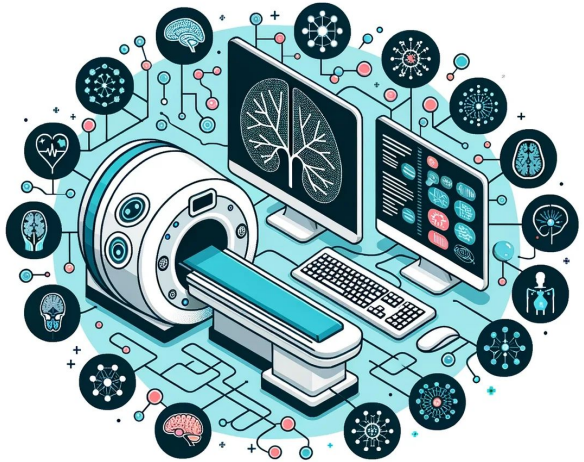
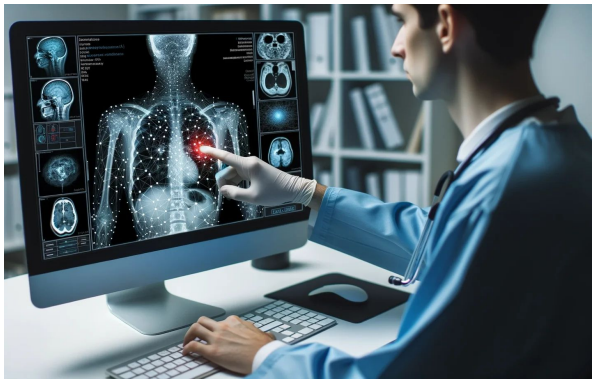
Applied Deep Learning and Generative Models in Healthcare



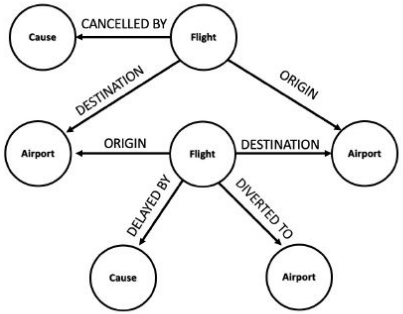


Session 3: Graph neural networks
Date: Jan 25 2025

Instructor: Mahmoud E. Khani, Ph.D.

Many types of data are graphs

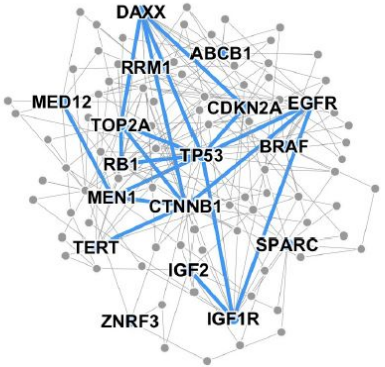
- Graphs are a general language for describing and analyzing entities with relations/interactions



Event Graphs

Image credit: SalientNetworks

Computer Networks



Disease Pathways

Many types of data are graphs

- Graphs are a general language for describing and analyzing entities with relations/interactions

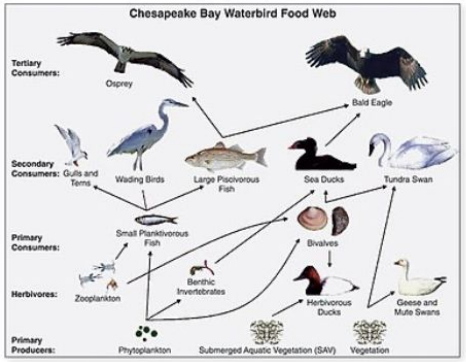


Image credit: Wikipedia

Food Webs

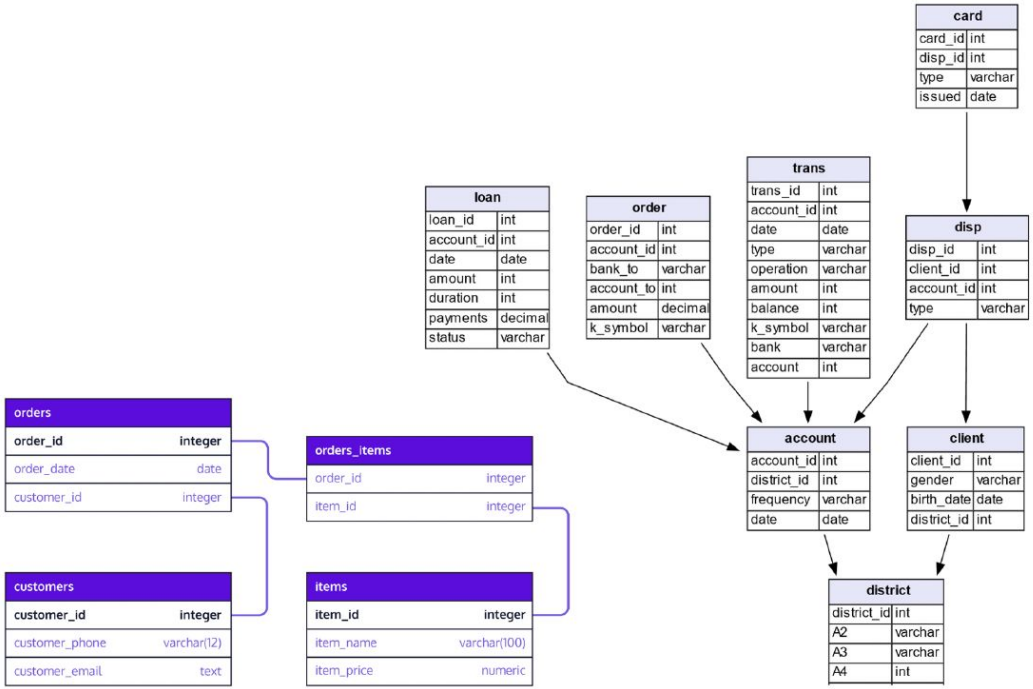
Image credit: Pinterest

Particle Networks

Image credit: visitlondon.com

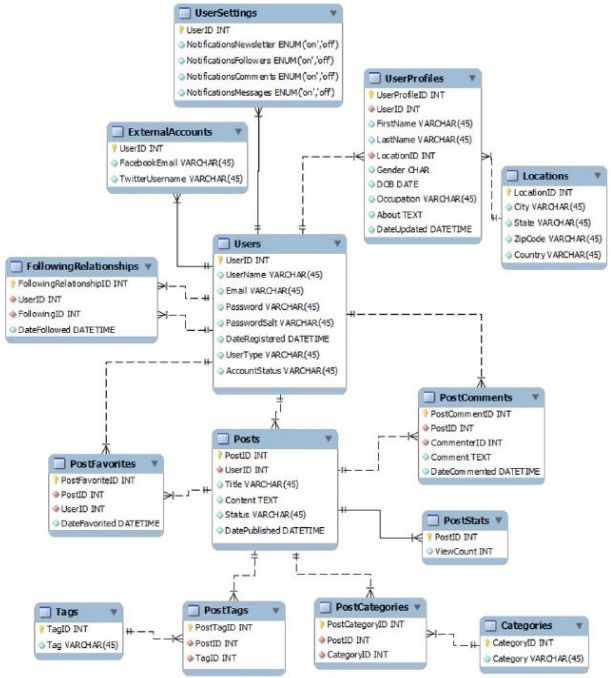
Underground Networks

Databases are Graphs!



Commerce

Finance



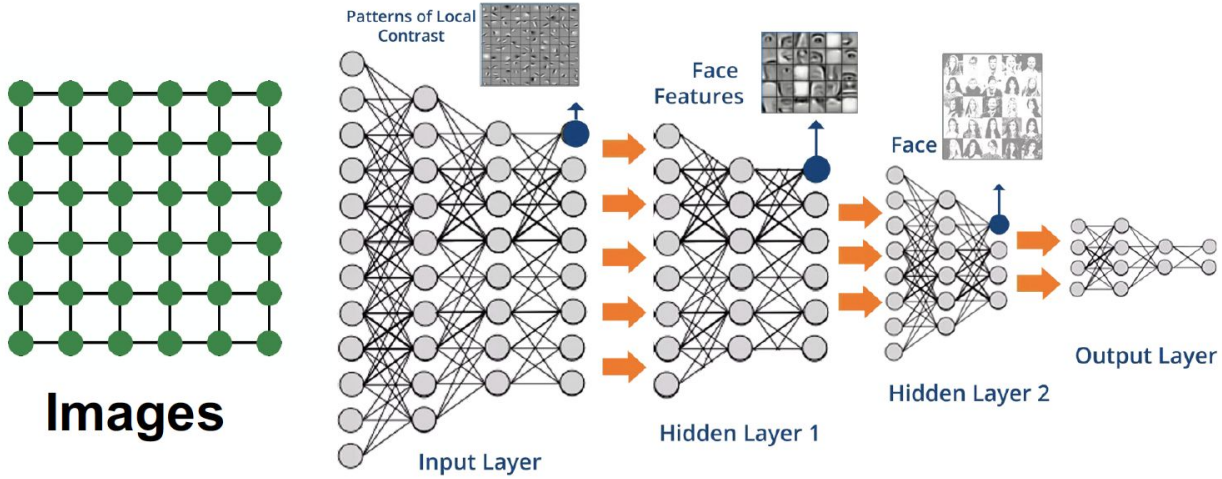
Social Media

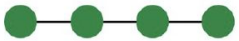
Graphs: Machine Learning

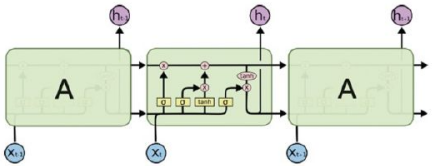
- Complex domains have a rich relational structure, which can be represented as a relational graph!
- How do we take advantage of relational structure for better prediction?

Modern ML Toolbox

- Modern deep learning toolbox is designed for simple sequences & grids!



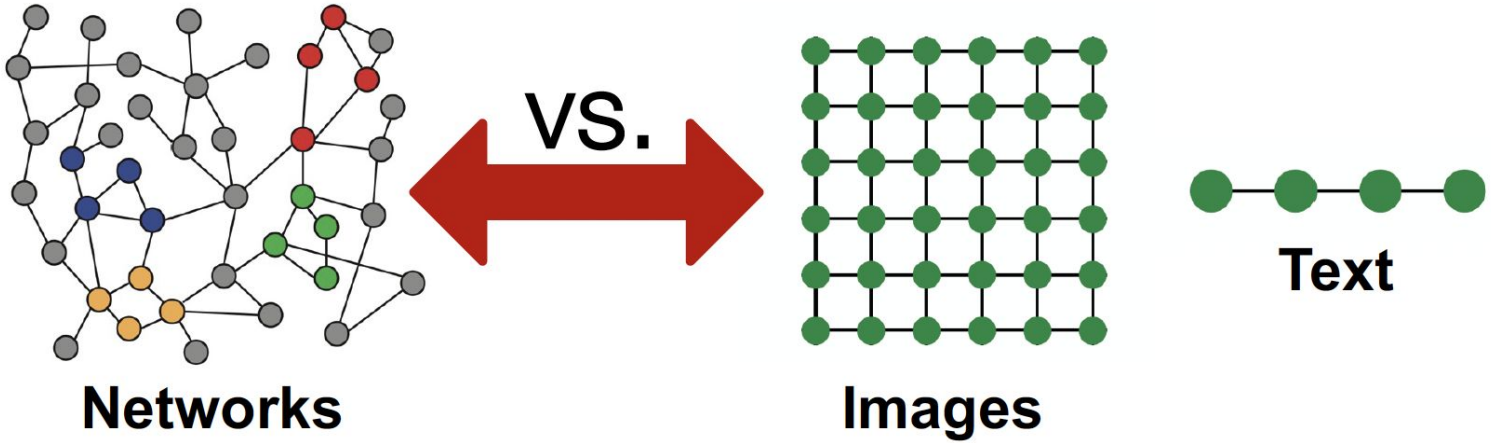




Why is Graph Deep Learning Hard?

Networks are complex

- Arbitrary size & complex topological structure (i.e., no spatial locality like grids)
- No fixed node ordering or reference point



Images as graphs

- Images as rectangular grids with image channels (244x244x3 floats)
- images as graphs with regular structure, where each pixel represents a node connected by edges.

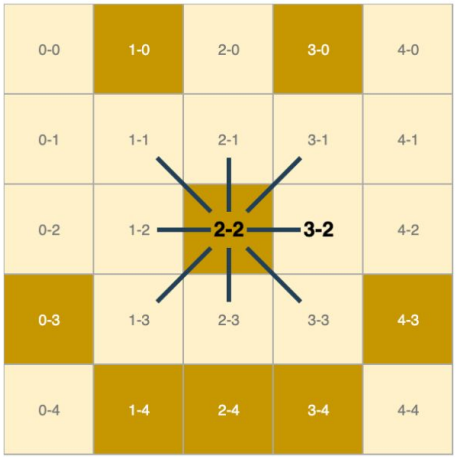
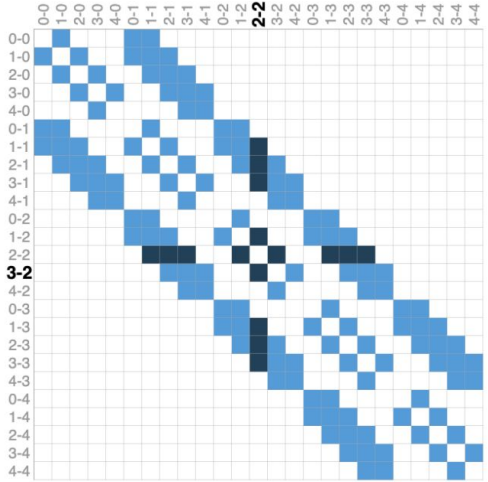
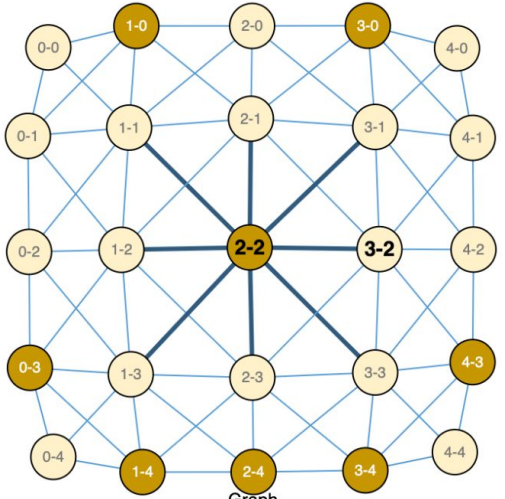


Image Pixels


Adjacency Matrix

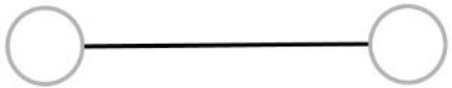


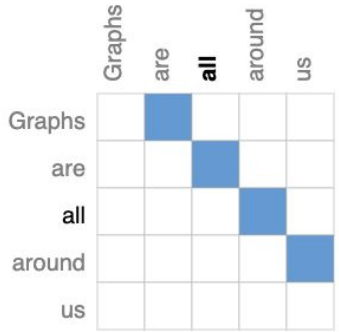
Graph

Text as graphs

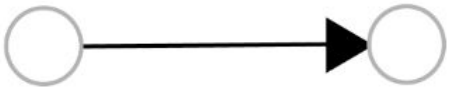
- We can digitize text by associating indices to each character, word, or token, and representing text as a sequence of these indices.
- This creates a directed graph!

Undirected edge

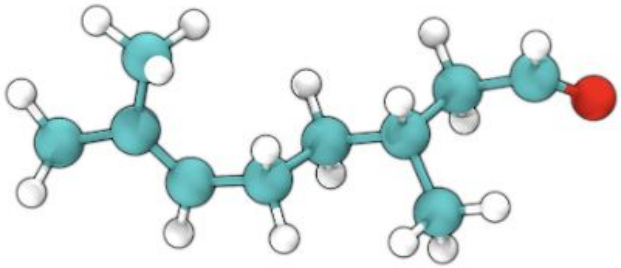




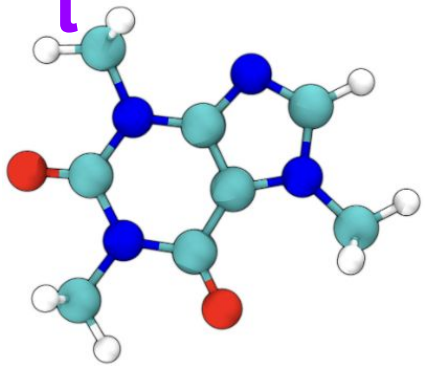
Directed edge



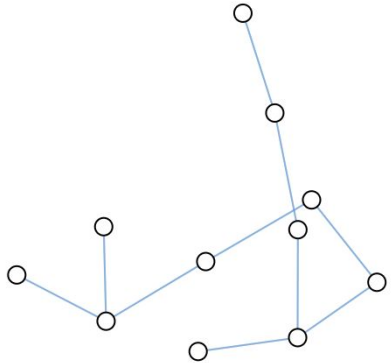
Molecules as graphs

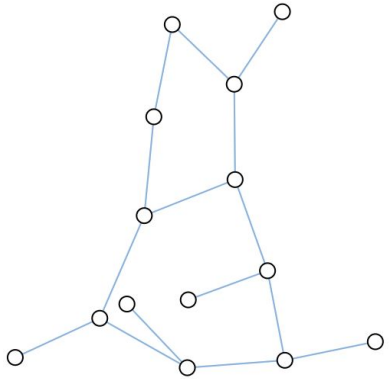


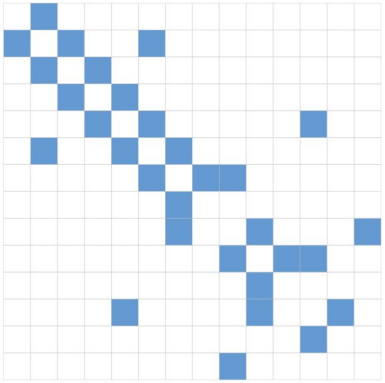
Citronellal

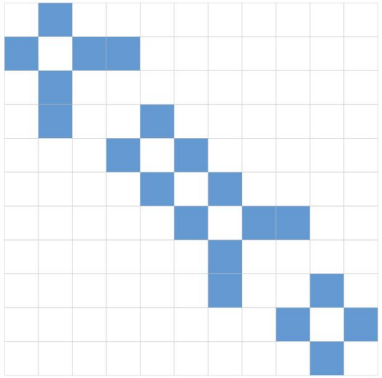


Caffeine

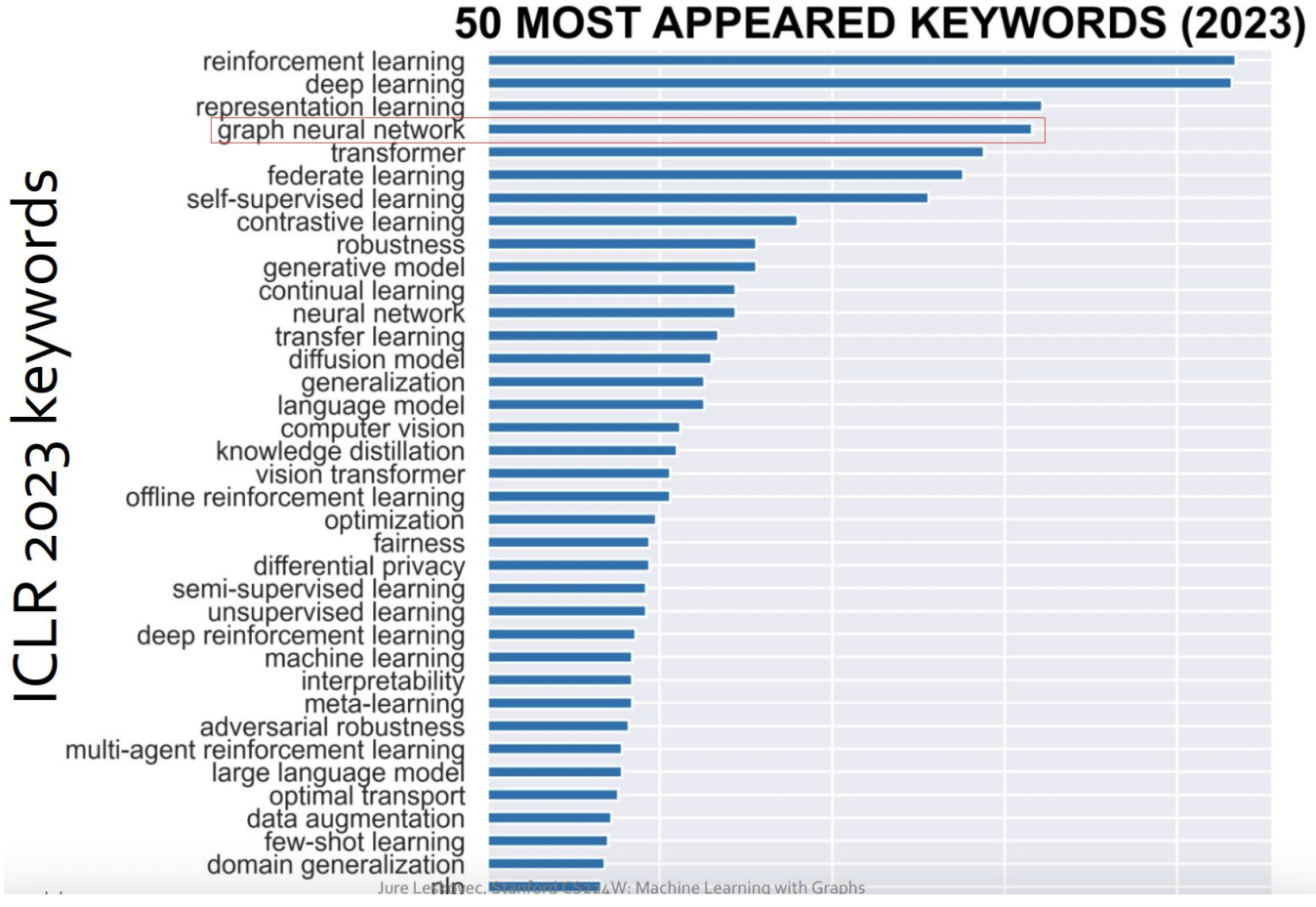






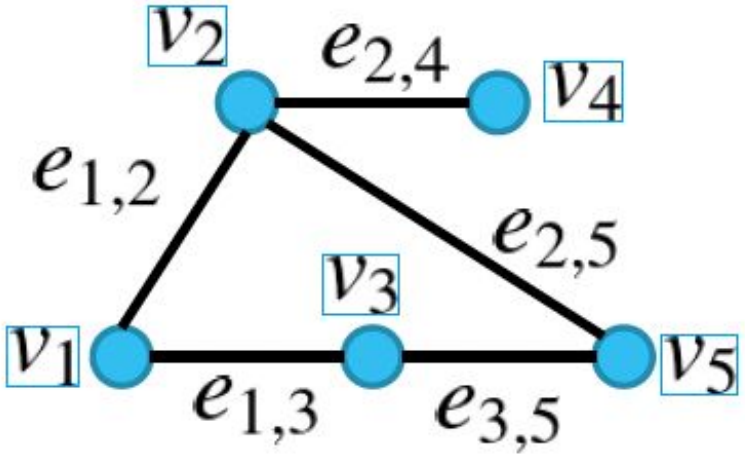


Hot subfiled in ML



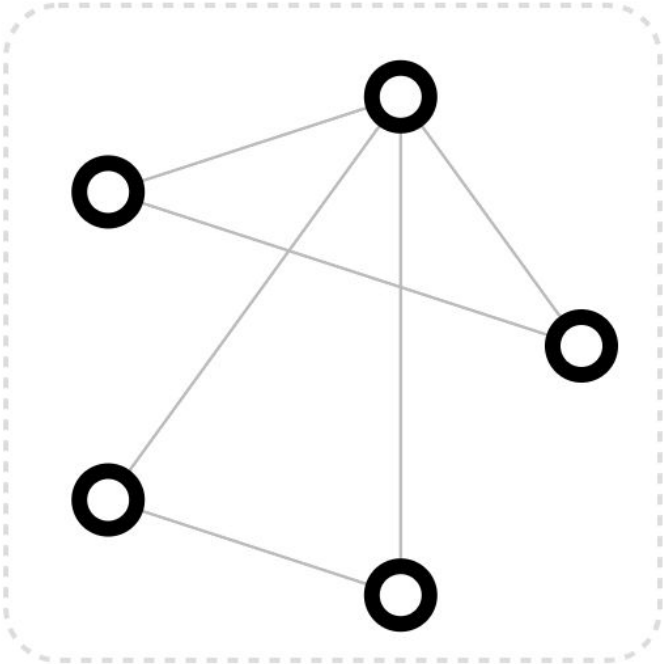
Graph neural networks

- Graph $G = \{v, \varepsilon\}$
- Set of nodes (vertices) $v = \{v_i\}$
- Set of edges $\varepsilon = \{e_{i,j}\}$
  - directional/non-directional



- Nodes and/or edges can have some features
  - Label, numbers, vectors

graph represents the relations (edges) between a collection of entities (nodes)

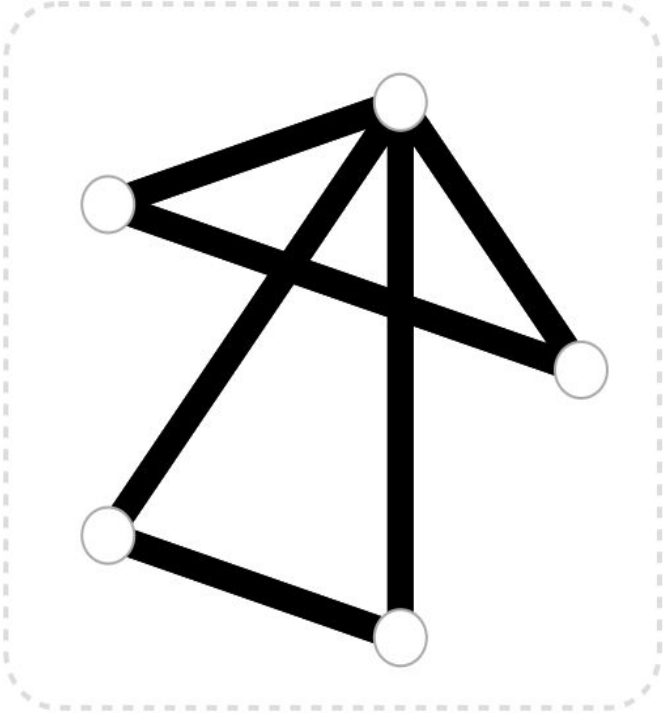


V Vertex (or node) attributes
e.g., node identity, number of neighbors

E Edge (or link) attributes and directions
e.g., edge identity, edge weight

U Global (or master node) attributes
e.g., number of nodes, longest path

graph represents the relations (edges) between a collection of entities (nodes)



V Vertex (or node) attributes

e.g., node identity, number of neighbors

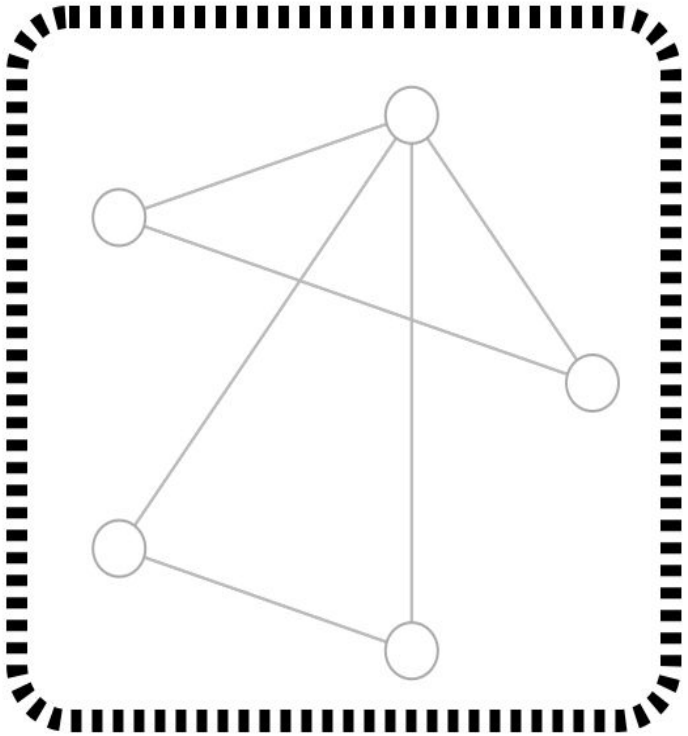
E Edge (or link) attributes and directions

e.g., edge identity, edge weight

U Global (or master node) attributes

e.g., number of nodes, longest path

graph represents the relations (edges) between a collection of entities (nodes)



V Vertex (or node) attributes

e.g., node identity, number of neighbors

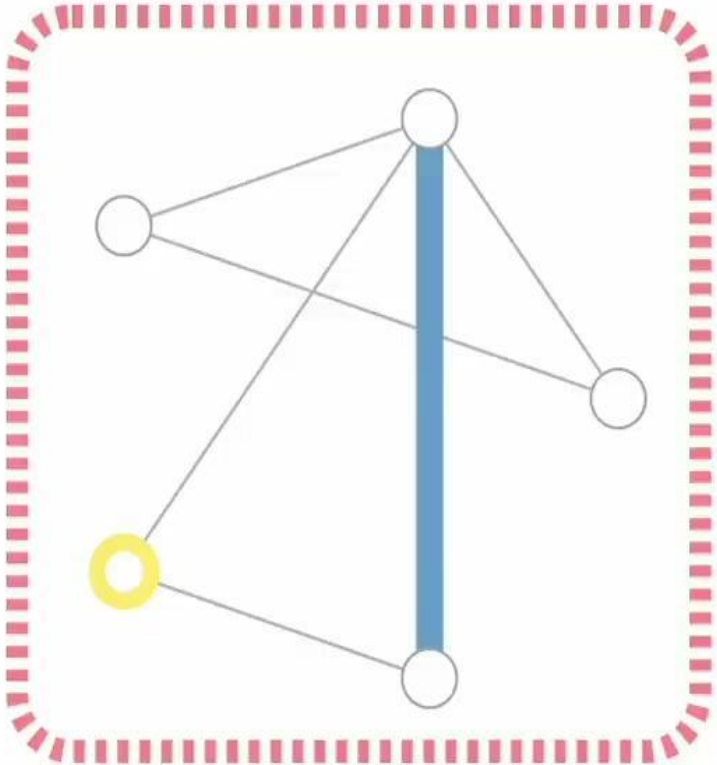
E Edge (or link) attributes and directions

e.g., edge identity, edge weight

U Global (or master node) attributes

e.g., number of nodes, longest path

we can store information in each of pieces of the graph



Vertex (or node) embedding

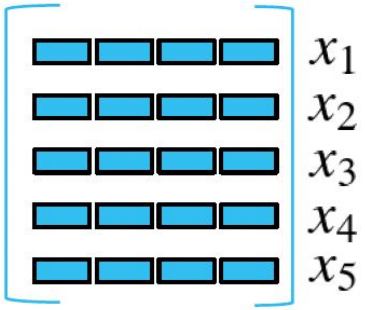
Edge (or link) attributes and embedding

Global (or master node) embedding

Matrix representation of attributed graphs

Feature Matrix $X = (x_{i,d}) = (x_1\ x_2\ \dots)^T$

$X \in \mathbb{R}^{N \times D}$ N nodes, D dim. features



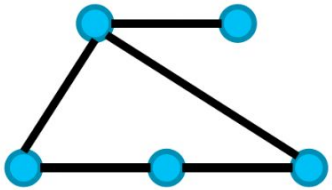
Adjacency Matrix $A = (a_{i,j})_{i,j=1,2,\dots}$

$A \in \{0,1\}^{N \times N}$ Edge existence between N X N node pairs

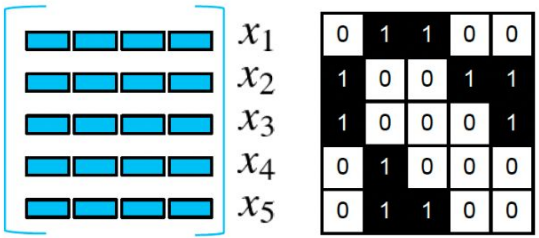
Symmetric A <-- non-directional edges

0	1	1	0	0
1	0	0	1	1
1	0	0	0	1
0	1	0	0	0
0	1	1	0	0

GNN functionality: Compute latent representation



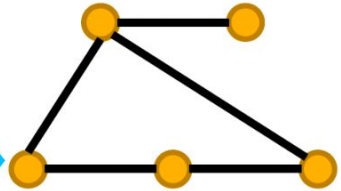
$G = (X, A)$

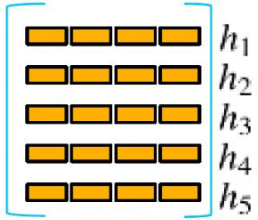


$X \in \mathbb{R}^{N \times D}$ $A \in \{0, 1\}^{N \times N}$

L-layered GNN

Latent repr. Matrix H of nodes





$H \in \mathbb{R}^{N \times D'}$

Node Prediction tasks

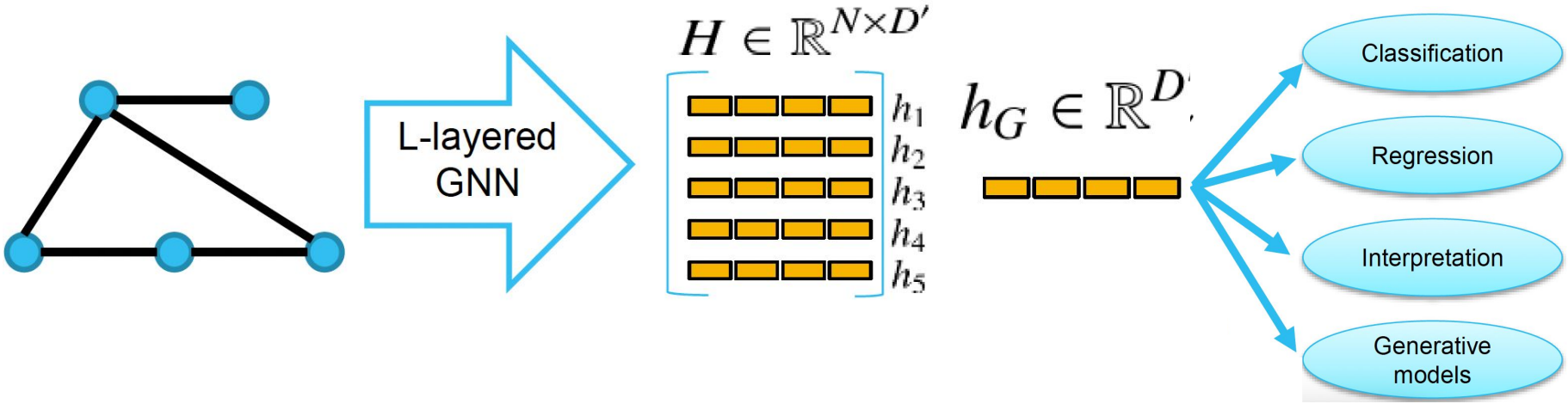
Latent repr. Vector h_G of a graph

$h_G \in \mathbb{R}^{D''}$

Graph Prediction tasks

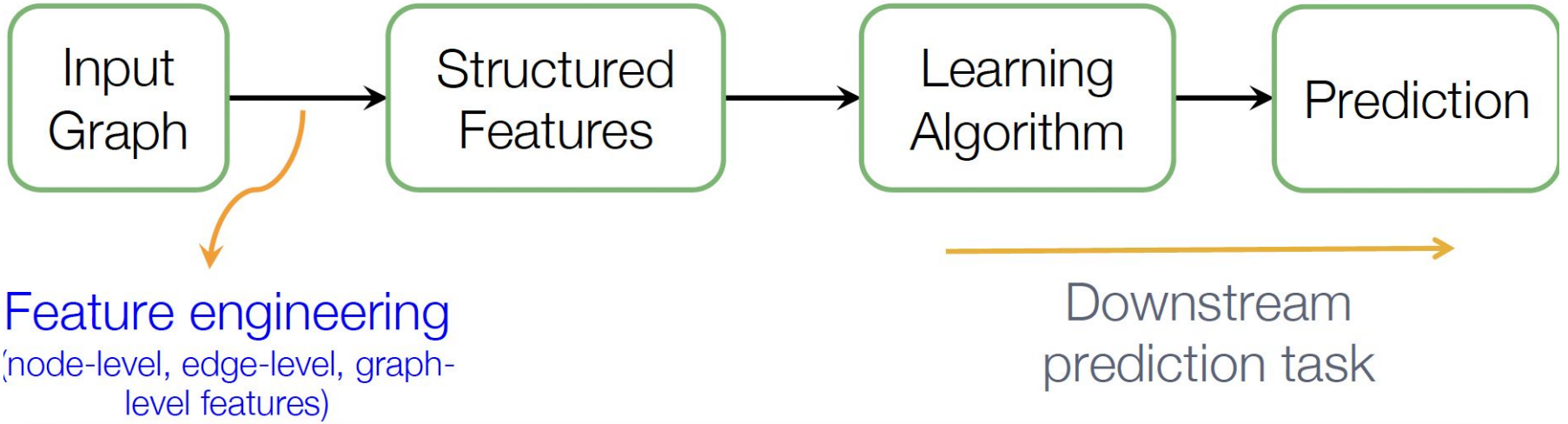
GNN functionality: Compute latent representation

- H is informative and easy-to-handle “data” for machine learning (ML) models/tools.



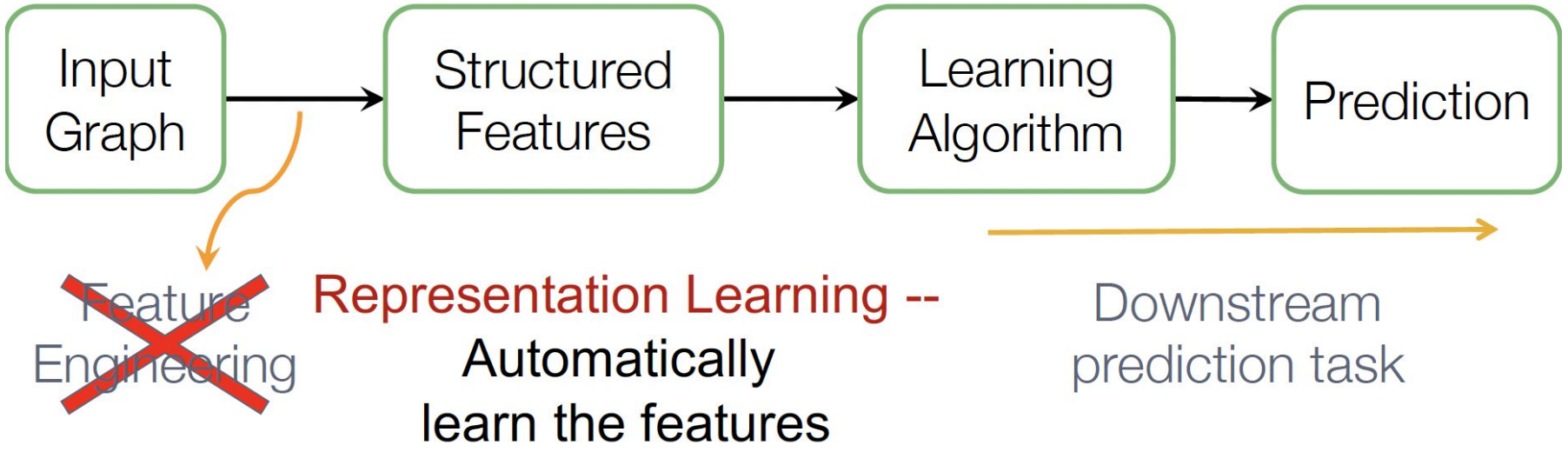
Traditional ML

- **Given an input graph, extract node, link and graph-level features, then learn a model (SVM, neural network, etc.) that maps features to labels.**



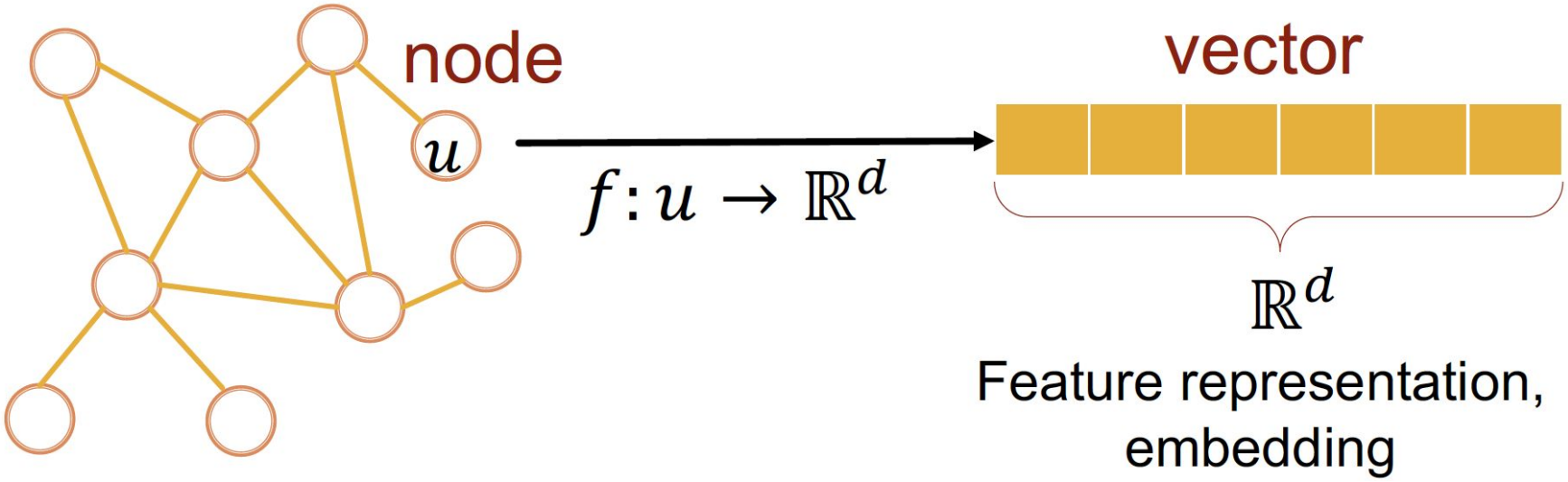
Graph representation learning

- **Graph Representation Learning alleviates the need to do feature engineering every single time.**



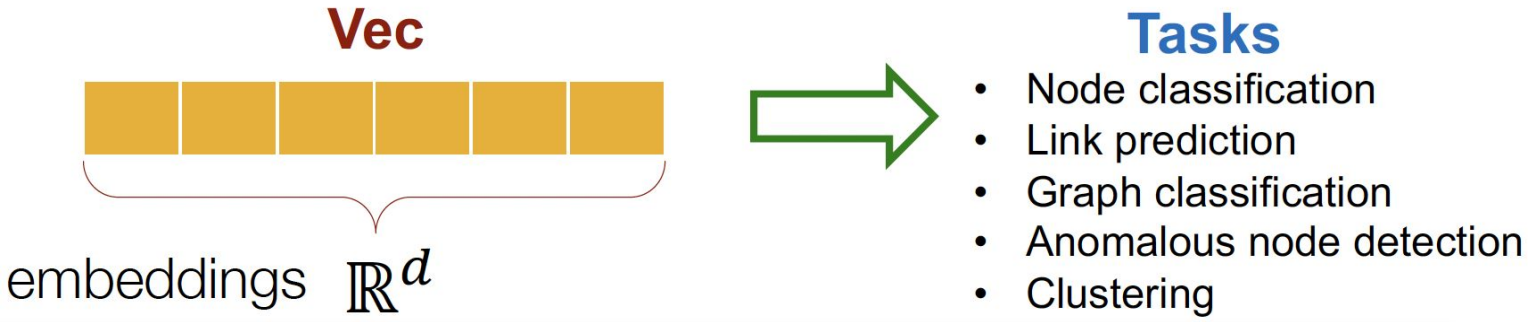
Graph representation learning

- **Goal: Efficient task-independent feature learning for machine learning with graphs!**



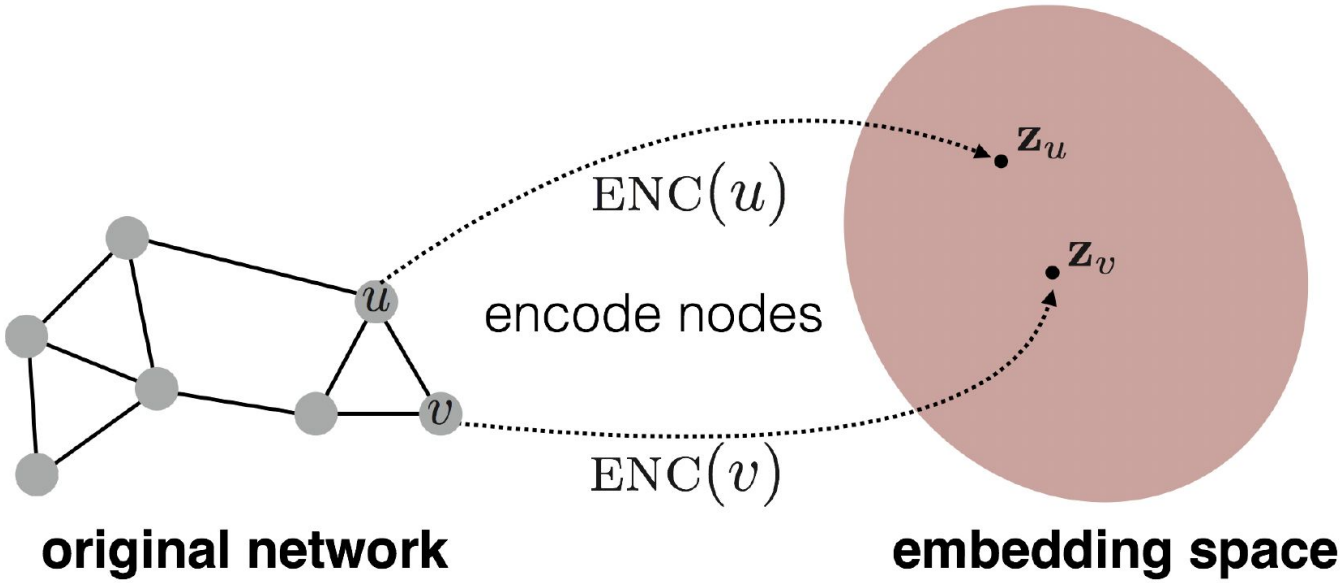
Why Embedding

- **Task: Map nodes into an embedding space**
- **Similarity of embeddings between nodes indicates**
  - their similarity in the network. For example:
  - Both nodes are close to each other (connected by an edge)
- **Encode network information**
- **Potentially used for many downstream predictions**

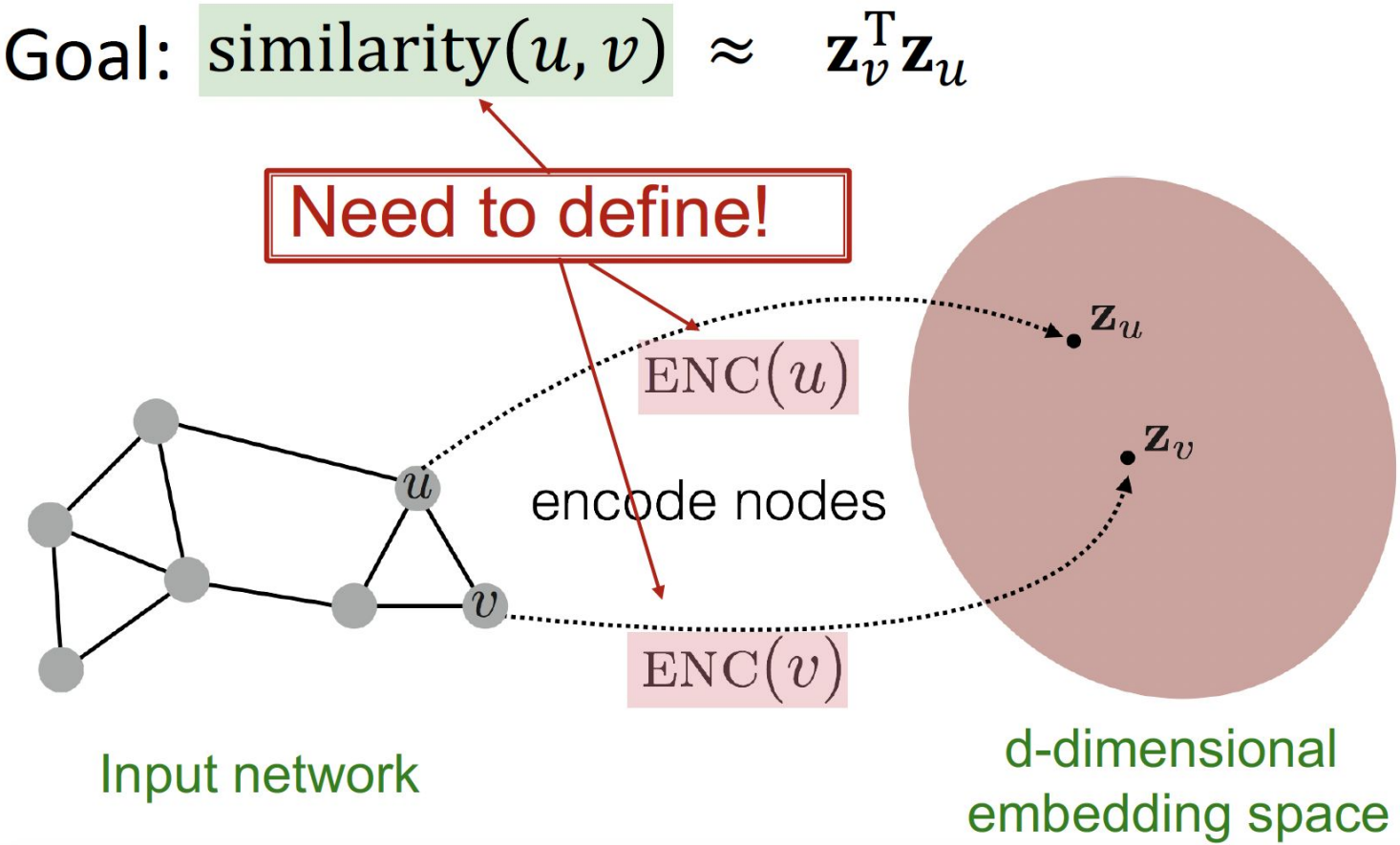


Embedding nodes

- Goal is to encode nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the graph



Embedding nodes

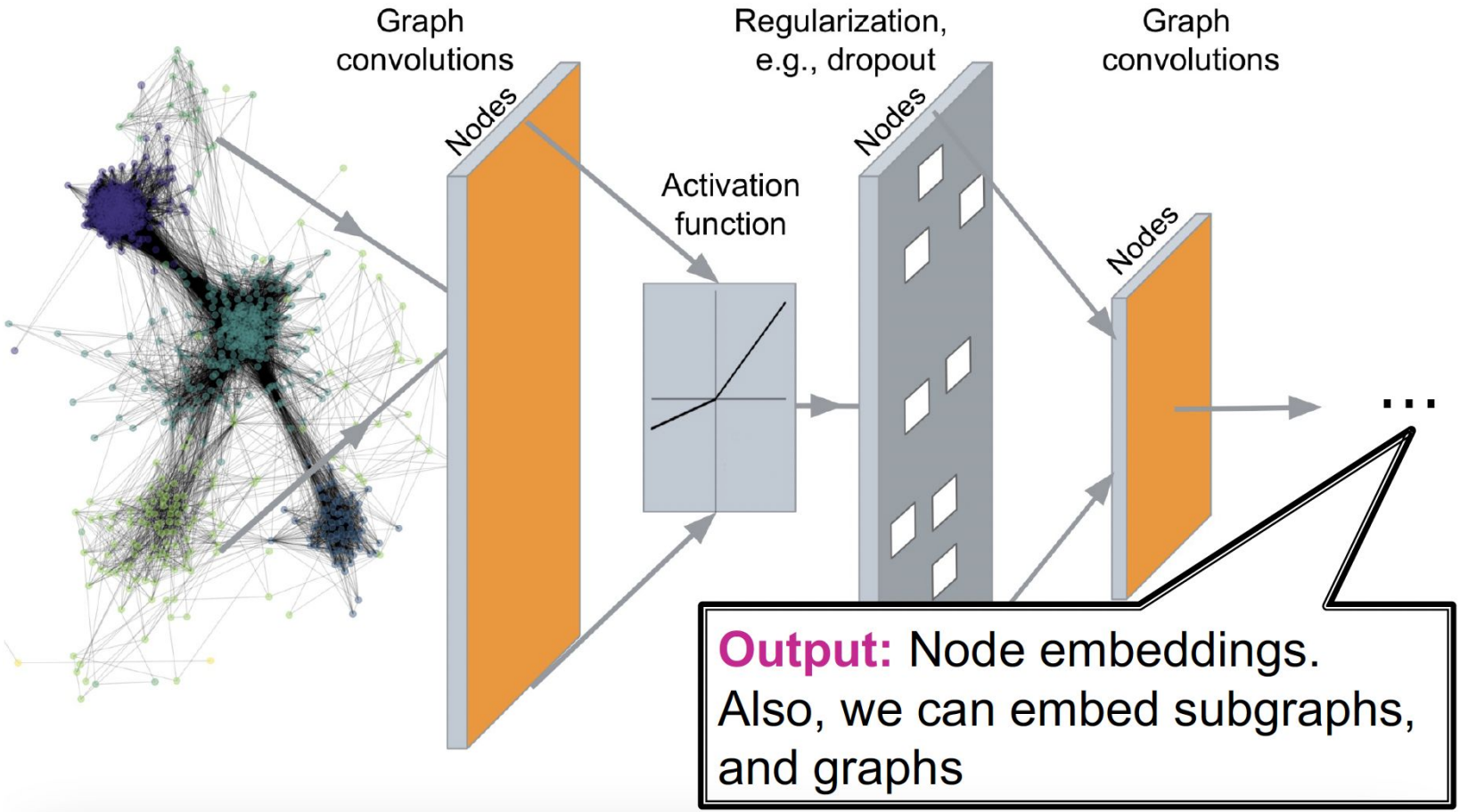


Deep graph encoders

- Deep learnig methods based on graph neural networks (GNNs):

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

Deep graph encoders



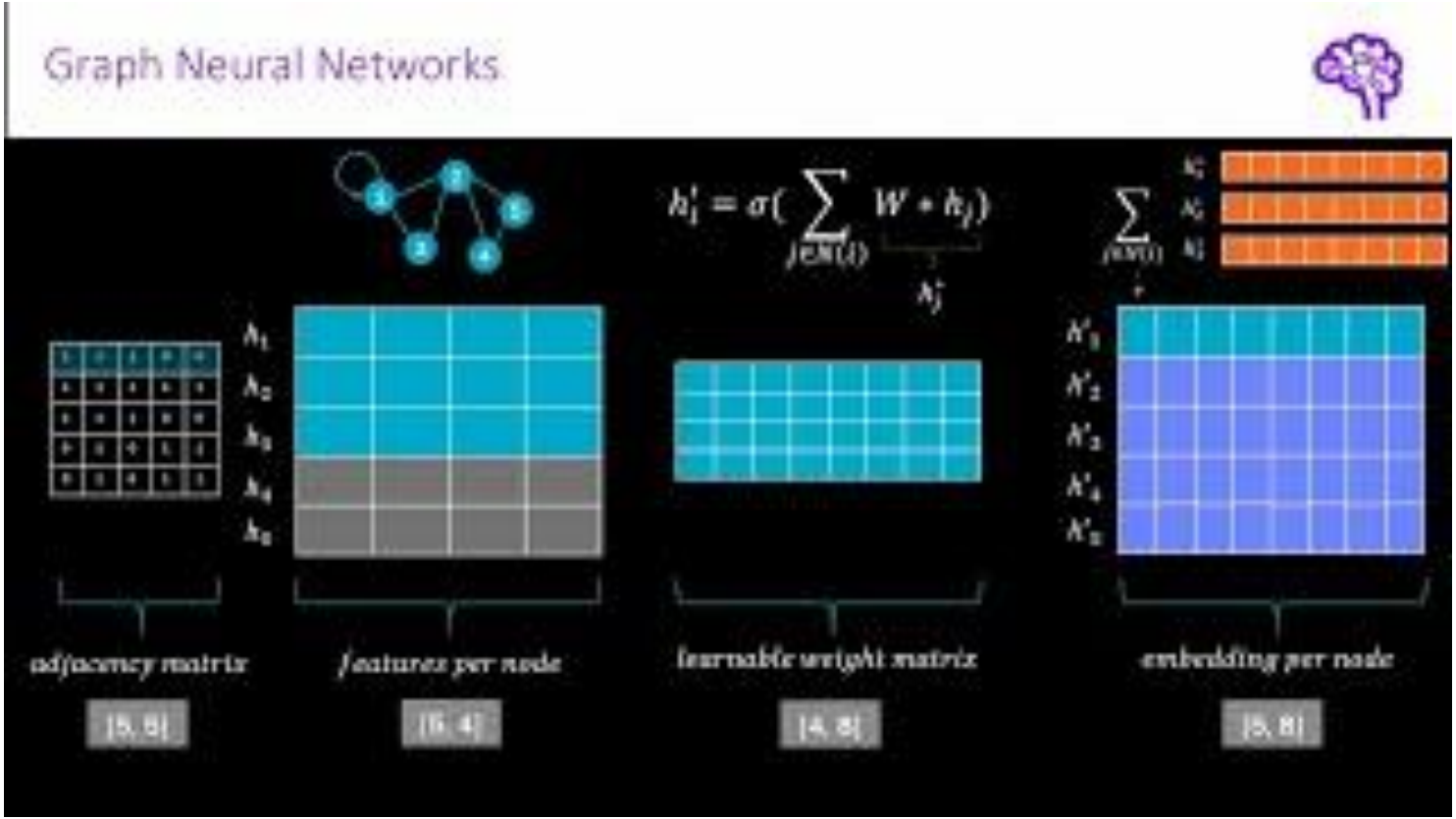
Recap: Basics of deep learning

- **Loss function $\mathcal{L}$:**

$$\min_{\Theta} \mathcal{L}(\boldsymbol{y}, f_{\Theta}(\boldsymbol{x}))$$

- f can be a simple linear layer, an MLP, or other neural networks (e.g., a GNN later)
- Sample a minibatch of input data $\boldsymbol{x}$
- **Forward propagation:** Compute $\mathcal{L}$ given $\boldsymbol{x}$
- **Back-propagation:** Obtain gradient $\nabla_{\Theta}\mathcal{L}$ using a chain rule.
- Use **stochastic gradient descent (SGD)** to optimize $\mathcal{L}$ for weights Θ over many iterations.

GNN functionality



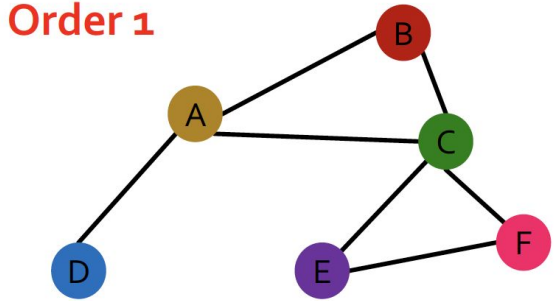
https://youtu.be/A-yKQamf2Fc?si=cfoIMTadg2UmDyyk

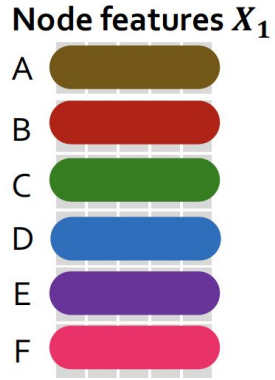
Permutation invariance

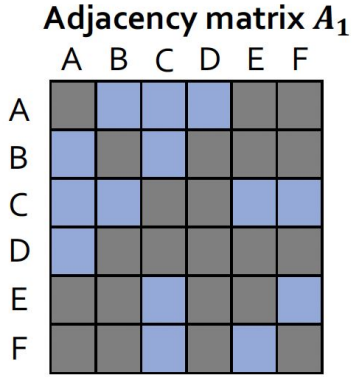
- **Consider we want to embed an entire graph**
- **Observation: A graph does not have a canonical ordering of its nodes**
  - We can have many different node orderings of the same graph.
- **What do we want: If we learn an embedding function over a graph, we should get the same result (the same embedding) regardless of how the nodes are numbered.**

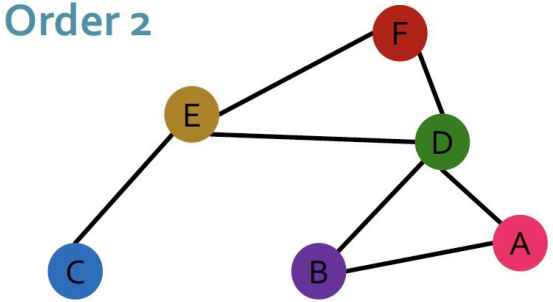
Permutation invariance

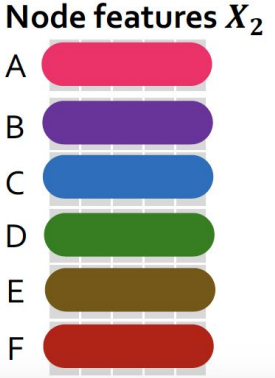
- Graph does not have a canonical ordering of the nodes!

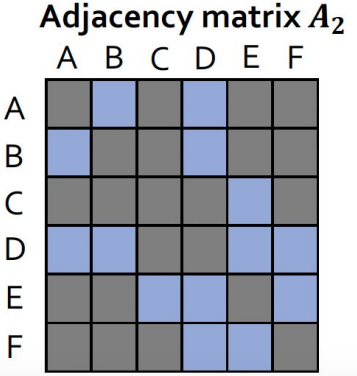






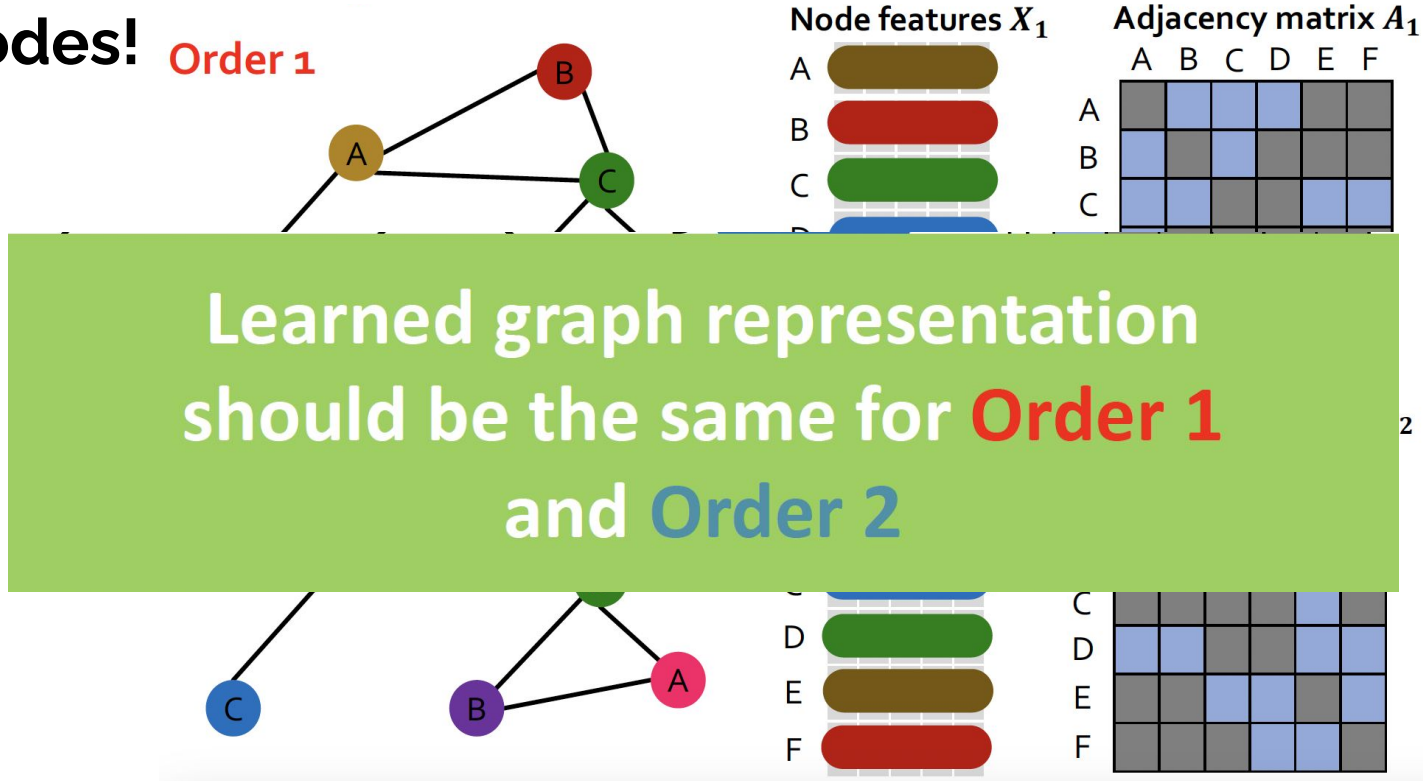






Permutation invariance

- Graph does not have a canonical ordering of the nodes!



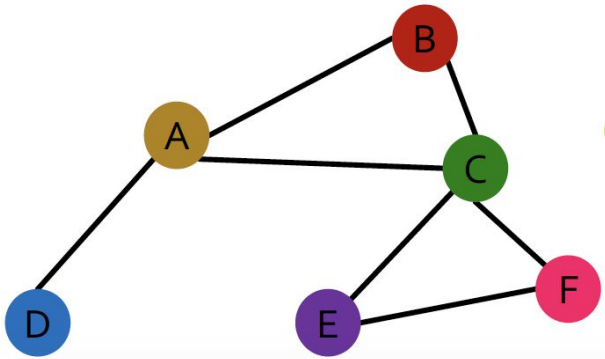
Permutation invariance

- What do we mean by "graph representation is the same for two orderings"?

$$f(\boldsymbol{A}_1, \boldsymbol{X}_1) = f(\boldsymbol{A}_2, \boldsymbol{X}_2)$$

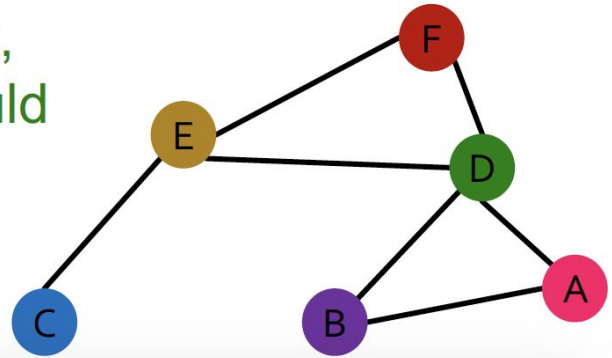
$\boldsymbol{A}$ is the adjacency matrix
$\boldsymbol{X}$ is the node feature matrix

Order 1: A_1, X_1



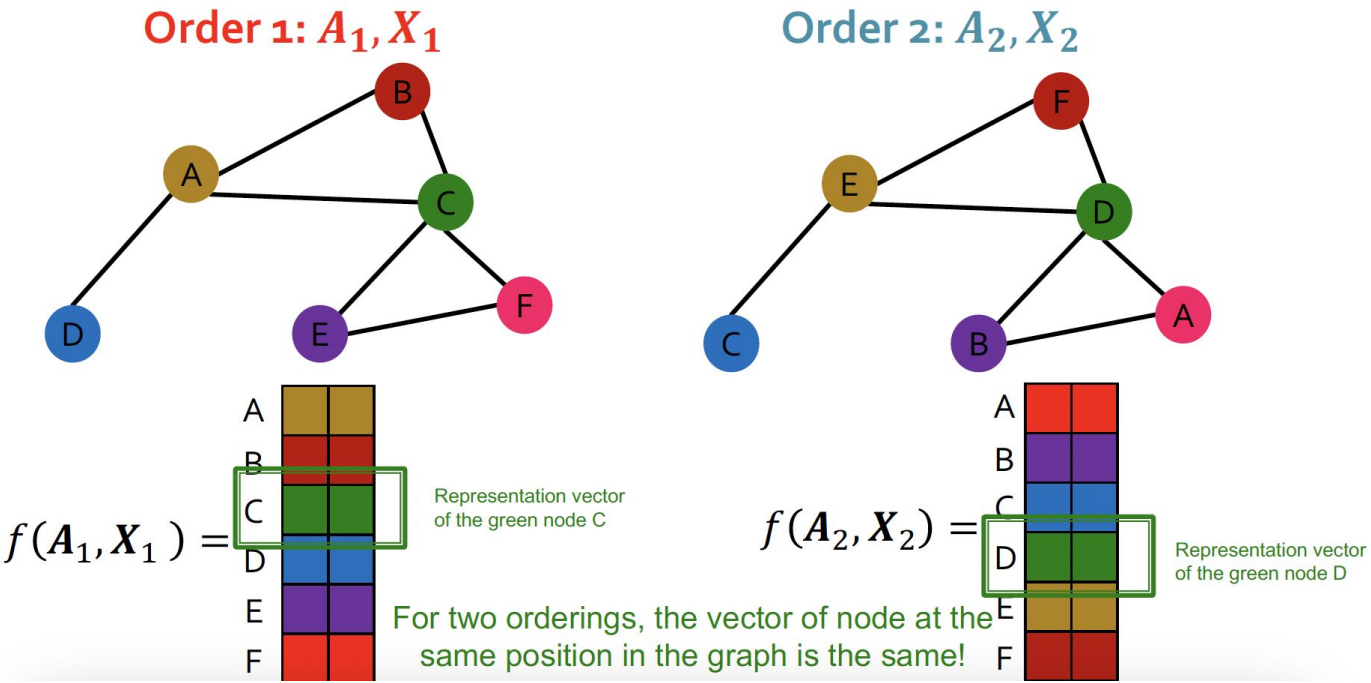
For two orders, output of f should be the same!

Order 2: A_2, X_2



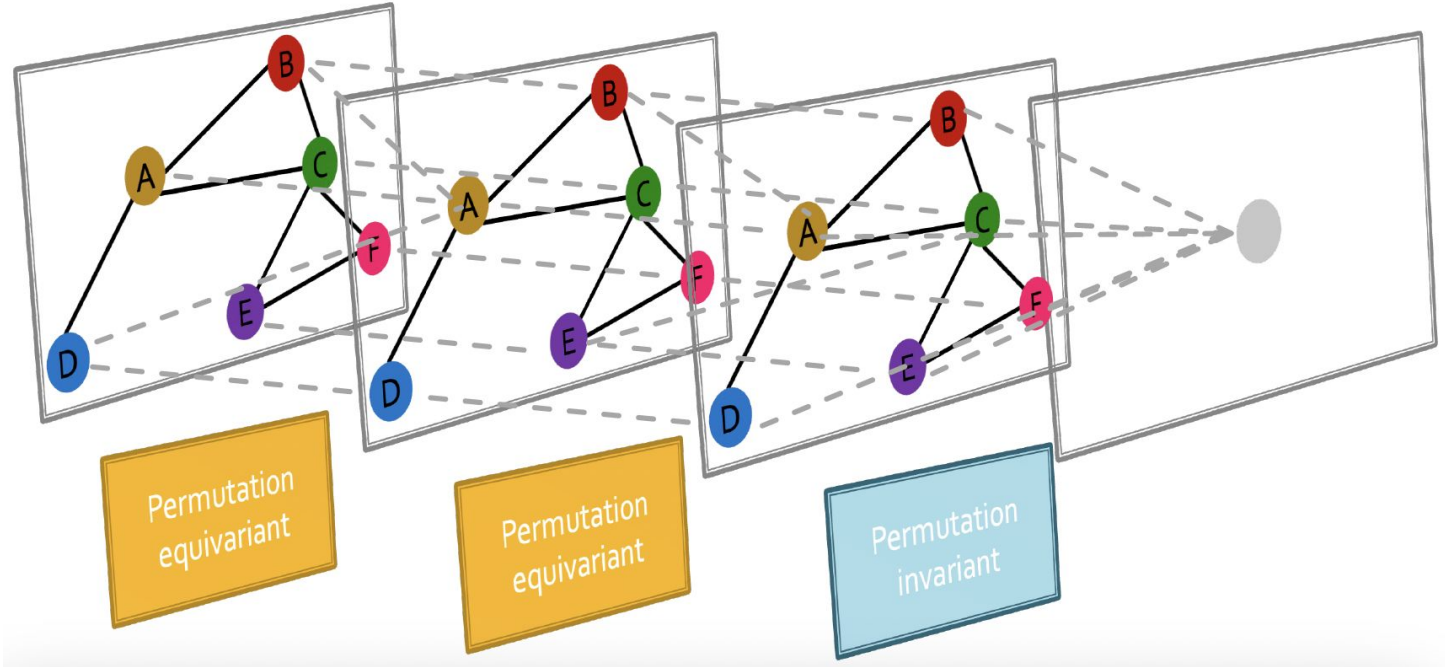
Permutation equivariance

- If the output vector of a node at the same position in the graph remains unchanged for any ordering



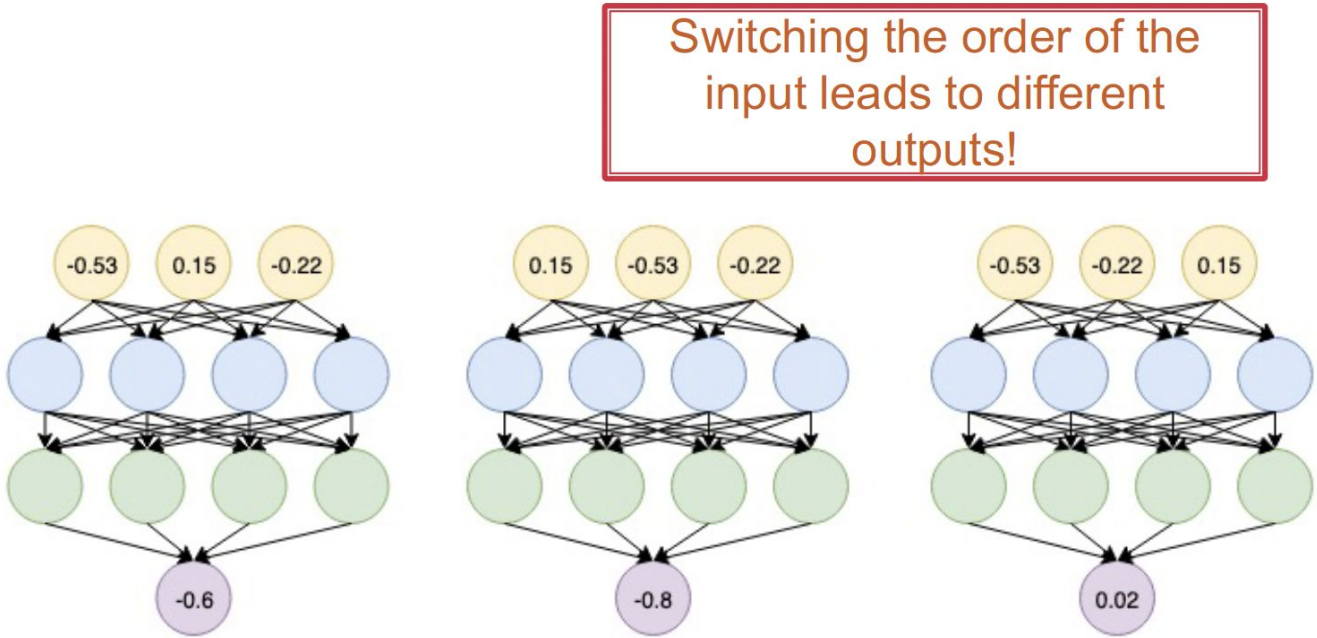
Graph neural networks overview

- Graph neural networks consist of multiple permutation equivariant / invariant functions



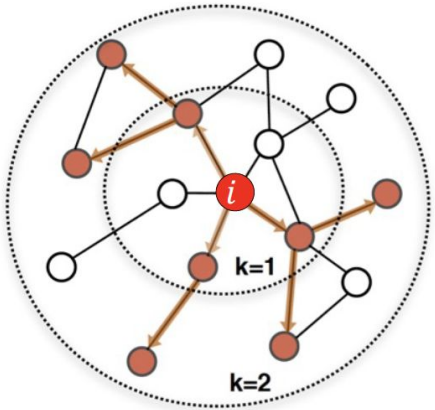
Graph neural networks overview

- Are other neural network architectures, e.g., MLPs, permutation invariant / equivariant? **NO!**

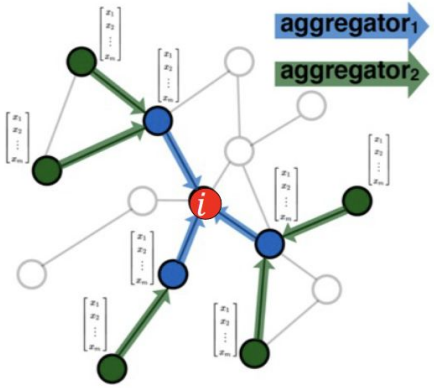


Graph convolutional networks

- Next: Design graph neural networks that are permutation invariant / equivariant by passing and aggregating information from neighbors!
- IDEA: Node's neighborhood defines a computation graph



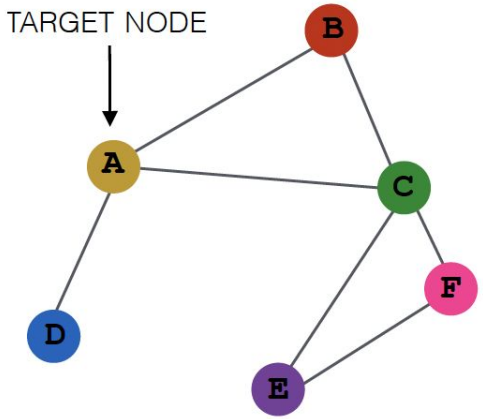
Determine node computation graph



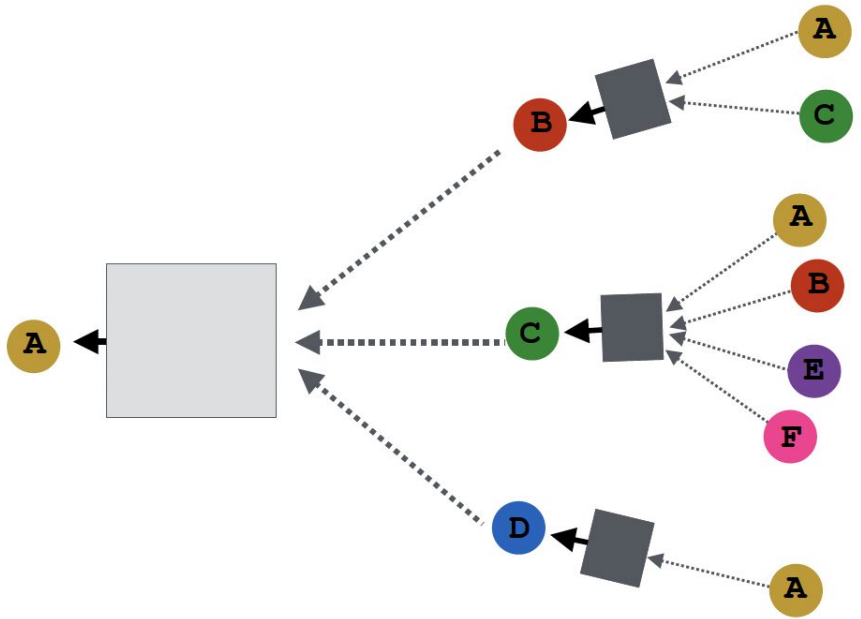
Propagate and transform information

Aggregate neighbors

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

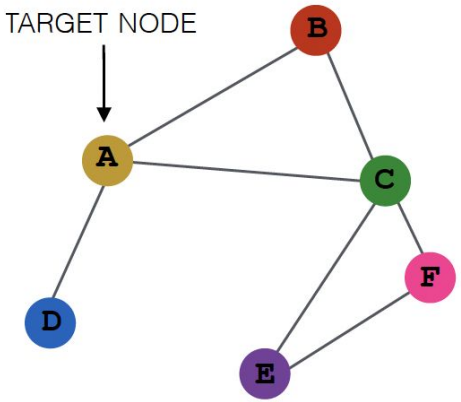


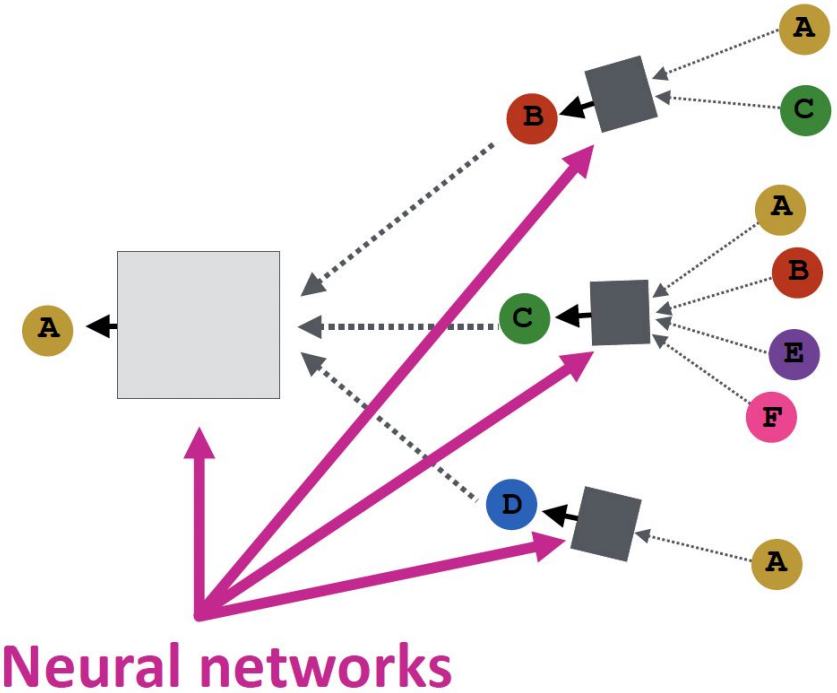
INPUT GRAPH



Aggregate neighbors

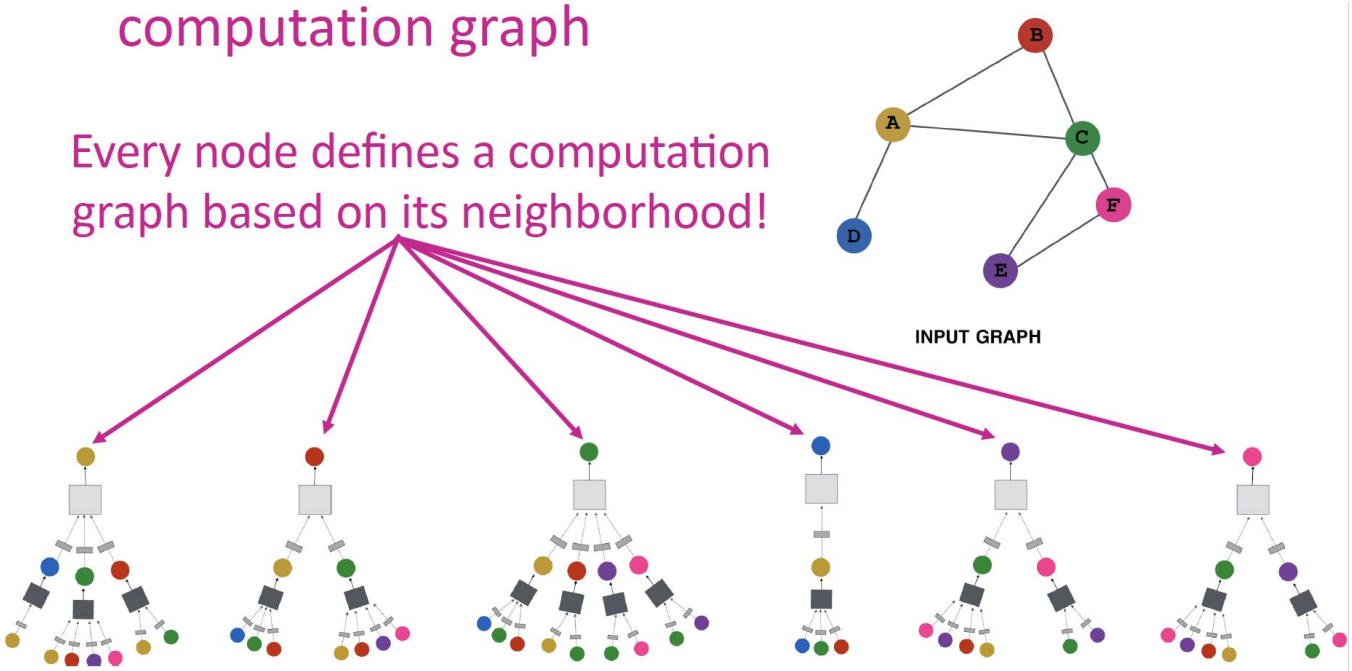
- **Intuition:** Nodes aggregate information from their neighbors using neural networks





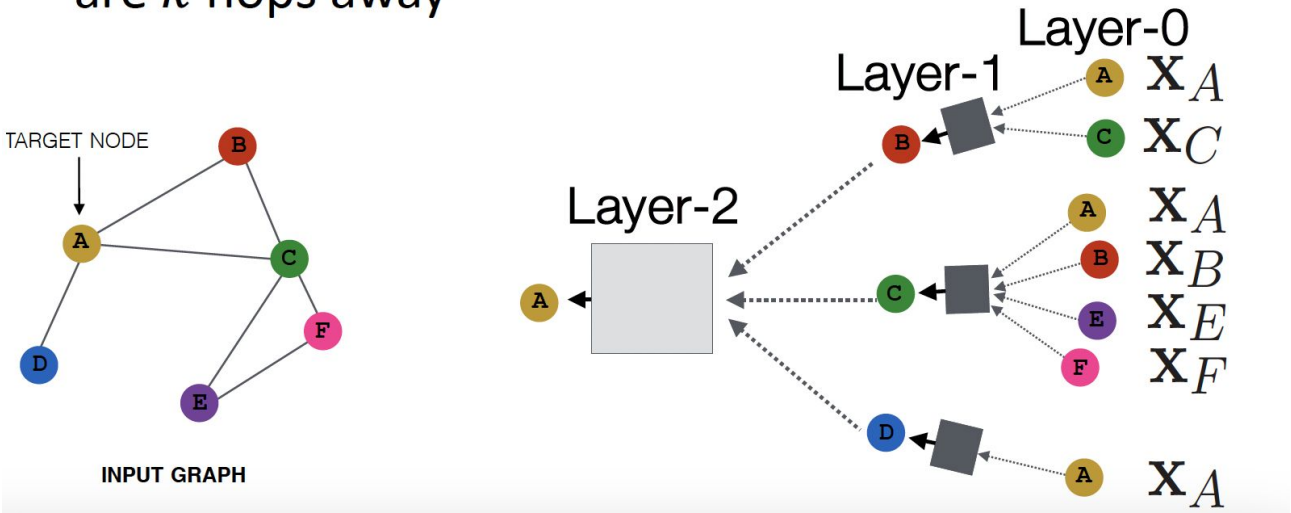
Aggregate neighbors

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



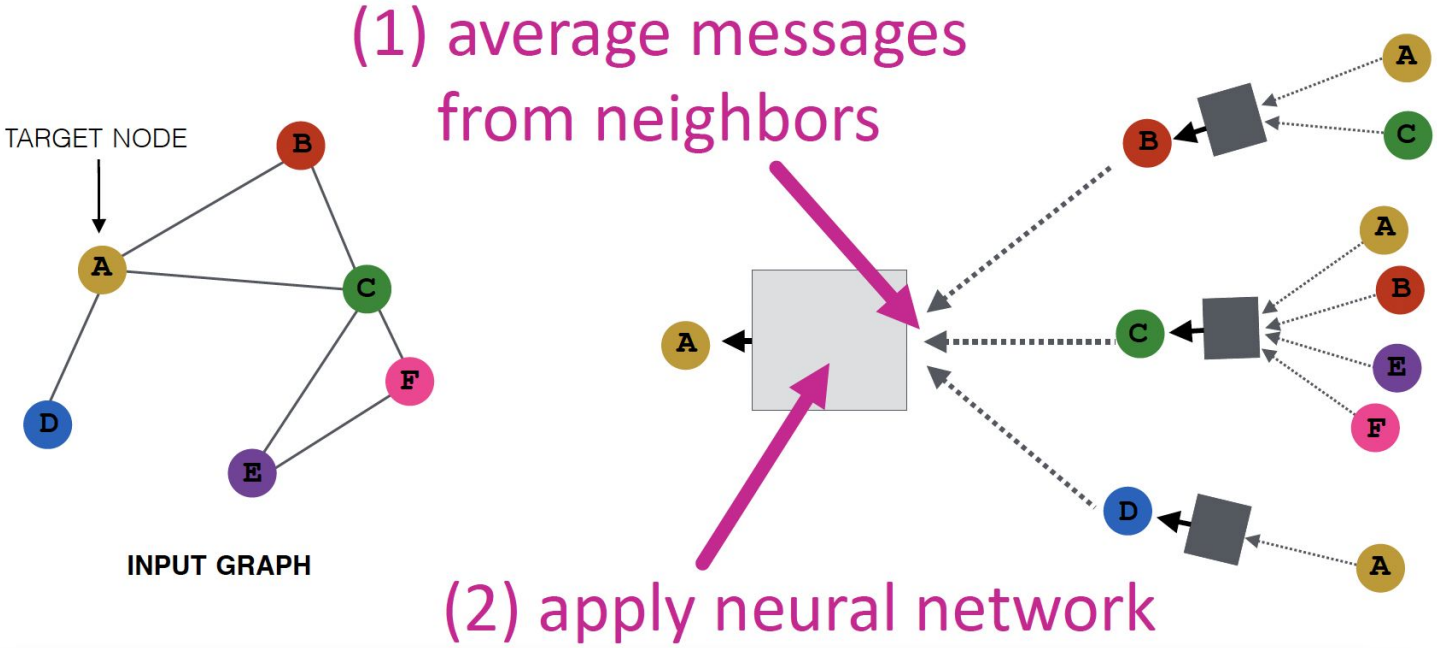
Deep model: many layers

- **Model can be of arbitrary depth**
  - Nodes have embeddings at each layer
  - Layer-0 embedding of node v is its input feature, x_v
  - Layer-k embedding gets information from nodes that are k hops away



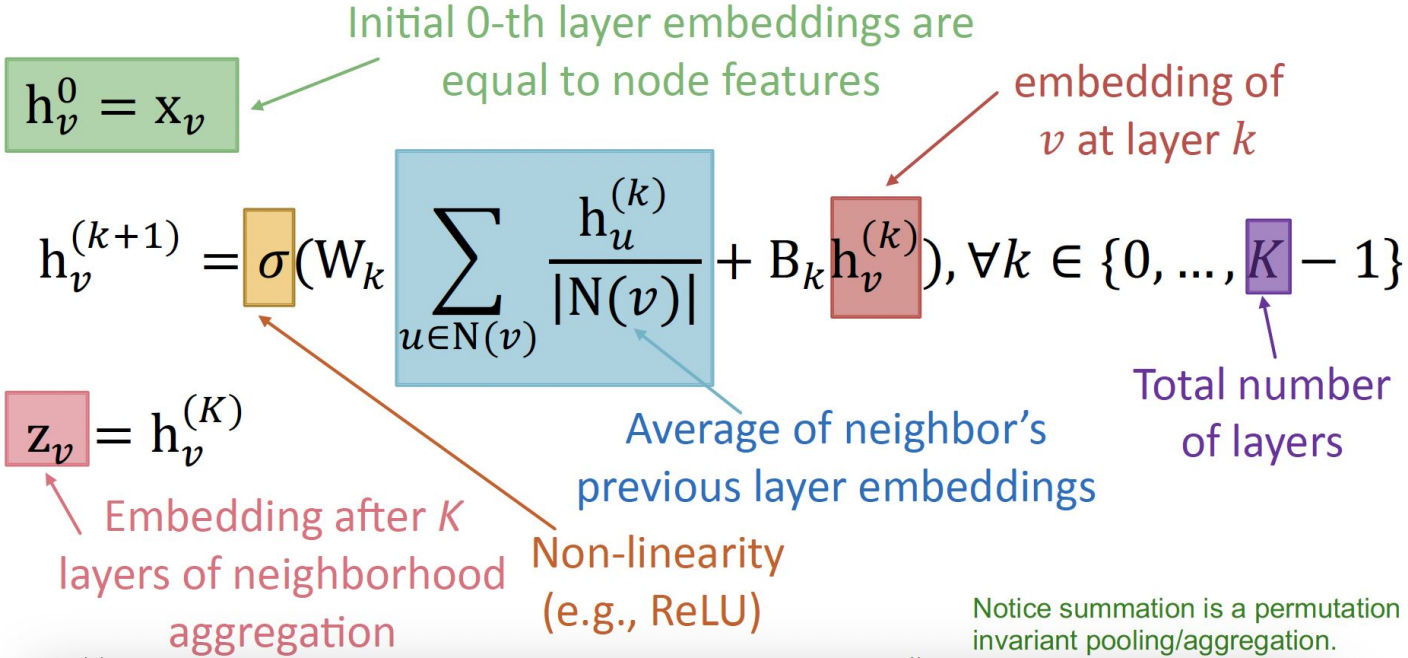
Neighborhood aggregation

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



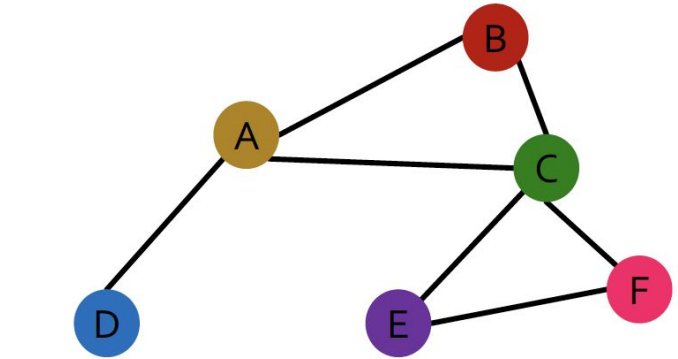
The math: deep encoder of a GCN

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

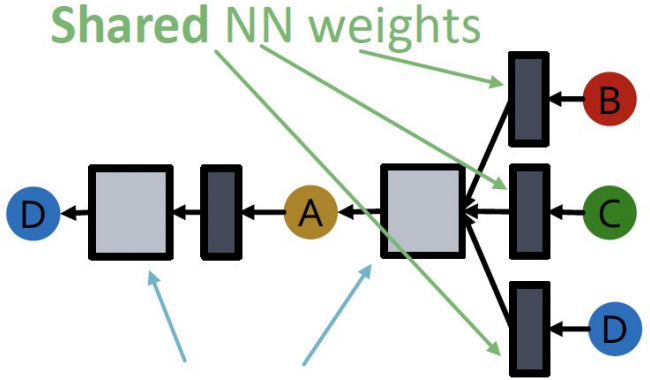


GCN: Invariance and equivariance

- What are the invariance and equivariance properties for a GCN?
- Given a node, the GCN that computes its embedding is permutation invariant



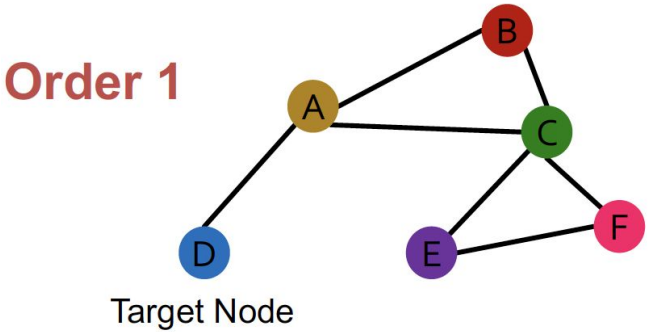
Target Node

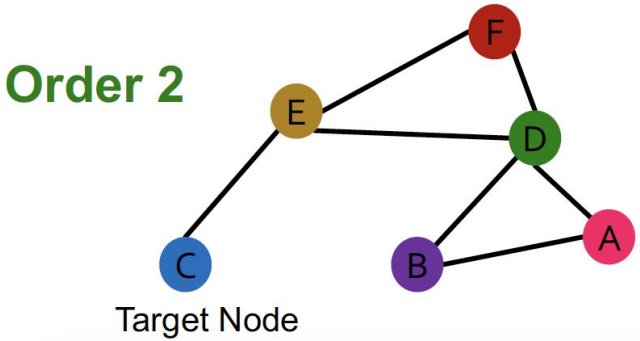


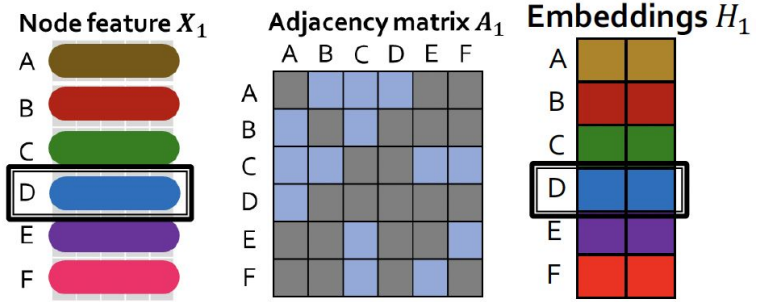
Average of neighbor's previous layer embeddings - **Permutation invariant**

GCN: Invariance and equivariance

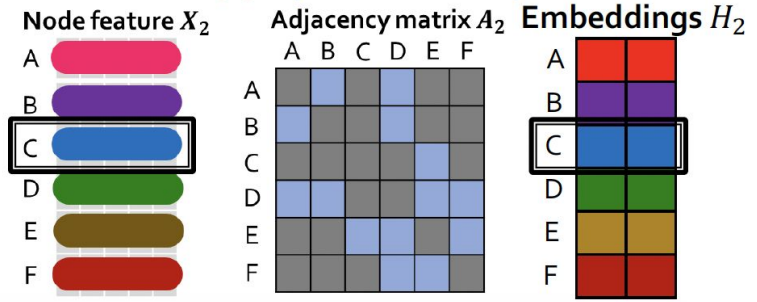
- Considering all nodes in a graph, GCN computation is **permutation equivariant!**







Permute the input, the output also permutes accordingly - permutation equivariant



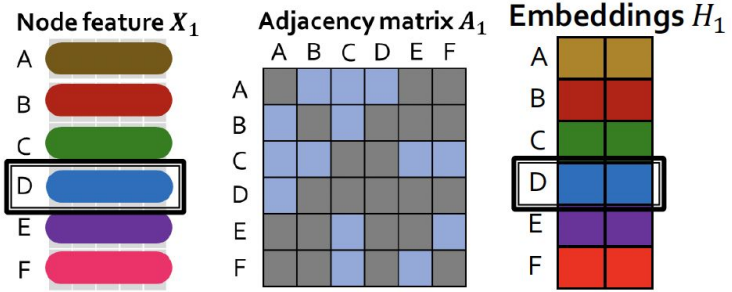
GCN: Invariance and equivariance

- **Considering all nodes in a graph, GCN computation is permutation equivariant!**

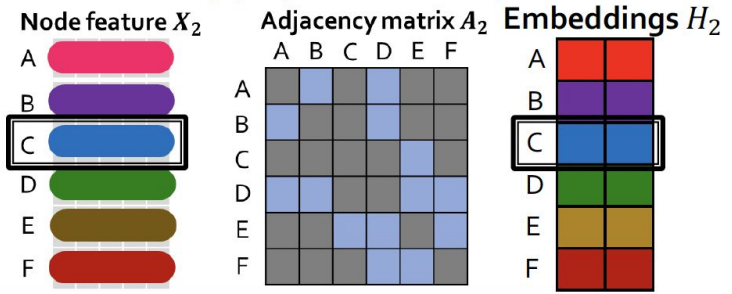
Detailed reasoning:

1. The rows of **input node features** and **output embeddings** are **aligned**
2. We know computing the embedding of **a given node** with GCN is **invariant.**
3. So, after permutation, the **location** of **a given node** in the **input node feature** matrix is changed, and the **the output embedding of a given node stays the same** (the colors of node feature and embedding are **matched**)

This is permutation equivariant



Permute the input, the output also permutes accordingly - permutation equivariant



Training the model

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

$$h_v^{(0)} = x_v$$

$$h_v^{(k+1)} = \sigma\left(W_k \sum_{u \in N(v)} \frac{h_u^{(k)}}{|N(v)|} + B_k h_v^{(k)}\right), \forall k \in \{0..K-1$$

$$z_v = h_v^{(K)}$$

Final node embedding

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

h_v^k: the hidden representation of node v at layer k

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

Training the model

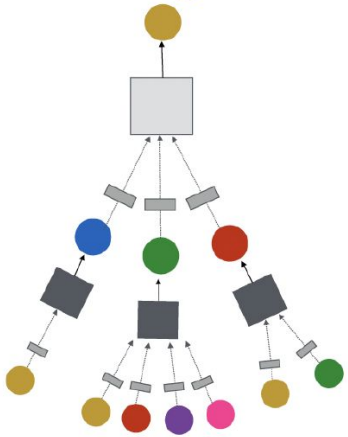
- Node embedding $\boldsymbol{z}_v$ is a function of input graph
- **Supervised setting**: We want to minimize loss $\mathcal{L}$:

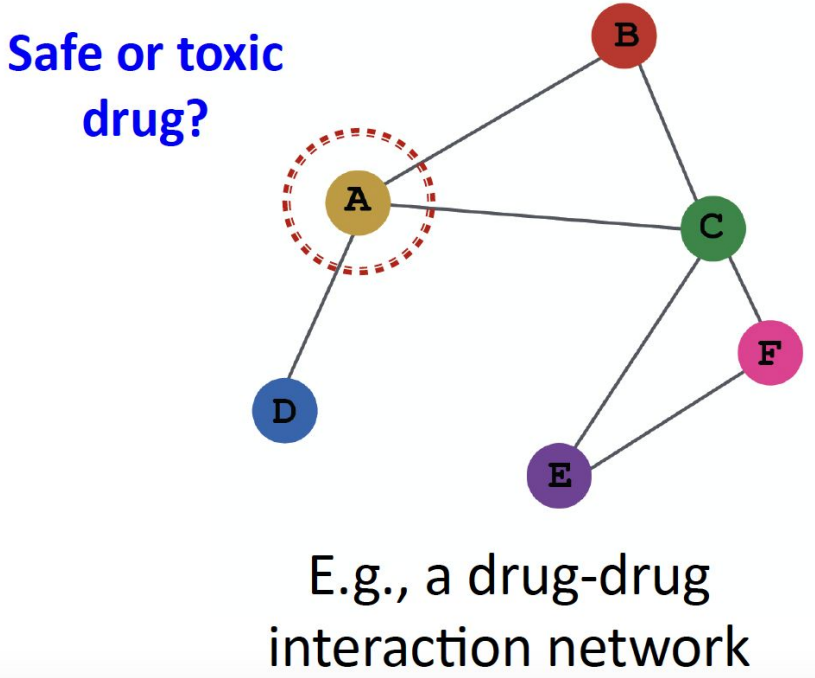
$$\min_{\Theta} \mathcal{L}(\boldsymbol{y}, f_{\Theta}(\boldsymbol{z}_v))$$

  - $\boldsymbol{y}$: node label
  - $\mathcal{L}$ could be L2 if $\boldsymbol{y}$ is real number, or cross entropy if $\boldsymbol{y}$ is categorical
- **Unsupervised setting:**
  - No node label available
  - **Use the graph structure as the supervision!**

Supervised training

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

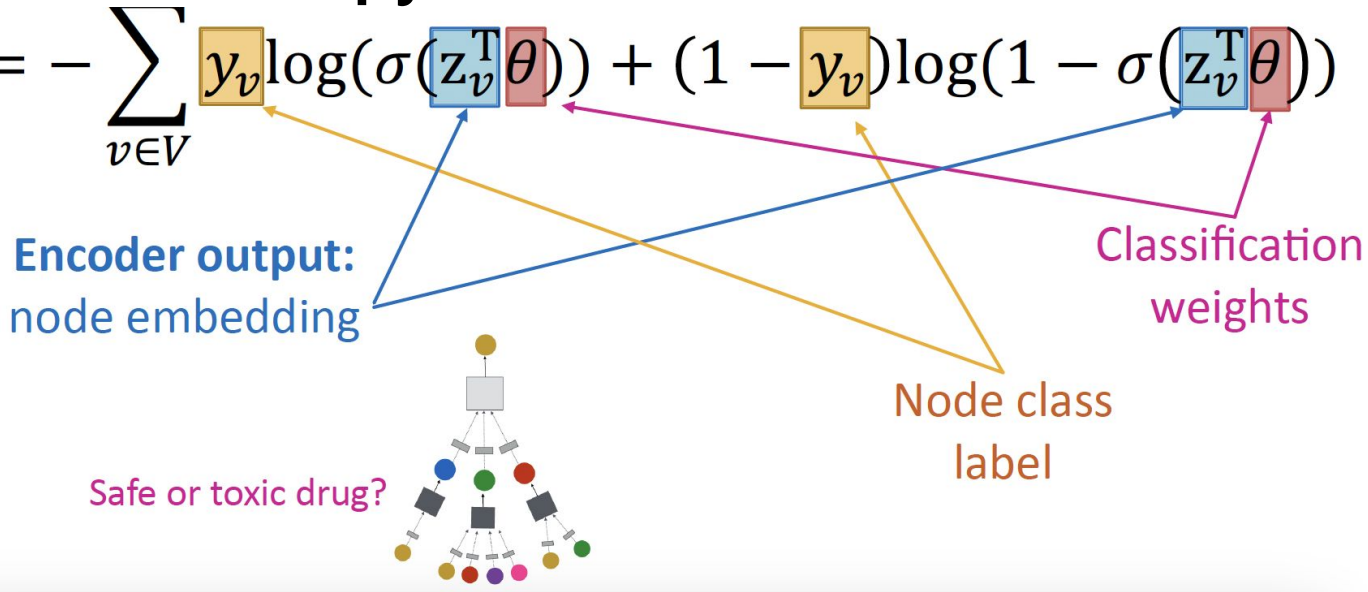




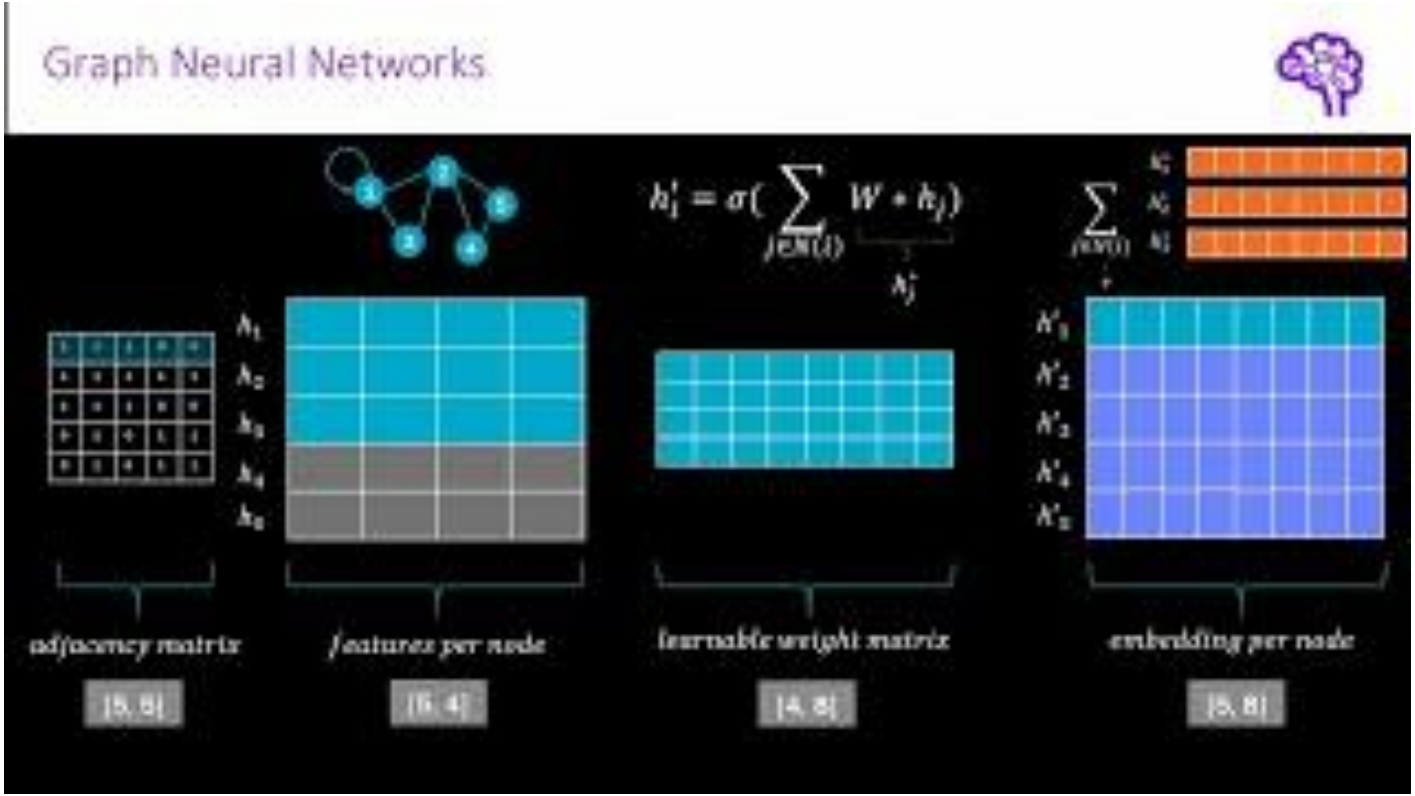
Supervised training

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

$$\mathcal{L} = -\sum_{v \in V} y_v \log(\sigma(z_v^T \theta)) + (1 - y_v)\log(1 - \sigma(z_v^T \theta))$$



Graph Attention Networks!



https://youtu.be/A-yKQamf2Fc?si=cfoIMTadg2UmDyyk

Protein Folding

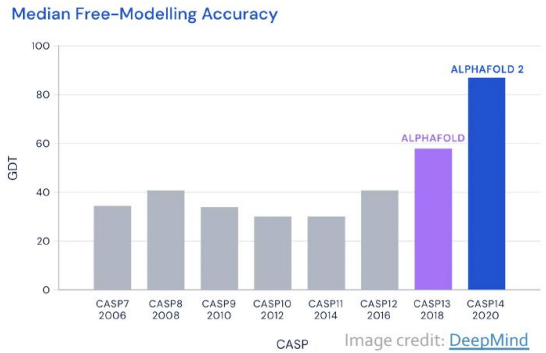


Image credit: DeepMind

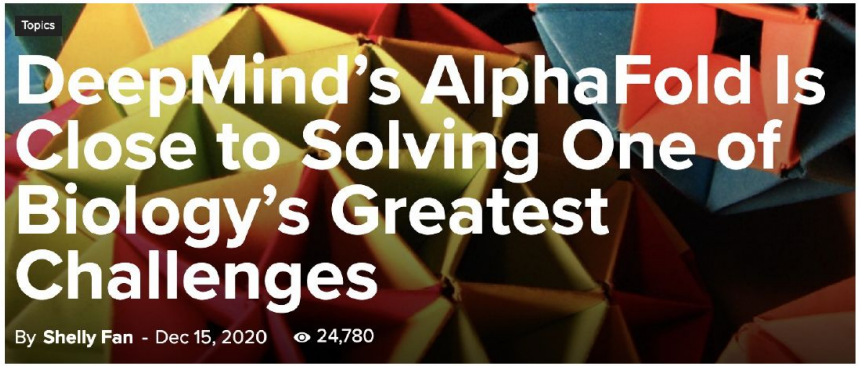


Image credit: SingularityHub

AlphaFold's AI could change the world of biological science as we know it

DeepMind's latest AI breakthrough can accurately predict the way proteins fold

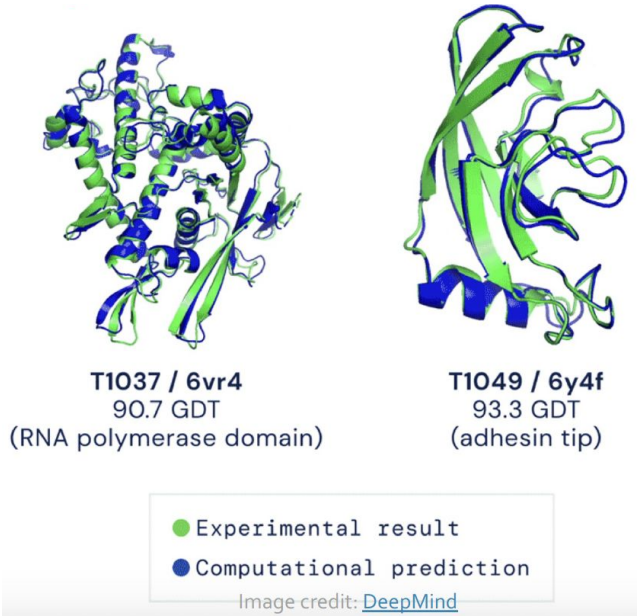
Has Artificial Intelligence 'Solved' Biology's Protein-Folding Problem?

12-14-20

DeepMind's latest AI breakthrough could turbocharge drug discovery

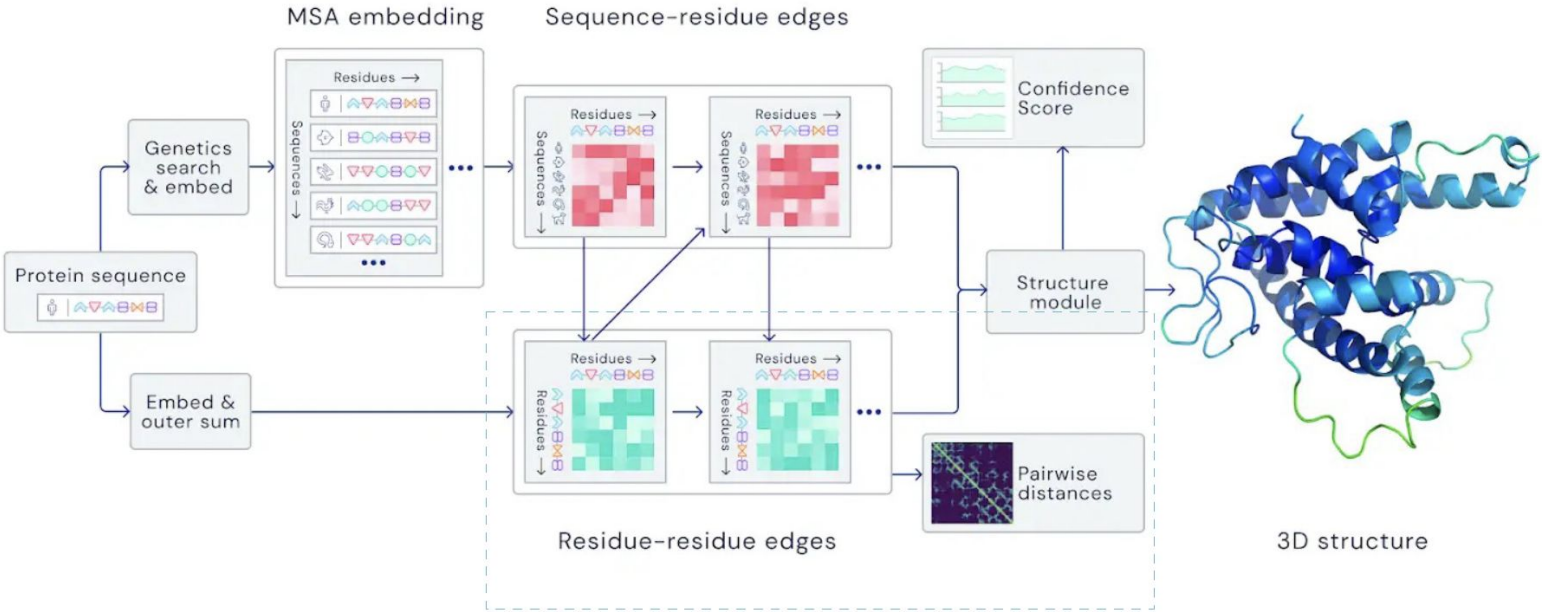
Protein Folding

- **Computational predict a protein's 3D structure based solely on its amino acid sequence**
  - For each graph node predict its 3D coordinates



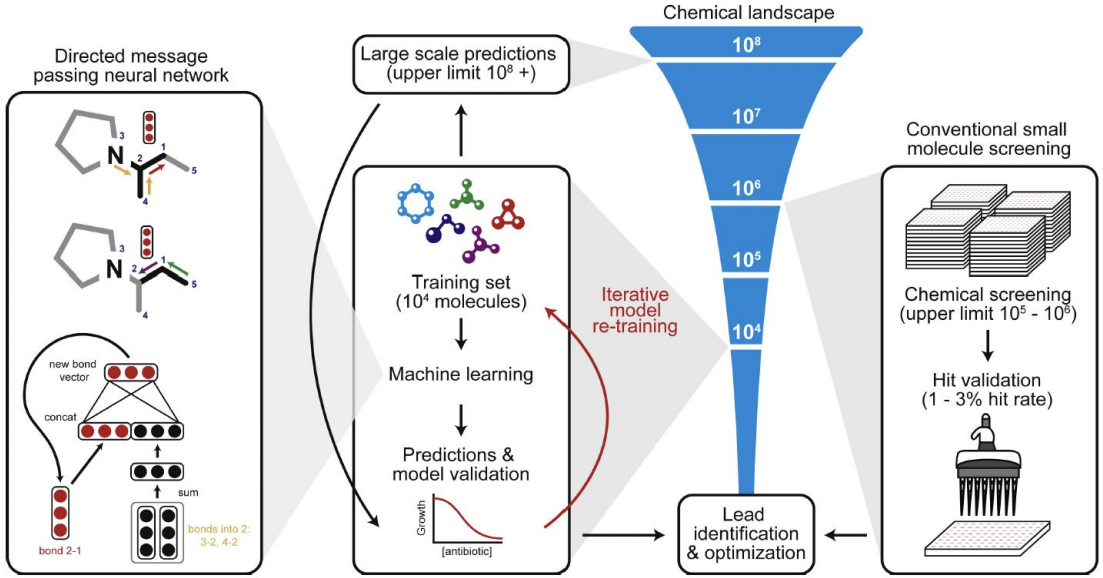
AlphaFold: Solving Protein Folding

- **Nodes:** Amino acids in a protein sequence
- **Edges:** Proximity between amino acids (residues)



GNN for Antibiotic Discovery

- A GNN graph classification model
- Predict promising molecules (hits) from a pool of candidates



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

Graph machine learning tools

- **PyG (PyTorch Geometric)**
  - The ultimate library for Graph Neural Networks (GNN).
- **GraphGym**
  - Platform for designing graph neural networks.
  - Modularized GNN implementation, simple hyperparameter tuning, flexible user customization