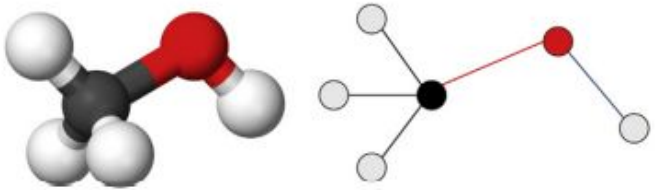
# Graph Neural NETWORK (GNN)

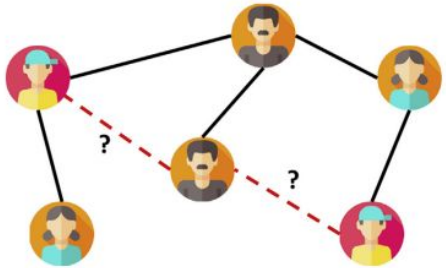
Zahra Akhlaghi

# Graph Structured Data

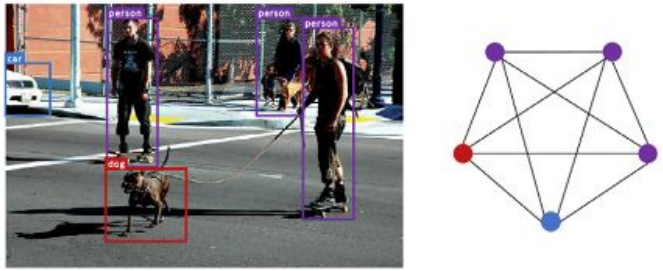
a) Molecule

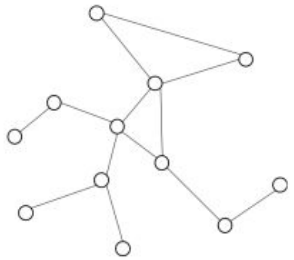


b) Social Network

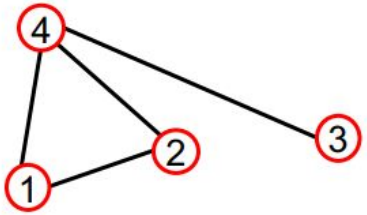


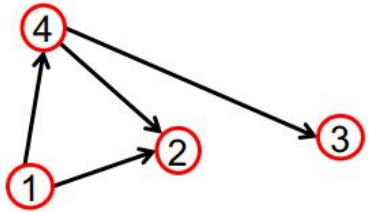

b) Image



# Representation Graph ( Adjacency Matrix )



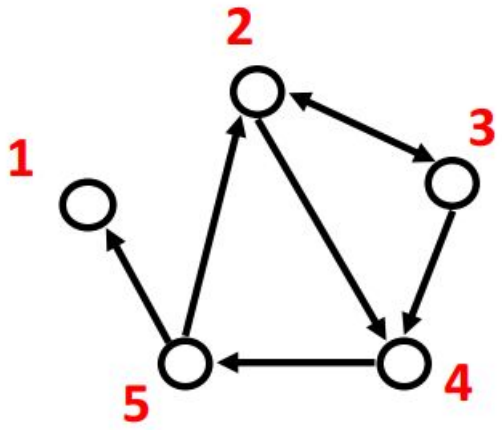

$A_{ij} = 1$ if there is a link from node $i$ to node $j$

$A_{ij} = 0$ otherwise

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

# Representation Graph ( Edge List )

- (2, 3)
- (2, 4)
- (3, 2)
- (3, 4)
- (4, 5)
- (5, 2)
- (5, 1)

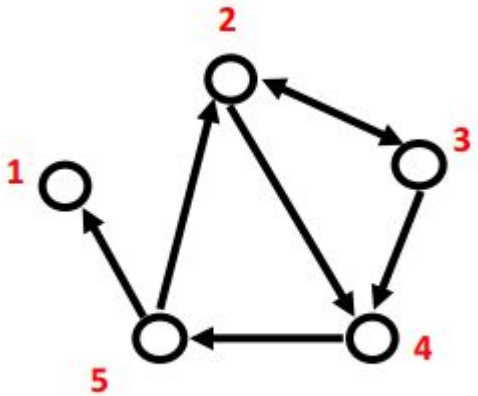


# Representation Graph ( Adjacency List )

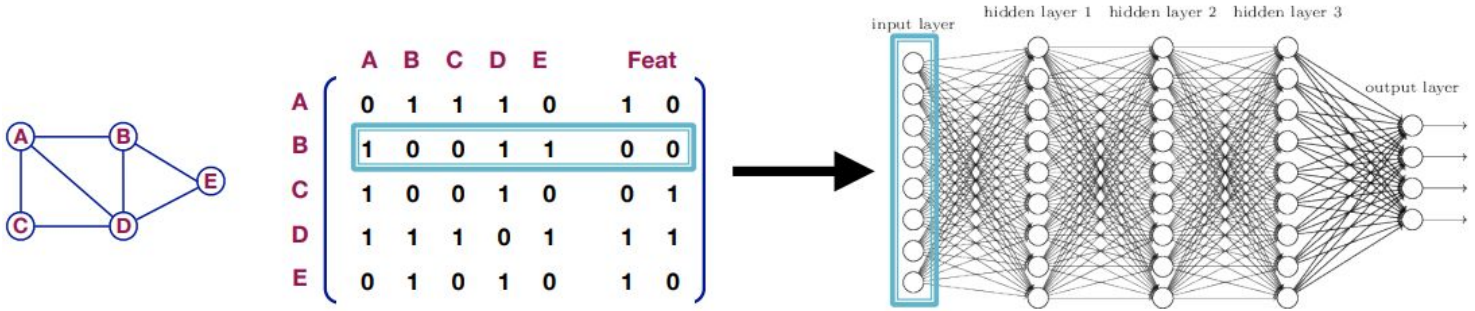
Easier to work with if network is:

- Large
- Sparse

- 1:
- 2: 3, 4
- 3: 2, 4
- 4: 5
- 5: 1, 2



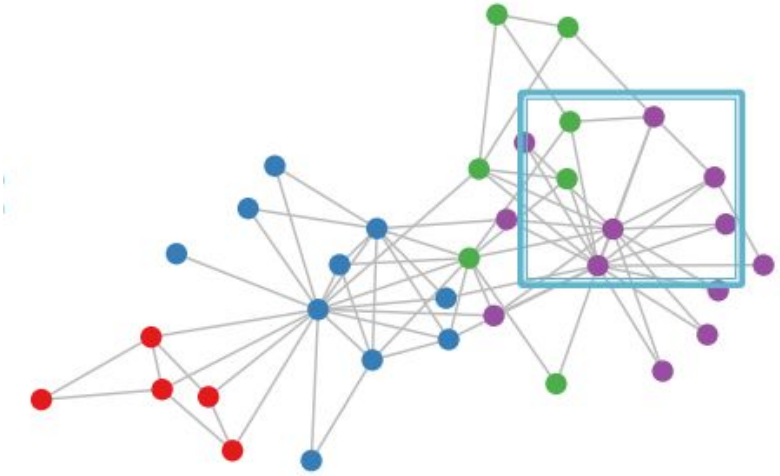
# A Naive Approach



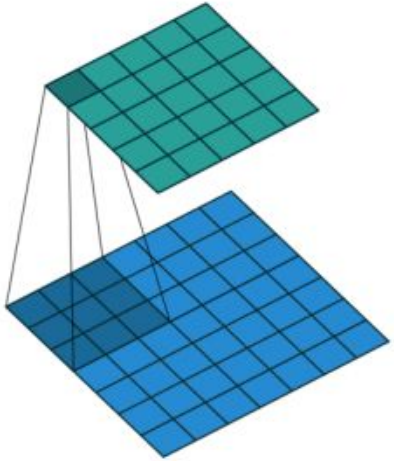

- O(|V|) parameters
- Not applicable to graphs of different sizes
- Sensitive to node ordering

# Real-World Graphs

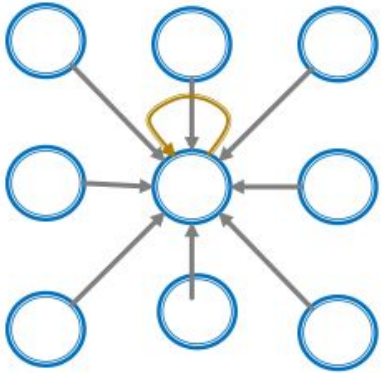
- There is no fixed notion of locality or sliding window on the graph
- Graph is permutation invariant



# From Images to Graph

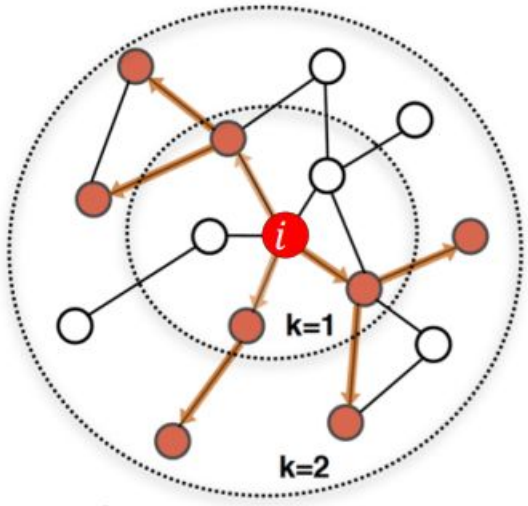


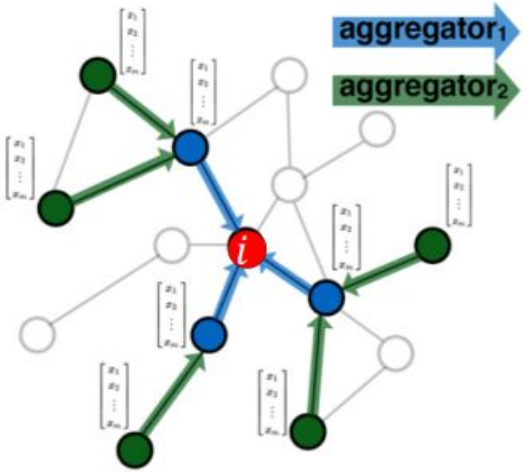
Images



Graph

# Graph Neural Network

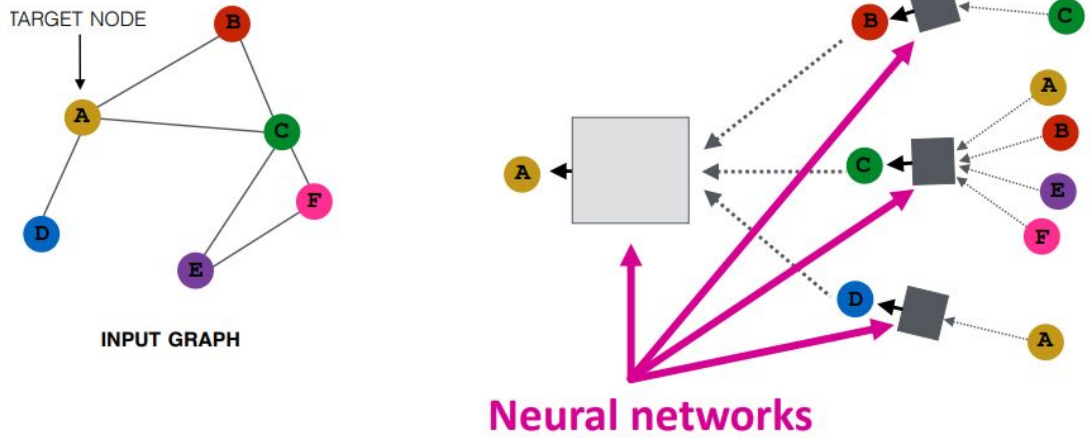



Idea: Node's neighborhood defines a computation graph

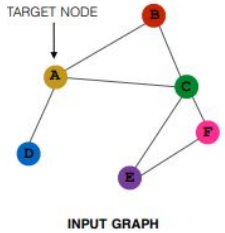
# Idea: Aggregate Neighbors

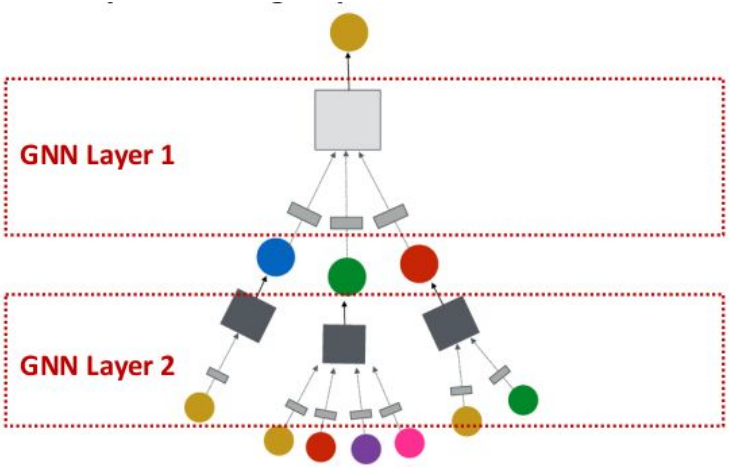
**Intuition:** Nodes aggregate information from their neighbors using neural networks



# A General GNN Framework

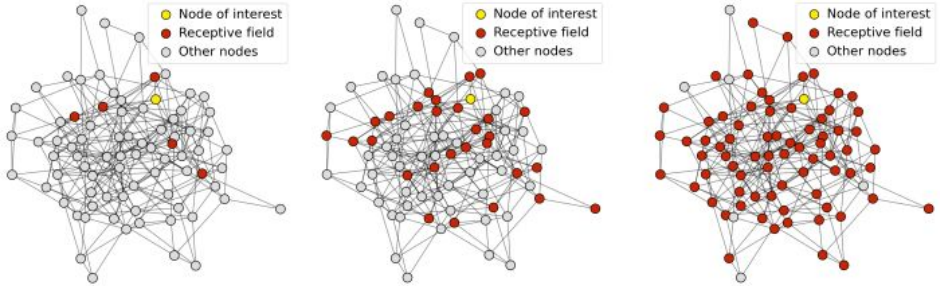
GNN Layer = Message + Aggregation



# The Over-Smoothing Problem

- The Issue of stacking many GNN layers
  - GNN suffers from the over-smoothing problem
- The over-smoothing problem: all the node embeddings converge to the same value
  - This is bad because we want to use node embeddings to differentiate nodes



# Why Manipulate Graph

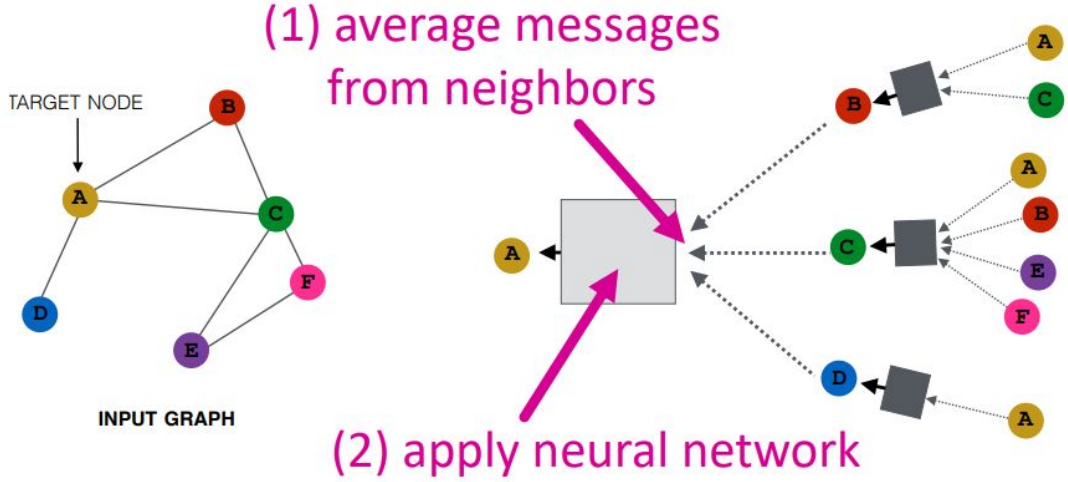
**Raw input graph = computational graph**

- **Feature level:**
  - The input graph lacks features → feature augmentation
- **Structure level:**
  - The graph is too sparse → inefficient message passing
  - The graph is too dense → message passing is too costly
  - The graph is too large → cannot fit the computational graph into a GPU

# Graph Manipulation Approaches

- Graph Feature manipulation
  - The input graph lacks features → feature augmentation
- Graph Structure manipulation
  - The graph is too sparse → Add virtual nodes / edges
  - The graph is too dense → Sample neighbors when doing message passing
  - The graph is too large → Sample subgraphs to compute embeddings

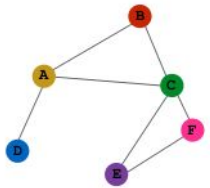
# Graph Convolutional Networks

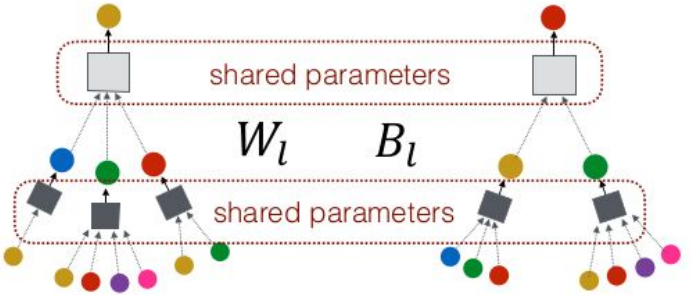



$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

# Inductive Capability

- The same aggregation parameters are shared for all nodes:
  - The number of model parameters is sublinear in |V| and we can generalize to unseen nodes!



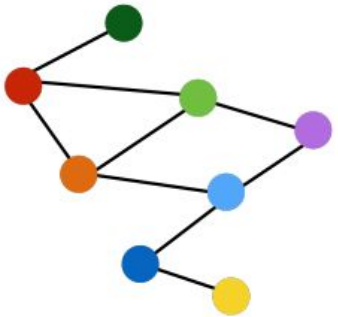




Compute graph for node A

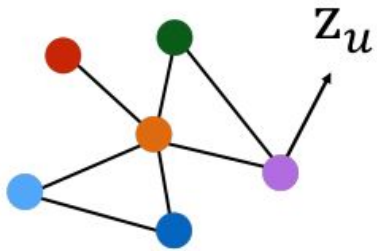
Compute graph for node B

# Inductive Capability : New Graphs

Inductive node embedding → Generalize to entirely unseen graphs

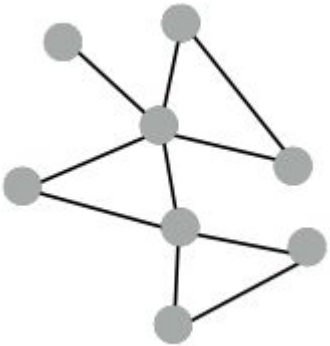


Train on one graph

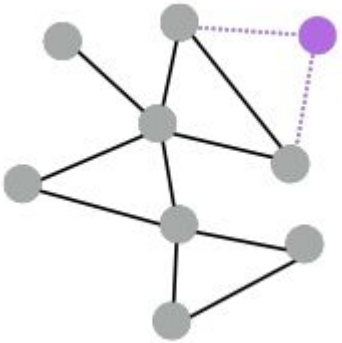



Generalize to new graph

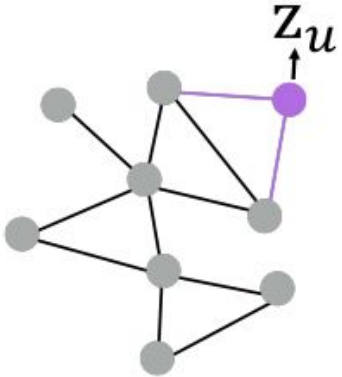
# Inductive Capability : New Nodes



Train with snapshot



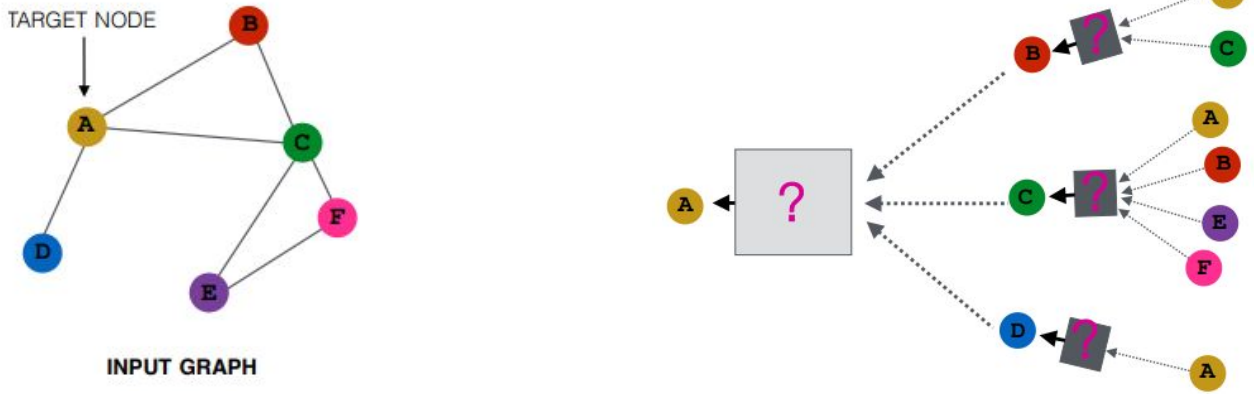
New node arrives




Generate embedding
for new node

# GraphSAGE (I)

Generalized neighborhood aggregation




$$\mathrm{h}_v^{(l+1)} = \sigma([\mathrm{W}_l \cdot \mathrm{AGG}\left(\left\{\mathrm{h}_u^{(l)}, \forall u \in N(v)\right\}\right), \mathrm{B}_l \mathrm{h}_v^{(l)}])$$

# GraphSAGE (II)

- Neighbor Aggregation:
  - **Mean:** Take a weighted average of neighbors
  - **Pool:** Transform neighbor vectors and apply symmetric vector function
  - **LSTM:** Apply LSTM to reshuffled of neighbors

- L2 Normalization:
  - Apply L2 normalization to the embedding in each layer

# Graph Attention Networks (I)

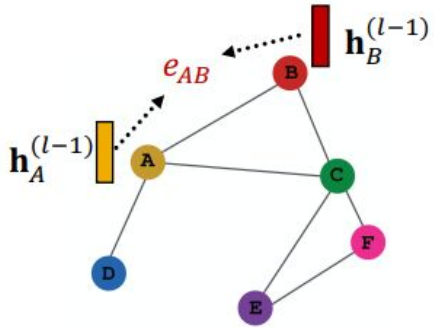
Not all node's neighbors are equally important

- Attention is inspired by cognitive attention.
- The attention focuses on the important parts of the input data and fades out the rest.
  - **Idea:** the NN should devote more computing power on that small but important part of the data.
  - Which part of the data is more important depends on the context and is learned through training.

# Graph Attention Networks (II)

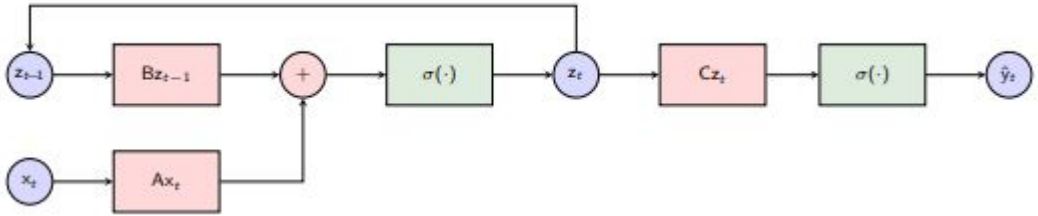
- **Goal:** Specify arbitrary importance to different neighbors of each node in the graph
- **Idea:** Compute embedding of each node in the graph following an attention strategy:
  - Nodes attend over their neighborhoods' message
  - Implicitly specifying different weights to different nodes in a neighborhood

$$e_{AB} = a(\mathbf{W}^{(l)}\mathbf{h}_A^{(l-1)}, \mathbf{W}^{(l)}\mathbf{h}_B^{(l-1)})$$

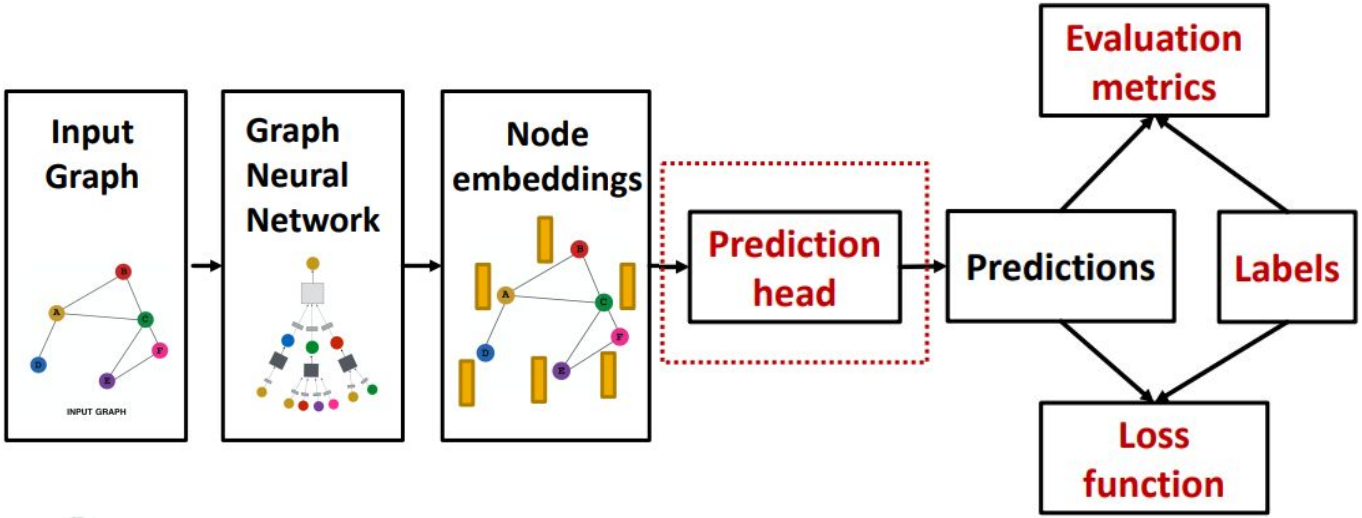


# Graph Recurrent Neural Network

- A graph recurrent neural network (GRNN) combines:
  - GNN because xt is supported on a graph.
  - RNN because xt is a sequence

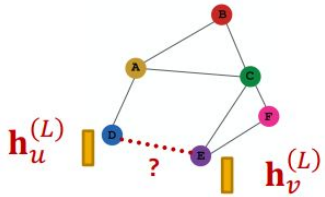


# Prediction With GNN

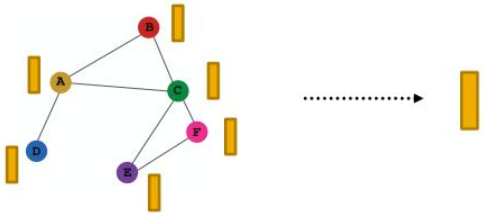


# Different prediction heads

- **Node-level tasks:** We can directly make prediction using node embeddings
- **Edge-level tasks:** Make prediction using pairs of node embeddings




- **Graph-level tasks:** Make prediction using pairs of node embeddings



# References


[1] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016)

[2] Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems* 30 (2017).

[3] You, Jiaxuan, Zhitao Ying, and Jure Leskovec. "Design space for graph neural networks." *Advances in Neural Information Processing Systems* 33 (2020): 17009-17021.

[4] Zhou, Jie, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. "Graph neural networks: A review of methods and applications." *AI open* 1 (2020): 57-81.

[5] CS224W: Machine Learning with Graphs (Stanford university)

# THANK YOU FOR YOUR ATTENTION!!!