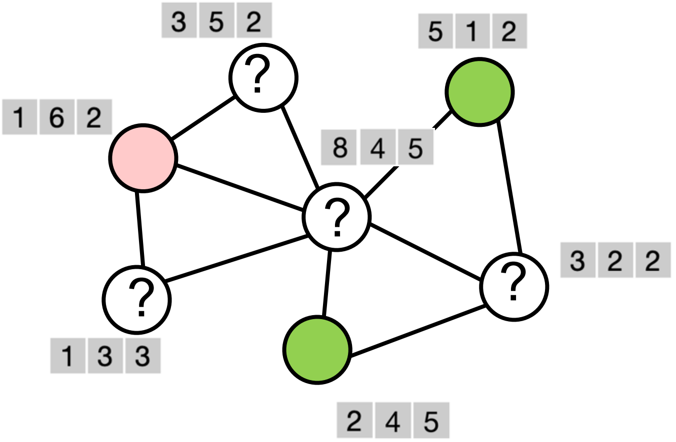
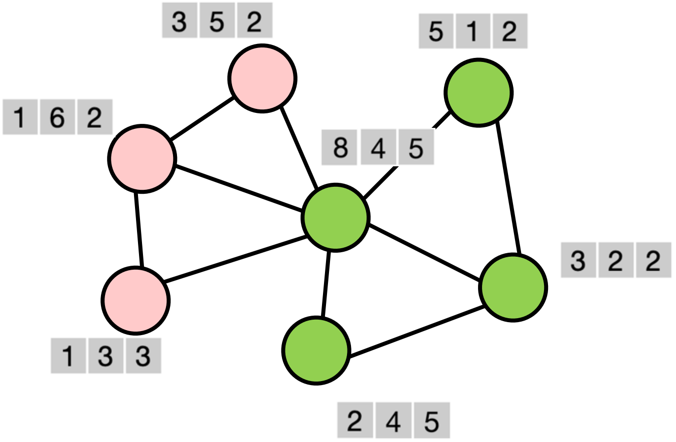
Graph Random Neural Network for Semi-Supervised Learning on Graphs

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan,
Qian Xu, Qiang Yang, Evgeny Kharlamov, Jie Tang

Semi-Supervised Learning on Graphs



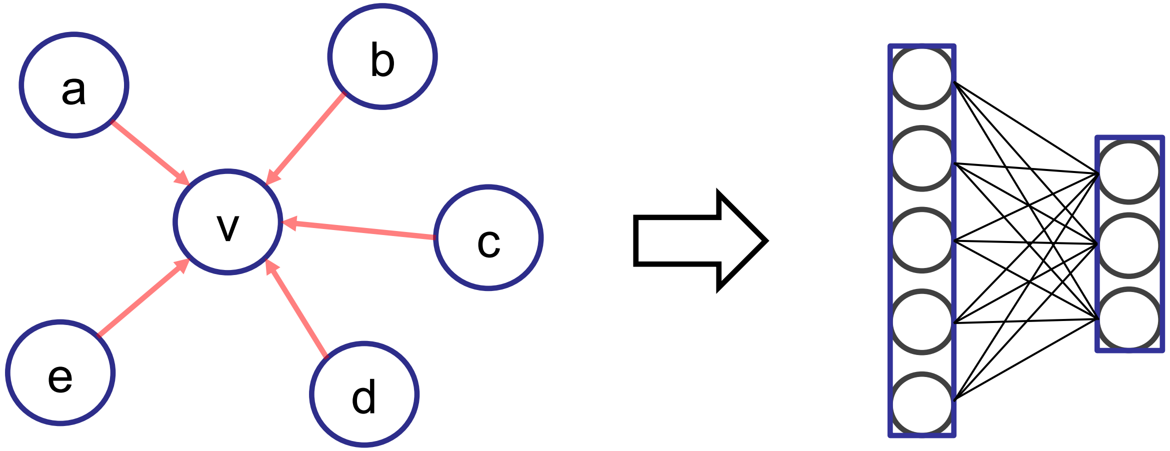


Input: a partially labeled & attributed graph

Output: infer the labels of unlabeled nodes

Graph Neural Network (GNN)

Graph Convolution Network:



node v's embedding at $k+1$

non-linear activation function (e.g. ReLU)

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

normalized Laplacian matrix

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

the neighbors of node v

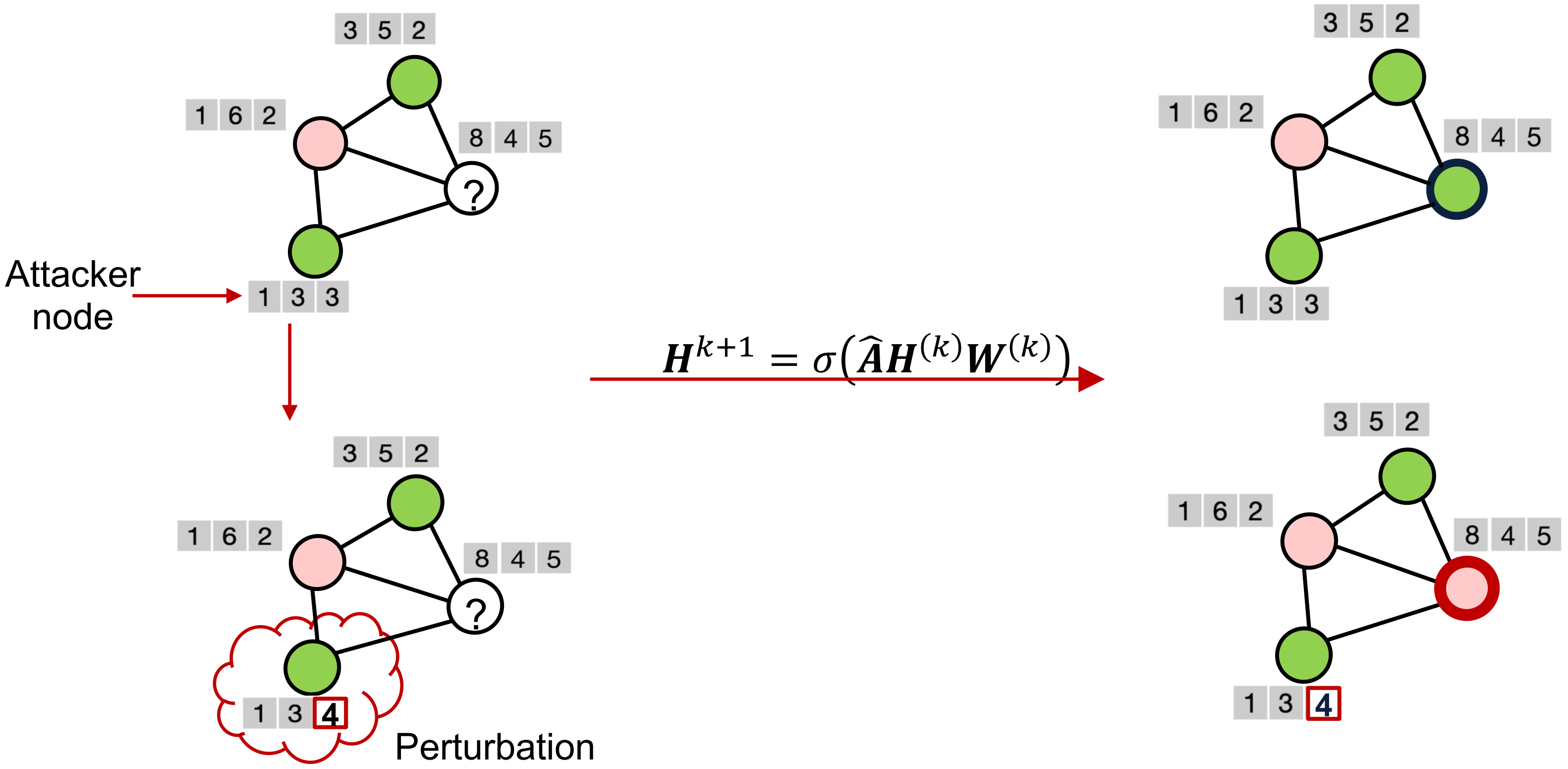
- Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In ICLR 2017

Graph Neural Networks

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

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.

Graph Neural Networks

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

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

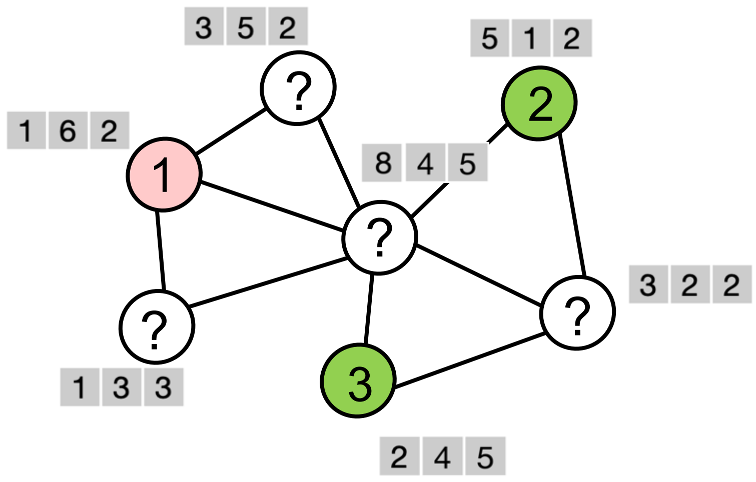
1. Each node is highly dependent with its neighborhoods, making GNNs **non-robust** to noises
2. Stacking many GNNs layers may cause **over-smoothing.**

- 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.

Graph Neural Networks

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

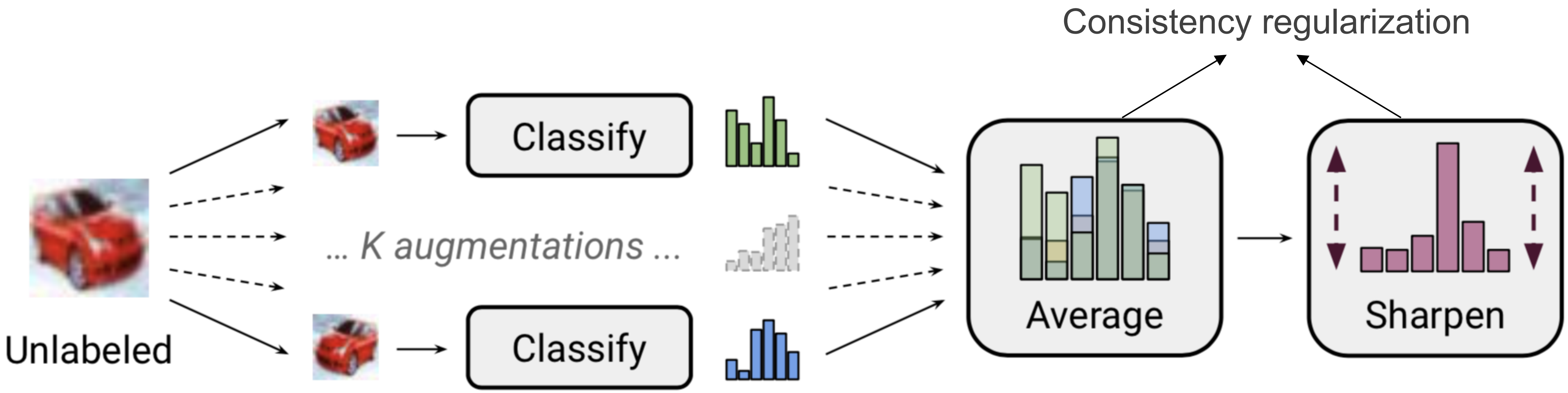
Loss function:

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

Cannot fully leverage unlabeled data

Recent advances in Semi-Supervised Image Classification

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

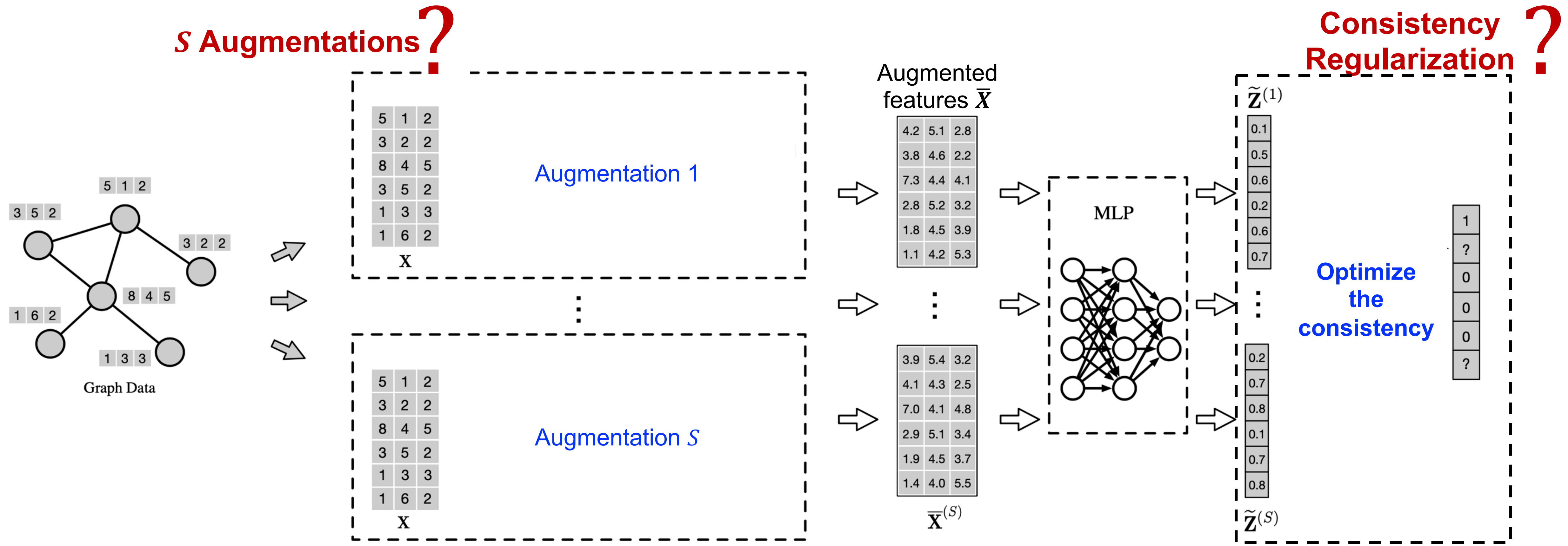


(Picture from MixMatch's paper)

- 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.



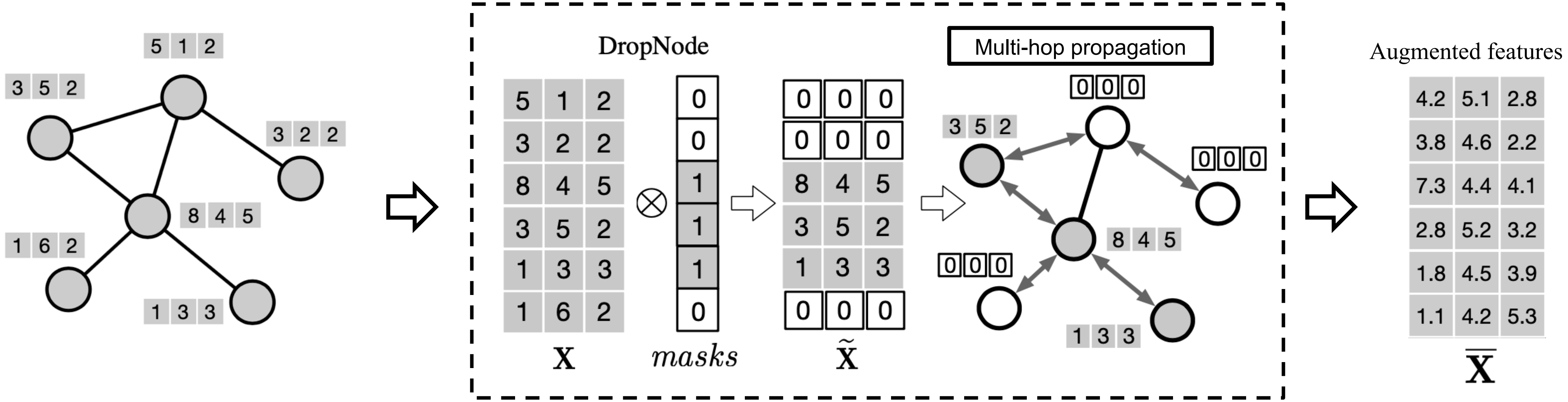
- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. https://arxiv.org/abs/2005.11079, 2020
- Code & data for Grand: https://github.com/THUDM/GRAND

Graph Random Neural Network (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.

Random Propagation

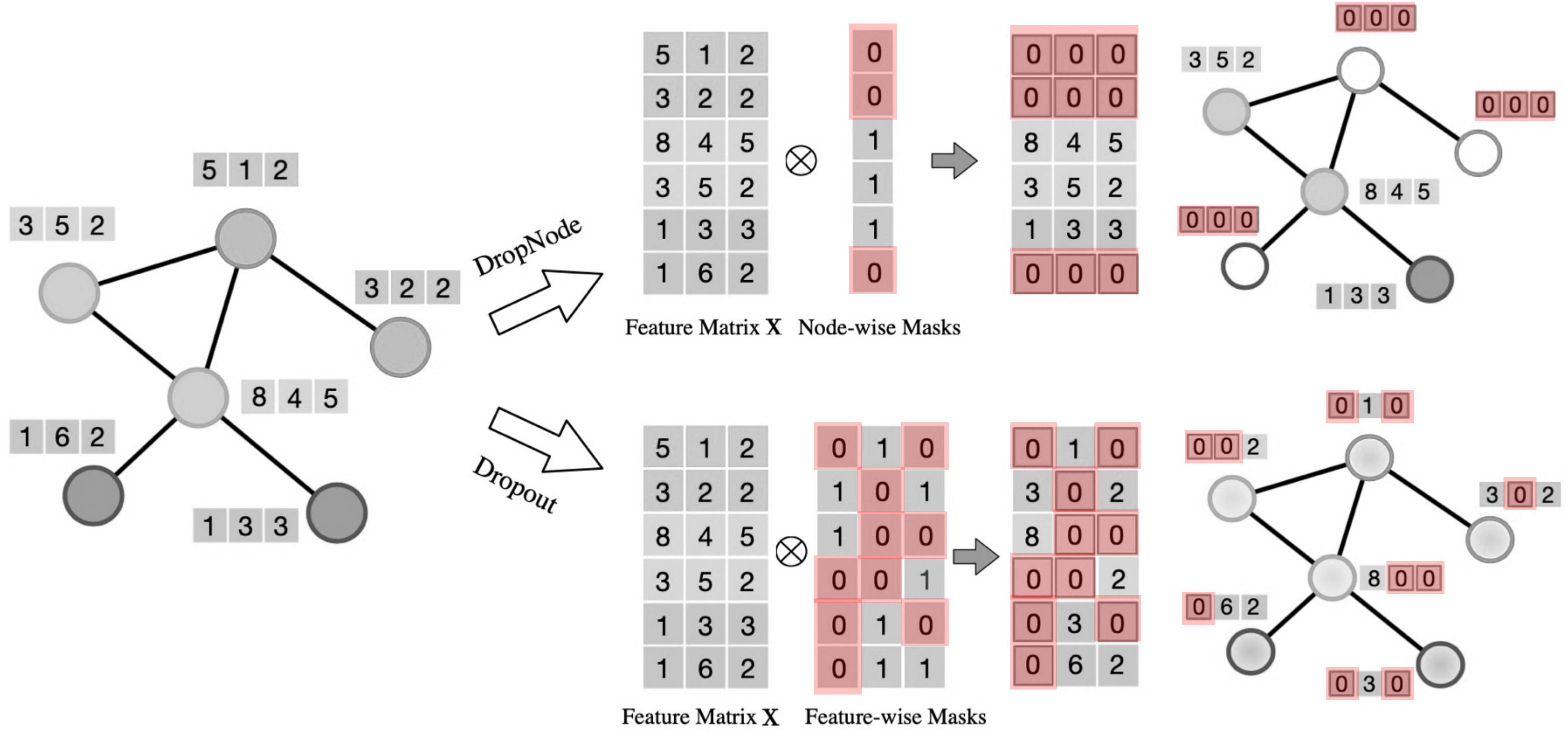
$$\overline{X} = \frac{\sum_{i=0}^{K} A^i X}{K+1}$$



- 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/THUDM/GRAND

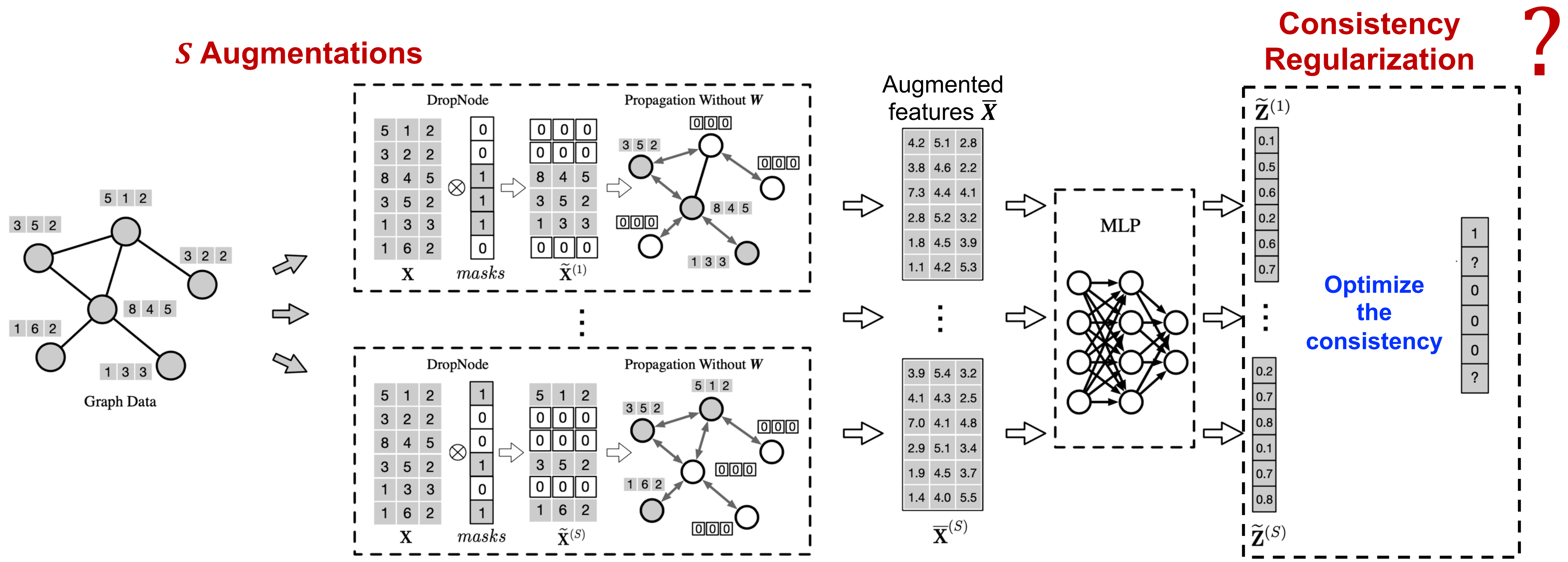
Random propagation: DropNode vs Dropout

- 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/THUDM/GRAND

Graph Random Neural Network (GRAND)



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/THUDM/GRAND https://github.com/THUDM/GRAND

GRAND: Consistency Regularization

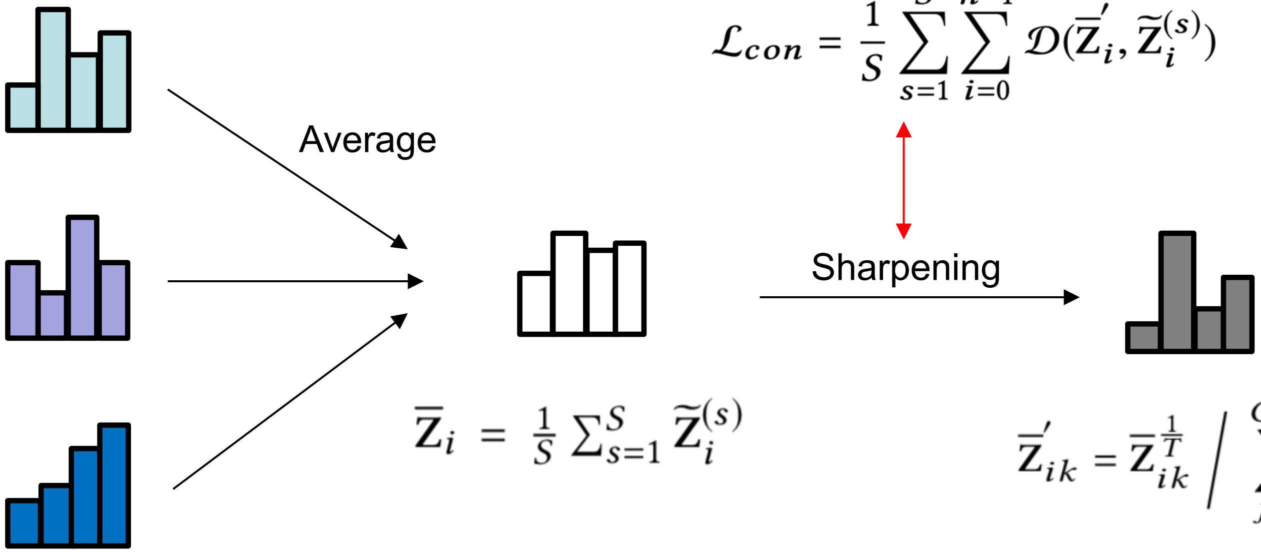
$$\mathcal{L}_{sup} = -\frac{1}{S}\sum_{s=1}^{S}\sum_{i=0}^{m-1}\mathbf{Y}_i^\top \log \widetilde{\mathbf{Z}}_i^{(s)}$$

$+$

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

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

Distributions of a node after augmentations



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

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

- 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/THUDM/GRAND

Graph Random Neural Networks (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 $\mathbf{Z}$.

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

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

end while
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/THUDM/GRAND

Graph Random Neural Network (GRAND)

- With Consistency Regularization Loss:
  - Random propagation can enforce the consistency of the classification confidence between each node and its all multi-hop neighborhoods.

$$\mathbb{E}_\epsilon\left(\mathcal{L}_{con}\right) \approx \mathcal{R}^c(\mathbf{W}) = \sum_{i=0}^{n-1} z_i^2(1-z_i)^2 \mathrm{Var}_\epsilon\left(\overline{\mathbf{A}}_i \widetilde{\mathbf{X}} \cdot \mathbf{W}\right)$$

$$\mathcal{R}^c_{DN}(\mathbf{W}) = \frac{\delta}{1-\delta} \sum_{j=0}^{n-1} \left[(\mathbf{X}_j \cdot \mathbf{W})^2 \sum_{i=0}^{n-1} (\overline{\mathbf{A}}_{ij})^2 z_i^2 (1-z_i)^2 \right]$$

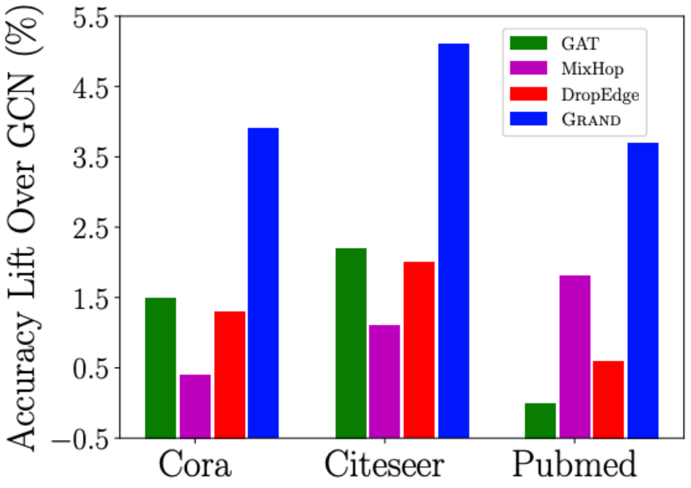
$$\mathcal{R}^c_{Do}(\mathbf{W}) = \frac{\delta}{1-\delta} \sum_{h=0}^{d-1} \mathbf{W}_h^2 \sum_{j=0}^{n-1} \left[\mathbf{X}_{jh}^2 \sum_{i=0}^{n-1} z_i^2 (1-z_i)^2 (\overline{\mathbf{A}}_{ij})^2 \right]$$

- With Supervised Cross-Entropy Loss:
  - Random propagation can enforce the consistency of the classification confidence between each node and its labeled multi-hop neighborhoods.

- 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/THUDM/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/THUDM/GRAND

Results

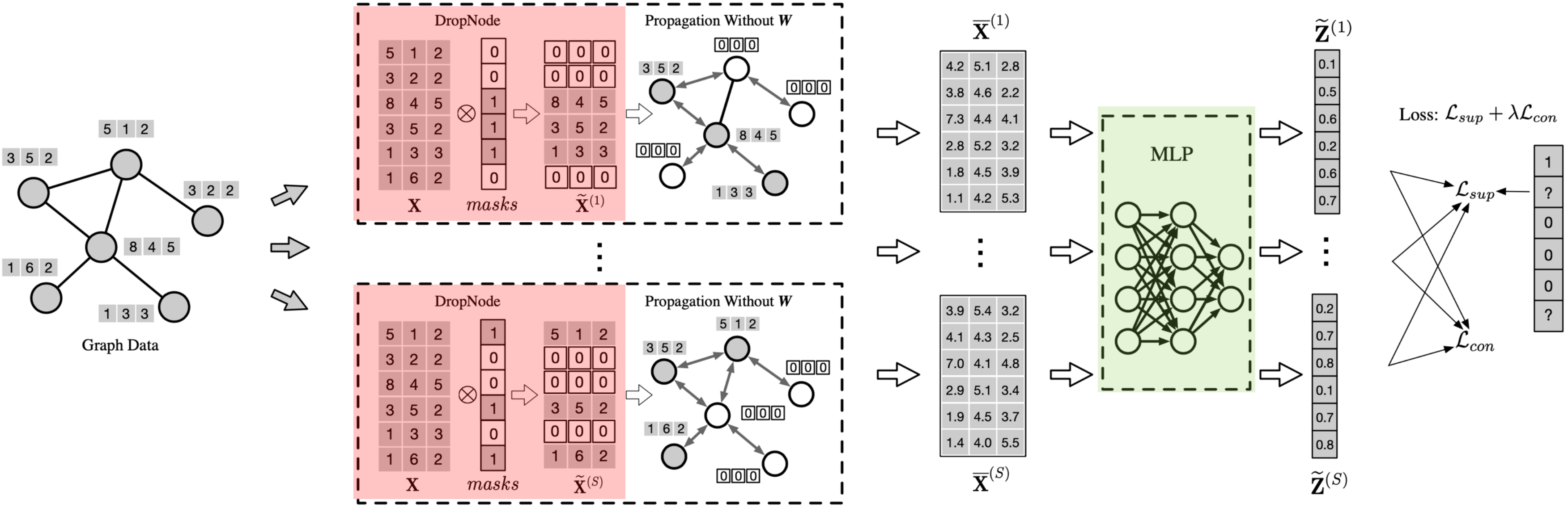
Table 5: Results on large datasets.

Method	Cora Full	Coauthor CS	Coauthor Physics	Amazon Computer	Amazon Photo	Citation CS
GCN	62.2 ± 0.6	91.1 ± 0.5	92.8 ± 1.0	82.6 ± 2.4	91.2 ± 1.2	49.9 ± 2.0
GAT	51.9 ± 1.5	90.5 ± 0.6	92.5 ± 0.9	78.0 ± 19.0	85.7 ± 20.3	49.6 ± 1.7
GRAND	**63.5 ±0.6**	**92.9 ± 0.5**	**94.6 ± 0.5**	**85.7 ± 1.8**	**92.5 ± 1.7**	**52.8 ± 1.2**

More experiments on larger graph 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/THUDM/GRAND

Results



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

Results

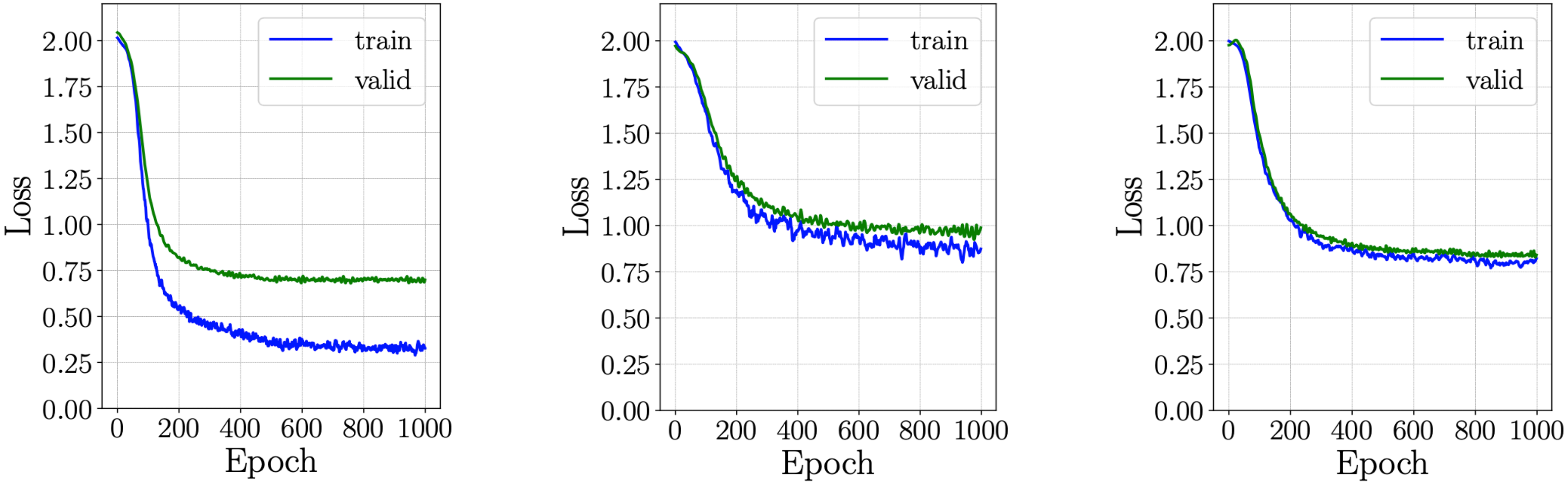
Method	Cora	Citeseer	Pubmed
GCN [19]	81.5	70.3	79.0
GAT [32]	83.0±0.7	72.5±0.7	79.0±0.3
APPNP [20]	83.8±0.3	71.6± 0.5	79.7 ± 0.3
Graph U-Net [11]	84.4±0.6	73.2±0.5	79.6±0.2
SGC [36]	81.0 ±0.0	71.9 ± 0.1	78.9 ± 0.0
MixHop [1]	81.9± 0.4	71.4±0.8	80.8±0.6
GMNN [28]	83.7	72.9	81.8
GraphNAS [12]	84.2±1.0	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/THUDM/GRAND

Results



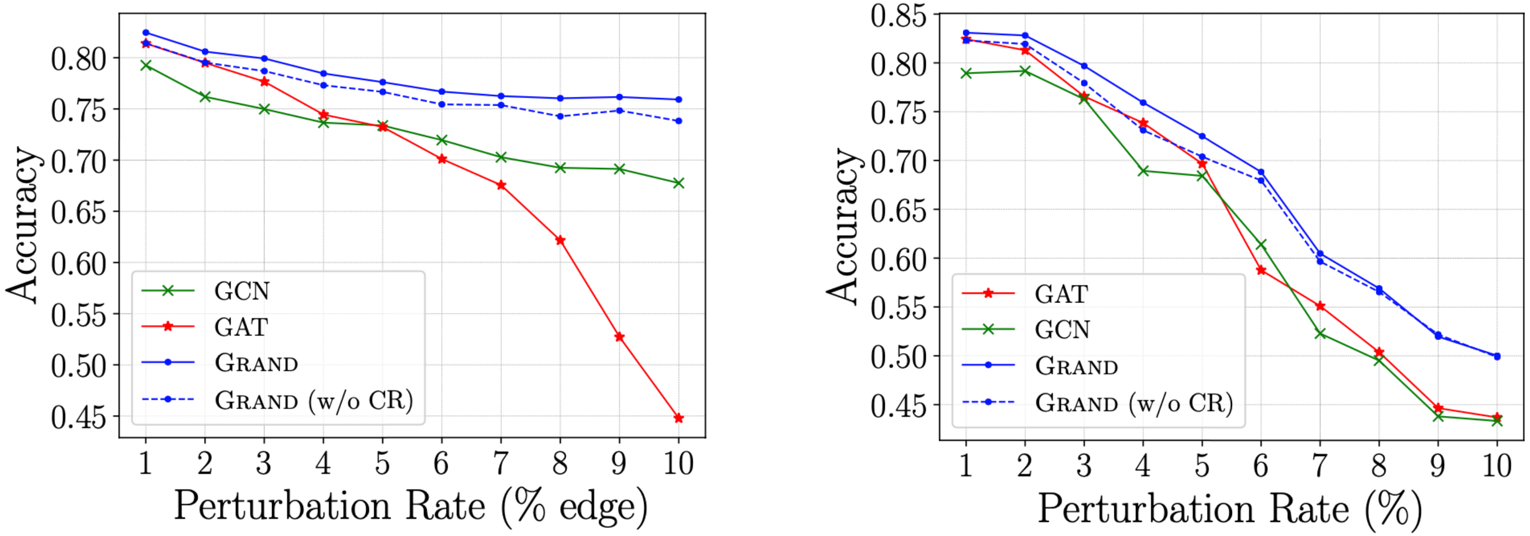
(a) Without CR and RP (b) Without CR (c) GRAND

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/THUDM/GRAND

Results



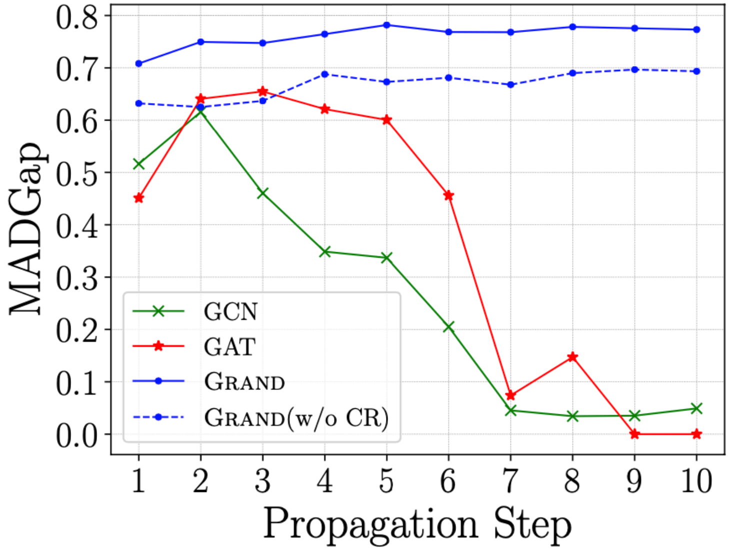
(a) Random Attack (b) Metattack

Robustness

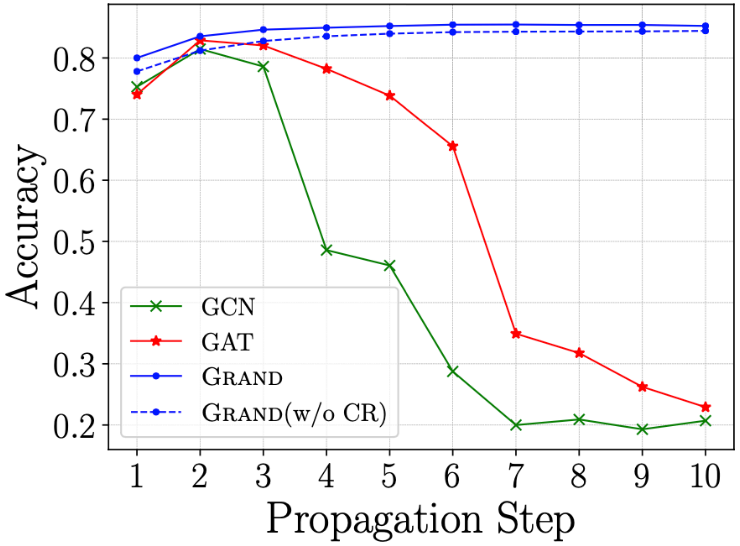
1. GRAND (with or w/o) consistency regularization is more robust than GCN and GAT.

- 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/THUDM/GRAND

Results



(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/THUDM/GRAND

Thanks!

Code & data for GRAND: https://github.com/THUDM/GRAND

Wechat: