



Modelling Structure with Graph Neural Networks

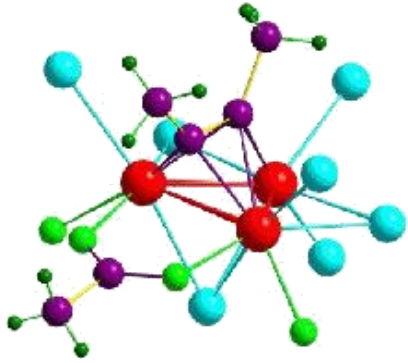
Lei Li



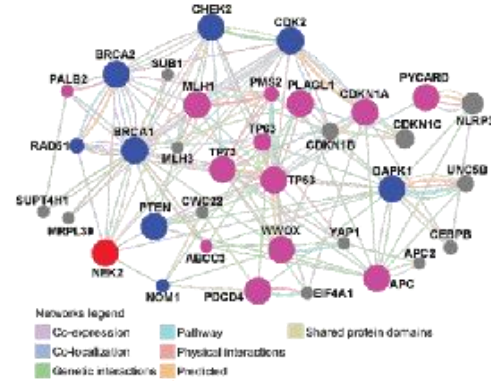
Language
Technologies
Institute

Carnegie Mellon University
School of Computer Science

Graph Data is everywhere



Molecular Graphs



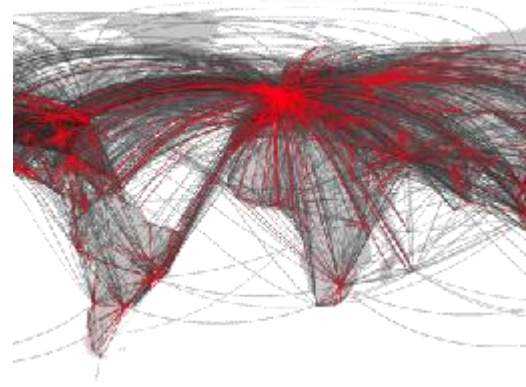
Gene Graphs



Brain Graphs



Web Graphs



Transportation Graphs



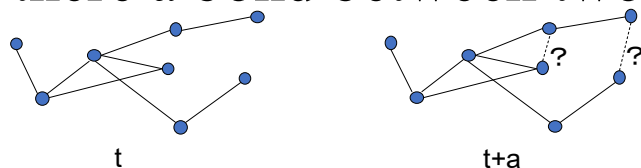
Social Graphs

Tasks on Graph Data

Node-level

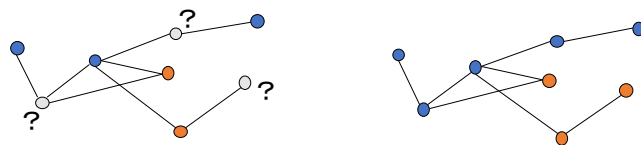
Link Prediction

is there a bond between two atoms?



Node Classification

is this atom C or N?

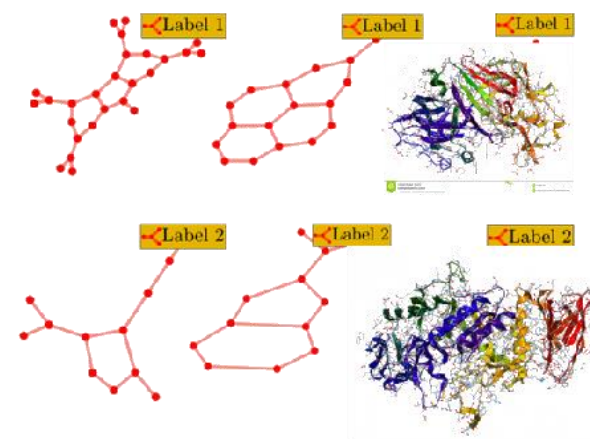


Structure prediction

what are 3D coordinates of atoms/residues?

Graph-level

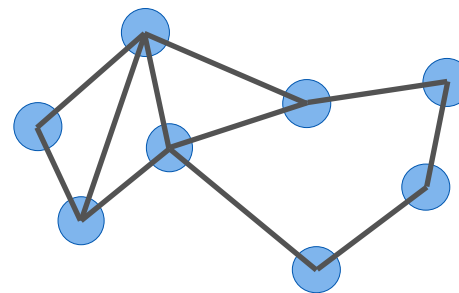
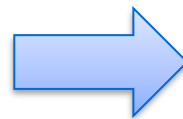
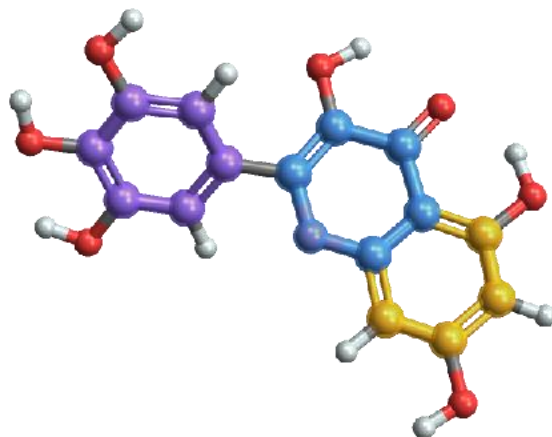
Predicting properties of a graph



Is a molecule toxic to animals?

Example: predict toxicity of a drug

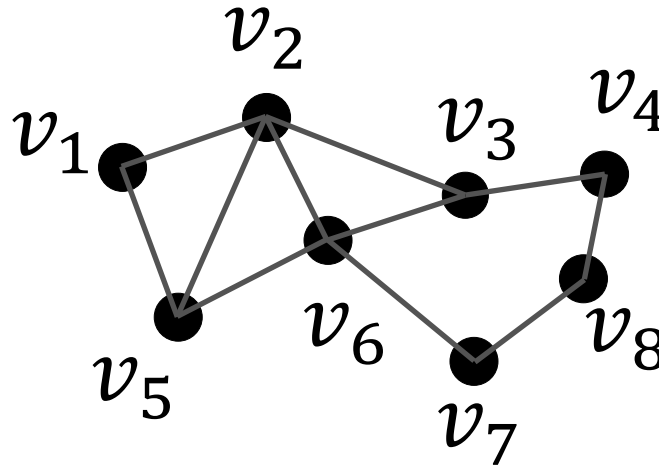
Toxic?



Outline

- Message Passing Neural Network
- Equivariant Graph Neural Network
- Code Example

Graph Representation



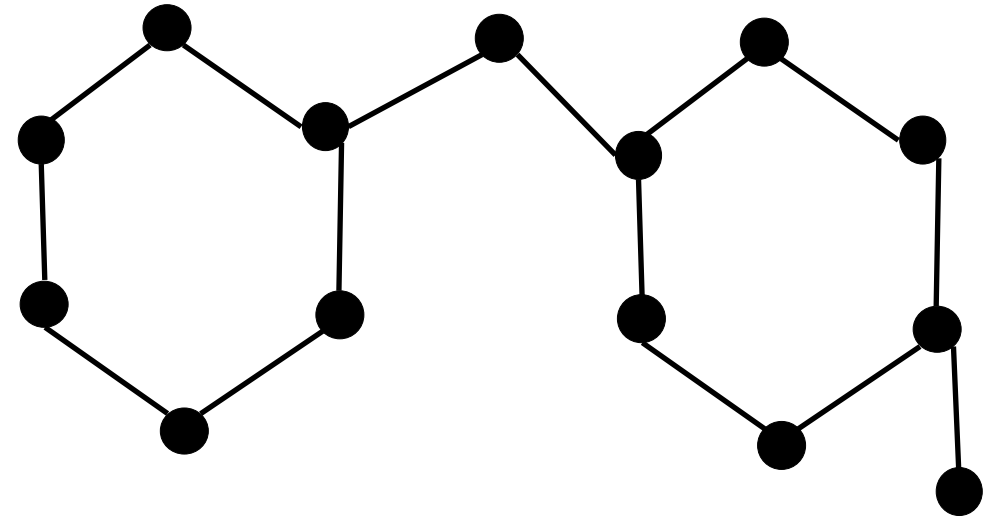
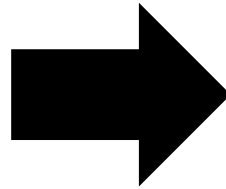
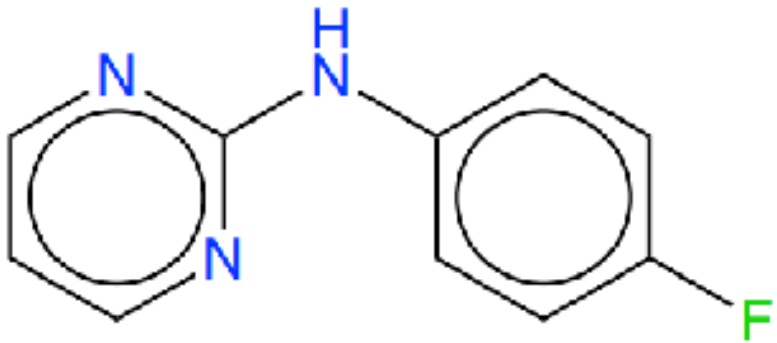
Graph: $G = \{V, E\}$

Nodes: $V = \{v_1, v_2, \dots, v_N\}$

Edges: $E = \{e_1, e_2, \dots, e_M\} \subset V \times V$

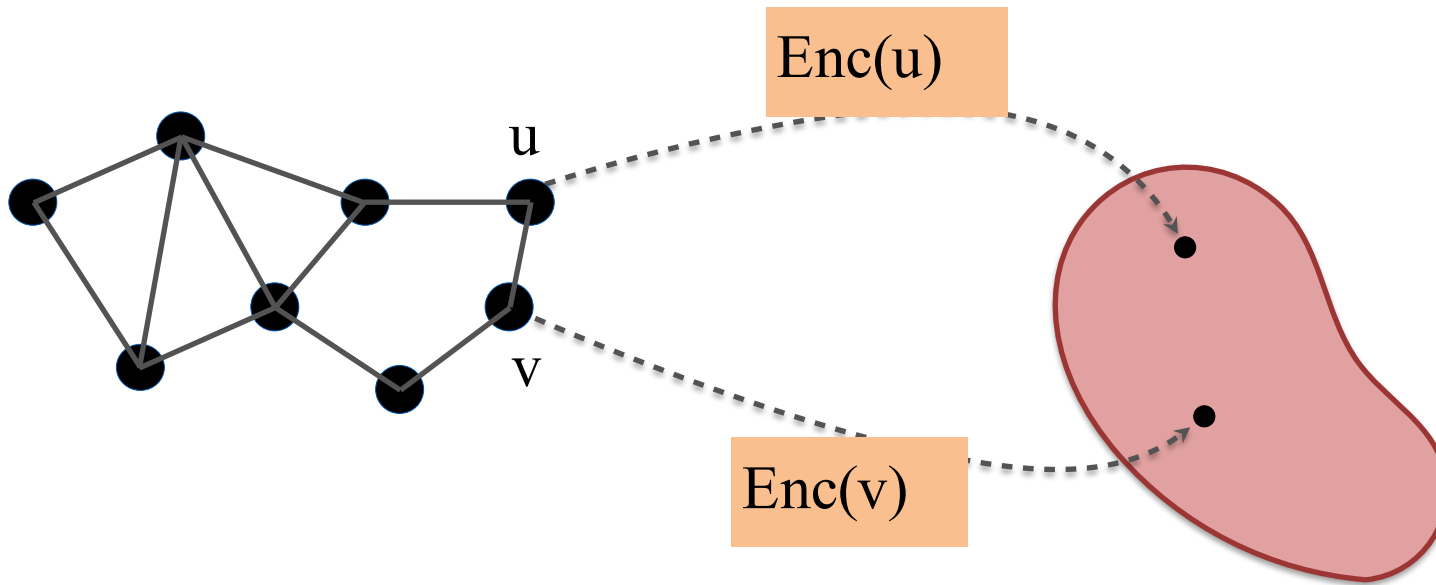
Atom-level Molecule Graph

- Node: heavy atoms (non-hydrogen)
- Edge: bond



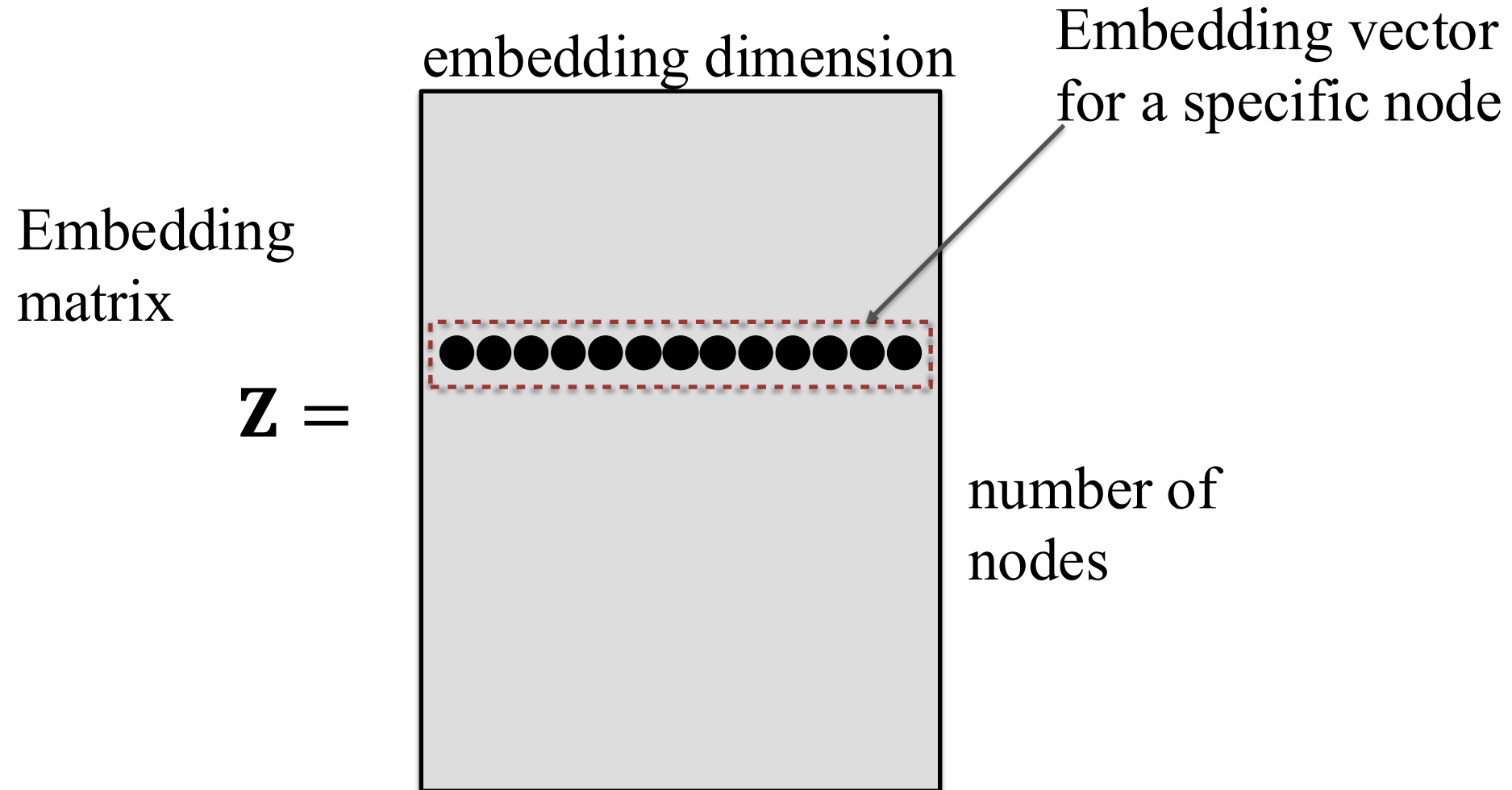
Node Embedding

$$Enc(\cdot): V \rightarrow \mathbb{R}^d$$

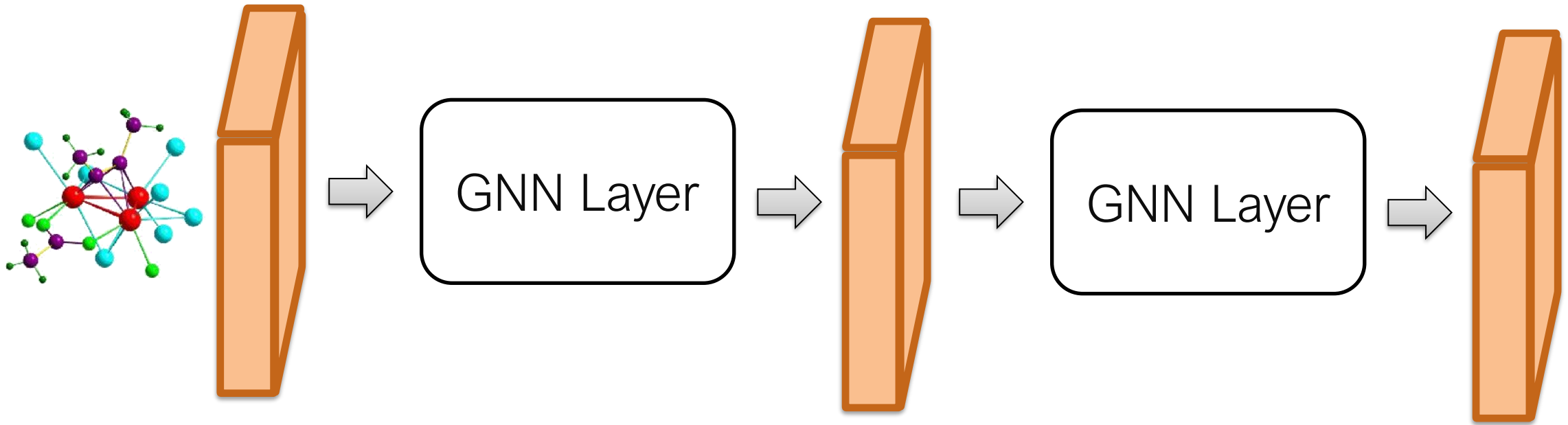


“Shallow” Node Embedding

- is just a lookup-table



Graph Neural Network



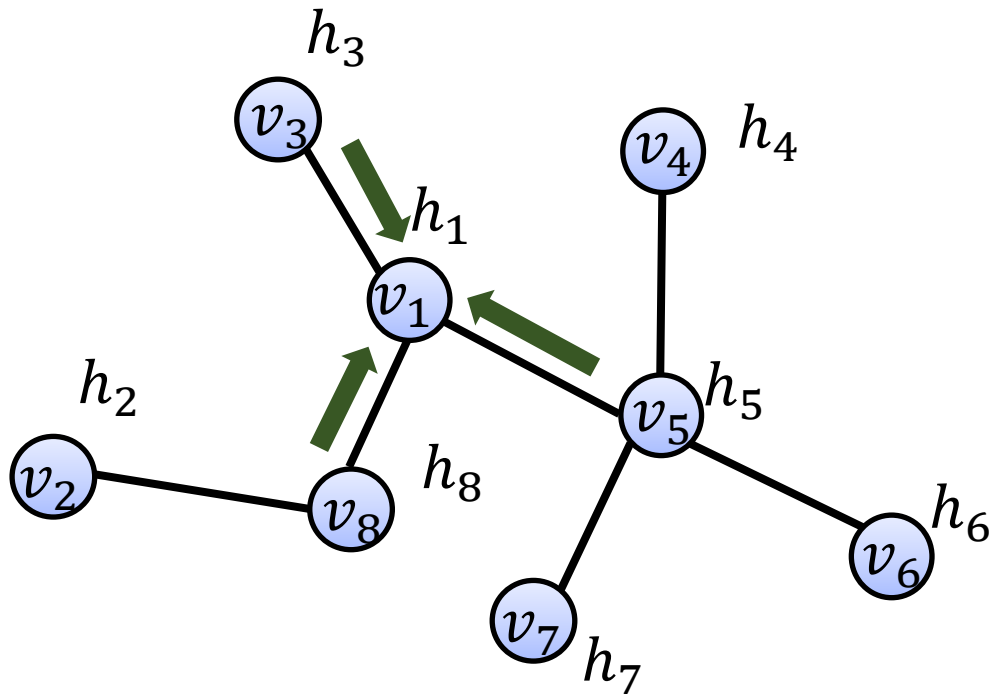
Output is an embedding matrix for nodes
for further downstream tasks: e.g. node property prediction

GNN-based Foundation Models for Biomedicine

- Predicting docking
 - EquiDock
 - EquiBind
- Property Prediction
 - EnviroDetaNet
 - TorchMD-NET
- Modeling Protein
 - GearNet
- Gene regulator network
 - DeepSEM
- Drug design
 - JT-VAE
 - GraphAF

Message Passing Neural Network

Every node receives message from its neighbors, aggregate and update



h_i : node embedding vector

aggregate information from its neighbors

$$h_i^{k+1} = U^k \left(\text{Aggregate}_{v_j \in N(v_i)} m_{j \rightarrow i}^k \right), \forall v_i \in V$$

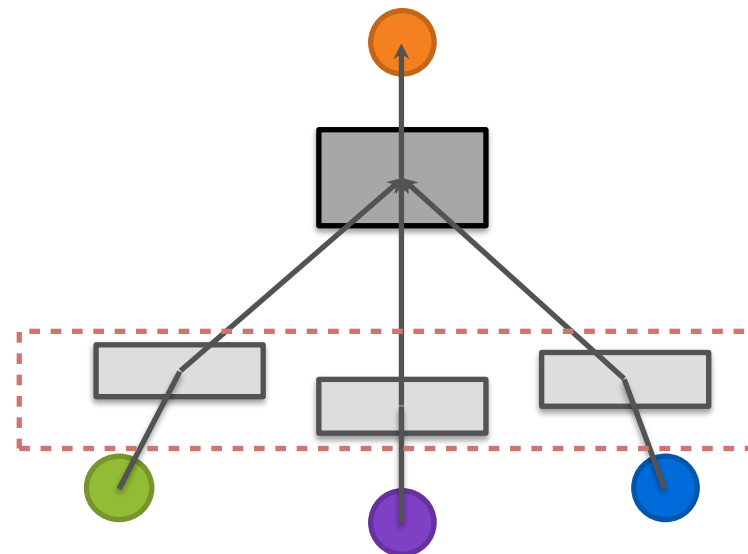
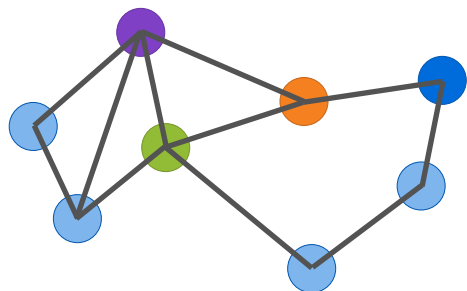
$N(v_i)$: Neighbors of the node v_i

$m_{j \rightarrow i}^k$: message from node j to i at k -th layer

Aggregate can be avg, sum, max/min etc.

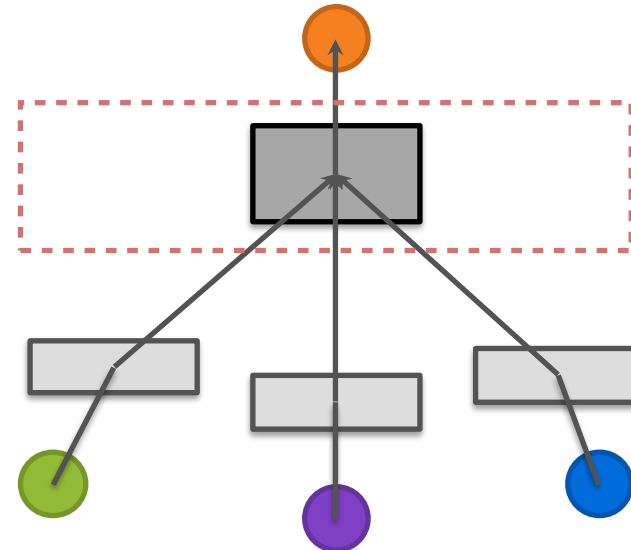
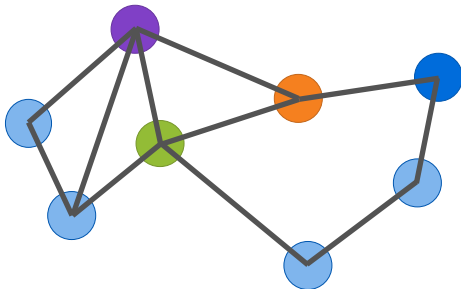
MPNN Message Computation

- Each node will create a message $m_{j \rightarrow i} = M^k(h_i^k, h_j^k, e_{ij})$
 - Dependent on incoming node (h_j^k), target node (h_i^k), edge feature
- e.g. Linear projection or FFN
 $\sigma(W_k \cdot [h_i^k, h_j^k, e_{ij}])$



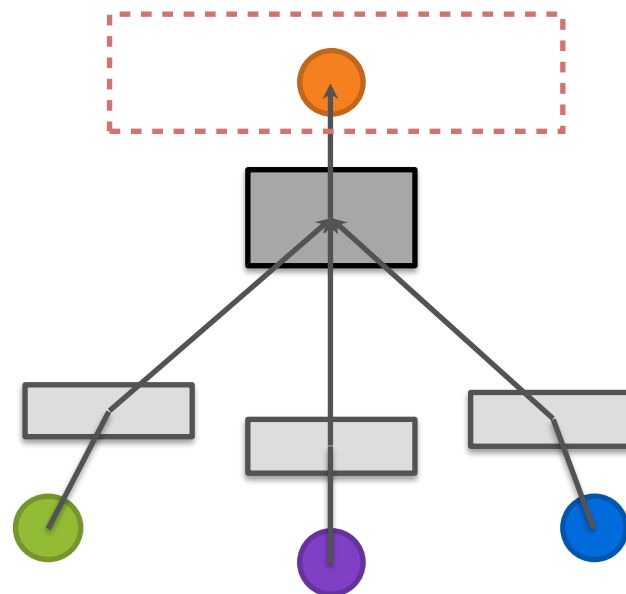
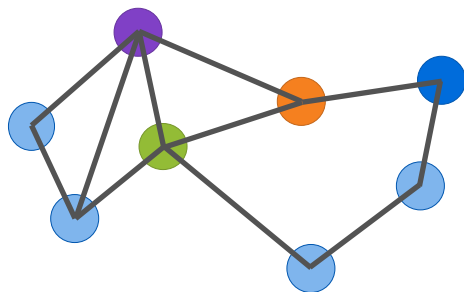
Message Aggregation/Pooling

- Each node will aggregate messages from its neighbors
- Aggregate function can be
 - Sum, Mean, Max operator
 - $m_i^{k+1} = \sum_{v_j \in N(v_i)} M^k(h_i^k, h_j^k, e_{ij})$



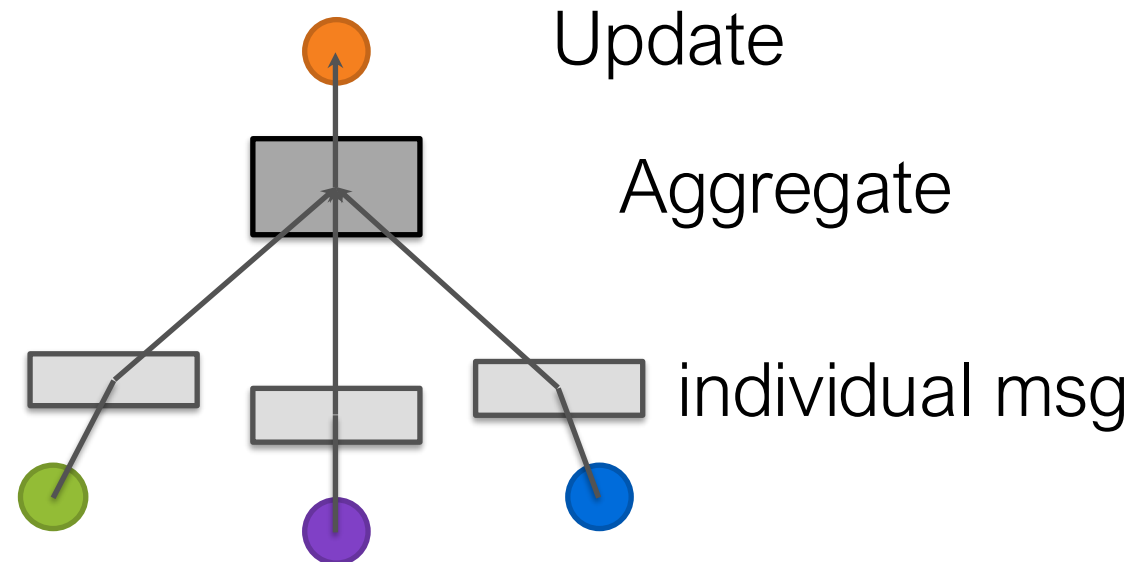
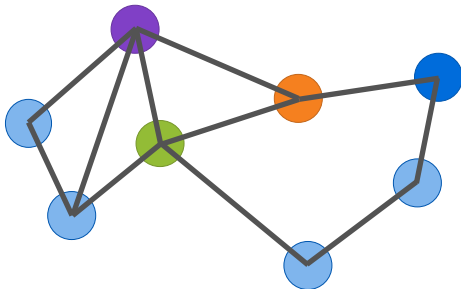
Message Update

- Combine message and current node embedding
- Apply FFN
 - $h_i^{k+1} = U^k(h_i^k, m_i^{k+1})$



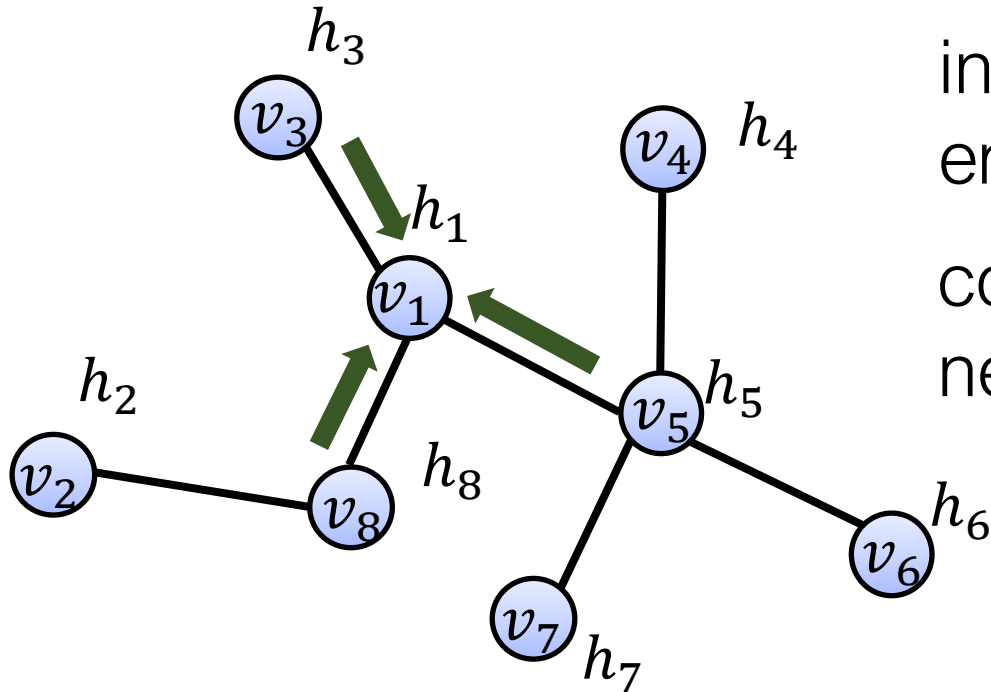
Generic framework: Message Passing NN

- MPNN = Message passing + Aggregation + Update
 - propagate nonlinear messages along graph topology
 - flexible and generic design choices under this framework
 - Graph convolutional network (GCN)
 - GraphSAGE
 - Graph Attention Network (GAT)



A Simple Graph Convolution Network

- averaging neighbor's message and apply nonlinear transformation



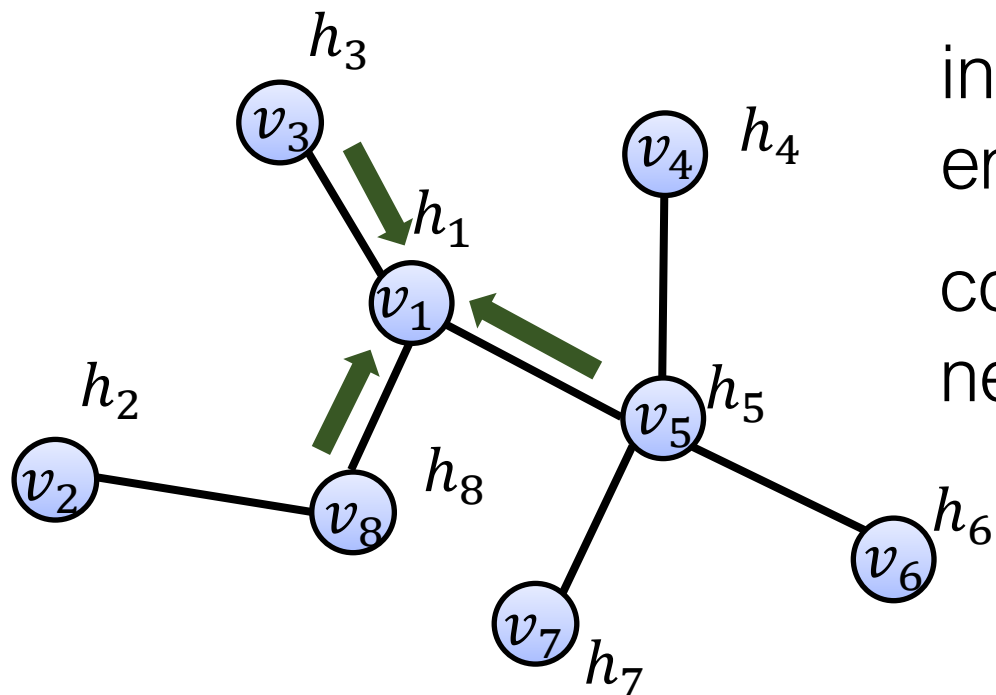
initial
embedding: $h_i^0 = x_i$

computing
next layer: $h_i^{k+1} = \sigma \left(W_k \frac{1}{|N(v_i)|} \sum_{v_j \in N(v_i)} h_j^k + B_k h_i^k \right)$

example: $h_1^2 = \tanh \left(W_1 \cdot \frac{1}{3} (h_3^1 + h_5^1 + h_8^1) + B_1 h_1^1 \right)$

A Simple Graph Convolution Layer

- averaging neighbor's message and apply nonlinear transformation



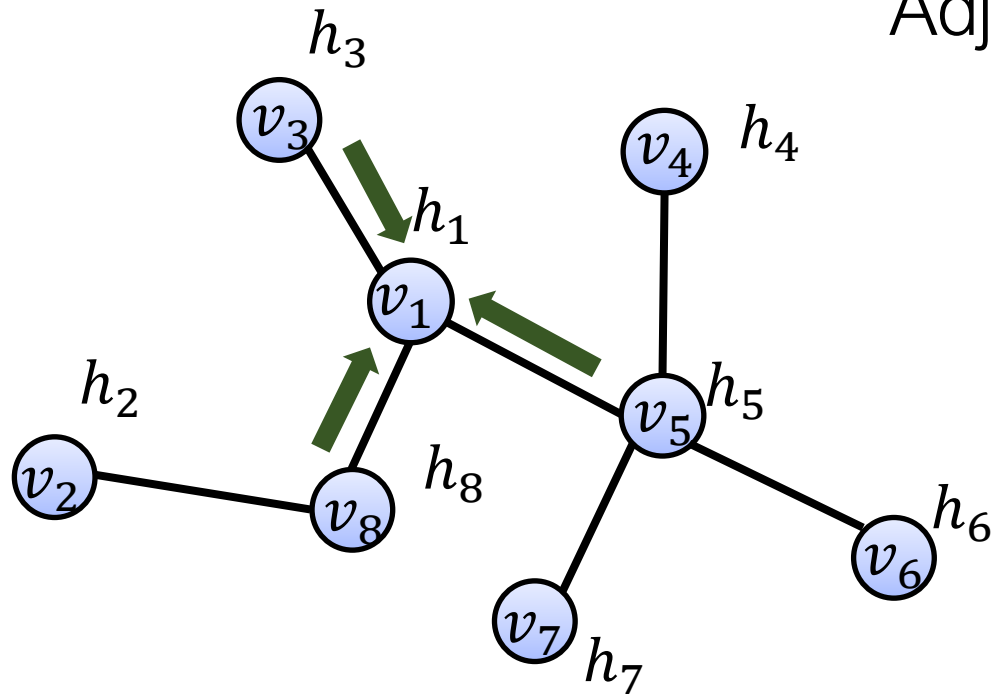
initial
embedding: $h_i^0 = x_i$

computing
next layer: $h_i^{k+1} = \sigma \left(W_k \frac{1}{|N(v_i)|} \sum_{v_j \in N(v_i)} h_j^k + B_k h_i^k \right)$

next layer: $h_1^{(3)} = \tanh \left(W_2 \cdot \frac{1}{3} (h_3^{(2)} + h_5^{(2)} + h_8^{(2)}) + B_2 h_1^{(2)} \right)$

Matrix Representations of Graphs

Adjacency Matrix: $A[i, j] = 1$ if v_i is adjacent to v_j
 $A[i, j] = 0$, otherwise



Adjacency Matrix **A**

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

Matrix Representation of GCN

- Neighbor Aggregation can be performed efficiently using matrix operations

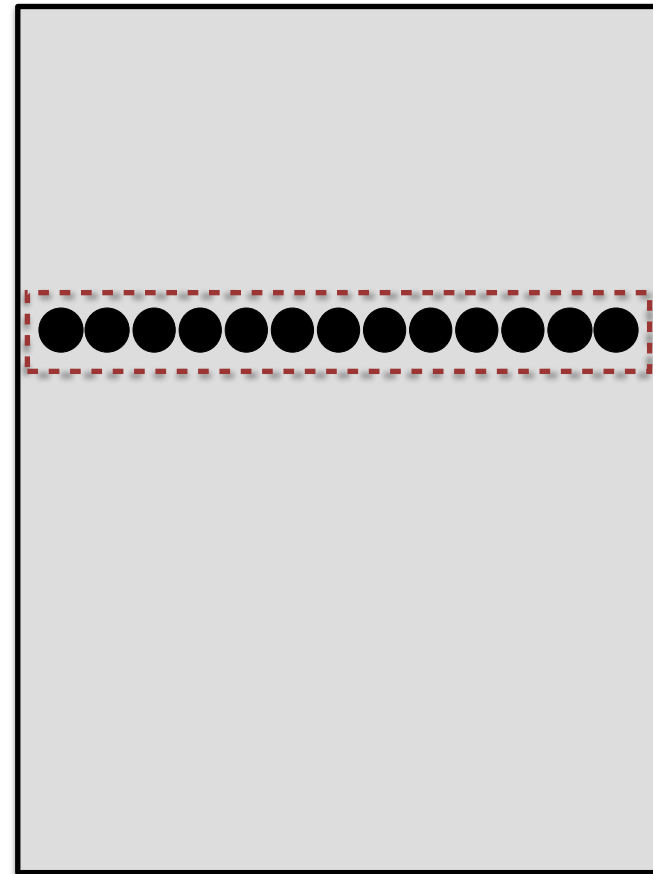
$$H^k = [h_1^k, \dots, h_{|V|}^k]^T$$

$$\text{Then } \sum_{v_j \in N(v_i)} h_j^k = A_{i,:} H^k$$

Let D be diagonal matrix (0 elsewhere)

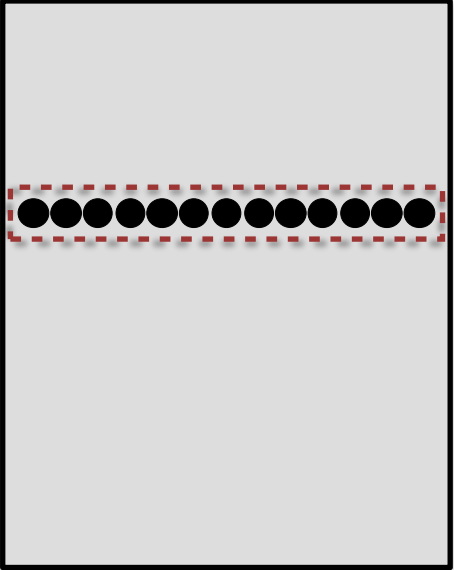
$$D_{i,i} = \text{Degree}(v_i) = \sum_j A_{i,j}$$

$$\text{Then } \frac{1}{|N(v_i)|} \sum_{v_j \in N(v_i)} h_j^k = D^{-1} A H^k$$



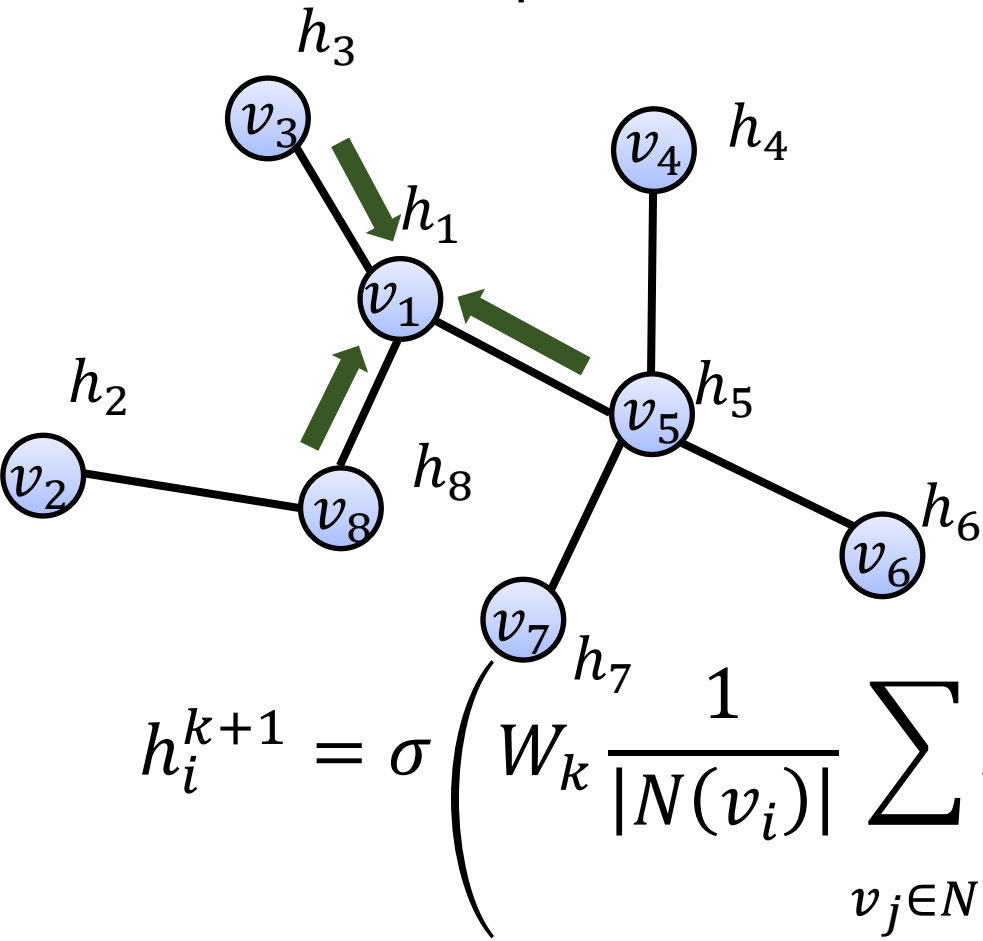
Aggregation Neighbor's Information in Matrix form

$$\frac{1}{|N(v_i)|} \sum_{v_j \in N(v_i)} h_j^k = D^{-1} A H^k$$

$$\begin{matrix}
 & D^{-1} & & A & & H^k \\
 \begin{pmatrix}
 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & \boxed{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
 \end{pmatrix}^{-1} & \cdot & \begin{pmatrix}
 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 1 & 0 & \boxed{1} & 0 & 0 & 1 & 0 & 0 \\
 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
 \end{pmatrix} & \cdot & \begin{matrix} \text{Matrix } H^k \end{matrix}
 \end{matrix}$$


Graph Convolution in Matrix Form

- Neighbor Aggregation can be performed efficiently using matrix operations



$$H^k = [h_1^k, \dots, h_{|V|}^k]^T$$

$$\tilde{A} = D^{-1}A$$

$$H^{k+1} = \sigma(\tilde{A}H^k \cdot W_k^T + H^k B_k^T)$$

$$h_i^{k+1} = \sigma \left(W_k \frac{1}{|N(v_i)|} \sum_{v_j \in N(v_i)} h_j^k + B_k h_i^k \right)$$

(Better) Graph Convolution Network

- To make $\tilde{A}$ symmetric

$$H^k = [h_1^k, \dots, h_{|V|}^k]^T$$

$$\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

$$H^{k+1} = \sigma(\tilde{A} H^k \cdot W_k^T + H^k B_k^T)$$

Prediction Layer

$$H^k = [h_1^k, \dots, h_{|V|}^k]^T$$
$$\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$
$$H^{k+1} = \sigma(\tilde{A} H^k \cdot W_k^T + H^k B_k^T)$$

- For node classification (e.g. predicting atom type):

$$o_i = \text{Softmax}(h_i^{(m)})$$

- For graph classification (e.g. predicting toxicity):

$$o = \text{Softmax}\left(\frac{1}{N} \sum_i h_i^{(m)}\right)$$

Property of GCN: Equivariant

- the embeddings computed from graph convolution layers is invariant to node order permutation

$$h_i^0 = x_i$$

$$h_i^{k+1} = \sigma\left(W_k \frac{1}{|N(v_i)|} \sum_{v_j \in N(v_i)} h_j^k + B_k h_i^k\right)$$

$$H^k = [h_1^k, \dots, h_{|V|}^k]^T$$

$$\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

$$H^{k+1} = \sigma(\tilde{A} H^k \cdot W_k^T + H^k B_k^T)$$

Training GCN

- Parameters: weight matrix for each layer

$$H^k = [h_1^k, \dots, h_{|V|}^k]^T$$

$$\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

$$H^{k+1} = \sigma(\tilde{A} H^k \cdot W_k^T + H^k B_k^T)$$

- Supervised training: e.g. Node classification

- Cross entropy loss

$$L = - \sum y_i \cdot \log f(h_i^K) \quad f_i = \text{Softmax}(h_i^{(K)})$$

- y_i is node label (in one-hot vector or smoothed version)

Training GCN

- Parameters: weight matrix for each layer

$$H^k = [h_1^k, \dots, h_{|V|}^k]^T$$

$$\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

$$H^{k+1} = \sigma(\tilde{A} H^k \cdot W_k^T + H^k B_k^T)$$

- Unsupervised training:

- Linked nodes have similar embedding

$$L = \sum_{i,j} CE(y_{i,j}, Sim(h_i^K, h_j^K))$$

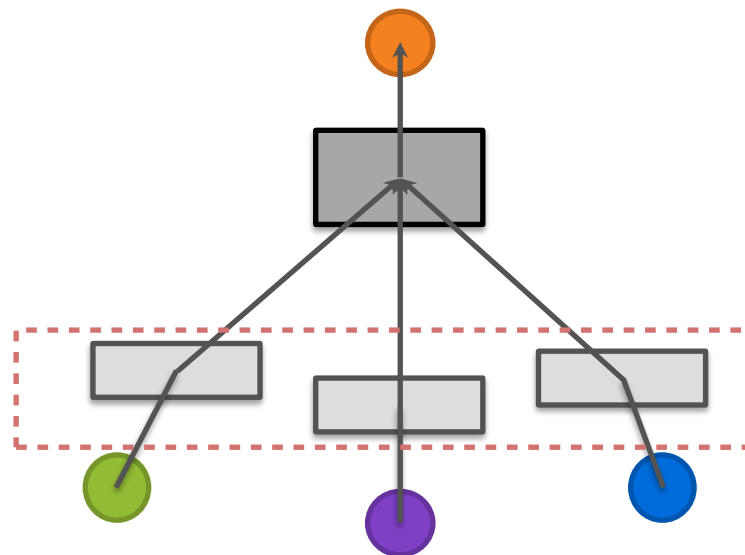
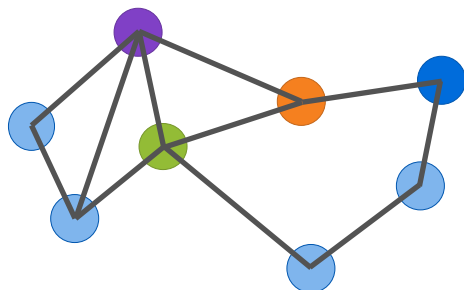
- $y_{i,j} = 1$ if there is edge from v_i to v_j
- Similarity can be defined in many ways: e.g. inner product $h_i \cdot h_j$

GraphSAGE

Also a special case of MPNN

$$h_i^{k+1} = \sigma \left(W_k \cdot \text{CONCAT}(h_i^k, \text{AGG}(\{h_j^k, \forall v_j \in N(v_i)\})) \right)$$

AGG can be designed in multiple ways, like pooling (sum, avg, max)



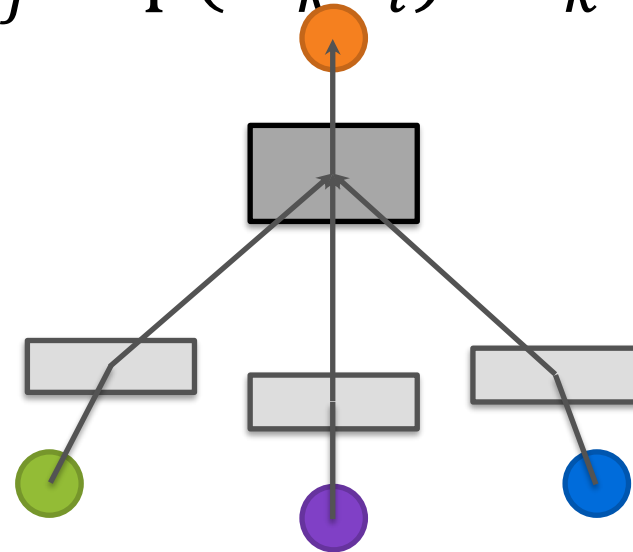
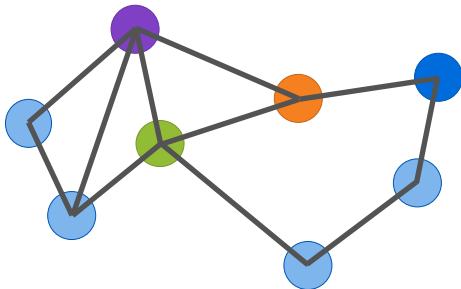
Graph Attention Network (GAT)

not strictly MPNN, but a generalized one

$$h_i^{k+1} = \sigma\left(\sum_{v_j \in N(v_i)} \alpha_{ij} W_k h_{v_j}^k\right)$$

attention weight

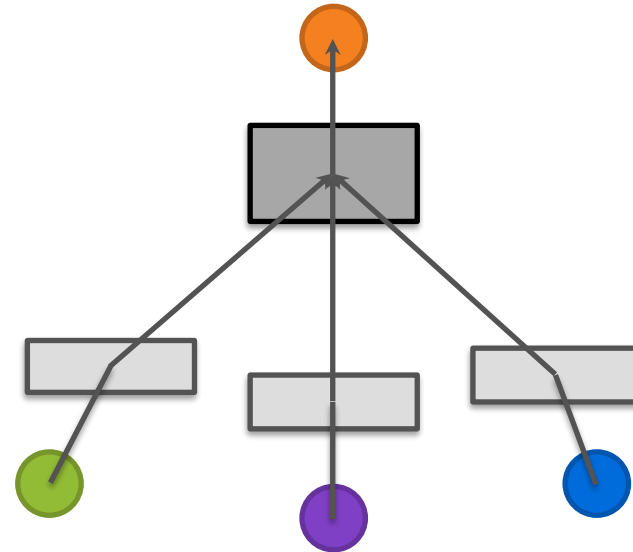
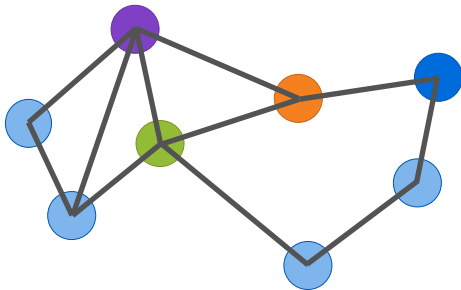
$$\alpha_{ij} = \text{Attention}(W_k h_i, W_k h_j) = \frac{\exp(W_k h_i)^T W_k h_j}{\sum_{j'} \exp(W_k h_i)^T W_k h_{j'}}$$



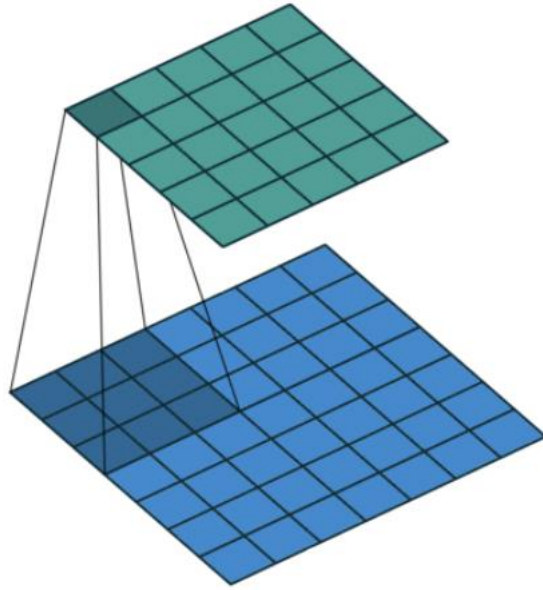
Multi-head Attention for GAT? Yes

$$h_i^{k+1} = \sigma\left(\sum_{v_j \in N(v_i)} \alpha_{ij} W_k h_{v_j}^k\right)$$

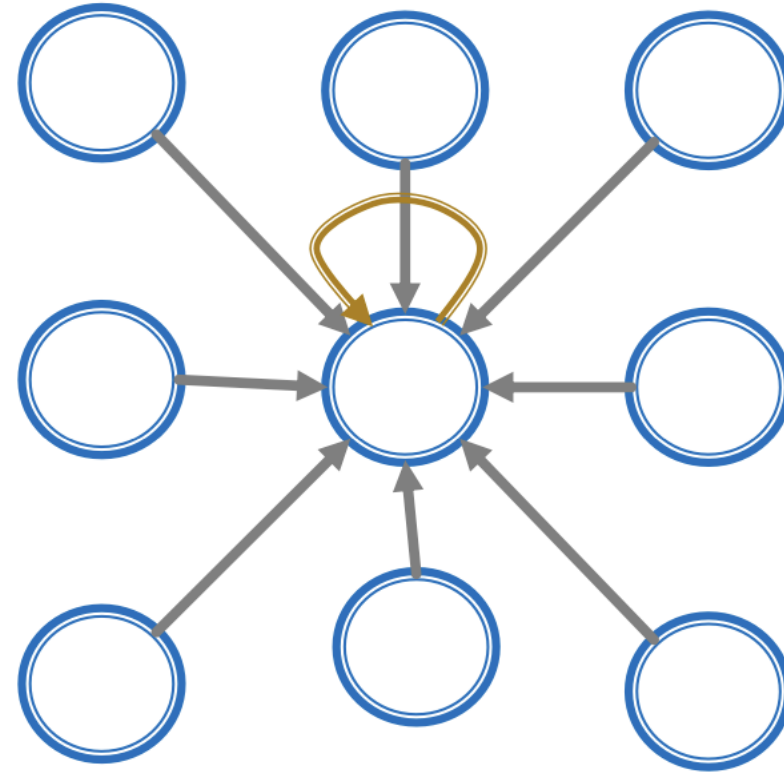
$$\alpha_{ij} = \text{Attention}(W_k h_i, W_k h_j) = \frac{\exp(W_k h_i)^T W_k h_j}{\sum_{j'} \exp(W_k h_i)^T W_k h_{j'}}$$



Relation between GNN and CNN



Image



Graph

CNN can be viewed as a special GNN on grid graph

GNN vs. Transformer

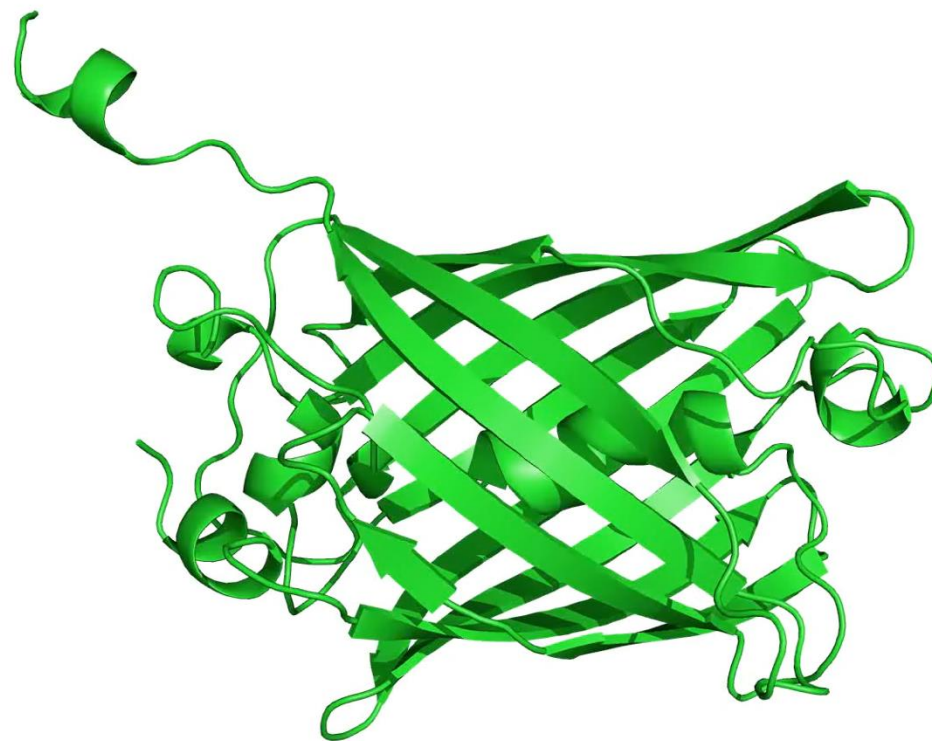
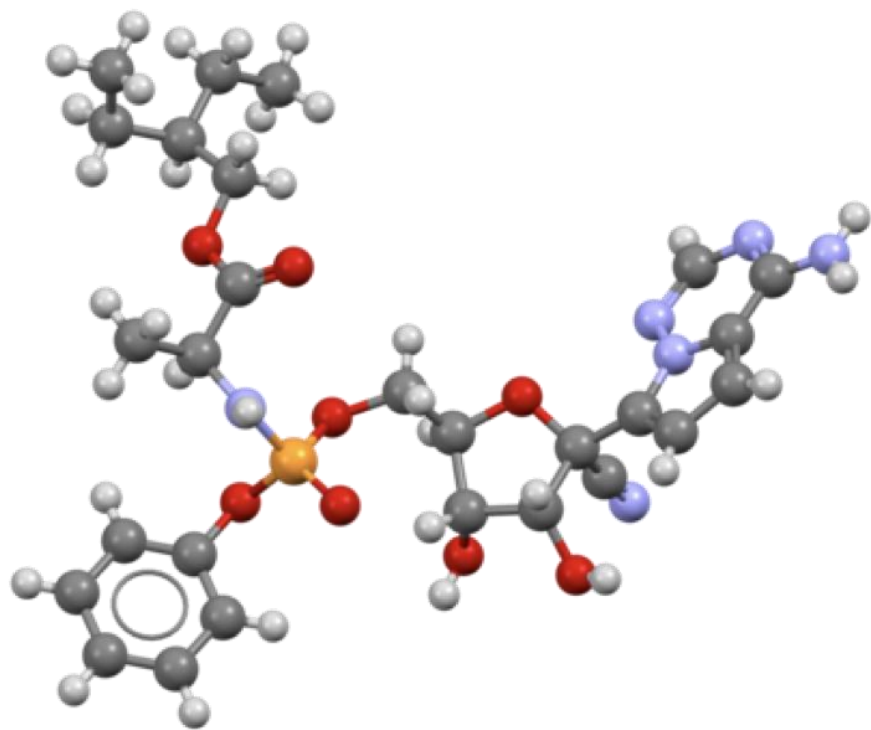
- Transformer is special GNN on a full-connected graph

Outline

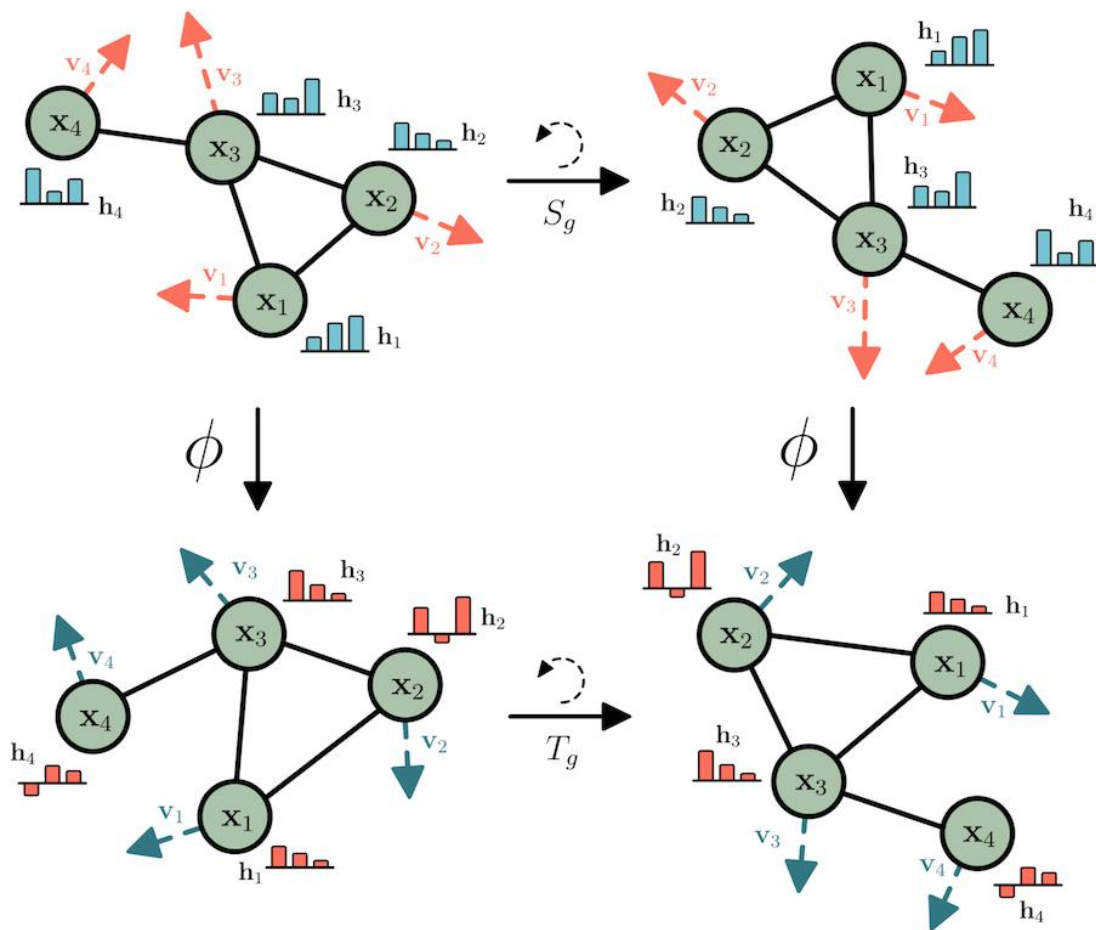
- Message Passing Neural Network
- ➔ • Equivariant Graph Neural Network
- Code Example

Structure and Geometry of Molecules

- Matrix of 3D coordinates, in addition to node features
- Equivariance: translation, rotation invariant



Equivariance in Embedding Transformation



f is Equivariant (SE3 invariant):

$$f(R(x) + z) = f(x)$$

No matter how we rotate and move the molecule, the NN will compute the same embeddings
 X are coordinates

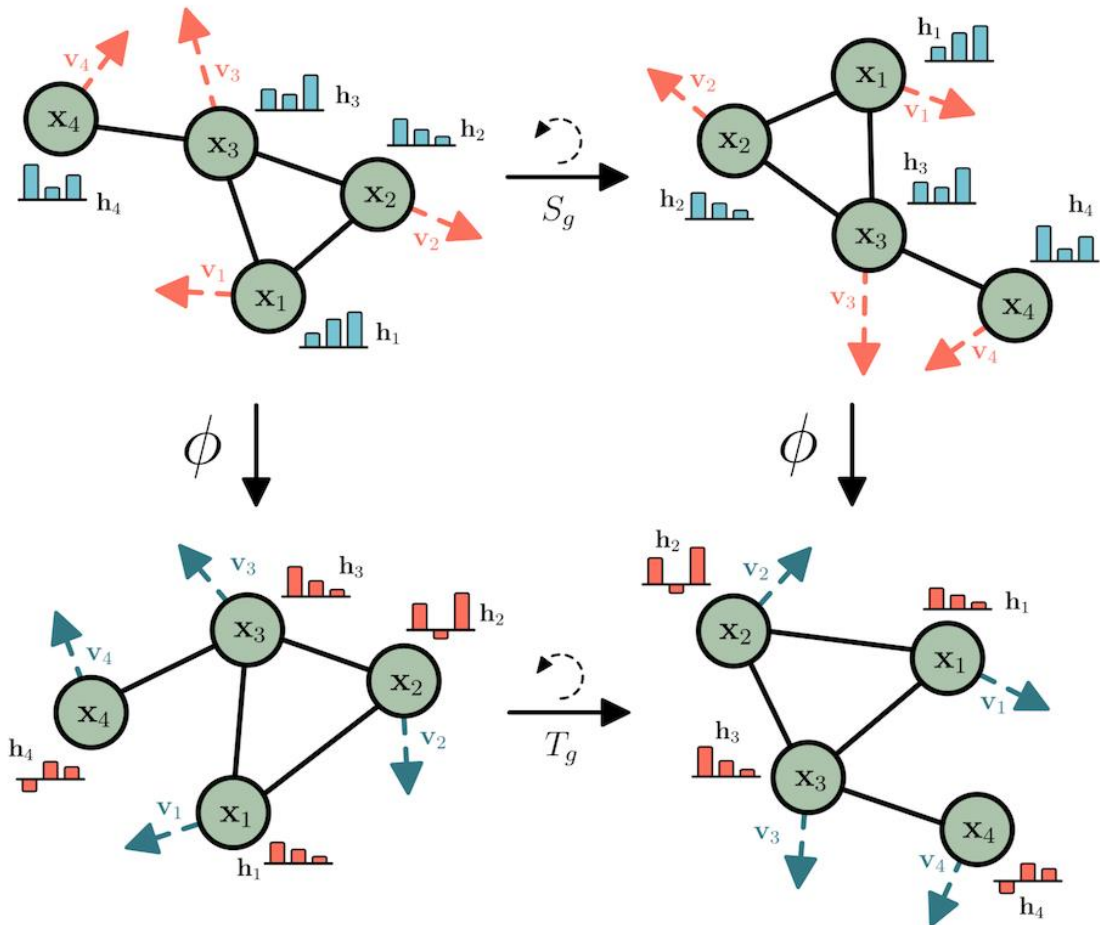
H are features

g is the rotation/translation

$$F(H, g(X)) = F(H, X)$$

Equivariant Graph Neural Network (EGNN)

- EGNN is an equivariant MPNN



$$d_{ij} = \|x_i - x_j\|_2$$

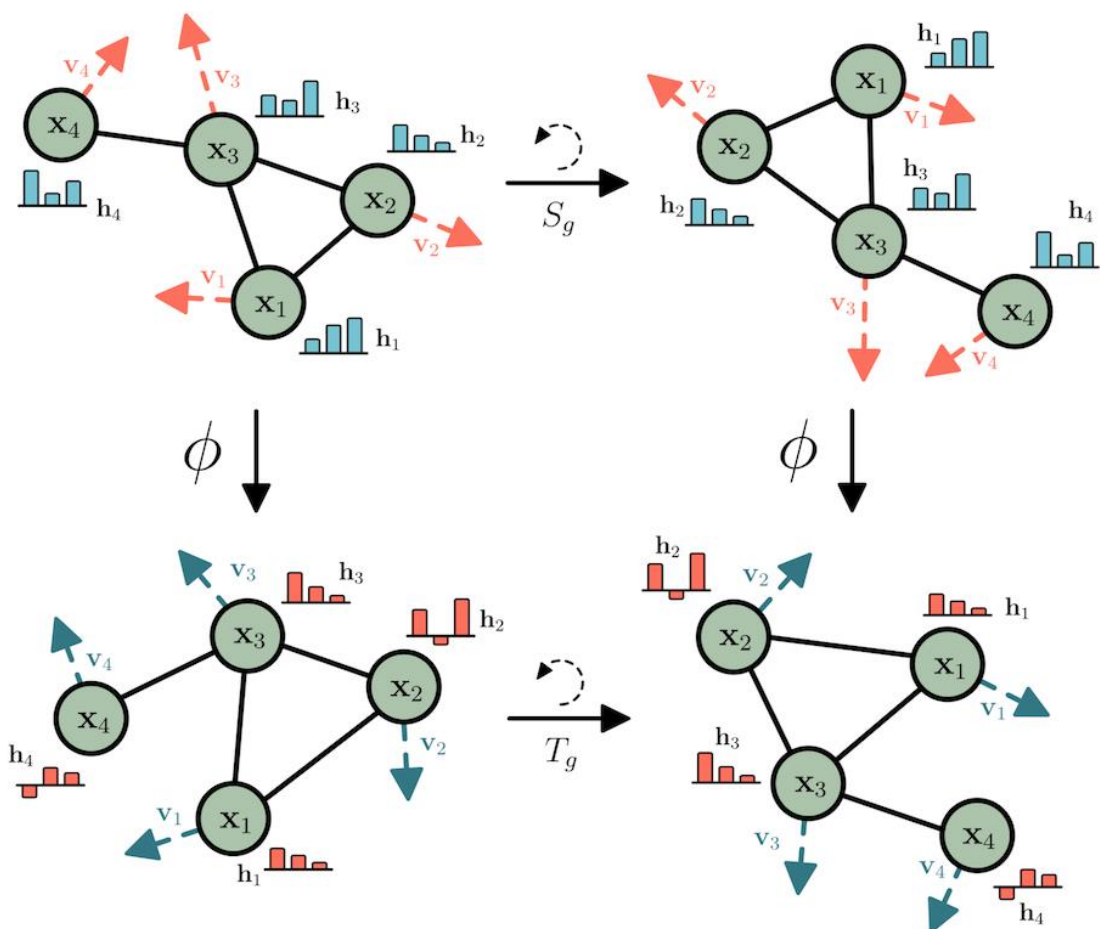
$$m_{j \rightarrow i} = M^k(h_i^k, h_j^k, d_{ij})$$

$$m_i^{k+1} = \sum_{v_j \in N(v_i)} m_{j \rightarrow i}$$

$$h_i^{k+1} = U^k(h_i^k, m_i^{k+1})$$

Equivariant Graph Neural Network (EGNN)

- EGNN is an equivariant MPNN, optionally updating coordinates



$$d_{ij}^k = \|x_i^k - x_j^k\|_2$$

$$m_{j \rightarrow i} = M^k(h_i^k, h_j^k, d_{ij}^k)$$

$$m_i^{k+1} = \sum_{v_j \in N(v_i)} m_{j \rightarrow i}$$

$$h_i^{k+1} = U^k(h_i^k, m_i^{k+1})$$

$$x_i^{k+1} = x_i^k + \sum_{v_j \in N(v_i)} (x_i^k - x_j^k) F^k(m_{j \rightarrow i})$$

Using MPNN and EGNN

- Code example at

https://projects.volkamerlab.org/teachopencadd/talktorials/T036_e3_equivariant_gnn.html

Summary

- Message Passing Neural Network
 - Nodes carry embeddings
 - message (embeddings) passed along graph edges
 - message + aggregation + update
 - many variants: GCN, GAT
- Equivariant Graph Neural Network
 - A special case of MPNN
 - modelling the embedding of nodes, along with 3D structures
 - rotation/translation invariant

Presentation Assignment

- Canvas

Quiz