Stanford CS520: Introduction to Graph Neural Networks

Jiaxuan You, Stanford University

(Slides adapted from CS224W: Machine Learning with Graphs)

Many Types of Data are Graphs

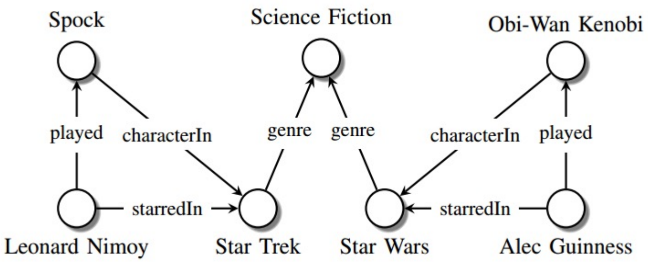


Image credit: Maximilian Nickel et al

Knowledge Graphs

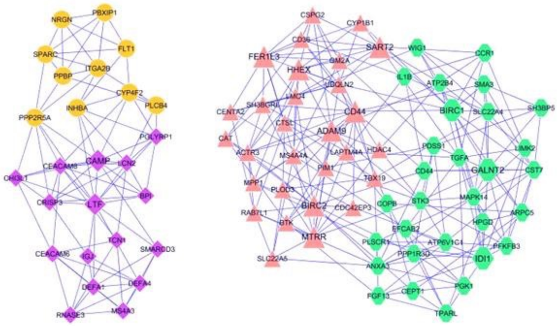


Image credit: ese.wustl.edu

Regulatory Networks

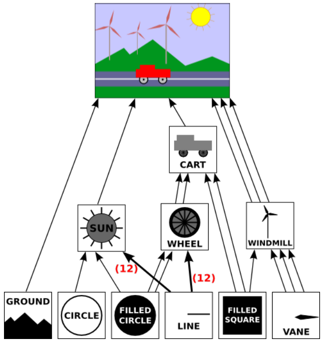


Image credit: math.hws.edu

Scene Graphs

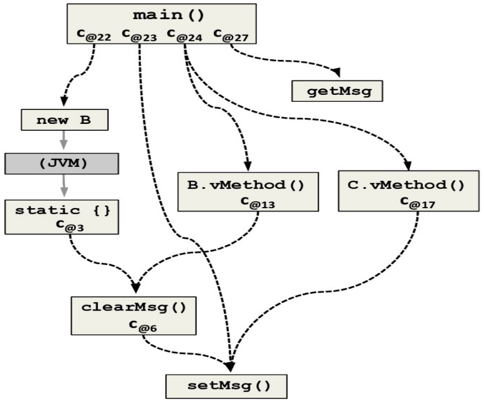


Image credit: ResearchGate

Code Graphs

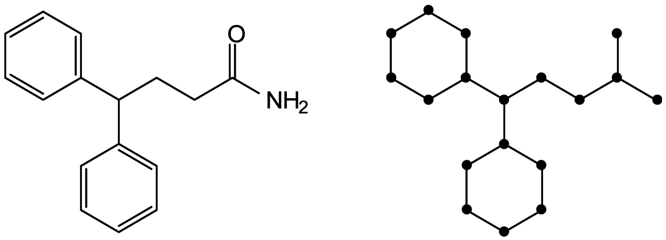


Image credit: MDPI

Molecules

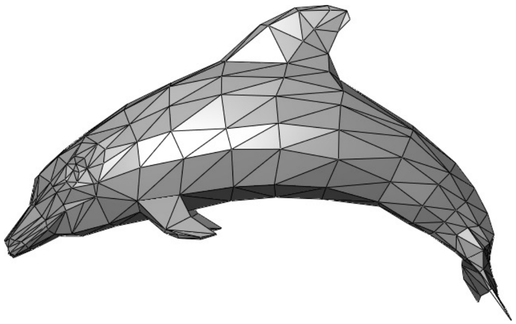


Image credit: Wikipedia

3D Shapes

Many Types of Data are Graphs

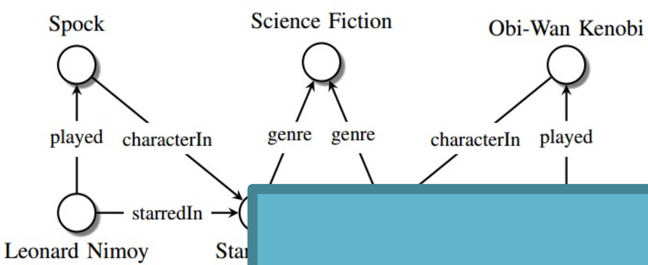
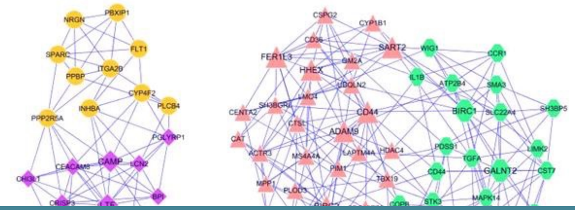
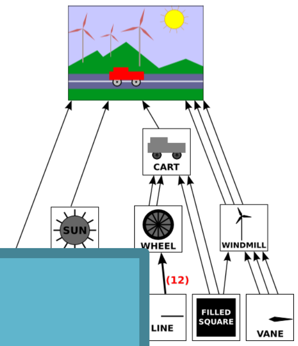


Image cred

Know





ath.hws.edu

Graphs

Main question:

How do we take advantage of relational structure for better prediction?

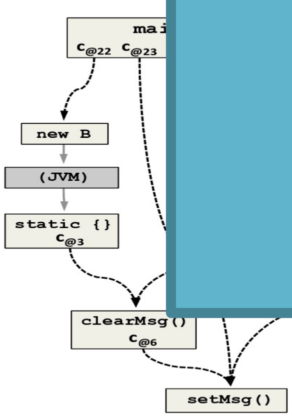


Image credit: ResearchGate

Code Graphs

Image credit: MDPI

Molecules

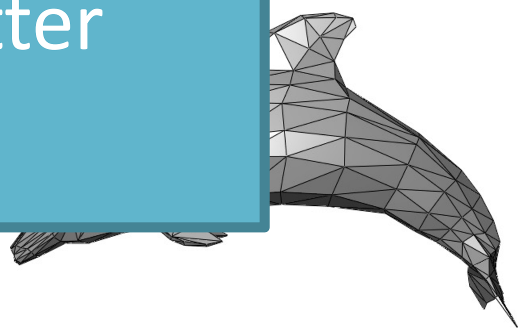


Image credit: Wikipedia

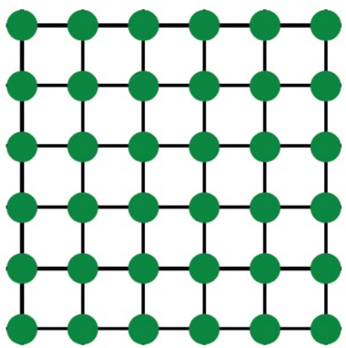
3D Shapes

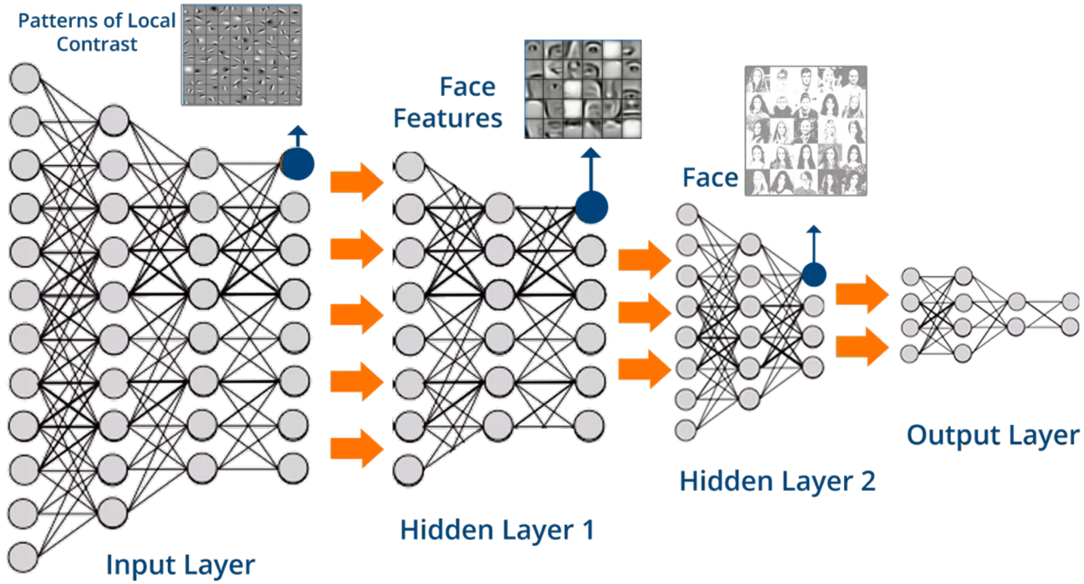
Graphs: Machine Learning

Complex domains have a rich relational structure, which can be represented as a **relational graph**

By explicitly modeling relationships we achieve better performance!

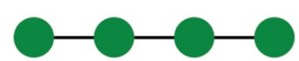
Modern ML Toolbox

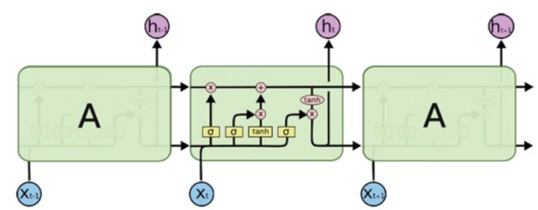




Images

Text/Speech



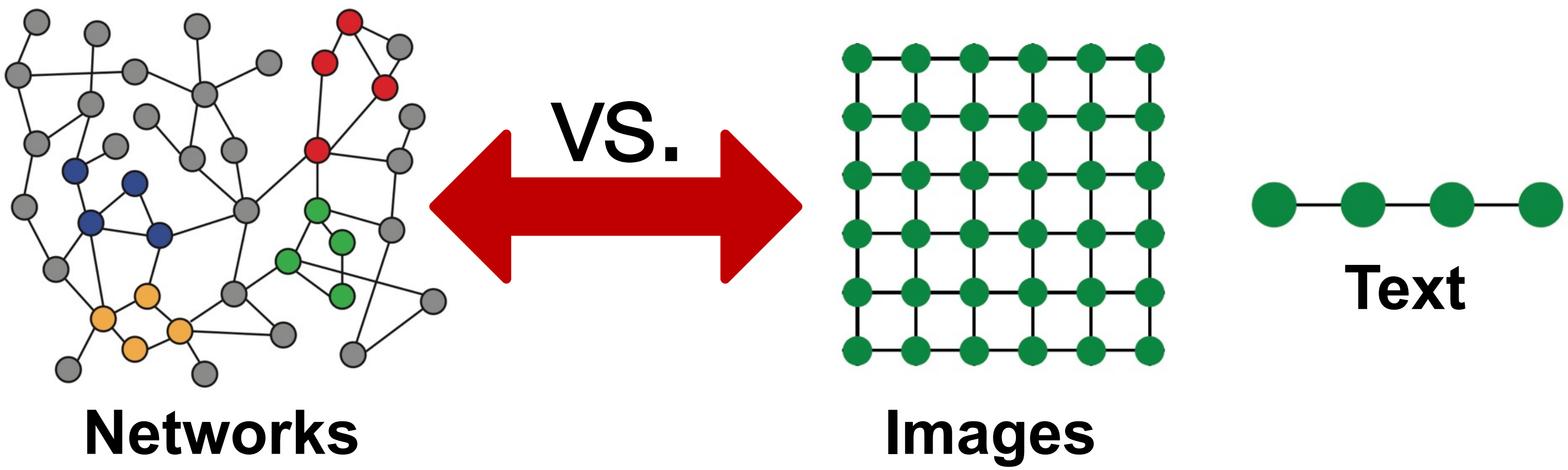


Modern deep learning toolbox is designed for simple sequences & grids

Why is it Hard?

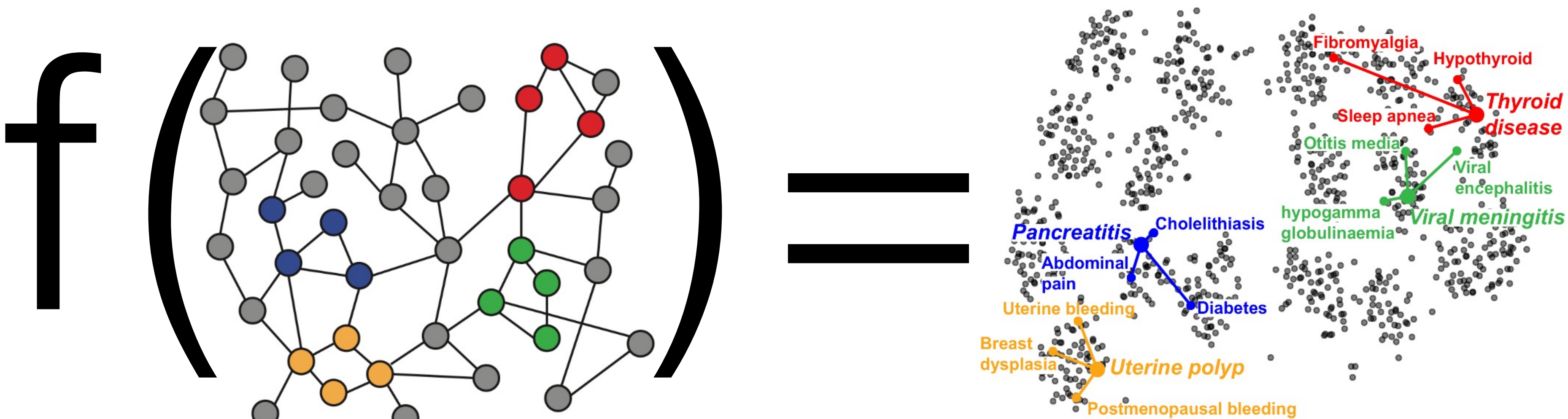
Networks are complex.

- Arbitrary size and complex topological structure (*i.e.*, no spatial locality like grids)



Node Embeddings

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



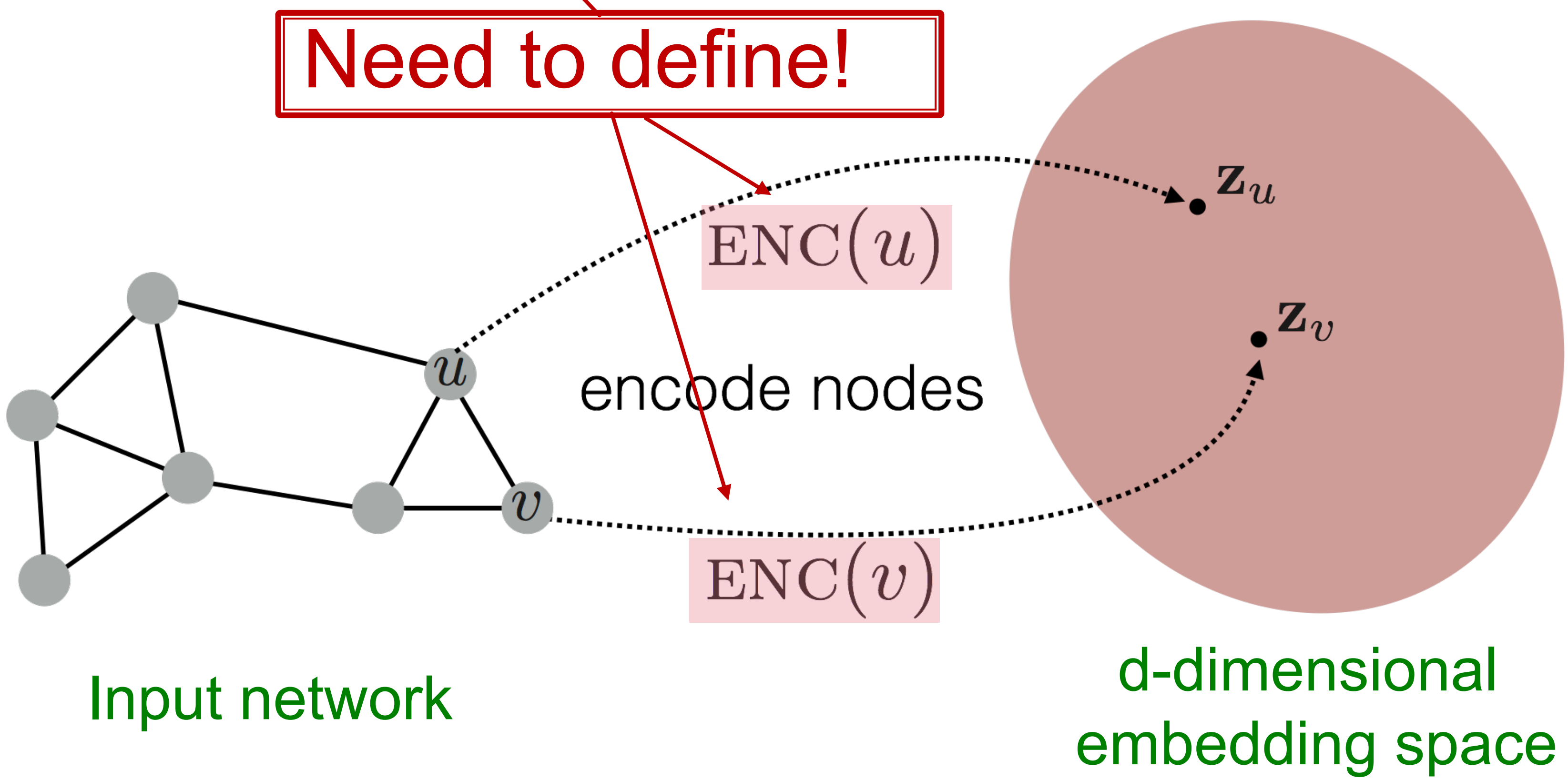
Input graph

2D node embeddings

How to <u>learn</u> mapping function f?

Node Embeddings

Goal: $\text{similarity}(u, v) \approx \mathbf{z}_v^{\mathrm{T}} \mathbf{z}_u$



Two Key Components

- **Encoder:** maps each node to a low-dimensional vector

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

d-dimensional embedding

node in the input graph

- **Similarity function:** specifies how the relationships in vector space map to the relationships in the original network

$$\text{similarity}(u, v) \approx \mathbf{z}_v^{\mathrm{T}} \mathbf{z}_u$$

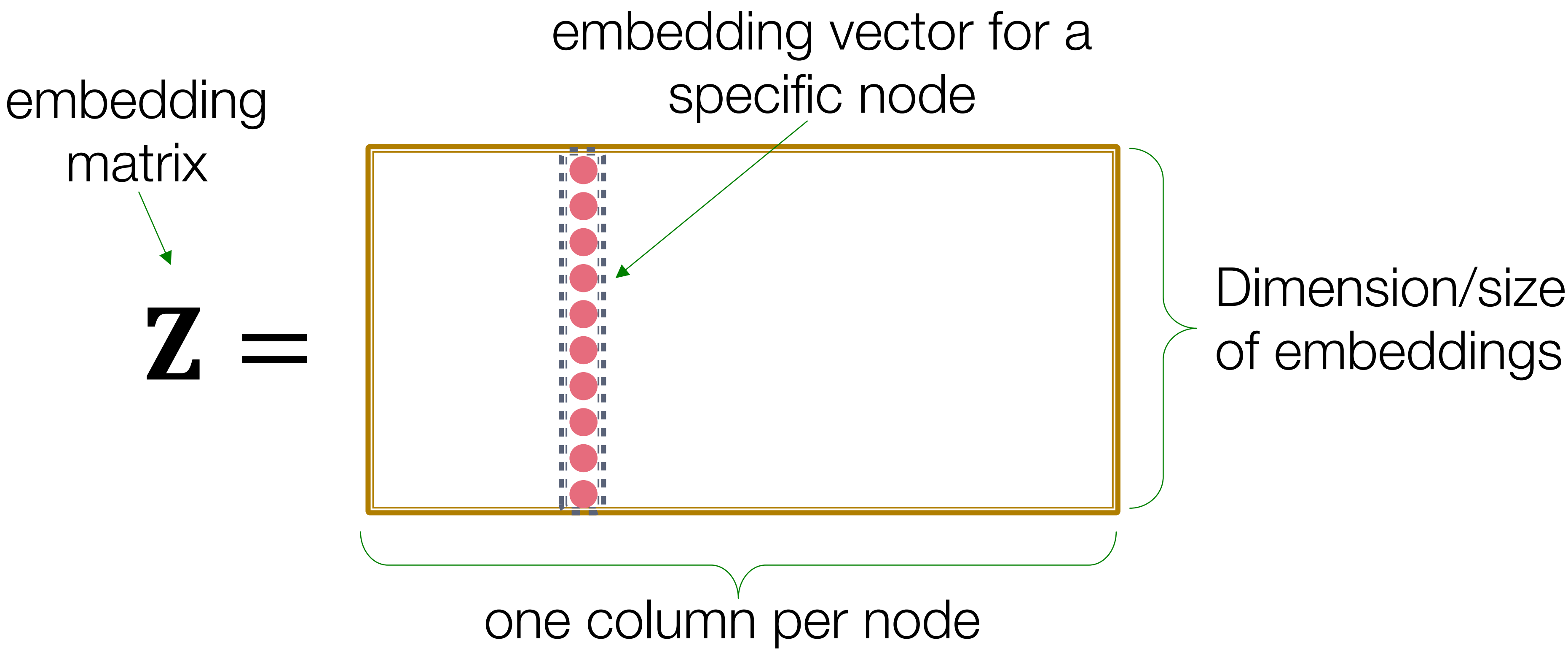
Decoder

Similarity of u and v in the original network

dot product between node embeddings

"Shallow" Encoding

Simplest encoding approach: **encoder is just an embedding-lookup**



KG: shallow encoder + similarity decoder

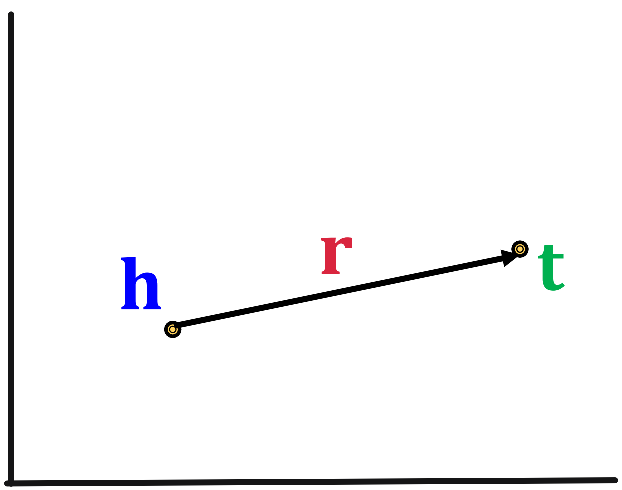
- Edges in KG are represented as **triplets** (h, r, t)
  - head (h) has relation (r) with tail (t)
- **Knowledge Graph embeddings**
  - **Shallow encoder:** Model entities and relations in the embedding/vector space $\mathbb{R}^d$ via embedding lookup
    - Associate entities and relations with **shallow embeddings**
  - **Similarity-based decoder:** Given a true triple (h, r, t), the goal is that the embedding of (h, r) **should be close** to the embedding of t.

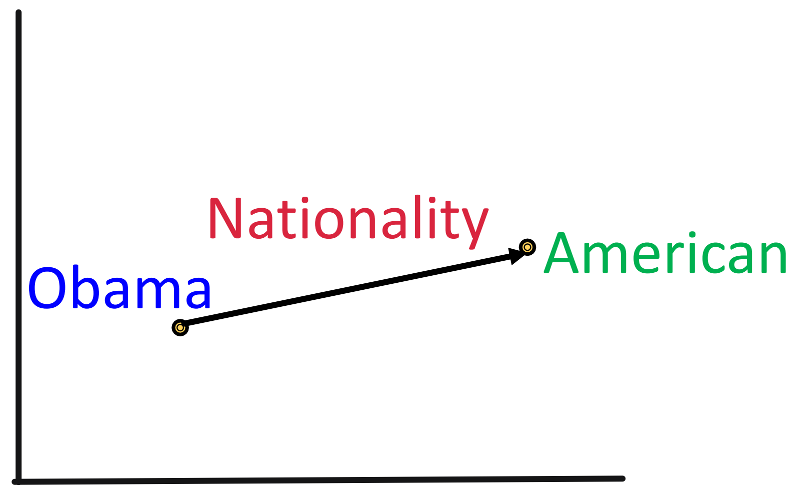
Bordes, et al., Translating embeddings for modeling multi-relational data, NeurIPS 2013.

Example: TransE

- **Translation Intuition**:
  For a triple (h, r, t), $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^d$,
  $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ if the given fact is true
  else $\mathbf{h} + \mathbf{r} \neq \mathbf{t}$

Scoring function: $f_r(h, t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||$





Shallow Encoders: Limitations

- **Limitations** of shallow embedding methods:
  - $O(|V|)$ **parameters are needed**:
    - No sharing of parameters between nodes
    - Every node has its own unique embedding
  - **Inherently "transductive"**:
    - Cannot generate embeddings for nodes that are not seen during training
  - **Do not incorporate node features**:
    - Many graphs have features that we can and should leverage

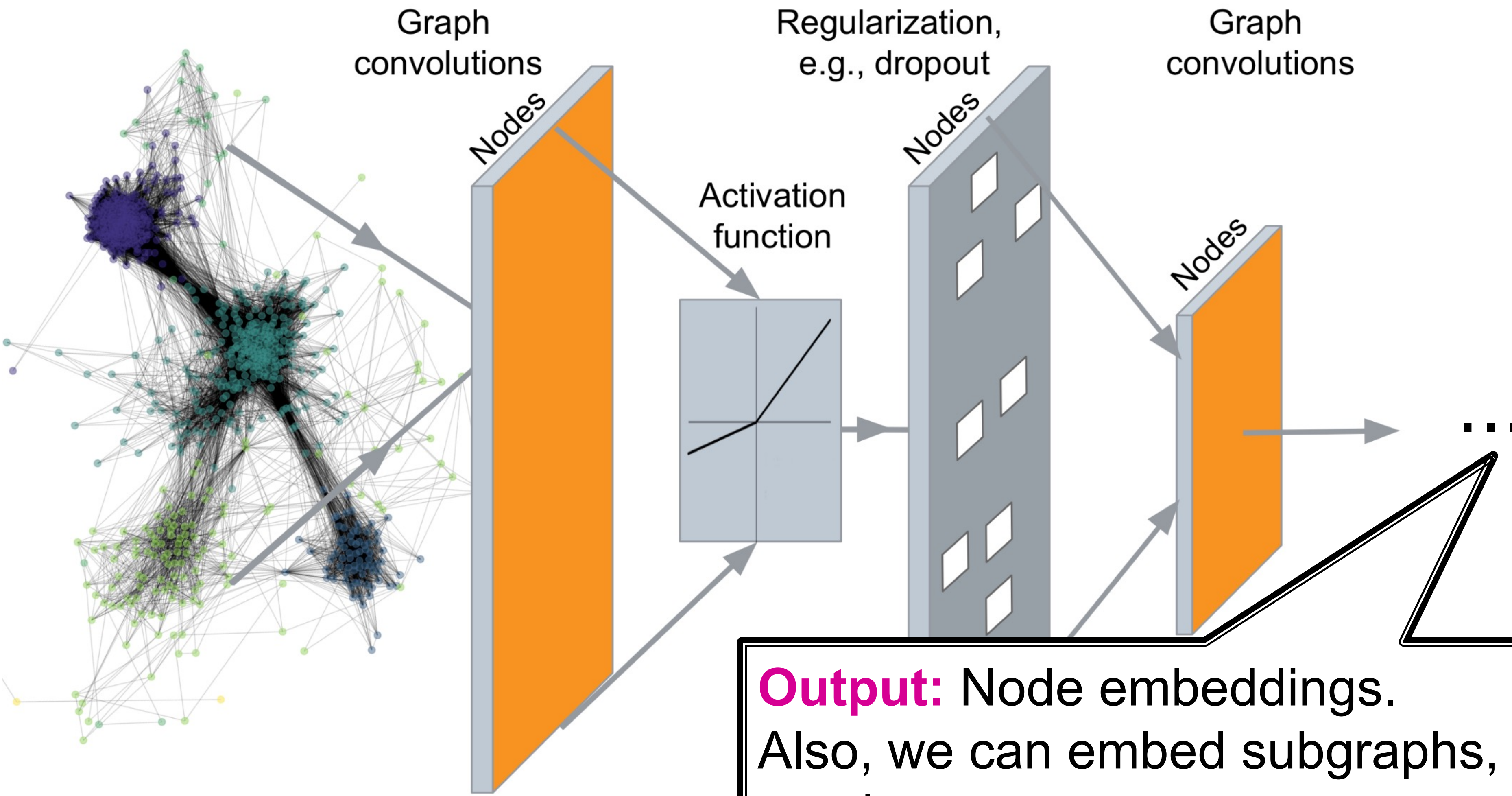
Today: Deep Graph Encoders

- **Today:** We will now discuss deep methods based on **graph neural networks (GNNs):**

$$\mathrm{ENC}(v) = $$ **multiple layers of non-linear transformations based on graph structure**

- **Note:** All these deep encoders can be **combined with node similarity functions** defined in Knowledge Graph models

Deep Graph Encoders



Output: Node embeddings. Also, we can embed subgraphs, graphs

Stanford CS520: Graph Neural Networks

Jiaxuan You, Stanford University

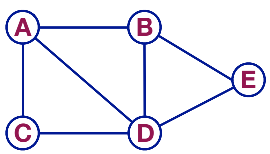
(Slides adapted from CS224W: Machine Learning with Graphs)

Setup

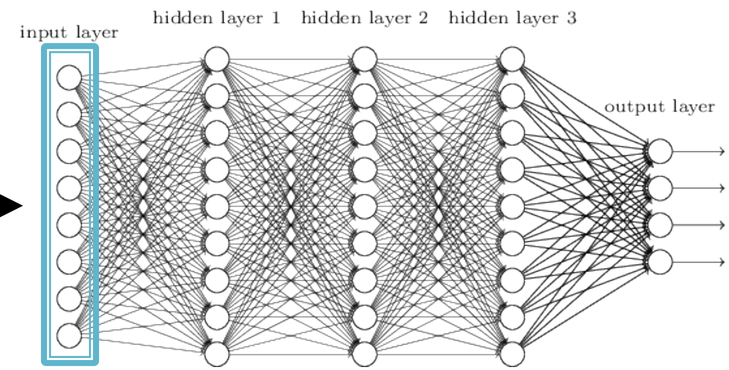
- **Assume we have a graph G:**
  - V is the **vertex set**
  - $\boldsymbol{A}$ is the **adjacency matrix** (assume binary)
  - $\boldsymbol{X} \in \mathbb{R}^{m \times |V|}$ is a matrix of **node features**
  - v: a node in V; $N(v)$: the set of neighbors of v.
  - **Node features:**
    - Social networks: User profile, User image
    - Biological networks: Gene expression profiles, gene functional information
    - When there is no node feature in the graph dataset:
      - Indicator vectors (one-hot encoding of a node)
      - Vector of constant 1: [1, 1, …, 1]

A Naïve Approach

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



$$
\begin{array}{c}
 \\ A \\ B \\ C \\ D \\ E
\end{array}
\begin{array}{c}
\begin{array}{cccccc} A & B & C & D & E & \quad \text{Feat} \end{array} \\
\begin{bmatrix}
0 & 1 & 1 & 1 & 0 & 1 & 0 \\
\boxed{1 \;\; 0 \;\; 0 \;\; 1 \;\; 1 \;\; 0 \;\; 0} \\
1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0
\end{bmatrix}
\end{array}
\longrightarrow
$$

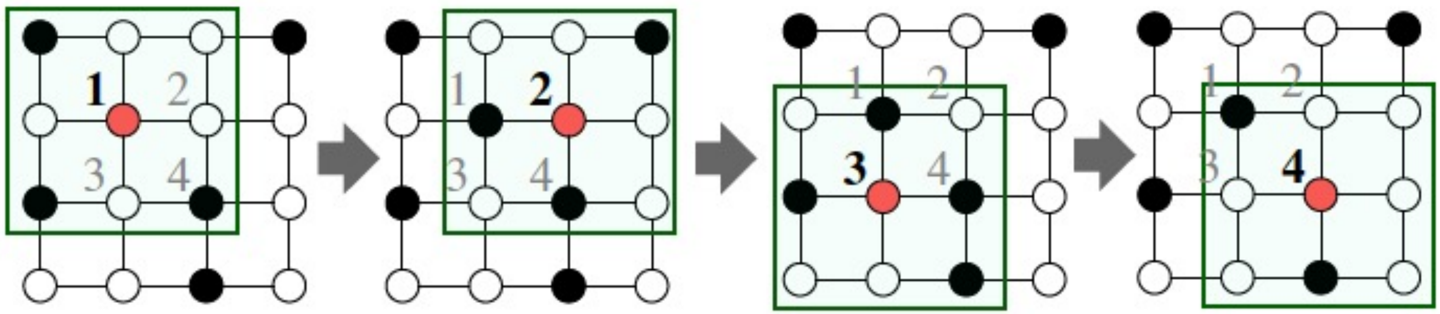


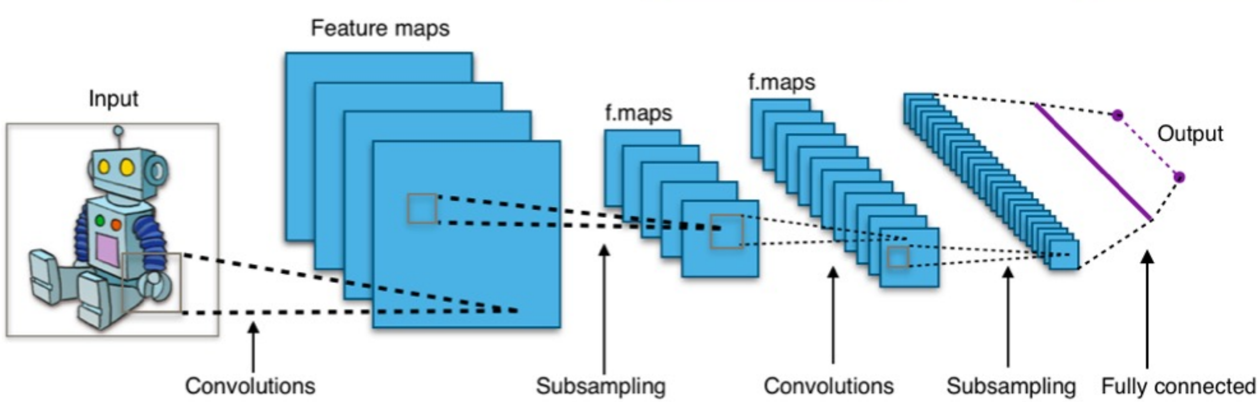
?

- 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:

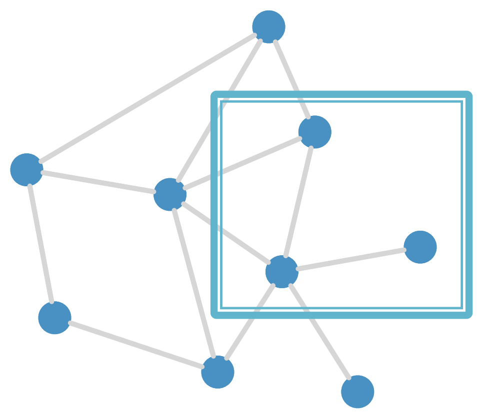




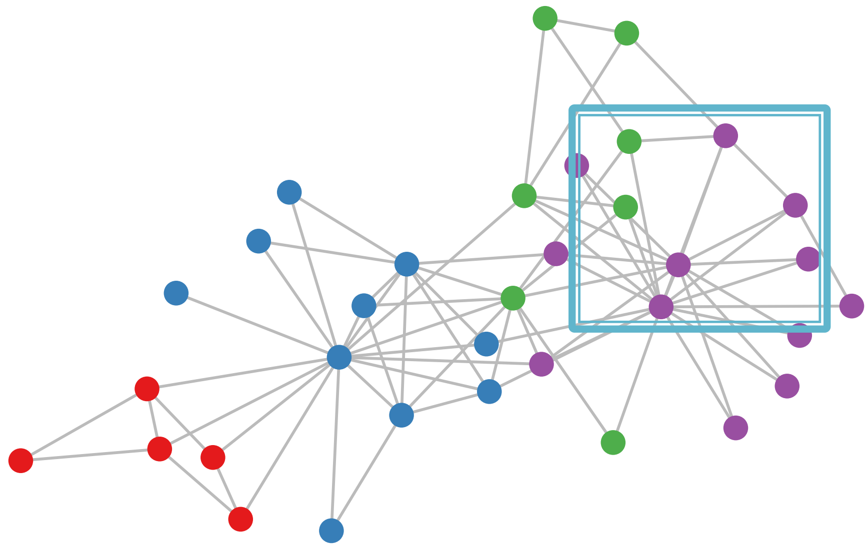
Goal is to generalize convolutions beyond simple lattices
Leverage node features/attributes (e.g., text, images)

Real-World Graphs

But our graphs look like this:



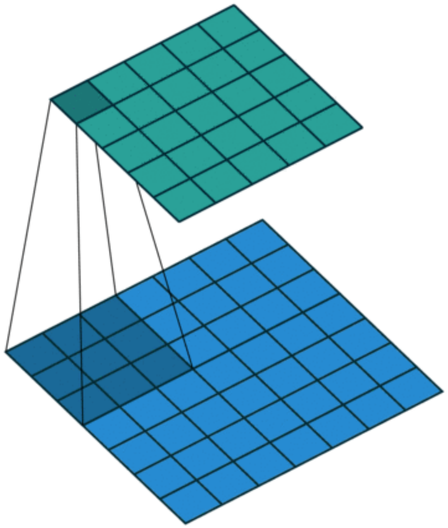
or this:



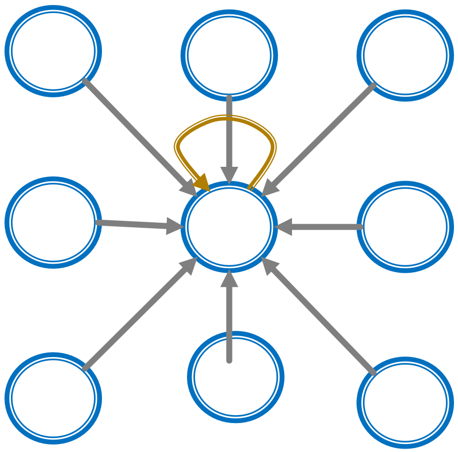
- There is no fixed notion of locality or sliding window on the graph
- Graph is permutation invariant

From Images to Graphs

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



Image



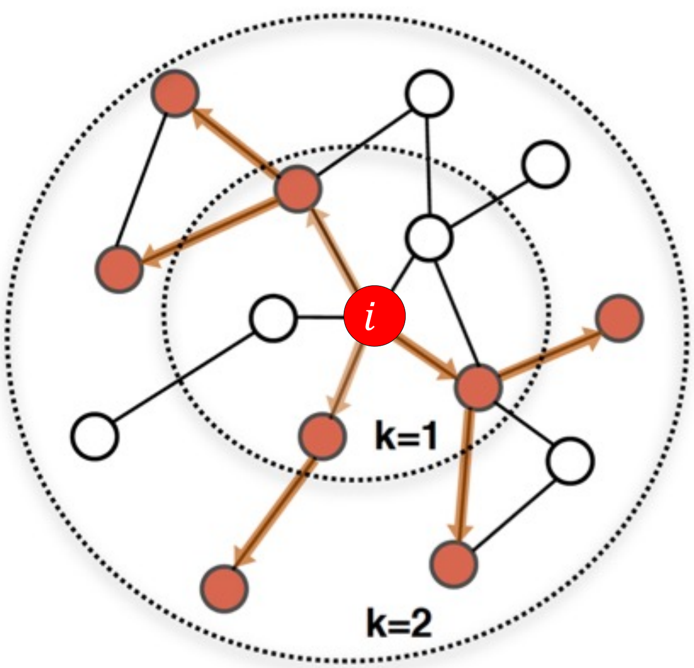
Graph

Idea: transform information at the neighbors and combine it:

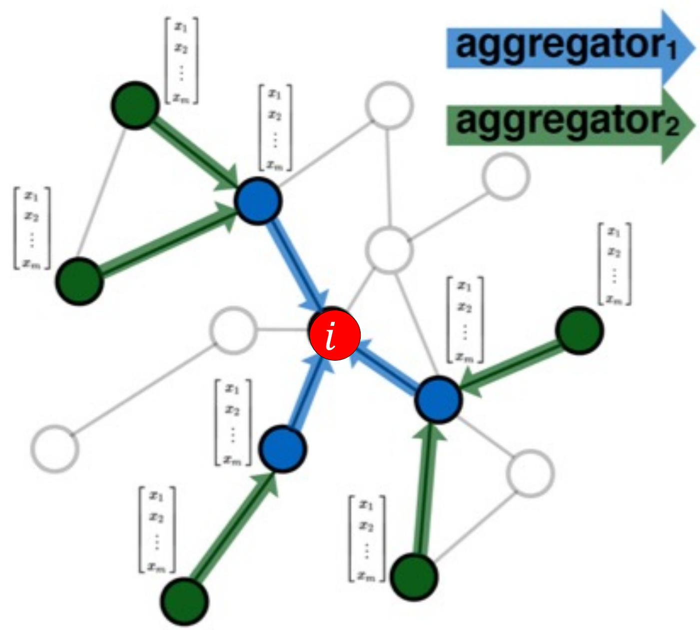
- Transform "messages" h_i from neighbors: $W_i \, h_i$
- Add them up: $\sum_i W_i \, h_i$

[Kipf and Welling, ICLR 2017]

Graph Convolutional Networks



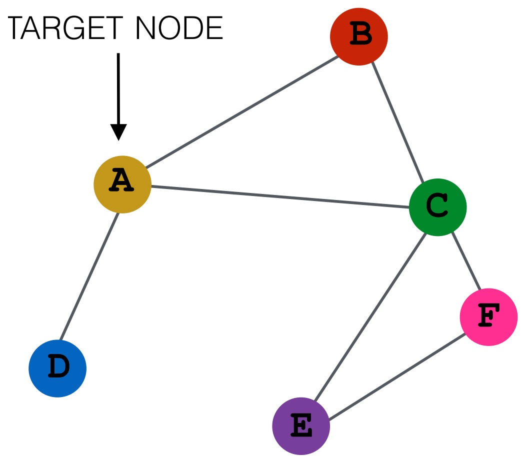
Determine node computation graph



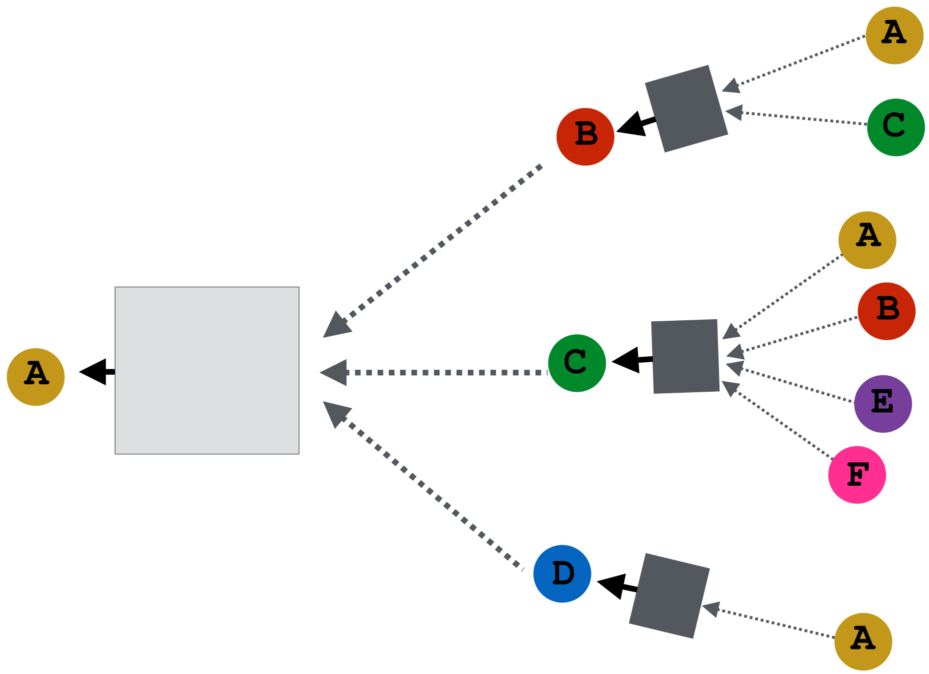
Propagate and transform information

Idea: Aggregate Neighbors

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

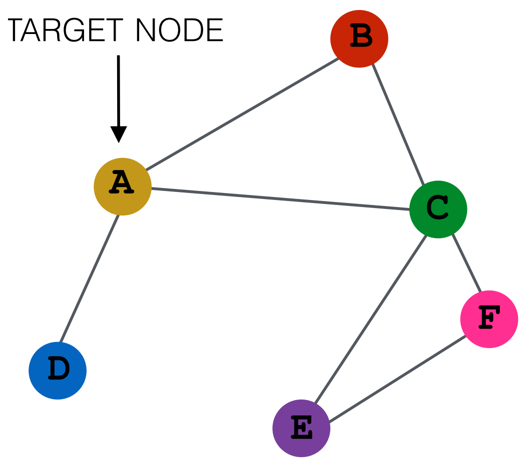


INPUT GRAPH

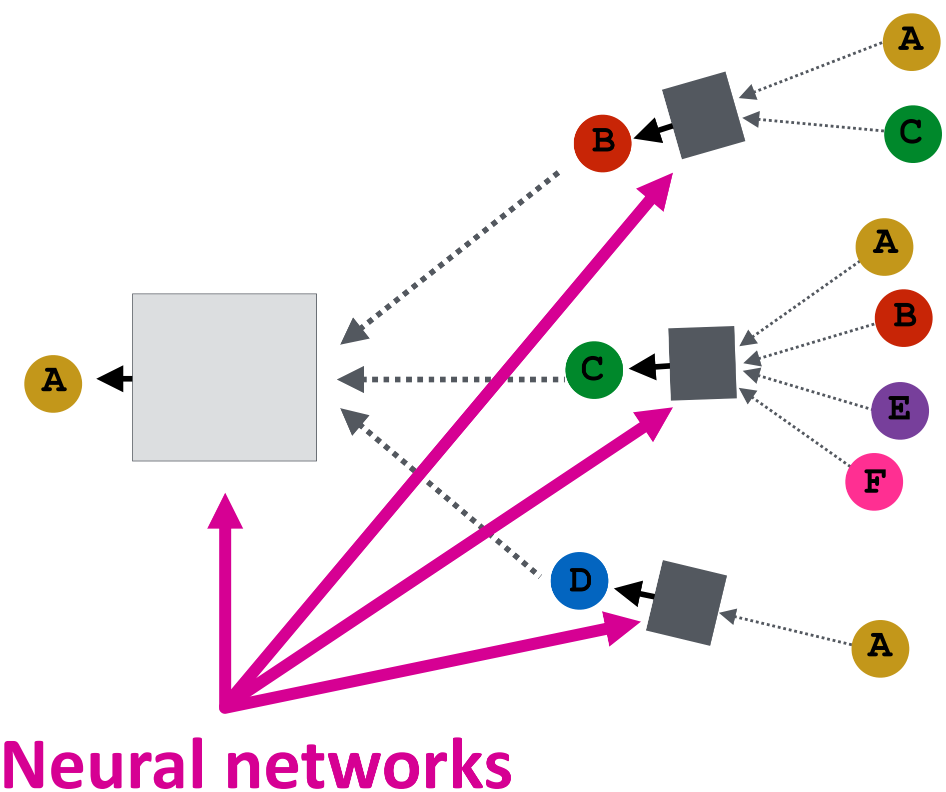


Idea: Aggregate Neighbors

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

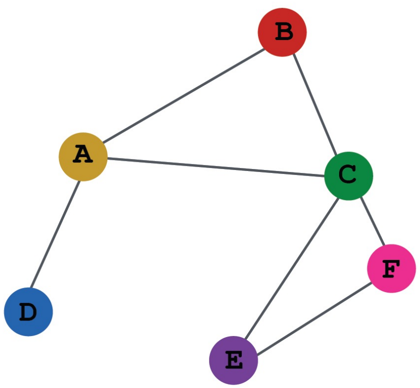


INPUT GRAPH



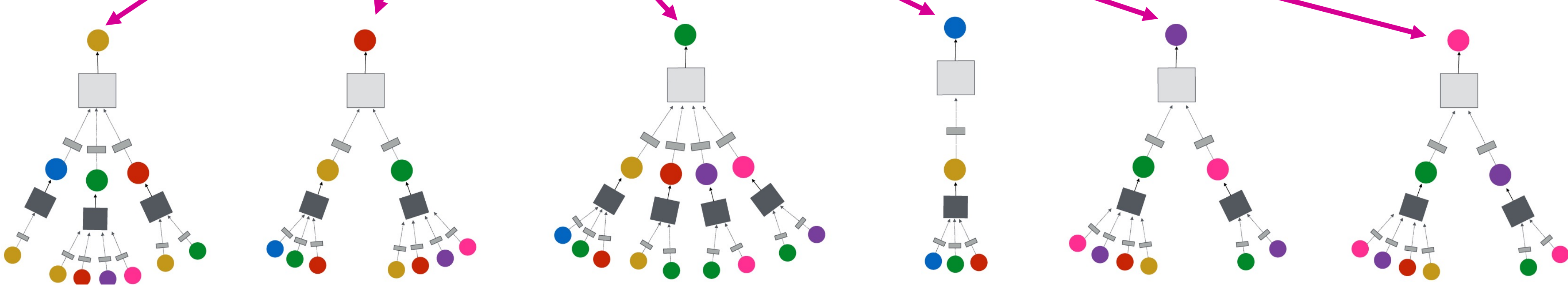
Idea: Aggregate Neighbors

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



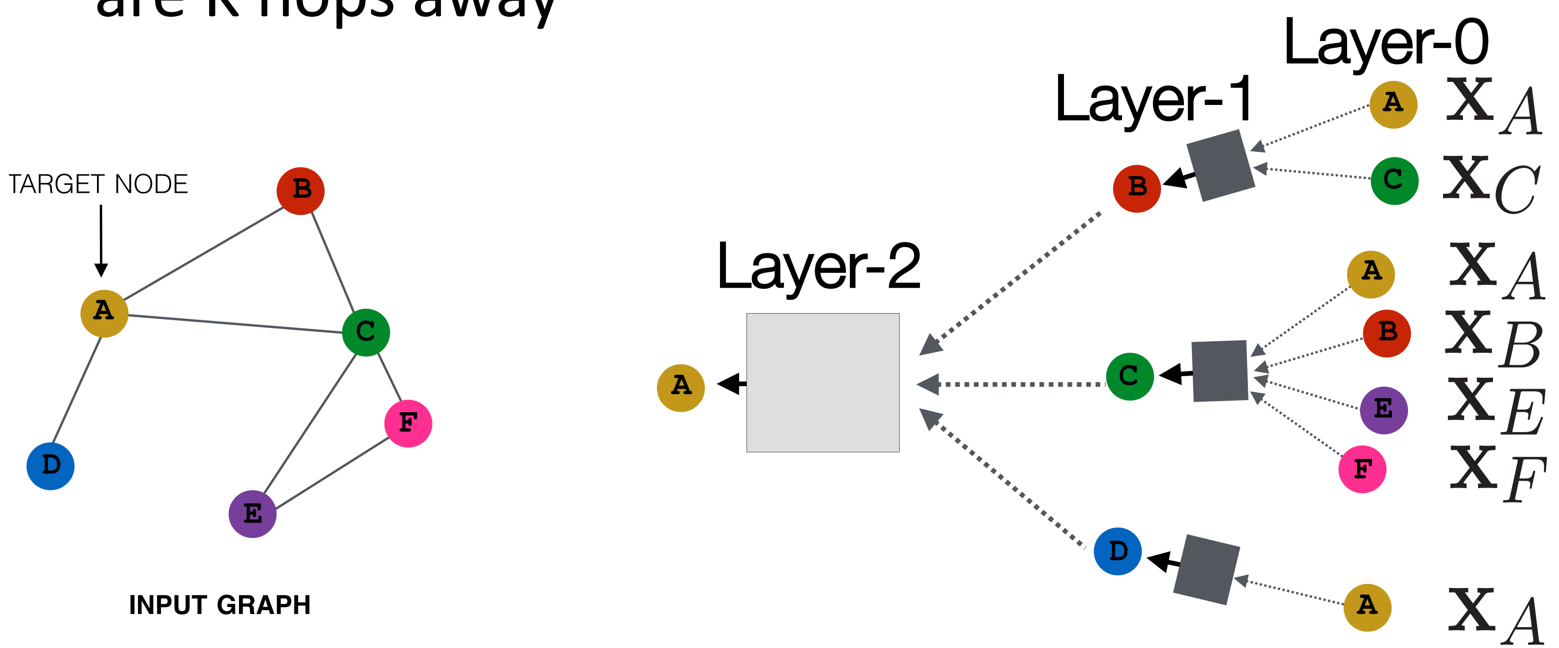
INPUT GRAPH

Every node defines a computation graph based on its neighborhood!



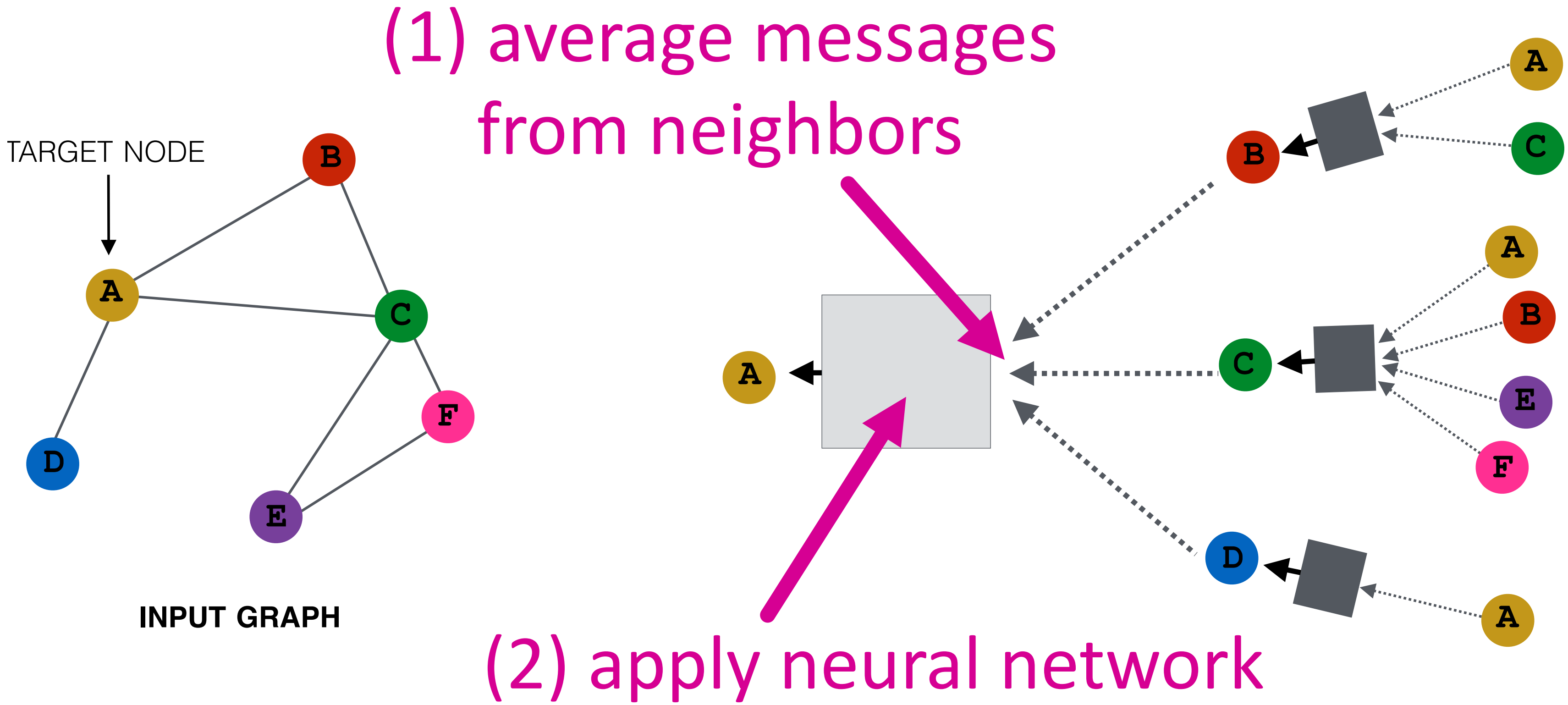
Deep Model: Many Layers

- 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



Neighborhood Aggregation

- **Basic approach:** Average information from neighbors and apply a neural network



The Math: Deep Encoder

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

Initial 0-th layer embeddings are equal to node features

$$\boxed{\mathrm{h}_v^0 = \mathrm{x}_v}$$

embedding of v at layer l

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

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

Average of neighbor's previous layer embeddings

Total number of layers

Embedding after L layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

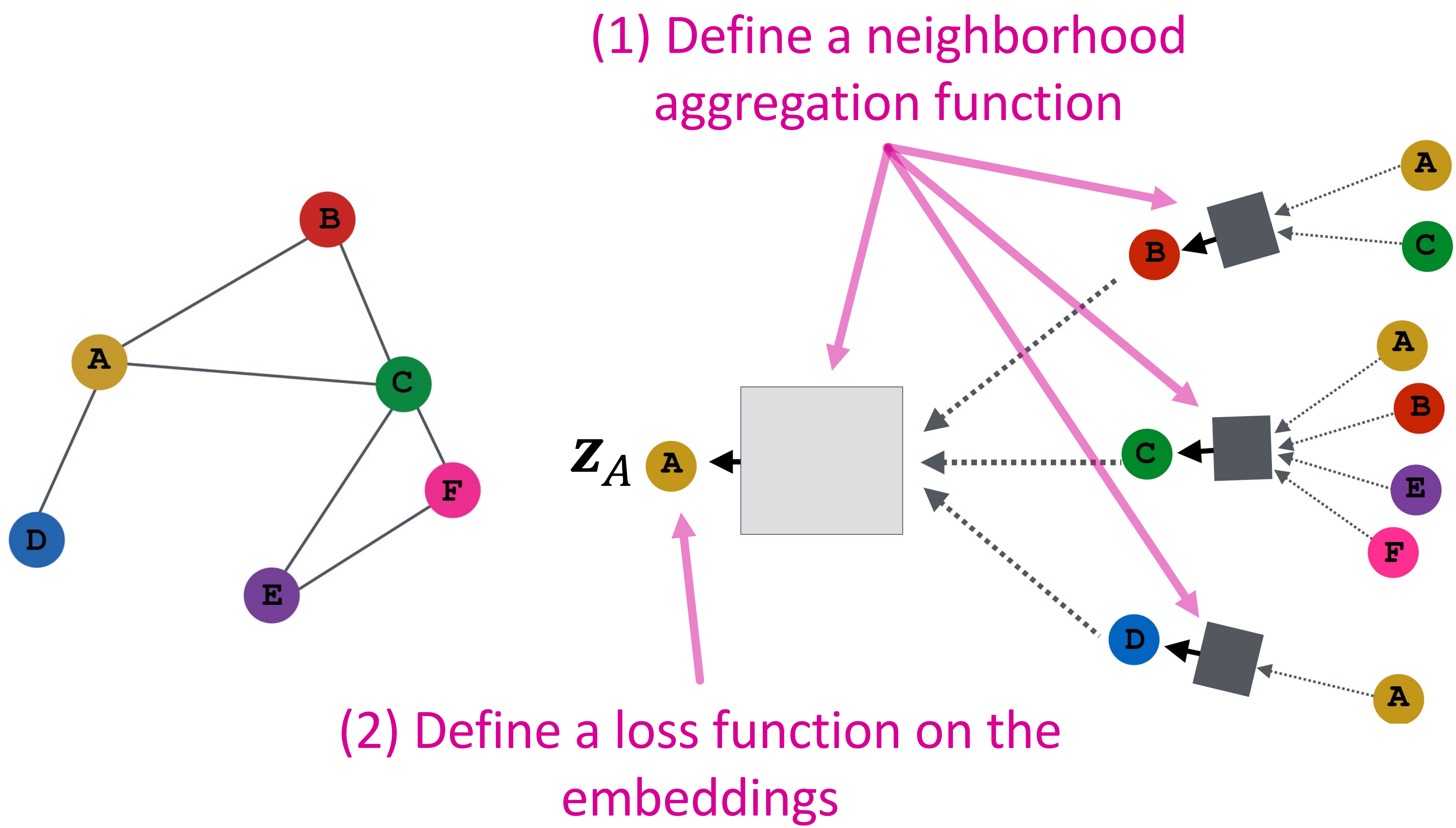
How to train a GNN

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

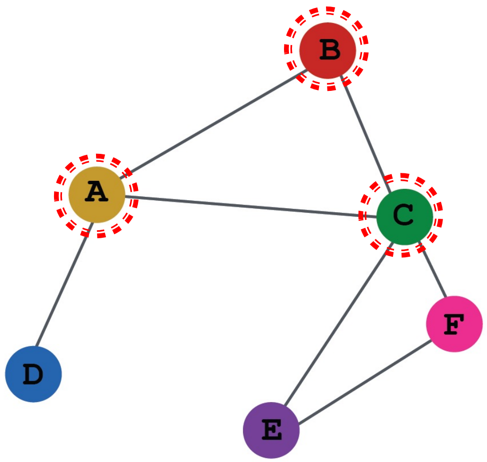
$$\min_{\Theta} \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{z}_v))$$

  - $\boldsymbol{y}$: 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 itself as supervision
    - E.g.: random walks, KG objective (TransE, RotatE, …)

Model Design: Overview

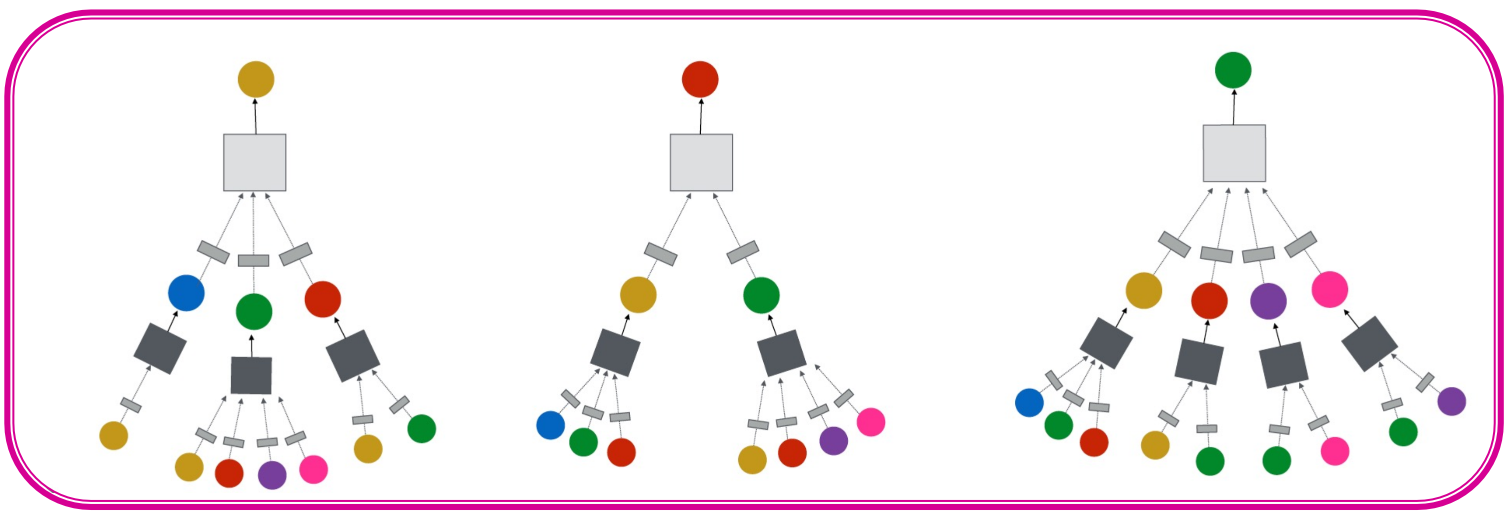


Model Design: Overview

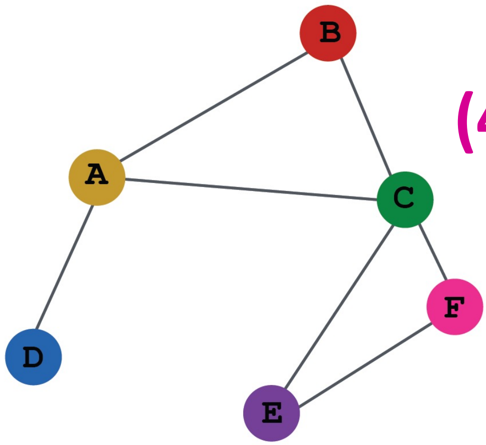


INPUT GRAPH

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



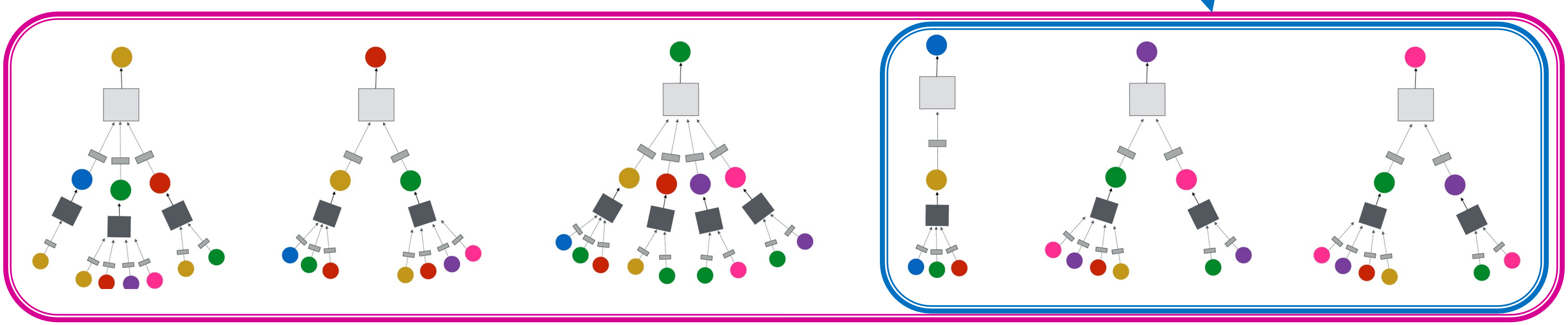
Model Design: Overview



INPUT GRAPH

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

Even for nodes we never trained on!



Stanford CS520: Applications of GNNs

Jiaxuan You, Stanford University

(Slides adapted from CS224W: Machine Learning with Graphs)

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
- Community detection
  - Identify densely linked clusters of nodes
- Graph classification
  - Classify different graphs

Example of
Node-level ML Tasks

Example (1): Protein Folding

A protein chain acquires its native 3D structure

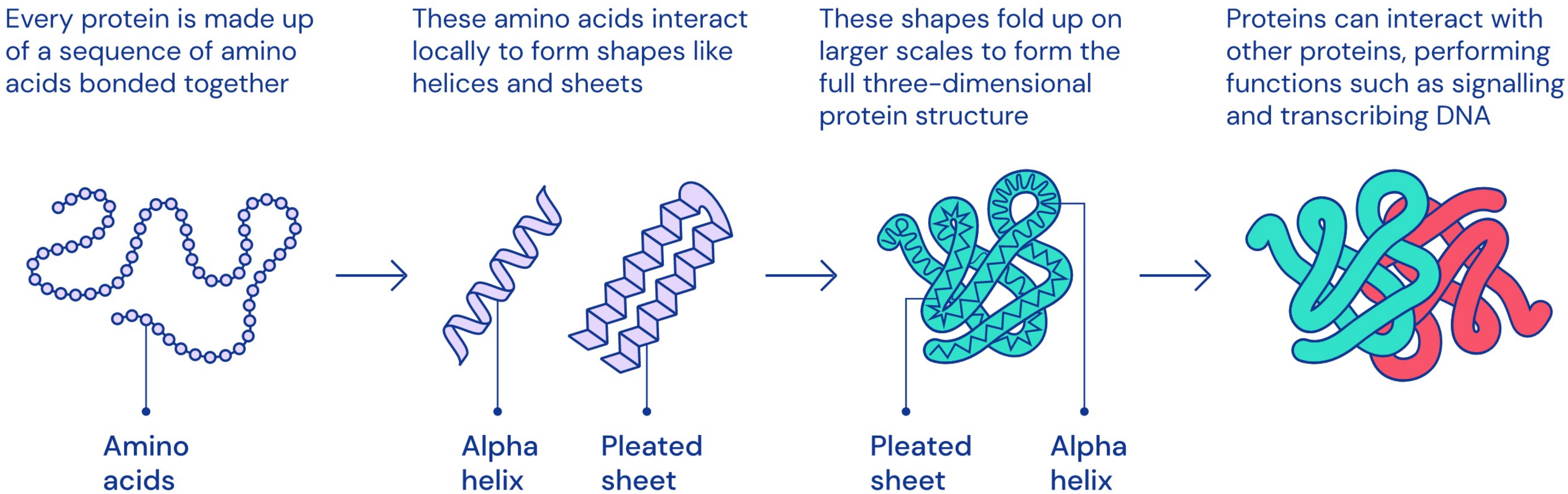


Image credit: [DeepMind]

The Protein Folding Problem

Computationally predict a protein's 3D structure based solely on its amino acid sequence

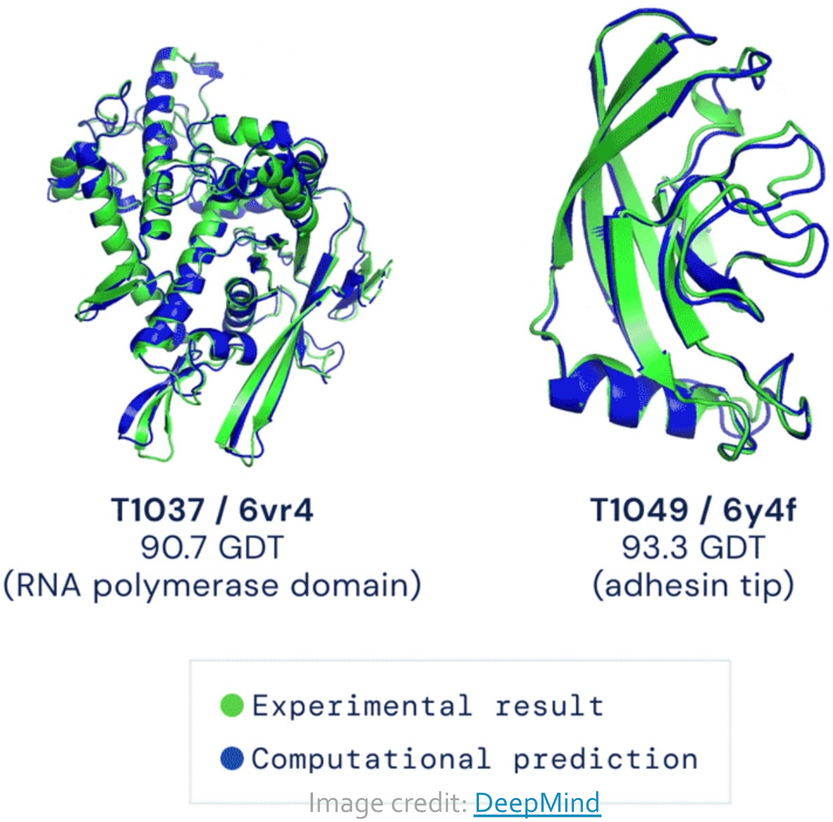


Image credit: DeepMind

AlphaFold: Solving Protein Folding

- **Key idea:** "Spatial graph"
  - **Nodes:** Amino acids in a protein sequence
  - **Edges:** Proximity between amino acids in 3D

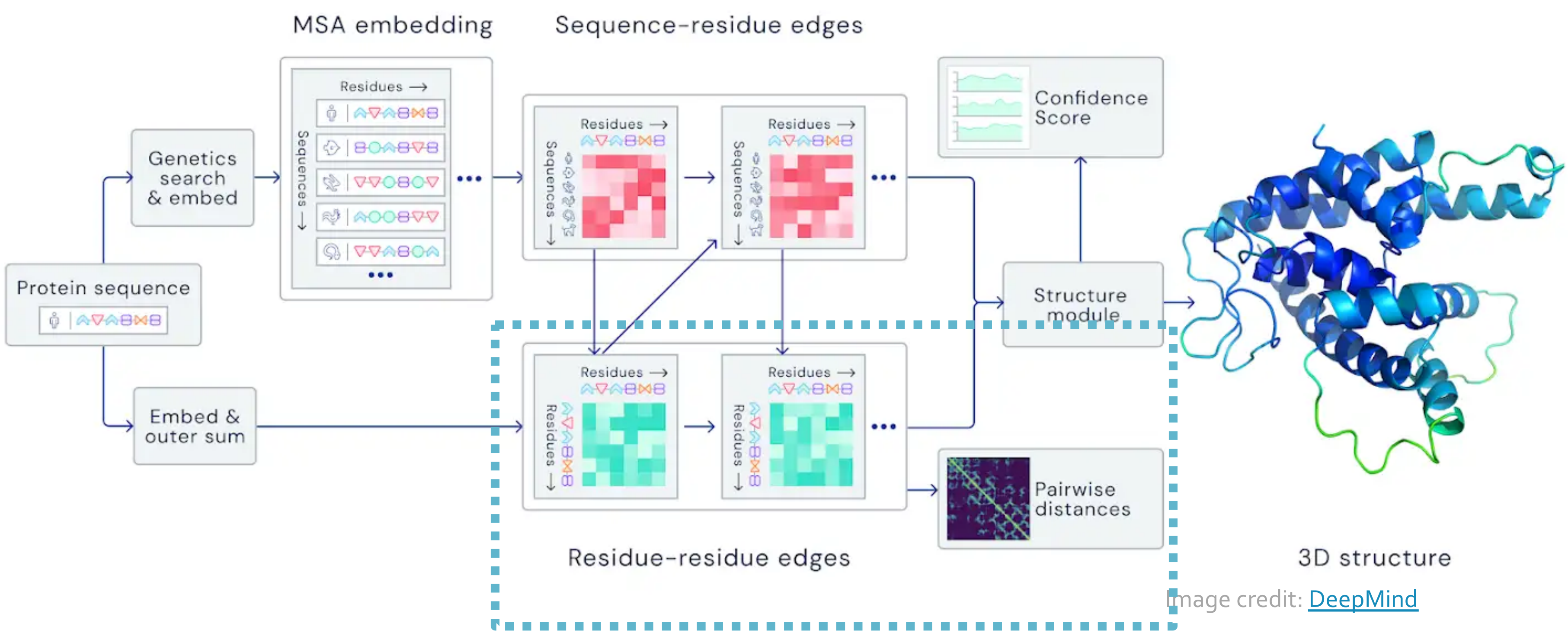


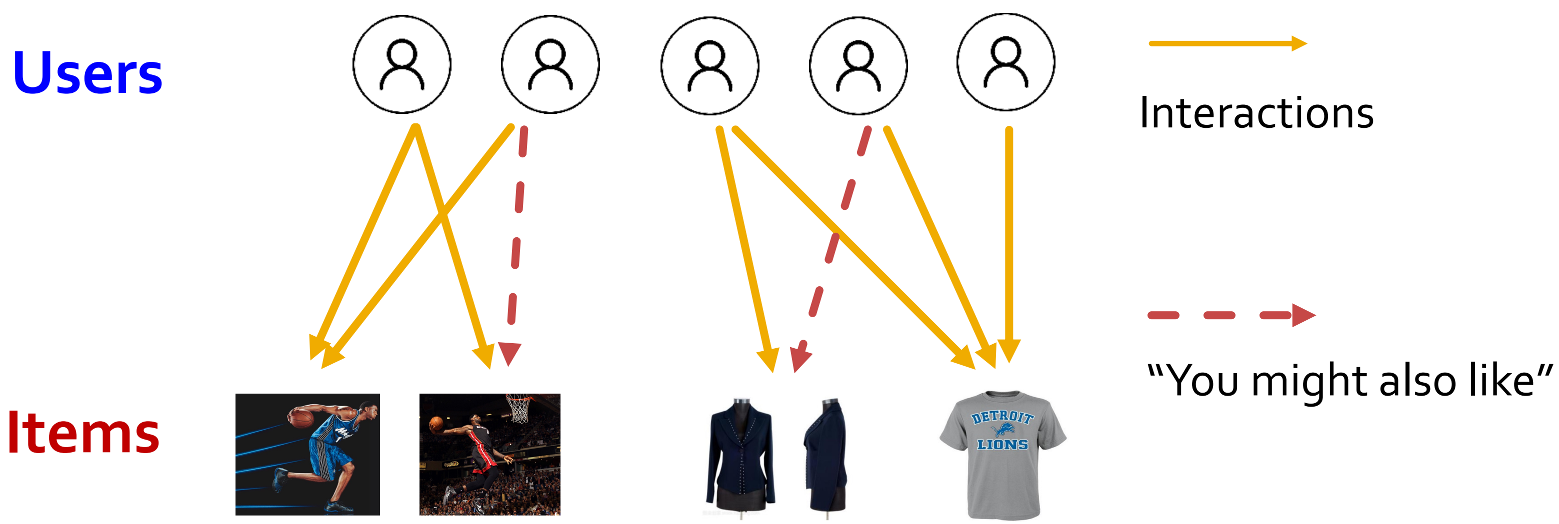
Image credit: DeepMind

Spatial graph

Examples of Edge-level ML Tasks

Example (2): Recommender Systems

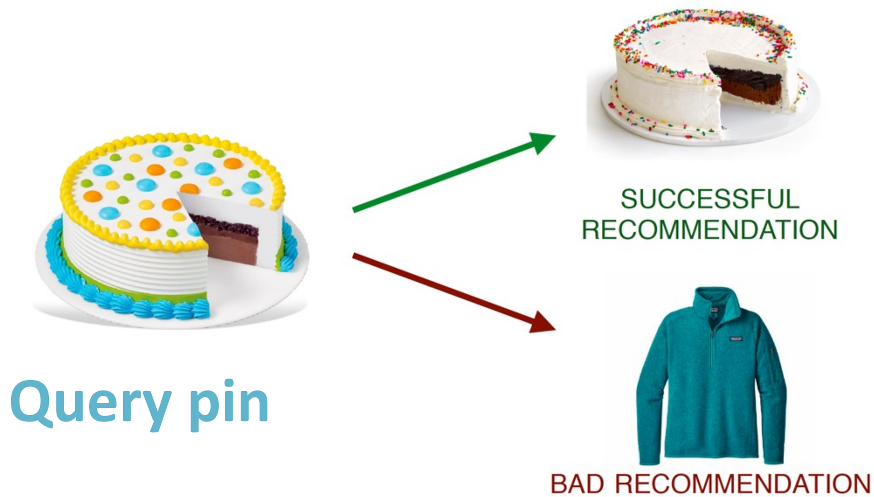
- **Users interacts with items**
  - Watch movies, buy merchandise, listen to music
  - **Nodes:** Users and items
  - **Edges:** User-item interactions
- **Goal: Recommend items users might like**



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

PinSage: Graph-based Recommender

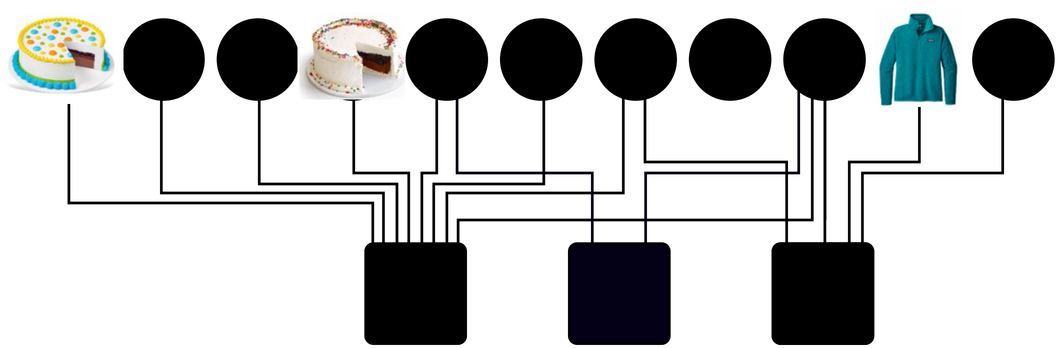
Task: Recommend related pins to users

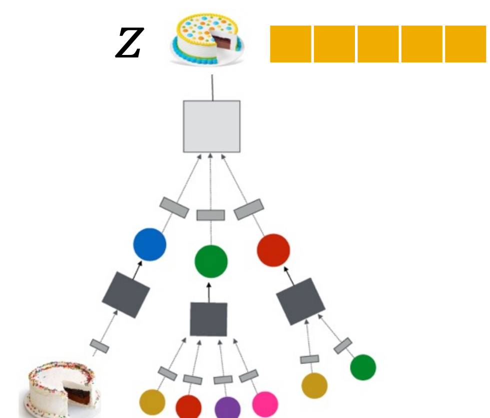


Task: Learn node embeddings z_i such that

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

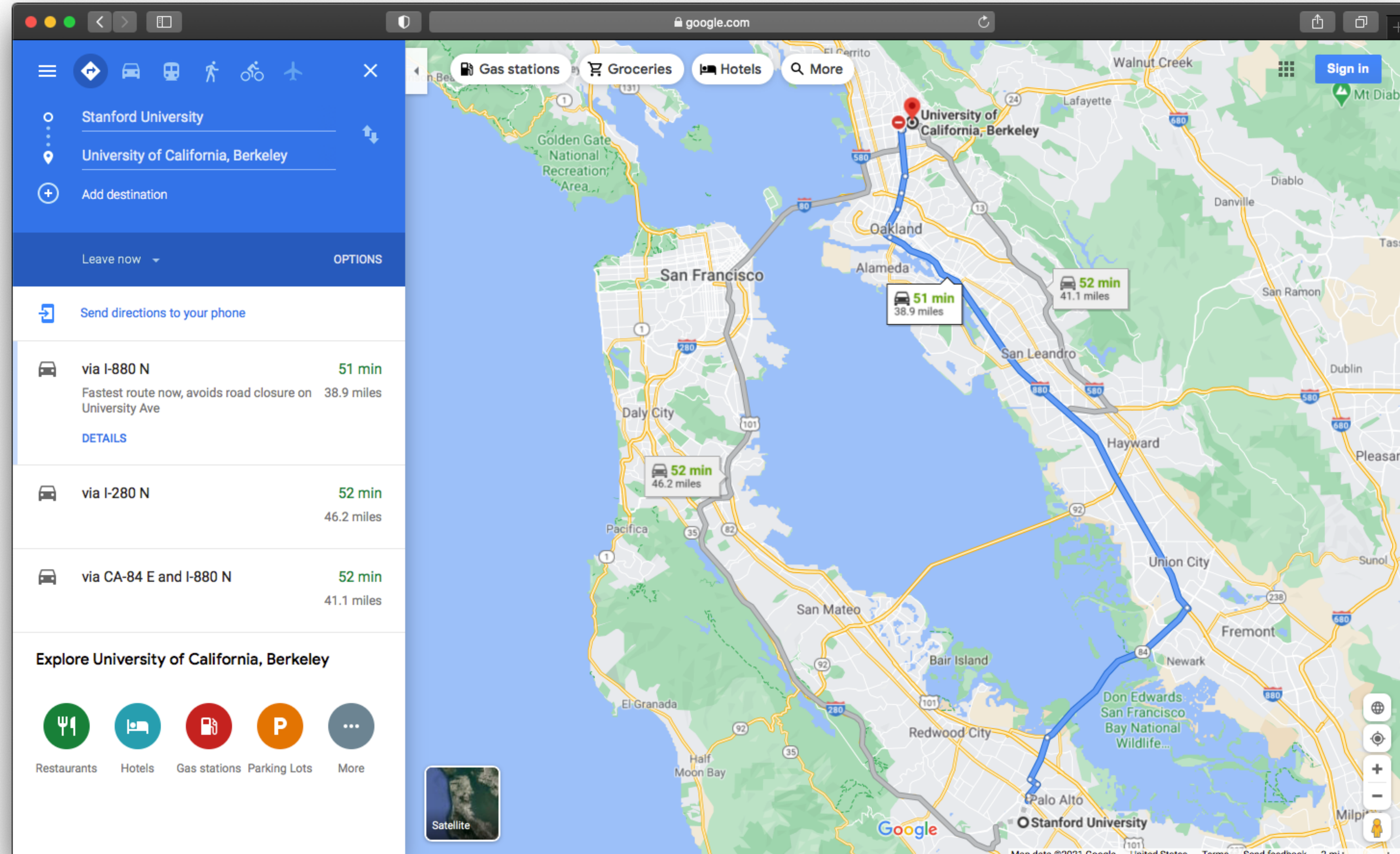
Predict whether two nodes in a graph are related





Examples of Subgraph-level ML Tasks

Example (3): Traffic Prediction



Road Network as a Graph

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

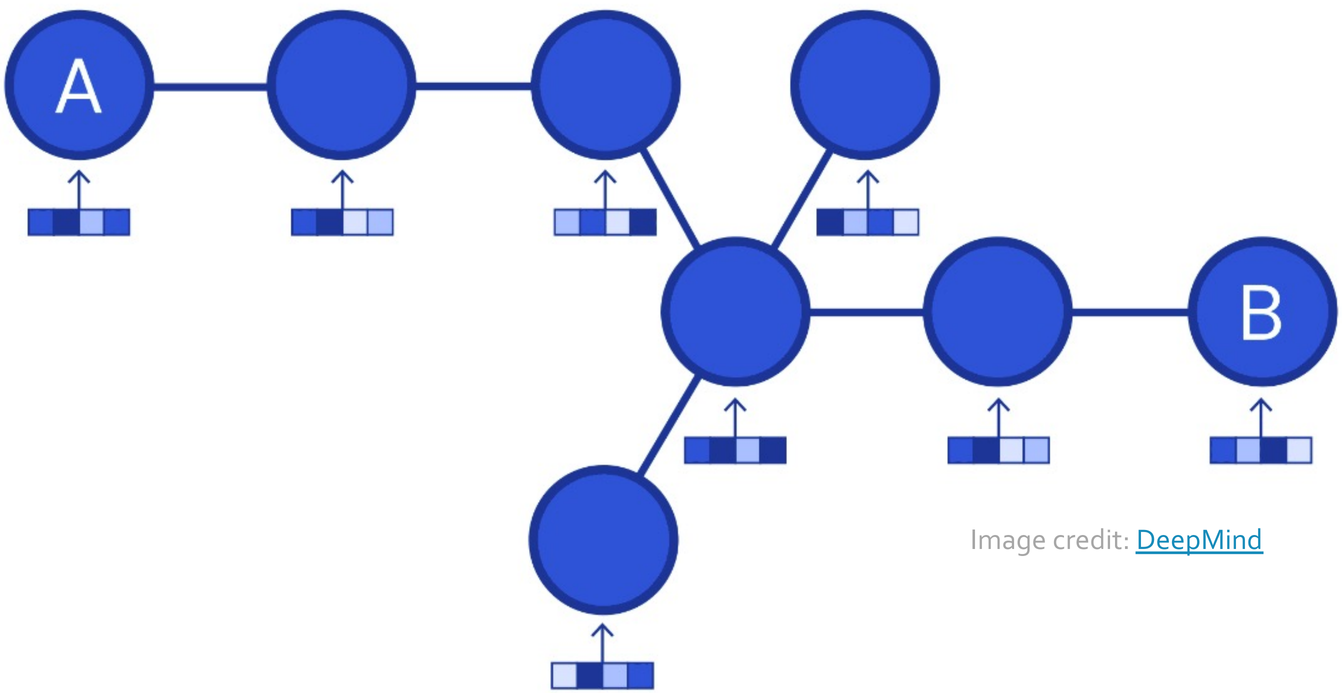
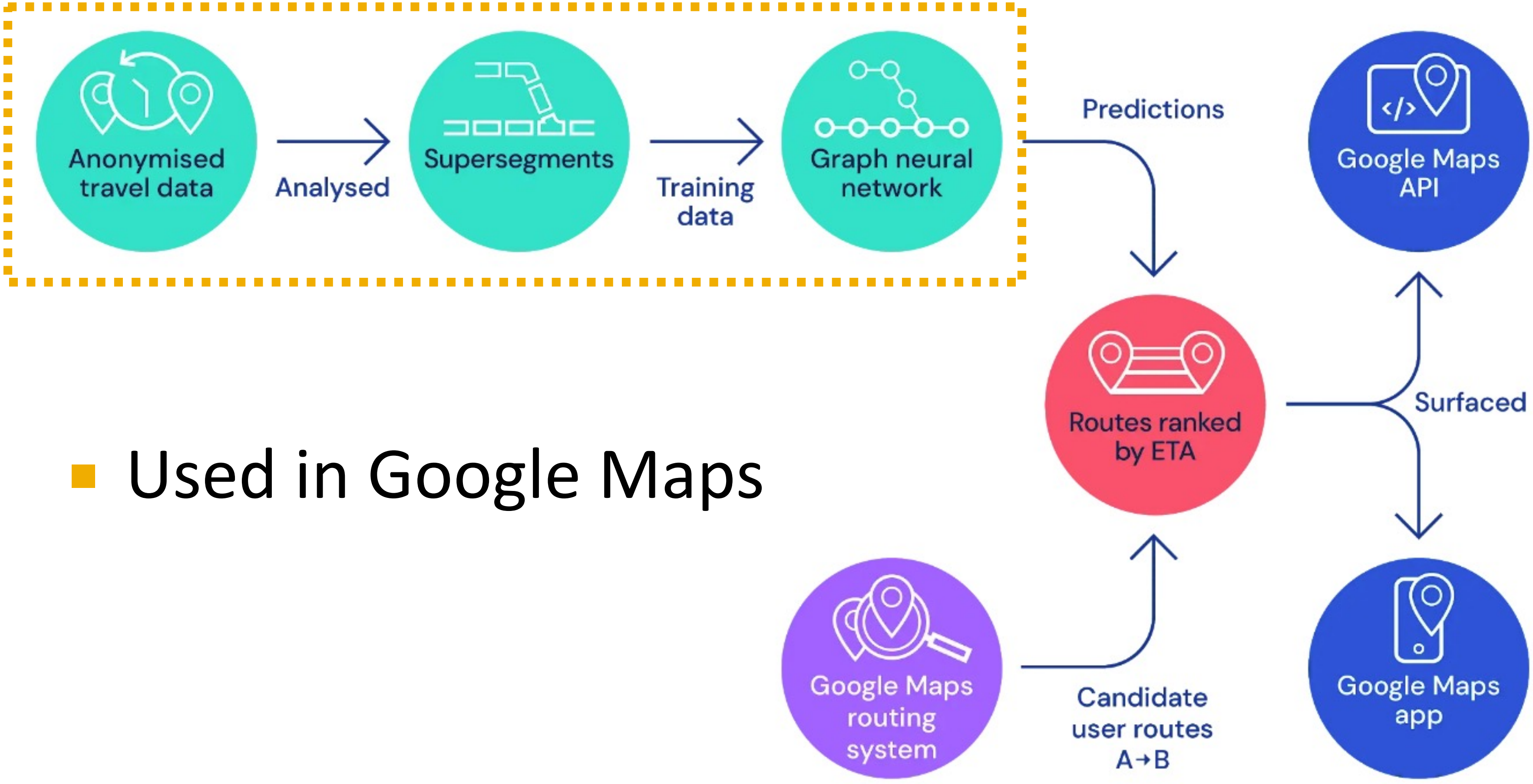


Image credit: DeepMind

Traffic Prediction via GNN

Predict via Graph Neural Networks



- Used in Google Maps

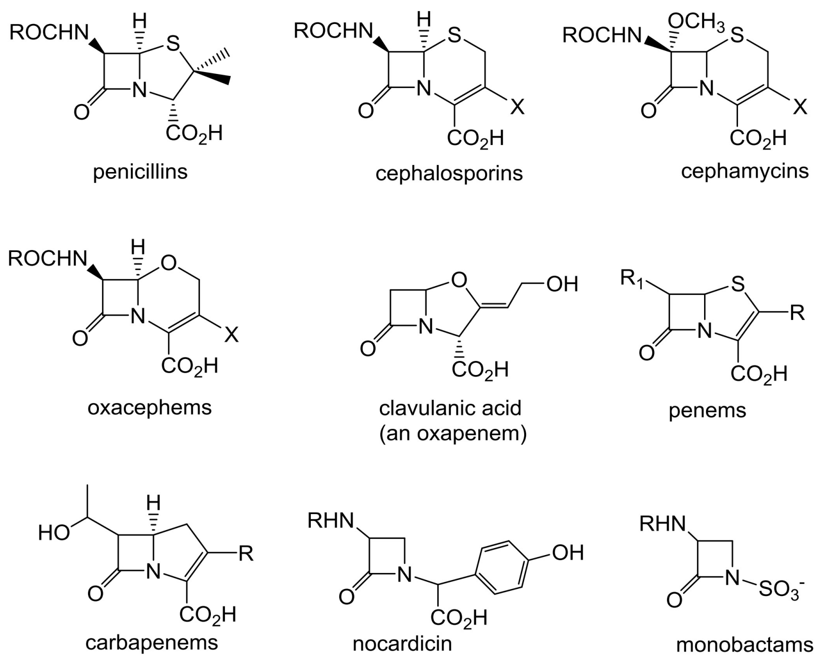
THE MODEL ARCHITECTURE FOR DETERMINING OPTIMAL ROUTES AND THEIR TRAVEL TIME.

Image credit: DeepMind

Examples of Graph-level ML Tasks

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.

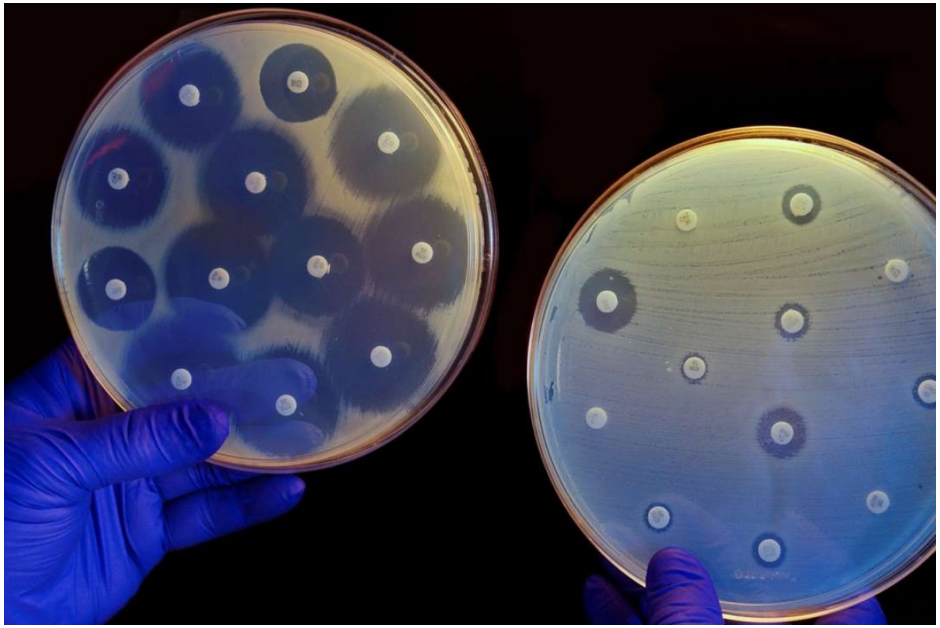
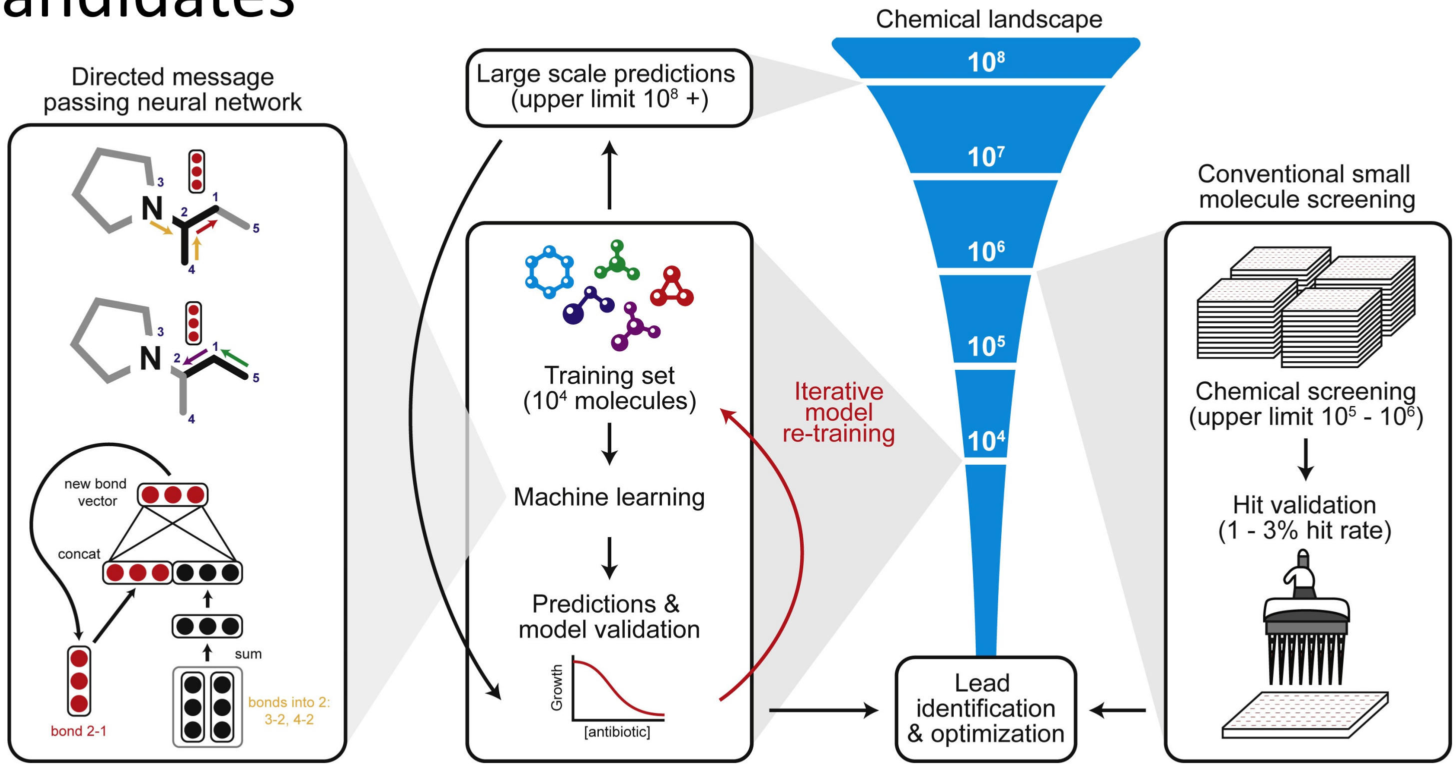


Image credit: [CNN]

Stokes et al., A Deep Learning Approach to Antibiotic Discovery, Cell 2020

Deep Learning for Antibiotic Discovery

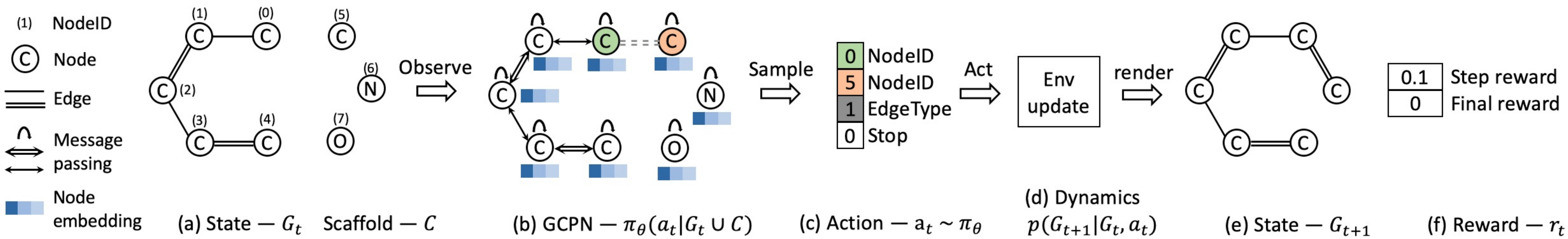
- A **graph classification task**
- Predict promising molecules from a pool of existing candidates



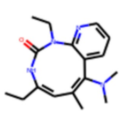
Stokes, Jonathan M., et al. "A deep learning approach to antibiotic discovery." Cell 180.4 (2020): 688-702.

Molecule Generation / Optimization

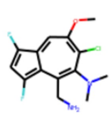
Graph generation: Generating novel molecules



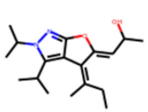
Use case 1: Generate novel molecules with high drug likeness



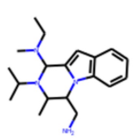
0.948



0.945

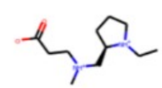


0.944

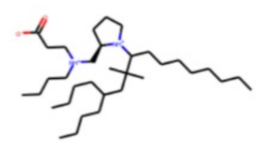


0.941

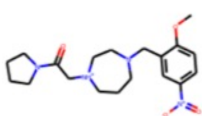
Use case 2: Optimize existing molecules to have desirable properties



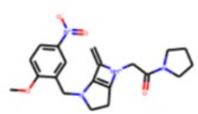
-8.32



-0.71



-5.55



-1.78

Summary

- Motivations for GNNs
  - Expressive and scalable
- What is a GNN
  - **Key:** A node neighborhood aggregation function
  - Define losses and training procedure
- Applications of GNNs
  - **Different levels:** Node, edge, subgraph, graph

- **More materials:**
  - Stanford CS224W
    - Course website: http://web.stanford.edu/class/cs224w/