Relationship Modeling: Graph Neural Networks

Ju Sun
Computer Science & Engineering

Dec 13, 2022

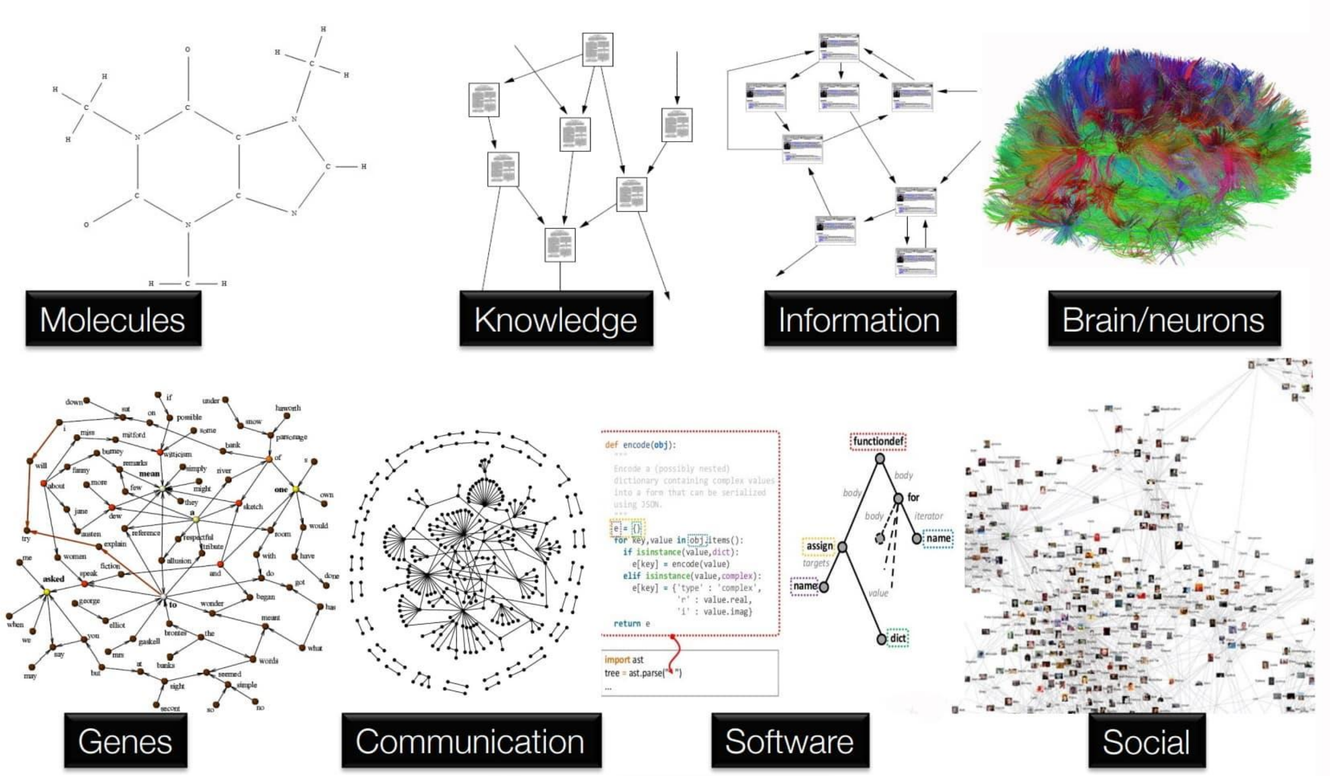
University of Minnesota
Driven to Discover℠

Outline

- Why graphs?
- Graphs: basic notions
- Graph neural networks
- Scaling up training

Why graphs?

Graphs are everywhere!



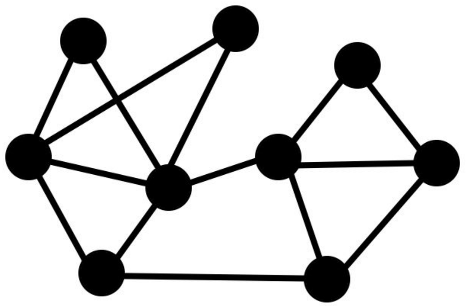


Image credit: Stanford CS224W

Graphs model **relationships/ interactions**

Image credit: https://blogs.nvidia.com/blog/2022/10/24/what-are-graph-neural-networks/

Different tasks on graphs

The whole graph is all available data

The whole graph is part of all available data, i.e., a data point

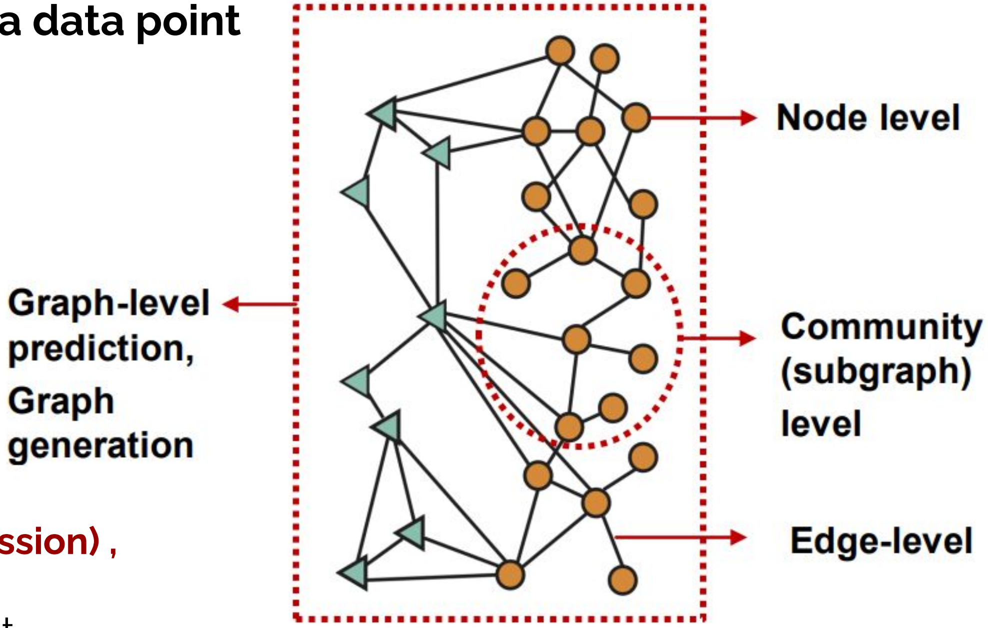


Image credit: Stanford CS224W

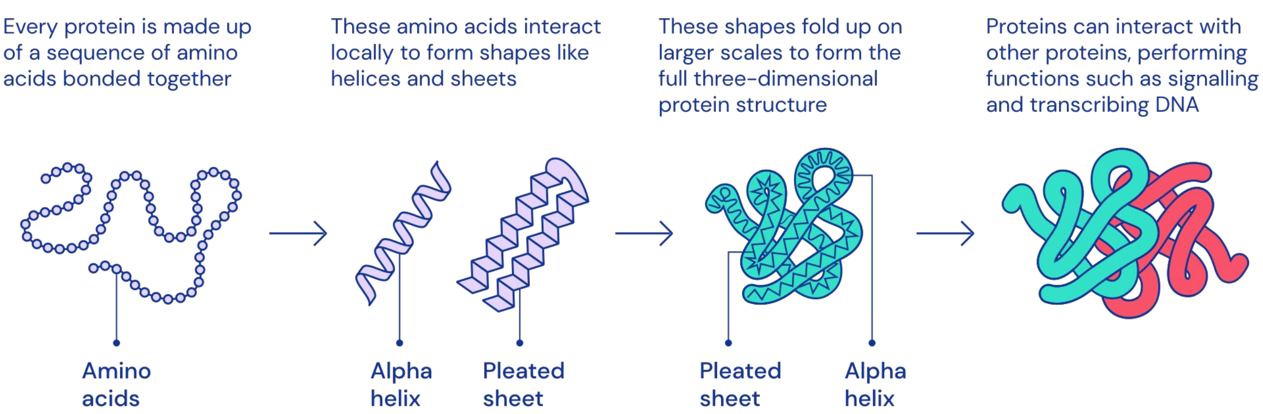
Node prediction (classification/regression)
given labels over part of the graph

Community detection/clustering
Clustering nodes/edges

Link prediction
Predict links should exists or not

Graph prediction (classification/regression) , graph generation
A graph is a data point

Example task 1: Protein folding



Protein folding: predict a protein's **3D structure** based on its **amino acid sequence**

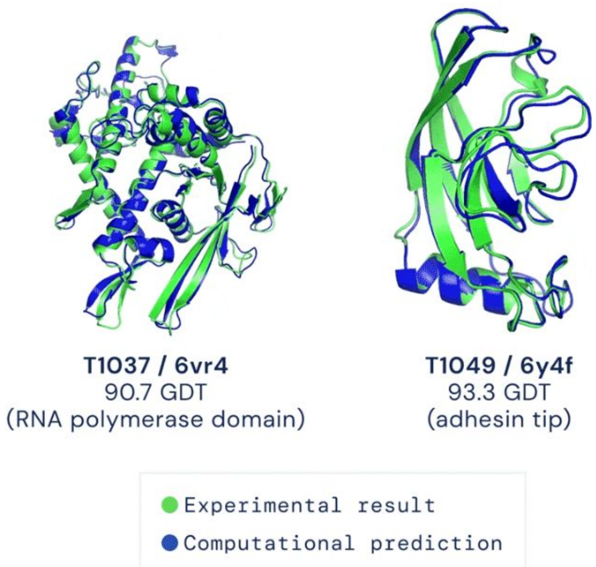


Image credit: Deep Mind

Example task 1: Protein folding

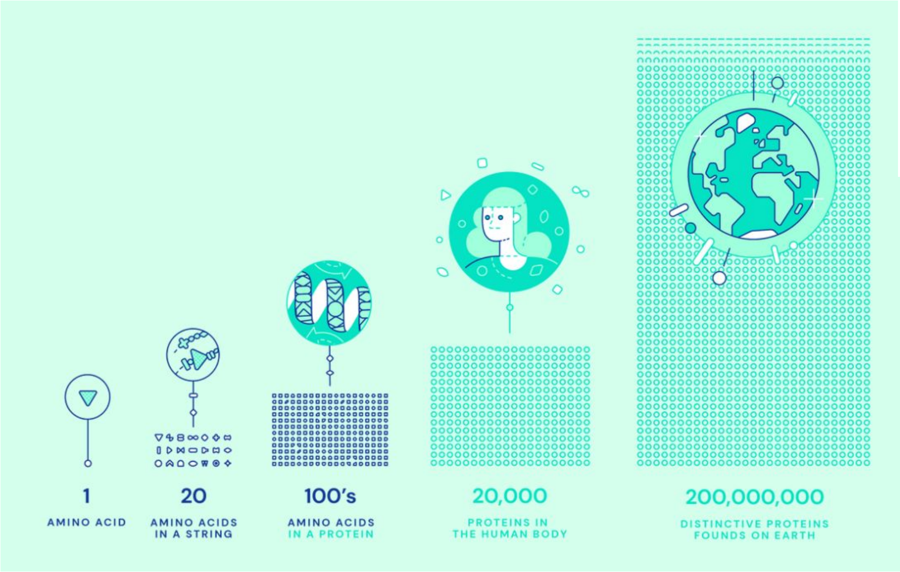
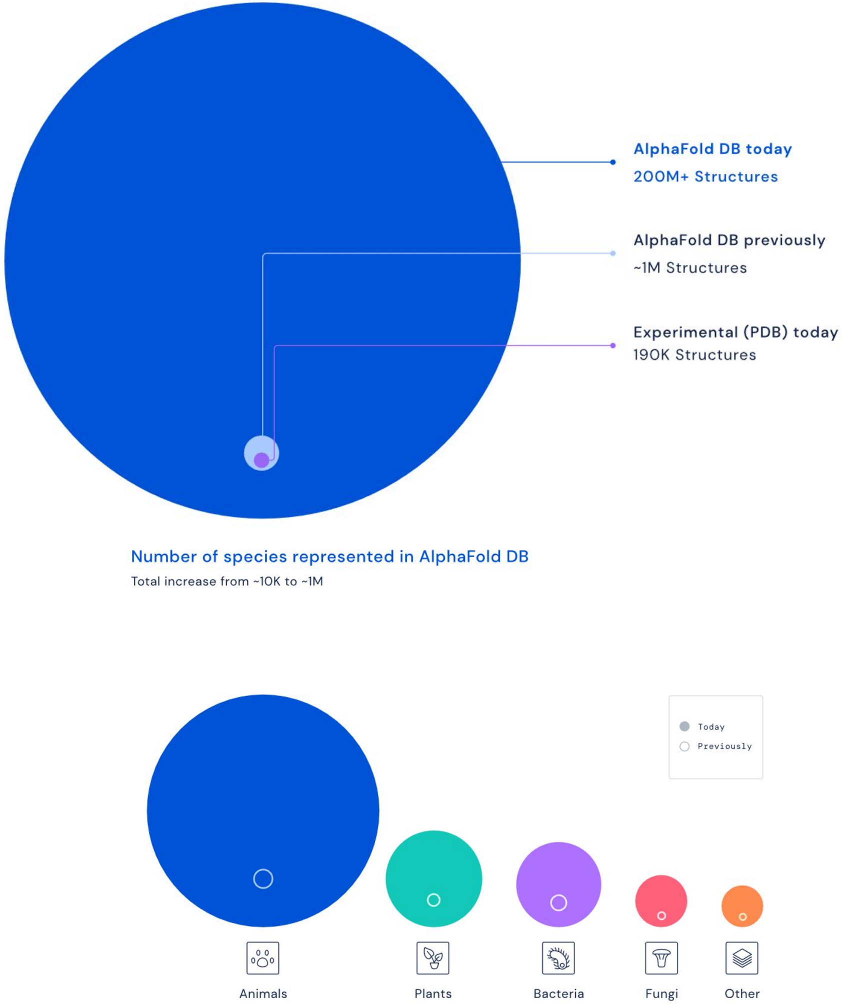


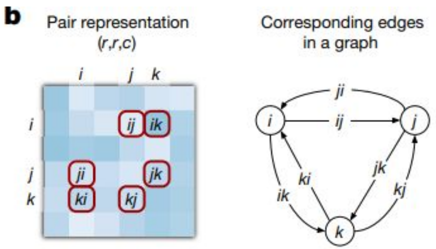
Image credit: Deep Mind

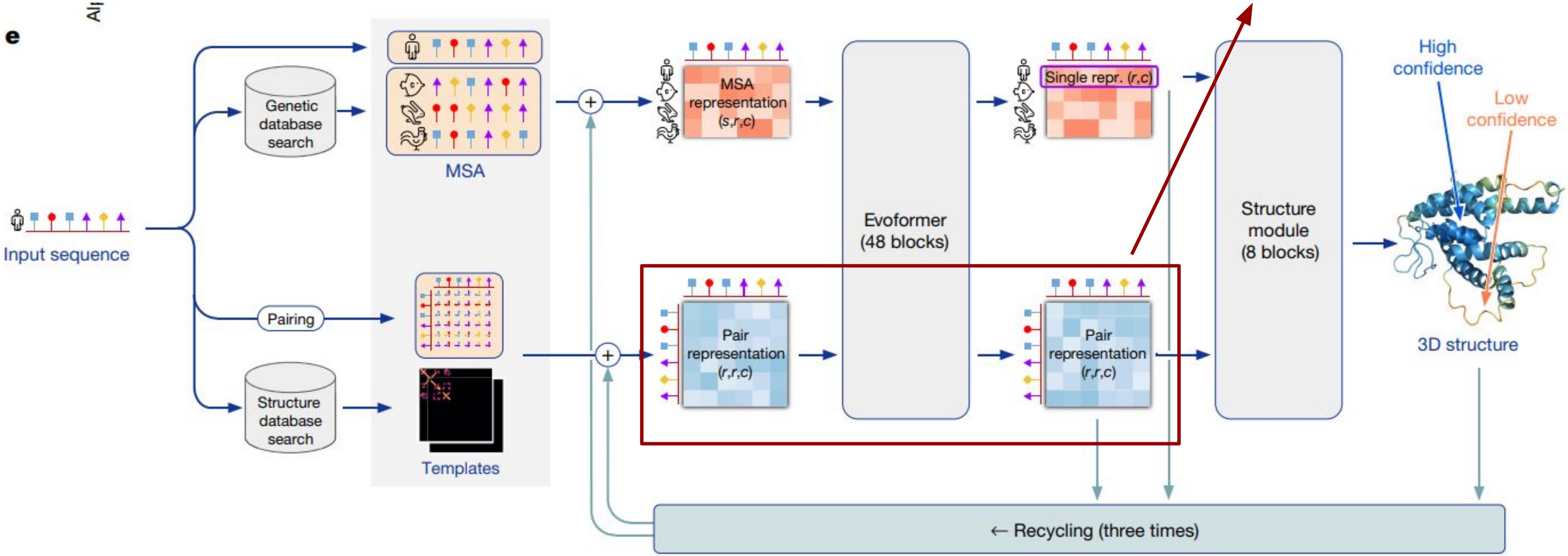


Example task 1: Protein folding

AlphaFold

Key concept: **Spatial graph**





Complete story: https://www.deepmind.com/research/highlighted-research/alphafold
Paper: https://www.nature.com/articles/s41586-021-03819-2

Example task 2: Recommendation systems

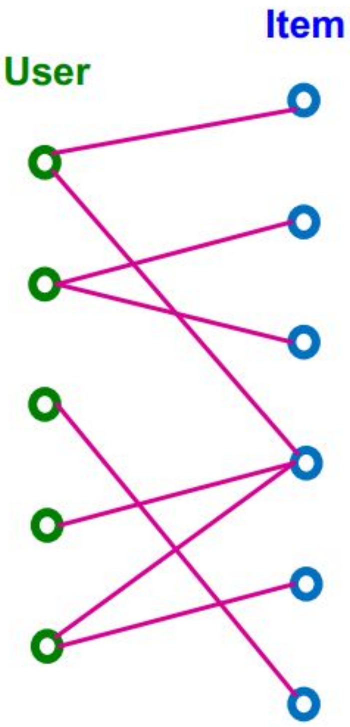


Image credit: Stanford CS224W

online shopping, music/movie recommendation

Nodes:
Users, items
Edges:
User-item interactions

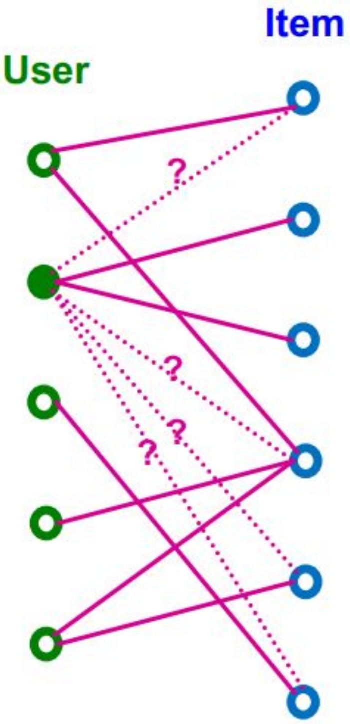
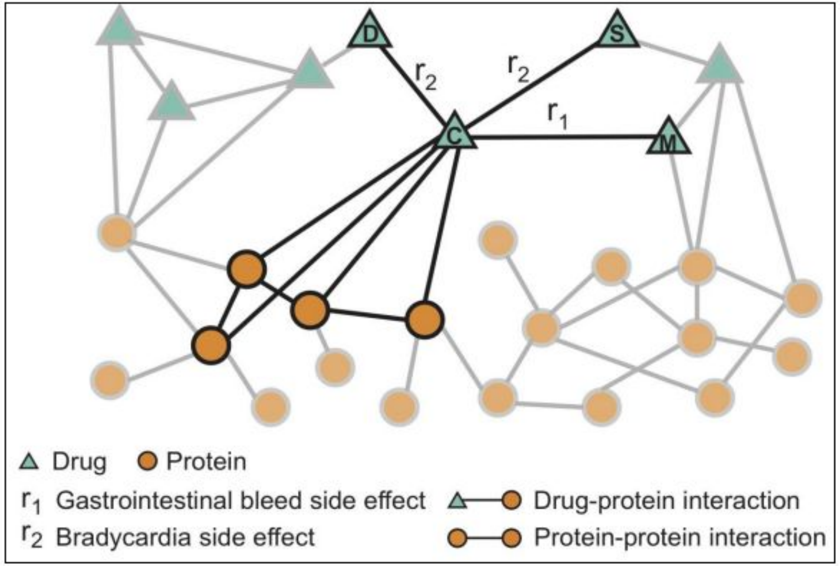


Image credit: Stanford CS224W

Example task 3: Drug adverse effect discovery

- **Nodes**: Drugs & Proteins
- **Edges**: Interactions



Query: How likely will Simvastatin and Ciprofloxacin, when taken together, break down muscle tissue?

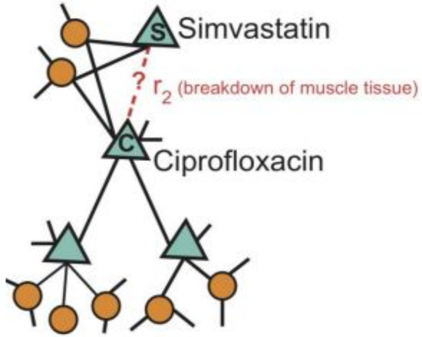
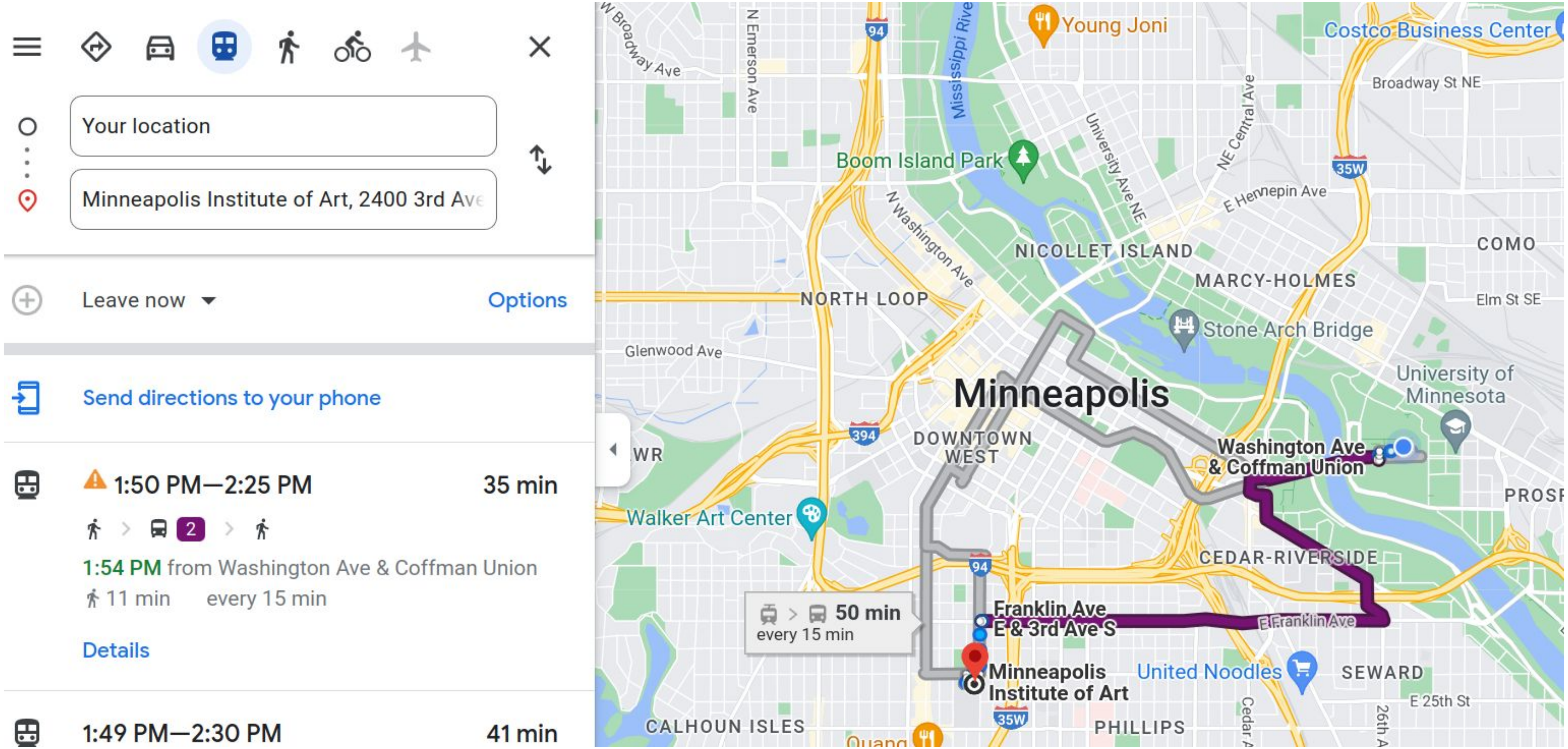


Image credit: Stanford CS224W

Example task 4: Traffic prediction



Example task 4: Traffic prediction

- **Nodes:** Road segments
- **Edges:** Connectivity between road segments
- **Prediction:** Time of Arrival (ETA)

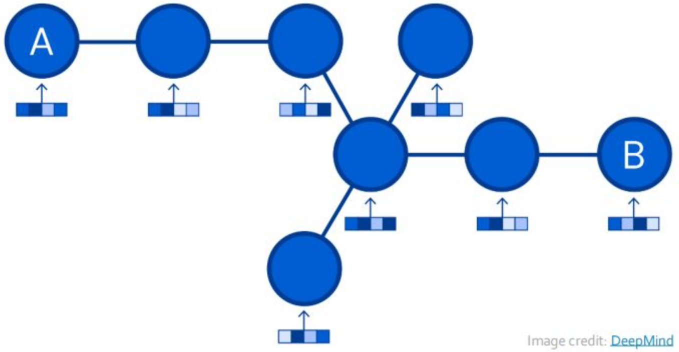


Image credit: Stanford CS224W

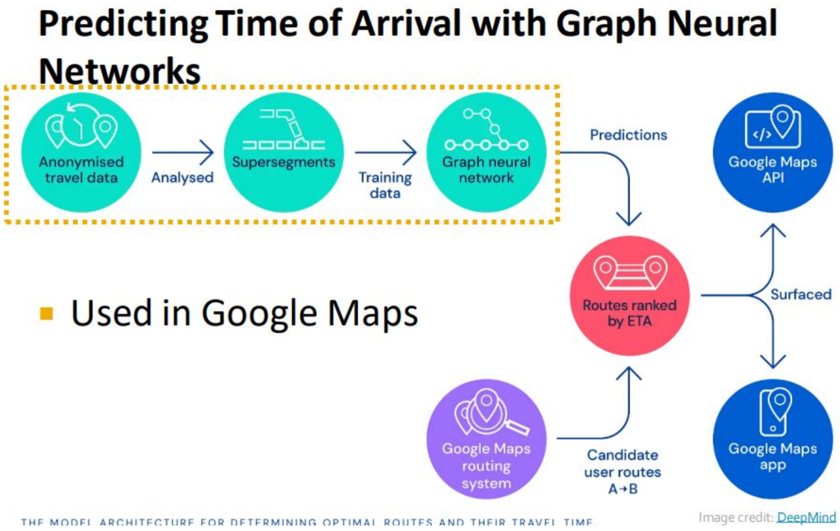
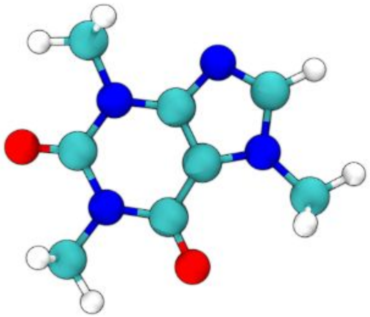


Image credit: Stanford CS224W

Subgraph discovery

Example task 5: Drug discovery



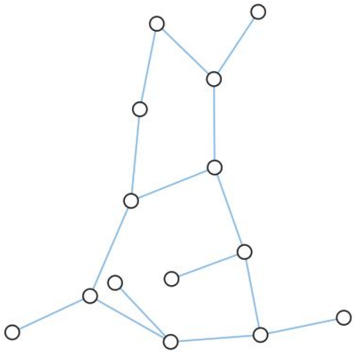
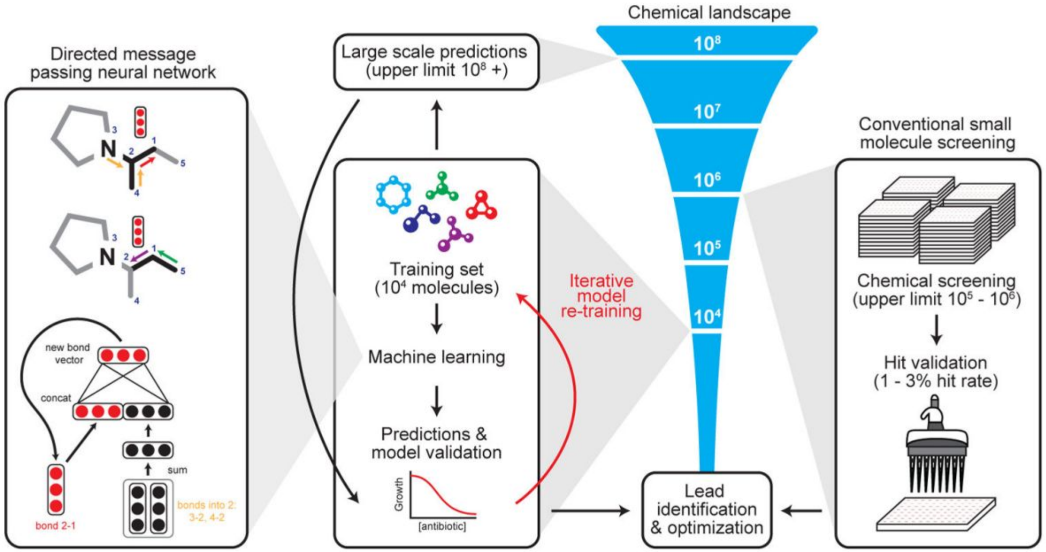


Image credit: https://distill.pub/2021/gnn-intro/

Molecules as graphs:
Nodes: atoms **Edges**: chemical bounds

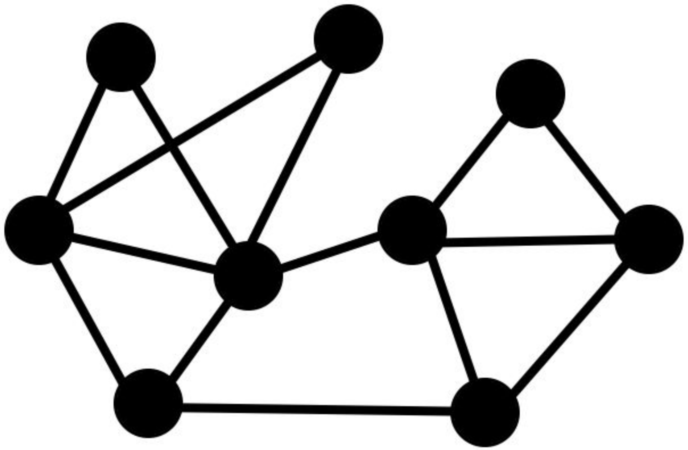


full-graph prediction

Image credit: https://pubmed.ncbi.nlm.nih.gov/32084340/

Graphs: basic notions

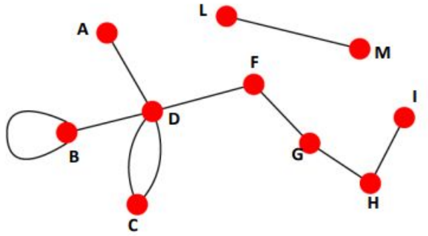
Basic objects



- N : Nodes (also vertices)
- E : Edges (also links)
- $G(N, E)$: Graph

Undirected

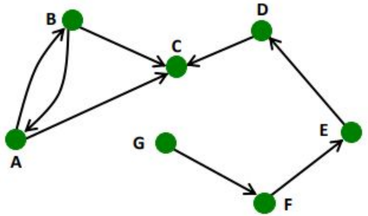
- **Links:** undirected (symmetrical, reciprocal)



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

Directed

- **Links:** directed (arcs)



- **Examples:**
  - Phone calls
  - Following on Twitter

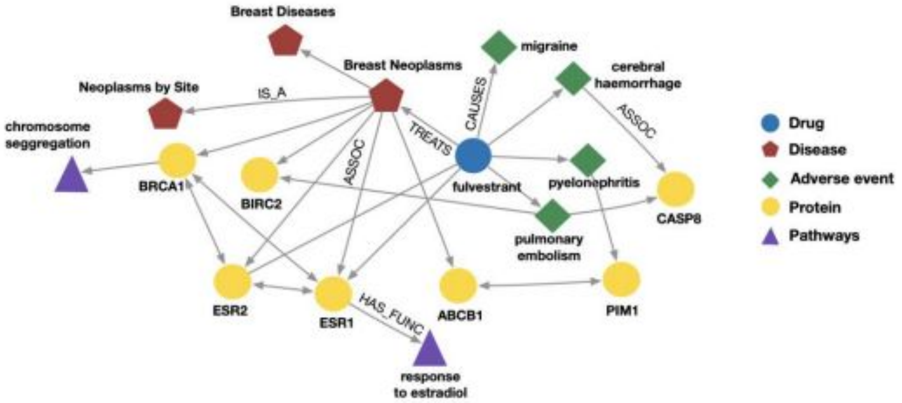
Image credit: Stanford CS224W

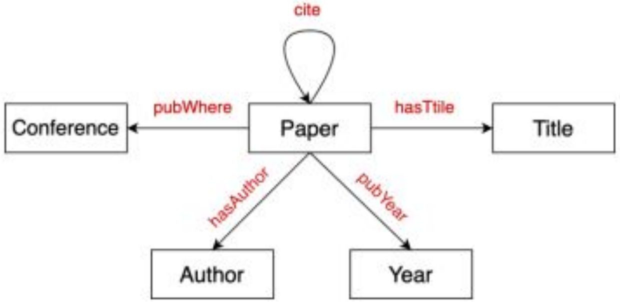
Heterogeneous graphs

- Nodes/Edges are multi-typed

$$G(N, E, T, R)$$

T: types of nodes
R: types of relationships





Biomedical Knowledge Graphs

Example node: Migraine
Example edge: (fulvestrant, Treats, Breast Neoplasms)
Example node type: Protein
Example edge type (relation): Causes

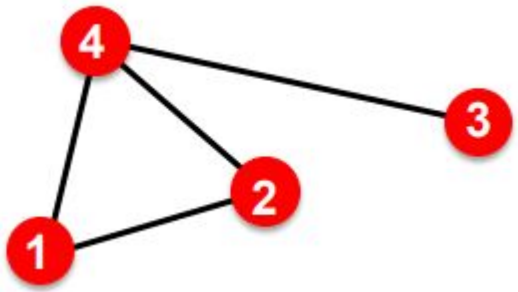
Academic Graphs

Example node: ICML
Example edge: (GraphSAGE, NeurIPS)
Example node type: Author
Example edge type (relation): pubYear

Image credit: Stanford CS224W

Graph representation

Undirected graphs



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

(1, 4), (1, 2)
(2, 1), (2, 4)
(3, 4)
(4, 1), (4, 2), (4, 3)

1: 2, 4
2: 1, 4
3: 4
4: 1, 2, 3

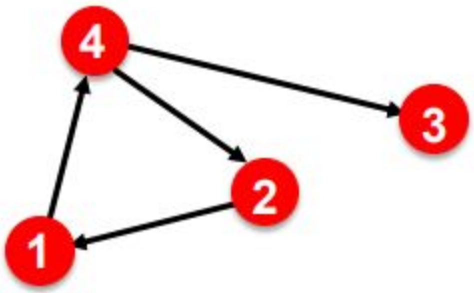
Adjacency matrix

Edge list

Adjacency list

Graph representation

Directed graphs



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

Adjacency matrix

(1, 4)
(2, 1)

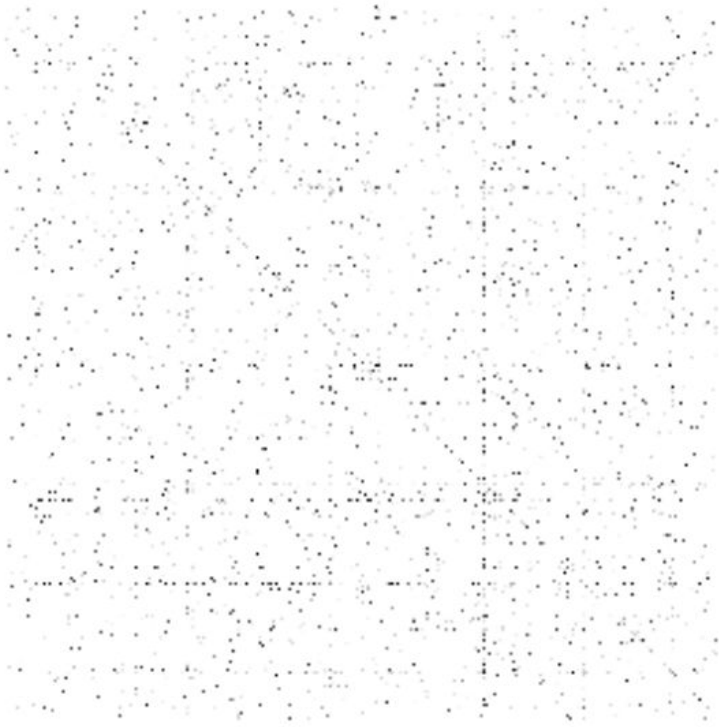
(4, 2), (4, 3)

Edge list

1: 4
2: 1
3:
4: 2, 3

Adjacency list

Adjacency matrix is often inefficient



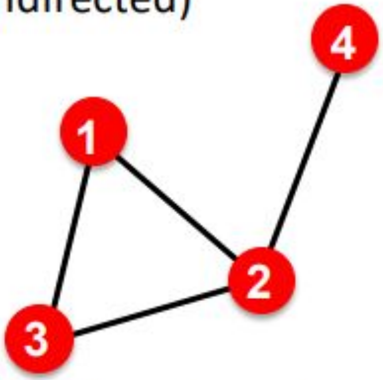
NETWORK	NODES	LINKS	DIRECTED/ UNDIRECTED	N	E
Internet	Routers	Internet connections	Undirected	192,244	609,066
WWW	Webpages	Links	Directed	325,729	1,497,134
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594
Phone Calls	Subscribers	Calls	Directed	36,595	91,826
Email	Email Addresses	Emails	Directed	57,194	103,731
Science Collaboration	Scientists	Co-authorship	Undirected	23,133	93,439
Actor Network	Actors	Co-acting	Undirected	702,388	29,397,908
Citation Network	Paper	Citations	Directed	449,673	4,689,479
E. Coli Metabolism	Metabolites	Chemical reactions	Directed	1,039	5,802
Protein Interactions	Proteins	Binding interactions	Undirected	2,018	2,930

Density = |E|/|N|^2

Image credit: Stanford CS224W

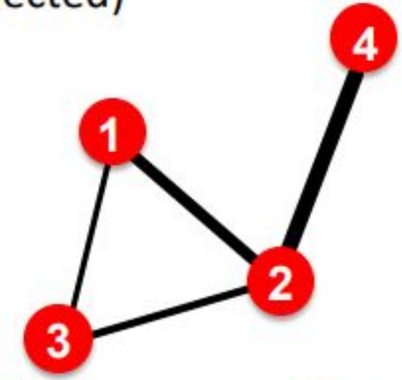
Weighted graphs

- **Unweighted**
  (undirected)



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

- **Weighted**
  (undirected)



$$A = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

Image credit: Stanford CS224W

Graph isomorphism/equivalence

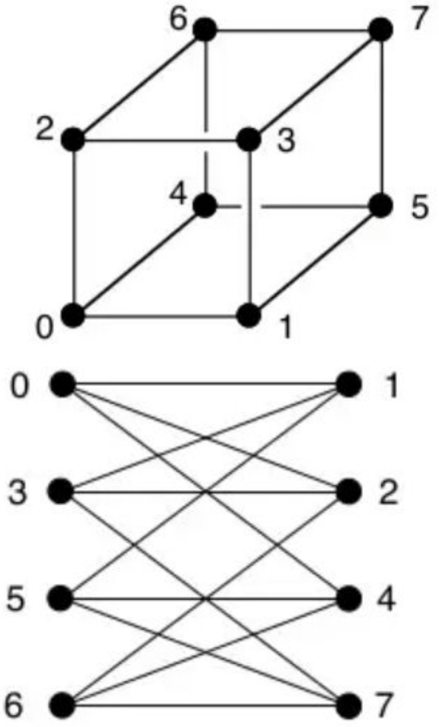


Image credit: https://tonicanada.medium.com/brute-force-code-for-isomorphisms-1241ef180570

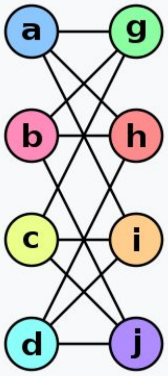
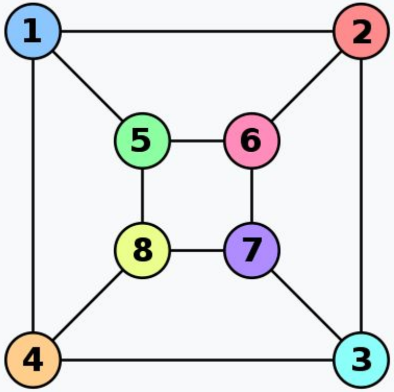
Graph G	Graph H	An isomorphism between G and H
		$f(a) = 1$ $f(b) = 6$ $f(c) = 8$ $f(d) = 3$ $f(g) = 5$ $f(h) = 2$ $f(i) = 4$ $f(j) = 7$

Image credit: Image credit: https://en.wikipedia.org/wiki/Graph_isomorphism

Isomorphism: there exists a bi-injective mapping, i.e., **permutation**, results in the same neighborhood structure

Permutation invariance

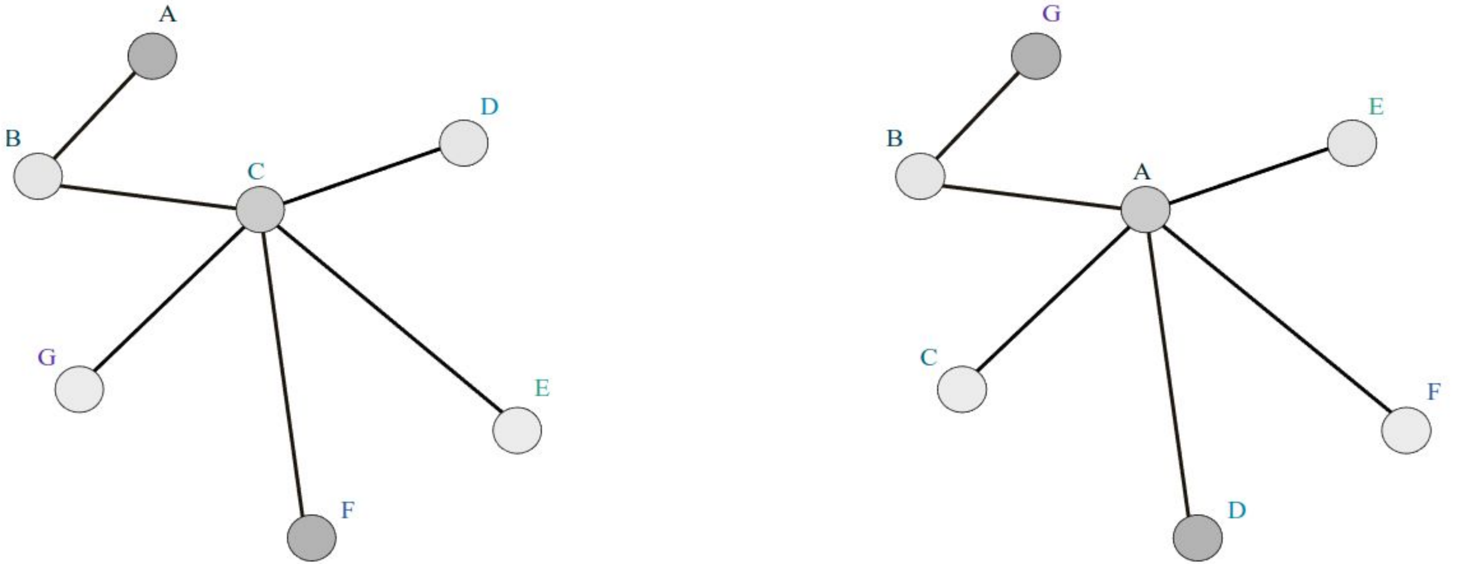
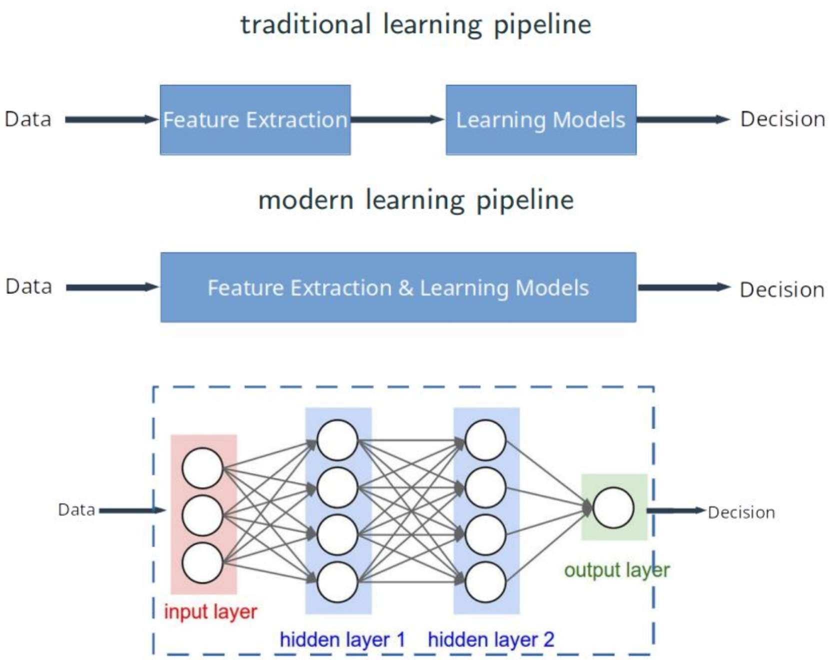


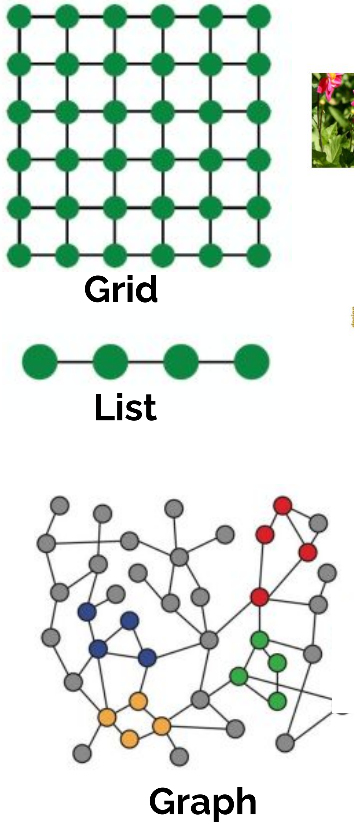
Image credit: Image credit: https://distill.pub/2021/understanding-gnns/

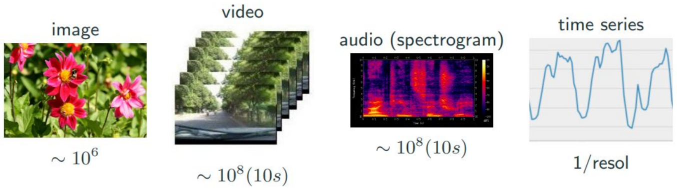
Permutation invariance: permuting the names of the nodes doesn't change the graph, as graph nodes are intrinsically orderless

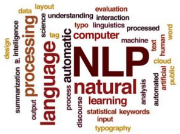
Graph neural networks

Representation learning for graphs

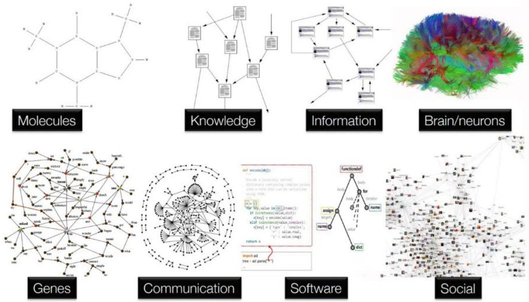




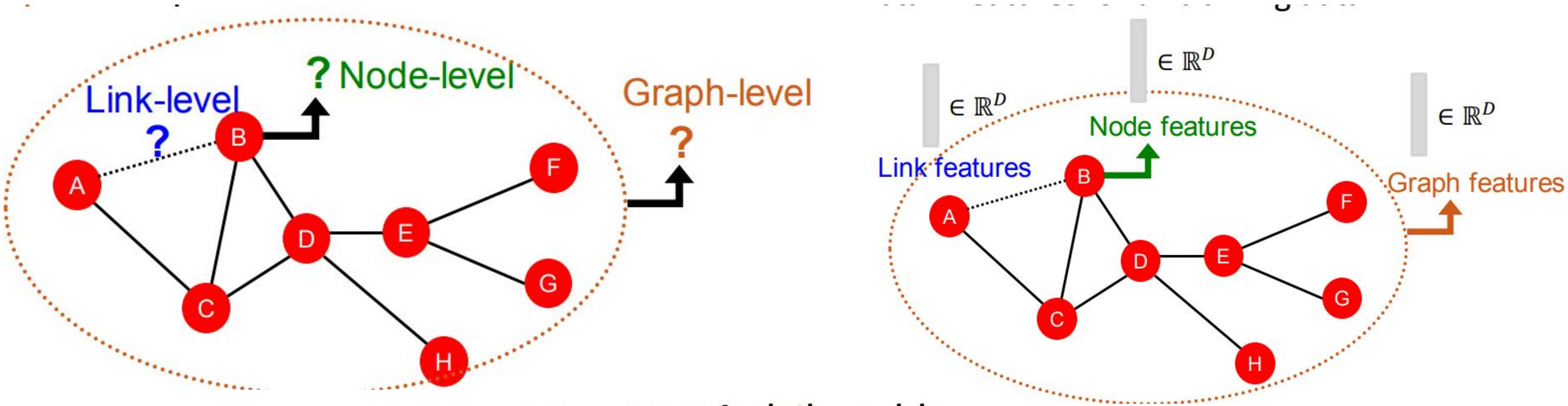




- machine translation, e.g., English ⇆ Chinese
- typing/writing prediction (smart compose)
- semantic classification



Where to put the features?



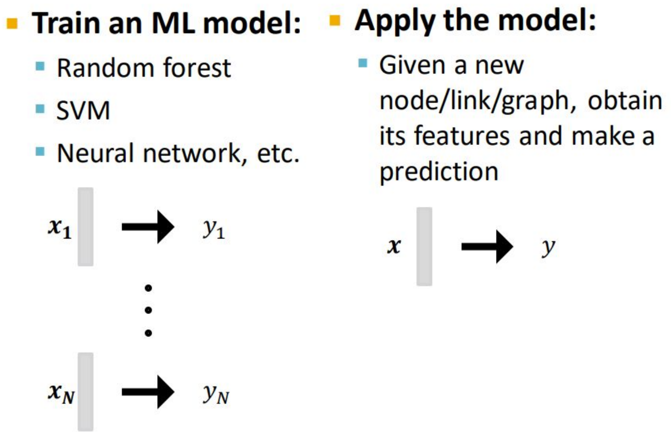


Image credit: Stanford CS224W

Node embedding

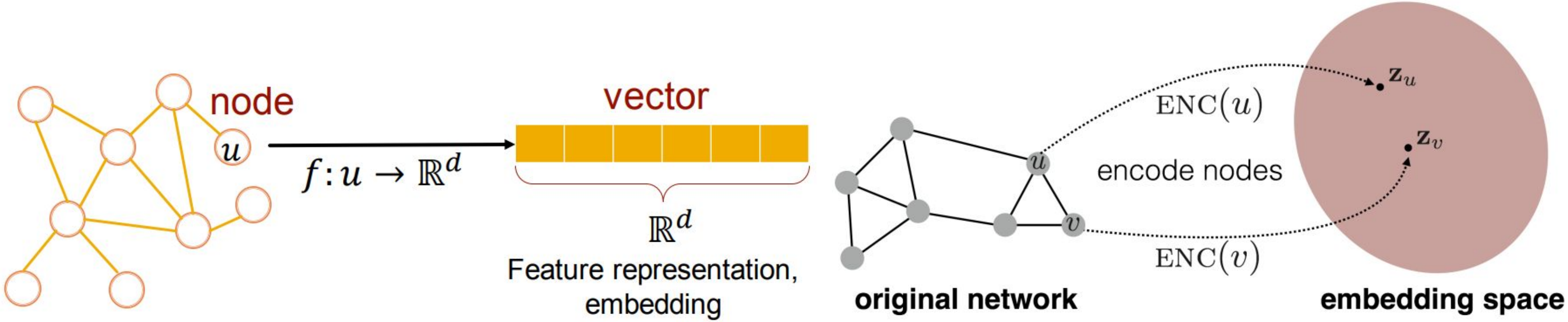


Image credit: Stanford CS224W

N : set of nodes

A : adjacency matrix

$\boldsymbol{X} \in \mathbb{R}^{|N| \times d}$: node (raw) features

u, v : nodes in N

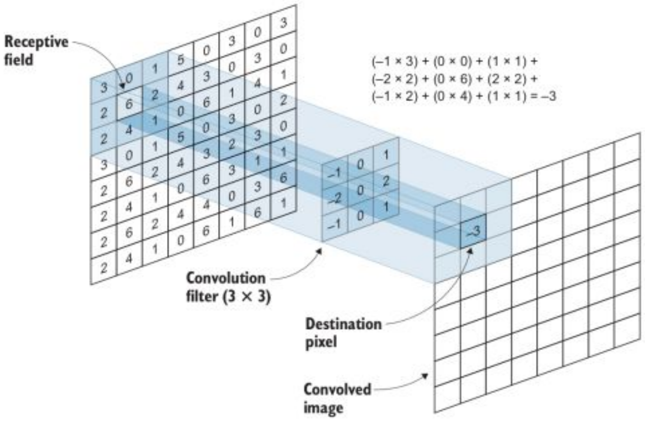
$\mathcal{N}(u)$: neighbors of u

Node raw features: e.g.,

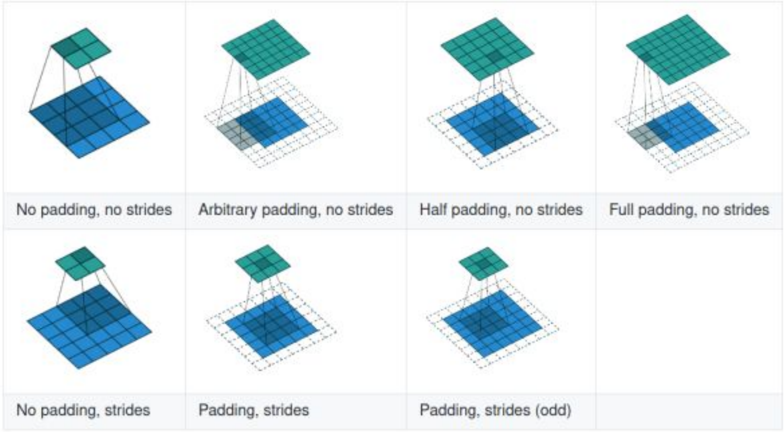
- Biomedical graphs: patient's EHR
- Social network graphs: user profile and images
- When no features: node indicator vector, constant vector

How to define the f ?

We'll bypass fully connected networks directly



(Credit: [Elgendy, 2020])



https://github.com/vdumoulin/conv_arithmetic

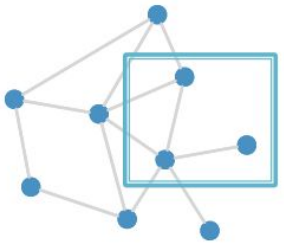
Convolution as performing **local info aggregation** (or **message passing**):

- Each time, the conv window focus on a **local neighborhood** of the current pixel
- Conv effectively **aggregates** the local info by **weighted summation**

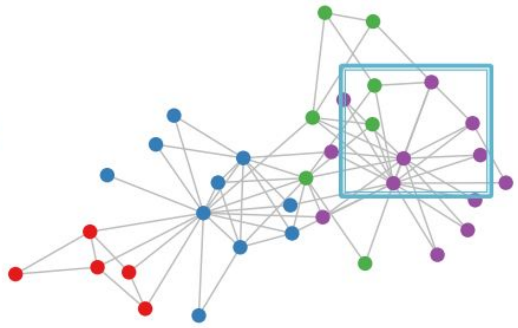
How to define the f ?

Convolution as performing **local info aggregation** (or **message passing**):

- **Neighborhood**: Each time, the conv window focus on a **local neighborhood** of the current pixel
- **Aggregation**: Conv effectively **aggregates** the local info by **weighted summation**



or this:



The rigid, grid-based neighborhood doesn't work!

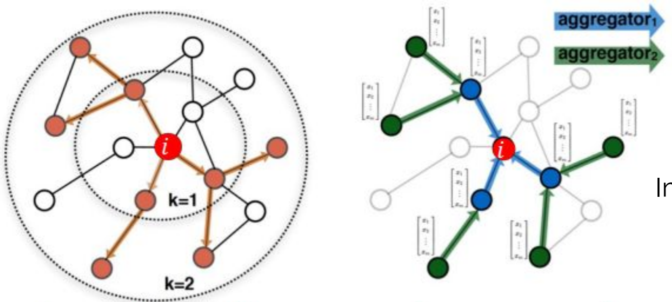
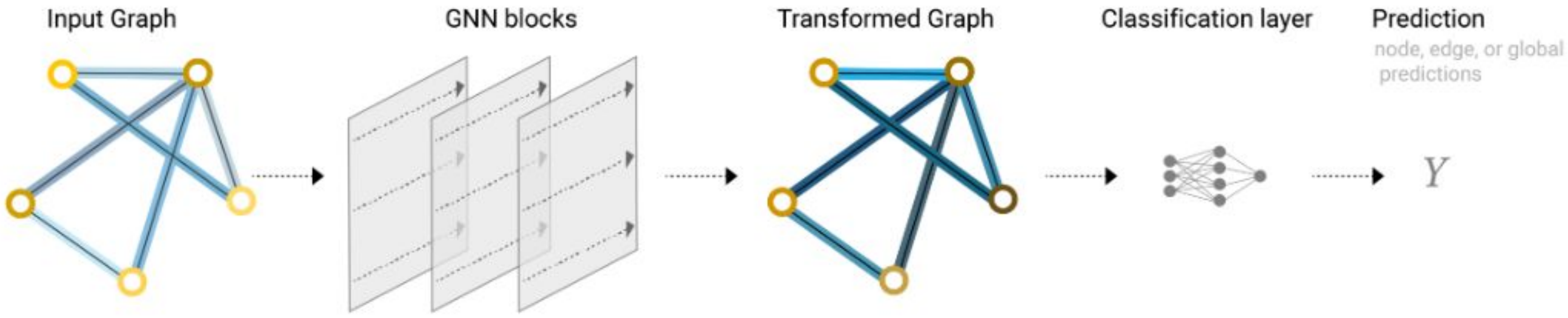


Image credit: Stanford CS224W

We need neighbors on the graph!

One layer of a graph neural network (GNN)!

Graph in, graph out

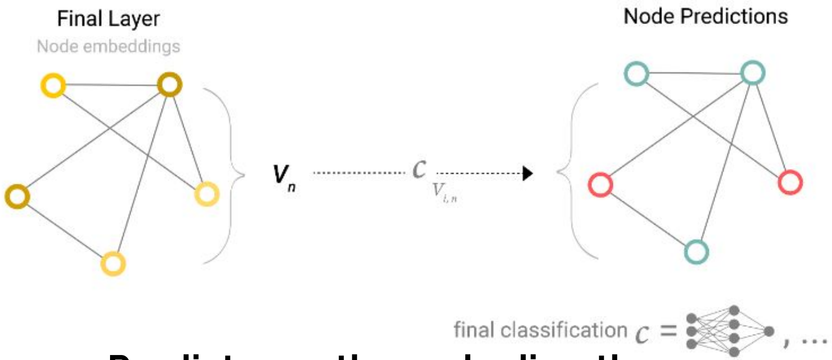


An end-to-end prediction task with a GNN model.

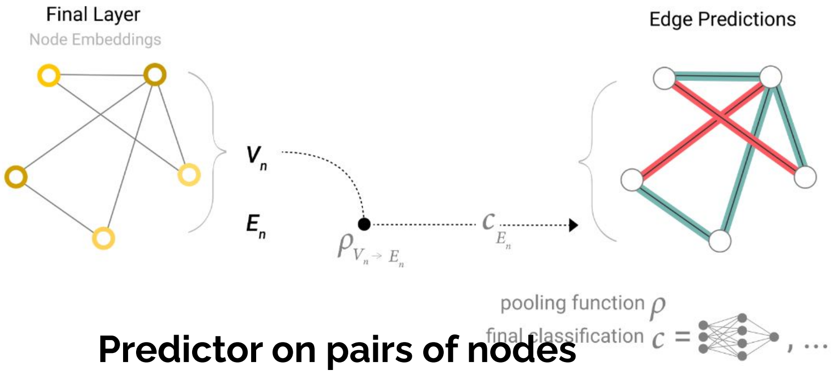
Image credit: https://distill.pub/2021/gnn-intro/

How to make supervised predictions?

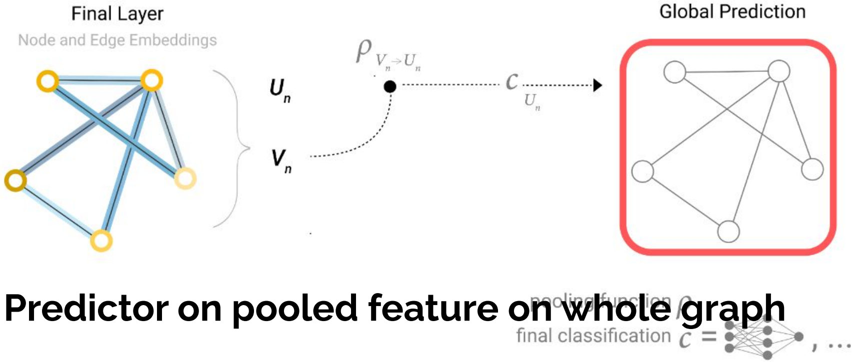
Image credit: https://distill.pub/2021/gnn-intro/



Predictor on the node directly

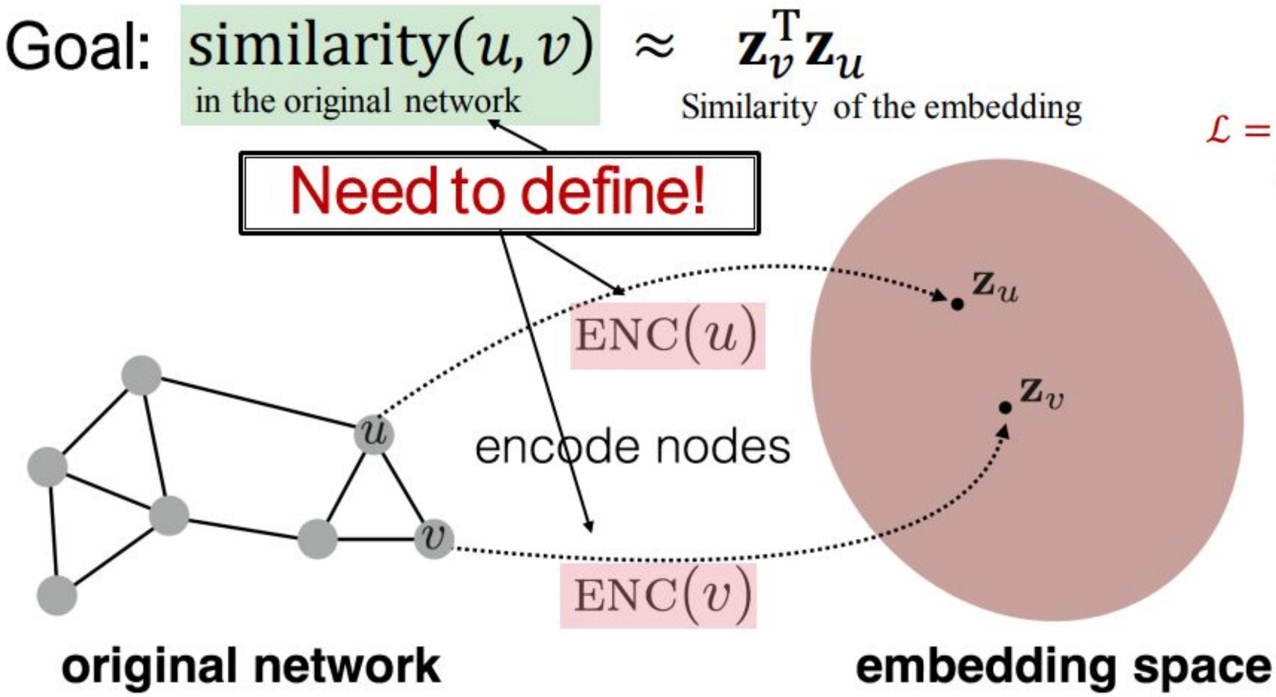


Predictor on pairs of nodes



Predictor on pooled feature on whole graph

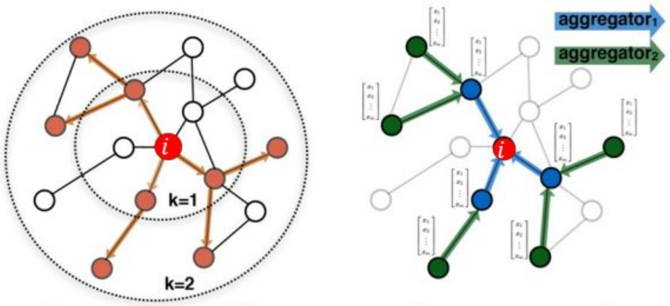
How to perform unsupervised learning?



$$\mathcal{L} = \sum_{z_u, z_v} \mathrm{CE}(y_{u,v}, \mathrm{DEC}(z_u, z_v))$$

Image credit: Stanford CS224W

Look into GNN layers



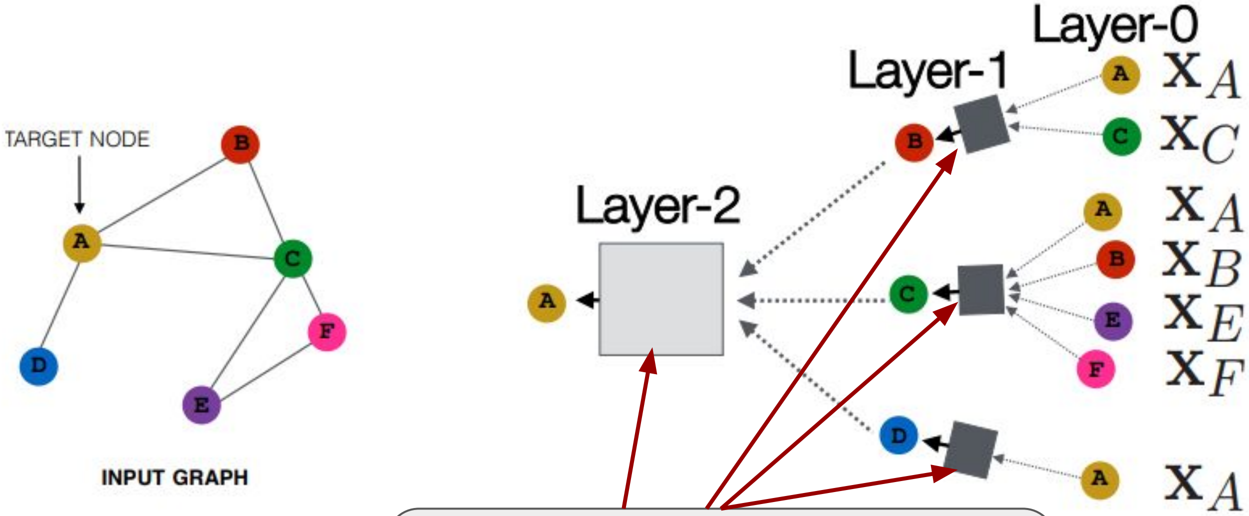


Image credit: Stanford CS224W

These **aggregators** sum up neighboring info, and can be represented by **DNNs**

Permutation invariance

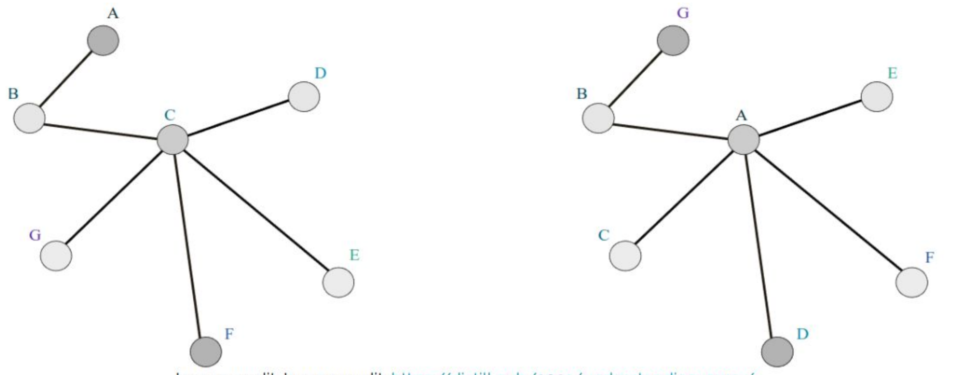


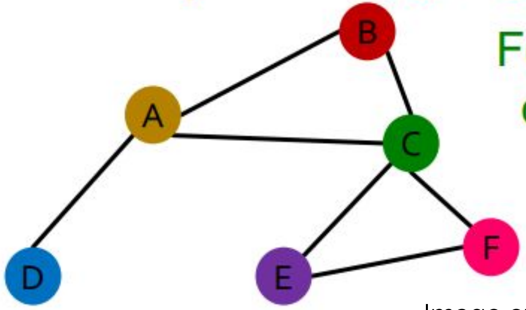
Image credit: Image credit: https://distill.pub/2021/understanding-gnns/

Permutation invariance: permuting the names of the nodes doesn't change the graph, as graph nodes are intrinsically orderless

For $f : G(\boldsymbol{A}, \boldsymbol{X}) \mapsto \boldsymbol{h}$

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

Order plan 1: A_1, X_1



For two order plans, output of f should be the same!

Order plan 2: A_2, X_2

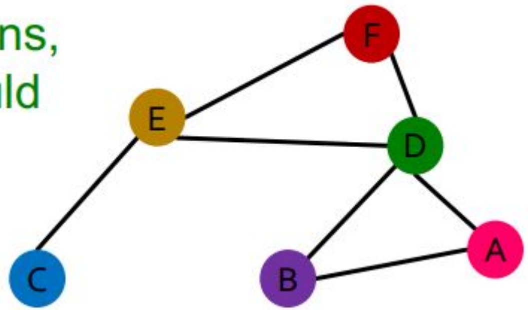


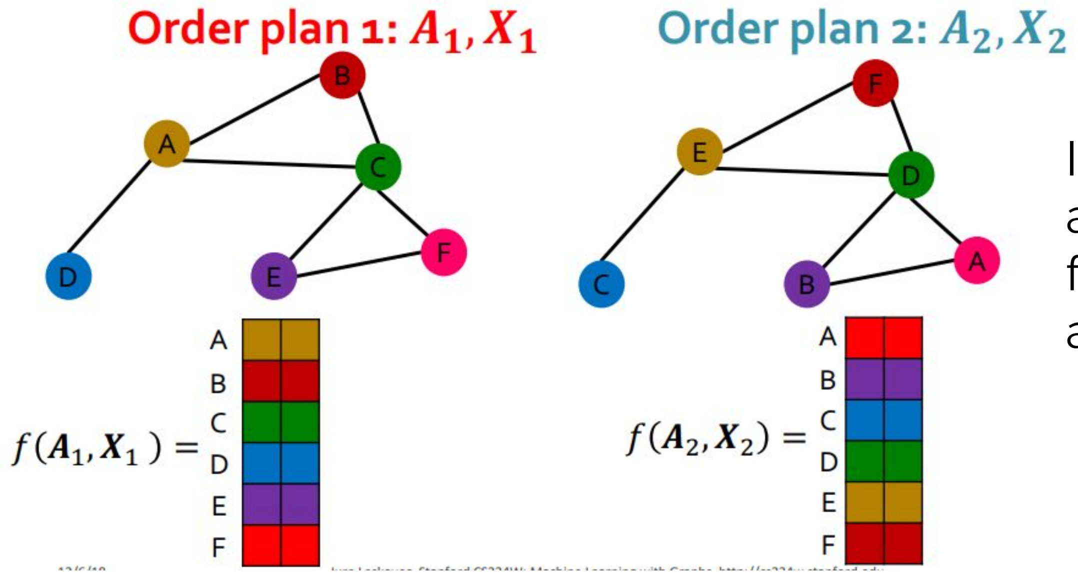
Image credit: Stanford CS224W

Permutation equivariance

For $f : G(\boldsymbol{A}, \boldsymbol{X}) \mapsto \boldsymbol{H} \in \mathbb{R}^{|N| \times d}$

$$f(\boldsymbol{\Pi} \boldsymbol{A} \boldsymbol{\Pi}^{\intercal}, \boldsymbol{\Pi} \boldsymbol{X}) = \boldsymbol{\Pi} f(\boldsymbol{A}, \boldsymbol{X})$$

for any permutation $\boldsymbol{\Pi}$



In other words, if the nodes are re-ordered, the learned features are re-ordered accordingly

Image credit: Stanford CS224W

A typical GNN consists of multiple permutation equivariant/invariant layers

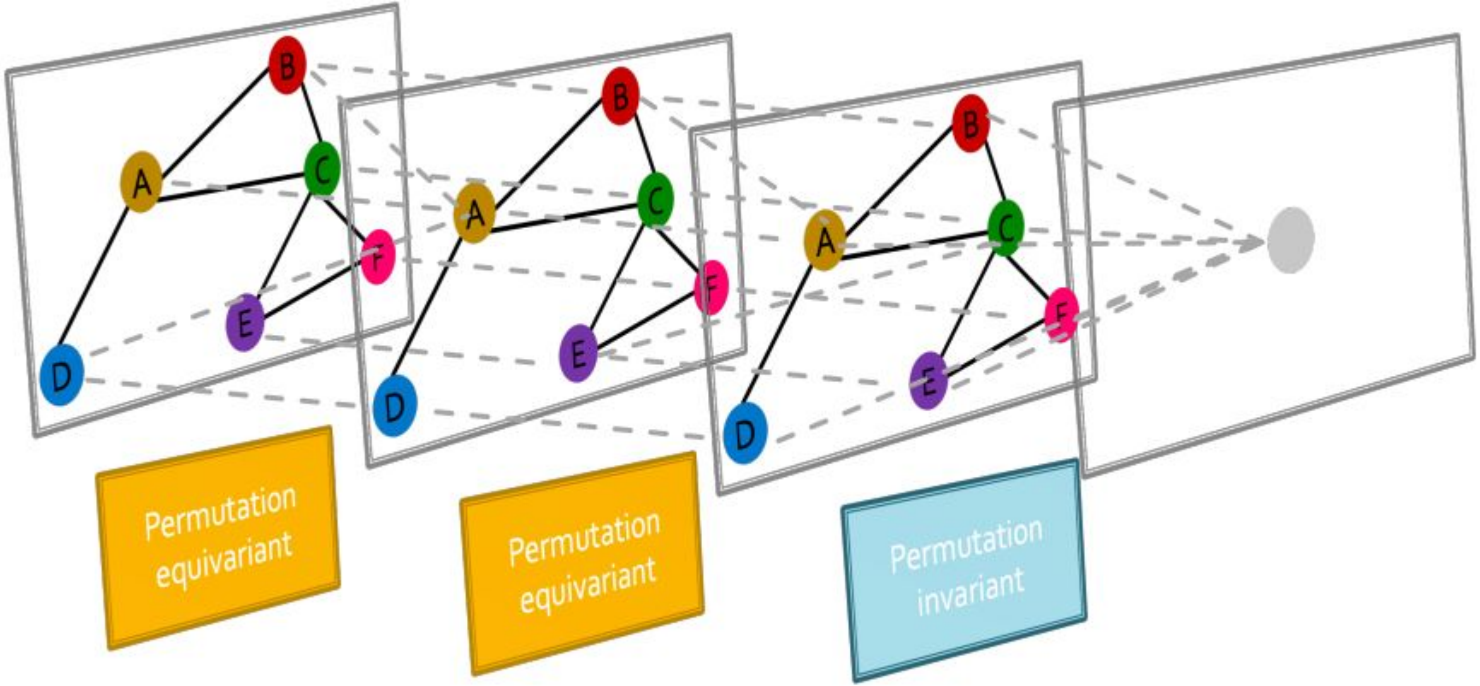


Image credit: Stanford CS224W

Graph convolutional networks (GCNs)

$$h_v^{(0)} = x_v \quad \text{for all } v \in V.$$

Node v's initial embedding.

... is just node v's original features.

and for $k = 1, 2, \ldots$ upto K:

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \frac{\sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}}{|\mathcal{N}(v)|} + B^{(k)} \cdot h_v^{(k-1)}\right) \quad \text{for all } v \in V.$$

Node v's embedding at step k.

Mean of v's neighbour's embeddings at step $k-1$.

Node v's embedding at step $k-1$.

Color Codes:

- Embedding of node v.
- Embedding of a neighbour of node v.
- (Potentially) Learnable parameters.

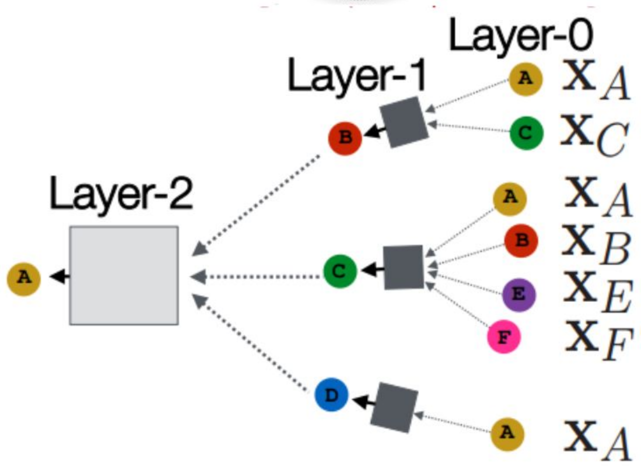


Image credit: https://distill.pub/2021/understanding-gnns/

Graph attention networks (GATs)

$$h_v^{(0)} = x_v \quad \text{for all } v \in V.$$

Node v's initial embedding. ... is just node v's original features.

and for $k = 1, 2, \ldots$ upto K:

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \left[\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^{(k-1)} h_u^{(k-1)} + \alpha_{vv}^{(k-1)} h_v^{(k-1)}\right]\right) \quad \text{for all } v \in V.$$

Node v's embedding at step k.

Weighted mean of v's neighbour's embeddings at step $k-1$.

Node v's embedding at step $k-1$.

where the attention weights $\alpha^{(k)}$ are generated by an attention mechanism $A^{(k)}$, normalized such that the sum over all neighbours of each node v is 1:

$$\alpha_{vu}^{(k)} = \frac{A^{(k)}(h_v^{(k)}, h_u^{(k)})}{\sum_{w \in \mathcal{N}(v)} A^{(k)}(h_v^{(k)}, h_w^{(k)})} \quad \text{for all } (v, u) \in E.$$

Color Codes:

- Embedding of node v.
- Embedding of a neighbour of node v.
- (Potentially) Learnable parameters.

Image credit: https://distill.pub/2021/understanding-gnns/

Graph sample and aggregate (GraphSAGE)

$$h_v^{(0)} = x_v \quad \text{for all } v \in V.$$

Node v's initial embedding. ... is just node v's original features.

and for $k = 1, 2, \ldots$ upto K:

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \left[\underset{u \in \mathcal{N}(v)}{\text{AGG}}(\{h_u^{(k-1)}\}),\ h_v^{(k-1)}\right]\right) \quad \text{for all } v \in V.$$

Node v's embedding at step k.

Aggregation of v's neighbour's embeddings at step $k-1$ ...

... concatenated with ...

... Node v's embedding at step $k-1$.

Color Codes:

- Embedding of node v.
- Embedding of a neighbour of node v.
- (Potentially) Learnable parameters.

Image credit: https://distill.pub/2021/understanding-gnns/

Graph isomorphism networks (GINs)

$$h_v^{(0)} = x_v \quad \text{for all } v \in V.$$

Node v's initial embedding. ... is just node v's original features.

and for $k = 1, 2, \ldots$ upto K:

$$h_v^{(k)} = f^{(k)}\left(\sum_{u \in \mathcal{N}(v)} h_u^{(k-1)} + \left(1 + \epsilon^{(k)}\right) \cdot h_v^{(k-1)}\right) \quad \text{for all } v \in V.$$

Node v's embedding at step k.

Sum of v's neighbour's embeddings at step $k-1$.

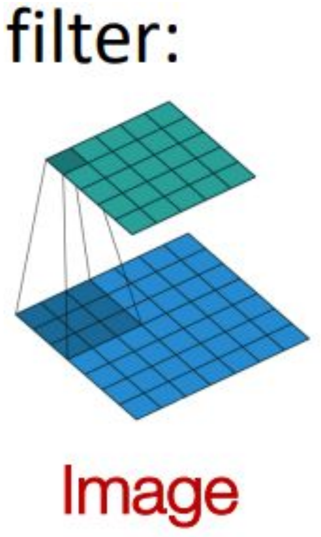
Node v's embedding at step $k-1$.

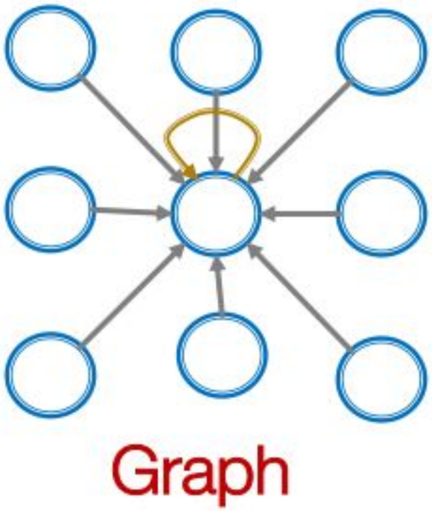
Color Codes:

- Embedding of node v.
- Embedding of a neighbour of node v.
- (Potentially) Learnable parameters.

Image credit: https://distill.pub/2021/understanding-gnns/

Connection to CNNs and Transformers





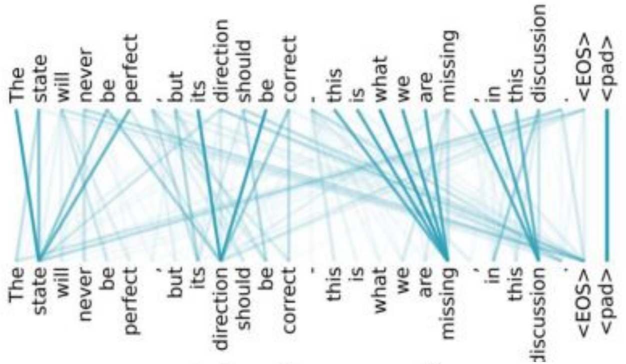
- CNN is GNN that keeps local ordering
- CNN not permutation-invariant

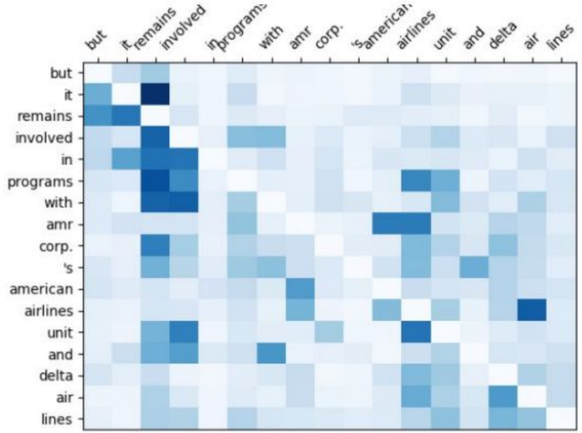
GNN formulation: $\mathrm{h}_v^{(l+1)} = \sigma\left(\mathbf{W}_l \sum_{u \in \mathrm{N}(v)} \frac{\mathrm{h}_u^{(l)}}{|\mathrm{N}(v)|} + \mathrm{B}_l \mathrm{h}_v^{(l)}\right), \forall l \in \{0, \ldots, L-1\}$

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

Image credit: Stanford CS224W

Connection to CNNs and Transformers





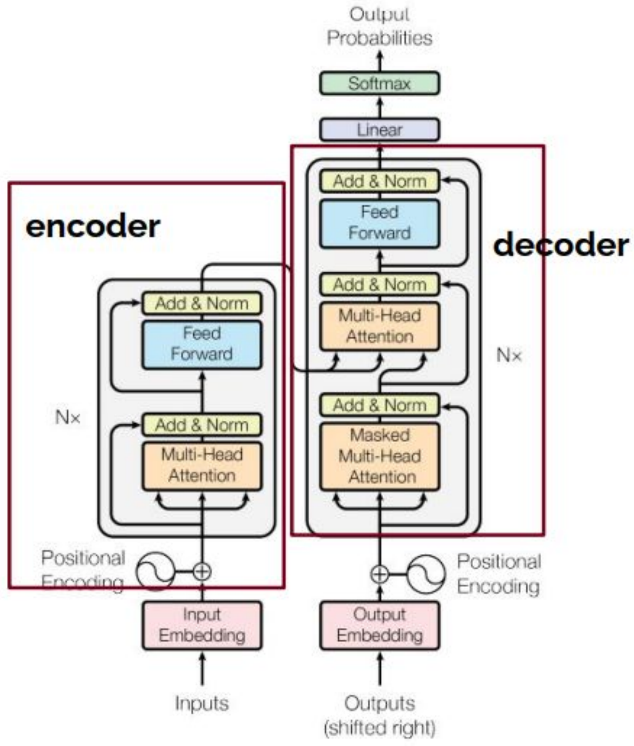


Figure 1: The Transformer - model architecture.

Self-attention (plus feed forward) is a layer of GAT on a complete graph!

Scaling up training

Practical graphs are large yet sparse

- **Knowledge Graphs (KGs):**
  - Wikidata
  - Freebase
- **ML tasks:**
  - KG completion
  - Reasoning

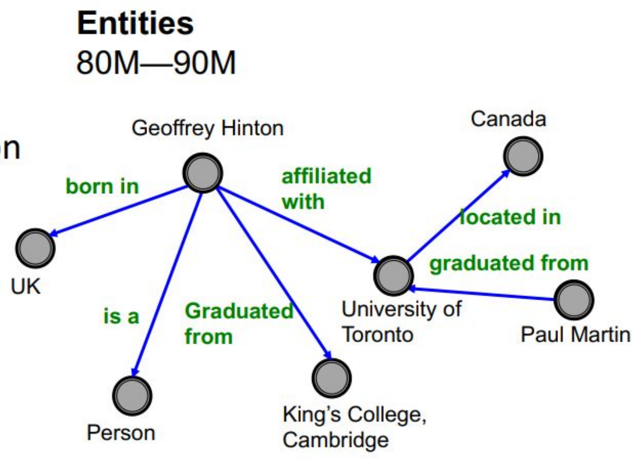


Image credit: Stanford CS224W

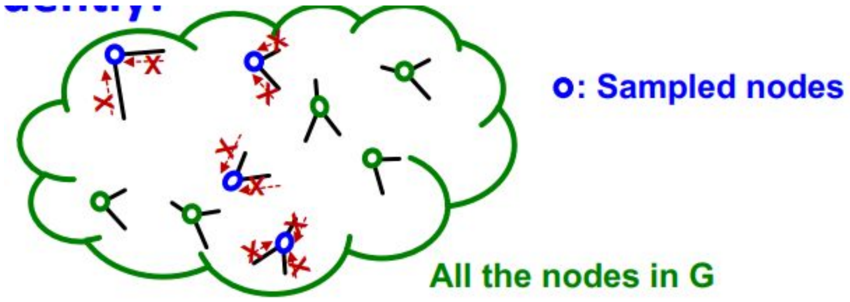
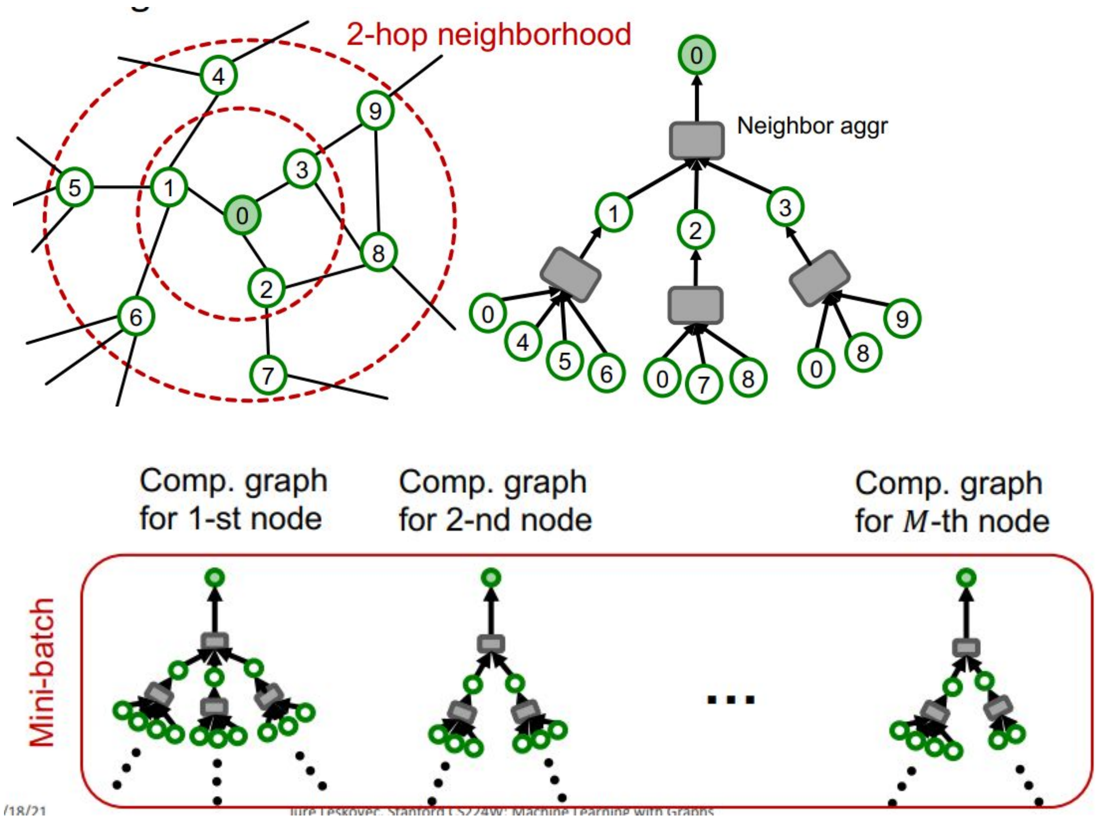
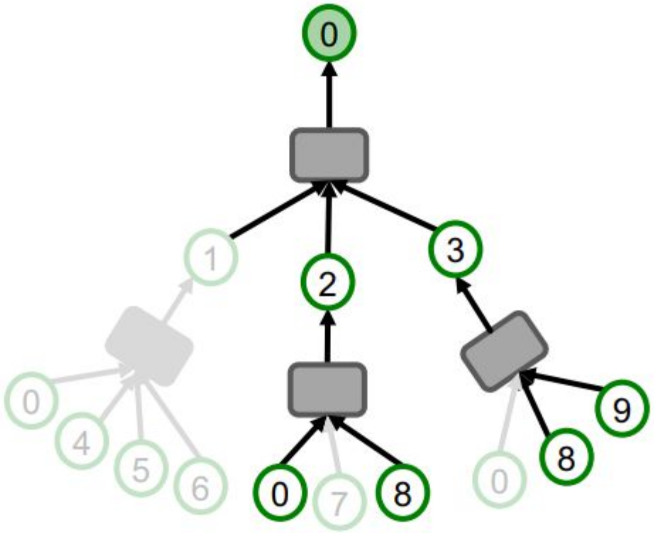


Image credit: Stanford CS224W

- Mini-batch subsampling induces isolated nodes
- No info to aggregate inside the mini-batch for most nodes

Two careful sub-sampling strategies

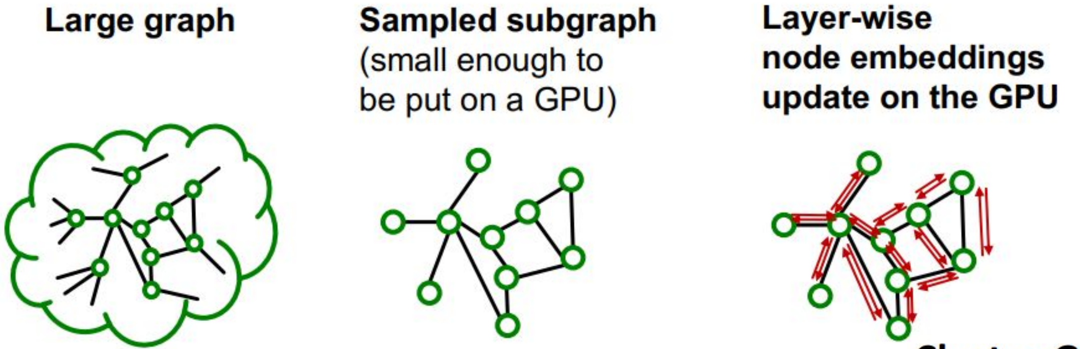




Neighborhood sampling

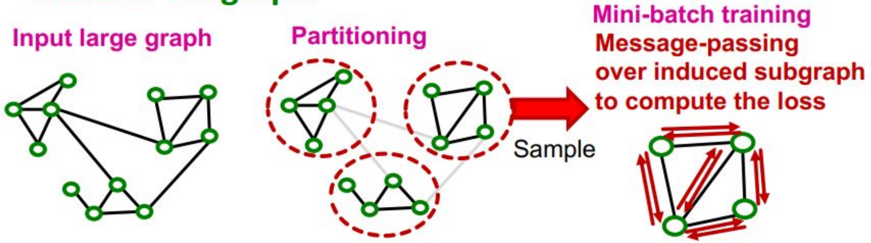
Image credit: Stanford CS224W

Two careful sub-sampling strategies



Cluster-GCN consists of two steps:

- **Pre-processing**: Given a large graph, partition it into groups of nodes (i.e., subgraphs).
- **Mini-batch training**: Sample one node group at a time. Apply GNN's message passing over the **induced subgraph**.



Rationale: important to keep the community structures, i.e., keep the "backbone" nodes

Image credit: Stanford CS224W

Software

PyTorch Geometric (PyG)

Deep Graph Library (DGL)

https://pytorch-geometric.readthedocs.io/en/latest/

https://www.dgl.ai/

Further reading

- What are graph neural networks?
  https://blogs.nvidia.com/blog/2022/10/24/what-are-graph-neural-networks/
- A Gentle Introduction to Graph Neural Networks
  https://distill.pub/2021/gnn-intro/
- Understanding Convolutions on Graphs
  https://distill.pub/2021/understanding-gnns/
- Graph Neural Networks: A Review of Methods and Applications
  https://arxiv.org/abs/1812.08434
- Stanford CS224W: Machine Learning with Graphs
  https://web.stanford.edu/class/cs224w/index.html
- Graph Representation Learning https://www.cs.mcgill.ca/~wlh/grl_book/