DSAIL Data Science & Artificial Intelligence

KAIST

Learning on Graphs

박찬영

Assistant Professor, KAIST

Industrial and Systems Engineering
Graduate School of Data Science
Graduate School of AI

cy.park@kaist.ac.kr

BRIEF BIO

Chanyoung Park, Ph.D.
Assistant Professor
ISYSE KAIST
GSDS/GSAI KAIST

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

Research Interest

- **Data-Centric AI, Model-Centric AI, Multimodal Data Mining, AI for Science**
  - *Mining meaningful knowledge from multimodal data to develop artificial intelligence solutions for various real-world applications across different disciplines*
  - Multi-modal Learning, Graph neural network
  - **Application domains**: Recommendation system, Social network analysis, Fraud detection, Sentiment analysis, Purchase/Click prediction, Anomaly detection, Knowledge-graph construction, Time-series analysis, Bioinformatics, Chemistry etc.

Professional Experience

- Assistant Professor, **KAIST** (2020.11 – Present)
- Postdoctoral Research Fellow, **University of Illinois at Urbana-Champaign**, Dept. of Computer Science (2019. 1 – 2020. 10)
- Research Intern, **Microsoft Research Asia** (2017. 9 – 2017. 12)
- Research Intern, **NAVER** (2017. 3 – 2017. 6)

Outline

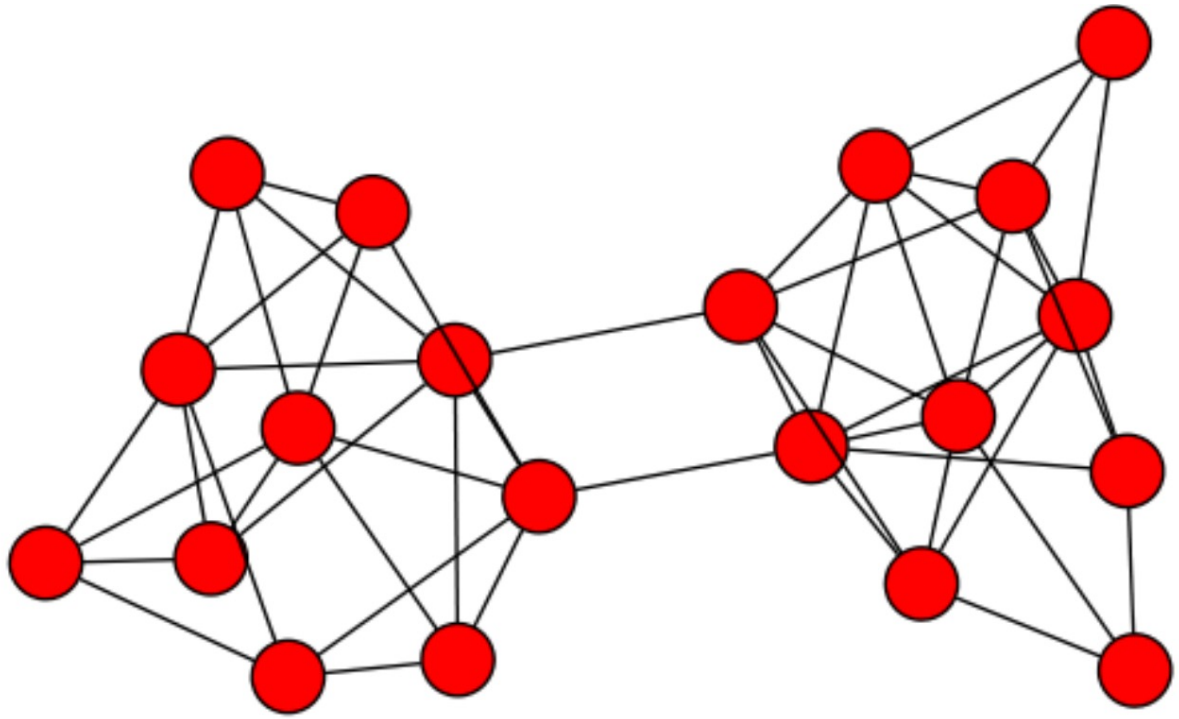
- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - ~~GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)~~
  - ~~GNNs for Computer vision (Scene Understanding) (AAAI'23)~~

Outline

- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)
  - GNNs for Computer vision (Scene Understanding) (AAAI'23)

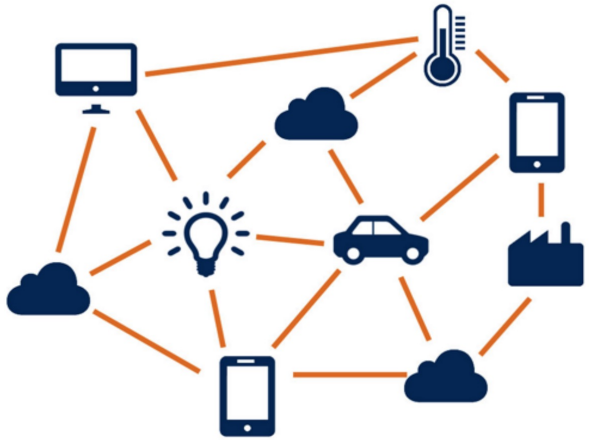
Graph (network)

- A general description of data and their relations
- A collection of objects (i.e., nodes), along with a set of interactions (i.e., edges) between pairs of these objects.

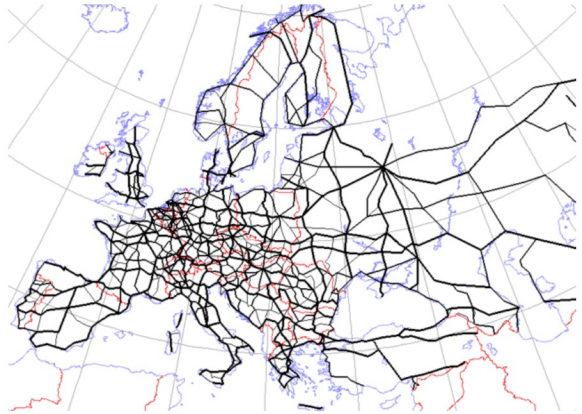


Various real-world graphs (networks)

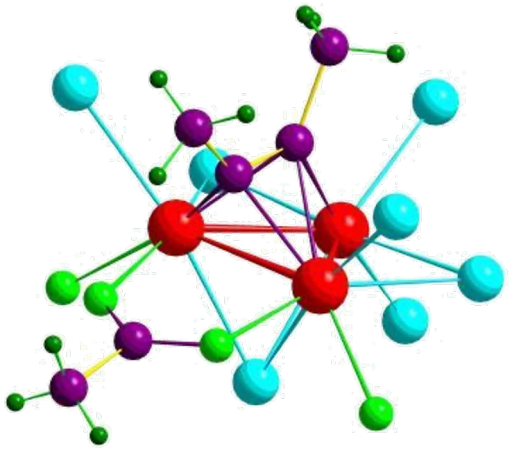
Social graph



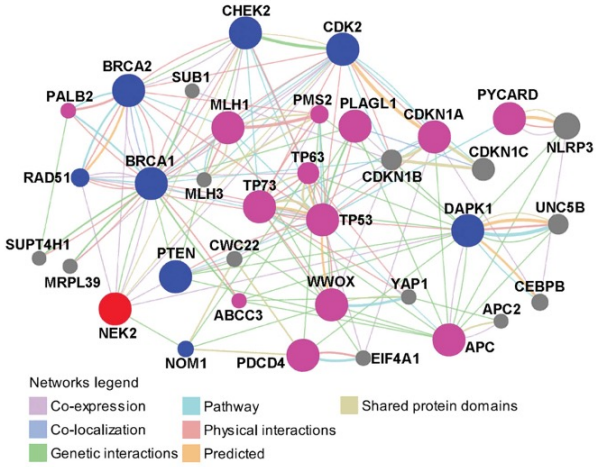
Internet-of-Things



Transportation



Molecular graph



Gene network

Web graph

(Figure credit) Web

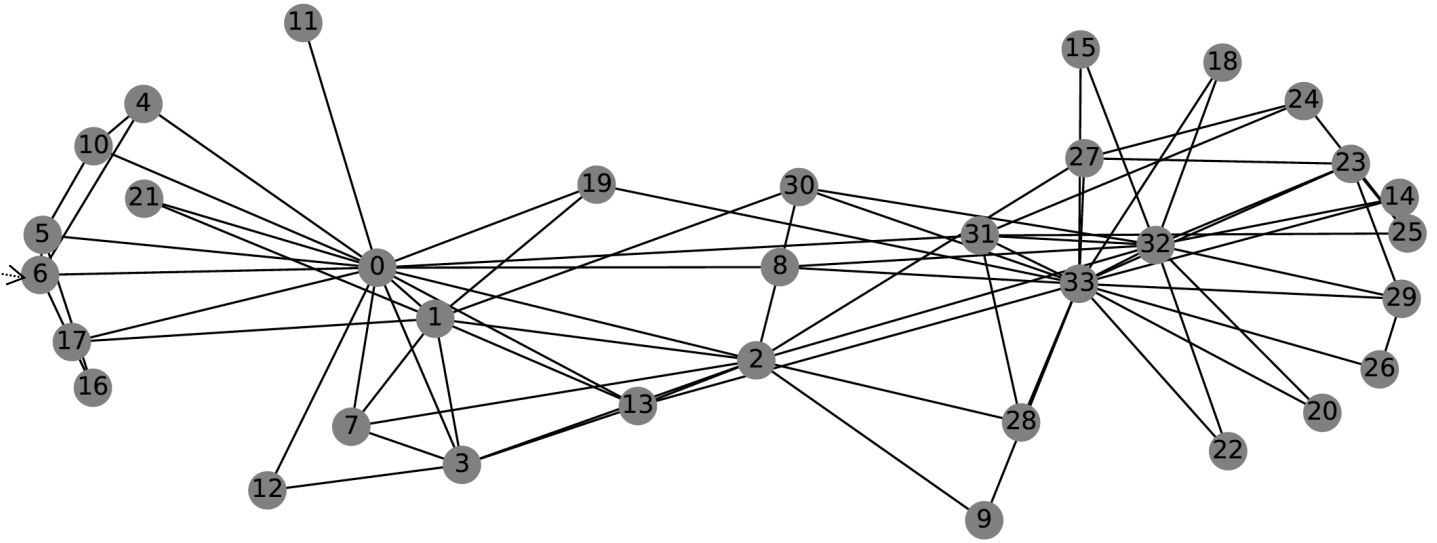
Types of graph

- Undirected / Directed
- Homogeneous network
- Heterogeneous network
  - Multiplex network
  - Bipartite graph

Formal definition of graphs

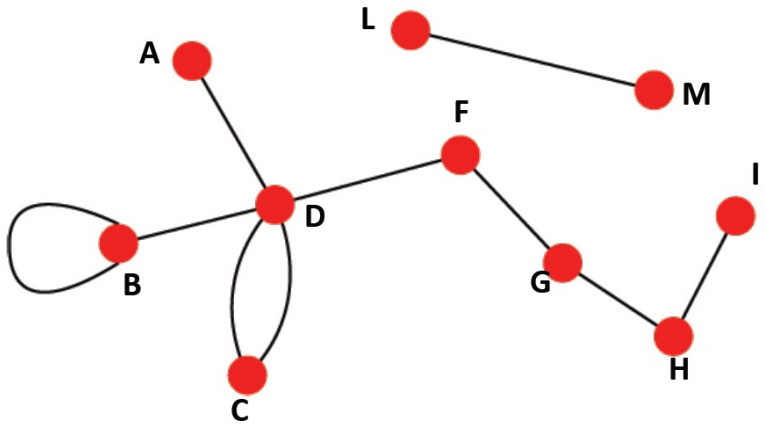
- A graph $G = (V, E)$ is defined by a set of nodes V and a set of edges E between these nodes
- An edge going from node $i \in V$ to node $j \in V$ is denoted as $(i, j) \in E$
- A convenient way to represent graphs is through an adjacency matrix $\boldsymbol{A} \in R^{|V| \times |V|}$
  - $\boldsymbol{A}_{ij} = 1$ if $(i, j) \in E$, and $\boldsymbol{A}_{ij} = 0$ otherwise
  - Some graphs have ***weighted*** edges, i.e., entries of $\boldsymbol{A}$ are arbitrary real-values rather than $\{0,1\}$
- Feature information $\boldsymbol{X} \in R^{|V| \times F}$

$$\boldsymbol{X_6} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}^T$$



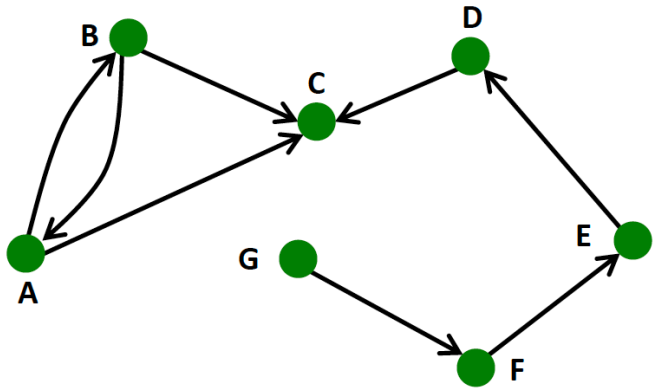
Undirected graph vs. directed graph

- **Undirected Graph**
  - Adjacency matrix A is **symmetric**
  - $(u, v) \in E \leftrightarrow (v, u) \in E$



- **Examples**
  - Collaborations
  - Friendship on Facebook

- **Directed Graph**
  - Adjacency matrix A is **not symmetric**
  - $(u, v) \in E \nleftrightarrow (v, u) \in E$



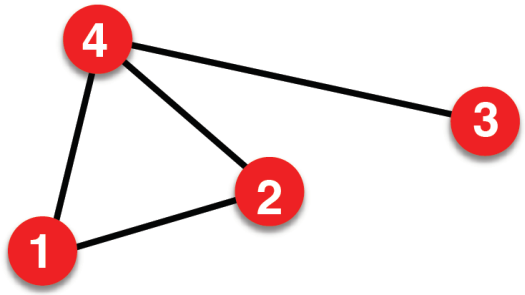
- **Examples**
  - Paper citation
  - Follow on Twitter

Adjacency matrix

$A_{ij} = 1$ if there is a link from node *i* to node *j*

$A_{ij} = 0$ otherwise

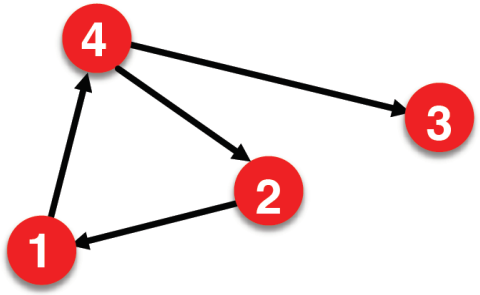
Undirected Graph



$$A_{ij} = A_{ji}$$
$$A_{ii} = 0$$

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Directed Graph



$$A_{ij} \neq A_{ji}$$
$$A_{ii} = 0$$

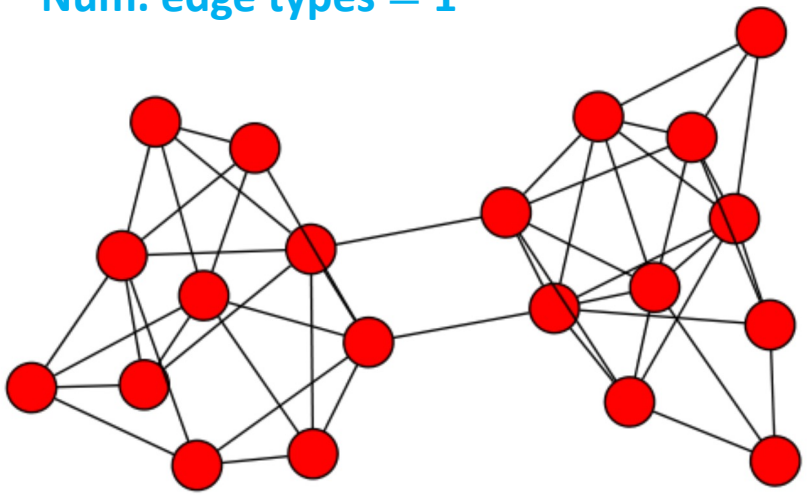
$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Homogeneous network

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

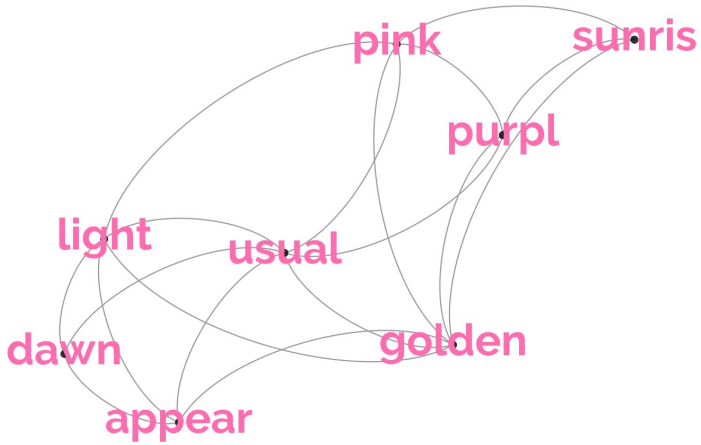
Num. node types = 1

Num. edge types = 1

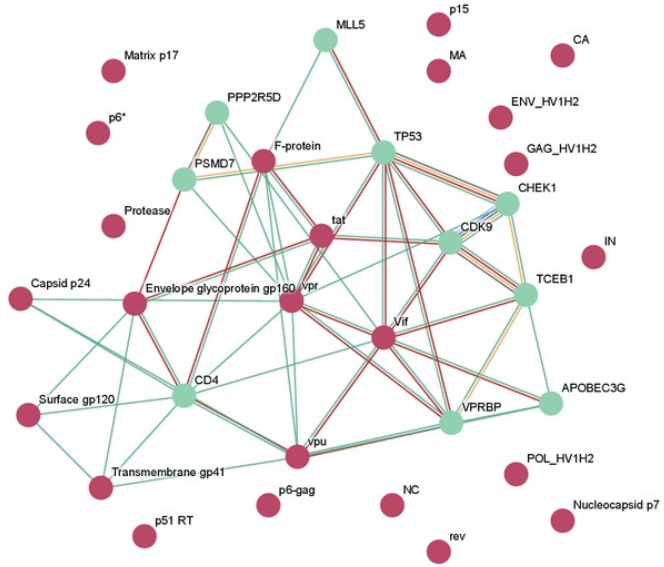


Homogeneous network

Social graph



Word cooccurrence graph



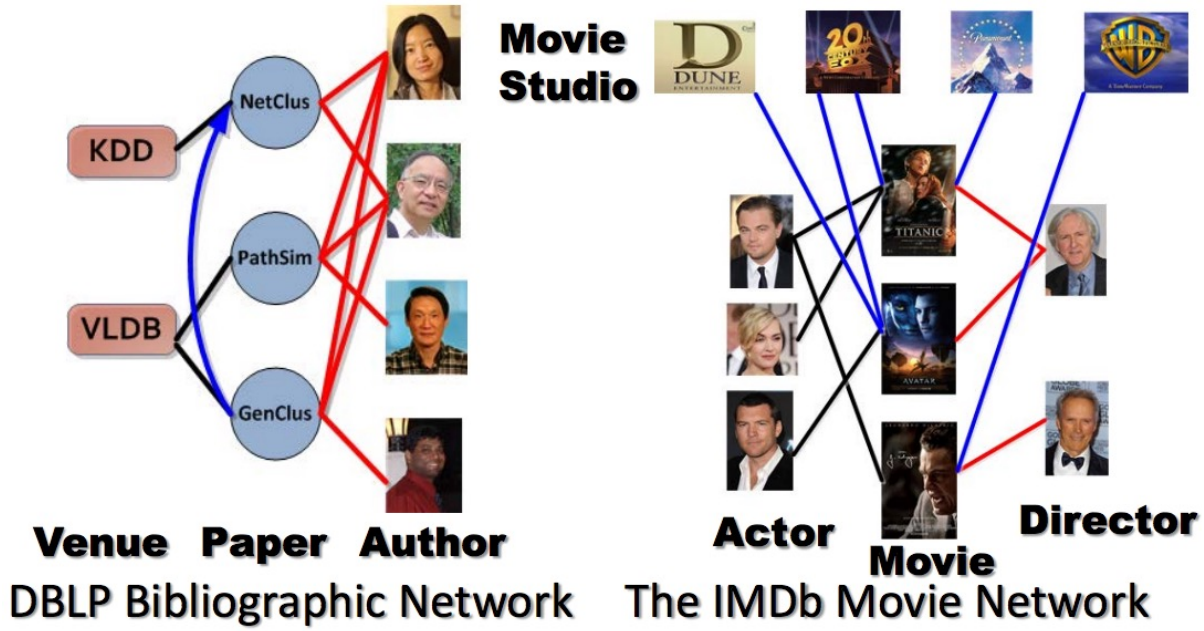
Protein-Protein Interaction Graph

(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

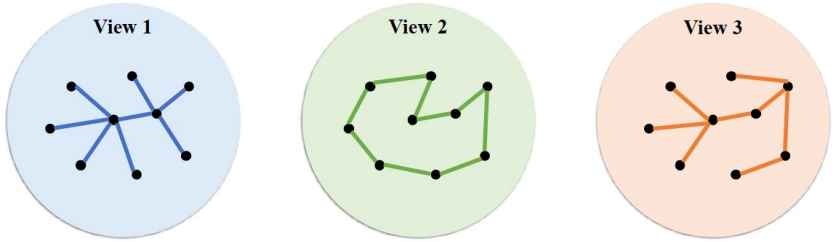
Heterogeneous network (HetNet)

- In reality a lot of graphs have **multiple types of nodes** and **multiple types of edges**
- Such networks are called **"heterogeneous network"**

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

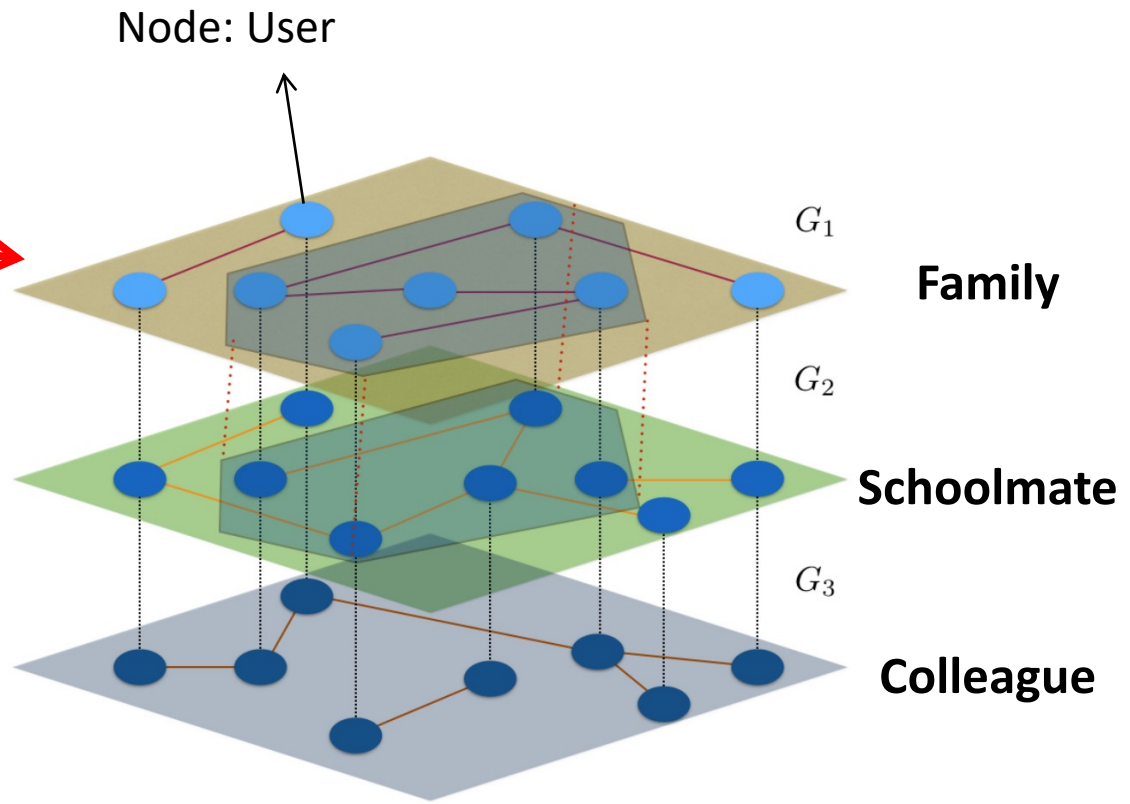


Multiplex (Multi-layer) network



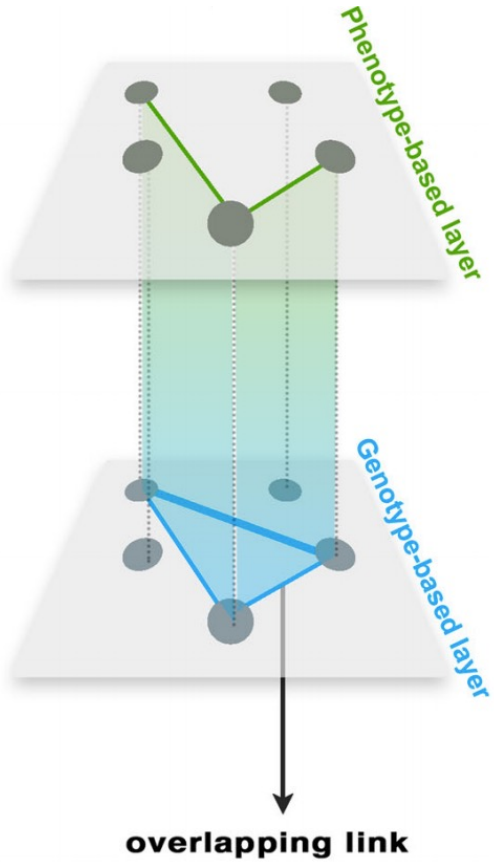
- A type of heterogeneous network
  - A single node type, multiple edge types
- **Example 1: Social network**
  - Relationship between users
- **Example 2: E-commerce**
  - Relationship between items
- **Example 3: Publication network**
  - Relationship between papers (Citation, share authors)
  - Relationship 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

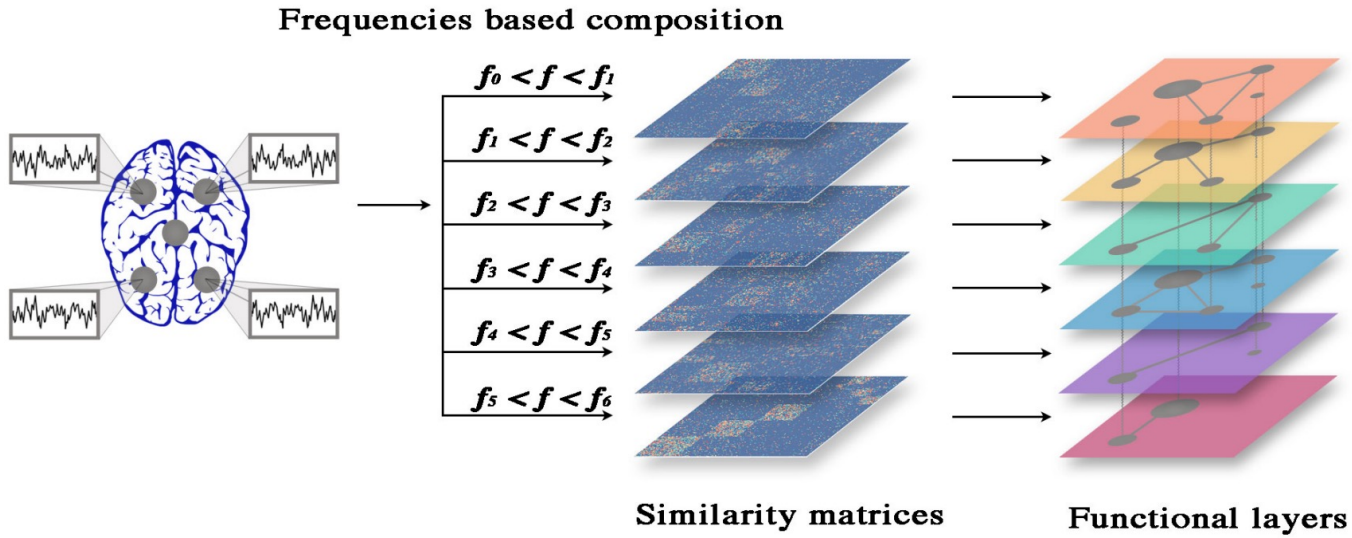
Other examples of multiplex network



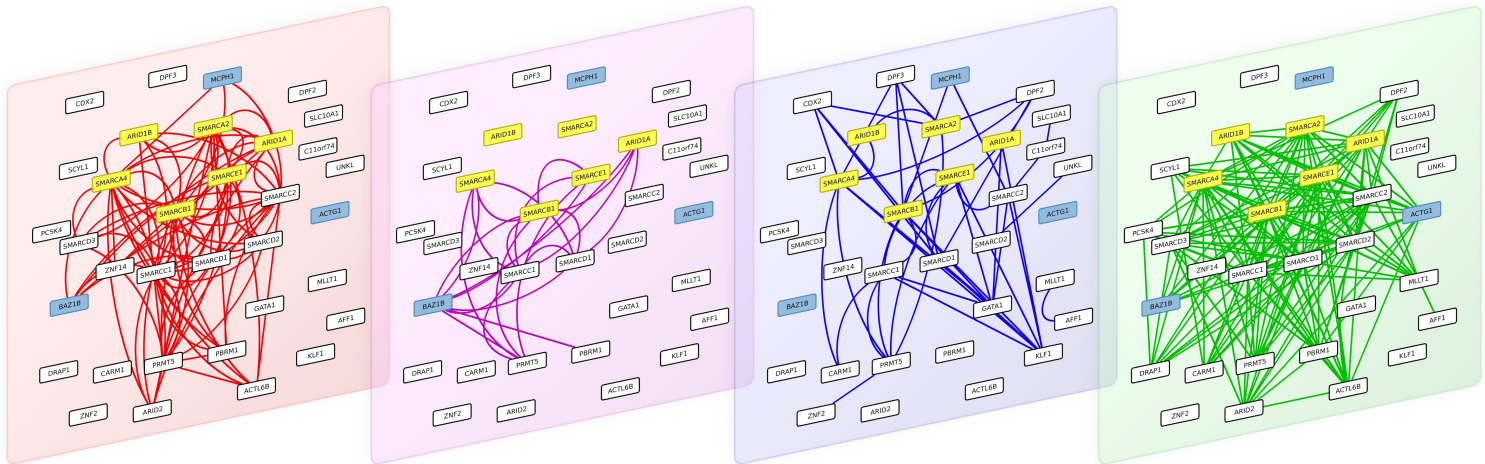
Share symptom

Share gene

Disease-Disease network



Frequency based composition of brain

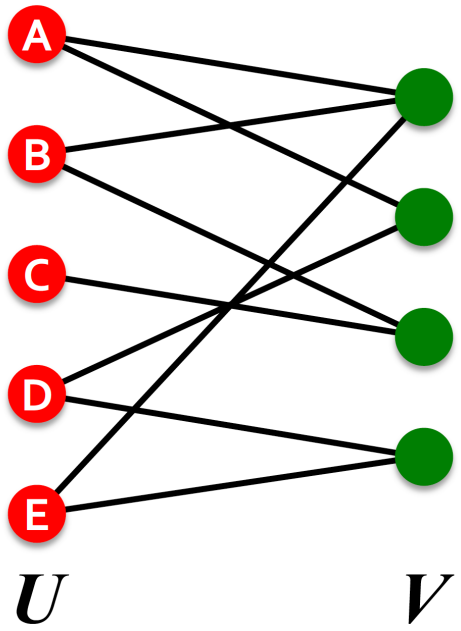


Multiplex biological networks

De Domenico, Manlio. "Multilayer modeling and analysis of human brain networks." *Giga Science* 6.5 (2017): gix004.
Halu, Arda, et al. "The multiplex network of human diseases." NPJ systems biology and applications 5.1 (2019): 1-12.
Didier, Gilles, Christine Brun, and Anaïs Baudot. "Identifying communities from multiplex biological networks." PeerJ 3 (2015): e1525.

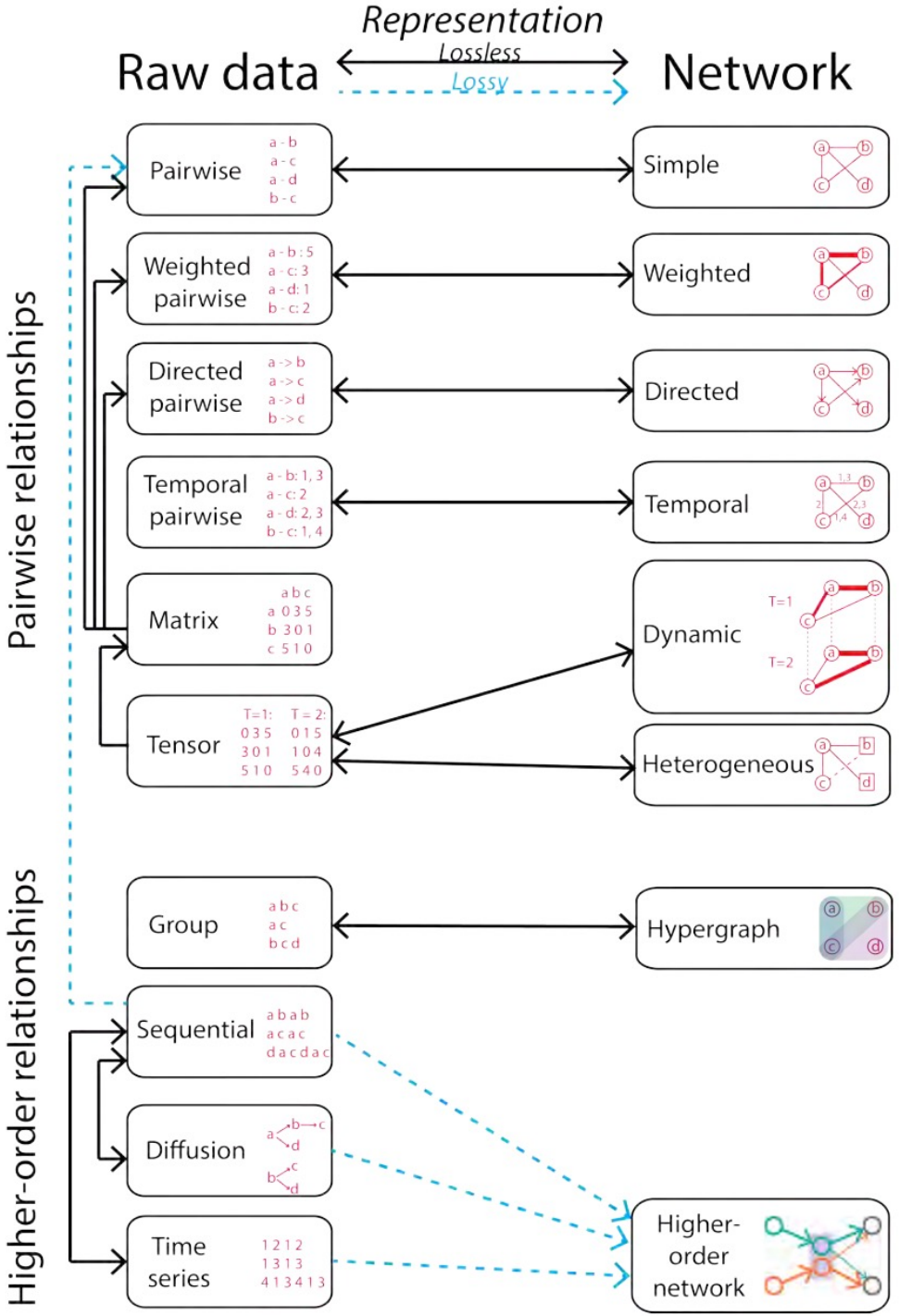
Bipartite graph

- Nodes can be divided into two disjoint sets ***U*** and ***V*** such that every link connects a node in ***U*** to one in ***V***
  - ***U*** and ***V*** are independent sets



- Examples
  - Authors-to-Papers (they authored)
  - Actors-to-Movies (they appeared in)
  - Users-to-Movies (they rated)
  - Recipes-to-Ingredients (they contain)

Overview: Data as Graphs

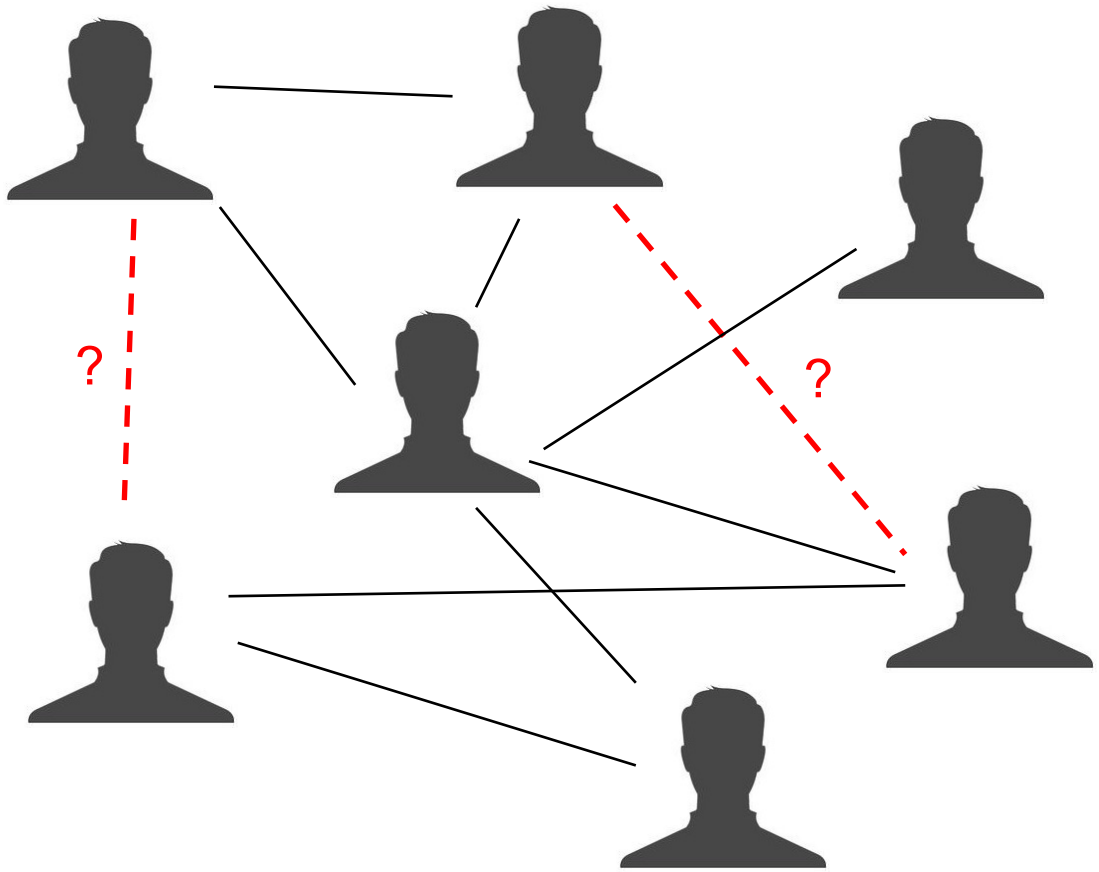


Xu, Jian. "Representing Big Data as Networks: New Methods a nd Insights." *arXiv preprint arXiv:1712.09648* (2017).

Machine learning on graphs

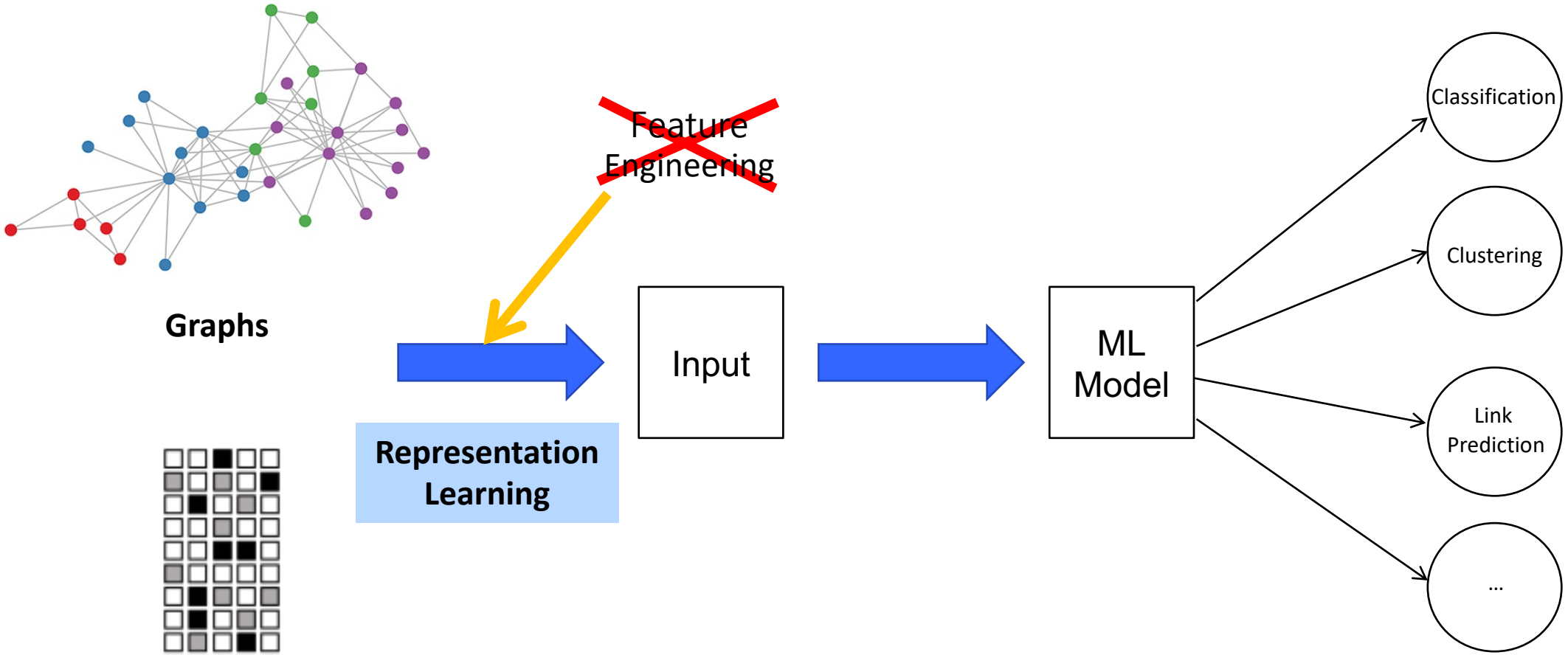
Classical ML tasks in graphs:

- 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



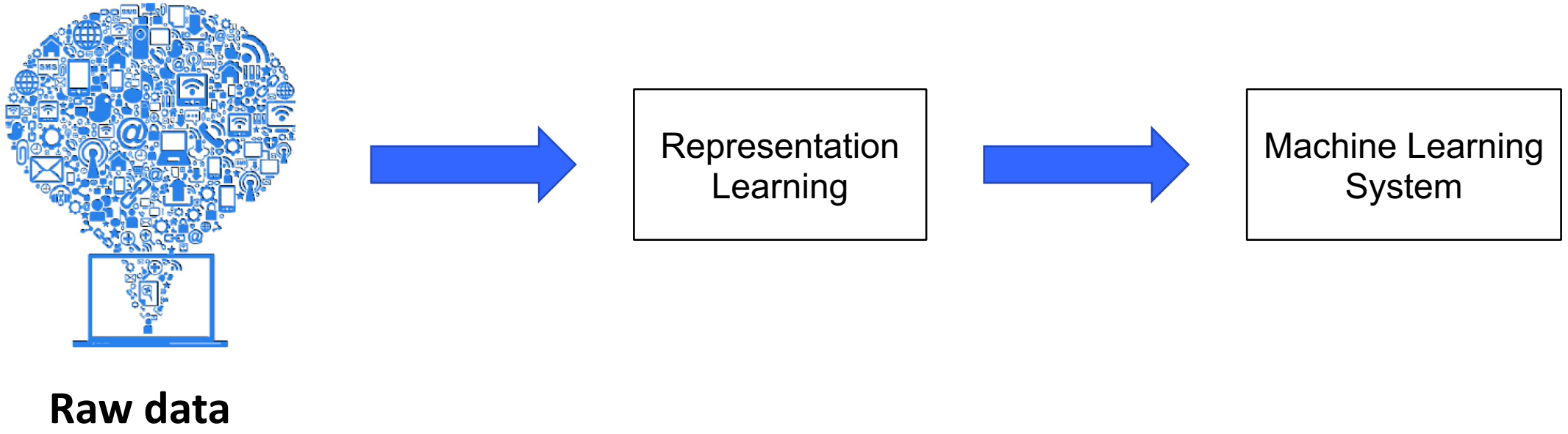
Link Prediction
(Friend Recommendation)

Machine learning on graphs



Machine learning in general

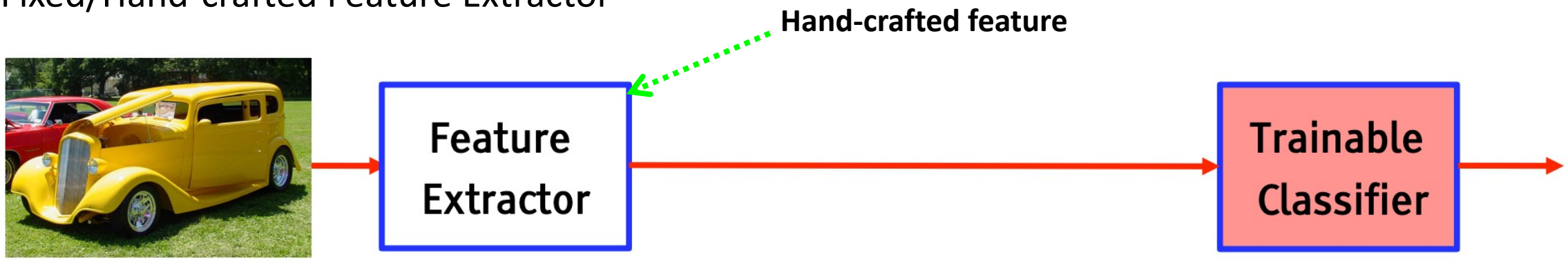
- **Machine Learning** = **Representation** + Objective + Optimization

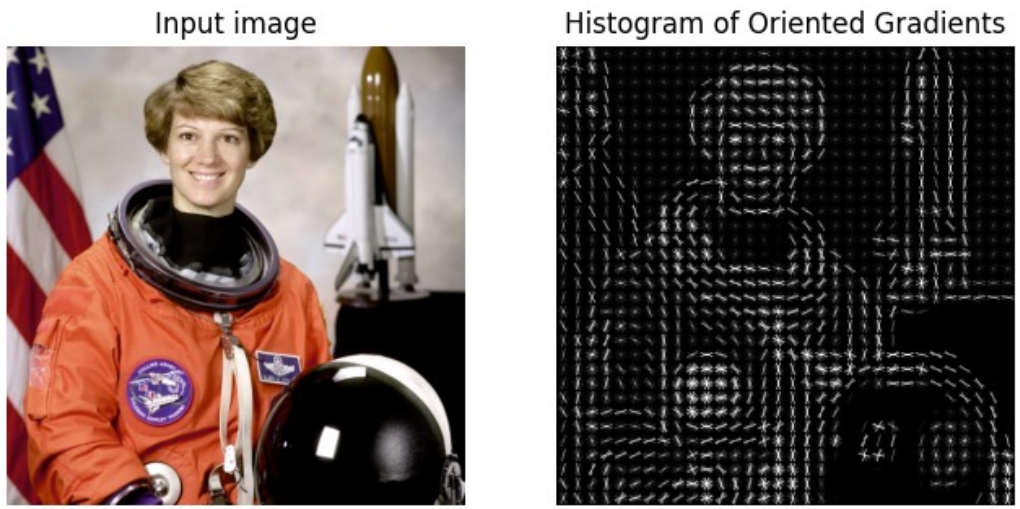


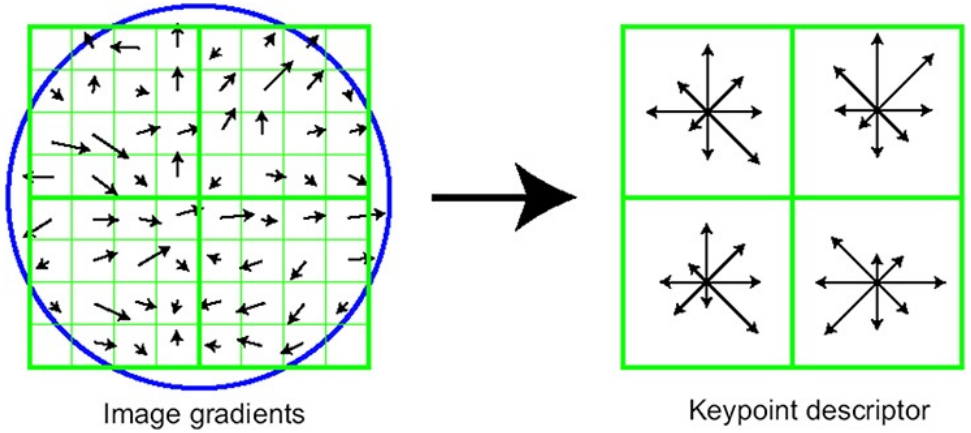
Good **Representation** is Essential for
Good **Machine Learning**

Traditional feature extraction for images

- Fixed/Hand-crafted Feature Extractor



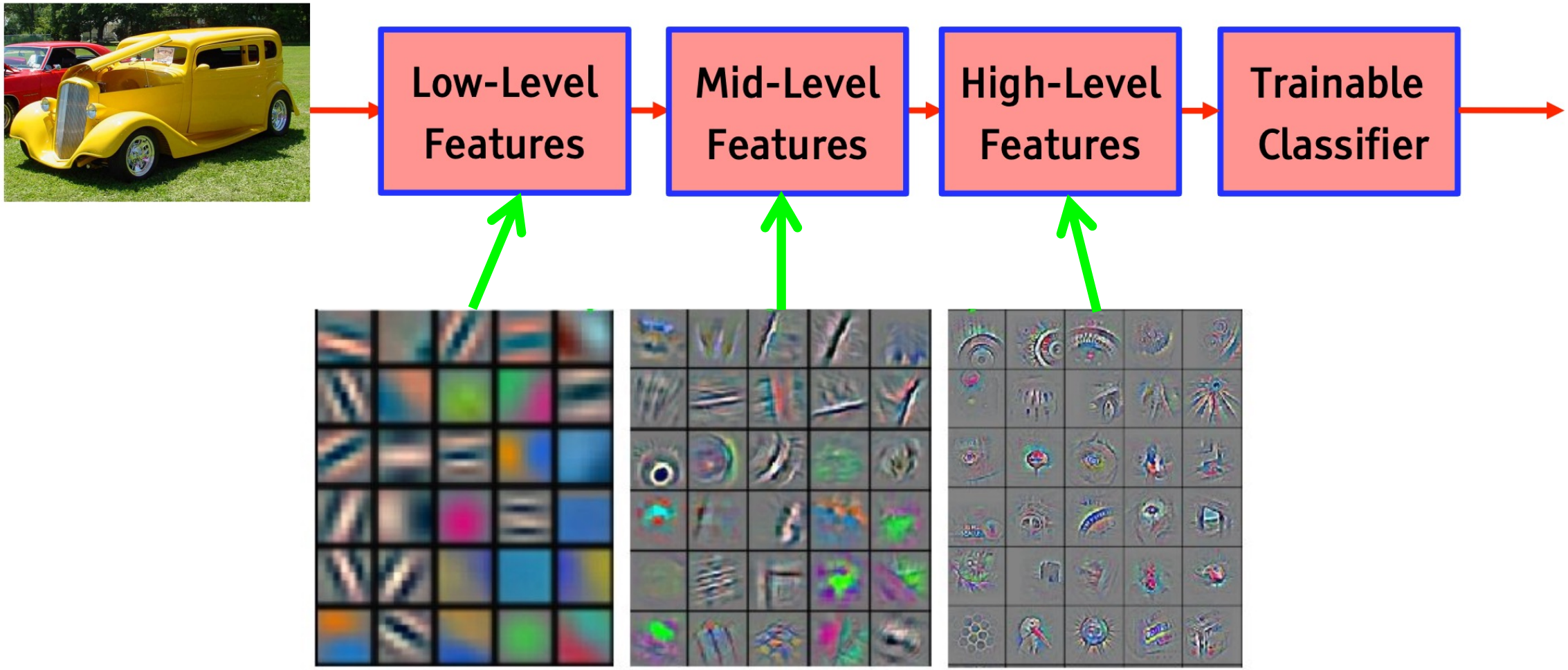




- Based on Yann Lecun's slides
- Lowe, David G. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision* 60.2 (2004): 91-110.

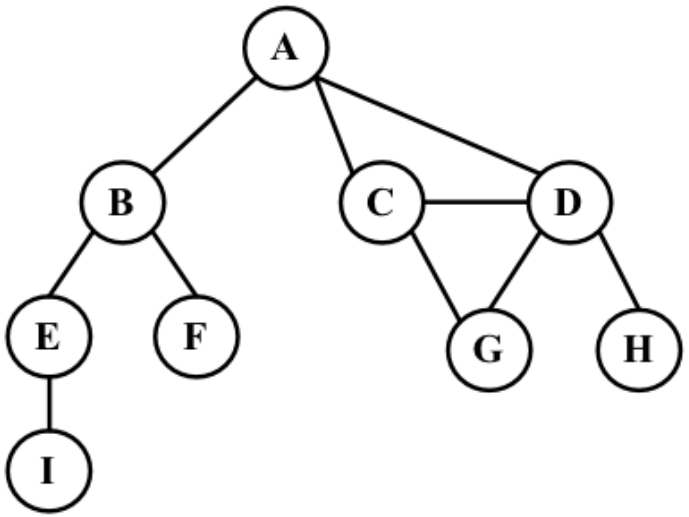
Machine (Deep) learning based Representation learning

- Multiple layers trained end-to-end



Based on Yann Lecun's slides

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"
- …

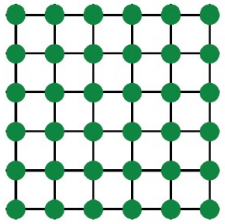
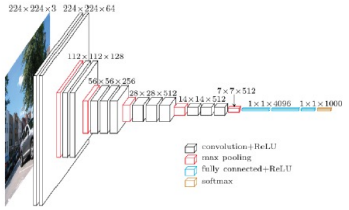
How to effectively and efficiently represent graphs is the key!

→ Deep learning-based approach?

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

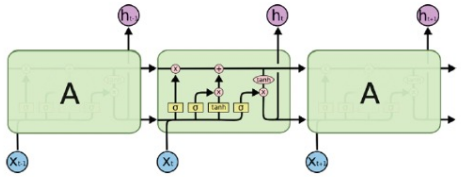
Challenges of Graph Representation Learning

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

  - RNNs for text/sequences …



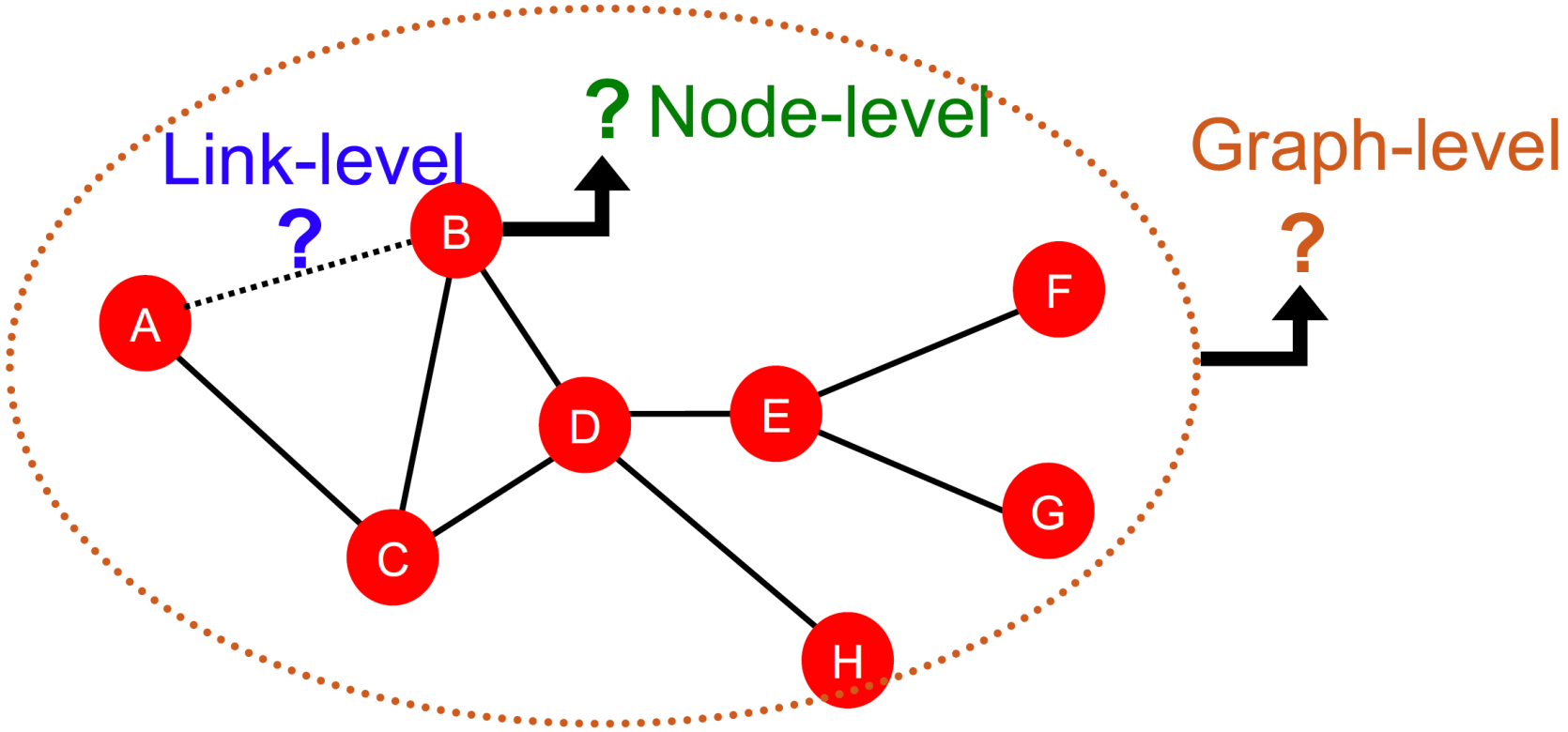


- **Graphs are very complex**
  - Arbitrary structures (no spatial locality like grids / no fixed orderings)
  - Heterogeneous: Directed/undirected, binary/weighted/typed, multimodal features
  - Large-scale: More than millions of nodes and billions of edges

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

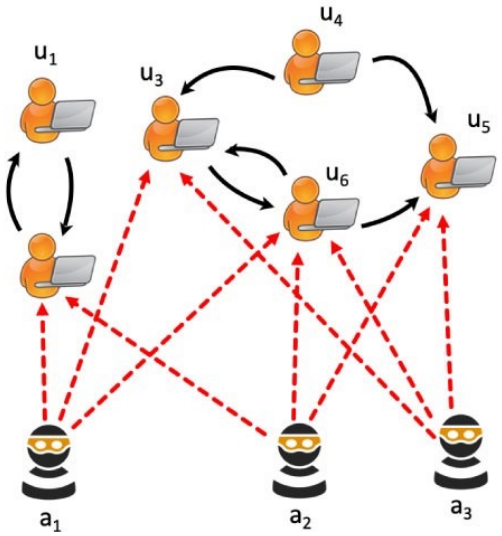
Typical tasks

- **Node-level** prediction
- **Edge-level** prediction
- **Graph-level** prediction

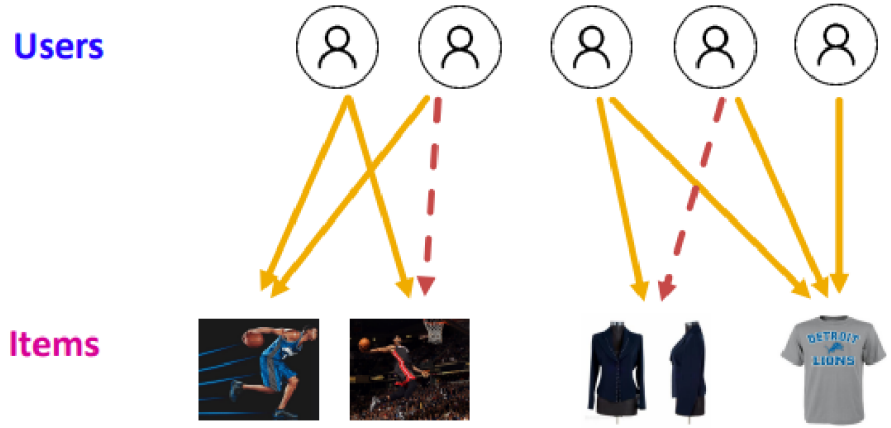


Typical tasks

- **Node-level tasks (or edge-level tasks)**
  - Node label classification, including node-level anomaly detection
  - Node label regression
  - Link label binary classification, i.e., link prediction
  - Link label multi-class classification, i.e., relation classification

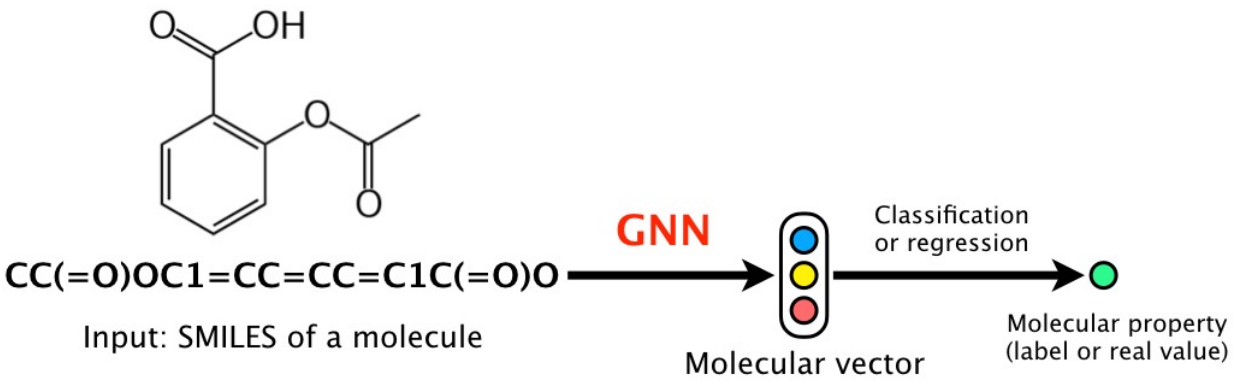


- Social network analysis (e.g., demographic info prediction)
- Spam / fraud detection (e.g., transaction networks)
- Link prediction (e.g., social networks, chemical interaction networks, biological networks, transportation networks)
- Knowledge graph population / completion / relation reasoning
- Recommender system (bipartite graphs, hyper-graphs)

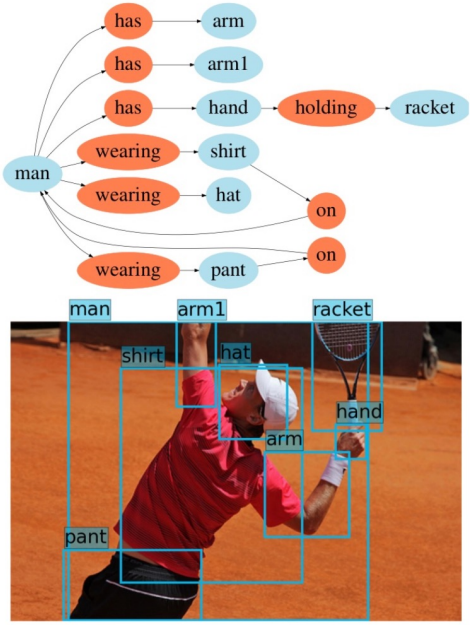


Typical tasks

- **Graph-level tasks**
  - Graph label classification
  - Graph label regression



- Molecular property prediction
- Drug discovery
- Scene understanding (i.e., objects graph)

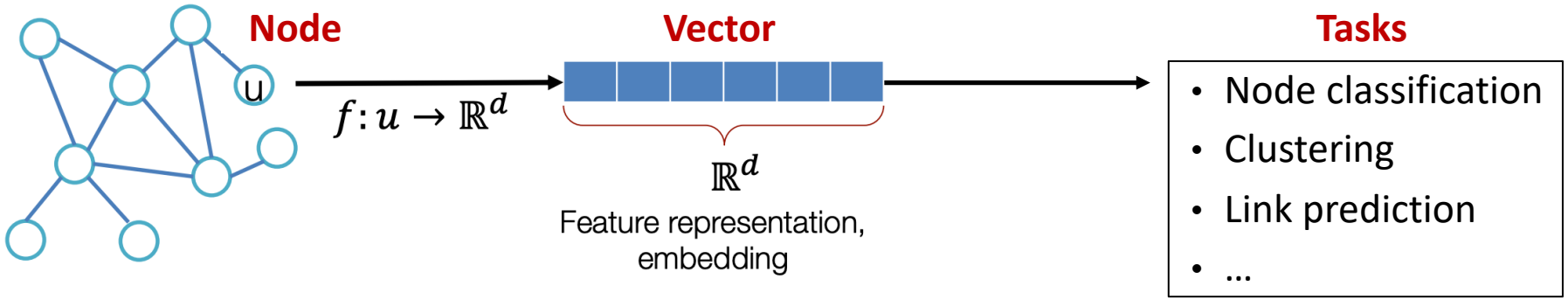


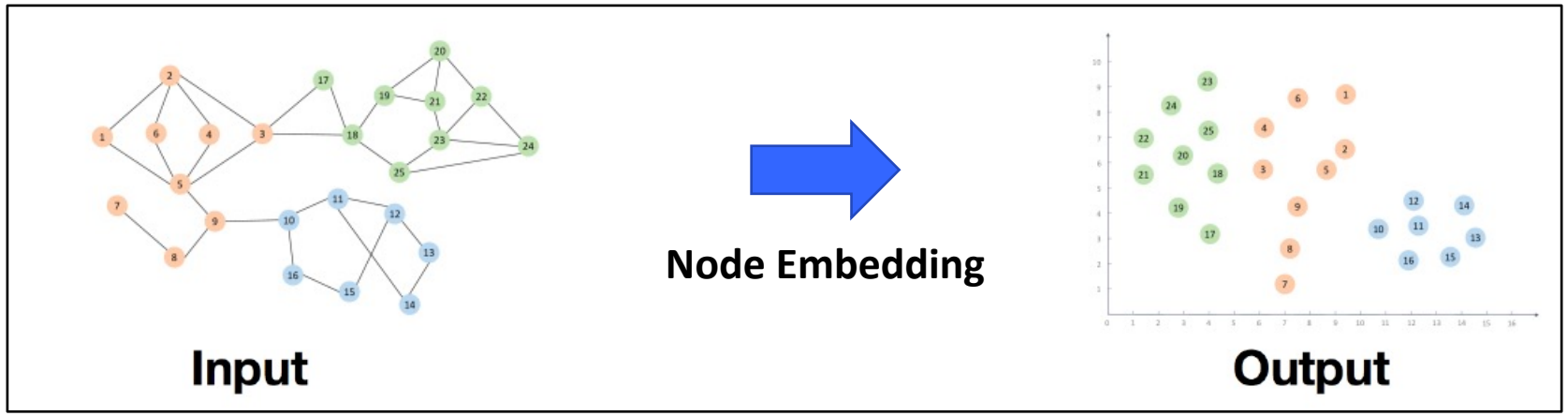
Outline

- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)
  - GNNs for Computer vision (Scene Understanding) (AAAI'23)

GRAPH REPRESENTATION LEARNING

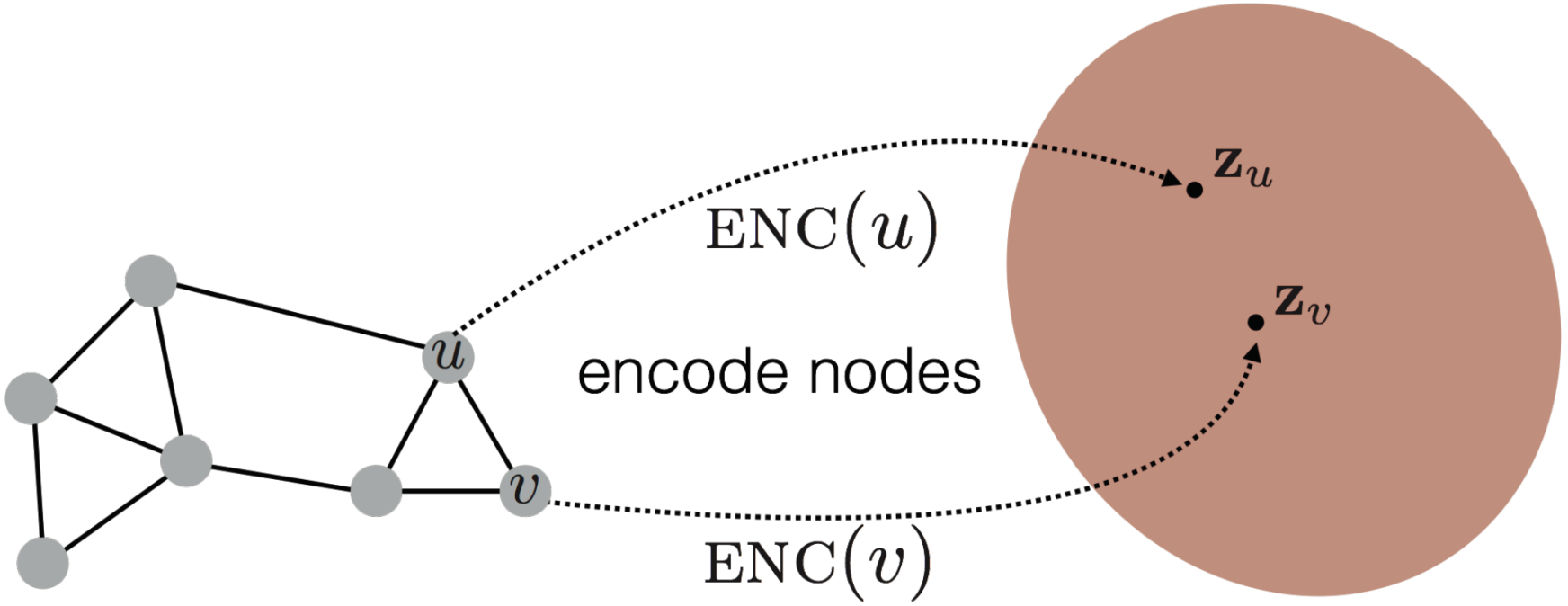
- **Goal**: Encode nodes so that **similarity in the embedding space** approximates **similarity in the original net work**
- **Similar nodes in a network have similar vector representations**





NODE EMBEDDING

- **Main idea:** Encode nodes so that **similarity in the embedding space** approximates **similarity in the graph**



Original graph

Embedding space (Latent space)

- **Two things to consider**
  - 1. How to encode nodes?
    - Encoder
  - 2. How to define similarity in the embedding space?
    - Decoder (Similarity function)

ENCODER

- Maps each node to a low-dimensional vector

$$ENC(v) = \mathbf{z}_v$$

$\mathbf{z}_v$: d-dimensional embedding vector

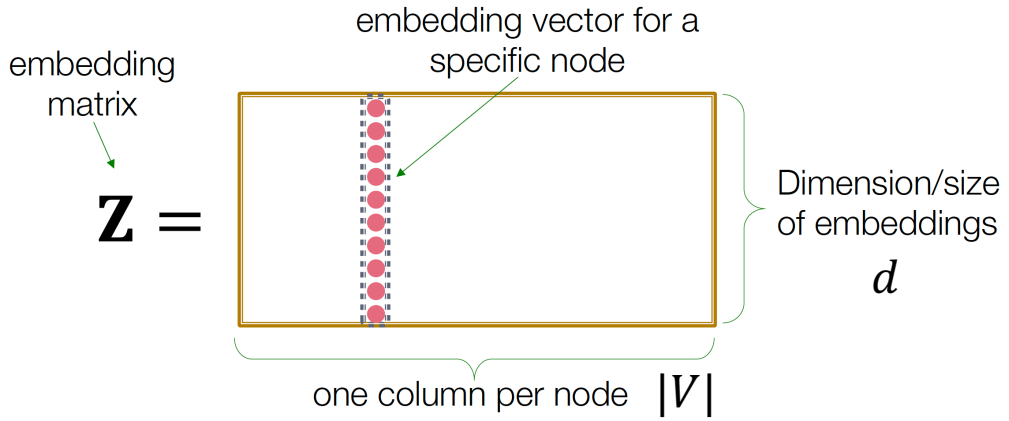
v: Node in the input graph

- **Simplest encoding approach**: Encoder is just an embedding-lookup (Shallow model)

$$ENC(v) = \mathbf{z}_v = \mathbf{Z} \cdot v$$

$\mathbf{Z} \in R^{d \times |V|}$

$v \in I^{|V|}$



$v =$ ex) $node\ 4$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Each node is assigned a unique embedding vector (i.e., we directly optimize the embedding of each node)

DECODER (SIMILARITY FUNCTION)

- Specifies how the **relationships in the original graph** map to **relationships in embedding space**

Relationships in the original graph

$similarity\ (u, v)$

Similarity between node u and node v in the original network

$\approx$

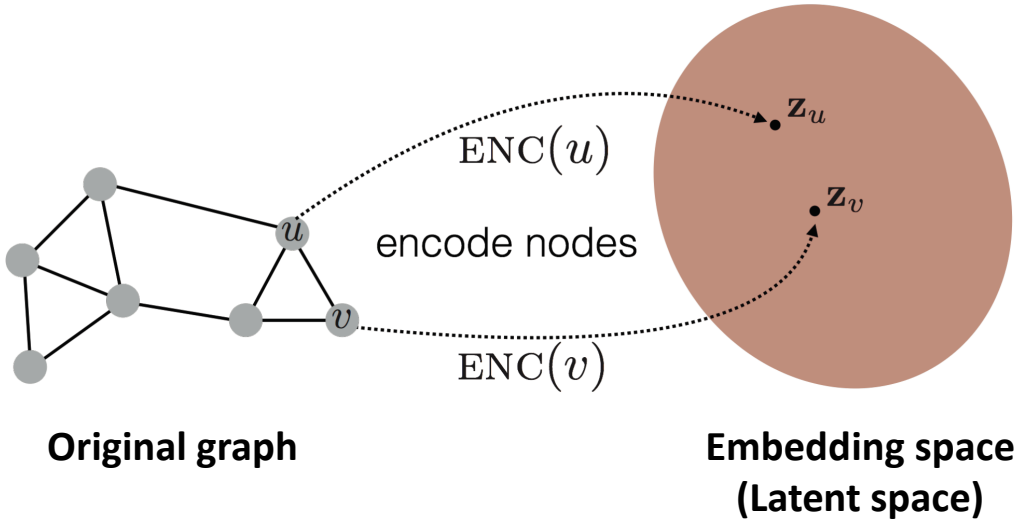
Relationships in embedding space

$\mathbf{z}_v^T \mathbf{z}_u$

Dot product between embeddings of node u and node v

ENCODER + DECODER FRAMEWORK

- **Encoder**: Embedding look-up (Shallow model)
  - Deep encoders (GNNs) later in the lecture
- **Decoder**: Based on dot product



Objective

Maximize $\boldsymbol{z}_v^T \boldsymbol{z}_u$ for node pairs (u, v) that are **similar**

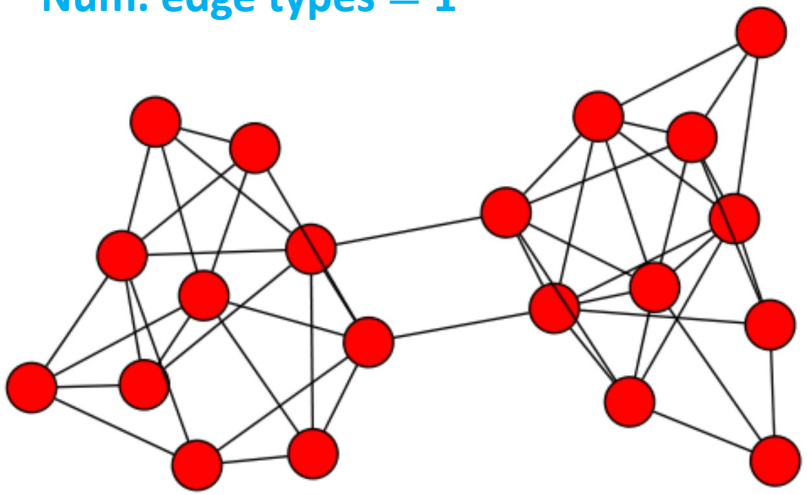
- **How can we define node similarity?**
- **Possible choice**
  - Are two nodes linked?
  - Do they share neighbors?
  - Do they have similar structural roles?
  - …

RECALL: HOMOGENEOUS NETWORK

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

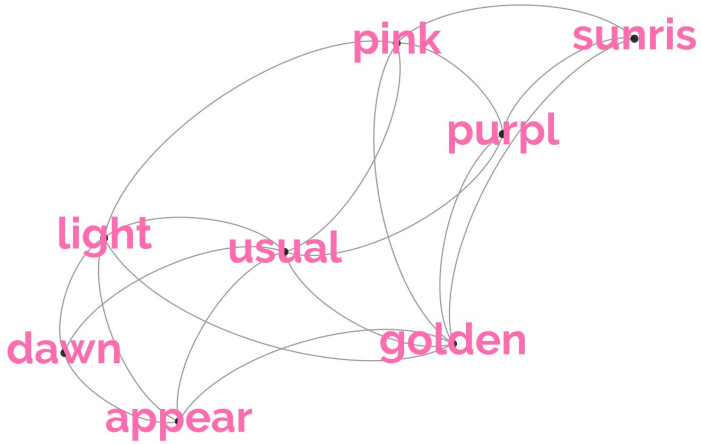
Num. node types = 1

Num. edge types = 1

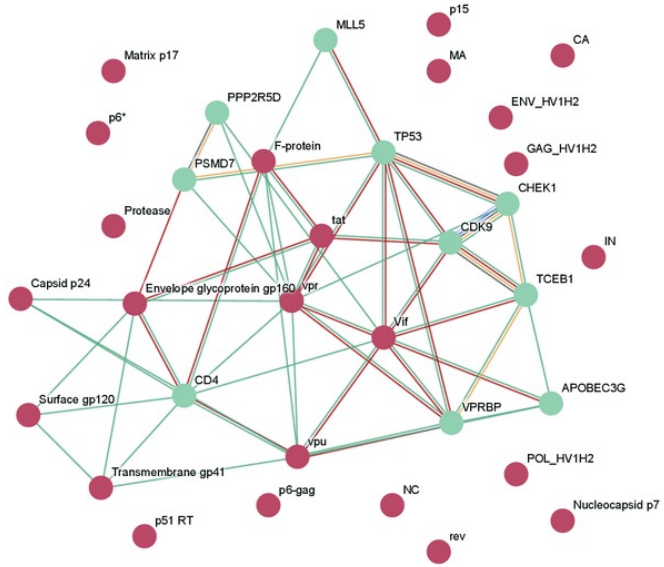


Homogeneous network

Social graph



Word cooccurrence graph

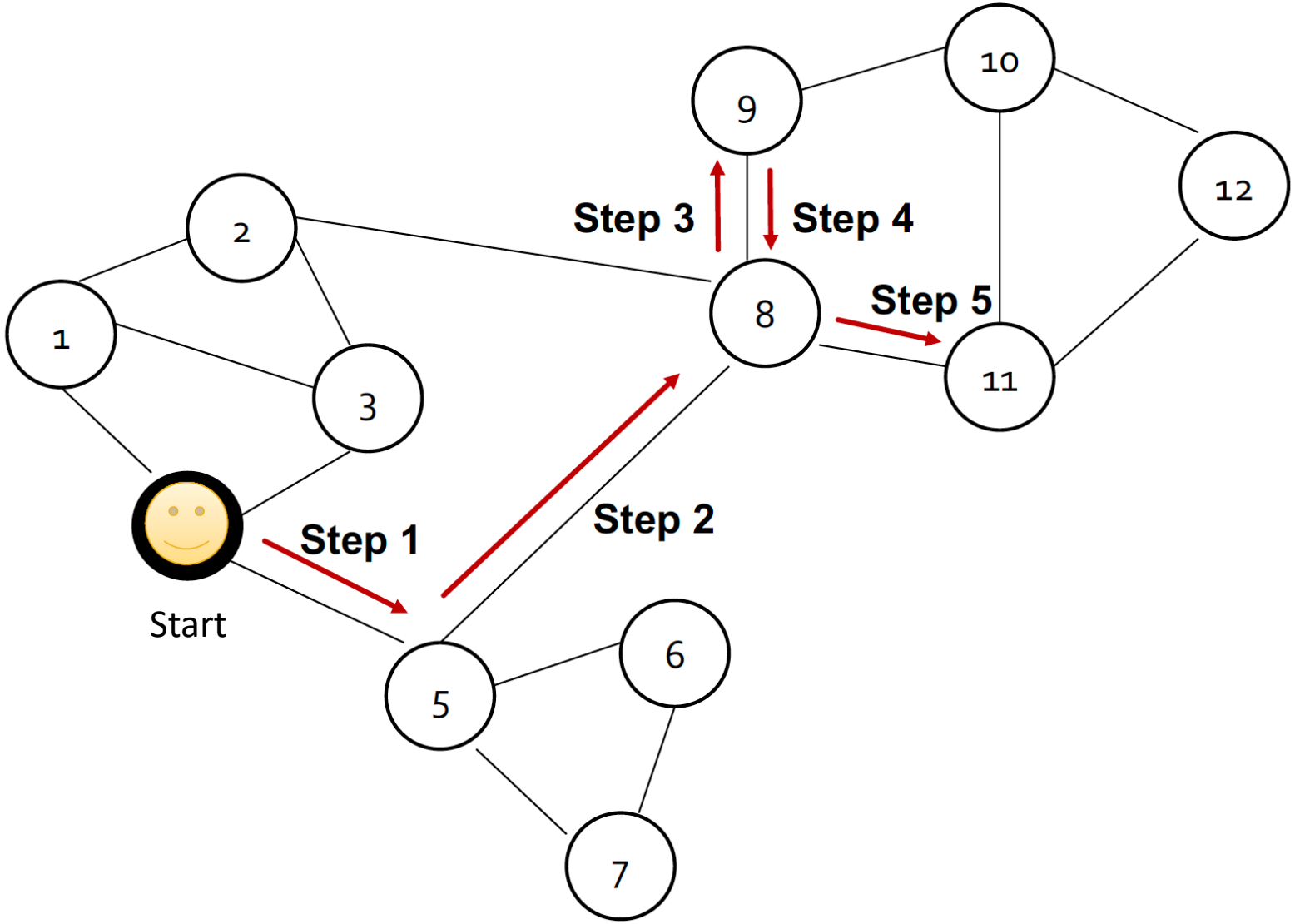


Protein-Protein Interaction Graph

(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

WHAT IS RANDOM WALK?

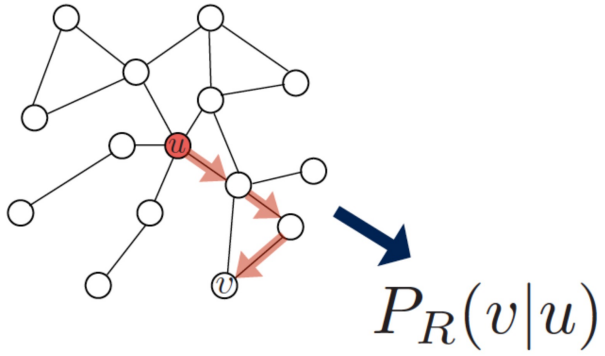
- **Given a graph and a starting node,**
  - 1. Select a neighbor of it at random,
  - 2. Move to this neighbor
  - Repeat 1,2
- Example of random walk
  - Start → 5 → 8 → 9 → 8 → 11
  - (Random) Sequence of nodes



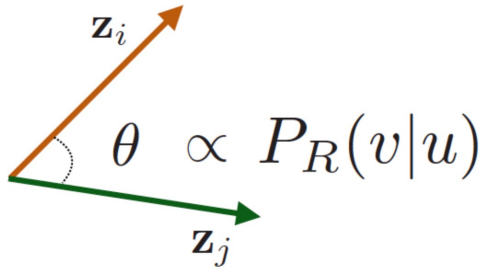
RANDOM WALK-BASED NODE EMBEDDINGS: OVERVIEW

- **Idea**: Learn node embedding such that nearby nodes in the graph are close together in the embedding space
- **Q.** Given a node u, how do we define nearby nodes?
- **A.** Through **random walk!**

- **Step 1.** Estimate the probability of visiting node v on a random walk starting from node u using some random walk strategy R



- **Step 2.** Optimize embeddings to encode these random walk statistics
  - e.g., If two nodes co-occur, maximize their similarity



cf) Dot product = cosine similarity, if node embeddings are unit vectors

Why random walk?

Random walk can reflect both **local** and **high-order** neighborhood information

RANDOM WALK-BASED NODE EMBEDDINGS: DETAILED ALGORITHM

- **Given**: $G = (V, E)$
- **Goal**: To learn a mapping function $f: u \rightarrow R^d$ for $u \in V$
  - $f(u) = \boldsymbol{z}_u \in R^d$
- **Step 1**: Run fixed-length random walks starting from each node u in the graph using some random walk str ategy R
- **Step 2**: For each node u collect $N_R(u)$, the multiset of nodes visited on random walks starting from u
  - $N_R(u)$: Neighboring nodes of node u under random walk strategy R
- **Step 3**: Optimize embeddings according to the following objective
  - Objective: Given node u, predict its neighbors $N_R(u)$

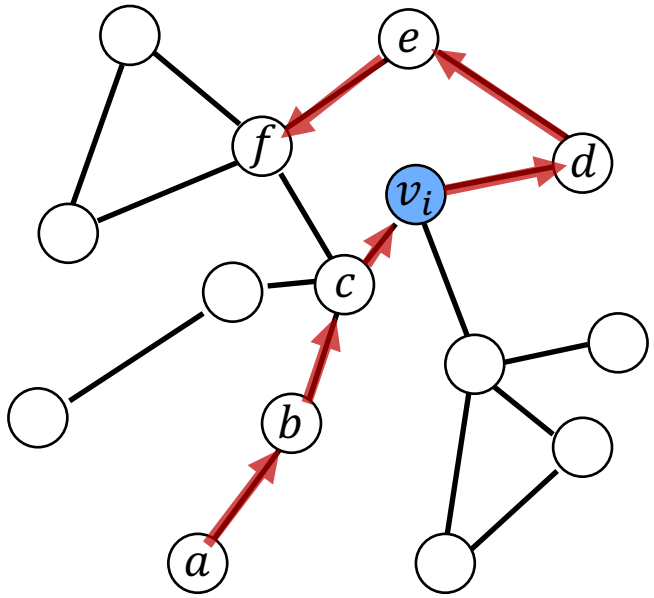
$$\max_f \sum_{u \in V} \log P(N_R(u) | \boldsymbol{z}_u)$$

(Maximum likelihood objective)

Given node u, we aim to maximize the probability of its neighboring nodes

i.e., we want to learn embedding of node u that is predictive of its neighboring nodes

HOW TO DEFINE NEIGHBORING NODES?



Random walk strategy	R
Example sequence	$a \rightarrow b \rightarrow c \rightarrow v_i \rightarrow d \rightarrow e \rightarrow f$
Window size=2	$a \rightarrow b \rightarrow c \rightarrow v_i \rightarrow d \rightarrow e \rightarrow f$
Center node	v_i
Neighborhood	$N_R(v_i) = b, c, d, e$

RANDOM WALK-BASED NODE EMBEDDINGS: OPTIMIZATION

$$\max_f \sum_{u \in V} \log P(N_R(u)|\boldsymbol{z}_u)$$

Equivalent

$$L = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\boldsymbol{z}_u))$$

- **Intuition:** Optimize embeddings $\boldsymbol{z}_u$ to maximize the likelihood of random walk co-occurrences
- **Approach:** Parameterize $P(v|\boldsymbol{z}_u)$ using **softmax**

$$P(v|\boldsymbol{z}_u) = \frac{\exp(\boldsymbol{z}_u^T \boldsymbol{z}_v)}{\sum_{j \in V} \exp(\boldsymbol{z}_u^T \boldsymbol{z}_j)}$$

$$L = \sum_{u \in V} \sum_{v \in N_R(u)} -\log\left(\frac{\exp(\boldsymbol{z}_u^T \boldsymbol{z}_v)}{\sum_{j \in V} \exp(\boldsymbol{z}_u^T \boldsymbol{z}_j)}\right)$$

Sum over all nodes u

Sum over nodes v seen on random walks starting from u

Predicted probability of u and v cooccurring on random walk

Optimizing random walk embeddings = Finding embeddings $\boldsymbol{z}_u$ that minimizes the loss L

NEGATIVE SAMPLING

- $O(|V|^2)$ complexity
- We can approximate this normalization term

- **But, optimizing the loss L is expensive!**

$$L = \sum_{u \in V} \sum_{v \in N_R(u)} -\log\left(\frac{\exp(\boldsymbol{z}_u^T \boldsymbol{z}_v)}{\sum_{j \in V} \exp(\boldsymbol{z}_u^T \boldsymbol{z}_j)}\right)$$

$$\log\left(\frac{\exp(\boldsymbol{z}_u^T \boldsymbol{z}_v)}{\sum_{j \in V} \exp(\boldsymbol{z}_u^T \boldsymbol{z}_j)}\right) \approx \log\left(\sigma(\boldsymbol{z}_u^T \boldsymbol{z}_v)\right) - \sum_{j=1}^{k} \log(\sigma(\boldsymbol{z}_u^T \boldsymbol{z}_j)), \quad j \sim P_V$$

https://arxiv.org/pdf/1402.3722.pdf

- $\sigma(x) = \frac{1}{1+e^{-x}}$ (Sigmoid function)
  - Makes each term a "probability" between 0 and 1
- P_v: Random distribution over nodes

- **Instead of normalizing w.r.t. all nodes, just normalize against k random "negative samples" j**
- How do we sample from P_V to help the training process?
  - Sample k negative nodes considering the degree of each node

RANDOM WALK-BASED NODE EMBEDDINGS: OPTIMIZATION

- After we obtained the objective function, **how do we optimize (minimize) it?**

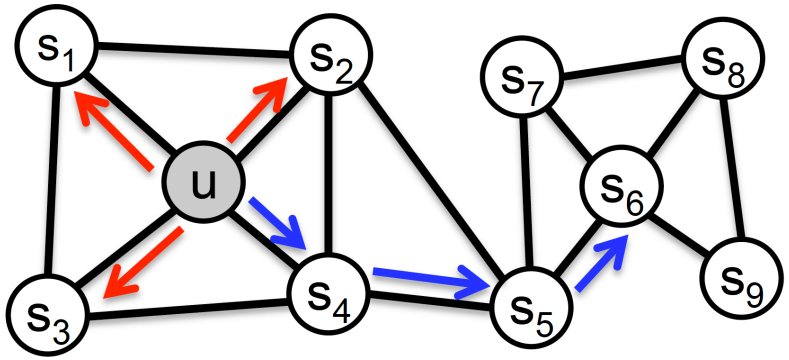
$$L = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\boldsymbol{z}_u))$$

- **Gradient descent:** A simple and the most common way to minimize L
  - Step 1: Randomly initialize $\boldsymbol{z}_i$ for all $i \in V$
  - Step 2: Iterate until convergence
    - For all $i \in V$, compute the derivative w.r.t. the loss L, i.e., $\frac{\partial L}{\partial \boldsymbol{z}_i}$
    - For all $i \in V$, update $\boldsymbol{z}_i \leftarrow \boldsymbol{z}_i - \eta \frac{\partial L}{\partial \boldsymbol{z}_i}$

- **Stochastic Gradient descent:** Instead of evaluating gradients over all examples, evaluate for a **single** node
  - Step 1: Randomly initialize $\boldsymbol{z}_i$ for all $i \in V$
  - Step 2: Iterate until convergence: $L^{(u)} = \sum_{v \in N_R(u)} -\log(P(v|\boldsymbol{z}_u))$
    - Sample a node i, for all $j \in N_R(i)$ compute the derivative w.r.t. the loss L, i.e., $\frac{\partial L^{(i)}}{\partial \boldsymbol{z}_j}$
    - For all $j \in N_R(i)$, update $\boldsymbol{z}_j \leftarrow \boldsymbol{z}_j - \eta \frac{\partial L^{(i)}}{\partial \boldsymbol{z}_j}$

NODE2VEC: BIASED WALKS

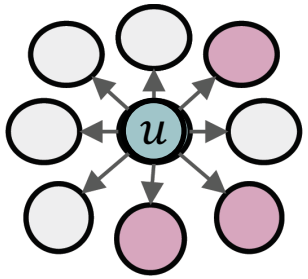
- **Idea**: Use flexible, biased random walks that can trade off between **local** and **global** views of the network
  - Deepwalk's simple random walk mainly focuses on the global view



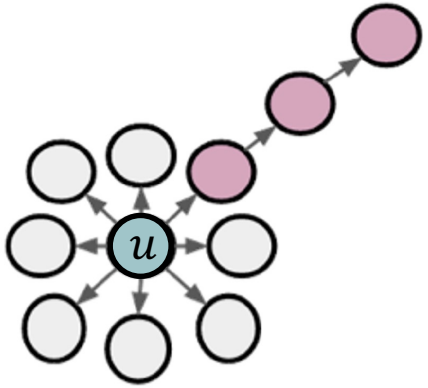
→ BFS (Breadth First Search)

→ DFS (Depth First Search)

- **Two strategies to define a neighborhood** $N_R(u)$ of node u: BFS and DFS
- Example: Walk of length 3 from node u
  - $N_{BFS}(u) = \{s_1, s_2, s_3\}$ **Local** microscopic view
  - $N_{DFS}(u) = \{s_4, s_5, s_6\}$ **Global** macroscopic view



BFS

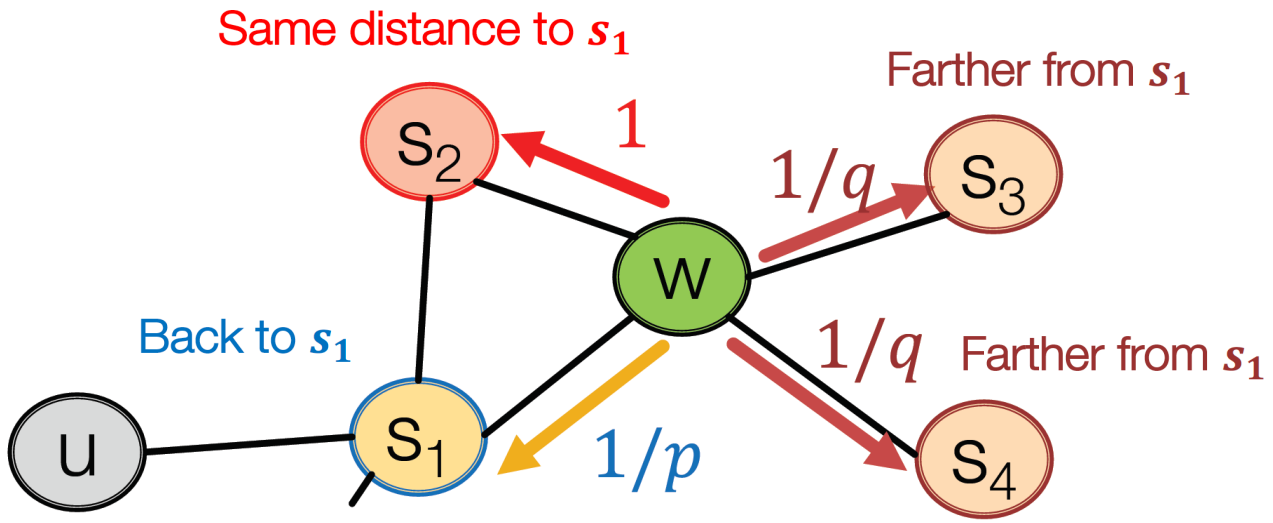


DFS

How can we interpolate between BFS and DFS?

NODE2VEC: INTERPOLATING BFS AND DFS

- Biased fixed-length random walk R starting fro m node u generates neighborhood $N_R(u)$
- **Two parameters to control the interpolation**
  - **Return parameter** p
    - Return back to the previous node
  - **In-out parameter** q
    - Moving outwards (DFS) vs. inwards (BFS)
    - Intuitively, q is the "ratio" of BFS vs. DFS



Current situation

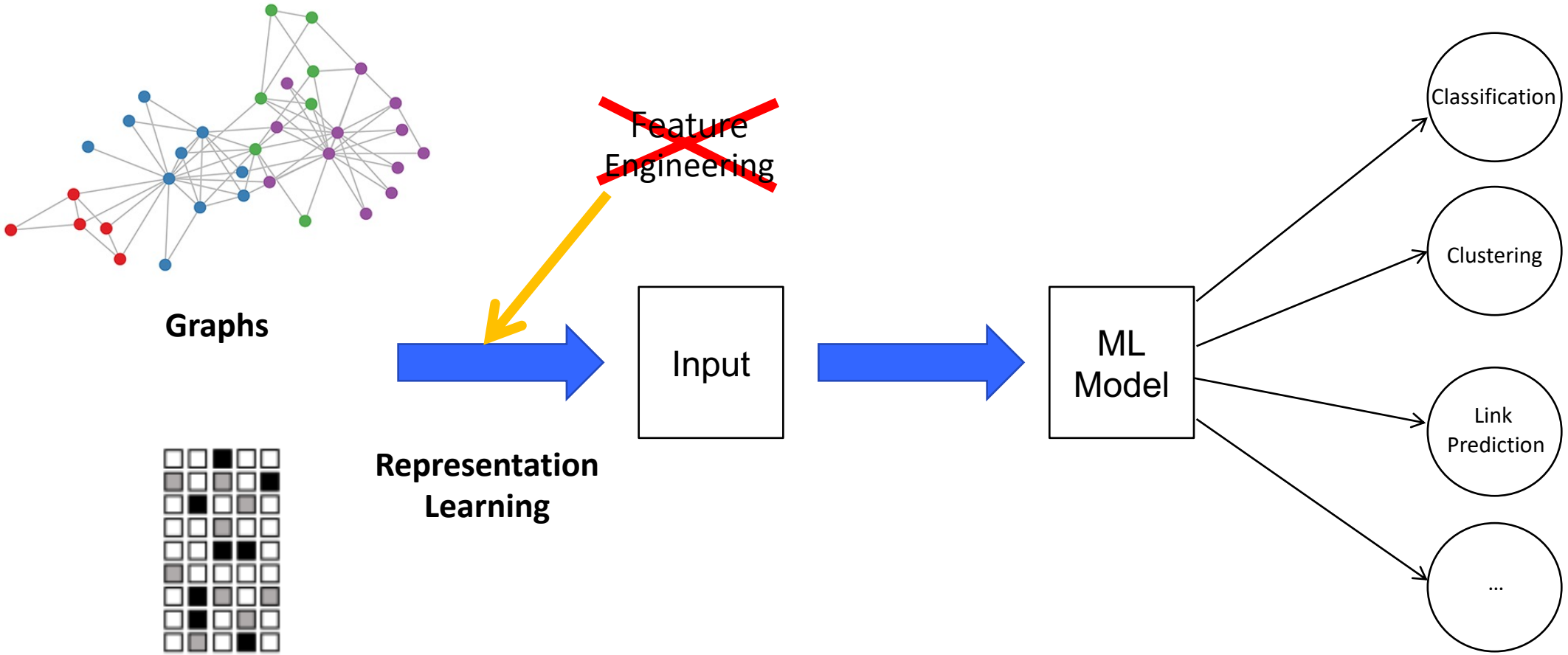
- Random walk that started from node u just traversed edge (s_1, w) and is now at w
- At this point, neighbors of w can be s_1, s_2, s_3 or s_4

Idea of biased random walk
Remember where the walk came from!

OUTLINE

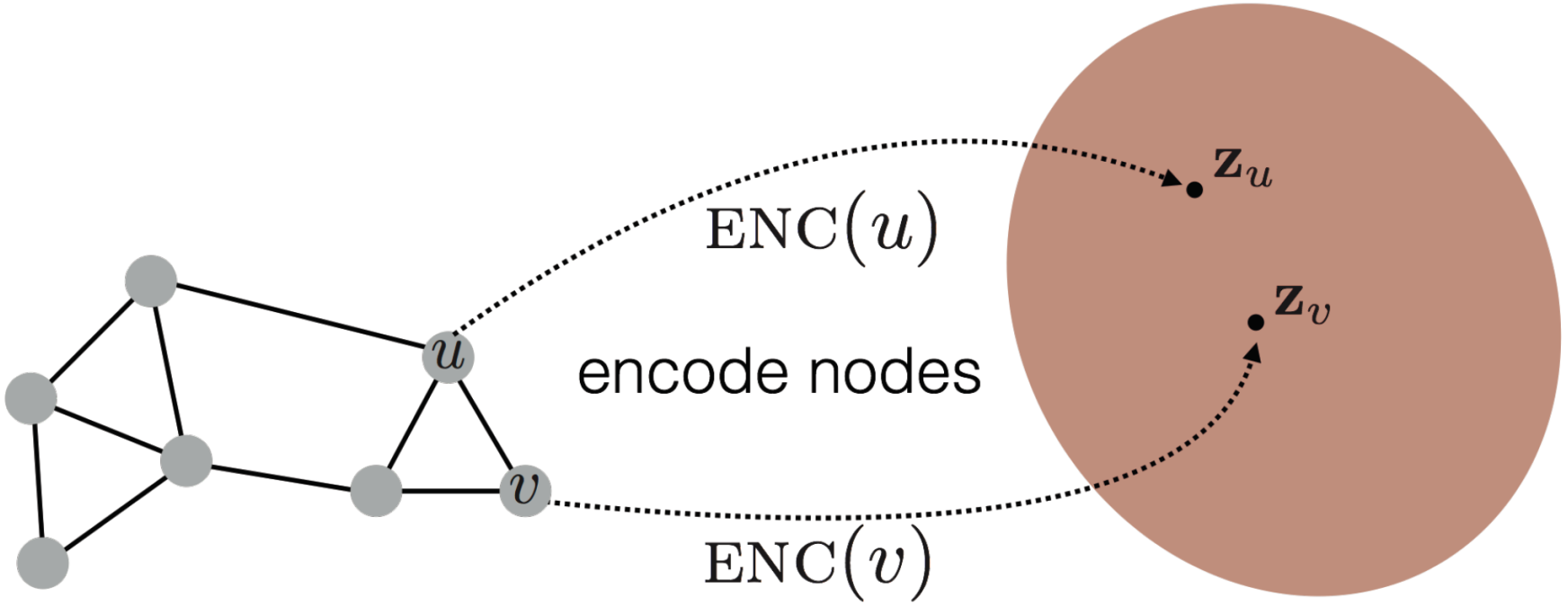
- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)
  - GNNs for Computer vision (Scene Understanding) (AAAI'23)

RECALL: MACHINE LEARNING ON GRAPHS



RECAP: NODE EMBEDDING

- **Main idea:** Encode nodes so that **similarity in the embedding space** approximates **similarity in the graph**



Original graph

Embedding space (Latent space)

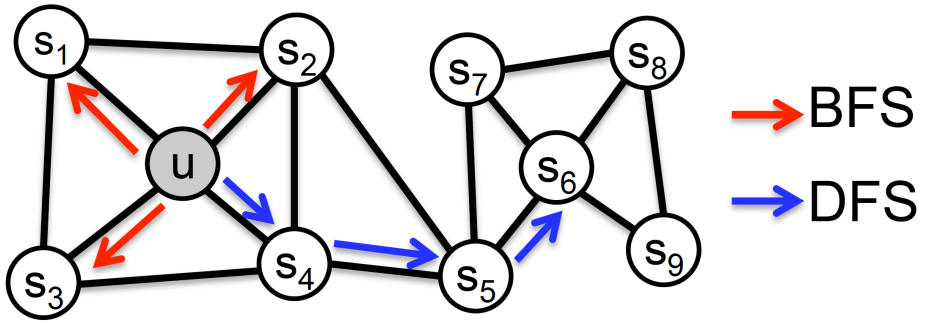
- **Two things to consider**
  - 1. How to encode nodes?
    - Encoder
  - 2. How to define similarity in the embedding space?
    - Decoder (Similarity function)

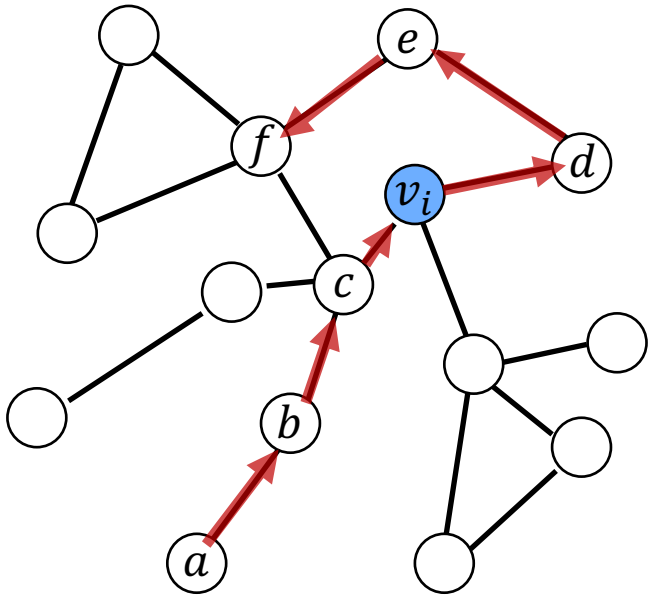
RECAP: DEEPWALK/NODE2VEC

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

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

- **Biased random walk**





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	$o = (N(v_i), v_i) = (\{b, c, d, e\}, v_i)$

Liu, Ninghao, et al. "Is a single vector enough? exploring node polysemy for network embedding." KDD 2019

RECALL: ENCODER

- Maps each node to a low-dimensional vector

$$ENC(v) = \mathbf{z}_v$$

$\mathbf{z}_v$: d-dimensional embedding vector

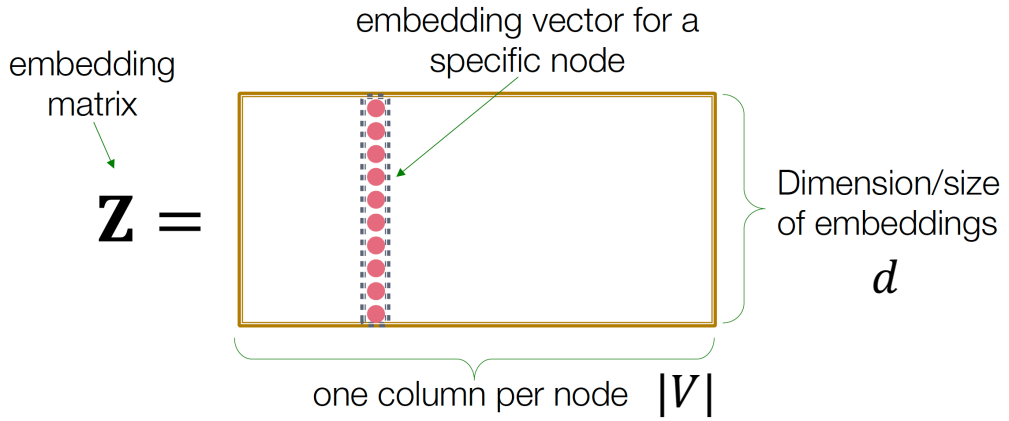
v: Node in the input graph

- **Simplest encoding approach**: Encoder is just an embedding-lookup (Shallow model)

$$ENC(v) = \mathbf{z}_v = \mathbf{Z} \cdot v$$

$\mathbf{Z} \in R^{d \times |V|}$

$v \in I^{|V|}$



$$\mathbf{Z} =$$

$$v = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

ex) $node\ 4$

Each node is assigned a unique embedding vector (i.e., we directly optimize the embedding of each node)

From now on: Deep encoder

DEEP GRAPH ENCODER

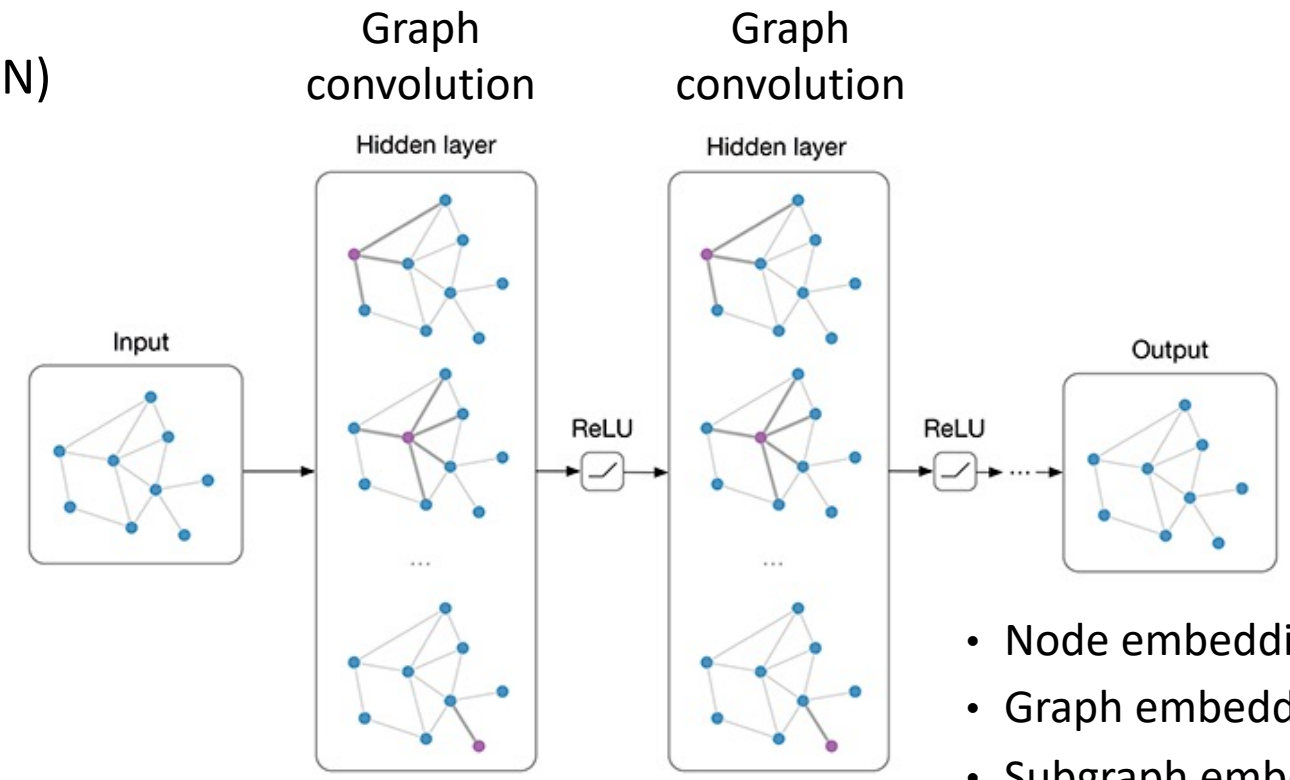
- **Deep Encoder** = **Graph neural network** (GNN)

Shallow model

$$ENC(v) = \mathbf{z}_v$$

Deep model

$ENC(v) =$ Multiple layers of non-linear transformations-based on graph structure

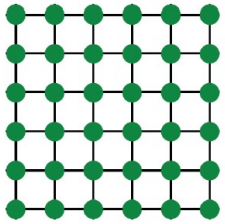
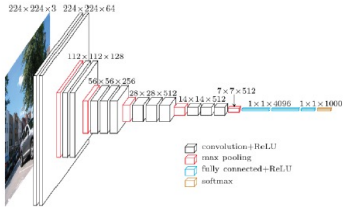


- Node embedding
- Graph embedding
- Subgraph embedding
- Edge embedding
- …

Can we use existing deep learning models? e.g., CNN, RNN, etc

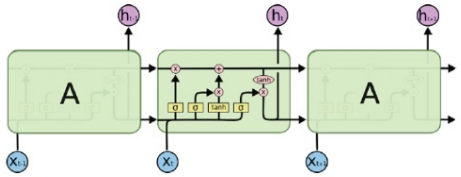
RECAP: CHALLENGES OF GRAPH REPRESENTATION LEARNING

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

  - RNNs for text/sequences …



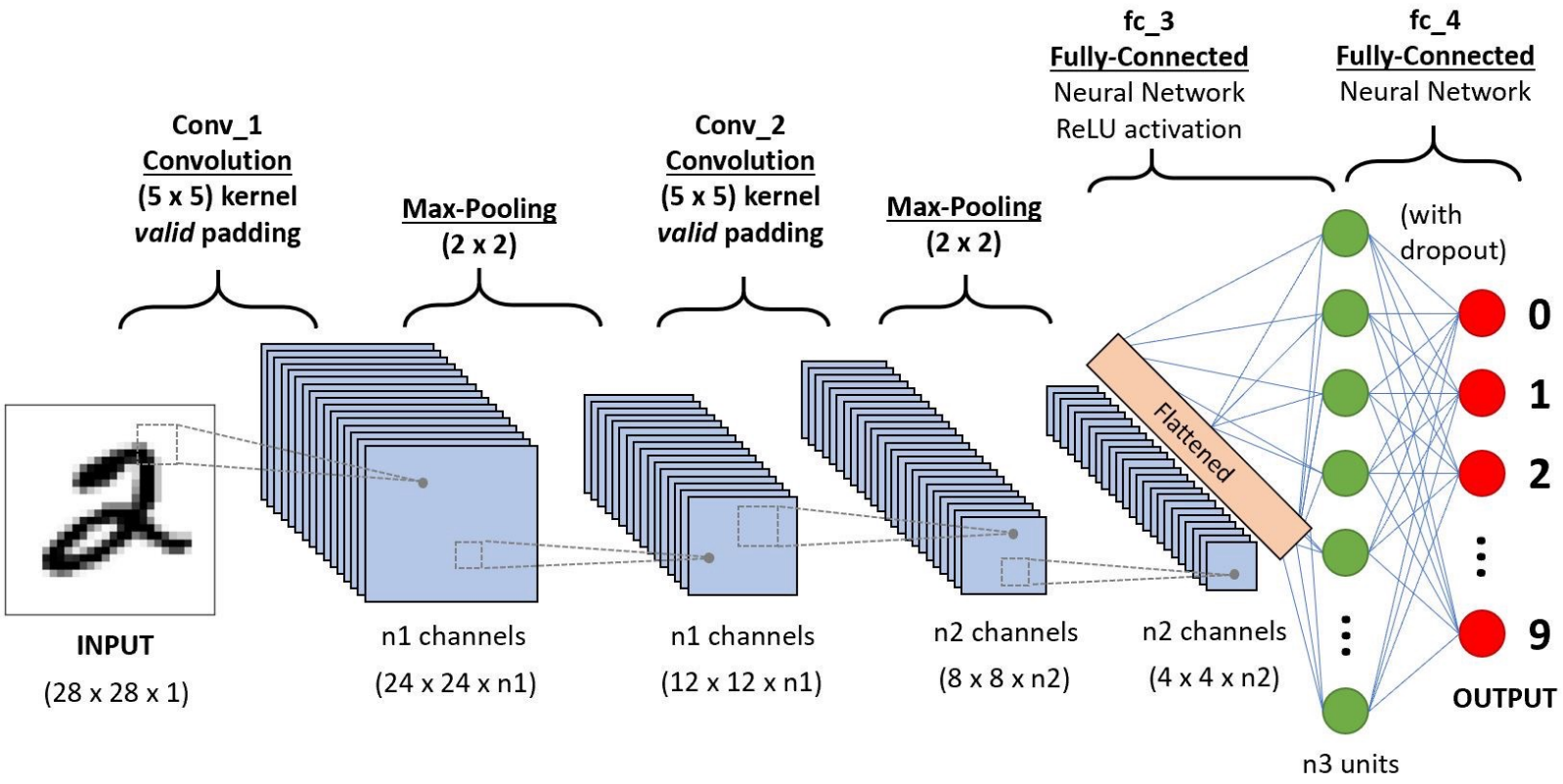


- **Graphs are very complex**
  - Arbitrary structures (no spatial locality like grids / no fixed orderings)
  - Heterogeneous: Directed/undirected, binary/weighted/typed, multimodal features
  - Large-scale: More than millions of nodes and billions of edges

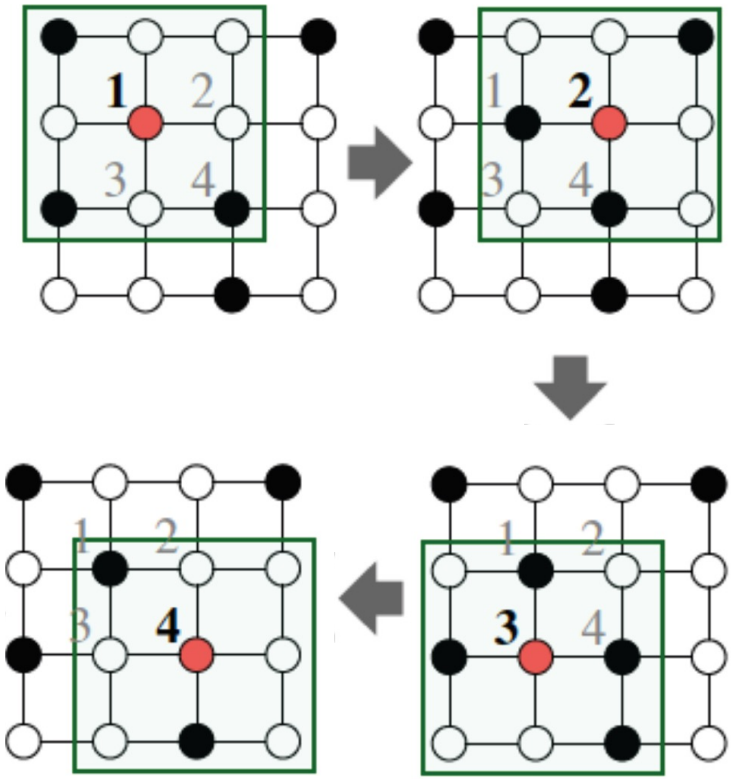
(Figure credit) Tutorial on Graph Representation Learning, AAAI 2019

BACKGROUND: CONVOLUTIONAL NEURAL NETWORKS FOR IMAGES

- Convolutional filters
  - Local feature detectors
  - A feature is learned in each **local receptive field** by a convolutional filter



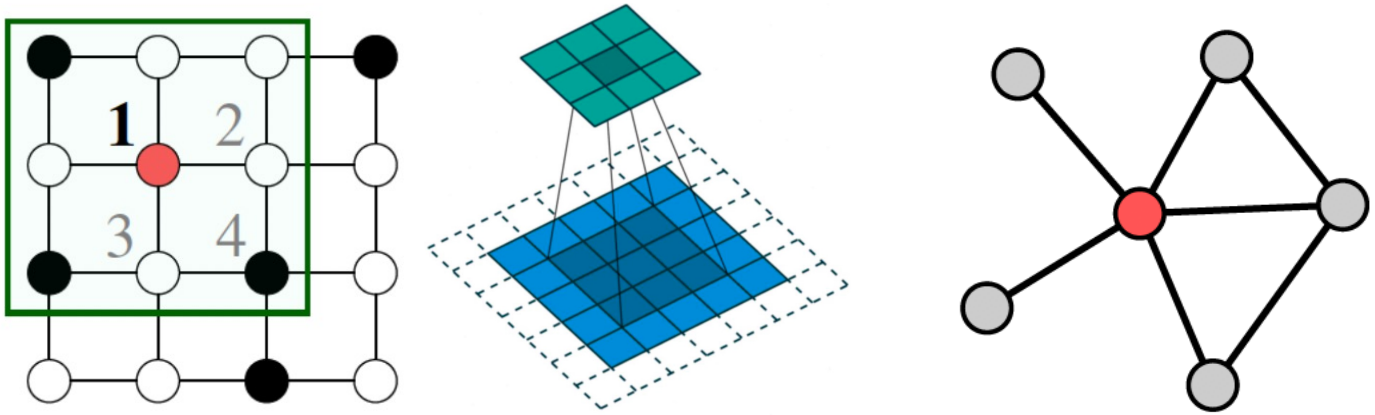
CNN on an image



(Figure credit) https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

FROM IMAGES TO GRAPHS: LOCAL RECEPTIVE FIELD ON GRAPHS

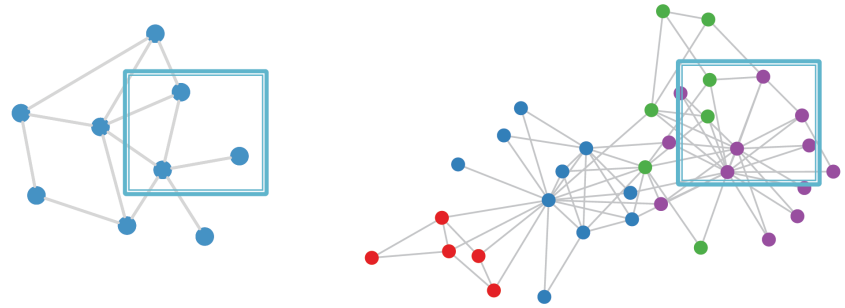
- How should we define local receptive fields on graphs?
  - Local subgraphs



Image

Graph

Graphs look like this



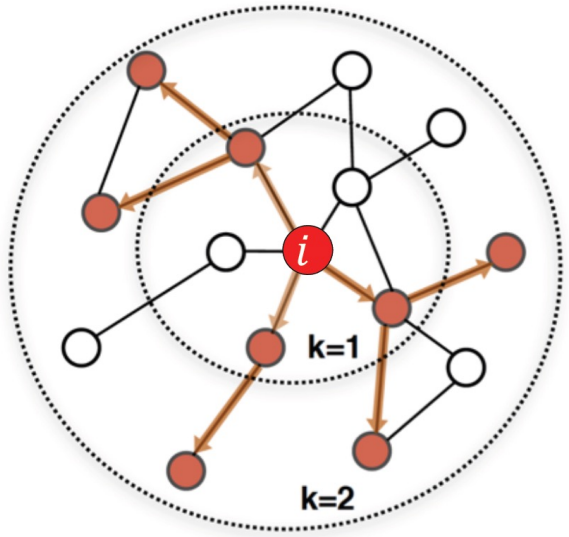
- There is no fixed notion of locality or sliding window on the graph
- No order among neighboring nodes
  - Permutation invariant

- **Idea**: Transform information from the neighboring nodes and combine it
  - **Step 1:** For each node v_i, transform "messages" from neighbors $N(i)$
    - $W_j h_j$ for $v_j \in N(i)$, h_j: "Message" from v_j
  - **Step 2:** Add them up: $\sum_{v_j \in N(i)} W_j h_j$

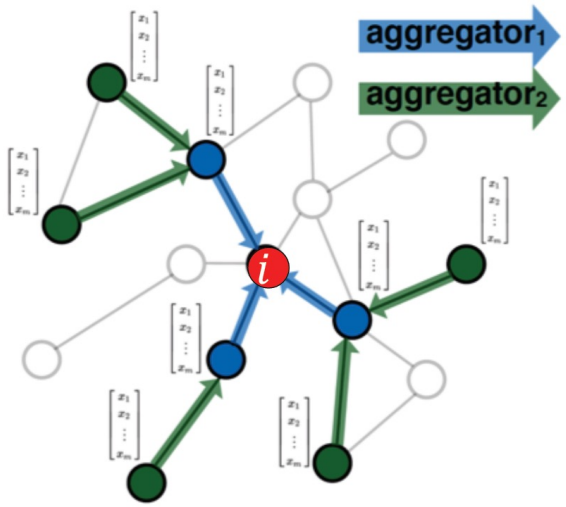
(Figure credit) https://deepgraphlearning.github.io/coursewebsite/schedule

GRAPH CONVOLUTIONAL NETWORK (GCN)

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



Determine node computation graph

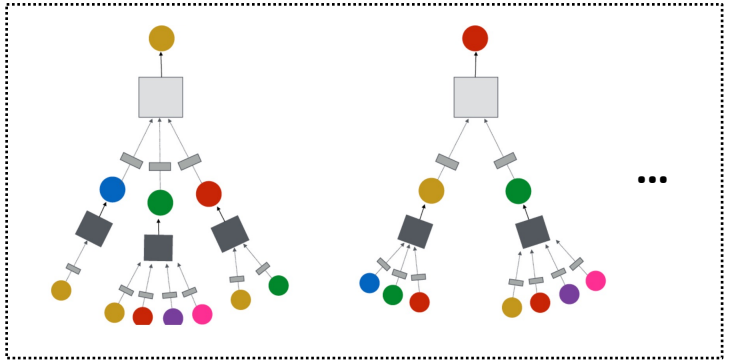


Propagate messages and transform information

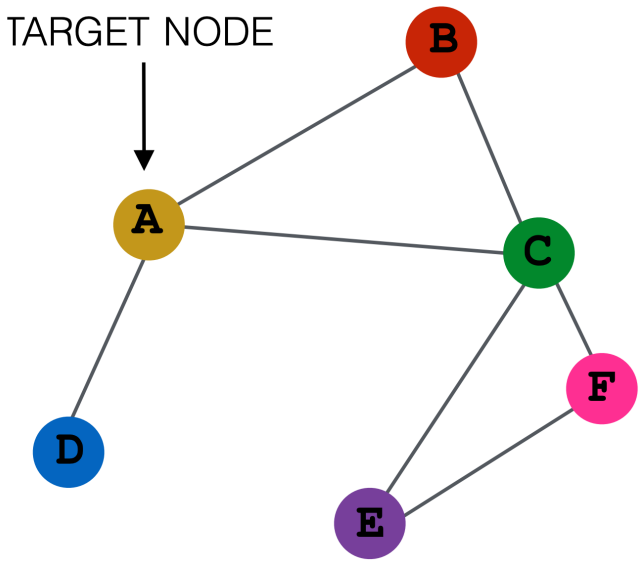
Learn how to propagate information across the graph to compute node features

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

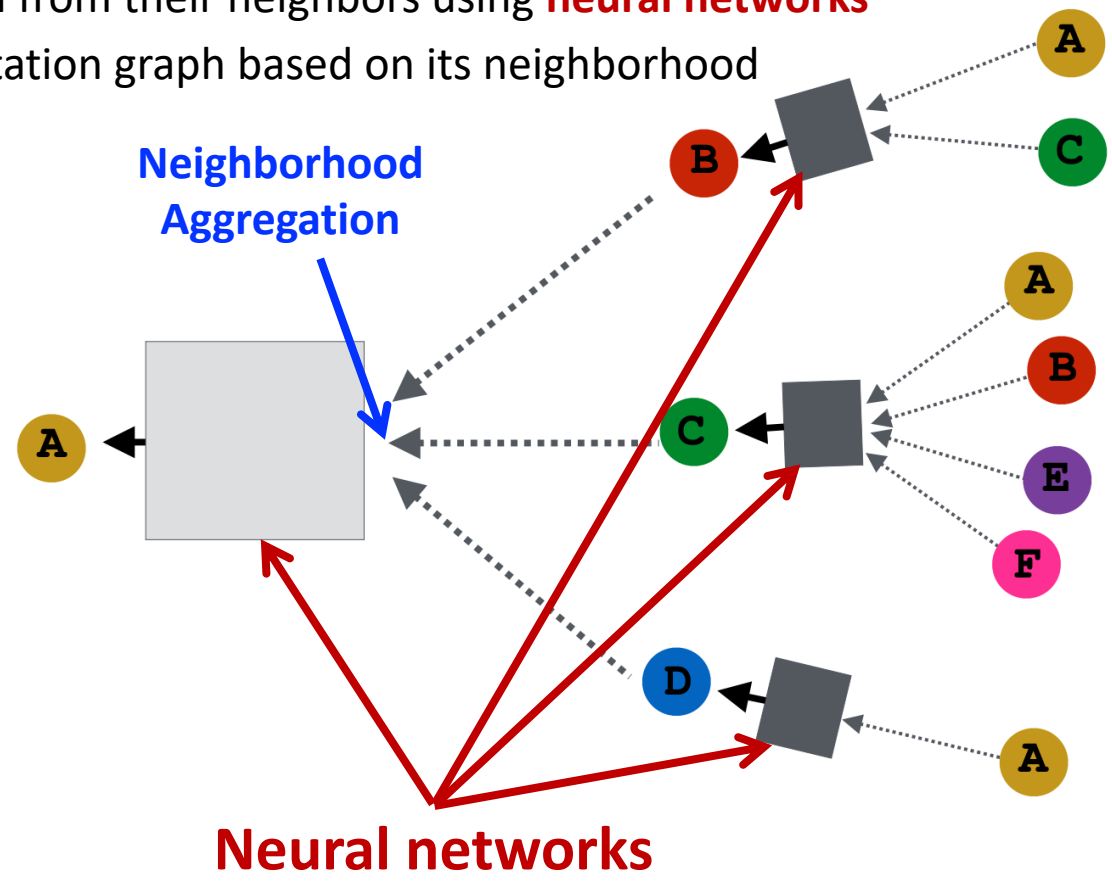
GCN: NEIGHBORHOOD AGGREGATION



- Generate node embeddings based on local network neighborhoods
- **Neighborhood aggregation**
  - Nodes aggregate information from their neighbors using **neural networks**
  - Every node defines a computation graph based on its neighborhood



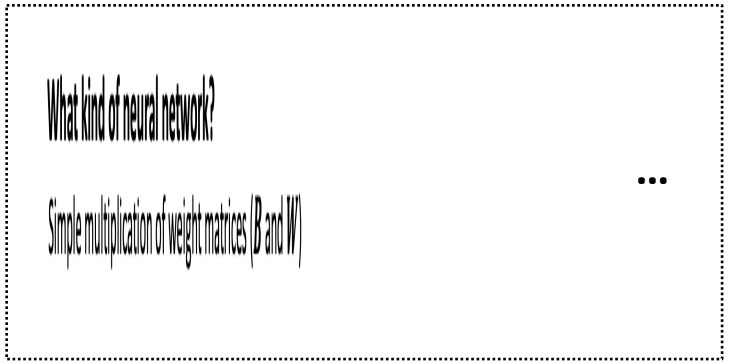
Input graph



- **Things to consider**
  - 1. What kind of neural network?
  - 2. How do we aggregate neighboring nodes?

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

GCN: BASIC APPROACH



- **1. What kind of neural network?**
  - Simple multiplication of weight matrices (B and W)
- **2. What kind of aggregation?**
  - Average

Weight matrix

Embedding of v at layer l

Total number of layers

$$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\boldsymbol{W}_l \sum_{u \in N(v)} \frac{\boldsymbol{h}_u^{(l)}}{|N(v)|} + \boldsymbol{B}_l \boldsymbol{h}_v^{(l)}\right), \quad \forall l \in \{0,1,\ldots,L-1\}$$

Initial embedding of v

$$\boldsymbol{h}_v^0 = \boldsymbol{x}_v$$

Feature of node v

Average of neighboring nodes' previous layer embeddings

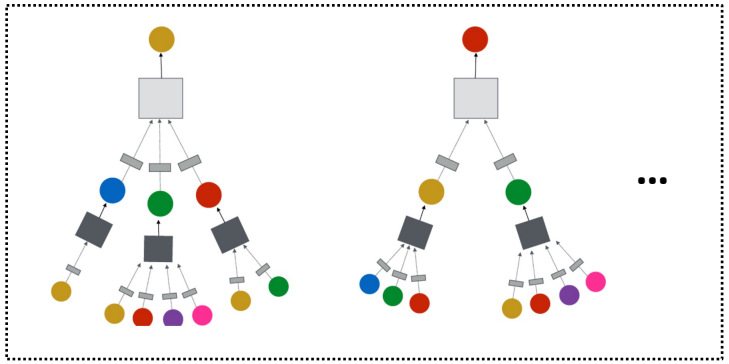
$$\boldsymbol{z}_v = \boldsymbol{h}_v^{(L)}$$

Final embedding of v

How do we train the embeddings?

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

GCN: MATRIX FORMULATION



- GCN can be efficiently computed in a matrix form

$$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\boldsymbol{W}_l \sum_{u \in N(v)} \frac{\boldsymbol{h}_u^{(l)}}{|N(v)|} + \boldsymbol{B}_l \boldsymbol{h}_v^{(l)}\right), \quad h_v^0 = x_v \ , \ z_v = h_v^{(L)}$$

$$\forall l \in \{0,1,\dots,L-1\}$$

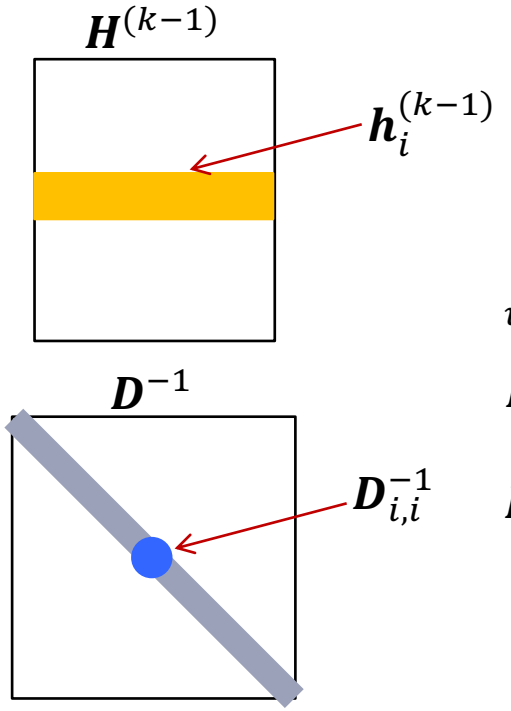
$\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{H}^{(l)}$ (Matrix form)

$$\boldsymbol{H}^{(l+1)} = \sigma(\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}_l^T + \boldsymbol{H}^{(l)}\boldsymbol{B}_l^T) \text{ where } \widetilde{\boldsymbol{A}} = \boldsymbol{D}^{-1}\boldsymbol{A}$$

Neighborhood aggregation

Self transformation

Since $\widetilde{\boldsymbol{A}}$ is sparse, sparse matrix multiplication can be used (efficient)



$$\sum_{u \in N(v)} \boldsymbol{h}_u^{(l)} = \boldsymbol{A}_v \boldsymbol{H}^{(l)}$$

$$\boldsymbol{D}_{v,v} = Deg(v) = |N(v)|$$

$$\boldsymbol{D}_{v,v}^{-1} = \frac{1}{|N(v)|}$$

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

GCN: TRAINING

- We need to define the loss function on the embeddings
- We can feed the **final embeddings z_v** into any loss function and run SGD to train the weight parameters
- **Types of loss function:** 1) Supervised loss, 2) Unsupervised loss
- **1) Supervised loss**

$$\min_{\theta} \sum_{v \in V} \mathcal{L}(y_v, f_\theta(\mathbf{z}_v))$$

  - y_v: Label of node v
  - f_θ: Classifier with parameter θ
  - $\mathcal{L}$ could be squared error if y is real number (regression), or cross entropy if y is categorical (classification)

- **2) Unsupervised loss**
  - No node label available
  - We can use the graph structure as the supervision
    - e.g., adjacency information
    - In this case, $\mathcal{L}$ is cross entropy ($\boldsymbol{A}_{v,u} = 1$ if an edge exists between node v and node u, otherwise 0)

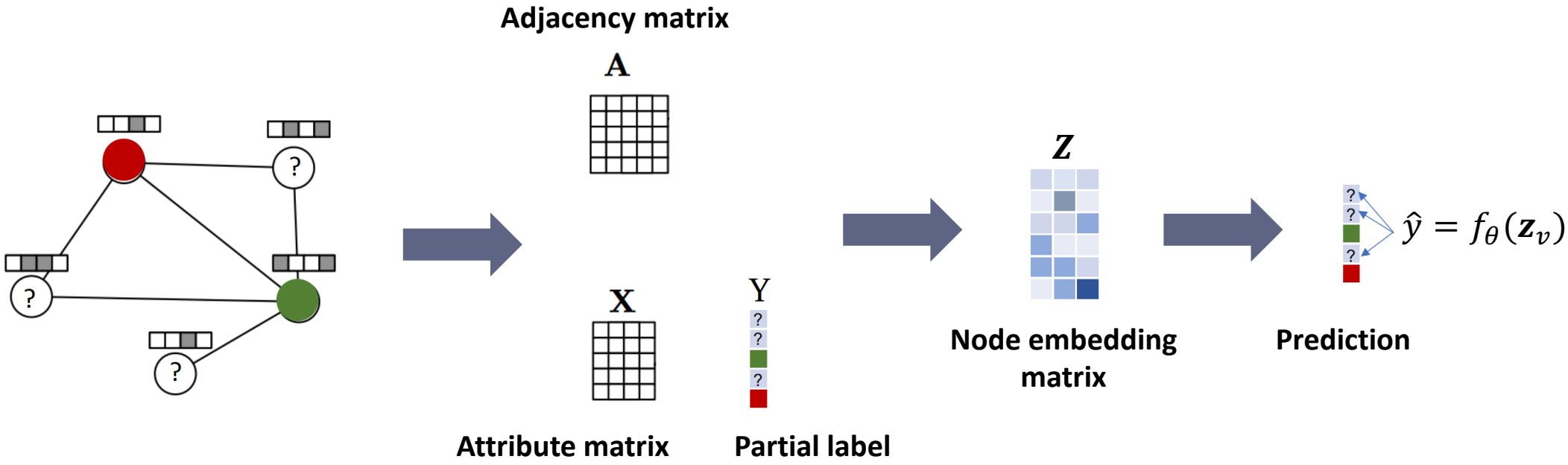
$$\min_{\theta} \sum_{v,u \in V} \mathcal{L}(\boldsymbol{A}_{v,u}, f_\theta(\mathbf{z}_v, \mathbf{z}_u))$$

f_θ: Encoder

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

GCN: SUPERVISED TRAINING

- Directly train the model for a supervised task (e.g., node classification)



$$\mathcal{L} = -\sum_{v_i \in V} y_v \log f_\theta(\boldsymbol{z}_v) + (1 - y_v) \log(1 - f_\theta(\boldsymbol{z}_v))$$

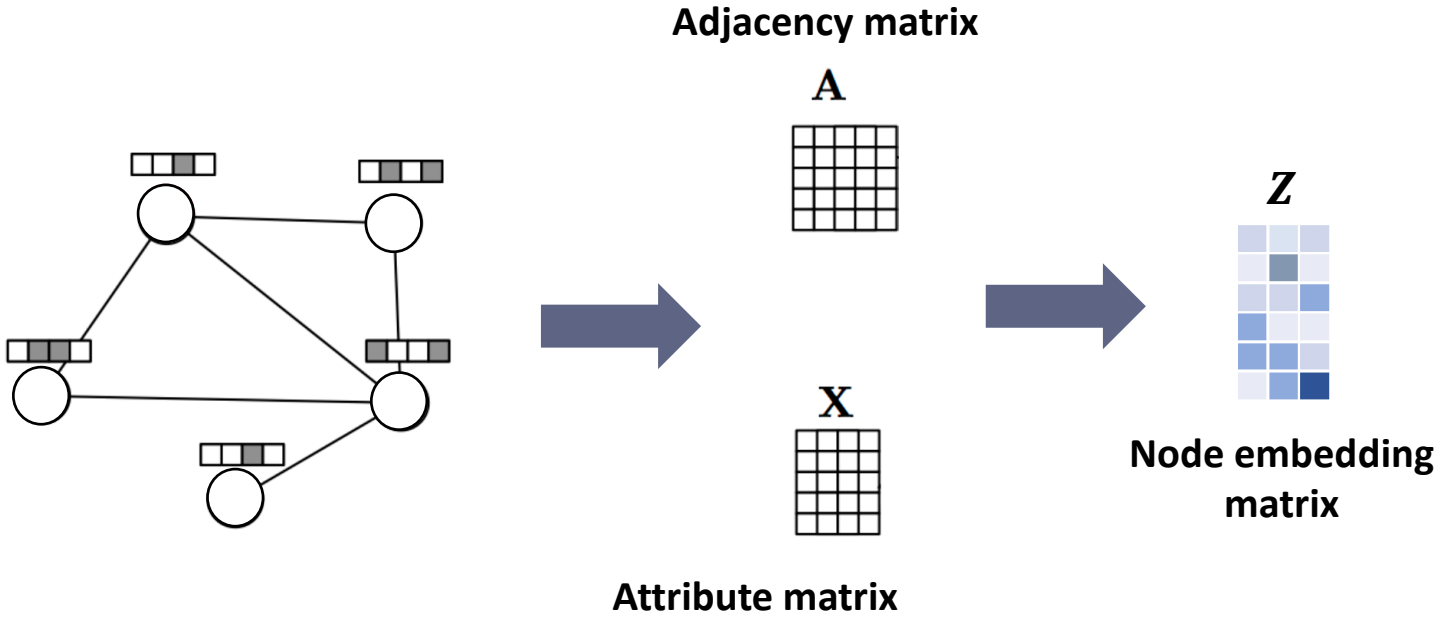
Ground truth label

Model prediction

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

GCN: UNSUPERVISED TRAINING

- As we are not given node labels, we define our task to reconstruct the graph, i.e., Adjacency matrix



$$\mathcal{L} = -\sum_{v,u \in V} \boldsymbol{A}_{v,u} \log f_\theta(\boldsymbol{z}_v, \boldsymbol{z}_u) + \left(1 - \boldsymbol{A}_{v,u}\right) \log(1 - f_\theta(\boldsymbol{z}_v, \boldsymbol{z}_u))$$

Ground truth label

Model prediction

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

GRAPH ATTENTION NETWORKS (GAT)

- **Idea**: Treat different neighboring nodes differently

$$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\boldsymbol{W}_l \sum_{u \in N(v) \cup v} \alpha_{vu} \boldsymbol{h}_u^{(l)}\right)$$

Attention weight

$$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\boldsymbol{W}_l \sum_{u \in N(v)} \frac{\boldsymbol{h}_u^{(l)}}{|N(v)|} + \boldsymbol{B}_l \boldsymbol{h}_v^{(l)}\right) \quad \text{(GCN)}$$

  - α_{vu}: Importance of node u to node v as its neighboring node

- In GCN, the importance was heuristically defined based on the structural property of the graph (node degree)
  - $\alpha_{vu} = \frac{1}{|N(v)|}$: Does not depend on the neighbors (it is fixed)
- All neighboring nodes $u \in N(v)$ are **equally important** to node v

Not all neighbors are equally important!

Veličković, Petar, et al. "Graph attention networks." ICLR 2018

GRAPH ATTENTION NETWORKS (GAT)

- **Idea**: Treat different neighboring nodes differently

$$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\boldsymbol{W}_l \sum_{u \in N(v) \cup v} \alpha_{vu} \boldsymbol{h}_u^{(l)}\right)$$

Attention weight

- α_{vu}: Importance of node u to node v as its neighboring node

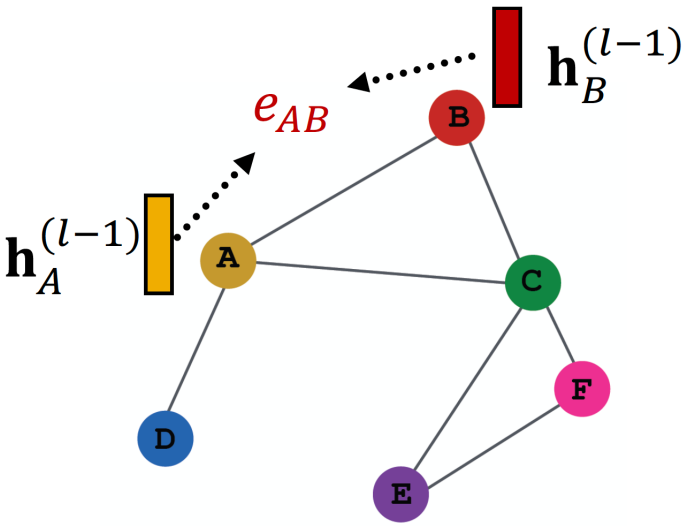
- Computing the attention weight

$$e_{vu} = a(\boldsymbol{W}_l \boldsymbol{h}_v^{(l)}, \boldsymbol{W}_l \boldsymbol{h}_u^{(l)})$$

(Importance of node u's message to node v)

$$\alpha_{vu} = \frac{\exp(e_{uv})}{\sum_{k \in N(v)} \exp(e_{vk})}$$

(Normalization)



$$e_{AB} = a(\boldsymbol{W}_{l-1} \boldsymbol{h}_A^{(l-1)}, \boldsymbol{W}_{l-1} \boldsymbol{h}_B^{(l-1)})$$

How do we define $a(\cdot)$?

GRAPH ATTENTION NETWORKS (GAT)

$$\alpha_{vu} = \frac{\exp(e_{uv})}{\sum_{k \in N(v)} \exp(e_{vk})}$$

- **Defining the function $a(\cdot)$**

$$e_{vu} = a(\boldsymbol{W}_l \boldsymbol{h}_v^{(l)}, \boldsymbol{W}_l \boldsymbol{h}_u^{(l)}) \quad \text{(Importance of node } u\text{'s message to node } v\text{)}$$

$$= \text{Linear}(\text{Concat}\left(\boldsymbol{W}_l \boldsymbol{h}_v^{(l)}, \boldsymbol{W}_l \boldsymbol{h}_u^{(l)}\right))$$

Parameters of Linear layer is jointly trained end-to-end with other parameters of GAT

- **Multi-head attention** (Refer to Lecture 5 - Transformer)
  - Create multiple attention scores using multiple copies of parameters

$$\boldsymbol{h}_v^{(l)}[1] = \sigma\left(\boldsymbol{W}_l \sum_{u \in N(v) \cup v} \alpha_{vu}^{[1]} \boldsymbol{h}_u^{(l)}\right)$$

$$\boldsymbol{h}_v^{(l)}[2] = \sigma\left(\boldsymbol{W}_l \sum_{u \in N(v) \cup v} \alpha_{vu}^{[2]} \boldsymbol{h}_u^{(l)}\right)$$

$$\boldsymbol{h}_v^{(l)}[3] = \sigma\left(\boldsymbol{W}_l \sum_{u \in N(v) \cup v} \alpha_{vu}^{[3]} \boldsymbol{h}_u^{(l)}\right)$$

$$\boldsymbol{h}_v^{(l)} = \text{AGG}(\boldsymbol{h}_v^{(l)}[1], \boldsymbol{h}_v^{(l)}[2], \boldsymbol{h}_v^{(l)}[3])$$

The final embedding aggregates the outputs of multi-head attention

GRAPHSAGE

- **Generalization of GCN/GAT**
  - So far we have aggregated the neighbor messages by taking their (weighted) average Can we do better?

Average of neighboring nodes

Add self representation

$$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\boldsymbol{W}_l \sum_{u \in N(v)} \frac{\boldsymbol{h}_u^{(l)}}{|N(v)|} + \boldsymbol{B}_l \boldsymbol{h}_v^{(l)}\right)$$

GCN

Attention weight

$$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\boldsymbol{W}_l \sum_{u \in N(v) \cup v} \alpha_{vu} \boldsymbol{h}_u^{(l)}\right)$$

GAT

GraphSAGE

Generalized representation

Generalized representation

$$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\left[\boldsymbol{W}_l \cdot \mathrm{AGG}\left(\left\{\boldsymbol{h}_u^{(l)} | u \in N(v)\right\}\right), \boldsymbol{B}_l \boldsymbol{h}_v^{(l)}\right]\right)$$

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

GRAPHSAGE

- **Generalization of GCN/GAT**
  - So far we have aggregated the neighbor messages by taking their (weighted) average Can we do better?

Generalized representation

Generalized representation
(Concatenation here)

$$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\left[\boldsymbol{W}_l \cdot \mathrm{AGG}\left(\left\{\boldsymbol{h}_u^{(l)} | u \in N(v)\right\}\right), \boldsymbol{B}_l \boldsymbol{h}_v^{(l)}\right]\right)$$

- **Variants of AGG**
  - **Mean:** Same as GCN
    - $\mathrm{AGG} = \sum_{u \in N(v)} \frac{\boldsymbol{h}_u^{(l)}}{|N(v)|}$
  - **Pool:** Transform neighbor vectors and apply symmetric vector function
    - $\mathrm{AGG} = \gamma\left(\left\{\mathrm{MLP}(\boldsymbol{h}_u^{(l)}) | u \in N(v)\right\}\right)$, where γ is element-wise mean/max/min
  - **LSTM:** Apply LSTM to shuffled neighbors
    - $\mathrm{AGG} = \mathrm{LSTM}\left(\left[\boldsymbol{h}_u^{(l)} | u \in \pi(N(v))\right]\right)$

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

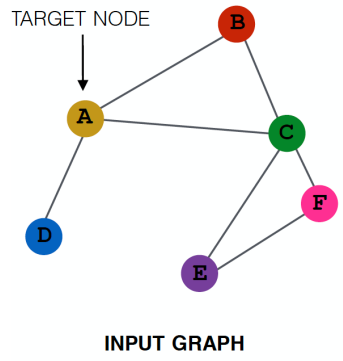
A GNN LAYER: OVERVIEW

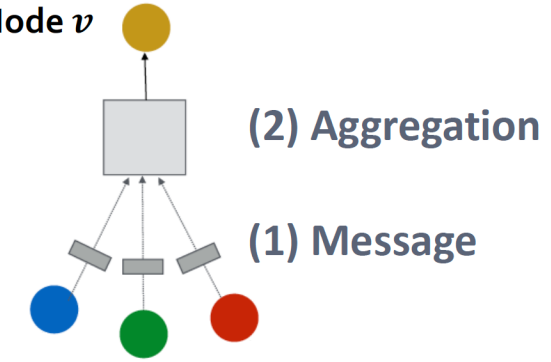
- GNN layer = 1) **Message** + 2) **Aggregation**
- Compress a set of vectors into a single vector
- Different types of GNN layers
  - GCN, GraphSAGE, GAT, …

(GCN) $$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\boldsymbol{W}_l \sum_{u \in N(v)} \frac{\boldsymbol{h}_u^{(l)}}{|N(v)|} + \boldsymbol{B}_l \boldsymbol{h}_v^{(l)}\right)$$

(GAT) $$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\boldsymbol{W}_l \sum_{u \in N(v) \cup v} \alpha_{vu} \boldsymbol{h}_u^{(l)}\right)$$

(GraphSAGE) $$\boldsymbol{h}_v^{(l+1)} = \sigma\left(\left[\boldsymbol{W}_l \cdot \mathrm{AGG}\left(\left\{\boldsymbol{h}_u^{(l)} | u \in N(v)\right\}\right), \boldsymbol{B}_l \boldsymbol{h}_v^{(l)}\right]\right)$$

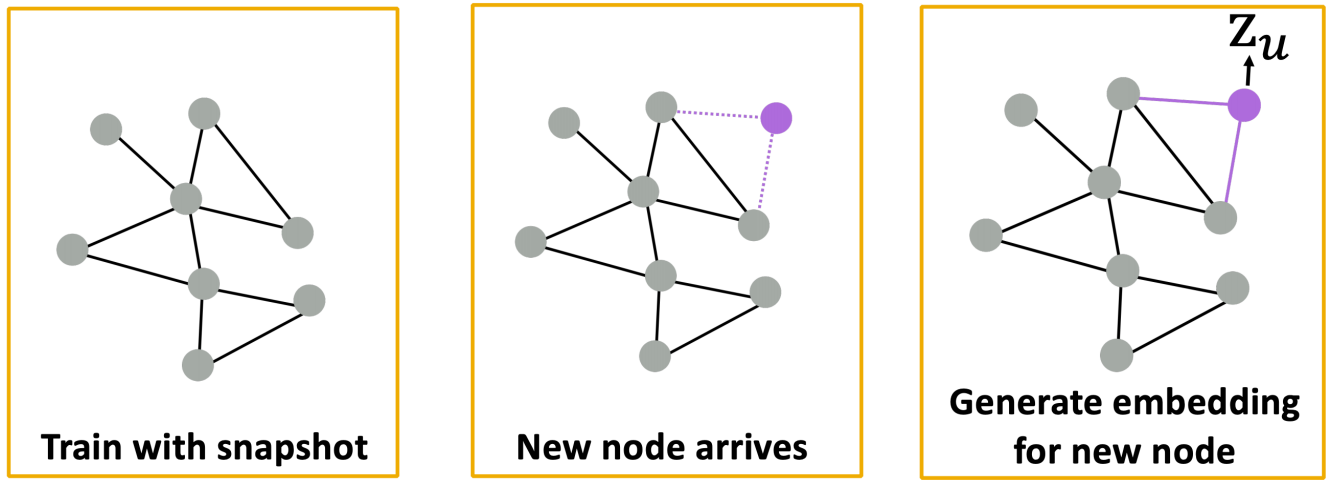




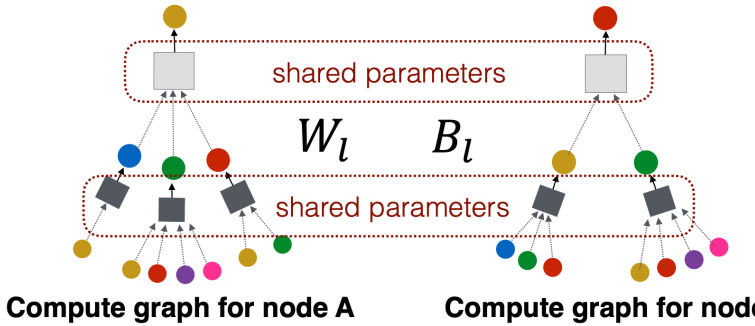
- (1) Message computation
  - Message function: $\boldsymbol{m}_u^{(l)} = \mathrm{MSG}^{(l)}(\boldsymbol{h}_u^{(l)})$
  - Example: A linear layer $\boldsymbol{m}_u^{(l)} = \boldsymbol{W}_l \boldsymbol{h}_u^{(l)}$
- (2) Aggregation
  - $\boldsymbol{h}_v^{(l)} = \mathrm{AGG}\left(\left\{\boldsymbol{m}_u^{(l)} | u \in N(v)\right\}\right)$
  - Example: SUM(), MEAN(), or MAX() aggregator
    - $\boldsymbol{h}_v^{(l)} = \mathrm{SUM}\left(\left\{\boldsymbol{m}_u^{(l)} | u \in N(v)\right\}\right) = \sum_{u \in N(v)} \boldsymbol{W}_l \boldsymbol{h}_u^{(l)}$
      $= \sum_{u \in N(v)} \boldsymbol{m}_u^{(l)}$

INDUCTIVE CAPABILITY OF GNN

- **Inductive learning**: We can obtain embeddings for nodes that have not appeared in the training time
  - e.g., In Amazon, new users are consistently added to the system, and it is impractical to re-train the system to get t he embeddings for the new users



- This is possible because we do not train an embedding matrix as done in Deepwalk/node2vec
- Instead, we train **aggregator** and **transformer**



OUTLINE

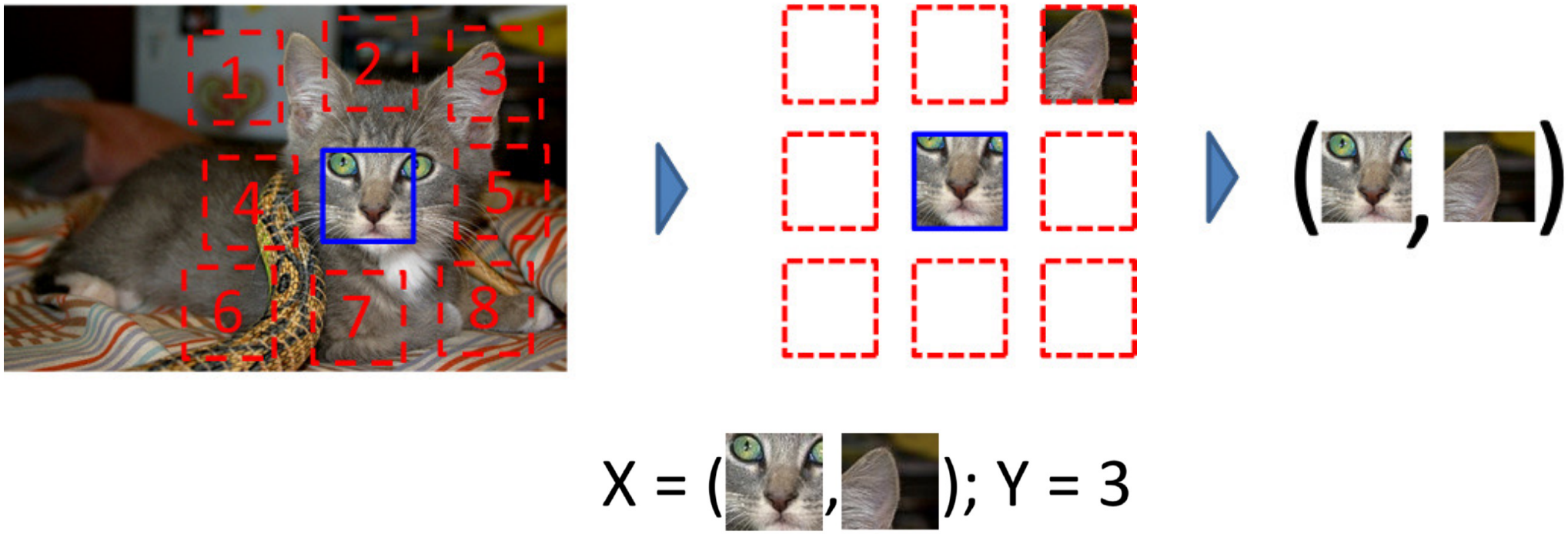
- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)
  - GNNs for Computer vision (Scene Understanding) (AAAI'23)

OUTLINE

- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)
  - GNNs for Computer vision (Scene Understanding) (AAAI'23)

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?

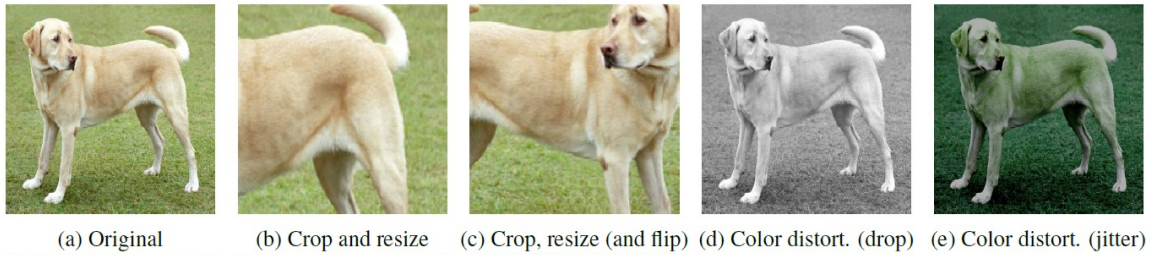
- Pretext task: **Jigsaw puzzle**

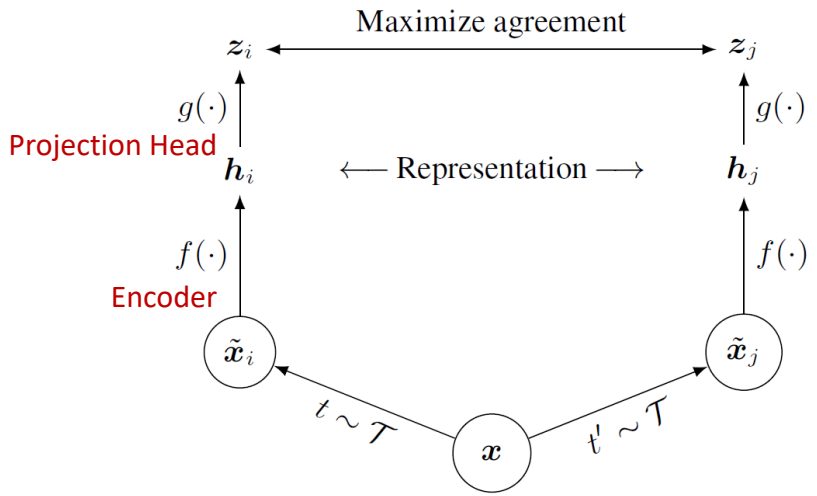
- Pretext task : **Colorization**

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

(Figure credit) Self-Supervised Learning, Andrew Zisserman

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!**

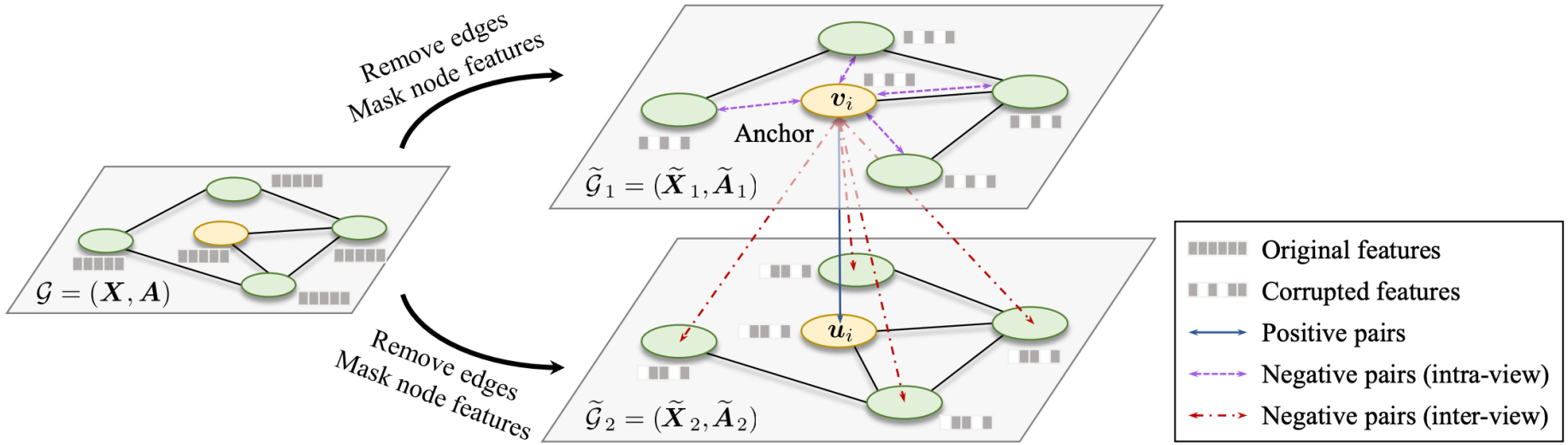
$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

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



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

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

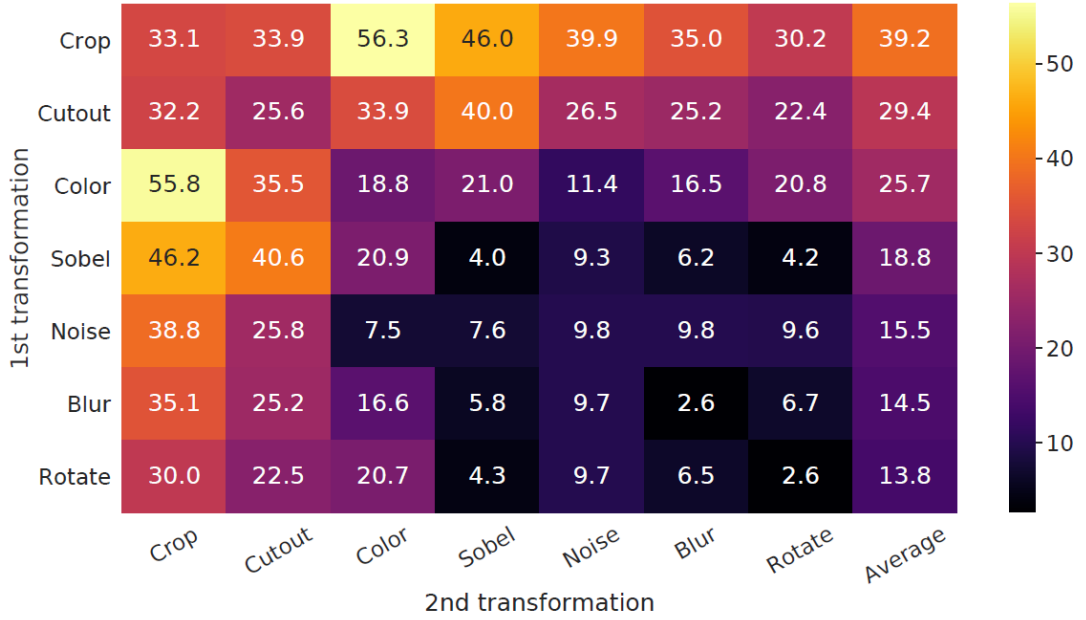
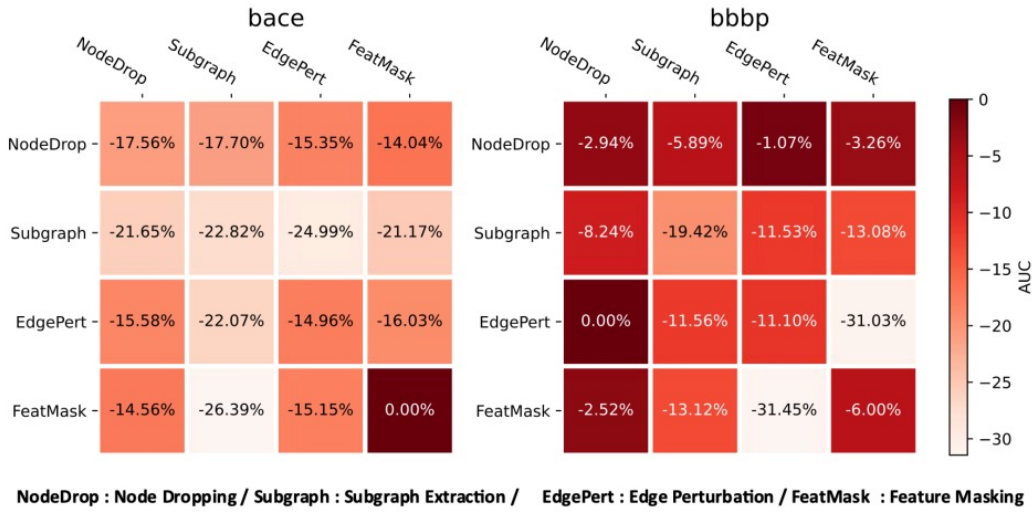


Image classification



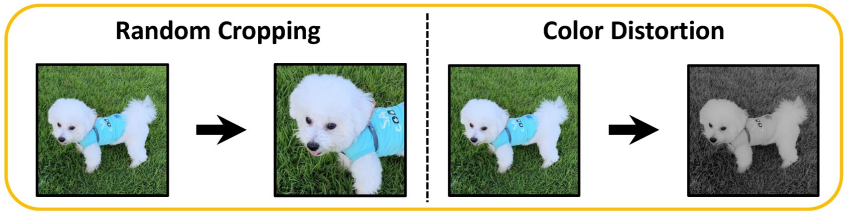
Graph classification

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

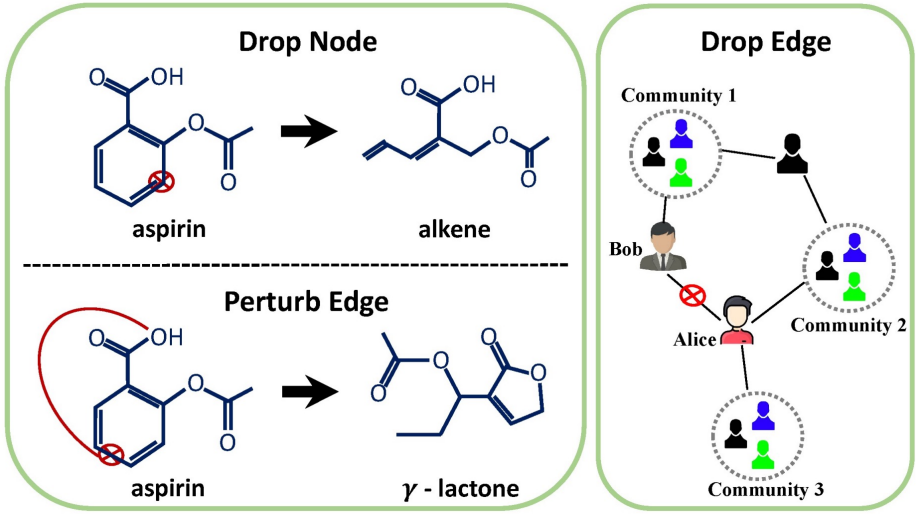
MoCL: Data-driven Molecular Fingerprint via Knowledge-aware Contrastive Learning from Molecular Graph, KDD 2020

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



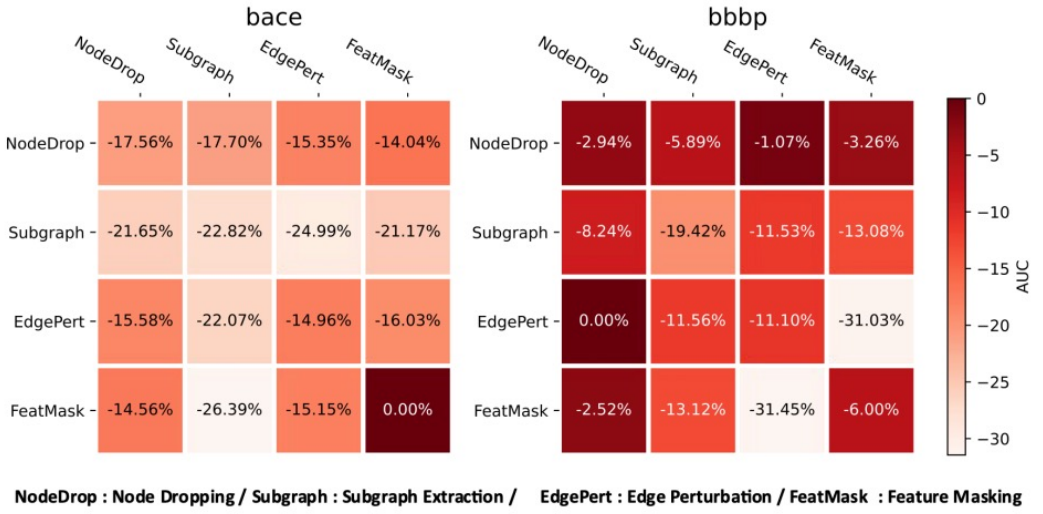
Because graphs contain not only the semantic but also the **structural information**

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



Graph-level task

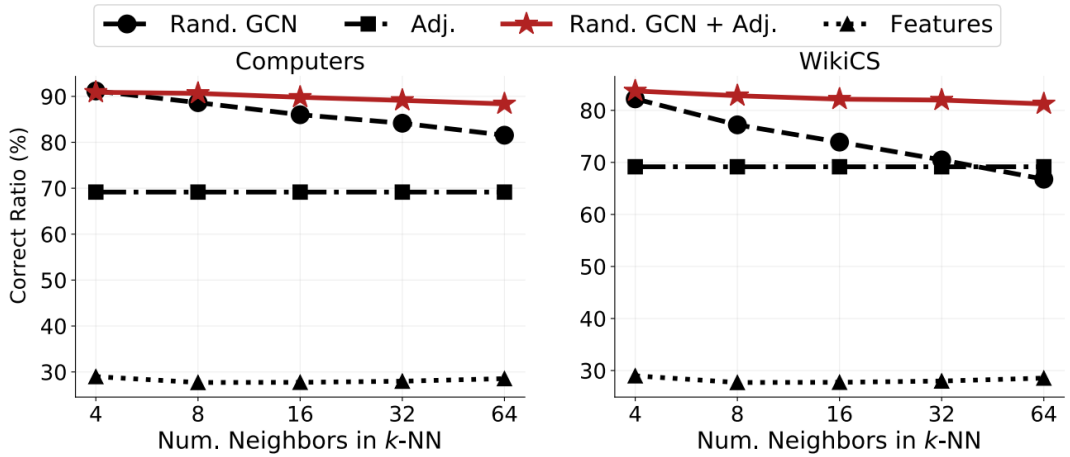
- The quality of the learned representations relies on the **choice of augmentation scheme**
  - Performance on various downstream tasks varies greatly according to the choice of augmentation hyperparameters

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

Augmentation-Free Graph Representation Learning

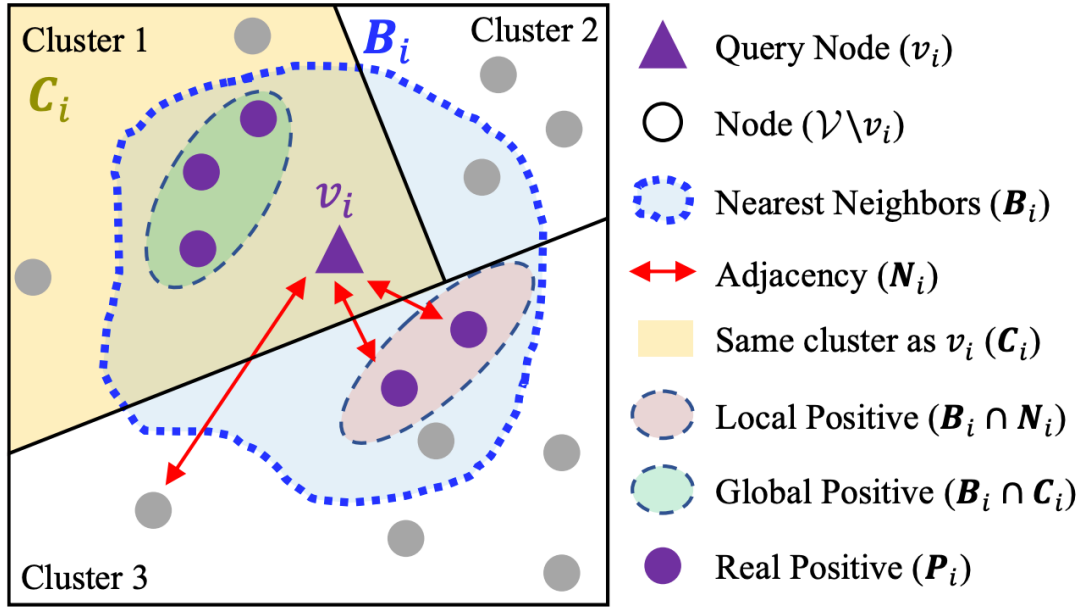
- Instead of creating two arbitrarily augmented views of a graph,
  - Use the original graph as-is as one view, and generate another view by discovering nodes that can serve as positive samples via **k-nearest neighbor search in embedding space**
- However, naively selected positive samples with k-NN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

% of same label among neighbors



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

Capturing Local and Global Semantics



- 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

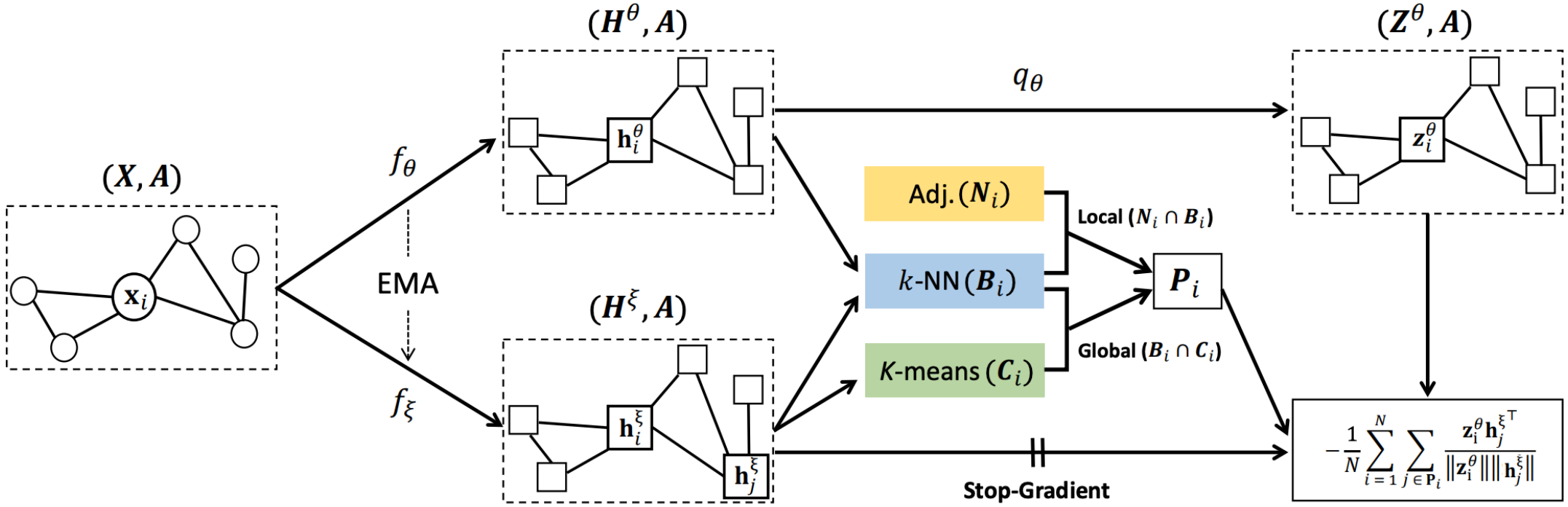
- Obtain real positives for v_i

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

- Minimize the cosine distance between query and real positives P_i

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

Overall Architecture of AFGRL



Experiments

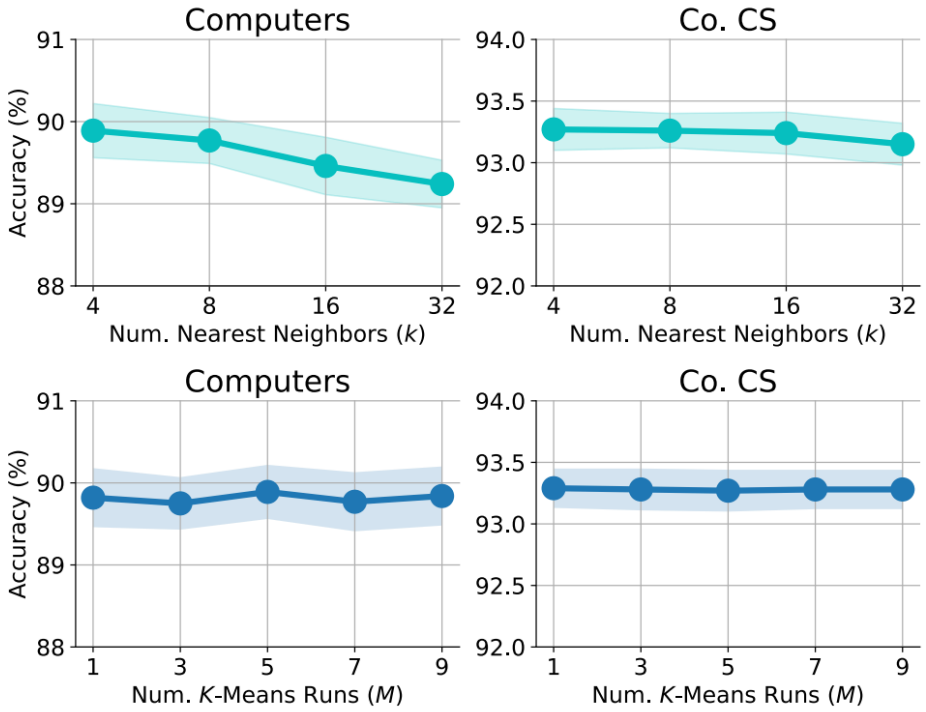
- **Task**: Node classification

	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**

Recall…

		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

AFGRL outperforms SOTA baselines



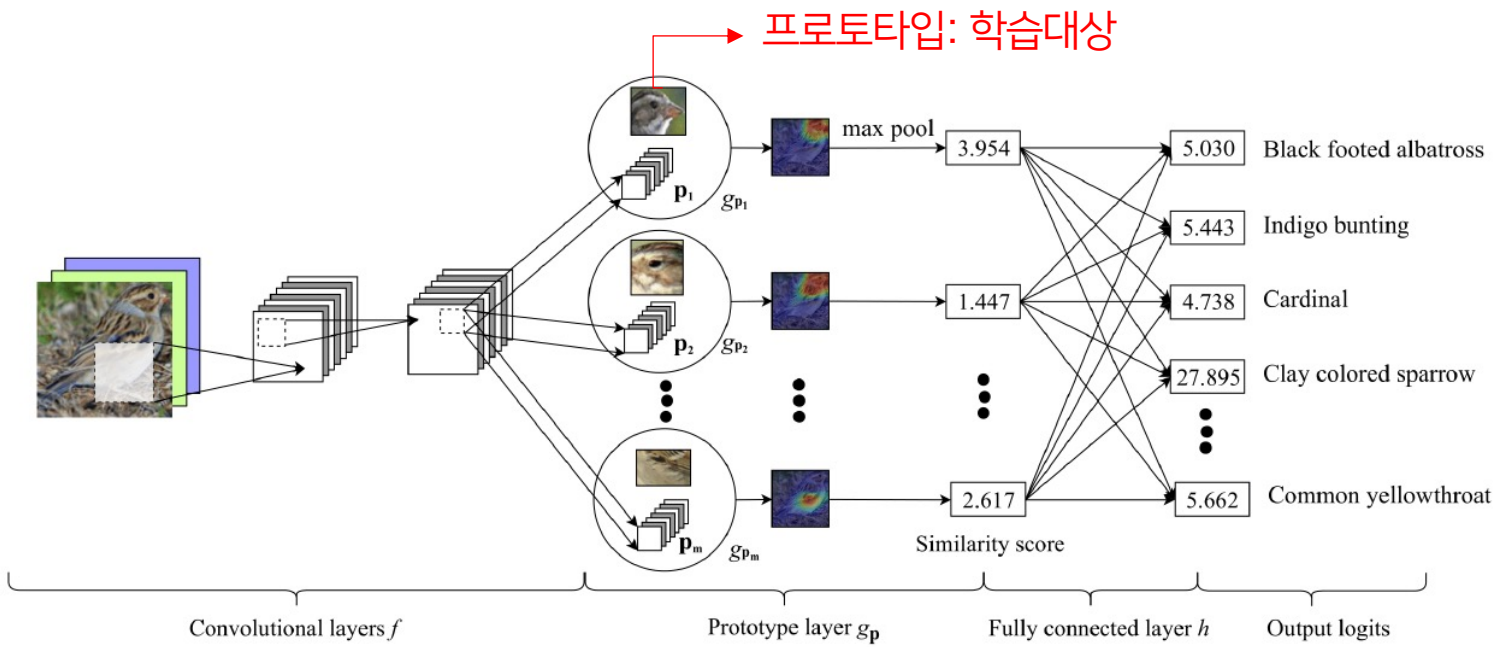
AFGRL is stable over hyperparameters
→ Can be easily trained compared with other augmentation-based methods.

Outline

- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)
  - GNNs for Computer vision (Scene Understanding) (AAAI'23)

INTRODUCTION

- Explainable AI
  - Reasoning process : 모델이 해당 결정을 내리기 위해 수행한 과정을 제시
    - 모델이 올바른 예측 또는 잘못된 예측을 한 이유에 대한 과정을 시각화하고 분석



1. 입력 이미지로부터 feature 추출
2. 각 class에 대한 prototype과 유사성 점수를 계산
3. 계산한 유사성 점수는 fully connected layer를 통해 output logit값을 생성

- 학습된 Prototype을 활용하여 reasoning process 제시
  - 각 class에 대해 고정된 수의 prototype을 할당

Chen, Chaofan, et al. "This looks like that: deep learning for interpretable image recognition." Advances in neural information processing systems 32 (2019).

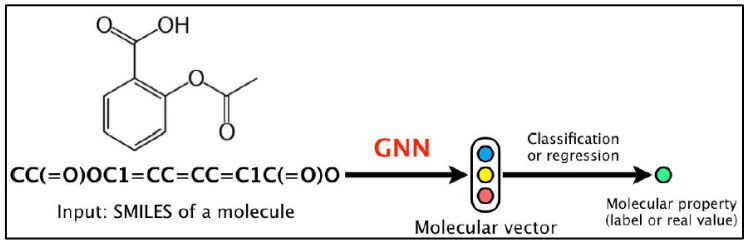
INTRODUCTION

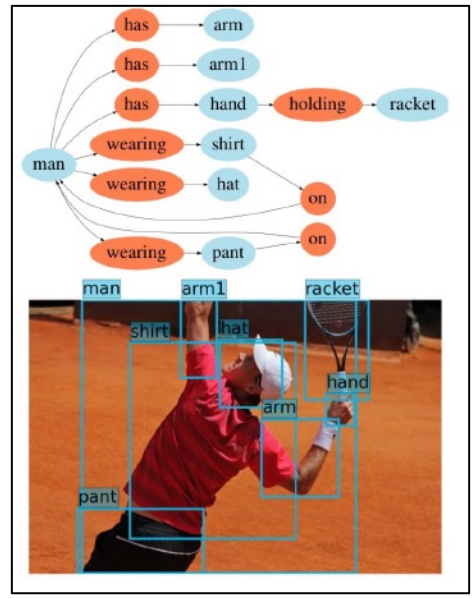
- Reasoning process : 모델이 해당 결정을 내리기 위해 수행한 과정을 제시

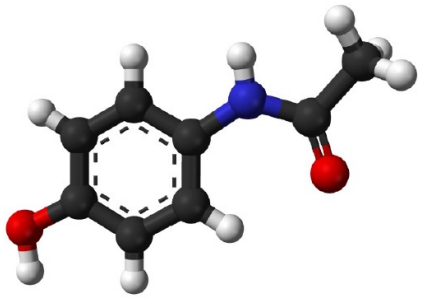
Chen, Chaofan, et al. "This looks like that: deep learning for interpretable image recognition." Advances in neural information processing systems 32 (2019).

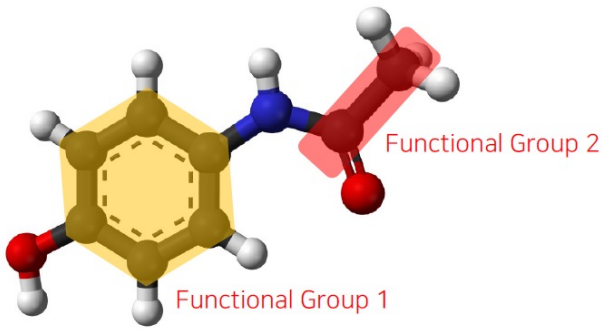
INTRODUCTION

- Graph data
  - Node와 Edge로 구성, 개체 간의 관계를 표현하기 위해 사용
  - Graph Classification : Graph가 어떤 레이블에 속하는지 예측
    - Application : 분자 특성 예측, 약물 발견, 장면 이해
- 분자 구조의 Graph data
  - 분자는 원자 간의 결합으로 구성되어 있어 Graph로 표현하기 적합
  - Functional Group : 분자 전체의 특성을 결정 짓는 서브그래프









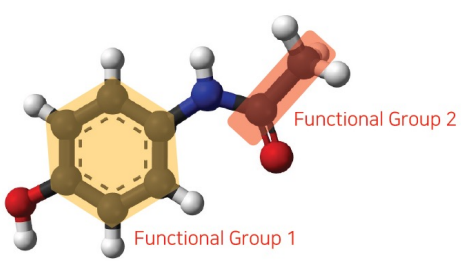
분자 = Graph

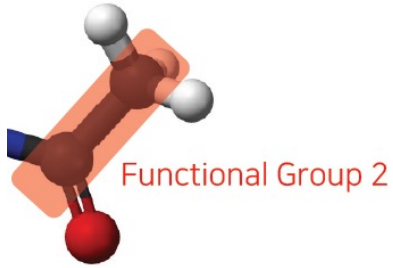
Functional Group = Subgraph

→ 중요한 subgraph를 탐지하기 위해 정보 이론 기반 방식이 제안됨

INTRODUCTION

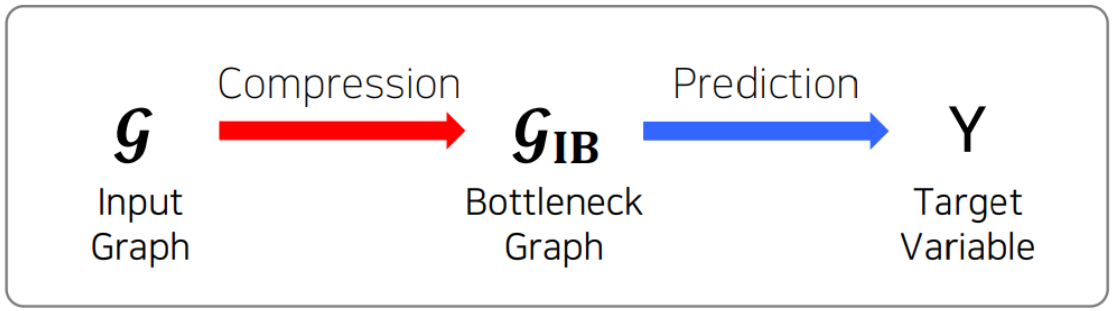
- Graph Information Bottleneck
  - Original graph $\mathcal{G}$를 bottleneck graph $\mathcal{G}_{IB}$로 압축하는 동시에 prediction Y와 관련된 정보를 유지하는 subgraph를 탐지





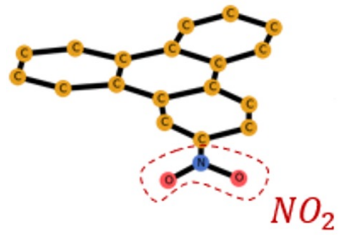
  - 목표 : Label Y를 예측하는 데 중요한 Bottleneck graph $\mathcal{G}_{IB}$을 찾는다.
  - Mutual Information(I) : 특정 variable이 다른 variable에 영향을 미치는 정도

$$\min_{\mathcal{G}_{IB}} \underbrace{-I(Y, \mathcal{G}_{IB})}_{\text{Prediction}} + \beta \underbrace{I(\mathcal{G}; \mathcal{G}_{IB})}_{\text{Compression}}$$



INTRODUCTION

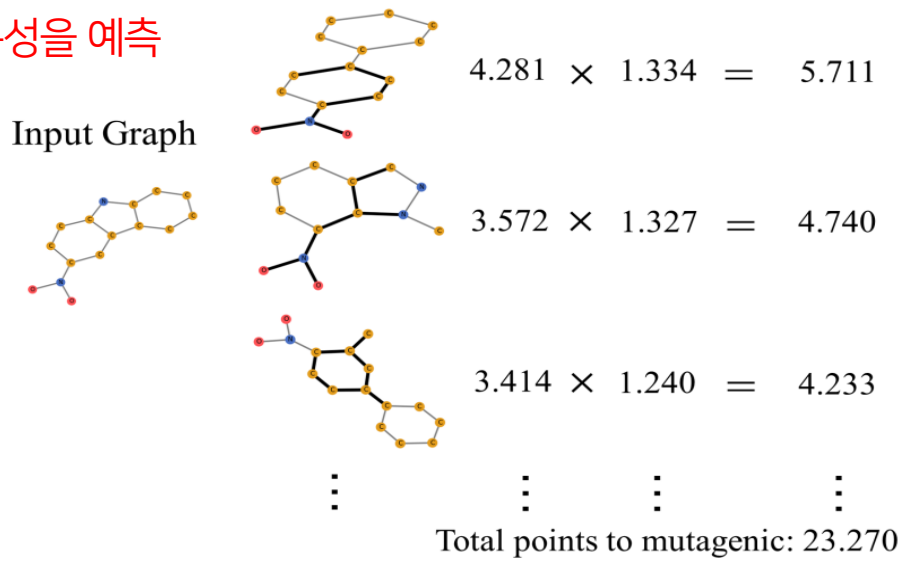
- ProtGNN
  - 그래프 데이터에서 Prototype을 활용하여 reasoning process를 제시한 모델
  - 모델이 특정 그래프를 해당 레이블로 예측하게 된 과정을 시각화



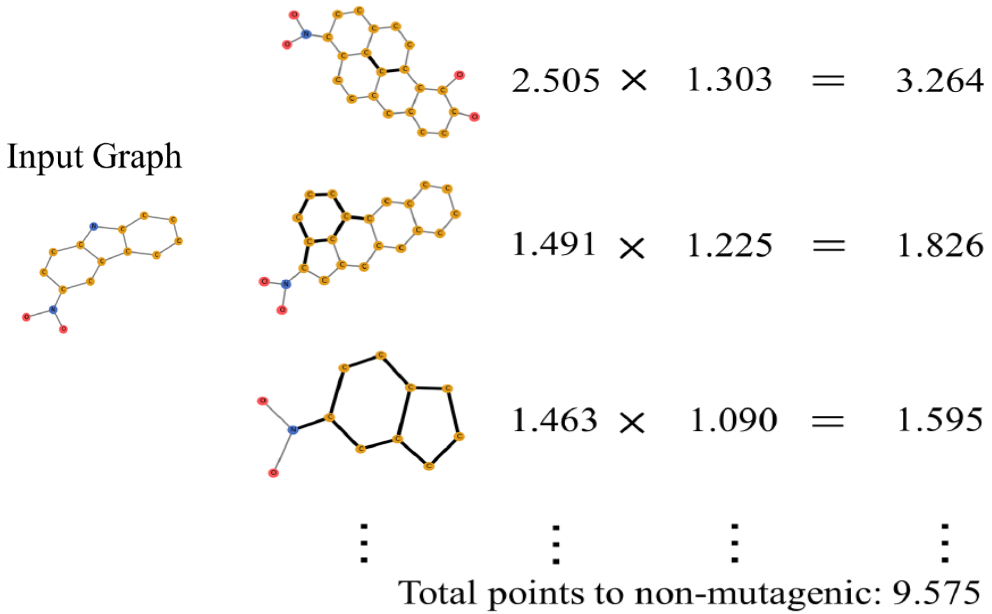
Functional Group

돌연변이성

분자에 대한 특성을 예측



비돌연변이성

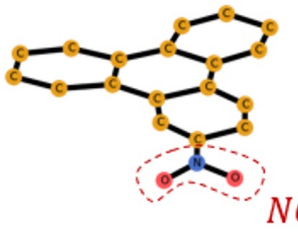


Zhang, Zaixi, et al. "Protgnn: Towards self-explaining graph neural networks." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. No. 8. 2022.

INTRODUCTION

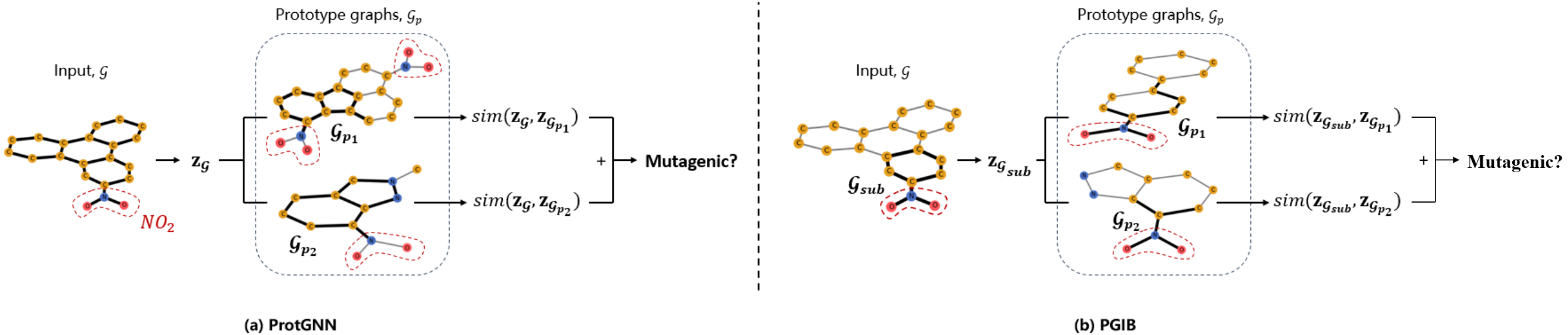
- ProtGNN
  - 학습 과정에서 prototype이 <u>중요한 subgraph를 포착하지 못하거나 불필요한 정보를 포함</u>
  - $\mathbf{NO_2}$(Functional group)를 제외한 다른 공통의 subgraph에 대해서 높은 similarity score를 계산

    → Prototype이 예측에 중요한 결정 근거를 충분히 포함하도록 해야함.



Functional Group

→ Information Bottleneck를 Prototype 관점에서 접근하여 <u>prototype이 예측에 중요한 subgraph에 관한 정보를 전달받음</u>

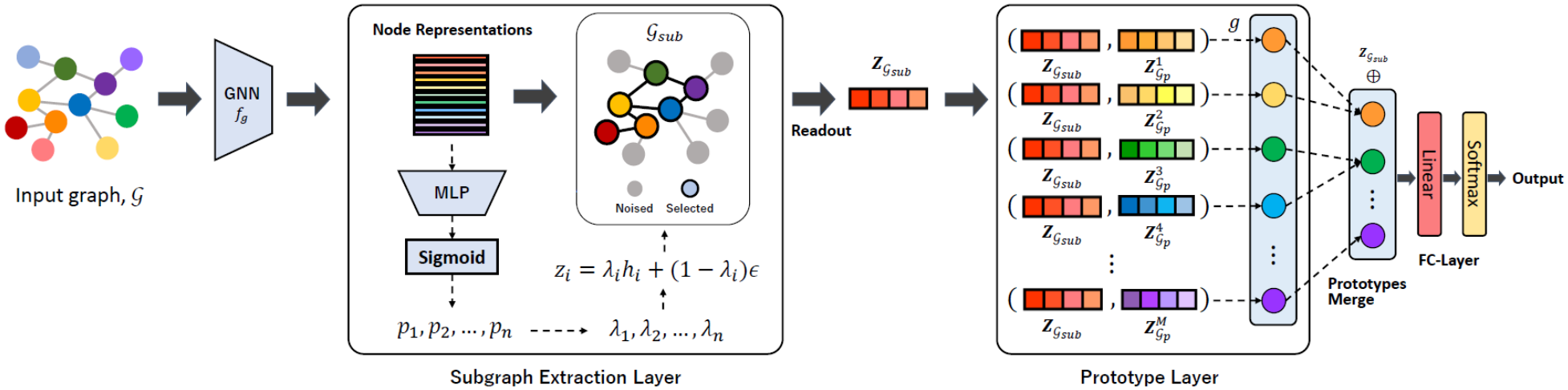


Zhang, Zaixi, et al. "Protgnn: Towards self-explaining graph neural networks." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. No. 8. 2022.

PROPOSED METHOD

Interpretable Prototype-based Graph Information Bottleneck

- Architecture



PGIB

1. Identify core subgraphs.
2. Calculate similarity scores with prototypes.
3. Merge prototypes.
4. Predict the label from the fully connected layer.

$$\min_{\mathcal{G}_{sub}} -I(Y;\mathcal{G}_{sub}) + \beta I(\mathcal{G};\mathcal{G}_{sub})$$

$$\min_{\mathcal{G}_{sub}} \underbrace{-I(Y;\mathcal{G}_{sub},\mathcal{G}_p) + I(Y;\mathcal{G}_p|\mathcal{G}_{sub})}_{\text{Prototype Layer}} + \underbrace{\beta\, I(\mathcal{G};\mathcal{G}_{sub})}_{\text{Subgraph Extraction Layer}}$$

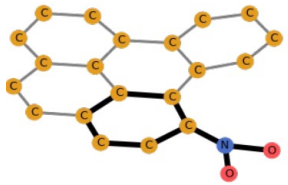
EXPERIMENT

Graph Classification

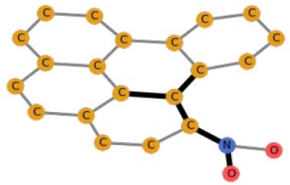
Dataset	Methods								
	GCN	GIN	GAT	ProtGNN	GIB	VGIB	GSAT	**PGIB**	**PGIB$_{cont}$**
MUTAG	74.50±7.89	80.50±7.89	73.50±7.43	80.50±9.07	79.00±6.24	81.00±6.63	80.88±8.94	85.00±7.07	**85.50±5.22**
PROTEINS	72.83±4.23	70.30±4.84	71.35±4.85	73.83±4.22	75.25±5.92	73.66±3.32	69.64±4.71	77.14±2.19	**77.50±2.42**
NCI1	73.16±3.49	75.04±2.08	66.05±1.03	74.13±2.10	64.65±6.78	63.75±3.37	68.13±2.64	77.65±2.20	**78.25±2.13**
DD	72.53±4.51	72.04±3.62	70.81±4.33	69.15±4.33	72.61±8.26	72.77±5.63	71.93±2.74	73.36±1.80	**73.70±2.14**

- 제안된 모델의 그래프 분류 성능이 여러 최신 모델들을 능가함.
- 탐지된 subgraph가 분류성능을 향상시키는데 기여

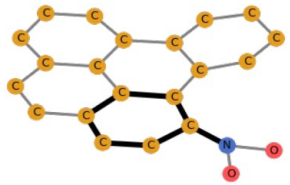
Graph Interpretation



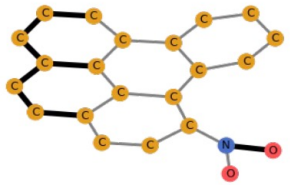
(a) PGIB$_{cont}$



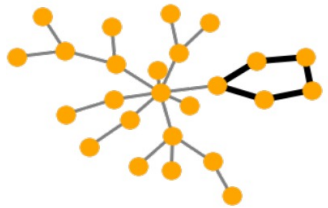
(b) VGIB



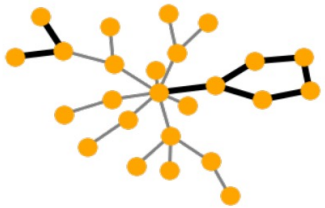
(c) ProtGNN



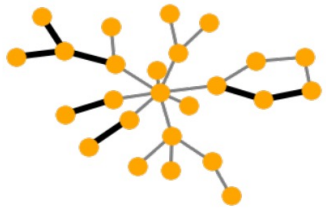
(d) GNNexplainer



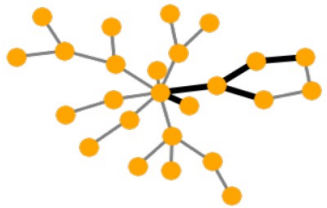
(e) PGIB$_{cont}$



(f) VGIB



(g) ProtGNN



(h) GNNexplainer

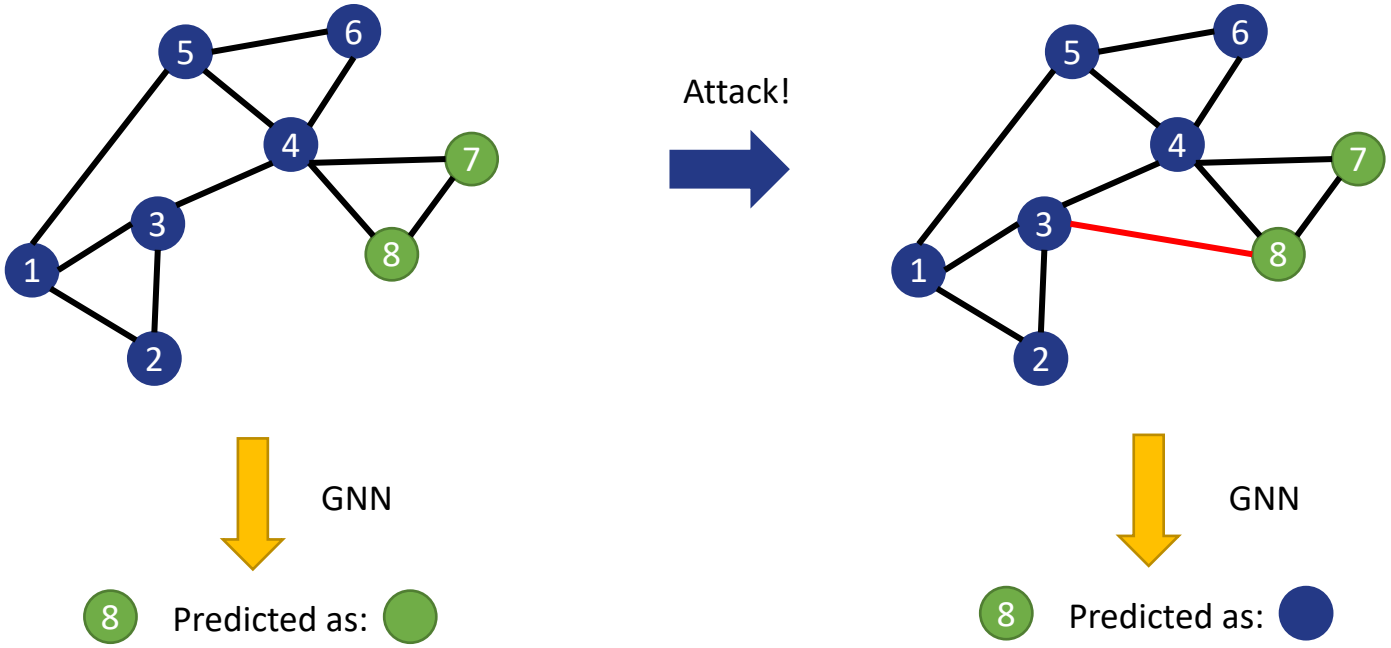
- 제안된 모델에 의해서 탐지된 subgraph가 functional group을 올바르게 포착함

Outline

- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)
  - GNNs for Computer vision (Scene Understanding) (AAAI'23)

Background

Adversarial Attacks on Graph Structures



- Graph Neural Networks are vulnerable to adversarial attacks on graph structures.
- Unsupervised GRL models are also vulnerable to such attacks.

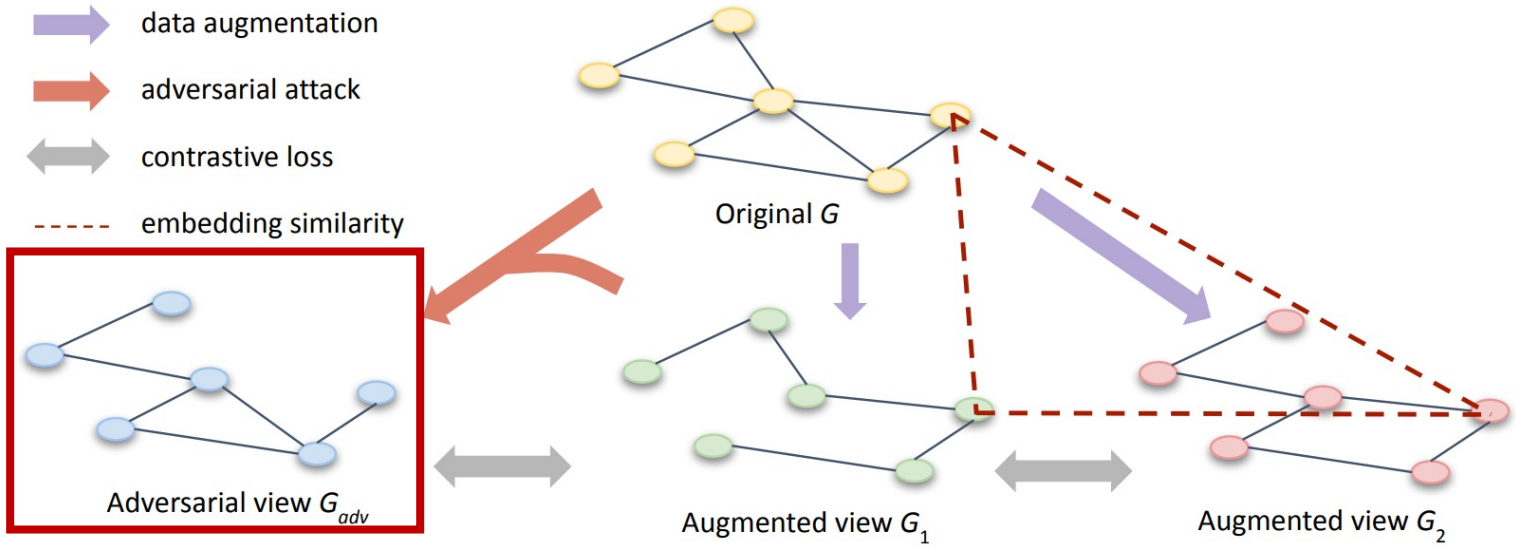
➡ *Leads to the requirements of robust graph representation learning methods*

Figure: Adversarial Attacks and Defenses: Frontiers, Advances and Practice, KDD'20 tutorial

Motivation

Applying Adversarial Training (AT) to Graph Contrastive Learning (GCL)

A graph with the worst-case attack maximizing contrastive loss



Formulation of the adversarial attack in GCL models

$$\delta_{\mathbf{A}}^*, \delta_{\mathbf{X}}^* = \arg \max_{\delta_{\mathbf{A}}, \delta_{\mathbf{X}} \in \Delta} \mathbb{E}\left[\mathcal{L}(f(\mathbf{A}^1 + \delta_{\mathbf{A}}, \mathbf{X}^1 + \delta_{\mathbf{X}}), f(\mathbf{A}^2, \mathbf{X}^2))\right]$$

Contrastive loss

$$\Delta = \{(\delta_{\mathbf{A}}, \delta_{\mathbf{X}}) | \|\delta_{\mathbf{A}}\|_0 \leq \Delta_{\mathbf{A}}, \|\delta_{\mathbf{X}}\|_0 \leq \Delta_{\mathbf{X}}\}$$

Perturbation budgets

- **Goal:** to find the optimal edges and node feature perturbations for the $\mathbf{A}^1, \mathbf{X}^1$ that maximally increase the contrastive loss.
- Since we consider unsupervised adversarial attacks, a contrastive loss is employed instead of a supervised loss.

Figure: Adversarial Graph Contrastive Learning with Information Regularization, WWW 2022

Motivation

Characteristic of Adversarial Attacks on GCL

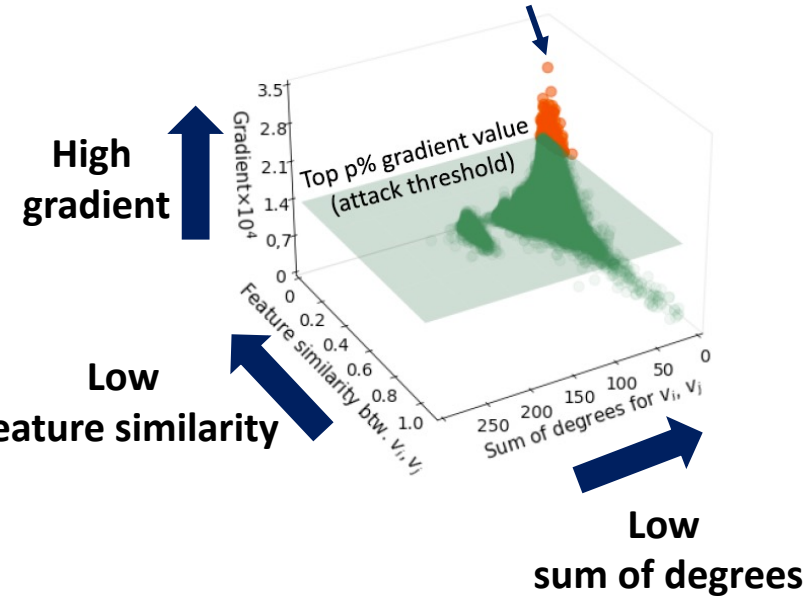
Assumption for simplicity
- GCL model with a 1-layer GCN w/o nonlinearity.
- Perturbs only one edge $v_i \rightarrow v_k$.
- Attacked graph $(\mathbf{A}^1 + \boldsymbol{\delta}_A, \mathbf{X}^1)$
- $\mathbf{Z}^{atk} = f(\mathbf{A}^1 + \boldsymbol{\delta}_A, \mathbf{X}^1)$

- If $\mathbf{z}_i^2 - \mathbf{z}_i^{atk}$ is large, $\boldsymbol{\delta}_A$ is effective perturbation.
- $\mathbf{z}_i^2 - \mathbf{z}_i^{atk}$ is computed as follows:

$$\mathbf{z}_i^2 - \mathbf{z}_i^{\mathrm{atk}} = (\mathbf{z}_i^2 - \mathbf{z}_i^1) + (\mathbf{z}_i^1 - \mathbf{z}_i^{\mathrm{atk}})$$

$$= \mathbf{e}_i + \underbrace{\frac{1}{\sqrt{|\mathcal{N}_{\mathbf{A}^1}^i| + 1}}}_{\text{Degree term}} \underbrace{\left(\sum_{j \in \mathcal{N}_{\mathbf{A}^1}^i \cup \{i\}} \frac{\alpha \mathbf{W} \mathbf{x}_j}{\sqrt{|\mathcal{N}_{\mathbf{A}^1}^i|} \sqrt{|\mathcal{N}_{\mathbf{A}^1}^j|}} - \frac{\mathbf{W} \mathbf{x}_k}{\sqrt{|\mathcal{N}_{\mathbf{A}^1}^k| + 1}} \right)}_{\text{Feature difference term}} \quad (4)$$

Each point: the node pair



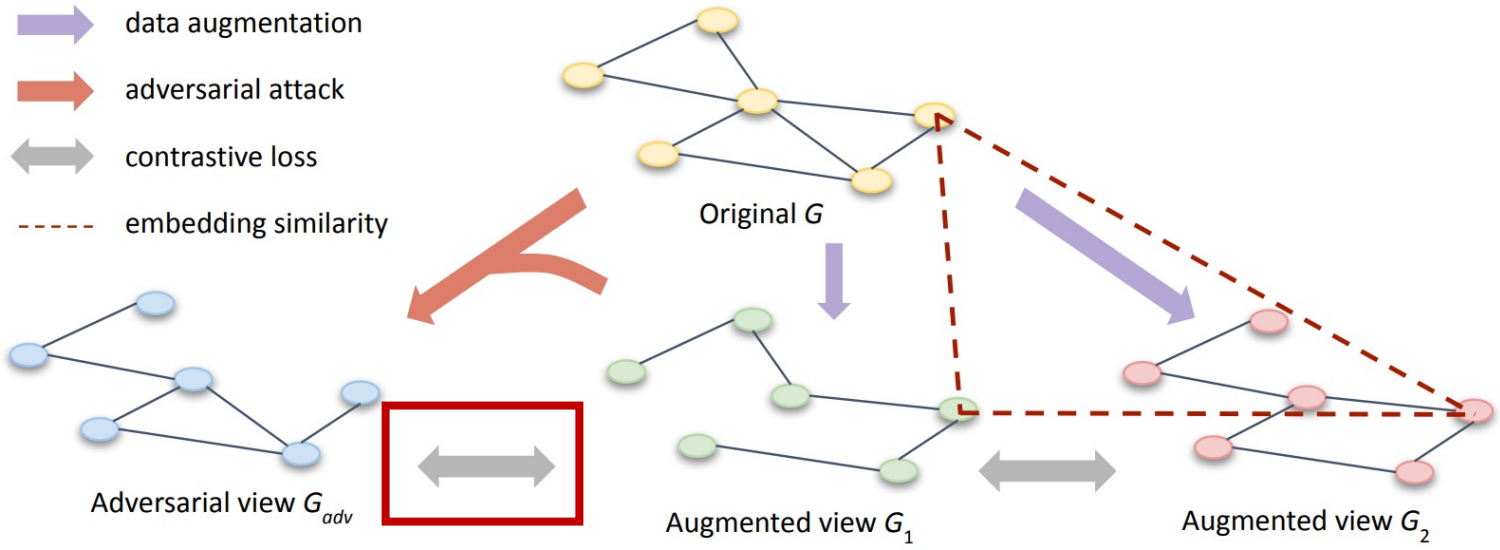
- $\mathbf{z}_i^2 - \mathbf{z}_i^{atk}$ becomes large when degree term ↓ and feature diff. term ↑
  - The degree of v_i is small (*low-degree nodes*)
  - The *features of node v_k (i.e., $\mathbf{x}_k$) is dissimilar* from the aggregation of neighborhood features in a clean graph.

Characteristic of a generated adversarial view by contrastive loss
1. Attack the nodes that have low-degree.
2. **Connect the nodes with dissimilar feature**

Motivation

Applying Adversarial Training (AT) to Graph Contrastive Learning (GCL)

Using the attacked graph as an additional augmentation !



Formulation of Adversarial Graph Contrastive Learning (AGCL)

$$\min_{\Theta} \underbrace{\mathcal{L}(\mathbf{Z}^1, \mathbf{Z}^2)}_{\text{GCL term}} + \lambda_1 \underbrace{\mathcal{L}(\mathbf{Z}^1, \mathbf{Z}^{\text{adv}})}_{\text{AT term}}$$

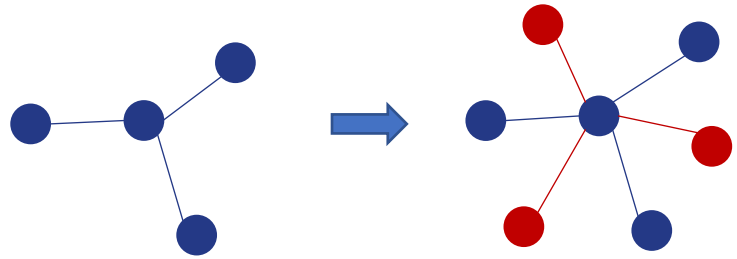
$$\mathbf{Z}^{\text{adv}} = \underbrace{f(\mathbf{A}^1 + \delta_{\mathbf{A}}^*, \mathbf{X}^1 + \delta_{\mathbf{X}}^*)}_{\text{Adversarial graph view}}$$

- **Goal:** robust graph representation learning based on adversarial training (AT).
- **Main idea:** to force the representations in the clean graph to be close to those of the attacked graphs.
  - The adversarial graph contrastive learning model minimizes the training objective.

Figure: Adversarial Graph Contrastive Learning with Information Regularization, WWW 2022

Motivation

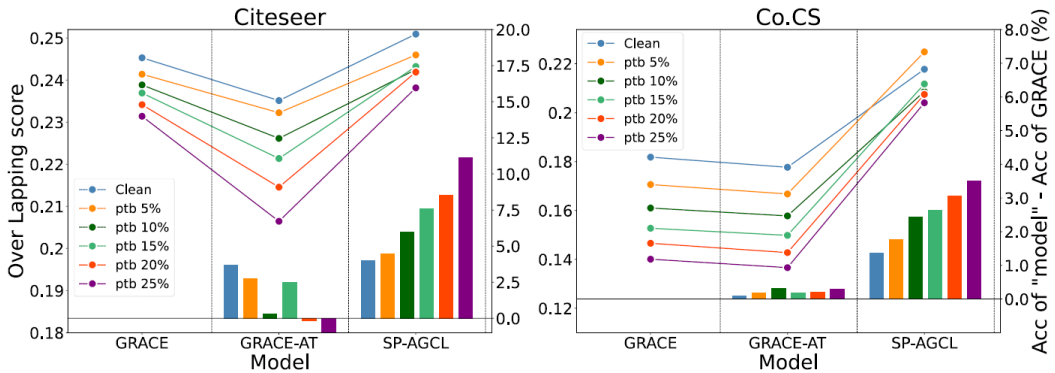
AT fails to preserve node similarity !



- As previously demonstrated, adversarial attacks on graphs tend to connect nodes with dissimilar features.
  - ➢ The neighborhood feature distribution is changed by the adversarial attacks.

- And AGCL reduces the distance between the clean view and the adversarial view to achieve robustness.
  - ➢ Neglecting the changes in the neighborhood feature distributions in the adversarial view.

We argue that existing AGCL models obtain robustness *at the expense of losing the feature information.*



- Solid line: OL score
- Bar plot: performance improvement compared to GRACE

$$OL(\mathbf{A}^{k\mathrm{NN}(\mathbf{Z})}, \mathbf{A}^{k\mathrm{NN}(\mathbf{X})}) = \frac{|\mathbf{A}^{k\mathrm{NN}(\mathbf{Z})} \cap \mathbf{A}^{k\mathrm{NN}(\mathbf{X})}|}{|\mathbf{A}^{k\mathrm{NN}(\mathbf{X})}|}$$

- indicates how much the feature information the representations have

We observe
- GRACE-AT have higher accuracy than GRACE
  - ➢ They obtain robustness.
- GRACE-AT have lower OL score than GRACE
  - ➢ They lose the feature information.

Motivation

Node similarity preservation is crucial !

- As previously demonstrated, existing AGCL models obtain robustness *at the expense of losing the feature information.*
- However, the node feature information is crucial for the robustness against graph structure attacks [1, 2].

We argue that the robustness of AGCL model can be further enhanced by fully exploiting the node feature information.

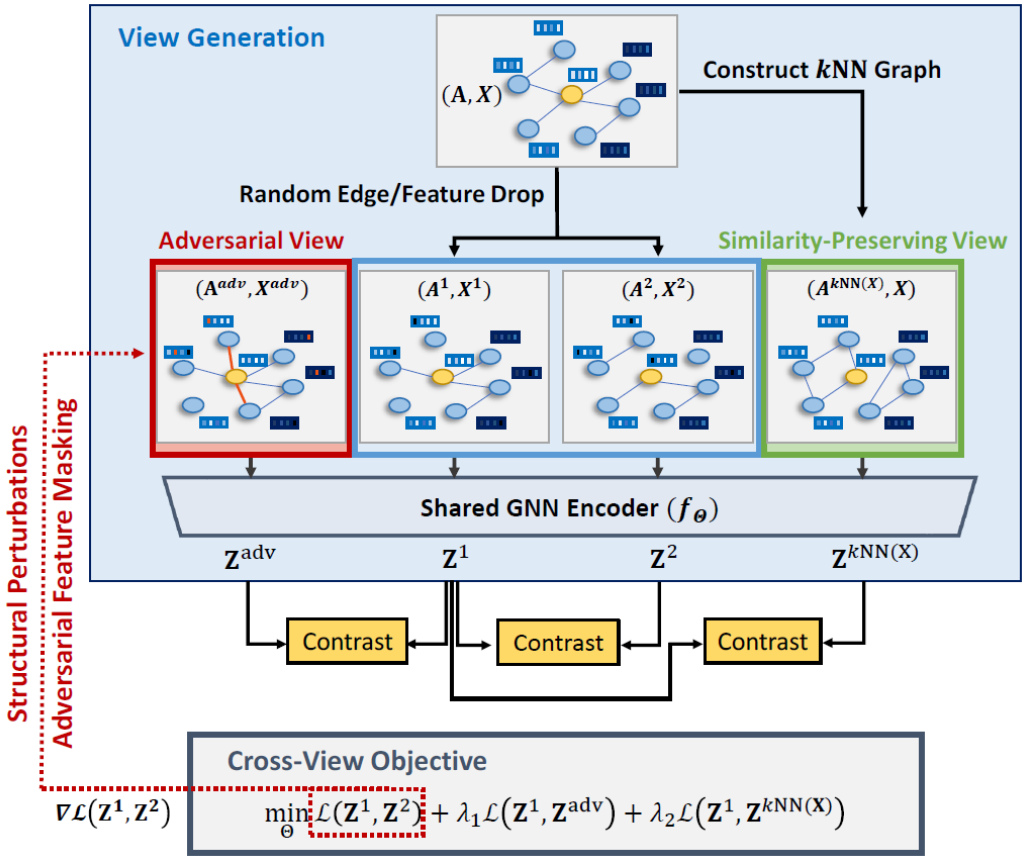
- Moreover, preserving the node feature similarity becomes especially useful for most real-world graphs.
  - Graphs with noisy node labels
  - Graphs with heterophilous neighbors
  - Low-degree nodes

➡ To this end, we propose a *similarity-preserving adversarial graph contrastive learning* (SP-AGCL) framework

[1] Graph Structure Learning for Robust Graph Neural Networks, KDD 2020
[2] Node Similarity Preserving Graph Convolutional Networks, WSDM 2021

Proposed Method

Similarity Preserving Adversarial Graph Contrastive Learning (SP-AGCL)



View generation

- *Step 1.* Two stochastically augmented views, $(\mathbf{A}^1, \mathbf{X}^1)$ and $(\mathbf{A}^2, \mathbf{X}^2)$
  - Same as the previous GCL models

- *Step 2.* Adversarial View
  - Structural perturbations

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}^1} + \frac{\partial \mathcal{L}}{\partial \mathbf{A}^2} = \mathbf{G}_\mathbf{A} \in \mathbb{R}^{N \times N}$$

  - Adversarial feature masking

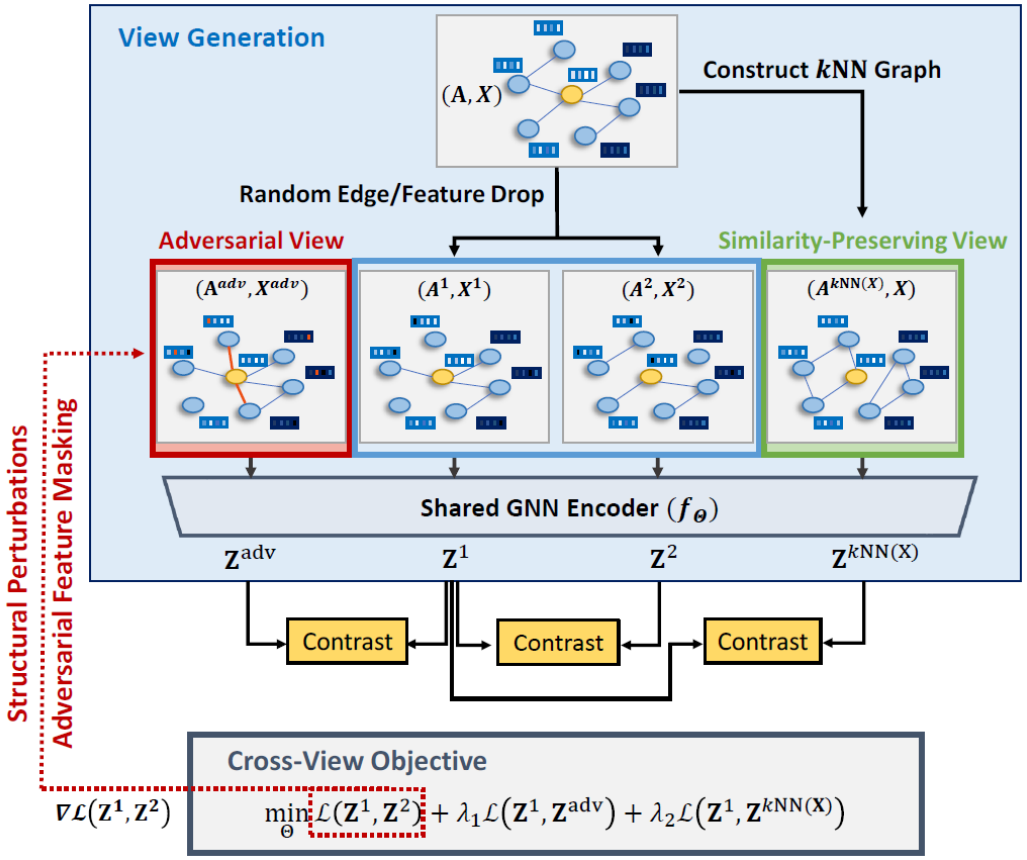
$$\frac{\partial \mathcal{L}}{\partial \mathbf{X}^1} + \frac{\partial \mathcal{L}}{\partial \mathbf{X}^2} = \mathbf{G}_\mathbf{X} \in \mathbb{R}^{N \times F}$$

    - Existing works flip the node feature
    - But, it corrupts the co-occurrence/correlation statistics.
    - By masking instead of flipping, we maintaining them.

- *Step 3.* Similarity preserving view
  - Aims to preserve the node feature similarity.
  - kNN graph of node features $\left(\mathbf{A}^{k\mathrm{NN}(\mathbf{X})}, \mathbf{X}\right)$

Proposed Method

Similarity Preserving Adversarial Graph Contrastive Learning (SP-AGCL)



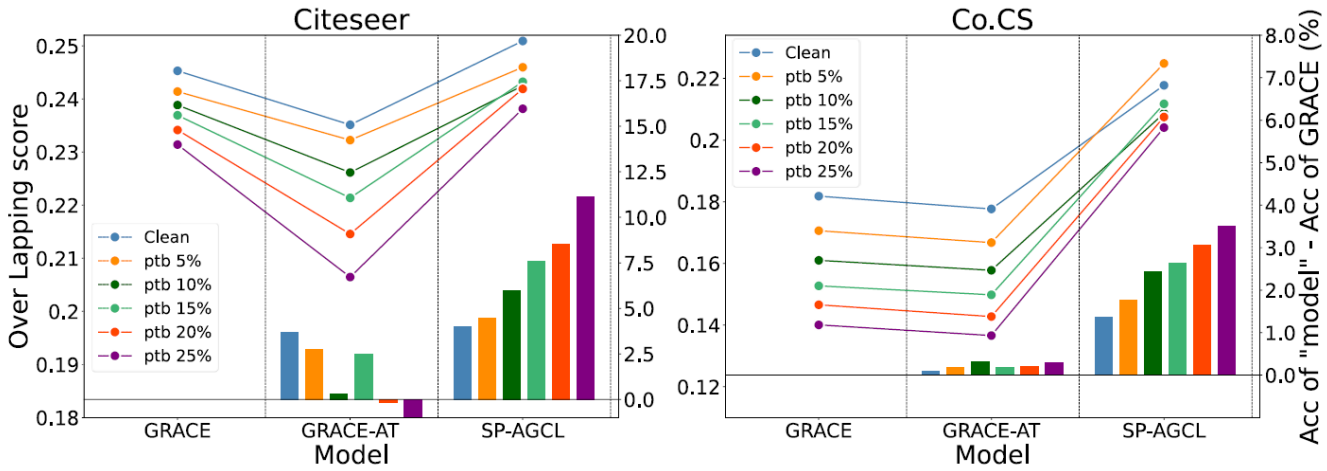
Cross-view Training for Robust GCL

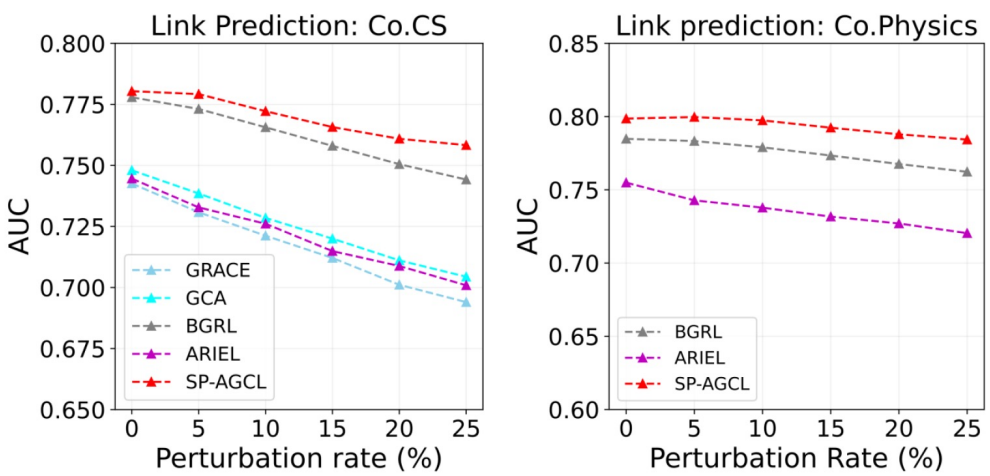
$$\min_{\Theta} \underbrace{\mathcal{L}(\mathbf{Z}^1, \mathbf{Z}^2)}_{\text{GCL term}} + \lambda_1 \underbrace{\mathcal{L}(\mathbf{Z}^1, \mathbf{Z}^{\text{adv}})}_{\text{AT term}} + \lambda_2 \underbrace{\mathcal{L}(\mathbf{Z}^1, \mathbf{Z}^{k\text{NN}(\text{X})})}_{\text{Similarity-preserving term}}$$

The **representations of nodes with similar features are pulled together** , which in turn preserves the node feature similarity.

Experiment

Preserving Feature Similarity is beneficial !





- SP-AGCL preserves the node feature similarity, which results in the robust graph representation.

- SP-AGCL consistently predicts reliable links compared with other baselines across all the perturbation ratios.
  - Moreover, ARIEL, the sota AGCL model, shows the worst performance

Node feature information is beneficial to predicting reliable links
since nodes with similar features tend to be adjacent in many real-world graphs.

Outline

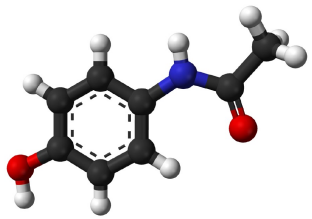
- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)
  - GNNs for Computer vision (Scene Understanding) (AAAI'23)

Outline

- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)
  - GNNs for Computer vision (Scene Understanding) (AAAI'23)

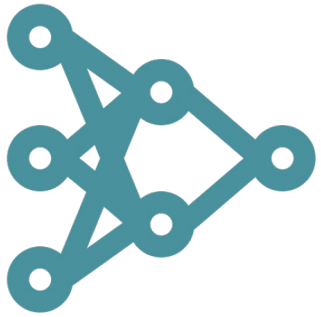
Introduction: Molecular Property Prediction

- Predict the properties of a molecule (소재 물성 예측)



Features
- **Reaction Rate**
- **Pressure**
- **Bond Type**
- **Activation Energy**
- **Stoichiometry**
- **Bond Energy**
- **Intermolecular Forces**
- **Temperature ⋯**

→



Graph Neural Network

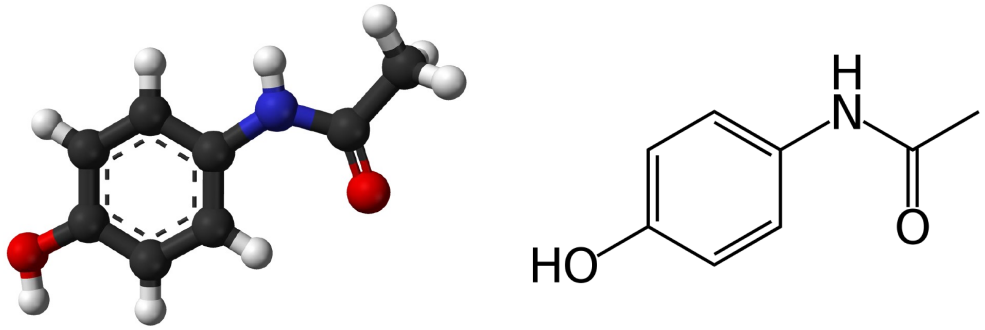
→

Prediction

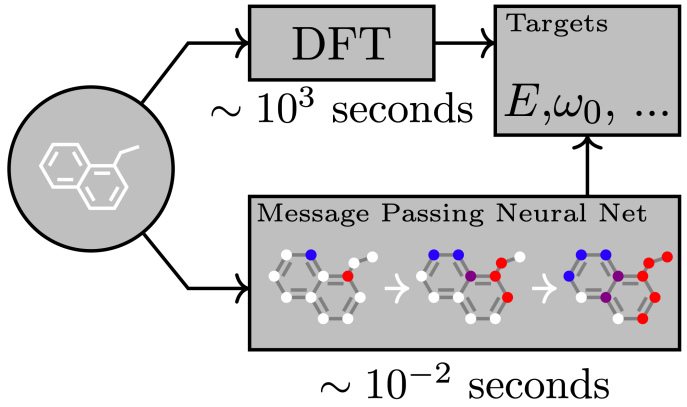
ex) Band gap, DOS, Fermi

Molecular Graphs

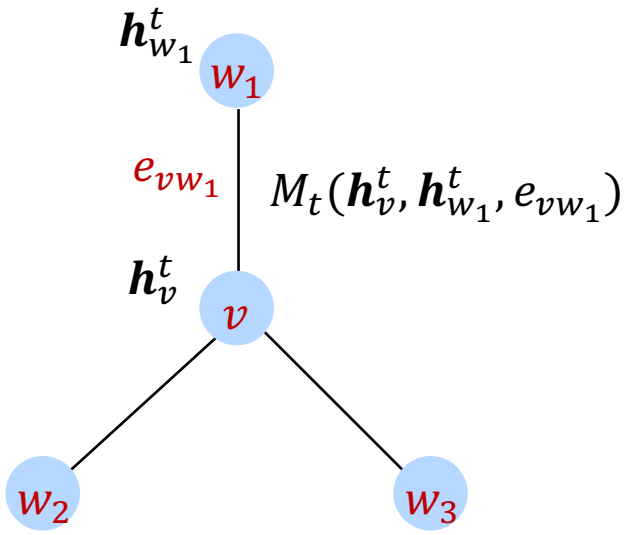
- Molecules can be represented as a graph with node features and edge features
  - Node features: atom type, atom charges…
  - Edge features: valence bond type…



Message Passing Neural Network



- Unified various graph neural network and graph convolutional network approaches



Edge embedding

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

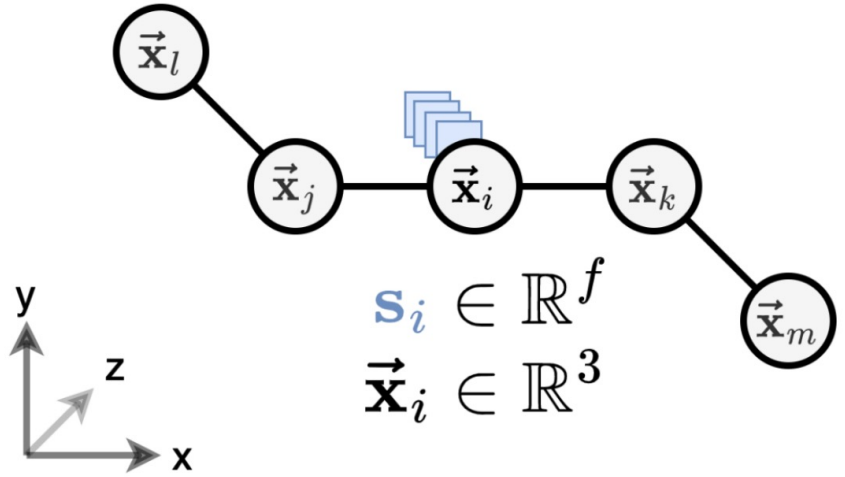
Neighbor of v

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

$$\hat{y} = R(\{h_v^T \mid v \in G\})$$

Geometric Graphs

- Sometimes, we also know the 3D positions of atoms, which is actually more informative
- A geometric graph $G = (A, S, X)$ is a graph where each node is embedded in d-dimensional **Euclidean space**:

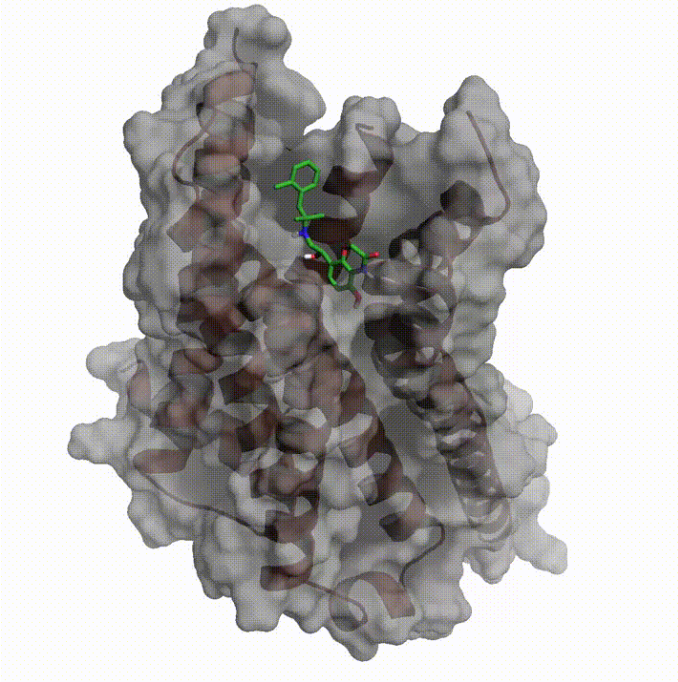


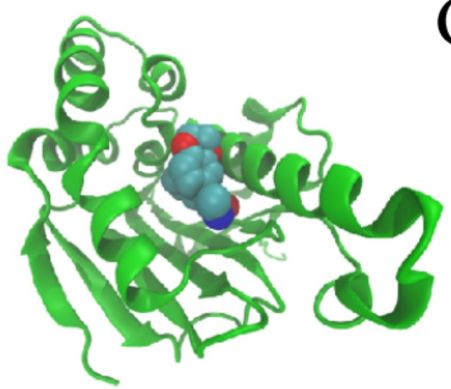
- A: an $n \times n$ adjacency matrix
- $S \in R^{n \times f}$: Scalar features (atom type, atom charges, …)
- $X \in R^{n \times d}$: tensor features, e.g., coordinates

http://web.stanford.edu/class/cs224w/slides/17-geometric.pdf

Broad Impact on Sciences

- Supervised Learning: Prediction
  - Properties prediction
  - 3D Protein-ligand interaction (binding)



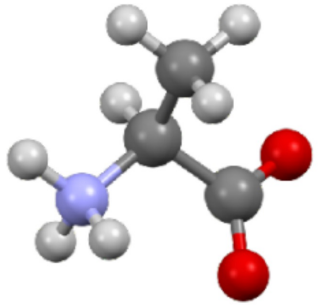


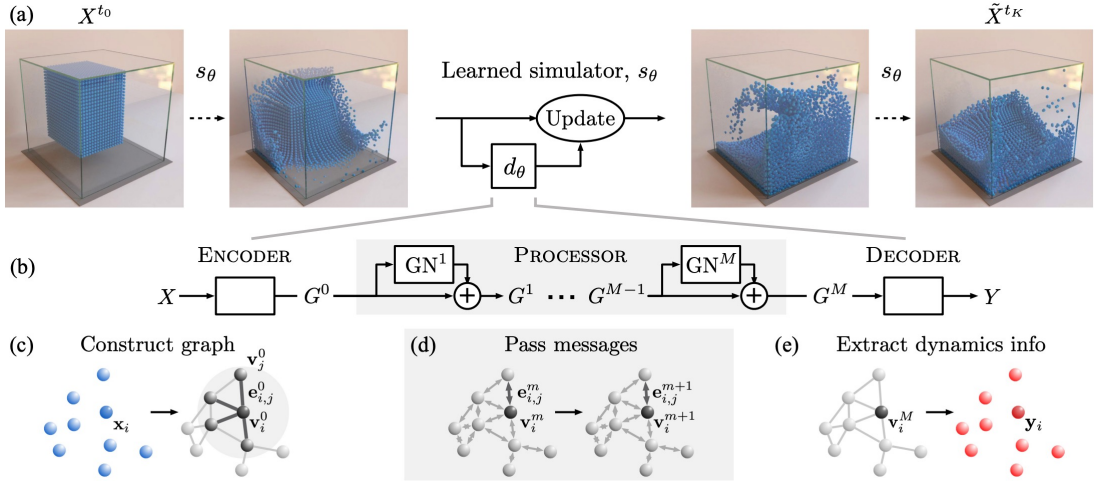
Geometric Graph → Geometric GNN → Prediction

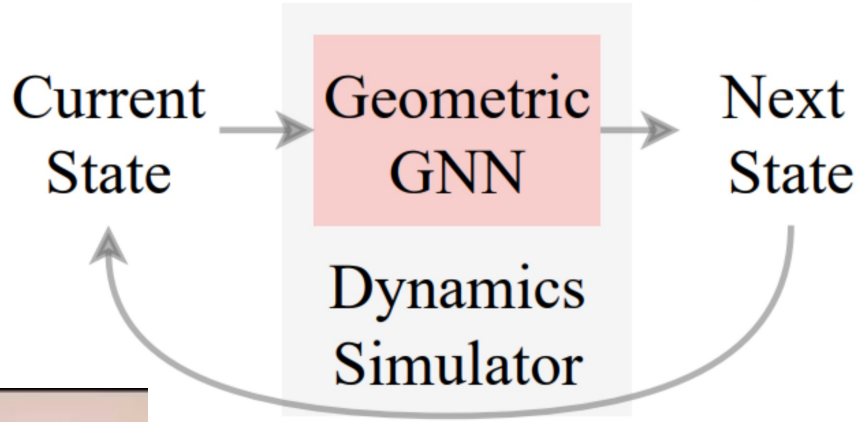
- Functional properties?
- Ligand binding affinity?
- Ligand efficacy?

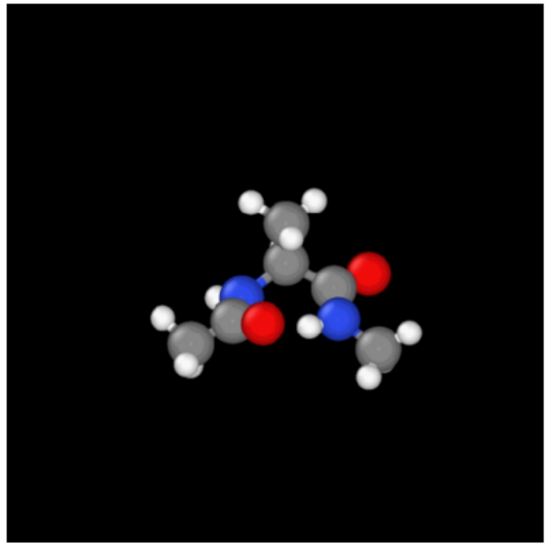
Broad Impact on Sciences

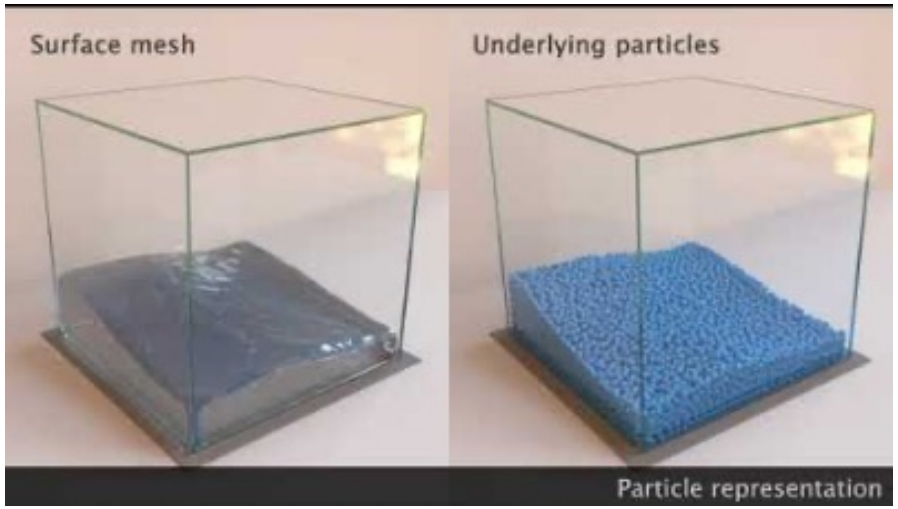
- Supervised Learning: Structured Prediction
  - Molecular Simulation







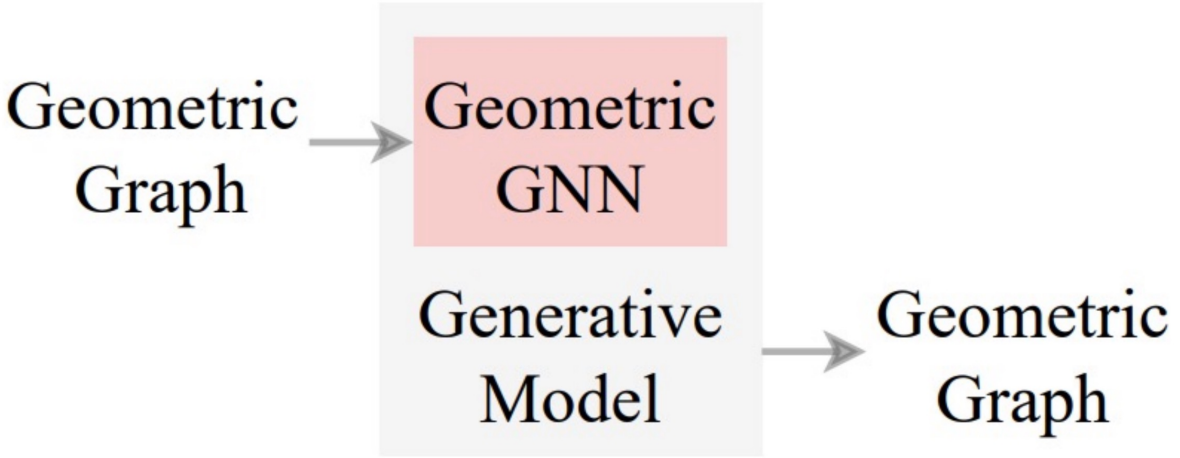


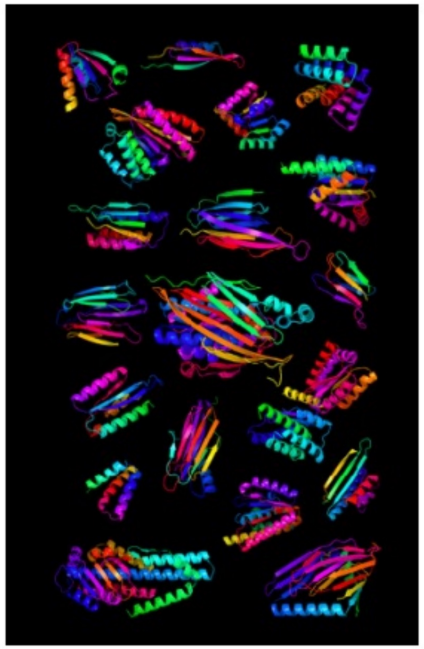


http://web.stanford.edu/class/cs224w/slides/17-geometric.pdf

Broad Impact on Sciences

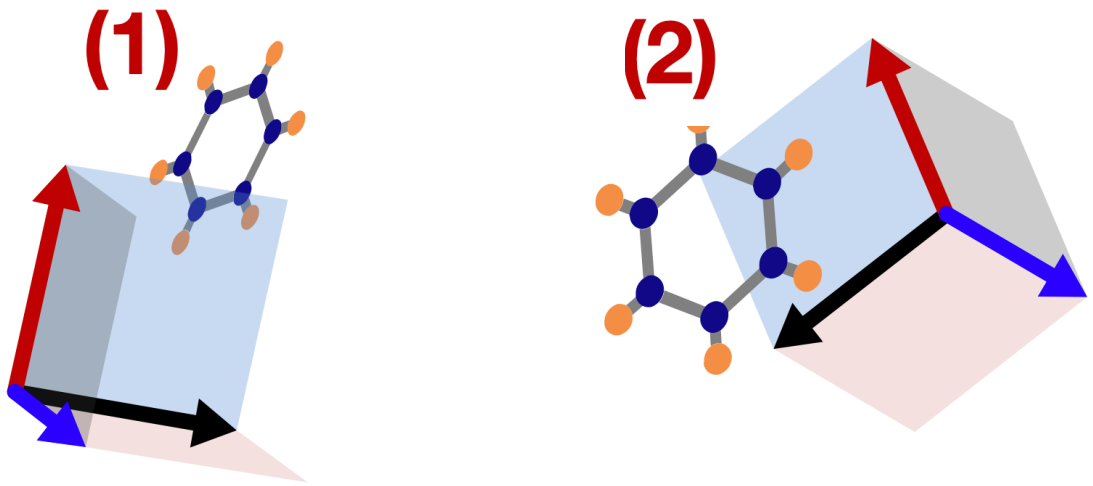
- Generative Models
  - Drug or material design





Geometric graph is more challenging than Molecular graph

- To describe geometric graphs, we use **coordinate systems**
  - **(1)** and **(2)** use different coordinate systems to describe the **same** molecular geometry.
- We can describe the transform between coordinate systems with **symmetries** of Euclidean space
  - 3D rotations, translations



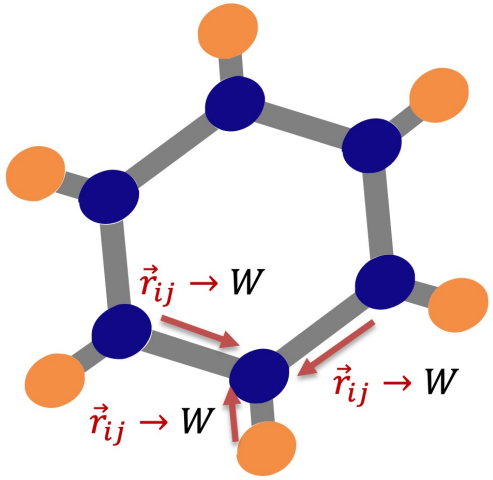
However, output of traditional GNNs given **(1)** and **(2)** are completely different!
→ Enforcing symmetry is crucial (Invariant GNNs)

Schnet: Overview

- **Input**
  - Feature representations of n atoms $X^l = (\boldsymbol{x}_1^l, \dots, \boldsymbol{x}_n^l)$ with $\boldsymbol{x}_i^l \in R^F$
  - At locations $R = (\boldsymbol{r}_1, \dots, \boldsymbol{r}_n)$ with $\boldsymbol{r}_i \in R^D$ ($D = 3$ for 3-dim coordinates)
- **Output**
  - Molecular total energy $E(\boldsymbol{r}_1, \dots, \boldsymbol{r}_n)$
- SchNet updates the node embeddings at the l-th layer by message passing layers

$$\mathbf{x}_i^{l+1} = (X^l * W^l)_i = \sum_j \mathbf{x}_j^l \circ W^l(\mathbf{r}_i - \mathbf{r}_j),$$

- A filter generating function $W^l: R^D \to R^F$ is determined by the relative position from neighbor atoms j to i
- $\circ$ is the element-wise multiplication



x^l: node embeddings at l layer
r: atomic coordinates

Schnet: Invariance

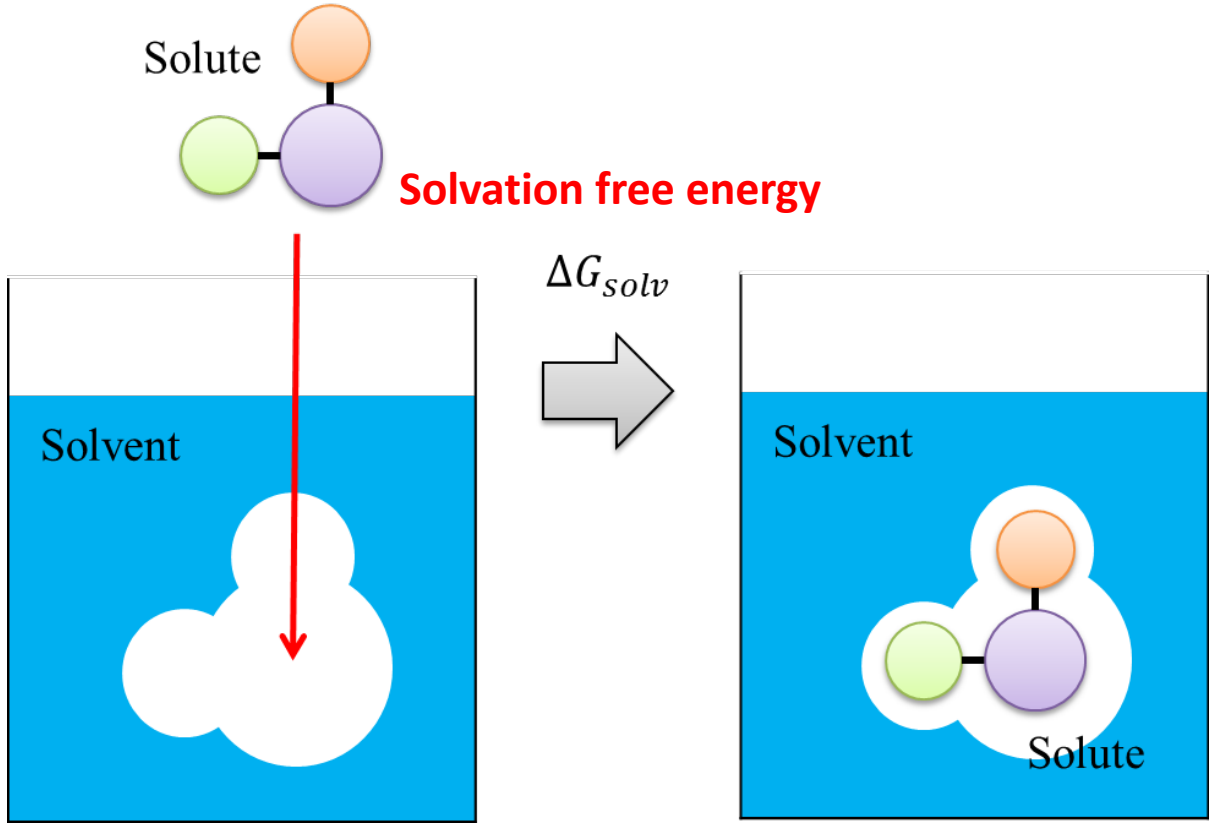
- W is invariant by scalarizing relative positions with relative distances ($\left\|r_i - r_j\right\| = \left\|r_{ij}\right\| = d_{ij}$)
  - $\|r_{ij}\|$ is invariant to **rotations** and **translations**
- Hence, each message passing layer W^l is invariant

→ Aggregated node embeddings $\sum_j \mathbf{x}_j^l \circ W^l(\mathbf{r}_i - \mathbf{r}_j)$ is invariant

→ **Node embeddings are invariant!**

Predicting Solvation Free Energy (용매화 자유 에너지)

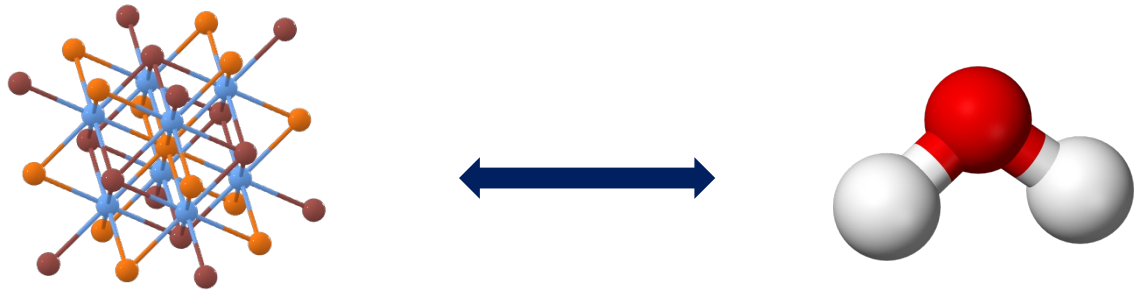
- **Solvation free energy**
  - Change in free energy for a molecule to be transferred from gas phase to a given solvent
  - Quantifies **solubility** of drug molecules
    - A large negative value → high solubility
    - A lower magnitudes/positive value → poor solubility



https://www.ibric.org/upload/geditor/201704/0.95691900_1493095501.png

Introduction: Relational Learning

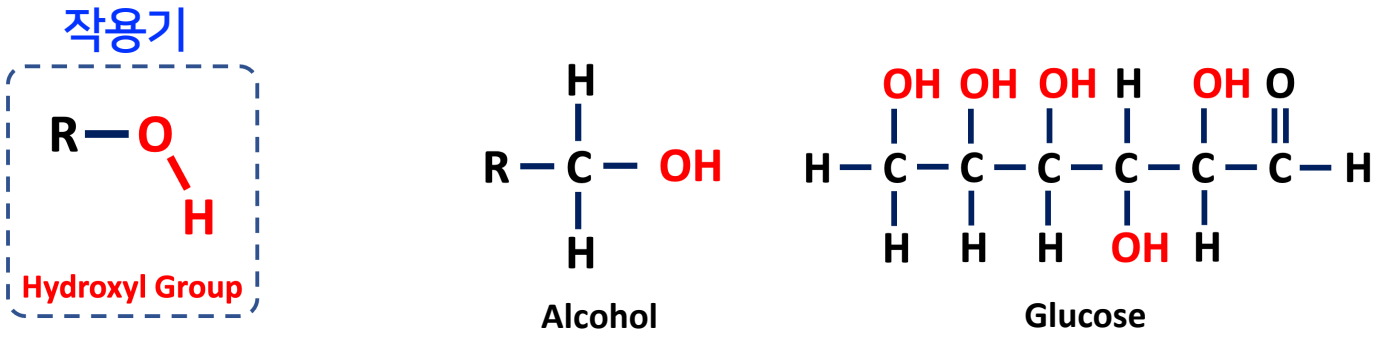
- Relational Learning
  - 두 개체 사이의 관계를 예측하는 기계 학습 분야
  - 특히, 두 화학 물질 사이의 관계를 예측하는 것은 화학 분야에서 매우 중요함



  - Examples
    - 발색체 (Chromophore)와 용매 (Solvent)가 반응했을 때의 광학적 성질 예측
    - 용질 (Solute)와 용매 (Solvent) 가 반응했을 때의 용해도 예측
    - 두 종류의 약물 (Drug) 을 동시에 섭취했을 때의 부작용 예측

Introduction: Functional Group

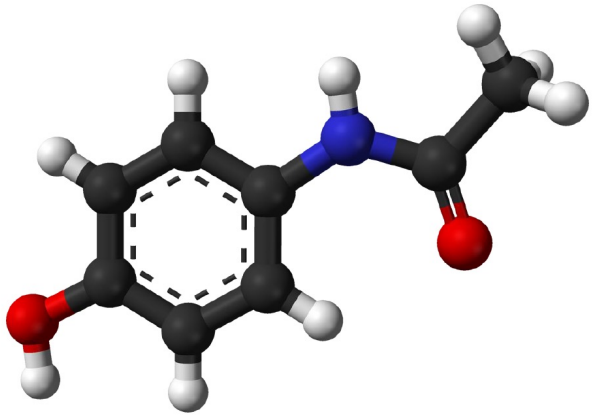
- **작용기 (Functional Group)**
  - 유기화합물의 **화학 반응적 특성을 결정하는데 중요한 역할**을 하는 특정 원자단 혹은 구조
  - 유기화학 반응 전후에 **작용기의 변환에 따라 반응물과 생성물의 물리적/화학적 성질이 변화**함
  - 같은 **작용기를 갖는 화합물의 특성은 대부분 유사**하며, 유사한 화학 반응이 일어남
  - Examples
    - Hydroxyl Group 구조는 분자의 극성을 증가시키는 특성을 가지고 있음
    - → Alcohol, Glucose 같이 Hydroxyl 구조를 포함한 분자들은 공통적으로 물에 대해 용해도가 높음



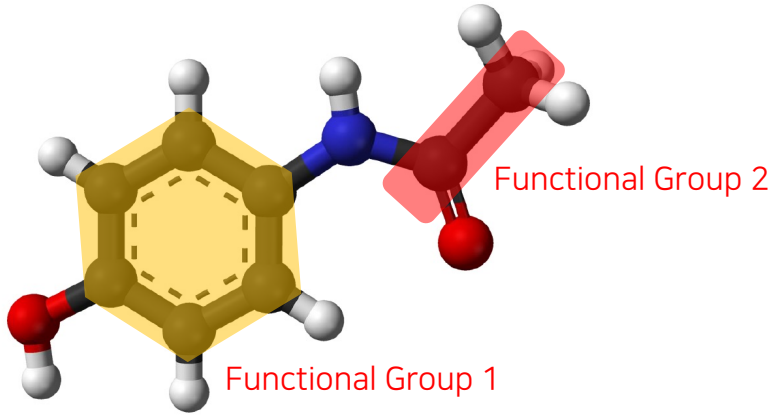
따라서, 화학 물질 사이의 상호작용을 예측하는 모델은 작용기를 기반으로 학습하는 것이 매우 중요함

Introduction: Representing Molecules as a Graph

- Molecule → Graph로 표현 가능
- Functional Group → Subgraph 로 표현 가능



Molecule
(=Graph)



Functional Group
(=Subgraph)

최근 Information Theory를 기반으로 Graph 구조에서 중요한 Subgraph를 찾는 모델이 주목

Information Bottleneck

- 기계학습 관점에서 그래프에서 중요한 Subgraph를 어떻게 찾아낼 것인가?
- Solution: Information Bottleneck Theory
  - 정보에 대한 압축 / 보존의 trade-off 에 대해 Theoretical 한 방법론
  - Random variable X, Y 가 주어졌을 때, Y 의 정보를 최대한 유지하면서 X 의 정보를 최소한으로 담는 Bottleneck variable T를 학습
    - → 즉, input데이터 X를 최대한 압축하면서 target 값 Y는 계속 잘 맞출 수 있는 T를 학습
    - → Noise 에 robust한 representation 을 학습할 때 많이 사용됨

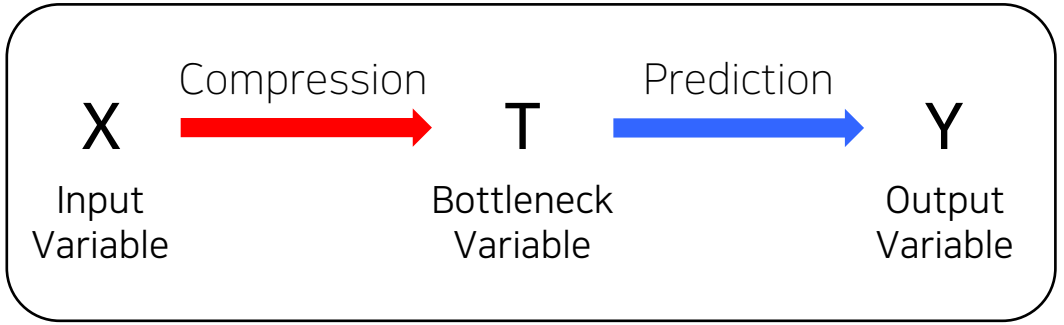
$$\min_T -I(Y;T) + \beta I(X;T)$$

X와 T 사이의 상호 정보량 최소화
→ T가 X 에 대한 정보를 최소한으로 갖도록
→ Compression

T와 Y 사이의 상호 정보량 최대화
→ T가 Y 에 대한 정보를 최대한 가지고 있어야함
→ Prediction

Information Bottleneck Objective

($I(X,Y)$: Mutual information between X and Y)

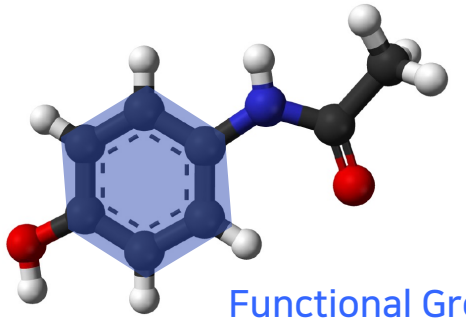


Graph Information Bottleneck: Overview

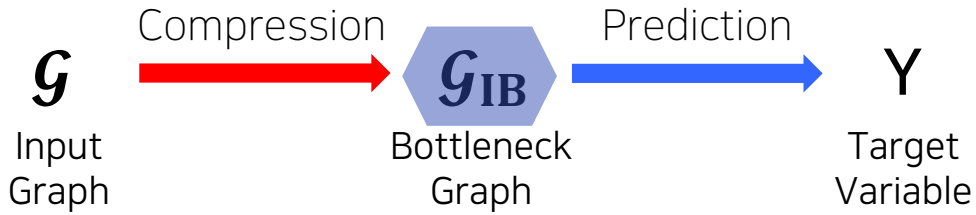
- Information bottleneck theory를 그래프에 어떻게 적용할 것인가?
- Information Bottleneck Graph (IB-Graph)
  - 기존 Graph 의 성질을 최대한 보존하는 Subgraph
  - Subgraph를 Bottleneck variable 로 모델링

    → Target Y를 맞추는데 가장 중요한 Subgraph G_{IB}를 찾는 문제로 Formulation

$$\mathcal{G}_{\mathrm{IB}} = \arg\min_{\mathcal{G}_{\mathrm{IB}}} -I(\mathrm{Y};\mathcal{G}_{\mathrm{IB}}) + \beta I(\mathcal{G};\mathcal{G}_{\mathrm{IB}})$$

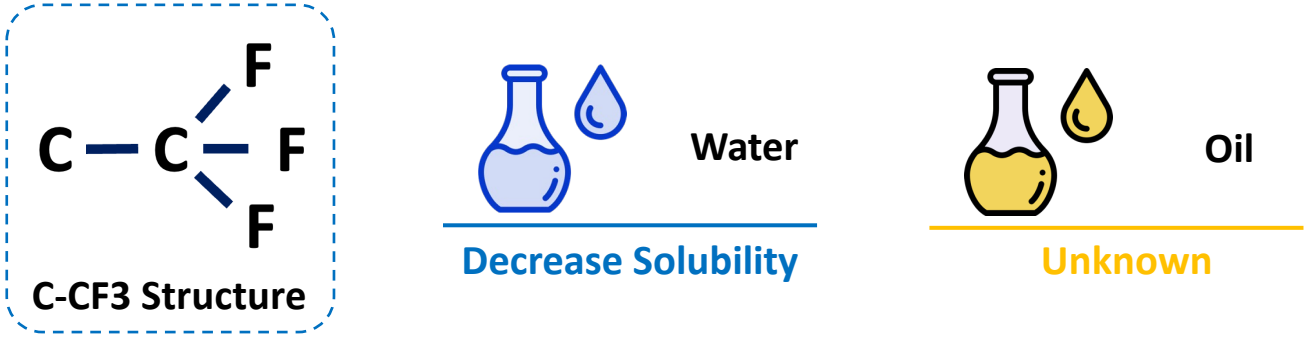


Functional Group



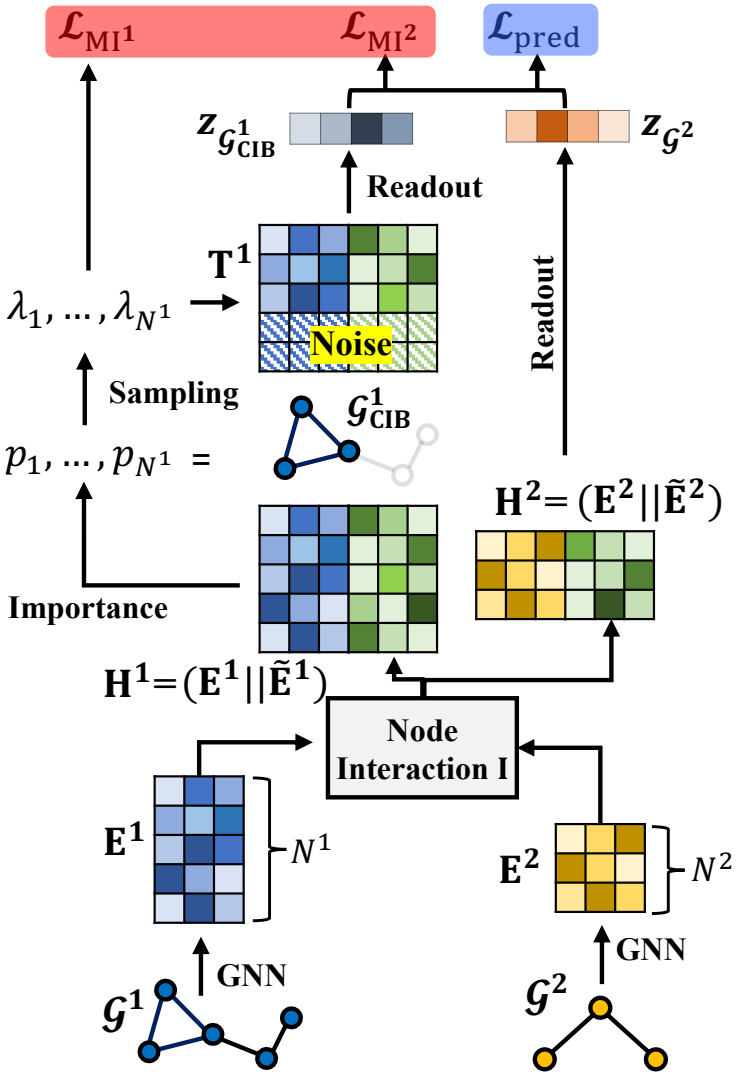
Recall: Functional Group

- **작용기 (Functional Group)**
  - 유기화합물의 **화학 반응적 특성을 결정하는데 중요한 역할**을 하는 특정 원자단 혹은 구조
  - 같은 **작용기를 갖는 화합물의 특성은 대부분 유사**하며, 유사한 화학 반응이 일어남
- 한편, 용질 (Chromophore)이 어떤 용매 (Solvent)와 반응하는지에 따라서 중요하게 작용하는 작용기가 다름
  - Examples: C-CF3 구조는 분자의 **물**에 대한 용해도를 낮추는 역할을 함
    - 하지만, C-CF3 구조가 분자의 **기름**에 대한 용해도에 미치는 영향에 대해서는 알려진 바가 없음
    - 따라서, 화학 반응에서 중요한 작용기를 추출할 때 용매의 종류를 고려할 필요성이 있음



기존의 Information Bottleneck이론으로는 Material Science지식을 제대로 모델링할 수 없음

Proposed Method: Conditional Graph Information Bottleneck



$$\min -I(\mathrm{Y}; \mathcal{G}^1_{\text{CIB}} | \mathcal{G}^2) + \beta I(\mathcal{G}^1; \mathcal{G}^1_{\text{CIB}} | \mathcal{G}^2)$$

Mutual Information 의 Chain Rule 에 따라 Conditional Mutual Information 을 분해 후 각 term 의 upper bound 유도

$$-I(\mathrm{Y}; \mathcal{G}^1_{\text{CIB}} | \mathcal{G}^2) = -I(\mathrm{Y}; \mathcal{G}^1_{\text{CIB}}, \mathcal{G}^2) + I(\mathrm{Y}; \mathcal{G}^2)$$

$$-I(\mathrm{Y}; \mathcal{G}^1_{\text{CIB}}, \mathcal{G}^2) \le \mathbb{E}_{\mathcal{G}^1_{\text{CIB}}, \mathcal{G}^2, \mathrm{Y}}[-\log p_\theta(\mathrm{Y} | \mathcal{G}^1_{\text{CIB}}, \mathcal{G}^2)]$$

Prediction Loss

$$I(\mathcal{G}^1; \mathcal{G}^1_{\text{CIB}} | \mathcal{G}^2) = I(\mathcal{G}^1_{\text{CIB}}; \mathcal{G}^1, \mathcal{G}^2) - I(\mathcal{G}^1_{\text{CIB}}; \mathcal{G}^2)$$

$$I(\mathcal{G}^1_{\text{CIB}}; \mathcal{G}^1, \mathcal{G}^2) \le \mathbb{E}_{\mathcal{G}^1, \mathcal{G}^2}[-\frac{1}{2}\log A + \frac{1}{2N^1}A + \frac{1}{2N^1}B^2]$$

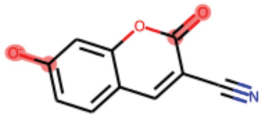
$$:= \mathcal{L}_{\text{MI}^1}(\mathcal{G}^1_{\text{CIB}}, \mathcal{G}^1, \mathcal{G}^2)$$

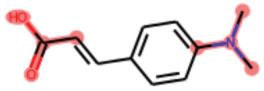
$$-I(\mathcal{G}^1_{\text{CIB}}; \mathcal{G}^2) \le \mathbb{E}_{\mathcal{G}^1_{\text{CIB}}, \mathcal{G}^2}[-\log p_\xi(\mathcal{G}^2 | \mathcal{G}^1_{\text{CIB}})]$$

$$:= \mathcal{L}_{\text{MI}^2}(\mathcal{G}^1_{\text{CIB}}, \mathcal{G}^2)$$

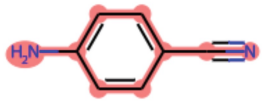
Compression Loss

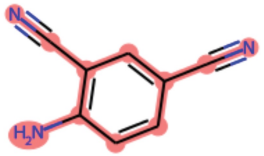
Result: Qualitative analysis





(a) Reaction with ordinary solvent

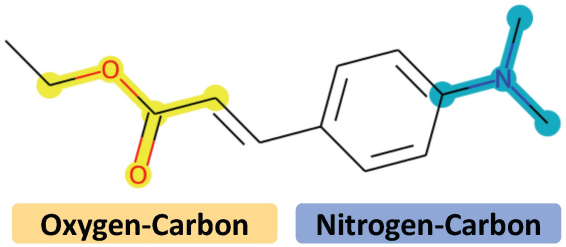




(b) Reaction with liquid oxygen solvent

(a) 일반적인 solvent와 반응
화학 구조의 끝 부분을 중요시 여기는 기존의 화학 지식과 align

(b) 단순한 solvent와 반응 (ex. 액체 산소)
chromophore의 전체 부분이 반응한다고 여기는 기존의 화학 지식과 align



(c) Chromophore: EDAC

(c) 다양한 solvent와 반응 (Trans-ethyl p-(dimethylamino) cinnamate (EDAC))
solvent의 화학적 극성에 따라 같은 Chromophore에서도 중요한 부분을 다르게 탐지함

- Case1) Benzene solvent
Nitrogen-carbon 구조와 같은 극성이 낮은 구조가 중요함
→ Benzene같은 무극성 solvent와 반응하기 때문

- Case2) Ethanol, THF, 1-hexanol, 1-butanol solvent
Oxygen-carbon 구조와 같은 극성이 높은 구조가 중요함
→ Ethanol 과 THF 같은 극성 solvent와 반응하기 때문
→ 일반적으로 1-hexanol 과 1-butanol은 무극성 solvent로 분류되지만 -OH 구조로 인해 local 한 극성을 띄는 특성을 가짐

Outline

- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)
  - GNNs for Computer vision (Scene Understanding) (AAAI'23)

Problem definition

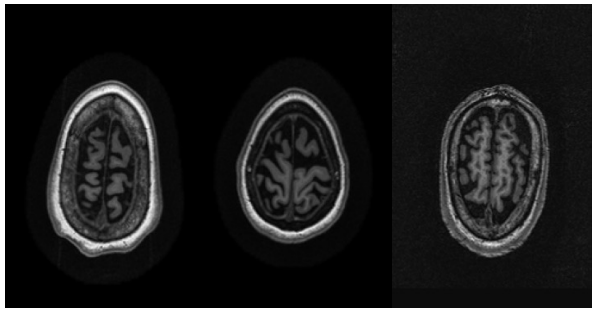


Image data

Patient	Age	DIG	TRA	ADA	MMS	City	APOE	RAV
1	42	34	-4	18.6	28	2	0	6
2	73	25	-15.8	31	20	9	1	4
3	80	38	-1.5	14.6	29	9	0	4
4	29	25	-9.4	21.3	27	4	1	4
5	65	34	-10.8	25.6	25	7	0	5

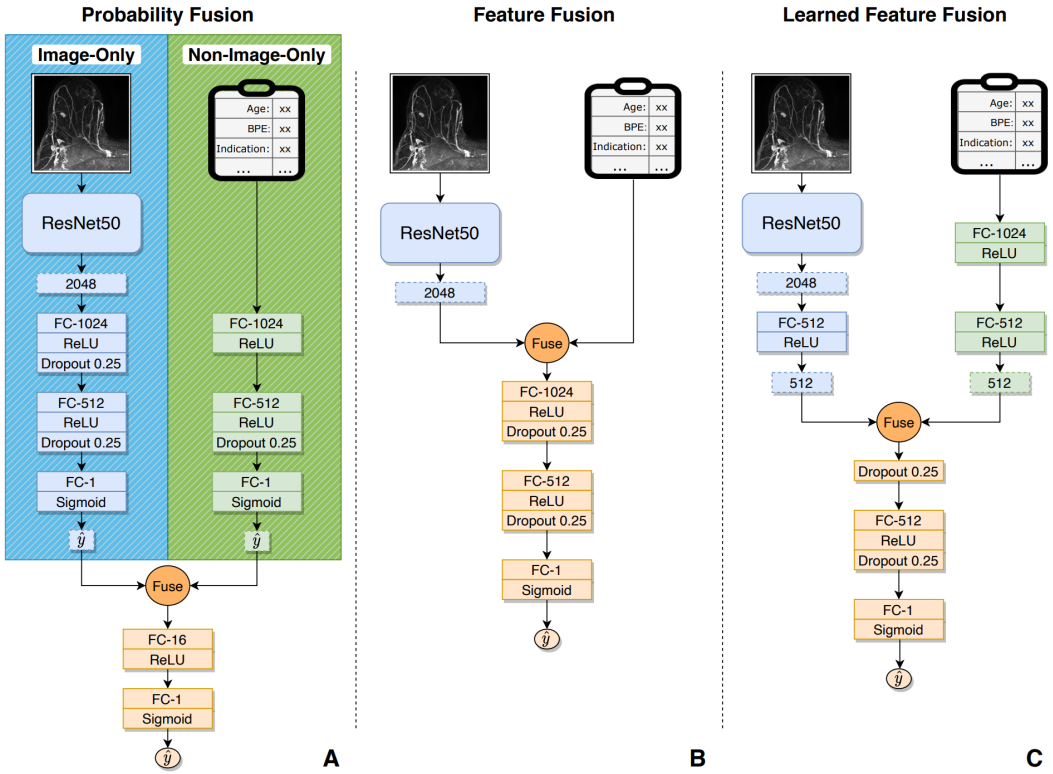
Non-image data

Fusing Method
(research goal)

Clinical Decision

- **Integrate** the multi-modal medical data (image and non-image data) for **more accurate clinical decisions**
- **Capture** important **information** from various aspects of the given data

Previous works

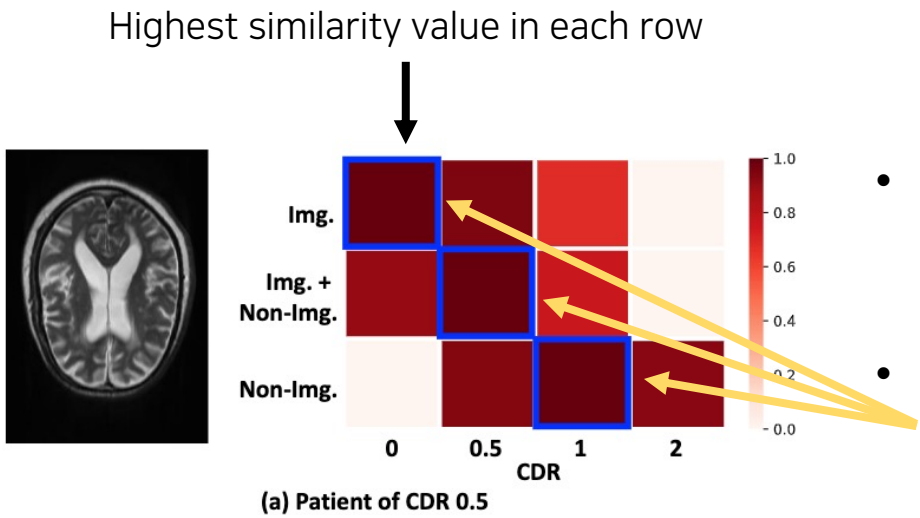


Naive multimodal fusion method

- Previous work demonstrates incorporating non-image data with images can significantly improve predictive performance
- But naïve integration of the modalities **cannot fully benefit from the complementary relationship** between the modalities.

G. Holste, S. C. Partridge, H. Rahbar, D. Biswas, C. I. Lee and A. M. Alessio, "End-to-End Learning of Fused Image and Non-Image Features for Improved Breast Cancer Classification from MRI," 2021 IEEE/CVF

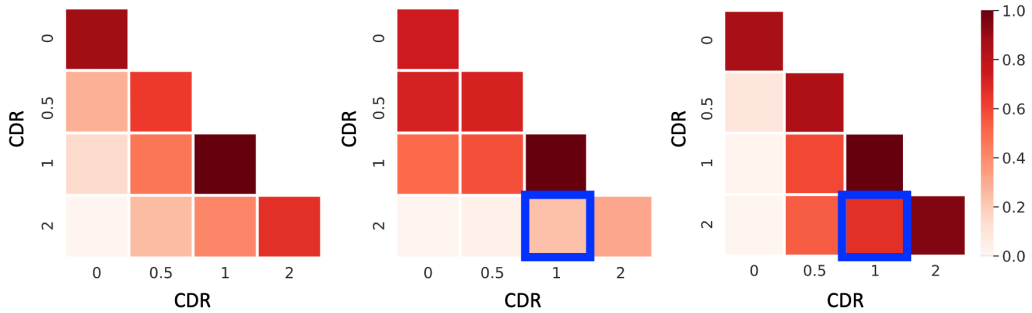
Motivation Medical data is multi-modal in nature



- Routine clinical visits of a patient produce not only image data, but also non-image data (i.e., clinical information).
- Multiple modalities of medical data provide **different and complementary views** of the same patient.

Integration of diverse and complementary views from medical data can make **more informed** and **accurate** clinical decisions.

Motivation Diverse aspects of Clinical data provides rich information on patients



All non-image Features

Non-image Features type 1 (Family history)

Non-image Features type 2 (Cognitive Abilities)

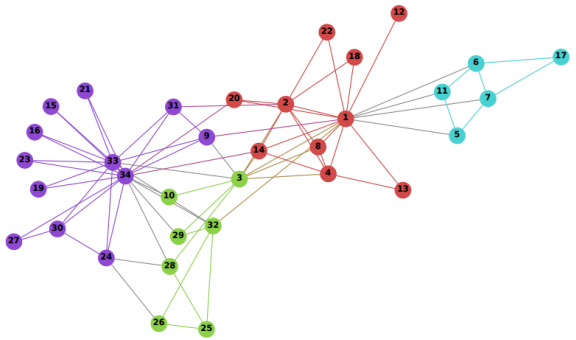
- Patients who suffer from the same disease share highly similar non-image data compared with those in different classes
- But diverse aspects of non-image data induce complex similarity relationships between patients

Example

- Type 1 feature fail to find connection between CDR 1 and 2
- Type 2 feature can make preemptive clinical decision on CDR 1

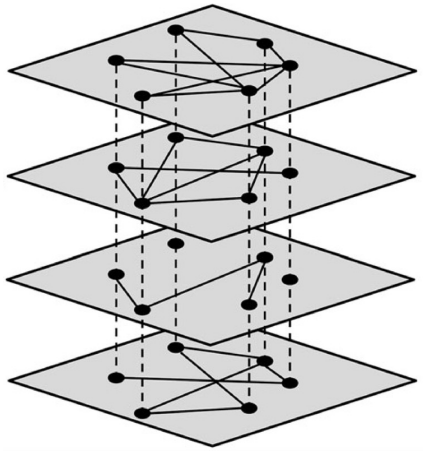
RQ1: How to incorporate complex similarity relationship between patients?

Methodology How can we capture the relationship between patients



Single Graph

- Single graph may miss the inherent complex relationships between patients
  - Single graph considers all relationships as the same
  - Whereas in reality, different relationships can have different characteristics and properties



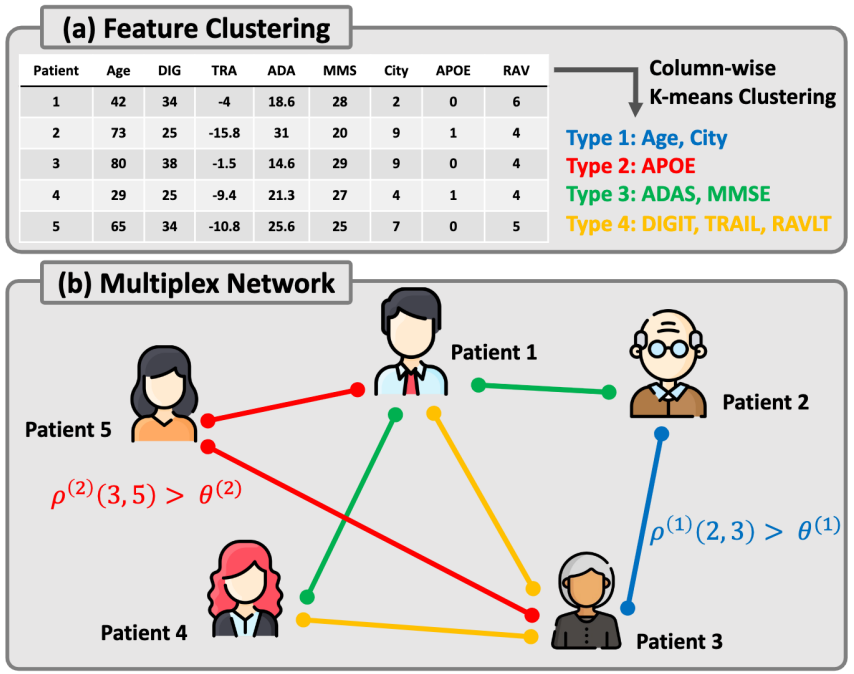
Multiplex Network

- Multiplex network considers the **multiple relationships** between nodes as different layers in the network
  - Each layer represents a specific type of relationship
  - Multiplex network can capture the complex relationships between patients, as it acknowledges the heterogeneity of relationships in the network

Goldblum, B.L., Reddie, A.W., Hickey, T.C. *et al.* The nuclear network: multiplex network analysis for interconnected systems. *Appl Netw Sci* **4**, 36 (2019).

Methodology

How can we capture the inherent complex relationship between patients ?



$$\boldsymbol{C} \in \mathbb{R}^{|\mathcal{V}| \times F_{non-img}} \xrightarrow{\text{Column-wise K-means}} \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, \cdots, \boldsymbol{c}^{(r)}, \cdots \boldsymbol{c}^{(K)}$$

$$\mathbf{A}^{(r)}(i,j) = \begin{cases} 1 & \text{if } \rho^{(r)}(i,j) > \theta^{(r)}, \\ 0 & \text{otherwise} \end{cases}$$

$$\rho^{(r)}(i,j) = \frac{\mathbf{c}_i^{(r)} \cdot \mathbf{c}_j^{(r)}}{\|\mathbf{c}_i^{(r)}\| \cdot \|\mathbf{c}_j^{(r)}\|}$$

- Using Column-wise K-means Clustering, divide non-image data into non-overlapping $|\mathcal{R}|$ types of features
- Using cosine-similarity and threshold for each type of non-image feature, construct multiplex network

Methodology Overall Frameworks

Overall Frameworks of HetMed

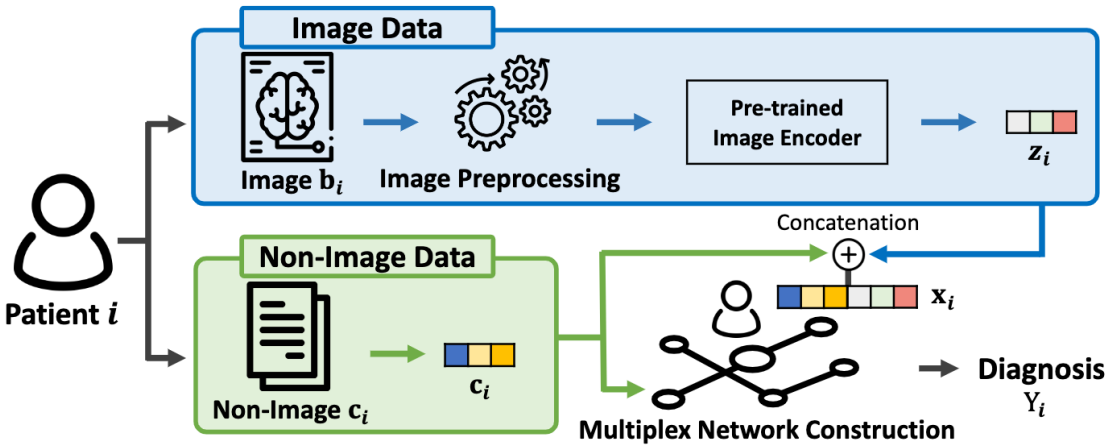


Image Data

- Image Preprocessing (atlas transform)
- Learn the image encoder (pre-trained with non-medical image)
- Extract embeddings of images

Non-image Data

- Column wise K-means Clustering
- Construct multiplex network via cosine similarity of each patient

Learn Multiplex Network

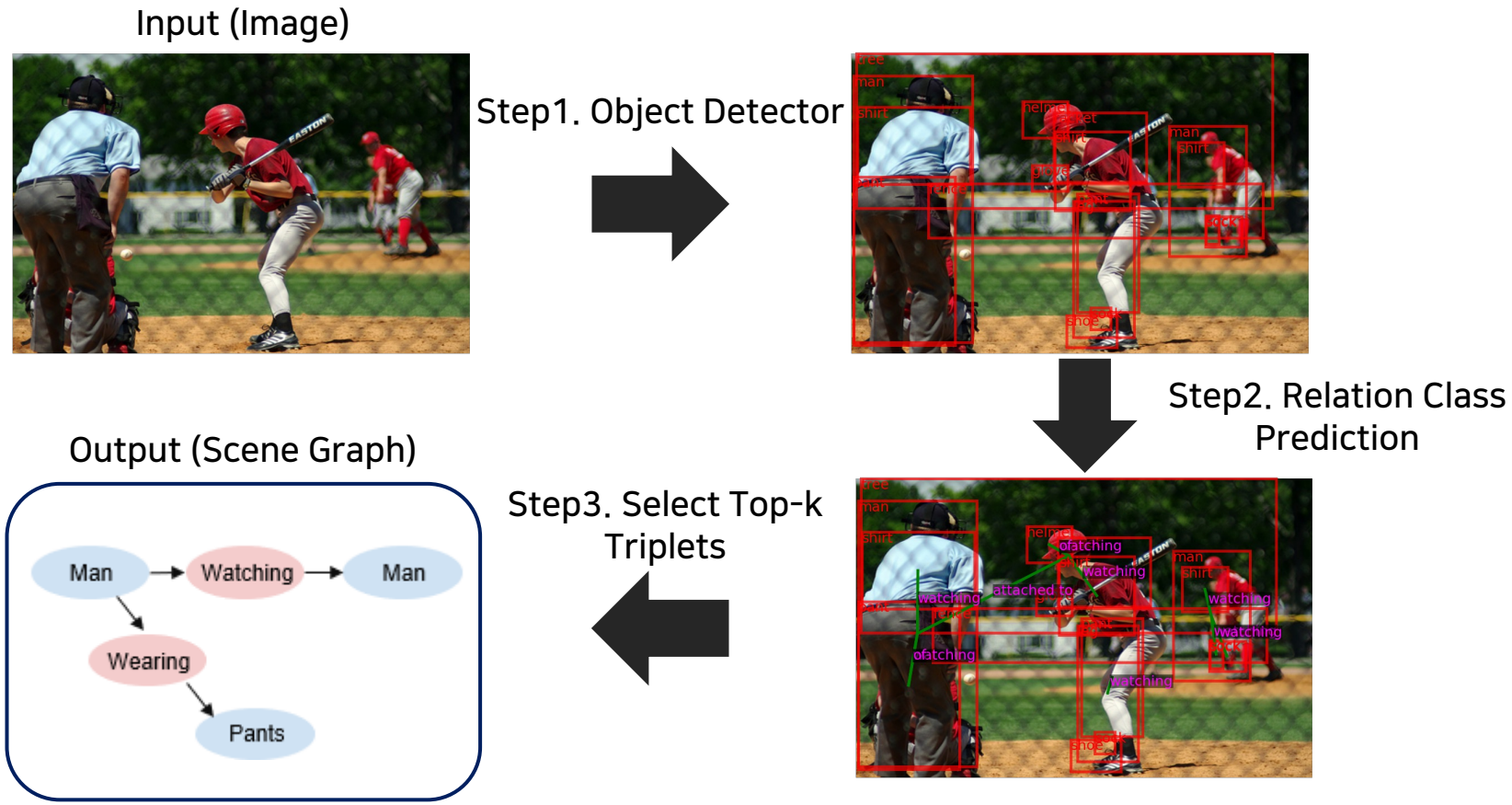
- Node feature = Concatenate (image embedding, non-image data)
- Learn consensus embedding
- Diagnosis

Outline

- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)
  - GNNs for Computer vision (Scene Understanding) (AAAI'23)

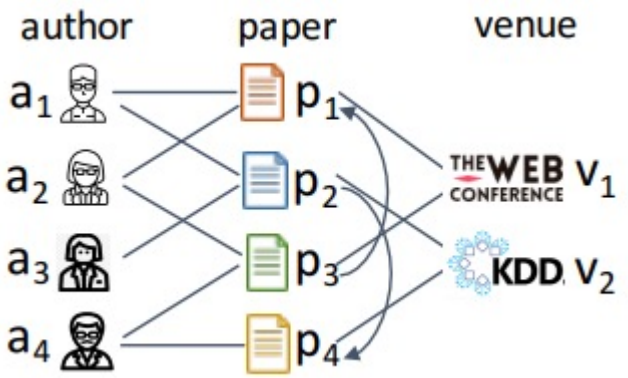
Scene graph generation (SGG)

- SGG aims to represent observable knowledges in an image in the form of a graph
  - The Knowledges include 1) object information and 2) their relation information
    - E.g., Object information: *man, horse, glasses, …* Relation information between objects: *feeding, wearing, …*

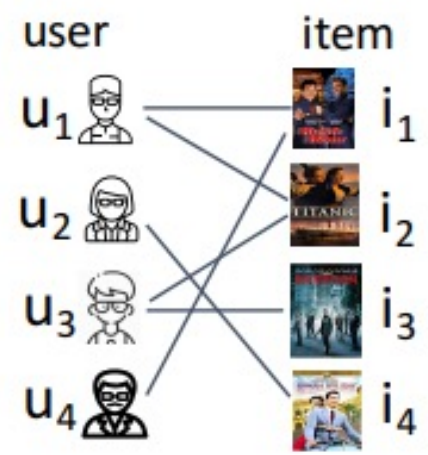


Heterogeneous graph

- **Heterogeneous graph** is a graph-structured data with more than one type of nodes or edges
  - By considering associations between multiple types of nodes or edges, many works demonstrate that considering the heterogeneity of nodes/edges are helpful for learning the representations with the semantic information.



[Academic Graph]

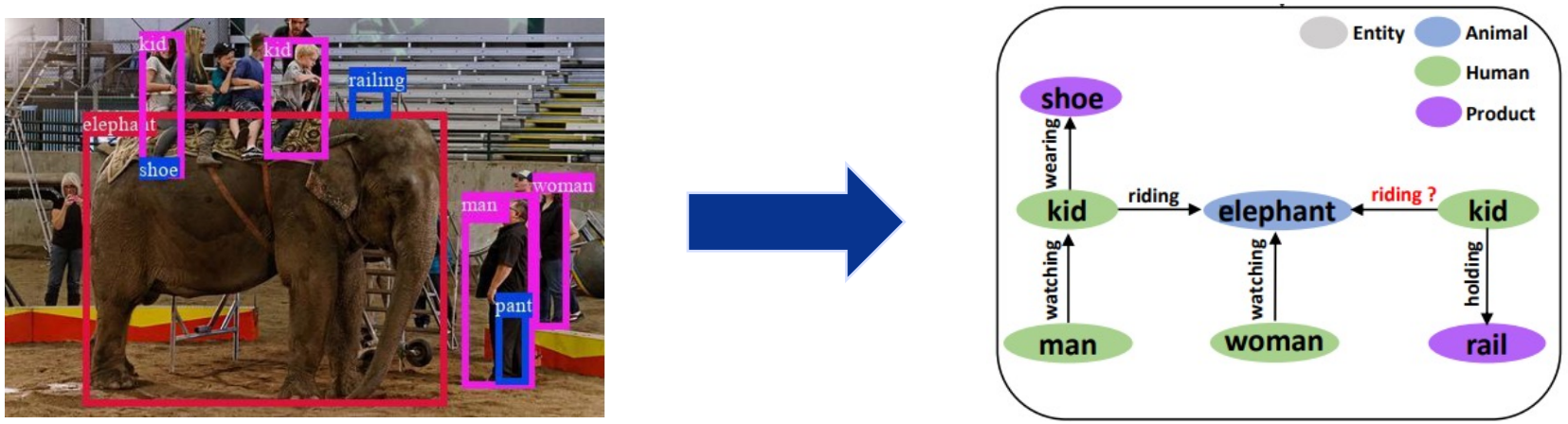


[Review Graph]

[KDD'19] Heterogeneous Graph Neural Network. Zhang et al.

Previous works

- In the literature of SGG, it's important to capture the context of neighborhood
  - Considering *<kid, holding, rail>* and *<woman, watching, elephant>* is helpful for predicting *<kid, riding, elephant>*
    - Compared with when **kid** and **elephant** are considered independently

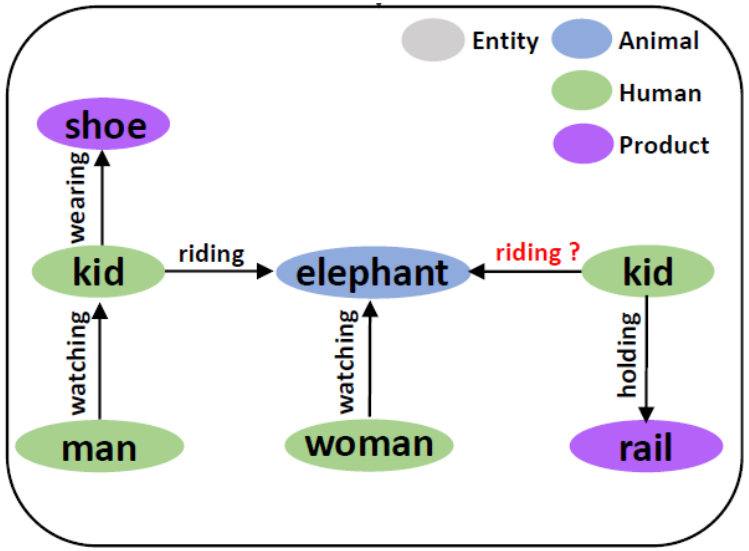


[Example of a context-aware model]

  - Context-aware SGG employs RNN, GNN, ⋯, Transformer to aggregate features of neighboring objects.

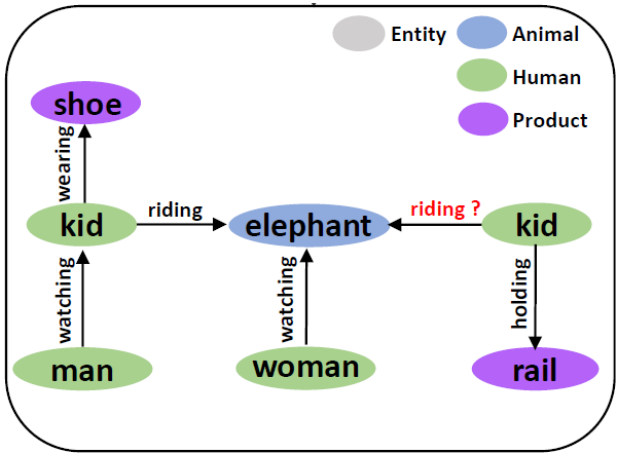
Limitations of previous works

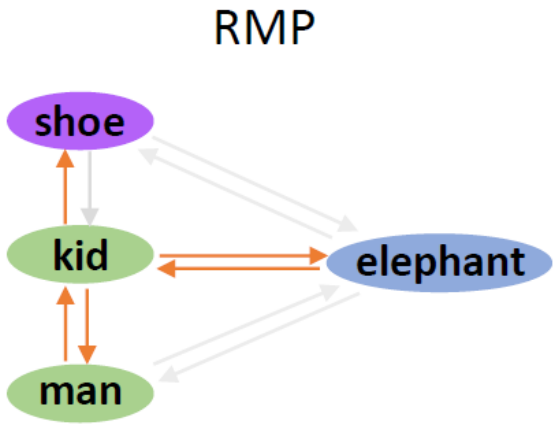
- Previous works consider the scene graph as homogeneous graph
  - The assumption of homogeneity restricts the context-awareness of the visual relations between objects.
    - Since it neglects the fact that predicates highly dependent on the objects where the predicates are associated.
    - For example, when we consider <*kid, riding, elephant*>, we know the opposite triplet <*elephant , riding, kid*> is not likely to appear.
    - Because it is usually "Human" that rides "Animal".



Our Goal

- We propose the Heterogeneous scene graph generation (HetSGG) framework
  - HetSGG generates a scene graph with relation-aware context
    - Consider both object types (e.g., Human, Animal, Product) & relation types (e.g., Human-Animal, Human-Human, ⋯,).
  - We propose a novel message-passing called relation aware message-passing (RMP)
  - Naturally captures the semantic between "Human" and "Animal" to predict <*kid, riding, elephant*>

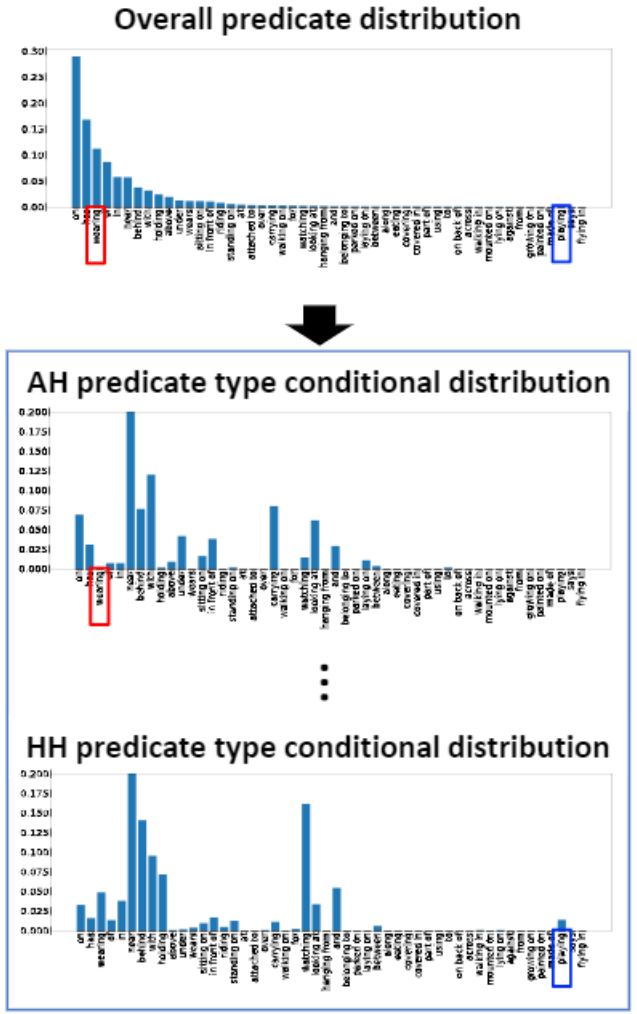




Relieving long-tailedness

- Overall predicate distribution is long-tailed
  - Problem: Model primarily predicts the meaningless predicate (i.e., on, has)

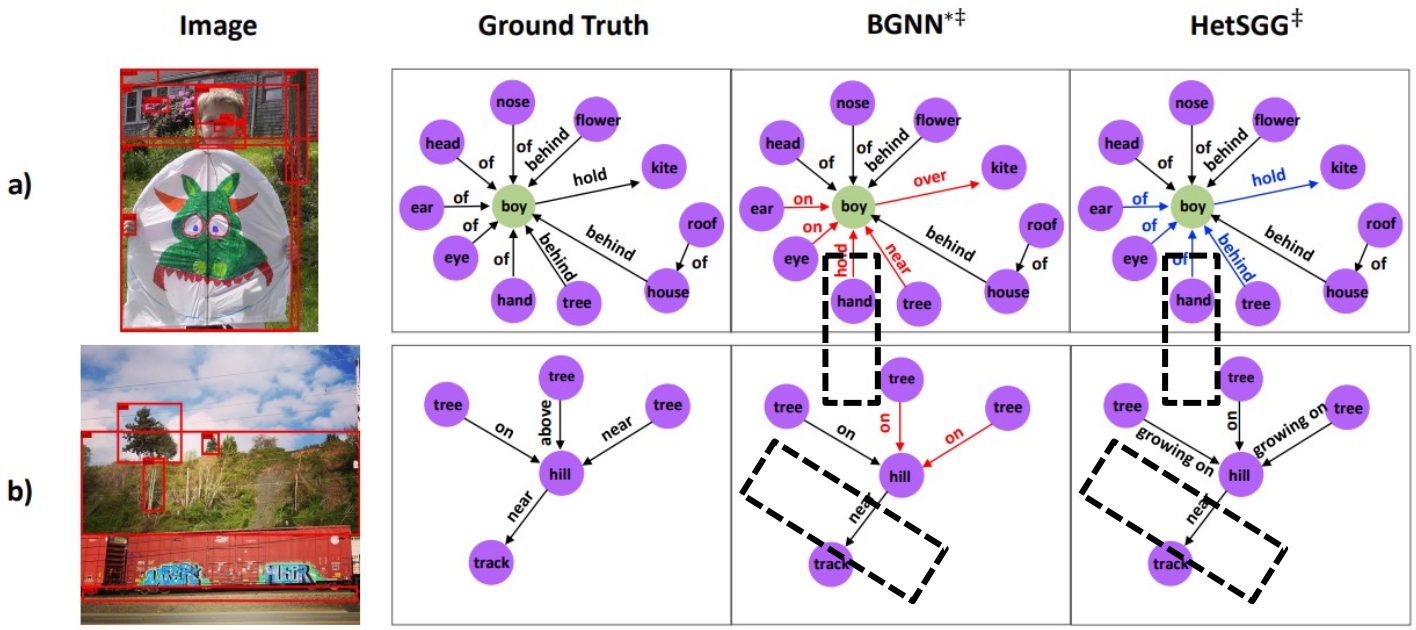
- Observation of the reformulated distribution in condition of predicate types
  - **Animal-Human**(AH): head predicate (e.g., "wearing") in overall distribution becomes tail predicate in AH distribution
  - **Human-Human**(HH): tail predicate (e.g., "playing") makes up a small proportion of the overall distribution, but the proportion improves in HH distribution



We expect the long-tailed problem is naturally alleviated in the formulation of heterogeneous graph distinguishing the relation type

Experiment: Qualitative results

- a) BGNN predicts “hand hold boy”, but HetSGG predicts “hand of boy”
  - HetSGG predicts the correct predicate by filtering the non-sense semantic relation, such as “hand hold boy”
- b) BGNN predicts “tree on hill”, but HetSGG predicts the fine-grained predicate (i.e., growing on)
  - HetSGG alleviates the long-tailed predicate distribution, thus predicts the fine-grained predicate



[Qualitative Result]

Red predicate: Incorrect for BGNN
Blue predicate: Correct for HetSGG and Incorrect for BGNN

Outline

- Overview
- Random walk-based Methods
- Graph Neural Networks (GNNs)
- How to effectively train GNNs?
  - Self-supervised Learning (AAAI'22)
  - Explainable Model (NeurIPS'23)
  - Robustness (KDD'23)
- Applications of GNNs
  - GNNs for Science (화학/소재/생물) (ICML'23)
  - GNNs for Medical Data (멀티모달 의료 데이터 분석) (AAAI'23)
  - GNNs for Computer vision (Scene Understanding) (AAAI'23)