.3±0.6	71.8±0.7	76.8±0.6
GMI	82.7±0.2	73.0±0.3	80.1±0.2
GIN	77.6±1.1	66.1±0.9	77.0±1.2
SGC	81.0±0.0	71.9±0.1	78.9±0.0
LGCN	83.3±0.5	73.0±0.6	79.5±0.2
APPNP	83.3±0.5	71.8±0.5	80.1±0.2
GAT	83.0±0.7	72.5±0.7	79.0±0.3
Pairnorm*	81.4±0.6	68.5±0.9	79.1±0.5
ADSF*	83.8±0.5	72.8±0.7	80.1±0.8
MixHop*	82.1±0.4	71.4±0.8	80.0±1.1
MADReg*	82.3±0.8	71.6±0.9	79.5±0.6
GCN*	81.8±0.5	70.8±0.5	79.3±0.7
JK-Net*	81.8±0.5	70.7±0.7	78.8±0.7
ResGCN*	82.2±0.6	70.8±0.7	78.3±0.6
DenseGCN*	82.1±0.5	70.9±0.8	79.1±0.9
Lasagne (Weighted)*	84.1±0.2	73.2±0.5	79.5±0.4
Lasagne (Stochastic)*	**84.2±0.5**	73.1±0.6	**80.2±0.5**
Lasagne (Max pooling)*	84.1±0.8	**73.3±0.5**	79.6±0.6

IV. EXPERIMENTS

End-to-end Comparison. We compare our method with 20 representative state-of-the-art methods on benchmark datasets. Table I shows that Lasagne outperforms the baselines by a significant margin (e.g., o.4% on Cora, 0.3% on Citeseer and 0.1% on Pubmed). Besides, different aggregators may result in different performance.

Model Interpretation We also the mutual information between the last layer's hidden representation and the input feature. The results of a 10-layer Lasagne and other baselines on Cora shows that, our method indeed helps deep GCN to preserve more information than other approaches, thus improving the model effectiveness. More detailed experimental results are in [4].

ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China under Grant (No. 61832001, U1936104), Beijing Academy of Artificial Intelligence (BAAI), PKU-Baidu Fund 2019BD006, and PKU-Tencent Joint Research Lab.

REFERENCES

[1] W. Zhang, X. Miao, Y. Shao, J. Jiang, L. Chen, O. Ruas, and B. Cui, "Reliable data distillation on graph convolutional network," in *SIGMOD*. ACM, 2020, pp. 1399–1414.
[2] X. Miao, H. Zhang, Y. Shi, X. Nie, Z. Yang, Y. Tao, and B. Cui, "Het: Scaling out huge embedding model training via cache-enabled distributed framework," in *Proc. VLDB Endow.*, 2022.
[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[4] X. Miao, W. Zhang, Y. Shao, B. Cui, L. Chen, C. Zhang, and J. Jiang, "Lasagne: A multi-layer graph convolutional network framework via node-aware deep architecture," *TKDE*, pp. 1–1, 2021.