165B
Machine Learning
Graph Neural Network

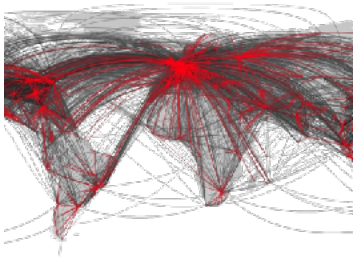
Lei Li (leili@cs)
UCSB

Recap

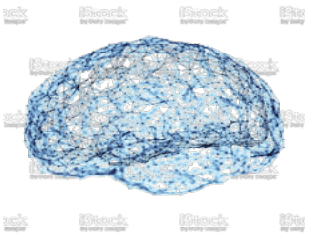
- Key components in Transformer
  - Positional Embedding (to distinguish tokens at different pos)
  - Multihead attention
  - Residual connection
  - layer norm
- Transformer is effective for machine translation, and many other tasks
- Pre-training: using raw unlabeled data for training, also known as self-supervised learning

Graph Data is everywhere

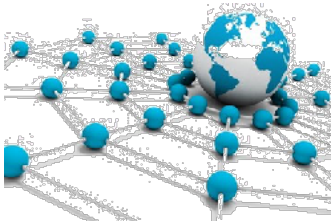
Social Graphs



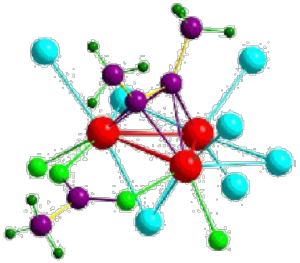
Transportation Graphs



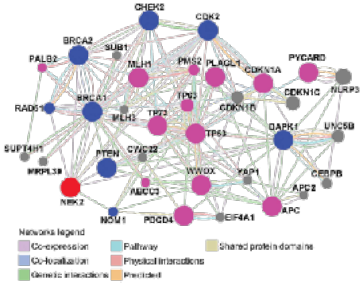
Brain Graphs



Web Graphs



Molecular Graphs



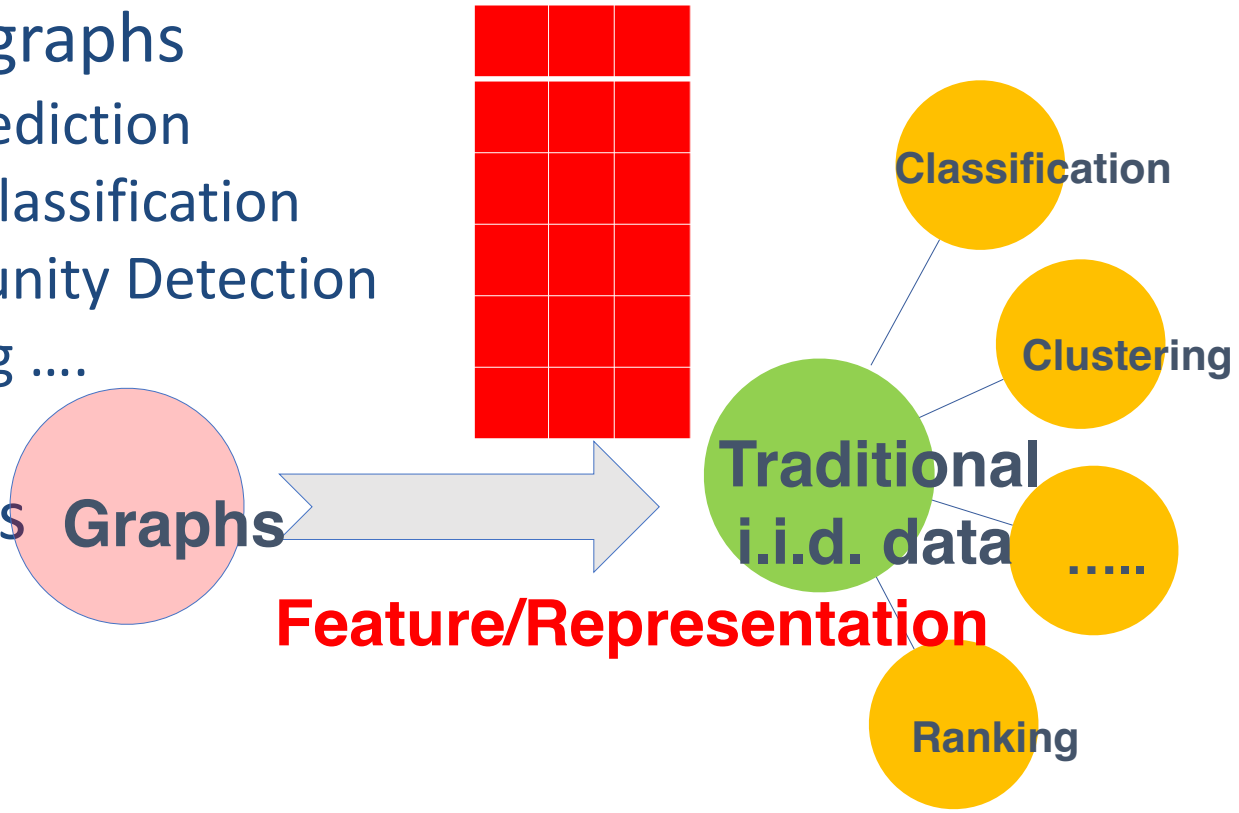
Gene Graphs

ML on Graphs

Numerous real-world problems can be summarized as a set of tasks on graphs

- Link prediction
- Node Classification
- Community Detection
- Ranking ….

ML solutions

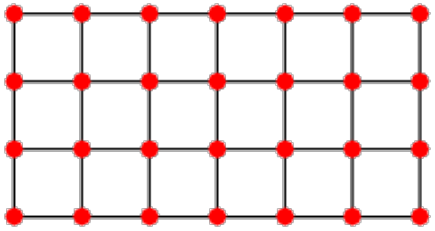


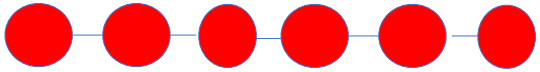
Slides adapted from Yao Ma & Jiliang Tang@MSU

Deep Learning Meets Graphs: Challenges

Traditional DL is designed for simple grids or sequences

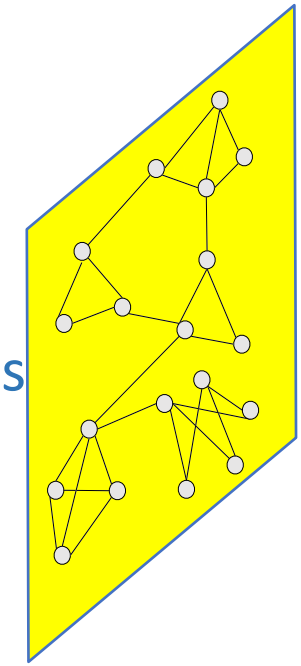
- CNNs for fixed-size images/grids
- RNNs for text/sequences





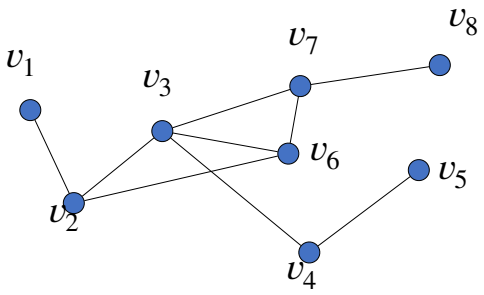
But nodes on graphs have different connections

- Arbitrary neighbor size
- Complex topological structure
- No fixed node ordering



Slides adapted from Yao Ma & Jiliang Tang@MSU

Graph Representation



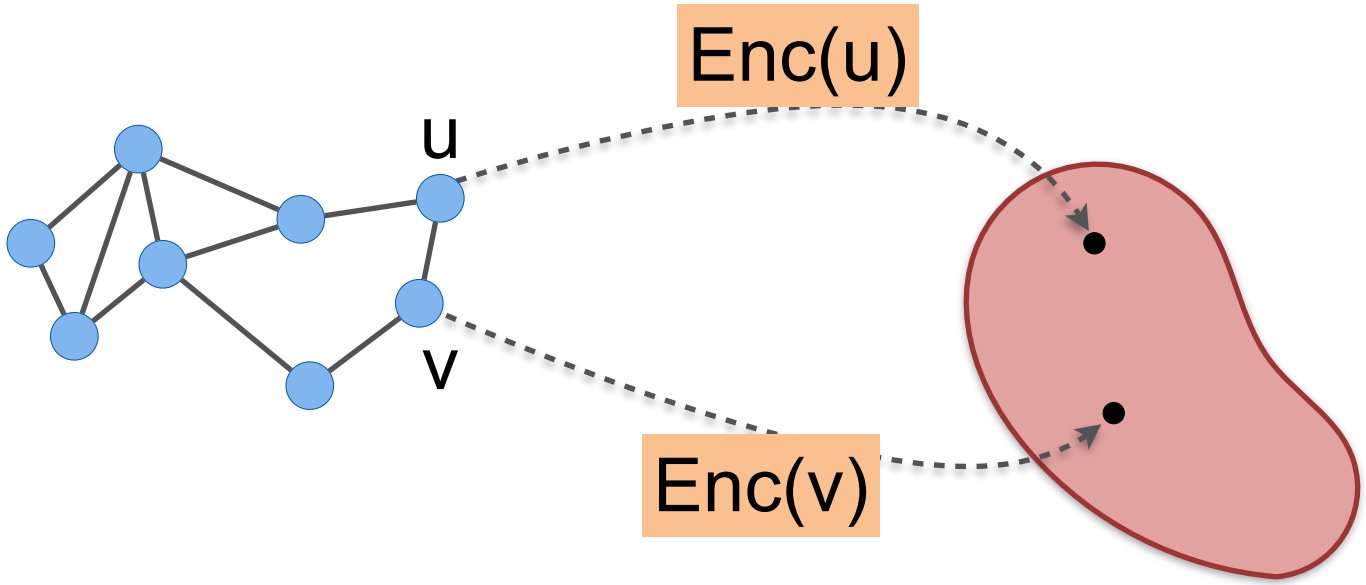
$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

$$\mathcal{E} = \{e_1, \ldots, e_M\}$$

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$

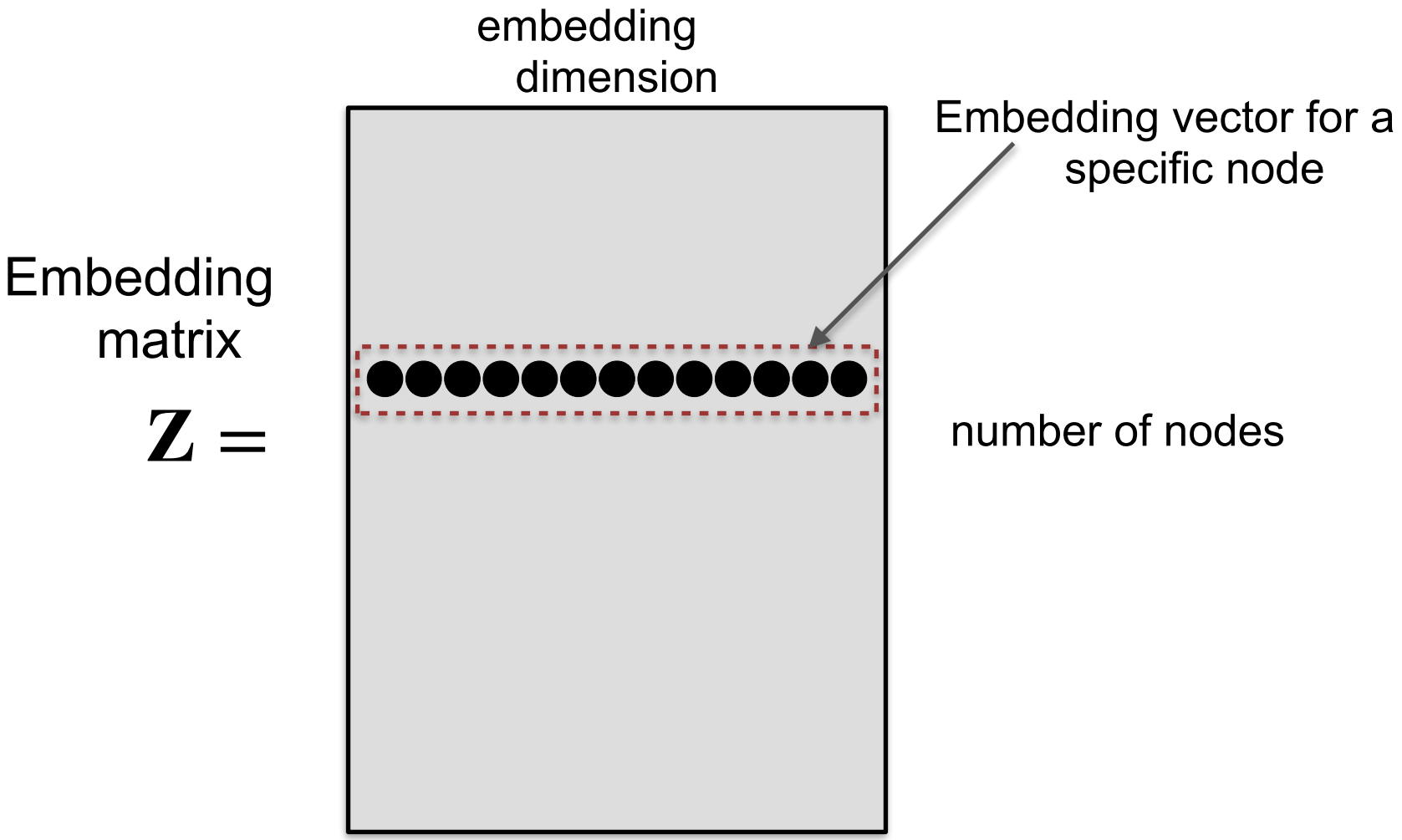
Node Embedding

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

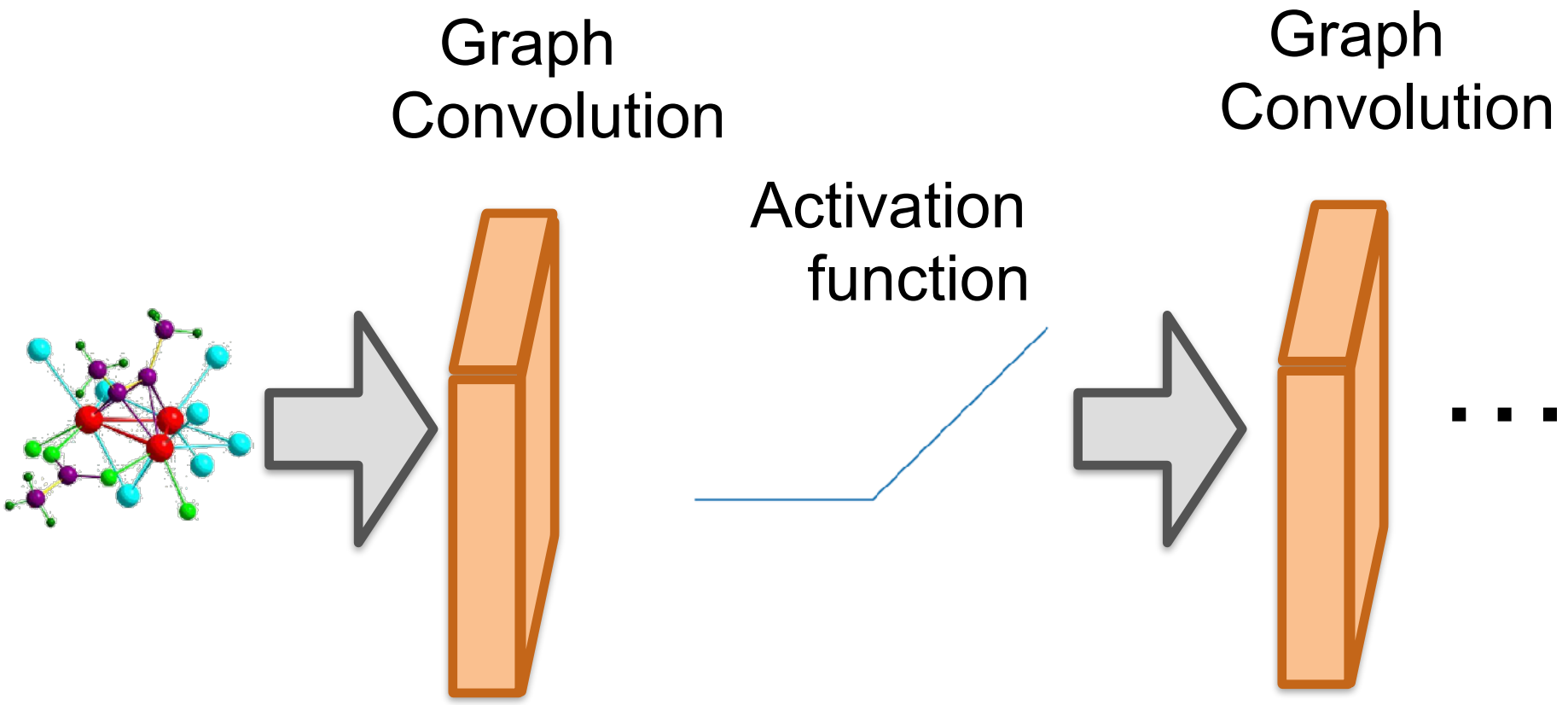


“Shallow” Node Embedding

- is just a lookup-table



Deep Graph Neural Network

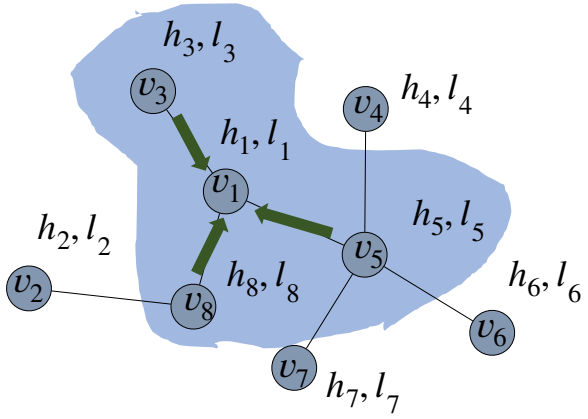


Output is embedding matrix for nodes

for further downstream tasks: e.g. node classification

From 2D-Convolution to Graph Convolution

Every node's neighbor defines a convolutional kernel

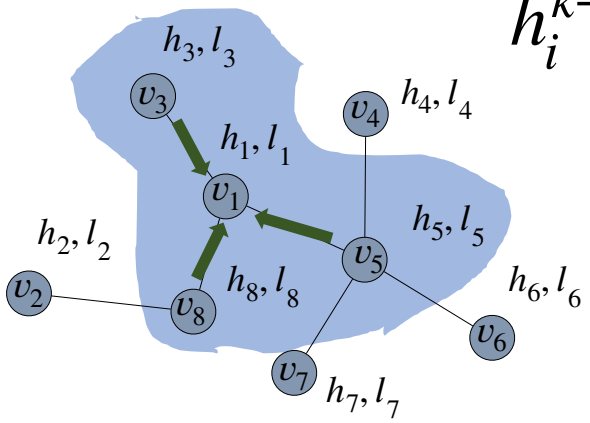


aggregate information from its neighbors

Aggregate Neighbors

h_i: node (hidden) embedding vector

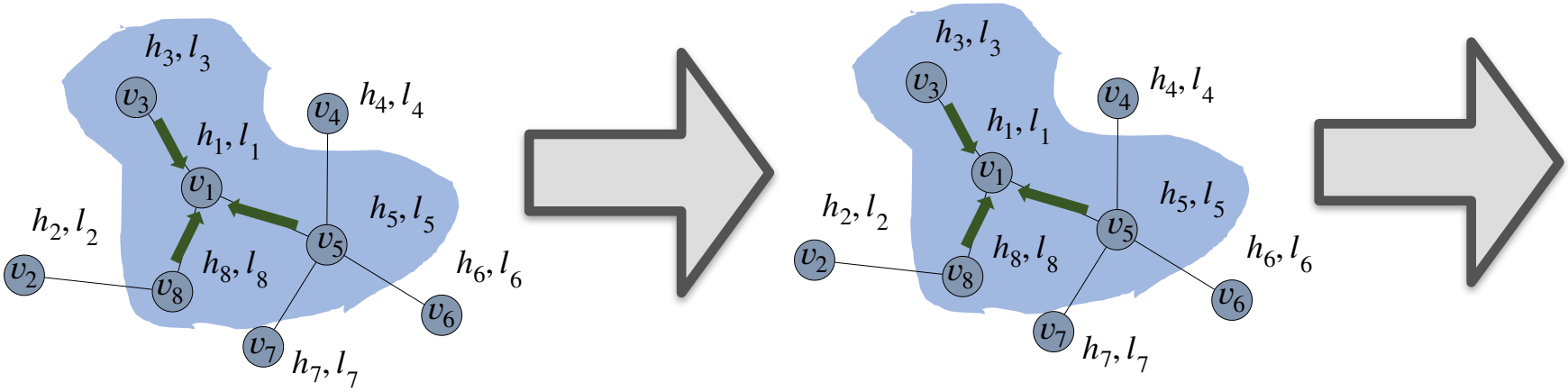
$$h_i^{k+1} = \text{Aggregate}_{v_j \in N(v_i)} f(h_i^k, h_j^k), \forall v_i \in V$$



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

$f(\cdot)$: Feedforward network.

Multiple Computation Layers



A Simple Graph Convolution Layer

- Simple approach: averaging neighbor's message and apply nonlinear transformation

$$h_i^0 = x_i$$

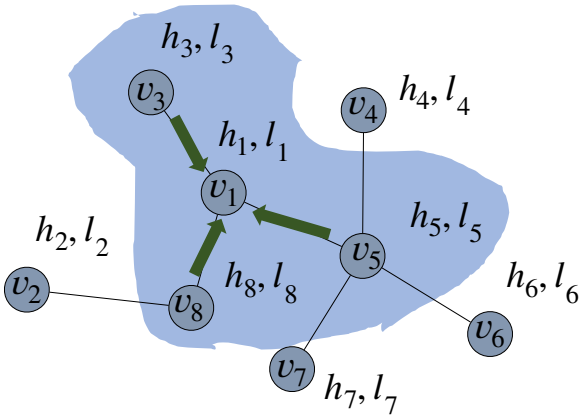
$$h_i^{k+1} = \sigma(W_k \frac{1}{|N(v_i)|} \sum_{v_j \in N(v_i)} h_j^k + B_k h_i^k)$$

Property: Equivariant

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

$$h_i^0 = x_i$$

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



Model Training

- Parameters: weight matrix for each layer

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

- 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

Model Training

- Parameters: weight matrix for each layer

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

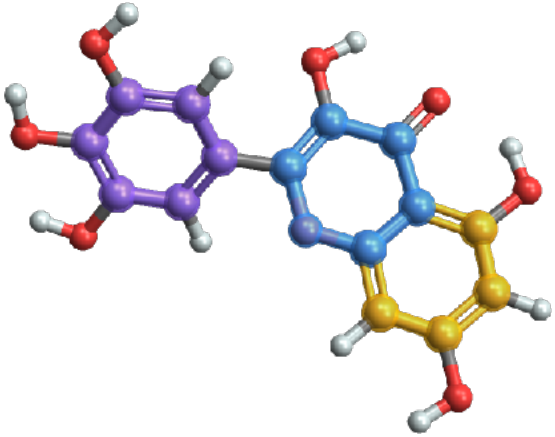
- Supervised training: e.g. Node classification
  - Linked nodes have similar embedding

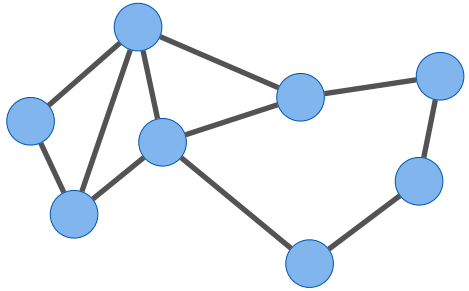
$$L = \sum_i CE(y_i, f(h_i^K))$$

  - y_i is node label

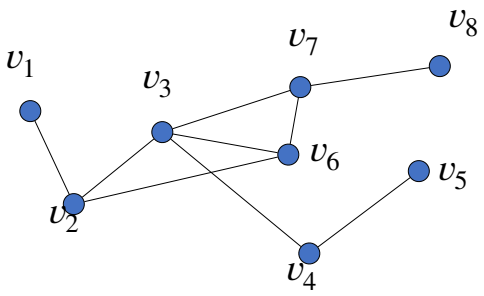
Example: predict toxicity of a drug

Toxic?





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

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

$\boldsymbol{A}$

Spectral graph theory. American Mathematical Soc.; 1997.

Matrix Representation of GCN

- Neighbor Aggregation can be performed efficiently using matrix operations

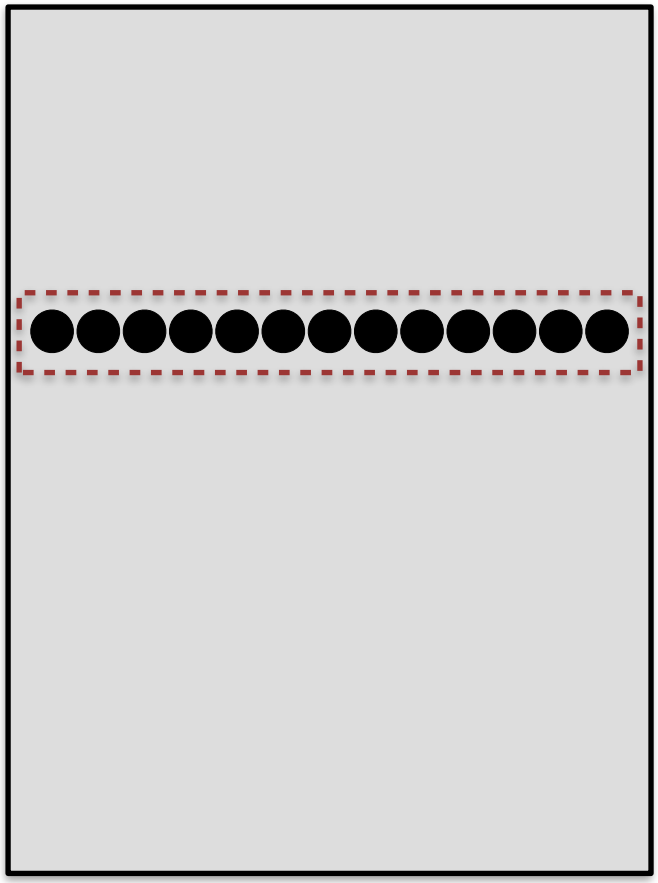
$$H^k = [h_1^k, \ldots, h_{|V|}^k]^T$$

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

Let D be diagonal matrix

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

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



Matrix Representation of GCN

- Neighbor Aggregation can be performed efficiently using matrix operations

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

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

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

Graph Convolution Network

- Neighbor Aggregation can be performed efficiently using matrix operations
- To make $\tilde{A}$ symmetric

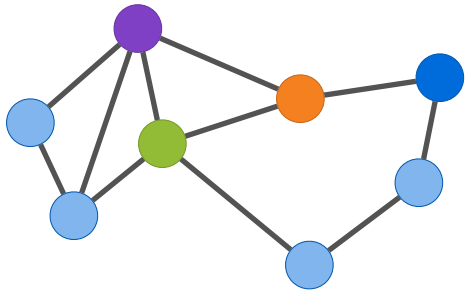
$$H^k = [h_1^k, \ldots, h_{|V|}^k]^T$$

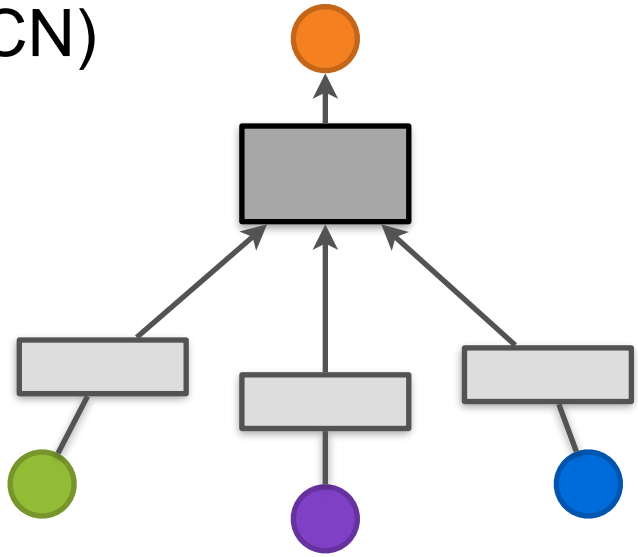
$$\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

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

Generic GNN framework

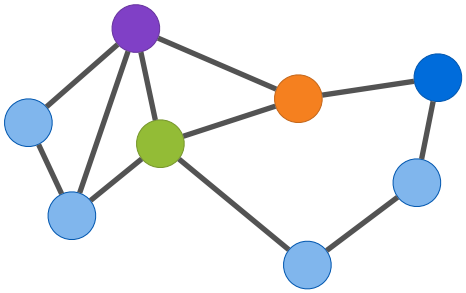
- GNN layer = message passing + Aggregation
  - different design choices under this framework
  - Graph convolutional network (GCN)
  - GraphSAGE
  - GAT

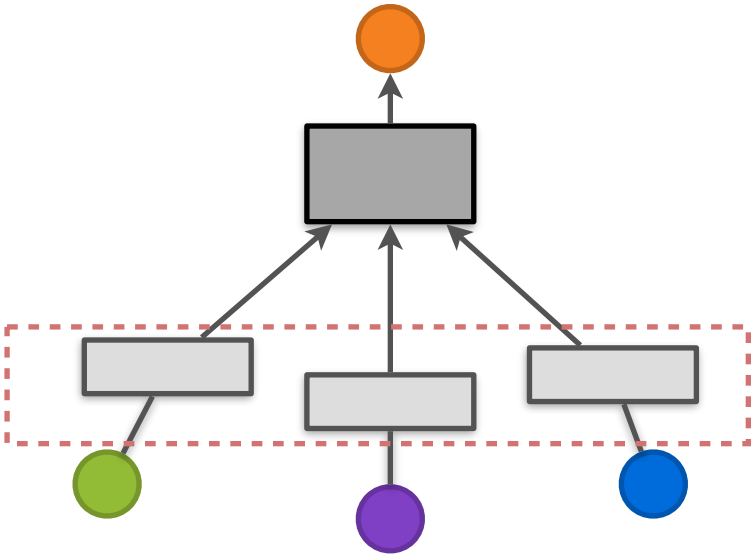




Message Computation

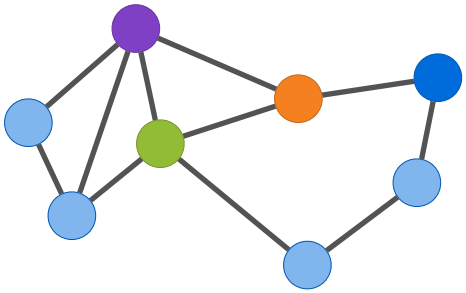
- Each node will create a message
- e.g.
  - Linear projection

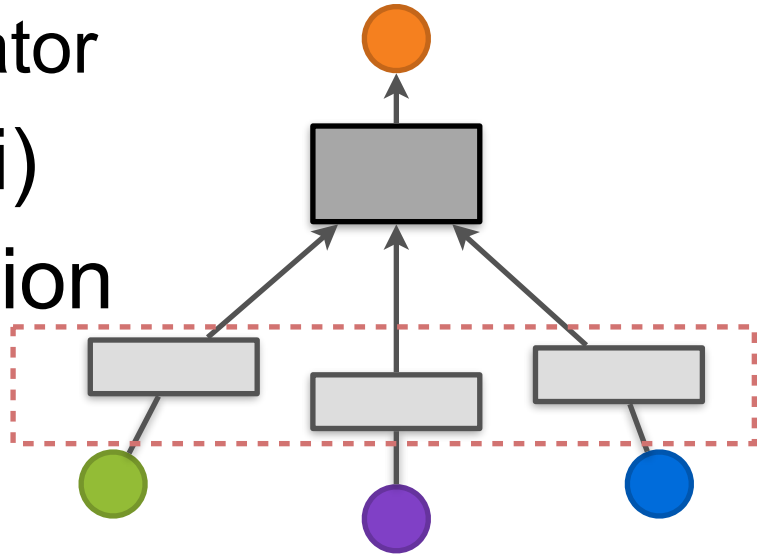




Aggregation/Pooling

- Each node will aggregate messages from its neighbors
- e.g.
  - Sum, Mean, Max operator
- Concat(AGG{m_j}, m_i)
- Apply nonlinear activation





GraphSAGE

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

AGG can be designed in multiple ways

GraphSAGE

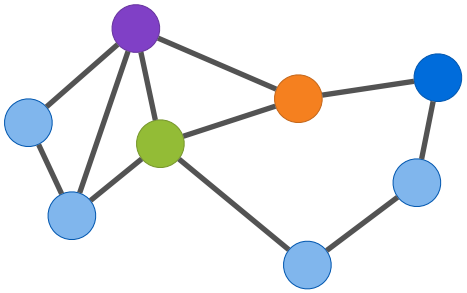
$$h_i^{k+1} = \sigma\left(W_k \cdot \text{CONCAT}\left(h_i^k, \text{AGG}(\{h_j^k, \forall v_j \in N(v_i)\}) \right) \right)$$

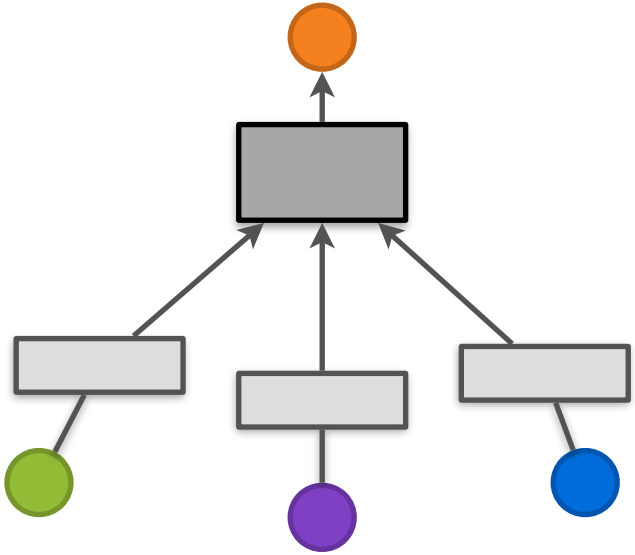
AGG can be designed in multiple ways

Graph Attention Network (GAT)

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

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

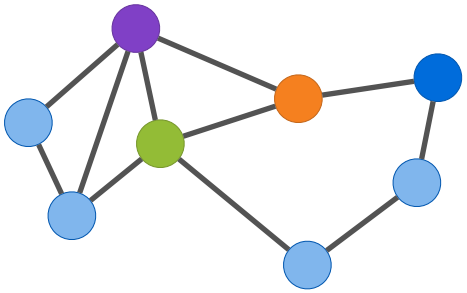


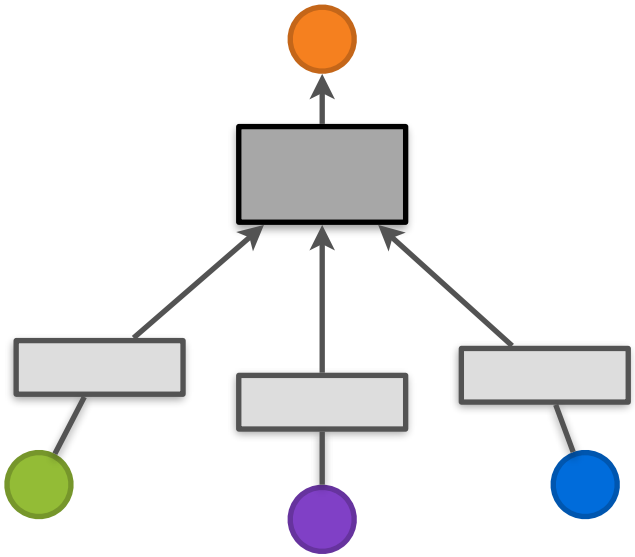


Multi-head Attention for GAT? Yes

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

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

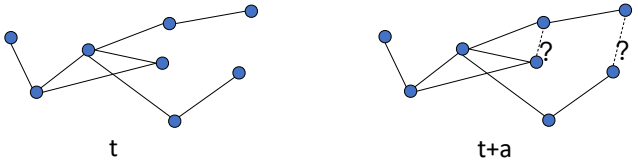




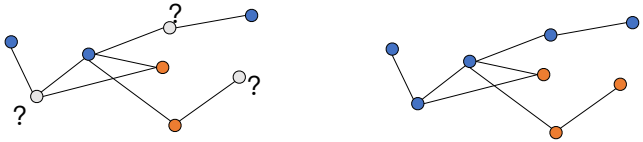
Tasks on Graph-Structured Data

Node-level

Link Prediction

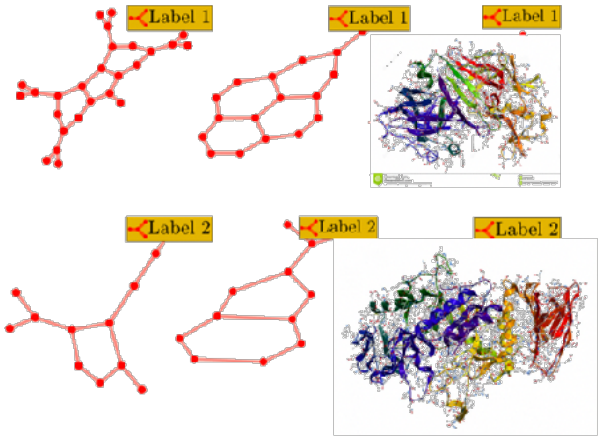


Node Classification

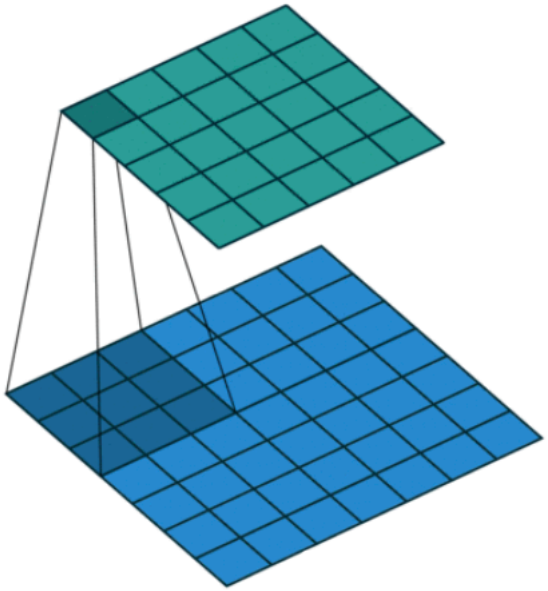


Graph-level

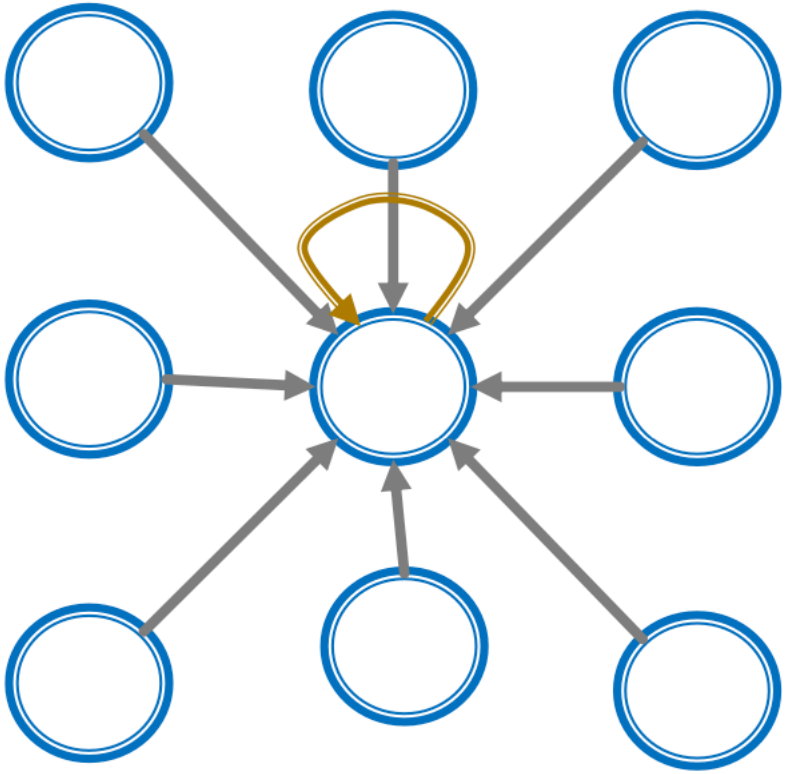
Graph Classification



Relation between GNN and CNN



Image



Graph

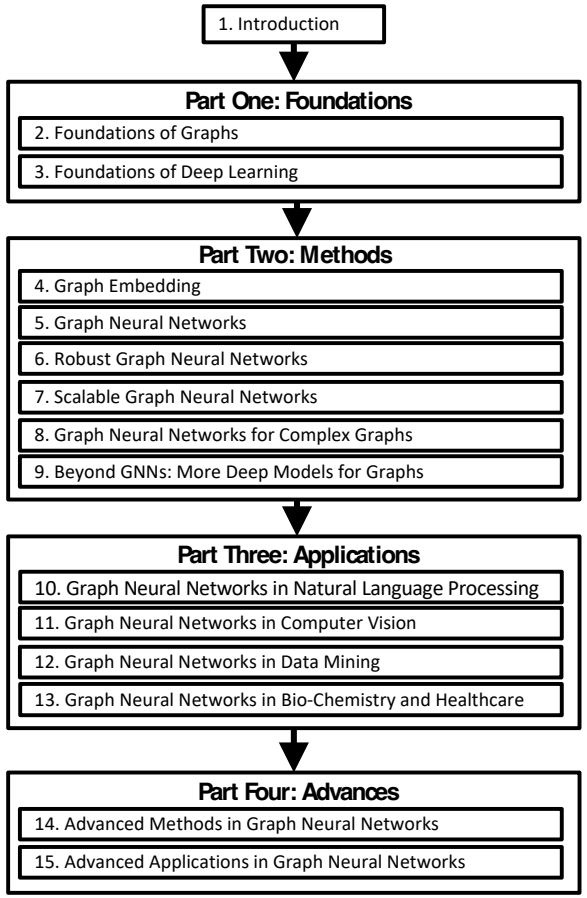
CNN can be viewed as a special GNN on grid graph

GNN vs. Transformer

- Transformer is special GNN on a full-connected graph

Book: Deep Learning on Graphs

https://cse.msu.edu/~mayao4/dlg_book/



Summary

- Graph neural network
  - message passed along graph edges
  - aggregate message/embedding by FFN
  - many variants

Next Up

- Variational Auto-Encoder