Efficient Topology-aware Data Augmentation for High-Degree Graph Neural Networks

Yurui Lai, Xiaoyang Lin, Renchi Yang, Hongtao Wang

Hong Kong Baptist University

KDD 2024

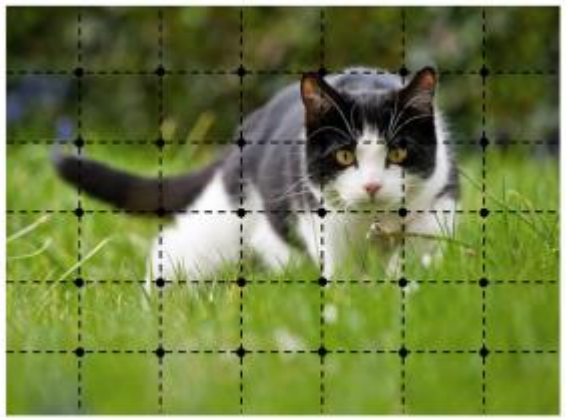
발표자 : 박기연

Index

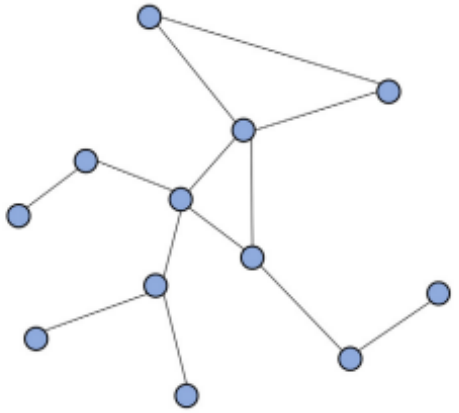
Introduction

Graph Neural Networks

- Non-Euclidean Space의 표현 및 학습이 가능
  - 실생활에서 들어오는 비 유클리드 데이터를 처리하기 적합
  - 관계, 상호작용과 같은 추상적 개념을 다루기 적합



Euclidean

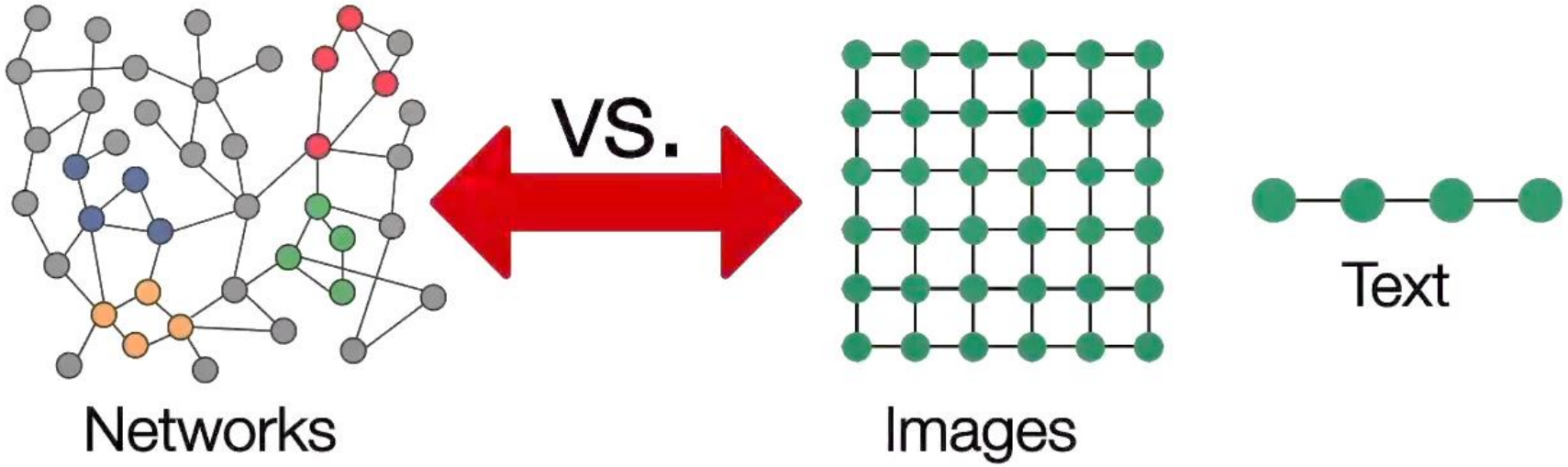


Non-Euclidean

Introduction

Graph Neural Networks

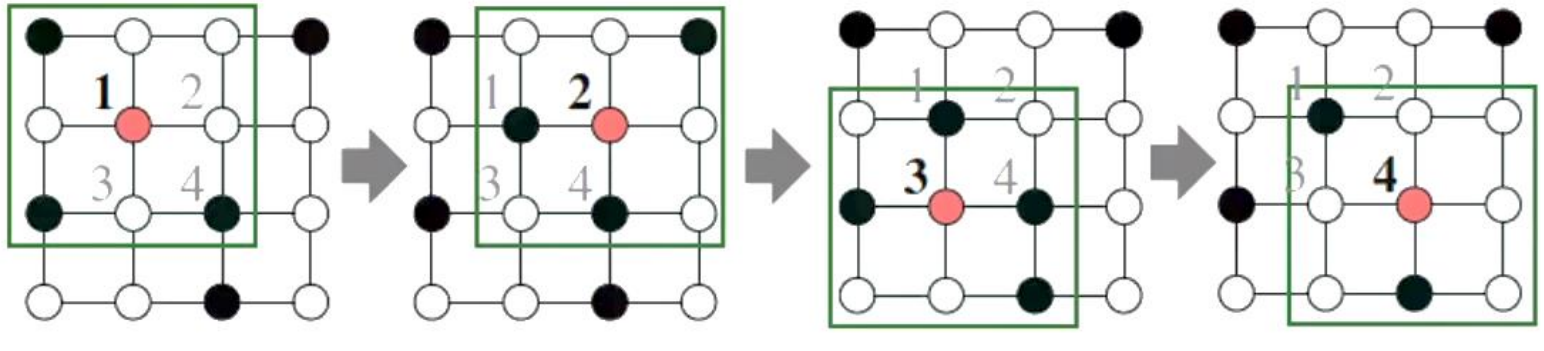
- Non-Euclidean Space의 표현 및 학습이 가능
  - 그렇다면 Non-Euclidean 데이터를 어떻게 입력으로 변환할 것인가?
    - 인접행렬을 통한 표현은 Node Ordering에 Sensitive 하다는 문제
    - Feature Vector를 어떻게 합칠 것인지 또한 난관

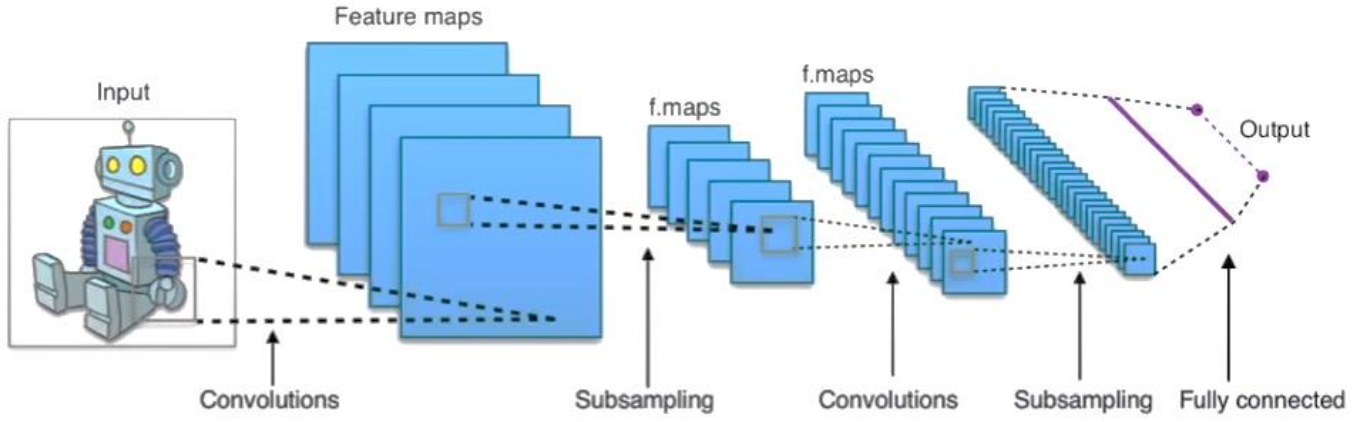


Introduction

Graph Convolutional Networks

- 그래프 구조를 CNN처럼 Locality를 토대로 묶어보자

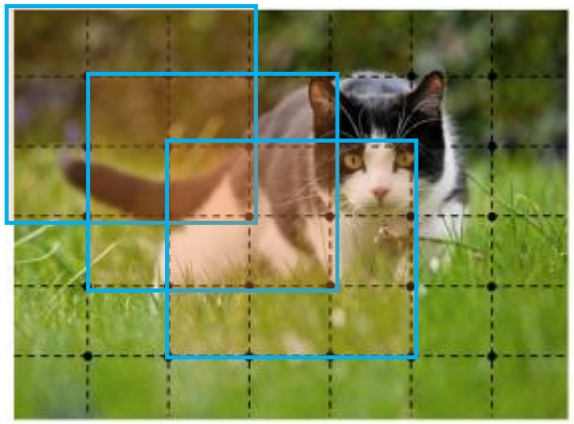




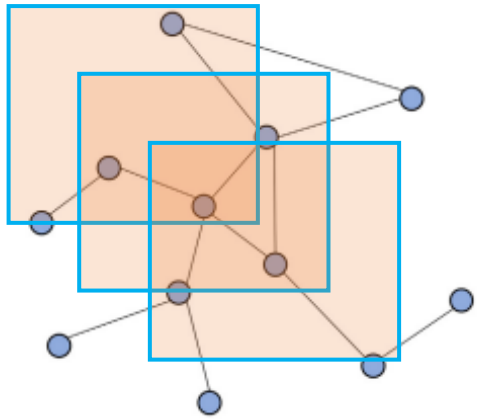
Introduction

Graph Convolutional Networks

- 그래프 구조를 CNN처럼 Locality를 토대로 묶어보자
  - 하지만 Graph는 Permutation Invariant
  - Locality나 Sliding Window에 대한 고정된 개념이 없음



Euclidean

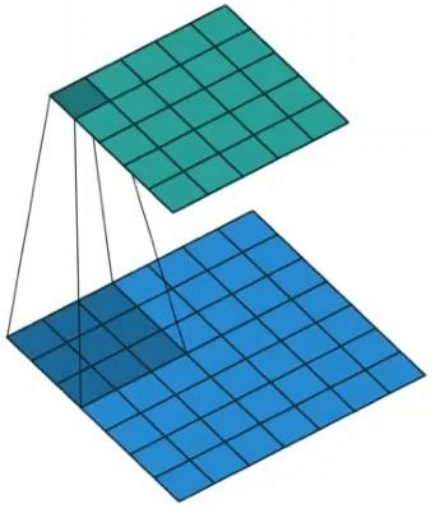


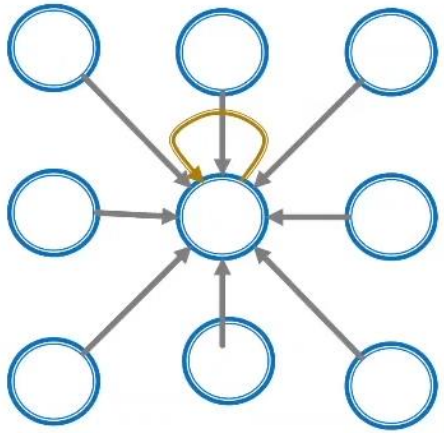
Non-Euclidean

Introduction

Graph Convolutional Networks

- 그래프 구조를 CNN처럼 Locality를 토대로 묶어보자
  - 하지만 Locality나 Sliding Window에 대한 고정된 개념이 없음
  - Graph는 Permutation Invariant

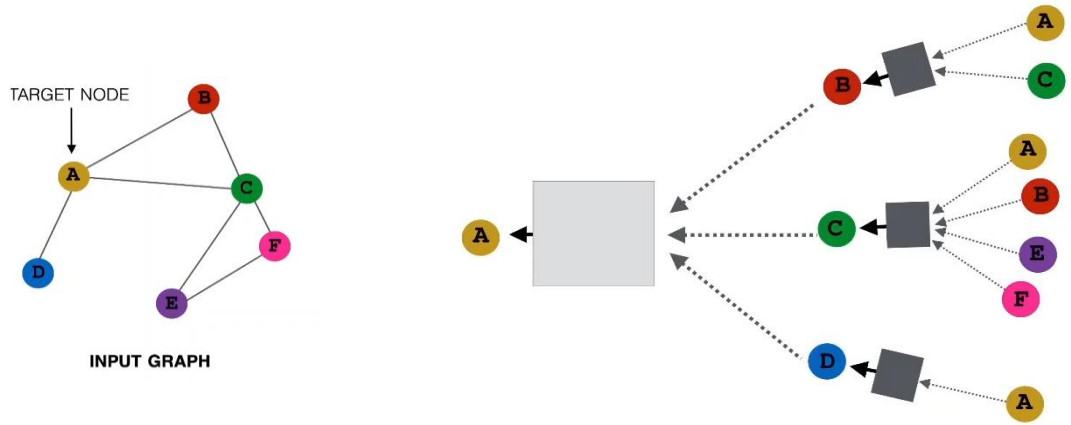


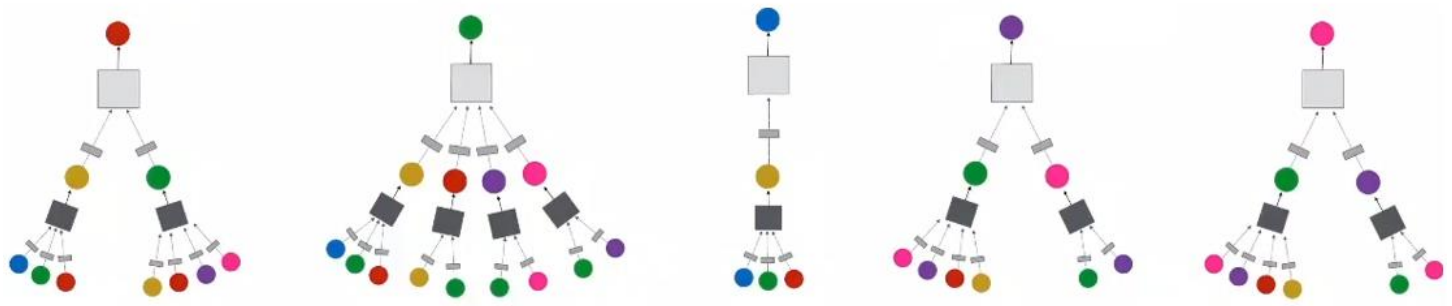


Introduction

Message Passing

- 타겟 노드를 기준으로 1-hop 이웃 노드들의 Feature들을 Aggregate
  - K-Layer → K-hop Receptive Field

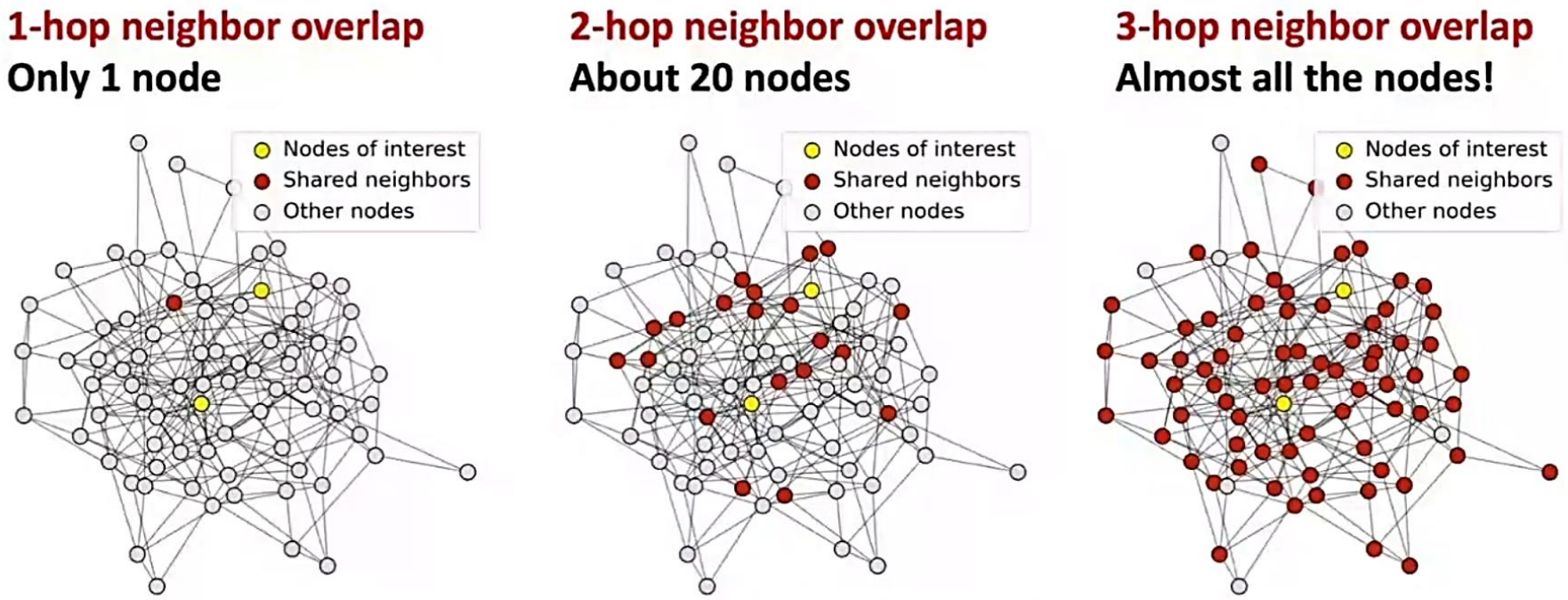




Introduction

Message Passing

- 그런데 High Degree Graphs로 가게 된다면?
  - 한 노드 당 이웃의 수가 수십, 수백 개



Related Work

Data Augmentations for GNNs

- Rule-Based Methods
  - Heuristic에 의존하여 그래프 데이터 수정/조작
    - DropEdge (DropEdge: Towards Deep Graph Convolutional Networks on Node Classification, 2019, Rong et al.)
      - Dropout과 유사한 방식으로 작동
      - 무작위로 에지, 노드, Feature, 서브그래프, 메시지 등을 제거
      - 그래프의 모든 요소를 동일하게 취급하여 정보 손실 발생 → Sub-optimal
    - 가상 노드 추가 (Do transformers really perform badly for graph representation?, 2021, Ying et al.)
      - 모든 노드에 연결된 가상 노드 추가
    - Node Feature Augmentation (Affinity-Aware Graph Networks, 2023, Velingker et al.)
      - Node Embedding → Node Feature Expanding

Related Work

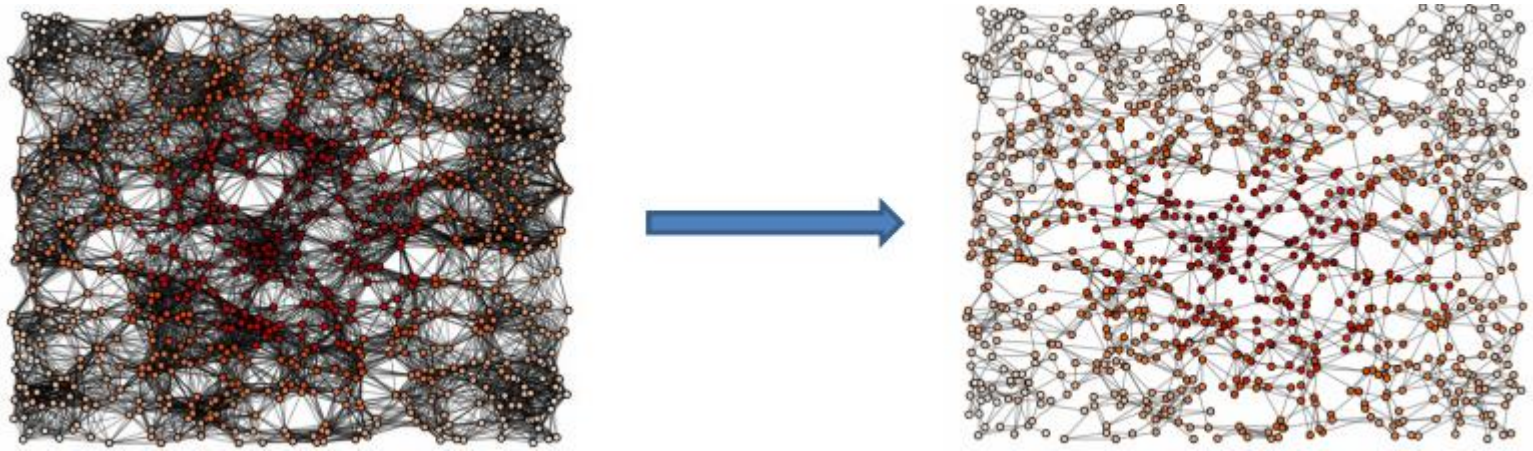
Data Augmentations for GNNs

- Learning-Based Methods
  - Graph Structure Learning
    - 그래프 구조를 학습 가능한 파라미터로 취급 → 나은 형태의 그래프 구조를 학습
  - Graph Structure Augmentation
  - Rationalization Methods
  - Reinforcement Learning
    - 그래프에서 서브 그래프를 학습 or 서브그래프 혹은 그래프에 대한 최적의 증강 전략을 학습

Related Work

Graph Sparsification

- 주어진 그래프를 회소 그래프로 근사화



Spectral Sparsification

- 라플라시안 행렬 기반
- For all n-dimensional vectors x,
- $G = (V, E) \rightarrow G_\epsilon = (V, E_\epsilon)$ with $\tilde{O}(\frac{n}{\epsilon^2})$

$$(1 - \epsilon)x^T L(G)x \leq x^T L(G_\epsilon)x \leq (1 + \epsilon)x^T L(G)x$$

$L = D - A$

Cut Sparsification

- Edge 가중치 기반
- For all cuts (S, V-S),
- $G = (V, E) \rightarrow G_\epsilon = (V, E_\epsilon)$ with $O(\frac{n \log n}{\epsilon^2})$

$$(1 - \epsilon)E(S) \leq E_\epsilon(S) \leq (1 + \epsilon)E(S)$$

Related Work

Structure Embedding

- 각 노드를 둘러싼 그래프 구조를 저차원 특징 벡터로 변환
  - Random walk-based methods
    - Skip-gram 모델을 최적화하는 방식으로 노드 임베딩 학습
  - Matrix factorization-based methods
    - Node-to-node affinity matrices를 분해하여 노드 임베딩 생성
  - Deep learning-based models
    - DNN을 통해 Non-attribute 그래프에서 노드 표현을 학습

  - Resistive Embedding or Spectral Embedding
    - as complementary node features

Method

Problem Definition

- Overcome Over-Smoothing Problem
- Overcome Expense of Calculating & Training on HDGs
- Capture Graph Topology and Node Attributes

Method

TADA (Topology Aware Data Augmentation)

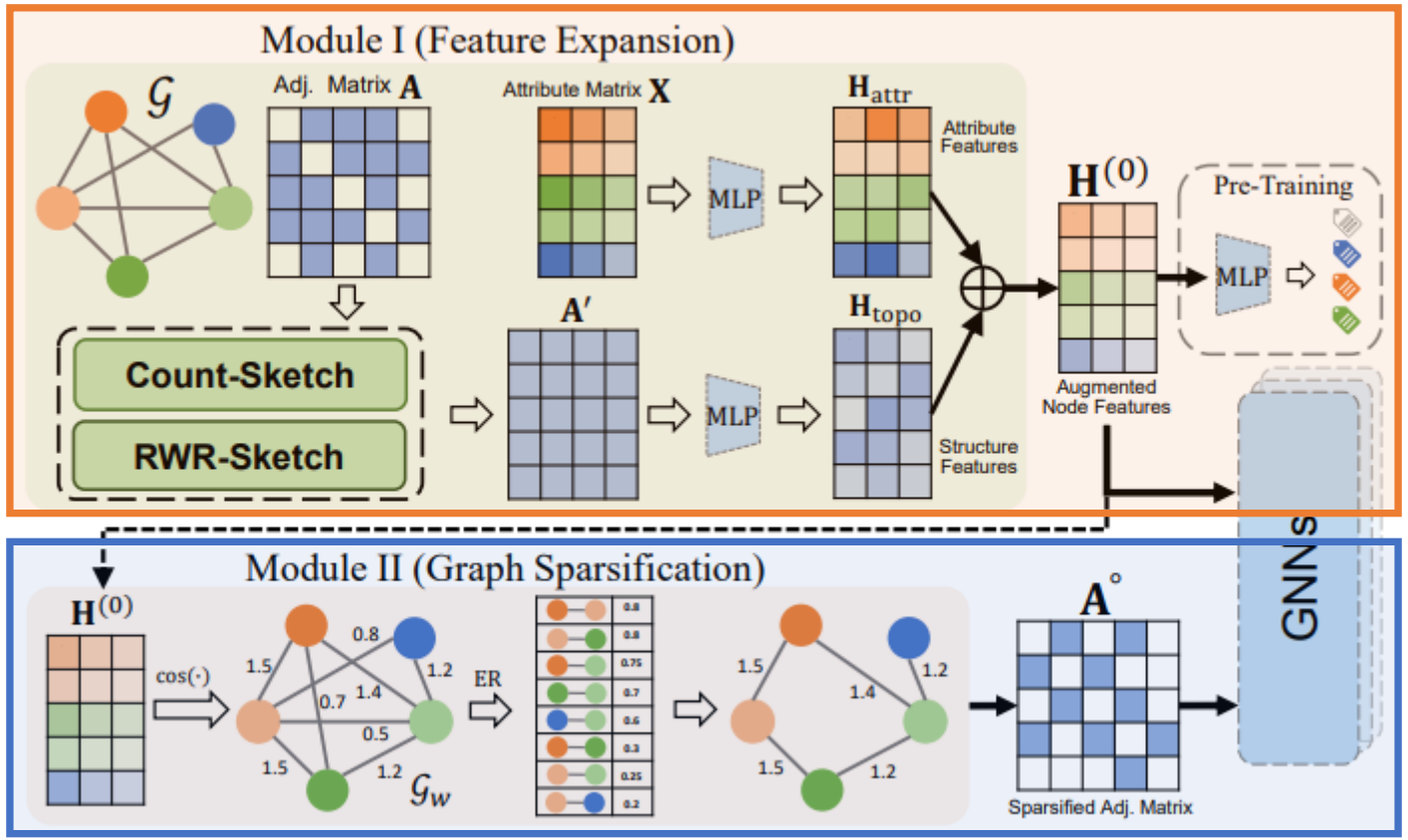
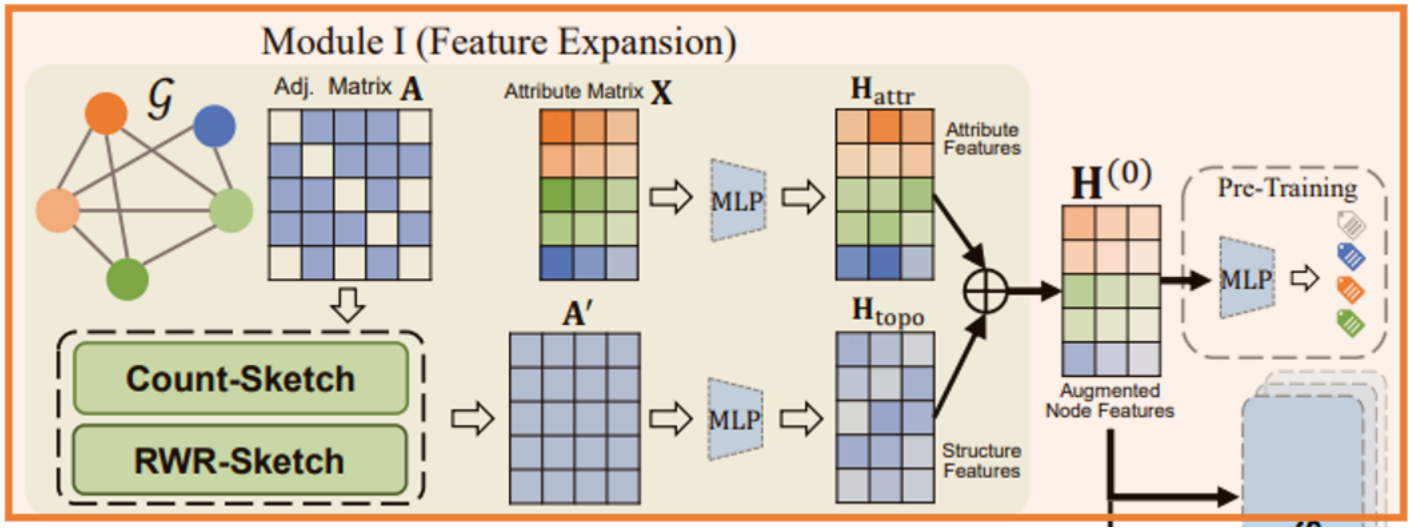


Figure 1: Overview of TADA

Method

TADA (Topology Aware Data Augmentation)



- 그래프의 구조적 의미를 더욱 많이 내포하는 Node Features를 Augmentation

Method

Feature Expansion

- Hybrid sketching technique을 인접행렬 A 에 적용
  - Count-Sketch + RWR-Sketch(Random Walk with Restart)
    - $A \rightarrow Count + RWR \rightarrow A' \in \mathbb{R}^{n \times k}$ $(k \ll n, typically\ k = 128)$
    - $A' \rightarrow MLP \rightarrow \Omega_{topo} \in \mathbb{R}^{k \times h} \rightarrow Structure\ embeddings\ H_{topo} \in \mathbb{R}^{n \times h}$ of all nodes
    - $H_{topo} = \sigma(A'\Omega_{topo})$

- 속성 행렬 $X \rightarrow MLP \rightarrow \Omega_{attr} \in \mathbb{R}^{d \times h}$
  - $H_{attr} = \sigma(X\Omega_{attr})$
  - $H_{attr} \in \mathbb{R}^{n \times h}$

- $H^{(0)} = (1-\gamma) \cdot H_{attr} + \gamma \cdot \mathrm{H_{topo}} \in \mathbb{R}^{n \times h}$
- $H^{(0)}$: Augmented Node Features

$$\mathbf{H}^{(t)} = \sigma(f_{\text{trans}}(f_{\text{aggr}}(\mathcal{G}, \mathbf{H}^{(t-1)}))),$$
$$\mathbf{H}^{(0)} = \sigma(\mathbf{X}\boldsymbol{\Omega}_{\text{orig}}) \in \mathbb{R}^{n \times h}$$

Ω_{topo} : 학습 가능한 변환 가중치
Ω_{attr} : 학습 가능한 가중치

Pretrained by single-layer MLP
By task (i.e., node classification)

Method

Feature Expansion

- HDGs 에서도 A는 희소 행렬임 ($m \ll n^2$)
  - 노드의 차수 분포가 매우 왜곡된 상태
  - 기존 밀집 행렬을 위한 sketch methods는 부적합
- Count-Sketch Method

  스케치된 인접 행렬 $A' \in \mathbb{R}^{n \times k}$를 O(nnz(A)) = O(m) 시간 안에 계산
  - $A' = AR^{\top}$, $R = \Phi\Delta$
    - $\Delta = \mathbb{R}^{n \times n}$: 대각 성분이 1 또는 -1로 0.5의 확률에 따라 선택된 대각 행렬
    - $\Phi \in \{0,1\}^{k \times n}$: 각 열에서 랜덤한 하나의 값이 1이고 나머지는 0인 이진 행렬
    - $\mathbf{\Phi} \in \{0,1\}^{k \times n}$ is a binary matrix with $\mathbf{\Phi}_{h(i),i} = 1$ and 0 otherwise $\forall 1 \leq i \leq n$. The function $h(\cdot)$ maps i $(1 \leq i \leq n)$ to $h(i) = j \in \{1, 2, \cdots, k\}$ uniformly at random.

Method

Feature Expansion

- Limitation of Count-Sketch Method
  - 근사가 보장됨 & 높은 효율성 But!
  - 데이터 비의존적 (스케칭 행렬의 무작위성)
    - Φ가 무작위로 생성 → A'에서 왜곡된 분포 발생 가능
      - 먼 노드가 동일한 클러스터
      - 가까운 노드가 다른 클러스터

Method

Feature Expansion

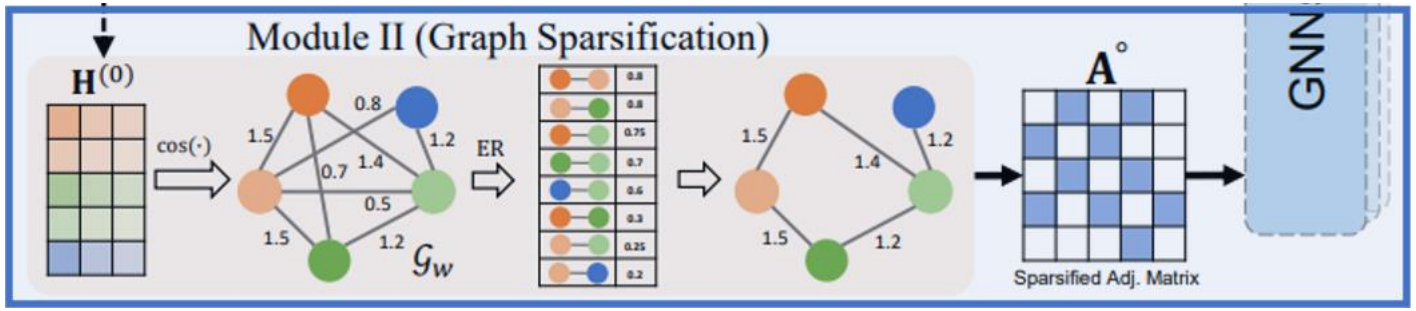
- Optimization via RWR-Sketch
  - RWR-Sketch → $S \in \mathbb{R}^{k \times n}$
  - $A' = A \cdot (R^{\top} + \beta \cdot S^{\top})$
  - S는 n개의 노드를 k개의 분리된 클러스터로 묶음
    - Random walk with Restart → 다중 홉 연결성 요약
      - 진입 차수가 가장 높은 노드 집합 C → 클러스터 후보 $(k \leq |C| \ll n)$
      - 각 노드v_i에 대해 $v_j \in C$ 에 대한 RWR 점수를 구함

$$\pi(v_i, v_j) = \sum_{t=0}^{T} (1-\alpha)\alpha^t \mathbf{P}_{i,j}$$

      - 각 v_j의 centraility 값 $\pi(v_j) = \sum_{v_i \in \mathcal{V}} \frac{\pi(v_i, v_j)}{n}$ 을 계산
        - 가장 큰 k개의 노드를 최종 Cluster Center로 선정
      - 각 $v_i \in V$ 에 대해, 가장 높은 RWR 점수를 가진 $v_j \in C_k$ 선택, $S_{j,i} = 1$로 설정
      - 각 행에 대해 L2 정규화

Method

TADA (Topology Aware Data Augmentation)



- Module I의 output에 기반하여 그래프 구조 희소화
- 그래프 구조에서 중복되거나 노이즈가 많은 connections 제거

→ 그래프 구조 및 노드 속성을 반영한 희소화

Method

Graph Sparsification

- Module I의 output $H^{(0)}$
  - 모든 에지의 centrality values를 계산 가능함

    Centrality values는 곧 그래프에서 edge의 중요성에 대한 지표
    - 이를 토대로 Edge Reweighting
      - $w(e_{i,j}) = \cos\left(\mathbf{H}_i^{(0)}, \mathbf{H}_j^{(0)}\right)$
    - 노드 v_i의 차수 또한 연결된 edge들의 가중치 합으로 계산
      - $d_w(v_i) = \sum_{v_j \in \mathcal{N}(v_i)} w(e_{i,j})$
    - 위의 과정을 거친 그래프 $G_w = (V, E_w)$
    - 그래프 G_w 에서의 Effective Resistance를 근사화
      - $\frac{1}{2}\left(\frac{1}{d_w(v_i)} + \frac{1}{d_w(v_j)}\right) \le r_w(e_{i,j}) \le \frac{1}{1-\lambda_2}\left(\frac{1}{d_w(v_i)} + \frac{1}{d_w(v_j)}\right)$
        - 각 edge $e_{i,j}$의 ER은 $\frac{1}{d_w(v_i)} + \frac{1}{d_w(v_j)}$ 에 비례한 값을 갖게 됨

Method

Graph Sparsification

- Edge Ranking and Sparsification of G_w
  - 희소화된 그래프 구성을 위해 Centrality Values에 따라 오름차순으로 에지를 정렬
  - Centrality Value $C_w(e_{i,j}) = w(e_{i,j}) \cdot \left(\frac{1}{d_w(v_i)} + \frac{1}{d_w(v_j)}\right)$
    - 직관적으로 에지 $e_{i,j}$가 노드 v_i와 v_j에 연결된 모든 에지들 사이에서 얼마나 중요한 지
  - 희소화 비율 ρ가 주어지면
    - $m \cdot \rho$개의 하위 Centrality Values를 가지는 Edge 부분집합 ε_{rm}을 삭제
    - $\forall e_{i,j} \in \varepsilon_w \setminus \varepsilon_{rm} \quad \mathrm{A}^{\circ}_{\mathrm{i,j}} = w(e_{i,j})$
      - A°: 희소화된 그래프의 인접행렬
      - $\varepsilon_w \setminus \varepsilon_{rm}$: 제거되지 않고 남은 에지들
      - 즉, 제거되지 않은 edge들의 가중치 → 희소화된 인접행렬 A°에 그대로 반영됨

Experiments

Datasets & Setup

Table 2: Statistics of Datasets (K = 10^3 and M = 10^6).

Dataset	n	m	d	$\lvert\mathcal{Y}\rvert$	m/n	HR
Photo [65]	7.7K	238.2K	745	8	31.1	0.83
WikiCS [54]	11.7K	431.7K	300	10	36.9	0.65
Reddit2 [107]	233K	23.2M	602	41	99.6	0.78
Amazon2M [12]	2.45M	61.9M	100	47	25.3	0.81
Squirrel [58]	5.2K	396.9K	2.1K	5	76.3	0.22
Penn94 [32]	41.6K	1.4M	128	2	32.8	0.47
Ogbn-Proteins [32]	132.5K	39.6M	8	112	298.5	0.38
Pokec [45]	1.6M	30.6M	65	2	18.8	0.45

- 8개의 Benchmark HDGs ($18 \leq m/n$인 고-차수 그래프)
- $|Y|$: 그래프 G 내의 노드들에 대한 클래스 라벨의 개수
- HR: Homophily Ratio, 동일한 클래스의 노드들간 연결된 에지 비율 (0.5 미만 = Heterophilic)

Linux Machine with NVIDIA Ampere A100 GPU (80GB), AMD EPYC 7513 CPU (2.6 GHz), 1TB RAM

Experiments

Results

Table 3: Node classification results (% test accuracy) of different GNN backbones with and without TADA on homophilic and heterophilic graphs. We conduct 10 trials and report mean accuracy and standard deviation over the trials.

Method	*Photo*	*WikiCS*	*Reddit2*	*Amazon2M*	*Squirrel*	*Penn94*	*Ogbn-Proteins*	*Pokec*
GCN	94.63±0.15	84.05±0.76	92.58±0.03	74.12±0.19	54.85±2.02	75.9±0.74	69.75±0.6	**75.47±1.36**
GCN + TADA	**94.92±0.45**	**84.62±0.53**	**94.86±0.22**	**76.14±0.23**	**73.48±1.61**	**76.06±0.43**	**73.79±0.76**	75.01±0.27
GAT	93.84±0.46	83.74±0.75	OOM	OOM	55.70±3.26	71.09±1.35	OOM	73.20±7.02
GAT + TADA	**94.58±0.12**	**84.97±0.84**	**95.97±0.04**	**59.16±0.36**	**72.99±2.81**	**71.19±0.78**	**74.94±0.25**	**74.26±0.94**
SGC	93.29±0.79	83.47±0.83	94.78±0.02	59.86±0.04	52.18±1.49	56.77±0.14	70.33±0.04	**67.40±5.56**
SGC + TADA	**94.93±0.39**	**83.97±0.71**	**95.65±0.02**	**73.39±0.35**	**72.32±2.72**	**71.02±0.53**	**74.31±0.42**	62.06±0.52
APPNP	94.95±0.33	85.04±0.60	90.86±0.19	65.51±0.36	54.47±2.06	69.25±0.38	75.19±0.58	62.79±0.11
APPNP + TADA	**95.42±0.53**	**85.19±0.56**	**95.34±0.18**	**69.81±0.24**	**73.24±1.38**	**71.08±0.62**	**75.52±0.32**	**67.03±0.27**
GCNII	95.12±0.12	85.13±0.56	94.66±0.07	OOM	53.13±4.29	74.97±0.35	73.11±1.93	76.49±0.88
GCNII + TADA	**95.54±0.44**	**85.42±0.60**	**96.62±0.08**	**77.83±0.62**	**72.89±2.45**	**75.84±3.13**	**75.34±1.33**	**77.64±0.32**

- Homophilic & Heterophilic 그래프 모두에서 정확도 향상
- Squirrel에서 가장 큰 성능 향상 폭을 보임
  - 정보가 거의 없는 노드 Attribute → 구조적 특징이 크게 기여
- Reddit2 & Ogbn-Proteins 데이터셋의 경우 m/n 이 특히 큰 HDG
  - Over-smoothing & edge noise 문제를 의도한 대로 해결
- Pokec의 GCN+TADA & SGC+TADA의 경우, 평균 정확도는 낮아졌지만 성능 안정성 향상

Experiments

Results

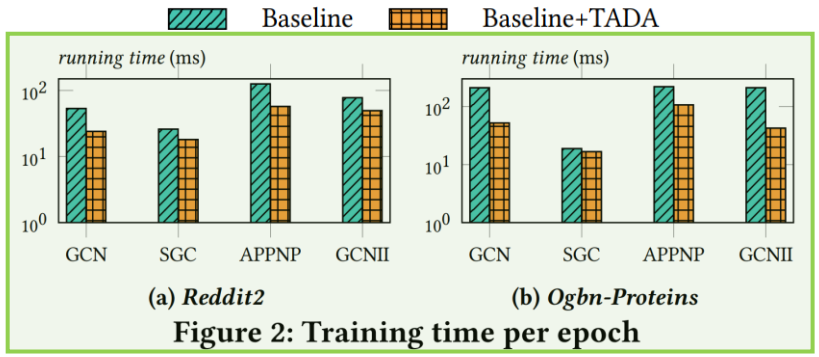


Figure 2: Training time per epoch

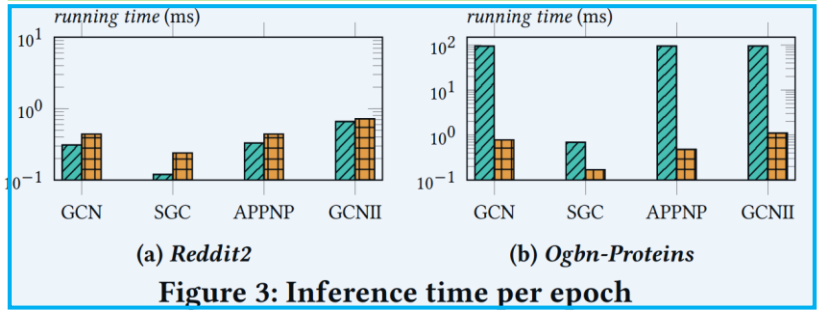


Figure 3: Inference time per epoch

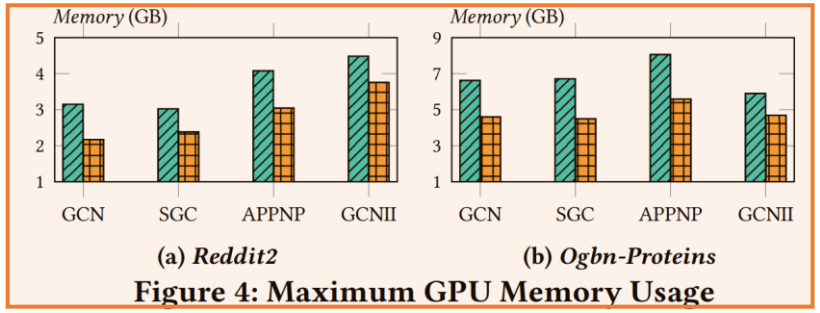


Figure 4: Maximum GPU Memory Usage

- Feature Aggregation Overhead
  - Ogbn-Proteins (Heterophilic)
  - Reddit2 (Homophilic)

- 추론 속도
  - Ogbn-Proteins
    - GCN, APPNP, GCNII 추론 속도 약 121, 198, 86배 향상
  - Reddit에서는 유사하거나 더 부진

- 에포크 당 학습 시간
  - 전체적 향상

- 메모리 사용량
  - 24% ~ 16% 까지 감소

- 전체적으로 HDG에서 GNN이 겪는 문제를 성공적으로 해결

Experiments

Results

Table 4: Comparison with GDA Baselines.

Method	*Reddit2*		*Ogbn-Proteins*	
	Acc (%)	Trng. / Inf. (ms)	Acc (%)	Trng. / Inf. (ms)
GCN	$92.58_{\pm 0.03}$	53.4 / 0.31	$69.75_{\pm 0.6}$	210.59 / 94.01
GCN+DropEdge	$93.59_{\pm 0.05}$	49.51 / 0.31	$61.46_{\pm 3.33}$	62.65 / 93.53
GCN+GraphMix	$92.60_{\pm 0.07}$	128.58 / 0.38	$72.41_{\pm 1.34}$	441.23/93.95
GCN+TADA	**$94.86_{\pm 0.22}$**	**24 / 0.44**	**$73.79_{\pm 0.76}$**	**52.26 / 0.78**
GCNII	$94.66_{\pm 0.07}$	125.6 / 0.66	$73.11_{\pm 1.93}$	211.31 / 95.52
GCNII+DropEdge	$96.23_{\pm 0.05}$	72.39 / 0.66	$60.50_{\pm 5.42}$	67.45 / 95.39
GCNII+GraphMix	$96.19_{\pm 0.05}$	172.66/0.72	$63.75_{\pm 1.72}$	456.59 / 95.34
GCNII+TADA	**$96.62_{\pm 0.08}$**	**49.5 / 0.72**	**$75.34_{\pm 1.33}$**	**42.68 / 1.11**

*Best is bolded and runner-up underlined.

- Comparison with Graph Data Augmentation Baselines
  - Reddit2와 Ogbn-Proteins에서 GCN, GCNII Backbone, +DropEdge, +GraphMix vs. +TADA 비교 (정확도, 학습, 추론)
  - +TADA의 경우 두 데이터셋 모두에서 정확도, 훈련 및 추론 속도에서 크게 향상
  - GDA Baseline들을 접목한 경우에는 Backbone보다 성능 및 추론 속도가 떨어지는 경우도 존재
    - HDG에서 기존 GDA 방법들의 한계에 대한 분석과 일치

Experiments

Results

Table 5: Ablation Study

Method	*Reddit2*	*Obgn-Proteins*
GCN	$92.58_{\pm 0.03}$	$69.75_{\pm 0.60}$
+ Count-Sketch	$93.81_{\pm 0.88}$	$72.90_{\pm 2.14}$
+ RWR-Sketch	$94.25_{\pm 0.66}$	$70.33_{\pm 2.54}$
+ Module II (i.e., GCN+TADA)	**$94.86_{\pm 0.22}$**	**$73.79_{\pm 0.76}$**
Random Projection (Module I)	$93.99_{\pm 0.74}$	$72.26_{\pm 0.77}$
k-SVD (Module I)	$93.36_{\pm 0.51}$	$69.79_{\pm 1.37}$
DeepWalk (Module I)	$94.48_{\pm 0.34}$	$72.56_{\pm 0.94}$
node2vec (Module I)	$94.47_{\pm 0.41}$	$\underline{73.05}_{\pm 1.50}$
LINE (Module I)	$\underline{94.49}_{\pm 0.34}$	$72.49_{\pm 1.66}$
RS (Module II)	$91.04_{\pm 0.04}$	$72.54_{\pm 1.11}$
k-Neighbor Spar (Module II)	$93.97_{\pm 0.7}$	$72.90_{\pm 1.47}$
SCAN (Module II)	$89.93_{\pm 0.78}$	$71.85_{\pm 1.25}$
DSpar (Module II)	$93.58_{\pm 0.08}$	$72.75_{\pm 1.11}$

*Best is bolded and runner-up underlined.

- Ablation Study
  - Reddit2 & Obgn-Proteins
    - GCN Backbone
    - +TADA
      - Module I 의 각 단계
      - Module I + Module II
    - Module I의 Sketching 기법 변환 (Module II 고정)
    - Module II를 대체하여 실험
  - Module I + Module II가 다른 대안들에 비해 월등한 성능

Conclusion

Limits

- Heterophilic에서의 향상이 크긴 하지만, Homophilic에 비해서 여전히 낮은 정확도

Contribution

- HDGs에서 기존 방식이 가지는 문제점을 효과적으로 개선
  - OOM → 59.16~77.83% Accuracy
- Heterophilic에도 적용이 가능한 Method이며 성능 향상을 보임

감사합니다