# Graph Random Neural Networks for Semi-Supervised Learning on Graphs

School of Industrial and Management Engineering, Korea University

Eun Ji Koh

KOREA UNIVERSITY

DMQA

# Contents

# Research Purpose

❖ Graph Random Neural Networks for Semi-Supervised Learning on Graphs (NeurIPS, 2020)

- 2022년 03월 18일 기준으로 79회 인용됨

## Graph Random Neural Networks for Semi-Supervised Learning on Graphs

**Wenzheng Feng**[1*], **Jie Zhang**[2*‡], **Yuxiao Dong**[3], **Yu Han**[1], **Huanbo Luan**[1], **Qian Xu**[2], **Qiang Yang**[2], **Evgeny Kharlamov**[4], **Jie Tang**[1§]
[1] Department of Computer Science and Technology, Tsinghua University
[2]WeBank Co., Ltd [3]Microsoft Research [4] Bosch Center for Artificial Intelligence
fwz17@mails.tsinghua.edu.cn, {zhangjie.exe, ericdongyx, yhanthu, luanhuanbo}@gmail.com
{qianxu, qiangyang}@webank.com, evgeny.kharlamov@de.bosch.com, jietang@tsinghua.edu.cn

**Abstract**

We study the problem of semi-supervised learning on graphs, for which graph neural networks (GNNs) have been extensively explored. However, most existing GNNs inherently suffer from the limitations of over-smoothing [7, 27, 28, 34], non-robustness [54, 51], and weak-generalization when labeled nodes are scarce. In this paper, we propose a simple yet effective framework—GRAPH RANDOM NEURAL NETWORKS (GRAND)—to address these issues. In GRAND, we first design a random propagation strategy to perform graph data augmentation. Then we leverage consistency regularization to optimize the prediction consistency of unlabeled nodes across different data augmentations. Extensive experiments on graph benchmark datasets suggest that GRAND significantly outperforms state-of-the-art GNN baselines on semi-supervised node classification. Finally, we show that GRAND mitigates the issues of over-smoothing and non-robustness, exhibiting better generalization behavior than existing GNNs. The source code of GRAND is publicly available at `https://github.com/Grand20/grand`.

# Research Purpose

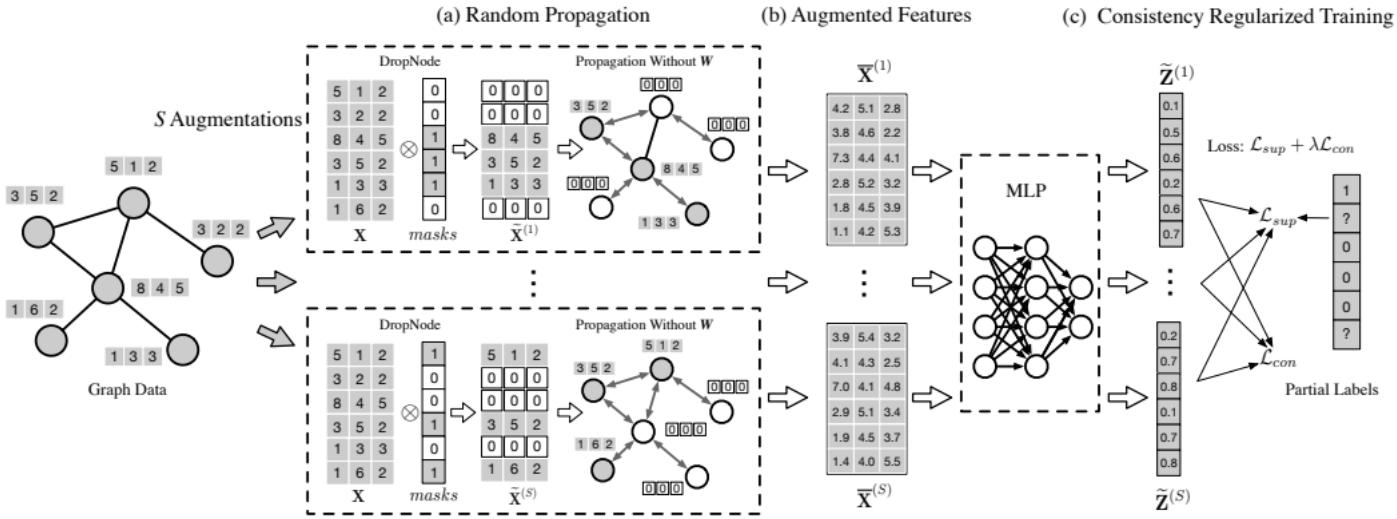
❖ Graph Random Neural Networks for Semi-Supervised Learning on Graphs (NeurIPS, 2020)

- Given graph의 labeled node가 적은 상황에서 semi-supervised learning을 통해 node 카테고리 예측 문제 수행
- Semi-supervised learning을 위해 주로 사용되던 GNN의 한계 3가지
  - ➢ *Over-smoothing*: graph convolution은 Laplacian smoothing의 특별한 형태에 해당하기 때문에 GNN layer 를 많이 쌓으면 node들의 feature가 indistinguishable 해짐
  - ➢ *Not robust to graph attack*: propagation으로 인해 multi-hop 내의 노드끼리 의존적이게 함
  - ➢ *Easily overfit*: Train 과정에서 부족한 labeled data에 overfit 될 수 있음
- Unlabeled data를 효과적으로 사용하는 semi-supervised learning을 통해 위의 문제점을 핸들링하고자 함
  - ➢ Vision 분야에서 augmentation을 사용한 방법론 (MixMatch, UDA 등 )을 통해 성공적으로 unlabeled data를 활용한 것에서 아이디어를 얻음

# Graph Random Neural Networks (GRAND)

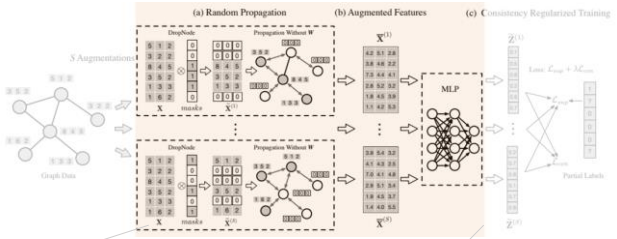
## ❖ GRAND architecture

- 그래프 기반의 semi-supervised learning을 위한 Graph Random Neural Networks(GRAND) 제안
- GRAND 작동 원리의 핵심
  - ➢ Graph augmentation을 위한 random propagation 모듈
  - ➢ Consistency regularized training
  - ➢ Semi-supervised learning을 위한 setting 하에서 generalization capacity 향상



# Graph Random Neural Networks (GRAND)

❖ Random Propagation for Graph Data Augmentation

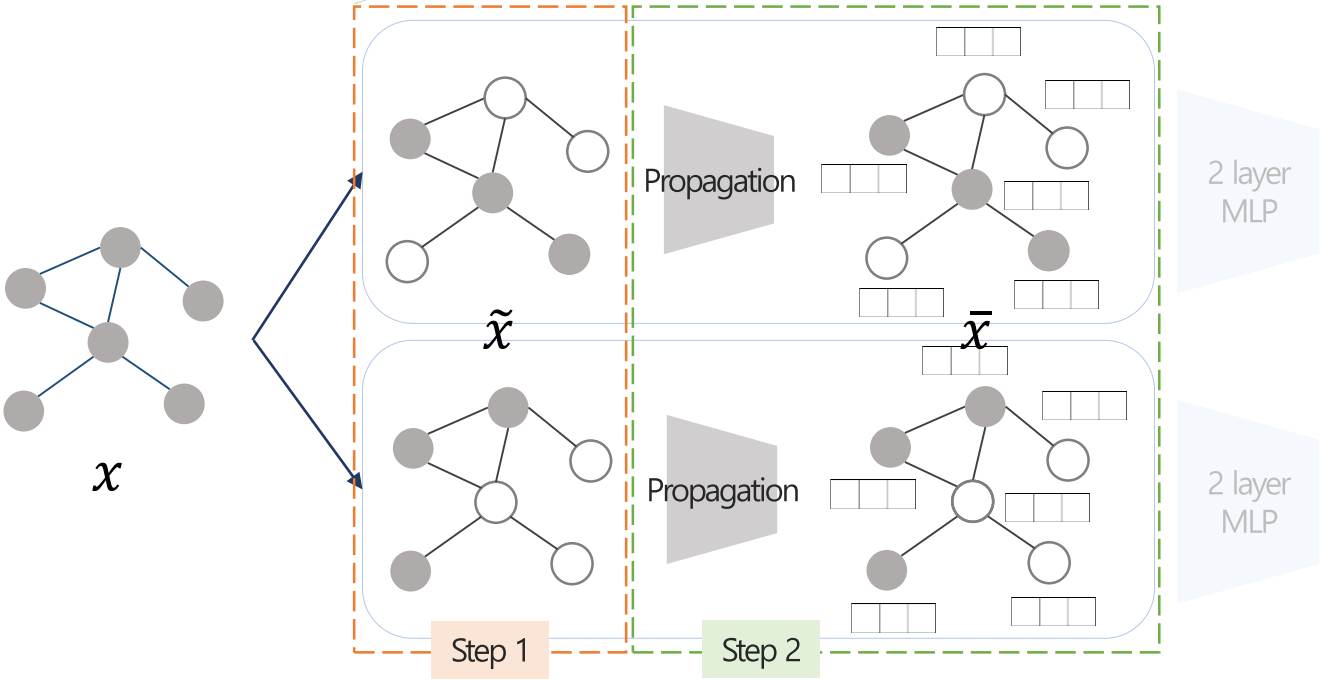


- Random propagation은 2 step으로 진행

**Step 1) 그래프 augmentation (generate perturbed feature)**

- Training 단계에서만 수행

**Step 2) Feature propagation**





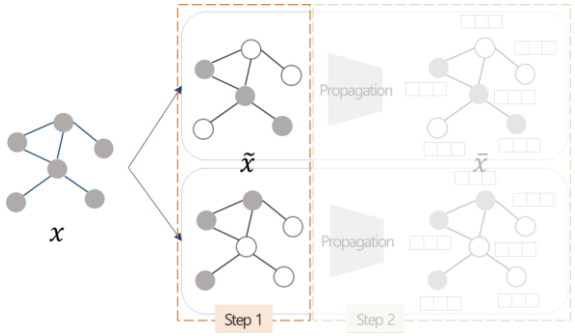



DMQA

# Graph Random Neural Networks (GRAND)

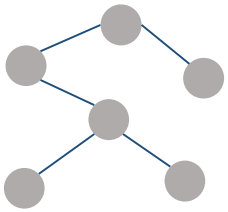
## ❖ Random Propagation for Graph Data Augmentation

- Random propagation은 2 step으로 진행




**Step 1) 그래프 augmentation (generate perturbed feature)** $\tilde{X} = \epsilon_i \cdot \mathbf{X}$

- Randomly drop을 통한 그래프 augmentation
- Randomly drop 방식으로는 DropNode 방식을 사용
  - ➢ 노드 feature의 일부 element를 지우는 dropout과 달리 노드의 전체 feature를 지우는(masking) 방식
  - ➢ DropNode 방식은 해당 node의 정보를 완전히 지우고 이웃 node들의 정보들을 재구성하므로 특정 이웃에 대한 의존도가 낮아져 모델을 더욱 강건하게 하는 효과가 있음(실험적으로 입증)



Given graph

| | | |
|---|---|---|
| 5 | 1 | 2 |
| 3 | 2 | 2 |
| 8 | 4 | 5 |
| 3 | 5 | 2 |
| 1 | 3 | 3 |
| 1 | 6 | 2 |

Feature $X$

$\otimes$

| |
|---|
| 0 |
| 0 |
| 1 |
| 1 |
| 1 |
| 0 |

Masking $*\epsilon_i$

$=$

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 8 | 4 | 5 |
| 3 | 5 | 2 |
| 1 | 3 | 3 |
| 0 | 0 | 0 |

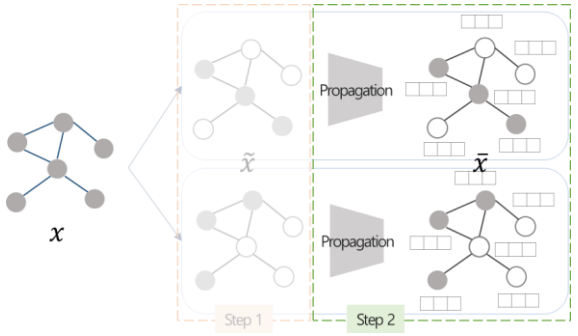
Perturbed feature $\tilde{X}$

$*\epsilon_i \sim Bernoulli(1-\delta)$

# Graph Random Neural Networks (GRAND)

## ❖ Random Propagation for Graph Data Augmentation

- Random propagation은 2 step으로 진행




**Step 2) Feature propagation** $\bar{X} = \hat{A} \times \hat{A}^k \tilde{X}$

- Mixed-order propagation을 통해 feature 업데이트
- Over-smoothing 위험을 낮추고 local한 feature를 더욱 잘 학습함
- Propagation을 통해 임의로 drop한 정보를 이웃 node로부터 보충 받아서 기존 node의 feature와 유사한 feature가 형성
  - ➢ 인접한 노드는 유사한 feature와 label을 갖을 것이라는 homophily assumption에 근거함

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 8 | 4 | 5 |
| 3 | 5 | 2 |
| 1 | 3 | 3 |
| 0 | 0 | 0 |

Perturbed feature $\tilde{x}$

**Propagation Rule**

$\bar{X} = \hat{A} \times \hat{A}^k \tilde{X}$

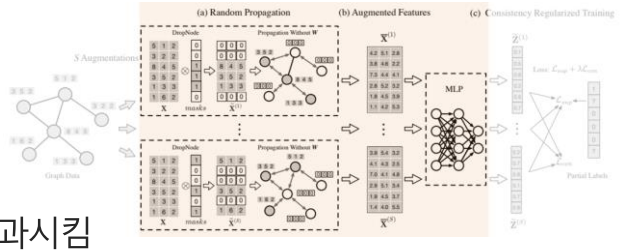
$*\hat{A}$ *from order* 0 *to* $K$

| | | |
|---|---|---|
| 4.2 | 5.1 | 2.8 |
| 3.8 | 4.6 | 2.2 |
| 7.3 | 4.4 | 4.1 |
| 2.8 | 5.2 | 3.2 |
| 1.8 | 4.5 | 3.9 |
| 1.1 | 4.2 | 5.3 |

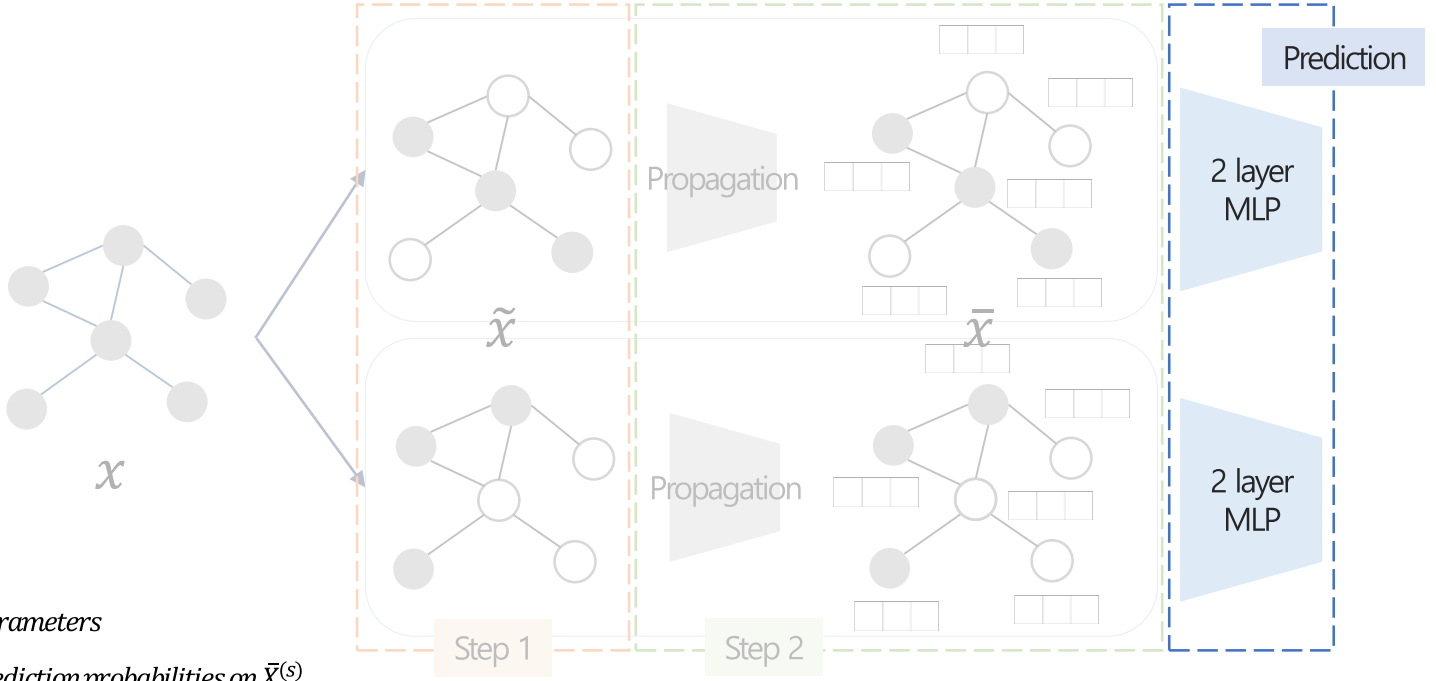
Propagated feature $\bar{X}$

# Graph Random Neural Networks (GRAND)

❖ Random Propagation for Graph Data Augmentation



- Prediction $\tilde{\boldsymbol{Z}}^{(s)} = \boldsymbol{f}_{\boldsymbol{mlp}}(\bar{\boldsymbol{X}}^{(s)}, \boldsymbol{\theta})$
  - Propagation을 $S$ 회 수행하여 얻은 각 feature에 대해 2 layer MLP를 통과시킴
  - Propagation 부분과 prediction 부분을 별개로 구성함으로써 high-order feature propagation이 가능하고 over-smoothing 위험이 감소




*$\theta$ are the model parameters*

*$\tilde{Z}^{(s)}$ denotes the prediction probabilities on $\bar{X}^{(s)}$*

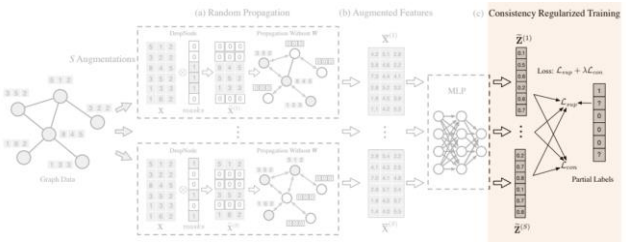




DMQA

# Graph Random Neural Networks (GRAND)

## ❖ Consistency Regularized Training




- Supervised loss와 consistency regularization loss의 합으로 구성

$$L = L_{sup} + L_{con}$$

➢ Supervised loss

- 전체 n 개의 노드 중 $m$개의 labeled node를 사용하여 node classification을 수행
- S개의 augmentations에 대한 cross-entropy loss의 평균으로 정의됨

$$L_{sup} = -\frac{1}{s}\sum_{s=1}^{s}\sum_{i=0}^{m-1} Y_i^T \log \widetilde{Z}_i^{(s)}$$

➢ Consistency regularization loss

- Unlabeled 노드만을 사용한 S개의 augmentations의 예측 결과가 consistency하게 하는 것이 목적
- 모든 distribution의 평균을 구해서 label distribution center를 계산한 후, sharpening trick을 사용

$$L_{con} = \frac{1}{s}\sum_{s=1}^{s}\sum_{i=0}^{n-1} \left\| \widehat{Z}_i' - \widetilde{Z}_i^{(s)} \right\|_2^2 \qquad \widehat{Z}_i' = \bar{Z}_{ij}^{\frac{1}{T}} \Big/ \sum_{c=0}^{C-1} \bar{Z}_{ic}^{\frac{1}{T}}, \quad (0 \le j \le C-1)$$

# Experiments

## ❖ Evaluation

- Standard GNN setting(on semi-supervised learning)을 동일하게 사용
  - ➢ Datasets: 3 benchmark graph(Cora, Citeseer, Pubmed) 사용
  - ➢ Baselines: perturbation 방식으로 DropNode 사용, 14개의 GNN baselines 사용
    - Eight graph convolutions: GCN [24], GAT [41], APPNP [25], Graph U-Net [13], SGC [45], MixHop [1], GMNN [36] and GrpahNAS [14].
    - Two sampling based GNNs: GraphSAGE [19] and FastGCN [8].
    - Four regularization based GNNs: VBAT [10], $G^3$NN [29], GraphMix [42] and Dropedge [37]. We report the results of these methods with GCN as the backbone model.
    - Four GRAND variants: GRAND_dropout, GRAND_DropEdge, GRAND_GCN and GRAND_GAT. In GRAND_dropout and GRAND_DropEdge, we use dropout and DropEdge as the perturbation method respectively, instead of DropNode. In GRAND_GCN and GRAND_GAT, we replace MLP with more complex models, i.e., GCN and GAT, respectively.

# Experiments

## ❖ Results

- GRAND는 baselines의 결과보다 모든 데이터셋에서 좋은 classification 성능을 냄
- DropNode는 DropEdge보다 좋은 classification 성능을 냄
- Prediction을 위해 MLP를 사용하는 것이 GCN과 GAT를 사용하는 것보다 좋은 classification 성능을 냄
- Figure 2를 통해 GRAND의 generalization 성능이 좋음을 확인 가능

| Method | Cora | Citeseer | Pubmed |
|---|---|---|---|
| GCN [24] | 81.5 | 70.3 | 79.0 |
| GAT [41] | 83.0±0.7 | 72.5±0.7 | 79.0±0.3 |
| APPNP [25] | 83.8±0.3 | 71.6± 0.5 | 79.7 ± 0.3 |
| Graph U-Net [13] | 84.4±0.6 | 73.2±0.5 | 79.6±0.2 |
| SGC [45] | 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 [36] | 83.7 | 72.9 | 81.8 |
| GraphNAS [14] | 84.2±1.0 | 73.1±0.9 | 79.6±0.4 |
| GraphSAGE [19] | 78.9±0.8 | 67.4±0.7 | 77.8±0.6 |
| FastGCN [8] | 81.4±0.5 | 68.8±0.9 | 77.6±0.5 |
| VBAT [10] | 83.6±0.5 | 74.0±0.6 | 79.9±0.4 |
| $G^3$NN [29] | 82.5±0.2 | 74.4±0.3 | 77.9 ±0.4 |
| GraphMix [42] | 83.9±0.6 | 74.5±0.6 | 81.0±0.6 |
| DropEdge [37] | 82.8 | 72.3 | 79.6 |
| GRAND_dropout | 84.9±0.4 | 75.0±0.3 | 81.7±1.0 |
| GRAND_DropEdge | 84.5±0.3 | 74.4±0.4 | 80.9±0.9 |
| 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** |
| 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 |

Table 1: Overall classification accuracy (%).

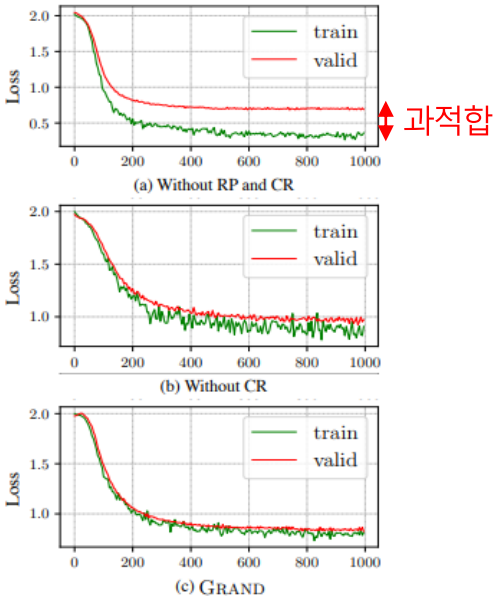



Figure 2: Generalization on Cora ($x$: epoch).



DMQA

# Experiments

## ❖ Results

- Robustness analysis를 위해 2가지 adversarial attack methods(random attack, metattack)를 사용
  - ➢ Figure 3을 보면 GRAND가 2가지 attack methods에 대해 모두 가장 좋은 classification 성능을 냄
  - ➢ GRAND가 GCN, GAT보다 consistency regularization을 더욱 잘 함
- Over-smoothing analysis를 위해 node representations의 over-smoothness를 측정하는 MADGap을 사용
  - ➢ MADGap 값이 작을수록 over-smoothing 문제가 심각한 것을 의미함
  - ➢ Figure 4를 보면 propagation step이 증가할수록 GCN, GAT의 성능은 급격히 하락
  - ➢ GRAND가 타 모델에 비해 over-smoothing 문제가 덜 발생함을 알 수 있음

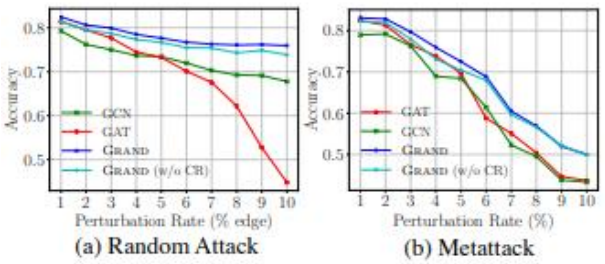



Figure 3: Robustness Analysis on Cora.

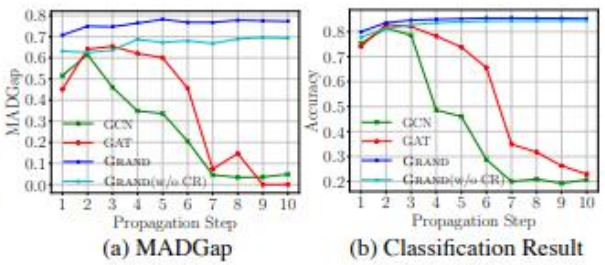



Figure 4: Over-Smoothing on Cora



DMQA

# Conclusion

## ❖ conclusion

- 본 연구는 Graph Random Neural Networks(GRAND) 제안
  - ➢ 그래프 augmentation 방법으로 random propagation을 제안
  - ➢ Unlabeled data에 대한 모델의 generalization을 개선하고자 consistency regularization 사용
  - ➢ 실험을 통해 타 방법론보다 over-smoothing문제를 잘 해결하며 robustness한 것을 보임
- 단순하고 효과적인 아이디어로 graph node classification문제를 semi-supervised learning으로 해결

# Reference

1. Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., ... & Tang, J. (2020). Graph random neural networks for semi-supervised learning on graphs. Advances in neural information processing systems, 33, 22092-22103.


Thank You