On Provable Benefits of Depth in Training Graph Convolutional Networks

Weilin Cong
Pennsylvania State University
wxc272@psu.edu

Morteza Ramezani
Pennsylvania State University
morteza@cse.psu.edu

Mehrdad Mahdavi
Pennsylvania State University
mzm616@psu.edu

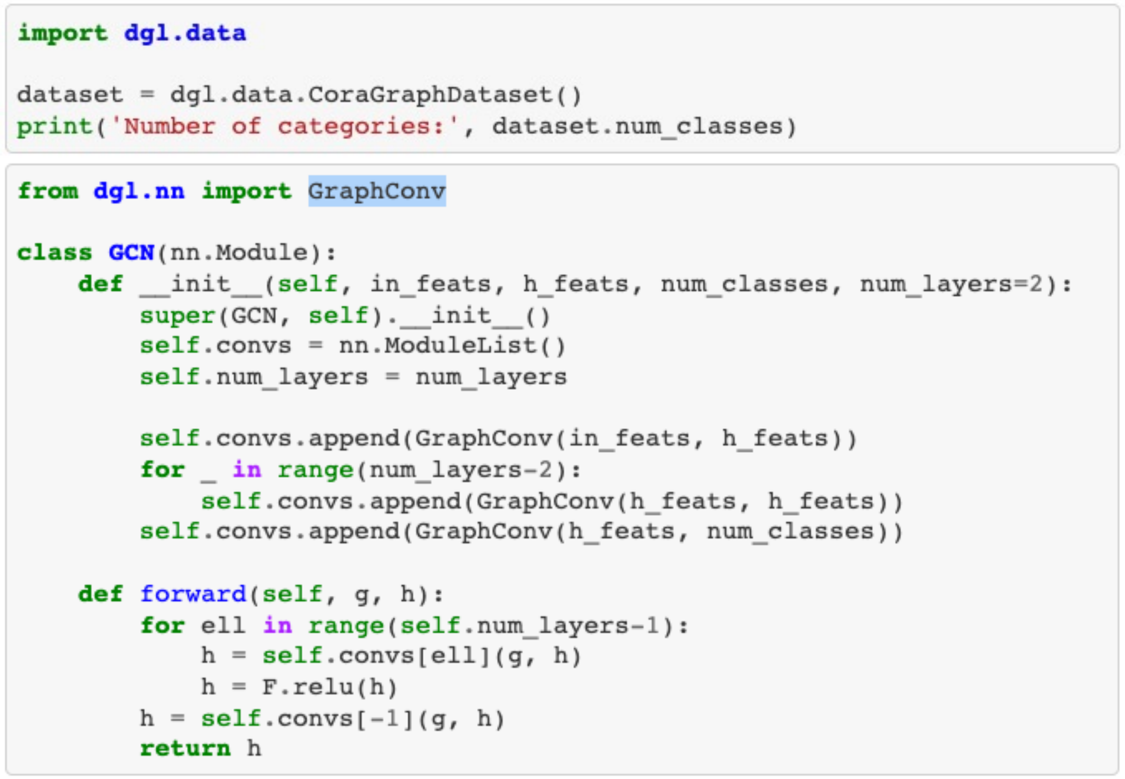
PennState

Motivation

- Graph neural networks have achieved state-of-the-art performance in many graph-structured applications.
- Existing GNNs are limited to very shallow structures because GNNs suffer from performance degradation issue as the number of layers increases.
- The conventional wisdom is that adding the number of layers cause **over-smoothing**.
- We observe that *there exists a discrepancy between the theoretical understanding of the inherent capabilities of GNN and their practical performance.*

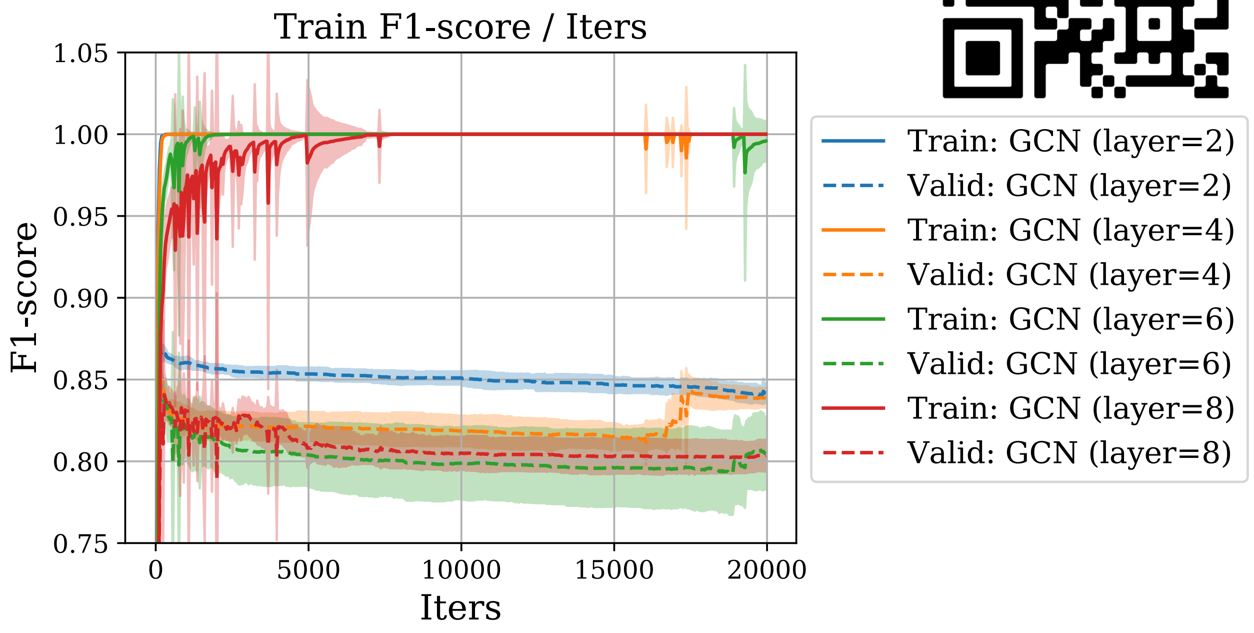
Motivation

- Experiment observations



```python
import dgl.data

dataset = dgl.data.CoraGraphDataset()
print('Number of categories:', dataset.num_classes)
```

```python
from dgl.nn import GraphConv

class GCN(nn.Module):
    def __init__(self, in_feats, h_feats, num_classes, num_layers=2):
        super(GCN, self).__init__()
        self.convs = nn.ModuleList()
        self.num_layers = num_layers

        self.convs.append(GraphConv(in_feats, h_feats))
        for _ in range(num_layers-2):
            self.convs.append(GraphConv(h_feats, h_feats))
        self.convs.append(GraphConv(h_feats, num_classes))

    def forward(self, g, h):
        for ell in range(self.num_layers-1):
            h = self.convs[ell](g, h)
            h = F.relu(h)
        h = self.convs[-1](g, h)
        return h
```



Example code

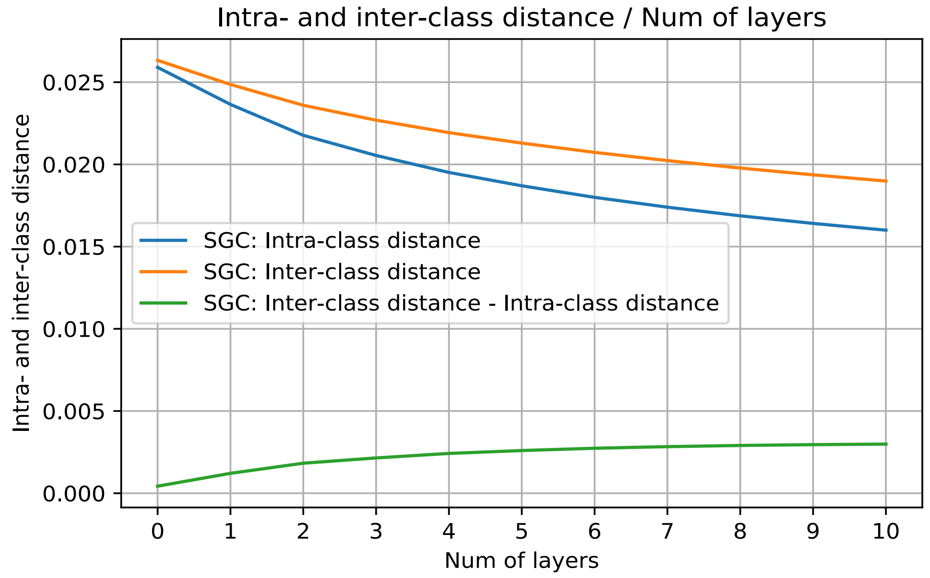
Motivation

- In this paper, we aim at answering two fundamental questions:
  - *Q1: Does increasing depth really impair the expressive power of GCNs?*
  - *Q2: If deep GCN is expressive, then why does it generalize poorly?*

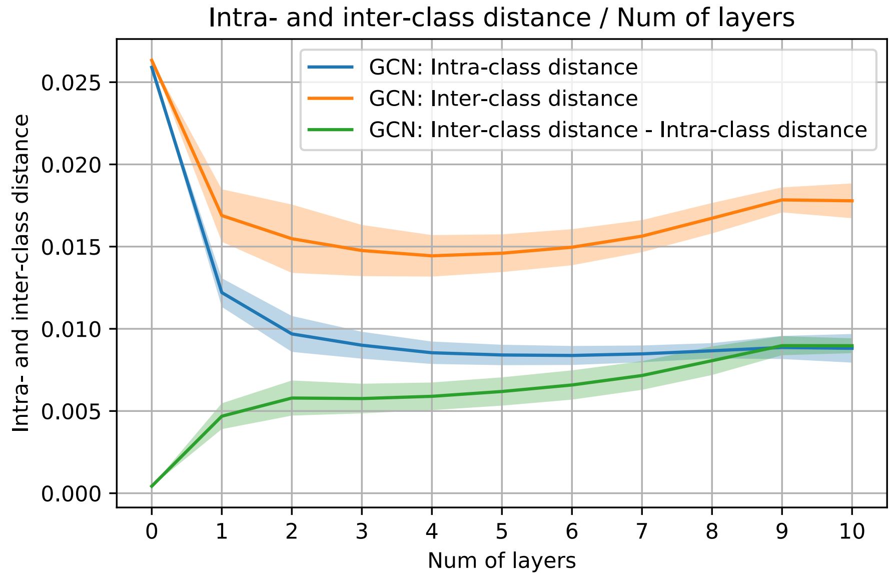
Q1: *Does increasing depth really impair the expressive power of GCNs?*

- Over-smoothing [1] : a phenomenon where all node embeddings converge to a single vector after applying multiple graph convolution operations to the node features

$$\mathbf{H}^{(\ell)} = \mathbf{L}\mathbf{H}^{(\ell-1)}, \mathbf{H}^{(0)} = \mathbf{X}$$



$$\mathbf{H}^{(\ell)} = \sigma(\mathbf{L}\mathbf{H}^{(\ell-1)}\mathbf{W}^{(\ell)}), \mathbf{H}^{(0)} = \mathbf{X}$$



[1] Li, Qimai, Zhichao Han, and Xiao-Ming Wu. "Deeper insights into graph convolutional networks for semi-supervised learning." Thirty-Second AAAI conference on artificial intelligence. 2018.

Q1: *Does increasing depth really impair the expressive power of GCNs?*

- [2] takes non-linearity and weight matrices into consideration.
- Notations:
  - Expressive power $d_{\mathcal{M}}(\mathbf{H}^{(\ell)})$ as the distance of node embeddings $\mathbf{H}^{(\ell)}$ to a subspace $\mathcal{M}$ that only has node degree information.
  - λ_L as the second largest eigenvalue of Laplacian, λ_W as the largest singular value of weight matrices
- They show $d_{\mathcal{M}}\left(\mathbf{H}^{(\ell)}\right) \leq (\lambda_L \lambda_W)^{\ell}\, d_{\mathcal{M}}\left(\mathbf{H}^{(0)}\right)$, i.e., the expressive power will be exponentially decreasing (if $\lambda_L \lambda_W < 1$) or increasing (if $\lambda_L \lambda_W > 1$) as the number of layers increases.

[2] Oono, Kenta, and Taiji Suzuki. "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification." *International Conference on Learning Representations*. 2019.

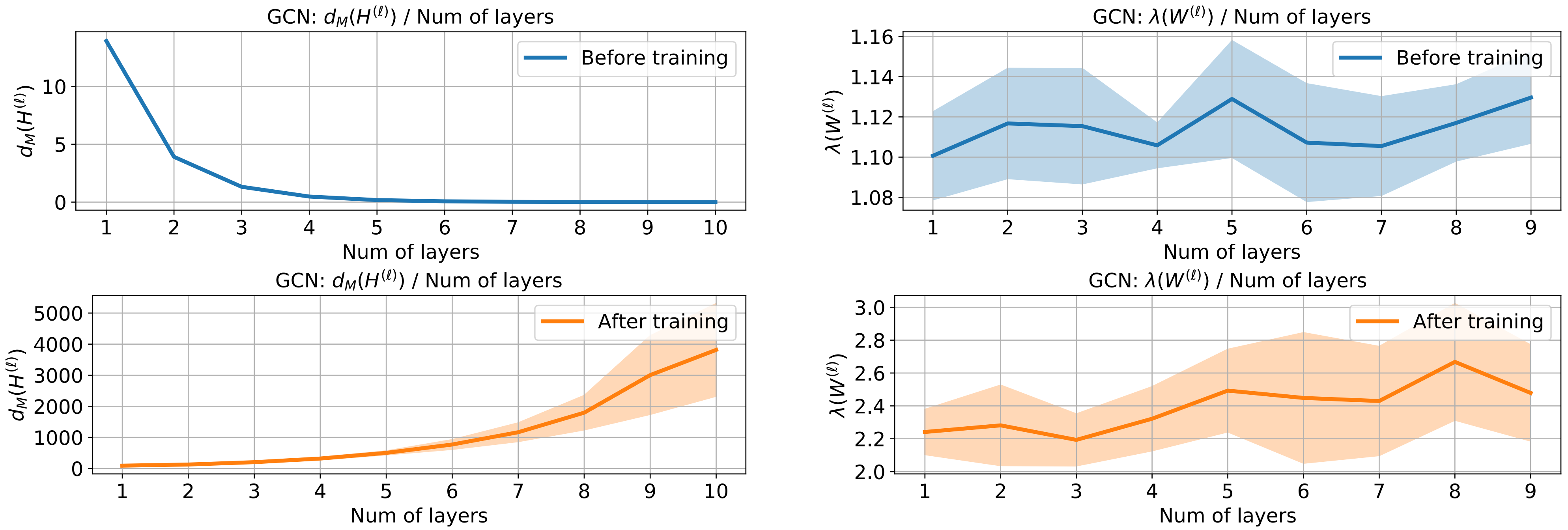
Q1: *Does increasing depth really impair the expressive power of GCNs?*

- However, the above assumption (*i.e.*, $\lambda_L \lambda_W < 1$) not always hold.
- For example,
  - Let assume weight matrices $W^{(\ell)} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$ is initialized by uniform distribution $\mathcal{N}(0, \sqrt{1/d_{\ell-1}})$.
  - By the Gordon's theorem for Gaussian matrices, we know that the expected largest singular value is bounded by $\mathbb{E}[\lambda_W] \leq 1 + \sqrt{d_\ell / d_{\ell-1}}$.
  - This also hold for other initializations.
- Besides, since real-world graphs are sparse, λ_L is close to 1.
  - Cora λ_L=0.9964, Citeseer λ_L=0.9987, PubMed λ_L=0.9905

[2] Oono, Kenta, and Taiji Suzuki. "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification." *International Conference on Learning Representations*. 2019.

Q1: *Does increasing depth really impair the expressive power of GCNs?*

- Besides, we empirically test on real-world dataset



Q1: *Does increasing depth really impair the expressive power of GCNs?*

- Deeper GCNs have stronger expressive power than the shallow GCNs.
  - [3] shows an appropriately trained GCNs is as expressive as 1-WL test
  - An L-layer GCN can encode any different computation tree into different representations.
  - Then, we can characterize the expressiveness of L-layer GCN by the number of computation graphs it can encode

Theorem 1. *Suppose $\mathcal{T}^L$ is a computation tree with binary node features and node degree at least d. Then the richness of the output of L-GCN defined on $\mathcal{T}^L$ is at least $|L\text{-}GCN(\mathcal{T}^L)| \geq 2(d-1)^{L-1}$.*

[3] Morris, Christopher, et al. "Weisfeiler and leman go neural: Higher-order graph neural networks." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. No. 01. 2019.

Q1: *Does increasing depth really impair the expressive power of GCNs?*

- Besides, we provide global convergence of GCNs

Theorem 2. *Let* $\boldsymbol{\theta}_t = \{\mathbf{W}_t^{(\ell)} \in \mathbb{R}^{d_{\ell-1} \times d_\ell}\}_{\ell=1}^{L+1}$ *be the model parameter at the* t*-th iteration and using square loss* $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{H}^{(L)}\mathbf{W}^{(L+1)} - \mathbf{Y}\|_{\mathrm{F}}^2,\ \mathbf{H}^{(\ell)} = \sigma(\mathbf{L}\mathbf{H}^{(\ell-1)}\mathbf{W}^{(\ell)})$ *as objective function. Then, under the condition that* $d_L \geq N$ *we can obtain* $\mathcal{L}(\boldsymbol{\theta}_T) \leq \epsilon$ *if* $T \geq C(L)\log(\mathcal{L}(\boldsymbol{\theta}_0)/\epsilon)$*, where* ϵ *is the desired error and* $C(L)$ *is a function of GCN depth* L *that grows as GCN becomes deeper.*

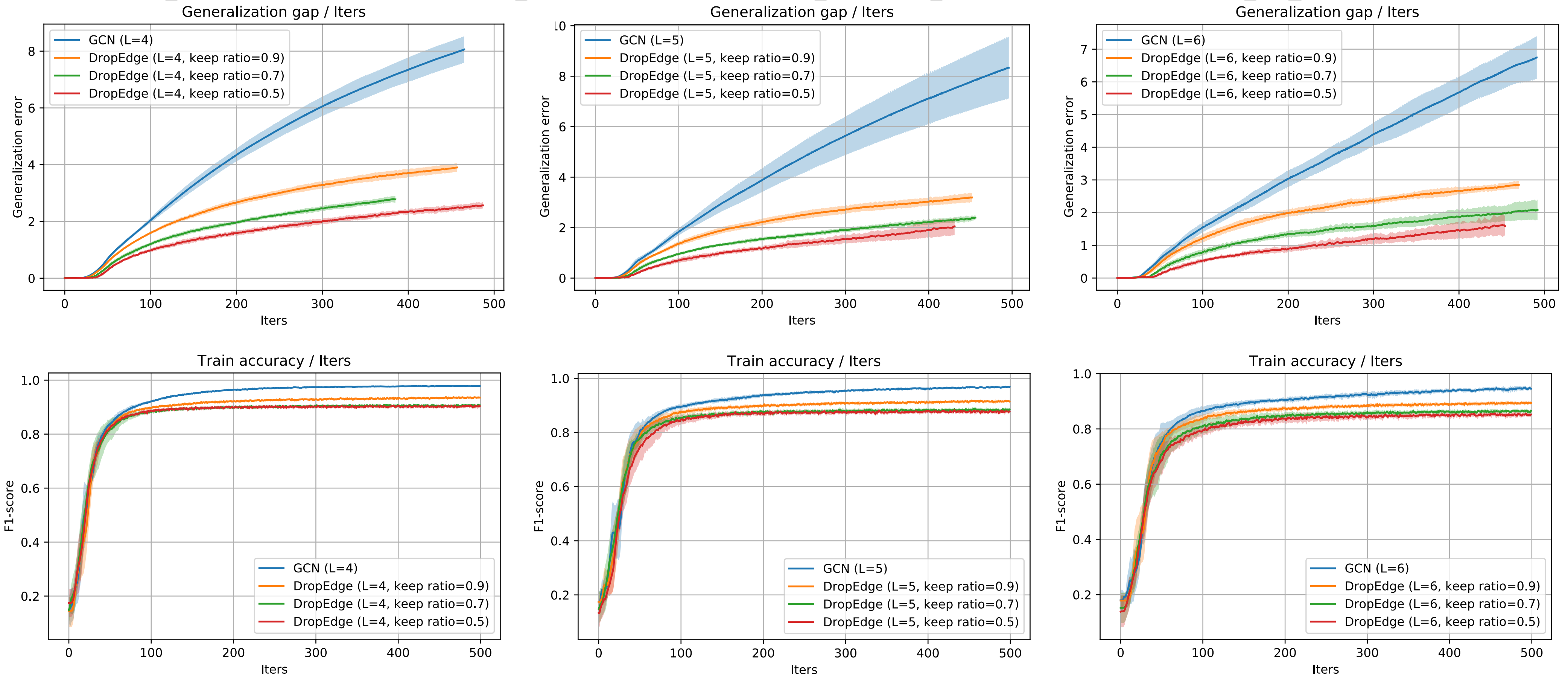
- It is still unclear why a deeper GCN has worse performance than a shallow GCN during the evaluation phase.

Q2: If deep GCN is expressive, then why it generalize poorly?

- To answer this question, we provide a different view by analyzing the impact of GCN structures on the generalization.
- We study the generalization ability of GCNs via *transductive uniform stability*:
  - difference between the training and testing errors for the random partition of a full dataset into training and testing sets.
- Interesting observation:
  - Existing methods that originally designed to alleviate the over-smoothing issue (*e.g., SGC, APPNP, GCNII, DropEdge, PairNorm*) all enjoys a better generalization power than classical GCN.

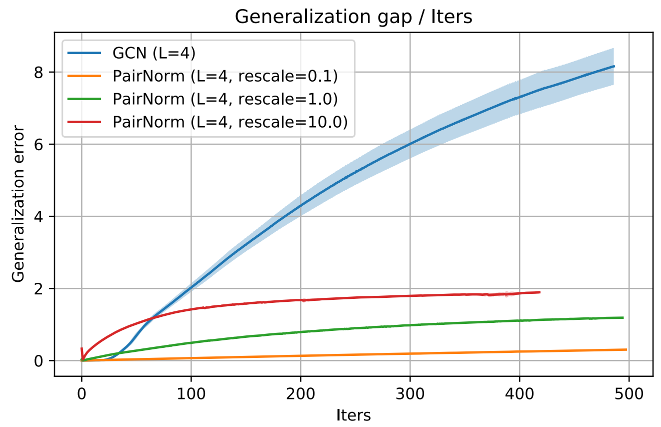
Q2: If deep GCN is expressive, then why it generalize poorly?

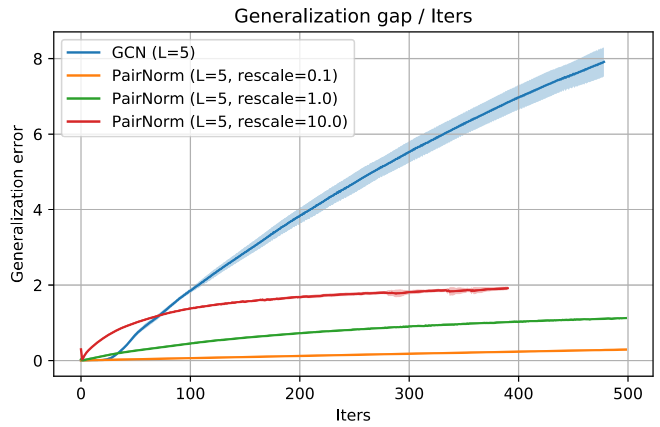
- For example, **DropEdge** is hurting the training accuracy (i.e., not alleviating over-smoothing) but reducing the generalization gap

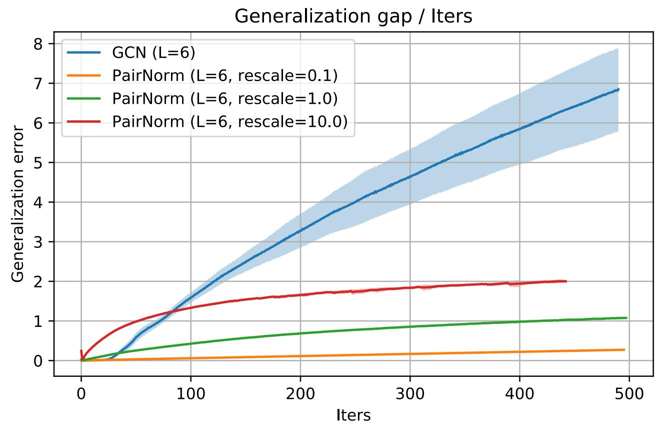


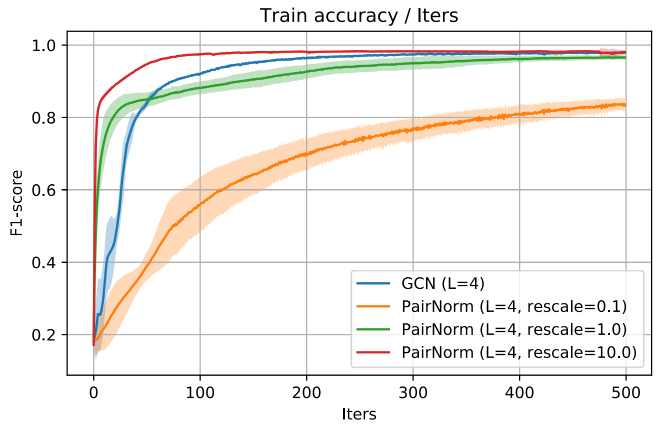
Q2: *If deep GCN is expressive, then why it generalize poorly?*

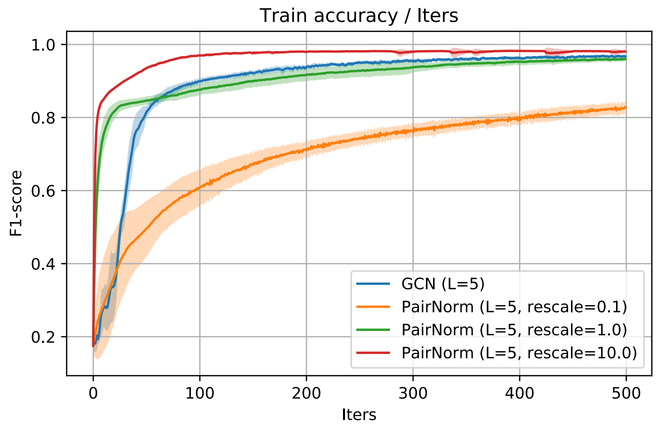
- For example, **PairNorm** is hurting the training accuracy (i.e., not alleviating over-smoothing) but reducing the generalization gap

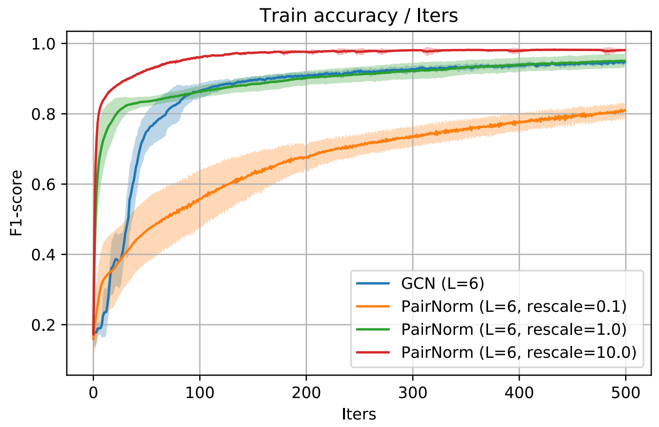












Q2: If GCN is expressive, then why do deep GCNs generalize poorly?

- Informal statement on generalization result

Theorem 4 (Informal)**.** *We say model is ϵ-uniformly stable with $\epsilon = \frac{2\eta\rho_f G_f}{m}\sum_{t=1}^{T}(1+\eta L_f)^{t-1}$ where the result of ρ_f, G_f, L_f are summarized in Table 1, and other related constants as*

$$\begin{aligned} B_d^{\alpha} &= (1-\alpha)\textstyle\sum_{\ell=1}^{L}(\alpha\sqrt{d})^{\ell-1} + (\alpha\sqrt{d})^{L},\ B_w^{\beta} = \beta B_w + (1-\beta), \\ B_{\ell,d}^{\alpha,\beta} &= \max\left\{\beta\big((1-\alpha)L + \alpha\sqrt{d}\big), (1-\alpha)LB_w^{\beta} + 1\right\}. \end{aligned} \tag{1}$$

Table 1: Comparison of uniform stability constant ϵ of GCN variants, where $\mathcal{O}(\cdot)$ is used to hide constants that shared between all bounds.

	ρ_f and G_f	L_f	C_1 and C_2
ϵ_{GCN}	$\mathcal{O}(C_1^L C_2)$	$\mathcal{O}\big(C_1^L C_2\big((L+2)C_1^L C_2 + 2\big)\big)$	$C_1 = \max\{1, \sqrt{d}B_w\},\ C_2 = \sqrt{d}(1+B_x)$
ϵ_{ResGCN}	$\mathcal{O}(C_1^L C_2)$	$\mathcal{O}\big(C_1^L C_2\big((L+2)C_1^L C_2 + 2\big)\big)$	$C_1 = 1 + \sqrt{d}B_w,\ C_2 = \sqrt{d}(1+B_x)$
ϵ_{APPNP}	$\mathcal{O}(C_1)$	$\mathcal{O}\big(C_1\big(C_1C_2\big) + 1\big)$	$C_1 = B_d^{\alpha}B_x, C_2 = \max\{1, B_w\}$
ϵ_{GCNII}	$\mathcal{O}(\beta C_1^L C_2)$	$\mathcal{O}\big(\alpha\beta C_1^L C_2\big((\alpha\beta L + 2)C_1^L C_2 + 2\beta\big)\big)$	$C_1 = \max\{1, \alpha\sqrt{d}B_w^{\beta}\},\ C_2 = \sqrt{d} + B_{\ell,d}^{\alpha,\beta}B_x$
ϵ_{DGCN}	$\mathcal{O}(C_1)$	$\mathcal{O}(C_1(C_1C_2) + 1)$	$C_1 = (\sqrt{d})^L B_x, C_2 = \max\{1, B_w\}$

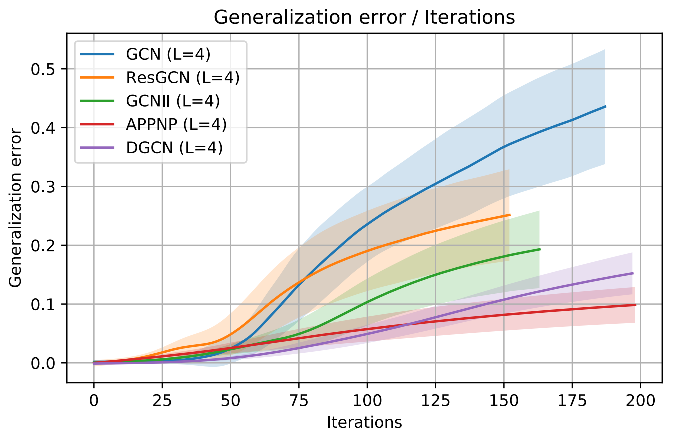
Proposed GNN architecture

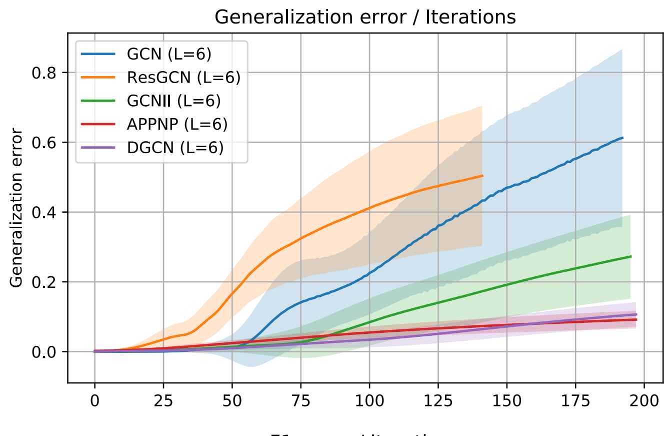
- Based on our generalization analysis, we propose ***Decoupled GCN***, with the following forward propagation rule.
  - α_ℓ, β_ℓ are trainable parameters
  - $\mathbf{P} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and $\mathbf{P}^\ell$ stands for $\mathbf{P}$ to the power of ℓ

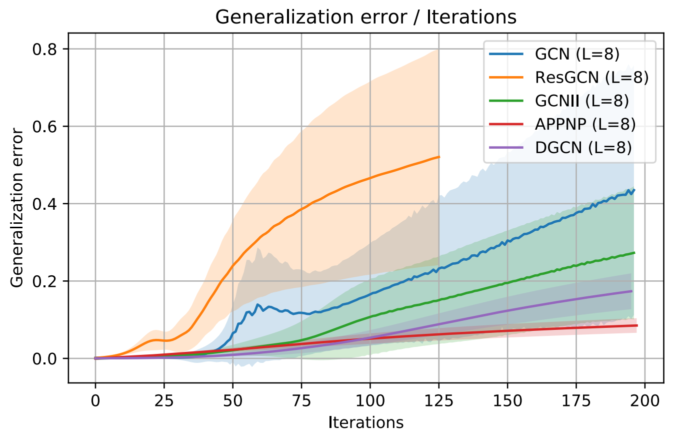
$$\mathbf{Z} = \sum_{\ell=1}^{L} \alpha_\ell f^{(\ell)}(\mathbf{X}),\ \ f^{(\ell)}(\mathbf{X}) = \mathbf{P}^\ell \mathbf{X} \left(\beta_\ell \mathbf{W}^{(\ell)} + (1-\beta_\ell)\mathbf{I}\right)$$

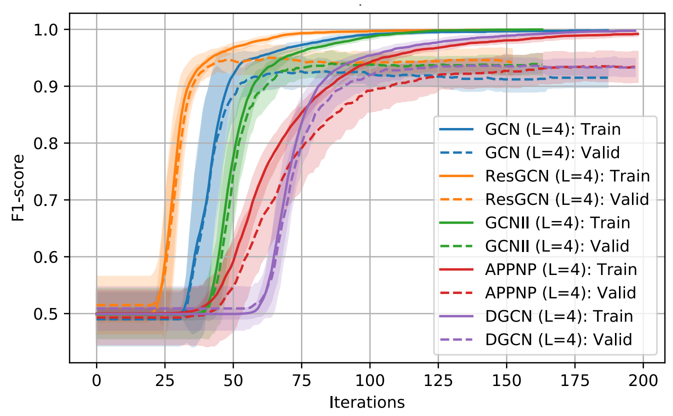
Empirical validation

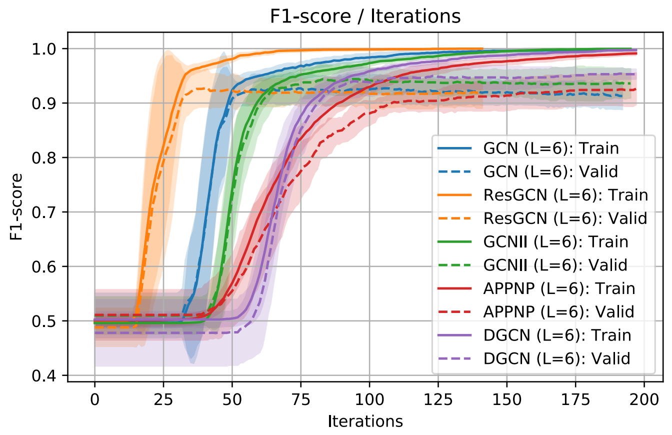
- Validate the correctness of the theoretical results on synthetic dataset

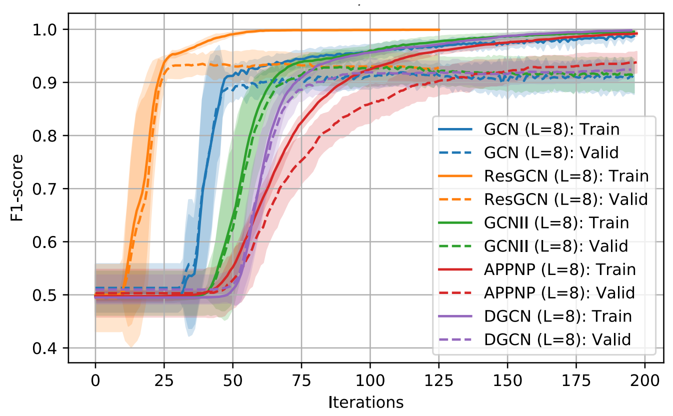












Empirical validation

- Validate the effectiveness of our model on real-world dataset

Table 2: Comparison of F1-score on OGB dataset.

%	**Products**	**Proteins**	**Arvix**
GCN	75.39 ± 0.21	71.66 ± 0.48	71.56 ± 0.19
ResGCN	75.53 ± 0.12	74.50 ± 0.41	72.56 ± 0.31
APPNP	66.35 ± 0.10	71.78 ± 0.29	68.02 ± 0.55
GCNII	$71.93 \pm 0.35^{\dagger}$	75.60 ± 0.47	$72.57 \pm 0.23^{\ddagger}$
DGCN	76.09 ± 0.29	75.45 ± 0.24	72.63 ± 0.12

Table 3: Comparison of F1-score on OGB-Arxiv dataset for different number of layers

Model	α	**2 Layers**	**4 Layers**	**8 Layers**	**12 Layers**	**16 Layers**
GCN	–	$71.02\% \pm 0.14$	$71.56\% \pm 0.19$	$71.28\% \pm 0.33$	$70.28\% \pm 0.23$	$69.37\% \pm 0.46$
ResGCN	–	$70.66\% \pm 0.48$	$72.41\% \pm 0.31$	$72.56\% \pm 0.31$	$72.46\% \pm 0.23$	$72.11\% \pm 0.28$
GCNII	0.9	$71.35\% \pm 0.21$	$72.57\% \pm 0.23$	$72.06\% \pm 0.42$	$71.31\% \pm 0.62$	$69.99\% \pm 0.80$
GCNII	0.8	$71.14\% \pm 0.27$	$72.32\% \pm 0.19$	$71.90\% \pm 0.41$	$71.21\% \pm 0.23$	$70.56\% \pm 0.72$
GCNII	0.5	$70.54\% \pm 0.30$	$72.09\% \pm 0.25$	$71.92\% \pm 0.32$	$71.24\% \pm 0.47$	$71.02\% \pm 0.58$
APPNP	0.9	$67.38\% \pm 0.34$	$68.02\% \pm 0.55$	$66.62\% \pm 0.48$	$67.43\% \pm 0.50$	$67.42\% \pm 1.00$
APPNP	0.8	$66.71\% \pm 0.32$	$68.25\% \pm 0.43$	$66.40\% \pm 0.89$	$66.51\% \pm 2.09$	$66.56\% \pm 0.74$
DGCN	–	$71.21\% \pm 0.25$	$72.29\% \pm 0.18$	$72.39\% \pm 0.21$	$\mathbf{72.63}\% \pm \mathbf{0.12}$	$72.41\% \pm 0.07$