Recent Advances in Machine learning on Graphs

Chanyoung Park

Assistant Professor
Department of Industrial & Systems Engineering
KAIST
cy.park@kaist.ac.kr

This talk

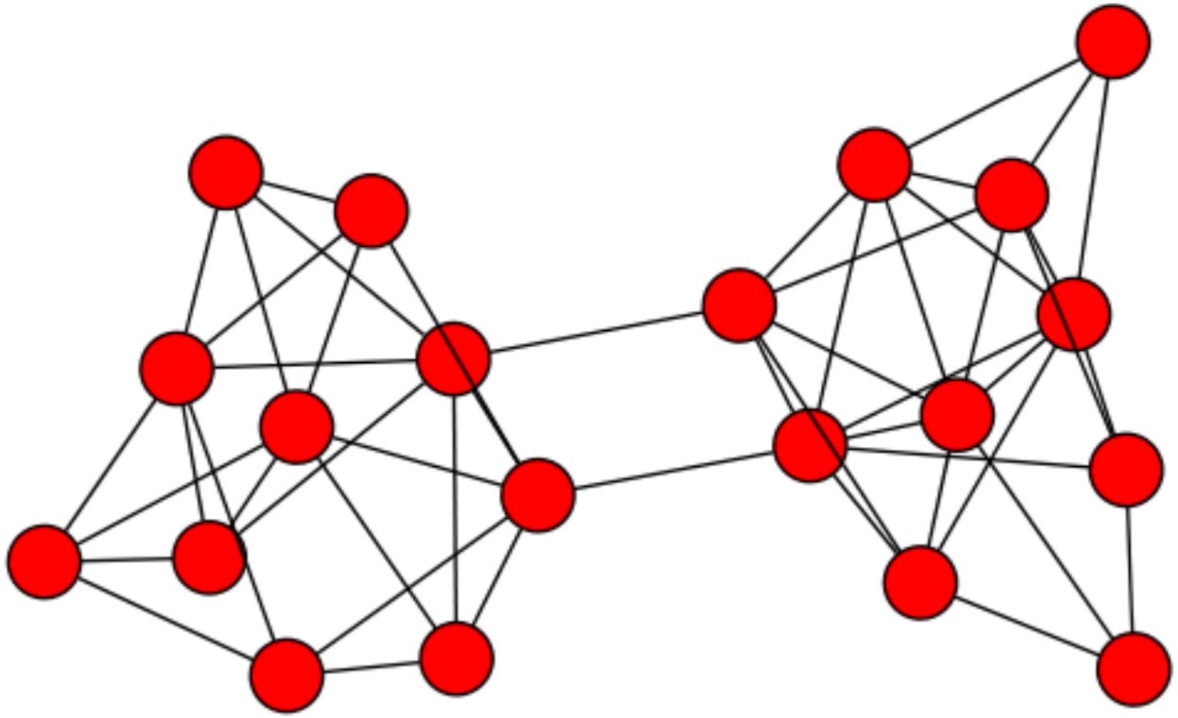
- How to learn graph representation in **various types of graphs**?
  - GNNs for Homogeneous Graph
  - GNNs for Multi-aspect Graph
  - GNNs for Multi-relational Graph
- How to effectively **train GNNs**?
  - Self-supervised learning
  - Alleviating Long-tail problem
  - Robustness of GNN

This talk

- How to learn graph representation in **various types of graphs**?
  - ~~GNNs for Homogeneous Graph~~
  - GNNs for Multi-aspect Graph
  - GNNs for Multi-relational Graph
- How to effectively **train GNNs**?
  - Self-supervised learning
  - Alleviating Long-tail problem
  - Robustness of GNN

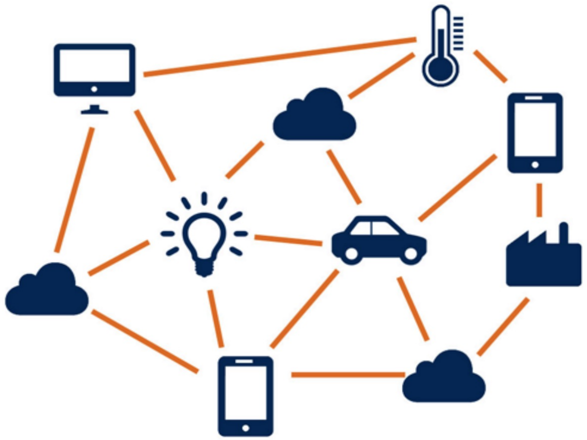
Graph (Network)

- A general description of data and their relations

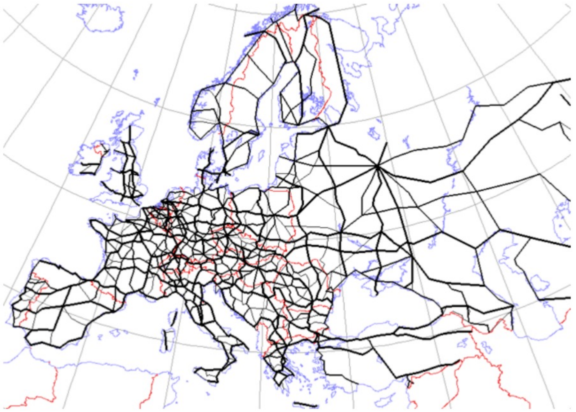


Various Real-World Graphs

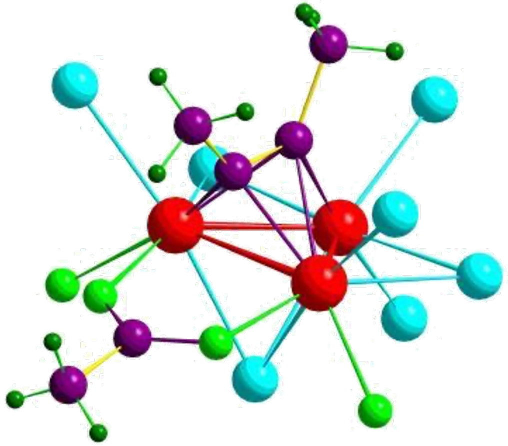
Social graph



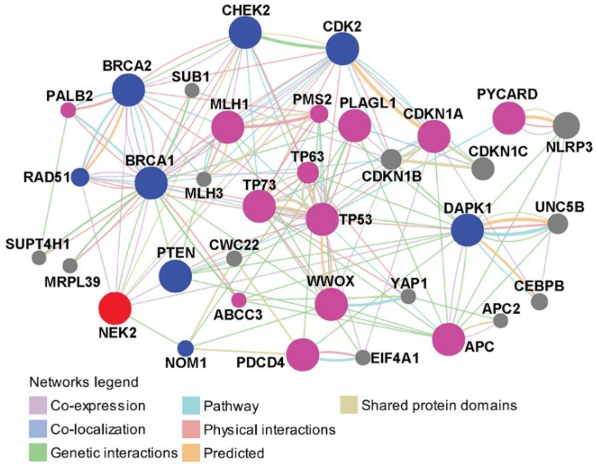
Internet-of-Things



Transportation



Molecular graph



Gene network

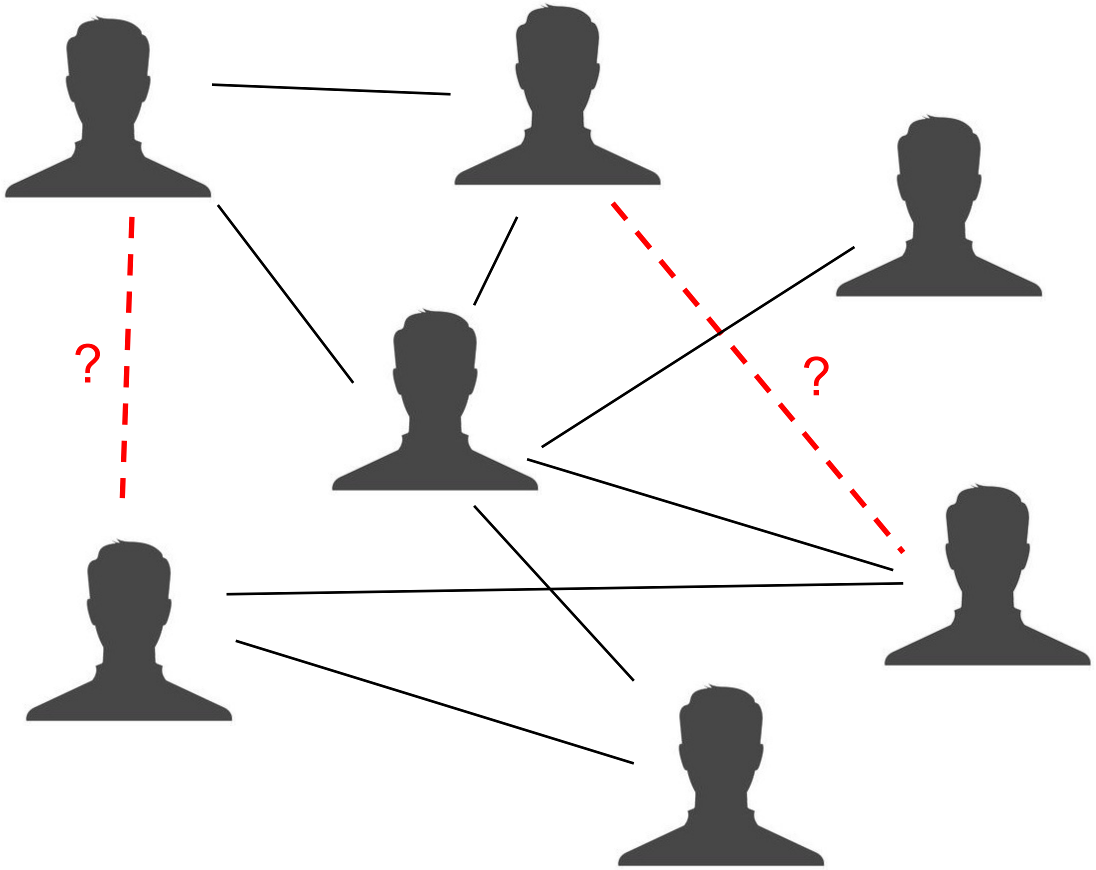
Web graph

(Figure credit) Web

Machine Learning on Graphs

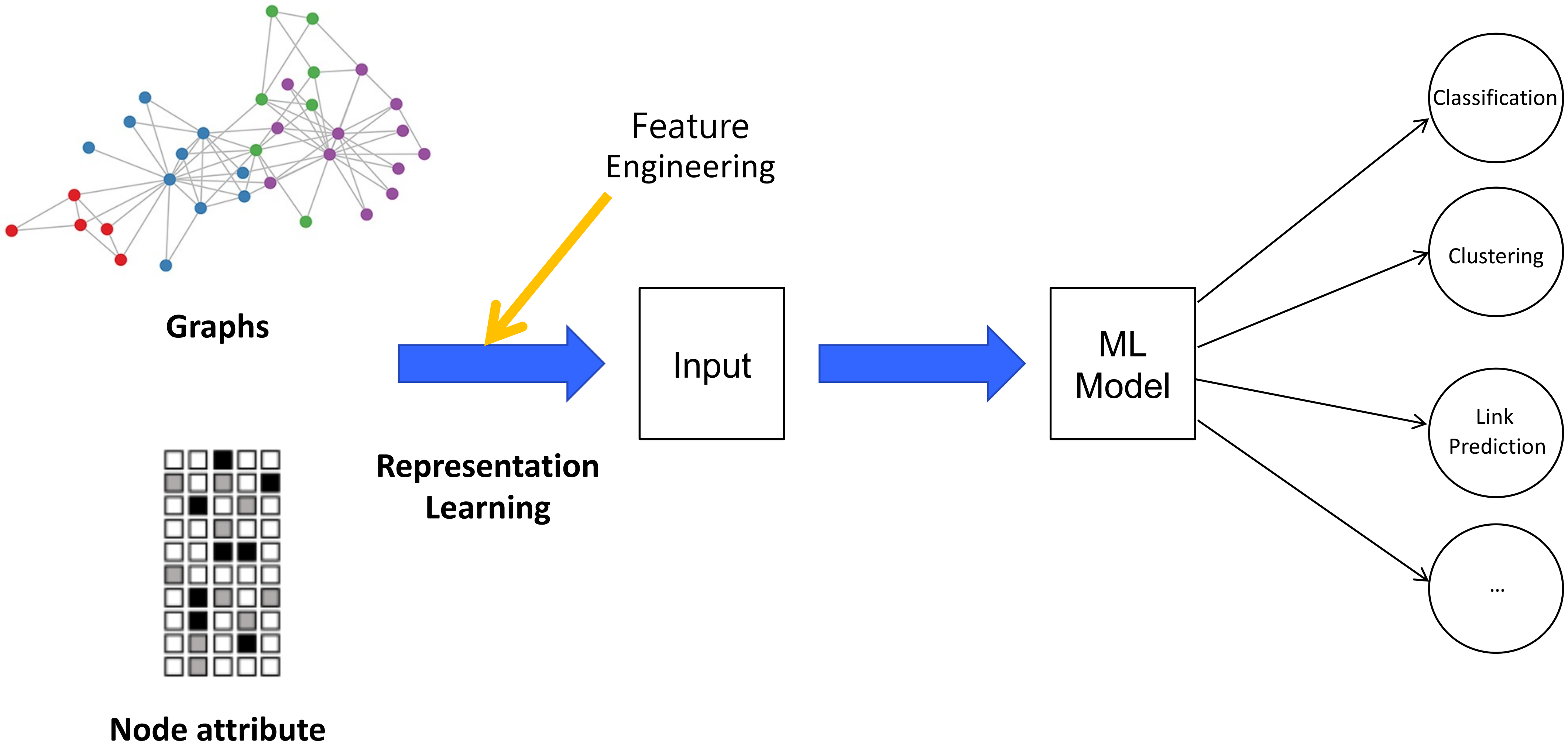
Classical ML tasks in graphs:

- Node classification
  - Predict a type of a given node
- Link prediction
  - Predict whether two nodes are linked
- Community detection
  - Identify densely linked clusters of nodes
- Network similarity
  - How similar are two (sub)networks

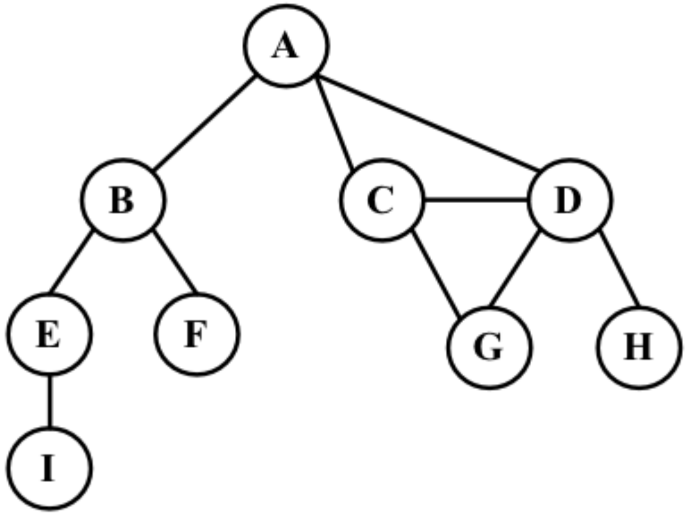


Link Prediction (Friend Recommendation)

Machine Learning on Graphs



Traditional Graph Representation



	A	B	C	D	E	F	G	H	I
A	0	1	1	1	0	0	0	0	0
B	1	0	0	0	1	1	0	0	0
C	1	0	0	1	0	0	1	0	0
D	1	0	1	0	0	0	1	1	0
E	0	1	0	0	0	0	0	0	1
F	0	1	0	0	0	0	0	0	0
G	0	0	1	1	0	0	0	0	0
H	0	0	0	1	0	0	0	0	0
I	0	0	0	0	1	0	0	0	0

Adjacency matrix

Problems

- Suffer from data sparsity
- Suffer from high dimensionality
- High complexity for computation
- Does not represent “semantics”
- …

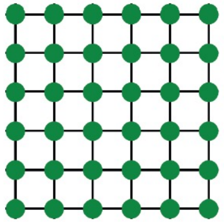
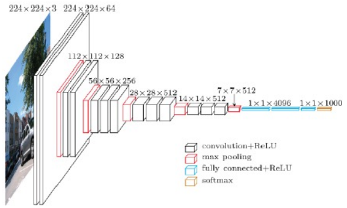
How to effectively and efficiently represent graphs is the key!

→ Deep learning-based approach?

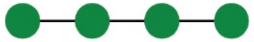
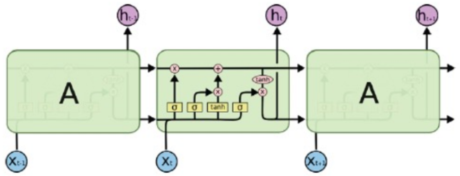
(Figure credit) https://www.oreilly.com/library/view/learning-javascript-data/9781788623872/8a7d3187-7c57-418c-a426-3aceab96f47f.xhtml

Challenges of Graph Representation Learning

- Existing deep neural networks are designed for data with regular-structure (grid or sequence)
  - CNNs for fixed-size images/grids …

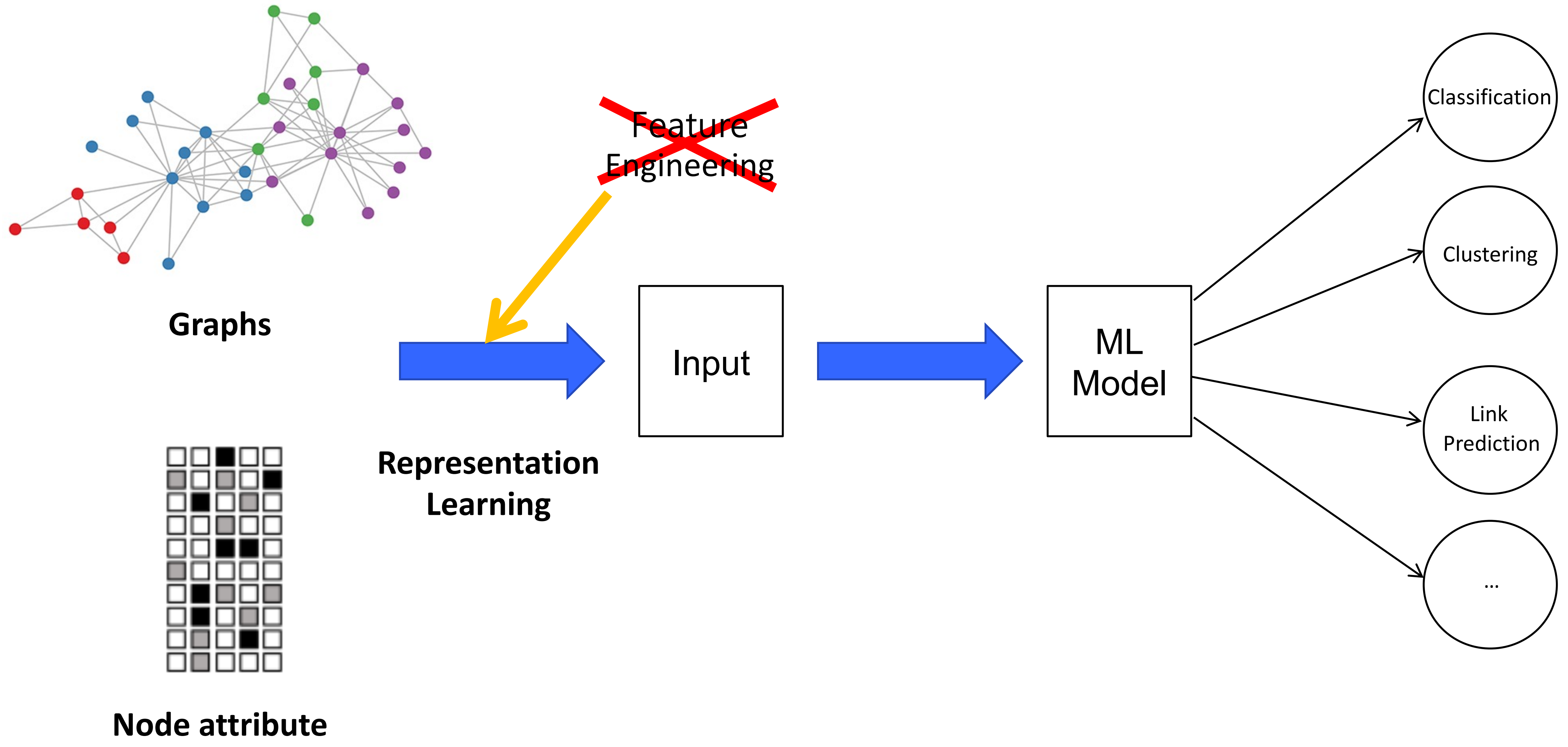
  - RNNs for text/sequences …

- **Graphs are very complex**
  - Arbitrary structures (no spatial locality like grids / no fixed orderings)
  - Heterogeneous: Directed/undirected, binary/weighted/typed, multimodal features
  - Large-scale: More than millions of nodes and billions of edges

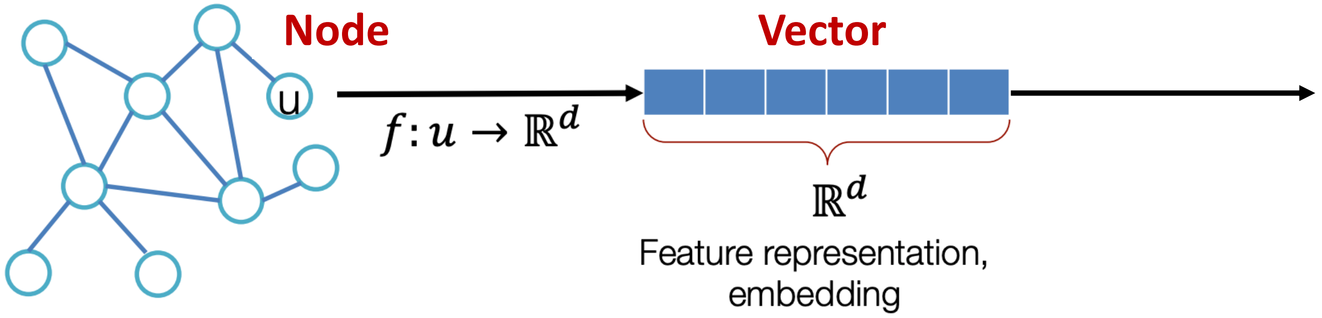
(Figure credit) Tutorial on Graph Representation Learning, AAAI 2019

Machine Learning on Graphs



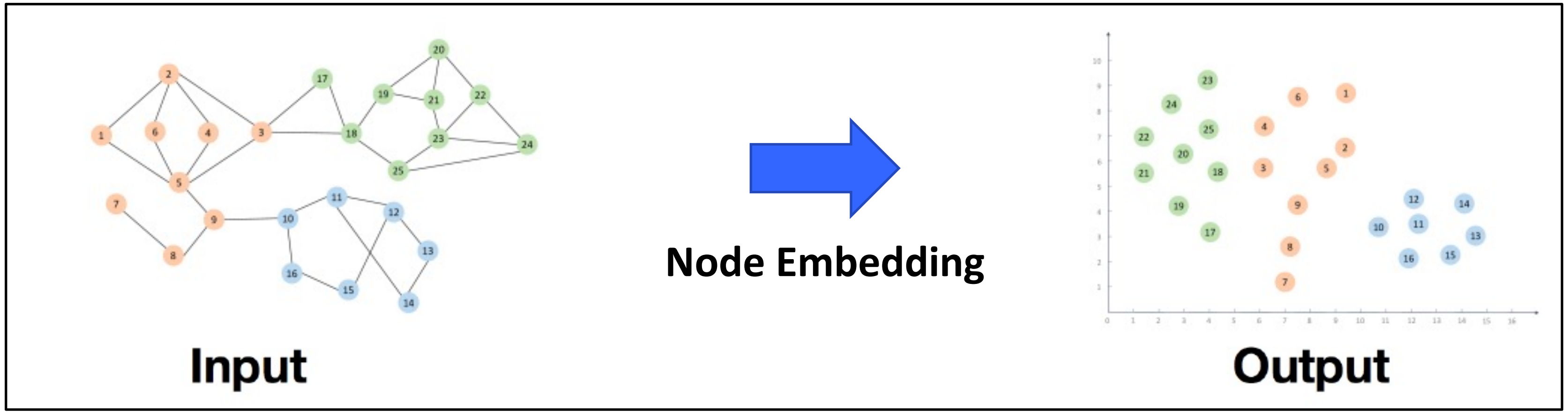
Graph Representation Learning

- **Goal**: Encode nodes so that **similarity in the embedding space** approximates **similarity in the original network**
- **Similar nodes in a network have similar vector representations**



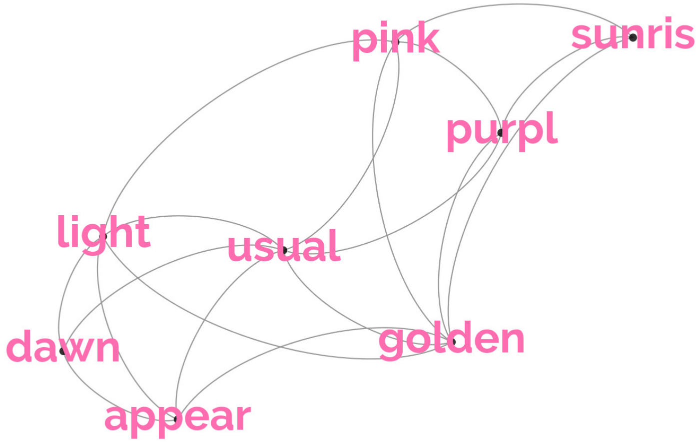
Tasks

- Node classification
- Clustering
- Link prediction
- …



Homogeneous Graph

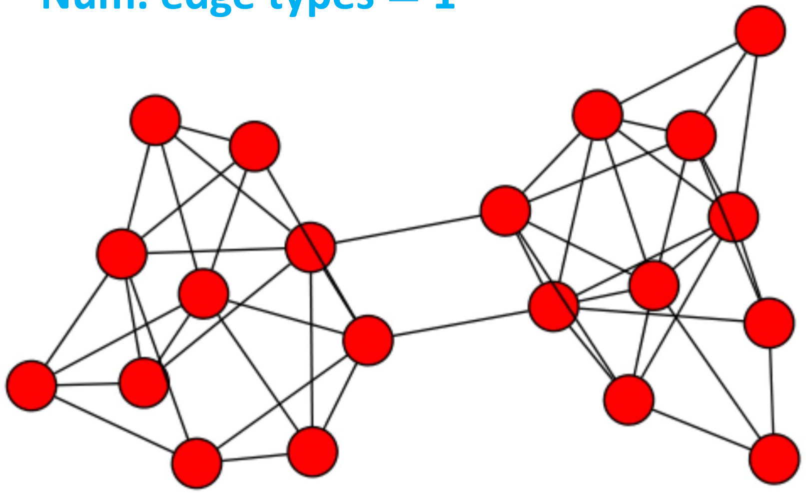
- A graph with a single type of node and a single type of edge



Word cooccurrence graph

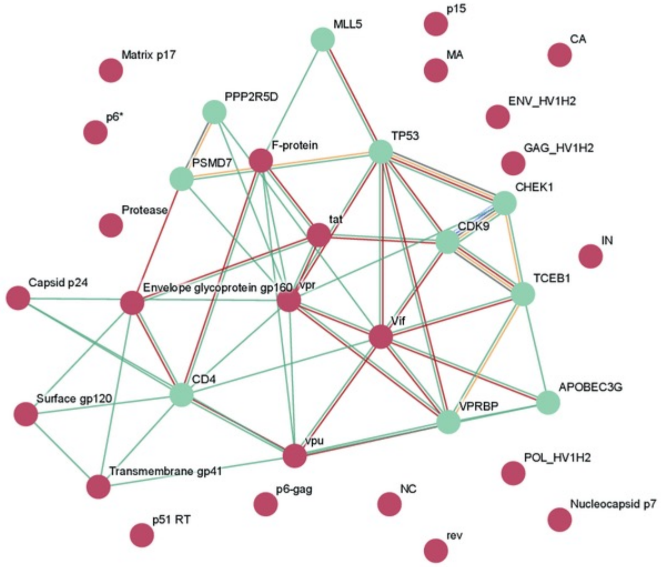
Num. node types $= 1$

Num. edge types $= 1$



Homogeneous graph

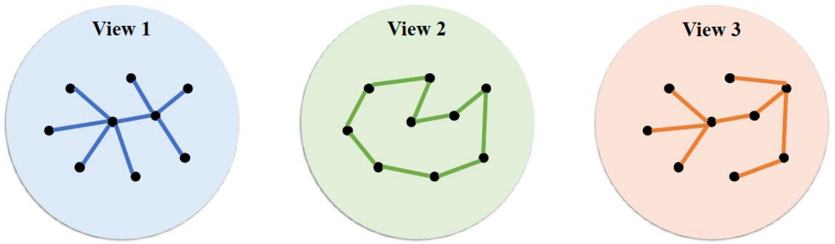
Social graph



Protein-Protein Interaction Graph

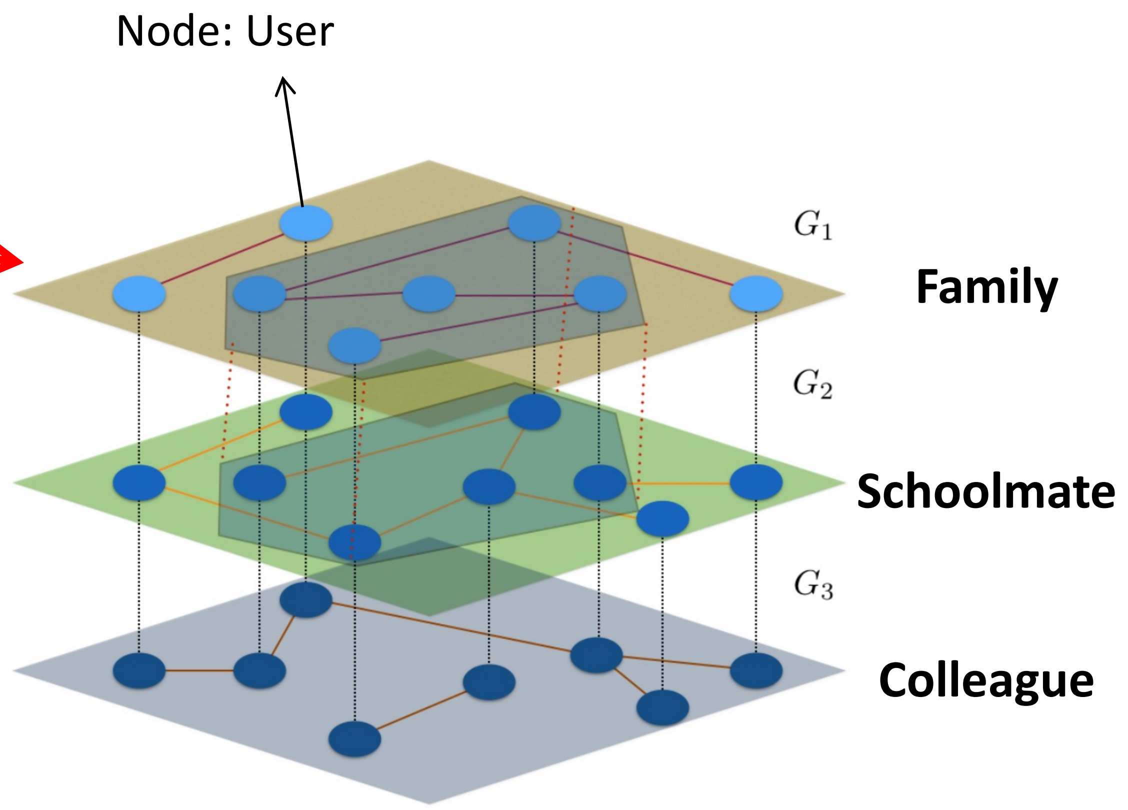
(Figure credit) https://medium.com/analytics-vidhya/social-network-analytics-f082f4e21b16
https://www.researchgate.net/publication/327854066/figure/fig2/AS:674567748075520@1537840892354/HIV-1-and-Homo-sapiens-interaction-network-in-virusesSTRING-HIV-1-and-Homo-sapiens.png
https://commons.wikimedia.org/wiki/File:Word_co-occurrence_network_(range_3_words)_-_ENG.jpg

Multi-layer (Multiplex) Graph



- A type of heterogeneous network
  - A single node type, multiple edge types
- **Example 1: Social network**
  - Relationship between users
- **Example 2: E-commerce**
  - Relationship between items
- **Example 3: Publication network**
  - Relationship between papers (Citation, share authors)
  - Relationship between authors (Co-author, co-citation)
- **Example 4: Movie database**
  - Relationship between movies
    - Common director, common actor
- **Example 5: Transportation network in a city**
  - Relation between locations in a city
    - Bus, train, car, taxi

Num. node types $= 1$
Num. edge types > 1

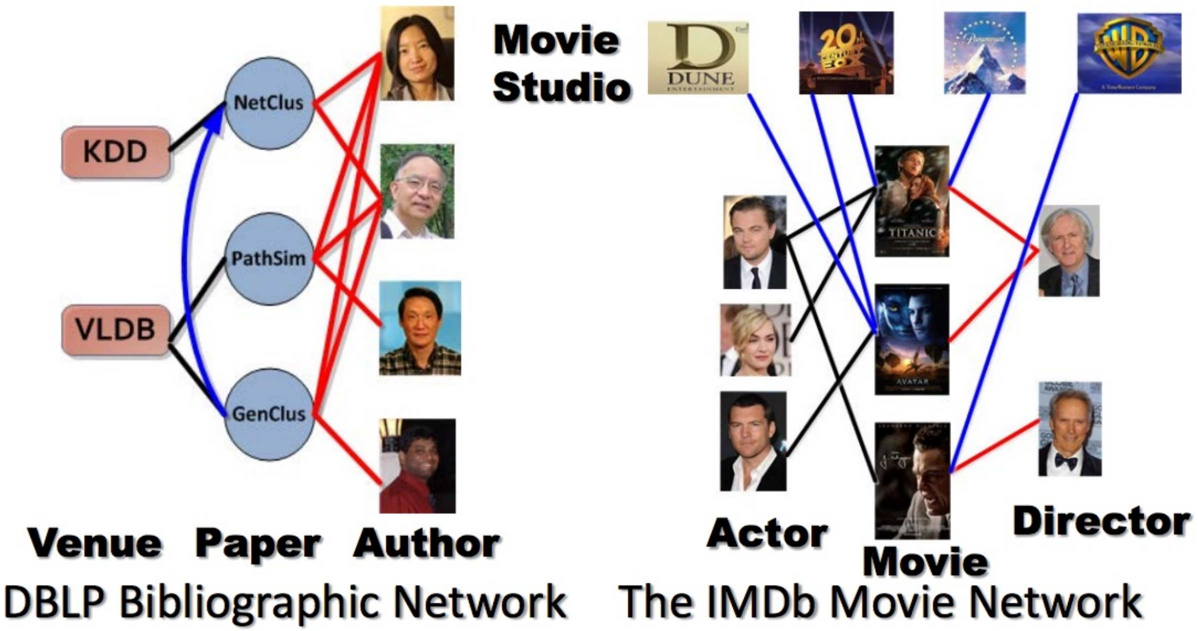


Social Network

Heterogeneous Graph

- So far, we have look at graphs with a single type of node and a single type of edges
- However, in reality a lot of graphs have **multiple types of nodes** and **multiple types of edges**
- Such networks are called **“heterogeneous graph”**

Num. node types $>$ 1
Num. edge types $>$ 1

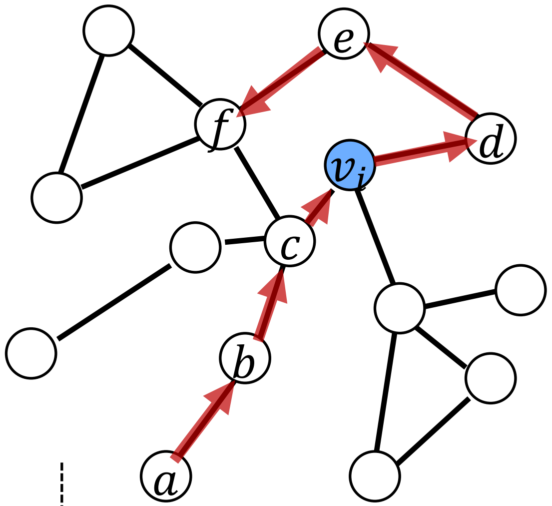


This talk

- How to learn graph representation in **various types of graphs**?
  - ~~GNNs for Homogeneous Graph~~
  - GNNs for Multi-aspect Graph
  - GNNs for Multi-relational Graph
- How to effectively **train GNNs**?
  - Self-supervised learning
  - Alleviating Long-tail problem
  - Robustness of GNN

Deepwalk

- Deepwalk converts a graph into a collection of node sequences through random walk
- Treat random walks on networks as sentences
- Distributional hypothesis
  - **Word embedding**: Words in similar contexts have similar meanings
  - **Node embedding**: Nodes in similar structural contexts are similar



Deepwalk

$$\mathcal{L}_{DW}(\theta) = \sum_{o \in \mathcal{O}} \log p(o|\theta) = \sum_{o \in \mathcal{O}} \log p\big((\mathcal{N}(v_i), v_i)|\theta\big)$$

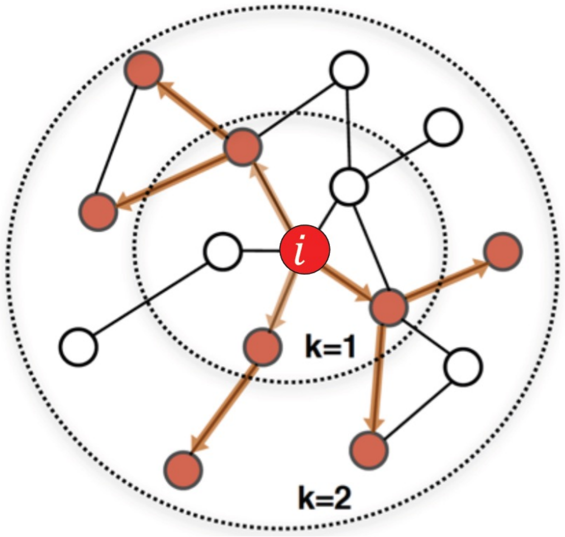
$$= \sum_{o \in \mathcal{O}} \sum_{v_j \in \mathcal{N}(v_i)} \log p(v_j|v_i),$$

- $\mathcal{O}$: The set of all observations obtained from random walks
- $o = (N(v_i), v_i) \in \mathcal{O}$
  - Center node v_i
  - Neighboring nodes $N(v_i)$

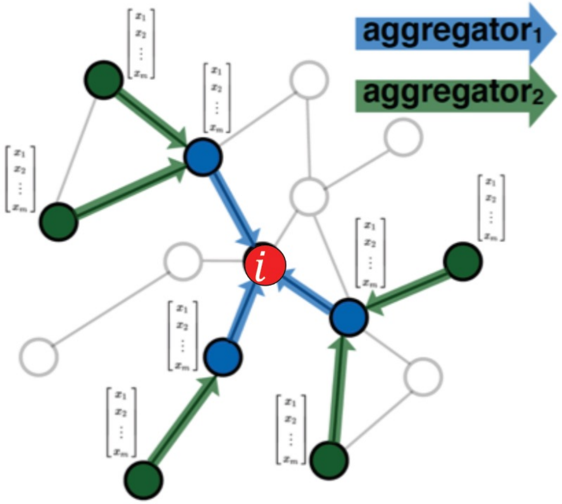
Example seq	$a \to b \to c \to v_i \to d \to e \to f$
Window size=2	$a \to b \to c \to v_i \to d \to e \to f$
Center node	v_i
Neighborhood	$N(v_i) = b, c, d, e$
Observation o	$o = (N(v_i), v_i)$=($\{b, c, d, e\}$,v_i)

Graph Convolutional Network (GCN)

- **Idea**: Node's neighborhood defines a computation graph
  - Messages contain **relational information** + **attribute information**



Determine node computation graph



Propagate messages and transform information

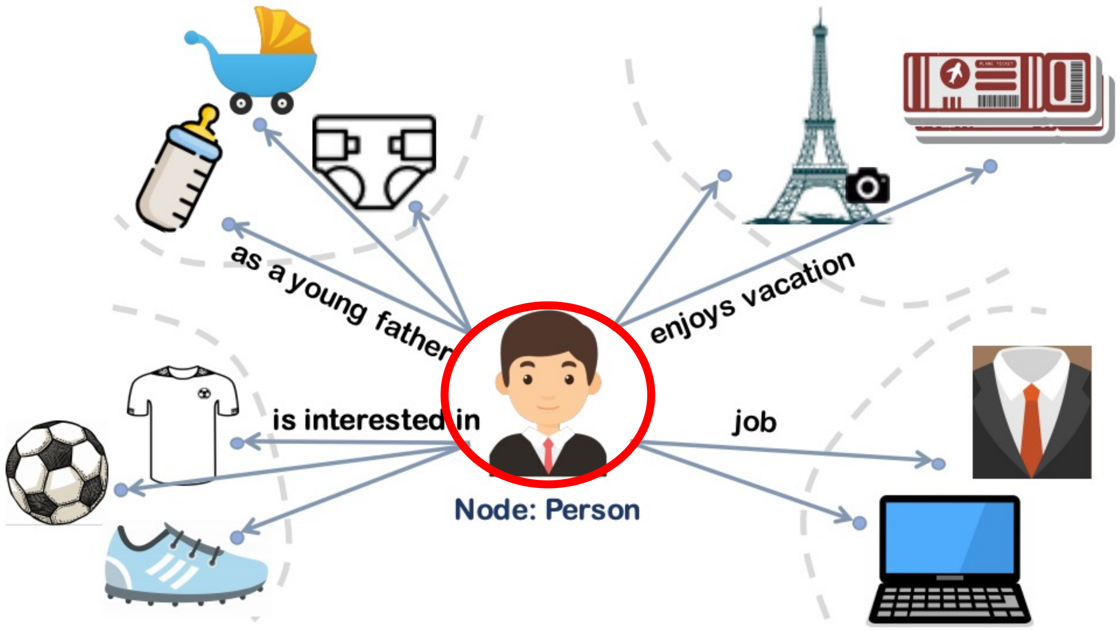
Learn how to propagate information across the graph to compute node features

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

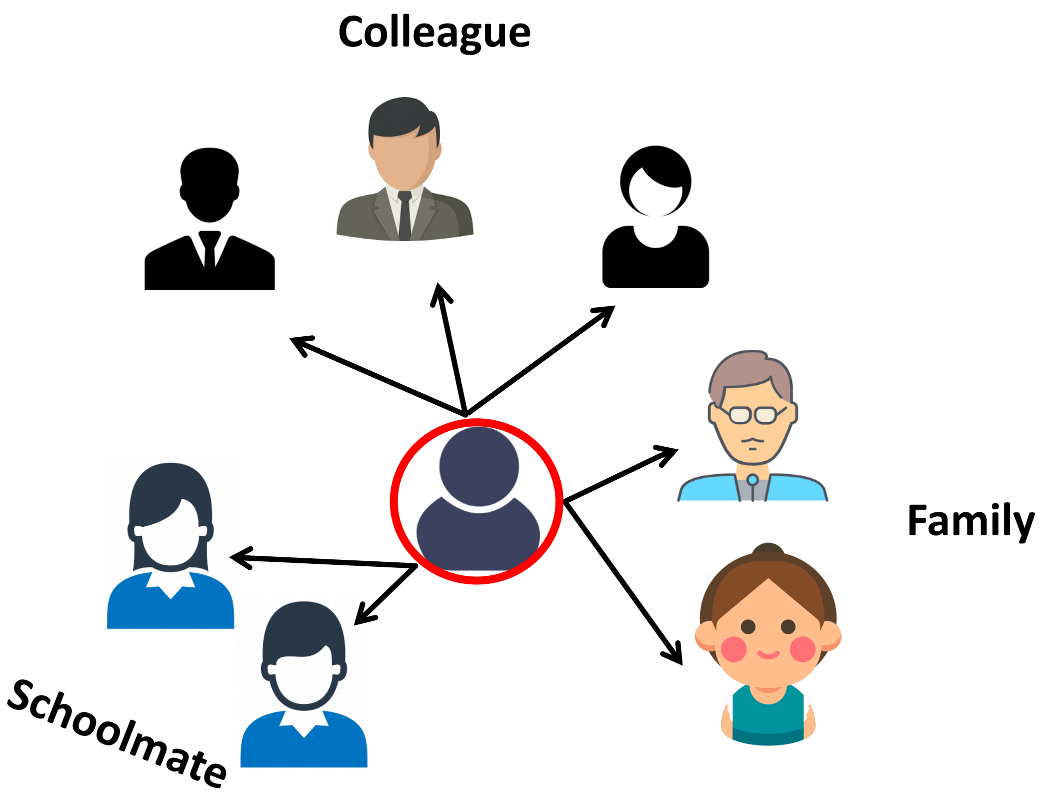
This talk

- How to learn graph representation in **various types of graphs**?
  - ~~GNNs for Homogeneous Graph~~
  - GNNs for Multi-aspect Graph
  - GNNs for Multi-relational Graph
- How to effectively **train GNNs**?
  - Self-supervised learning
  - Alleviating Long-tail problem
  - Robustness of GNN

Is a Single Representation Enough?



Purchase history



Social network

How to differentiate among multiple aspects?

(Figure credit) Is a single vector enough? exploring node polysemy for network embedding, KDD19

PolyDW

- **Idea:** Similar to the idea of Deepwalk, but consider multi-aspect of each node
  - Define the aspect (sense) of each node by clustering the adjacency matrix (offline clustering)
  - For each node and its context nodes, sample an aspect
  - Update the node embeddings of the sampled aspect only

Deepwalk

$$\mathcal{L}_{DW}(\theta) = \sum_{o \in O} \log\ p(o|\theta) = \sum_{o \in O} \log\ p((\mathcal{N}(v_i), v_i)|\theta)$$
$$= \sum_{o \in O} \sum_{v_j \in \mathcal{N}(v_i)} \log\ p(v_j|v_i),$$

PolyDW

$$\mathcal{L}_{PolyDW}(\theta) = \sum_{o \in O} \log\ p(o|\mathcal{P}, \theta)$$

Prior (obtained from clustering)

$$= \sum_{o \in O} \log\left[\sum_{s(o)} p(o|s(o), \mathcal{P}, \theta) \cdot p(s(o)|\mathcal{P}, \theta)\right]$$

Jensen's inequality

$$\geq \sum_{o \in O} \sum_{s(o)} p(s(o)|\mathcal{P}, \theta) \cdot \log\ p(o|s(o), \mathcal{P}, \theta)$$

$$= \sum_{o \in O} \sum_{s(o)} p(s(o)|\mathcal{P}) \cdot \left[\sum_{v_j \in \mathcal{N}(v_i)} \log\ p(v_j|v_i, s(o))\right] = \mathcal{L}^{*}_{PolyDW}(\theta)$$

Cluster membership

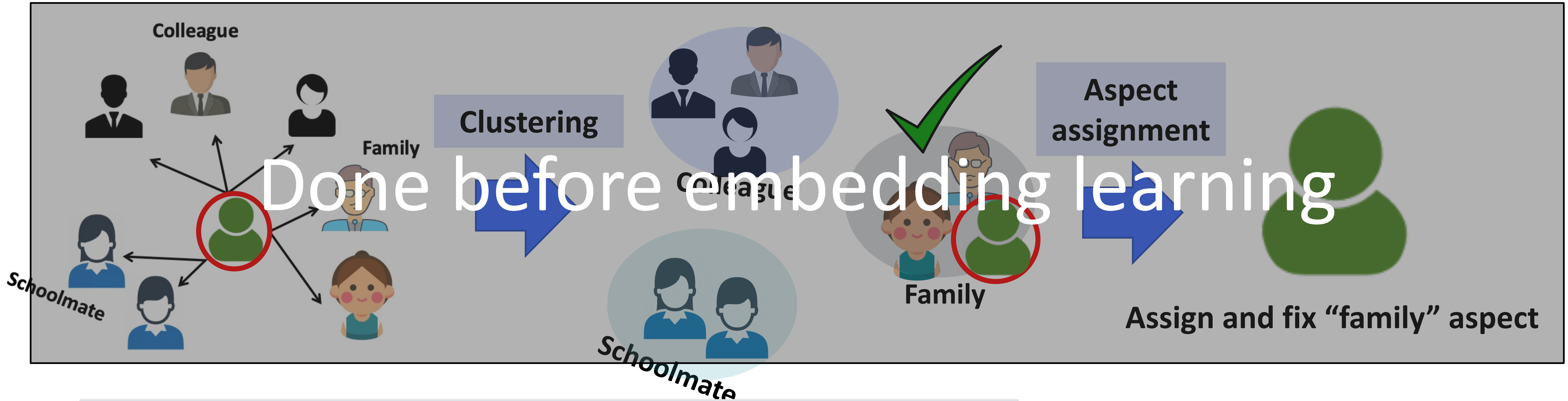
$s(o)$: A set of possible aspects within an observation o

Is a single vector enough? exploring node polysemy for network embedding, KDD 2019

PolyDW

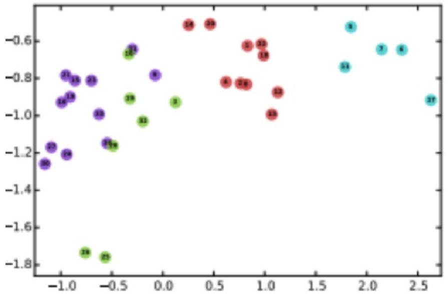

Random Walk 1 (o_1)
v_1 v_2 v_3 v_4 v_6
average
$\mathbf{p}(o_1)$
min
facet sampling for context nodes
facet sampling for target node
$\mathbf{U}_3^1$ $\mathbf{U}_3^2$ $\mathbf{U}_3^3$
$\mathbf{H}_1^1$ $\mathbf{H}_2^1$ $\mathbf{H}_4^2$ $\mathbf{H}_6^1$
Random Walk 2 (o_2)
v_6 v_7 v_8 v_9 v_{10}
average
$\mathbf{p}(o_2)$
min
facet sampling for context nodes
facet sampling for target node
$\mathbf{U}_8^1$ $\mathbf{U}_8^2$ $\mathbf{U}_8^3$
$\mathbf{H}_6^2$ $\mathbf{H}_7^2$ $\mathbf{H}_9^3$ $\mathbf{H}_{10}^3$
Input Network
Node Facet Distribution
Facet Assignment
Objective Optimization

PolyDW: Summary and Limitation



1. Each node always has the same **fixed aspect** regardless of its current context
2. Final network embedding **quality depends on the performance of clustering**
   - Training **cannot be done end-to-end**

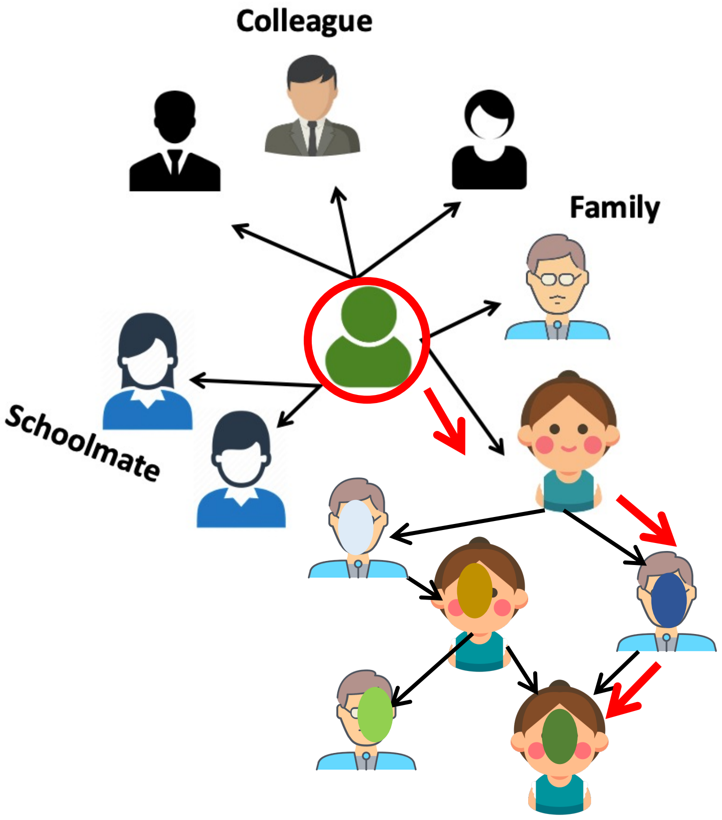
Start network embedding



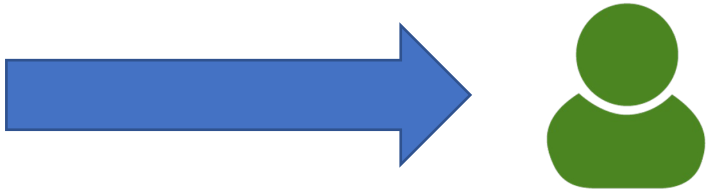
Done!

Asp2vec: Motivation

- **Idea:** Each node should have different aspect according to its neighborhood (context)



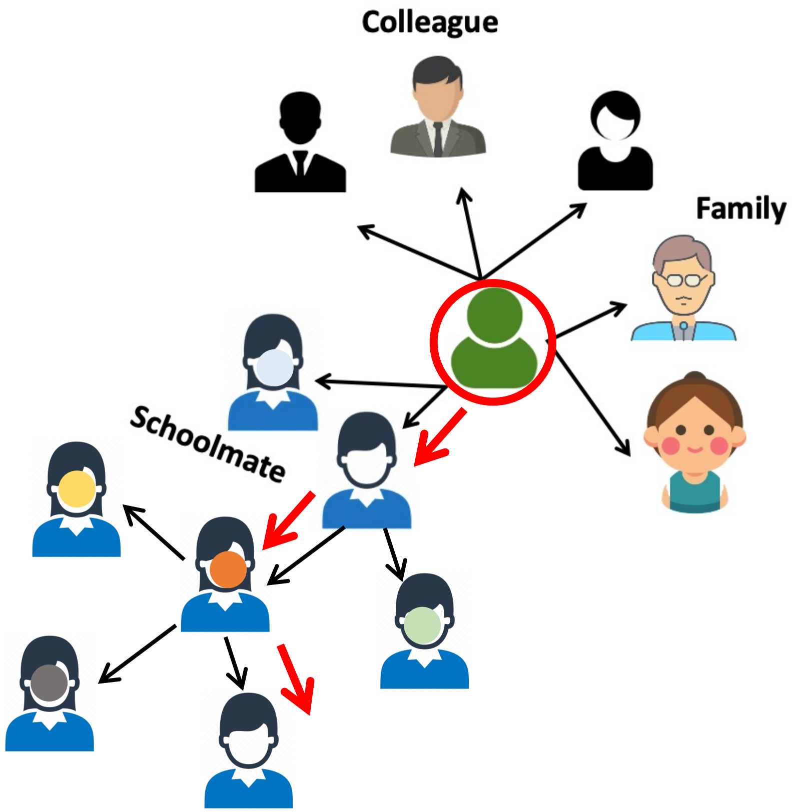
Context: Family



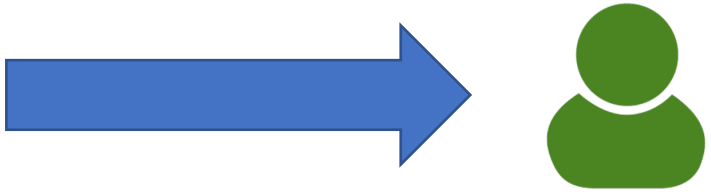
Assign
“Family” aspect

Asp2vec: Motivation

- **Idea:** Each node should have different aspect according to its neighborhood (context)



Context: Schoolmate



Assign
“Schoolmate” aspect

Assign a single aspect for each node based on the context

However, this assignment process is **non-differentiable**

Asp2vec: Overview

- Adopt the **Gumbel-softmax trick** [1] to **dynamically sample aspects based on the context**
  - **Continuous relaxation of discrete random variable**

Gumbel-softmax

Embedding of v_i

Embedding of $N(v_i)$ regarding aspect s

Aspect of node v_i

$$p(\delta(v_i) = s \mid \mathcal{N}(v_i)) = \boxed{\frac{\exp[\langle \mathbf{P}_i, \mathrm{Readout}^{(s)}(\mathcal{N}(v_i))\rangle + g_s]/\tau}{\sum_{s'=1}^{K} \exp[\langle \mathbf{P}_i, \mathrm{Readout}^{(s')}(\mathcal{N}(v_i))\rangle + g_{s'}]/\tau}}$$

Local context of v_i

Sample the aspect that gives the highest value

(Continuous relaxation of discrete random variable **)**

Probability of v_i being selected as aspect s given its context $N(v_i)$

Gumbel-softmax

Differentiable

Gumbel noise drawn from Gumbel(0,1)

$$g_i = -\log(-\log(u_i))$$

$$u_i \sim Uniform(0, 1)$$

$$z_i = \mathrm{softmax}\left[\log \pi_i + g_i\right] = \frac{\exp\left((\log \pi_i + g_i)/\tau\right)}{\sum_{j=1}^{K} \exp\left((\log \pi_j + g_j)/\tau\right)} \quad \text{for } k = 1, \ldots, K$$

Temperature parameter

As $\tau \to 0$, samples from the Gumbel-Softmax distribution become one-hot

[1] Categorical reparameterization with gumbel-softmax, ICLR 2017

Asp2vec: Single-aspect → Multi-aspect

Single-aspect (Deepwalk)

$$\mathcal{J}_{\mathrm{DW}}^{(\mathbf{w})} = \sum_{v_i \in \mathbf{w}} \sum_{v_j \in \mathcal{N}(v_i)} \log \boxed{p(v_j | v_i)}$$

$$\frac{\exp(\langle \mathbf{P}_i, \mathbf{Q}_j \rangle)}{\sum_{v_{j'} \in \mathcal{V}} \exp(\langle \mathbf{P}_i, \mathbf{Q}_{j'} \rangle)}$$

$$\frac{\exp[(\log \langle \mathbf{P}_i, \mathrm{Readout}^{(s)}(\mathcal{N}(v_i)) \rangle + g_s)/\tau]}{\sum_{s'=1}^{K} \exp[(\log \langle \mathbf{P}_i, \mathrm{Readout}^{(s')}(\mathcal{N}(v_i)) \rangle + g_{s'})/\tau]}$$

Multi-aspect (asp2vec)

$$\mathcal{J}_{\mathrm{asp2vec}}^{(\mathbf{w})} = \sum_{v_i \in \mathbf{w}} \sum_{v_j \in \mathcal{N}(v_i)} \sum_{s=1}^{K} \boxed{p(\delta(v_i) = s | \mathcal{N}(v_i))} \log \boxed{p(v_j | v_i, p(\delta(v_i) = s))}$$

Aspect selection probability

$$\frac{\exp(\langle \mathbf{P}_i, \mathbf{Q}_j^{(s)} \rangle)}{\sum_{v_{j'} \in \mathcal{V}} \exp(\langle \mathbf{P}_i, \mathbf{Q}_{j'}^{(s)} \rangle)}$$

Final objective function

$$\mathcal{L}_{\mathrm{asp2vec}} = - \sum_{\mathbf{w} \in \mathcal{W}} \mathcal{J}_{\mathrm{asp2vec}}^{(\mathbf{w})}$$

Asp2vec: Is Multi-aspect Enough?

- Authors can belong to multiple research communities
- **These communities interact with one another**

	More Related		Less related	
Database	⟷	**Data Mining**	⟷	**Computer Architecture**

Interactions among aspects should be captured

- **Goal**: Aspect embeddings should be
  - 1. Related to each other (**Relatedness**)
    - To capture some common information shared among aspects (e.g., DM ↔ DB)
  - 2. Diverse from each other (**Diversity**)
    - To independently capture the inherent properties of individual aspects (e.g., DM ↔ CA)

How can we capture both relatedness and diversity among aspects?

Asp2vec: Capturing Diversity and Relatedness among Aspects

- **Capturing diversity**: Minimize similarity among aspect embeddings (= maximize diversity)

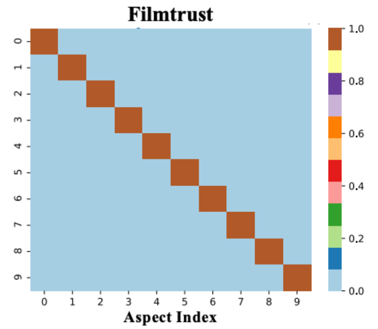
$$\mathrm{reg}_{\mathrm{asp}} = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \mathrm{A\text{-}Sim}(\mathbf{Q}_*^{(i)}, \mathbf{Q}_*^{(j)}) \qquad \mathbf{Q}_*^{(i)} \in \mathbb{R}^{n \times d}$$

(Aspect embedding matrix w.r.t. aspect i)

Aspect similarity between aspect i and j

$$\mathrm{A\text{-}Sim}(\mathbf{Q}_*^{(i)}, \mathbf{Q}_*^{(j)}) = \sum_{h=1}^{|V|} f(\mathbf{Q}_h^{(i)}, \mathbf{Q}_h^{(j)}) \qquad f(\mathbf{Q}_h^{(i)}, \mathbf{Q}_h^{(j)}) = \frac{\langle \mathbf{Q}_h^{(i)}, \mathbf{Q}_h^{(j)} \rangle}{\|\mathbf{Q}_h^{(i)}\| \|\mathbf{Q}_h^{(j)}\|}, \quad -1 \le f(\mathbf{Q}_h^{(i)}, \mathbf{Q}_h^{(j)}) \le 1$$

(Cosine similarity)

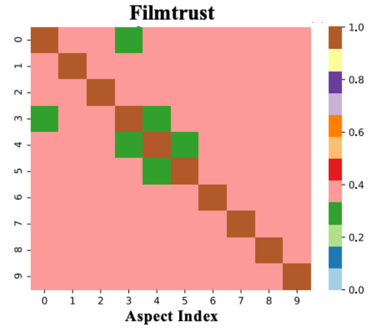


$\epsilon = 0.1$

- **Capturing relatedness**: Allow similarity among aspects **to some extent**

$$\mathrm{A\text{-}Sim}(\mathbf{Q}_*^{(i)}, \mathbf{Q}_*^{(j)}) = \sum_{h=1}^{|V|} w_{i,j}^h f(\mathbf{Q}_h^{(i)}, \mathbf{Q}_h^{(j)})$$

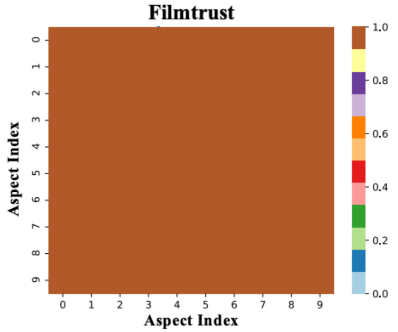
(Maximize diversity + allow some similarity)



$\epsilon = 0.9$

$$w_{i,j}^h = \begin{cases} 1, & \left| f(\mathbf{Q}_h^{(i)}, \mathbf{Q}_h^{(j)}) \right| \ge \epsilon \\ 0, & \text{otherwise} \end{cases}$$

- Enforce loss if similarity is larger than ϵ
  - Allow similarity as much as ϵ



$\epsilon = 1.0$

(No reg_{asp})

Asp2vec: Final Objectives

$$\mathcal{L} = \mathcal{L}_{\text{asp2vec}} + \lambda\, \text{reg}_{\text{asp}}$$

Multi-aspect embedding | **Aspect regularization**

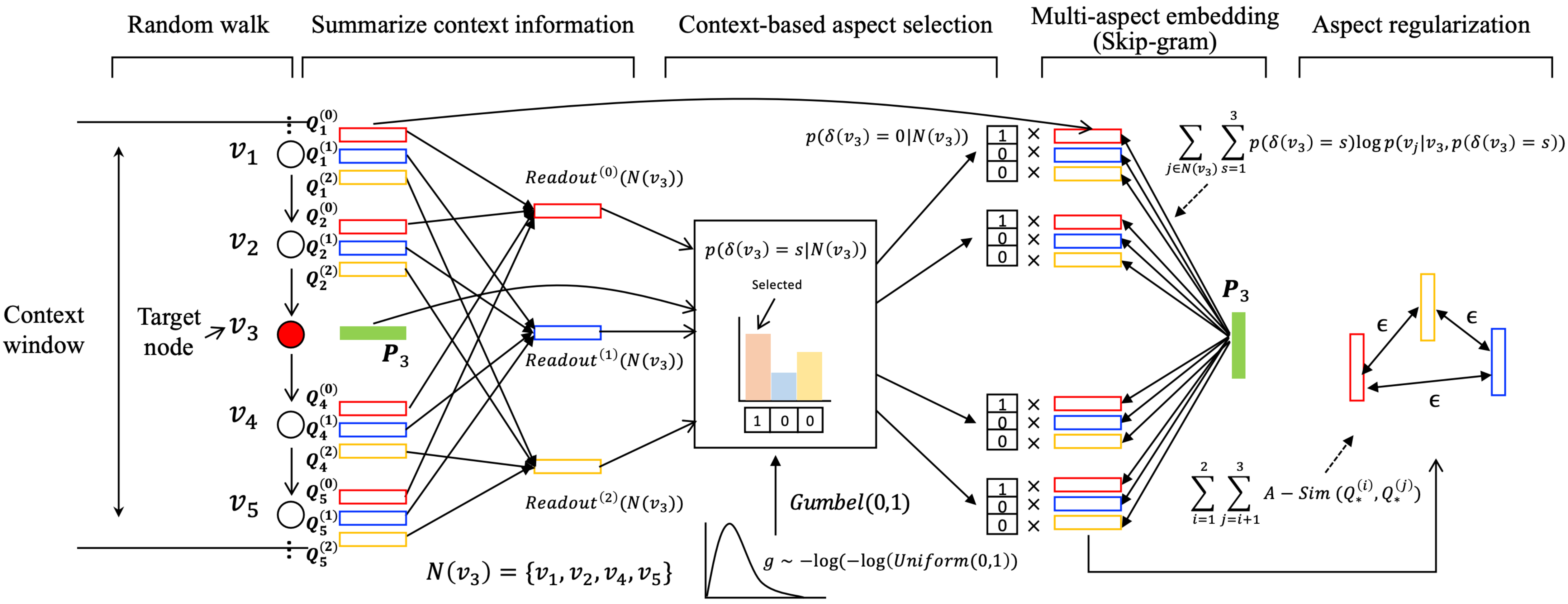
$$\mathcal{L}_{\text{asp2vec}} = -\sum_{\mathbf{w}\in\mathcal{W}} \mathcal{J}^{(\mathbf{w})}_{\text{asp2vec}}$$

$$\mathcal{J}^{(\mathbf{w})}_{\text{asp2vec}} = \sum_{v_i\in\mathbf{w}} \sum_{v_j\in\mathcal{N}(v_i)} \sum_{s=1}^{K} p(\delta(v_i)=s|\mathcal{N}(v_i)) \log p(v_j|v_i, p(\delta(v_i)=s))$$

$$\text{reg}_{\text{asp}} = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \text{A-Sim}(\mathbf{Q}_*^{(i)}, \mathbf{Q}_*^{(j)})$$

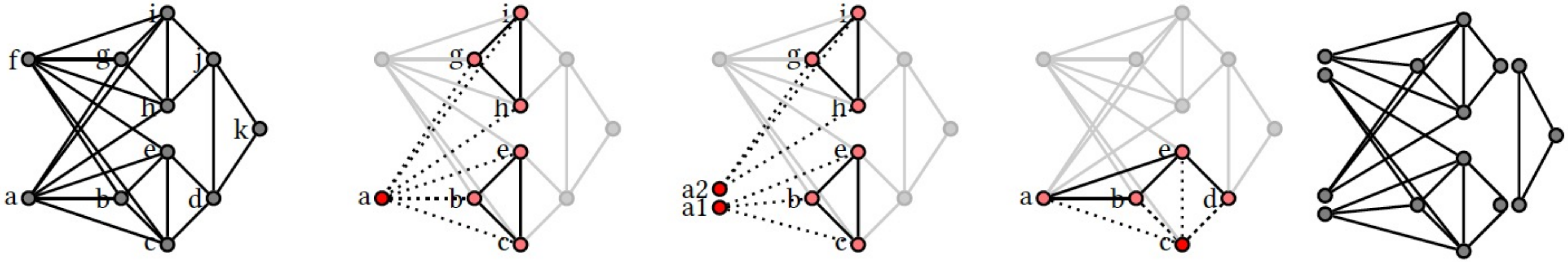
$$\text{A-Sim}(\mathbf{Q}_*^{(i)}, \mathbf{Q}_*^{(j)}) = \sum_{h=1}^{|V|} w_{i,j}^{h} f(\mathbf{Q}_h^{(i)}, \mathbf{Q}_h^{(j)})$$

Asp2vec: Architecture



Splitter

- Given an original graph, compute a **persona graph**
  - Add constraints on Deepwalk to relate the persona graph with the original graph



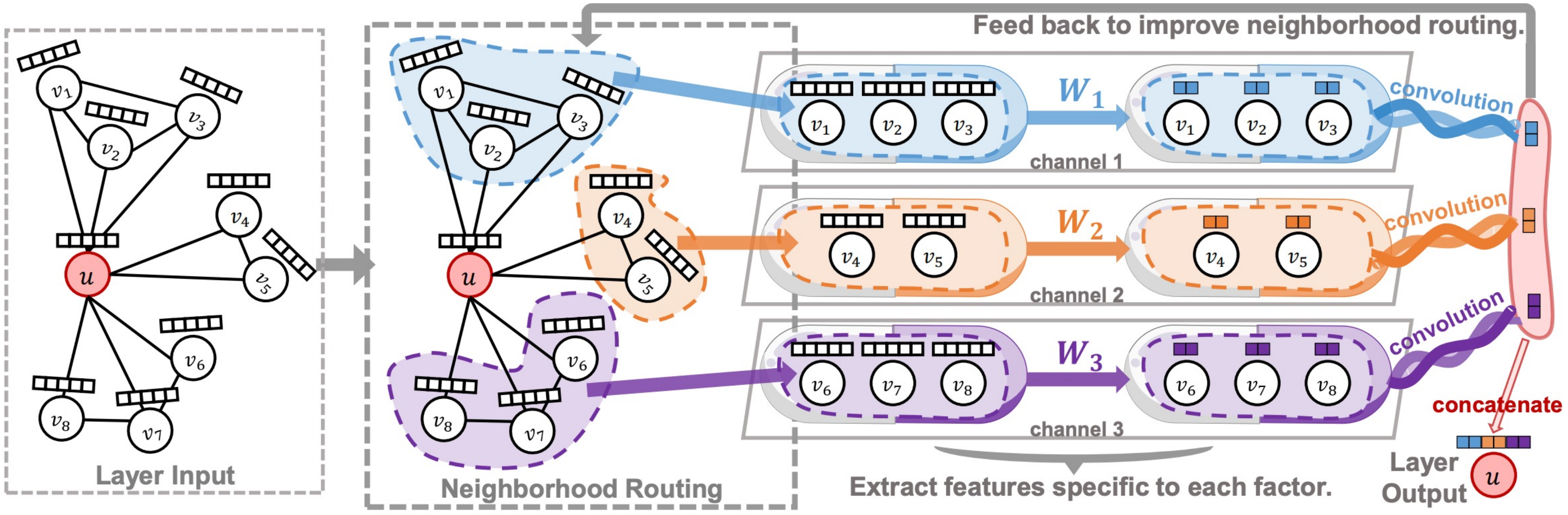
(a) original graph G (b) ego-net of a (c) splitting a in two personas (d) ego-net of c (one persona) (e) persona graph

$$\underset{\Phi_{G_P}}{\text{minimize}} \quad -\log \Pr\left(\{v_{i-w}, \cdots, v_{i+w}\} \setminus v_i \mid \Phi_{G_P}(v_i)\right) - \lambda \log \Pr\left(v_o \mid \Phi_{G_P}(v_i)\right).$$

Predict the $N(v_i)$ using the persona of v_i

Predict the original embedding of v_i using its persona

DisenGCN: Disentangled Graph Convolutional Networks



Step 1: Project x_i into K different subspaces (aspects)

$$\mathbf{z}_{i,k} = \frac{\sigma(\mathbf{W}_k^\top \mathbf{x}_i + \mathbf{b}_k)}{\left\|\sigma(\mathbf{W}_k^\top \mathbf{x}_i + \mathbf{b}_k)\right\|_2} \qquad \mathbf{W}_k \in \mathbb{R}^{d_{in} \times \frac{d_{out}}{K}} \quad \mathbf{b}_k \in \mathbb{R}^{\frac{d_{out}}{K}}$$

- l2-normalization to ensure numerical stability
- $z_{i,k}$ approximately describes the aspect of node i that are related with the k-th factor

Step 2: How do we know which of the neighbors belong to which channel?

$$p_{v,k}^{(t)} = \frac{\exp(\mathbf{z}_{v,k}{}^\top \mathbf{c}_k^{(t)}/\tau)}{\sum_{k'=1}^{K} \exp(\mathbf{z}_{v,k'}{}^\top \mathbf{c}_{k'}^{(t)}/\tau)}$$

$p_{v,k}$: probability that factor k is the reason why node u reaches neighbor

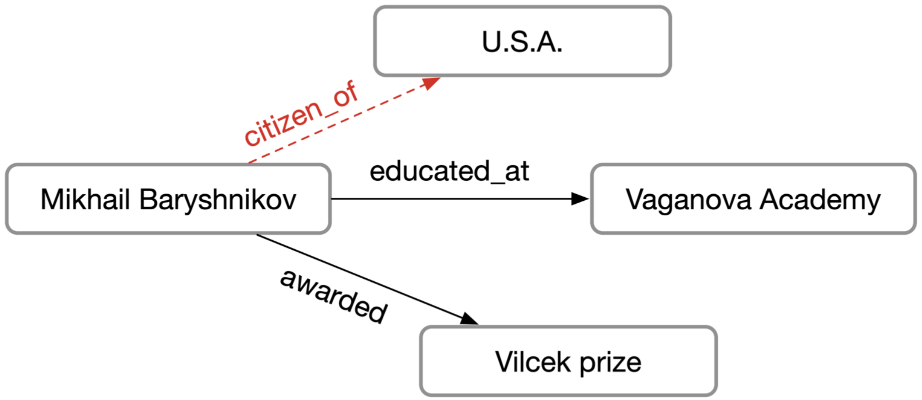
$$\mathbf{c}_k^{(t)} = \frac{\mathbf{z}_{u,k} + \sum_{v:(u,v)\in G} p_{v,k}^{(t-1)} \mathbf{z}_{v,k}}{\|\mathbf{z}_{u,k} + \sum_{v:(u,v)\in G} p_{v,k}^{(t-1)} \mathbf{z}_{v,k}\|_2}$$

c_k: The final output of the k-th channel (combination of the current node u and its neighbors)

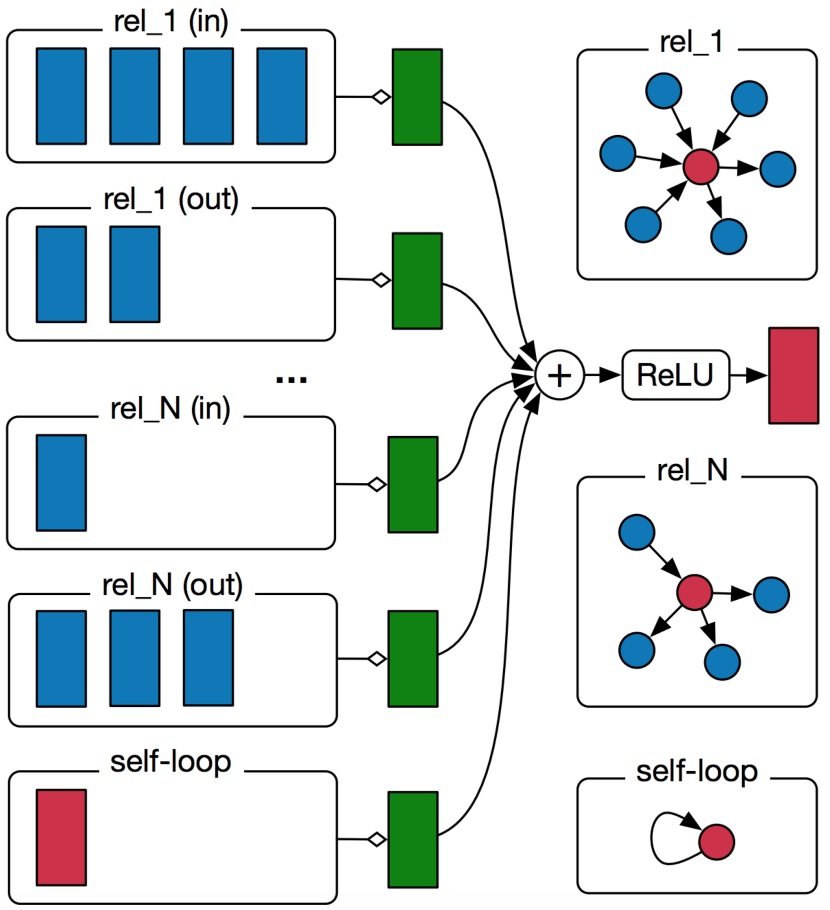
This talk

- How to learn graph representation in **various types of graphs**?
  - ~~GNNs for Homogeneous Graph~~
  - GNNs for Multi-aspect Graph
  - GNNs for Multi-relational Graph
- How to effectively **train GNNs**?
  - Self-supervised learning
  - Alleviating Long-tail problem
  - Robustness of GNN

R-GCN: Relational GCN



- Knowledge graph is a type of multiplex network
  - Nodes are entities, the edges are relations labeled with their types



GCN

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathrm{N_i}} \frac{1}{c_i} \mathrm{W}^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)}\right)$$

R-GCN

$$h_i^{(l+1)} = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)}\right)$$

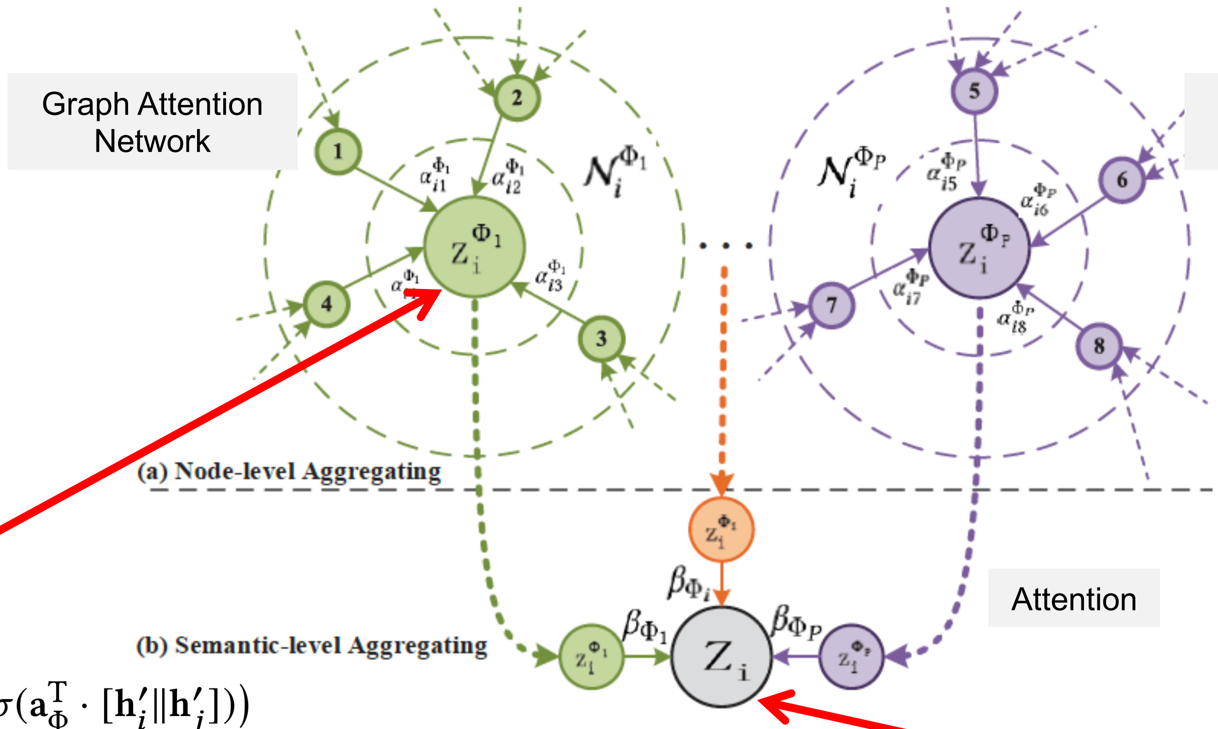
$$W_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} V_b^{(l)}$$

Address overfitting and rapid growth in # parameters

Modeling Relational Data with Graph Convolutional Networks, ESWC2018

HAN: Heterogeneous Graph Attention Network

- **Idea:** Apply graph attention networks to each network and then aggregate through attention



$$\mathbf{z}_i^{\Phi} = \sigma\left(\sum_{j \in \mathcal{N}_i^{\Phi}} \alpha_{ij}^{\Phi} \cdot \mathbf{h}_j'\right)$$

$$\alpha_{ij}^{\Phi} = softmax_j(e_{ij}^{\Phi}) = \frac{\exp\left(\sigma(\mathbf{a}_{\Phi}^{\mathrm{T}} \cdot [\mathbf{h}_i' \| \mathbf{h}_j'])\right)}{\sum_{k \in \mathcal{N}_i^{\Phi}} \exp\left(\sigma(\mathbf{a}_{\Phi}^{\mathrm{T}} \cdot [\mathbf{h}_i' \| \mathbf{h}_k'])\right)},$$

$$e_{ij}^{\Phi} = att_{node}(\mathbf{h}_i', \mathbf{h}_j'; \Phi).$$

$$\mathrm{Z} = \sum_{p=1}^{P} \beta_{\Phi_p} \cdot \mathrm{Z}_{\Phi_p}$$

Background: Mutual Information (MI)

- Measures the amount of information that two variables share
- If X and Y are independent, then $P_{XY} = P_X P_Y$ → in this case, MI = 0

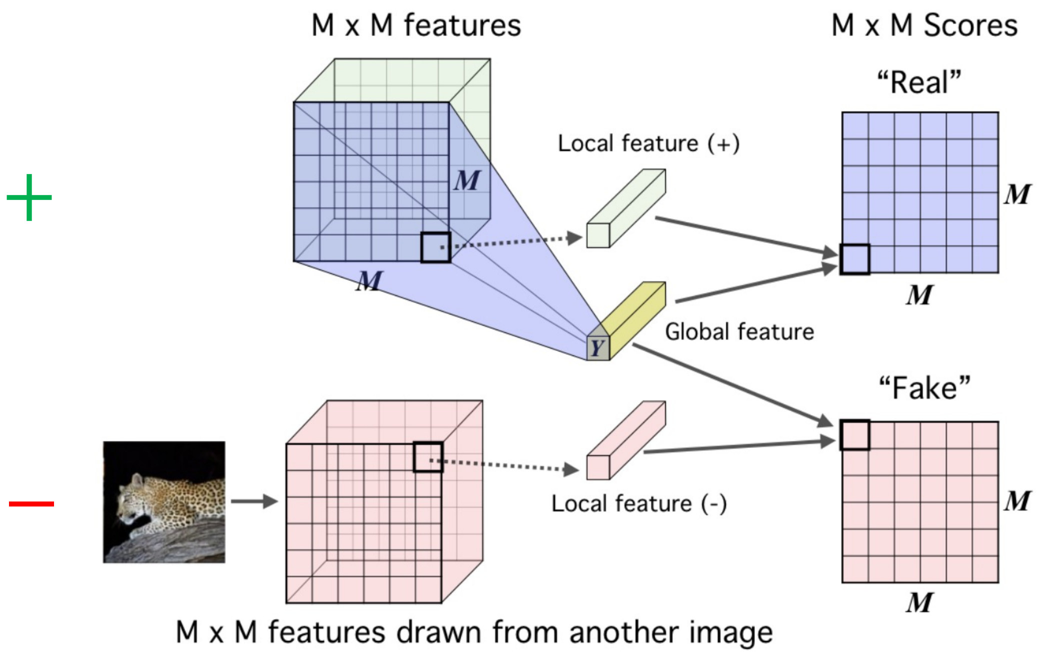
$$I(X;Y) = \mathbb{E}_{P_{XY}}\left[\log \frac{P_{XY}}{P_X P_Y}\right]$$
$$= D_{KL}(P_{XY}||P_X P_Y)$$

- High MI? → One variable is always indicative of the other variable
- Recently, scalable estimation of mutual information was made both possible and practical through **Mutual Information Neural Estimation** (MINE)

MINE: Mutual Information Neural Estimation, ICML 2018

Deep Infomax

- Unsupervised representation learning method for image data
- Intuition: **Maximize mutual information (MI)** between local patches and the global representation of an image



Discriminator tries to discriminate between "Real" and "Fake"

Deep Graph Infomax

- Deep Graph Infomax (DGI) applies Deep Infomax on graph domain
- Unsupervised graph representation learning method that considers node features
- Notations

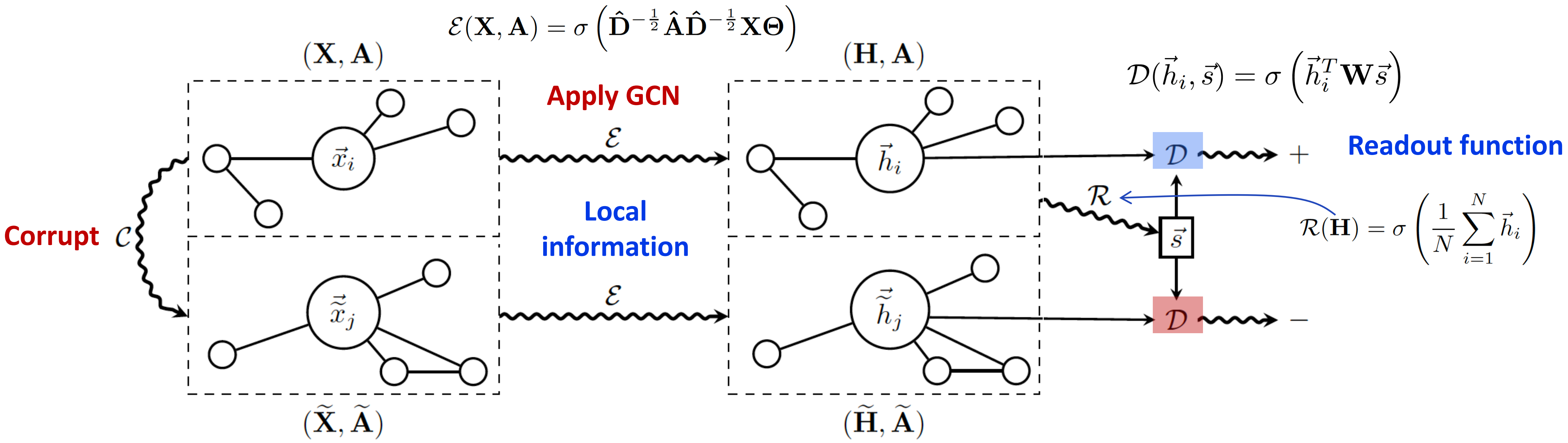
  $\mathbf{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$: A set of node features (N: number of nodes) $\vec{x}_i \in \mathbb{R}^F$

  $\mathbf{A} \in \mathbb{R}^{N \times N}$: Adjacency matrix

- Learn a **graph convolutional** encoder $\mathcal{E}(\mathbf{X}, \mathbf{A}) = \mathbf{H} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}$ $\vec{h}_i \in \mathbb{R}^{F'}$
  - Generates node representations by **repeated aggregation over local node neighborhoods**
  - $\vec{h}_i$ summarizes a patch of the graph centered around node i (≈ patch representation)

Analogy: Local patch representation in an image == Node representation in a graph

Deep Graph Infomax



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

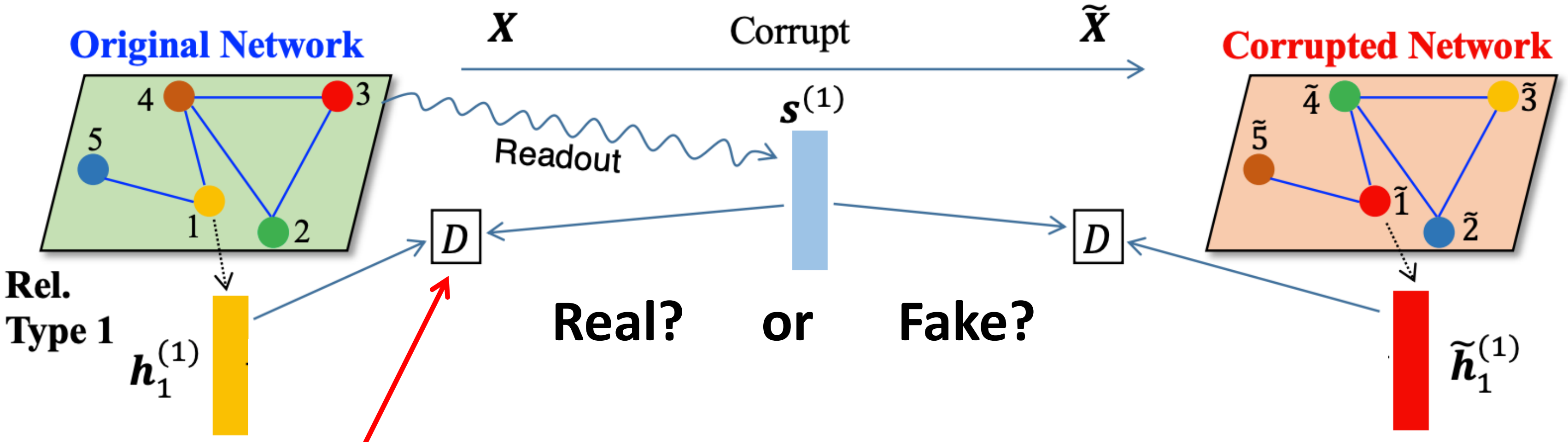
Maximizes the mutual information between the local patches (h_i) and the graph-level global representation (s)

Deep Graph Infomax, ICLR 2019

DMGI: Unsupervised Attributed Multiplex Network Embedding

- **Idea:** Adopt infomax principal to multiplex network

Schoolmate

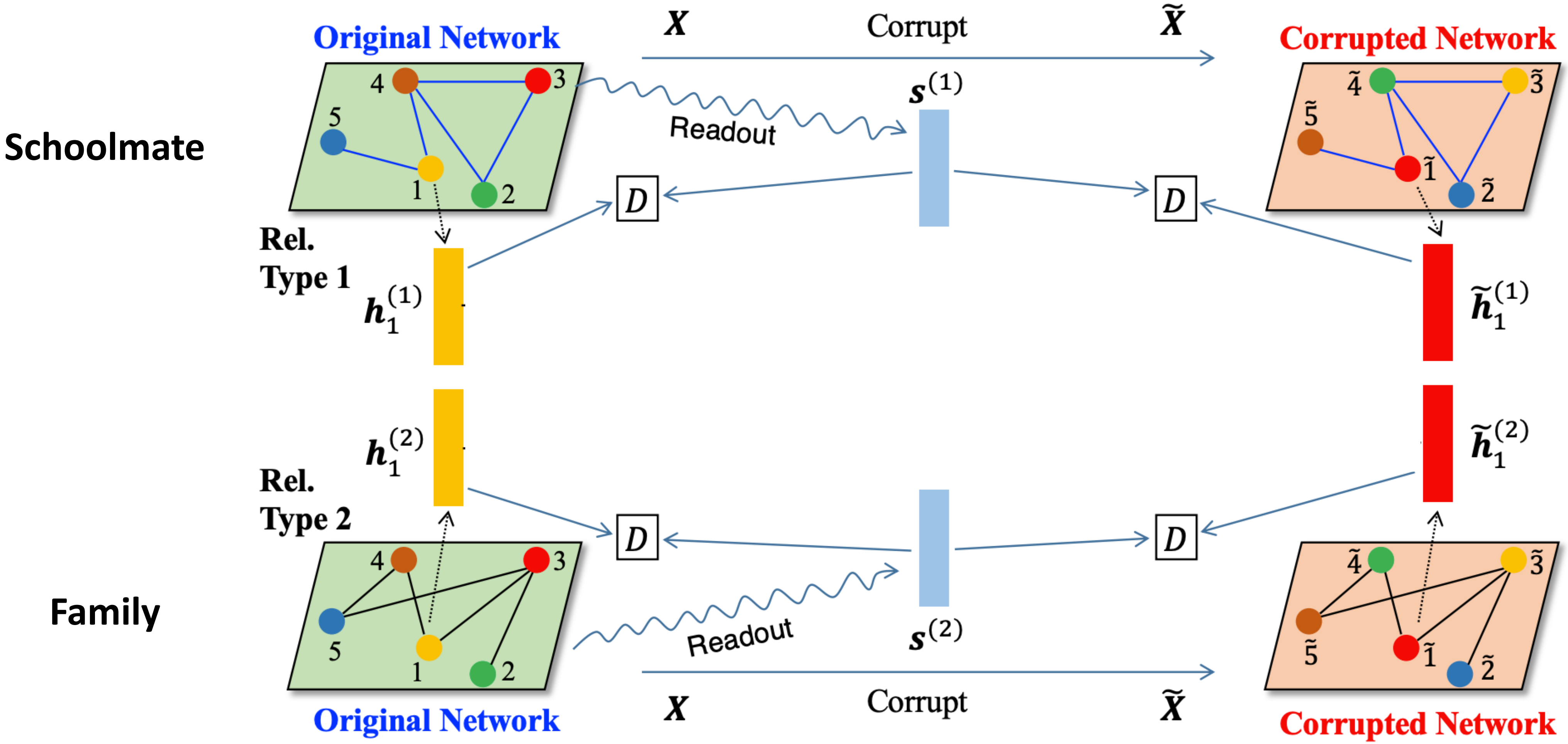


DGI

$$\mathcal{D}\left(\mathbf{h}_i^{(r)}, \mathbf{s}^{(r)}\right) = \sigma(\mathbf{h}_i^{(r)T}\mathbf{M}^{(r)}\mathbf{s}^{(r)})$$

Score matrix w.r.t. relation r

DMGI: Unsupervised Attributed Multiplex Network Embedding

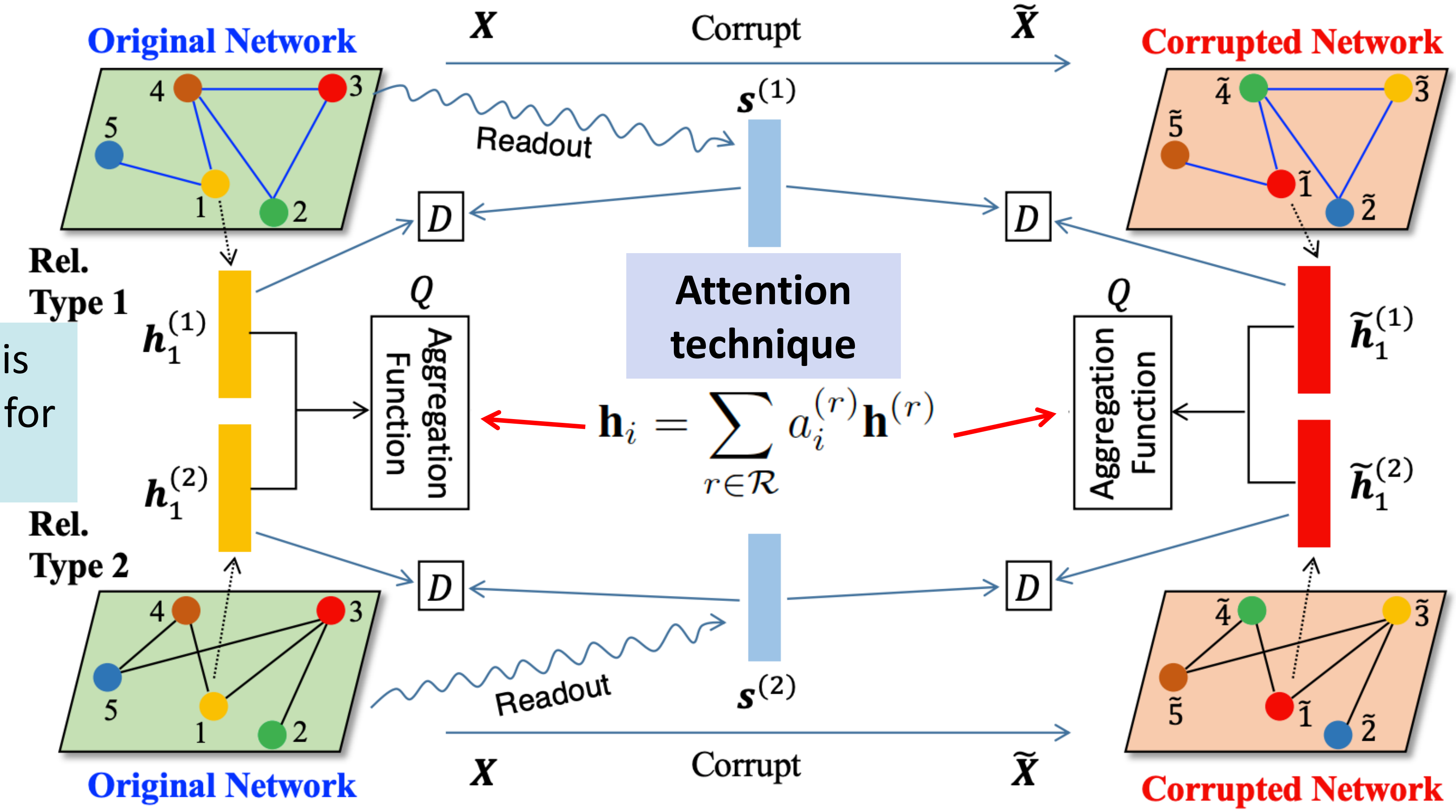


DMGI: Unsupervised Attributed Multiplex Network Embedding

Schoolmate

Which relation is **more important** for each node?

Family

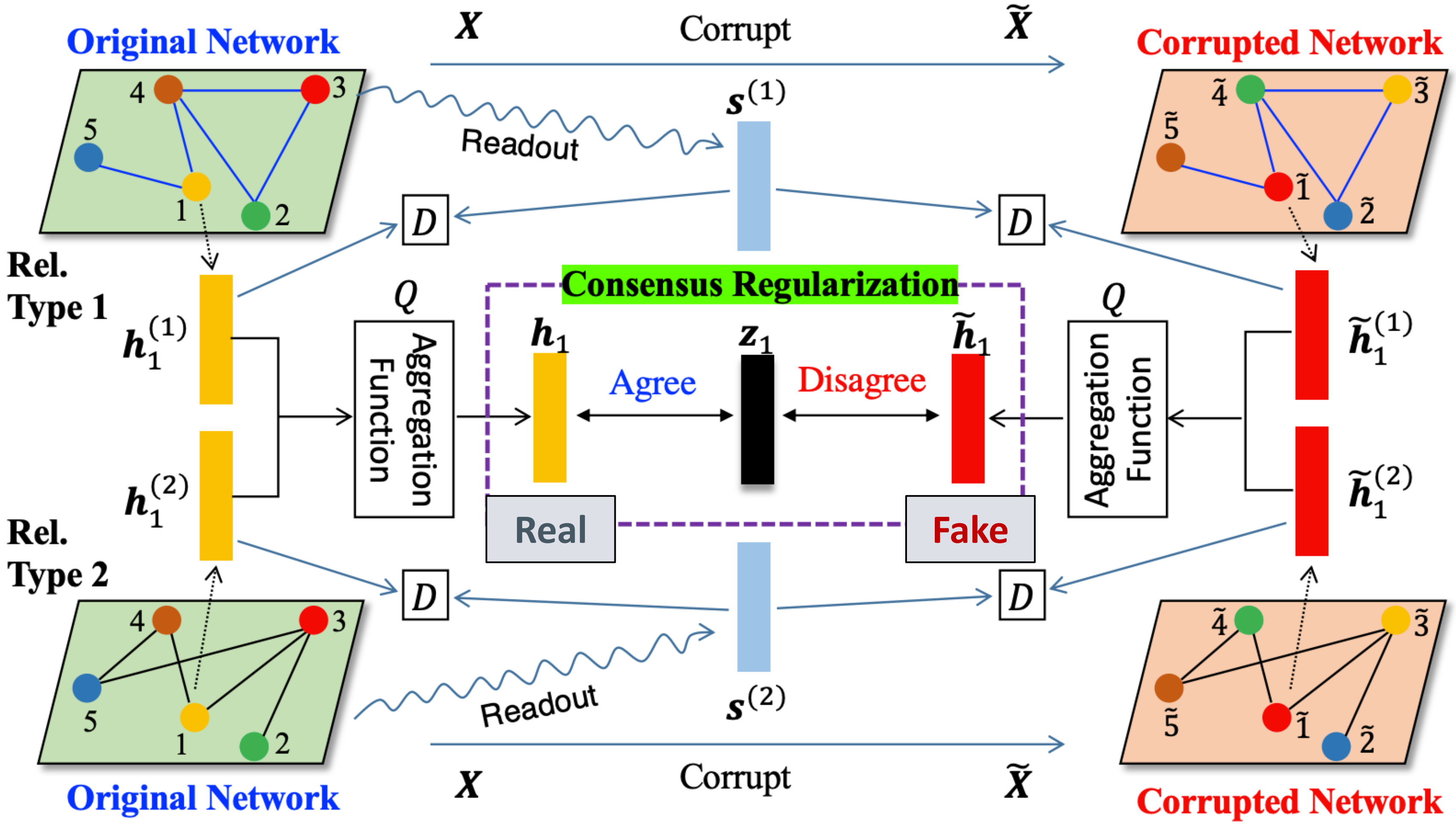


Unsupervised Attributed Multiplex Network Embedding, AAAI 2020

DMGI: Unsupervised Attributed Multiplex Network Embedding

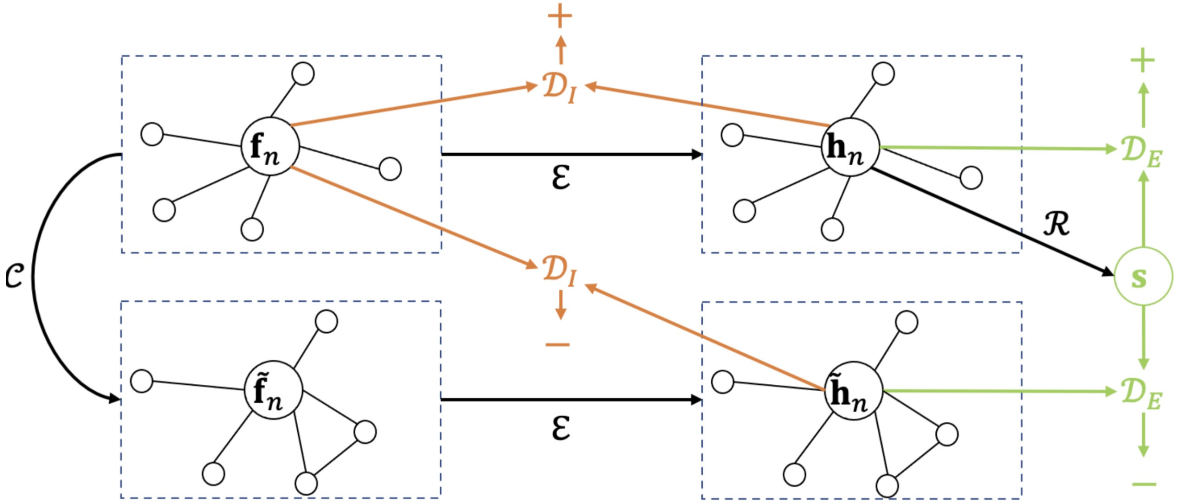
Schoolmate

Family



HDGI: High-order Deep Graph Infomax

- **Idea: High-order Mutual Information**
  - We should not only consider the extrinsic supervision signal, i.e., $\boldsymbol{s} \leftrightarrow \boldsymbol{h}$, but also **intrinsic signal**, i.e., $\boldsymbol{f} \leftrightarrow \boldsymbol{h}$



$$I(\mathbf{h}_n; \mathbf{s}; \mathbf{f}_n) = I(\mathbf{h}_n; \mathbf{s}) + I(\mathbf{h}_n; \mathbf{f}_n) - I(\mathbf{h}_n; \mathbf{s}, \mathbf{f}_n)$$

Difference-based estimation (Mukherjee et al, 2020)

$$\max I(\mathbf{h}_n; \mathbf{s}; \mathbf{f}_n) = \max I(\mathbf{h}_n; \mathbf{s}) + \max I(\mathbf{h}_n; \mathbf{f}_n) - \max I(\mathbf{h}_n; \mathbf{s}, \mathbf{f}_n)$$

Empirical finding

$$\mathcal{L} = \lambda_E I(\mathbf{h}_n; \mathbf{s}) + \lambda_I I(\mathbf{h}_n; \mathbf{f}_n) + \lambda_J I(\mathbf{h}_n; \mathbf{s}, \mathbf{f}_n)$$

$$I(X; Y) = H(X) + H(Y) - H(X, Y)$$

DGI

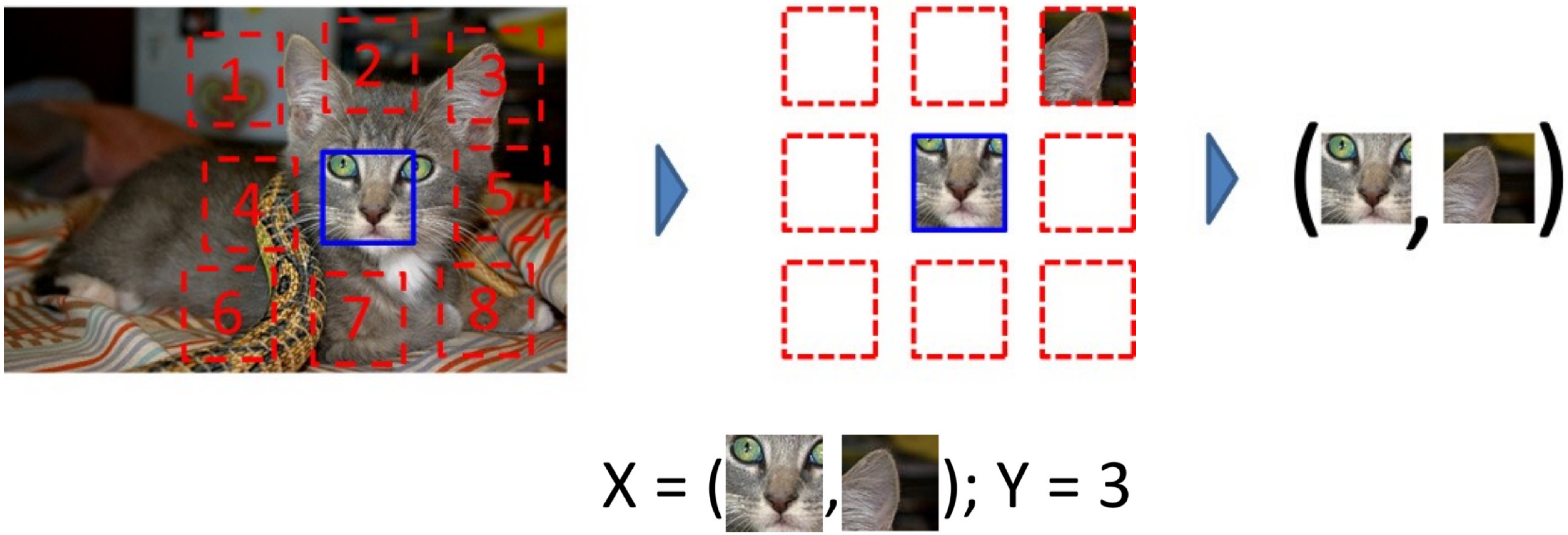
$$\begin{aligned} I(X_1; X_2; X_3) = & H(X_1) + H(X_2) + H(X_3) \\ & - H(X_1, X_2) - H(X_1, X_3) - H(X_2, X_3) \\ & + H(X_1, X_2, X_3) \\ = & H(X_1) + H(X_2) - H(X_1, X_2) \\ & + H(X_1) + H(X_3) - H(X_1, X_3) \\ & - H(X_1) - H(X_2, X_3) + H(X_1, X_2, X_3) \\ = & I(X_1; X_2) + I(X_1; X_3) - I(X_1; X_2, X_3) \end{aligned}$$

This talk

- How to learn graph representation in **various types of graphs**?
  - ~~GNNs for Homogeneous Graph~~
  - GNNs for Multi-aspect Graph
  - GNNs for Multi-relational Graph
- How to effectively **train GNNs**?
  - Self-supervised learning
  - Alleviating Long-tail problem
  - Robustness of GNN

What is self-supervised learning?

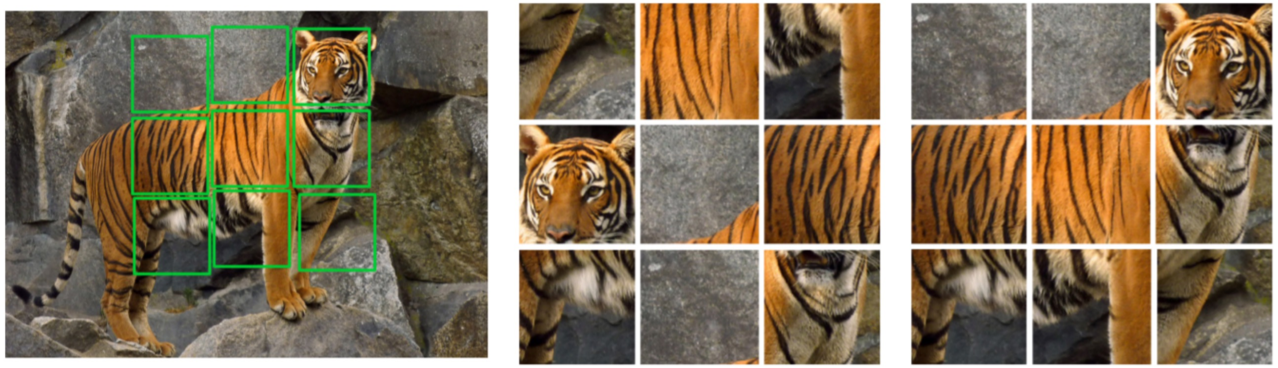
- A form of unsupervised learning where the data provides the supervision
- In general, withhold some part of the data, and task the network with predicting it
- An example of **pretext task**: **Relative positioning**
  - Train network to predict relative position of two regions in the same image



(Figure credit) Unsupervised visual representation learning by context prediction, ICCV2015

What is self-supervised learning?

- Pretext task: **Jigsaw puzzle**



(a) (b) Shuffled (c) Solved

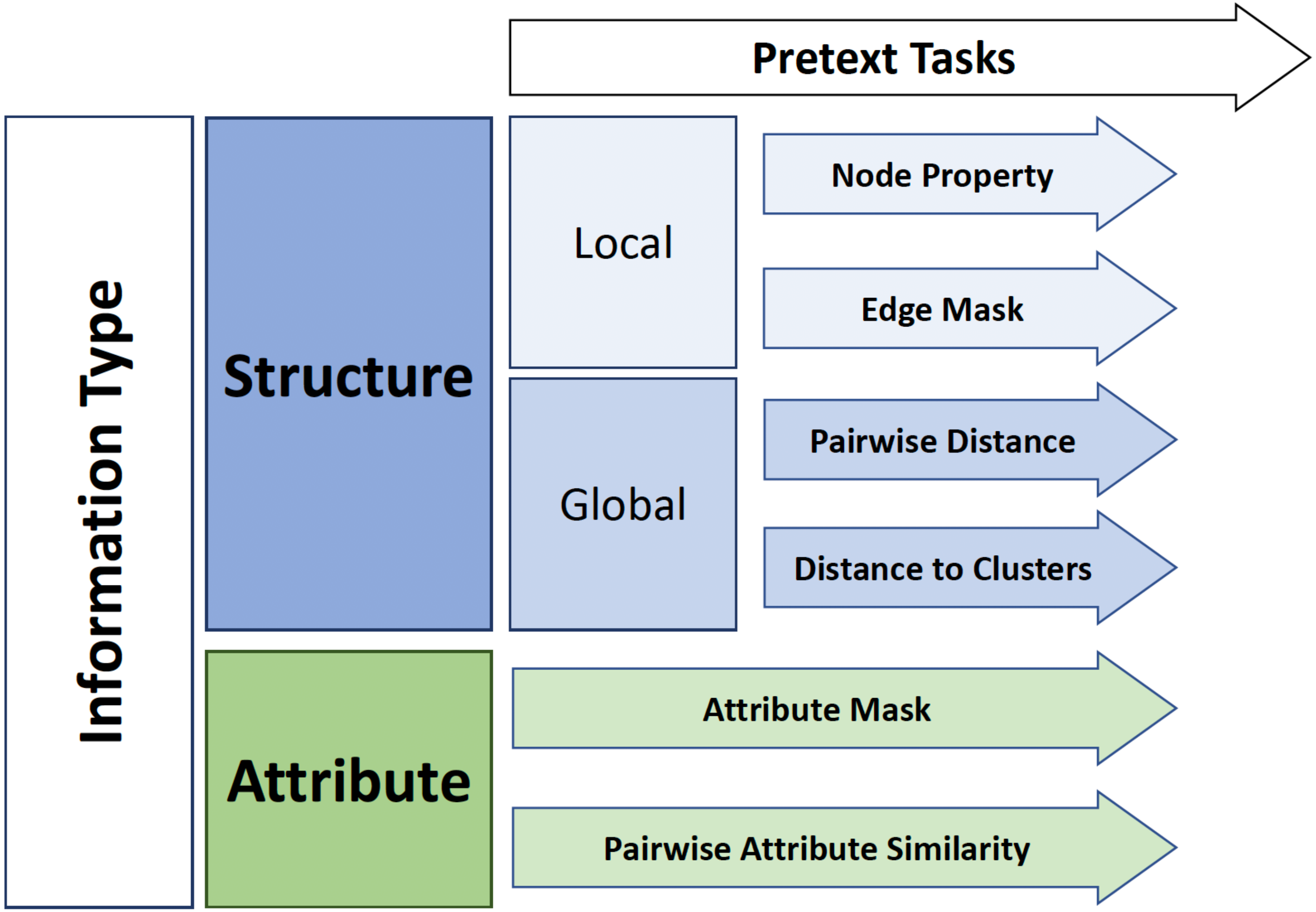
- Pretext task : **Colorization**

Input **Output** **Input** **Output**

- Pretext task : **Rotation**
  - Which one has the correct rotation?

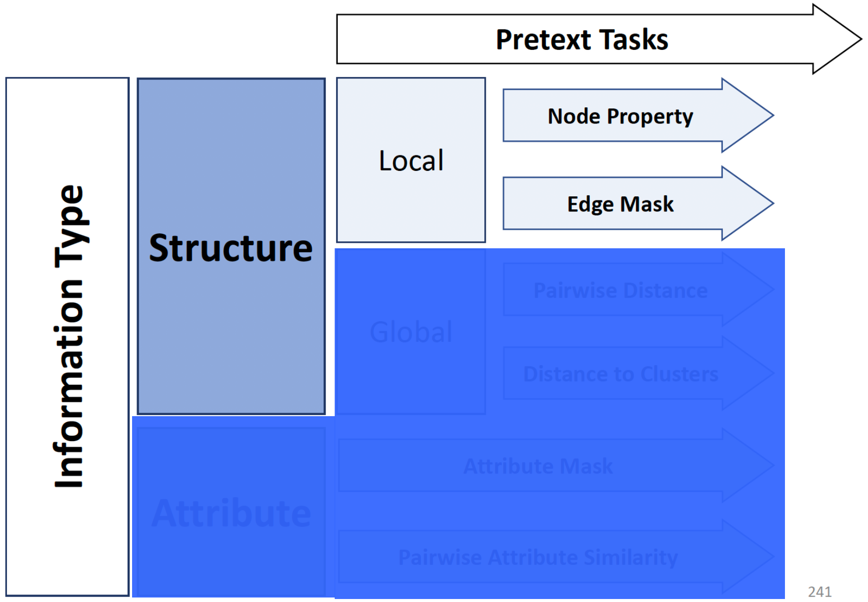
(Figure credit) Self-Supervised Learning, Andrew Zisserman

Examples of Pretext tasks on graphs



(Figure credit) Tutorial in Graph Neural Networks: Models and Applications, AAAI2021

Local Structure-based Pretext Task



- Node property
  - **Goal**: To predict the property for each node in the graph such as their *degree*, *local node importance*, and *local clustering coefficient*.

Predicted degree of node v_i

$$\mathcal{L}_{self}(\theta', \mathbf{A}, \mathbf{X}, \mathcal{D}_U) = \frac{1}{|\mathcal{D}_U|} \sum_{v_i \in \mathcal{D}_U} \left(f_{\theta'}(\mathcal{G})_{v_i} - d_i \right)^2$$

Degree of node v_i

- Edge mask
  - **Goal**: To predict *whether or not there exists a link between a given node pair*

$$\mathcal{L}_{self}(\theta', \mathbf{A}, \mathbf{X}, \mathcal{D}_U) =$$

Cross-entropy loss

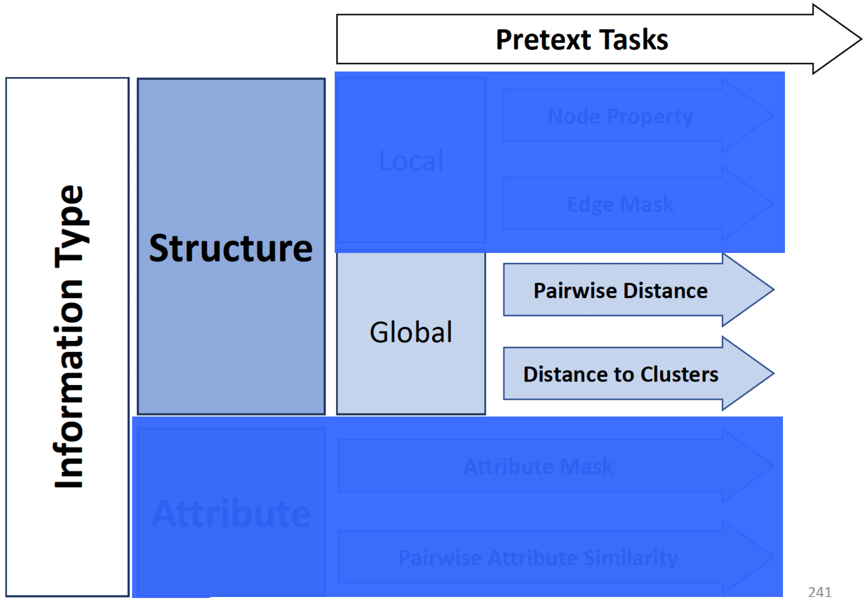
$$\frac{1}{|\mathcal{M}_e|} \sum_{(v_i, v_j) \in \mathcal{M}_e} \ell\left(f_w(|f_{\theta'}(\mathcal{G})_{v_i} - f_{\theta'}(\mathcal{G})_{v_j}|), 1 \right) + \frac{1}{|\overline{\mathcal{M}}_e|} \sum_{(v_i, v_j) \in \overline{\mathcal{M}}_e} \ell\left(f_w(|f_{\theta'}(\mathcal{G})_{v_i} - f_{\theta'}(\mathcal{G})_{v_j}|), 0 \right)$$

Connected edges

Not connected edges

Self-supervised Learning on Graphs: Deep Insights and New Directions, arxiv2020

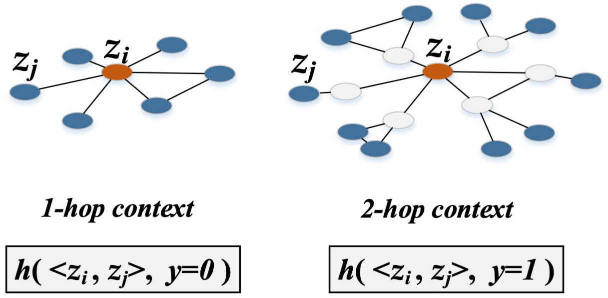
Global Structure-based Pretext Task



- Pairwise distance
  - **Goal**: To predict the distance between different node pair.

$$\mathcal{L}_{self}(\theta', \mathbf{A}, \mathbf{X}, \mathcal{D}_U) = \frac{1}{|\mathcal{S}|} \sum_{(v_i, v_j) \in \mathcal{S}} \ell\left(f_w(|f_{\theta'}(\mathcal{G})_{v_i} - f_{\theta'}(\mathcal{G})_{v_j}|), C_{p_{ij}}\right)$$

Pairwise distance between node v_i and v_j



- Distance2Clusters
  - **Goal**: To predict the distance from the unlabeled nodes to predefined graph clusters
  - Step 1: Apply graph clustering to get k clusters $\{C_1, C_2, \ldots, C_k\}$
  - Step 2: In each cluster C_j, assume the node with the highest degree as the center node

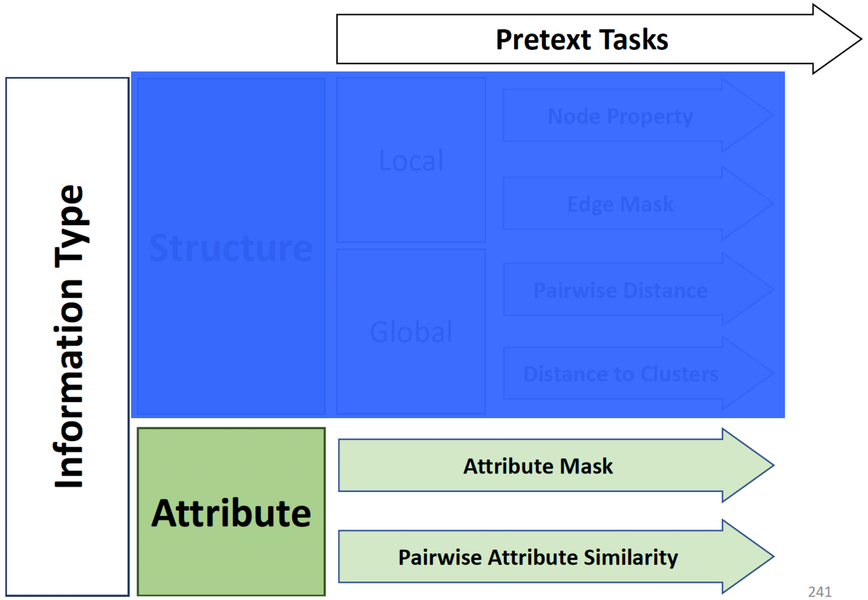
$$\mathcal{L}_{self}(\theta', \mathbf{A}, \mathbf{X}, \mathcal{D}_U) = \frac{1}{|\mathcal{D}_U|} \sum_{v_i \in \mathcal{D}_U} \|f_{\theta'}(\mathcal{G})_{v_i} - \mathbf{d}_i\|^2$$

$$\boldsymbol{d_i} = [d_{i1}, d_{i2}, \ldots, d_{ik}]$$

Distance from node v_i to cluster c_2

Self-Supervised Graph Representation Learning via Global Context Prediction, arxiv2020
Self-supervised Learning on Graphs: Deep Insights and New Directions, arxiv2020

Attribute-based Pretext Task



- Attribute mask
  - **Goal**: To predict the masked attribute
  - Apply PCA to reduce the dimensionality of features

$$\mathcal{L}_{self}(\theta', \mathbf{A}, \mathbf{X}, \mathcal{D}_U) = \frac{1}{|\mathcal{M}_a|} \sum_{v_i \in \mathcal{M}_a} \| f_{\theta'}(\mathcal{G})_{v_i} - \mathbf{x}_i \|^2$$

Feature of node v_i

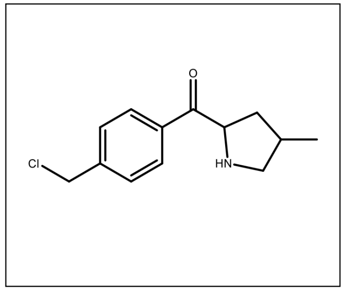
- Pairwise attribute similarity
  - **Goal**: To predict the similarity of pairwise node features

$$\mathcal{L}_{self}(\theta', \mathbf{A}, \mathbf{X}, \mathcal{D}_U) = \frac{1}{|\mathcal{T}|} \sum_{(v_i, v_j) \in \mathcal{T}} \| f_w(|f_{\theta'}(\mathcal{G})_{v_i} - f_{\theta'}(\mathcal{G})_{v_j}|) - s_{ij} \|^2$$

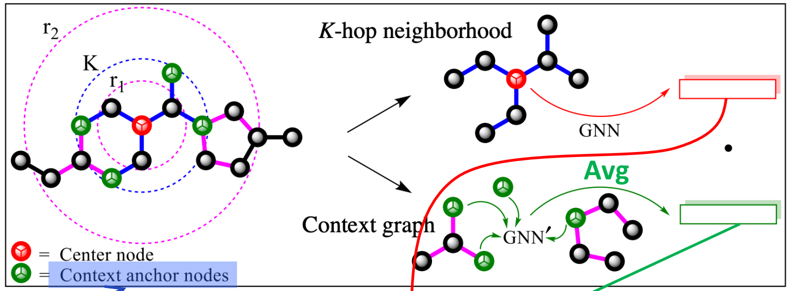
Context prediction

- **Pretext task:** Context prediction

Input graph



(a) Context Prediction



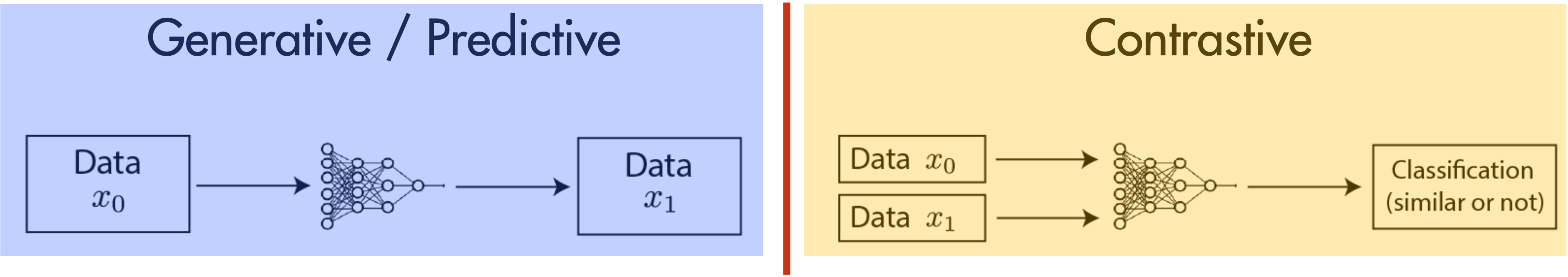
Nodes that are shared between the neighborhood and the context graph

$$\sigma\left(h_v^{(K)\top} c_{v'}^{G'}\right) \approx \mathbf{1}\{v \text{ and } v' \text{ are the same nodes}\},$$

	Chemistry			Biology		
	Non-pre-trained	Pre-trained	Gain	Non-pre-trained	Pre-trained	Gain
GIN	67.0	**74.2**	**+7.2**	64.8 ± 1.0	$\mathbf{74.2 \pm 1.5}$	**+9.4**
GCN	**68.9**	72.2	+3.4	63.2 ± 1.0	70.9 ± 1.7	+7.7
GraphSAGE	68.3	70.3	+2.0	65.7 ± 1.2	68.5 ± 1.5	+2.8
GAT	66.8	60.3	-6.5	$\mathbf{68.2 \pm 1.1}$	67.8 ± 3.6	-0.4

Taxonomy of Self-Supervised Learning

So far | **From now on…**

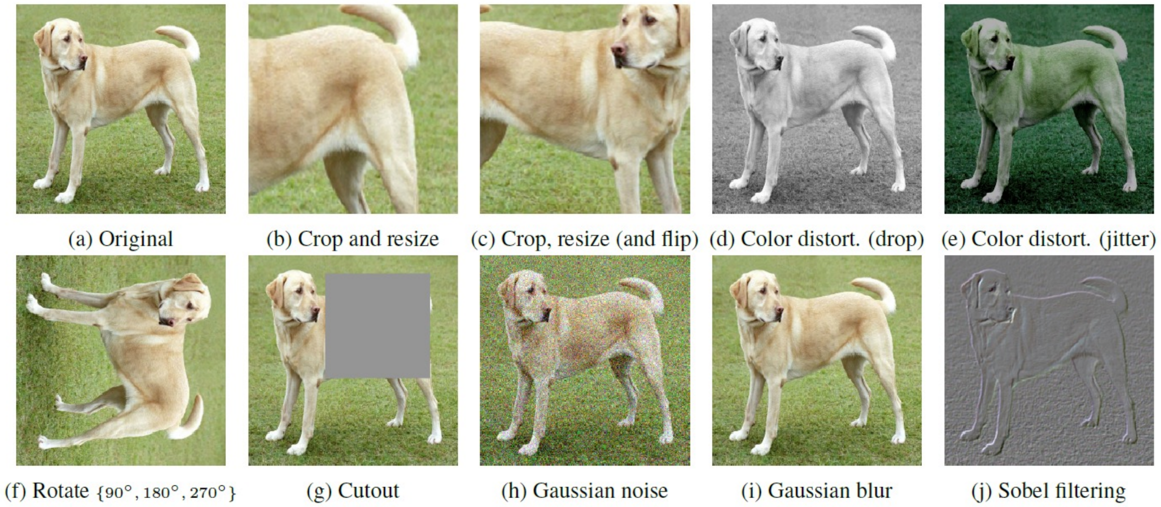


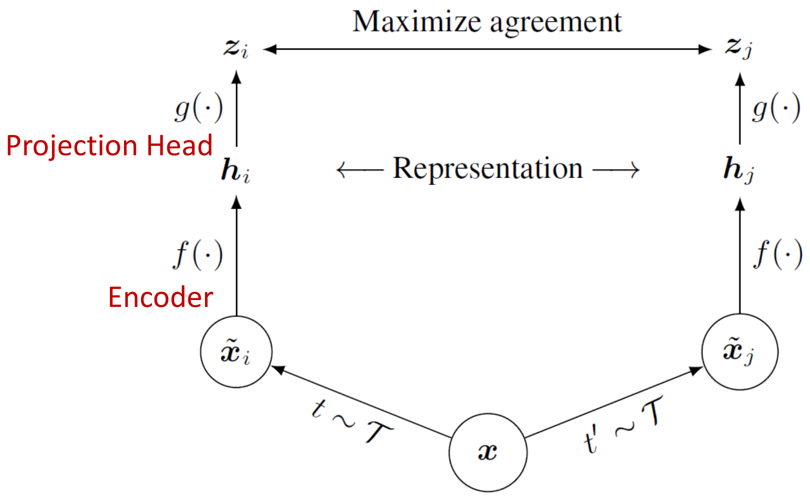
- **Contrastive learning**
  - **Given:** $X = \{x, \ x^+, \ x_1^-, \ldots, x_{N-1}^-\}$; Similarity function $s(\cdot)$ (e.g., cosine similarity)
  - **Goal:** $s\big(f(x), f(x^+)\big) > s\big(f(x), f(x^-)\big)$
  - **Contrastive/InfoNCE Loss**

$$\mathcal{L}_N = -\mathbb{E}_X\left[\log \frac{\exp\Big(s\big(f(x), f(x^+)\big)\Big)}{\exp\Big(s\big(f(x), f(x^+)\big)\Big) + \sum_{j=1}^{N-1} \exp\Big(s\big(f(x), f(x_j^-)\big)\Big)}\right]$$

(Figure credit) https://ankeshanand.com/blog/2020/01/26/contrative-self-supervised-learning.html

The Contrastive Learning Paradigm





- Algorithm
  - 1) Sample mini batch of N examples
  - 2) Create $2N$ data points via Data Augmentation
  - 3) Given a positive pair, treat other $2(N-1)$ points as negative examples
  - → **Instance Discrimination!**

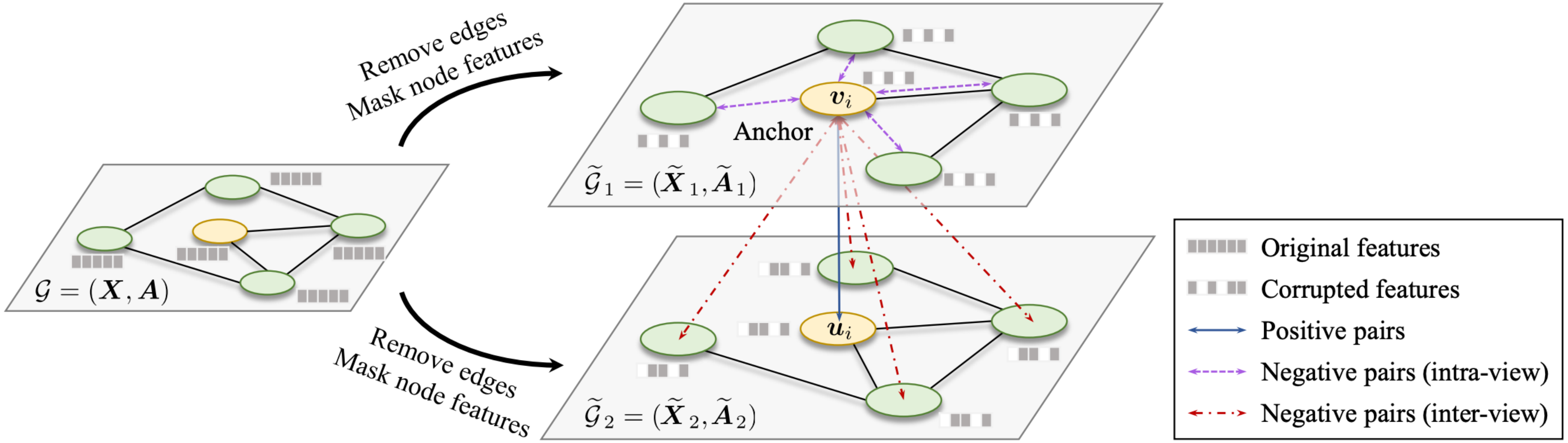
$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

Reduce: Dist. between representations of different augmented views of the same image (Positive)
Increase: Dist. between representations of augmented views from different images (Negative)

A simple framework for contrastive learning of visual representations, ICML 2020

Deep Graph Contrastive Representation Learning (GRACE)

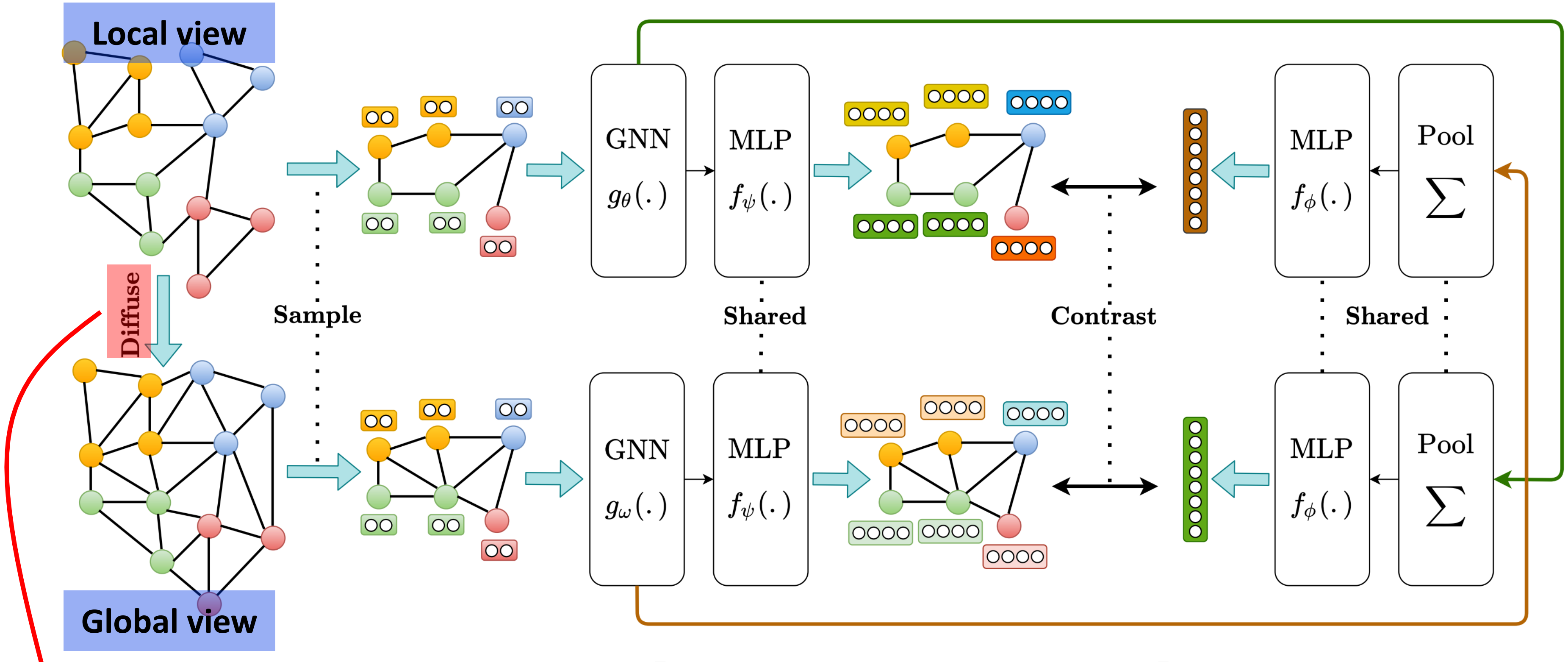
- **Pull** the representation of the same node in the two augmented graphs
- **Push** apart representations of every other node



$$\ell(\boldsymbol{u}_i, \boldsymbol{v}_i) = \log \frac{e^{\theta(\boldsymbol{u}_i, \boldsymbol{v}_i)/\tau}}{\underbrace{e^{\theta(\boldsymbol{u}_i, \boldsymbol{v}_i)/\tau}}_{\text{the positive pair}} + \underbrace{\sum_{k=1}^{N} \mathbb{1}_{[k \neq i]} e^{\theta(\boldsymbol{u}_i, \boldsymbol{v}_k)/\tau}}_{\text{inter-view negative pairs}} + \underbrace{\sum_{k=1}^{N} \mathbb{1}_{[k \neq i]} e^{\theta(\boldsymbol{u}_i, \boldsymbol{u}_k)/\tau}}_{\text{intra-view negative pairs}}},$$

Contrastive Multi-View Representation Learning on Graphs

- **Idea**: Contrast encodings from first-order neighbors and a general graph diffusion
  - Maximize MI between node representations of one view and graph representation of another view and vice versa

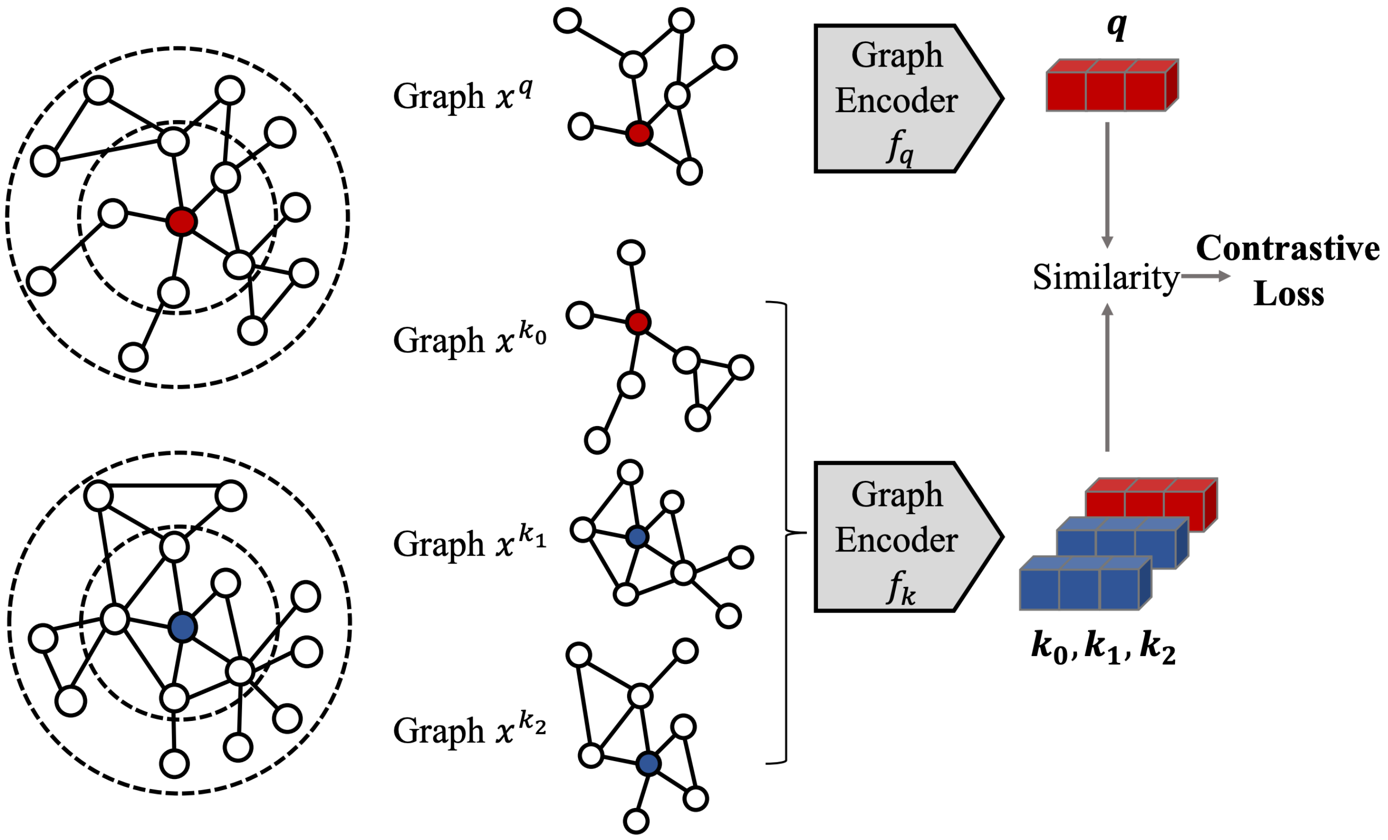


$$\max_{\theta,\omega,\phi,\psi} \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \left[\frac{1}{|g|} \sum_{i=1}^{|g|} \left[\mathrm{MI}\left(\vec{h}_i^{\alpha}, \vec{h}_g^{\beta}\right) + \mathrm{MI}\left(\vec{h}_i^{\beta}, \vec{h}_g^{\alpha}\right) \right] \right]$$

Structural Augmentation

GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training

- **Idea**: **Subgraph instance discrimination** in and across networks



$$\mathcal{L} = -\log \frac{\exp\left(\boldsymbol{q}^\top \boldsymbol{k}_+ / \tau\right)}{\sum_{i=0}^{K} \exp\left(\boldsymbol{q}^\top \boldsymbol{k}_i / \tau\right)}$$

- Query instance x^q
- Key instances $\{x^{k_0}, x^{k_1}, x^{k_2}\}$
- Embedding
  - $\boldsymbol{q}$ (embedding of x^q)
    - i.e., $\boldsymbol{q} = f(x^q)$
  - $\boldsymbol{k_0}, \boldsymbol{k_1}, \boldsymbol{k_2}$ (embedding of $\{x^{k_0}, x^{k_1}, x^{k_2}\}$)
    - i.e., $\boldsymbol{k_i} = f(x^{k_i})$

Shortcomings of Contrastive Methods

- 1) Requires negative samples → **Sampling bias**
  - Treat different image as negative even if they share the semantics
- 2) Requires careful augmentation

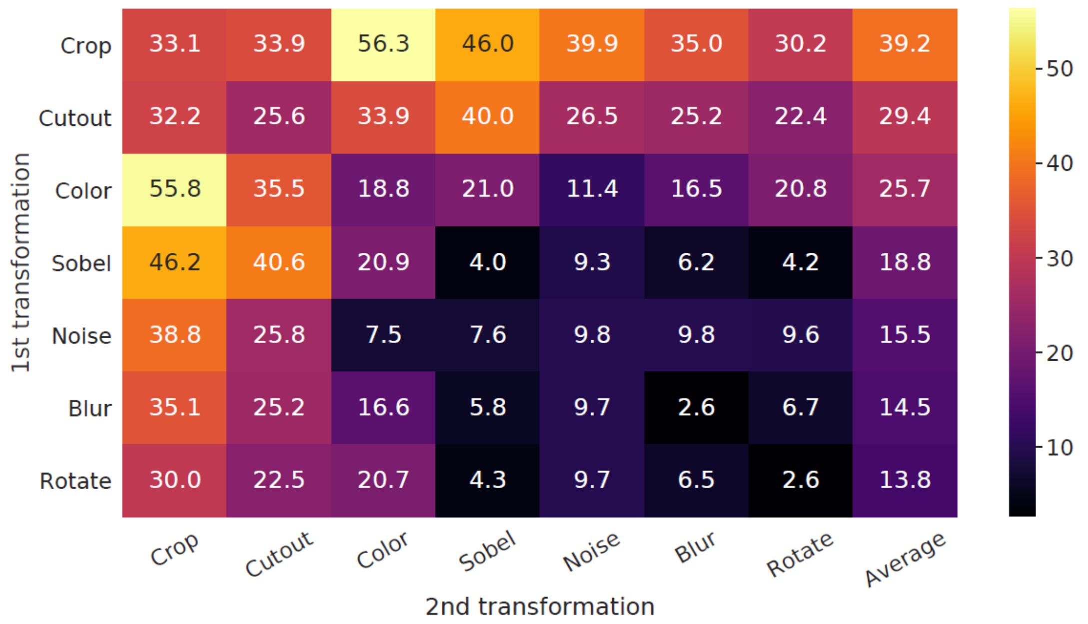
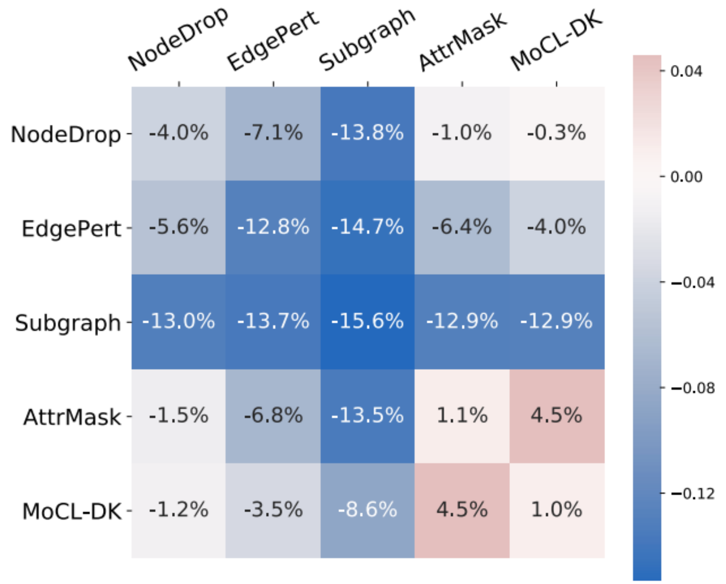


Image classification



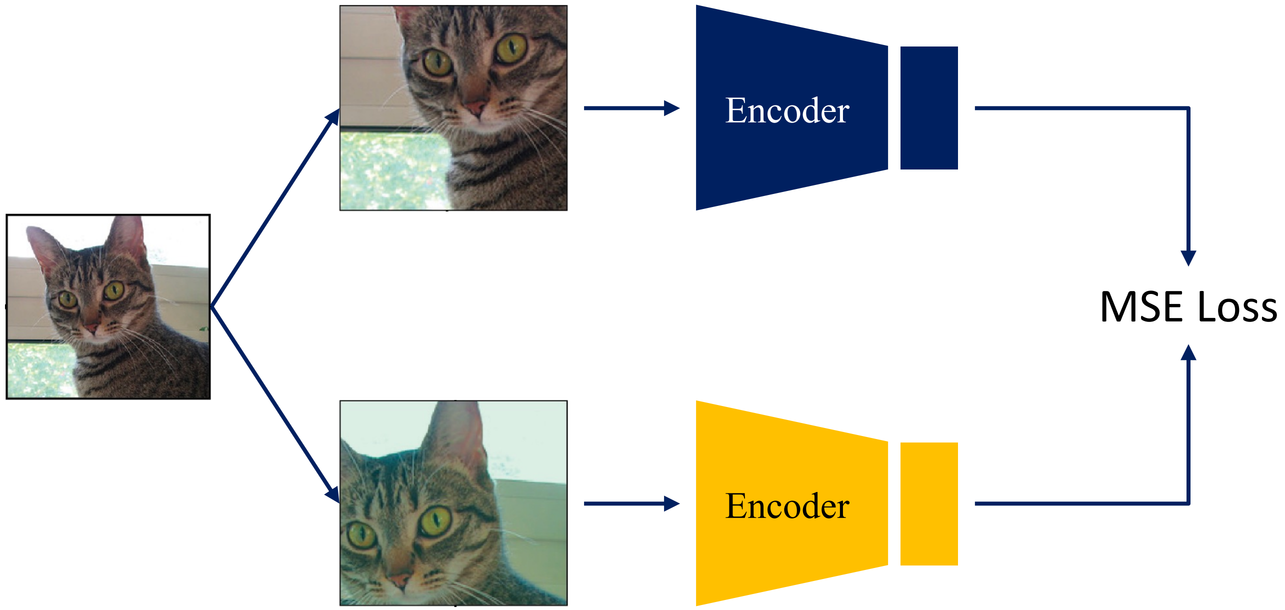
Graph classification

A simple framework for contrastive learning of visual representations, ICML 2020

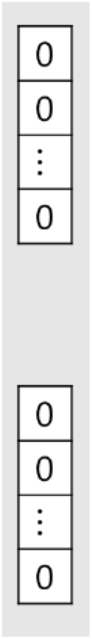
MoCL: Data-driven Molecular Fingerprint via Knowledge-aware Contrastive Learning from Molecular Graph, KDD 2020

Can We Remove Negative Sampling?

- **Cross-view prediction framework without negative samples?**
  - Learn representations by predicting different views of the same image from one another
- **Problem:** Predicting directly in representation space can lead to collapsed representation
  - Contrastive methods circumvents this by reformulating the prediction problem discrimination task (Pos ↔ Neg)



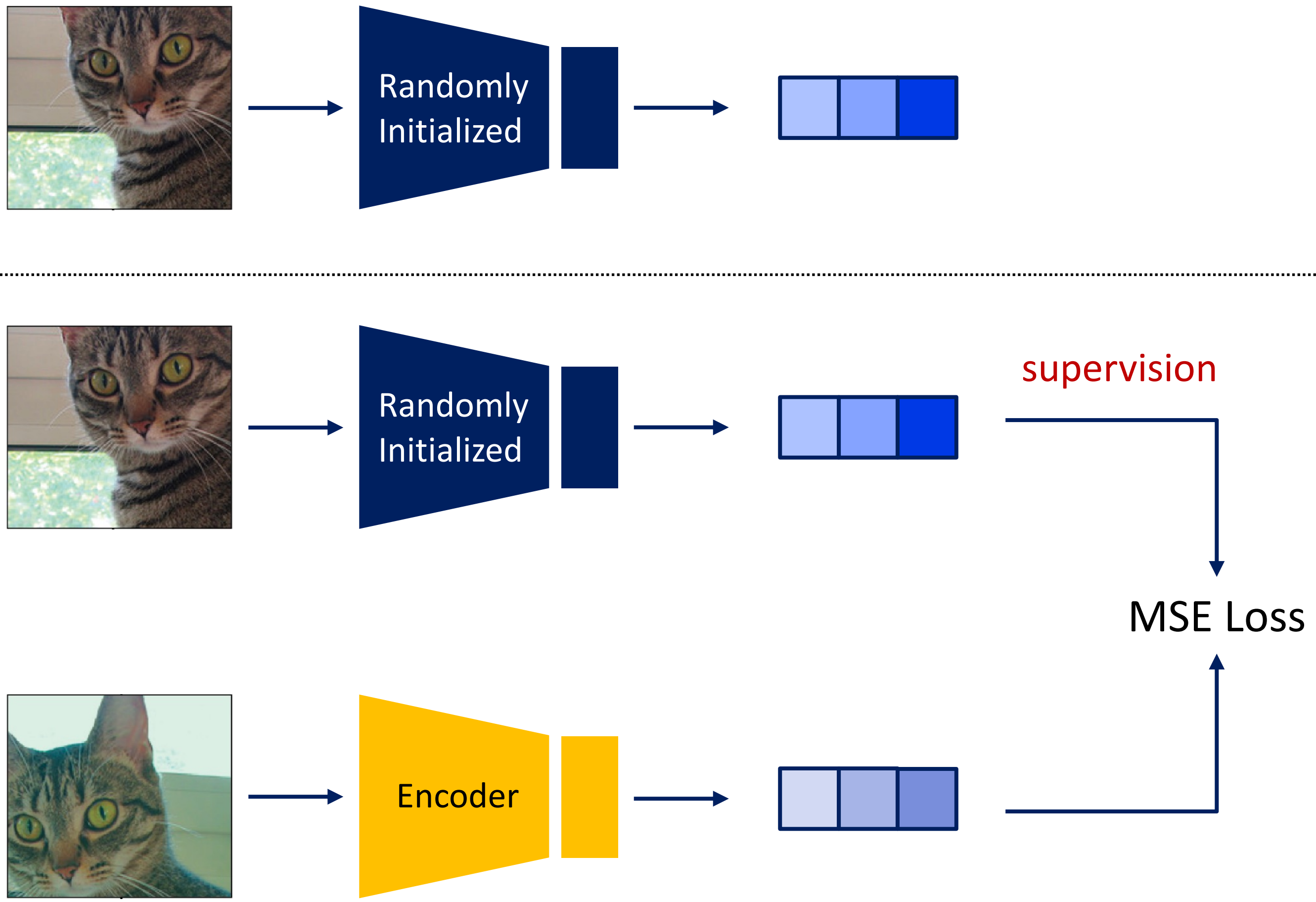
Cross-view prediction framework



Trivial Solution → Constant Vector

Straightforward Solution to Overcome Collapsed Representation

- **Use a fixed randomly initialized network to produce targets for our predictions**



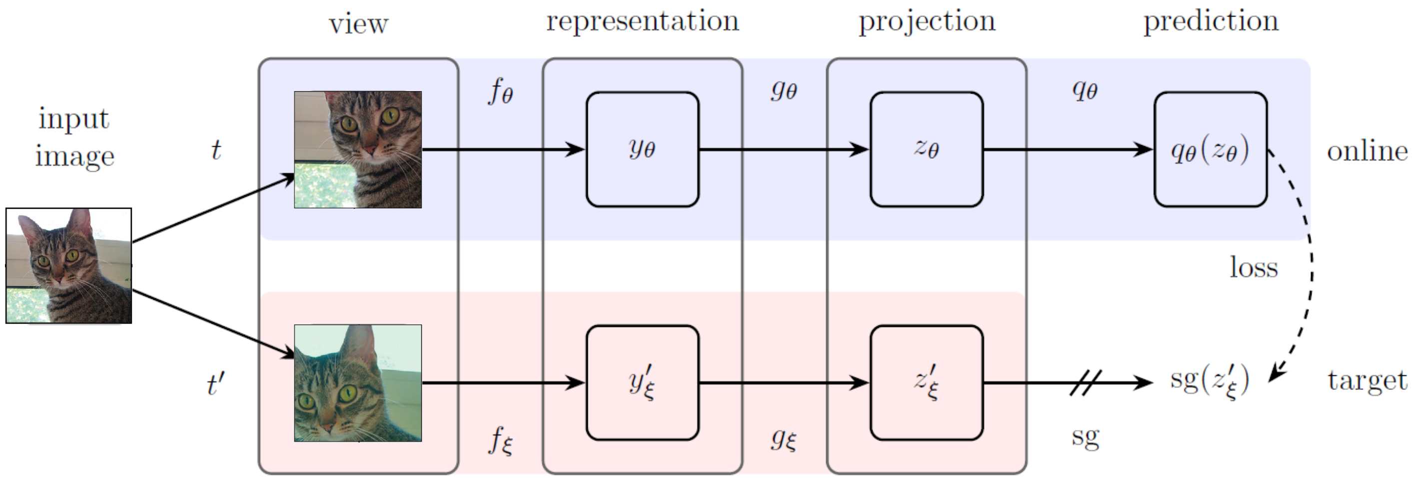
Top-1 Accuracy → 1.4 %

Top-1 Accuracy → 18.8 %
even with random supervision

Core motivation of
non-contrastive methods!

Bootstrap Your Own Latent (BYOL)

- BYOL uses two neural networks to learn: 1) online and 2) target networks
- From a given target representation, we train a new online representation by predicting the target representation



Only online parameters are updated to reduce the loss, while the target parameters follow a different objective

→ Avoid Collapsed Representation

1) Online Network Update → Gradient-based update

$$\mathcal{L}_{\theta,\xi} \triangleq \left\|\overline{q_\theta}(z_\theta) - \bar{z}'_\xi\right\|_2^2, \quad \mathcal{L}^{\mathrm{BYOL}}_{\theta,\xi} = \mathcal{L}_{\theta,\xi} + \widetilde{\mathcal{L}}_{\theta,\xi} \text{ (Symmetrize)}$$

$$\theta \leftarrow \mathrm{optimizer}\left(\theta, \nabla_\theta \mathcal{L}^{\mathrm{BYOL}}_{\theta,\xi}, \eta\right)$$

Online network

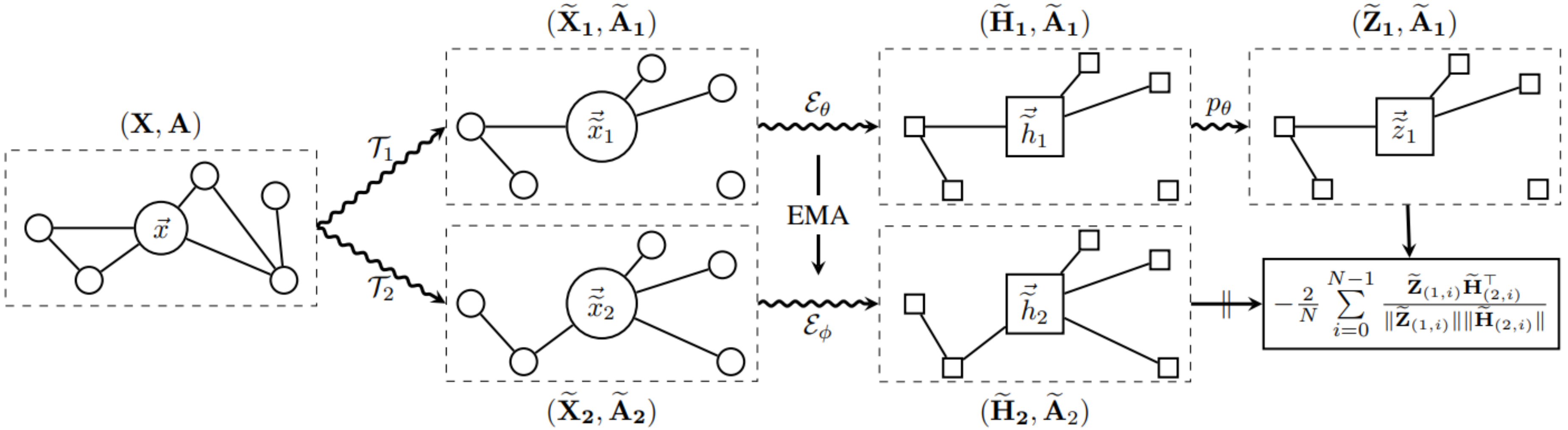
2) Target Network Update → Exponential Moving Average

$$\xi \leftarrow \tau\xi + (1-\tau)\theta$$

Target network Online network

Large-Scale Representation Learning on Graphs via Bootstrapping

- BGRL is a simple extension of BYOL to graph domain
- Representations are directly learned by **predicting the representation of each node in one view of the graph, using the representation of the same node in another view**



- Graph Augmentation → Node attribute masking + Edge masking

Shortcomings of Contrastive Methods

- 1) Requires negative samples → **Sampling bias**
  - Treat different image as negative even if they share the semantics
- 2) Requires careful augmentation

Research Question

Is augmentation appropriate for graph-structured data?

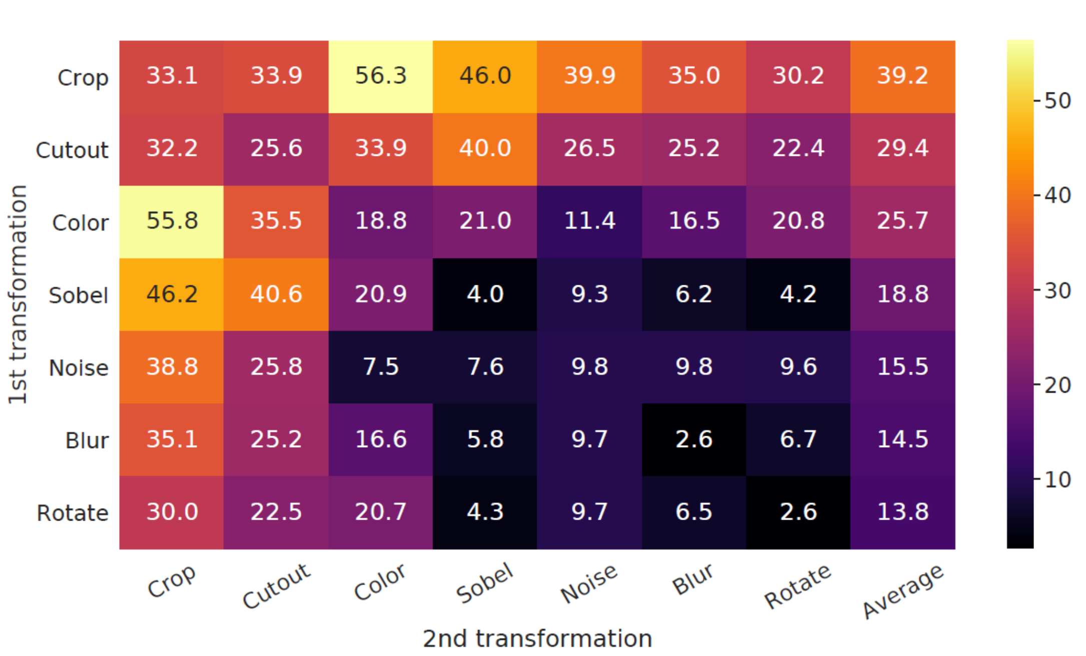
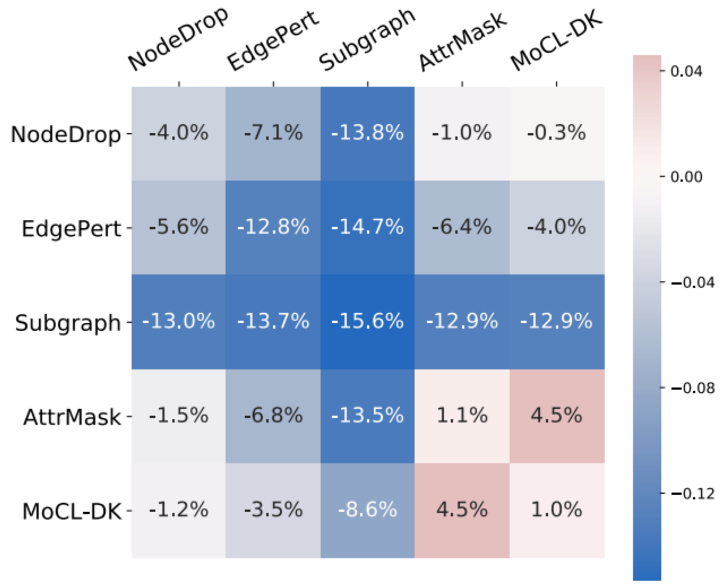


Image classification



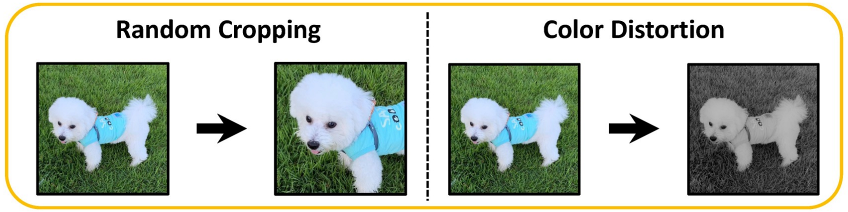
Graph classification

A simple framework for contrastive learning of visual representations, ICML 2020

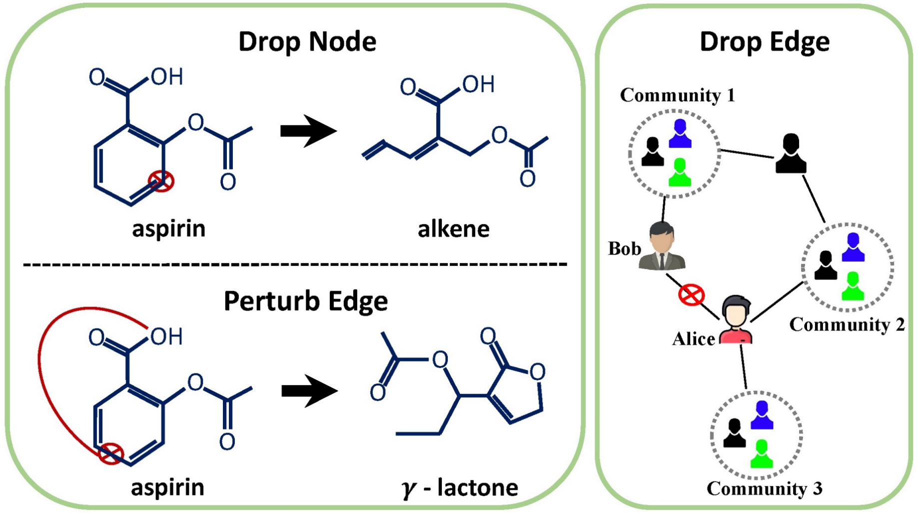
MoCL: Data-driven Molecular Fingerprint via Knowledge-aware Contrastive Learning from Molecular Graph, KDD 2020

Motivation: Is Augmentation Appropriate for Graph-structured Data?

- Image's underlying semantic is hardly changed after augmentation



- However in the case of graphs, we cannot ascertain whether the augmented graph would be positively related to original graph



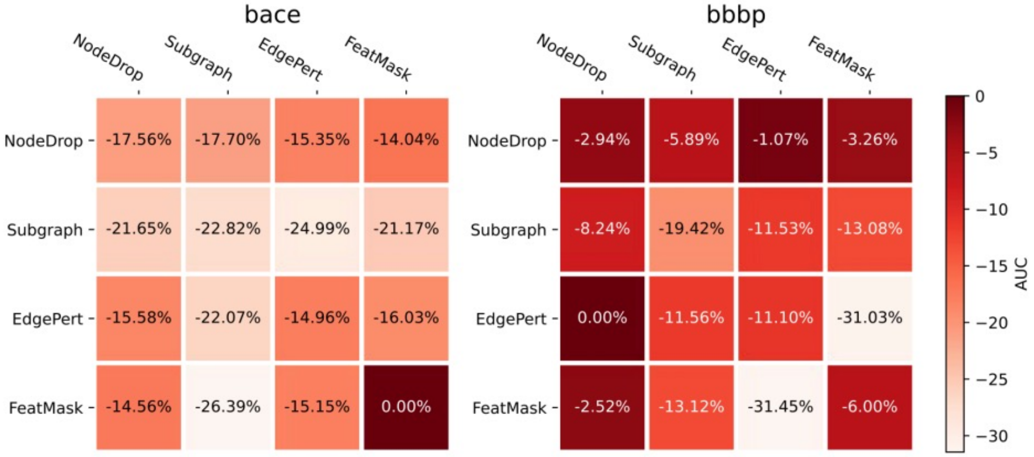
Because graphs contain not only the semantic but also the **structural information**

Motivation: Is Augmentation Appropriate for Graph-structured Data?

- Performance sensitivity according to hyperparameters for augmentations

		Comp.	Photo	CS	Physics
Node Classi.	BGRL	-4.00%	-1.06%	-0.20%	-0.69%
	GCA	-19.18%	-5.48%	-0.27%	OOM
Node Clust.	BGRL	-11.57%	-13.30%	-0.78%	-6.46%
	GCA	-26.28%	-23.27%	-1.64%	OOM

Node-level task



NodeDrop : Node Dropping / Subgraph : Subgraph Extraction / EdgePert : Edge Perturbation / FeatMask : Feature Masking

Graph-level task

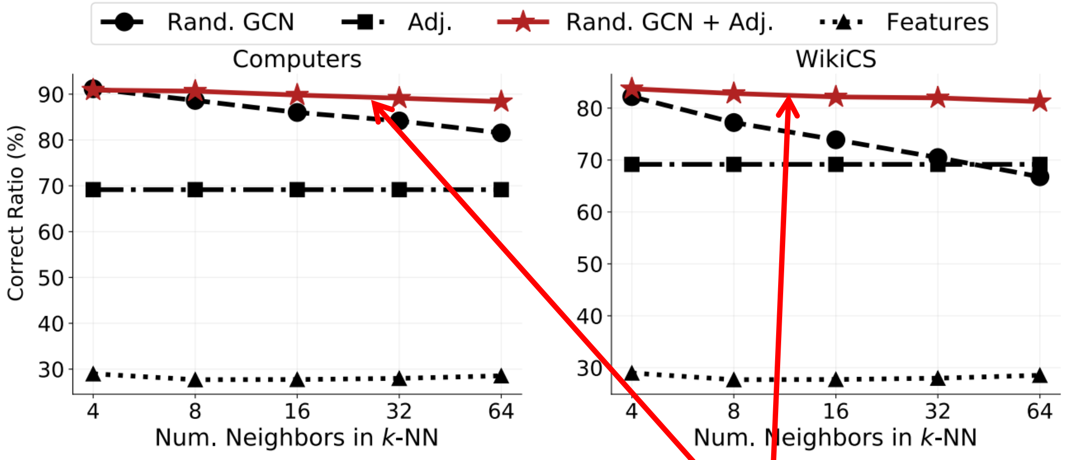
- The quality of the learned representations relies on the **choice of augmentation scheme**
  - Performance on various downstream tasks varies greatly according to the choice of augmentation hyperparameters

We need more stable and general framework for generating alternative view of the original graph **without relying on augmentation**

\+ remove negative sampling process

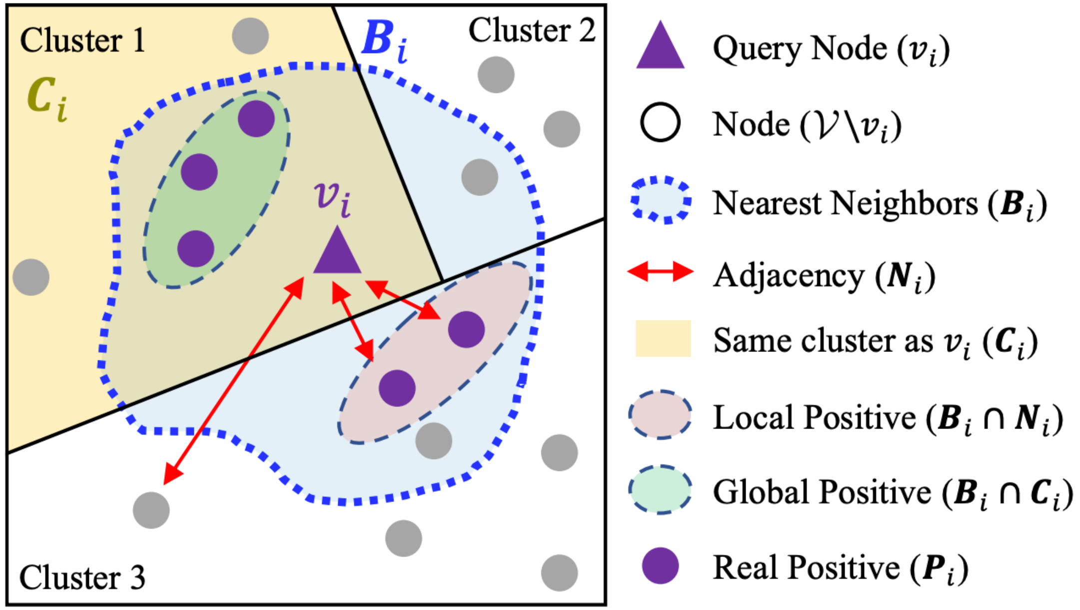
Augmentation-Free Graph Representation Learning

- Instead of creating two arbitrarily augmented views of graph,
  - Use the original graph per se as one view, and generate another view by discovering nodes that can be serve as pos itive samples via k-nearest neighbor search in embedding space.
- However, naively selected positive samples with k-NN includes false positives
  - More than 10% of false negatives



We need to filter out false positives regarding local and global perspective!

Capturing Local and Global Semantics



- Obtain real positives for v_i

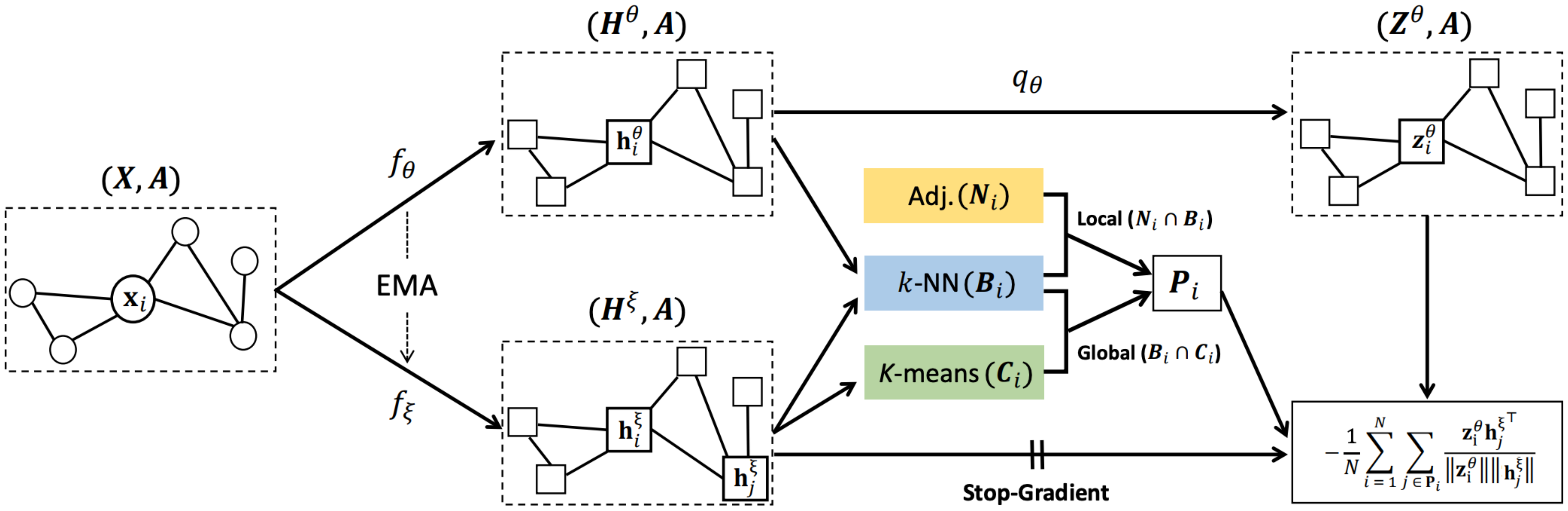
$$\mathbf{P}_i = (\mathbf{B}_i \cap \mathbf{N}_i) \cup (\mathbf{B}_i \cap \mathbf{C}_i)$$

- Minimize the cosine distance between query and real positives P_i

$$\mathcal{L}_{\theta,\xi} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{v_j \in \mathbf{P}_i} \frac{\mathbf{z}_i^{\theta} \mathbf{h}_j^{\xi \top}}{\left\|\mathbf{z}_i^{\theta}\right\| \left\|\mathbf{h}_j^{\xi}\right\|}$$

- B_i: Set of k-NNs of query v_i
- N_i: Set of adjacent nodes of query v_i
- C_i: Set of nodes that are in the same cluster with query v_i

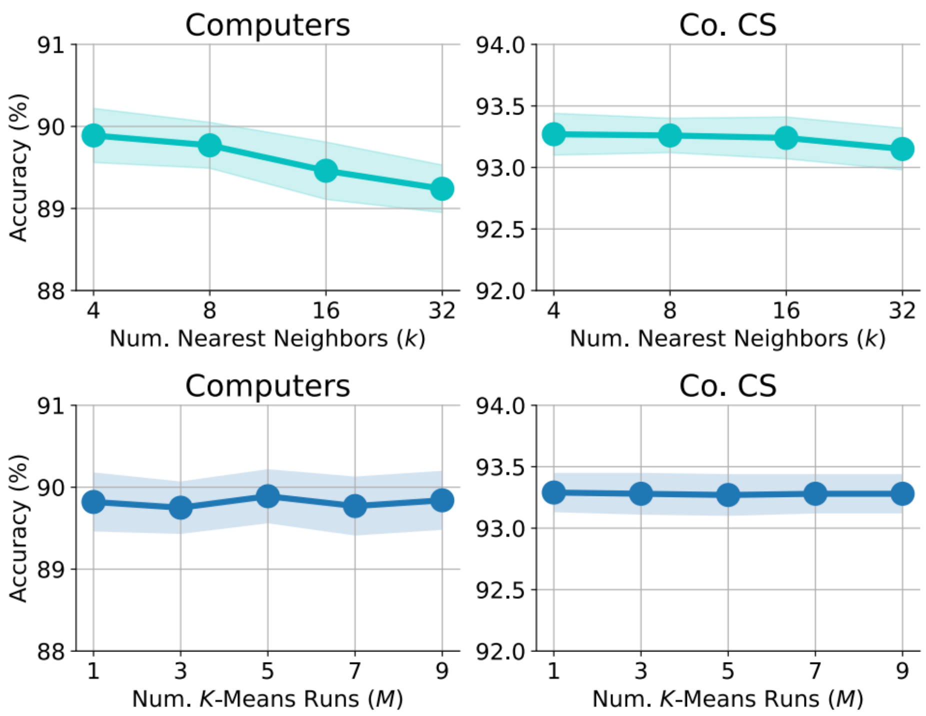
Overall Architecture of AFGRL



Experiments

- **Task**: Node classification

	WikiCS	Computers	Photo	Co.CS	Co.Physics
Sup. GCN	77.19 ± 0.12	86.51 ± 0.54	92.42 ± 0.22	93.03 ± 0.31	95.65 ± 0.16
Raw feats.	71.98 ± 0.00	73.81 ± 0.00	78.53 ± 0.00	90.37 ± 0.00	93.58 ± 0.00
node2vec	71.79 ± 0.05	84.39 ± 0.08	89.67 ± 0.12	85.08 ± 0.03	91.19 ± 0.04
DeepWalk	74.35 ± 0.06	85.68 ± 0.06	89.44 ± 0.11	84.61 ± 0.22	91.77 ± 0.15
DW + feats.	77.21 ± 0.03	86.28 ± 0.07	90.05 ± 0.08	87.70 ± 0.04	94.90 ± 0.09
DGI	75.35 ± 0.14	83.95 ± 0.47	91.61 ± 0.22	92.15 ± 0.63	94.51 ± 0.52
GMI	74.85 ± 0.08	82.21 ± 0.31	90.68 ± 0.17	OOM	OOM
MVGRL	77.52 ± 0.08	87.52 ± 0.11	91.74 ± 0.07	92.11 ± 0.12	95.33 ± 0.03
GRACE	**77.97 ± 0.63**	86.50 ± 0.33	92.46 ± 0.18	92.17 ± 0.04	OOM
GCA	77.94 ± 0.67	87.32 ± 0.50	92.39 ± 0.33	92.84 ± 0.15	OOM
BGRL	76.86 ± 0.74	89.69 ± 0.37	93.07 ± 0.38	92.59 ± 0.14	95.48 ± 0.08
AFGRL	77.62 ± 0.49	**89.88 ± 0.33**	**93.22 ± 0.28**	**93.27 ± 0.17**	**95.69 ± 0.10**

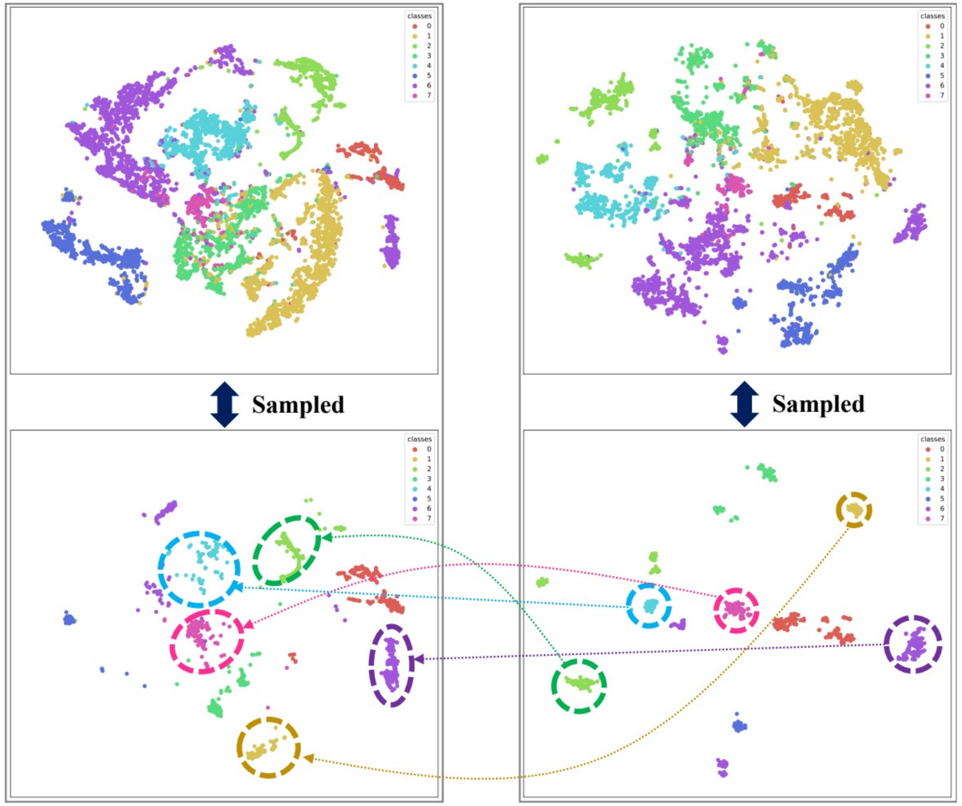


AFGRL outperforms SOTA baselines

AFGRL is stable over hyperparameters
→ Can be easily trained compared with other augmentation-based methods.

Experiments

- **Task**: T-SNE visualization



(a) GCA (b) AFGRL

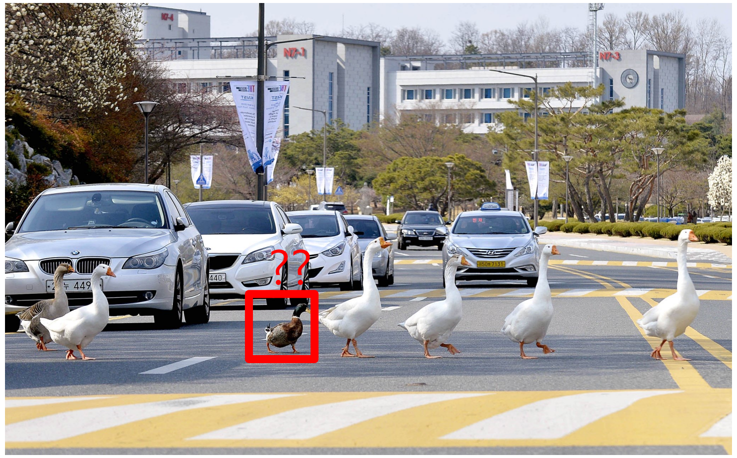
Nodes are more tightly grouped in AFGRL
→ Captures fine-grained class information

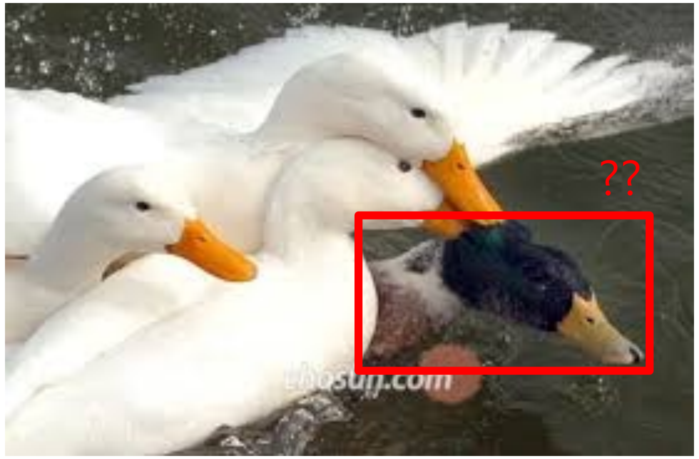
This talk

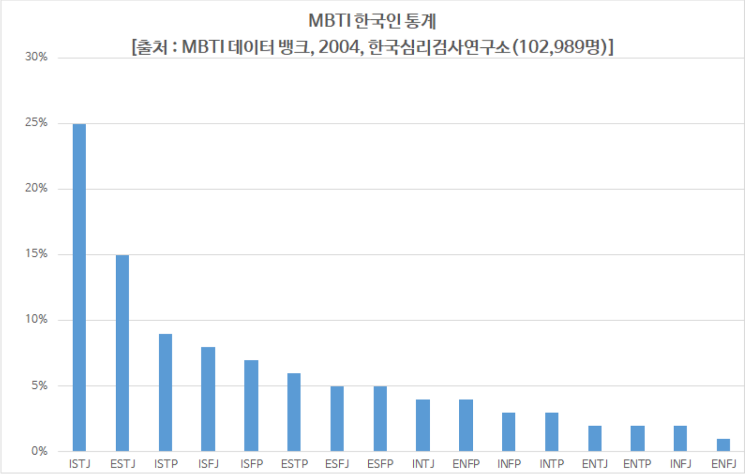
- How to learn graph representation in **various types of graphs**?
  - ~~GNNs for Homogeneous Graph~~
  - GNNs for Multi-aspect Graph
  - GNNs for Multi-relational Graph
- How to effectively **train GNNs**?
  - Self-supervised learning
  - Alleviating Long-tail problem
  - Robustness of GNN

Motivation: Long-tail (Class imbalance)

Purpose of ML: “***To generalize well*** ”

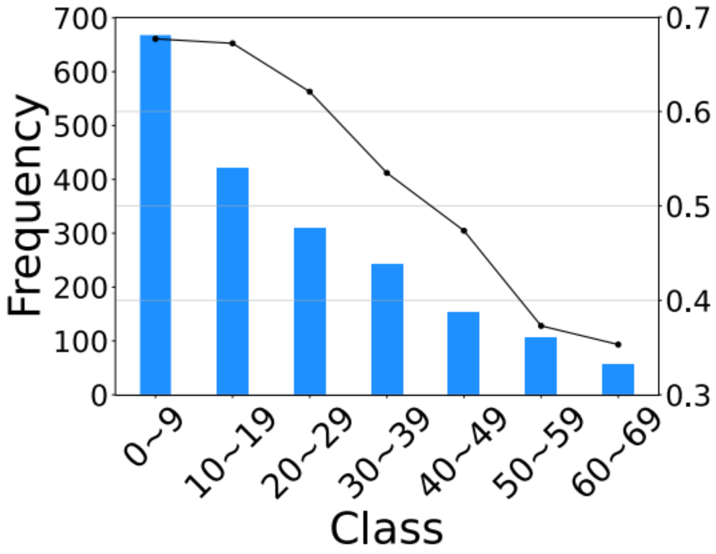


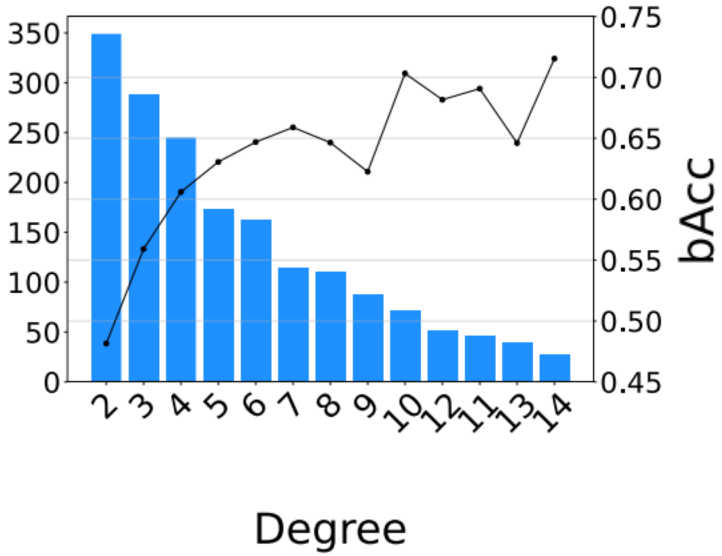




Motivation: Long-tail in GNNs

- Graphs exhibit long-tail problems in two perspectives: 1) Class long-tailedness, 2) Degree long-tailedness





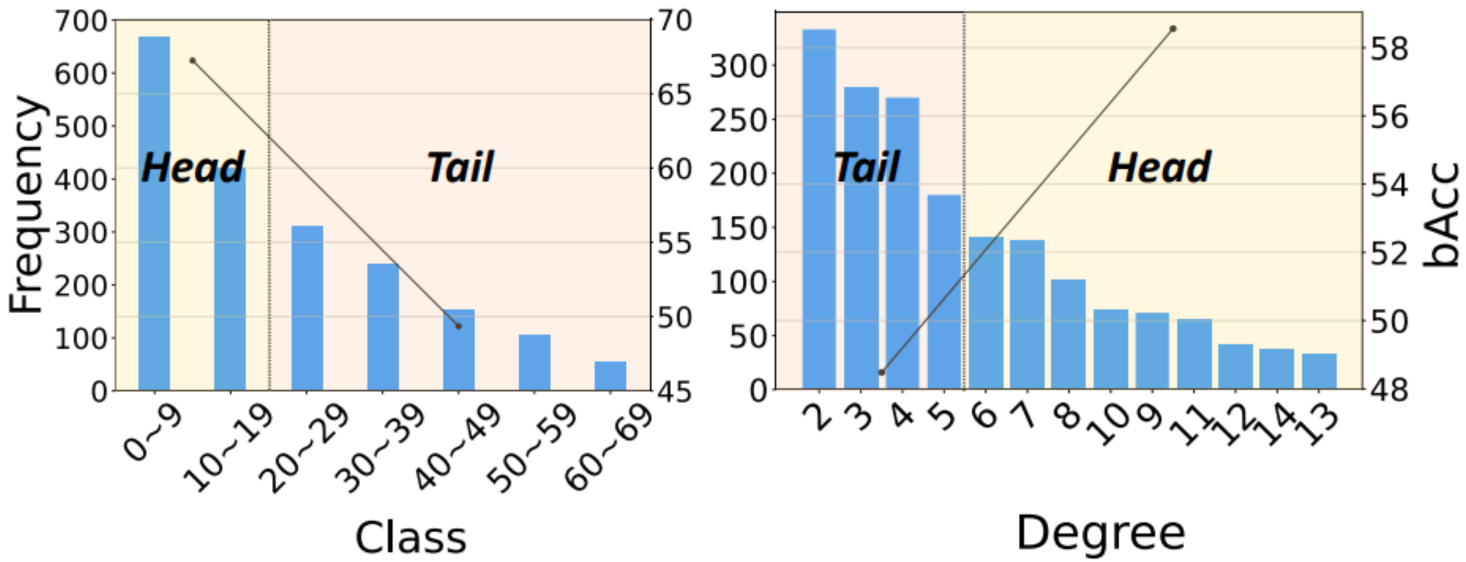
Balanced Accuracy (bAcc) = $\frac{(\text{True Positive Rate} + \text{True Negative Rate})}{2}$

Predict

Actual	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Motivation: Long-tail in GNNs

- Graphs exhibit long-tail problems in two perspectives: 1) Class long-tailedness, 2) Degree long-tailedness



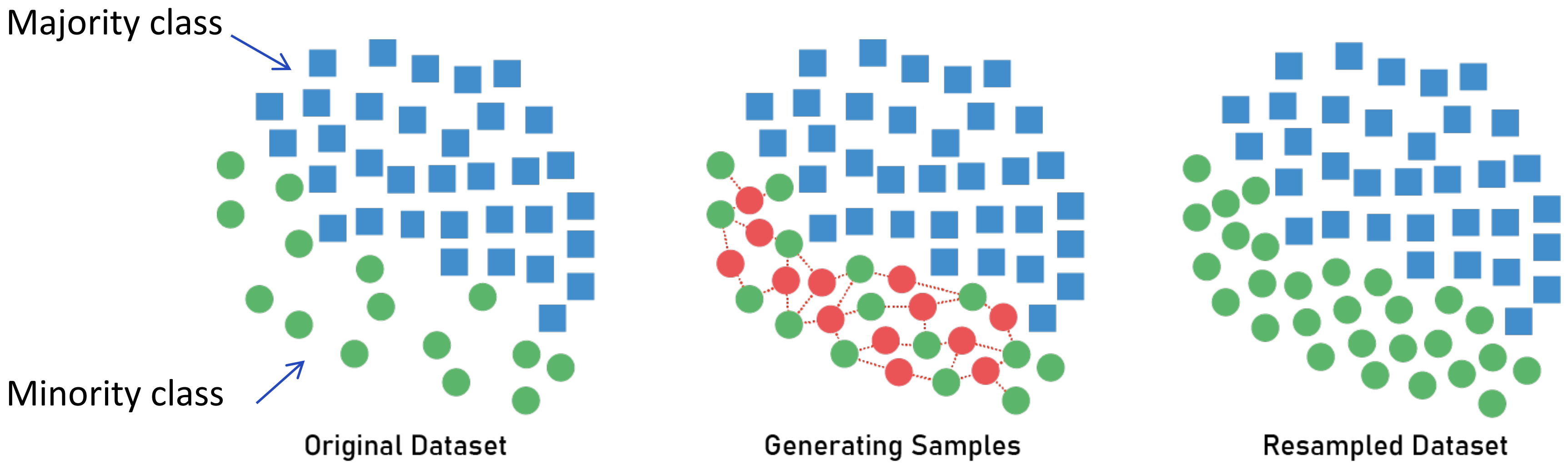
$$\textbf{Balanced Accuracy (bAcc)} = \frac{(\text{True Positive Rate} + \text{True Negative Rate})}{2}$$

Predict

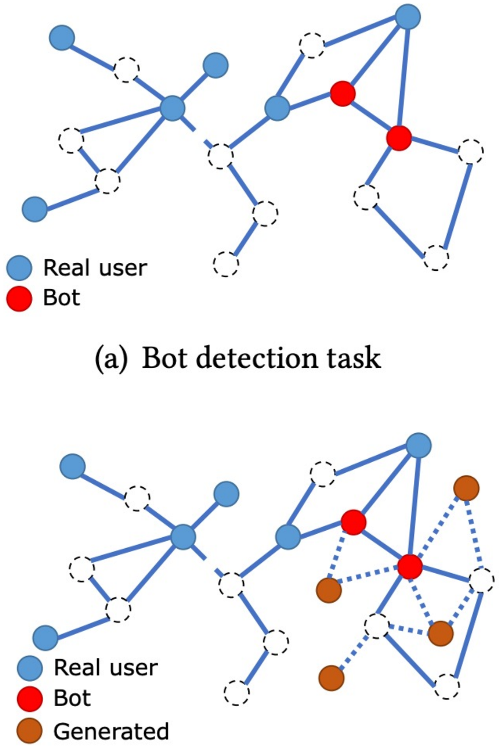
Actual	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Long-Tailedness: Class Perspective
Imbalanced Node Classification on Graphs with Graph Neural Networks (GraphSMOTE)

- **Motivation:** Extend a well-known imbalanced learning technique (SMOTE) to graph domain



(Figure credit) https://bit.ly/3S0YsoJ



(a) Bot detection task

(b) After over-sampling

Vanilla SMOTE fail to provide relation information for newly synthesized samples

GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks. WSDM 2021.

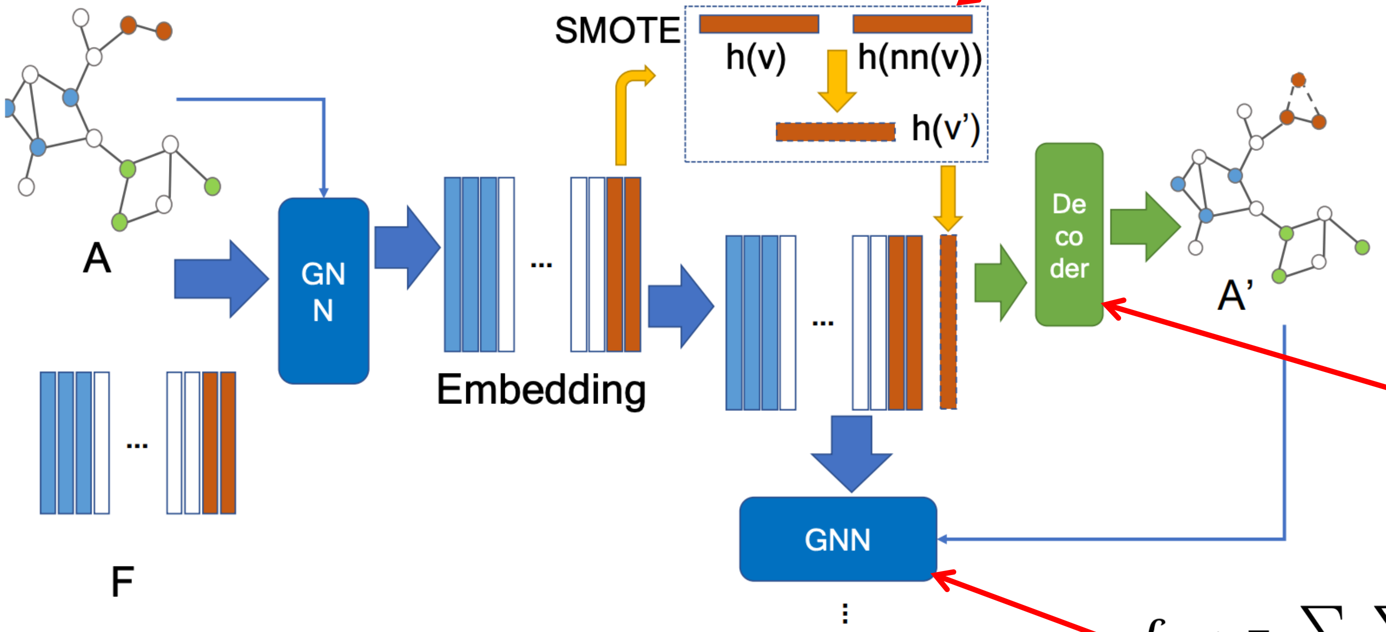
Long-Tailedness: Class Perspective

Imbalanced Node Classification on Graphs with Graph Neural Networks (GraphSMOTE)

- **Main idea**: Train edge generator based on existing nodes and use them for synthetic nodes:

$$nn(v) = \underset{u}{\operatorname{argmin}} \|\mathbf{h}_u^1 - \mathbf{h}_v^1\|, \quad \text{s.t.} \quad \mathbf{Y}_u = \mathbf{Y}_v$$

$$\mathbf{h}_{v'}^1 = (1-\delta) \cdot \mathbf{h}_v^1 + \delta \cdot \mathbf{h}_{nn(v)}^1$$



For existing nodes,

$$\mathbf{E}_{v,u} = softmax(\sigma(\mathbf{h}_v^1 \cdot \mathbf{S} \cdot \mathbf{h}_u^1)). \quad \mathcal{L}_{edge} = \|\mathbf{E} - \mathbf{A}\|_F^2,$$

For synthetic nodes,

$$\tilde{\mathbf{A}}[v', u] = \begin{cases} 1, & \text{if } \mathbf{E}_{v',u} > \eta \\ 0, & \text{otherwise.} \end{cases}$$

$$\mathcal{L}_{node} = \sum_{u \in \tilde{\mathcal{V}}_L} \sum_c (1(Y_u == c) \cdot log(\mathbf{P}_v[c]))$$

Final Loss

$$\min_{\theta,\phi,\varphi} \mathcal{L}_{node} + \lambda \cdot \mathcal{L}_{edge},$$

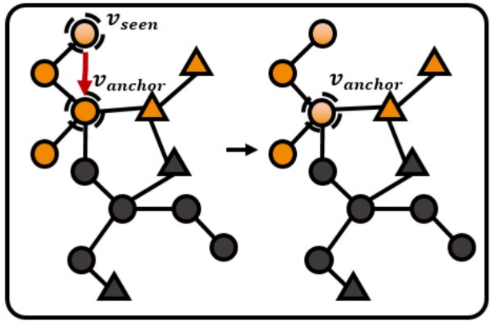
GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks. WSDM 2021.

Long-Tailedness: Class Perspective

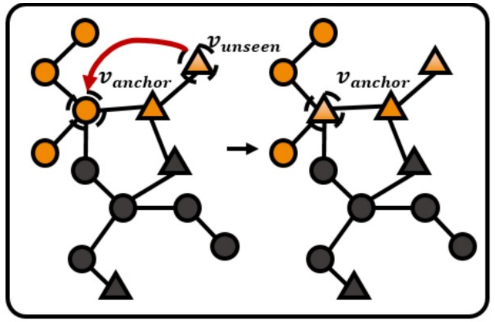
Neighbor-Aware Ego Network Synthesis for Class-Imbalanced Node Classification (GraphENS)

- **Motivation**: Neighbor memorization problem (Due to message passing of GNNs)
  - Existing approaches overfit to neighbor sets of minor class nodes, rather than to minor nodes themselves

Node replacing



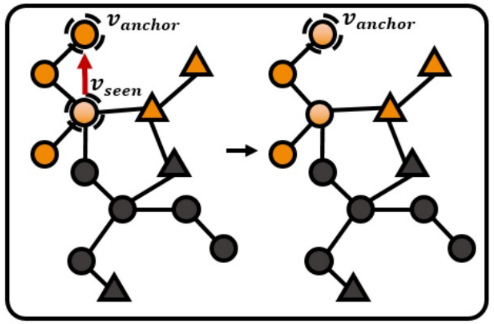
Seen nodes surrounded by **Seen** neighbors



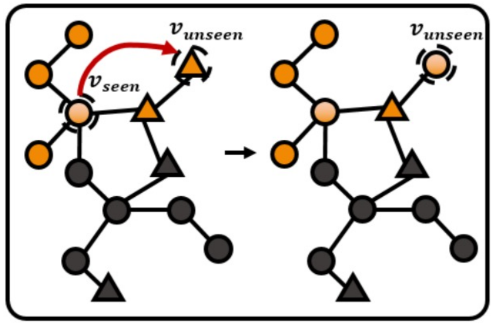
Unseen nodes surrounded by **Seen** neighbors

To see whether the model overfits to the node

Neighbor replacing



Seen nodes surrounded by **Seen** neighbors



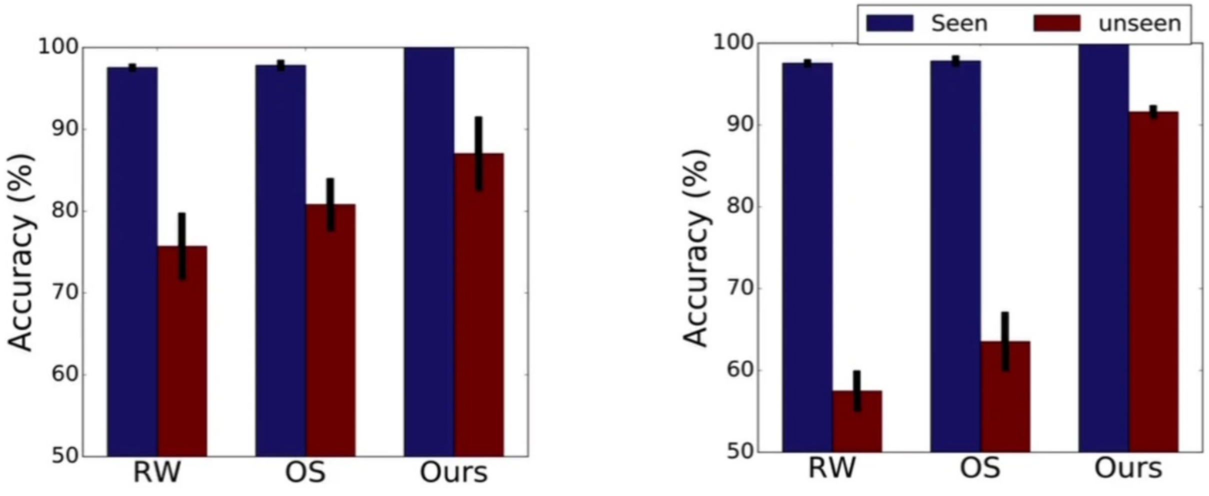
Seen nodes surrounded by **Unseen** neighbors

To see whether the model overfits to the neighbors

GraphENS: Neighbor-Aware Ego Network Synthesis for Class-Imbalanced Node Classification. ICLR 2022.

Long-Tailedness: Class Perspective
Neighbor-Aware Ego Network Synthesis for Class-Imbalanced Node Classification (GraphENS)

- **Motivation**: Neighbor memorization problem (Due to message passing of GNNs)
  - Existing approaches overfit to neighbor sets of minor class nodes, rather than to minor nodes themselves



(c) Node-Replacing

(d) Neighbor-Replacing

: Accuracy of **seen** nodes surrounded by seen neighbors

: Accuracy of **unseen** nodes surrounded by seen neighbors

: Accuracy of seen nodes surrounded by **seen** neighbors

: Accuracy of seen nodes surrounded by **unseen** neighbors

Performance drop of existing approaches in the neighbor replacing experiment is steeper than in the node replacing

→ **Neighbor memorization problem is a critical obstacle!**

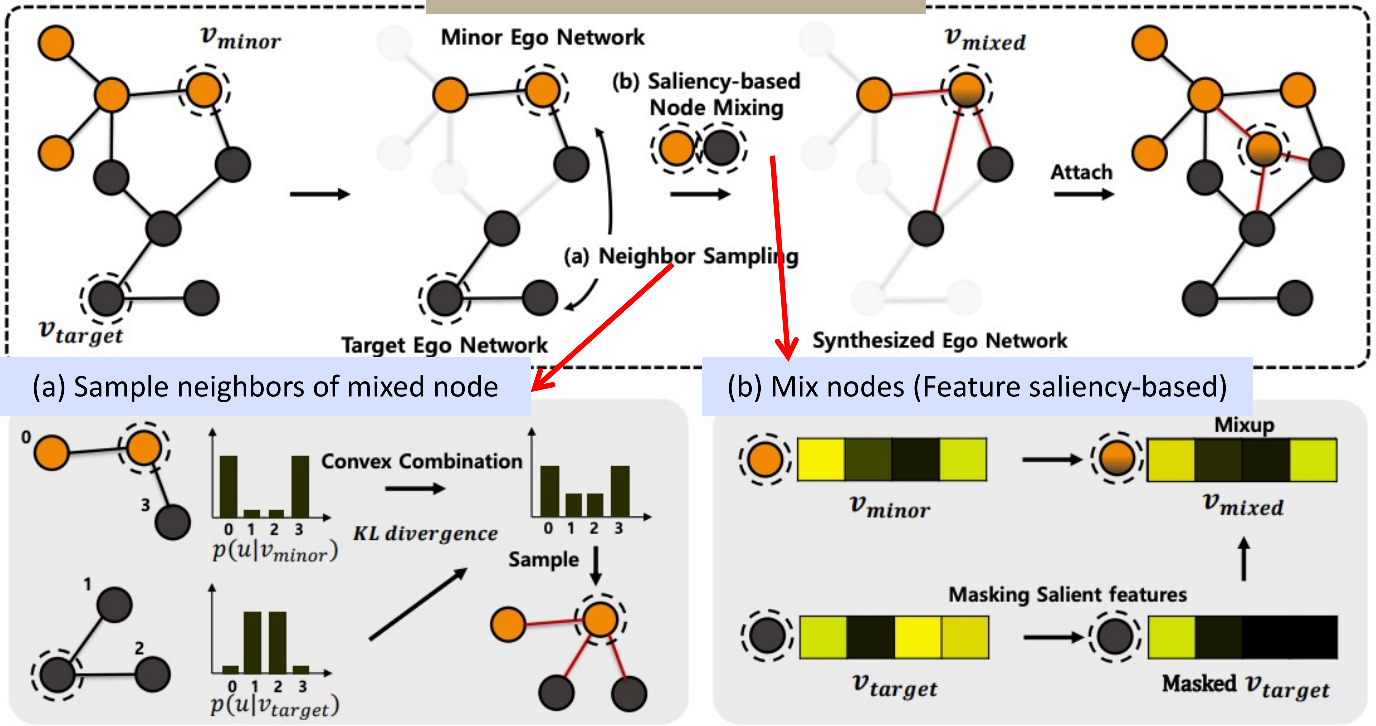
GraphENS: Neighbor-Aware Ego Network Synthesis for Class-Imbalanced Node Classification. ICLR 2022.

Long-Tailedness: Class Perspective

Neighbor-Aware Ego Network Synthesis for Class-Imbalanced Node Classification (GraphENS)

- **Main idea**: 1) Mix two ego-networks rather than mixing two nodes, 2) Selectively mix features

Overall framework



Neighbor distribution of mixed node

$$p(u|v_{mixed}) = \hat{\phi}\, p(u|v_{minor}) + (1-\hat{\phi})\, p(u|v_{target})$$

KLD btw logits of minor and target node

Rely on more similar target node

$$v_{mixed} = (1-\Lambda_K) \odot v_{minor} + \Lambda_K \odot v_{target}$$

Select top-K features with high gradients and mask them
→ Mix with general features (non class-specific)

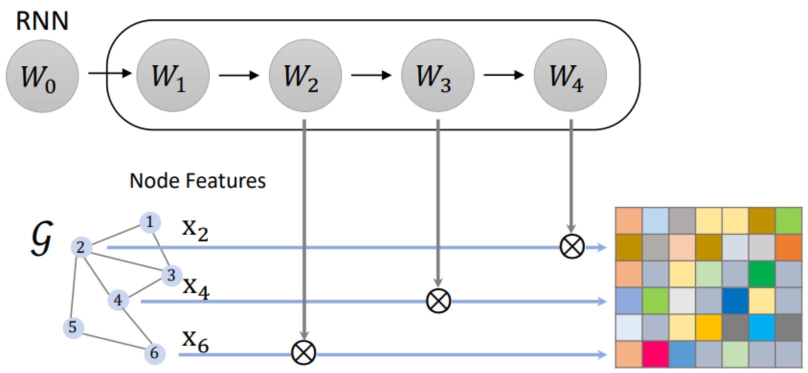
GraphENS: Neighbor-Aware Ego Network Synthesis for Class-Imbalanced Node Classification. ICLR 2022.

Long-Tailedness: Degree Perspective

Investigating and Mitigating Degree-Related Biases in Graph Convolutional Networks (SL-DSGC)

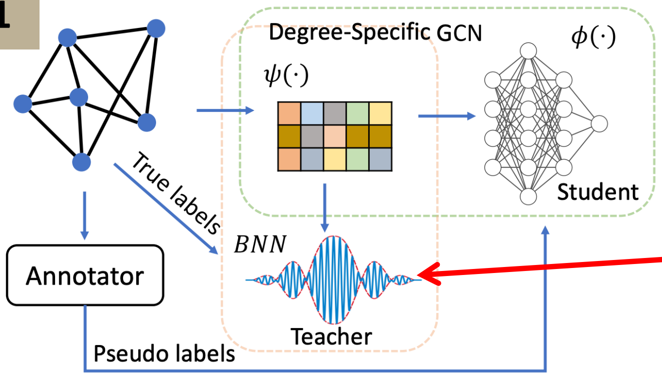
- **Motivation**: Given limited supervision, performance of GCNs becomes unsatisfying for low-degree nodes
- **Approach**
  - Bias reduction in **model perspective** (due to parameter sharing btw nodes)
    - **Degree-specific GCN** layer with RNN to generate degree-specific parameters:

$$\mathbf{x}_i^{l+1} = \sigma\Big(\sum_{j \in \mathcal{N}(i)} a_{ij}(\mathbf{W}^l + \underbrace{\mathbf{W}^l_{d(j)}}_{\text{Degree-specific parameter}})\mathbf{x}_j^l\Big) \qquad W^l_{k+1} = \mathrm{RNN}(W^l_k),\ k = 0, 1, \cdots, d_{\max},$$



  - Bias reduction in **data perspective**
    - Create **pseudo labels** with uncertainty scores → Pseudo labels increase the chance of connecting to low-degree nodes

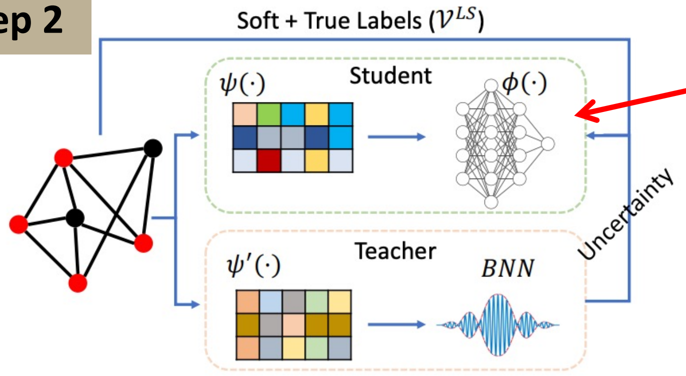
Step 1



Bayesian Neural Network (BNN)
Avoid overfitting on inaccurate pseudo labels

Pretrain student and teacher (independently)

Step 2



Use generated labels and uncertainty scores from teacher
→ Get clean pseudo label

Dynamic step size

$$\eta_i = \eta \cdot \exp(-\alpha c_i) \cdot \exp(\beta d_i)$$

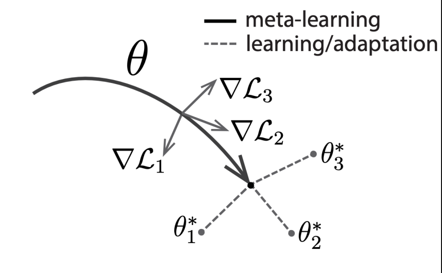
Confidence obtained from BNN

Fine-tune student using generated labels and uncertainty scores from teacher

Long-Tailedness: Degree Perspective

Towards Locality-Aware Meta-Learning of Tail Node Embeddings on Networks (meta-tail2vec)

- **Motivation**: How do we learn embedding vectors for tail nodes from limited structural information?
- **Idea**: Tail-degree node embeddings as a **few-shot regression** problem (i.e., few links on each tail node)
  - Meta-learning (Obtain information from head-degree node and transfer it to tail-degree node)

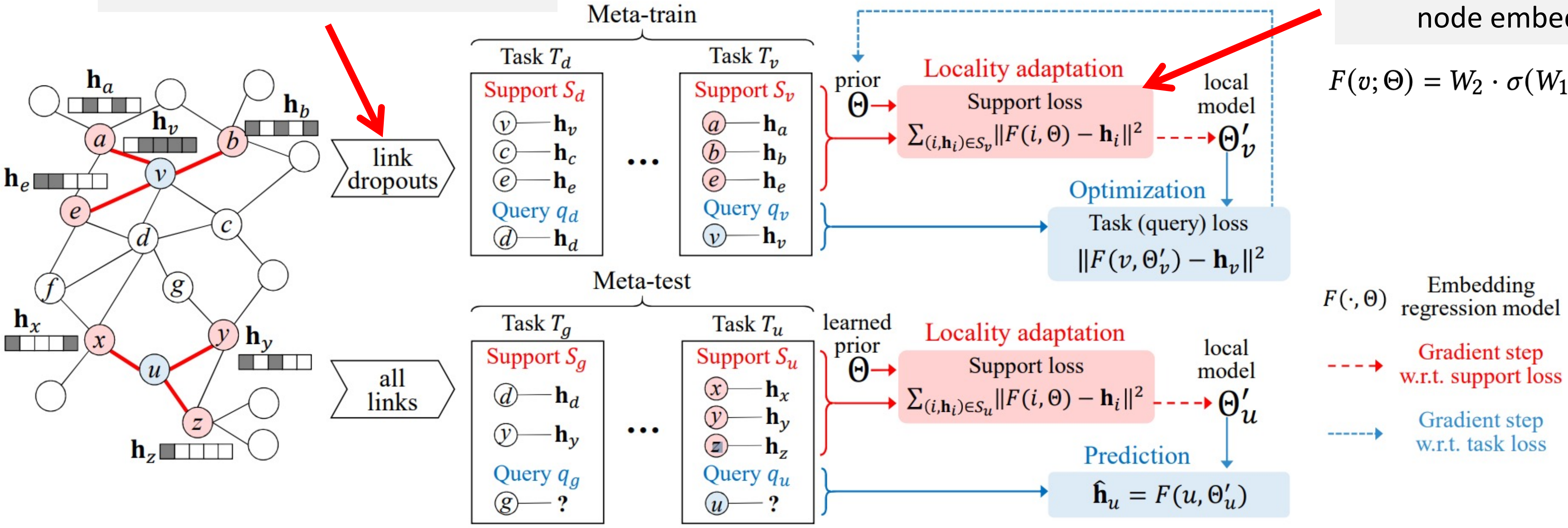


Model-agnostic Meta Learning

To simulate test environment

Reconstruct head-degree node embedding

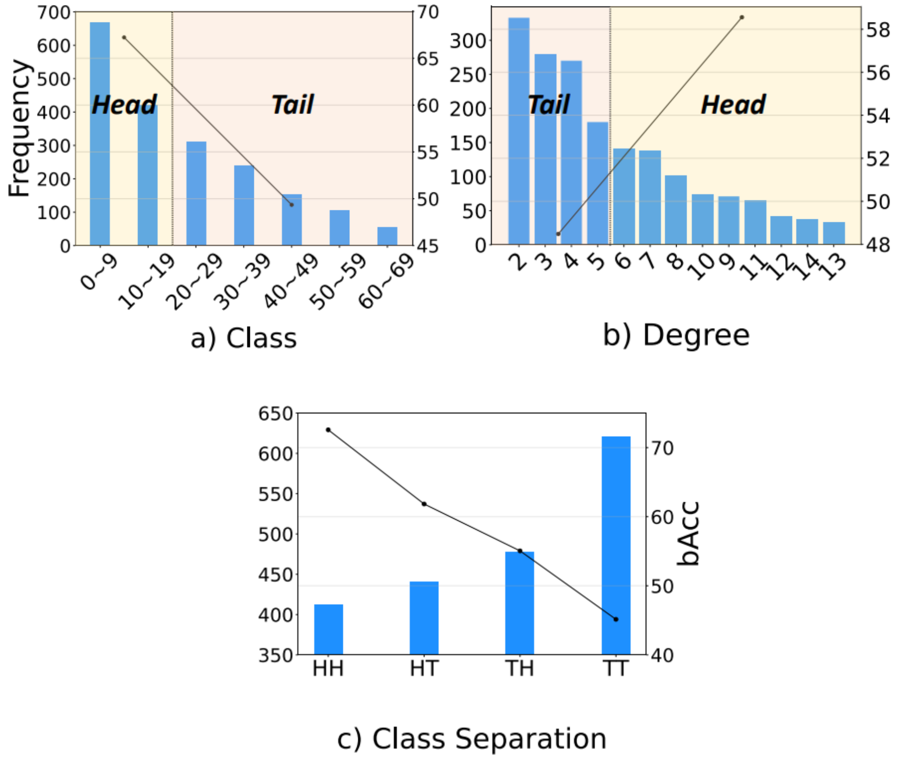
$$F(v;\Theta) = W_2 \cdot \sigma(W_1\mathbf{x}_v + \mathbf{b}_1) + \mathbf{b}_2$$

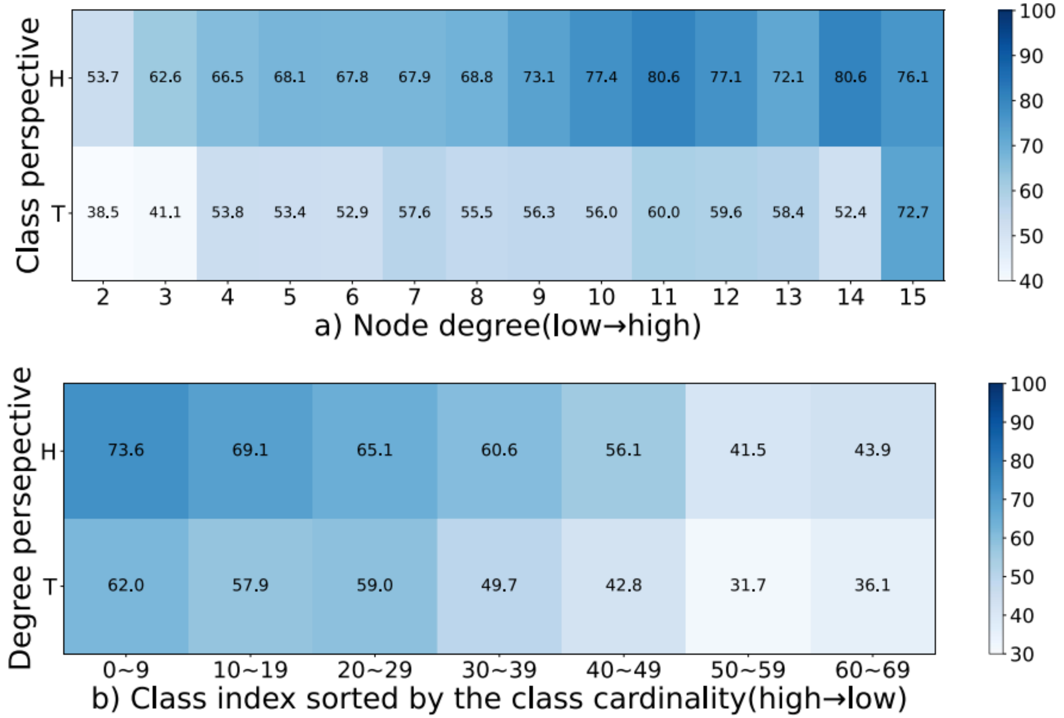


(a) Toy network (b) Locality-aware tasks (c) Locality-aware meta-learning

Long-Tailedness: Class & Degree Perspective
Long-Tail Experts for Graph Neural Networks (LTE4G)

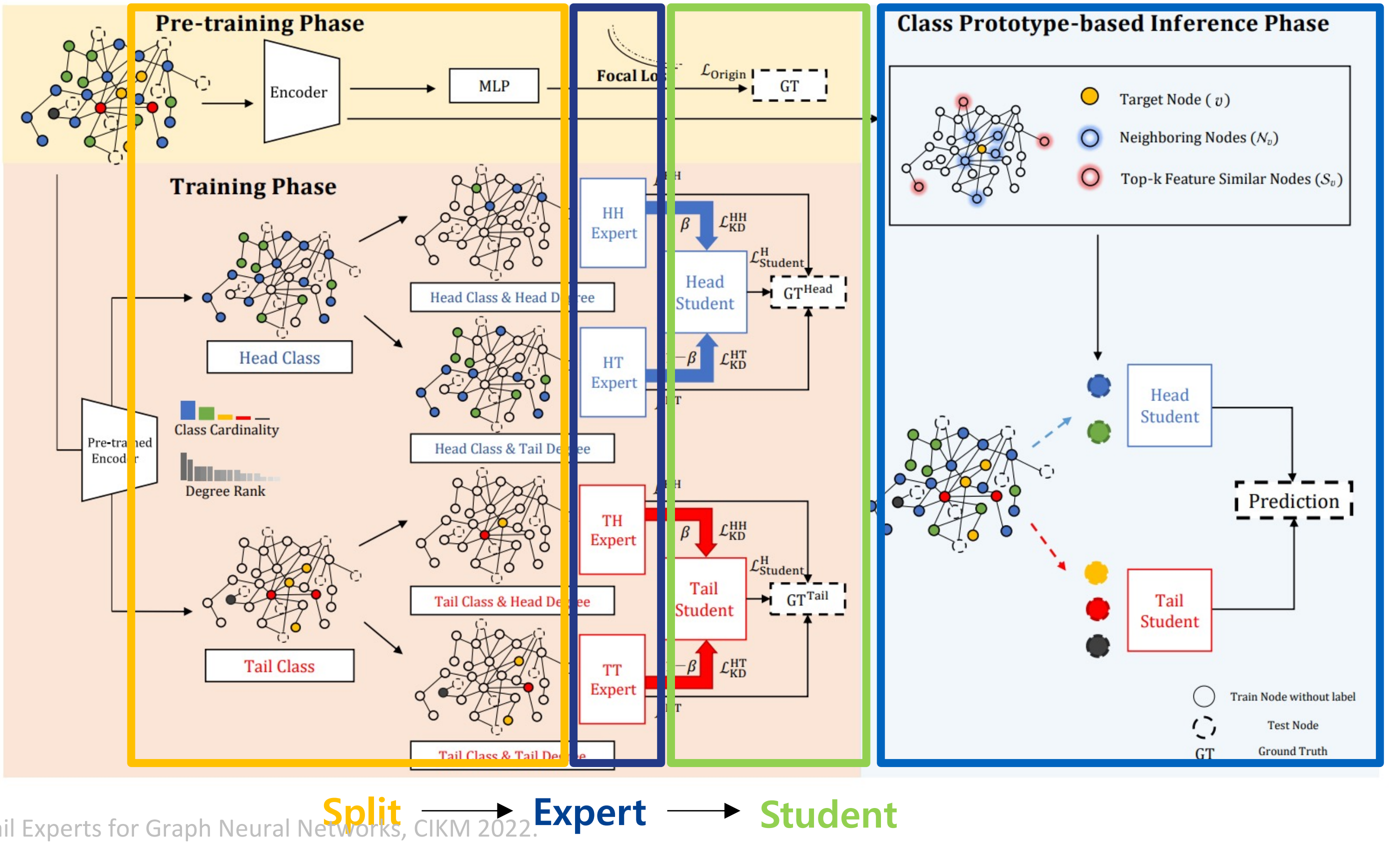
- **Motivation**: Both class and degree- longtailedness should be considered at the same time





Long-Tailedness: Class & Degree Perspective
Long-Tail Experts for Graph Neural Networks (LTE4G)

- **Idea**
  - Obtain balanced subsets of nodes and assign an expert to each subset (HH,HT,TH,TT)
  - Knowledge distillation between experts and class-wise students
    - Distill knowledge of **HH/HT experts to Head-class student** & **TH/TT experts to Tail-class student**



Pre-training Phase

- Obtain a Pre-trained Encoder

Training Phase

- Split nodes in a balanced manner
- Obtain Experts and Students
- Using Knowledge Distillation
- Using Head-to-Tail Learning

Class Prototype-based Inference Phase

- Using Candidates for Class Prototype
- Assign each test node to a student

Long-Tailedness: Class & Degree Perspective
Long-Tail Experts for Graph Neural Networks (LTE4G)

- Experiments: Datasets & Metrics

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2,708	5,429	1,433	7
CiteSeer	3,327	4,732	3,703	6
Cora-Full	19,793	146,635	8,710	70

Dataset	Imb. class	Imb. ratio	L_0	L_1	L_2	L_3	L_4	L_5	L_6
Cora	3	10%	23.3	23.3	23.3	23.3	2.4	2.4	2.4
		5%	24.1	24.1	24.1	24.1	1.2	1.2	1.2
	5	10%	40.0	40.0	4.0	4.0	4.0	4.0	4.0
		5%	44.4	44.4	2.2	2.2	2.2	2.2	2.2
	LT	1%	54.0	25.0	11.6	5.4	2.4	1.2	0.5
CiteSeer	3	10%	30.3	30.3	30.3	3.0	3.0	3.0	-
		5%	31.7	31.7	31.7	1.6	1.6	1.6	-
	5	10%	66.7	6.7	6.7	6.7	6.7	6.7	-
		5%	80.0	4.0	4.0	4.0	4.0	4.0	-
	LT	1%	60.7	24.1	9.5	3.8	1.5	0.5	-
Cora-Full	-	1.1%	34.0	18.9	14.1	10.9	6.9	4.8	2.6

1) Balanced Accuracy (bAcc) $= \frac{(\text{True Positive Rate} + \text{True Negative Rate})}{2}$

2) Macro-F1 $= \frac{1}{|C|} \sum_{c \in C} \frac{2*(\text{Precision}_c * \text{Recall}_c)}{\text{Precision}_c + \text{Recall}_c}$

3) Geometric Means (G-Means) $= (\prod_{c \in C} \text{Sensitivity}_c)^{\frac{1}{|C|}}$

4) Accuracy (Acc) $= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$

Predict

Actual

	Positive	**Negative**
Positive	TP	FN
Negative	FP	TN

Long-Tailedness: Class & Degree Perspective

Long-Tail Experts for Graph Neural Networks (LTE4G)

- 1) Overall Performance on **manual imbalanced** datasets

	Method	Imb. class num: 3						Imb. class num: 5					
		Imbalance_ratio: 10%			*Imbalance_ratio: 5%*			*Imbalance_ratio: 10%*			*Imbalance_ratio: 5%*		
		bAcc.	Macro-F1	G-Means	bAcc.	Macro-F1	G-Means	bAcc.	Macro-F1	G-Means	bAcc.	Macro-F1	G-Means
Cora	Origin	71.0±3.4	70.8±3.7	82.2±2.2	61.6±3.5	58.6±4.3	75.9±2.4	66.8±1.1	66.9±1.4	79.4±0.7	57.7±5.2	56.1±5.8	73.2±3.6
	Over-sampling	65.5±1.8	64.5±2.0	78.5±1.2	61.6±3.4	57.0±4.9	75.9±2.3	58.0±0.8	56.9±1.1	73.4±0.6	44.6±5.3	40.7±5.6	63.5±4.1
	Re-weight	72.9±2.7	72.3±3.7	83.4±1.7	64.7±4.5	62.5±5.5	78.0±3.0	67.5±1.8	67.3±2.2	79.9±1.2	59.1±1.7	56.7±2.7	74.2±1.2
	SMOTE	66.4±3.8	64.7±5.5	79.1±2.6	61.6±3.4	57.0±4.9	75.9±2.3	61.0±2.6	61.1±3.3	75.5±1.8	44.6±5.3	40.7±5.6	63.5±4.1
	Embed-SMOTE	65.5±4.2	63.4±4.7	78.6±2.8	59.3±5.5	54.2±7.4	74.3±3.7	57.5±4.9	55.2±5.5	73.0±3.4	44.3±6.9	41.0±9.0	63.2±5.5
	GraphSMOTE_T	71.2±2.4	70.2±3.0	82.3±1.6	65.7±1.5	63.3±2.7	78.7±1.0	67.2±1.8	67.2±2.4	79.7±1.2	58.7±2.8	58.0±2.2	73.9±1.9
	GraphSMOTE_O	70.7±1.9	70.0±2.5	82.0±1.3	64.2±4.0	62.5±4.4	77.7±2.7	67.6±1.8	66.9±2.1	80.0±1.2	61.6±3.0	59.9±3.5	75.9±2.1
	GraphSMOTE_{preT}	71.8±5.4	70.4±6.4	82.7±3.5	67.3±5.9	63.9±8.3	79.7±3.9	69.0±2.8	68.0±2.5	80.9±1.8	67.5±3.7	64.8±3.8	79.9±2.4
	GraphSMOTE_{preO}	73.4±2.1	72.5±2.0	83.8±1.4	68.2±0.4	65.8±1.9	80.4±0.3	67.6±5.5	65.7±5.8	79.9±3.6	67.2±3.4	64.6±3.5	79.7±2.2
	GraphENS	62.0±3.6	58.2±4.6	76.2±2.4	56.5±4.7	51.4±6.9	72.4±3.3	44.8±4.0	41.3±4.2	63.7±3.1	34.5±2.9	30.3±4.1	55.4±2.5
	Tail-GNN	63.1±3.5	60.4±3.5	76.9±2.4	54.7±4.4	48.0±7.6	71.1±3.1	55.7±6.2	54.7±6.9	71.7±4.3	39.2±6.9	33.6±9.5	59.2±5.6
	LTE4G	**73.6**±2.6	**73.0**±2.5	**83.9**±1.7	**70.9**±3.1	**69.4**±2.5	**82.1**±2.0	**74.2**±1.8	**73.9**±1.9	**84.3**±1.1	**71.9**±3.5	**70.9**±3.6	**82.8**±2.3
CiteSeer	Origin	46.3±2.5	37.2±3.4	64.2±1.9	43.3±1.2	33.1±2.1	62.0±1.0	41.1±2.9	37.2±3.7	60.2±2.4	29.8±1.4	23.4±1.6	50.6±1.2
	Over-sampling	48.1±2.7	41.4±5.3	65.6±2.1	45.7±3.2	36.6±4.3	63.8±2.5	34.9±2.9	31.2±3.6	55.1±2.5	33.8±2.9	27.5±0.8	54.1±2.5
	Re-weight	47.2±2.5	39.7±3.9	64.9±1.9	44.1±2.2	33.5±3.2	62.6±1.7	42.3±5.0	37.9±5.3	61.1±3.9	31.2±3.9	25.7±3.5	51.8±3.5
	SMOTE	46.4±2.6	37.6±3.3	64.3±2.0	45.7±3.2	36.6±4.3	63.8±2.5	34.7±0.7	27.3±3.2	55.0±0.6	33.8±2.9	27.5±0.8	54.1±2.5
	Embed-SMOTE	46.4±3.3	36.6±4.1	64.3±2.6	44.9±4.3	33.5±5.8	63.2±3.4	32.9±0.4	25.5±1.7	53.4±0.3	20.4±0.3	11.2±0.5	41.4±0.3
	GraphSMOTE_T	47.3±3.0	38.9±4.6	65.0±2.3	45.6±1.9	35.1±2.8	63.7±1.4	42.8±5.8	37.3±6.9	61.5±4.6	31.1±4.6	26.0±5.3	51.7±4.0
	GraphSMOTE_O	47.2±3.4	38.6±5.9	64.9±2.6	45.1±4.4	34.9±6.1	63.3±3.3	41.8±2.9	35.3±2.9	60.8±2.3	35.3±4.6	28.3±4.6	55.3±4.0
	GraphSMOTE_{preT}	45.5±3.7	37.3±4.5	63.6±2.9	41.2±2.8	31.0±2.6	60.3±2.3	46.3±4.9	42.9±4.9	64.2±3.8	34.1±7.7	28.6±8.4	54.1±6.8
	GraphSMOTE_{preO}	45.2±1.9	38.2±1.5	63.4±1.4	40.9±1.3	30.4±1.8	60.1±1.1	46.4±4.3	43.3±4.6	64.3±3.4	34.0±7.7	28.3±8.2	54.0±6.9
	GraphENS	46.7±2.4	39.2±3.9	64.6±1.8	44.2±1.2	35.4±1.9	62.7±1.0	28.9±5.0	23.6±6.2	49.6±4.6	25.4±2.0	20.4±4.1	46.4±2.0
	Tail-GNN	44.2±1.6	34.3±2.5	62.7±1.3	41.8±0.7	30.1±2.6	60.8±0.6	32.1±4.7	26.4±6.1	52.6±4.2	27.8±5.0	21.5±4.7	48.6±4.6
	LTE4G	**51.0**±1.6	**50.1**±0.7	**67.8**±1.2	**50.5**±0.8	**48.6**±1.2	**67.5**±0.6	**49.6**±2.3	**47.0**±3.7	**66.8**±1.7	**46.7**±0.6	**44.4**±4.8	**64.5**±4.7

LTE4G: Long-Tail Experts for Graph Neural Networks, CIKM 2022.

Long-Tailedness: Class & Degree Perspective
Long-Tail Experts for Graph Neural Networks (LTE4G)

- 2) Overall Performance on **manual LT** & **natural datasets**

Method	Cora-LT			CiteSeer-LT		
	bAcc.	Macro-F1	G-Means	bAcc.	Macro-F1	G-Means
Origin	63.3±1.4	58.4±1.4	77.1±0.9	48.3±1.8	41.7±1.4	65.8±1.4
Over-sampling	65.9±2.5	63.3±2.8	78.8±1.7	48.7±1.7	42.2±1.9	66.1±1.3
Re-weight	64.3±0.2	61.0±0.7	77.8±0.2	50.3±2.5	44.9±2.3	67.3±1.9
SMOTE	64.1±0.3	60.8±0.2	77.6±0.2	48.5±0.7	42.1±0.5	65.9±0.6
Embed-SMOTE	61.9±1.0	58.3±0.9	76.1±0.7	48.8±2.5	42.3±2.0	66.2±1.9
GraphSMOTE_T	65.2±2.2	62.3±2.9	78.4±1.4	50.8±1.8	45.6±1.8	67.7±1.3
GraphSMOTE_O	65.8±1.6	62.9±2.0	78.8±1.1	51.0±1.2	45.9±0.8	67.8±0.9
GraphSMOTE_{preT}	65.8±1.4	63.5±2.0	78.8±0.9	47.8±1.9	42.4±1.8	65.4±1.4
GraphSMOTE_{preO}	66.1±0.7	63.5±0.5	78.9±0.5	48.1±1.9	42.4±1.9	65.6±1.4
GraphENS	70.0±1.2	66.8±1.1	81.6±0.8	56.0±1.1	50.9±1.1	71.4±0.8
Tail-GNN	63.2±2.0	57.6±1.8	77.0±1.3	53.1±0.9	48.2±1.4	69.4±0.7
LTE4G	**72.6**±1.4	**72.4**±1.5	**83.3**±0.9	**60.6**±1.7	**55.0**±1.9	**74.7**±1.2

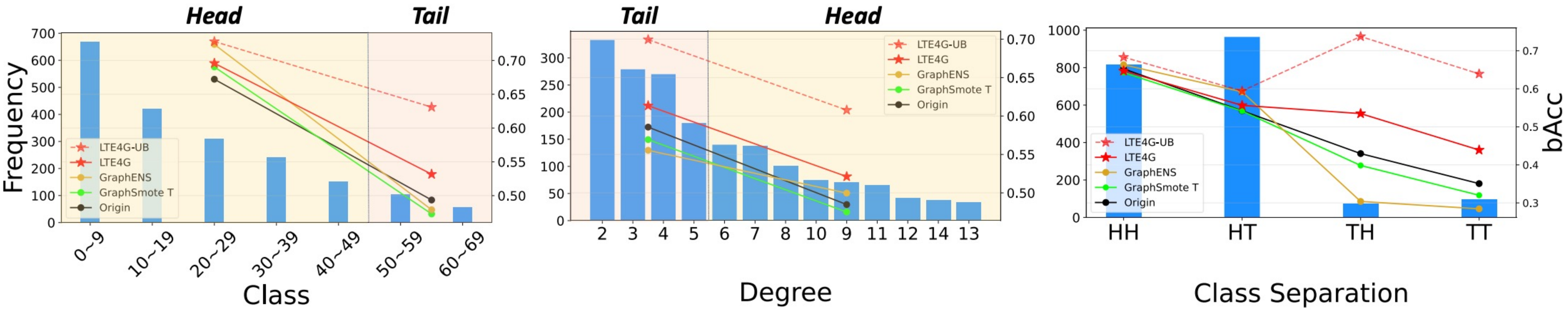
Method	Cora-Full			
	bAcc.	Macro-F1	G-Means	Acc.
Origin	52.0±1.0	52.5±0.8	71.9±0.7	60.5±0.2
Over-sampling	52.0±0.7	52.6±0.6	71.9±0.5	60.7±0.1
Re-weight	52.1±0.9	52.6±0.7	72.0±0.6	60.7±0.1
SMOTE	52.2±0.7	52.4±0.7	72.0±0.5	60.6±0.4
Embed-SMOTE	52.3±0.7	53.8±0.7	72.1±0.5	62.6±0.5
GraphSMOTE_T	51.9±0.6	52.4±0.4	71.8±0.4	60.6±0.2
GraphSMOTE_O	52.3±1.0	52.5±0.8	72.1±0.7	60.5±0.3
GraphSMOTE_{preT}	48.0±2.1	48.4±2.2	69.0±1.5	56.8±1.9
GraphSMOTE_{preO}	47.0±2.5	47.2±2.5	68.3±1.8	55.9±2.1
GraphENS	52.9±0.5	53.7±0.3	72.5±0.3	**63.4**±0.4
Tail-GNN	OOM	OOM	OOM	OOM
LTE4G	**56.3**±0.5	**55.2**±0.2	**74.8**±0.3	62.6±0.2

LTE4G outperforms SOTA in both manual and natural settings

LTE4G: Long-Tail Experts for Graph Neural Networks, CIKM 2022.

Long-Tailedness: Class & Degree Perspective
Long-Tail Experts for Graph Neural Networks (LTE4G)

- 3) Performance on each class, degree and joint consideration



LTE4G performs well in terms of **class** and **degree** + **class and degree jointly**

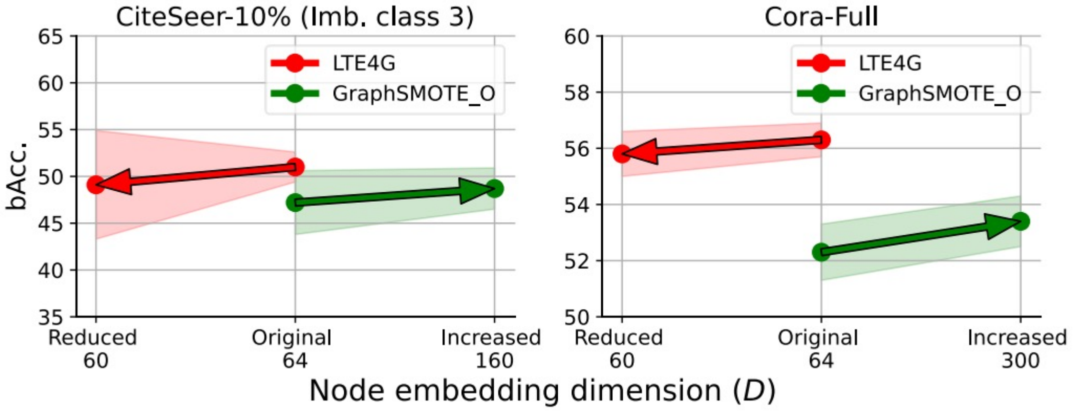
Long-Tailedness: Class & Degree Perspective
Long-Tail Experts for Graph Neural Networks (LTE4G)

- 4) Ablations on each component of LTE4G & balanced split of LTE4G

Components						Cora-5%(Imb Class 3)			Cora-5%(Imb Class 5)		
#	*C*	*D*	*KD*	*T2H*	*H2T*	bAcc.	Macro-F1	G-Means	bAcc.	Macro-F1	G-Means
(a)	✓					70.6±3.2	69.0±2.8	82.0±2.1	71.3±4.6	70.2±4.6	82.4±3.0
(b)		✓				50.6±1.2	44.8±1.8	68.1±0.9	43.5±2.8	40.1±4.5	62.7±2.2
(c)	✓	✓				59.9±3.3	59.0±3.6	74.8±2.3	55.6±4.6	56.2±0.5	71.7±3.3
(d)	✓	✓	✓			70.6±3.4	69.3±2.8	81.9±2.2	71.1±6.1	69.8±6.1	82.2±0.4
(e)	✓	✓	✓	✓		69.4±3.7	67.9±3.4	81.2±2.4	70.3±4.4	69.2±4.4	81.7±2.9
(f)	✓	✓	✓		✓	**70.9±3.1**	**69.4±2.5**	**82.1±2.0**	**71.9±3.5**	**70.9±3.6**	**82.8±2.3**

Balanced Split		Cora-LT			Cora-Full		
Class	Degree	bAcc.	Macro-F1	G-Means	bAcc.	Macro-F1	G-Means
✗	✗	65.7±1.4	61.6±1.5	78.7±0.9	53.2±0.9	54.1±0.9	72.8±0.6
✗	✓	63.0±1.4	58.7±1.6	76.9±1.0	53.3±0.9	54.0±0.9	72.8±0.6
✓	✗	72.3±0.9	72.0±1.0	83.0±0.6	55.4±0.9	54.6±0.6	74.2±0.6
✓	✓	**72.6**±1.4	**72.4**±1.5	**83.3**±0.9	**56.3**±0.5	**55.2**±0.2	**74.8**±0.3

- 5) Complexity analysis



Blindly increasing the number of parameters is **not beneficial**
→ Important to assign parameters in the right place where they are needed

This talk

- How to learn graph representation in **various types of graphs**?
  - ~~GNNs for Homogeneous Graph~~
  - GNNs for Multi-aspect Graph
  - GNNs for Multi-relational Graph
- How to effectively **train GNNs**?
  - Self-supervised learning
  - Alleviating Long-tail problem
  - Robustness of GNN

Adversarial Examples

- Assume a neural network that performs at human level accuracy
- Given a data point x, it is possible to build x' (an adversarial example) around x such that the neural netw ork makes nearly 100% error
- In many cases, x' is so similar to x that a human observer cannot tell the difference between x' and x
  - **Imperceptible** noise changes the prediction

Carefully calculated noise

Adversarial example

$+ .007 \times$

$=$

Classified as panda

Small adversarial noise

Classified as gibbon

x

ϵ

x'

Find x' satisfying $||x' - x|| \leq \Delta$ s.t. $C(x') \neq y$

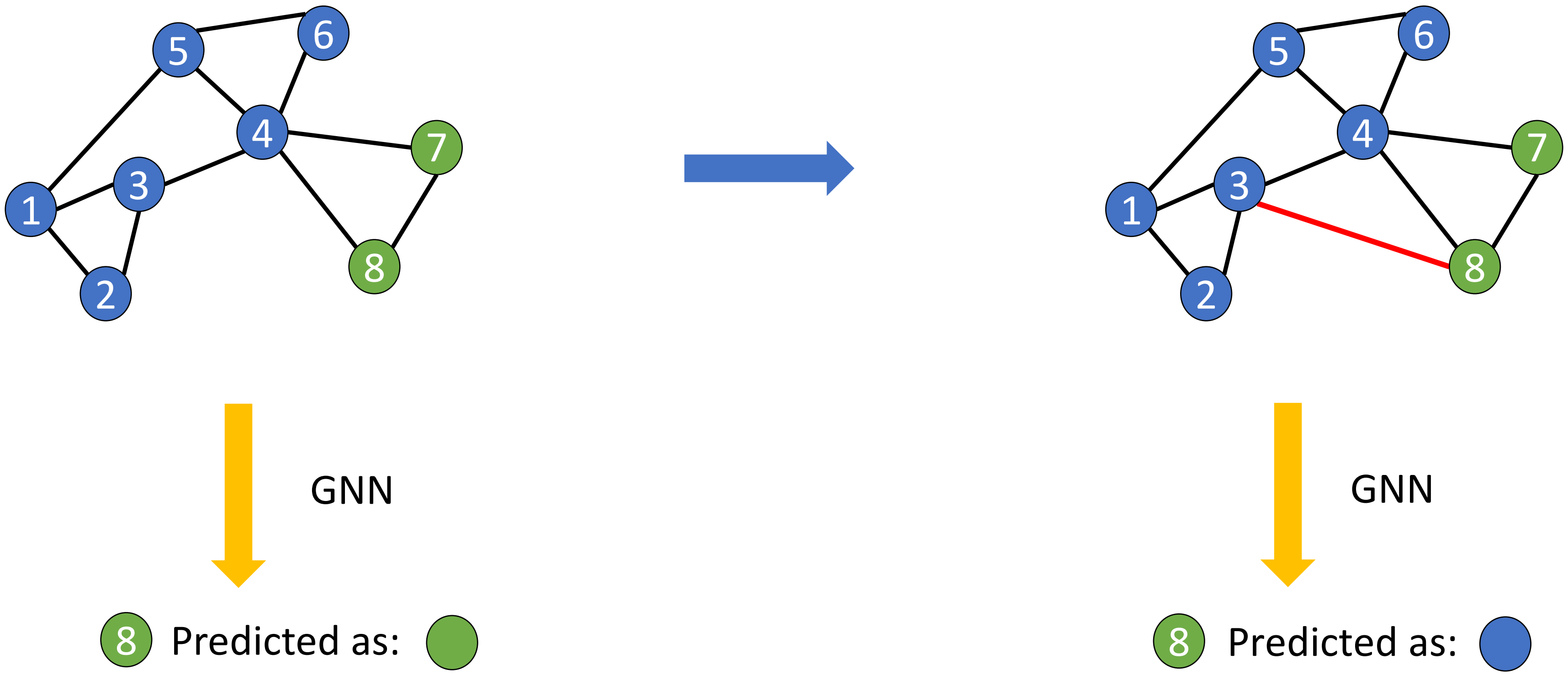
Implications of Adversarial Examples

- **The existence of adversarial examples prevents the reliable deployment of deep learning models to the real world**
  - Adversaries may try to actively hack the deep learning models.
  - The model performance can become much worse than we expect.
- **Deep learning models are often not robust**
  - In fact, it is an active area of research to make these models robust against adversarial examples

How about GNNs? Are they robust to adversarial examples?

(Figure credit) Graph Neural Networks: Models and Applications, AAAI 2021 Tutorial

Adversarial Attacks on GNN

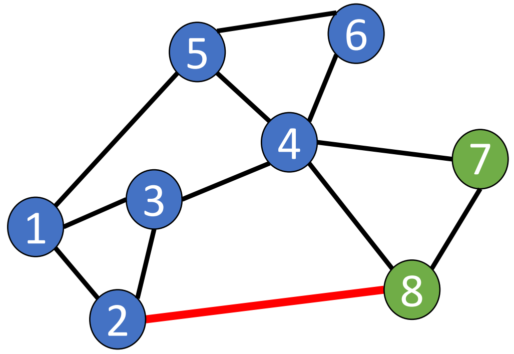


Why do we care about robust GNN?

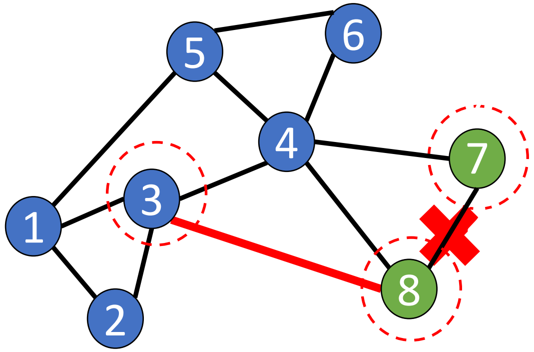
- Adversaries are very common in application scenarios, e.g. search engines, or recommender systems
  - Financial Systems
    - Credit Card Fraud Detection
  - Recommender Systems
    - Social Recommendation
    - Product Recommendation
  - Search engines
  - …
- These adversaries will exploit any vulnerabilities exposed

(Figure credit) Graph Neural Networks: Models and Applications, AAAI 2021 Tutorial

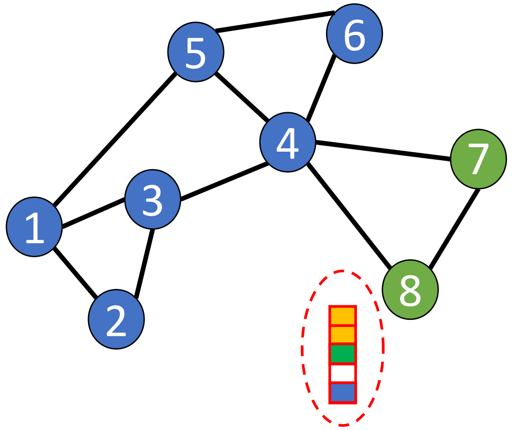
Perturbation Type



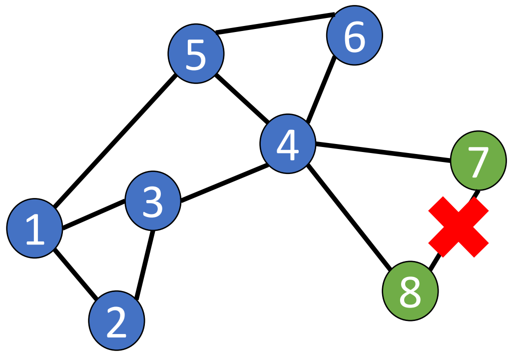
Adding an edge



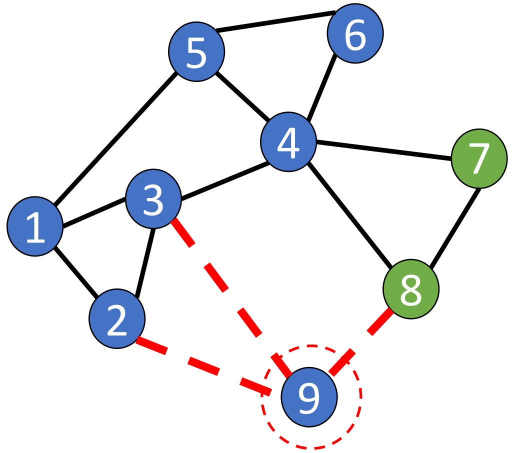
Rewiring



Modifying Features



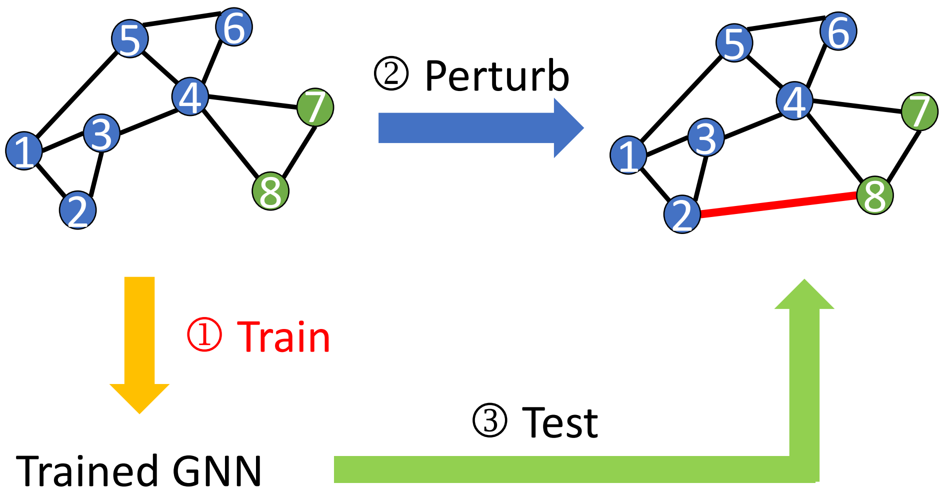
Deleting an edge



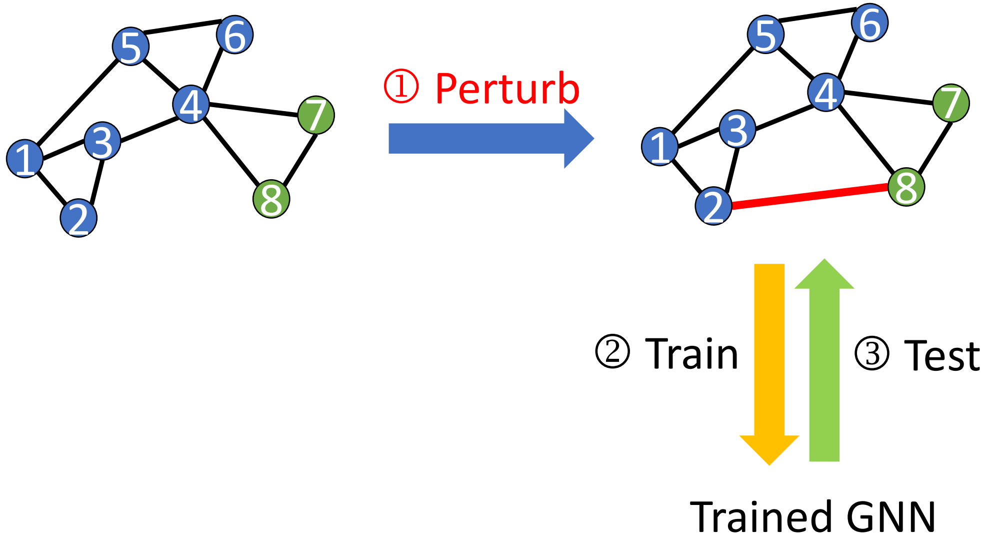
Node Injection

(Figure credit) Graph Neural Networks: Models and Applications, AAAI 2021 Tutorial

Evasion & Poisoning Attack



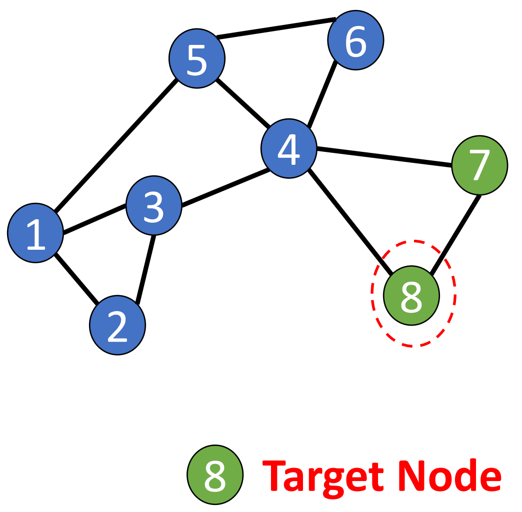
Evasion Attack



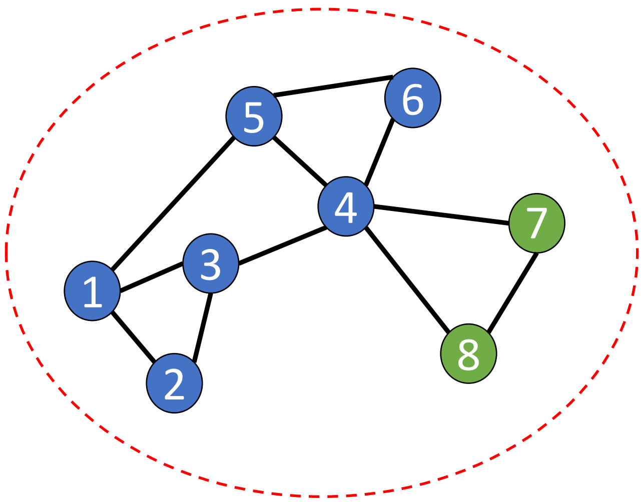
Poisoning Attack

Targeted & Non-Targeted Attack

Targeted Attack

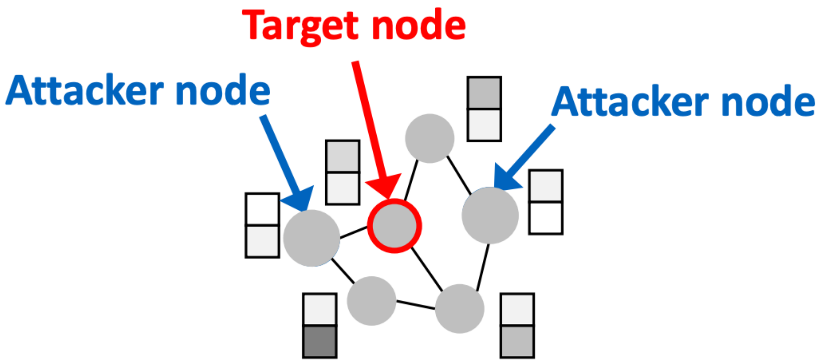


Non-Targeted Attack

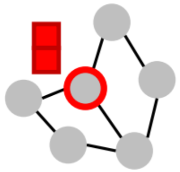
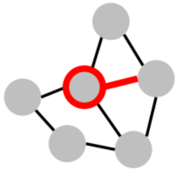


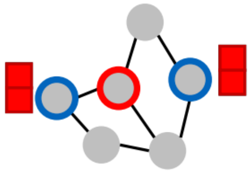
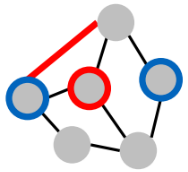
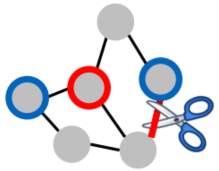
(Figure credit) https://www.cs.cit.tum.de/fileadmin/w00cfj/daml/nettack/kdd_talk.pdf

Direct & Indirect Attack



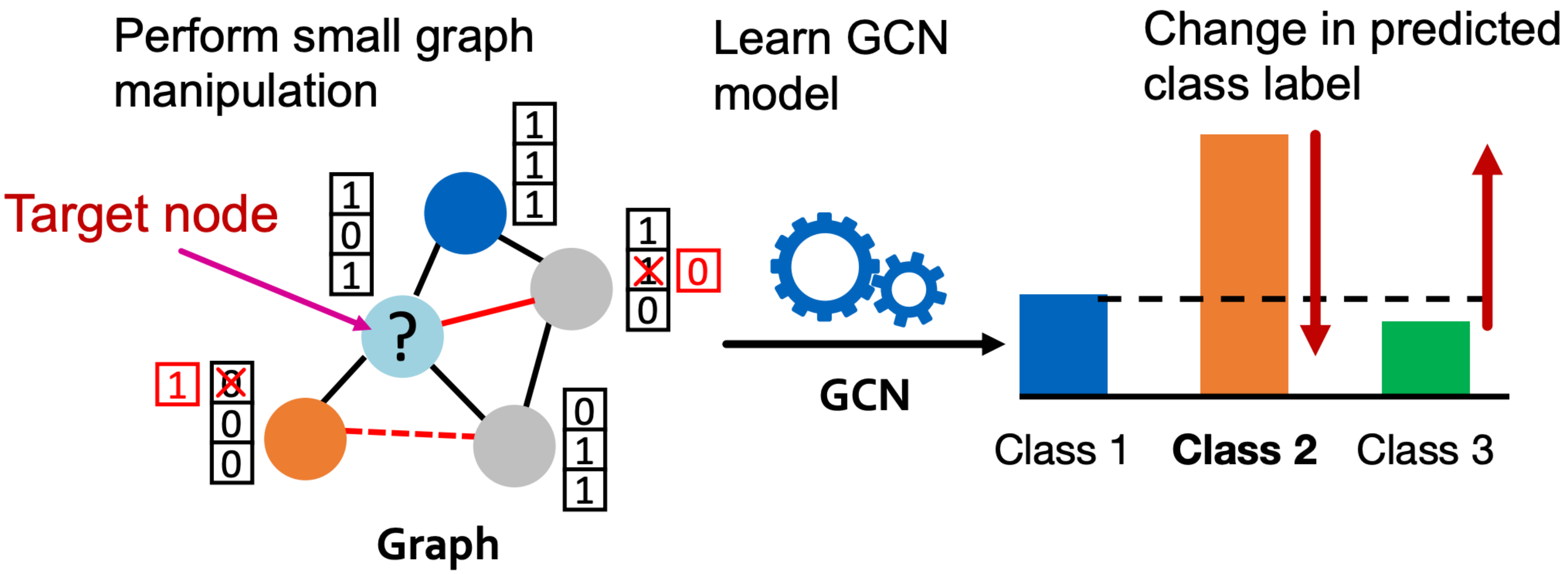
- **Target node $t \in V$**: node whose classification label we want to change
- **Attacker nodes $S \subset V$**: nodes the attacker can modify

Direct attack ($S = \{t\}$)		**Example**
• Modify the **target**'s features		Change website content
• Add connections to the **target**		Buy likes/ followers
• Remove connections from the **target**		Unfollow untrusted users

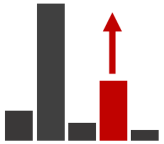
Indirect attack ($t \notin S$)		**Example**
• Modify the **attackers**' features		Hijack friends of target
• Add connections to the **attackers**		Create a link/ spam farm
• Remove connections from the **attackers**		

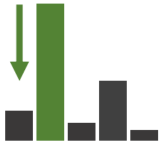
Objective for Attacker

- **Maximize**: Change of target node label prediction
- **Subject to**: Graph manipulation is small (imperceptible)
  - If graph manipulation is too large, it will easily be detected
  - Successful attacks should change the target prediction with "unnoticeably-small" graph manipulation



Poisoning Attack on Node Classification (Nettack)





$$\arg\max_{A',X'} \max_{c \neq c_{old}} \log Z^*_{v,c} - \log Z^*_{v,c_{old}}$$

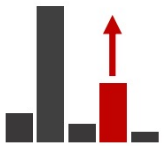
$$where\ Z^* = f_{\theta^*}(A', X') = softmax\left(\hat{A}'\ ReLU\left(\hat{A}'X'W^{(1)}\right)W^{(2)}\right),$$

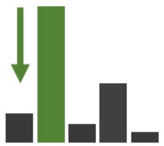
$$with\ \ \theta^* = \arg\min_{\theta} \mathcal{L}\,(\theta; A', X')$$

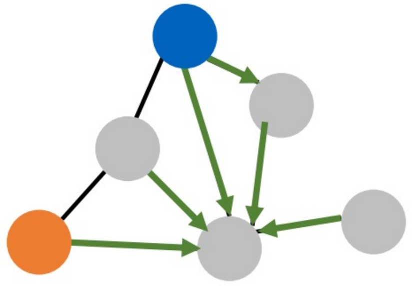
$A \in \{0,1\}^{N \times N}$: original adjacency matrix
$X \in \{0,1\}^{N \times D}$: (binary) node attributes
A': modified structure
X': modified features
v : target node

$$s.t.\,(A', X') \approx (A, X)$$

Poisoning Attack on Node Classification (Nettack)







Message passing

$$\arg\max_{A',X'} \max_{c \neq c_{old}} \log Z^*_{v,c} - \log Z^*_{v,c_{old}}$$

$$where\ Z^* = f_{\theta^*}(A',X') = softmax\left(\hat{A}'\ ReLU\left(\hat{A}'X'W^{(1)}\right)W^{(2)}\right),$$

$$with\ \ \theta^* = \arg\min_{\theta} \mathcal{L}\,(\theta; A', X')$$ (after re-train)

c.f. $\mathcal{L}(\theta; A, X)$: evasion

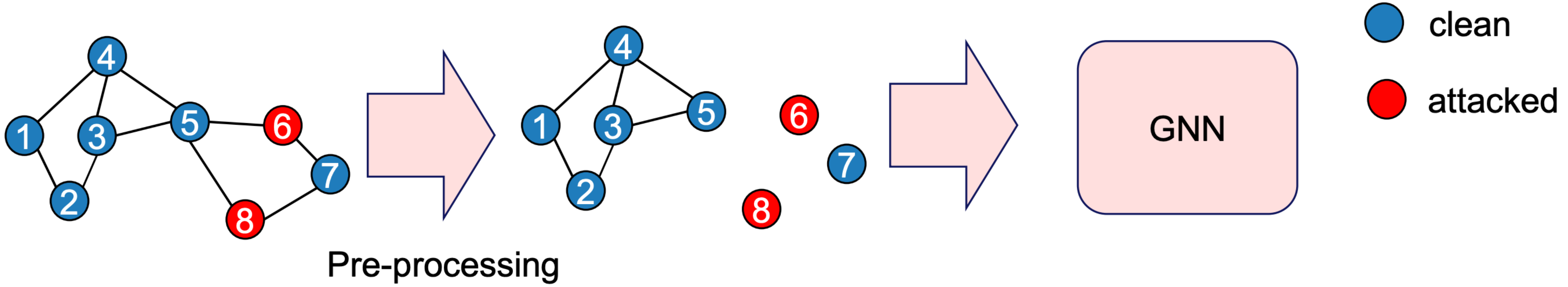
$s.t.\,(A',X') \approx (A,X)$

"Unnoticeability" constraint

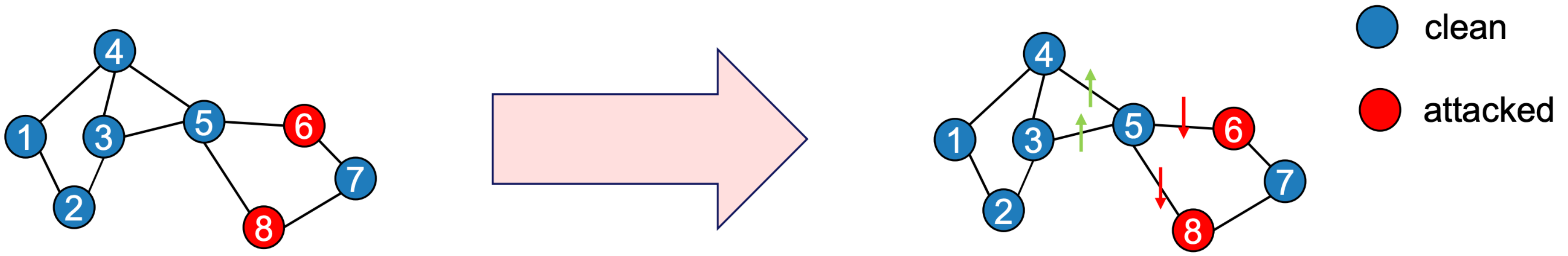
$A \in \{0,1\}^{N\times N}$: original adjacency matrix
$X \in \{0,1\}^{N\times D}$: (binary) node attributes
A': modified structure
X': modified features
v : target node

Adversarial Defense in GNN

- **Graph Purification-based Approach**
  - ProGNN (KDD 2020)

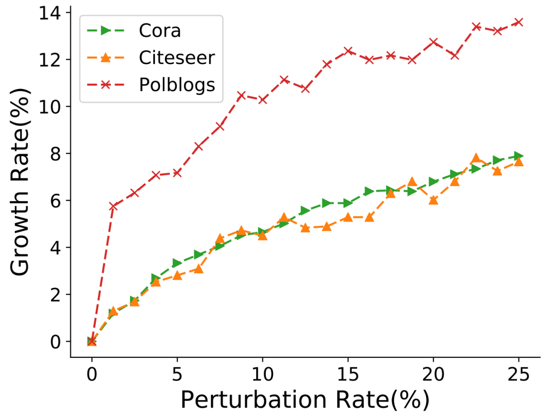


- **Attention-based Approach**
  - RGCN (KDD 2019), PA-GNN (WSDM 2020)

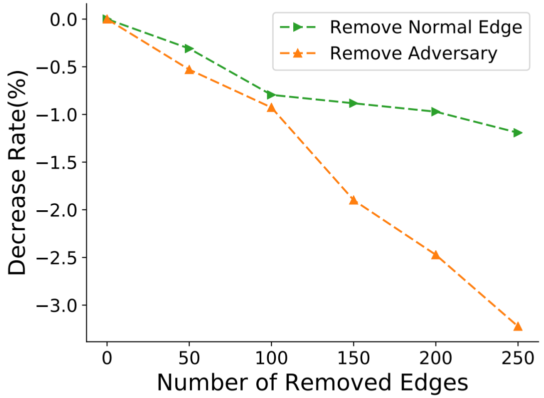


Defense: Graph Purify-based Approach
Graph Structure Learning for Robust Graph Neural Network (ProGNN)

- **Idea**: Preserve intrinsic properties of real-world graphs
  - Low-rank, Sparsity, Feature smoothness

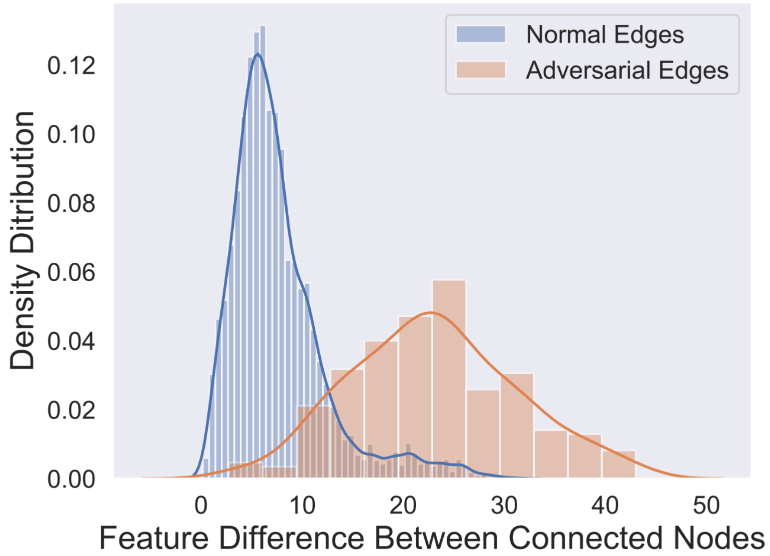


(b) Rank Growth



(c) Rank Decrease Rate

Low-rank



Feature smoothness

Defense: Graph Purify-based Approach
Graph Structure Learning for Robust Graph Neural Network (ProGNN)

- Approach
  - Add loss to ensure the graph is low-rank and sparse (**S**: the clean adjacent matrix we would like to learn)

$$\arg\min_{\mathbf{S}\in\mathcal{S}} \mathcal{L}_0 = \|\mathbf{A}-\mathbf{S}\|_F^2 + \alpha\|\mathbf{S}\|_1 + \beta\|\mathbf{S}\|_*, \; s.t., \mathbf{S}=\mathbf{S}^\top$$

$$||\mathbf{S}||_* = \Sigma_{i=1}^{rank(\mathbf{S})}\sigma_i$$

  - Add loss to penalize rapid changes in features between adjacent nodes:

$$\arg\min_{\mathbf{S}\in\mathcal{S}} \mathcal{L} = \mathcal{L}_0 + \lambda\cdot\mathcal{L}_s = \mathcal{L}_0 + \lambda\, tr(\mathbf{X}^T\hat{\mathbf{L}}\mathbf{X}), \; s.t., \; \mathbf{S}=\mathbf{S}^\top$$

$$\mathcal{L}_s = tr(\mathbf{X}^T\hat{\mathbf{L}}\mathbf{X}) = \frac{1}{2}\sum_{i,j=1}^{N}\mathbf{S}_{ij}\left(\frac{\mathbf{x}_i}{\sqrt{d_i}}-\frac{\mathbf{x}_j}{\sqrt{d_j}}\right)^2$$

  - Jointly learn the desired properties of graphs and the GNN model:

$$\begin{aligned}\arg\min_{\mathbf{S}\in\mathcal{S},\theta} \mathcal{L} &= \mathcal{L}_0 + \lambda\mathcal{L}_s + \gamma\mathcal{L}_{GNN} \\ &= \|\mathbf{A}-\mathbf{S}\|_F^2 + \alpha\|\mathbf{S}\|_1 + \beta\|\mathbf{S}\|_* + \gamma\mathcal{L}_{GNN}(\theta,\mathbf{S},\mathbf{X},\mathcal{Y}_L) + \lambda tr(\mathbf{X}^T\hat{\mathbf{L}}\mathbf{X}) \\ s.t. \quad & \mathbf{S}=\mathbf{S}^\top,\end{aligned}$$

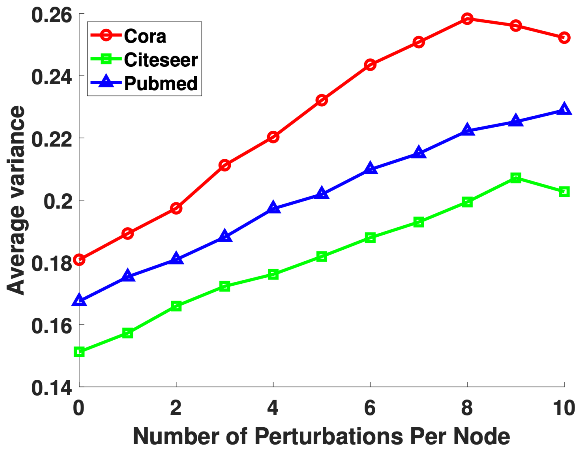
Graph Structure Learning for Robust Graph Neural Networks. KDD 2020.

Defense: Attention-based Approach

Robust Graph Convolutional Networks Against Adversarial Attacks (RGCN)

- **Motivation:** Attacked nodes may have high uncertainty → Give lower attention score to reduce their impact
- **Idea:** Adopt Gaussian distribution as the node representations $\mathbf{h}_i^{(l)} = \mathcal{N}(\boldsymbol{\mu}_i^{(l)}, diag(\boldsymbol{\sigma}_i^{(l)}))$

$$\mathbf{h}_{ne(i)}^{(l)} = \sum_{j \in ne(i)} \frac{1}{\sqrt{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}}} \mathbf{h}_j^{(l)} \sim \mathcal{N}\left(\sum_{j \in ne(i)} \frac{1}{\sqrt{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}}} \boldsymbol{\mu}_j^{(l)}, diag\left(\sum_{j \in ne(i)} \frac{1}{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}} \boldsymbol{\sigma}_j^{(l)}\right)\right)$$



- Variance-based attention mechanism
  - Assign different weights to node neighborhoods according to their variances
  - Attacked nodes have larger variances, give them small attention weights

    →Reduce influence of adversarial changes

$$\mathbf{h}_{ne(i)}^{(l)} = \sum_{j \in ne(i)} \frac{1}{\sqrt{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}}} (\mathbf{h}_j^{(l)} \odot \boldsymbol{\alpha}_j^{(l)})$$

$$\boldsymbol{\alpha}_j^{(l)} = exp(-\gamma \boldsymbol{\sigma}_j^{(l)})$$

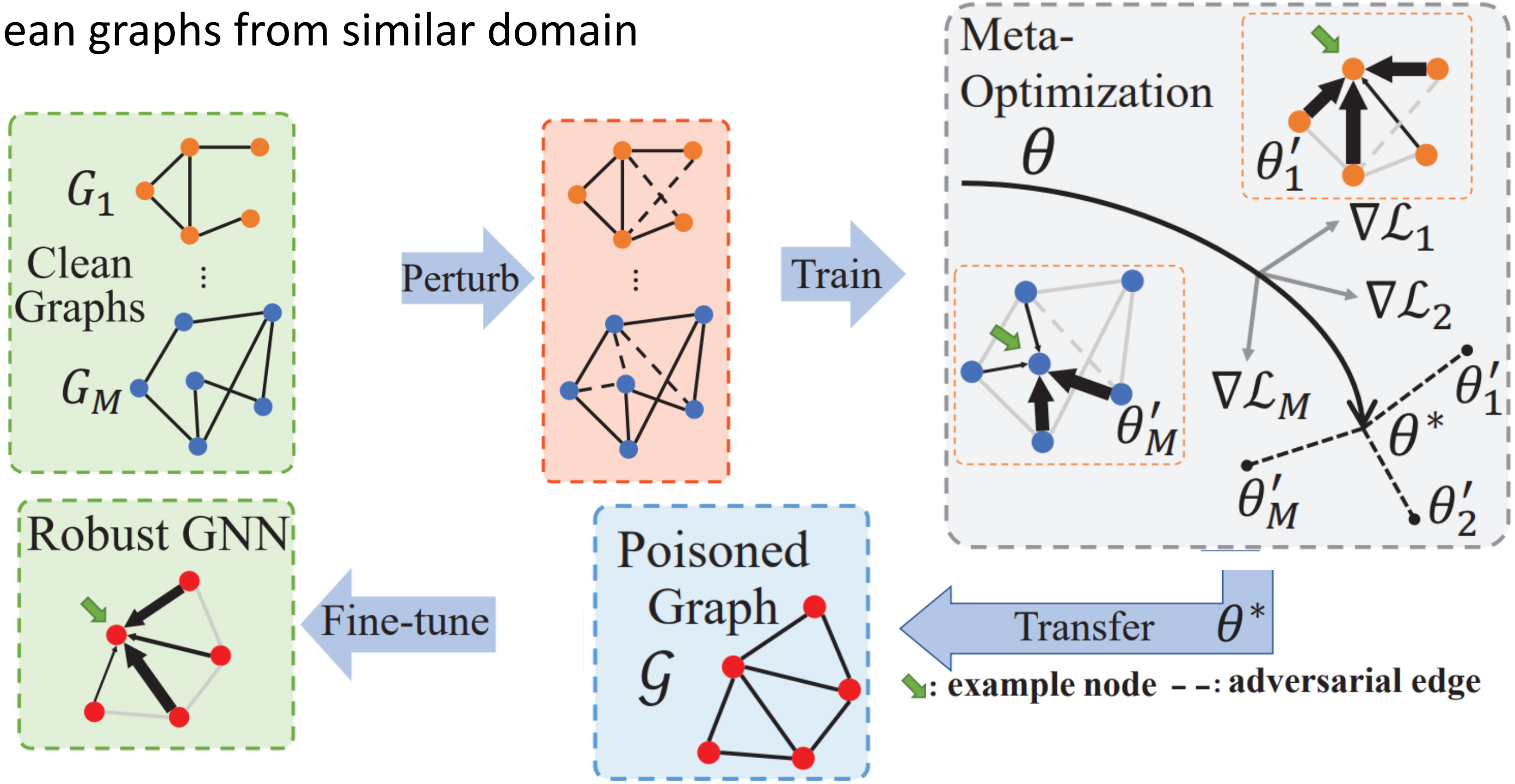
$$\mathcal{L}_{reg1} = \sum_{i=1}^{N} KL\left(\mathcal{N}(\boldsymbol{\mu}_i^{(1)}, diag(\boldsymbol{\sigma}_i^{(1)})) || \mathcal{N}(\mathbf{0}, \mathbf{I})\right)$$

Enforce Gaussian in the first layer

Defense: Attention-based Approach

Robust Graph Neural Network Against Poisoning Attacks via Transfer Learning (PA-GNN)

- **Motivation**
  - Only relying on perturbed graph to learn attention coefficients is not enough
  - We should exploit information from clean graphs
- **Assumption**: There are clean graphs from similar domain
  - Facebook & Twitter
  - Yelp & Foursquare



This talk

- How to learn graph representation in **various types of graphs**?
  - ~~GNNs for Homogeneous Graph~~
  - GNNs for Multi-aspect Graph
  - GNNs for Multi-relational Graph
- How to effectively **train GNNs**?
  - Self-supervised learning
  - Alleviating Long-tail problem
  - Robustness of GNN

Thank you

- Contact: cy.park@kaist.ac.kr
- Lab homepage: http://dsail.kaist.ac.kr