

Relationship Modeling: Graph Neural Networks

Ju Sun

Computer Science & Engineering

Dec 13, 2022



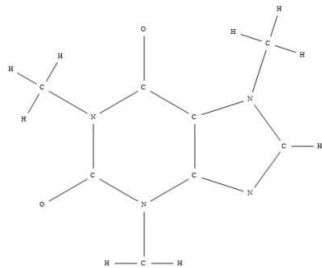
UNIVERSITY OF MINNESOTA
Driven to DiscoverSM

Outline

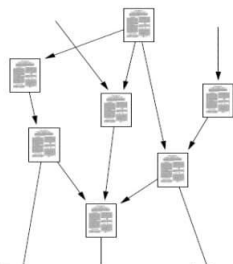
- Why graphs?
- Graphs: basic notions
- Graph neural networks
- Scaling up training

Why graphs?

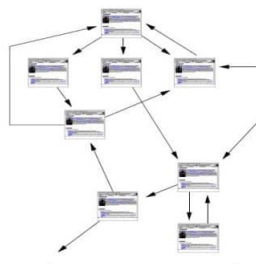
Graphs are everywhere!



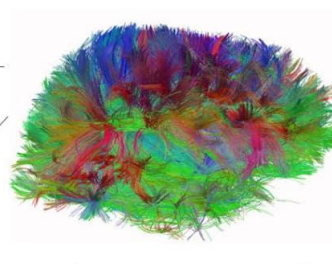
Molecules



Knowledge



Information



Brain/neurons

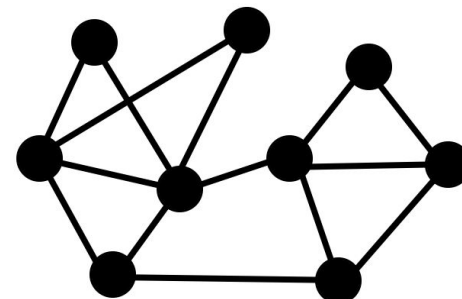
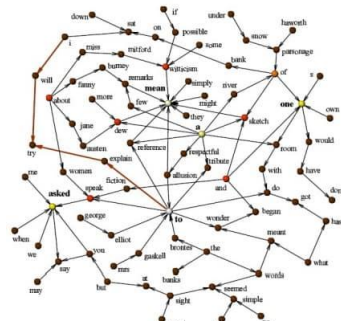
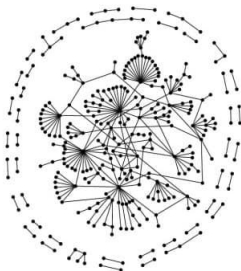


Image credit: Stanford CS224W



Genes

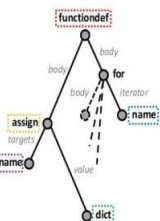


Communication

```
def encode(obj):
    """
    Encode a (possibly nested)
    dictionary containing complex values
    into a form that can be serialized
    using JSON.
    """
    if isinstance(obj, dict):
        e = {}
        for key, value in obj.items():
            if isinstance(value, dict):
                e[key] = encode(value)
            elif isinstance(value, complex):
                e[key] = {'type': 'complex',
                        'r': value.real,
                        'i': value.imag}
        return e
    elif isinstance(obj, list):
        e = []
        for value in obj:
            e.append(encode(value))
        return e
    else:
        return obj

import ast
tree = ast.parse("""
...
""")
```

Software



Social

Graphs model
**relationships/
interactions**

Different tasks on graphs

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

Graph prediction (classification/regression), graph generation

A graph is a data point

Graph-level prediction,
Graph generation

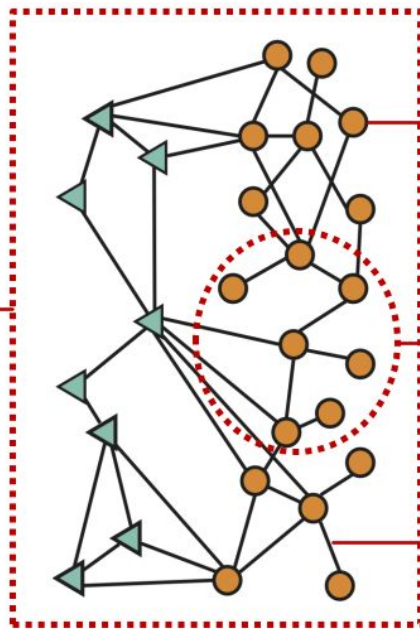


Image credit: Stanford CS224W

The whole graph is all available data

Node level

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

Community (subgraph) level

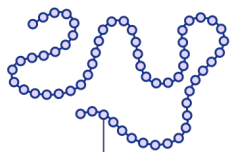
Community detection/clustering
Clustering nodes/edges

Edge-level

Link prediction
Predict links should exists or not

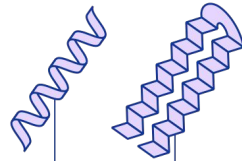
Example task 1: Protein folding

Every protein is made up of a sequence of amino acids bonded together



Amino acids

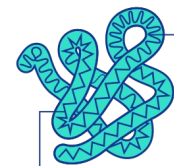
These amino acids interact locally to form shapes like helices and sheets



Alpha helix

Pleated sheet

These shapes fold up on larger scales to form the full three-dimensional protein structure



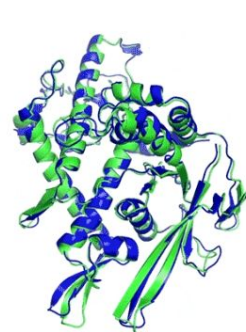
Pleated sheet

Alpha helix

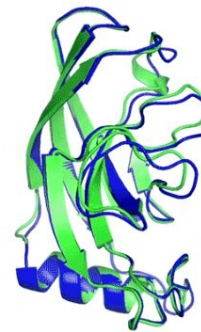
Proteins can interact with other proteins, performing functions such as signalling and transcribing DNA



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



T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)



T1049 / 6y4f
93.3 GDT
(adhesin tip)

Image credit: Deep Mind

● Experimental result
● Computational prediction

Example task 1: Protein folding

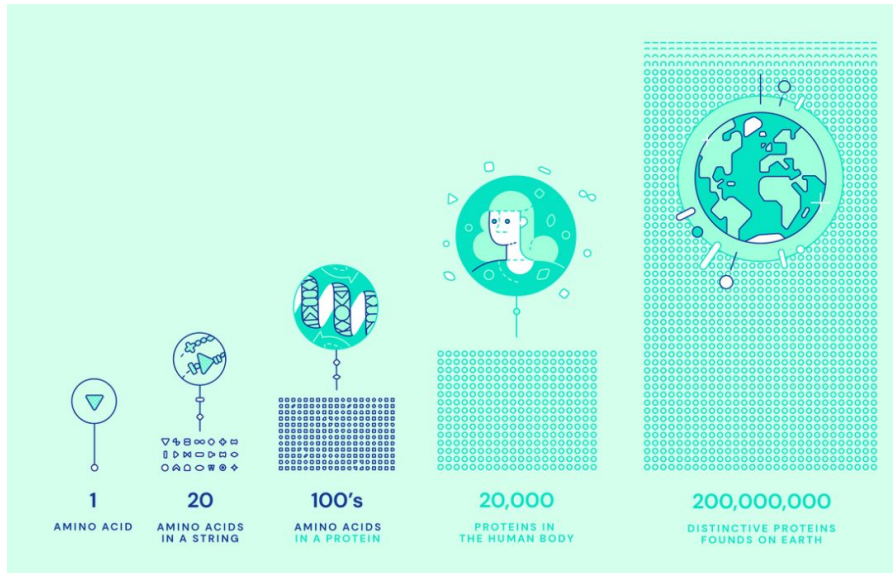
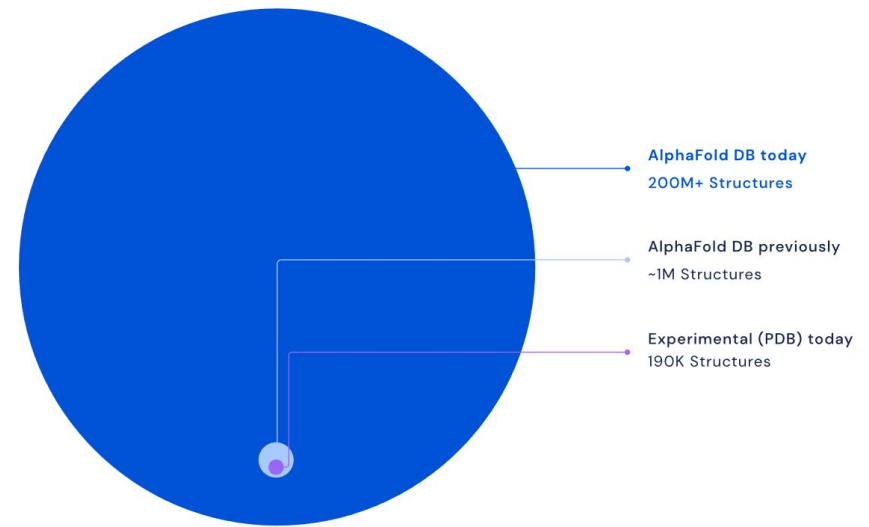
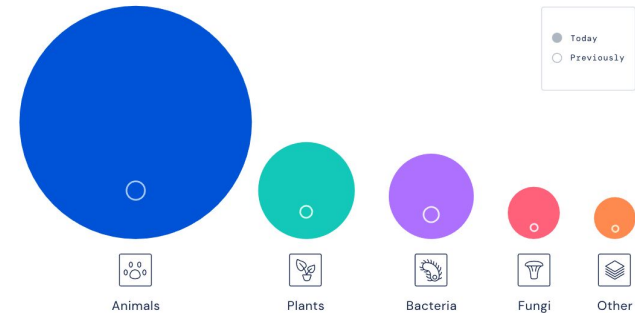


Image credit: Deep Mind



Number of species represented in AlphaFold DB

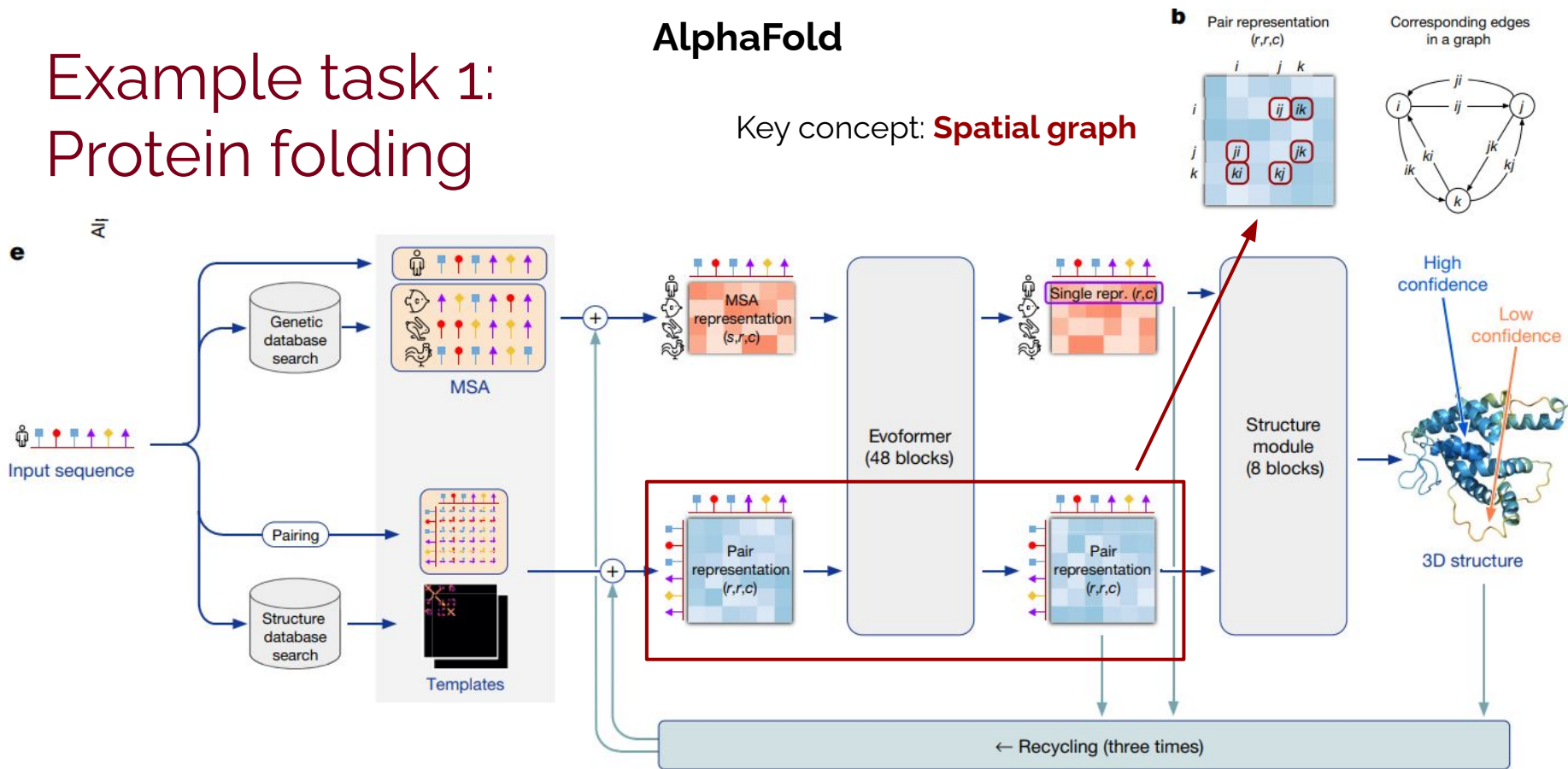
Total increase from ~10K to ~1M



Example task 1: Protein folding

AlphaFold

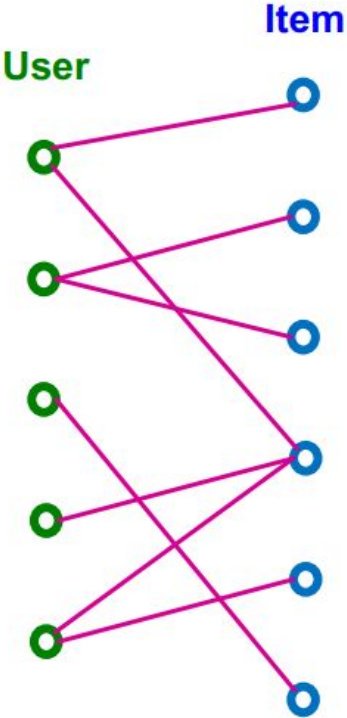
Key concept: **Spatial graph**



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



online shopping,
music/movie
recommendation

Nodes:
Users, items
Edges:
User-item
interactions

Image credit: Stanford CS224W

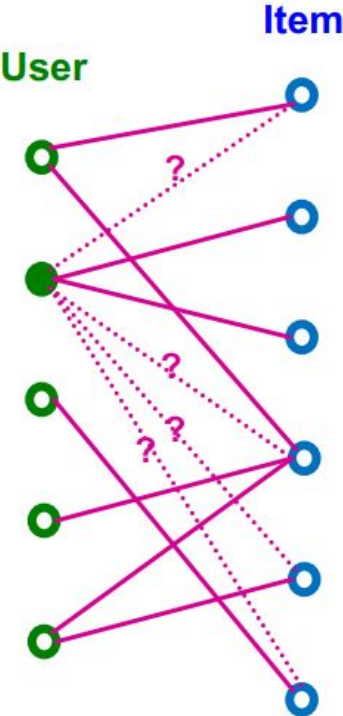
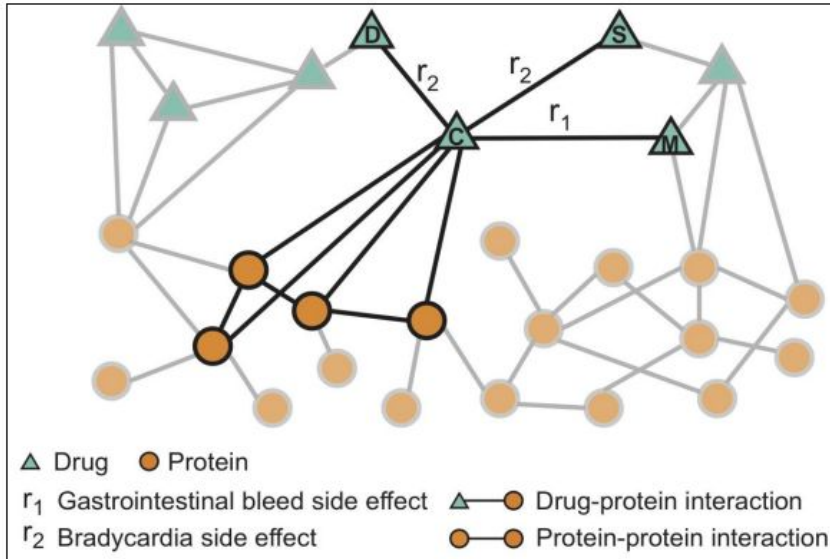


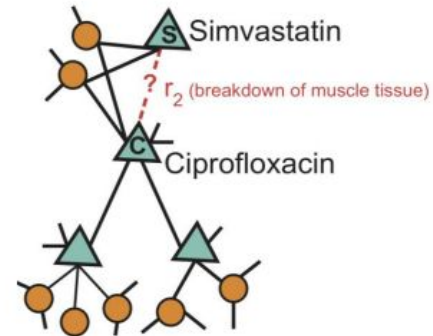
Image credit: Stanford CS224W

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?



Example task 4: Traffic prediction

The screenshot displays a navigation application interface. On the left, there is a search bar with "Your location" and "Minneapolis Institute of Art, 2400 3rd Ave". Below the search bar are icons for different travel modes: car, bus, walking, bicycle, and airplane. A "Leave now" button is visible, along with a "Send directions to your phone" option. A list of transit options is shown, including a 35-minute route from Washington Ave & Coffman Union to Minneapolis Institute of Art, and a 41-minute route from 1:49 PM to 2:30 PM. The map on the right shows the city of Minneapolis with a purple route highlighted, starting from Washington Ave & Coffman Union and ending at the Minneapolis Institute of Art. The map also shows major roads like I-94 and I-35W, and landmarks like Boom Island Park and the University of Minnesota.

Options

Leave now ▾

Send directions to your phone

Transit Options:

- 35 min**
1:50 PM—2:25 PM
1:54 PM from Washington Ave & Coffman Union
every 15 min
- 41 min**
1:49 PM—2:30 PM

Map Labels: Minneapolis, Washington Ave & Coffman Union, Franklin Ave, E & 3rd Ave S, Minneapolis Institute of Art, Walker Art Center, Stone Arch Bridge, University of Minnesota, Boom Island Park, Young Joni, Costco Business Center, Nicollet Island, Marcy-Holmes, Downtown West, Cedar-Riverside, Phillips, Seward, Calhoun Isles, North Loop, University Ave NE, E Hennepin Ave, Broadway St NE, Elm St SE, 26th A, E 25th St, Cedar A, 35W, 94, 394, WR, Glenwood Ave, N Washington Ave, N Emerson Ave, W Broadway Ave.

Example task 4: Traffic prediction

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

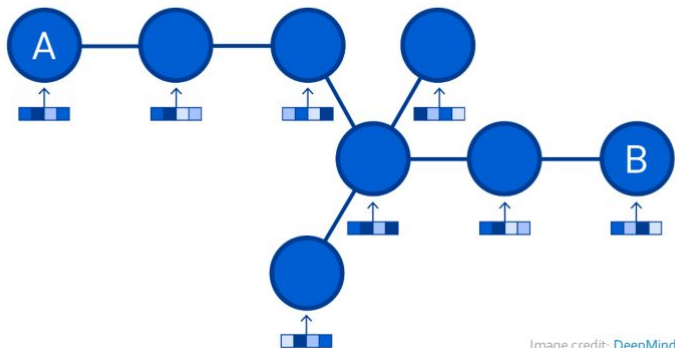


Image credit: Stanford CS224W

Predicting Time of Arrival with Graph Neural Networks

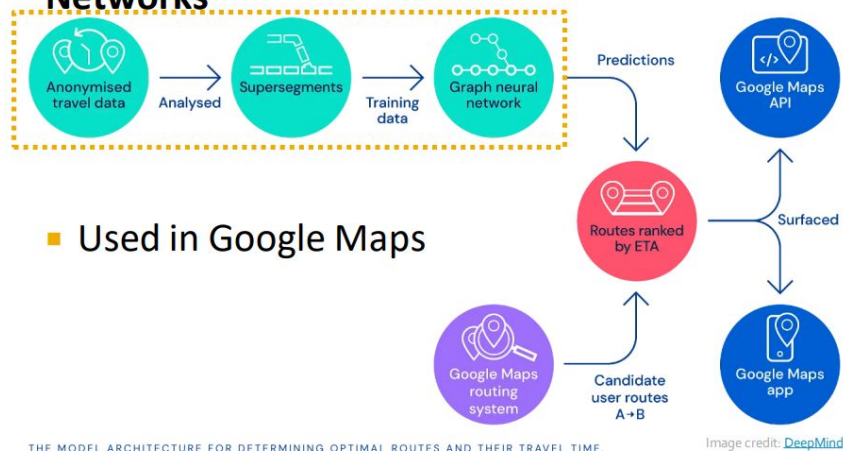


Image credit: Stanford CS224W

Subgraph discovery

Example task 5: Drug discovery

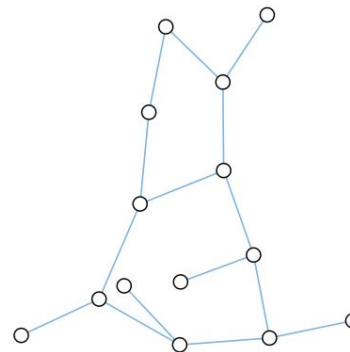
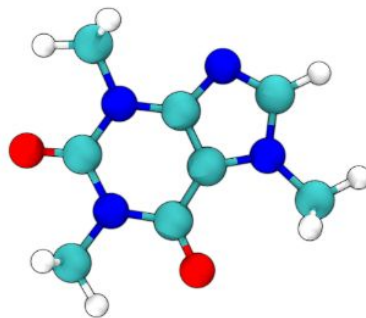
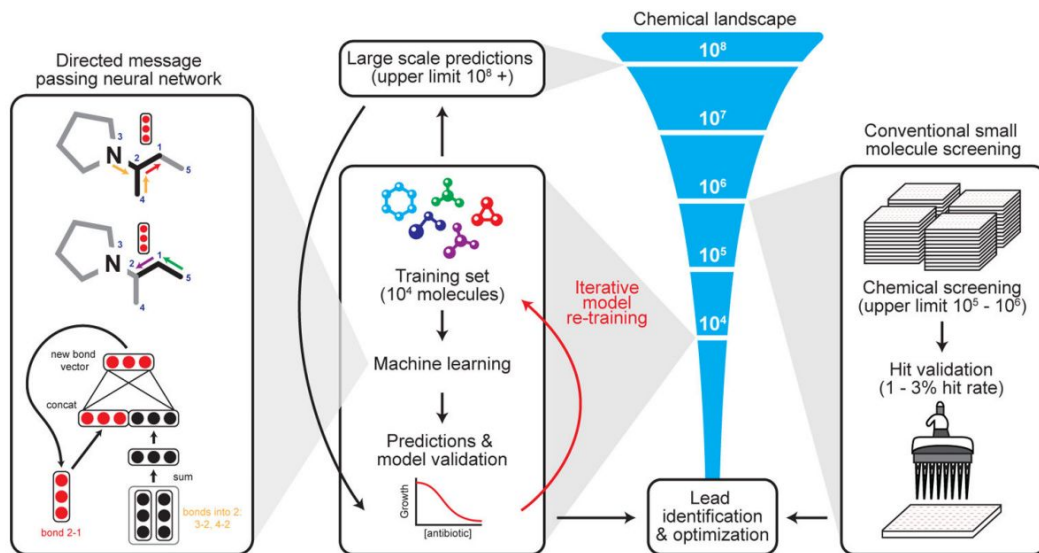


Image credit: <https://distill.pub/2021/gnn-intro/>

Molecules as graphs:

Nodes: atoms

Edges: chemical bounds



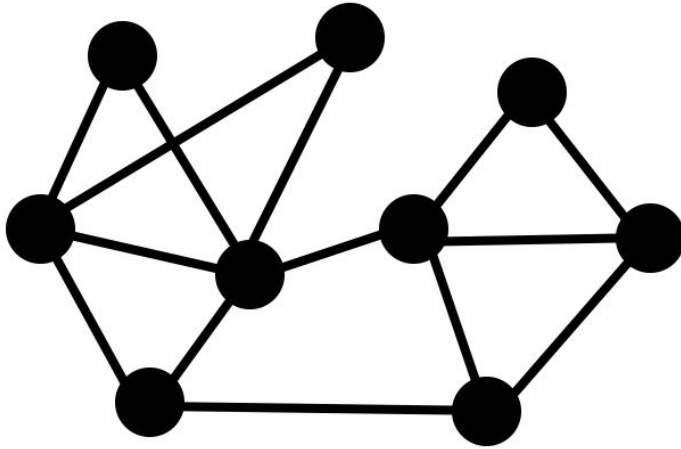
full-graph prediction

Image credit:

<https://pubmed.ncbi.nlm.nih.gov/32084340/>

Graphs: basic notions

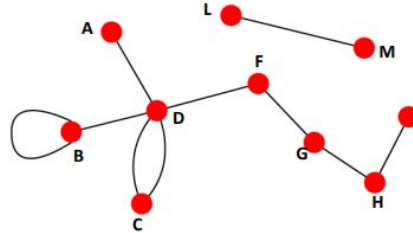
Basic objects



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

Undirected

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

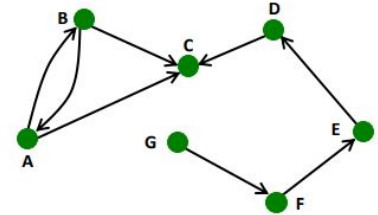


Examples:

- Collaborations
- Friendship on Facebook

Directed

- **Links:** directed (arcs)

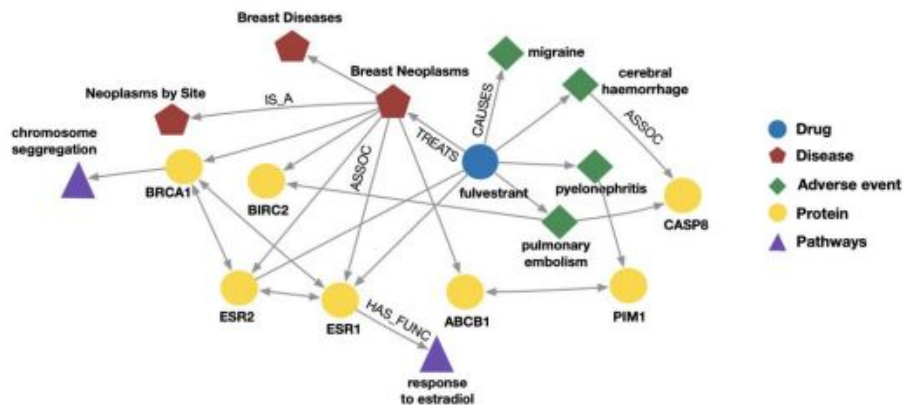


Examples:

- Phone calls
- Following on Twitter

Image credit: Stanford CS224W

Heterogeneous graphs

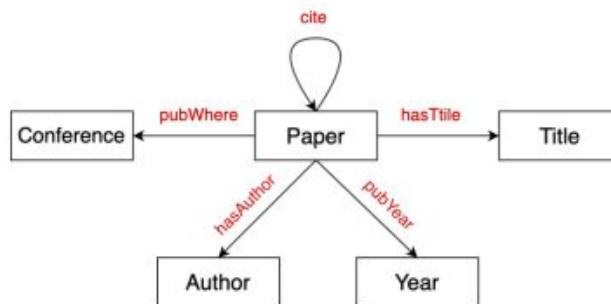


- Nodes/Edges are multi-typed

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

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**

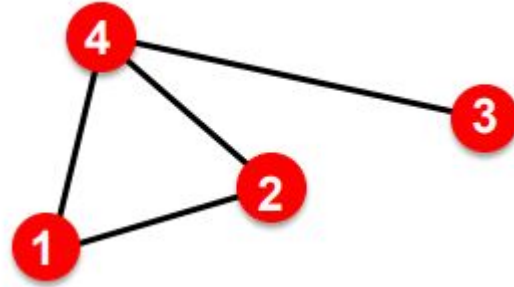
Example edge: **(GraphSAGE, NeurIPS)**

Example node type: **Author**

Example edge type (relation): **pubYear**

Graph representation

Undirected graphs



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

Adjacency matrix

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

Edge list

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

Adjacency list

Graph representation

Directed graphs

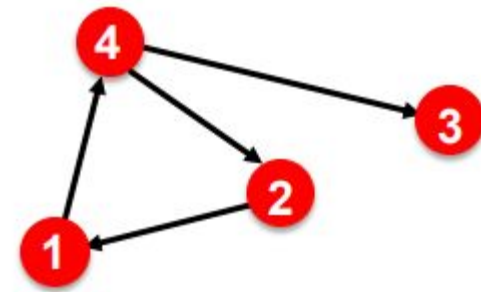
$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Adjacency matrix

(1, 4)
(2, 1)

(4, 2), (4, 3)

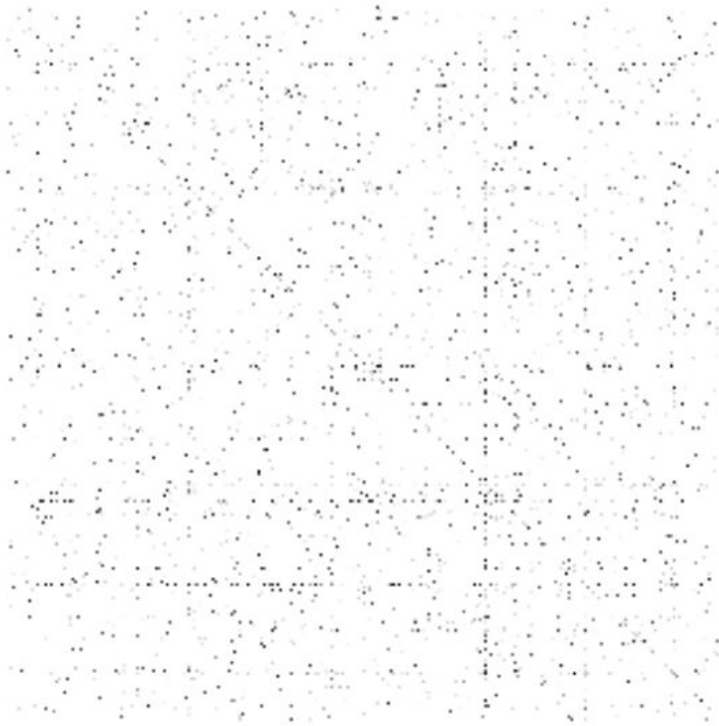
Edge list



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

Adjacency list

Adjacency matrix is often inefficient

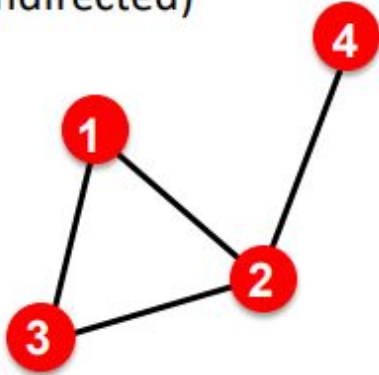


NETWORK	NODES	LINKS	DIRECTED/ UNDIRECTED	N	E
Internet	Routers	Internet connections	Undirected	192,244	609,066
WWW	Webpages	Links	Directed	325,729	1,497,134
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594
Phone Calls	Subscribers	Calls	Directed	36,595	91,826
Email	Email Addresses	Emails	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

$$\text{Density} = |E|/|N|^2$$

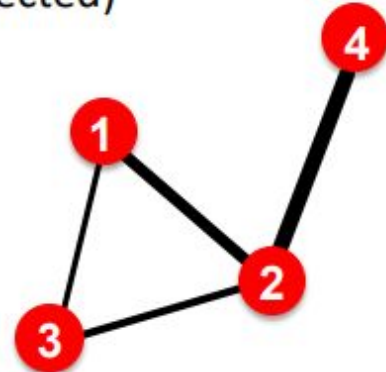
Weighted graphs

- **Unweighted**
(undirected)



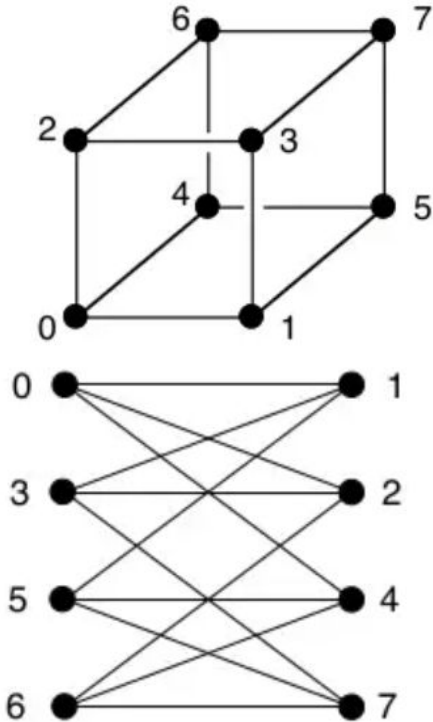
$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

- **Weighted**
(undirected)



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

Graph isomorphism/equivalence



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

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

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

Permutation invariance

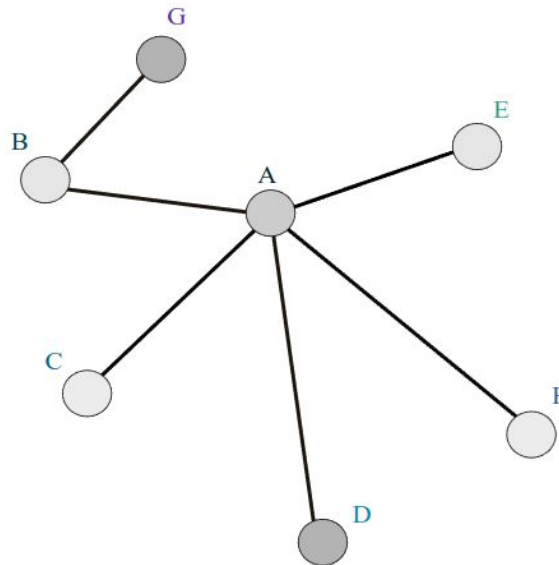
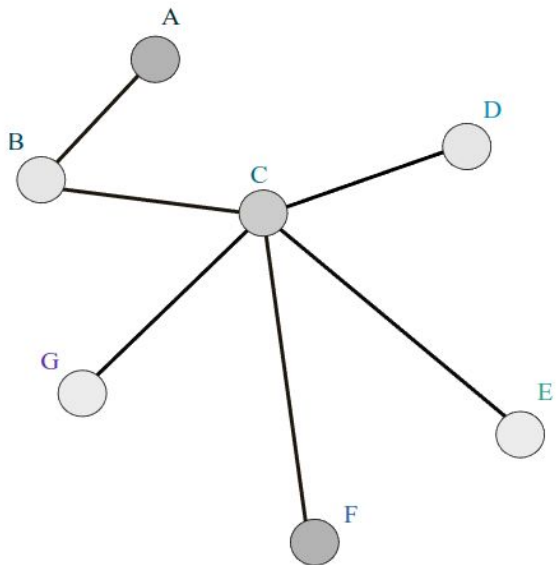


Image credit: Image credit: <https://distill.pub/2021/understanding-gnns/>

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

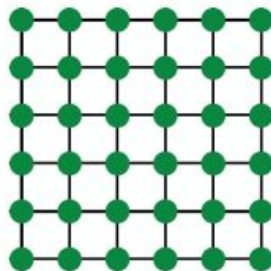
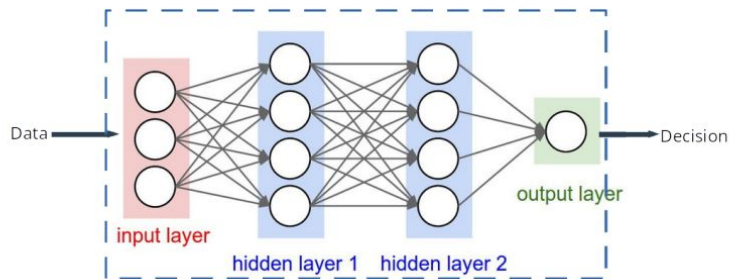
Graph neural networks

Representation learning for graphs

traditional learning pipeline



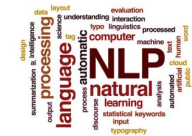
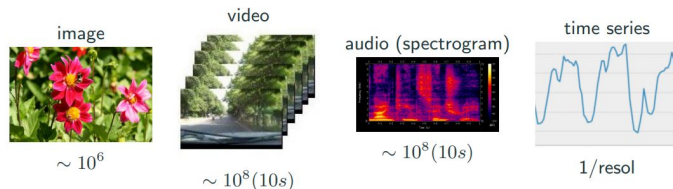
modern learning pipeline



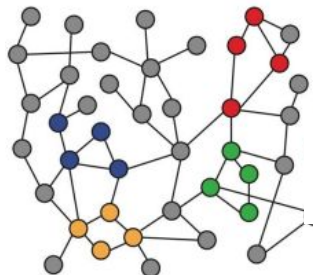
Grid



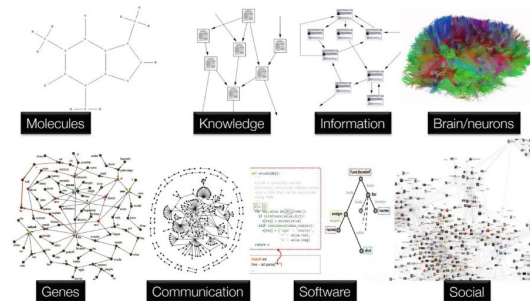
List



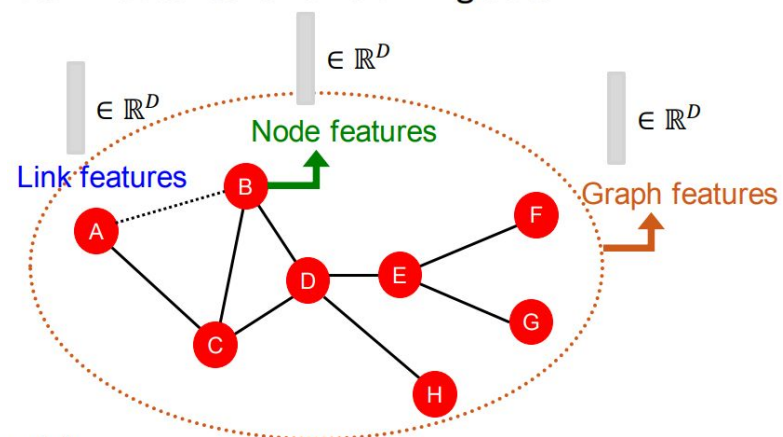
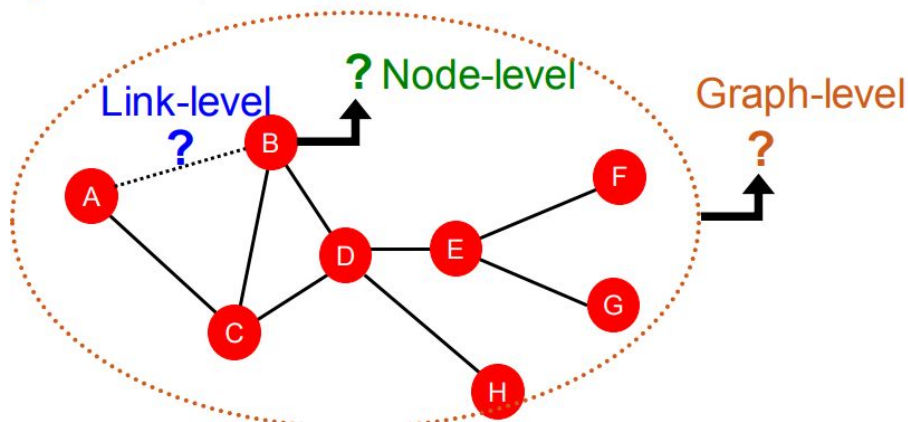
- machine translation, e.g., English \leftrightarrow Chinese
- typing/writing prediction (smart compose)
- semantic classification



Graph



Where to put the features?



Train an ML model:

- Random forest
- SVM
- Neural network, etc.

$$x_1 \rightarrow y_1$$

⋮

$$x_N \rightarrow y_N$$

Apply the model:

- Given a new node/link/graph, obtain its features and make a prediction

$$x \rightarrow y$$

Node embedding

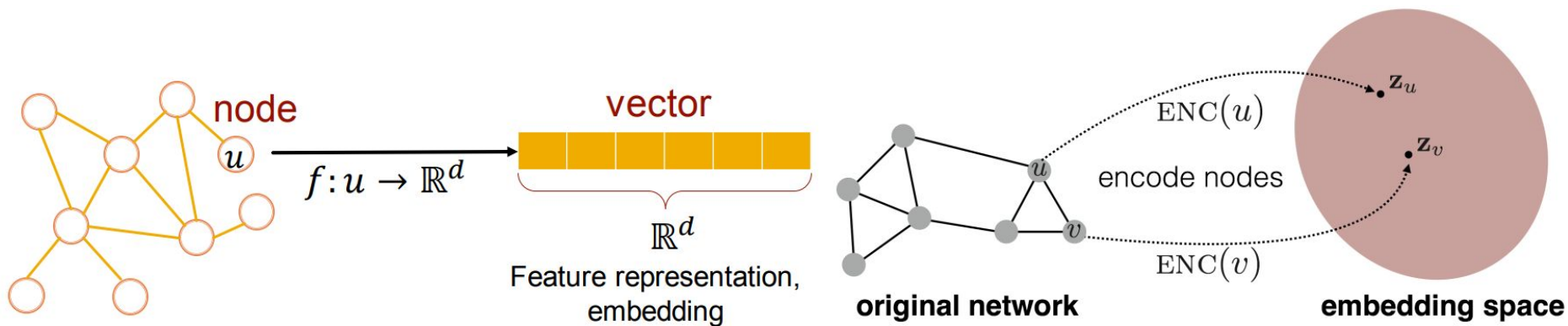


Image credit: Stanford CS224W

N : set of nodes

A : adjacency matrix

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

u, v : nodes in N

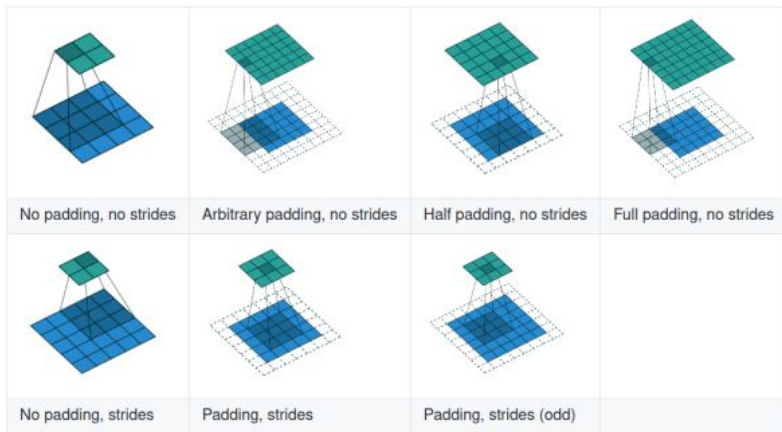
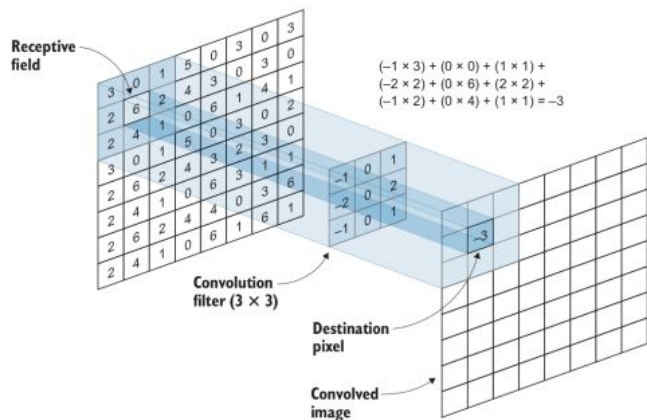
$\mathcal{N}(u)$: neighbors of u

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: [Elgandy, 2020])

https://github.com/vdumoulin/conv_arithmetic

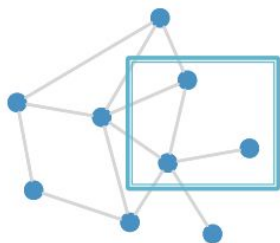
Convolution as performing **local info aggregation** (or **message passing**):

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

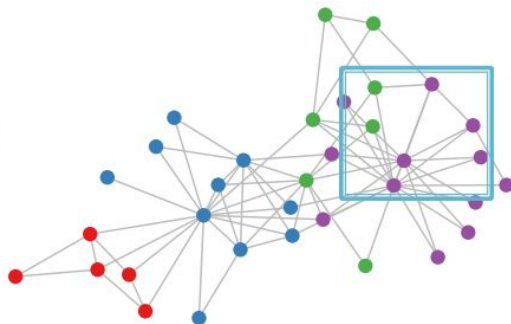
How to define the f ?

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

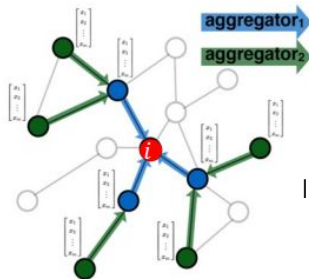
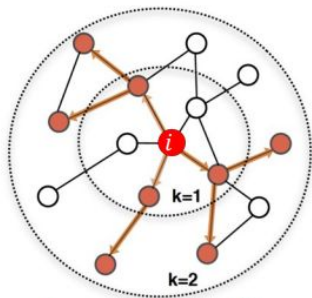
- **Neighborhood**: Each time, the conv window focus on a **local neighborhood** of the current pixel
- **Aggregation**: Conv effectively **aggregates** the local info by **weighted summation**



or this:



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

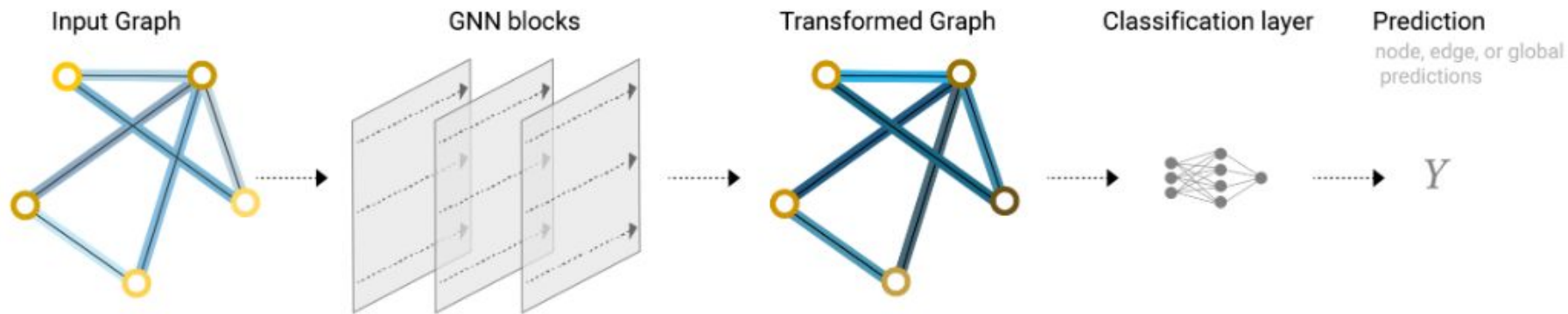


We need neighbors on the graph!

Image credit: Stanford CS224w

One layer of a graph neural network (GNN)!

Graph in, graph out

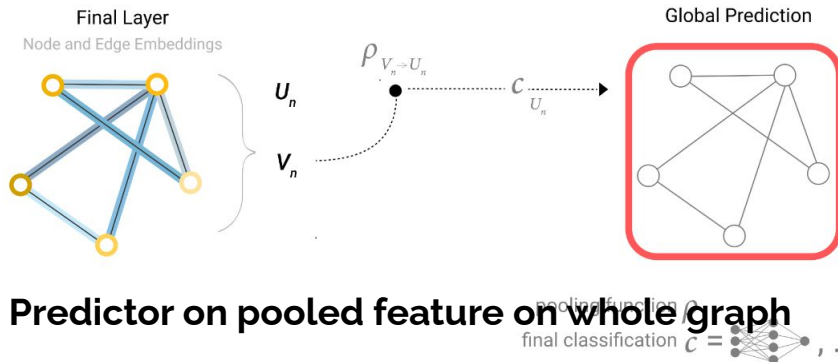
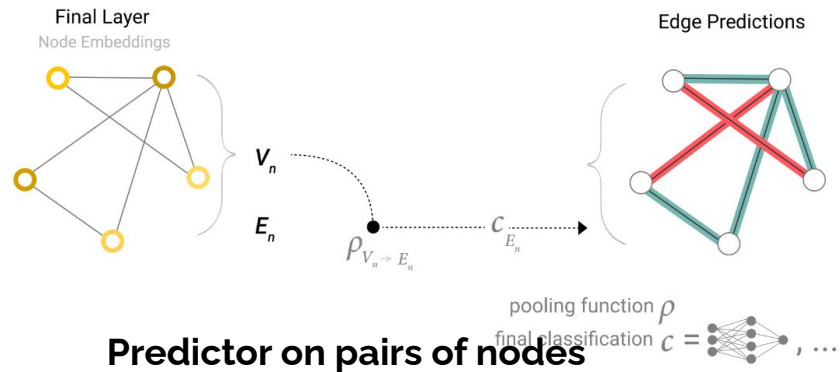
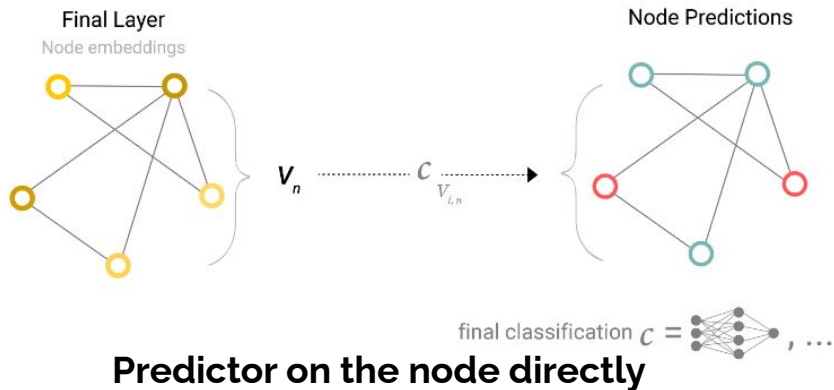


An end-to-end prediction task with a GNN model.

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

How to make supervised predictions?

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

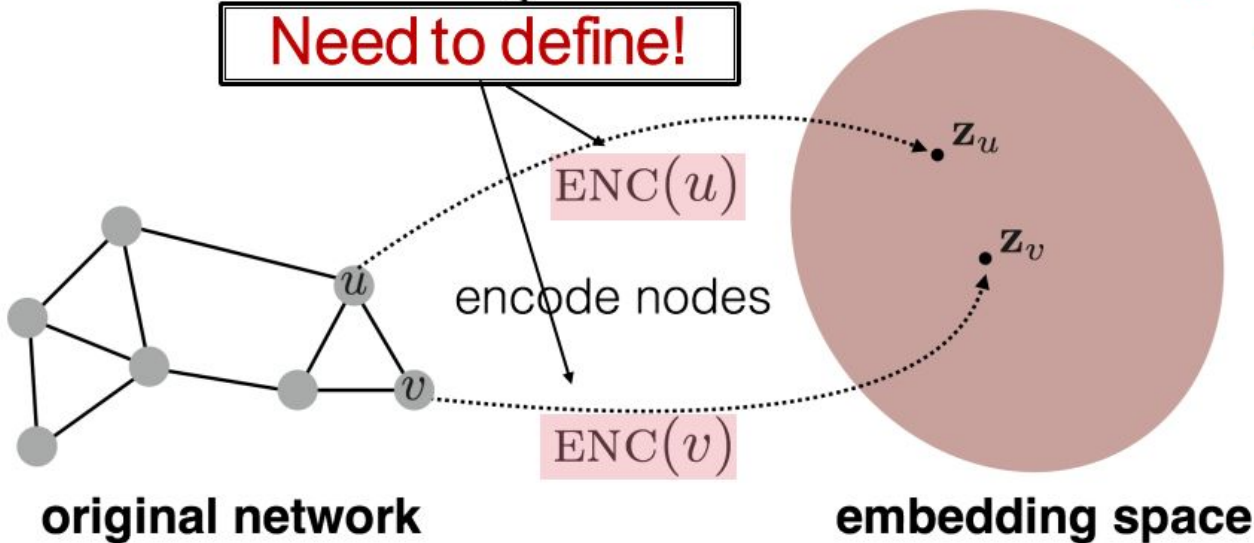


How to perform unsupervised learning?

Goal: $\text{similarity}(u, v)$ $\approx \mathbf{z}_v^T \mathbf{z}_u$
in the original network Similarity of the embedding

Need to define!

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



Look into GNN layers

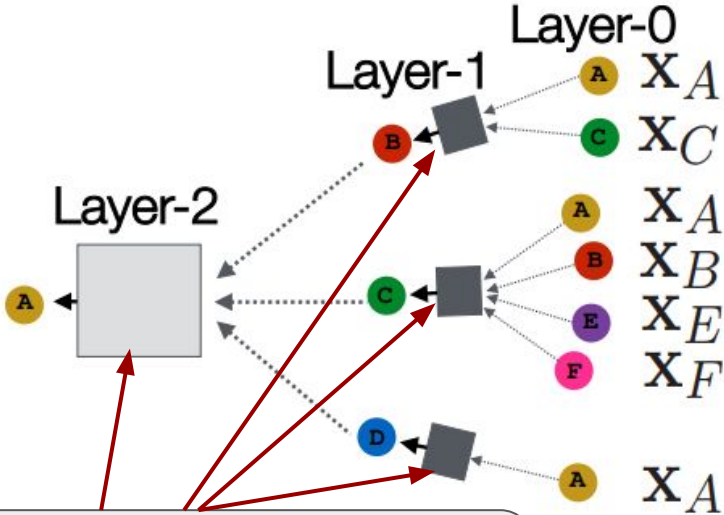
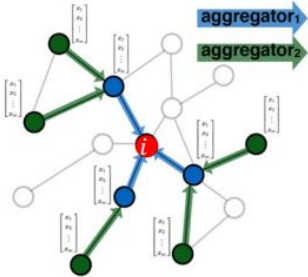
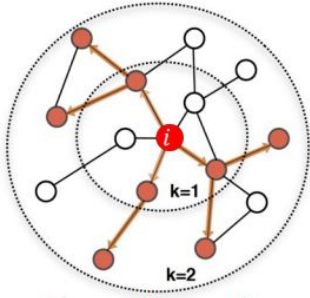
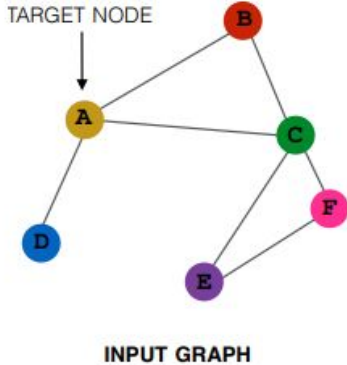


Image credit: Stanford CS224W

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

Permutation invariance

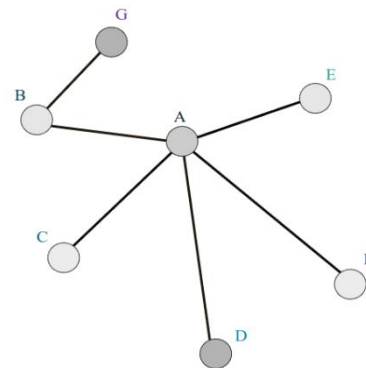
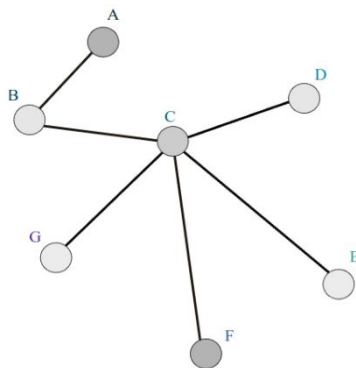
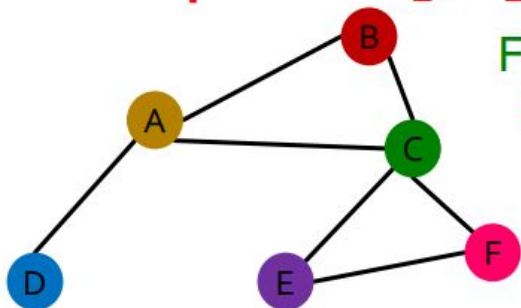


Image credit: <https://distill.pub/2021/understanding-gnns/>

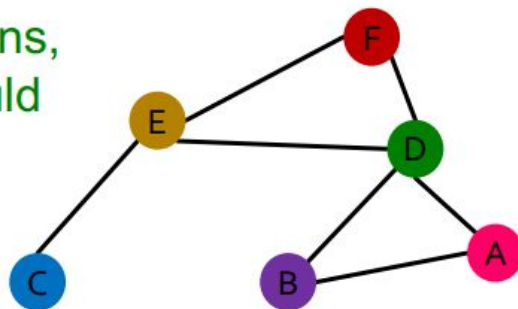
Permutation invariance: permuting the names of the nodes doesn't change the graph, as graph nodes are intrinsically orderless

$$\text{For } f : G(\mathbf{A}, \mathbf{X}) \mapsto \mathbf{h}$$
$$f(\mathbf{A}_1, \mathbf{X}_2) = f(\mathbf{A}_2, \mathbf{X}_2)$$

Order plan 1: $\mathbf{A}_1, \mathbf{X}_1$



Order plan 2: $\mathbf{A}_2, \mathbf{X}_2$



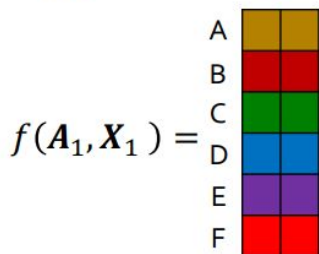
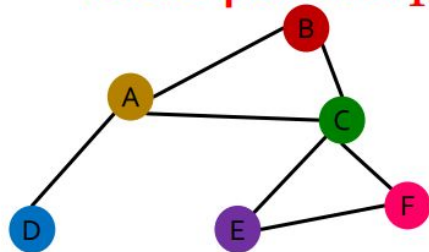
For two order plans,
output of f should
be the same!

Image credit: Stanford CS224W

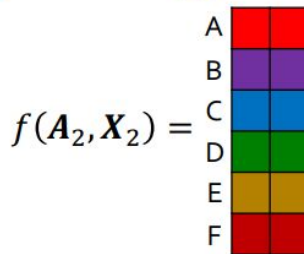
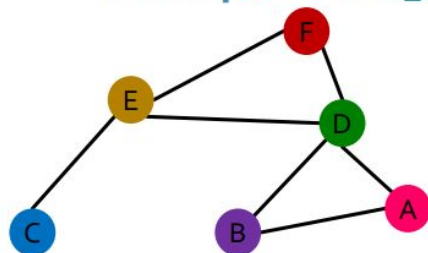
Permutation equivariance

For $f : G(\mathbf{A}, \mathbf{X}) \mapsto \mathbf{H} \in \mathbb{R}^{|N| \times d}$
 $f(\mathbf{\Pi A \Pi}^\top, \mathbf{\Pi X}) = \mathbf{\Pi f(A, X)}$
 for any permutation $\mathbf{\Pi}$

Order plan 1: $\mathbf{A}_1, \mathbf{X}_1$

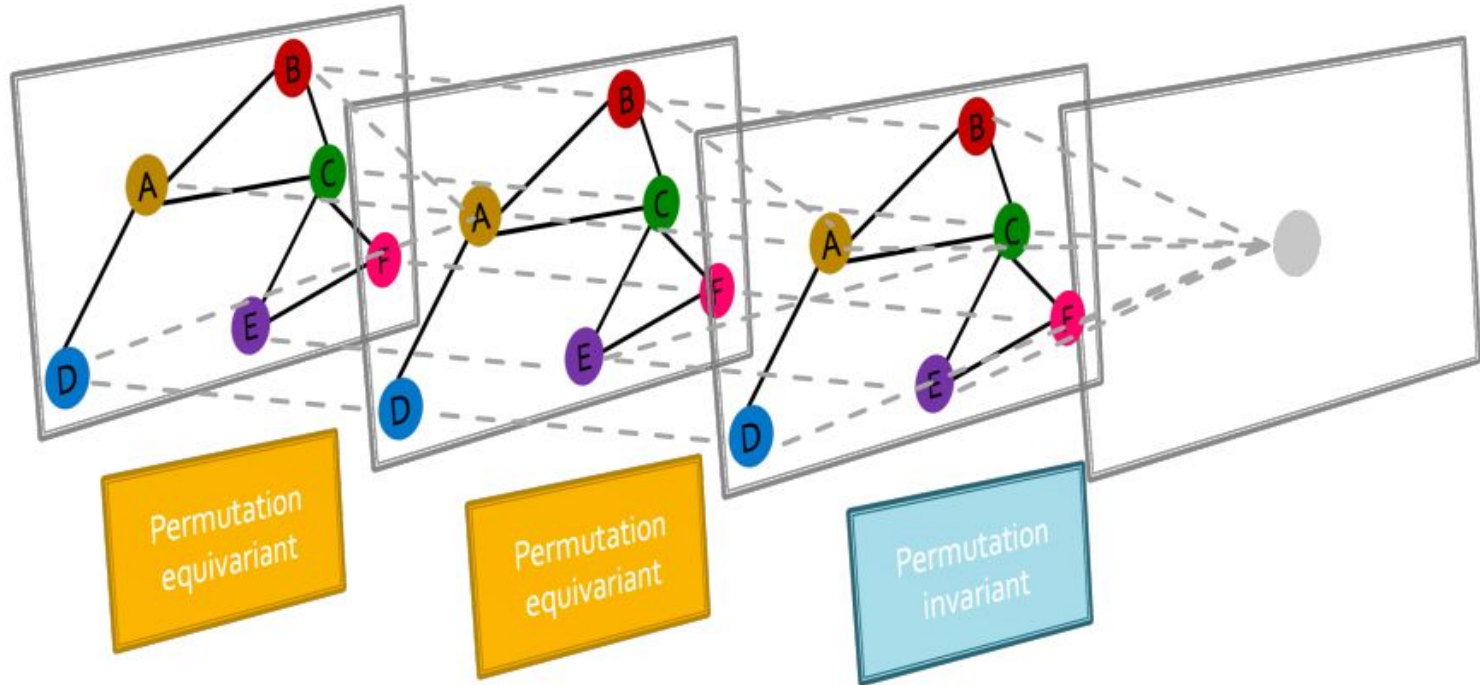


Order plan 2: $\mathbf{A}_2, \mathbf{X}_2$



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

A typical GNN consists of multiple permutation equivariant/invariant layers



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.

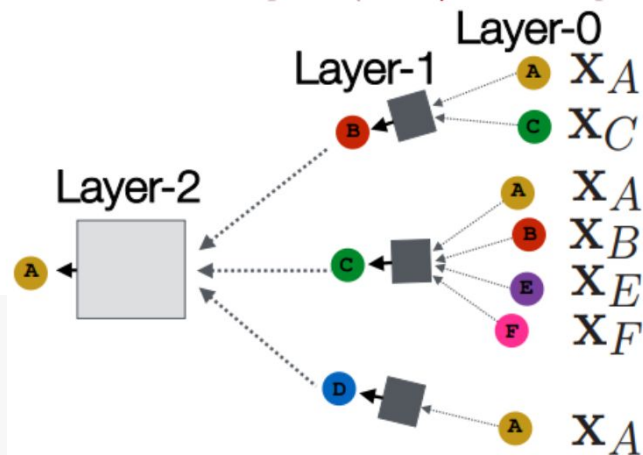
and for $k = 1, 2, \dots$ upto K :

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

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.



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 $k = 1, 2, \dots$ 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.

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 $k = 1, 2, \dots$ upto K :

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

Node v 's
embedding at
step k .

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

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

... concatenated
with ...

Color Codes:

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

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 $k = 1, 2, \dots$ upto K :

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

Node v 's
embedding at
step k .

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

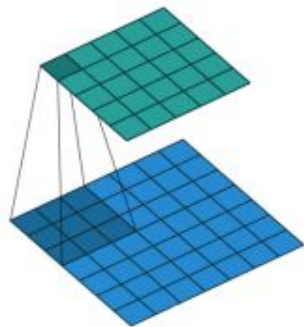
Node v 's
embedding at
step $k - 1$.

Color Codes:

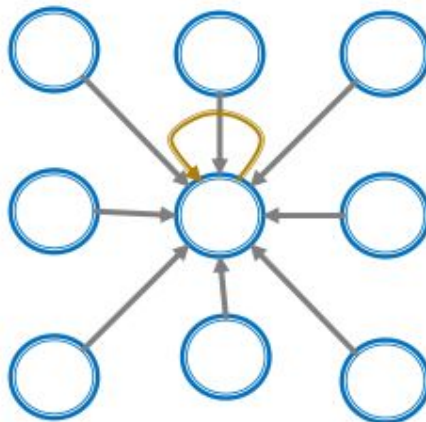
- Embedding of node v .
- Embedding of a neighbour of node v .
- (Potentially) Learnable parameters.

Connection to CNNs and Transformers

filter:



Image



Graph

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

$$\text{GNN formulation: } h_v^{(l+1)} = \sigma(\mathbf{W}_l \sum_{u \in \mathcal{N}(v)} \frac{h_u^{(l)}}{|\mathcal{N}(v)|} + \mathbf{B}_l h_v^{(l)}), \forall l \in \{0, \dots, L-1\}$$

$$\text{CNN formulation: } h_v^{(l+1)} = \sigma(\sum_{u \in \mathcal{N}(v)} \mathbf{W}_l^u h_u^{(l)} + \mathbf{B}_l h_v^{(l)}), \forall l \in \{0, \dots, L-1\}$$

Connection to CNNs and Transformers

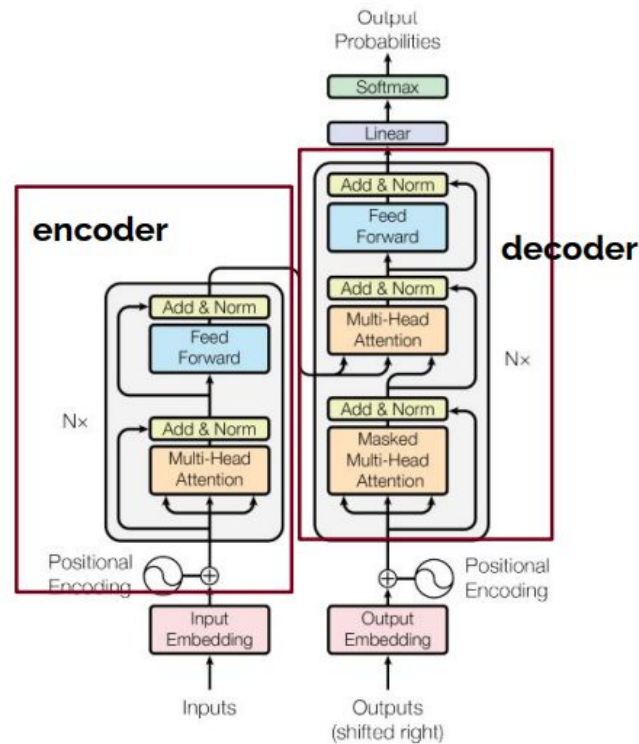
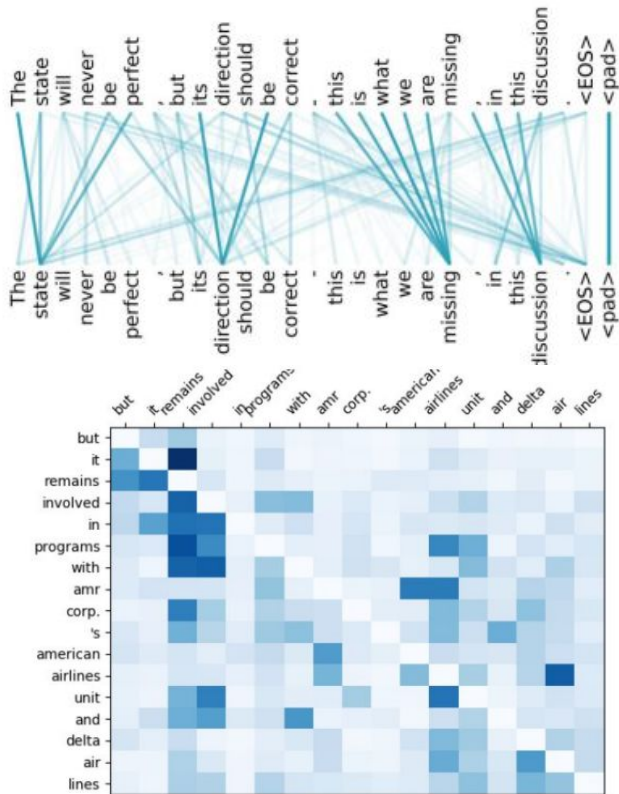


Figure 1: The Transformer - model architecture.

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

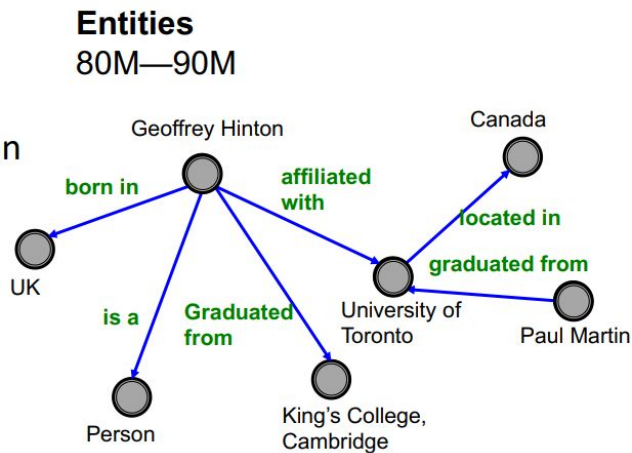


Image credit: Stanford CS224W

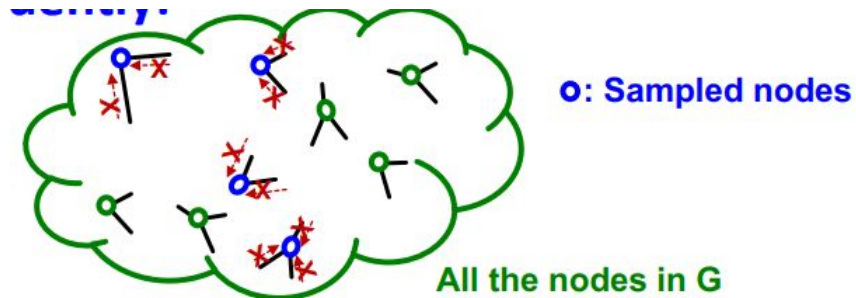
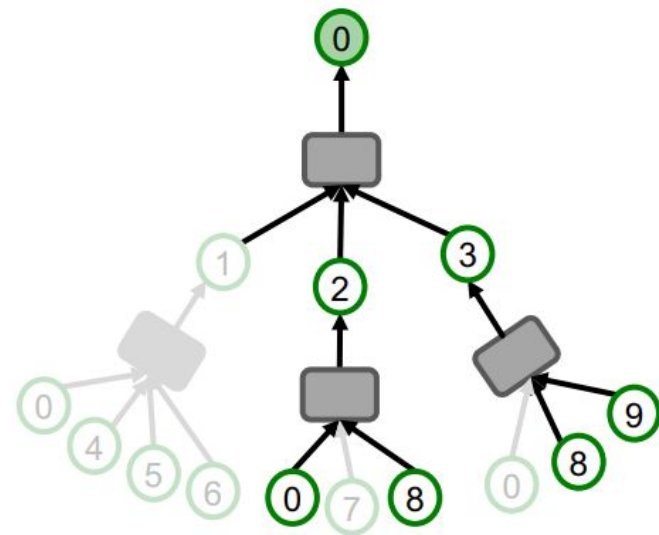
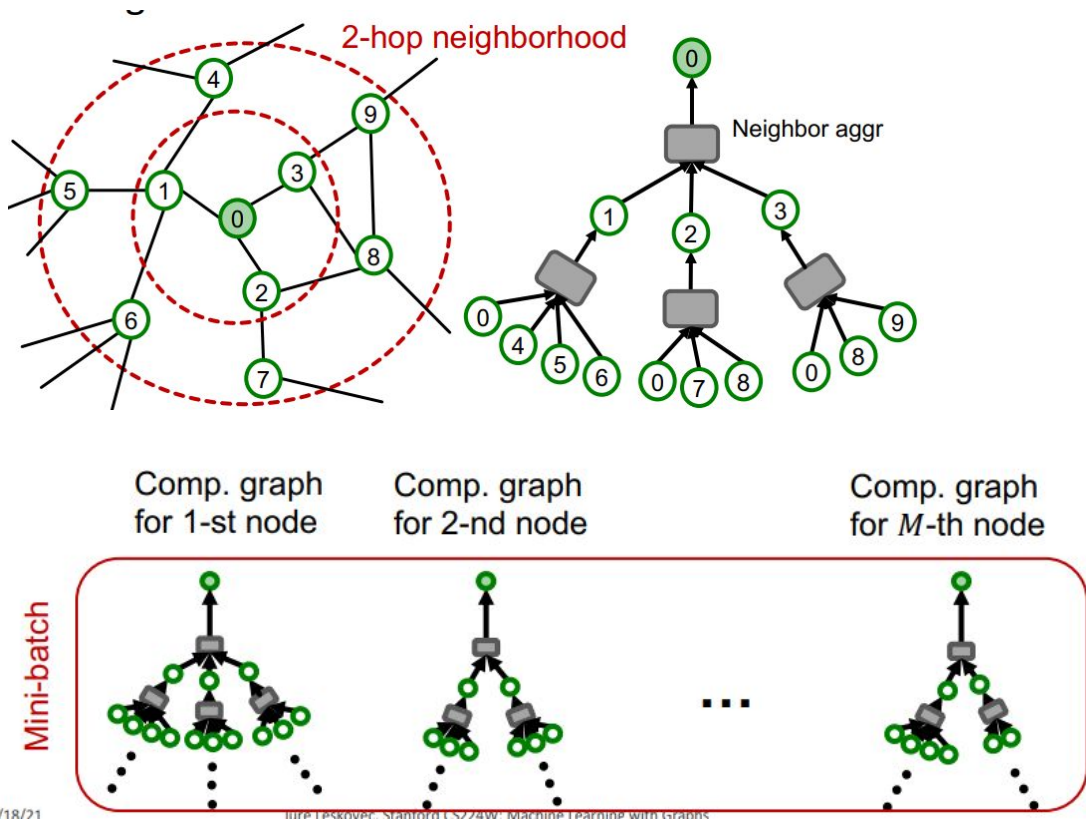


Image credit: Stanford CS224W

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

Two careful sub-sampling strategies

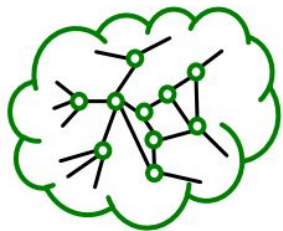


Neighborhood sampling

Image credit: Stanford CS224W

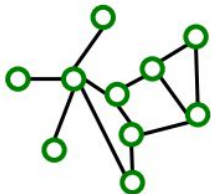
Two careful sub-sampling strategies

Large graph

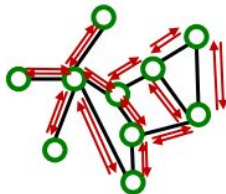


Sampled subgraph

(small enough to be put on a GPU)



Layer-wise node embeddings update on the GPU

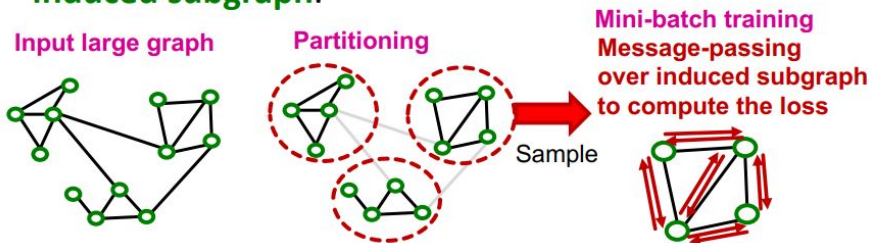


Rationale: important to keep the community structures, i.e., keep the “backbone” nodes

Image credit: Stanford CS224W

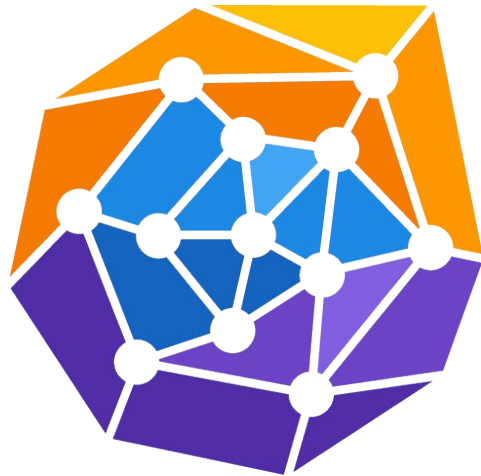
Cluster-GCN consists of two steps:

- **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**.



Software

PyTorch Geometric (PyG)



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

Deep Graph Library (DGL)



<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/