Consistency Regularization for GNNs

Yukuo Cen

GNN Center, Zhipu AI

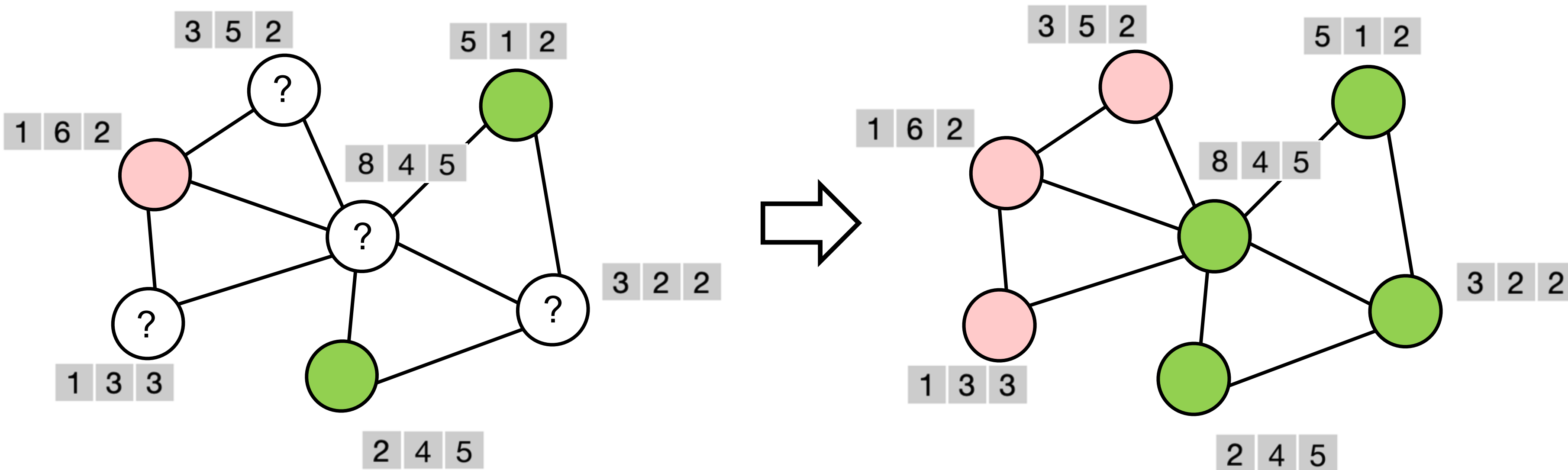
KEG, Tsinghua University

Advisors: Yuxiao Dong, Jie Tang

Course Link: https://cogdl.ai/gnn2022/

CogDL is publicly available at https://github.com/THUDM/cogdl

Semi-Supervised Learning on Graphs



Input: a partially labeled & attributed graph

Output: infer the labels of unlabeled nodes

Graph Neural Networks (GNN)

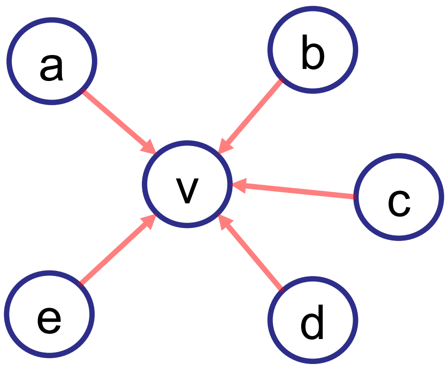
Message Passing:

Representation vector of the $l+1$-th layer

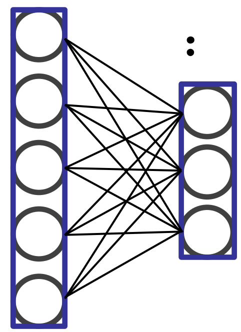
Activation function (e.g. ReLU)

GCN: $\boldsymbol{H}^{l+1} = \sigma\left(\widehat{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\right)$

Normalized adjacency matrix



Feature propagation



Non-linear transformation

$$\boldsymbol{H}^{l+1} = \sigma\left(\boldsymbol{W}^{l} \sum_{u \in N(v) \cup v} \frac{\boldsymbol{H}_{\boldsymbol{u}}^{\boldsymbol{l}}}{\sqrt{|N(u)||N(v)|}}\right)$$

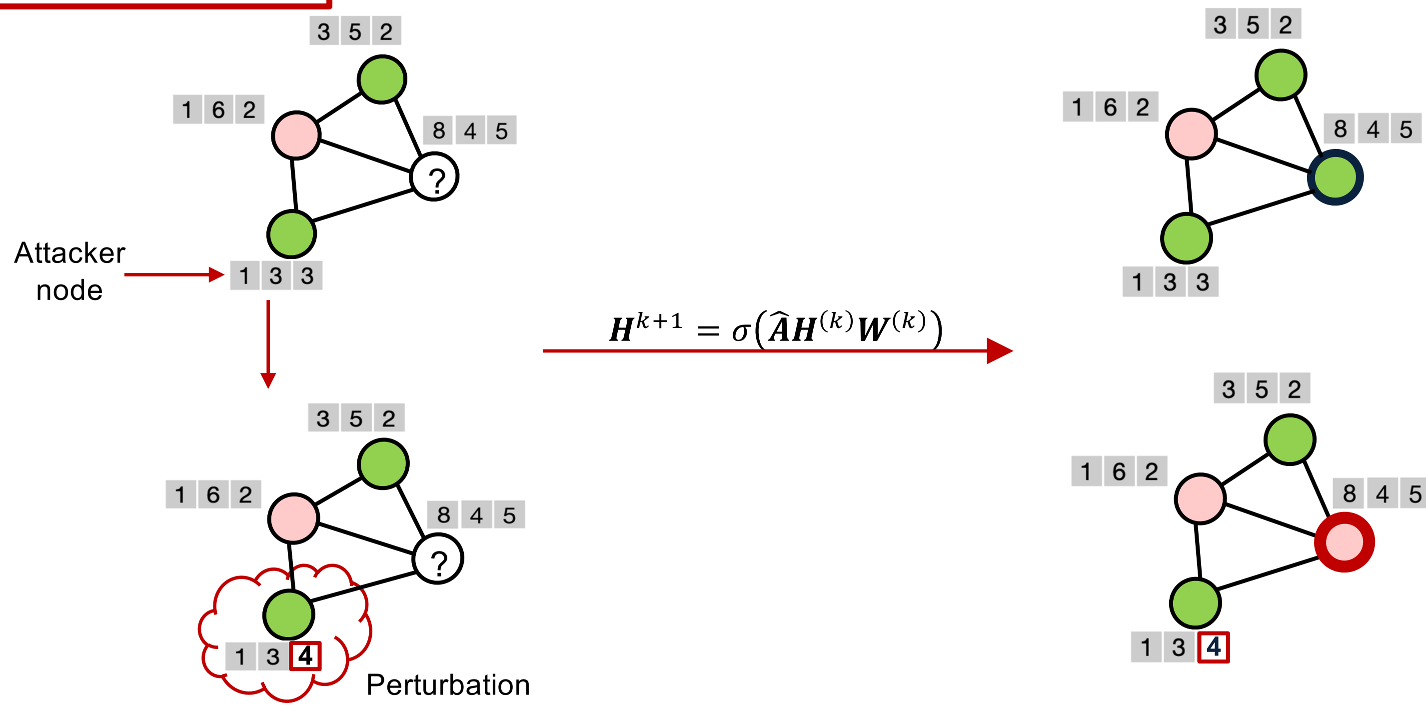
- Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In ICLR 2017

Potential Issues of GNNs

$$H^{k+1} = \sigma\left(\hat{A}H^{(k)}W^{(k)}\right)$$

a deterministic propagation

1. Each node is highly dependent with its neighborhoods, making GNNs **non-robust** to noises



- Zügner D, Akbarnejad A, Günnemann S. Adversarial attacks on neural networks for graph data. In KDD 2018.

Potential Issues of GNNs

$$\boldsymbol{H}^{k+1} = \sigma\left(\widehat{\boldsymbol{A}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}\right)$$

feature propagation is
Laplacian smoothing,
coupled with
non-linear transformation

1. Each node is highly dependent with its neighborhoods, making GNNs **non-robust** to noises
2. Stacking many GNNs layers may cause **over-smoothing.**

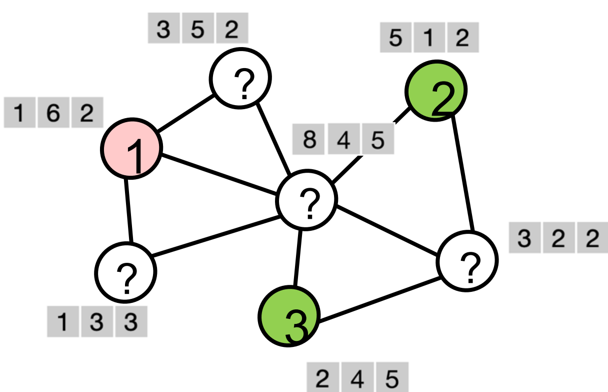
- Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI'18.*
- Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020.

Potential Issues of GNNs

1. Each node is highly dependent with its neighborhoods, making GNNs **non-robust** to noises
2. Stacking many GNNs layers may cause **over-smoothing.**
3. Under semi-supervised setting, standard training method is easy to **over-fit** the scarce label information.

Standard training method for GNN:

$$\boldsymbol{H}^{k+1} = \sigma\left(\widehat{\boldsymbol{A}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}\right)$$



GNN

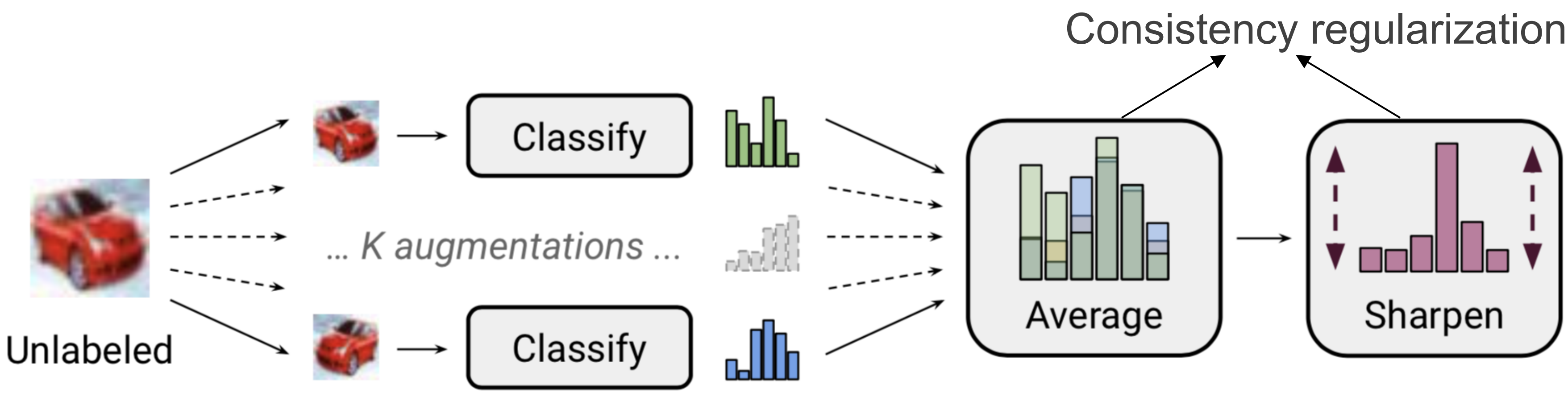
Loss function:

$$\boldsymbol{y}_1^{\mathrm{T}}\log(\widehat{\boldsymbol{y}}_1) + \boldsymbol{y}_2^{\mathrm{T}}\log(\widehat{\boldsymbol{y}}_2) + \boldsymbol{y}_3^{\mathrm{T}}\log(\widehat{\boldsymbol{y}}_3)$$

Cannot fully leverage unlabeled data

Recent advances in Semi-Supervised Learning

- Improving models' generalization through image data augmentation and consistency regularization.

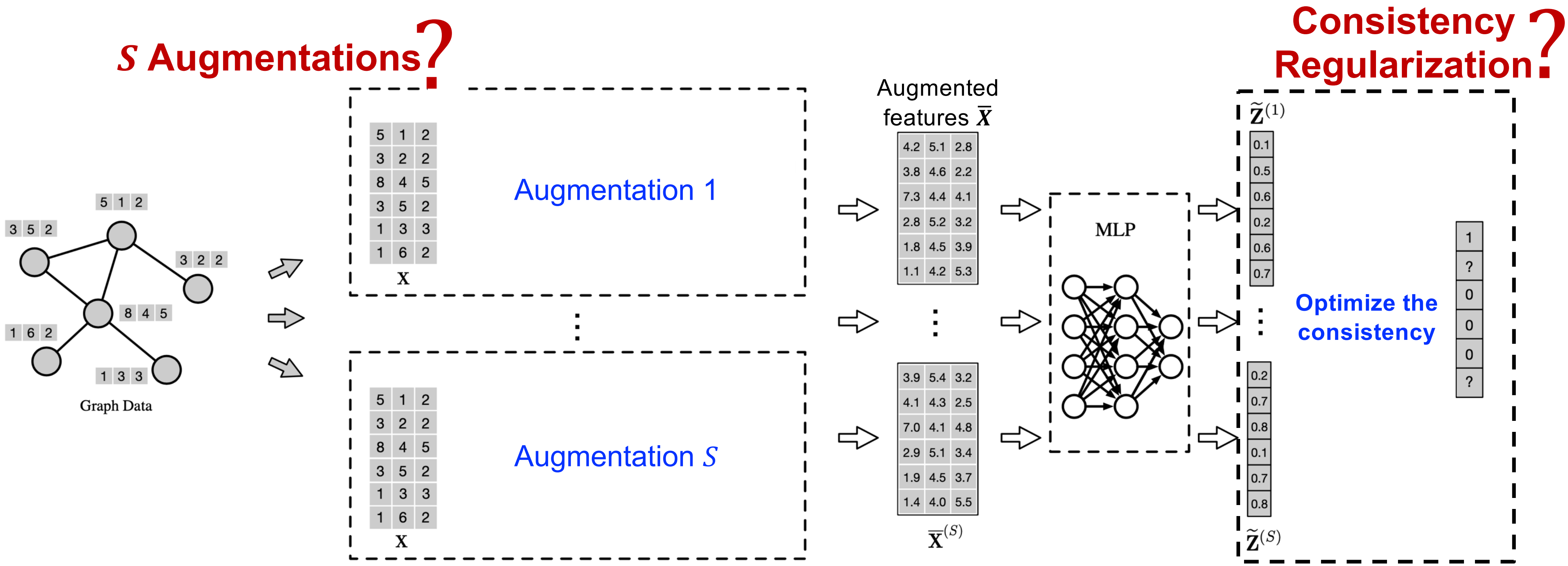


(Picture from MixMatch's paper)

- Berthelot D, Carlini N, Goodfellow I, et al. Mixmatch: A holistic approach to semi-supervised learning. In NIPS'19.

Graph Random Neural Network (GRAND)

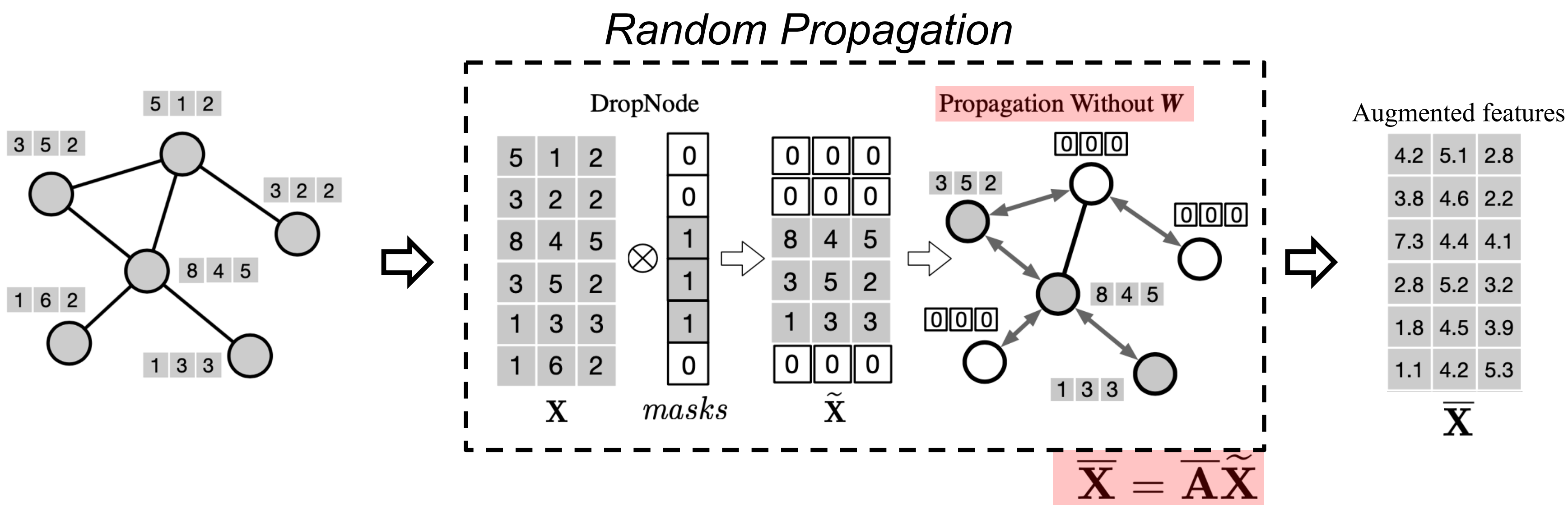
- Consistency Regularized Training:
  - Generates S data augmentations of the graph
  - Optimizing the consistency among S augmentations of the graph.



- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/Grand20/grand

Random Propagation in GRAND

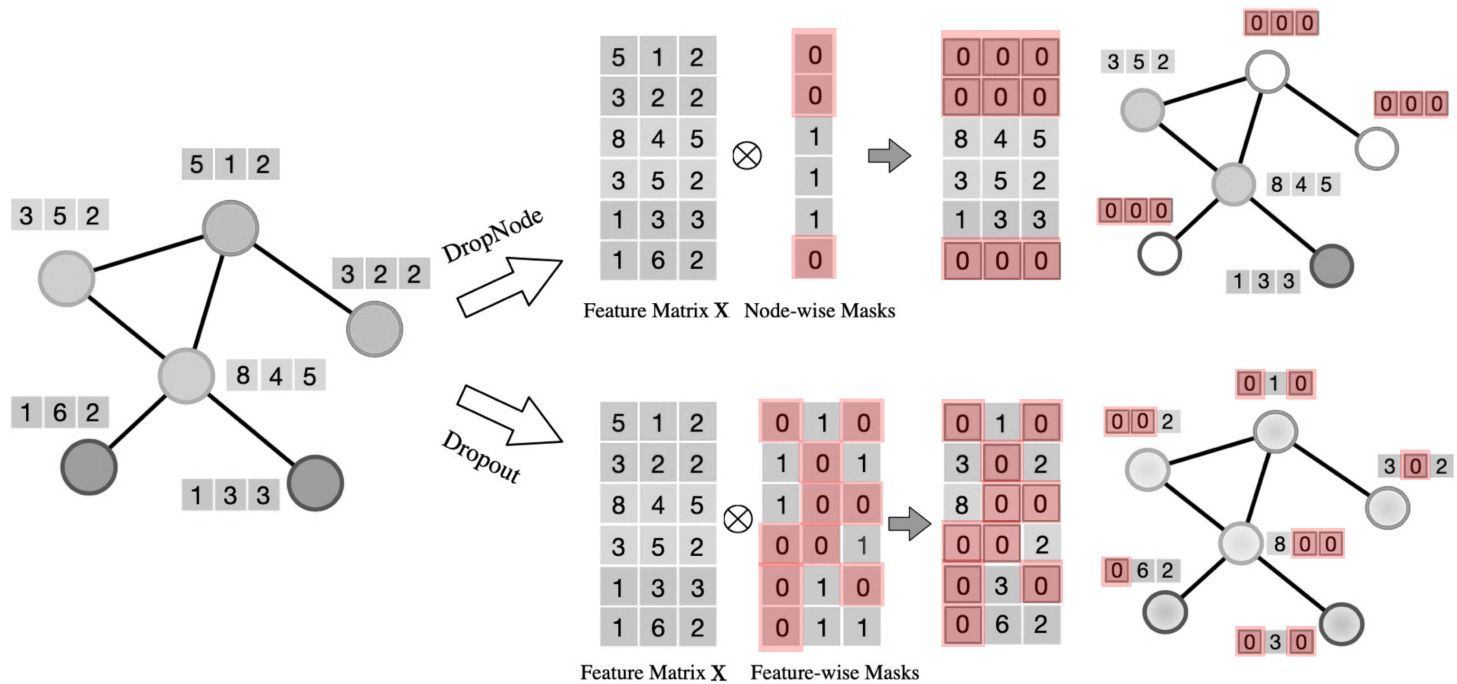
- **Random Propagation** (DropNode + Propagation):
  - Enhancing robustness: Each node is enabled to be not sensitive to specific neighborhoods.
  - Mitigating over-smoothing and overfitting: Decouple feature propagation from feature transformation.



- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/Grand20/grand

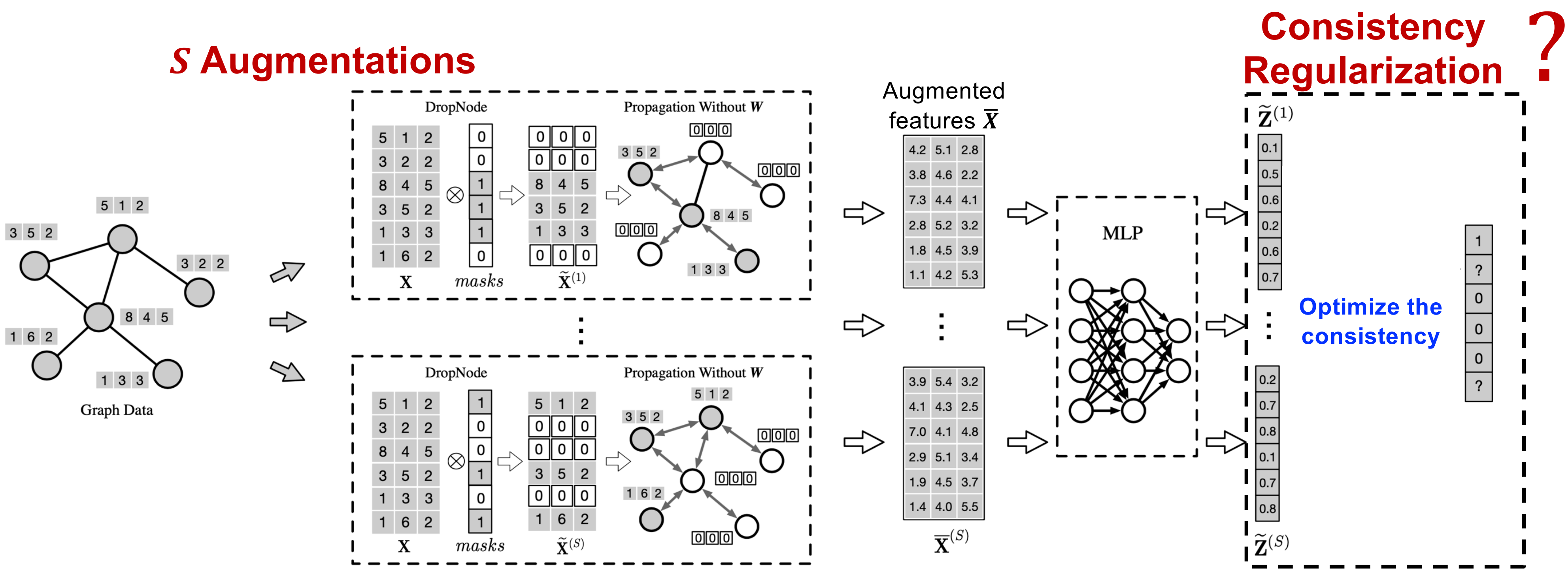
Random propagation: DropNode vs Dropout

- Dropout drops each element in $\boldsymbol{X}$ independently
- DropNode drops the entire features of selected nodes, i.e., the row vectors of $\boldsymbol{X}$, randomly



- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/Grand20/grand

Consistency Regularization?



Random Propagation as data augmentation

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/Grand20/grand

GRAND: Consistency Regularization

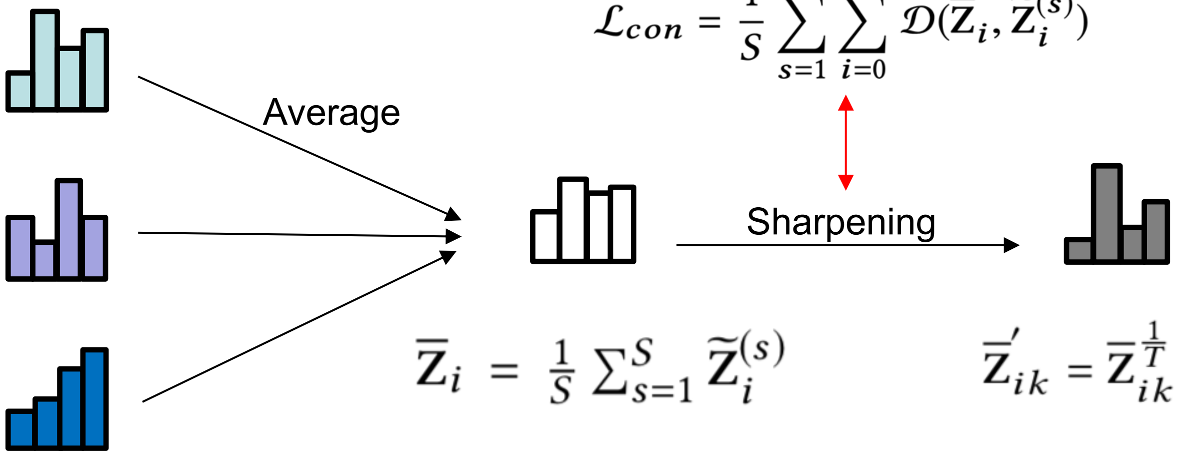
Distributions of a node after augmentations

$$\mathcal{L}_{sup} = -\frac{1}{S}\sum_{s=1}^{S}\sum_{i=0}^{m-1}\mathbf{Y}_i^{\top}\log\widetilde{\mathbf{Z}}_i^{(s)}$$

$$+$$

$$\mathcal{L}_{con} = \frac{1}{S}\sum_{s=1}^{S}\sum_{i=0}^{n-1}\mathcal{D}(\overline{\mathbf{Z}}'_i, \widetilde{\mathbf{Z}}_i^{(s)})$$

$$\Rightarrow \mathcal{L} = \mathcal{L}_{sup} + \lambda\mathcal{L}_{con}$$



$$\overline{\mathbf{Z}}_i = \frac{1}{S}\sum_{s=1}^{S}\widetilde{\mathbf{Z}}_i^{(s)}$$

$$\overline{\mathbf{Z}}'_{ik} = \overline{\mathbf{Z}}_{ik}^{\frac{1}{T}} \Big/ \sum_{j=0}^{C-1}\overline{\mathbf{Z}}_{ij}^{\frac{1}{T}}$$

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/Grand20/grand

Training Algorithm of GRAND

Input:

Adjacency matrix $\hat{\mathbf{A}}$, feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, times of augmentations in each epoch S, DropNode probability δ.

Output:

Prediction $\mathbf{Z}$.

1: **while** not convergence **do**
2: **for** $s = 1 : S$ **do**
3: Apply DropNode via Algorithm 1: $\widetilde{\mathbf{X}}^{(s)} \sim \mathrm{DropNode}(\mathbf{X}, \delta)$.
4: Perform propagation: $\overline{\mathbf{X}}^{(s)} = \frac{1}{K+1} \sum_{k=0}^{K} \hat{\mathbf{A}}^k \widetilde{\mathbf{X}}^{(s)}$.
5: Predict class distribution using MLP: $\widetilde{\mathbf{Z}}^{(s)} = P(\mathbf{Y} | \overline{\mathbf{X}}^{(s)}; \Theta)$.
6: **end for**
7: Compute supervised classification loss $\mathcal{L}_{sup}$ via Eq. 4 and consistency regularization loss via Eq. 6.
8: Update the parameters Θ by gradients descending:

$$\nabla_\Theta \mathcal{L}_{sup} + \lambda \mathcal{L}_{con}$$

9: **end while**
10: Output prediction $\mathbf{Z}$ via Eq. 8.

Generate *S* Augmentations

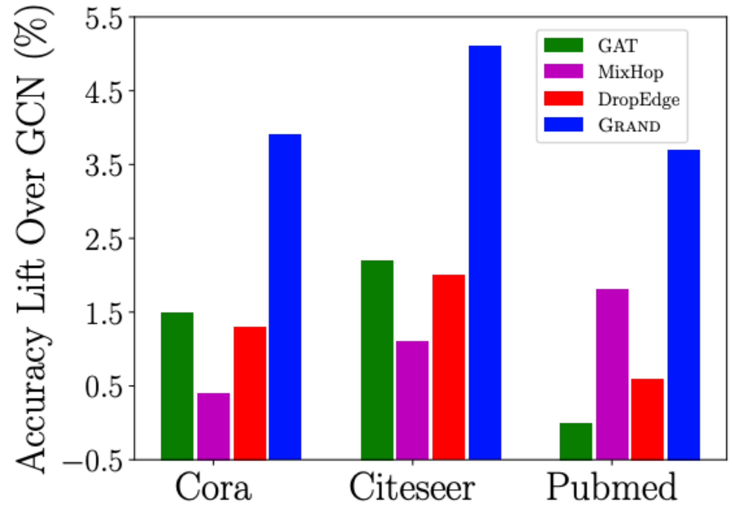
Consistency Regularization

Consistency Regularized Training Algorithm

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/Grand20/grand

GRAND Results

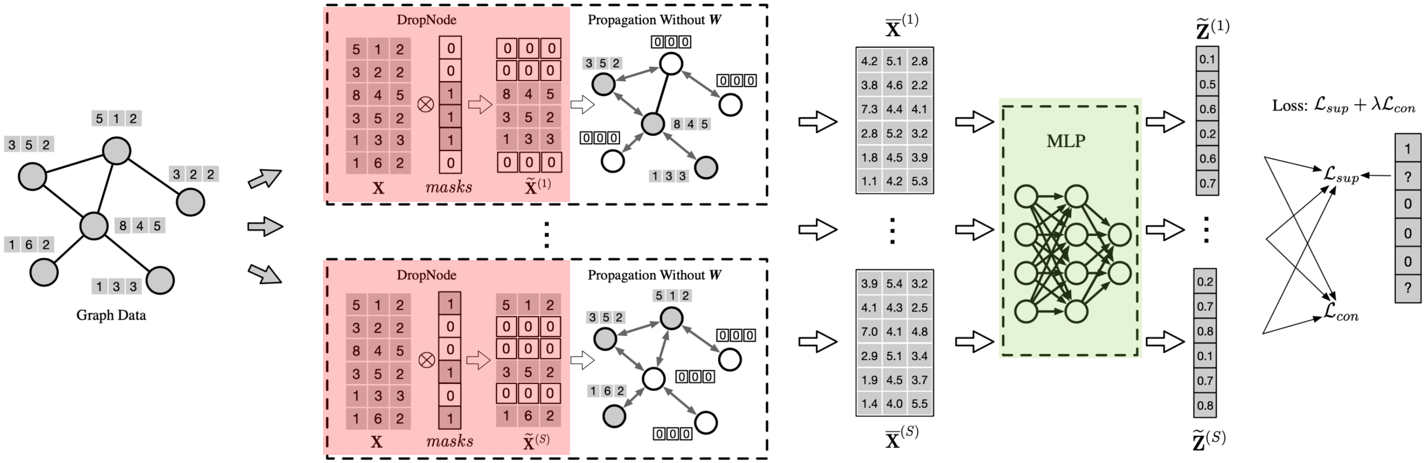
	Method	Cora	Citeseer	Pubmed
GCNs	GCN [19]	81.5	70.3	79.0
	GAT [32]	83.0±0.7	72.5±0.7	79.0±0.3
	APPNP [20]	83.8±0.3	71.6± 0.5	79.7 ± 0.3
	Graph U-Net [11]	84.4±0.6	73.2±0.5	79.6±0.2
	SGC [36]	81.0 ±0.0	71.9 ± 0.1	78.9 ± 0.0
	MixHop [1]	81.9± 0.4	71.4±0.8	80.8±0.6
	GMNN [28]	83.7	72.9	81.8
	GraphNAS [12]	84.2±1.0	73.1±0.9	79.6±0.4
Sampling GCNs	GraphSAGE [16]	78.9±0.8	67.4±0.7	77.8±0.6
	FastGCN [7]	81.4±0.5	68.8±0.9	77.6±0.5
Regularization GCNs	VBAT [10]	83.6±0.5	74.0±0.6	79.9±0.4
	G^3NN [24]	82.5±0.2	74.4±0.3	77.9 ±0.4
	GraphMix [33]	83.9±0.6	74.5±0.6	81.0±0.6
	DropEdge [29]	82.8	72.3	79.6
	GRAND	**85.4±0.4**	**75.4±0.4**	**82.7±0.6**



Instead of the marginal improvements by conventional GNN baselines over GCN, ***GRAND*** *achieves* ***much more significant*** *performance lift in* ***all three datasets***!

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/Grand20/grand

Results of Different Choices



GRAND_dropout	84.9±0.4	75.0±0.3	81.7±1.0
GRAND_GCN	84.5±0.3	74.2±0.3	80.0±0.3
GRAND_GAT	84.3±0.4	73.2± 0.4	79.2±0.6
GRAND	**85.4±0.4**	**75.4±0.4**	**82.7±0.6**

Evaluation of the design choices in GRAND

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/Grand20/grand

Ablation Study of GRAND

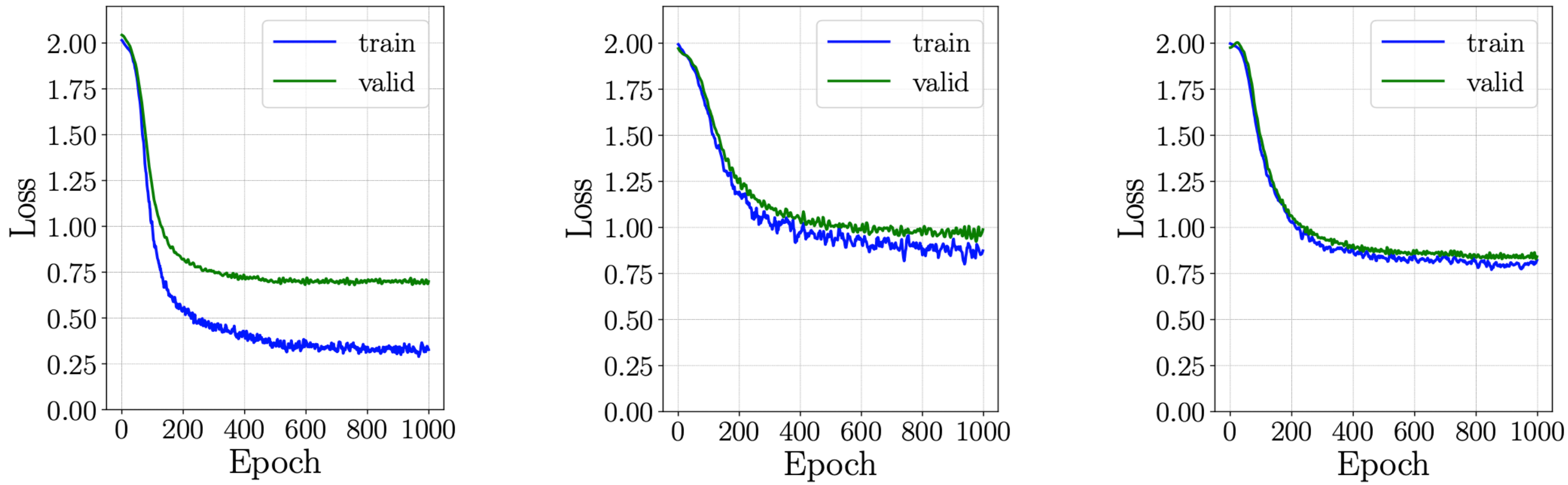
Method	Cora	Citeseer	Pubmed
GCN [19]	81.5	70.3	79.0
GAT [32]	83.0±0.7	72.5±0.7	79.0±0.3
APPNP [20]	83.8±0.3	71.6± 0.5	79.7 ± 0.3
Graph U-Net [11]	84.4±0.6	73.2±0.5	79.6±0.2
SGC [36]	81.0 ±0.0	71.9 ± 0.1	78.9 ± 0.0
MixHop [1]	81.9± 0.4	71.4±0.8	80.8±0.6
GMNN [28]	83.7	72.9	81.8
GraphNAS [12]	84.2±1.0	73.1±0.9	79.6±0.4
DropEdge [29]	82.8	72.3	79.6
w/o CR	84.4±0.5	73.1±0.6	80.9±0.8
w/o mDN	84.7±0.4	74.8±0.4	81.0±1.1
w/o sharpening	84.6±0.4	72.2±0.6	81.6±0.8
w/o CR & DN	83.2±0.5	70.3±0.6	78.5±1.4

Ablation Study

1. Each of the designed components contributes to the success of GRAND.
2. GRAND w/o consistency regularization outperforms almost *all 8 non-regularization based GCNs* & *DropEdge*

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/Grand20/grand

Analysis of CR and RP



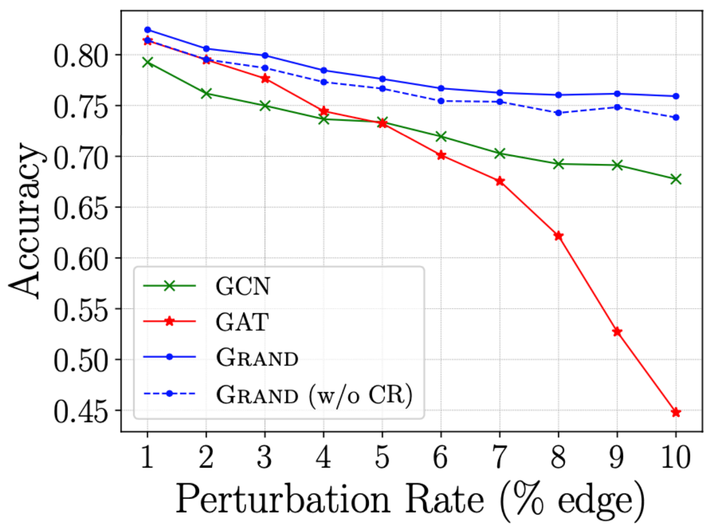
(a) Without CR and RP (b) Without CR (c) GRAND

Generalization

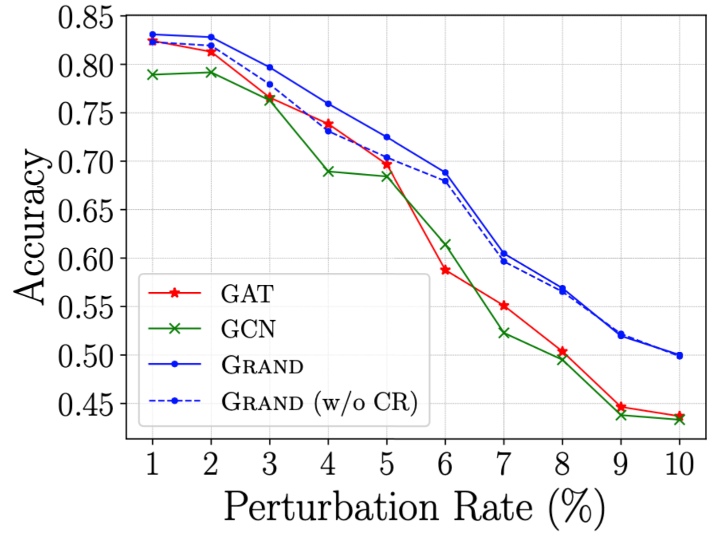
1. Both the random propagation and consistency regularization improve GRAND's generalization capability

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/Grand20/grand

Robustness Analysis



(a) Random Attack



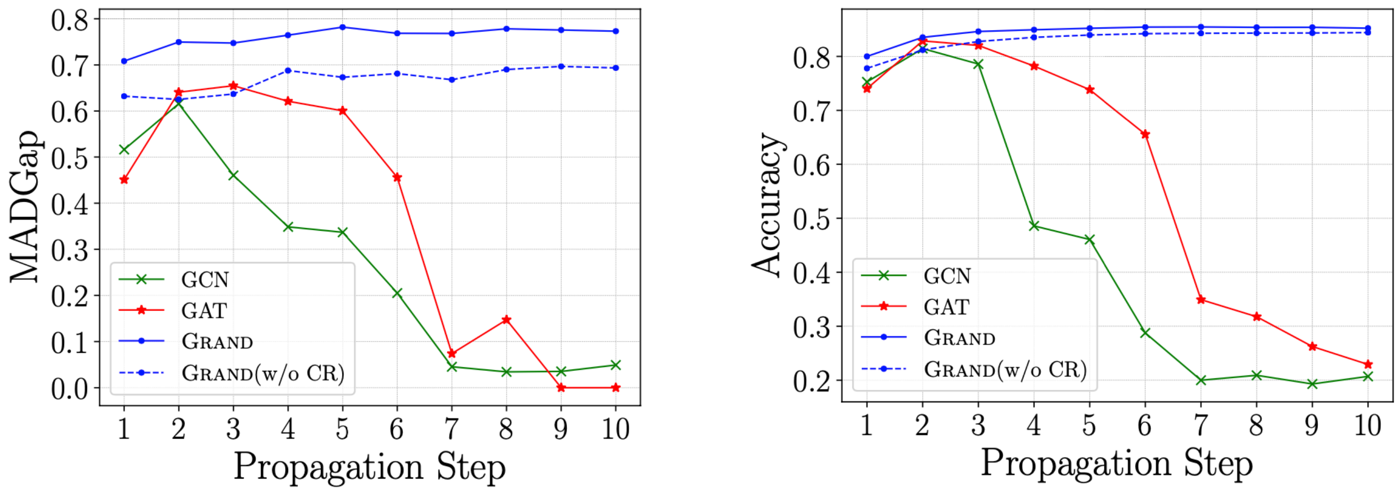
(b) Metattack

Robustness

1. GRAND (with or w/o) consistency regularization is more robust than GCN and GAT.

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/Grand20/grand

Over-smoothing Analysis



(a) MADGap (b) Classification Results

Over-Smoothing

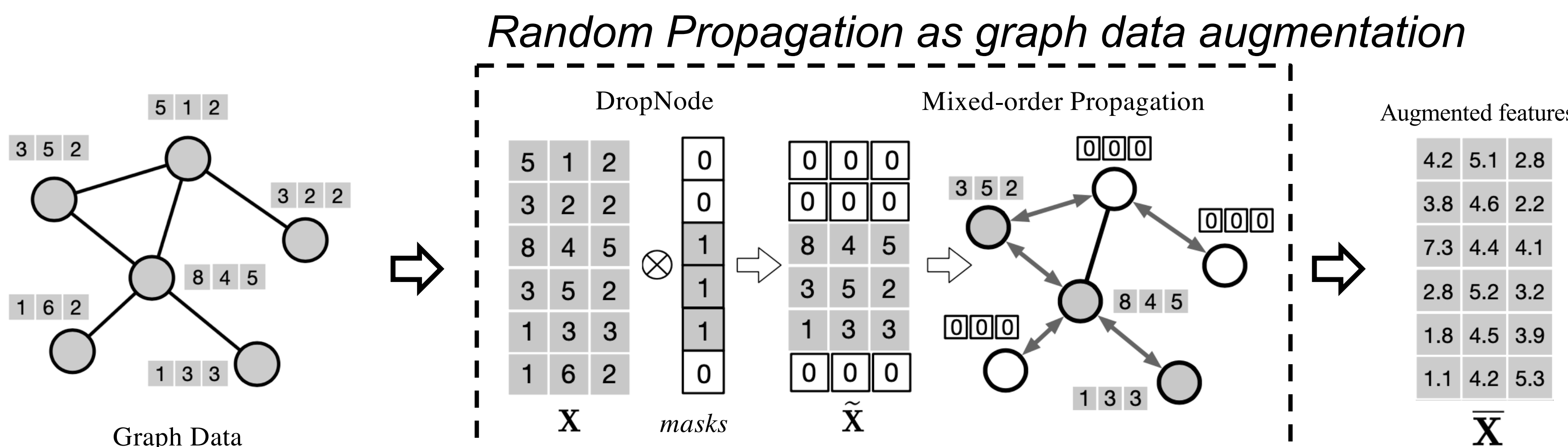
1. GRAND is very powerful to relieve over-smoothing, when GCN & GAT are very vulnerable to it

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/Grand20/grand

GRAND+: Scalable Graph Random Neural Networks

Review GRAND

- Random Propagation (DropNode + Propagation):
  - Decouple the feature propagation from non-linear feature transformation.
  - Propagate feature with a mixed-order adjacency matrix: $\mathbf{\Pi} = \sum_{n=0}^{N} \frac{1}{N+1} \hat{\mathbf{A}}^n$
  - Use DropNode to randomly aggregate neighbors' features



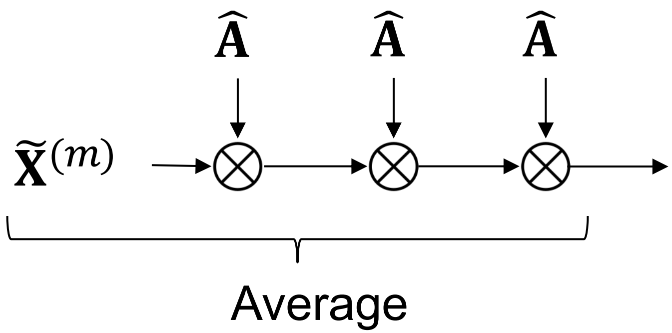
Feng W, Zhang J, Dong Y, et al. Graph random neural networks for semi-supervised learning on graphs[J]. Advances in neural information processing systems, 2020, 33: 22092-22103.

Scalability limitation of GRAND

- Random Propagation in GRAND:

$$\overline{\mathbf{X}}^{(m)} = \mathbf{\Pi}\, \widetilde{\mathbf{X}}^{(m)}, \qquad \mathbf{\Pi} = \sum_{n=0}^{N} \frac{1}{N+1} \widehat{\mathbf{A}}^{n}$$

- $\overline{\mathbf{X}}^{(m)}$ is calculated with power iteration:



Weak Scalability:

- Time/Memory complexity: $O(|E| + |V|)$.
- Random propagation needs to be formed for multiple times at each epoch.

GRAND+: General Idea

- Mini-batch Radom Propagation:
  - Select a batch of nodes at each training step, and generate augmented features by

DropNode mask

$$\overline{\mathbf{X}}_s^{(m)} = \sum_{v \in \mathcal{N}_v^{\pi}} \boldsymbol{z}_v \cdot \boldsymbol{\Pi}(\mathrm{s}, \mathrm{v}) \cdot \mathbf{X}_v, \qquad \boldsymbol{z}_{\boldsymbol{v}} \sim Bernoulli(1-\delta)$$

Non-zero elements in $\boldsymbol{\Pi}_{\mathbf{s}}$

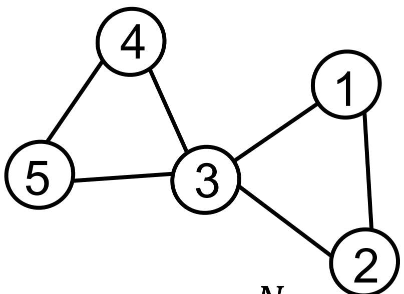
$$\boldsymbol{\Pi} = \sum_{n=0}^{N} \frac{1}{N+1} \widehat{\mathbf{A}}^n$$

How to efficiently calculate the row vector $\boldsymbol{\Pi}_{\mathbf{s}}$?

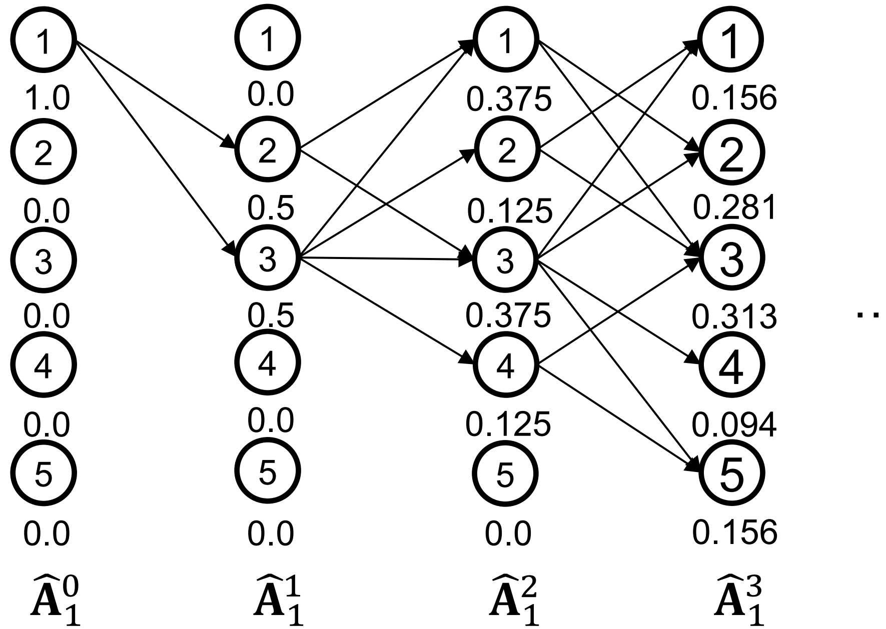
GRAND+：Matrix approximation

$$\mathbf{\Pi} = \frac{1}{N+1}\sum_{n=0}^{N} \widehat{\mathbf{A}}^{n}$$

$\widehat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{A}}$ is random walk reverse transition matrix. $\mathbf{P}(s, \mathrm{v})$ indicates the random walk probability from s to v.



$$\mathbf{\Pi_1} = \frac{1}{N+1}\sum_{n=0}^{N} \widehat{\mathbf{A}}_1^{n}$$



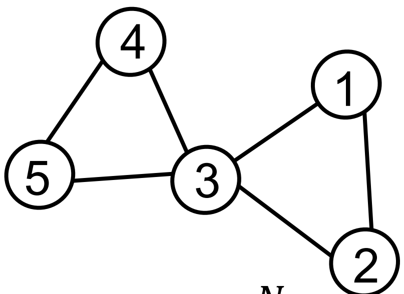
Random Walk Probability Diffusion

Complexity: $O(|E|)$ ☹

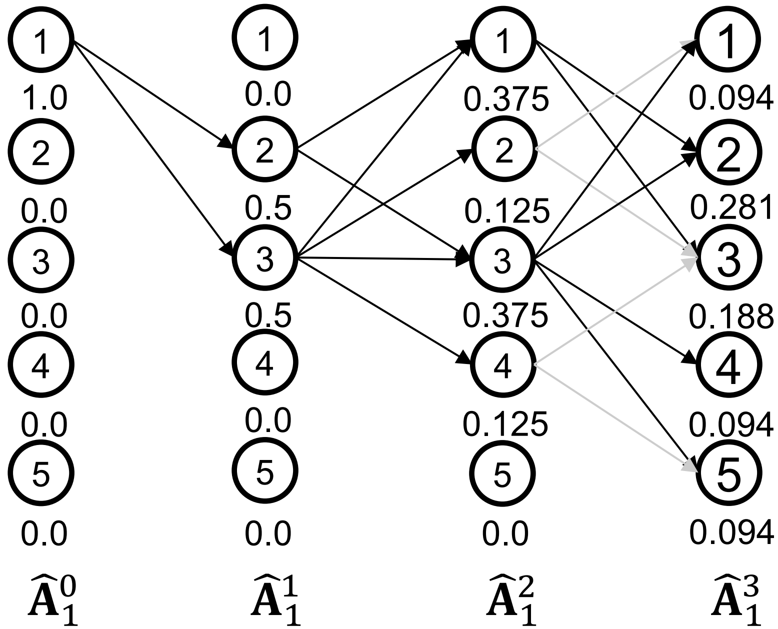
GRAND+：Matrix approximation

$$\mathbf{\Pi} = \frac{1}{N+1}\sum_{n=0}^{N}\hat{\mathbf{A}}^n$$

$\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$ is random walk reverse transition matrix. $\mathbf{P}(s, \mathrm{v})$ indicates the random walk probability from s to v.



$$\tilde{\mathbf{\Pi}}_{\mathbf{1}} = \frac{1}{N+1}\sum_{n=0}^{N}\hat{\mathbf{A}}_1^n$$



Generalized Forward Push (GFPush)

$r_{max} = 0.07$

$\cdots$

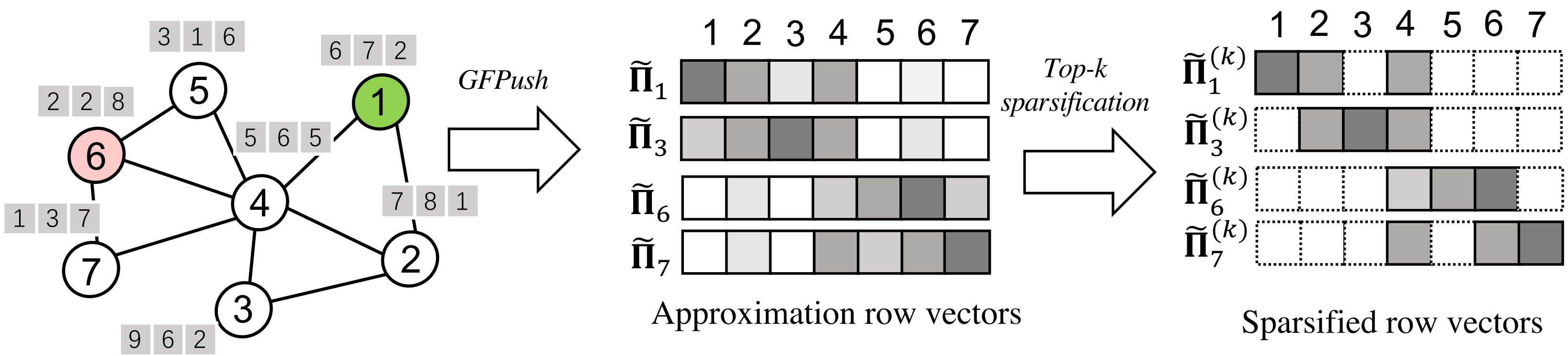
Error Bound：$T \cdot r_{max}$

Time Complexity：T / r_{max}

Memory Complexity：T / r_{max}

GRAND+: Matrix approximation

- Approximation method:
  - GFPush: Generate an error bounded approximation $\tilde{\boldsymbol{\Pi}}_{\mathbf{s}}$ for $\boldsymbol{\Pi}_{\mathbf{s}}$.
  - Top-*k* sparsification: Truncate $\tilde{\boldsymbol{\Pi}}_{\mathbf{s}}$ for top-*k* elements.



GRAND+: Mini-batch Radom Propagation

- Mini-batch Random Propagation with Approximation:

$$\overline{\mathbf{X}}_s^{(m)} = \sum_{v \in \mathcal{N}_v^{(k)}} \boldsymbol{z}_v \cdot \widetilde{\boldsymbol{\Pi}}^{(k)}(\mathrm{s}, \mathrm{v}) \cdot \mathbf{X}_v, \qquad \boldsymbol{z}_v \sim Bernoulli(1-\delta)$$

Non-zero elements in $\widetilde{\boldsymbol{\Pi}}_v^{(k)}$

- Prediction:

$$\hat{\mathbf{Y}}^{(m)} = \mathrm{MLP}(\overline{\mathbf{X}}_s^{(m)}, \Theta)$$

With batch size as b, the time complexity is $O(b \cdot k)$, which is independent of graph size

Scalability: Adopt GFPush to approximately calculate the propagation matrix ， and adopt mini-batch method for model training

GRAND+: Propagation matrix

- Propagation Matrix in GRAND:

$$\mathbf{\Pi} = \sum_{n=0}^{N} \frac{1}{N+1} \hat{\mathbf{A}}^{n}, \quad \hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1} \tilde{\mathbf{A}}$$

- Generalized Mixed-order Matrix:

$$\mathbf{\Pi} = \sum_{n=0}^{N} w_n \hat{\mathbf{A}}^{n}, \quad \hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1} \tilde{\mathbf{A}}$$

Flexibility: Using a set of tunable weights $\{w_t | 0 \leq t \leq T\}$ to control the importance of different orders of neighborhoods

Confidence-aware Consistency Regularization

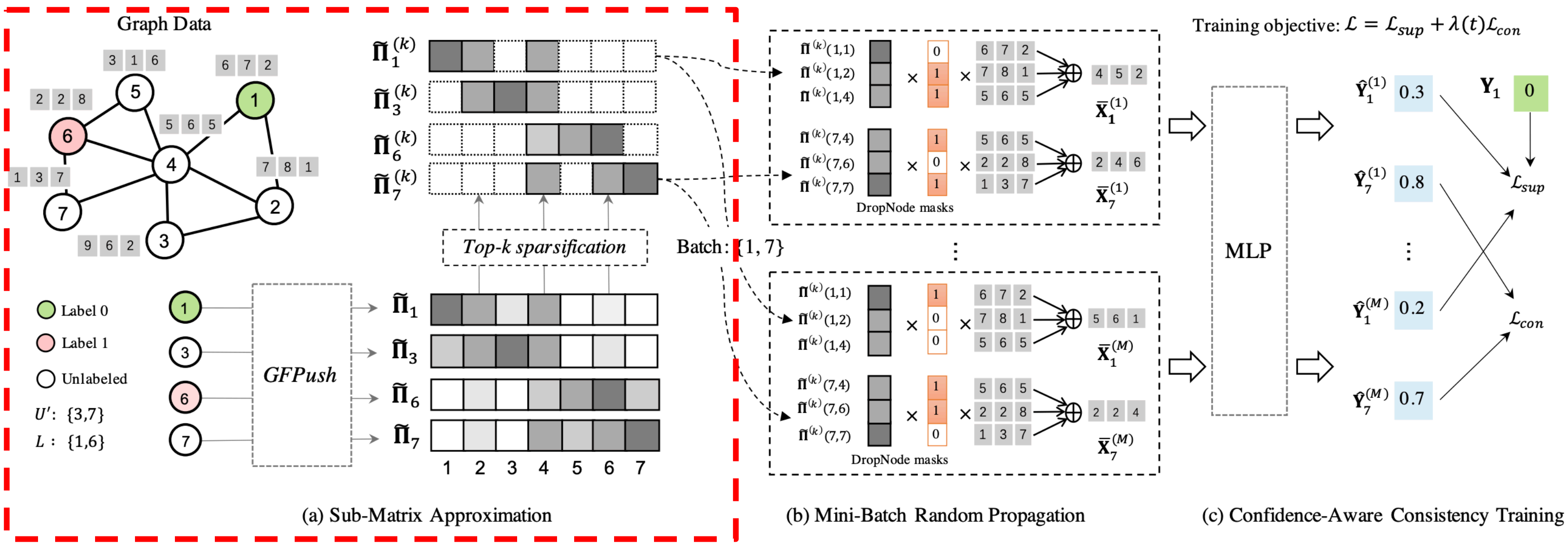
- Confidence-aware Consistency Loss：

$$\mathcal{L}_{con} = \frac{1}{b_u \cdot M} \sum_{s \in U_n} \mathbb{1}\left(\max(\overline{\mathrm{Y}}_s) \geq \gamma\right) \sum_{m=1}^{M} \mathcal{D}(\widetilde{\mathrm{Y}}_s, \hat{\mathrm{Y}}_s^{(m)}),$$

Confidence term: Filter out unlabeled nodes that have low confidence

Effectiveness: Further improving prediction performance

GRAND+ Architecture



Parallelization by OpenMP

GRAND+: Better scalability & generalization capability

GRAND+ Experiments

Table 2: Classification Accuracy (%) on Benchmarks.

Category	Method	Cora	Citeseer	Pubmed
Full-batch GNNs	GCN	81.5 ± 0.6	71.3 ± 0.4	79.1 ± 0.4
	GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
	APPNP	84.1 ± 0.3	71.6 ± 0.5	79.7 ± 0.3
	GCNII	85.5 ± 0.5	73.4 ± 0.6	80.3 ± 0.4
	GRAND	85.4 ± 0.4	75.4 ± 0.4	82.7 ± 0.6
Scalable GNNs	FastGCN	81.4 ± 0.5	68.8 ± 0.9	77.6 ± 0.5
	GraphSAINT	81.3 ± 0.4	70.5 ± 0.4	78.2 ± 0.8
	SGC	81.0 ± 0.1	71.8 ± 0.1	79.0 ± 0.1
	GBP	83.9 ± 0.7	72.9 ± 0.5	80.6 ± 0.4
	PPRGo	82.4 ± 0.2	71.3 ± 0.3	80.0 ± 0.4
Our Methods	GRAND+ (P)	**85.8 ± 0.4**	**75.6 ± 0.4**	84.5 ± 1.1
	GRAND+ (A)	85.5 ± 0.4	75.5 ± 0.4	**85.0 ± 0.6**
	GRAND+ (S)	85.0 ± 0.5	74.4 ± 0.5	84.2 ± 0.6

Better generalization performance: Achieves 2.3% improvements over GRAND on Pubmed.

GRAND+ Experiments

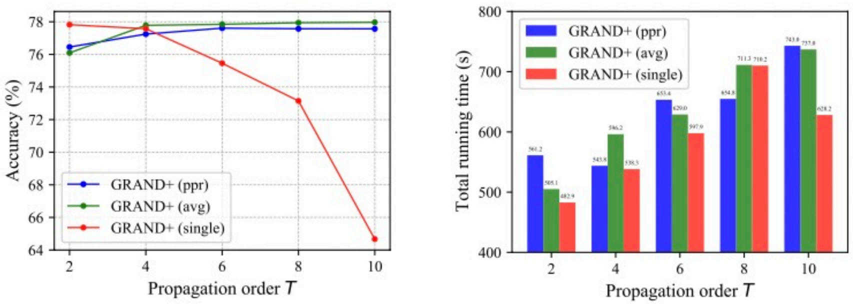
Table 3: Accuracy (%) and Running Time (s) on Large Graphs.

Method	AMiner-CS		Reddit		Amazon2M		MAG.	
	Acc	RT	Acc	RT	Acc	RT	Acc	RT
GRAND	53.1±1.1	750	OOM	–	OOM	–	OOM	–
FastGCN	48.9±1.6	69	89.6±0.6	158	72.9±1.0	239	64.3±5.6	4220
GraphSAINT	51.8±1.3	39	92.1±0.5	39	75.9±1.3	189	75.0±1.7	6009
SGC	50.2±1.2	9	92.5±0.2	31	74.9±0.5	69	–	>24h
GBP	52.7±1.7	21	88.7±1.1	370	70.1±0.9	280	–	>24h
PPRGo	51.2±1.4	11	91.3±0.2	233	67.6±0.5	160	72.9±1.1	434
GRAND+ (P)	53.9±1.8	17	93.3±0.2	183	75.6±0.7	188	77.6±1.2	653
GRAND+ (A)	54.2±1.7	14	**93.5±0.2**	174	75.9±0.7	136	**80.0±1.1**	737
GRAND+ (S)	**54.2±1.6**	10	92.8±0.2	62	**76.2±0.6**	80	77.8±0.9	483

Scalability:

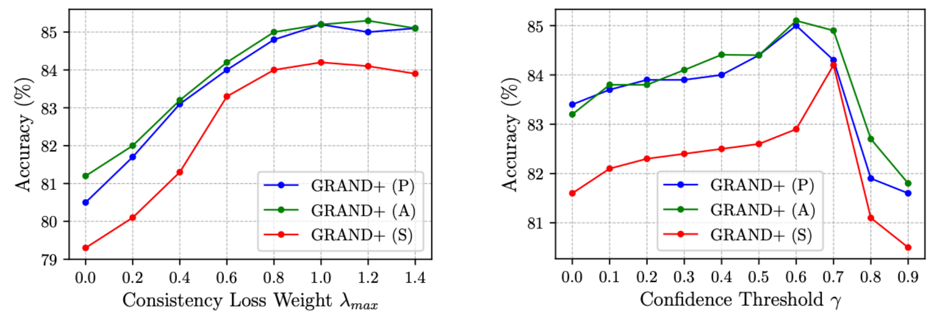
- 40 times faster than GRAND on Aminer-CS.
- 8 times faster than FastGCN on MAG.
- 12 times faster than GraphSAINT on MAG.
- Achieves comparable running time and 4.9% improvement than PPRGo on MAG.

Parameter Analysis



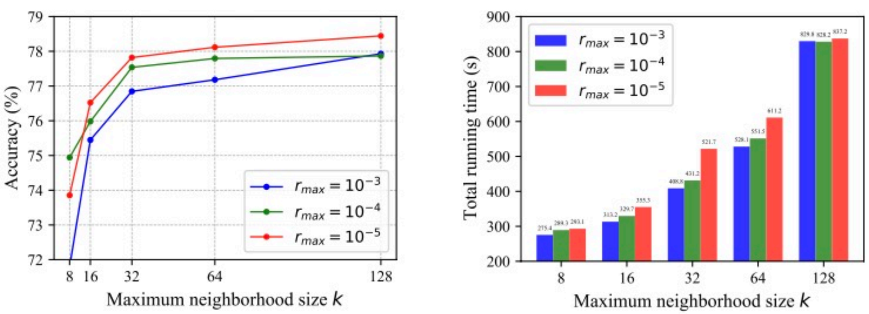
(a) Classification accuracy. (b) Total running time.

Figure 5: Effects of propagation order T on MAG-Scholar-C.



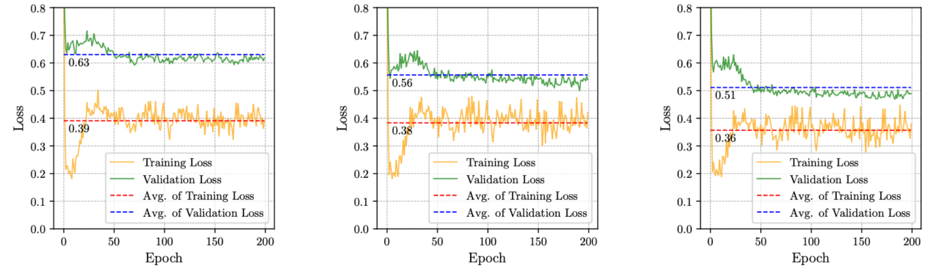
(a) Accuracy w.r.t. λ_{max}. (b) Accuracy w.r.t. γ

Figure 2: Effects of λ_{max} and γ on Pubmed.



(a) Classification accuracy. (b) Total running time.

Figure 4: GRAND+ w.r.t. k and r_{max} on MAG-Scholar-C.

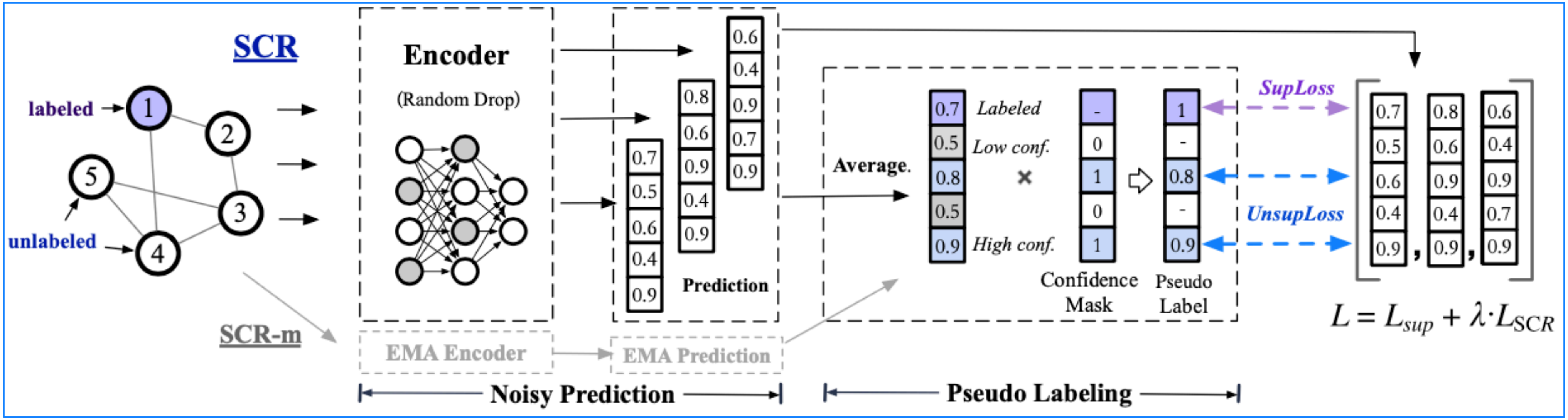


(a) $\lambda_{max} = 0.0$ (b) $\lambda_{max} = 1.0, \gamma = 0.0$ (c) $\lambda_{max} = 1.0, \gamma = 0.6$

Figure 3: Training and Validation Losses on Pubmed.

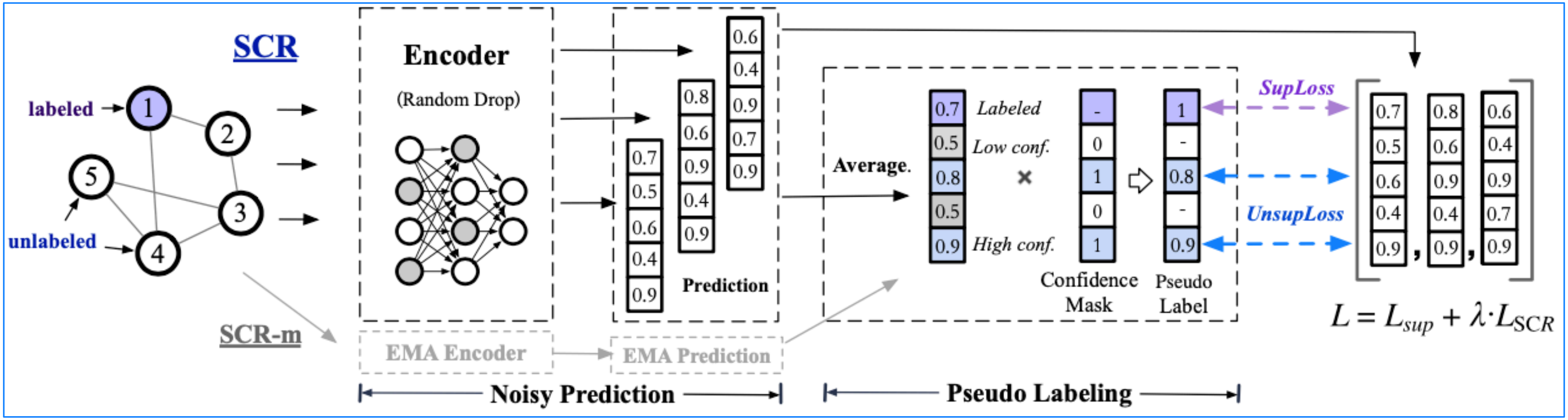
SCR: Training Graph Neural Networks with Consistency Regularization

The SCR Framework



- **Noisy Prediction Generation**: Get multiple predictions with different dropout masks
- Pseudo Labeling: obtain pseudo labels for unlabeled data
  - SCR: averages the noisy predictions
  - SCR-m: exploits an EMA teacher encoder

The SCR Framework



$$\mathcal{L} = \frac{1}{S \cdot N_L} \sum_{i \in \mathcal{B}_L} \sum_{s=1}^{S} \texttt{CrossEntropy}(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i^{(s)}) + \frac{\lambda}{S \cdot N_U} \sum_{i \in \mathcal{B}_U} \sum_{s=1}^{S} \texttt{dist}\left(\bar{\boldsymbol{y}}_i^{(sharp)}, \hat{\boldsymbol{y}}_i^{(s)}\right)$$

Pseudo label sharpening: $\bar{\boldsymbol{y}}^{(sharp)}[c] = \texttt{sharpen}(\bar{\boldsymbol{y}})[c] = \frac{(\bar{\boldsymbol{y}}[c])^{1/T}}{\sum_{c'=1}^{C} (\bar{\boldsymbol{y}}[c'])^{1/T}}$

SCR Results on OGB.

Table 2: Classification accuracy on ogbn-products. Results with gray are obtained by our proposed framework.

Methods	Arch.	C&S	Validation	Test
-	MLP	-	75.54 ± 0.14	61.06 ± 0.08
-	MLP	✓	91.47 ± 0.09	84.18 ± 0.07
-	GCN	-	92.00 ± 0.03	75.64 ± 0.21
-	GraphSAGE	-	92.24 ± 0.07	78.50 ± 0.14
-	SIGN	-	92.99 ± 0.04	80.52 ± 0.16
-	SAGN	-	93.09 ± 0.04	81.20 ± 0.07
-	GAMLP	-	93.12 ± 0.03	83.54 ± 0.09
SLE	SAGN	-	93.09 ± 0.07	84.68 ± 0.12
SLE	SAGN	✓	93.02 ± 0.03	84.85 ± 0.10
RLU	GAMLP	-	93.24 ± 0.05	84.59 ± 0.10
SCR	GAMLP	-	93.30 ± 0.06	84.07 ± 0.06
SCR-m	GAMLP	-	93.19 ± 0.03	84.62 ± 0.03
RLU + SCR	GAMLP	-	92.92 ± 0.05	85.05 ± 0.09
RLU + SCR	GAMLP	✓	93.04 ± 0.05	**85.20** ± 0.08
using node features generated by GIANT-XRT				
SLE	SAGN	-	93.63 ± 0.05	86.22 ± 0.22
SLE	SAGN	✓	93.52 ± 0.05	86.43 ± 0.20
SCR	SAGN	-	93.64 ± 0.05	86.67 ± 0.09
SCR	SAGN	✓	93.57 ± 0.04	**86.80** ± 0.07
SCR-m	SAGN	-	93.89 ± 0.02	86.51 ± 0.09
SCR-m	SAGN	✓	93.87 ± 0.02	**86.73** ± 0.08

+0.37%

Table 3: Classification accuracy on ogbn-mag. Results with gray are obtained by our proposed framework.

Methods	Arch.	Validation	Test
-	MLP	26.26 ± 0.16	26.92 ± 0.26
-	R-GCN	40.84 ± 0.41	39.77 ± 0.46
-	SIGN	40.68 ± 0.10	40.46 ± 0.12
-	NARS	53.72 ± 0.09	52.40 ± 0.16
-	NARS_SAGN	54.12 ± 0.15	52.32 ± 0.25
-	NARS_GAMLP	55.48 ± 0.08	53.96 ± 0.18
SLE	NARS_SAGN	55.91 ± 0.17	54.40 ± 0.15
RLU	NARS_GAMLP	57.02 ± 0.41	55.90 ± 0.27
SCR	NARS_GAMLP	56.54 ± 0.21	54.32 ± 0.18
SCR-m	NARS_GAMLP	55.90 ± 0.28	54.51 ± 0.19
RLU + SCR	NARS_GAMLP	57.34 ± 0.35	**56.31** ± 0.21

+0.41%

Table 4: Classification accuracy on ogbn-papers100M. Results with gray are obtained by our proposed framework.

Methods	Arch.	Validation	Test
-	MLP	49.60 ± 0.29	47.24 ± 0.31
-	SGC	66.48 ± 0.20	63.29 ± 0.19
-	SIGN	69.32 ± 0.06	65.68 ± 0.06
-	SIGN-XL	70.32 ± 0.11	67.06 ± 0.17
-	SAGN	70.34 ± 0.99	66.75 ± 0.84
-	GAMLP	71.17 ± 0.14	67.71 ± 0.20
SLE	SAGN	71.63 ± 0.07	68.30 ± 0.08
RLU	GAMLP	71.59 ± 0.05	68.25 ± 0.11
SCR	GAMLP	71.90 ± 0.07	68.14 ± 0.08
SCR-m	GAMLP	71.86 ± 0.08	68.16 ± 0.12
RLU + SCR	GAMLP	71.88 ± 0.07	**68.42** ± 0.15

+0.12%

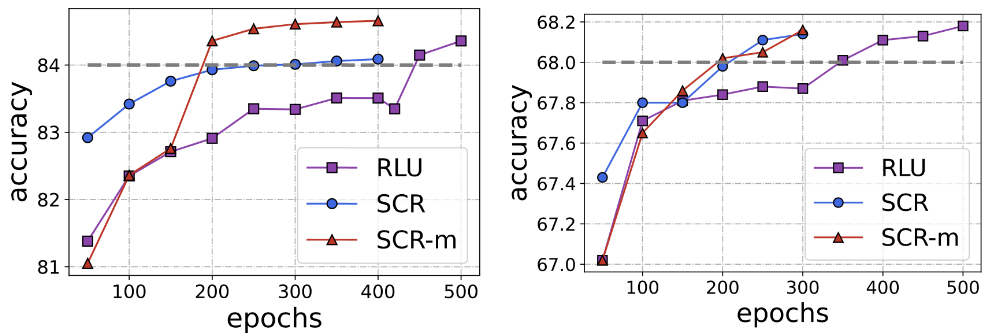
Experimental Results

Table 5: Classification accuracy with different GNNs as the base encoder on the ogbn-products dataset.

Methods	Arch.	Validation	Test
-	MLP	75.54 ± 0.14	61.06 ± 0.08
SCR	MLP	76.51 ± 0.08	62.83 ± 0.14 (+1.77)
-	GraphSAGE	92.14 ± 0.09	78.75 ± 0.38
SCR	GraphSAGE	92.30 ± 0.05	79.82 ± 0.39 (+1.07)
-	ClusterGCN	91.66 ± 0.06	78.37 ± 0.49
SCR	ClusterGCN	91.86 ± 0.08	78.72 ± 0.23 (+0.35)
-	GraphSAINT	91.67 ± 0.07	79.17 ± 0.30
SCR	GraphSAINT	91.71 ± 0.06	80.01 ± 0.33 (+0.84)
-	SIGN	92.99 ± 0.04	80.52 ± 0.16
SCR	SIGN	92.96 ± 0.08	81.64 ± 0.07 (+1.12)

Applicability to various GNN architectures.

Comparison with RLU.



(a) ogbn-products **(b) ogbn-papers100M**

SCR converges faster than multi-stage self-training.

Summary

- GRAND
  - non-robust, over-smoothing, over-fit problems
  - random propagation + consistency regularization
- GRAND+
  - scalable version of GRAND
  - mini-batch random propagation with approximation
- SCR
  - simple and general method for GNNs

Homework 6: GRAND Implementation

- Experiments on GRAND:
  - Due by 21st Aug.
  - Implement GRAND with **consistency regularization**
  - Test GRAND on the cora dataset
  - Discuss on consistency regularization
- Find the homework material from the course website: https://cogdl.ai/gnn2022/

- Bonus: post your discussion to: https://discuss.cogdl.ai/t/topic/83 .

Thank you！

Collaborators:

Zhenyu Hou, Yuxiao Dong, Jie Tang, et al. (**THU**)

Qingfei Zhao, Xinije Zhang, Peng Zhang, et al. (**Zhipu AI**)

Hongxiao Yang, Chang Zhou, et al. (**Alibaba**)

Yang Yang (**ZJU**)

Yukuo Cen, KEG, Tsinghua U.	https://github.com/THUDM/cogdl
Online Discussion Forum	https://discuss.cogdl.ai/