Deep Graph Infomax

Chao Chen

LEHIGH UNIVERSITY

Background: Graph and GNN

Given a graph G with N nodes:

Each node has a node feature vector $x_i \in \mathbb{R}^F$, and we denote $X = \{x_1, x_2, \dots, x_N\}$

Adjacency matrix $A \in \{0,1\}^{N \times N}$

In the l-th layer of Graph Neural Network (GNN)[1], to update **node** representation: [**Encoder**]

$$a_i^{(l)} = AGGRE^{(l)}\left(\left\{h_j^{(l-1)} : j \in \mathcal{N}(i)\right\}\right), \qquad h_i^{(l)} = COMBINE^{(l)}\left(h_i^{(l-1)}, a_i^{(l)}\right)$$

For example:

In GCN[2], two operations are integrated:

$$h_i^{(l)} = ReLU\left(W \cdot MEAN\left\{h_j^{(l-1)}, \forall j \in \mathcal{N}(i) \cup \{i\}\right\}\right)$$

In GraphSAGE[3]:

$$a_i^{(l)} = MAX\left(\left\{ReLU\left(W_1 \cdot h_j^{(l-1)}\right), \forall j \in \mathcal{N}(i)\right\}\right), \qquad h_i^{(l)} = W_2 \cdot \left[h_i^{(l-1)}, a_i^{(l)}\right]$$

$MAX(\cdot)$ here is element-wise max pooling.

Background: Graph and GNN

Given a graph G with N nodes:

Each node has a node feature vector $x_i \in \mathbb{R}^F$, and we denote $X = \{x_1, x_2, \ldots, x_N\}$

Adjacency matrix $A \in \{0,1\}^{N \times N}$

After the L-th layer, we can use a **readout** function for **graph** representation (summary):

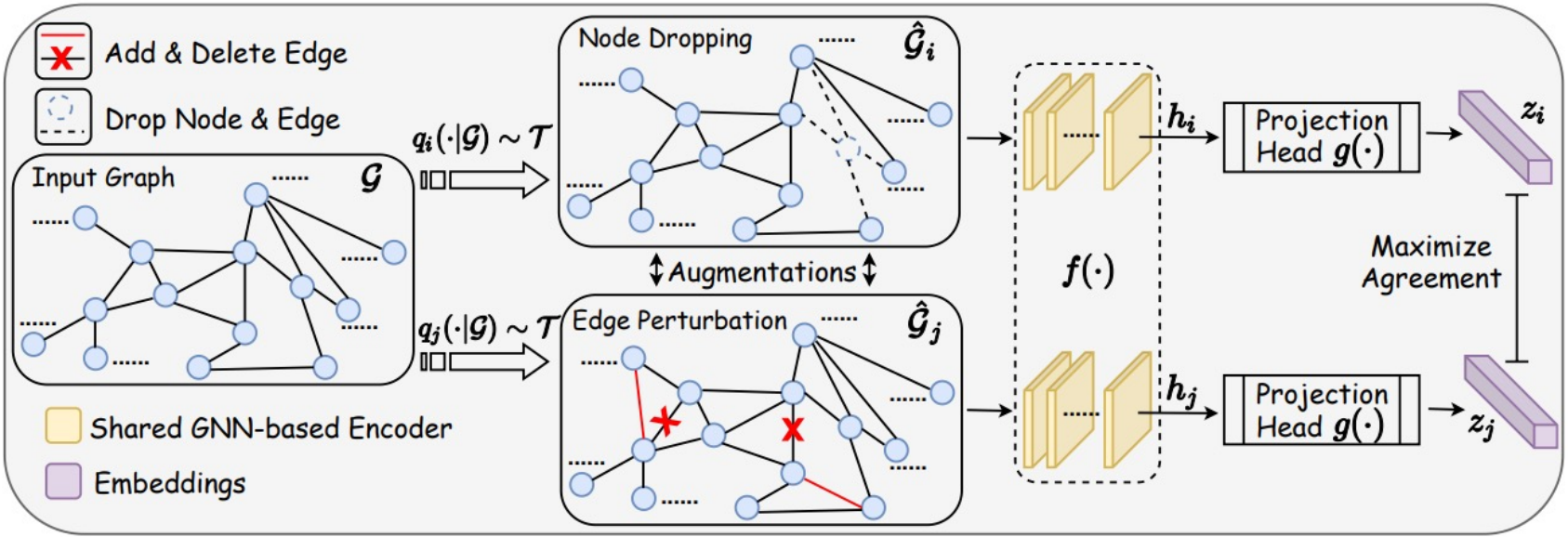
$$s = \mathcal{R}\left(\left\{h_i^{(L)} \middle| i \in G\right\}\right)$$

For example:

$$s = \frac{1}{N}\sum_{i \in G} h_i^{(L)}$$

Background: Contrastive Learning

Basic assumption: some types of transformation retain the important information[4].



For n-th graph, two transformed samples i and j:

$$l_n = -\log \frac{\exp\left(sim\left(z_{n,i}, z_{n,j}\right)/\tau\right)}{\sum_{n'=1,n'\neq n}^{N} \exp\left(sim\left(z_{n,i}, z_{n',j}\right)/\tau\right)}$$

Background: Notations

Given a graph with N nodes:

Node feature $x_i \in \mathbb{R}^F$, and feature matrix $X = \{x_1, x_2, \dots, x_N\} \in R^{N \times F}$, adjacency matrix $A \in \{0,1\}^{N \times N}$

[Goal] An encoder $\mathcal{E}: \mathbb{R}^{N \times F} \times \mathbb{R}^{N \times N} \to \mathbb{R}^{N \times F'}$ that encodes **node** with **high-level** representation:

$H = \mathcal{E}(X, A) = \{h_1, h_2, \dots, h_N\}$, where $h_i \in \mathbb{R}^{F'}$.

(Example: GCN or GraphSAGE can be considered as an encoder.)

A readout function $\mathcal{R}: \mathbb{R}^{N \times F} \to \mathbb{R}^F$ summarizes the nodes representations to a **graph-level** representation: $s = \mathcal{R}\big(\mathcal{E}(X, A)\big)$

(Example: MEAN or SUM can be considered as a readout function.)

Deep Graph Infomax

Original node feature matrix $X = \{x_1, x_2, \dots, x_N\}$, adjacency matrix A

An encoder $\mathcal{E}$ and new node representations: $H = \mathcal{E}(X, A)$

A readout function $\mathcal{R}$ results in a summary: $s = \mathcal{R}\big(\mathcal{E}(X, A)\big)$

A discriminator $\mathcal{D}: \mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}$ assigns probability to node: $\mathcal{D}(h_i, s)$

[To decide if the node should be contained within the summary]

[Example: cosine similarity. $\mathcal{D}(h_i, s) = \sigma\big(h_i^T W s\big)$ in this paper]

Negative samples for $\mathcal{D}$: $\tilde{h}_j$ from an alternative graph $\tilde{G} = \big(\tilde{X}, \tilde{A}\big)$

$\tilde{G} = \big(\tilde{X}, \tilde{A}\big)$ can come from another graph in the multiple-graphs dataset.

Or can be obtained by a corruption function $\mathcal{C}: \mathbb{R}^{N \times F} \times \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^{M \times F} \times \mathbb{R}^{M \times M}$: $\big(\tilde{X}, \tilde{A}\big) = \mathcal{C}(X, A)$

[Example: $\tilde{A} = A$ and random row-wise shuffling X to get $\tilde{X}$]

[$\mathcal{C}$ generates negative samples, while $\mathcal{T}$ in Contrastive Learning generates positive samples.]

Deep Graph Infomax

Original node feature matrix $X = \{x_1, x_2, \dots, x_N\}$, adjacency matrix A

An encoder $\mathcal{E}$ and new node representations: $H = \mathcal{E}(X, A)$

A readout function $\mathcal{R}$ results in a summary: $s = \mathcal{R}\big(\mathcal{E}(X, A)\big)$

A discriminator $\mathcal{D}$ assigns importance to node: $\mathcal{D}(h_i, s)$

Negative samples $(\tilde{h}_j, s)$ from an alternative graph $\big(\tilde{X}, \tilde{A}\big) = \mathcal{C}(X, A)$

Metric: Mutual Information (MI), evaluated by the JSD MI estimator[5]:

$$\max MI(X, A; h_i) \approx \max \Big[\log\big(\mathcal{D}(h_i; X, A)\big) + \log\Big(1 - \mathcal{D}\big(\tilde{h}_j; X, A\big)\Big)\Big]$$

Approximate (X, A) by $s = \mathcal{R}\big(\mathcal{E}(X, A)\big)$

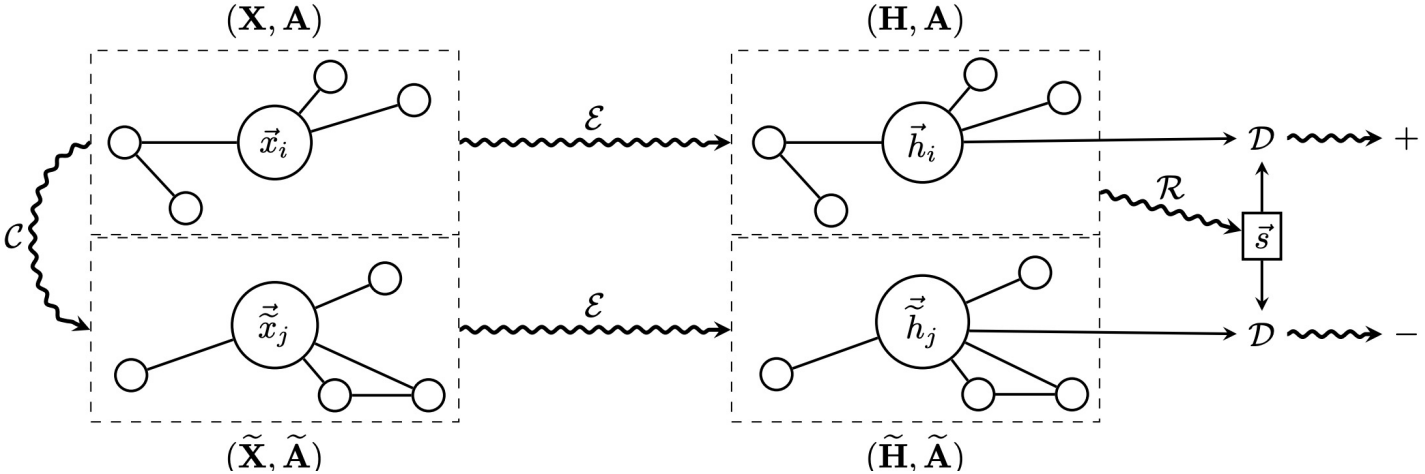
Objective: Noise Contrastive objective with BCE loss

$$\mathcal{L} = \frac{1}{N+M}\left(\sum_{i=1}^{N} \mathbb{E}_{(X,A)}\big[\log \mathcal{D}(h_i, s)\big] + \sum_{j=1}^{M} \mathbb{E}_{(\tilde{X},\tilde{A})}\Big[\log\Big(1 - \mathcal{D}\big(\tilde{h}_j, s\big)\Big)\Big]\right)$$

Deep Graph Infomax

1. Sample a negative example by $(\tilde{X}, \tilde{A}) \sim \mathcal{C}(X, A)$
2. Obtain $h_i, \tilde{h}_j$ by the encoder, $H = \mathcal{E}(X, A)$ and $\tilde{H} = \mathcal{E}(\tilde{X}, \tilde{A})$
3. Readout the input graph: $s = \mathcal{R}(H)$
4. Update parameters of $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{D}$ by gradient descent to maximize $\mathcal{L}$.

$$\mathcal{L} = \frac{1}{N+M}\left(\sum_{i=1}^{N} \mathbb{E}_{(X,A)}[\log \mathcal{D}(h_i, s)] + \sum_{j=1}^{M} \mathbb{E}_{(\tilde{X},\tilde{A})}\left[\log\left(1 - \mathcal{D}(\tilde{h}_j, s)\right)\right]\right)$$



Deep Graph Infomax – Theoretical Motivation

Q1: We approximate (X, A) by $s = \mathcal{R}\big(\mathcal{E}(X, A)\big)$, in which case they are the closest to each other?

A1: $\mathcal{R}$ is injective.

Given a probability distribution of graphs, $p(\boldsymbol{X})$, we can draw $\left\{\boldsymbol{X}^{(k)}\right\}_{k=1}^{|\boldsymbol{X}|}$ from $p(\boldsymbol{X})$ uniformly, e.g., $p\big(\boldsymbol{X}^{(k)}\big) = p\big(\boldsymbol{X}^{(k)'}\big)$. $s^{(k)} = \mathcal{R}\big(\boldsymbol{X}^{(k)}\big)$ is the summary of k-th graph with marginal distribution $p(s)$.

The optimal classifier between the $p(\boldsymbol{X}, s)$ and $p(\boldsymbol{X})p(s)$ has an error rate upper bounded by $Err^* = \frac{1}{2}\sum_{k=1}^{|\boldsymbol{X}|} p\big(s^{(k)}\big)^2$, and the upper bound is achieved when $\mathcal{R}$ is injective.

Deep Graph Infomax – Theoretical Motivation

Given a probability distribution of graphs, $p(\boldsymbol{X})$, we can draw $\{\boldsymbol{X}^{(k)}\}_{k=1}^{|\boldsymbol{X}|}$ from $p(\boldsymbol{X})$ uniformly, e.g., $p(\boldsymbol{X}^{(k)}) = p(\boldsymbol{X}^{(k)\prime})$. $s^{(k)} = \mathcal{R}(\boldsymbol{X}^{(k)})$ is the summary of k-th graph with marginal distribution $p(s)$.

Define $\mathcal{Q}^{(k)} = \{\boldsymbol{X}^{(j)} | \mathcal{R}(X^{(j)}) = s^{(k)}\}$ contains all graphs being mapped to $s^{(k)}$.

Sample $(\boldsymbol{X}^{(k)}, s^{(k)})$ drawn from the product of marginals with probability $p(\boldsymbol{X})p(s)$:

$$p(s^{(k)}) \sum_{s} p(\boldsymbol{X}^{(k)}, s) = p(s^{(k)}) p(\boldsymbol{X}^{(k)}, s^{(k)}) = p(s^{(k)}) p(\boldsymbol{X}^{(k)} | s^{(k)}) p(s^{(k)}) = p(s^{(k)})^2 \frac{p(\boldsymbol{X}^{(k)})}{\sum_{\boldsymbol{X}' \in \mathcal{Q}^{(k)}} p(\boldsymbol{X}')}$$

$p(\boldsymbol{X}^{(k)}, s^{(j)}) = 0$ when $j \neq k$ since $\mathcal{R}$ is deterministic.

$p(\boldsymbol{X}^{(k)}, s^{(k)}) = p(s^{(k)}) p(\boldsymbol{X}^{(k)} | s^{(k)})$ from the definition of conditional probability.

$p(\boldsymbol{X}^{(k)} | s^{(k)}) = \frac{p(\boldsymbol{X}^{(k)})}{\sum_{\boldsymbol{X}' \in \mathcal{Q}^{(k)}} p(\boldsymbol{X}')}$ from the definition of $\mathcal{Q}^{(k)}$.

Deep Graph Infomax – Theoretical Motivation

Given a probability distribution of graphs, $p(\boldsymbol{X})$, we can draw $\{\boldsymbol{X}^{(k)}\}_{k=1}^{|\boldsymbol{X}|}$ from $p(\boldsymbol{X})$ uniformly, e.g., $p(\boldsymbol{X}^{(k)}) = p(\boldsymbol{X}^{(k)'})$. $s^{(k)} = \mathcal{R}(\boldsymbol{X}^{(k)})$ is the summary of k-th graph with marginal distribution $p(s)$.

Define $\mathcal{Q}^{(k)} = \{\boldsymbol{X}^{(j)} | \mathcal{R}(X^{(j)}) = s^{(k)}\}$ contains all graphs being mapped to $s^{(k)}$.

Sample $(\boldsymbol{X}^{(k)}, s^{(k)})$ drawn from the product of marginals with probability $p(\boldsymbol{X})p(s)$:

$$\frac{p(\boldsymbol{X}^{(k)})}{\sum_{\boldsymbol{X}' \in \mathcal{Q}^{(k)}} p(\boldsymbol{X}')} p(s^{(k)})^2 = \rho^{(k)} p(s^{(k)})^2 \leq p(s^{(k)})^2$$

We have $\rho^{(k)} \leq 1$, since $\boldsymbol{X}^{(k)} \in \mathcal{Q}^{(k)}$.

When $\mathcal{Q}^{(k)} = \{\boldsymbol{X}^{(k)}\}$, ($\mathcal{R}$ is injective), $\rho^{(k)} = 1$.

Deep Graph Infomax – Theoretical Motivation

Given a probability distribution of graphs, $p(\boldsymbol{X})$, we can draw $\{\boldsymbol{X}^{(k)}\}_{k=1}^{|\boldsymbol{X}|}$ from $p(\boldsymbol{X})$ uniformly, e.g., $p(\boldsymbol{X}^{(k)}) = p(\boldsymbol{X}^{(k)'})$. $s^{(k)} = \mathcal{R}(\boldsymbol{X}^{(k)})$ is the summary of k-th graph with marginal distribution $p(s)$.

Define $\mathcal{Q}^{(k)} = \{\boldsymbol{X}^{(j)} | \mathcal{R}(X^{(j)}) = s^{(k)}\}$ contains all graphs being mapped to $s^{(k)}$.

Sample $(\boldsymbol{X}^{(k)}, s^{(k)})$ drawn from the joint $p(\boldsymbol{X}, s)$:

$$p(\boldsymbol{X}^{(k)}, s^{(k)}) = p(\boldsymbol{X}^{(k)}|s^{(k)})p(s^{(k)}) = \rho^{(k)}p(s^{(k)})$$

Compare it with previous results:

$$\rho^{(k)}p(s^{(k)}) \geq \rho^{(k)}p(s^{(k)})^2$$

The optimal classifier always classifies samples to the joint for a lower error

$$Err \leq \frac{1}{2}\sum_{k=1}^{|\boldsymbol{X}|}\rho^{(k)}p(s^{(k)})^2 \leq \frac{1}{2}\sum_{k=1}^{|\boldsymbol{X}|}p(s^{(k)})^2$$

When $\mathcal{R}$ is injective for all $\boldsymbol{X}^{(k)}$, $Err^* = \frac{1}{2}\sum_{k=1}^{|\boldsymbol{X}|}p(s^{(k)})^2$

Deep Graph Infomax – Theoretical Motivation

Q2: If we assume that $\mathcal{R}$ is injective and $|s^*| = |\boldsymbol{X}|$, what else can we know?

[$|s|$ is the number of allowable states in s, and s^* is the optimal summary w.r.t the classification error.]

A2: $s^* = argmax_s MI(\boldsymbol{X}; s)$. In other words, minimizing the classification error is equivalent to maximizing the mutual information.

Proof

MI is invariant under invertible transforms.

Since $\mathcal{R}$ is injective and $|s^*| = |\boldsymbol{X}|$, we always can find an inverse function $\mathcal{R}^{-1}$.

For any s:

$$MI(\boldsymbol{X}; s) \leq H(\boldsymbol{X}) = MI(\boldsymbol{X}; \boldsymbol{X}) = MI\big(\boldsymbol{X}; \mathcal{R}(\boldsymbol{X})\big) = MI(\boldsymbol{X}; s^*)$$

The definition of MI: $MI(\boldsymbol{X}; s) = H(\boldsymbol{X}) - H(\boldsymbol{X}|s)$

Experiments – Settings

Datasets:

- Transductive: Cora, Citeseer, Pubmed;
- Inductive: Reddit (large graph), PPI (multiple graphs).

In Cora, Citeseer, Pubmed (transductive):

Encoder $\mathcal{E}$ is one-layer GCN:

$$\mathcal{E}(X, A) = \sigma(\widehat{D}^{-\frac{1}{2}}\hat{A}\widehat{D}^{-\frac{1}{2}}X\Theta)$$

where $\hat{A} = A + I_N$ and σ is parametric ReLU.

Corruption $\mathcal{C}$ set $\tilde{A} = A$ and $\tilde{X}$ as randomly row-wise shuffling of X.

Experiments – Settings

Datasets:

- Transductive: Cora, Citeseer, Pubmed;
- Inductive: Reddit (large graph), PPI (multiple graphs).

In Reddit (large graph):

Encoder $\mathcal{E}$ is three-layer mean-pooling model:

$$MP(X, A) = \hat{D}^{-1}\hat{A}X\Theta,$$

$$\widetilde{MP}(X, A) = \sigma(X\Theta' \| MP(X, A)),$$

$$\mathcal{E}(X, A) = \widetilde{MP}_3\left(\widetilde{MP}_2\left(\widetilde{MP}_1(X, A), A\right), A\right)$$

where $\hat{A} = A + I_N$ and σ is parametric ReLU.

Corruption $\mathcal{C}$ set $\tilde{A} = A$ and $\tilde{X}$ as randomly row-wise shuffling of X.

Experiments – Settings

Datasets:

- Transductive: Cora, Citeseer, Pubmed;
- Inductive: Reddit (large graph), PPI (multiple graphs).

In Reddit (multiple graphs):

Encoder $\mathcal{E}$ is three-layer mean-pooling model with skip connections:

$$MP(X, A) = \widehat{D}^{-1}\hat{A}X\Theta,$$

$$H_1 = \sigma\big(MP_1(X, A)\big),$$

$$H_2 = \sigma\left(MP_2\big(H_1 + XW_{skip}, A\big)\right),$$

$$\mathcal{E}(X, A) = \sigma\left(MP_3\big(H_2 + H_1 + XW_{skip}, A\big)\right)$$

where W_{skip} is a learnable matrix, and σ is parametric ReLU.

Corruption $\mathcal{C}$ is to select other graphs.

Experiments – Settings

For all experiments:

Readout function $\mathcal{R}$:

$$\mathcal{R}(H) = \sigma\left(\frac{1}{N}\sum_{i=1}^{N} h_i\right)$$

σ is the sigmoid function.

Discriminator $\mathcal{D}$:

$$\mathcal{D}(h_i, s) = \sigma\left(h_i^T W s\right)$$

σ is the sigmoid function.

All models are initialized using Glorot initialization.

Experiments – Results

Accuracy in transductive

Transductive

Available data	Method	Cora	Citeseer	Pubmed
X	Raw features	47.9 ± 0.4%	49.3 ± 0.2%	69.1 ± 0.3%
A, **Y**	LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
A	DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
X, **A**	DeepWalk + features	70.7 ± 0.6%	51.4 ± 0.5%	74.3 ± 0.9%
X, **A**	Random-Init (ours)	69.3 ± 1.4%	61.9 ± 1.6%	69.6 ± 1.9%
X, **A**	**DGI** (ours)	**82.3** ± 0.6%	**71.8** ± 0.7%	**76.8** ± 0.6%
X, **A**, **Y**	GCN (Kipf & Welling, 2016a)	81.5%	70.3%	79.0%
X, **A**, **Y**	Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%

Micro-F1 in inductive

$$Precision_{mi} = \frac{\sum_i TP_i}{\sum_i [TP_i + FP_i]}$$

$$Recall_{mi} = \frac{\sum_i TP_i}{\sum_i [TP_i + FN_i]}$$

$$F1_{mi} = 2\frac{Precision_{mi} \times Recall_{mi}}{Precision_{mi} + Recall_{mi}}$$

Inductive

Available data	Method	Reddit	PPI
X	Raw features	0.585	0.422
A	DeepWalk (Perozzi et al., 2014)	0.324	—
X, **A**	DeepWalk + features	0.691	—
X, **A**	GraphSAGE-GCN (Hamilton et al., 2017a)	0.908	0.465
X, **A**	GraphSAGE-mean (Hamilton et al., 2017a)	0.897	0.486
X, **A**	GraphSAGE-LSTM (Hamilton et al., 2017a)	0.907	0.482
X, **A**	GraphSAGE-pool (Hamilton et al., 2017a)	0.892	0.502
X, **A**	Random-Init (ours)	0.933 ± 0.001	0.626 ± 0.002
X, **A**	**DGI** (ours)	**0.940** ± 0.001	**0.638** ± 0.002
X, **A**, **Y**	FastGCN (Chen et al., 2018)	0.937	—
X, **A**, **Y**	Avg. pooling (Zhang et al., 2018)	0.958 ± 0.001	0.969 ± 0.002

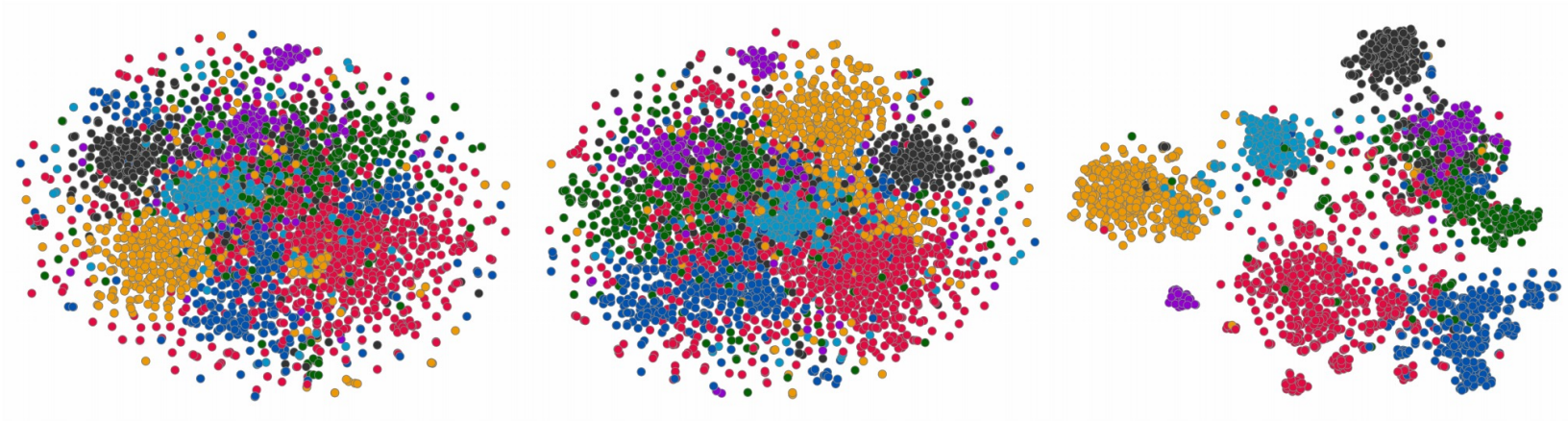
Experiments – Results

t-SNE embeddings of the nodes in the Cora dataset from:

the raw features (left),

features from a randomly initialized DGI model (middle),

and a learned DGI model (right).



Reference

[1] Xu, Keyulu, et al. "How powerful are graph neural networks?." ICLR, 2019.

[2] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." ICLR, 2017.

[3] Hamilton, William L., Rex Ying, and Jure Leskovec. "Inductive representation learning on large graphs." NeurIPS, 2017.

[4] You, Yuning, et al. "Graph contrastive learning with augmentations." NeurIPS, 2020.

[5] Hjelm, R. Devon, et al. "Learning deep representations by mutual information estimation and maximization." ICLR, 2019.

Thank you

LEHIGH UNIVERSITY