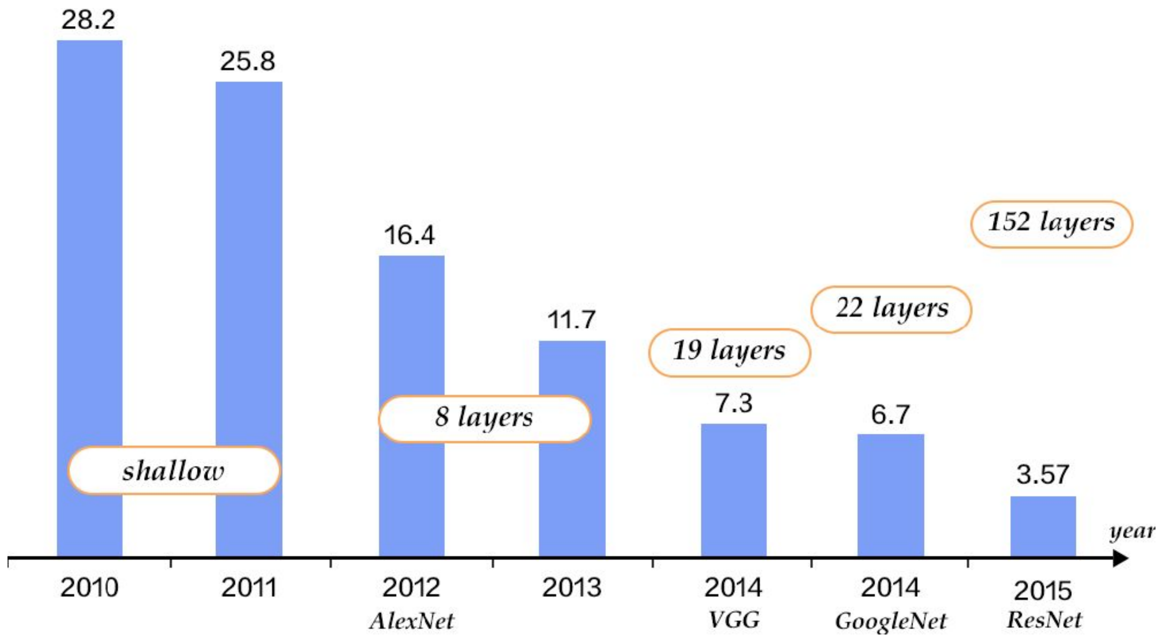
GNN: Over-smoothing

Kei Ishikawa

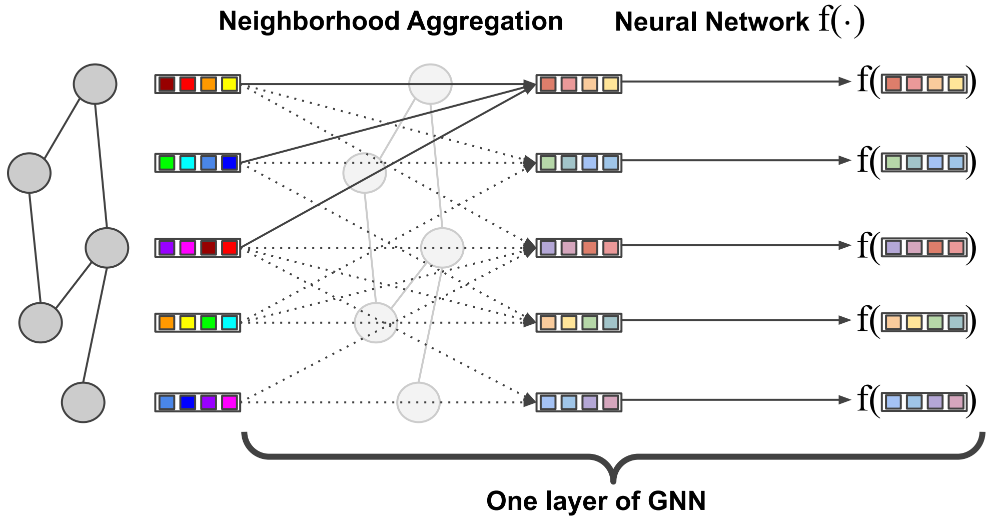
Motivation: The deeper, the better.



Top-5 error of the winners of the ImageNet Challenge. Image source http://paddlepaddle.org/.

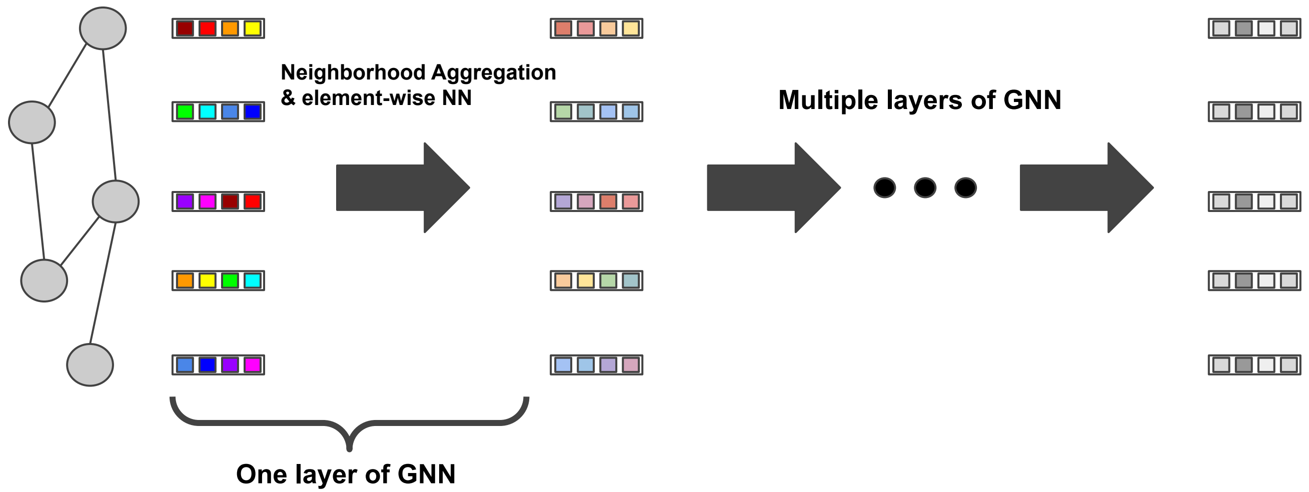
Reminder: Graph Neural Network (GNN)

- One layer GNN = Neighborhood Aggregation + Node-wise Neural Network



What is "Over-smoothing"?

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



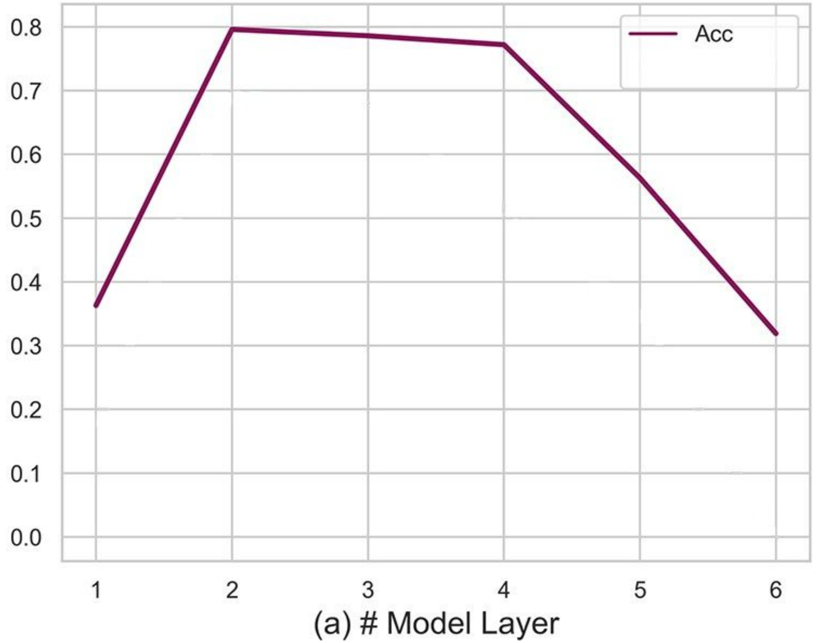
Agenda

- 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])



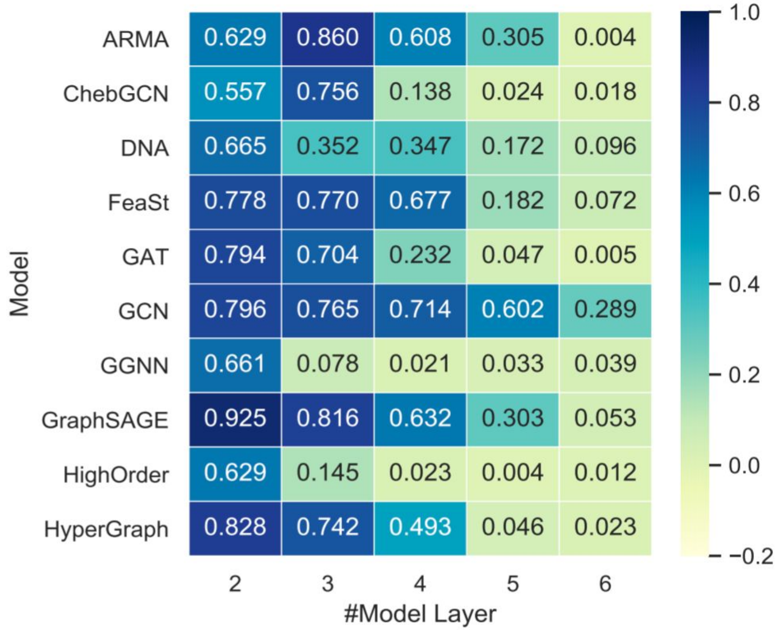
The node classification accuracy (Acc) of GCNs on the CORA dataset.

Similarity of Node Features

- A measure of similarity of node features.

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

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



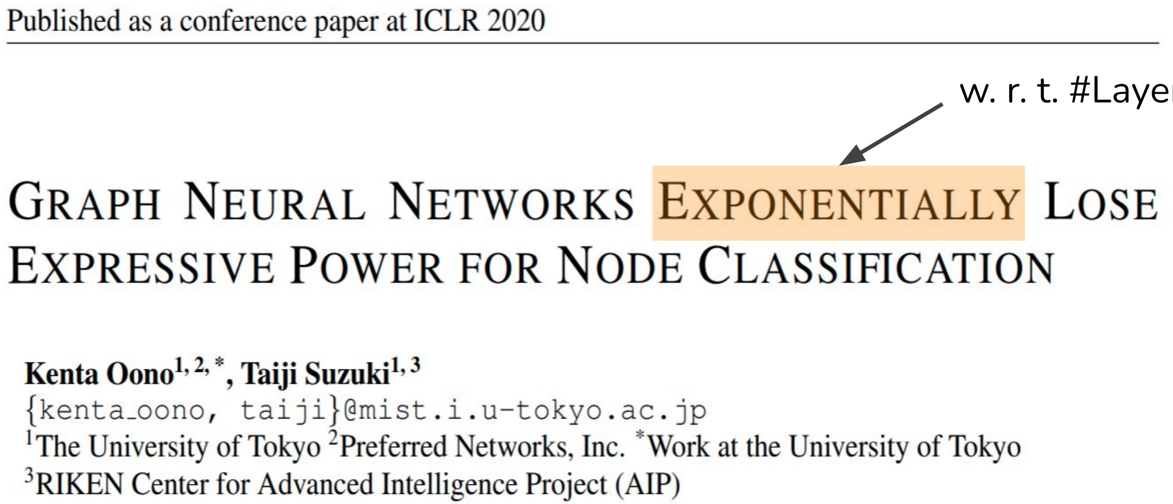
Model	2	3	4	5	6
ARMA	0.629	0.860	0.608	0.305	0.004
ChebGCN	0.557	0.756	0.138	0.024	0.018
DNA	0.665	0.352	0.347	0.172	0.096
FeaSt	0.778	0.770	0.677	0.182	0.072
GAT	0.794	0.704	0.232	0.047	0.005
GCN	0.796	0.765	0.714	0.602	0.289
GGNN	0.661	0.078	0.021	0.033	0.039
GraphSAGE	0.925	0.816	0.632	0.303	0.053
HighOrder	0.629	0.145	0.023	0.004	0.012
HyperGraph	0.828	0.742	0.493	0.046	0.023

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

w. r. t. #Layers

GRAPH NEURAL NETWORKS EXPONENTIALLY LOSE EXPRESSIVE POWER FOR NODE CLASSIFICATION

Kenta Oono[1, 2, *], Taiji Suzuki[1, 3]
{kenta_oono, taiji}@mist.i.u-tokyo.ac.jp
[1]The University of Tokyo [2]Preferred Networks, Inc. [*]Work at the University of Tokyo
[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 $x_1, ..., x_N \in \mathbb{R}^C$ are node feature vectors and N = (#nodes).

- Augmented adjacency/degree matrix $\tilde{\mathbf{A}} = \mathbf{A}+\mathbf{I},\ \tilde{\mathbf{D}} = \mathbf{D}+\mathbf{I}.$

- GCN $f : \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times C}$

$$f = f_L \circ \cdots \circ f_1$$

$$f_l(X) = \sigma(\underbrace{PX}_{\text{convolved node features}}W_l)$$

where

σ is Relu activation,
W_l 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.

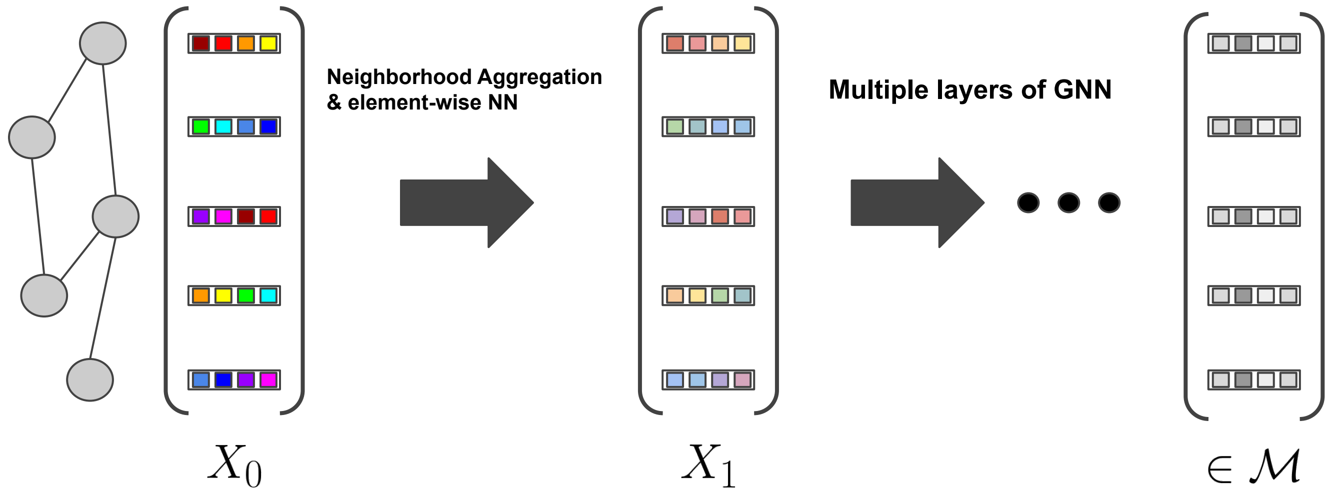
    i. e. $\mathcal{M} = \left\{ \begin{bmatrix} a_1 & a_2 & \cdots & a_C \\ a_1 & a_2 & \cdots & a_C \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \cdots & a_C \end{bmatrix} : a \in \mathbb{R}^C \right\} \subseteq \mathbb{R}^{N \times C}$. e. g. $X = \left\{ \begin{pmatrix} 2 & 7 & 3 & 5 & 1 \\ 2 & 7 & 3 & 5 & 1 \\ 2 & 7 & 3 & 5 & 1 \\ 2 & 7 & 3 & 5 & 1 \\ 2 & 7 & 3 & 5 & 1 \\ 2 & 7 & 3 & 5 & 1 \end{pmatrix} \right\} \in \mathcal{M}$.
  - Let $\{X_l\}_{l \in \mathbb{N}}$ be the sequence of node features defined by $X_l = f_l(X_{l-1}) \Big(:= \sigma(P X_{l-1} W_l) \Big)$.
  - Then,

$$\operatorname{dist}(X_l, \mathcal{M}) = \inf_{Y \in \mathcal{M}} ||X_l - Y||_{\text{Frob}} < C^l$$

for some constant $0 \leq C < 1$ (under some conditions).

Intuition of Exponential Decay

- $\mathrm{dist}(X_l, \mathcal{M}) = \inf_{Y \in \mathcal{M}} ||X_l - Y||_{\mathrm{Frob}} < C^l$



Analogy to Power Iteration

- Ignore the σ(·) and W from $f_l(X) = \sigma(PXW_l)$, and assume C=1

  ⇒ Power iteration to find the largest eigenvector of P ,

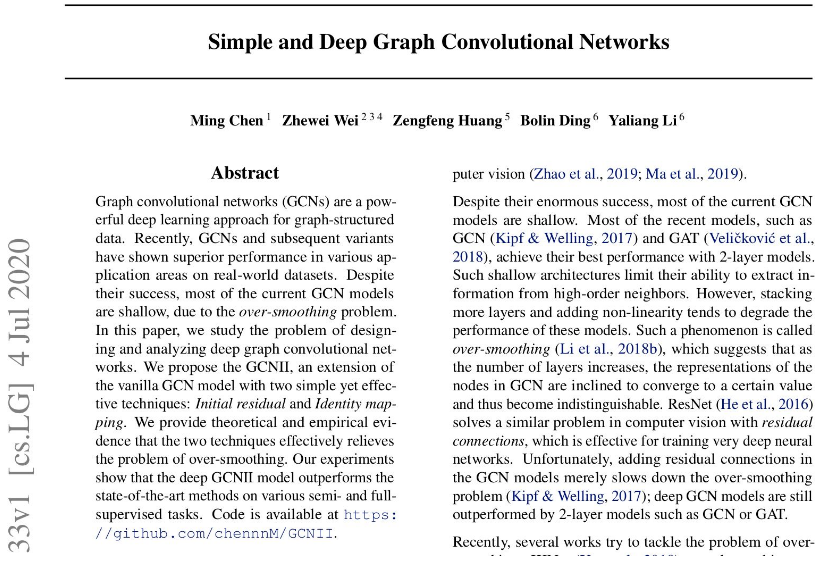
$$X_l = PX_{l-1}$$

X_l approaches to $\underbrace{\text{the eigenspace of largest eigenvalue of } P}_{\mathcal{M} \in \mathbb{R}^N}$.

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 Chen [1] **Zhewei Wei** [2 3 4] **Zengfeng Huang** [5] **Bolin Ding** [6] **Yaliang Li** [6]

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 GCNII, 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 relieves the problem of over-smoothing. Our experiments show that the deep GCNII model outperforms the state-of-the-art methods on various semi- and full-supervised tasks. Code is available at https://github.com/chennnM/GCNII.

puter vision (Zhao et al., 2019; Ma et al., 2019).

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 (Veličković 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

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\tilde{\mathbf{P}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}\right)$$

- GCNII = GCN + Initial residual connection + Identity mapping

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\left(\underbrace{(1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)} + \alpha_\ell\mathbf{H}^{(0)}}_{\text{initial residual connection}}\right)\left(\underbrace{(1-\beta_\ell)\mathbf{I}_n + \beta_\ell\mathbf{W}^{(\ell)}}_{\text{identity mapping}}\right)\right)$$

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.
α_ℓ's and β_ℓ's are hyperparameters.

Solution to Over-smoothing: Residual Connection (Chen et al [3])

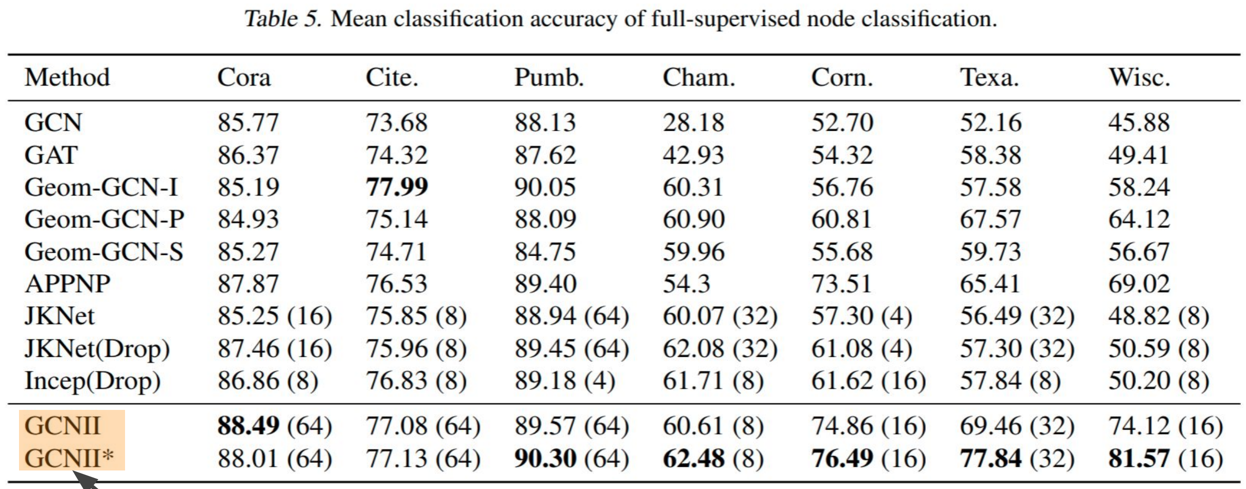


Table 5. Mean classification accuracy of full-supervised node classification.

Method	Cora	Cite.	Pumb.	Cham.	Corn.	Texa.	Wisc.
GCN	85.77	73.68	88.13	28.18	52.70	52.16	45.88
GAT	86.37	74.32	87.62	42.93	54.32	58.38	49.41
Geom-GCN-I	85.19	**77.99**	90.05	60.31	56.76	57.58	58.24
Geom-GCN-P	84.93	75.14	88.09	60.90	60.81	67.57	64.12
Geom-GCN-S	85.27	74.71	84.75	59.96	55.68	59.73	56.67
APPNP	87.87	76.53	89.40	54.3	73.51	65.41	69.02
JKNet	85.25 (16)	75.85 (8)	88.94 (64)	60.07 (32)	57.30 (4)	56.49 (32)	48.82 (8)
JKNet(Drop)	87.46 (16)	75.96 (8)	89.45 (64)	62.08 (32)	61.08 (4)	57.30 (32)	50.59 (8)
Incep(Drop)	86.86 (8)	76.83 (8)	89.18 (4)	61.71 (8)	61.62 (16)	57.84 (8)	50.20 (8)
GCNII	**88.49** (64)	77.08 (64)	89.57 (64)	60.61 (8)	74.86 (16)	69.46 (32)	74.12 (16)
GCNII*	88.01 (64)	77.13 (64)	**90.30** (64)	**62.48** (8)	**76.49** (16)	**77.84** (32)	**81.57** (16)

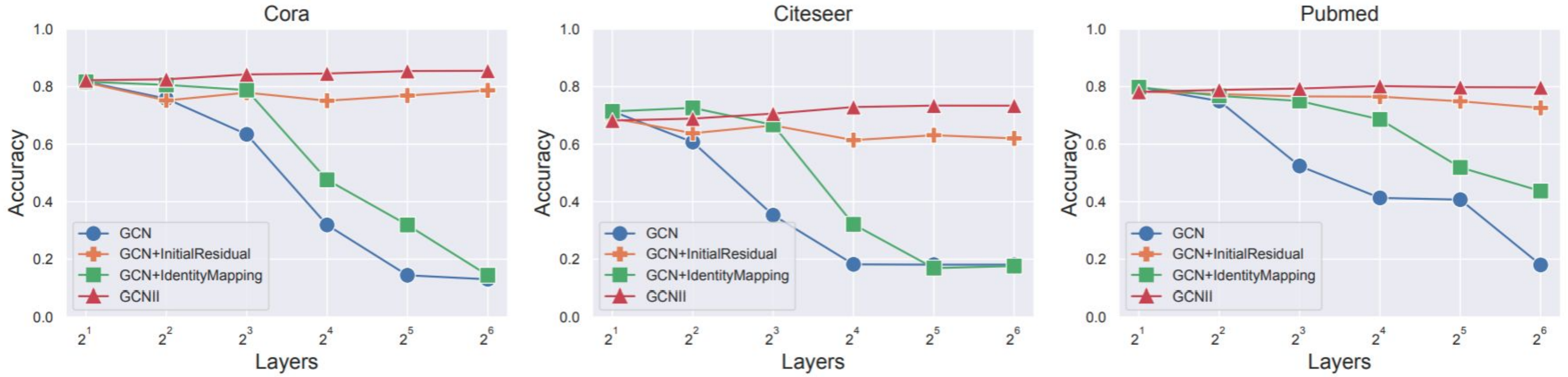
GCNII* uses **different weight matrix** for initial residual connection as

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\left(1-\alpha_\ell\right)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)}\left(\left(1-\beta_\ell\right)\mathbf{I}_n + \beta_\ell \mathbf{W}_1^{(\ell)}\right) + \alpha_\ell \mathbf{H}^{(0)}\left(\left(1-\beta_\ell\right)\mathbf{I}_n + \beta_\ell \mathbf{W}_2^{(\ell)}\right)\right)$$

Solution to Over-smoothing: Residual Connection (Chen et al [3])

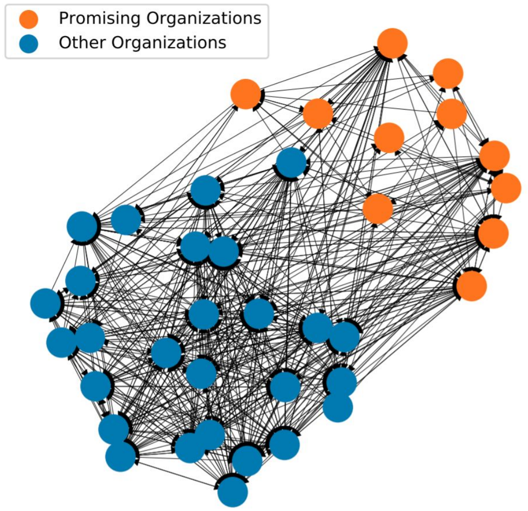
$$\mathbf{H}^{(\ell+1)} = \sigma\left(\left(\underbrace{(1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)} + \alpha_\ell \mathbf{H}^{(0)}}_{\text{initial residual connection}}\right)\left(\underbrace{(1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}^{(\ell)}}_{\text{identity mapping}}\right)\right)$$

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

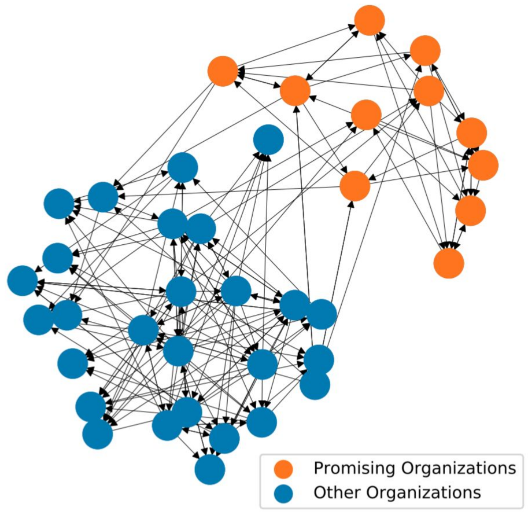


Agenda

Solution to Over-smoothing: Graph Sparsification

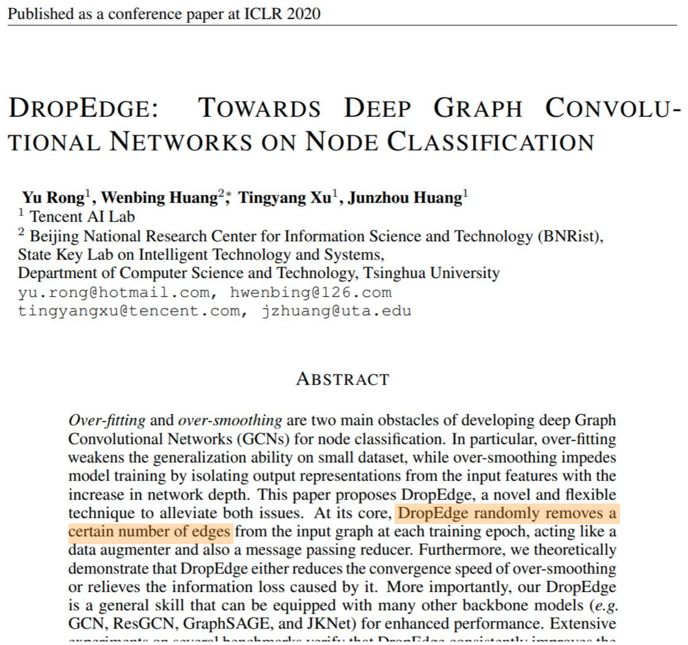


(a) Original Subgraph



(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[1], **Wenbing Huang**[2,*], **Tingyang Xu**[1], **Junzhou Huang**[1]
[1] Tencent AI Lab
[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, hwenbing@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

Solution to Over-smoothing: Graph Sparsification (Rong et al [4])

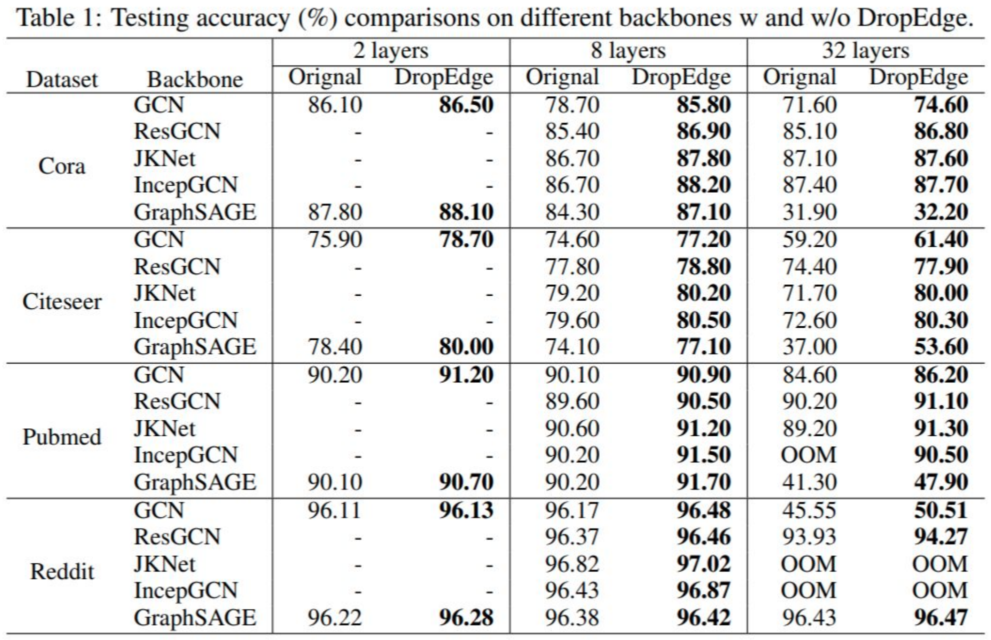
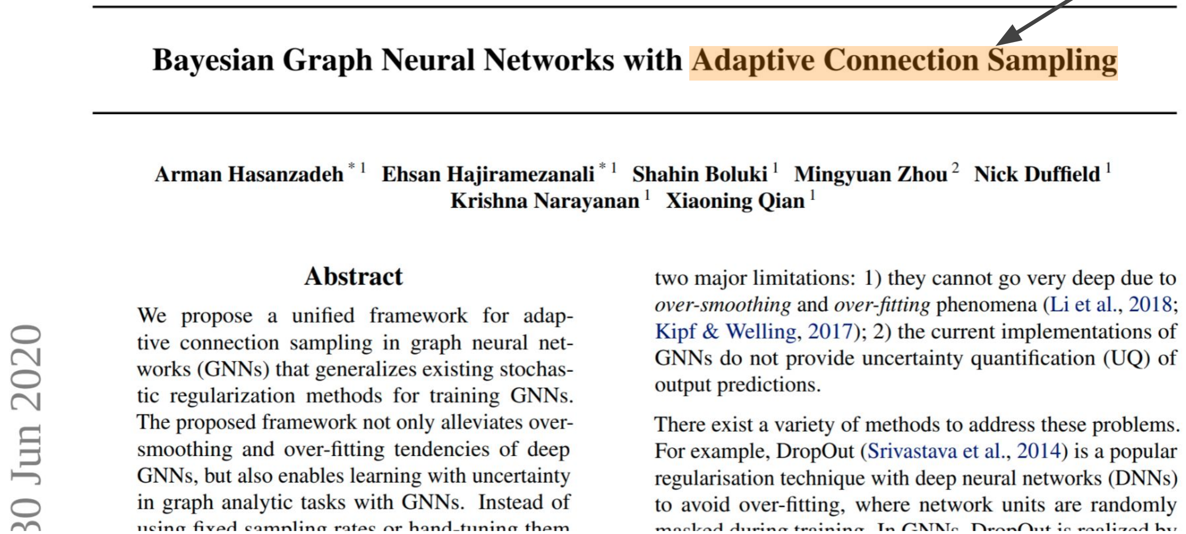


Table 1: Testing accuracy (%) comparisons on different backbones w and w/o DropEdge.

		2 layers		8 layers		32 layers	
Dataset	Backbone	Orignal	DropEdge	Orignal	DropEdge	Orignal	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

Arman Hasanzadeh [*1] **Ehsan Hajiramezanali** [*1] **Shahin Boluki** [1] **Mingyuan Zhou** [2] **Nick Duffield** [1] **Krishna Narayanan** [1] **Xiaoning Qian** [1]

30 Jun 2020

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

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

Solution to Over-smoothing: Graph Sparsification (Hasanzadeh et al [5])

- DropEdge (layer wise)

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathfrak{N}(\mathbf{A} \odot \mathbf{Z}^{(l)})\,\mathbf{H}^{(l)}\,\mathbf{W}^{(l)}\right)$$

- Graph Drop Connect (layer & channel wise)

$$\mathbf{H}^{(l+1)}[:,j] = \sigma\left(\sum_{i=1}^{f_l} \mathfrak{N}(\mathbf{A} \odot \mathbf{Z}_{i,j}^{(l)})\,\mathbf{H}^{(l)}[:,i]\,\mathbf{W}^{(l)}[i,j]\right), \quad \text{for} \quad j = 1, \ldots, f_{l+1} \tag{4}$$

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 π_l of edges as posterior inference

  ⇔ DropEdge as Bayesian approximation of $W_{j,i}^{(l)} \overset{\text{i.i.d.}}{\sim} N(0, I)$

Solution to Over-smoothing: Graph Sparsification (Hasanzadeh et al [5])

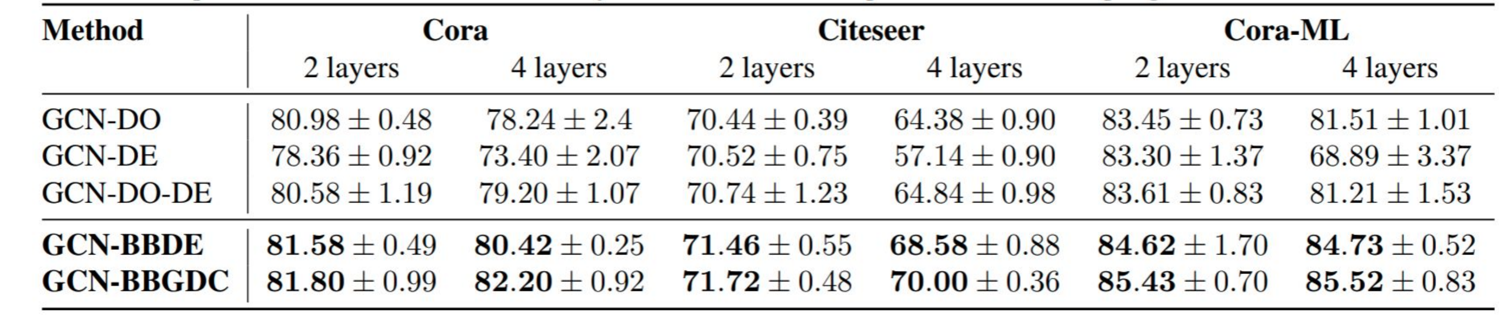


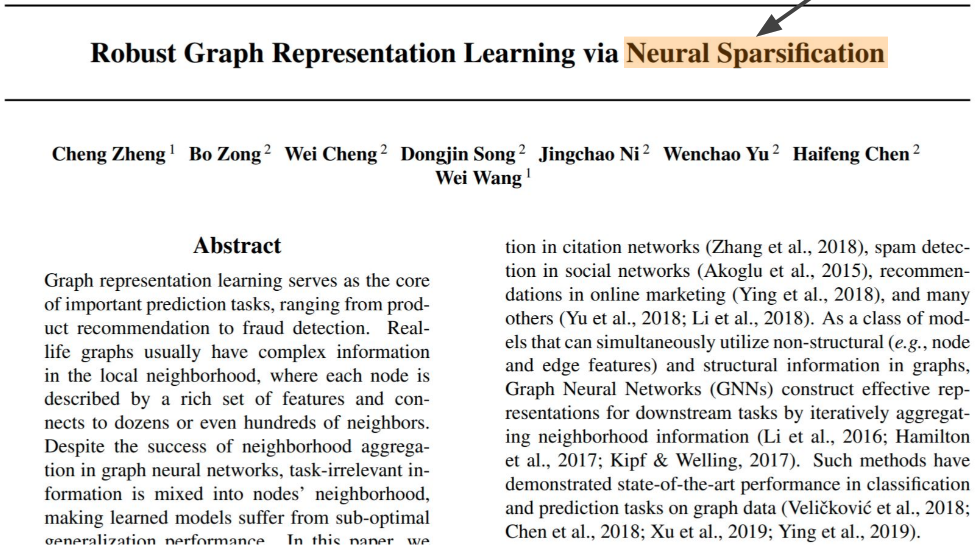
Table 1. Semi-supervised node classification accuracy of GCNs with our adaptive connection sampling and baseline methods.

Method	**Cora**		**Citeseer**		**Cora-ML**	
	2 layers	4 layers	2 layers	4 layers	2 layers	4 layers
GCN-DO	80.98 ± 0.48	78.24 ± 2.4	70.44 ± 0.39	64.38 ± 0.90	83.45 ± 0.73	81.51 ± 1.01
GCN-DE	78.36 ± 0.92	73.40 ± 2.07	70.52 ± 0.75	57.14 ± 0.90	83.30 ± 1.37	68.89 ± 3.37
GCN-DO-DE	80.58 ± 1.19	79.20 ± 1.07	70.74 ± 1.23	64.84 ± 0.98	83.61 ± 0.83	81.21 ± 1.53
GCN-BBDE	$\mathbf{81.58} \pm 0.49$	$\mathbf{80.42} \pm 0.25$	$\mathbf{71.46} \pm 0.55$	$\mathbf{68.58} \pm 0.88$	$\mathbf{84.62} \pm 1.70$	$\mathbf{84.73} \pm 0.52$
GCN-BBGDC	$\mathbf{81.80} \pm 0.99$	$\mathbf{82.20} \pm 0.92$	$\mathbf{71.72} \pm 0.48$	$\mathbf{70.00} \pm 0.36$	$\mathbf{85.43} \pm 0.70$	$\mathbf{85.52} \pm 0.83$

● 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 Zheng[1] Bo Zong[2] Wei Cheng[2] Dongjin Song[2] Jingchao Ni[2] Wenchao Yu[2] Haifeng Chen[2] Wei Wang[1]

Abstract

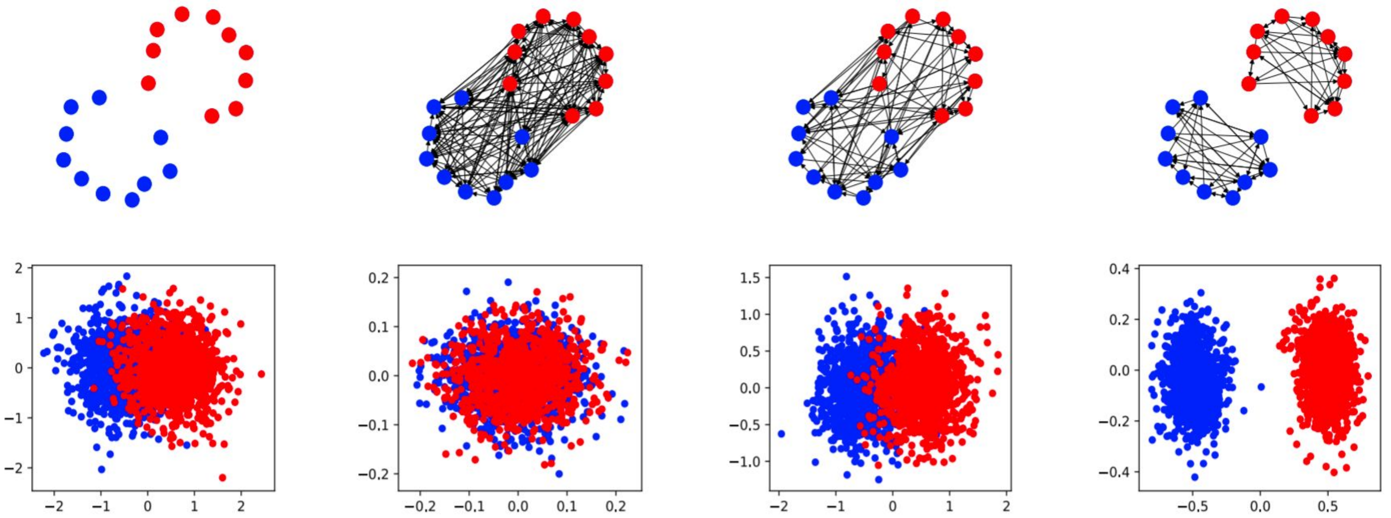
Graph representation learning serves as the core of important prediction tasks, ranging from product recommendation to fraud detection. Real-life graphs usually have complex information in the local neighborhood, where each node is described by a rich set of features and connects to dozens or even hundreds 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

tion in citation networks (Zhang et al., 2018), spam detection in social networks (Akoglu et al., 2015), recommendations in online marketing (Ying et al., 2018), and many others (Yu et al., 2018; Li et al., 2018). As a class of models that can simultaneously utilize non-structural (*e.g.*, node and edge features) and structural information in graphs, Graph Neural Networks (GNNs) construct effective representations for downstream tasks by iteratively aggregating neighborhood information (Li et al., 2016; Hamilton et al., 2017; Kipf & Welling, 2017). Such methods have demonstrated state-of-the-art performance in classification and prediction tasks on graph data (Veličković et al., 2018; Chen et al., 2018; Xu et al., 2019; Ying et al., 2019).

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

Input graph

Node features obtained by GCN



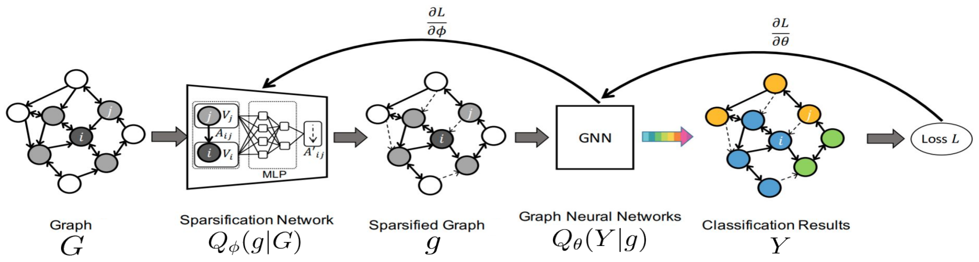
(a) Node Features (b) With Task-irrelevant Edges (c) By DropEdge (d) By NeuralSparse

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

- Consider a weighted objective

$$P(Y|G) \approx \sum_{g \in \mathbb{S}_G} \underbrace{Q_\theta(Y|g)}_{\text{original objective}} \underbrace{Q_\phi(g|G)}_{\text{weight}} = \mathbb{E}_{g \sim Q_\phi(g|G)} \left[Q_\theta(Y|g) \right]$$

  - $g \in \mathbb{S}_G$ is the latent variable
  - $\mathbb{S}_G$ 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 $z_{u,v} = \mathrm{MLP}_\phi(V(u), V(v), \mathbf{A}(u,v))$.



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

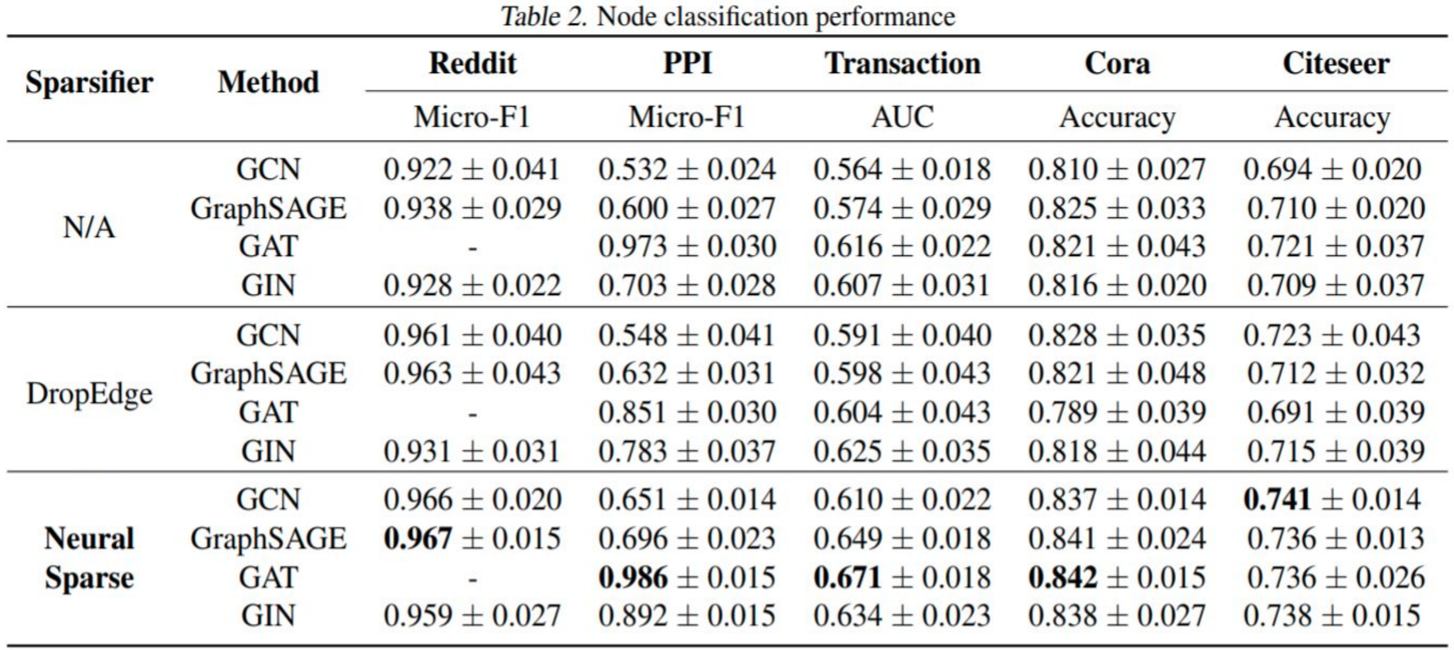


Table 2. Node classification performance

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

Summary

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

References

[1] Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., & Sun, X. (2020). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In Proceedings of the AAAI Conference on Artificial Intelligence.

[2] Oono, K., & Suzuki, T. (2020). Graph neural networks exponentially lose expressive power for node classification. In Proceedings of the International Conference on Learning Representation.

[3] Chen, M., Wei, Z., Huang, Z., Ding, B., & Li, Y. (2020). Simple and deep graph convolutional networks. In *International Conference on Machine Learning*.

[4] Rong, Y., Huang, W., Xu, T., & Huang, J. (2019). Dropedge: Towards deep graph convolutional networks on node classification. In Proceedings of the International Conference on Learning Representation.

[5] Hasanzadeh, A., Hajiramezanali, E., Boluki, S., Zhou, M., Duffield, N., Narayanan, K., & Qian, X. (2020, November). Bayesian graph neural networks with adaptive connection sampling. In *International Conference on Machine Learning*.

[6] Zheng, C., Zong, B., Cheng, W., Song, D., Ni, J., Yu, W., ... & Wang, W. (2020). Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*.