GNN: Over-smoothing

Kei Ishikawa
Motivation: The deeper, the better.

Reminder: Graph Neural Network (GNN)

- One layer GNN = Neighborhood Aggregation + Node-wise Neural Network
What is "Over-smoothing"?

- As the model gets deeper, node features become similar everywhere.
Agenda

- Understanding Over-smoothing
  - Experimental Evidence
  - Theoretical Analysis
- Solutions to Over-smoothing
  - Residual Connection
  - Graph Sparsification
Agenda

● Understanding Over-smoothing
  ○ Experimental Evidence
  ○ Theoretical Analysis
● Solutions to Over-smoothing
  ○ Residual Connection
  ○ Graph Sparsification
Experimental Evidence (Chen et al [1])

The node classification accuracy (Acc) of GCNs on the CORA dataset.
A measure of similarity of node features.

\[
\text{MAD} = \frac{1}{|V|^2} \sum_{v,u \in V} \left( 1 - \frac{x_v^T x_u}{||x_v||_2 \cdot ||x_u||_2} \right)
\]

- **MAD**: Mean Average Distance
- **\( V \)**: the vertex set of the graph \((V, E)\).
- **\( x_v \)**: the node feature of the node \( v \).

The MAD (mean average distance) values of various GNNs with different layers on the CORA dataset
Agenda

- Understanding Over-smoothing
  - Experimental Evidence
  - Theoretical Analysis
- Solutions to Over-smoothing
  - Residual Connection
  - Graph Sparsification
Understanding Over-smoothing: Theoretical Analysis (Oono and Suzuki [2])

Published as a conference paper at ICLR 2020

Graph Neural Networks \textit{exponentially} lose expressive power for node classification

\textbf{Kenta Oono}\textsuperscript{1,2,*}, Taiji Suzuki\textsuperscript{1,3}
\{kenta.oono, taiji\}@mist.i.u-tokyo.ac.jp
\textsuperscript{1}The University of Tokyo \textsuperscript{2}Preferred Networks, Inc. \textsuperscript{*}Work at the University of Tokyo
\textsuperscript{3}RIKEN Center for Advanced Intelligence Project (AIP)
Notations and Model Assumptions

- Node feature matrix

\[ X = \begin{bmatrix}
  -x_1^T \\
  -x_2^T \\
  \vdots \\
  -x_N^T
\end{bmatrix} \]

where \( x_1, \ldots, x_N \in \mathbb{R}^C \) are node feature vectors and \( N = \text{(\#nodes)} \).

- Augmented adjacency/degree matrix

\[ \tilde{A} = A + I, \quad \tilde{D} = D + I \]

- GCN \( f : \mathbb{R}^{N \times C} \rightarrow \mathbb{R}^{N \times C} \)

\[ f = f_L \circ \cdots \circ f_1 \]

\[ f_i(X) = \sigma(\underbrace{PXW_i}_{\text{convolved node features}}) \]

where

- \( \sigma \) is Relu activation,
- \( W_i \) is a weight matrix,
- \( P = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \in \mathbb{R}^{N \times N} \) is the graph convolution matrix.
Exponential Convergence of Node Features

- Theorem
  - Assume for simplicity that \( G = (V, E) \) is connected graph and the node degrees are same for all nodes.
  - Let \( \mathcal{M} \subseteq \mathbb{R}^{N \times C} \) be the linear subspace where all row vectors are equivalent.

\[
\mathcal{M} = \left\{ \begin{bmatrix} a_1 & a_2 & \cdots & a_C \\ a_1 & a_2 & \cdots & a_C \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \cdots & a_C \end{bmatrix} : a \in \mathbb{R}^C \right\} \subseteq \mathbb{R}^{N \times C}.
\]

- Let \( \{X_t\}_{t \in \mathbb{N}} \) be the sequence of node features defined by \( X_t = f_t(X_{t-1}) \left( := \sigma(PX_{t-1}W_t) \right) \).
- Then,

\[
\text{dist}(X_t, \mathcal{M}) = \inf_{Y \in \mathcal{M}} \|X_t - Y\|_{\text{Frob}} < C^t
\]

for some constant \( 0 \leq C < 1 \) (under some conditions).
Intuition of Exponential Decay

- \( \text{dist}(X_l, \mathcal{M}) = \inf_{Y \in \mathcal{M}} \|X_l - Y\|_{\text{Frob}} < C^l \)
Analogy to Power Iteration

- Ignore the $\sigma(\cdot)$ and $W$ from $f_\ell(X) = \sigma(PXW_\ell)$, and assume $C=1$

  $\Rightarrow$ Power iteration to find the largest eigenvector of $P$, 

  $$X_\ell = PX_{\ell-1}$$

  $X_\ell$ approaches to the eigenspace of largest eigenvalue of $P$. 

  $\mathcal{M} \in \mathbb{R}^N$
Agenda

- Understanding Over-smoothing
  - Experimental Evidence
  - Theoretical Analysis
- Solutions to Over-smoothing
  - Residual Connection
  - Graph Sparsification
Solution to Over-smoothing: Residual Connection (Chen et al [3])

Simple and Deep Graph Convolutional Networks

Ming Chen¹  Zhewei Wei² ³ ⁴  Zengfeng Huang⁵  Bolin Ding⁶  Yaliang Li⁶

Abstract

Graph convolutional networks (GCNs) are a powerful deep learning approach for graph-structured data. Recently, GCNs and subsequent variants have shown superior performance in various application areas on real-world datasets. Despite their success, most of the current GCN models are shallow, due to the over-smoothing problem. In this paper, we study the problem of designing and analyzing deep graph convolutional networks. We propose the GCNIL, an extension of the vanilla GCN model with two simple yet effective techniques: initial residual and identity mapping. We provide theoretical and empirical evidence that the two techniques effectively alleviate the problem of over-smoothing. Our experiments show that the deep GCNIL model outperforms the state-of-the-art methods on various semi- and full-supervised tasks. Code is available at https://github.com/chenm94/GCNIL.

Despite their enormous success, most of the current GCN models are shallow. Most of the recent models, such as GCN (Kipf & Welling, 2017) and GAT (Velicković et al., 2018), achieve their best performance with 2-layer models. Such shallow architectures limit their ability to extract information from high-order neighbors. However, stacking more layers and adding non-linearity tends to degrade the performance of these models. Such a phenomenon is called over-smoothing (Li et al., 2018b), which suggests that as the number of layers increases, the representations of the nodes in GCN are inclined to converge to a certain value and thus become indistinguishable. ResNet (He et al., 2016) solves a similar problem in computer vision with residual connections, which is effective for training very deep neural networks. Unfortunately, adding residual connections in the GCN models merely slows down the over-smoothing problem (Kipf & Welling, 2017); deep GCN models are still outperformed by 2-layer models such as GCN or GAT.

Recently, several works try to tackle the problem of over-
Solution to Over-smoothing: Residual Connection (Chen et al [3])

- GCN
  \[ H^{(\ell+1)} = \sigma \left( \tilde{P} H^{(\ell)} W^{(\ell)} \right) \]

- GCNII = GCN + Initial residual connection + Identity mapping
  \[ H^{(\ell+1)} = \sigma \left( (1-\alpha_\ell) \tilde{P} H^{(\ell)} + \alpha_\ell H^{(0)} \right) \left( (1-\beta_\ell) I_n + \beta_\ell W^{(\ell)} \right) \]

where \( \tilde{P} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \) is the graph convolution matrix and \( W \) is weight matrix. \( \alpha_\ell \)'s and \( \beta_\ell \)'s are hyperparameters.
Solution to Over-smoothing: Residual Connection (Chen et al [3])

Table 5. Mean classification accuracy of full-supervised node classification.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>85.77</td>
<td>73.68</td>
<td>88.13</td>
<td>28.18</td>
<td>52.70</td>
<td>52.16</td>
<td>45.88</td>
</tr>
<tr>
<td>GAT</td>
<td>86.37</td>
<td>74.32</td>
<td>87.62</td>
<td>42.93</td>
<td>54.32</td>
<td>58.38</td>
<td>49.41</td>
</tr>
<tr>
<td>Geom-GCN-I</td>
<td>85.19</td>
<td>77.99</td>
<td>90.05</td>
<td>60.31</td>
<td>56.76</td>
<td>57.58</td>
<td>58.24</td>
</tr>
<tr>
<td>Geom-GCN-P</td>
<td>84.93</td>
<td>75.14</td>
<td>88.09</td>
<td>60.90</td>
<td>60.81</td>
<td>67.57</td>
<td>64.12</td>
</tr>
<tr>
<td>Geom-GCN-S</td>
<td>85.27</td>
<td>74.71</td>
<td>84.75</td>
<td>59.96</td>
<td>55.68</td>
<td>59.73</td>
<td>56.67</td>
</tr>
<tr>
<td>APPNP</td>
<td>87.87</td>
<td>76.53</td>
<td>89.40</td>
<td>54.3</td>
<td>73.51</td>
<td>65.41</td>
<td>69.02</td>
</tr>
<tr>
<td>JKNNet</td>
<td>85.25 (16)</td>
<td>75.85 (8)</td>
<td>88.94 (64)</td>
<td>60.07 (32)</td>
<td>57.30 (4)</td>
<td>56.49 (32)</td>
<td>48.92 (8)</td>
</tr>
<tr>
<td>JKNNet(Drop)</td>
<td>87.46 (16)</td>
<td>75.96 (8)</td>
<td>89.45 (64)</td>
<td>62.08 (32)</td>
<td>61.08 (4)</td>
<td>57.30 (32)</td>
<td>50.59 (8)</td>
</tr>
<tr>
<td>Incep(Drop)</td>
<td>86.86 (8)</td>
<td>76.83 (8)</td>
<td>89.18 (4)</td>
<td>61.71 (8)</td>
<td>61.62 (16)</td>
<td>57.84 (8)</td>
<td>50.20 (8)</td>
</tr>
<tr>
<td>GCNII*</td>
<td>88.49 (64)</td>
<td>77.08 (64)</td>
<td>89.57 (64)</td>
<td>60.61 (8)</td>
<td>74.86 (16)</td>
<td>69.46 (32)</td>
<td>74.12 (16)</td>
</tr>
<tr>
<td>GCNII*</td>
<td>88.01 (64)</td>
<td>77.13 (64)</td>
<td>90.30 (64)</td>
<td>62.48 (8)</td>
<td>76.49 (16)</td>
<td>77.84 (32)</td>
<td>81.57 (16)</td>
</tr>
</tbody>
</table>

GCNII* uses different weight matrix for initial residual connection as

\[ H^{(\ell+1)} = \sigma \left( (1 - \alpha_\ell) \tilde{P} H^{(\ell)} \left( (1 - \beta_\ell) I_n + \beta_\ell W^{(\ell)} \right) + \alpha_\ell H^{(0)} \left( (1 - \beta_\ell) I_n + \beta_\ell W^{(0)} \right) \right) \]
**Solution to Over-smoothing:**

**Residual Connection (Chen et al [3])**

- Initial residual connection lets you go deeper.
- Identity mapping improves performance.

\[
H^{(\ell+1)} = \sigma \left( \left( (1-\alpha_\ell)\hat{P}H^{(\ell)} + \alpha_\ell H^{(0)} \right) (1-\beta_\ell)I_n + \beta_\ell W^{(\ell)} \right)
\]

- Initial residual connection
- Identity mapping
Agenda

- Understanding Over-smoothing
  - Experimental Evidence
  - Theoretical Analysis
- Solutions to Over-smoothing
  - Residual Connection
  - Graph Sparsification
Solution to Over-smoothing: Graph Sparsification

(a) Original Subgraph  
(b) Sparsified Subgraph
Solution to Over-smoothing: Graph Sparsification (Rong et al [4])

Published as a conference paper at ICLR 2020

DROPEdge: Towards Deep Graph Convolutional Networks on Node Classification

Yu Rong\textsuperscript{1}, Wenbing Huang\textsuperscript{2}, Tingyang Xu\textsuperscript{1}, Junzhou Huang\textsuperscript{1}
\textsuperscript{1} Tencent AI Lab
\textsuperscript{2} Beijing National Research Center for Information Science and Technology (BNRist), State Key Lab on Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University
yu.rong@hotmail.com, wenbing@126.com
tingyangxu@tencent.com, jzhuang@uta.edu

ABSTRACT

Over-fitting and over-smoothing are two main obstacles of developing deep Graph Convolutional Networks (GCNs) for node classification. In particular, over-fitting weakens the generalization ability on small dataset, while over-smoothing impedes model training by isolating output representations from the input features with the increase in network depth. This paper proposes DropEdge, a novel and flexible technique to alleviate both issues. At its core, DropEdge randomly removes a certain number of edges from the input graph at each training epoch, acting like a data augmenter and also a message passing reducer. Furthermore, we theoretically demonstrate that DropEdge either reduces the convergence speed of over-smoothing or relieves the information loss caused by it. More importantly, our DropEdge is a general skill that can be equipped with many other backbone models (e.g. GCN, ResGCN, GraphSAGE, and JKNet) for enhanced performance. Extensive experiments on standard benchmarks verify that DropEdge is dexterously powerful.
Solution to Over-smoothing: Graph Sparsification (Rong et al [4])

Table 1: Testing accuracy (%) comparisons on different backbones w and w/o DropEdge.

| Dataset | Backbone | 2 layers | | | 8 layers | | | 32 layers | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Original | DropEdge | Original | DropEdge | Original | DropEdge | Original | DropEdge |
| Cora | GCN | 86.10 | 86.50 | 78.70 | 85.80 | 71.60 | 74.60 |  |
| | ResGCN | - | - | 85.40 | 86.90 | 85.10 | 86.80 |  |
| | JKNet | - | - | 86.70 | 87.80 | 87.10 | 87.60 |  |
| | IncepGCN | - | - | 86.70 | 88.20 | 87.40 | 87.70 |  |
| | GraphSAGE | 87.80 | 88.10 | 84.30 | 87.10 | 31.90 | 32.20 |  |
| Citeseer | GCN | 75.90 | 78.70 | 74.60 | 77.20 | 59.20 | 61.40 |  |
| | ResGCN | - | - | 77.80 | 78.80 | 74.40 | 77.90 |  |
| | JKNet | - | - | 79.20 | 80.20 | 71.70 | 80.00 |  |
| | IncepGCN | - | - | 79.60 | 80.50 | 72.60 | 80.30 |  |
| | GraphSAGE | 78.40 | 80.00 | 74.10 | 77.10 | 37.00 | 53.60 |  |
| Pubmed | GCN | 90.20 | 91.20 | 90.10 | 90.90 | 84.60 | 86.20 |  |
| | ResGCN | - | - | 89.60 | 90.50 | 90.20 | 91.10 |  |
| | JKNet | - | - | 90.60 | 91.20 | 89.20 | 91.30 |  |
| | IncepGCN | - | - | 90.20 | 91.50 | OOM | 90.50 |  |
| | GraphSAGE | 90.10 | 90.70 | 90.20 | 91.70 | 41.30 | 47.90 |  |
| Reddit | GCN | 96.11 | 96.13 | 96.17 | 96.48 | 45.55 | 50.51 |  |
| | ResGCN | - | - | 96.37 | 96.46 | 93.93 | 94.27 |  |
| | JKNet | - | - | 96.82 | 97.02 | OOM | OOM |  |
| | IncepGCN | - | - | 96.43 | 96.87 | OOM | OOM |  |
| | GraphSAGE | 96.22 | 96.28 | 96.38 | 96.42 | 96.43 | 96.47 |  |
Solution to Over-smoothing: Graph Sparsification (Hasanzadeh et al [5])

drop edges in a smarter way

Bayesian Graph Neural Networks with Adaptive Connection Sampling

Abstract

We propose a unified framework for adaptive connection sampling in graph neural networks (GNNs) that generalizes existing stochastic regularization methods for training GNNs. The proposed framework not only alleviates over-smoothing and over-fitting tendencies of deep GNNs, but also enables learning with uncertainty in graph analytic tasks with GNNs. Instead of using fixed sampling rates or hand-tuning them, two major limitations: 1) they cannot go very deep due to over-smoothing and over-fitting phenomena (Li et al., 2018; Kipf & Welling, 2017); 2) the current implementations of GNNs do not provide uncertainty quantification (UQ) of output predictions.

There exist a variety of methods to address these problems. For example, DropOut (Srivastava et al., 2014) is a popular regularisation technique with deep neural networks (DNNs) to avoid over-fitting, where network units are randomly masked during training. In GNNs, DropOut is realised by
Solution to Over-smoothing: Graph Sparsification (Hasanzadeh et al [5])

- DropEdge (layer wise)

\[ H^{(l+1)} = \sigma \left( \mathcal{M}(A \odot Z^{(l)}) H^{(l)} W^{(l)} \right) \]

- Graph Drop Connect (layer & channel wise)

\[ H^{(l+1)}[:, j] = \sigma \left( \sum_{i=1}^{f_l} \mathcal{M}(A \odot Z^{(l)}_{i,j}) H^{(l)}[:, i] W^{(l)}[i, j] \right), \]

for \( j = 1, \ldots, f_{l+1} \) \hspace{1cm} (4)
Solution to Over-smoothing: Graph Sparsification (Hasanzadeh et al [5])

- Hierarchical prior for drop rate
  
  $z_e^{(l)} \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(\pi_l)$
  
  $\pi_l \overset{\text{i.i.d.}}{\sim} \text{Beta} \left( \frac{c}{L}, \frac{c(L - 1)}{L} \right)$

- Learn the drop rate $\pi_l$ of edges as posterior inference

  $\Leftrightarrow$ DropEdge as Bayesian approximation of $W_{ji}^{(l)} \overset{\text{i.i.d.}}{\sim} N(0, I)$
Solution to Over-smoothing: Graph Sparsification (Hasanzadeh et al [5])

Table 1. Semi-supervised node classification accuracy of GCNs with our adaptive connection sampling and baseline methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cora</th>
<th>Cora-ML</th>
<th>Citeseer</th>
<th>Citeseer-ML</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 layers</td>
<td>4 layers</td>
<td>2 layers</td>
<td>4 layers</td>
</tr>
<tr>
<td>GCN-DO</td>
<td>80.98 ± 0.48</td>
<td>78.24 ± 2.4</td>
<td>70.44 ± 0.39</td>
<td>64.38 ± 0.90</td>
</tr>
<tr>
<td>GCN-DE</td>
<td>78.36 ± 0.92</td>
<td>73.40 ± 2.07</td>
<td>70.52 ± 0.75</td>
<td>57.14 ± 0.90</td>
</tr>
<tr>
<td>GCN-DO-DE</td>
<td>80.58 ± 1.19</td>
<td>79.20 ± 1.07</td>
<td>70.74 ± 1.23</td>
<td>64.84 ± 0.98</td>
</tr>
<tr>
<td>GCN-BBDE</td>
<td>81.58 ± 0.49</td>
<td>80.42 ± 0.25</td>
<td>71.46 ± 0.55</td>
<td>68.58 ± 0.88</td>
</tr>
<tr>
<td>GCN-BBGDC</td>
<td>81.80 ± 0.99</td>
<td>82.20 ± 0.92</td>
<td>71.72 ± 0.48</td>
<td>70.00 ± 0.36</td>
</tr>
</tbody>
</table>

- GCN-DO: Dropout
- GCN-DE: DropEdge
- GCN-BBDE: Beta-Bernoulli DropEdge (layer-wise dropping)
- GCN-BBGDC: Beta-Bernoulli Graph Drop Connection (layer & channel-size dropping)
Solution to Over-smoothing: Graph Sparsification (Zheng et al [6])

drop edges in a smarter way using NNs

---

Robust Graph Representation Learning via **Neural Sparsification**

---

Cheng Zheng¹, Bo Zong², Wei Cheng², Dongjin Song², Jingchao Ni², Wenchao Yu², Haifeng Chen², Wei Wang¹

**Abstract**

Graph representation learning serves as the core of important prediction tasks, ranging from product recommendation to fraud detection. Real-life graphs usually have complex information in the local neighborhood, where each node is described by a rich set of features and connects to dozens or even hundreds of neighbors. Despite the success of neighborhood aggregation in graph neural networks, task-irrelevant information is mixed into nodes’ neighborhood, making learned models suffer from sub-optimal generalization performance. In this paper, we

...
Solution to Over-smoothing: Graph Sparsification (Zheng et al [6])

(a) Node Features
(b) With Task-irrelevant Edges
(c) By DropEdge
(d) By NeuralSparse
Consider a weighted objective

\[ P(Y|G) \approx \sum_{g \in \mathcal{S}_G} Q_\theta(Y|g)Q_\phi(g|G) = \mathbb{E}_{g \sim Q_\phi(g|G)} [Q_\theta(Y|g)] \]

- \( g \in \mathcal{S}_G \) is the latent variable
- \( \mathcal{S}_G \) is k-neighbor subgraphs of \( G \).
- \( Q_\theta(Y|g) \) is GNN model.
- \( Q_\phi(g|G) \) is realized by sampling from edge weight model \( z_{u,v} = \text{MLP}_\phi(V(u), V(v), A(u,v)) \).

Solution to Over-smoothing: Graph Sparsification (Zheng et al [6])

[Diagram of the process involving Graph, Sparsification Network, Sparsified Graph, GNN, and Classification Results]
## Solution to Over-smoothing: Graph Sparsification (Zheng et al [6])

<table>
<thead>
<tr>
<th>Sparsifier</th>
<th>Method</th>
<th>Reddit Micro-F1</th>
<th>Reddit Micro-F1</th>
<th>PPI AUC</th>
<th>Transaction AUC</th>
<th>Cora Accuracy</th>
<th>Citeseer Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>GCN</td>
<td>0.922 ± 0.041</td>
<td>0.532 ± 0.024</td>
<td>0.564 ± 0.018</td>
<td>0.810 ± 0.027</td>
<td>0.694 ± 0.020</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GraphSAGE</td>
<td>0.938 ± 0.029</td>
<td>0.600 ± 0.027</td>
<td>0.574 ± 0.029</td>
<td>0.825 ± 0.033</td>
<td>0.710 ± 0.020</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>-</td>
<td>0.973 ± 0.030</td>
<td>0.616 ± 0.022</td>
<td>0.821 ± 0.043</td>
<td>0.721 ± 0.037</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIN</td>
<td>0.928 ± 0.022</td>
<td>0.703 ± 0.028</td>
<td>0.607 ± 0.031</td>
<td>0.816 ± 0.020</td>
<td>0.709 ± 0.037</td>
<td></td>
</tr>
<tr>
<td>DropEdge</td>
<td>GCN</td>
<td>0.961 ± 0.040</td>
<td>0.548 ± 0.041</td>
<td>0.591 ± 0.040</td>
<td>0.828 ± 0.035</td>
<td>0.723 ± 0.043</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GraphSAGE</td>
<td>0.963 ± 0.043</td>
<td>0.632 ± 0.031</td>
<td>0.598 ± 0.043</td>
<td>0.821 ± 0.048</td>
<td>0.712 ± 0.032</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>-</td>
<td>0.851 ± 0.030</td>
<td>0.604 ± 0.043</td>
<td>0.789 ± 0.039</td>
<td>0.691 ± 0.039</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIN</td>
<td>0.931 ± 0.031</td>
<td>0.783 ± 0.037</td>
<td>0.625 ± 0.035</td>
<td>0.818 ± 0.044</td>
<td>0.715 ± 0.039</td>
<td></td>
</tr>
<tr>
<td>Neural Sparse</td>
<td>GCN</td>
<td>0.966 ± 0.020</td>
<td>0.651 ± 0.014</td>
<td>0.610 ± 0.022</td>
<td>0.837 ± 0.014</td>
<td><strong>0.741 ± 0.014</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GraphSAGE</td>
<td><strong>0.967 ± 0.015</strong></td>
<td>0.696 ± 0.023</td>
<td>0.649 ± 0.018</td>
<td>0.841 ± 0.024</td>
<td>0.736 ± 0.013</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>-</td>
<td><strong>0.986 ± 0.015</strong></td>
<td><strong>0.671 ± 0.018</strong></td>
<td><strong>0.842 ± 0.015</strong></td>
<td>0.736 ± 0.026</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIN</td>
<td>0.959 ± 0.027</td>
<td>0.892 ± 0.015</td>
<td>0.634 ± 0.023</td>
<td>0.838 ± 0.027</td>
<td>0.738 ± 0.015</td>
<td></td>
</tr>
</tbody>
</table>
Summary

- Understanding Over-smoothing
  - Experimental Evidence
  - Theoretical Analysis
- Solutions to Over-smoothing
  - Residual Connection
  - Graph Sparsification
References


