PROPAGATE DEEPER AND ADAPTABLE GRAPH CONVOLUTIONAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Convolutional Networks (GCNs) are the basic architecture for handling graph-structured data. Deeper GCNs are required for large and sparse graph data. As the number of layers increases, the performance of GCNs degrades, which is commonly attributed to over-smoothing but is constantly debated. In this paper, we eliminate the equivalence between model degradation and over-smoothing or gradient vanishing and propose a systematic solution, an Adaptive DeepGCN (ADGCN) architecture, which makes the model the potential to address all issues. We place learnable parameters at the appropriate locations to make adaptive adjustments to different graph-structured data. We conduct experiments on real-world datasets to verify the stability and adaptability of our architecture.

1 INTRODUTION

The studies have demonstrated that deeper architectures can increase the expression ability of deep neural networks in many domains, however, model degradation refers to the decline in generalization performance after stacking many GCN layers. Many studies have investigated the causes of it. We discover that there are causes of diversification that depend on different concerns. Some studies have focused primarily on over-smoothing caused by multiple graph convolution operations, others have concentrated on gradient vanishing, and still, others have expanded their perspective to consider the generalization gap based on the Lipschitz constraint to identify the key factors which can enhance model performance (Yang et al., 2020; Chen et al., 2020a; Feng et al., 2020; Zhu et al., 2021; Yan et al., 2021; Chien et al., 2021; Godwin et al., 2022; Cong et al., 2021; Zhang et al., 2022).

We argue that model degradation is a complex problem that isn't caused by a single issue but rather by the coexistence of various causes. The reason why the conclusions of the above studies are different is that the key reasons revealed have altered due to different settings. For instance, models with less than 10 layers are usually the choice of some research, the gradient remains normal, and the model is not complex, but the over-smoothing will be prominent. The gradient vanishing becomes noticeable and other issues are hidden as the number of layers increasing. We propose a systematic solution, an Adaptive DeepGCN (ADGCN) architecture, which makes the model the potential to address all issues. It is beneficial to consistently manage the interaction between various components through end-to-end learning in neural networks. We place learnable parameters at the appropriate locations so that the model has space to make adaptive adjustments to different graph-structured data. The effect is comparable to the architecture of precise manual parameter modification.

The ADGCN architecture is composed of three parts that regulate the model's information propagation, control the model complexity, and introduce the attention mechanism as the control switch to keep the model stable, respectively. We learn more about the "systematic" scheme's necessity through experiments. When you skip the propagation operation and directly consider reducing model complexity (i.e., the number of parameters in the model is too large for the current dataset), the generalization ability and training effectiveness will be improved in comparison to the basic GCNs, but the degradation will still happen.

2 ADAPTIVE DEEPGCN

When the performance of GCNs combined with residual connections or other technologies is still reduced (Li et al., 2019), it is clear that only guaranteeing the training accuracy (roughly equal to 1)

is not enough. Additionally, as the number of layers increased, the performance of the GCN variants, namely GCNII (Chen et al., 2020b) and Decoupled GCN (DGCN) (Cong et al., 2021), improved. In order to create a more stable and adaptable deep GCN architecture, we also propose the ADGCN.

The ADGCN model can be mathematically formulated as $\mathbf{H} = \mathrm{softmax}(\mathbf{Z})$, $\mathbf{Z} = \sum_{l=1}^{L} \gamma_l \mathbf{H}^{(l)}$, and $\mathbf{H}^{(l)} = \sigma((\alpha_l \mathbf{P}\mathbf{H}^{(l-1)} + (1-\alpha_l)\mathbf{H}^{(0)})(\beta_l \mathbf{W}^{(l)} + (1-\beta_l)\mathbf{I}))$, where α_l, β_l and γ_l are the learnable parameters for the l-th layer of the model, and $\gamma_l \geq 0 \; \forall l \in \{0, 1, \ldots, L\}$, $\sum_{l=0}^{L} \gamma_l = 1$. Intuitively, α can determine the trade-off between topology information and the initial node information of each layer, which preserves node information while also expanding the receptive field. β controls the model complexity, and γ can aggregate each hidden layer with attention mechanism. In addition, GCNII has a similar architecture with hyper-parameters, DGCN with learnable parameters β and γ, and ResGCN with the residual connections. Due to the space limit, the mathematical formulas and other details of the models are deferred to Appendix.

Table 1: Comparison of classification accuracy on citation network datasets.

Dataset	**Models**	2 Layers(epoch)	4 Layers	8 Layers	16 Layers	32 Layers	64 Layers	128 Layers(epoch)
Cora	GCN	0.8010(600)	**0.8011(600)**	0.7471(600)	FTT	–	–	–
	ResGCN	**0.8106(600)**	0.7993(600)	0.7735(600)	0.7885(600)	0.7793(1000)	FTT	–
	GCNII	0.8006(200)	0.7965(200)	0.8004(200)	0.8098(200)	0.8219(200)	0.8153(200)	**0.8233(200)**
	DGCN	0.8094(200)	0.8093(200)	0.8113(200)	0.8105(200)	0.8062(200)	0.8125(200)	**0.8132(200)**
	ADGCN(α, β, γ)	0.7871(200)	0.8147(200)	0.8151(200)	0.8152(200)	0.8181(200)	0.8214(200)	**0.8222(200)**
	ADGCN(β)	**0.8138(200)**	0.8071(200)	0.8048(200)	0.7963(200)	0.6380(200)	FTT(200)	FTT(200)
	ADGCN(uniform)	0.7700(200)	0.7993(200)	0.7788(200)	0.7898(200)	0.7876(200)	0.7987(200)	**0.8069(200)**
PubMed	GCN	**0.7301(600)**	0.7292(600)	0.6748(600)	FTT	–	–	–
	ResGCN	0.7398(600)	**0.7678(600)**	0.7590(600)	0.7222(600)	0.7275(600)	FTT	–
	GCNII	0.7687(200)	0.7699(200)	0.7804(200)	0.7962(200)	0.7975(200)	**0.7984(200)**	0.7949(200)
	DGCN	0.7795(200)	0.7725(200)	0.7844(200)	0.7838(200)	0.7849(200)	0.7854(200)	**0.7890(200)**
	ADGCN(α, β, γ)	0.7653(200)	0.7812(200)	0.7902(200)	0.7908(200)	0.7922(200)	**0.7981(200)**	0.7945(200)
	ADGCN(β)	0.7727(200)	0.7751(200)	0.7852(200)	**0.7901(200)**	0.7694(200)	FTT(200)	FTT(200)
	ADGCN(uniform)	0.7668(200)	0.7625(200)	0.7706(200)	0.7702(200)	0.7731(200)	0.7669(200)	**0.7736(200)**

3 EXPERIMENT AND MAIN RESULT

We use the citation network datasets (Cora, Citeseer, and PubMed) to assess each model's accuracy for the node classification task (Sen et al., 2008). Due to the space limit, statistics of the datasets and detailed comparisons between different architectures are supplemented in Appendix.

As shown in Table 1, we highlight the most important results in bold, and FTT is the abbreviation of failure to train, which means that the model is unable to guarantee training accuracy within an acceptable number of iterations. In comparison to other models, ADGCN achieves higher accuracy with deeper layers without having to manually adjust the hyper-parameters for every configuration. ADGCN performs comparably to GCNII and outperforms ResGCN and DGCN. The optimal accuracies for the GCN and ResGCN are between 2 and 8 layers, which are less than the optimal value for the GCNII, DGCN and ADGCN. The final row of each dataset indicates that the model's performance will be significantly influenced by the parameters' initialization method, which should be taken into consideration. We initialize $\alpha_l, \beta_l, \gamma_l$ to $\lambda(1-\lambda)^l$, where λ is a hyperparameter and the model is insensitive to it.

As seen from the degradation phenomenon of ADGCN that only retains β_l parameters, the increased weight matrices result in excessive transformation operations, which make the model more complex and cause overfitting problems. It is distinct from alleviating over-fitting during the training process by improving the generalization bound of the model through perturbation analysis. DGCN and ADGCN with γ and β parameters solve the model degradation, which indicates that the information is not transferred smoothly between layers of the neural network. On the premise of aggregating the information of each layer through γ to ensure the completion of training, adjust the parameter amount adaptively at each layer for different data sets, so as to obtain a deeper GCN. The superior performance of ADGCN demonstrates that graph data has unique properties that set it apart from other types of data. We need to establish independent paths to each layer for the node information in the graph, while DGCN ignored node information and concentrated primarily on the over-smoothing issue brought about by processing topological information. Model degradation is a broad problem in general. A stable and adaptable architecture can be obtained on the assumption that each component is given careful consideration.

URM STATEMENT

The authors acknowledge that at least one key author of this work meets the URM criteria of ICLR 2023 Tiny Papers Track.

REFERENCES

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3438–3445, 2020a.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020b.

E. Chien, J. Peng, P. Li, and O. Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

Weilin Cong, Morteza Ramezani, and Mehrdad Mahdavi. On provable benefits of depth in training graph convolutional networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103, 2020.

Jonathan Godwin, Michael Schaarschmidt, Alexander L. Gaunt, Alvaro Sanchez-Gonzalez, Yulia Rubanova, Petar Velickovic, James Kirkpatrick, and Peter W. Battaglia. Simple GNN regularisation for 3d molecular property prediction and beyond. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 2017. *ArXiv abs/1609.02907*, 2017.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9267–9276, 2019.

Y. Rong, W. Huang, T. Xu, and J. Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.

Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. *arXiv preprint arXiv:2102.06462*, 2021.

Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek Abdelzaher. Revisiting over-smoothing in deep gcns. *arXiv preprint arXiv:2003.13663*, 2020.

Wentao Zhang, Zeang Sheng, Ziqi Yin, Yuezihan Jiang, Yikuan Xia, Jun Gao, Zhi Yang, and Bin Cui. Model degradation hinders deep graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 2493–2503, 2022.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. 2020.

Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. Interpreting and unifying graph neural networks with an optimization framework. In *Proceedings of the Web Conference 2021*, pp. 1215–1226, 2021.

A APPENDIX

A.1 GCN AND ITS VARIANTS

The following architectures are the GCN and its variants.

GCN. Kipf & Welling (2017) indicated that the GCN is the one-order approximation of spectral GCN with propagation operation $\mathbf{P}$ and transformation operation $\mathbf{W}$. The normalized propagation matrix (i.e., graph convolution operation) $\mathbf{P} = \mathbf{I_N} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{A}$ denotes adjacency matrix and $\mathbf{D}$ denotes the diagonal degree matrix. The output of l-th graph convolution layer is formulated as the following:

$$\mathbf{H}^{(l)} = \sigma(\mathbf{P}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}), \tag{1}$$

where $\sigma(\cdot)$ is ReLU activation, and $\mathbf{W}^{(l)}$ is the weight matrix of l-th layer.

ResGCN. Li et al. (2019) suggest GCN combined with residual connections to address the the gradient vanishing during deepening. The output of l-th layer is formulated as the following:

$$\mathbf{H}^{(l)} = \sigma(\mathbf{P}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}) + \mathbf{H}^{(l-1)}. \tag{2}$$

The ResGCN model can solve the gradient vanishing, but we still observe model degradation issue in Table 1 and Table 3.

GCNII. Chen et al. (2020b) is one of the state-of-the-art deep GCNs so far, which is composed of initial residual and identity mapping techniques. To achieve optimal generalization performance, the critical parameters (i.e., α and β) for different datasets must be manually adjusted. The l-th layer is formulated as the following:

$$\mathbf{H}^{(l)} = \sigma\Big(\big((1-\alpha_l)\mathbf{P}\mathbf{H}^{(l-1)} + \alpha_l\mathbf{H}^{(0)}\big)\big((1-\beta_l)\mathbf{I}_n + \beta_l\mathbf{W}^{(l)}\big)\Big), \tag{3}$$

where $\alpha_l \in [0, 1]$ and $\beta_l \in [0, 1]$.

DGCN. Cong et al. (2021) computes the output of the model with learnable parameters α_l and β_l. By separating expressive power from generalization ability, it creates a decoupled structure for GCNs. This can lessen the generalization gap's reliance on the weight matrices. It is defined as:

$$\mathbf{Z} = \sum_{l=1}^{L} \alpha_l \mathbf{H}^{(l)},\ \mathbf{H}^{(l)} = \mathbf{P}^l\mathbf{H}^{(l-1)}((1-\beta_l)\mathbf{I}_n + \beta_l\mathbf{W}^{(l)}), \tag{4}$$

where $\beta_l \in [0, 1]$.

A.2 ILLUSTRATING THE MODEL DEGRADATION

A.2.1 EXPERIMENT SETUP

Here are the models' additional empirical evaluations using the widely used standard citation network datasets. Statistics of the datasets are summarized in Table 2. We consider semi-supervised node classification tasks and contrast how the models' generalization performance changes as the number of layers increases. We set the hidden dimension as 64, dropout rate as 0.1, learning rate as 0.01, $\lambda = 0.1$.

Table 2: Dataset statistics.

Dataset	Type	Nodes	Edge	Classes	Features
Cora	Citation network	2708	10556	7	1433
Citeseer	Citation network	3327	9104	6	3703
PubMed	Citation network	19717	88648	3	500

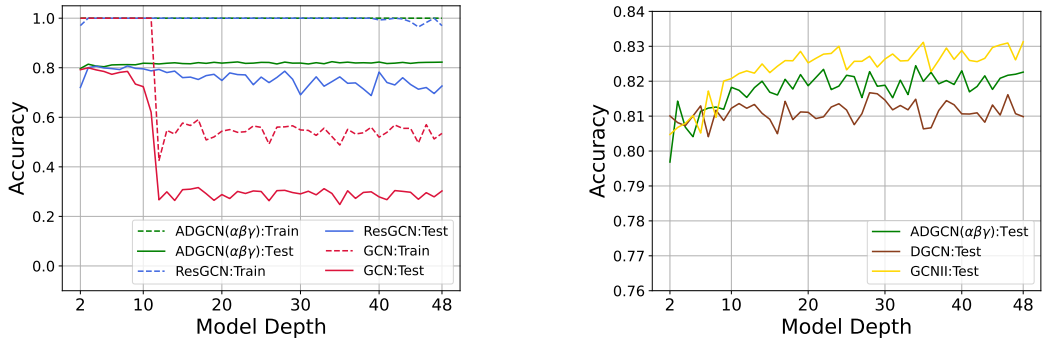
A.2.2 RESULTS

Table 3 is the supplementary results of Table 1 in section 3. It is obvious to see that all improved models have significantly higher accuracy when compared to GCN since Citeseer is a more dense data set than Cora and PubMed. Among them, ADGCN has attained decent accuracy at all depths.

Table 3: Accuracy for the node classification task.

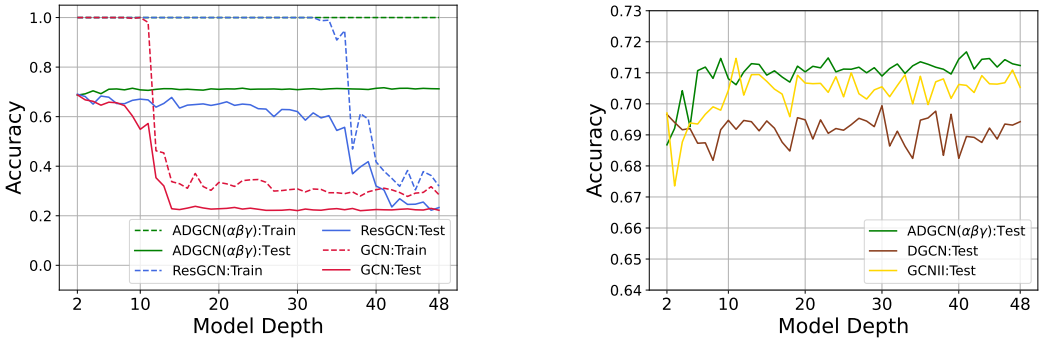
Datasets	Models	2 Layers(epoch)	4 Layers	8 Layers	16 Layers	32 Layers	64 Layers	128 Layers(epoch)
Citeseer	GCN	**0.6298(600)**	0.6010(600)	0.5745(600)	FTT	–	–	
	ResGCN	0.6927(600)	0.6801(600)	**0.6994(600)**	0.6889(600)	0.6353(1000)	FTT	–
	GCNII	0.6843(200)	0.6687(200)	0.6864(200)	0.6838(200)	0.6949(200)	0.6955(200)	**0.6971(200)**
	DGCN	0.6935(200)	0.6917(200)	0.6894(200)	0.6919(200)	0.6891(200)	0.6953(200)	**0.6980(200)**
	ADGCN(α, β, γ)	0.6933(200)	0.6951(200)	0.7082(200)	0.7002(200)	0.7030(200)	0.7031(200)	**0.7118(200)**
	ADGCN(β)	**0.6911(200)**	0.6867(200)	0.6771(200)	0.6576(200)	0.4909(200)	FTT(200)	FTT(200)
	ADGCN(uniform)	0.6433(200)	0.6630(200)	0.6605(200)	0.6845(200)	**0.7081(200)**	0.6917(200)	0.6746(200)

Figures 1 to Figures 4 depict the change in generalization performance as the depth of the models increases from 2 to 48 with various architectures. They complete the comparison on different datasets and show similar results. These indicate the effectiveness of the models.



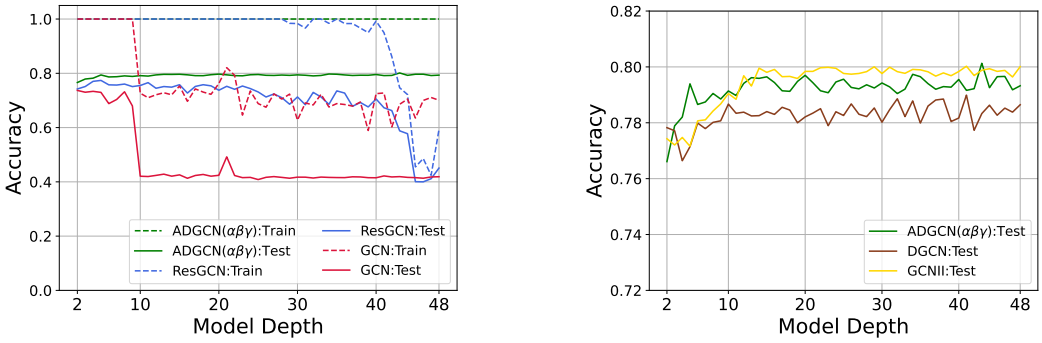
(a) (b)

Figure 1: On the Cora dataset, (a) compare the ADGCNs with learnable parameters, ResGCNs with residual connections and standard GCNs respectively. Note that the model degradation is reflected in the red solid line in ranges 2 to 11 and the blue solid line in ranges 2 to 48 showing a slightly decline in testing accuracy, meanwhile, the training accuracy roughly equal to 1. (b) compare the ADGCNs, DGCNs and GCNIIs in testing accuracy in ranges 2 to 48. The performance of ADGCN is superior to that of DGCN and nearly equal to that of GCNII with manual parameter adjustment. There is no model degradation in any of the three models.



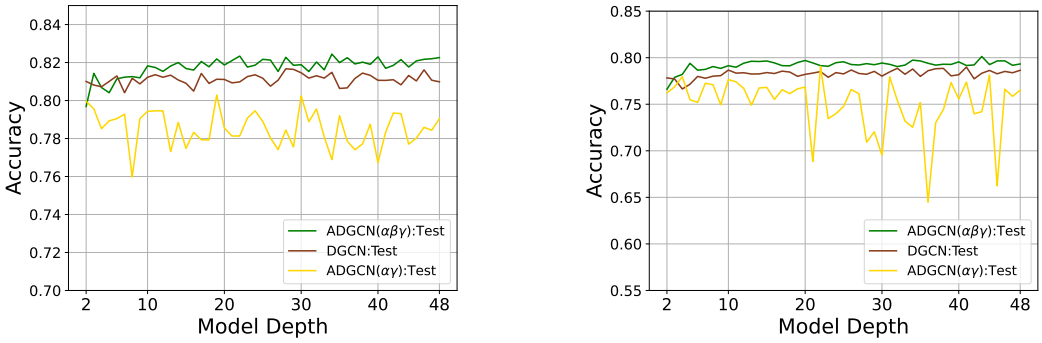
(a) (b)

Figure 2: On the Citeseer dataset, (a) compare the ADGCNs, ResGCNs and standard GCNs respectively. Note that the experimental results are similar to Cora. (b) compare the ADGCNs, DGCNs and GCNIIs in testing accuracy in ranges 2 to 48. The performance of ADGCN is superior to that of DGCN and that of GCNII with manual parameter adjustment. There is no model degradation in any of the three models, meanwhile, ADGCN and GCNII are more stable than DGCN.



(a) (b)

Figure 3: On the PubMed dataset, (a) compare the ADGCNs, ResGCNs and standard GCNs respectively. Note that the experimental results are similar to Cora and Citeseer. (b) compare the ADGCNs, DGCNs and GCNIIs in testing accuracy. The performance of ADGCN is superior to that of DGCN and nearly equal to that of GCNII. There is no model degradation in any of the three models.



(a) Cora (b)

Figure 4: Comparison of models with different parameter configurations. The yellow curves show instability compared with the others . This demonstrates the necessity of controlling model complexity through β.