# Relationship Modeling: Graph Neural Networks (GNNs)

Ju Sun
Computer Science & Engineering

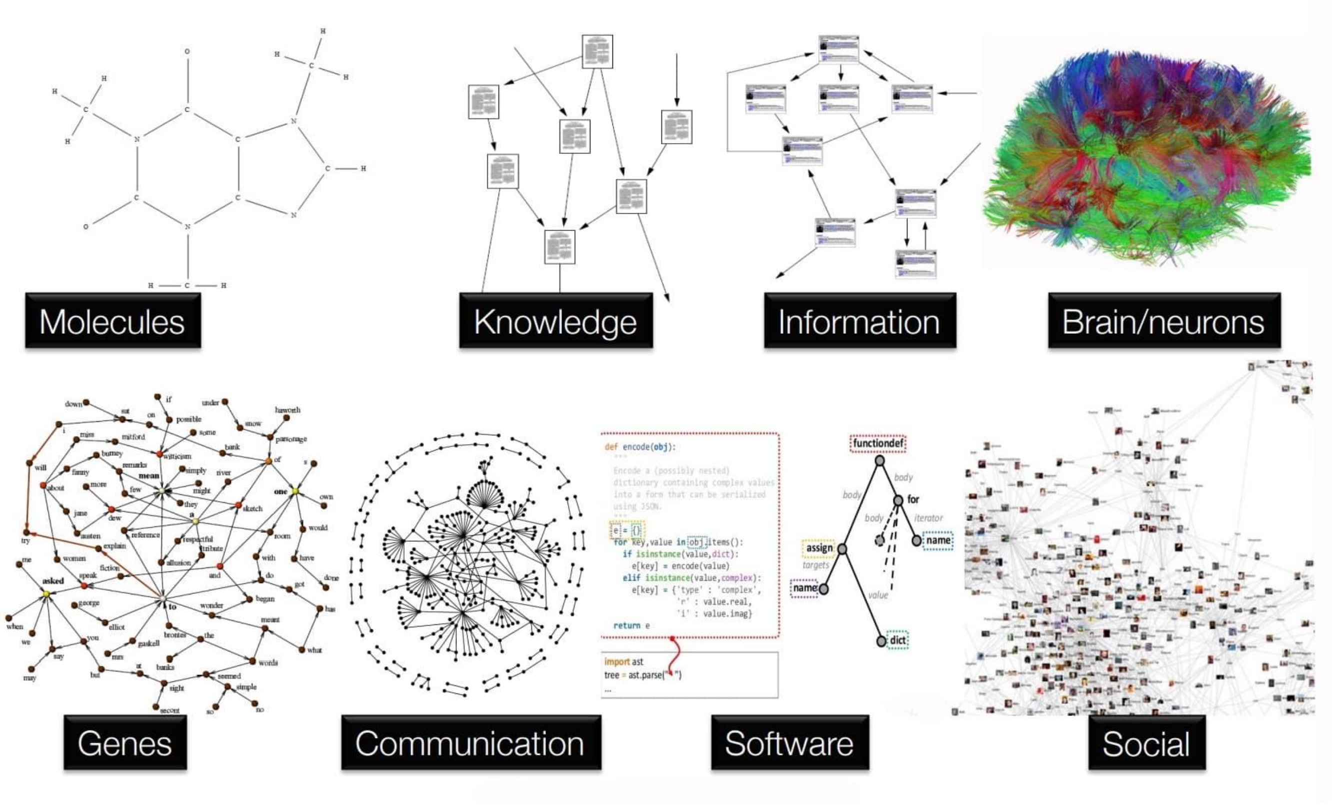
Nov 29, 2023

## Outline

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

## Why graphs?

## Graphs are everywhere!



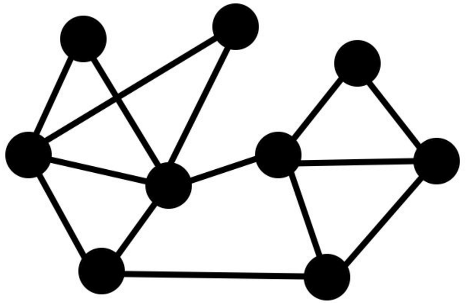


Image credit: Stanford CS224W

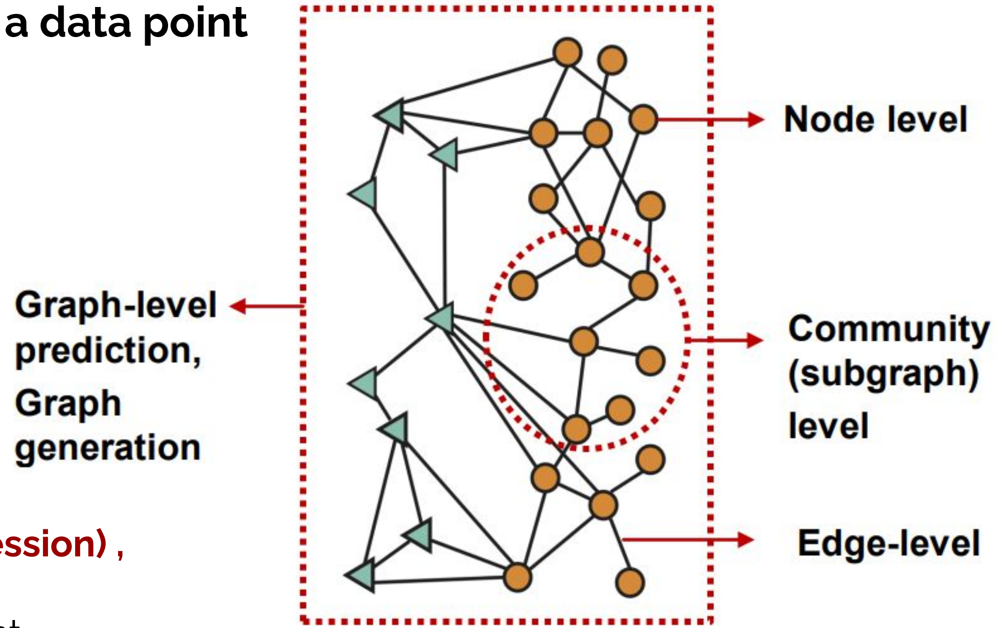
Graphs model **relationships/ interactions**

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

# Different tasks on graphs

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

**The whole graph is all available data**


Graph-level prediction, Graph generation
Node level
Community (subgraph) level
Edge-level

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

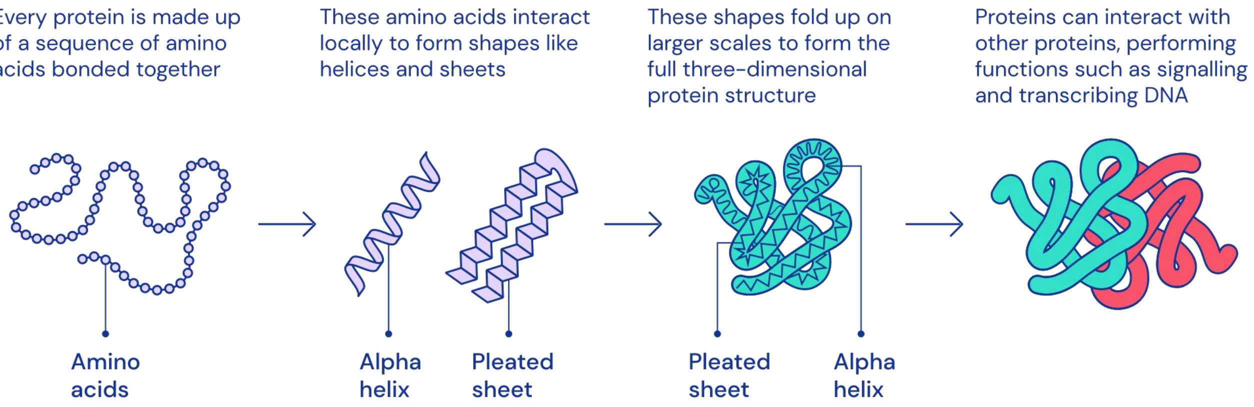
**Community detection/clustering**
Clustering nodes/edges

**Link prediction**
Predict links should exists or not

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

Image credit: Stanford CS224W

## Example task 1: Protein folding


Every protein is made up of a sequence of amino acids bonded together
These amino acids interact locally to form shapes like helices and sheets
These shapes fold up on larger scales to form the full three-dimensional protein structure
Proteins can interact with other proteins, performing functions such as signalling and transcribing DNA
Amino acids
Alpha helix
Pleated sheet
Pleated sheet
Alpha helix

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

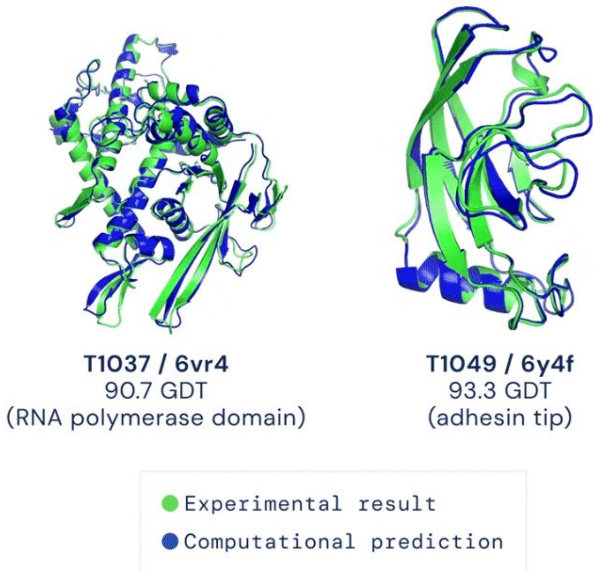


Image credit: Deep Mind

## Example task 1: Protein folding

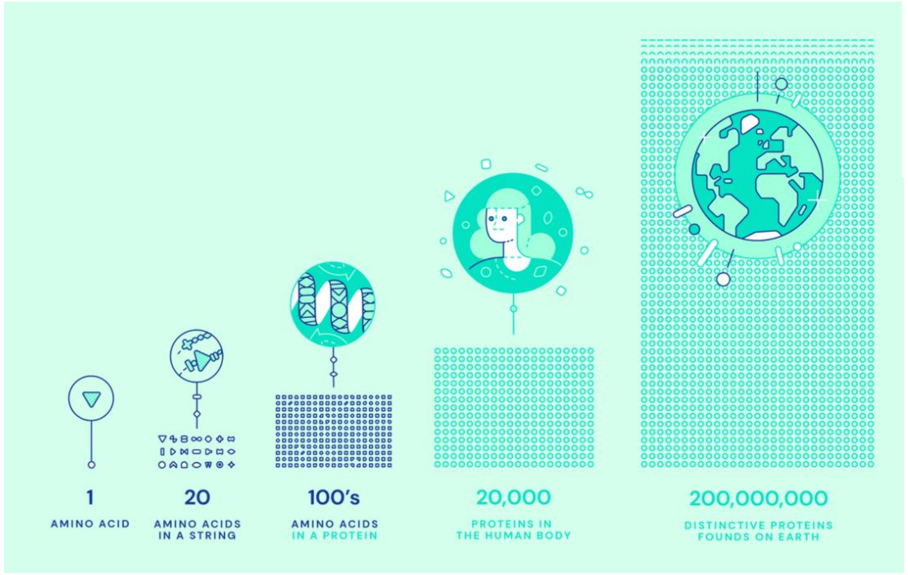
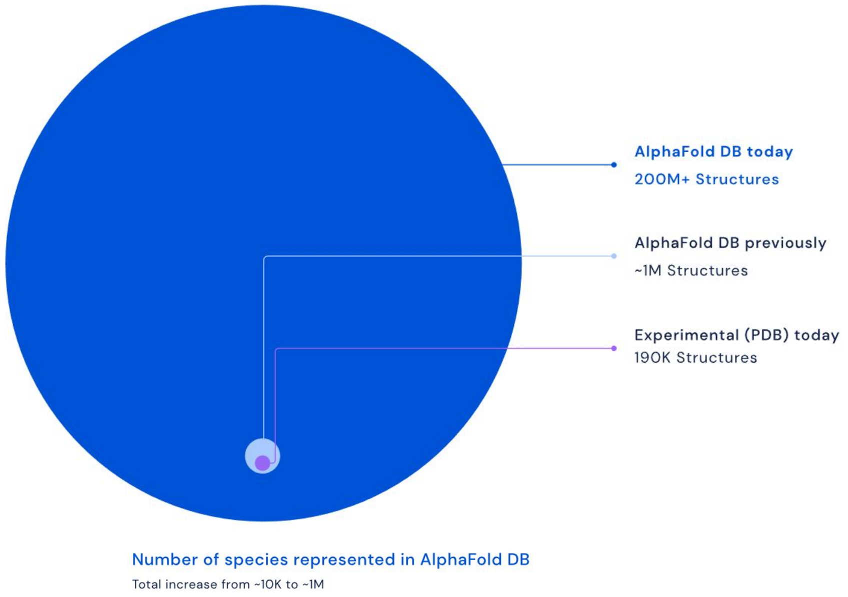
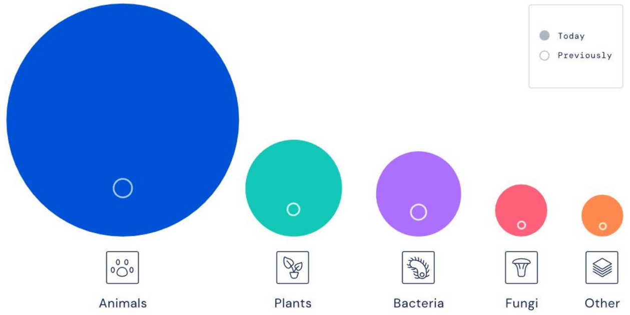


Image credit: Deep Mind

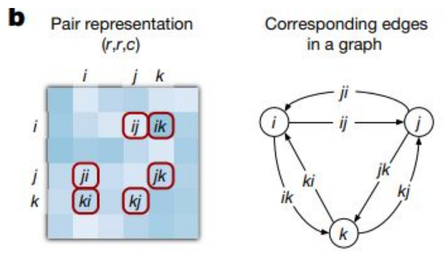


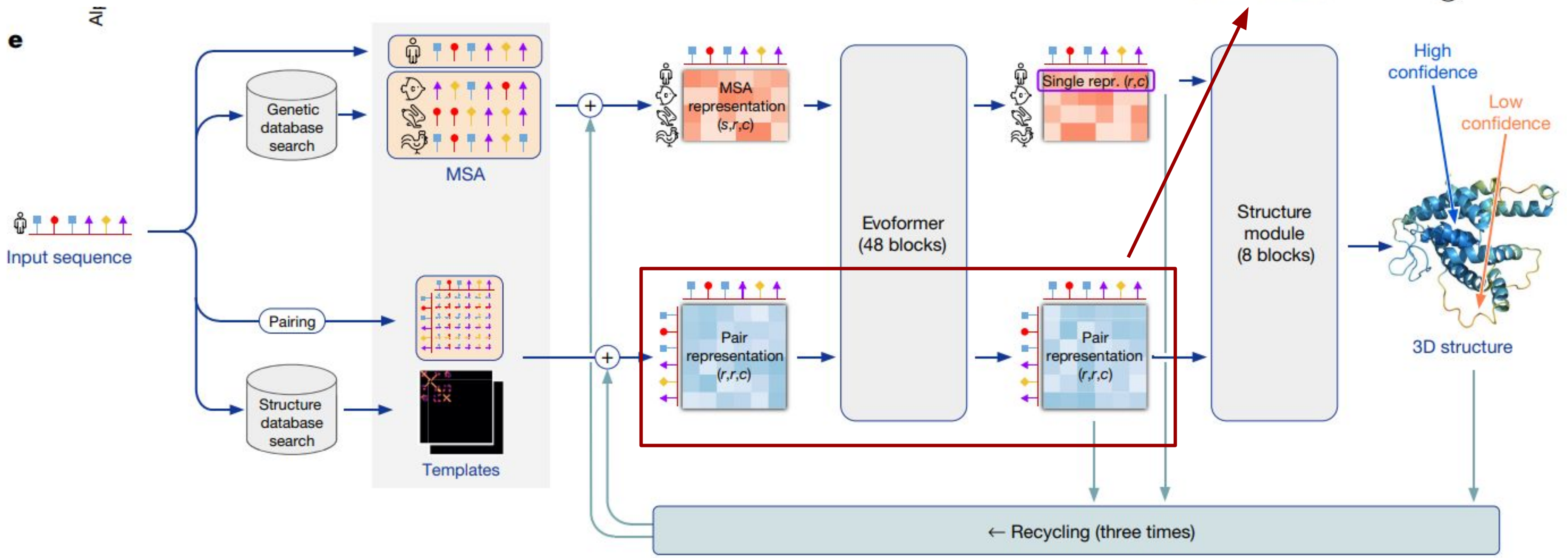


# Example task 1: Protein folding

## AlphaFold

Key concept: **Spatial graph**




e
Input sequence
Genetic database search
MSA
Pairing
Structure database search
Templates
MSA representation (s,r,c)
Pair representation (r,r,c)
Evoformer (48 blocks)
Single repr. (r,c)
Pair representation (r,r,c)
Structure module (8 blocks)
High confidence
Low confidence
3D structure
← Recycling (three times)

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

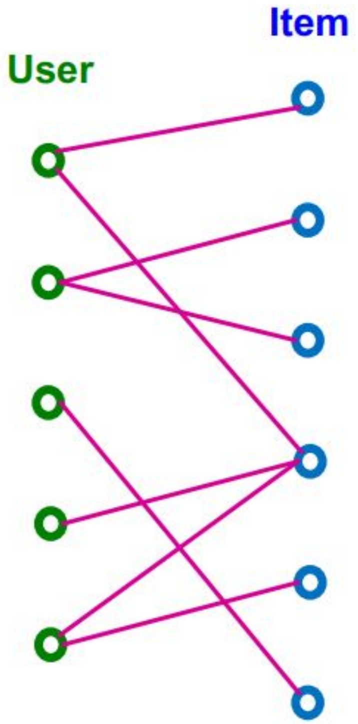


Image credit: Stanford CS224W

online shopping, music/movie recommendation

**Nodes:**
Users, items
**Edges:**
User-item interactions

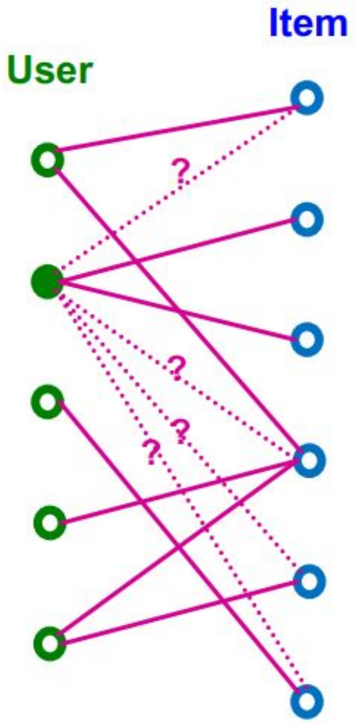
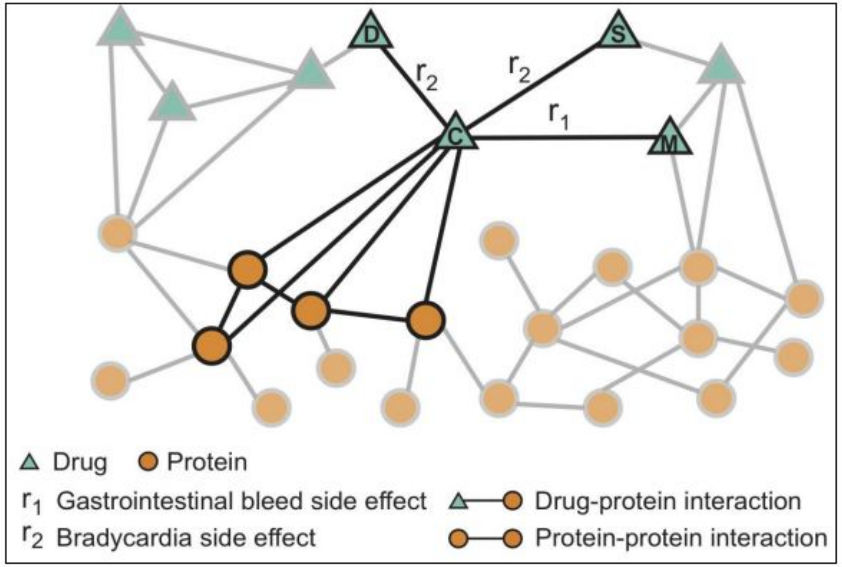


Image credit: Stanford CS224W

# Example task 3: Drug adverse effect discovery

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



D
S
C
M
$r_2$
$r_2$
$r_1$
△ Drug
● Protein
$r_1$ Gastrointestinal bleed side effect
$r_2$ Bradycardia side effect
Drug-protein interaction
Protein-protein interaction

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

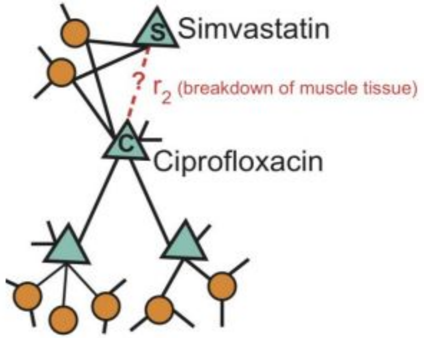
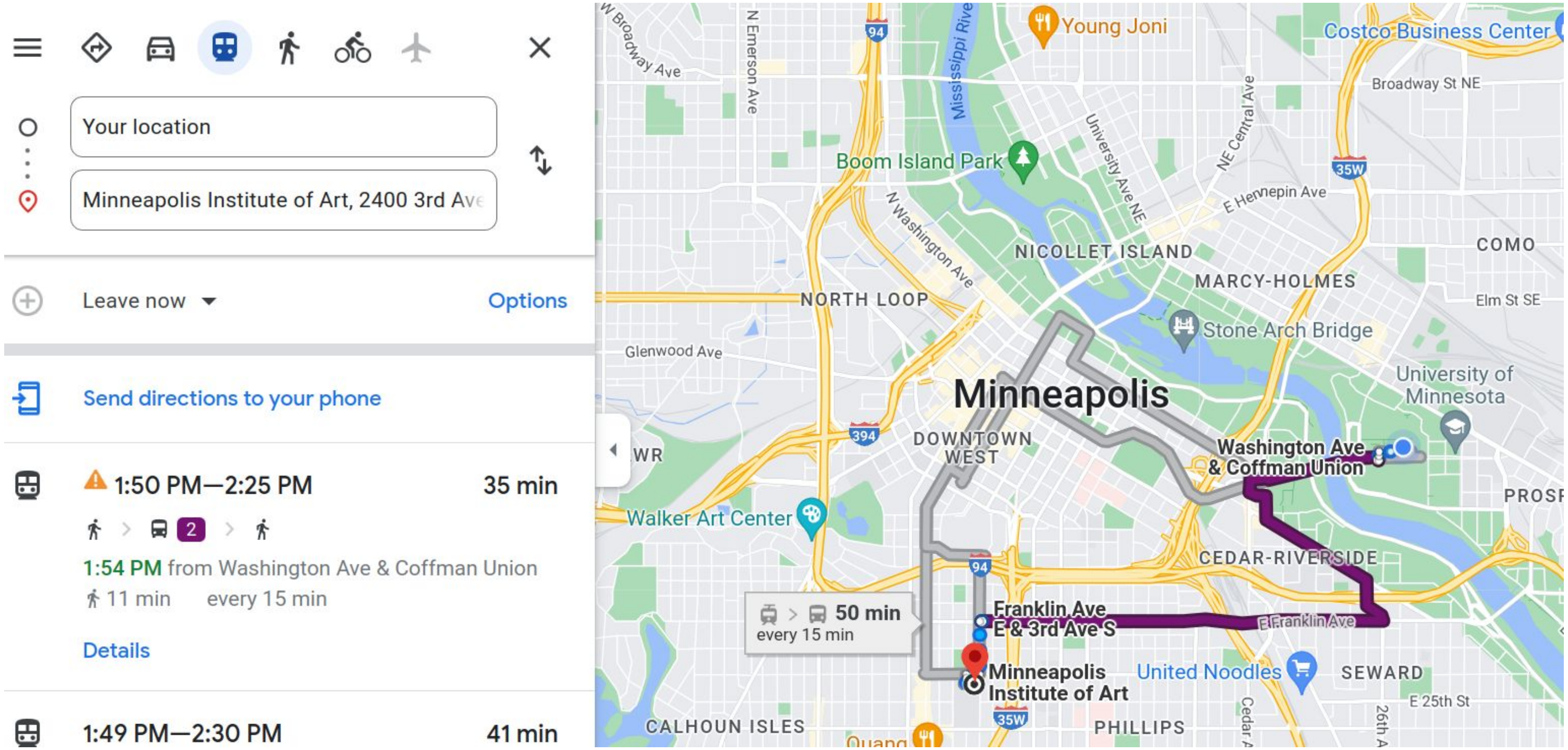


Image credit: Stanford CS224W

# Example task 4: Traffic prediction



Your location

Minneapolis Institute of Art, 2400 3rd Ave

Leave now

Options

Send directions to your phone

1:50 PM—2:25 PM 35 min

2

1:54 PM from Washington Ave & Coffman Union

11 min every 15 min

Details

1:49 PM—2:30 PM 41 min

# Example task 4: Traffic prediction

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

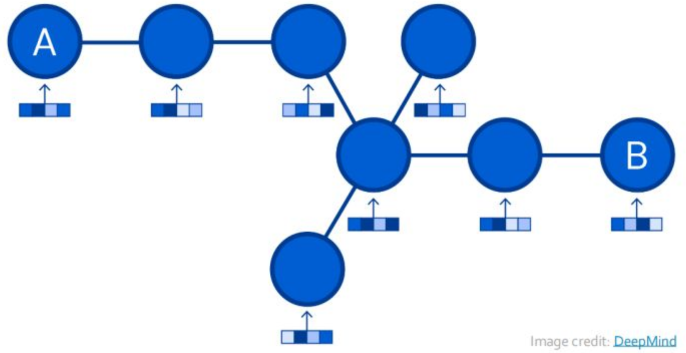
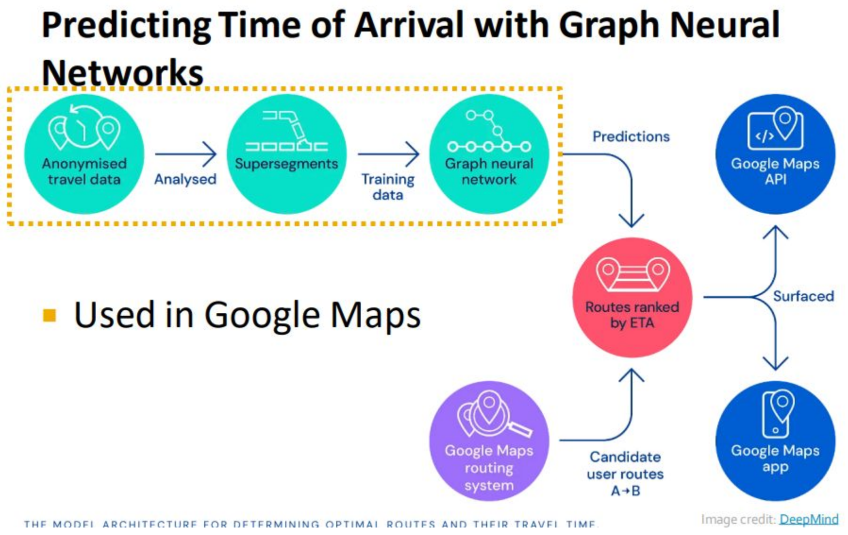


Image credit: Stanford CS224W

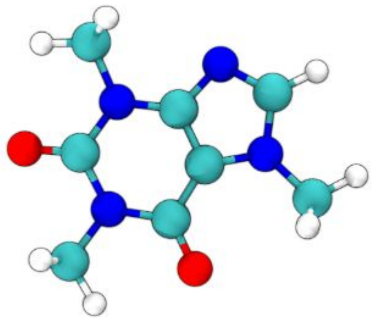

Predicting Time of Arrival with Graph Neural Networks
Anonymised travel data
Analysed
Supersegments
Training data
Graph neural network
Predictions
Google Maps API
Routes ranked by ETA
Surfaced
Google Maps routing system
Candidate user routes A→B
Google Maps app
THE MODEL ARCHITECTURE FOR DETERMINING OPTIMAL ROUTES AND THEIR TRAVEL TIME.
Image credit: DeepMind

- Used in Google Maps

Image credit: Stanford CS224W

**Subgraph discovery**

## Example task 5: Drug discovery



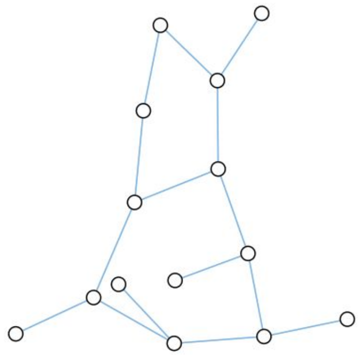
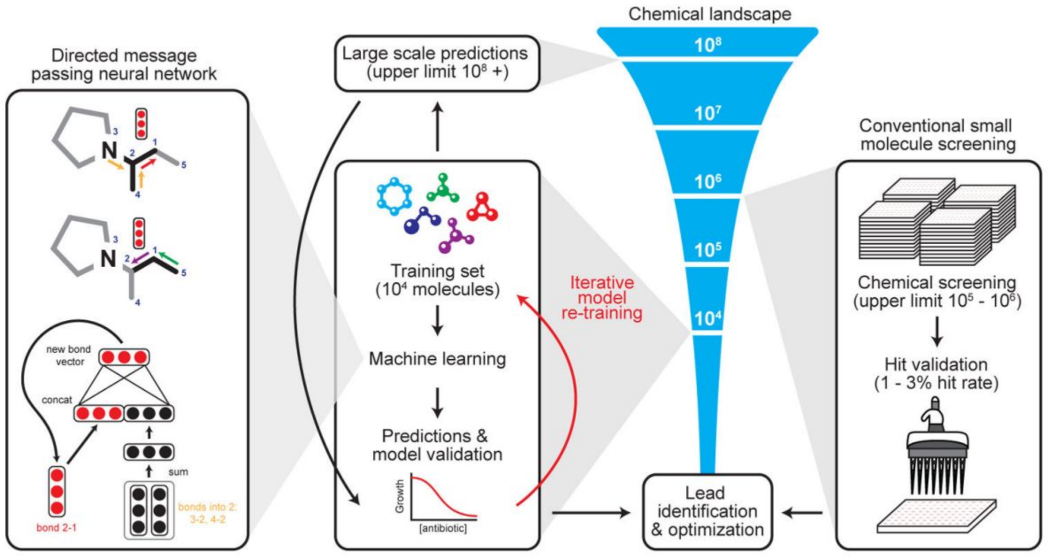


Image credit: https://distill.pub/2021/gnn-intro/

**Molecules as graphs:**

**Nodes**: atoms **Edges**: chemical bounds



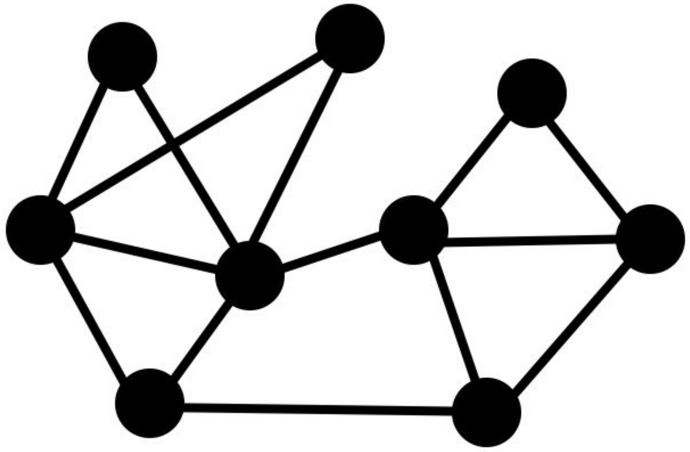
Chemical landscape
$10^8$
$10^7$
$10^6$
$10^5$
$10^4$
Directed message passing neural network
new bond vector
concat
sum
bond 2-1
bonds into 2: 3-2, 4-2
Large scale predictions (upper limit $10^8$ +)
Training set ($10^4$ molecules)
Machine learning
Predictions & model validation
Growth
[antibiotic]
Iterative model re-training
Conventional small molecule screening
Chemical screening (upper limit $10^5$ - $10^6$)
Hit validation (1 - 3% hit rate)
Lead identification & optimization

**full-graph prediction**

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

## Graphs: basic notions

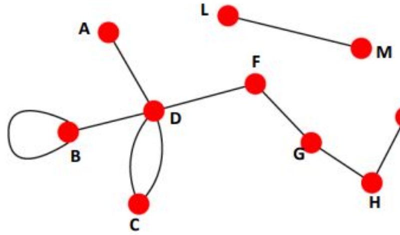
## Basic objects



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

**Undirected**

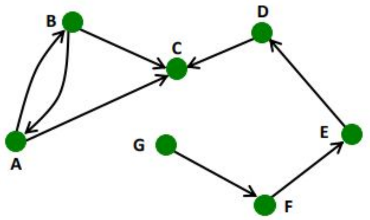
- **Links:** undirected (symmetrical, reciprocal)



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

**Directed**

- **Links:** directed (arcs)



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

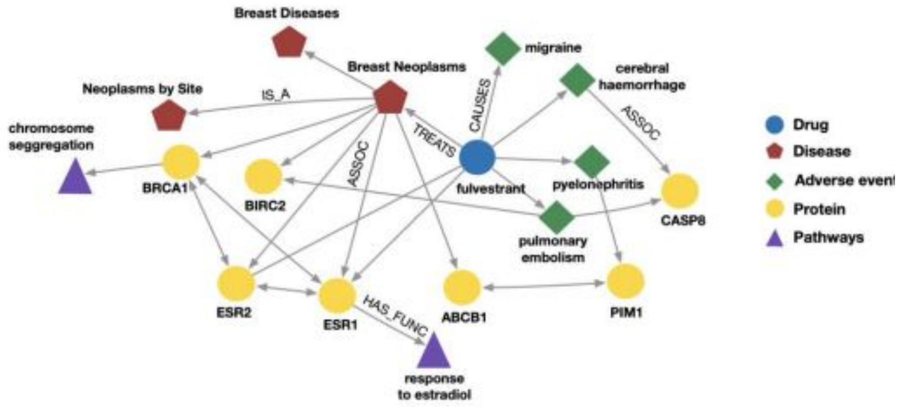
Image credit: Stanford CS224W

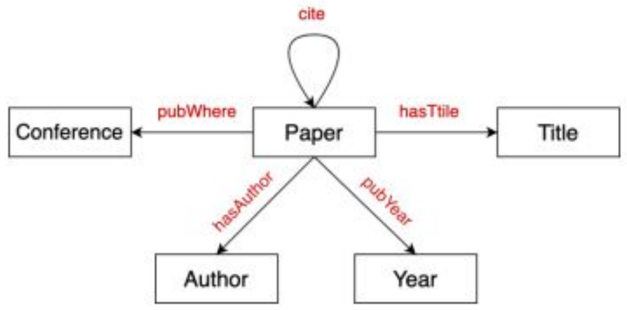
# Heterogeneous graphs

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

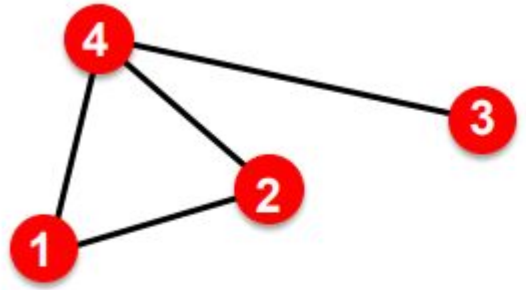
**Academic Graphs**

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

Image credit: Stanford CS224W

# Graph representation

**Undirected graphs**



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

**Adjacency matrix**

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

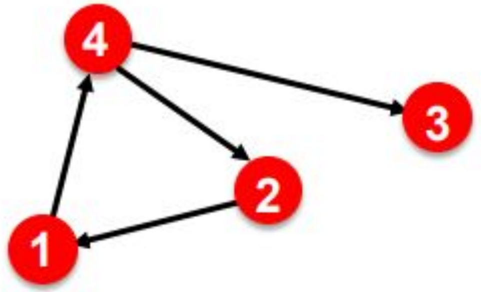
**Edge list**

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)

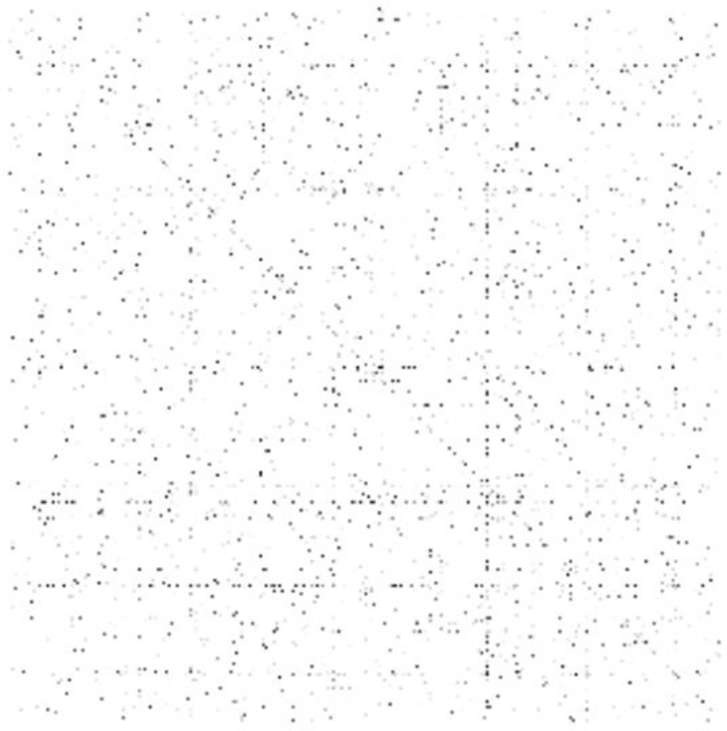
(4, 2), (4, 3)

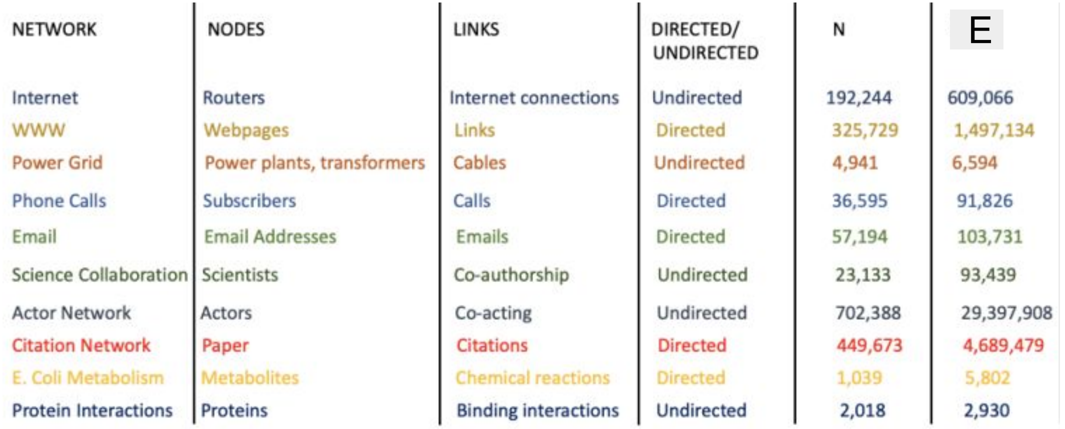
**Edge list**

| | |
|---|---|
| 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 |

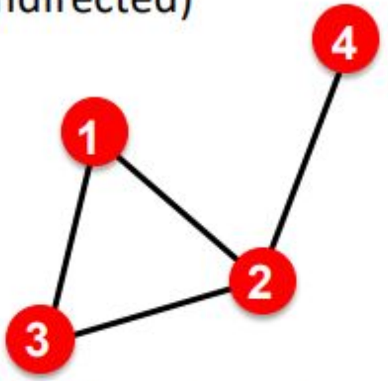
Density = |E|/|N|^2

**Real-world graphs are often very sparse**

Image credit: Stanford CS224W

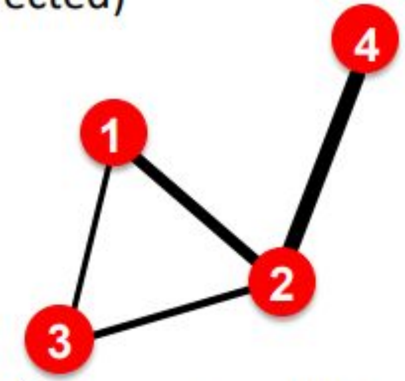
## Weighted graphs

- **Unweighted**
  (undirected)



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

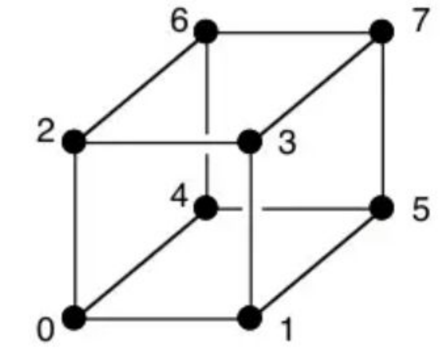
- **Weighted**
  (undirected)

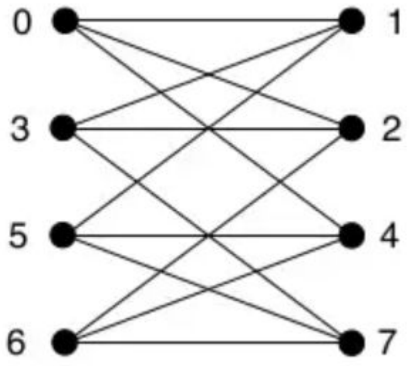


$$\begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

Image credit: Stanford CS224W

# Graph isomorphism/equivalence





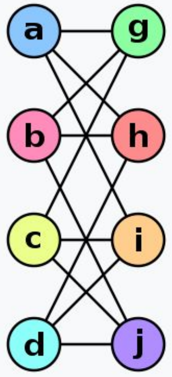
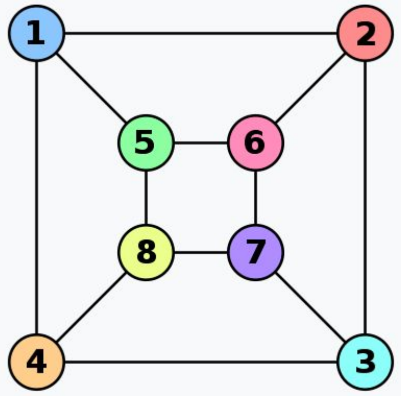
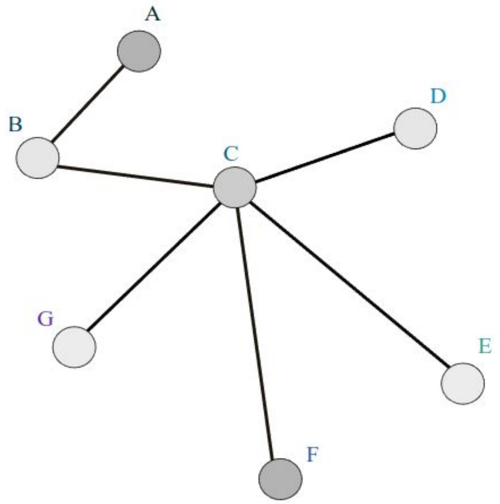
| Graph G | Graph H | An isomorphism between G and H |
|---|---|---|
|  |  | $f(a) = 1$<br>$f(b) = 6$<br>$f(c) = 8$<br>$f(d) = 3$<br>$f(g) = 5$<br>$f(h) = 2$<br>$f(i) = 4$<br>$f(j) = 7$ |

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

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

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

# Permutation invariance



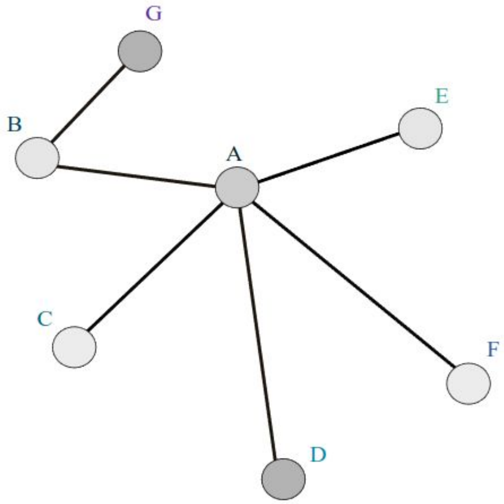


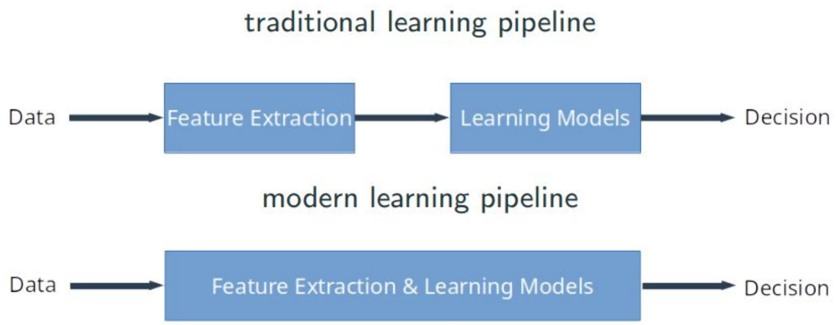
Image credit: Image credit: https://distill.pub/2021/understanding-gnns/

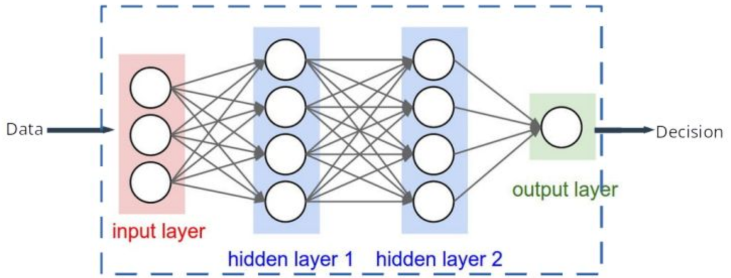
**Permutation invariance**: permuting the names of the nodes doesn't change the graph, as graph nodes are intrinsically orderless

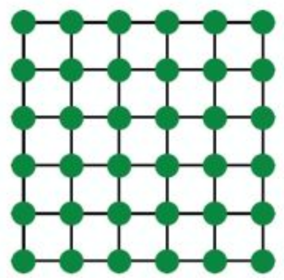
**Mathematically**: if $A$ is (the adjacency matrix of) a graph, $\Pi A \Pi^\intercal$ is (the adjacency matrix of) an equivalent graph for **any permutation matrix** $\Pi$

## Graph neural networks (GNNs)

# Representation learning for graphs

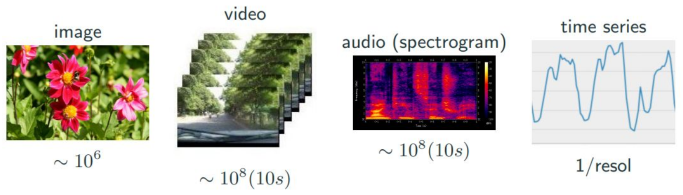


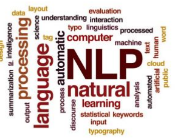




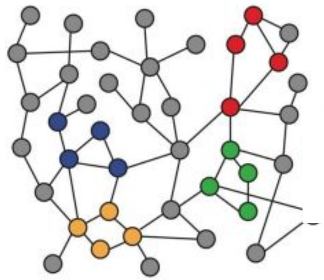
Grid

List

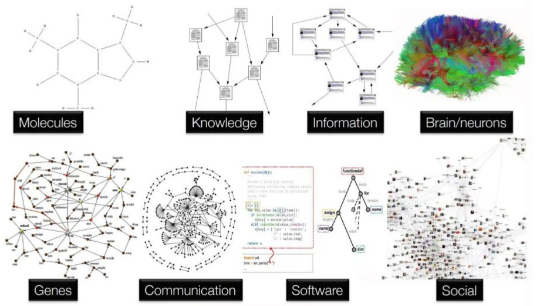




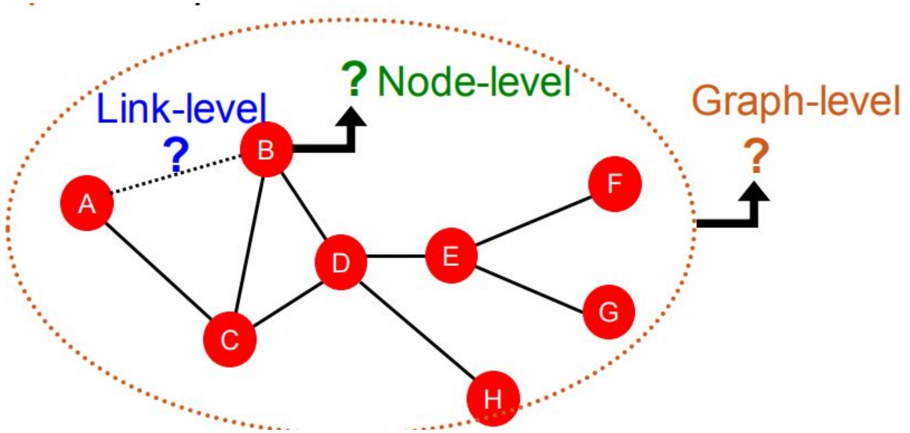
- machine translation, e.g., English ⇆ Chinese
- typing/writing prediction (smart compose)
- semantic classification

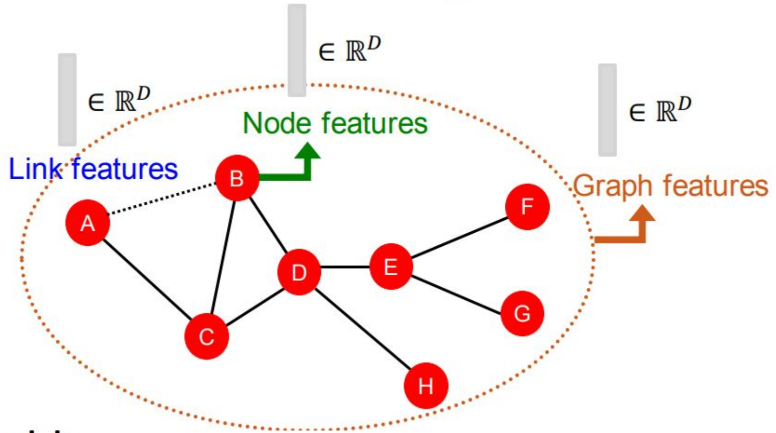


Graph



# Where to put the features?





- **Train an ML model:**
  - Random forest
  - SVM
  - Neural network, etc.

$$x_1 \rightarrow y_1$$
$$\vdots$$
$$x_N \rightarrow y_N$$

- **Apply the model:**
  - Given a new node/link/graph, obtain its features and make a prediction

$$x \rightarrow y$$

Image credit: Stanford CS224W

## Node embedding

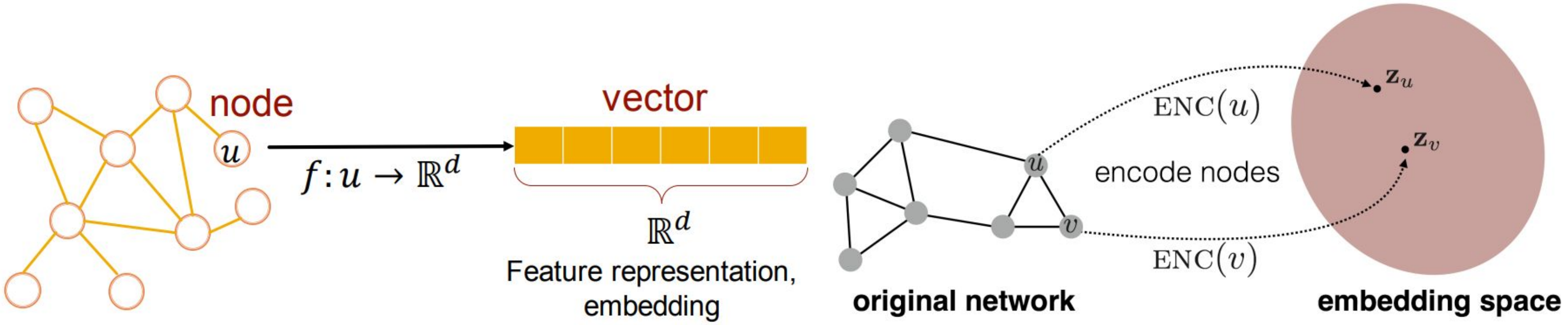


Image credit: Stanford CS224W

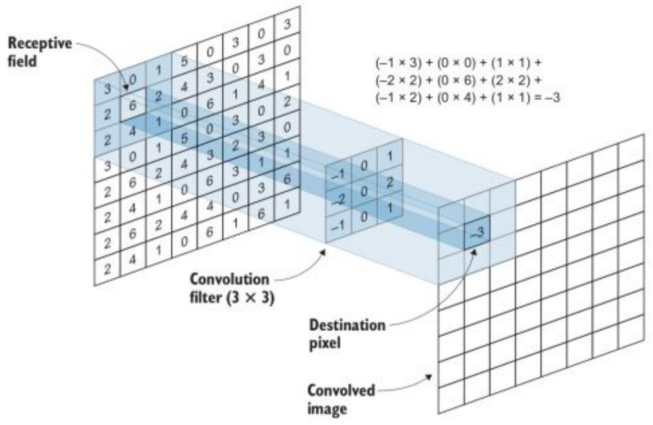
$N$ : set of nodes
$A$ : adjacency matrix
$\boldsymbol{X} \in \mathbb{R}^{|N| \times d}$ : node (raw) features
$u, v$ : nodes in $N$
$\mathcal{N}(u)$ : neighbors of $u$

**Node raw features: e.g.,**

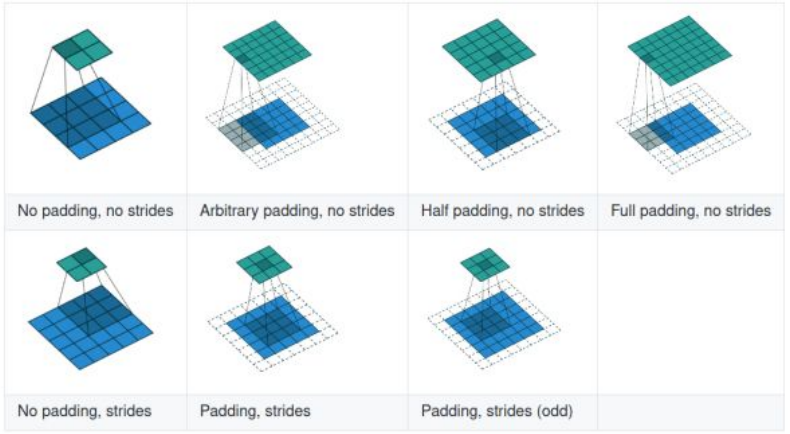
- Biomedical graphs: patient's EHR
- Social network graphs: user profile and images
- When no features: node indicator vector, constant vector

## How to define the $f$ ?

We'll bypass fully connected networks directly



(Credit: [Elgendy, 2020])



https://github.com/vdumoulin/conv_arithmetic

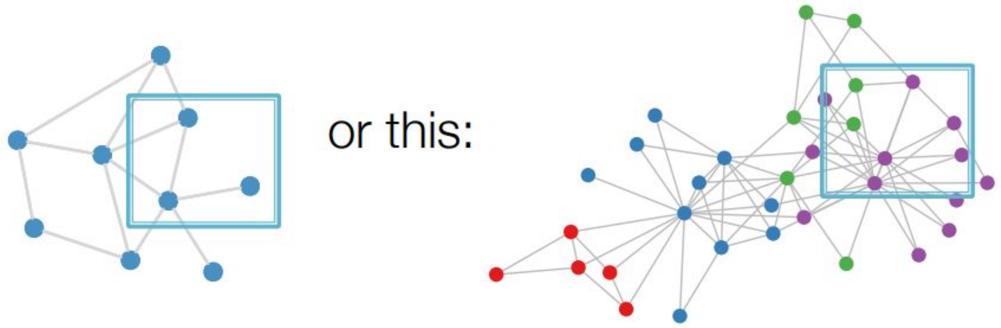
Convolution as performing **local info aggregation** (or **message passing**):

- 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 $f$ ?

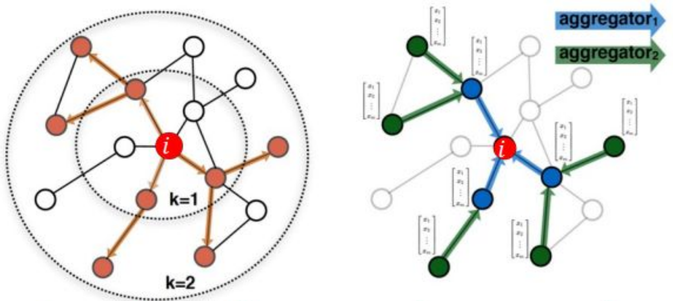
Convolution as performing **local info aggregation** (or **message passing**):

- **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**



We need neighbors on the graph!

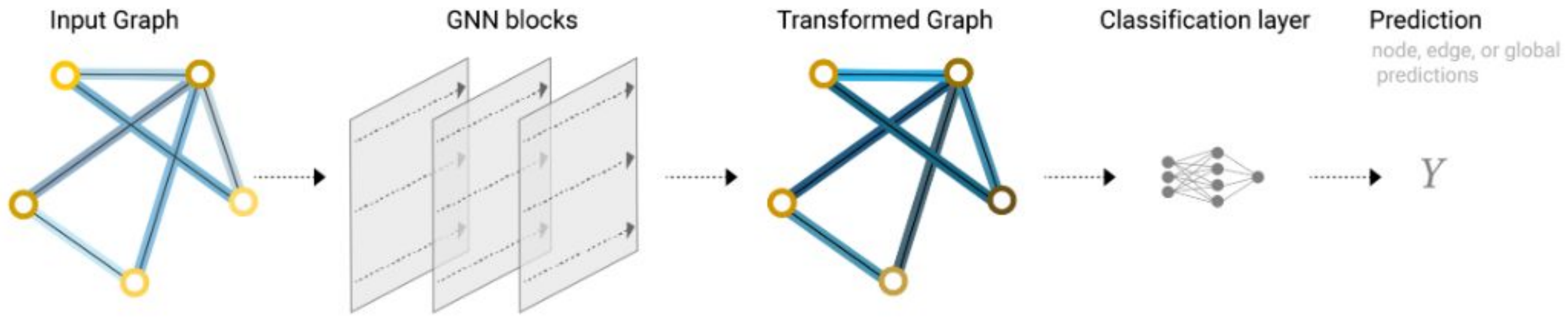
The rigid, grid-based neighborhood doesn't work!



**One layer of a graph neural network (GNN)!**

Image credit: Stanford CS224W

# Graph in, graph out

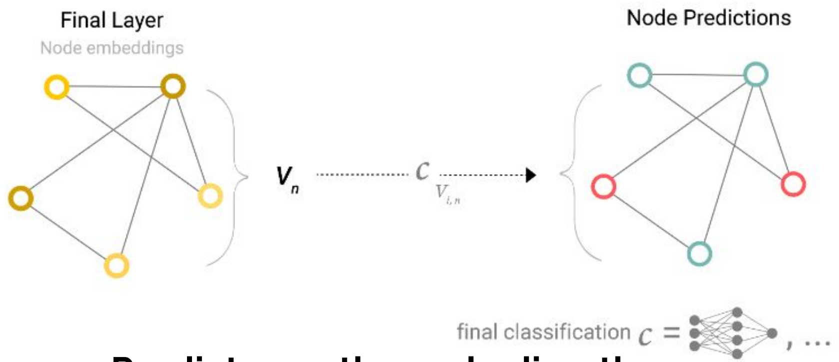


An end-to-end prediction task with a GNN model.

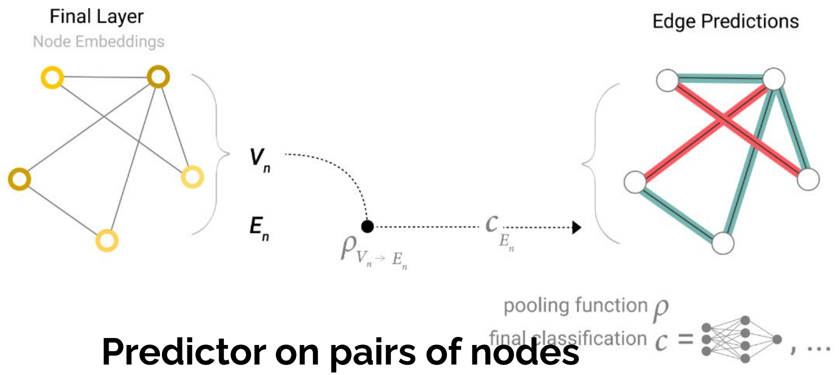
Image credit: https://distill.pub/2021/gnn-intro/

# How to make supervised predictions?

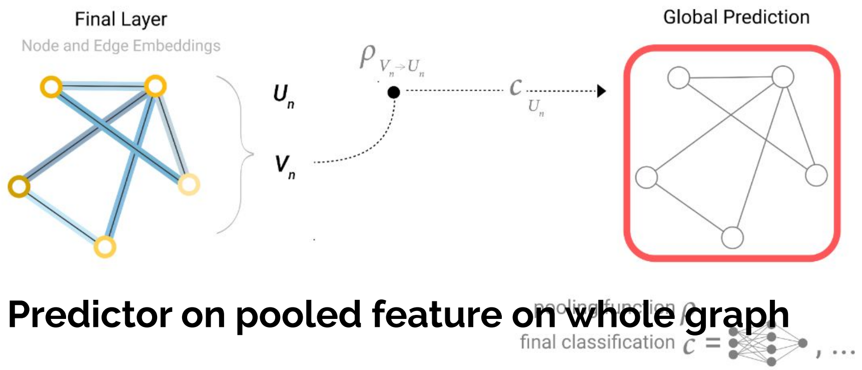
Image credit: [https://distill.pub/2021/gnn-intro/](https://distill.pub/2021/gnn-intro/)



**Predictor on the node directly**

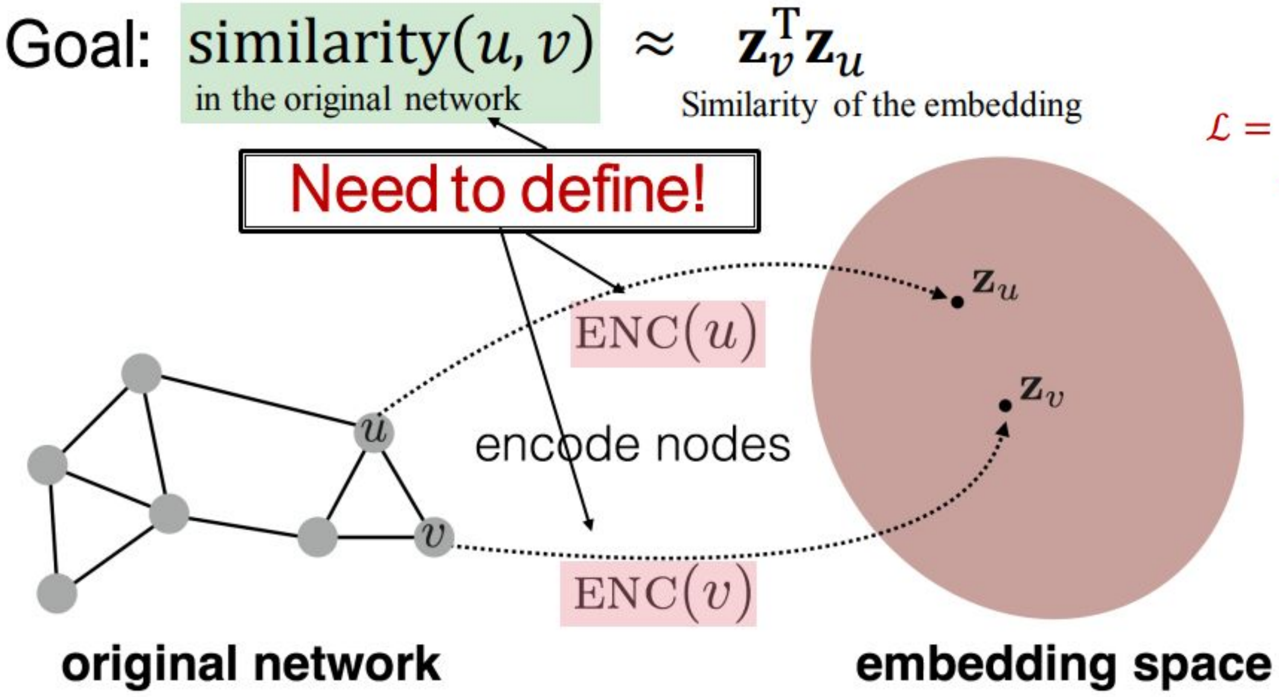


**Predictor on pairs of nodes**



**Predictor on pooled feature on whole graph**

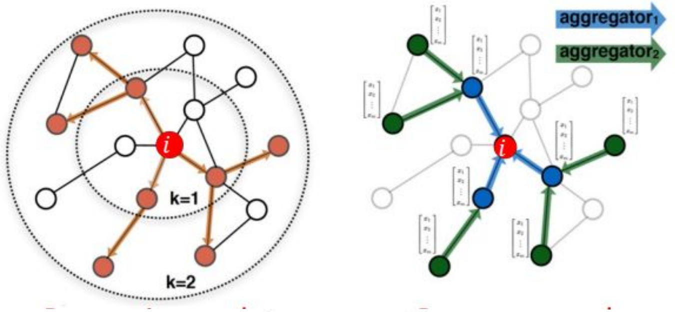
# How to perform unsupervised learning?

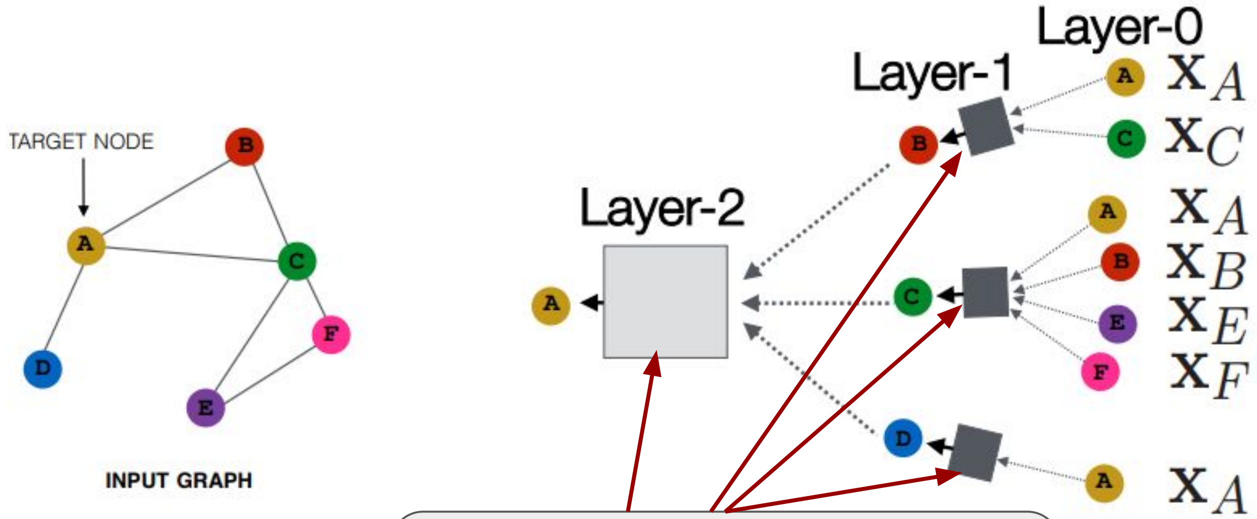


$$\mathcal{L} = \sum_{z_u, z_v} \mathrm{CE}(y_{u,v}, \mathrm{DEC}(z_u, z_v))$$

Image credit: Stanford CS224W

## Look into GNN layers





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

Image credit: Stanford CS224W

## Permutation invariance

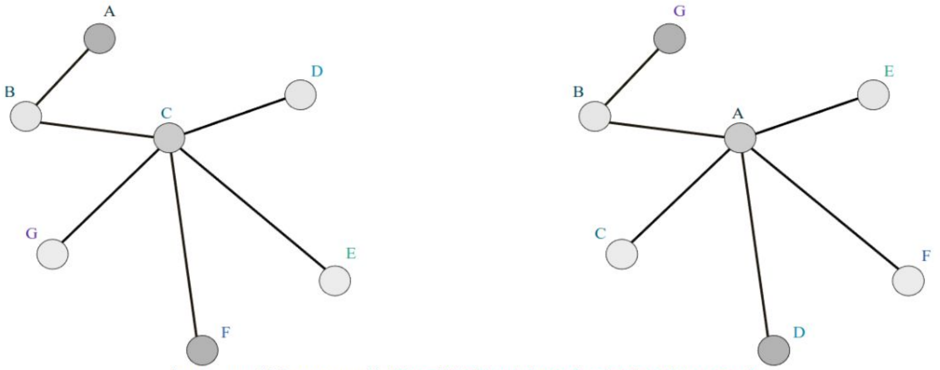
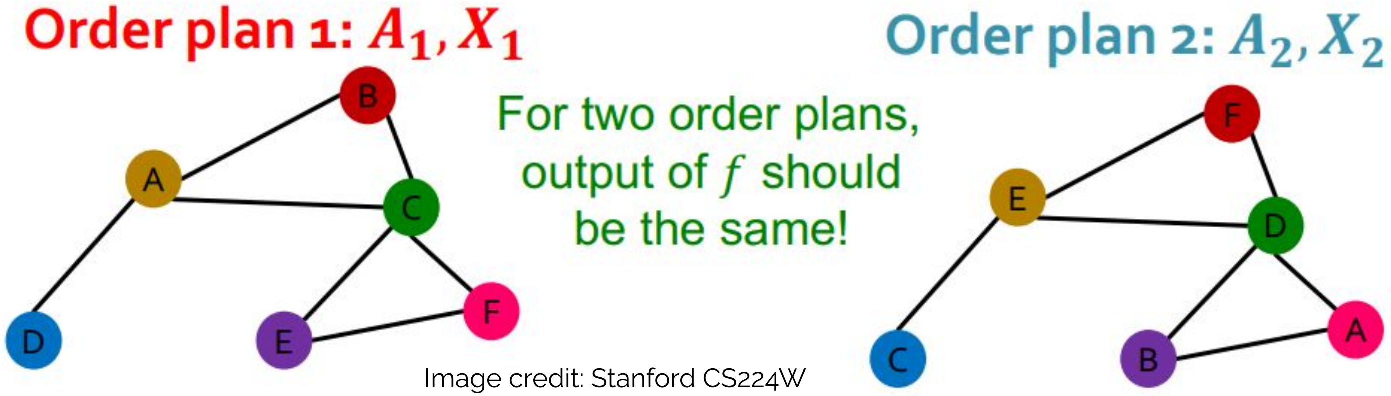


Image credit: Image credit: https://distill.pub/2021/understanding-gnns/

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

Feature vector for the **whole graph**

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



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

Image credit: Stanford CS224W

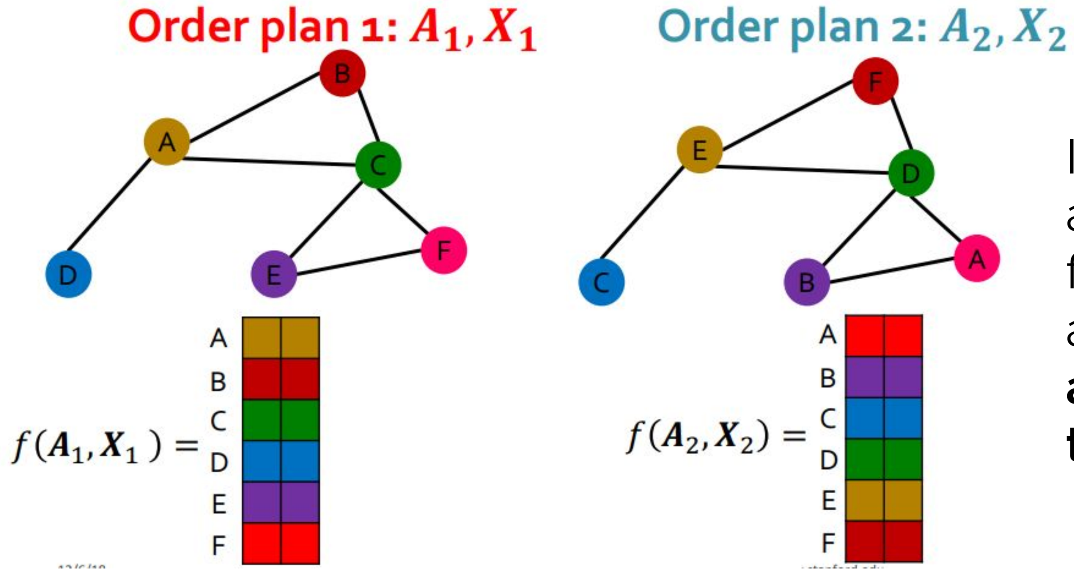
## Permutation equivariance

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 $\boldsymbol{\Pi}$

Collection of feature vectors on all nodes



In other words, if the nodes are re-ordered, the learned features are re-ordered accordingly, **so features are attached to nodes not their names**

Image credit: Stanford CS224W

# A typical GNN consists of multiple permutation equivariant/invariant layers

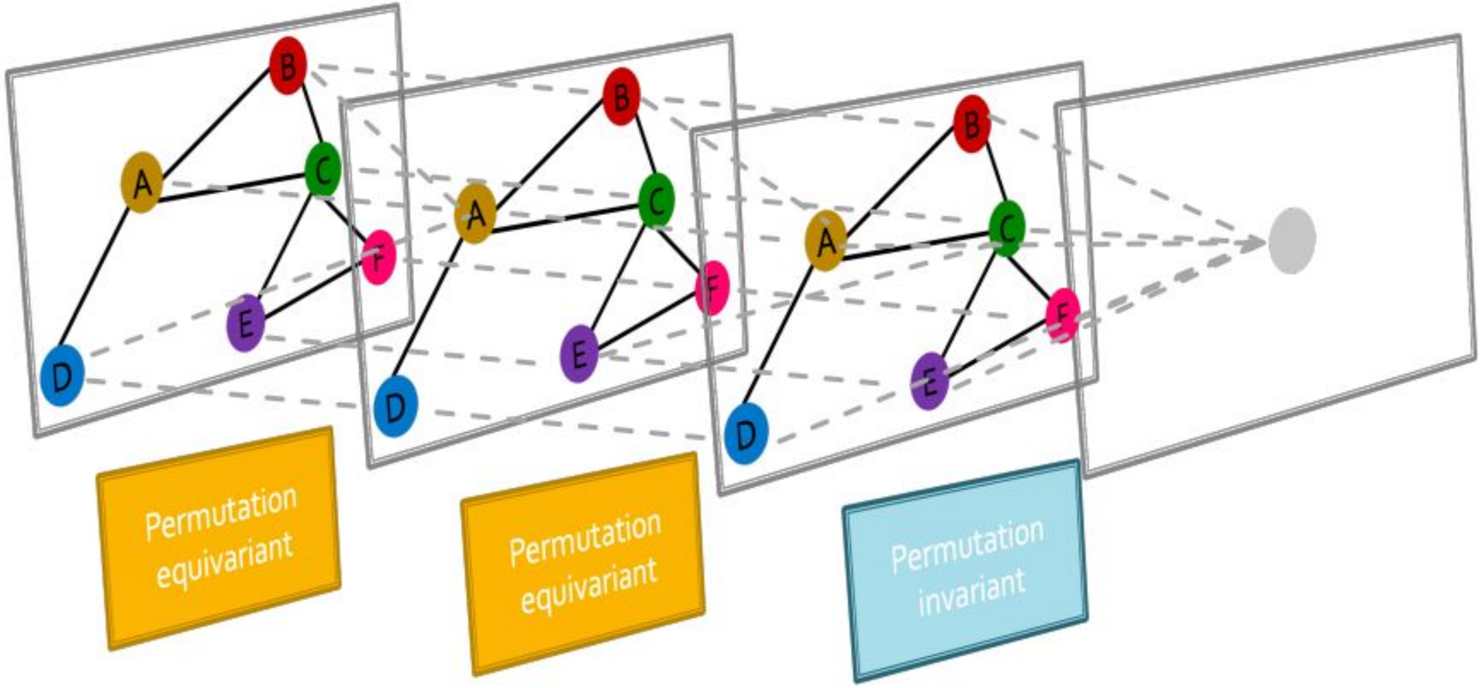
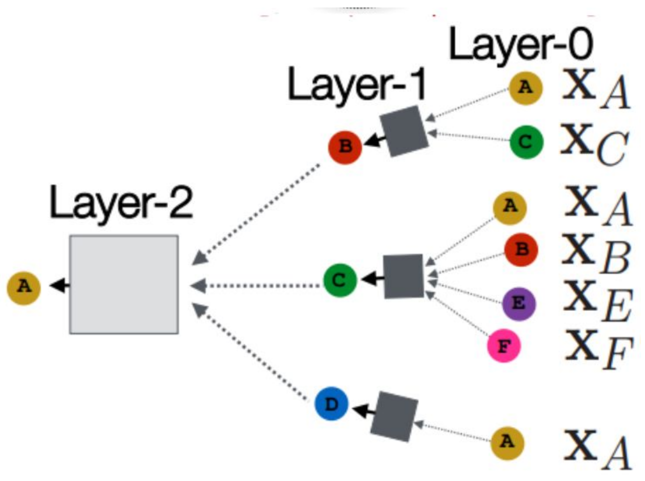
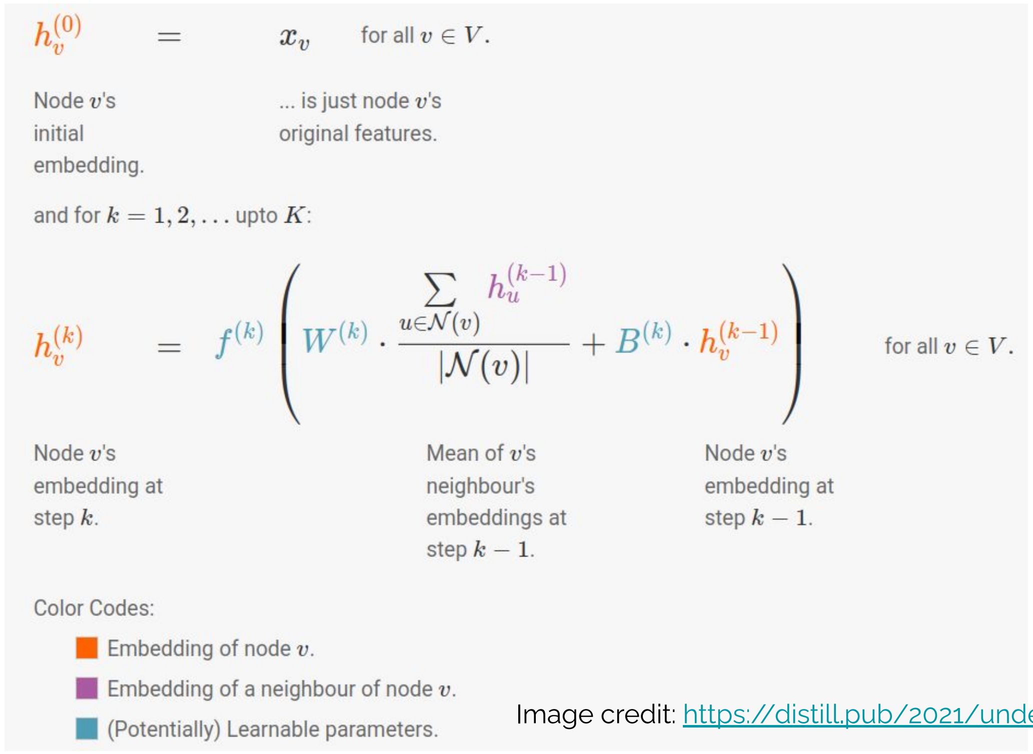


Image credit: Stanford CS224W

# 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 $k-1$.

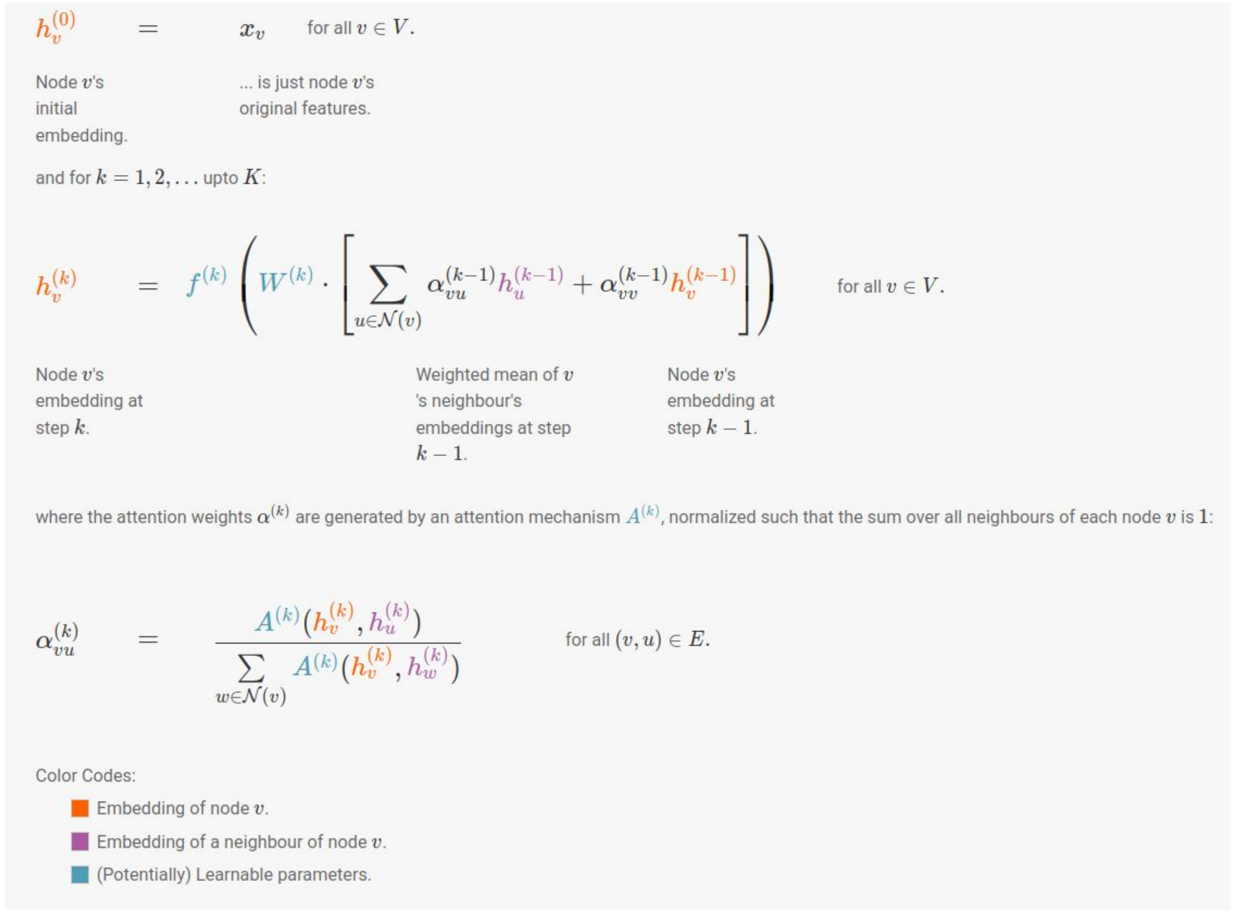
Node $v$'s embedding at step $k-1$.

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 $1$:

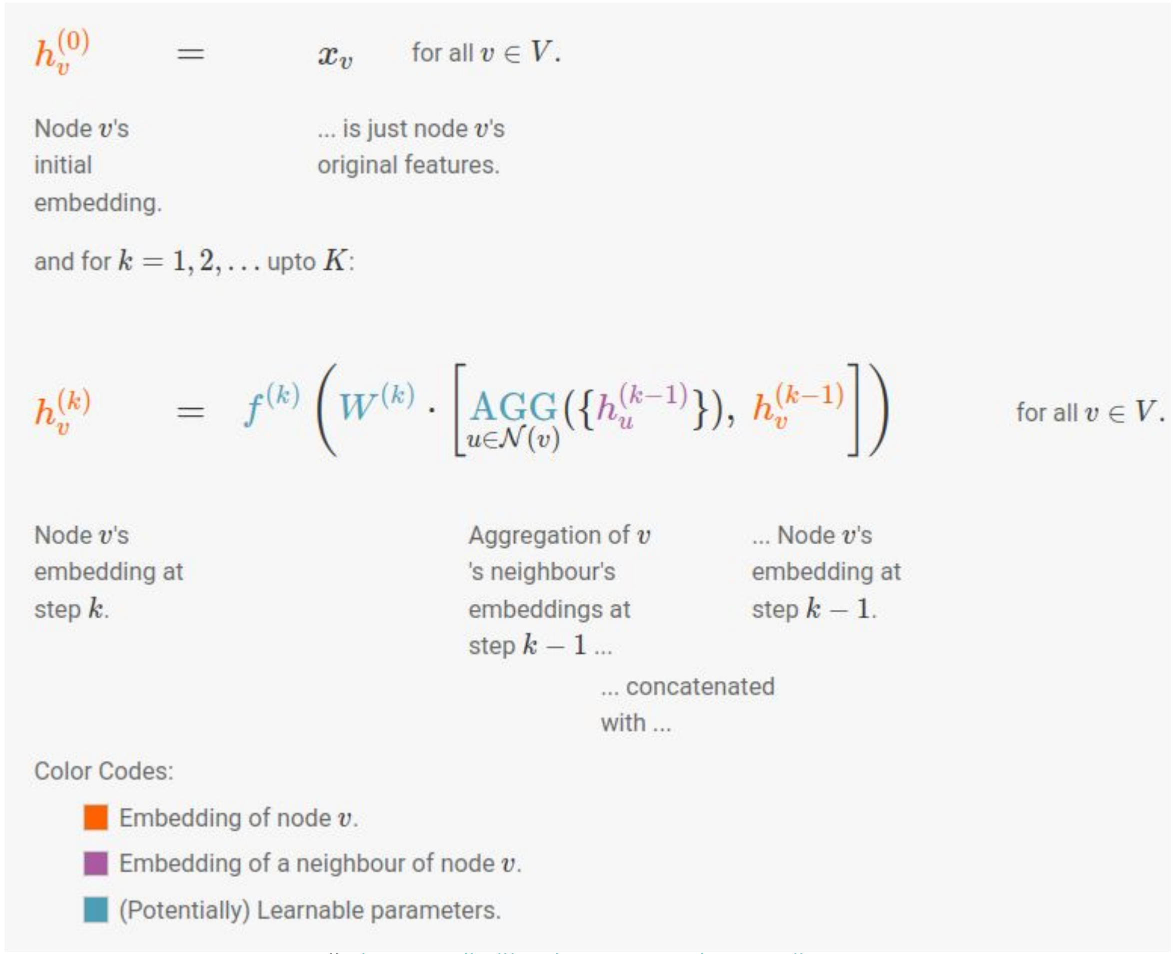
$$\alpha_{vu}^{(k)} = \frac{A^{(k)}(h_v^{(k)}, h_u^{(k)})}{\sum_{w \in \mathcal{N}(v)} A^{(k)}(h_v^{(k)}, h_w^{(k)})} \quad \text{for all } (v, u) \in E.$$

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 ...

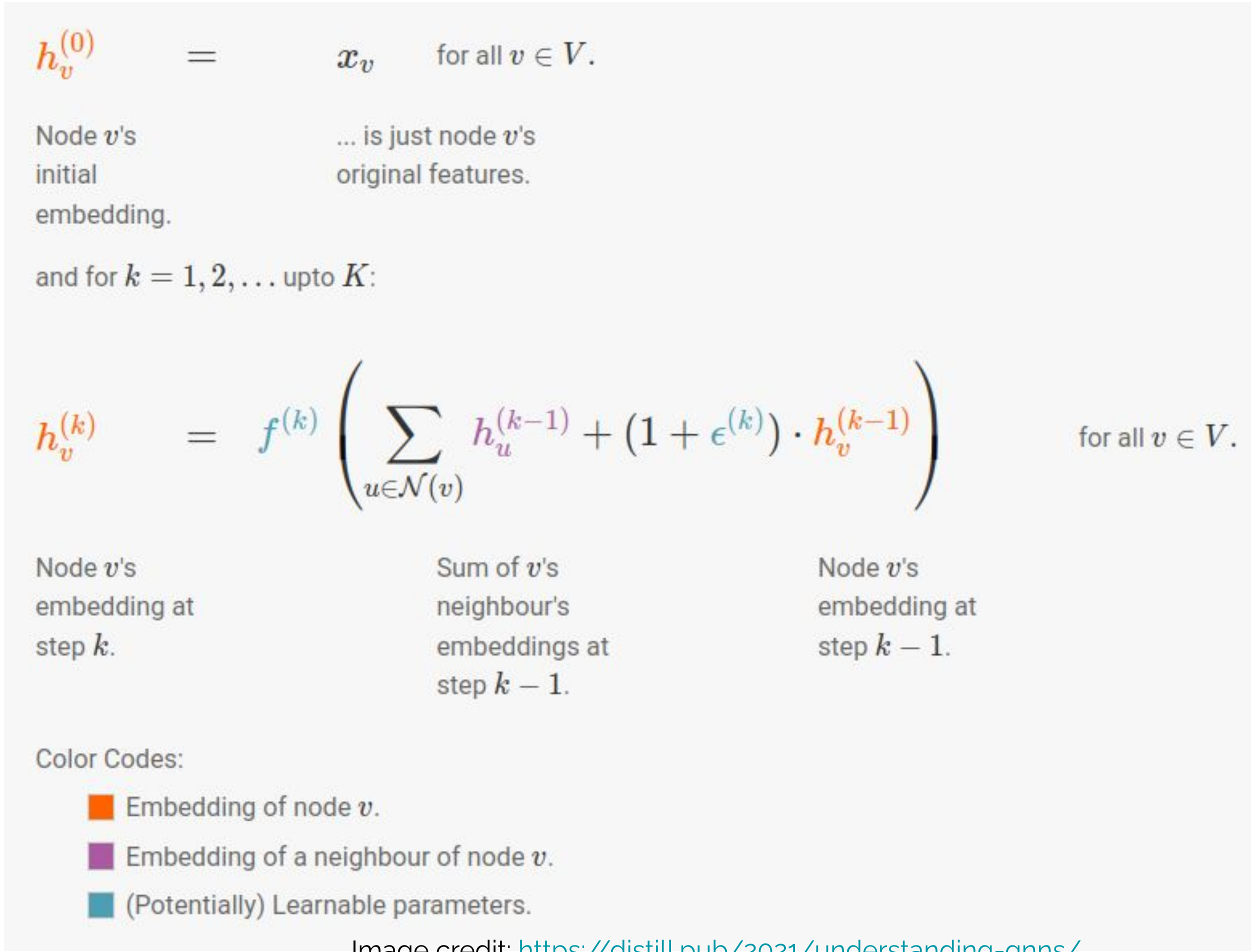
... Node $v$'s embedding at step $k - 1$.

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 $K$:

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

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

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

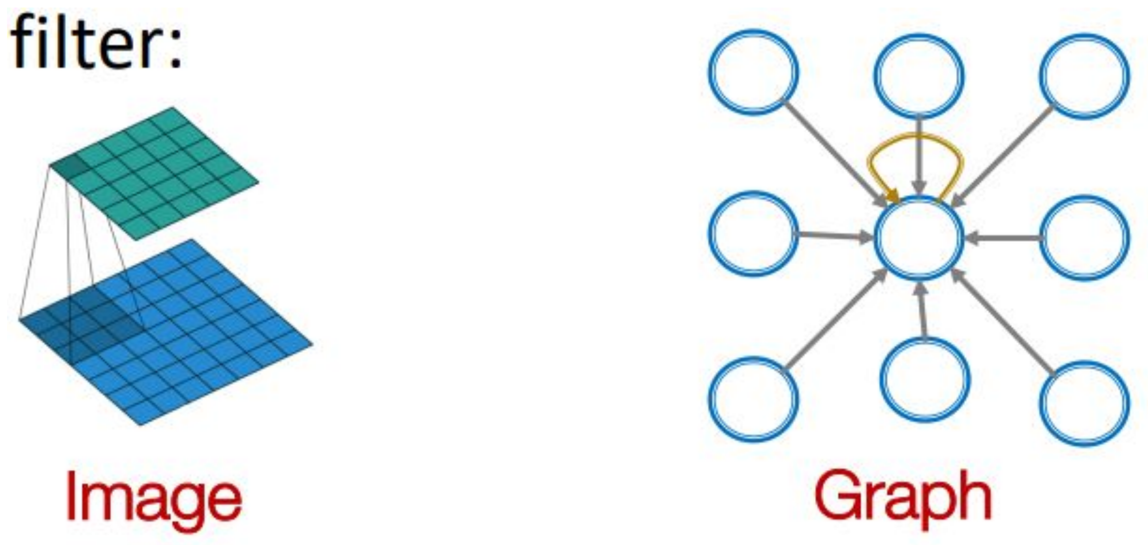
Node $v$'s embedding at step $k-1$.

Color Codes:

- 🟧 Embedding of node $v$.
- 🟪 Embedding of a neighbour of node $v$.
- 🟦 (Potentially) Learnable parameters.

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

# Connection to CNNs and Transformers



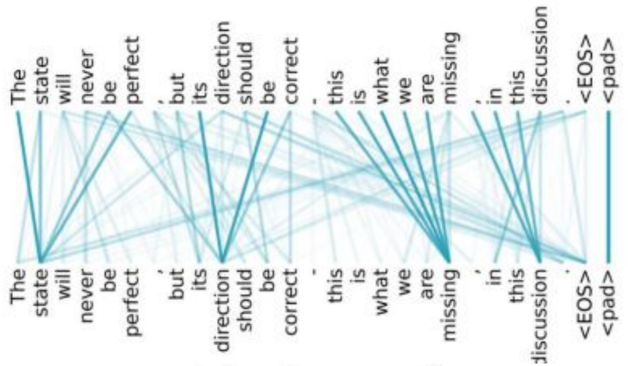
- CNN is GNN that keeps local ordering
- CNN not permutation-invariant

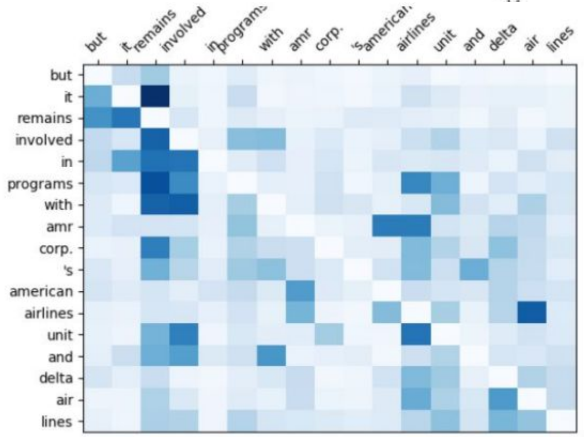
GNN formulation: $\mathrm{h}_v^{(l+1)} = \sigma(\mathbf{W}_l \sum_{u \in \mathrm{N}(v)} \frac{\mathrm{h}_u^{(l)}}{|\mathrm{N}(v)|} + \mathrm{B}_l \mathrm{h}_v^{(l)}), \forall l \in \{0, \dots, L-1\}$

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

Image credit: Stanford CS224W

# Connection to CNNs and Transformers





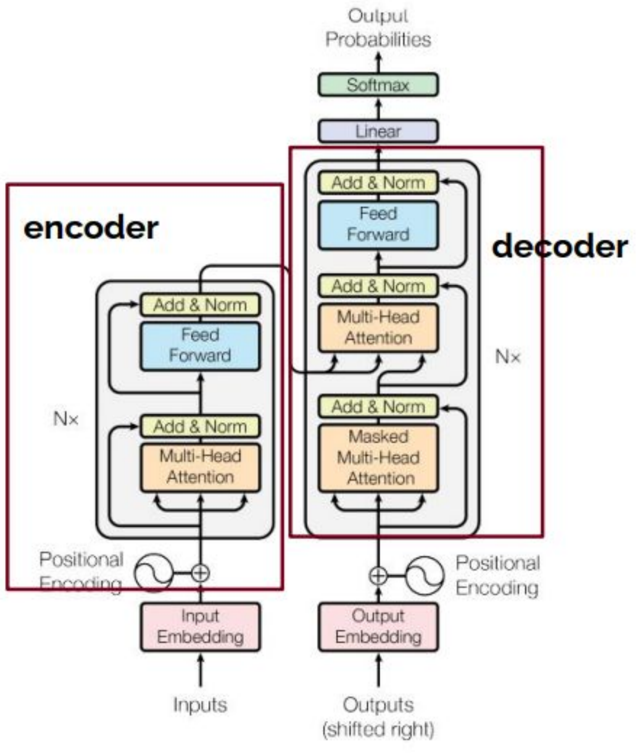


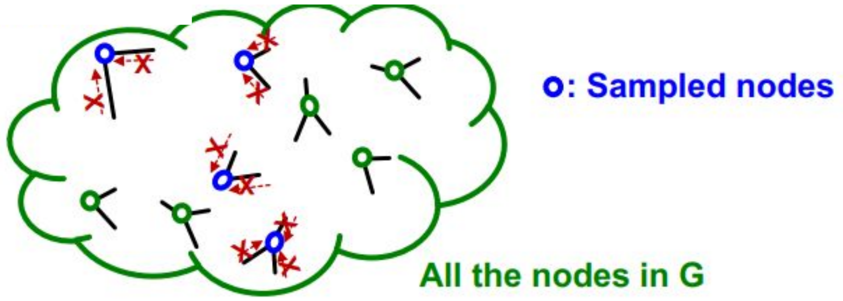
Figure 1: The Transformer - model architecture.

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

## Scaling up training

# Practical graphs are large yet sparse

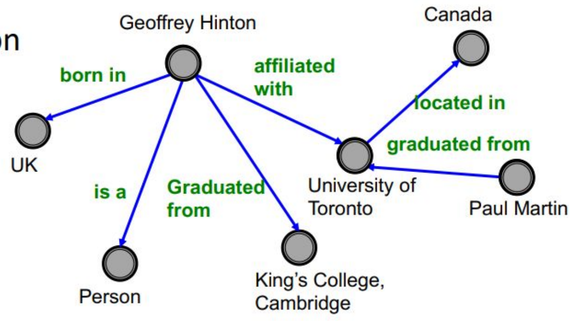
How to perform mini-batch training?



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

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

**Entities**
80M—90M



**Solution**: structured subsampling

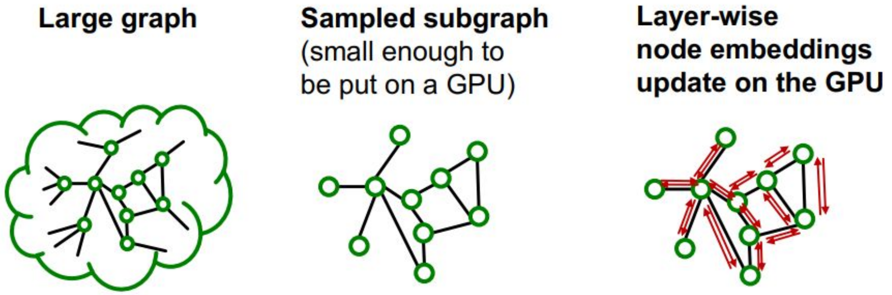
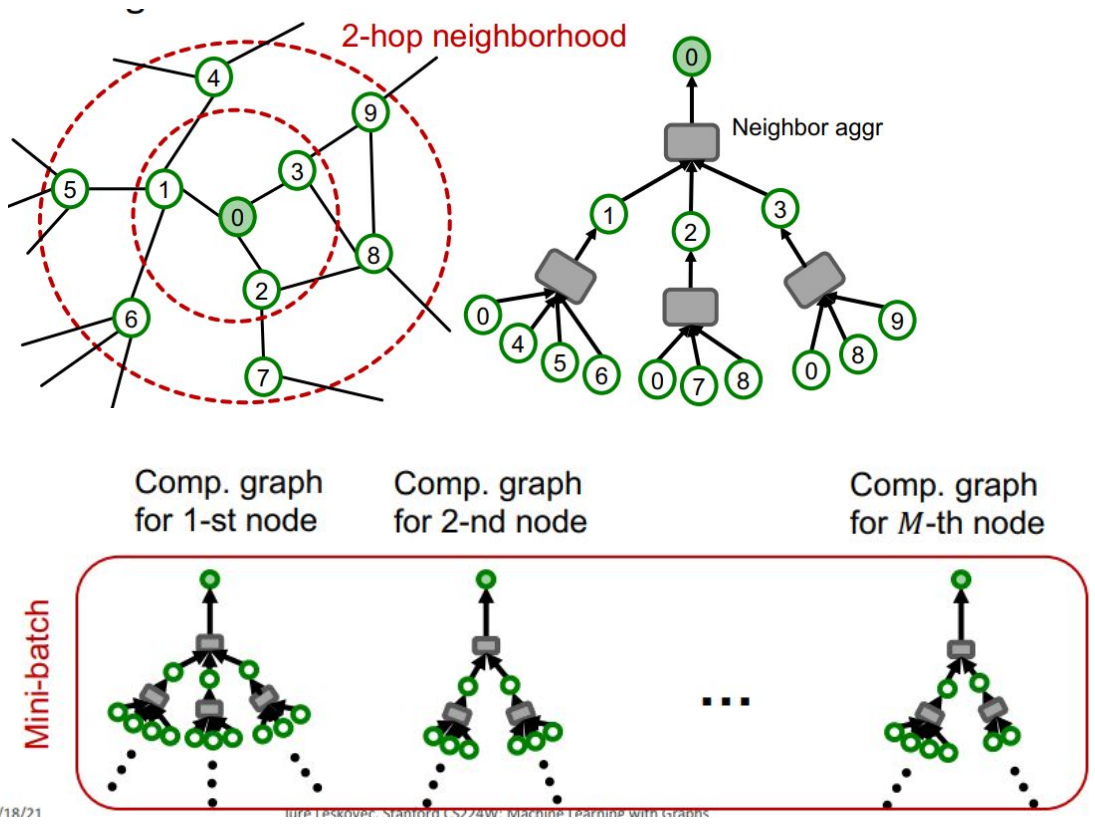
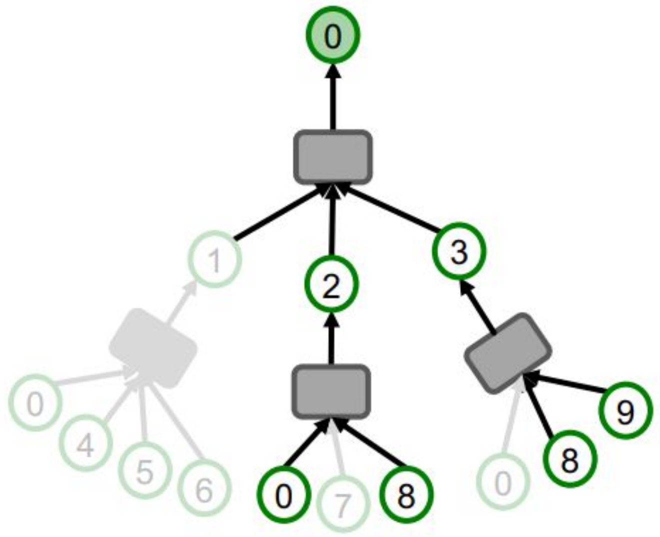

Large graph
Sampled subgraph (small enough to be put on a GPU)
Layer-wise node embeddings update on the GPU

Image credit: Stanford CS224W

# Two structured sub-sampling strategies—I

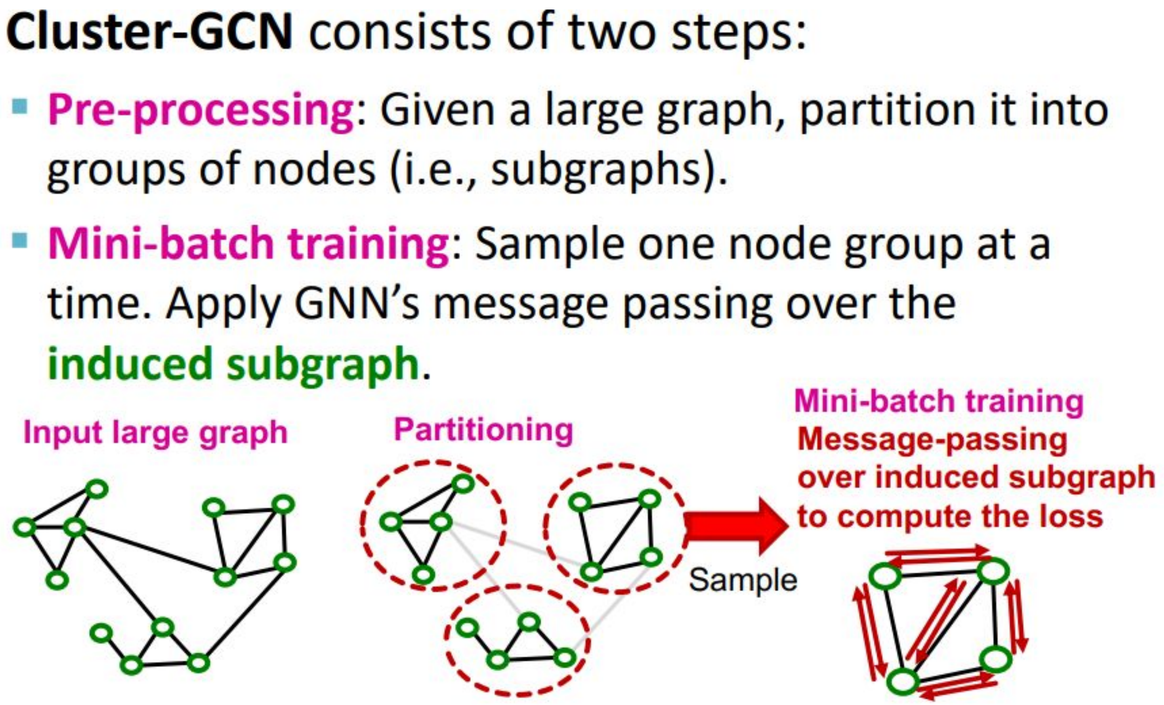

2-hop neighborhood
Neighbor aggr
Comp. graph for 1-st node
Comp. graph for 2-nd node
Comp. graph for $M$-th node
Mini-batch
...



i.e., **neighborhood sampling**, instead of iid uniform sampling

Image credit: Stanford CS224W

# Two structured sub-sampling strategies—II



**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/](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/](https://distill.pub/2021/gnn-intro/)
- Understanding Convolutions on Graphs [https://distill.pub/2021/understanding-gnns/](https://distill.pub/2021/understanding-gnns/)
- Graph Neural Networks: A Review of Methods and Applications [https://arxiv.org/abs/1812.08434](https://arxiv.org/abs/1812.08434)
- Stanford CS224W: Machine Learning with Graphs [https://web.stanford.edu/class/cs224w/index.html](https://web.stanford.edu/class/cs224w/index.html)
- Graph Representation Learning [https://www.cs.mcgill.ca/~wlh/grl_book/](https://www.cs.mcgill.ca/~wlh/grl_book/)