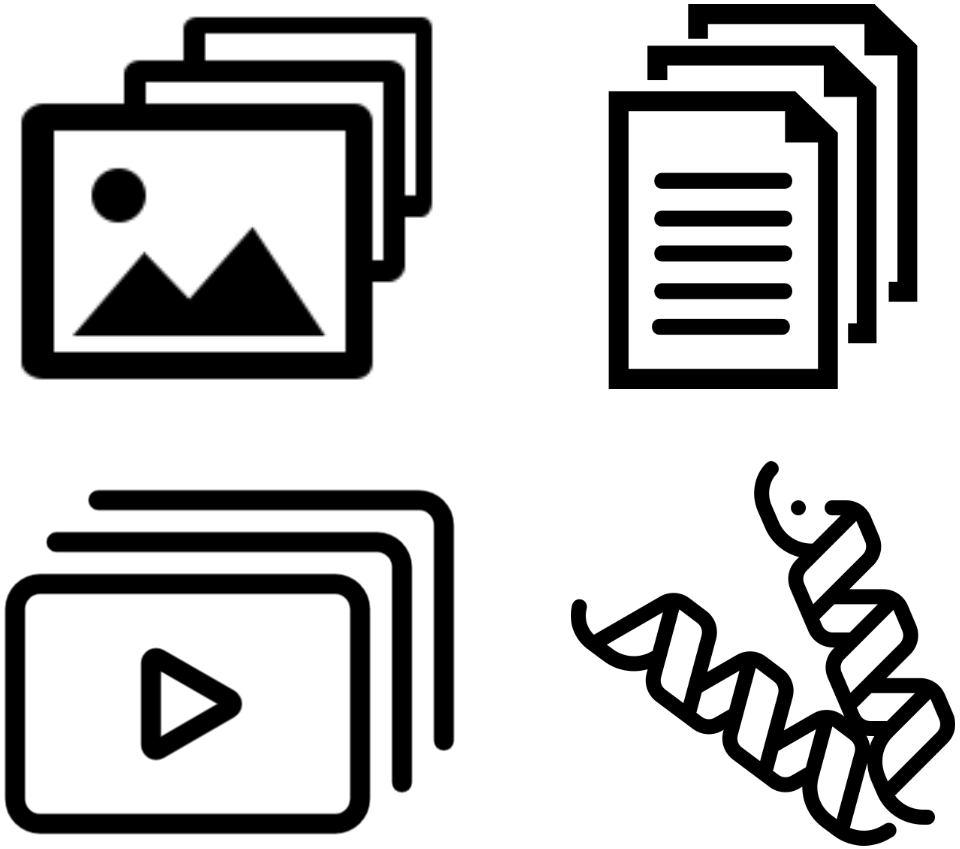
# Introduction to Graph Deep Learning

*Guest lecture for CS 7643 Deep Learning, Fall 2023*

**Jiaxuan You**

**Incoming Assistant Professor at UIUC CS**

**Gap**

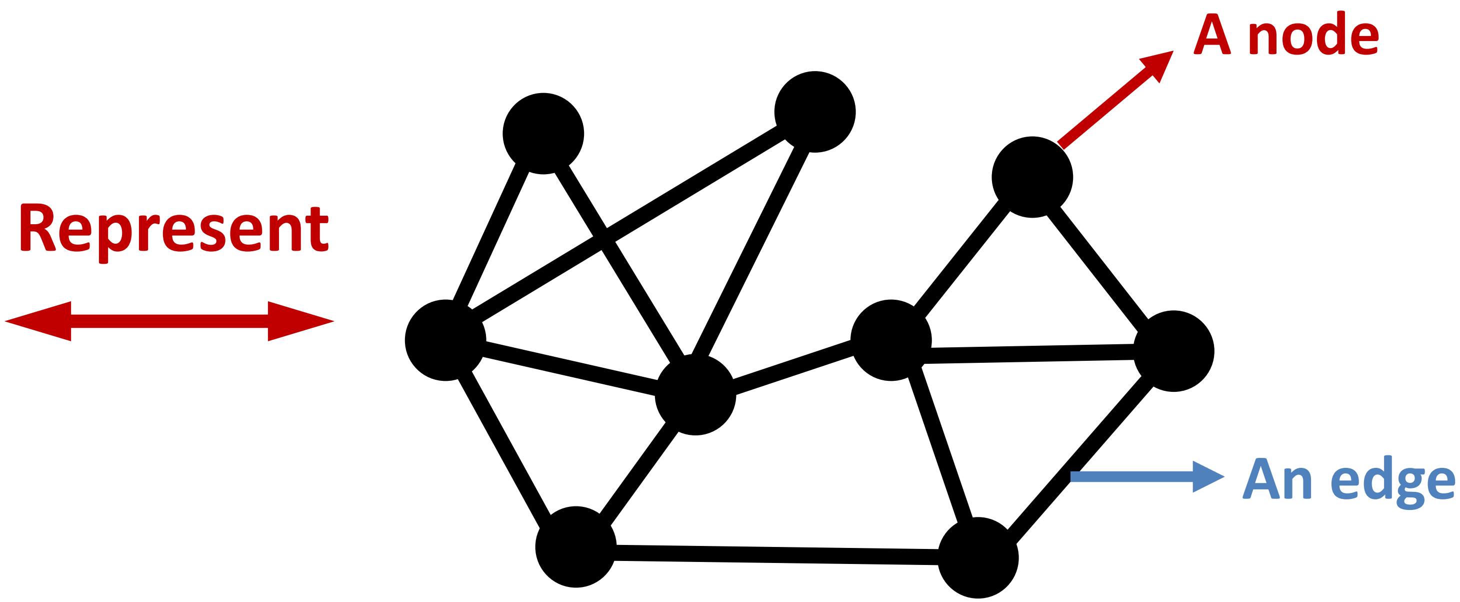


**Interconnected world**

**Modern ML**

# How to Represent **Interconnected Data?**

**Interconnected world**

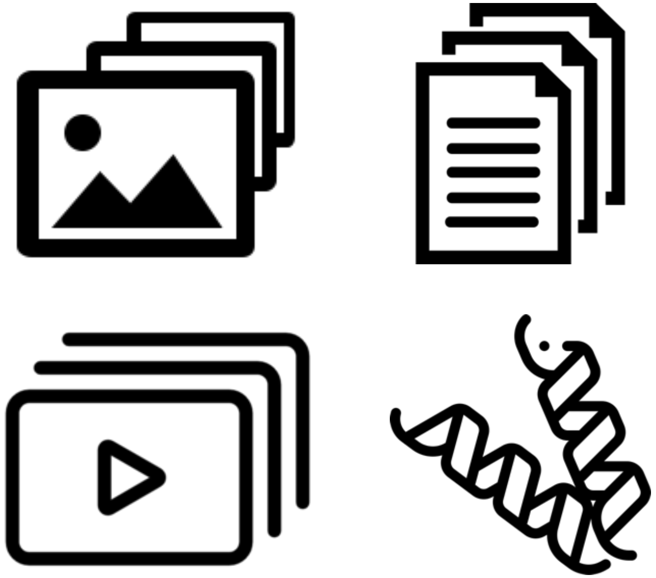



**Graph-structured data**

**Graph:** The language for **describing entities with relations**

Interconnected world

Gap



Modern ML

# Goal of Graph Deep Learning

## Enable DL research for the interconnected data

# Graph: Ubiquitous across Disciplines

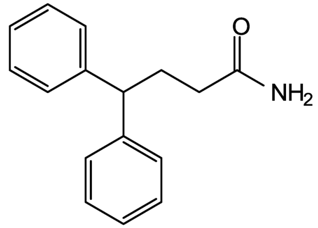



Image credit: MDPI

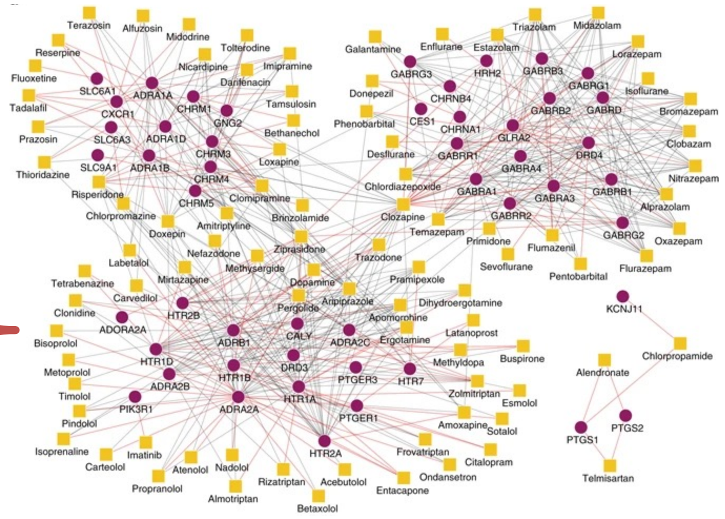


Image credit

Image credit: Medium

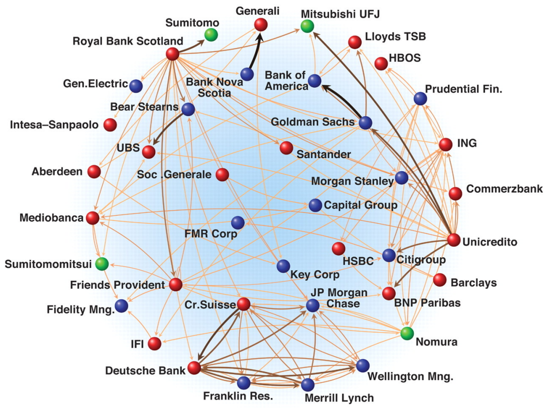



Image credit: Science

**Molecule**
*Molecule design*

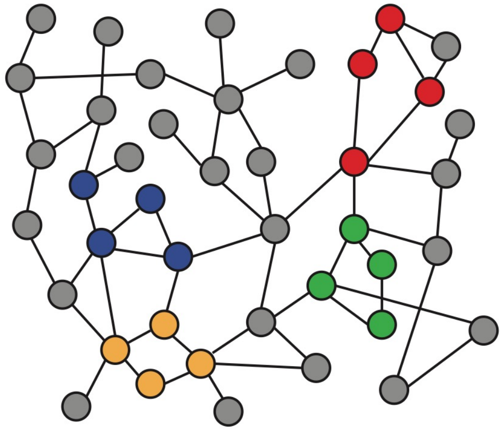
**Protein interaction**
*Drug discovery*

**Social network**
*Recommender systems*

**Economic network**
*Policy making*

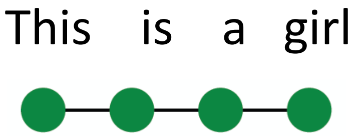
- **Graphs:** ***flexible*** and ***expressive***
- **Graphs** can **bridge interdisciplinary data**

# Machine Learning with Graphs is Hard

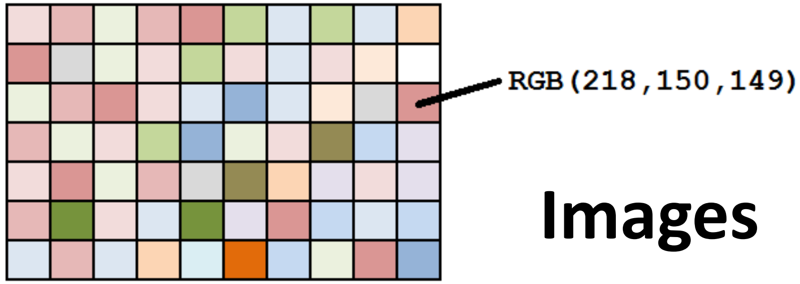


Graphs

***vs.***



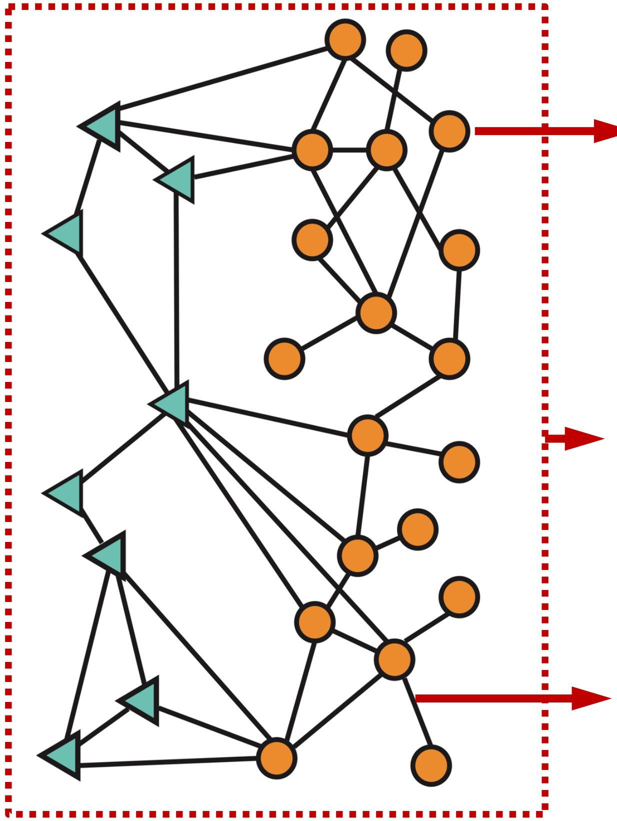

Text




Images

- **Arbitrary size** and topological **structure**
- Nodes have **no fixed ordering**

# Graph Machine Learning **Tasks**

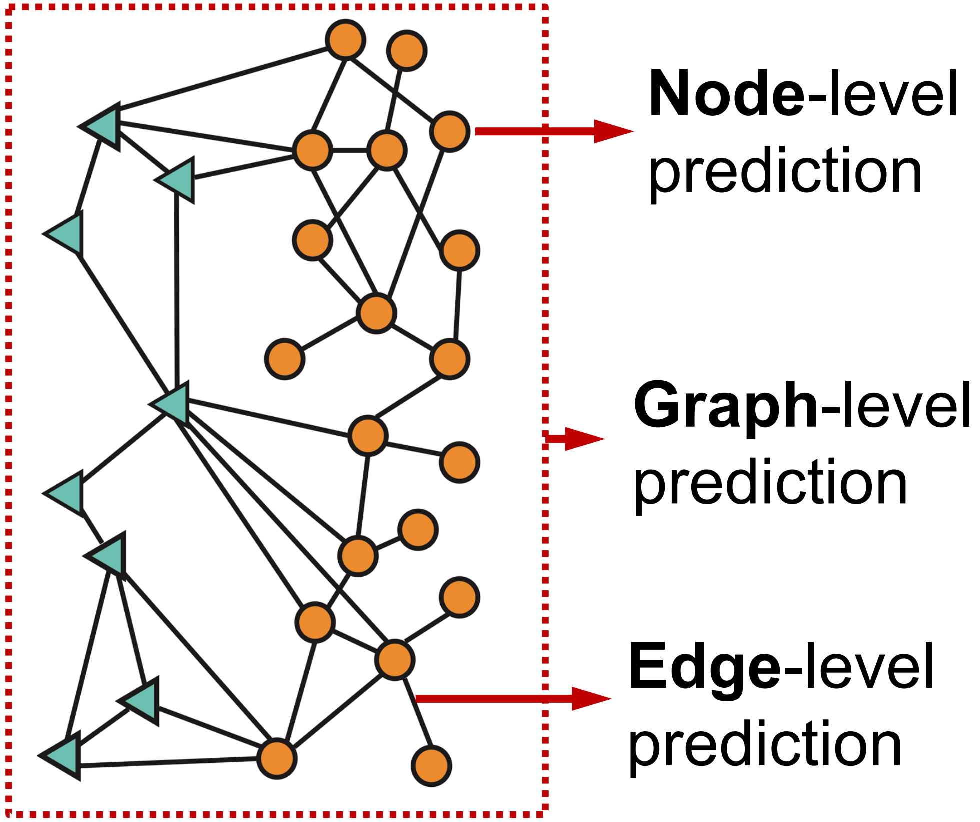


**Node**-level prediction
*"Classify user by their type in a social network"*

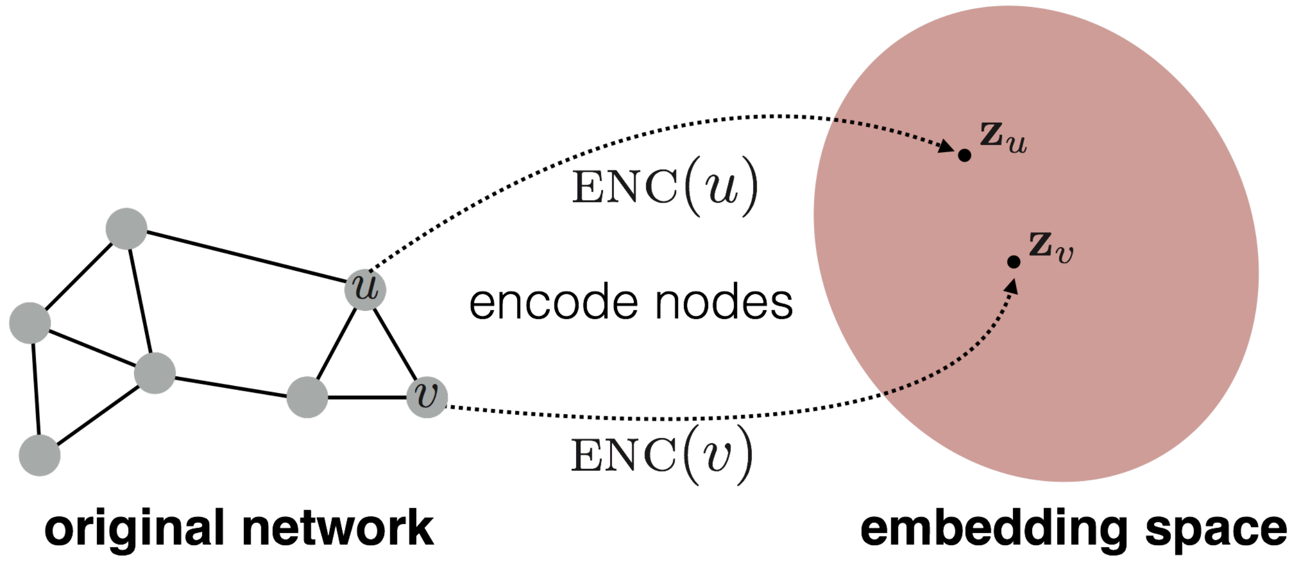
**Graph**-level prediction
*"Predict which molecules are drug-like"*

**Edge**-level prediction
*"Recommend item nodes to user nodes"*

## Graph ML **Tasks**

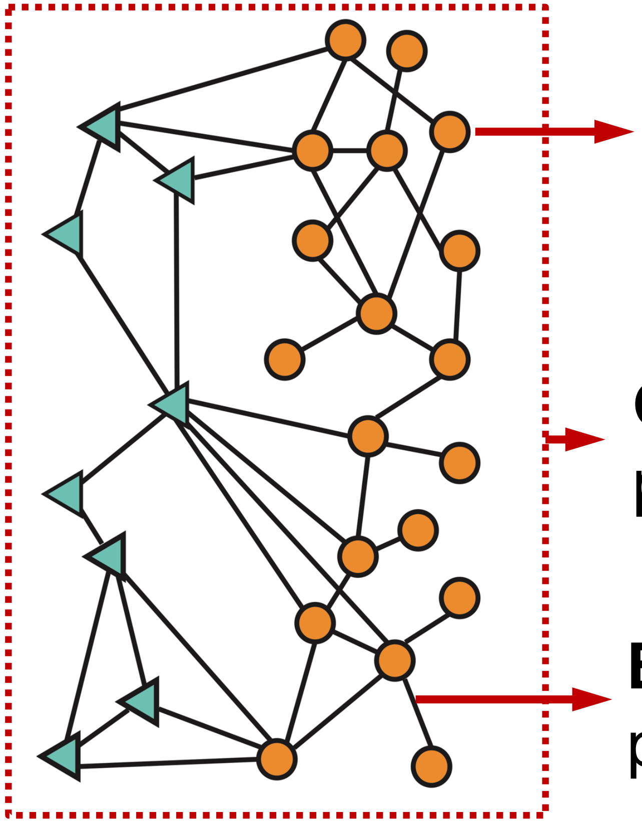



## Key Idea: **Node Embeddings**




**Intuition:** Map nodes to $d$-dimensional embeddings such that similar nodes in the graph are embedded close together

## Graph ML **Tasks**

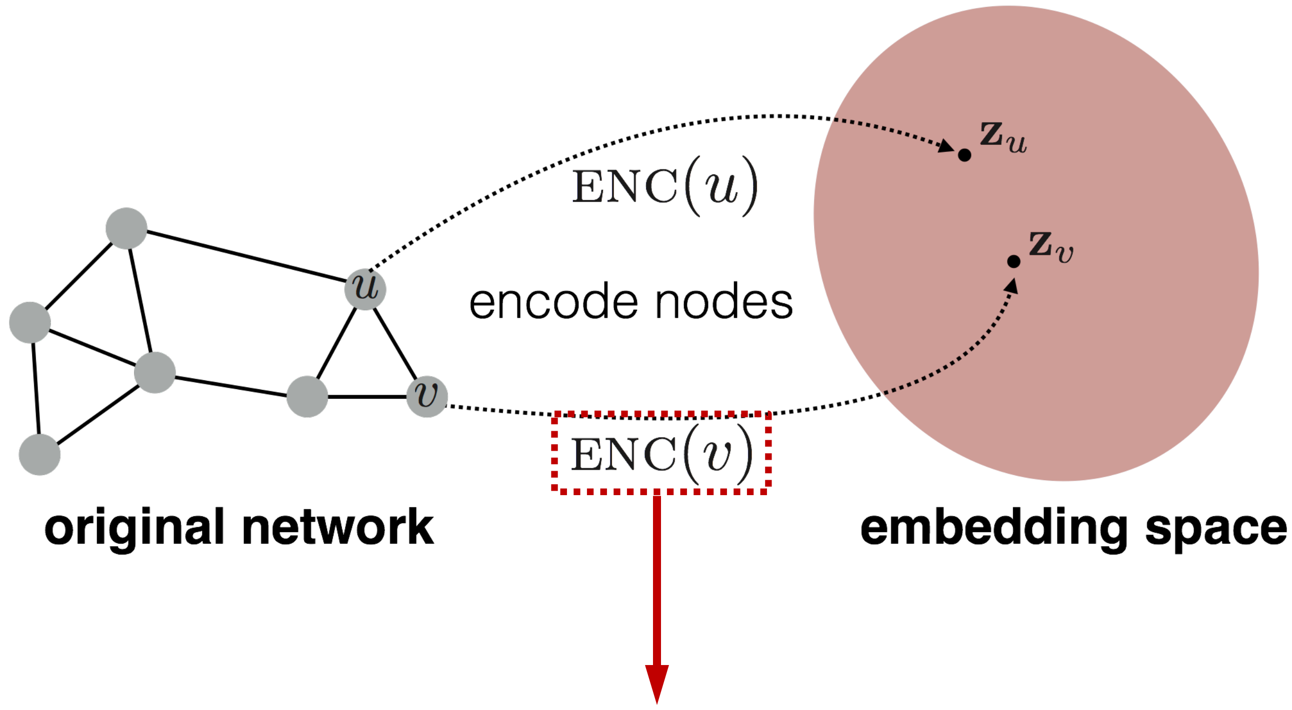


**Node**-level prediction

**Graph**-level prediction

**Edge**-level prediction

## Key Idea: **Node Embeddings**

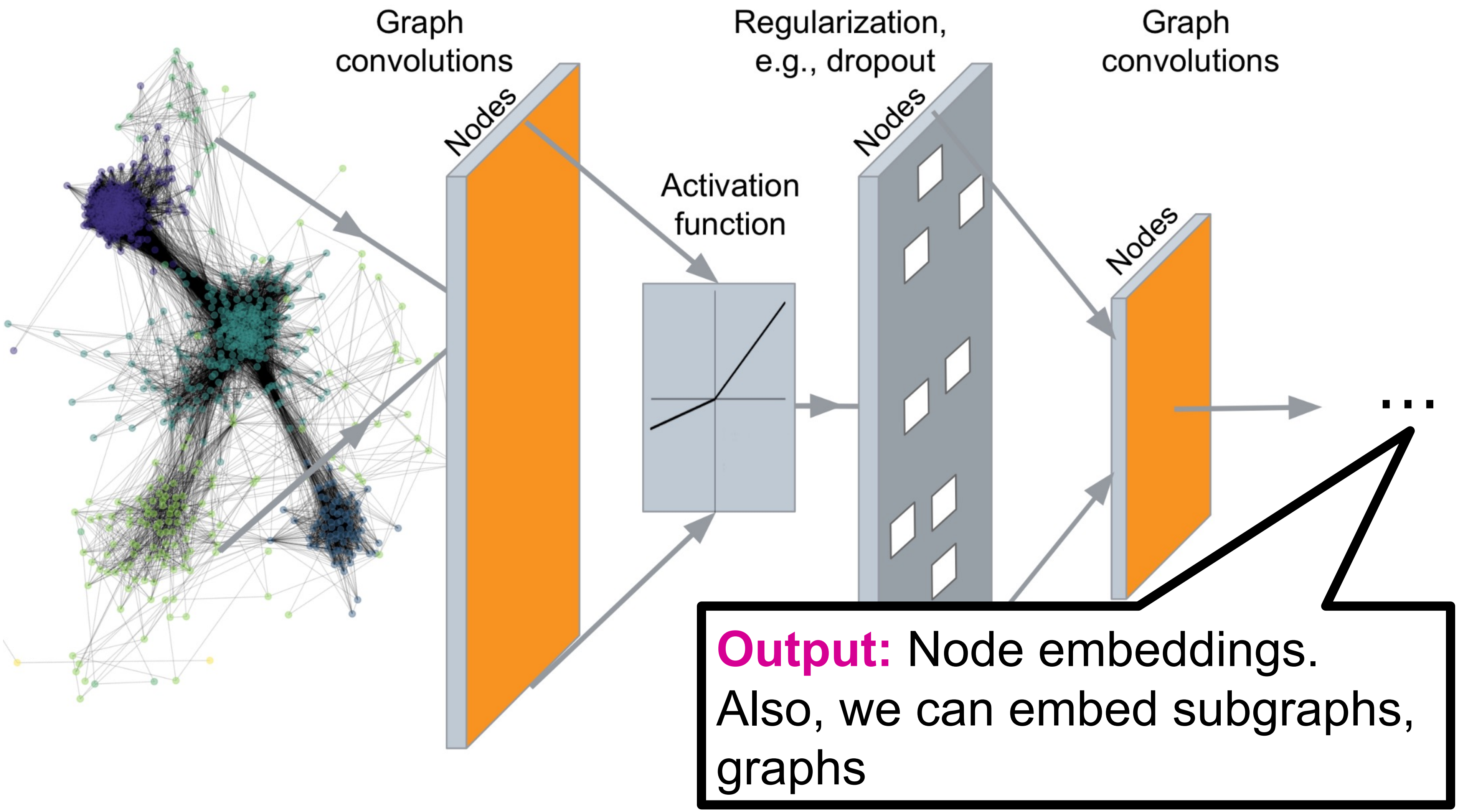



**Graph Neural Networks (GNNs)**

Slides adapted from Stanford CS224W Course

# Graph Neural Networks (GNNs)

# Deep Graph Encoders

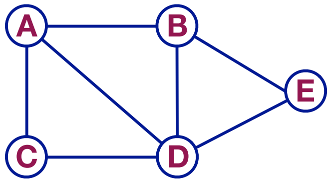


# Graph ML Setup

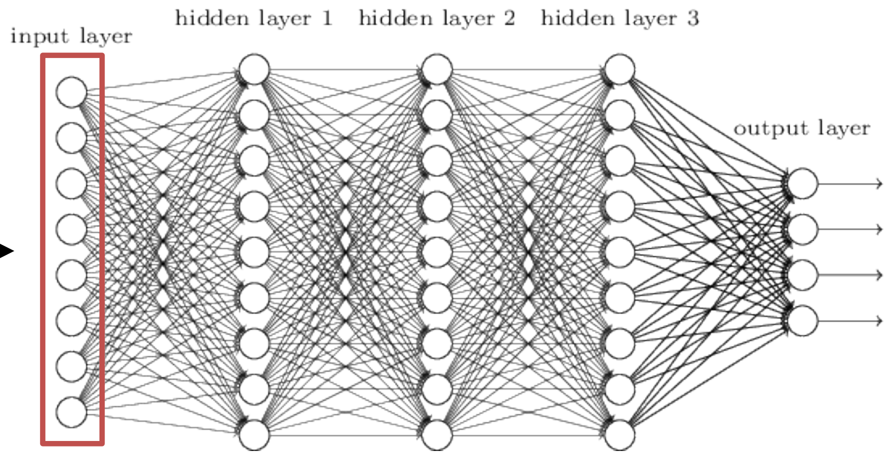
- **Assume we have a graph $\boldsymbol{G}$:**
  - $V$ is the **vertex set**
  - $\boldsymbol{A}$ is the **adjacency matrix** (assume binary)
  - $\boldsymbol{X} \in \mathbb{R}^{m \times |V|}$ is a matrix of **node features**
    - Social networks – user attributes, molecule – atom types, …
    - When there is no node feature in the graph dataset:
      - One-hot encodings – cannot generalize to new nodes
      - Vector of constant 1: [1, 1, …, 1] – inductive, but less expressive
    - **Edge feature** can be incorporated as well
  - $v$: a node in $V$; $N(v)$: the set of neighbors of $v$.
  - **Node features:**

# A Naïve Approach: MLP

- Join adjacency matrix and features
- Feed them into a deep neural net:




$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array}
\begin{array}{c}
\begin{array}{ccccccc} A & B & C & D & E & \text{Feat} & \end{array} \\
\left(\begin{array}{ccccccc}
0 & 1 & 1 & 1 & 0 & 1 & 0 \\
\boxed{1 \quad 0 \quad 0 \quad 1 \quad 1} & & & & & \boxed{0 \quad 0} & \\
1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0
\end{array}\right)
\end{array}$$

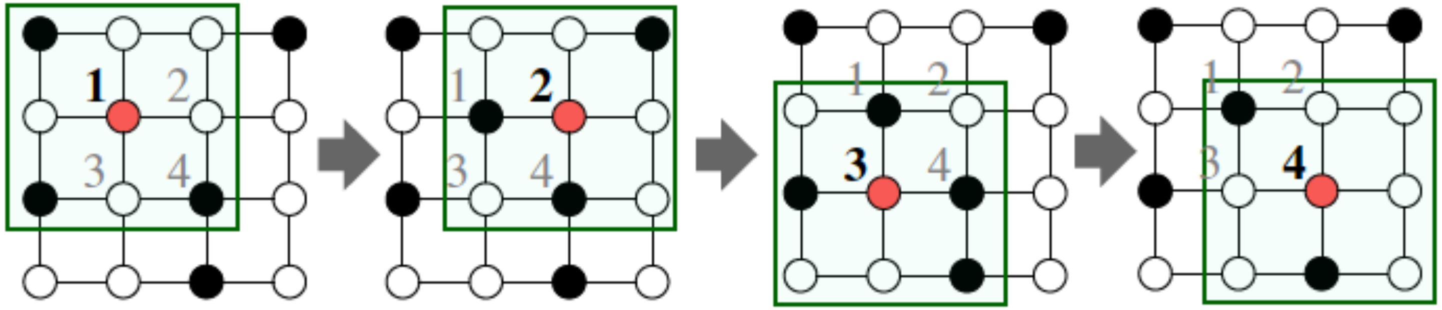


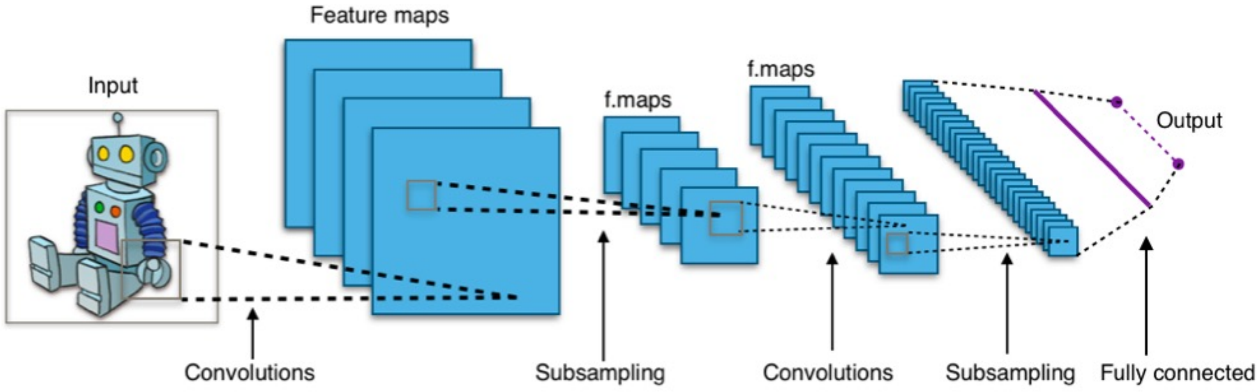

**?**

- Issues with this idea:
  - $O(|V|)$ parameters
  - Not applicable to graphs of different sizes
  - Sensitive to node ordering

# Idea: Convolutional Networks

## CNN on an image:

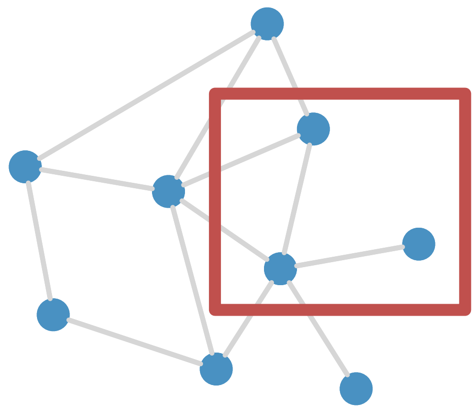


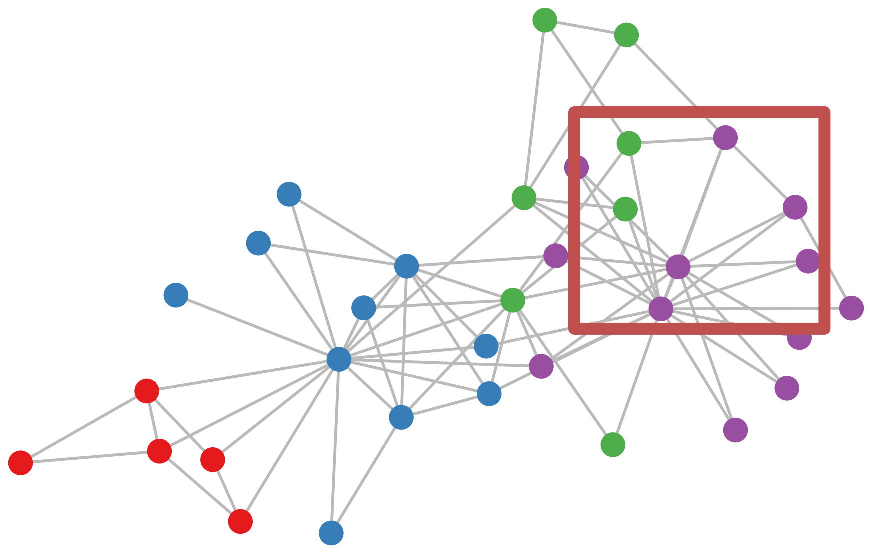

Goal is to generalize convolutions beyond simple lattices
Leverage node features/attributes (e.g., text, images)

# Real-World Graphs

**But our graphs look like this:**



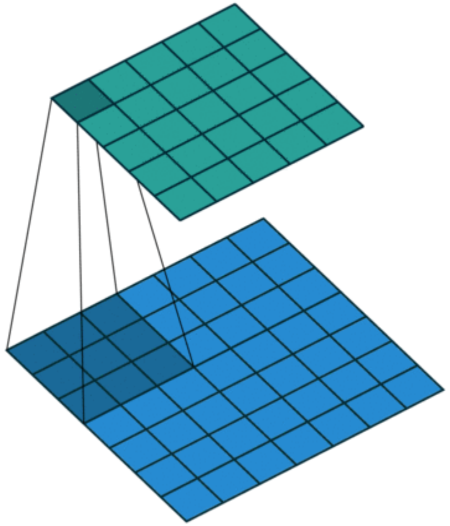
or this:



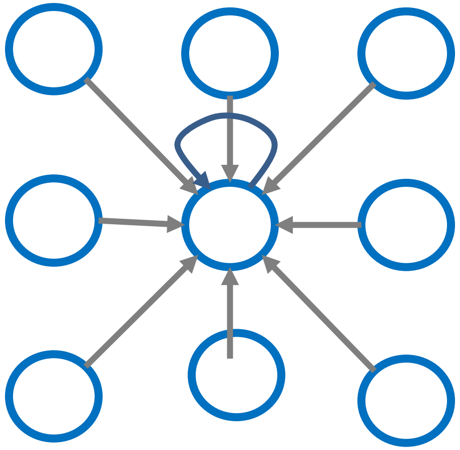
- There is no fixed notion of locality or sliding window on the graph
- Graph is permutation invariant

# From Images to Graphs

Single Convolutional neural network (CNN) layer with 3x3 filter:



Image



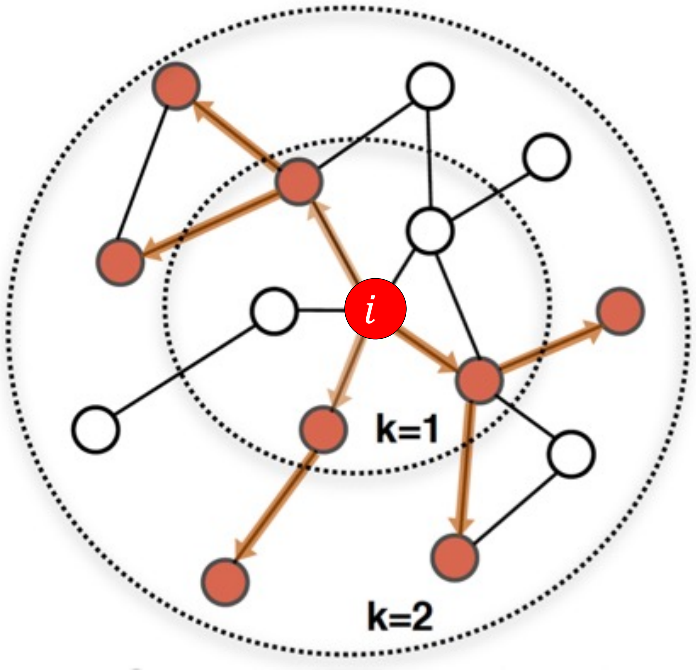
Graph

**Idea:** transform information at the neighbors and combine it:

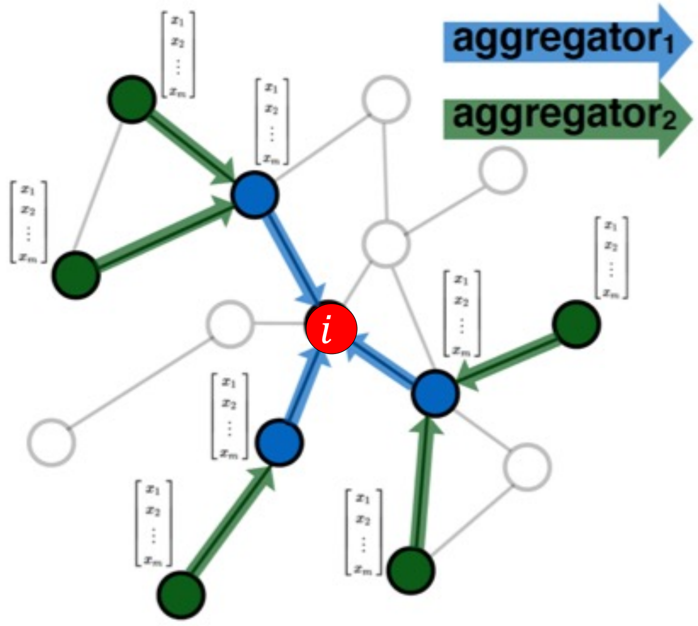
- Transform "messages" $h_i$ from neighbors: $W_i\ h_i$
- Add them up: $\sum_i W_i\ h_i$

# Graph Convolutional Networks

- Graph Convolutional Networks: one of the first GNN models



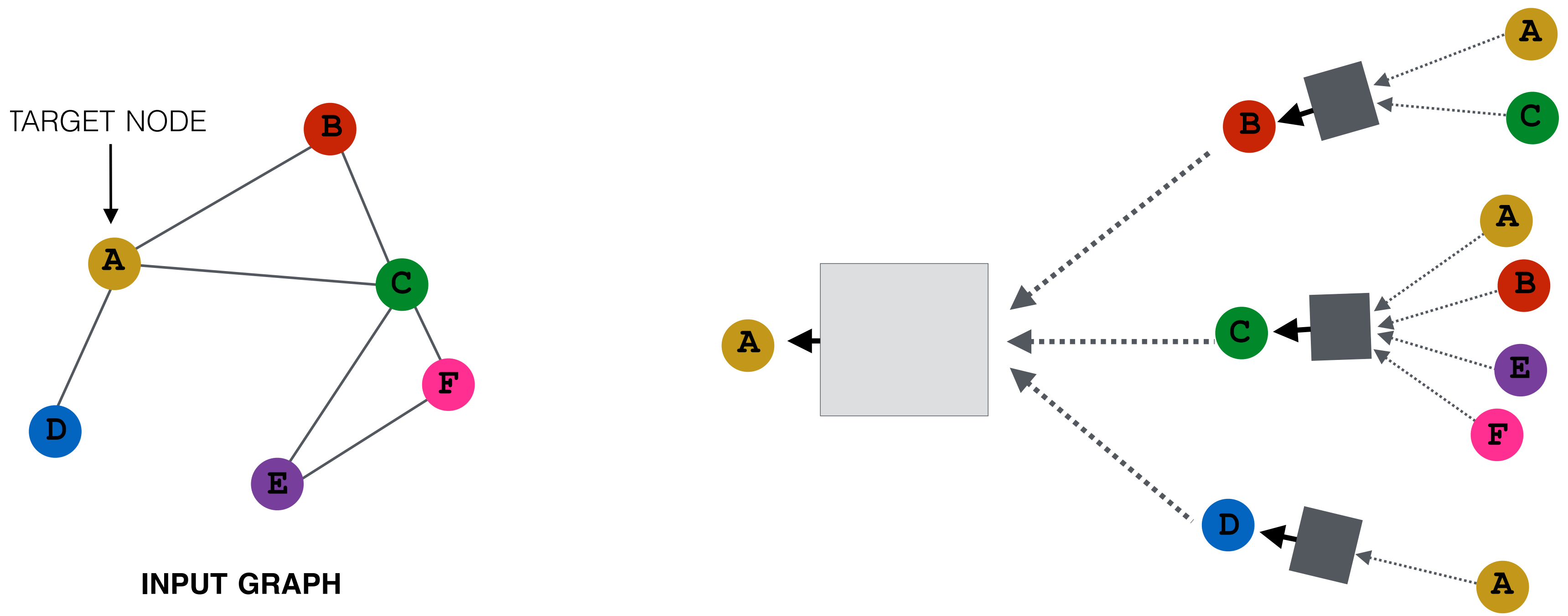

Determine node computation graph




Propagate and transform information

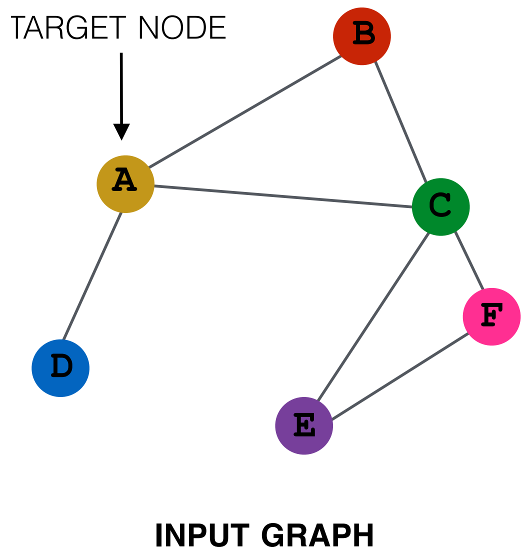
# Idea: Aggregate Neighbors

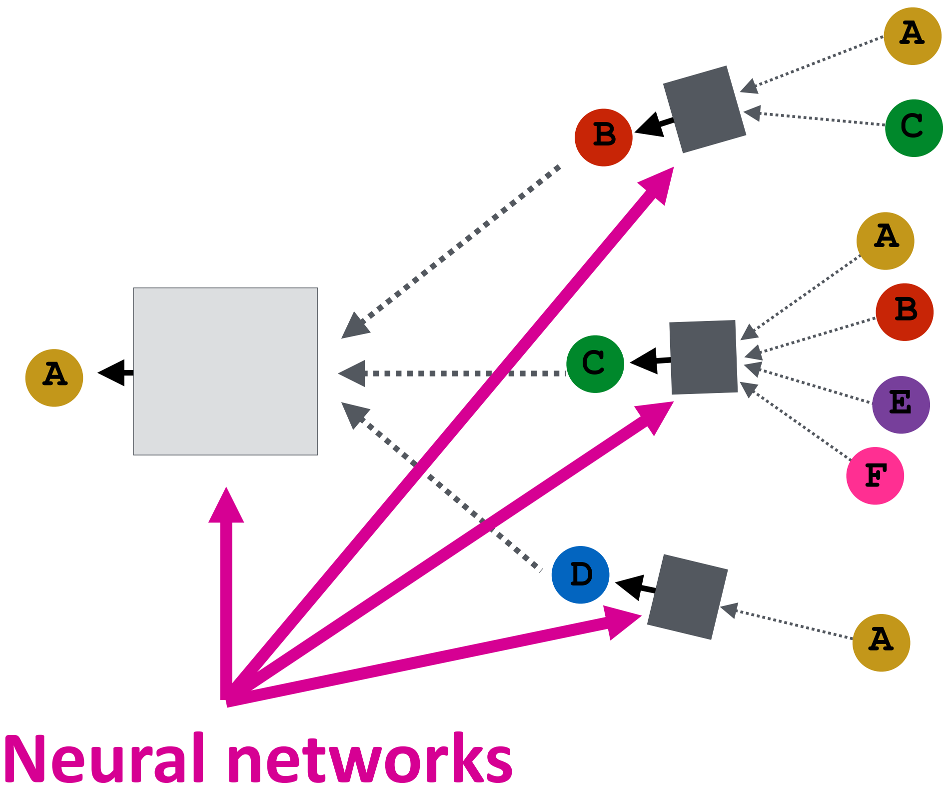
- **Key idea:** Generate node embeddings based on **local network neighborhoods**



# Idea: Aggregate Neighbors

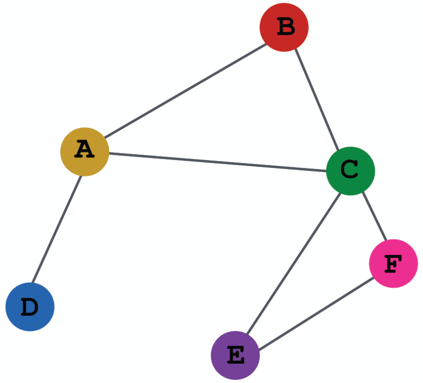
- **Intuition:** Nodes aggregate information from their neighbors using neural networks



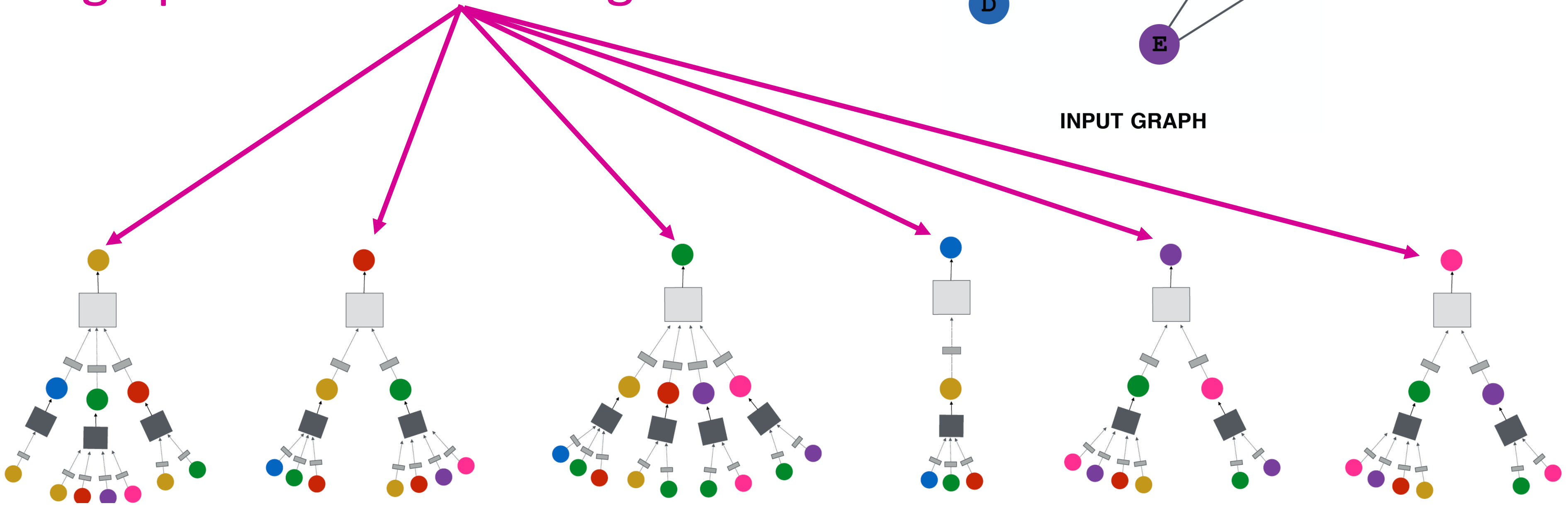
# Idea: Aggregate Neighbors

- **Intuition:** Network neighborhood defines a computation graph



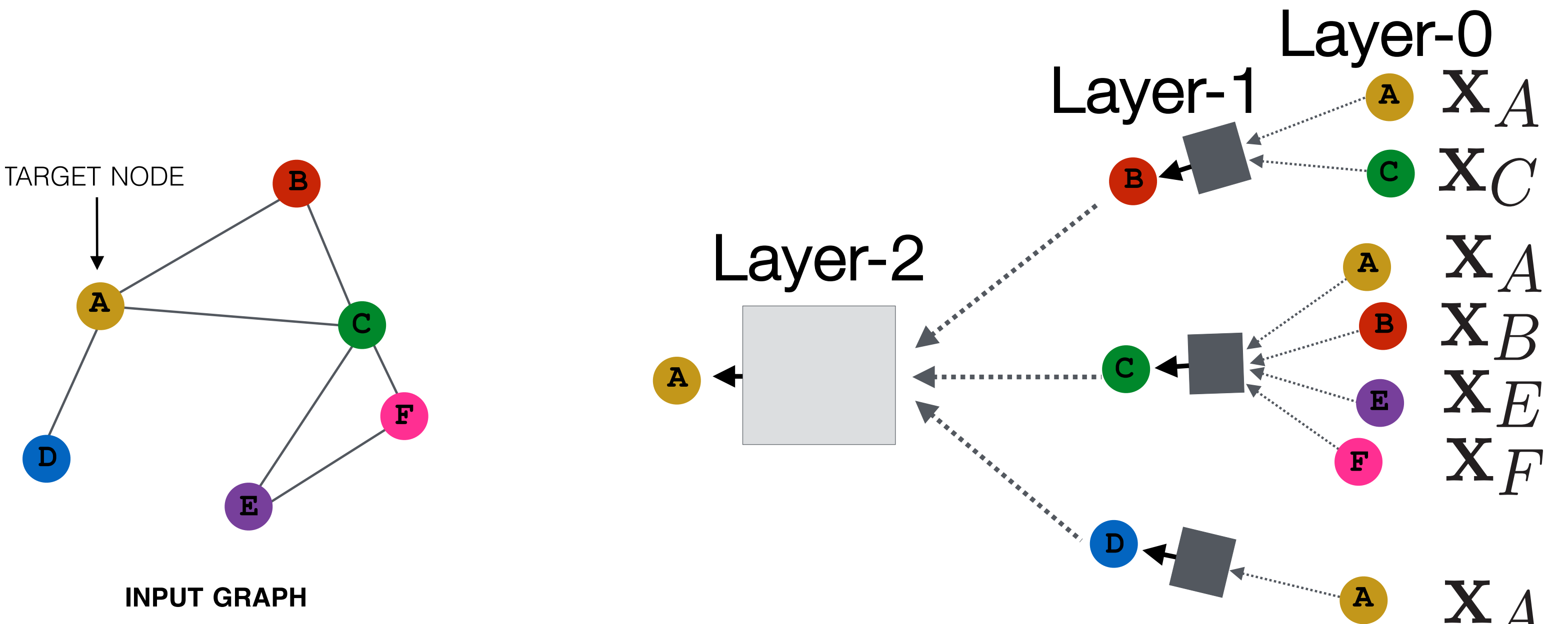

INPUT GRAPH

Every node defines a computation graph based on its neighborhood!



# Deep Model: Many Layers

- Model can be of arbitrary depth:
  - Nodes have embeddings at each layer
  - Layer-0 embedding of node $u$ is its input feature, $x_u$
  - Layer-$k$ embedding gets information from nodes that are K hops away



# The Math: GCN with Many Layers

- **Basic approach:** Average neighbor messages and apply a neural network

Initial 0-th layer embeddings are equal to node features

$$\boxed{h_v^0 = x_v}$$

embedding of $v$ at layer $l$

$$h_v^{(l+1)} = \boxed{\sigma}\left(W_l \boxed{\sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|}} + B_l \boxed{h_v^{(l)}}\right), \forall l \in \{0, \ldots, \boxed{L} - 1\}$$

$$\boxed{z_v} = h_v^{(L)}$$

Average of neighbor's previous layer embeddings

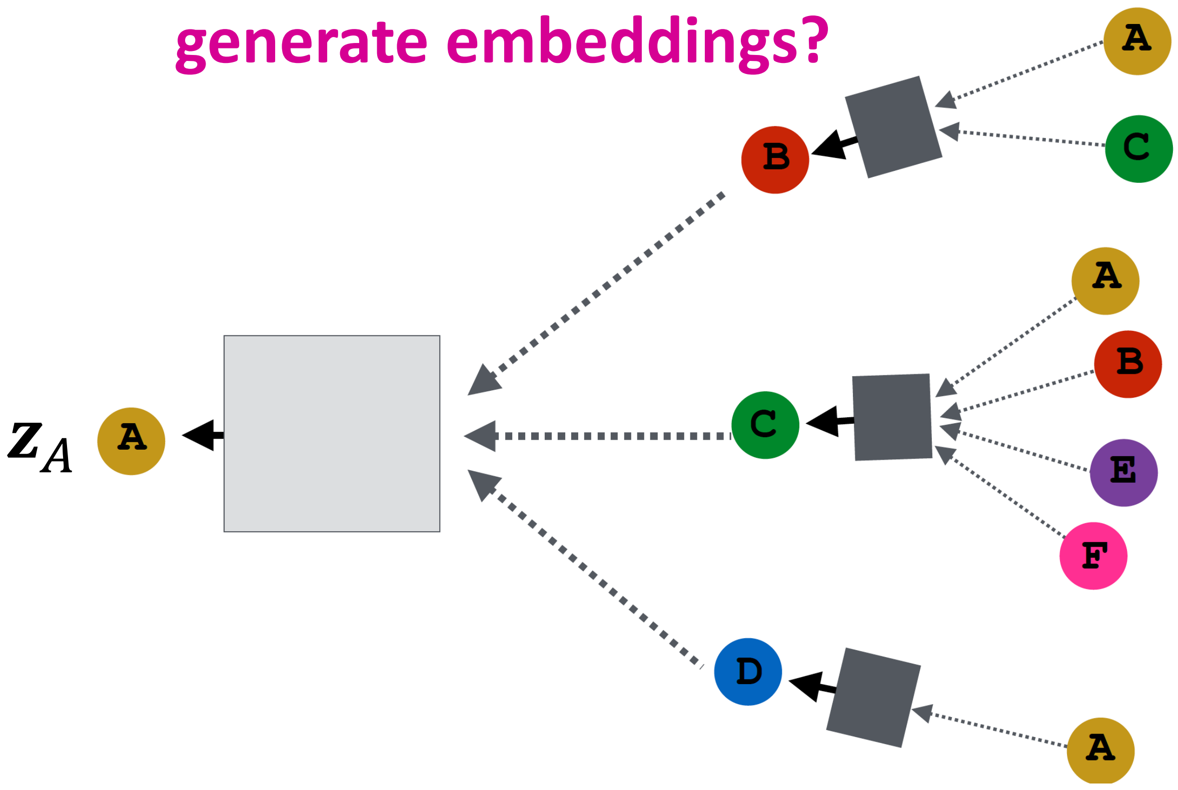
Total number of layers

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

# Training the GNN Model

**How do we train the model to generate embeddings?**




**Need to define a loss function on the embeddings**

# Model Parameters

Trainable weight matrices
(i.e., what we learn)

$$\mathrm{h}_v^{(0)} = \mathrm{x}_v$$

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

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

**Final node embedding**

We can feed these **embeddings into any loss function** and run SGD to **train the weight parameters**

$h_v^l$: the hidden representation of node $v$ at layer $l$

- $W_k$: weight matrix for neighborhood aggregation
- $B_k$: weight matrix for transforming hidden vector of self

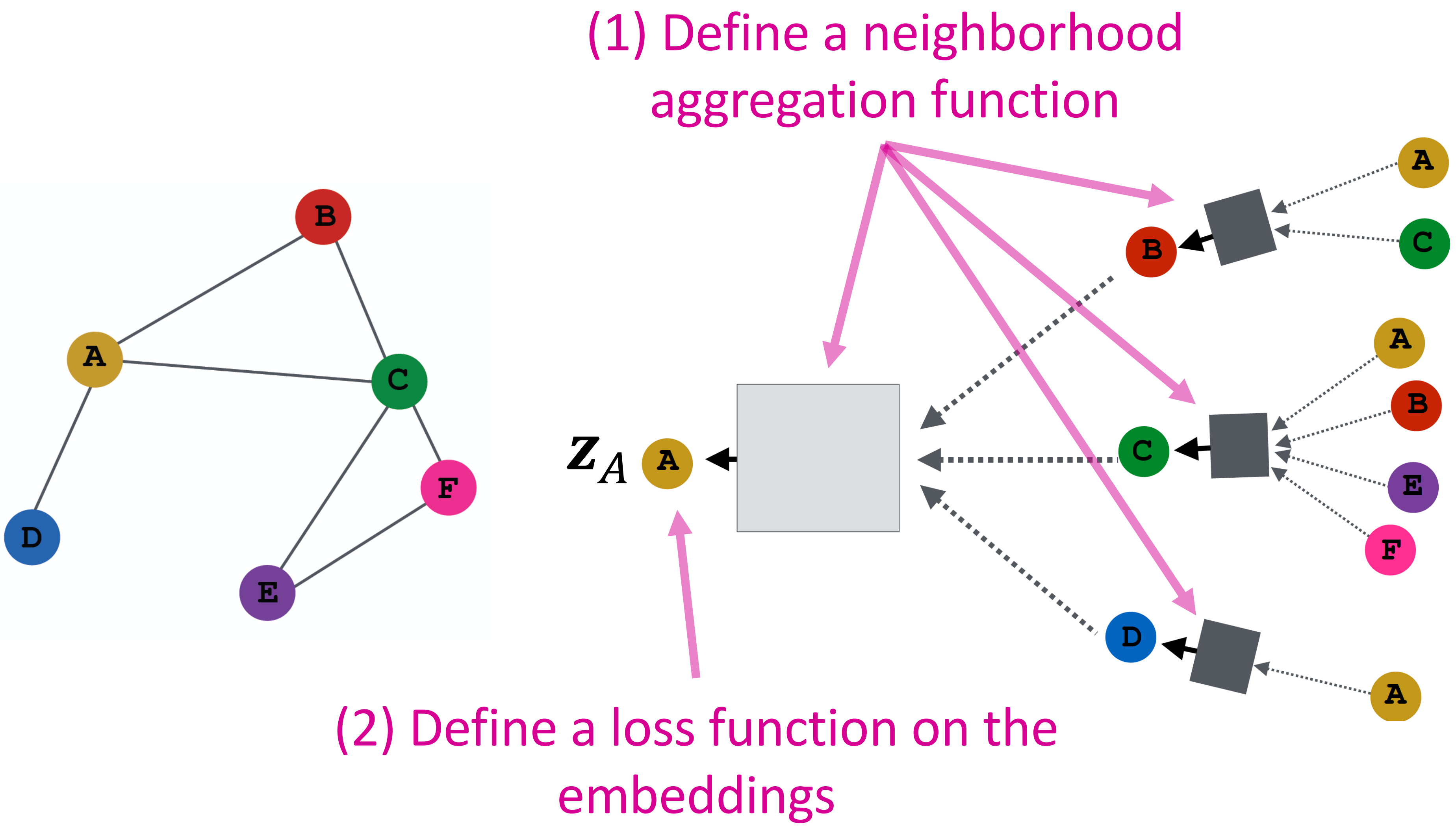
# How to train a GNN

- GNN provides us node embedding $\boldsymbol{z}_v$
- **Supervised setting**:
- we want to minimize the loss $\mathcal{L}$:

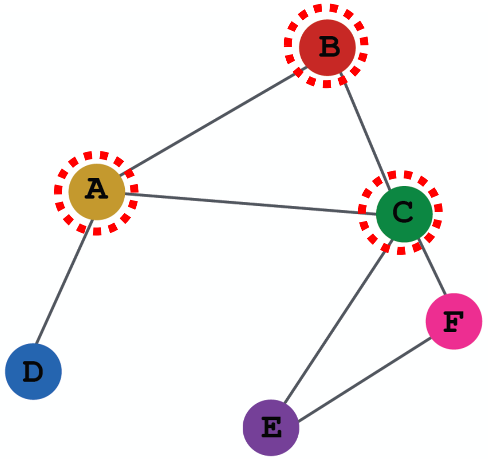
$$\min_{\Theta} \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{z}_v))$$

  - $\boldsymbol{y}$: node/egde/graph label (from external sources)
  - $\mathcal{L}$ could be L2 if $\boldsymbol{y}$ is real number, or cross entropy if $\boldsymbol{y}$ is categorical
- **Unsupervised setting:**
  - Use graph structure/feature itself as supervision
    - E.g., link prediction, masked feature prediction, …

# Model Design: Overview

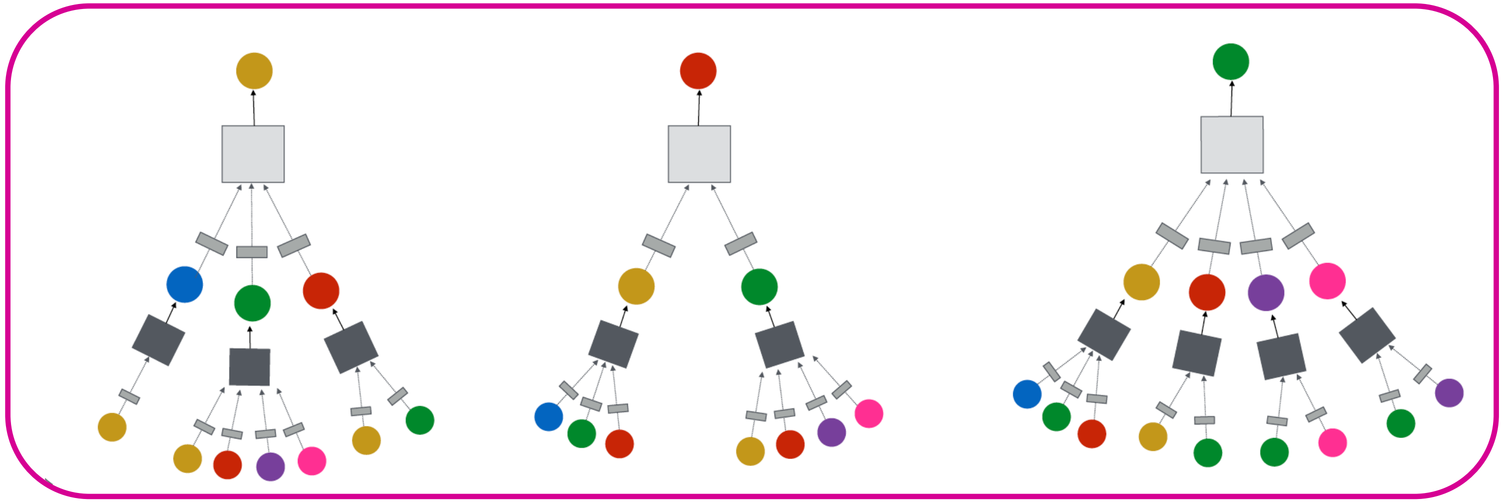


# Model Design: Overview

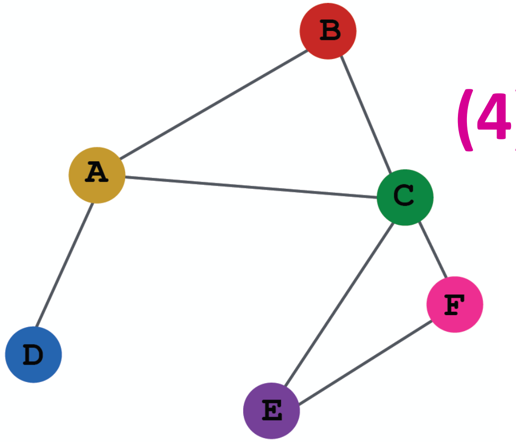



INPUT GRAPH

**(3) Train on a set of nodes, i.e., a batch of computational graphs**



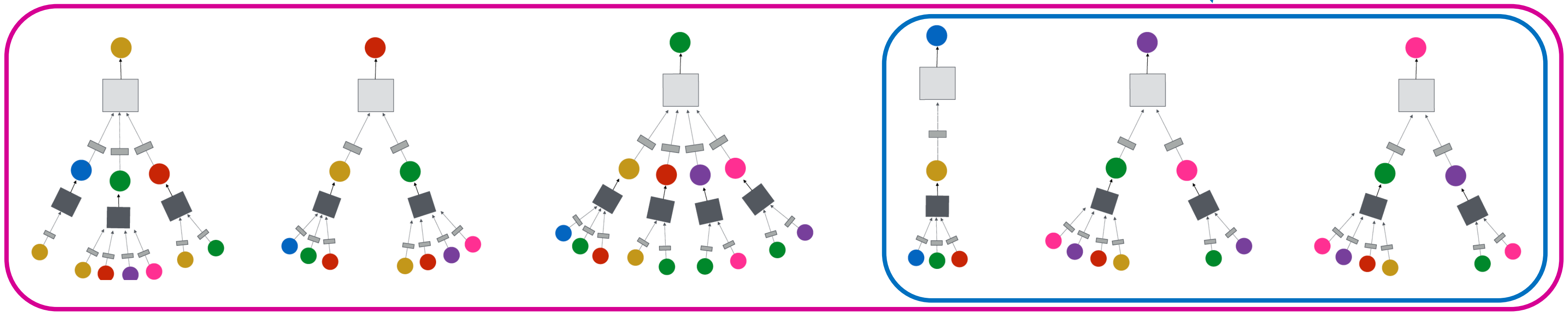
# Model Design: Overview




INPUT GRAPH

(4) Test time: Generate embeddings for nodes as needed

Even for nodes we never trained on!

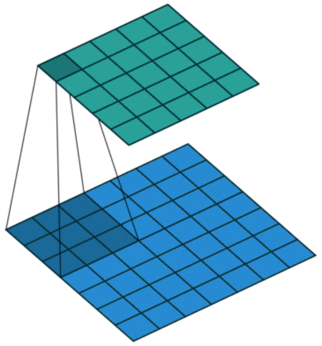


Slides adapted from Stanford CS224W Course

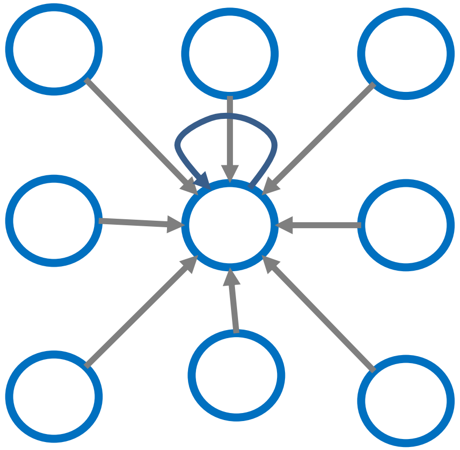
# GNN vs CNN & Transformer

# GNN vs CNN

Convolutional neural network (CNN) layer with 3x3 filter:



Image


Graph

- GNN formulation: $\mathrm{h}_v^{(l+1)} = \sigma(\mathbf{W}_l \sum_{u \in \mathrm{N}(v)} \frac{\mathrm{h}_u^{(l)}}{|\mathrm{N}(v)|} + \mathrm{B}_l \mathrm{h}_v^{(l)}), \forall l \in \{0, \ldots, L-1\}$
- CNN formulation: $\mathrm{h}_v^{(l+1)} = \sigma(\sum_{u \in \mathrm{N}(v)} \mathbf{W}_l^u \mathrm{h}_u^{(l)} + \mathrm{B}_l \mathrm{h}_v^{(l)}), \forall l \in \{0, \ldots, L-1\}$

**Key difference**: We can learn different $\boldsymbol{W}_l^u$ for different "neighbor" $u$ for pixel $v$ on the image

# GNN vs CNN

Convolutional neural network (CNN) layer with 3x3 filter:

**CNN can be seen as a special GNN with fixed neighbor size and ordering:**

- **The size of the filter is pre-defined for a CNN.**
- **The advantage of GNN is it processes arbitrary graphs with different degrees for each node.**

**CNN is not permutation invariant/equivariant.**

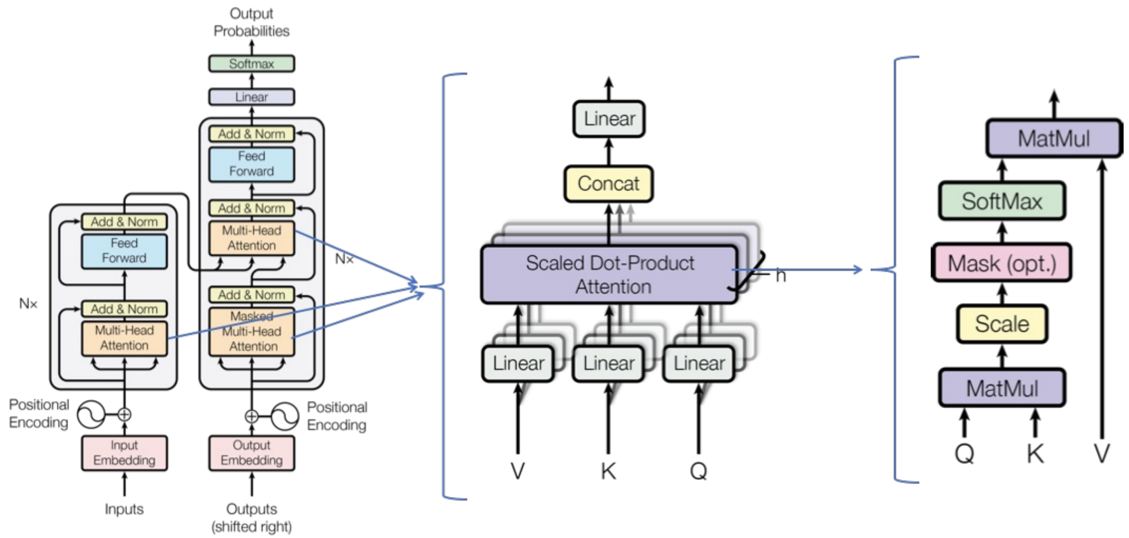
- **Switching the order of pixels will leads to different outputs.**

- GNN form
- CNN form $\sigma(\sum_{u \in N(v)} W_l^u h_u + B_l h_v), \forall l \in \{0, \dots, L-1\}$

**Key difference**: We can learn different $W_l^u$ for different "neighbor" $u$ for pixel $v$ on the image

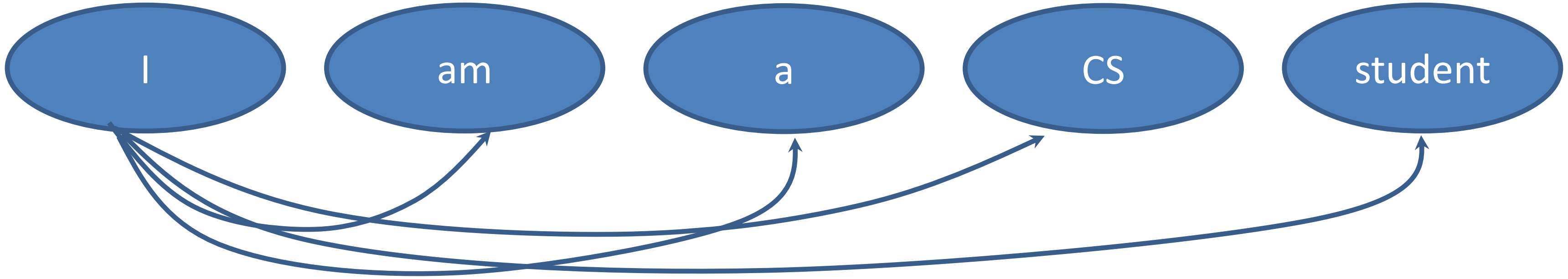
# Transformer

Transformer is one of the most popular architectures that achieves great performance in many sequence modeling tasks.




**Key component: self-attention**

- Every token/word attends to all the other tokens via matrix multiplication.

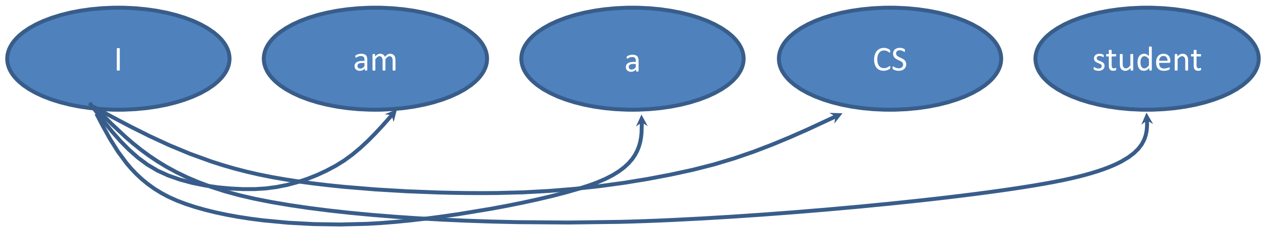


A nice blog plot for this: https://towardsdatascience.com/transformers-are-graph-neural-networks-bca9f75412aa

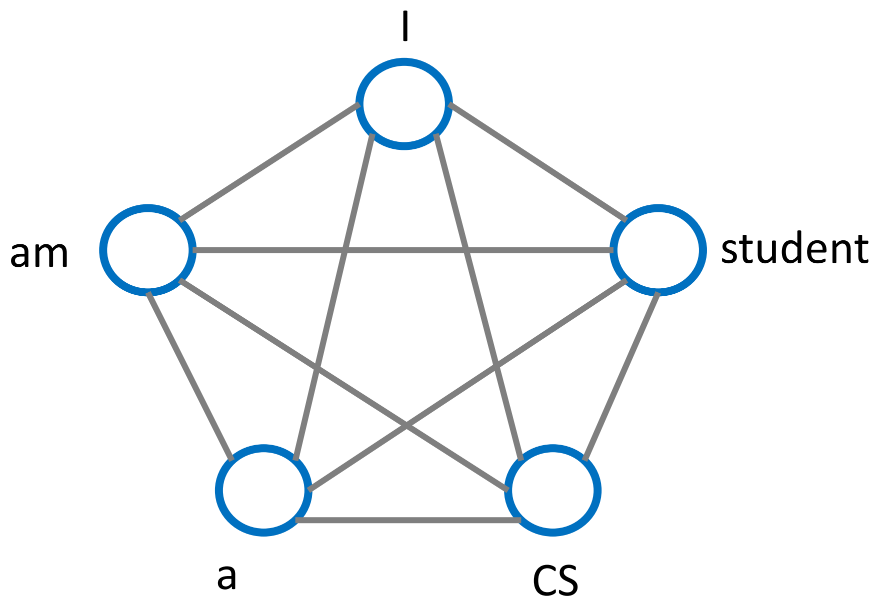
# GNN vs Transformer

Transformer layer can be seen as a special GNN that runs on a fully-connected “token graph”!

Since each word attends to **all the other tokens**, **the computation graph** of a transformer layer is identical to that of a GNN on the **fully-connected “token graph”**.




Text




Fully-connected Graph

Slides adapted from Stanford CS224W Course

# Applications of GNNs

# Tasks on Networks

**Tasks we will be able to solve:**

- Node classification
  - Predict a type of a given node
- Link prediction
  - Predict whether two nodes are linked
- Subgraph detection
  - Identify certain subgraphs or paths within a graph
- Graph classification
  - Classify different graphs

# Example (1): Financial Networks

- **Financial Networks:** Describe financial entities and their connections

### International banking

- *Nodes: Countries*
- *Edges: Capital flows*

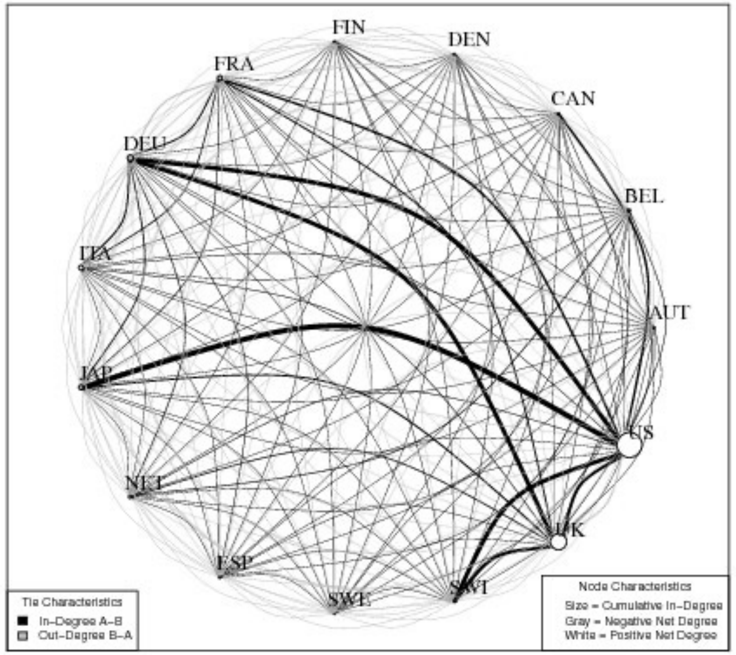



Image credit: The Political Economy of Global Finance: A Network Model

### Bitcoin transactions

- *Nodes: BTC wallets*
- *Edges: Transactions*

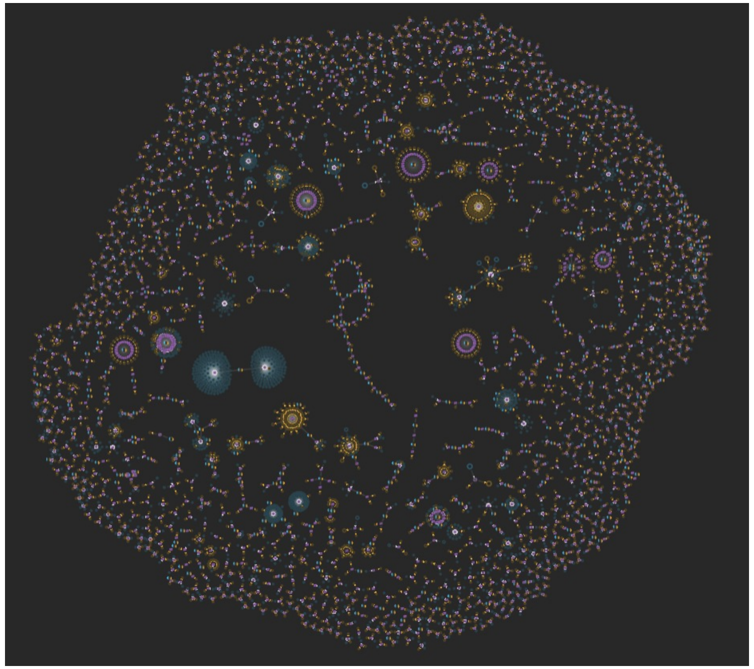


Image credit: https://dailyblockchain.github.io/

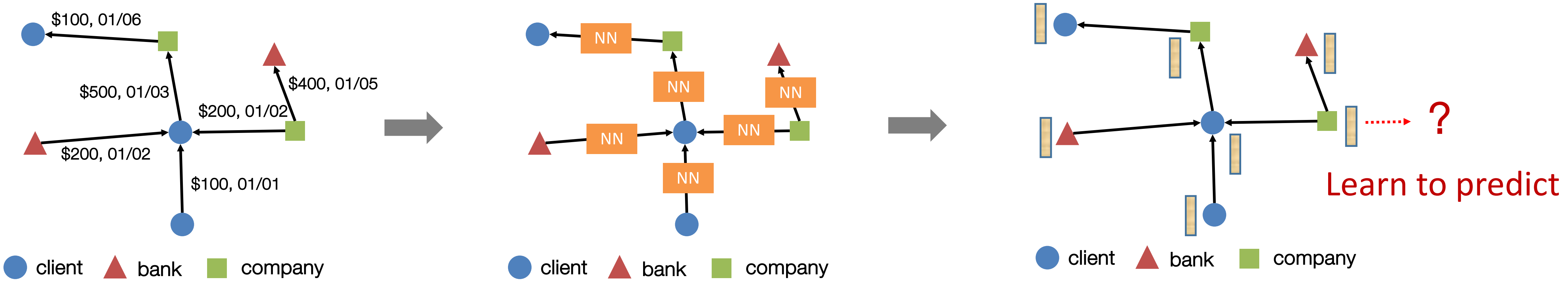
J. You, T. Du, J. Leskovec. ROLAND: Graph Learning Framework for Dynamic Graphs, KDD 2022

# ROLAND: GNN for Financial Networks

- **ROLAND framework:**
  - Transform financial networks as GNN computational graphs
  - Learning from diverse objectives (node and edge level)

Self-supervised (from raw data)
- Will a user make a transaction? Yes
- What is the amount? $500
- When will it happen? 01/03
- …

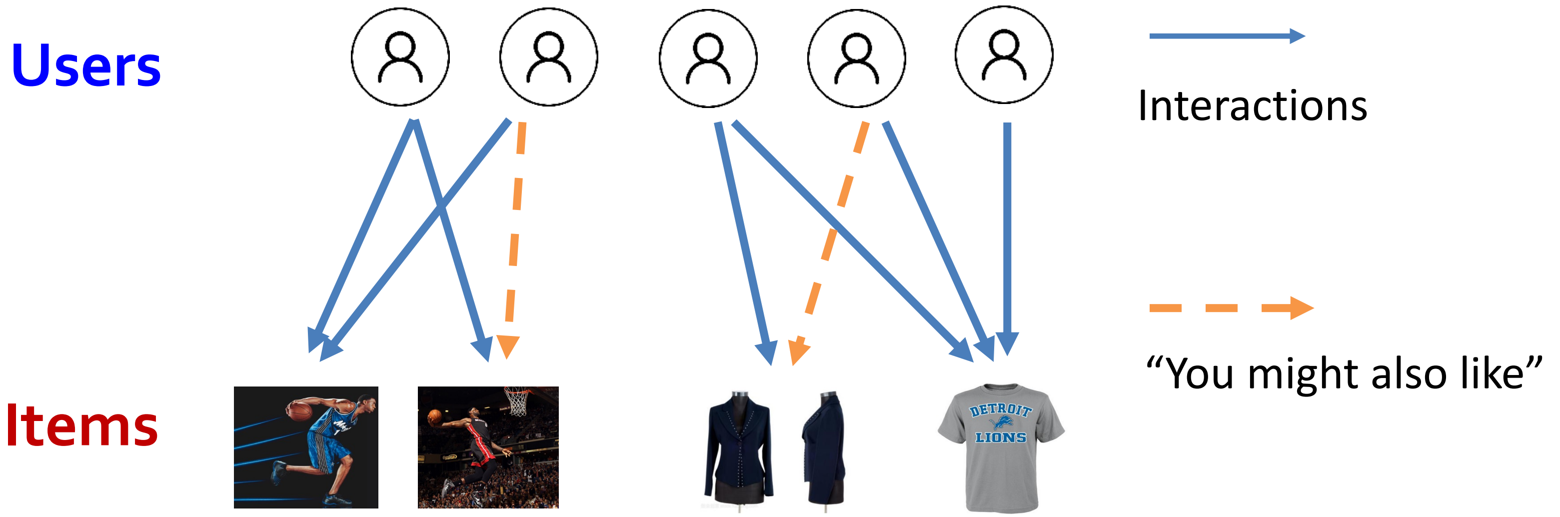
Supervised (from external sources)
- Does a user involve fraud? No
- Does a user involve money laundering? Yes
- …




Financial networks | Graph Neural Networks | Learning objectives

# Example (2): Recommender Systems

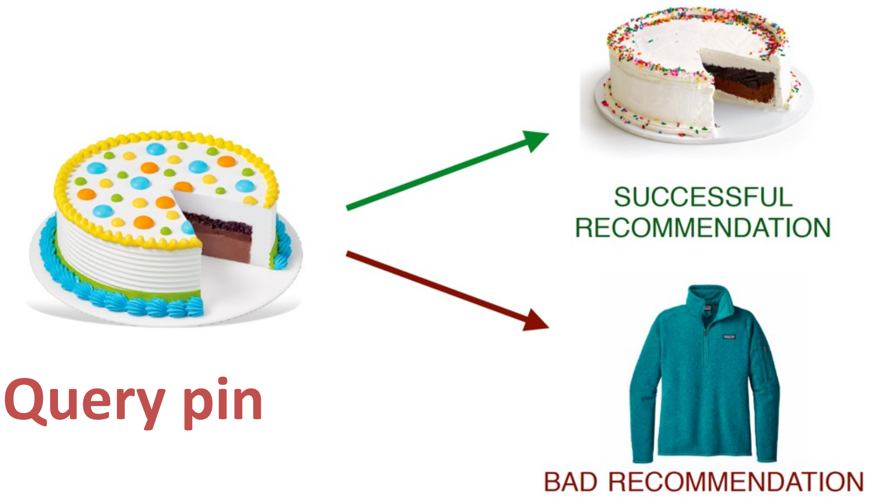
- **Users interacts with items**
  - Watch movies, buy merchandise, listen to music
  - **Nodes:** Users and items
  - **Edges:** User-item interactions
- **Goal: Recommend items users might like**



Ying et al., Graph Convolutional Neural Networks for Web-Scale Recommender Systems, KDD 2018

# PinSage: Graph-based Recommender

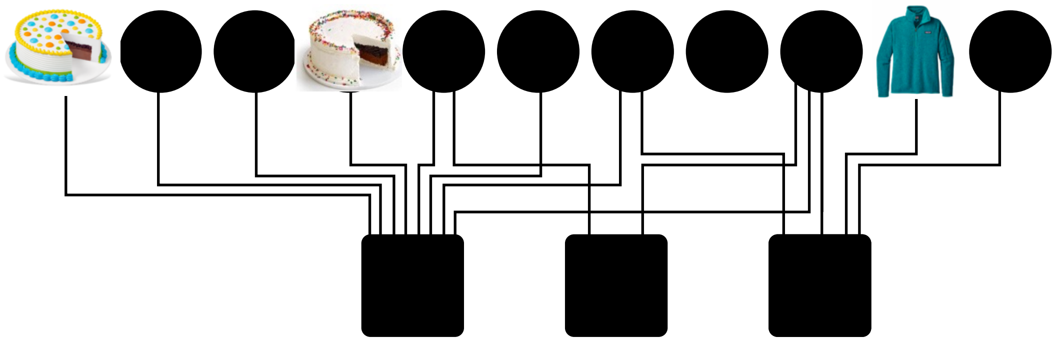
## Task: Recommend related pins to users

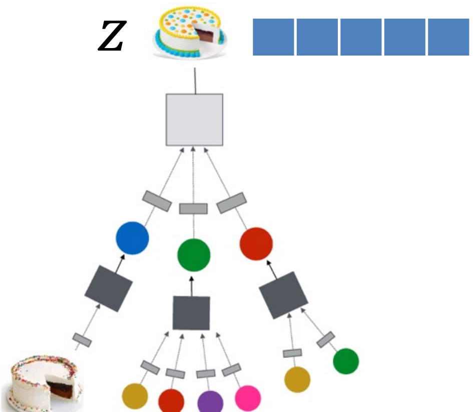



**Task:** Learn node embeddings $z_i$ such that

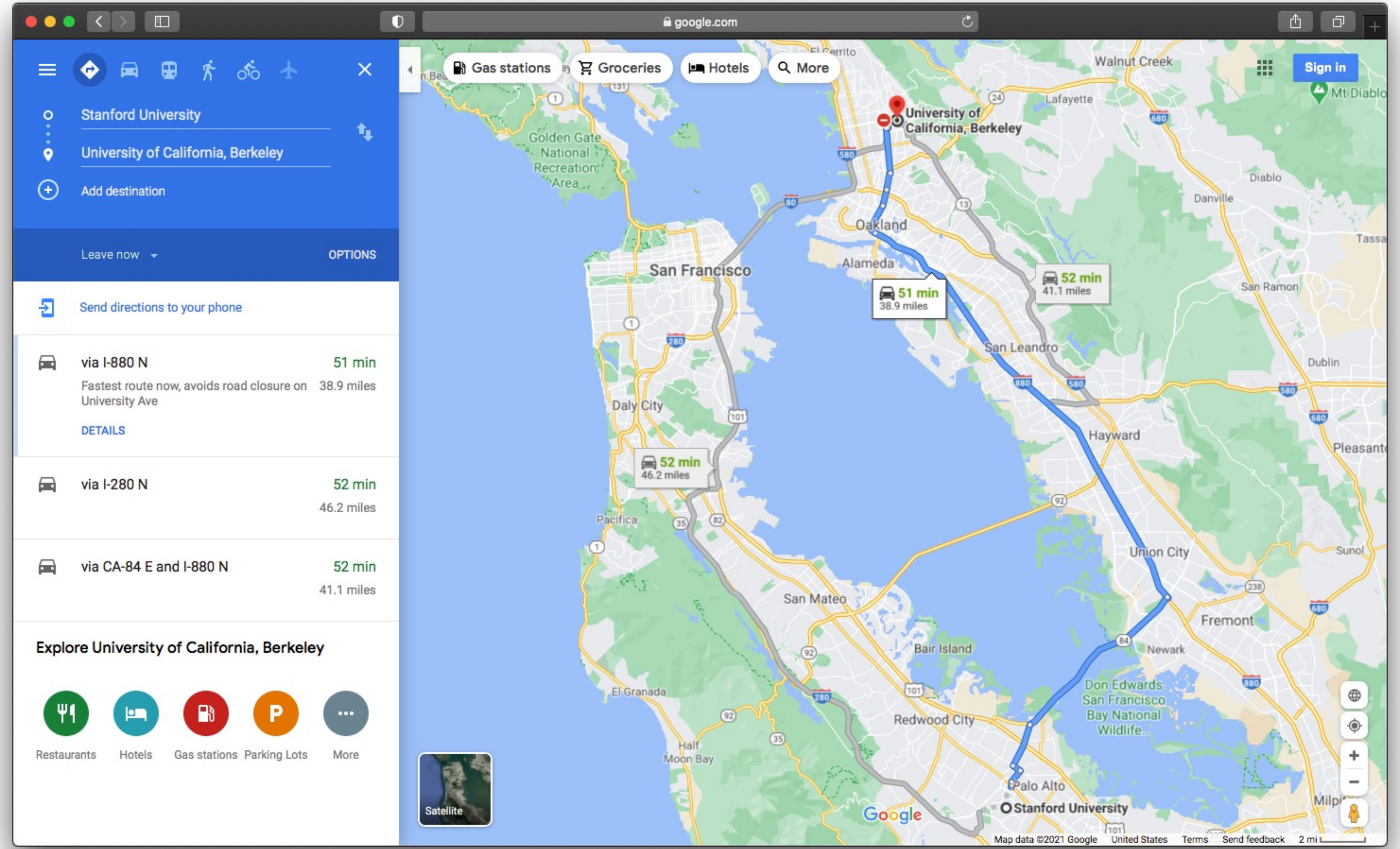
$$d(z_{cake1}, z_{cake2}) < d(z_{cake1}, z_{sweater})$$

## Predict whether two nodes in a graph are related





# Example (3): Traffic Prediction



# Road Network as a Graph

- **Nodes:** Road segments
- **Edges:** Connectivity between road segments

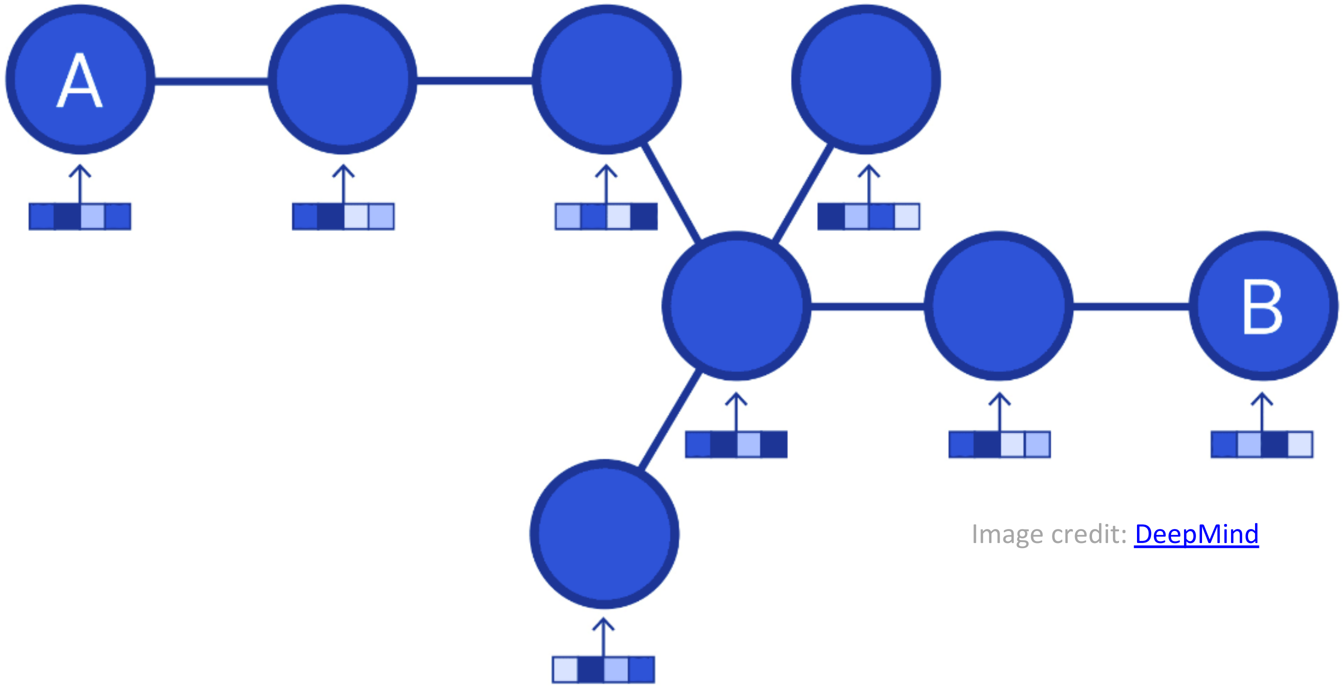
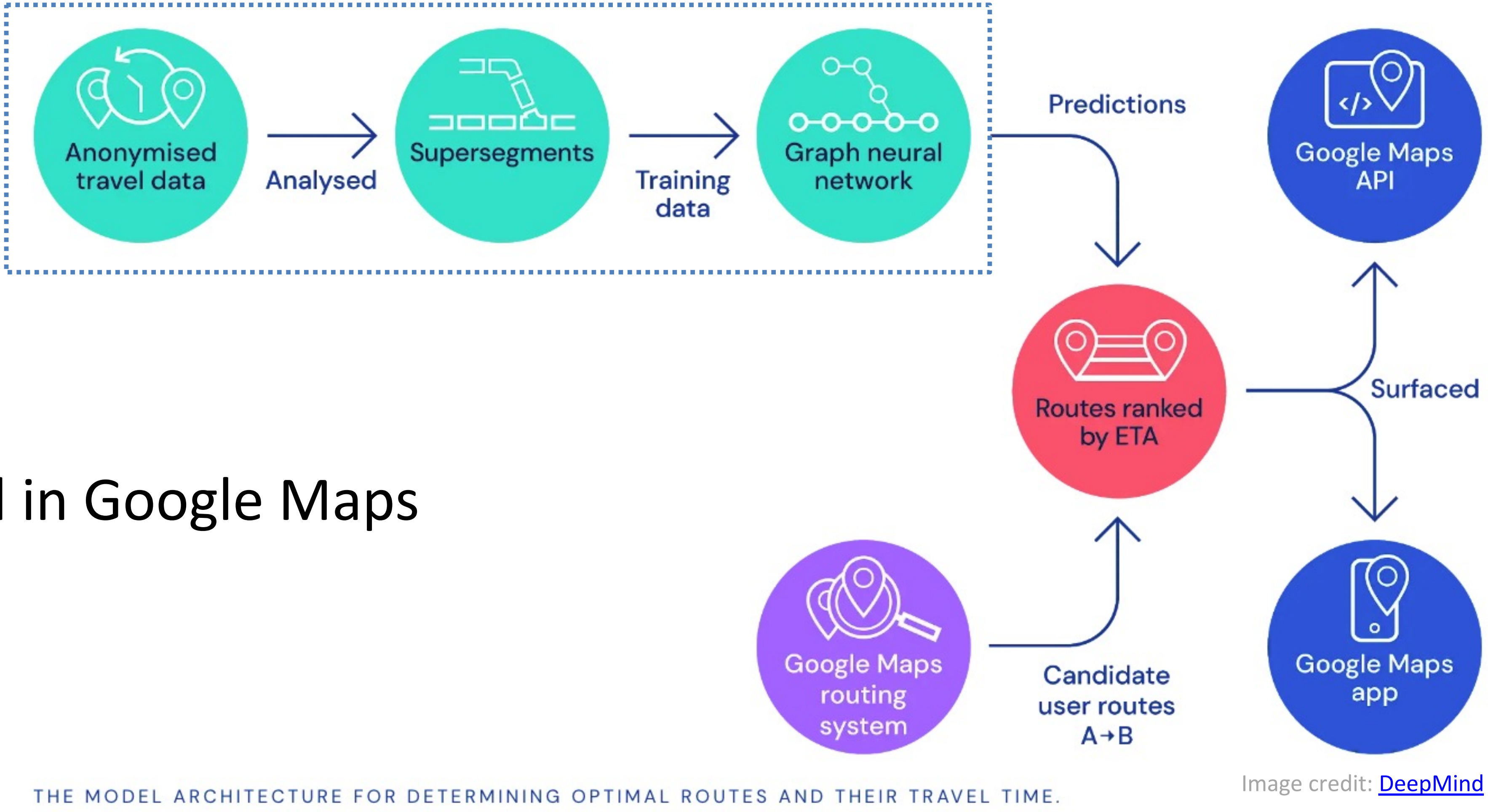



Image credit: DeepMind

# Traffic Prediction via GNN

**Predict the best route via Graph Neural Networks**

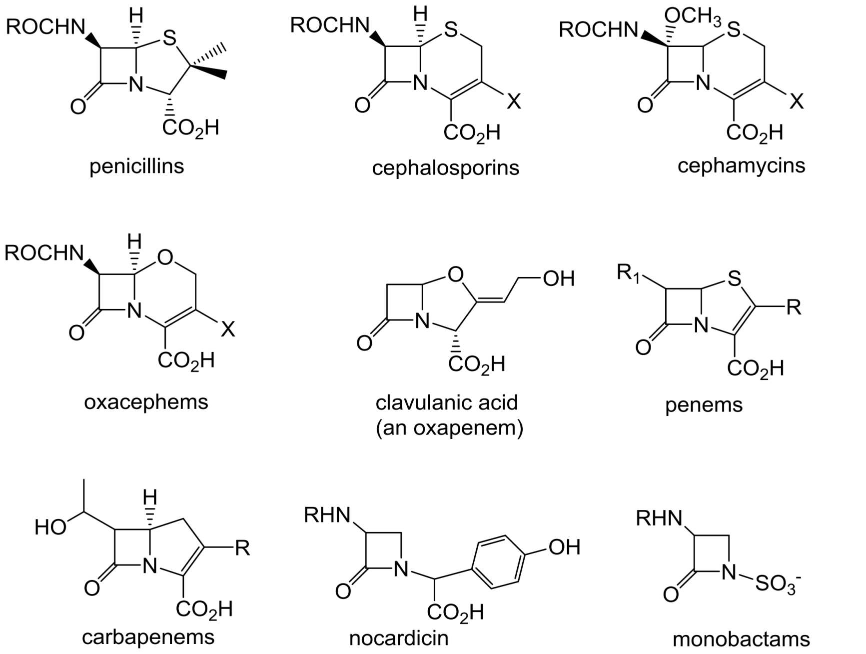



- Used in Google Maps

Image credit: DeepMind

# Example (4): Drug Discovery

- **Antibiotics are small molecular graphs**
  - **Nodes:** Atoms
  - **Edges:** Chemical bonds




Konaklieva, Monika I. "Molecular targets of β-lactam-based antimicrobials: beyond the usual suspects." Antibiotics 3.2 (2014): 128-142.

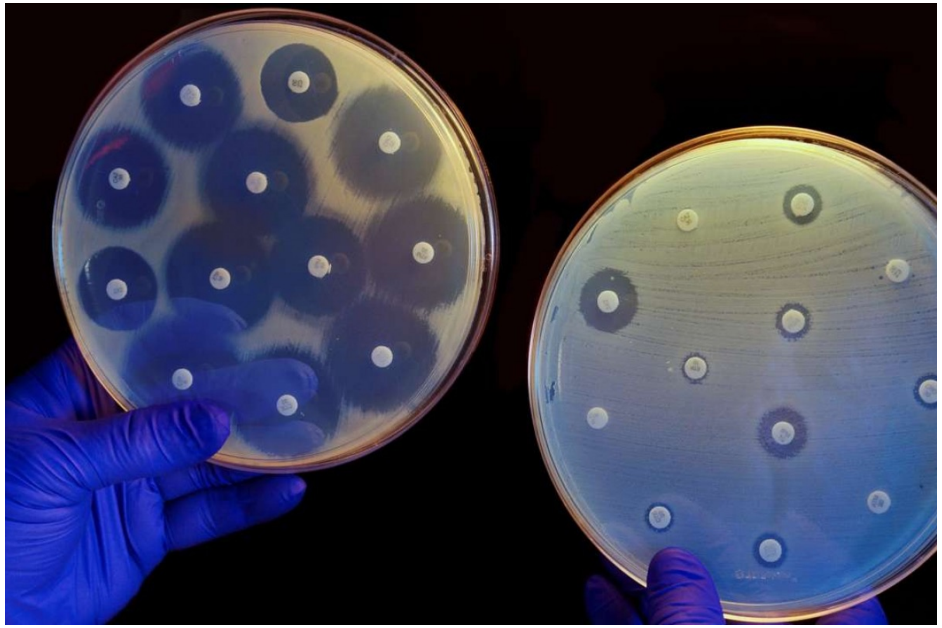
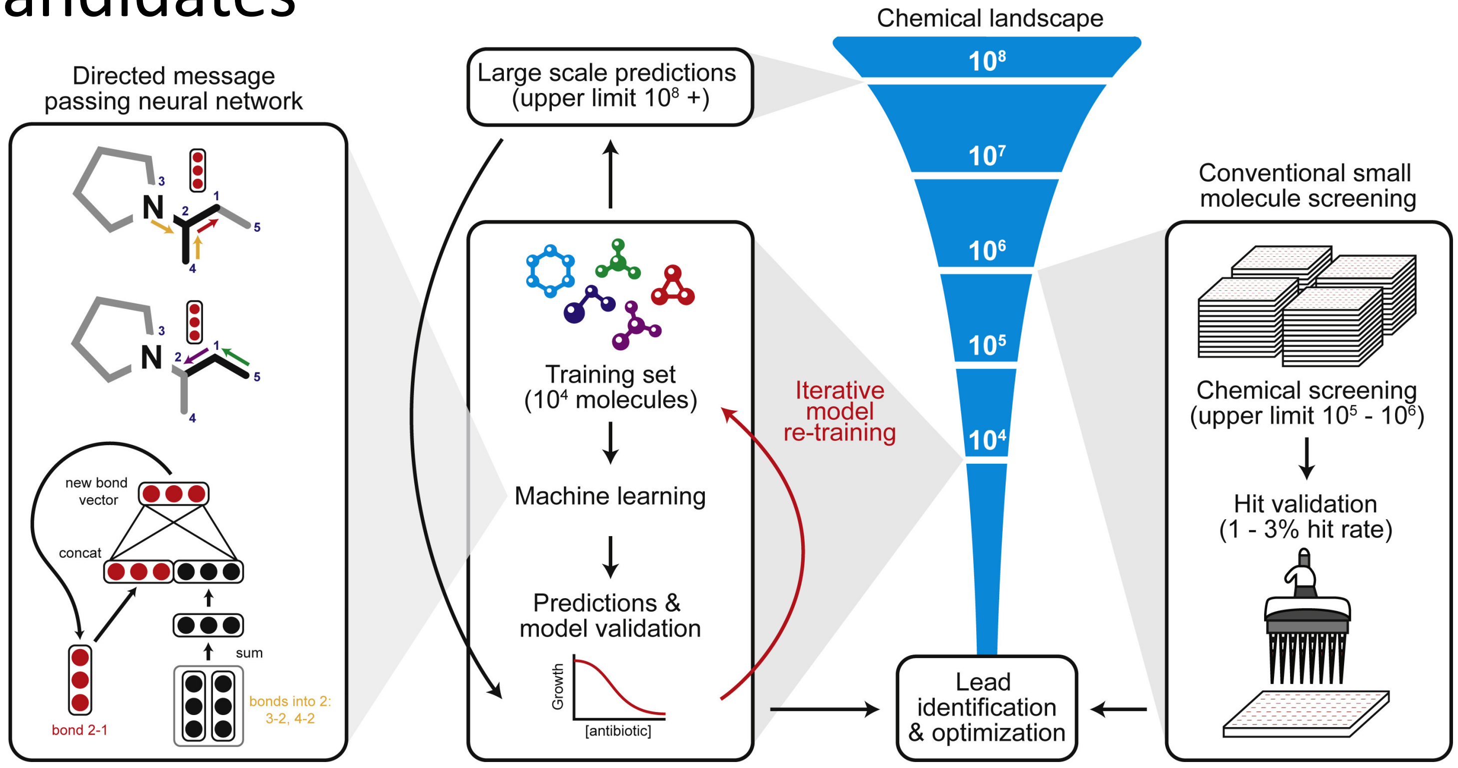


Image credit: CNN

# Deep Learning for Antibiotic Discovery

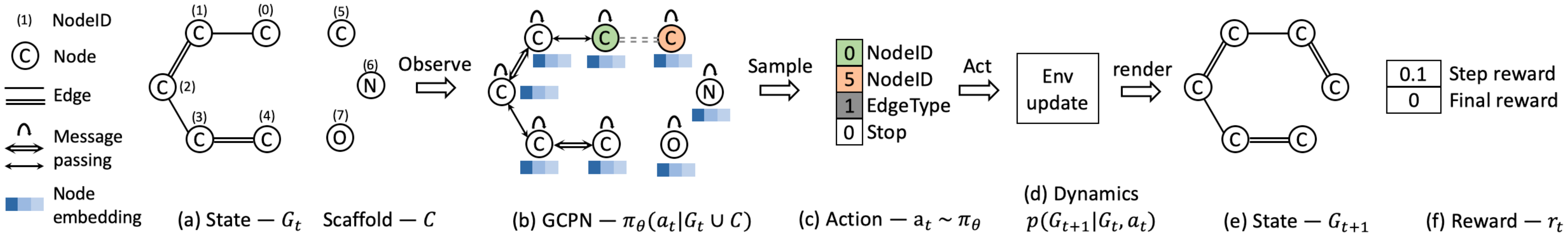
- A **graph classification task**
- Predict promising molecules from a pool of existing candidates



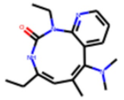

Stokes, Jonathan M., et al. "A deep learning approach to antibiotic discovery." Cell 180.4 (2020): 688-702.

# Molecule Generation / Optimization

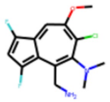
## **Graph generation**: Generating novel molecules



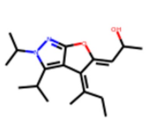

**Use case 1: Generate novel molecules with high drug likeness**



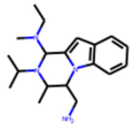
0.948



0.945

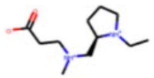


0.944

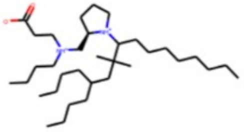


0.941

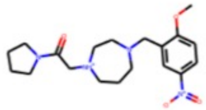
**Use case 2: Optimize existing molecules to have desirable properties**



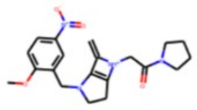
-8.32



-0.71



-5.55



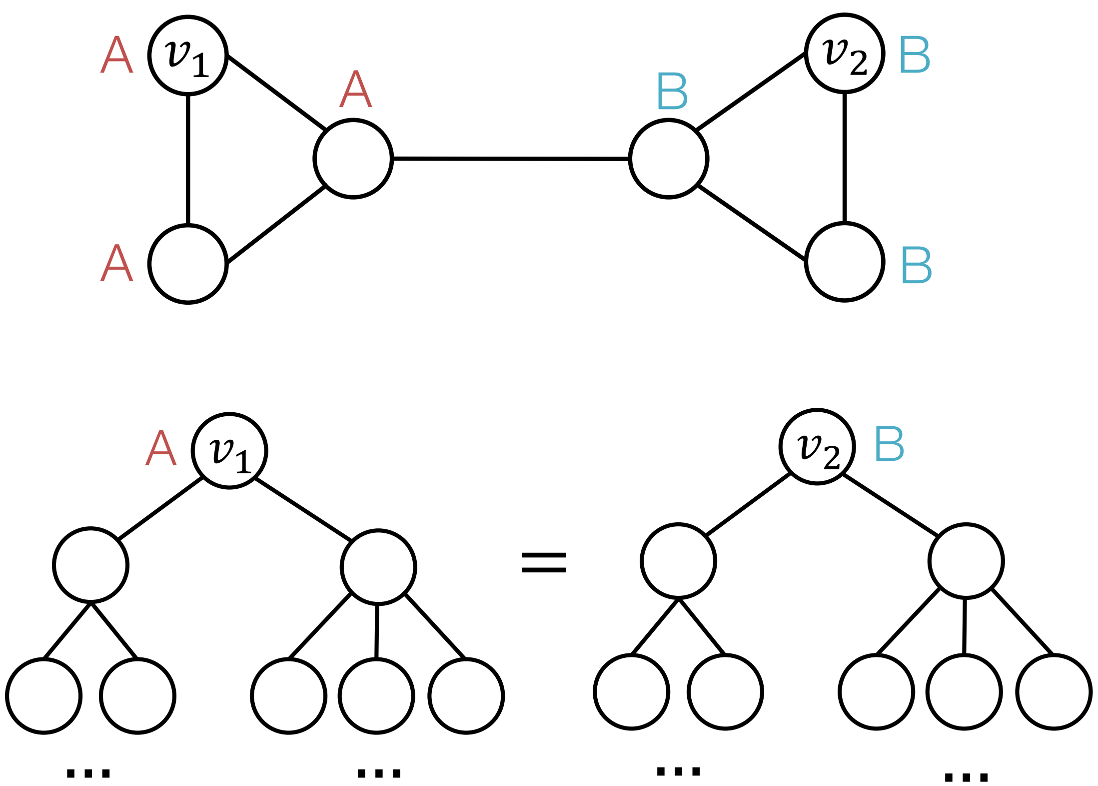
-1.78

# Frontiers of Graph ML Research

J. You, R. Ying, J. Leskovec. Position-aware Graph Neural Networks, ICML 2019.

# Designing more Expressive GNNs

**Position-aware task**



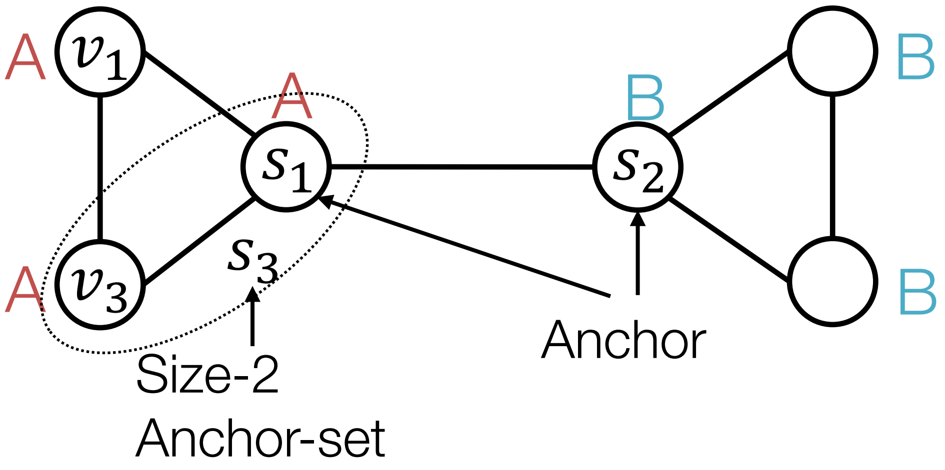

- GNNs fail at Position-aware tasks ☹
- $v_1$ and $v_2$ will always have the same computational graph, **due to structure symmetry**

- **Q:** Can we define deep learning methods that are position-aware?

J. You, R. Ying, J. Leskovec. Position-aware Graph Neural Networks, ICML 2019.

# Idea: P-GNN

- P-GNN proposes the first notion of **position embeddings for graphs**
  - Notably, Position embeddings are crucial for Transformers and LLMs



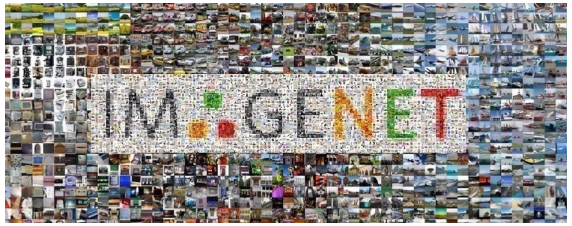

| | $s_1$ | $s_2$ | $s_3$ |
|---|---|---|---|
| $v_1$ | 1 | 2 | 1 |
| $v_3$ | 1 | 2 | 0 |

$v_1$'s Position embedding

$v_3$'s Position embedding

- P-GNN inspires many successful application of **Transformer + Graphs**
  - E.g., **GAT-POS** [Ma et al., 2021], **Graphormer** [Ying et al., 2021], …

# Graphs are Ubiquitous in ML problems

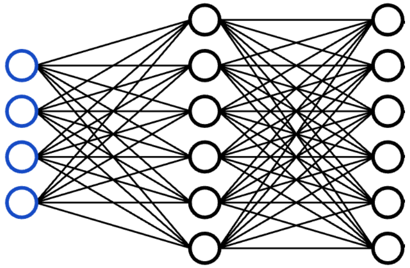



Q: What is your favorite animal?
A: My favorite animal is a dog.

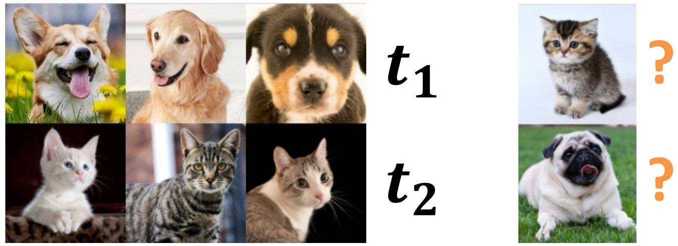
Q: Why?
A: Because dogs are loyal and friendly.

Q: What are two reasons that a dog might be in a bad mood?
A: Two reasons that a dog might be in a bad mood are if it is hungry

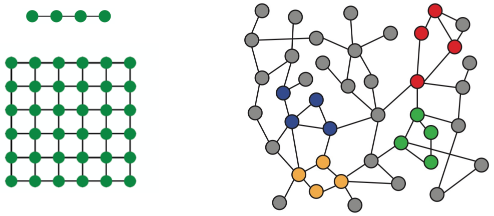
**Input data**



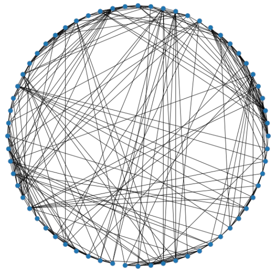
**Neural networks**



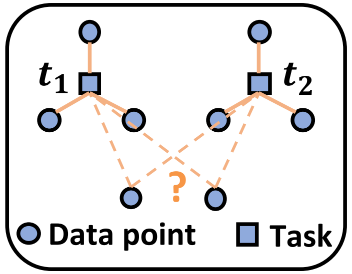

**ML tasks**



**Graph** is a superset for existing **ML input data**



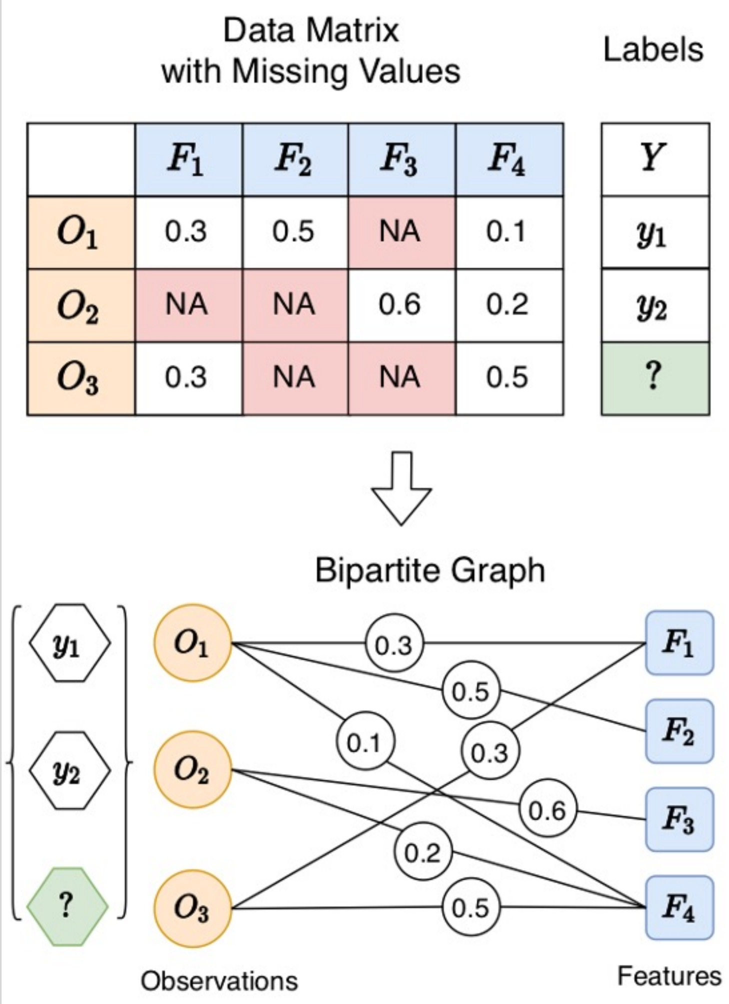
Understand and inspire **ML methods** with graphs




**Graph** can represent novel **ML applications**

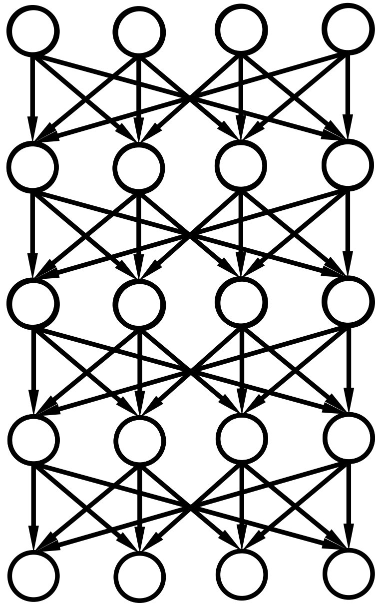
J. You, et al. Handling Missing Data with Graph Representation Learning, NeurIPS 2020

# (1) **Graphs** in Missing Data Problems



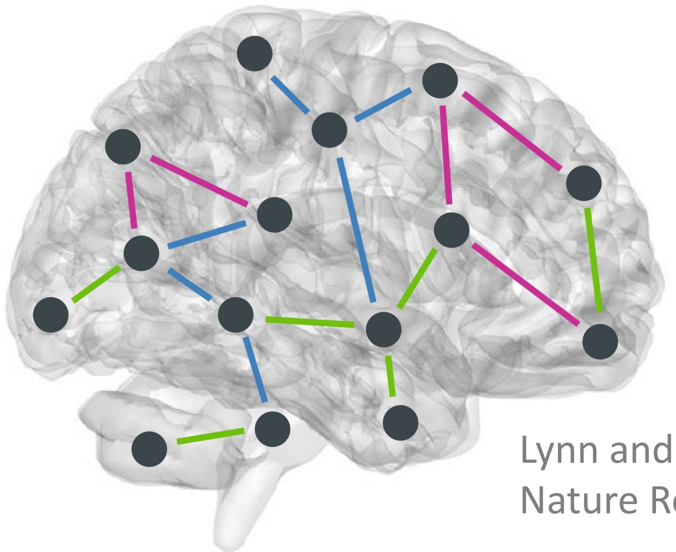

- **Real-world data often exhibit missing values**
- **Idea: Input data as heterogenous graph**
  - **Nodes: Data points** and **features**
  - **Edges:** Link data points with features
- **Graph offers unified solution for missing data problem**
  - **Feature imputation – edge-level prediction**
  - **Label prediction – node-level prediction**
- **10~20% lower MAE than SOTA baselines**

# (2) New NN representation: Relational Graph



**(Artificial) neural network**

**Gap**



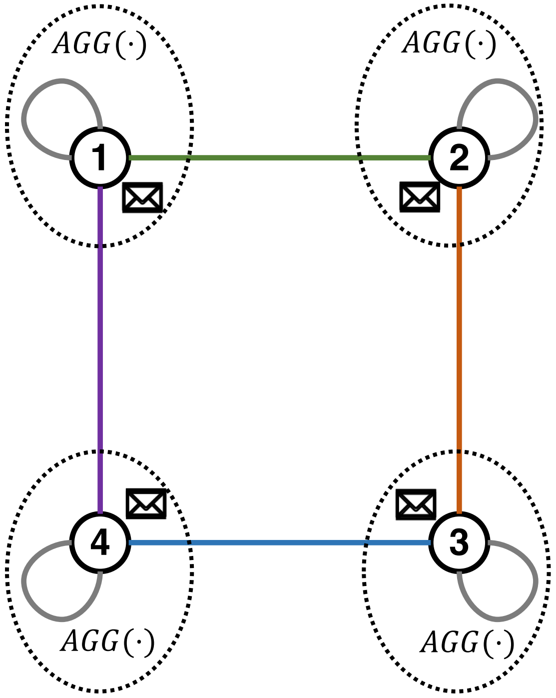
Lynn and Bassett, Nature Rev. Phys. 2019

**Brain network**

**Can we translate any graph** (e.g., brain network) **to a neural network**?

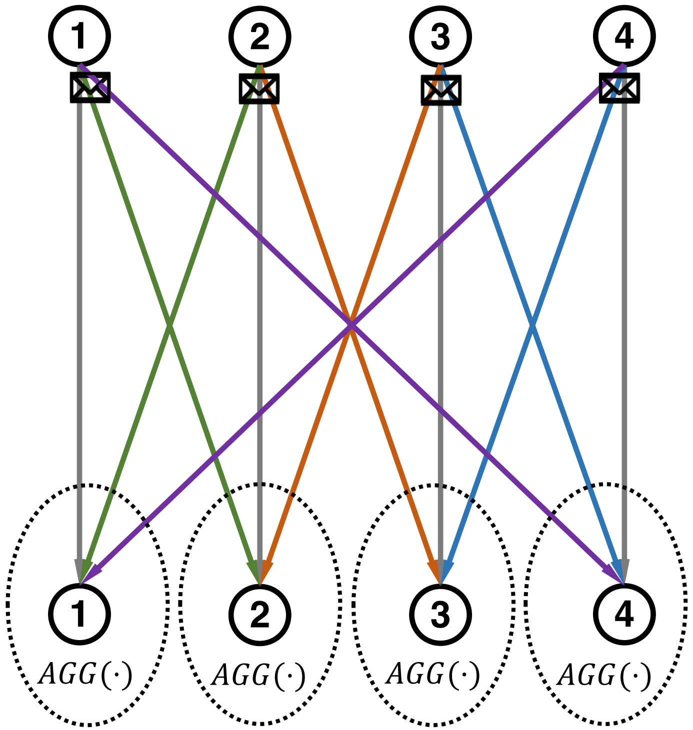
- Study the performance of NNs with **network science tools**
- Bridge deep learning with **neuroscience**

# (2) **New** NN representation: **Relational Graph**




**Relational Graph**

$\Longleftrightarrow$




**Neural network layer**

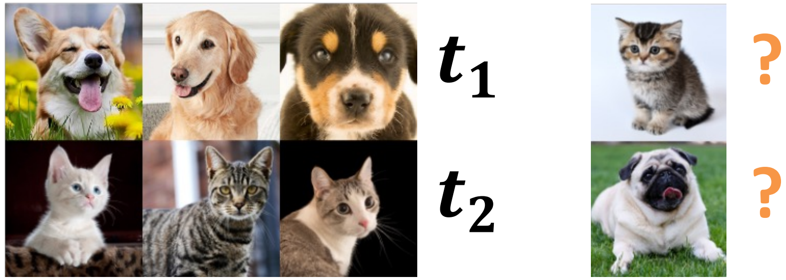
Directed message computation

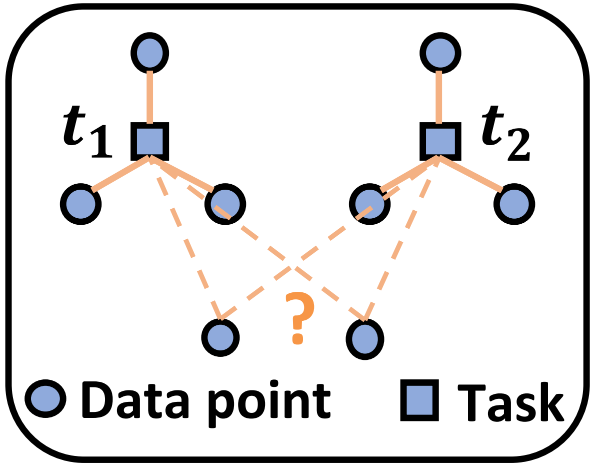
- **Translate any graph → NN**
- **Computation** is defined as **message passing** over the graph

K. Cao*, J. You*, J. Leskovec. Relational Multi-Task Learning: Modeling Relations between Data and Tasks, ICLR 2022

# (3) **Graphs** in Multi-task Learning Problems

- **Graph representation for multi-task learning** (supervised/meta learning)
  - **Nodes: Data points** and **ML tasks**
  - **Edges:** A data point labeled by a task
- **Innovations**
  - Solve various multi-task settings via **graph ML**
  - Explore **new multi-task learning settings**: Leverage **auxiliary labels** during inference
  - **~13% improvement** with auxiliary task info



# Summary

- **Why Graph Deep Learning?**
  - Enable DL for interconnected data
- **What is a GNN**
  - **Key:** iterative node neighborhood aggregation
  - CNN & Transformer can be considered as special GNNs
- **Applications of GNNs**
  - **Different levels:** Node, edge, subgraph, graph
- **Frontiers of Graph ML research**
  - Design more expressive GNNs
  - Empower general ML pipeline with graphs