

EECE 571F: Deep Learning with Structures

Lecture 3: Graph Neural Networks Message Passing Models

Renjie Liao

University of British Columbia

Winter, Term 1, 2022

Course Scope

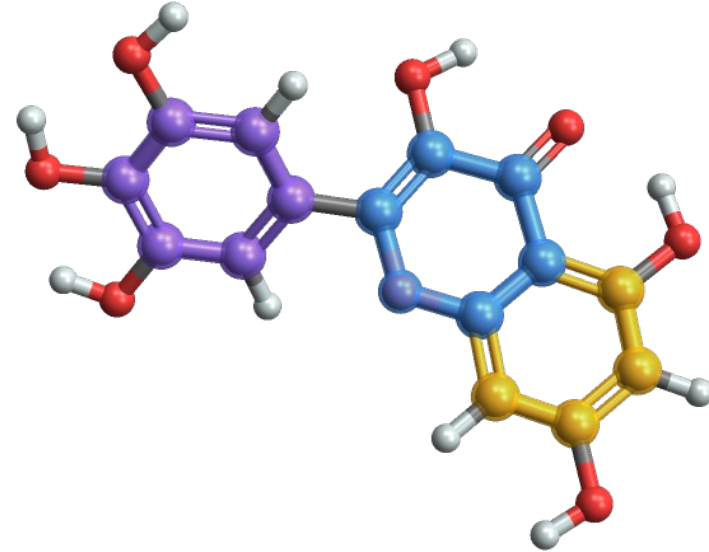
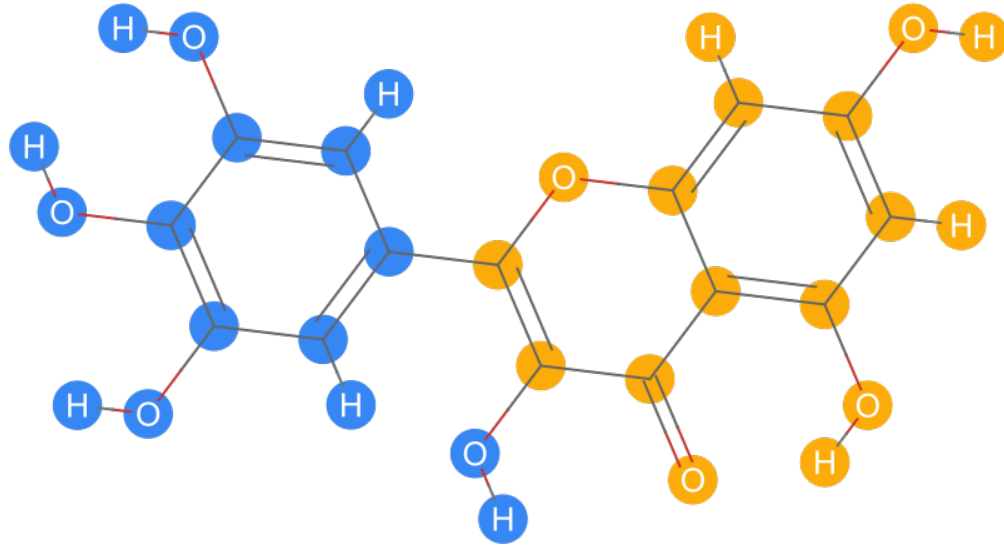
- Brief Intro to Deep Learning
- Geometric Deep Learning
 - Deep Learning Models for Sets and Sequences: Deep Sets & Transformers
 - Deep Learning Models for Graphs: Message Passing & Graph Convolution GNNs
 - Expressiveness & Generalizations of GNNs
 - Unsupervised/Self-supervised Graph Representation Learning
- Probabilistic Deep Learning
 - Deep Generative Models:
Auto-regressive models, GANs, VAEs, Diffusion/Score based models
 - Discrete/Hybrid Latent Variable Models: RBMs, Latent Graph Models
 - Stochastic Gradient Estimation

Course Scope

- Brief Intro to Deep Learning
- Geometric Deep Learning
 - Deep Learning Models for Sets and Sequences: Deep Sets & Transformers
 - Deep Learning Models for Graphs: **Message Passing** & Graph Convolution GNNs
 - Expressiveness & Generalizations of GNNs
 - Unsupervised/Self-supervised Graph Representation Learning
- Probabilistic Deep Learning
 - Deep Generative Models:
Auto-regressive models, GANs, VAEs, Diffusion/Score based models
 - Discrete/Hybrid Latent Variable Models: RBMs, Latent Graph Models
 - Stochastic Gradient Estimation

Motivating Applications of Graphs

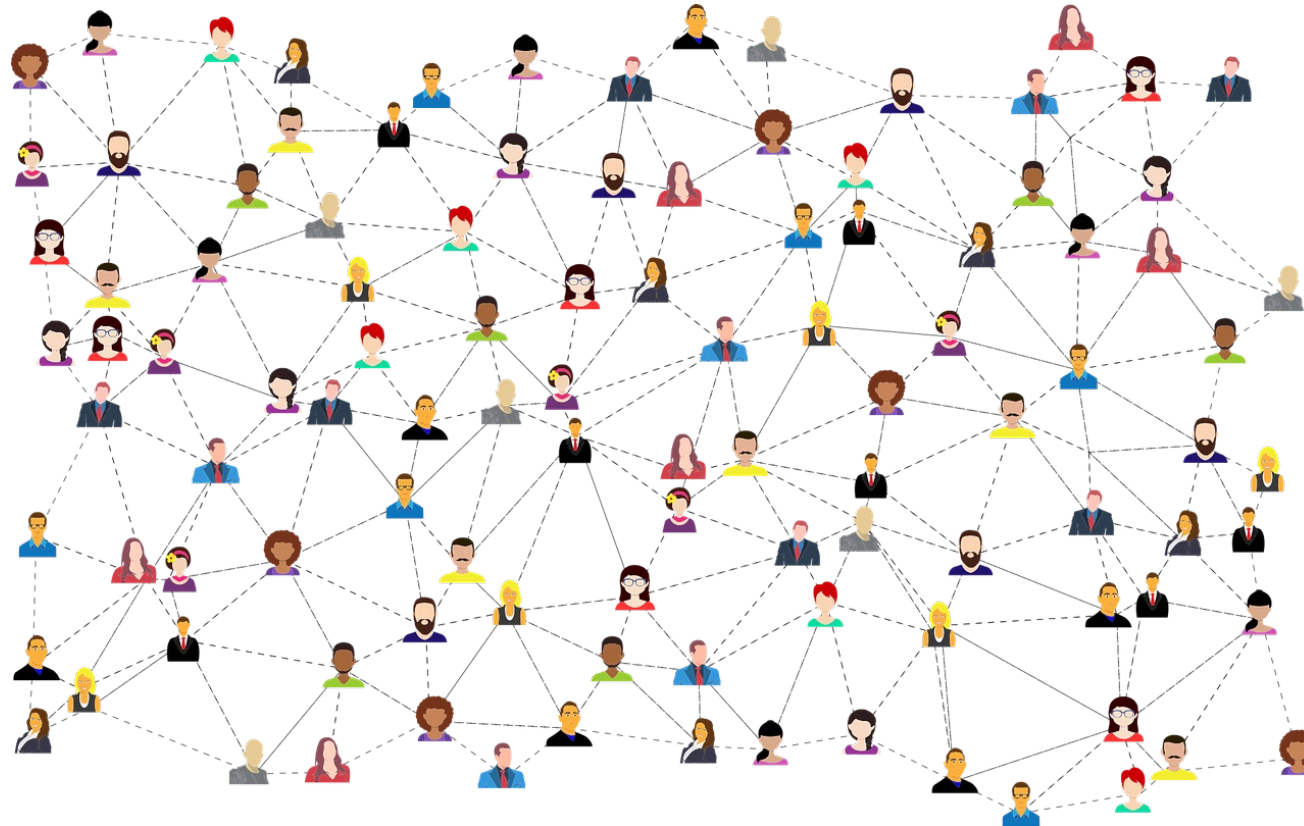
- Molecules



- Multi-edges exist
- Nodes have types
- Edges have types

Motivating Applications of Graphs

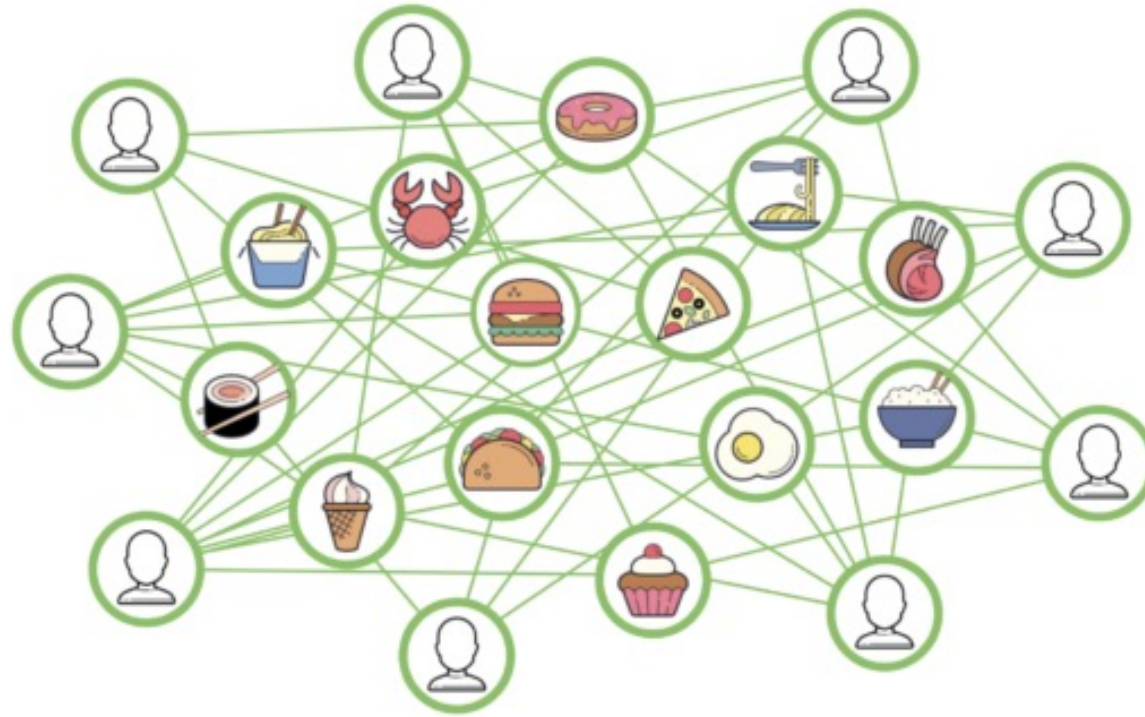
- Social Networks



Link Prediction

Motivating Applications of Graphs

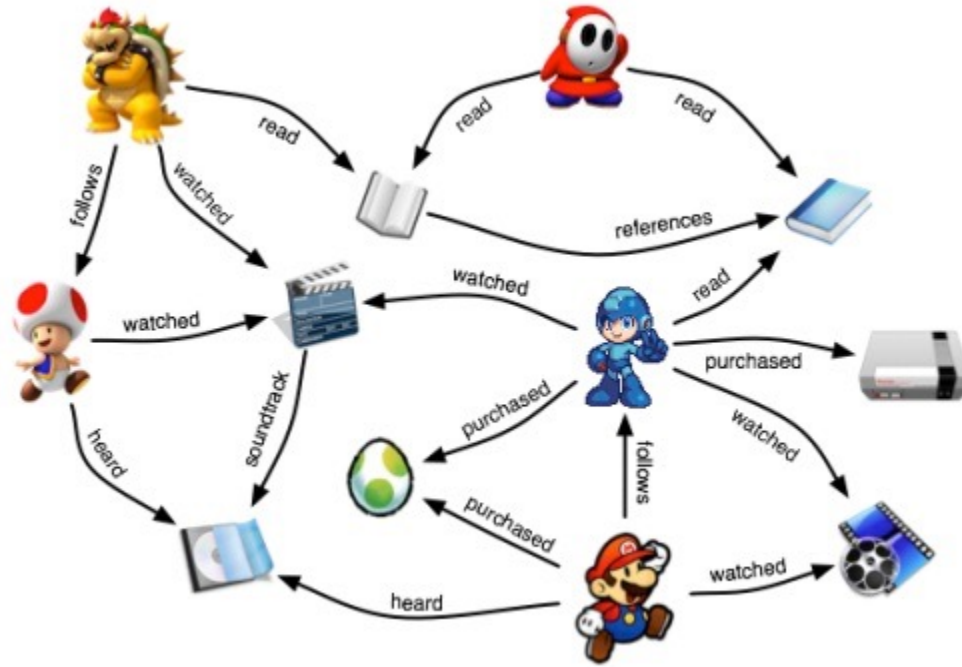
- Network-based Recommendations



Food Discovery

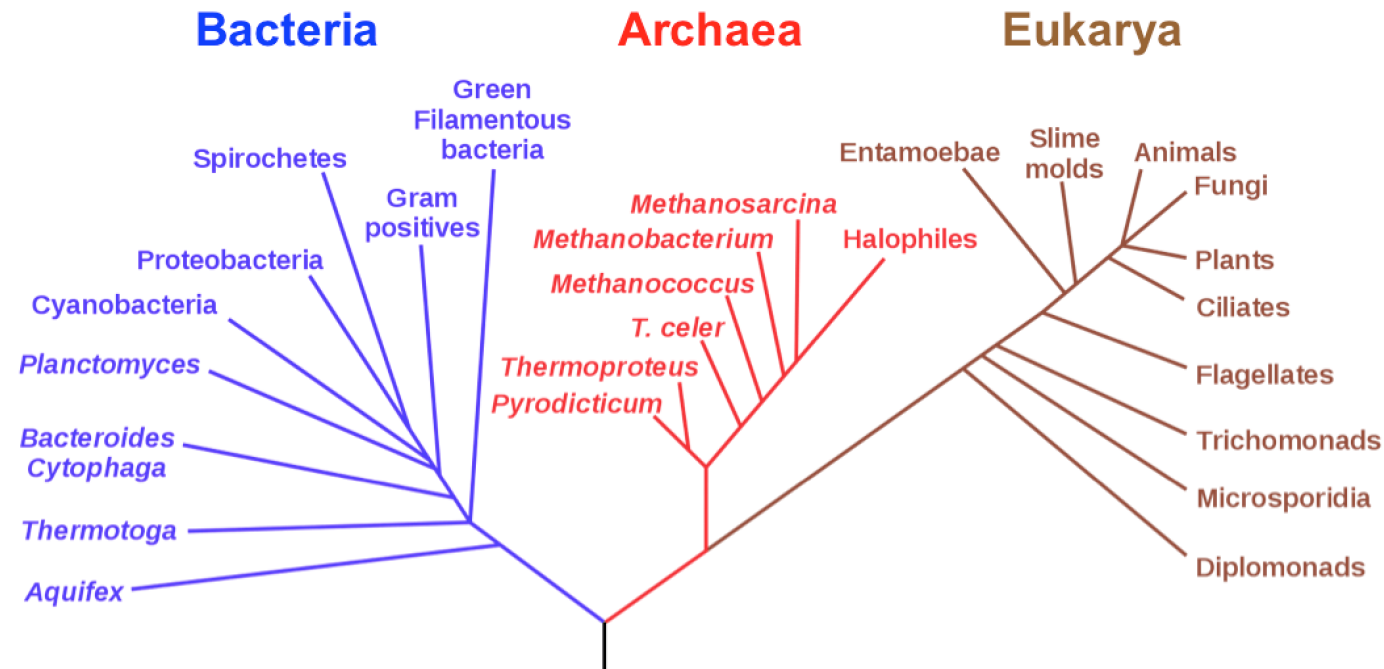
Motivating Applications of Graphs

- Network-based Recommendation



Motivating Applications of Graphs

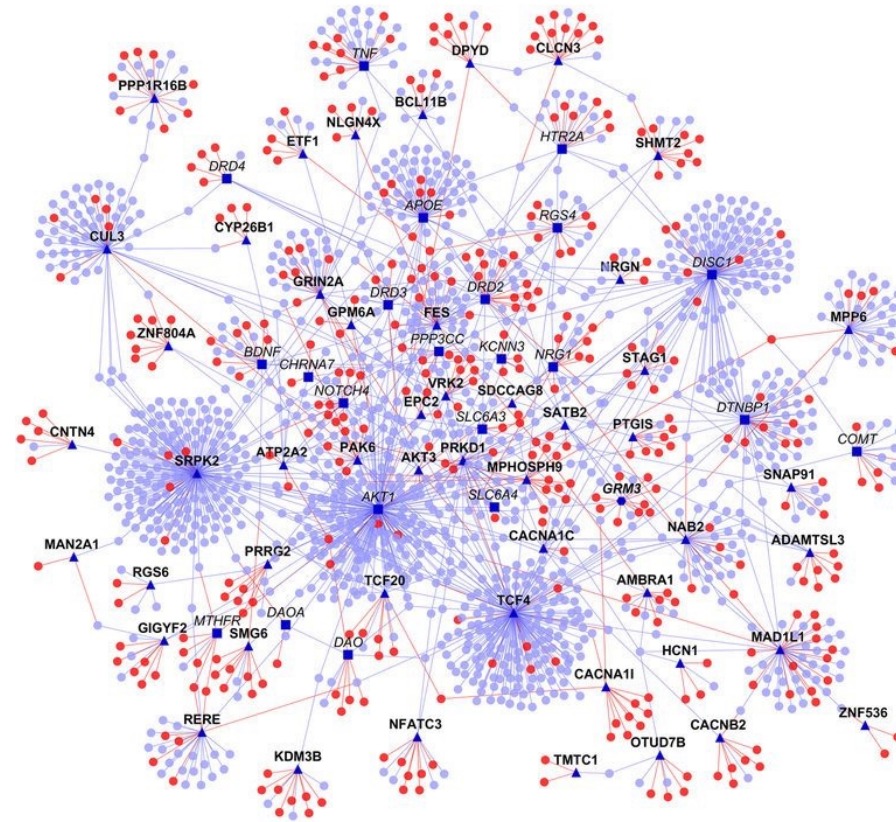
- Phylogenetic Tree



A phylogenetic tree based on rRNA genes showing the three life domains

Motivating Applications of Graphs

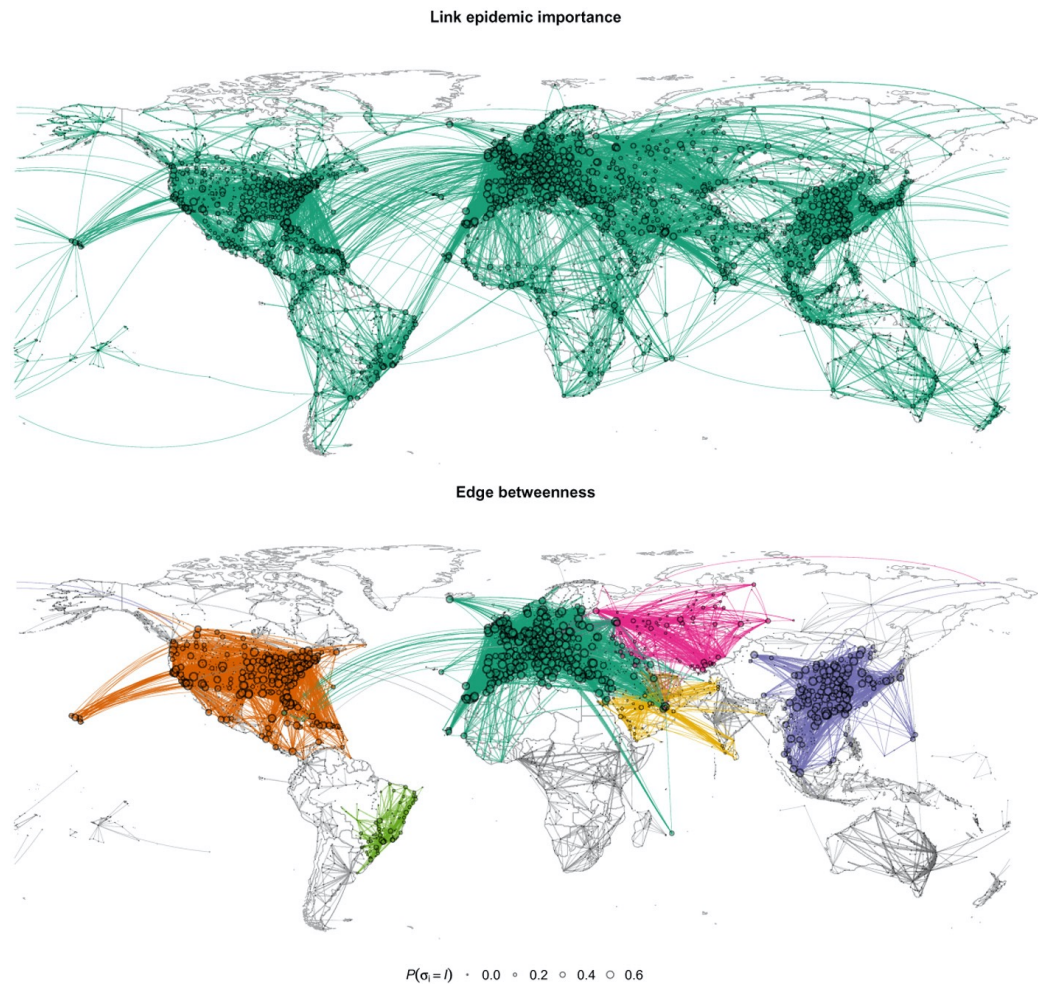
- Protein-Protein Interactions (PPIs)



Schizophrenia PPI

Motivating Applications of Graphs

- Epidemic Networks

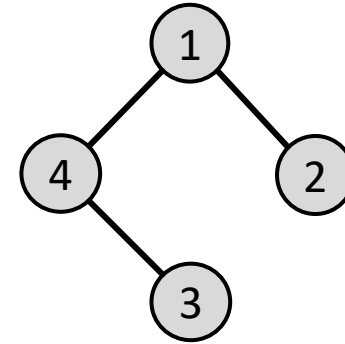


Deep Learning for Graphs

Graph Representations

- Connectivity

1. Adjacency List: $G = (V, E)$

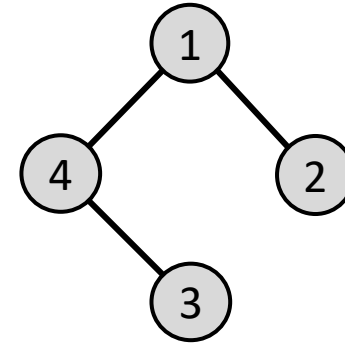


$$V = \{1,2,3,4\}, E = \{(1,2), (1,4), (4,3)\}$$

Deep Learning for Graphs

Graph Representations

- Connectivity
 1. Adjacency List: $G = (V, E)$
 2. Adjacency Matrix: A (sometimes we have weights)



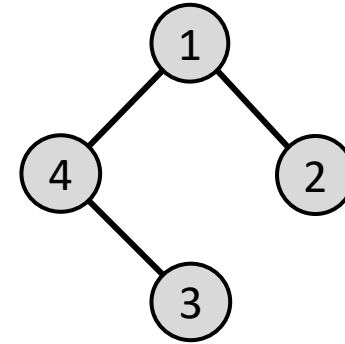
$$V = \{1,2,3,4\}, E = \{(1,2), (1,4), (4,3)\}$$

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Deep Learning for Graphs

Graph Representations

- Connectivity
 1. Adjacency List: $G = (V, E)$
 2. Adjacency Matrix: A (sometimes we have weights)
- Feature
 1. Node Feature: X
 2. Edge Feature
 3. Graph Feature



$$V = \{1,2,3,4\}, E = \{(1,2), (1,4), (4,3)\}$$

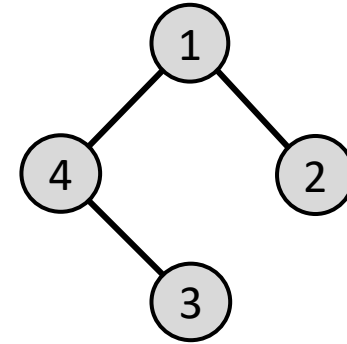
	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Deep Learning for Graphs

Graph Representations

- Connectivity
 1. Adjacency List: $G = (V, E)$
 2. Adjacency Matrix: A (sometimes we have weights)
- Feature
 1. Node Feature: X
 2. Edge Feature
 3. Graph Feature

Graph Data = (A, X)



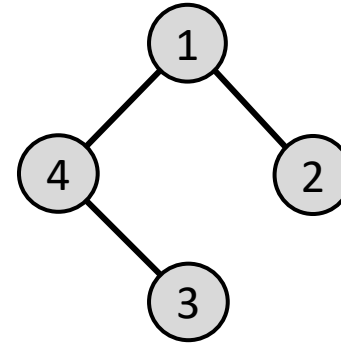
$V = \{1,2,3,4\}, E = \{(1,2), (1,4), (4,3)\}$

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Deep Learning for Graphs

Permutation

$$\begin{array}{l} V = [1,2,3,4] \\ E = [(1,2), (1,4), (4,3)] \end{array} \Rightarrow \begin{array}{l} V' = [2,1,3,4] \\ E' = [(2,1), (2,4), (4,3)] \end{array}$$



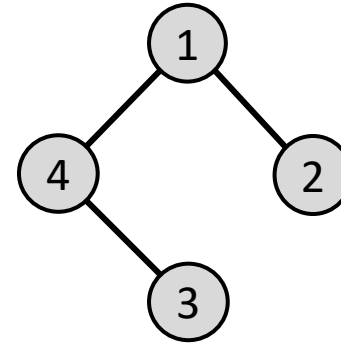
	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

Deep Learning for Graphs

Permutation

$$\begin{aligned} V = [1,2,3,4] & \Rightarrow V' = [2,1,3,4] \\ E = [(1,2), (1,4), (4,3)] & \Rightarrow E' = [(2,1), (2,4), (4,3)] \end{aligned}$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Original Adj Matrix

Deep Learning for Graphs

Permutation

$$V = [1,2,3,4]$$

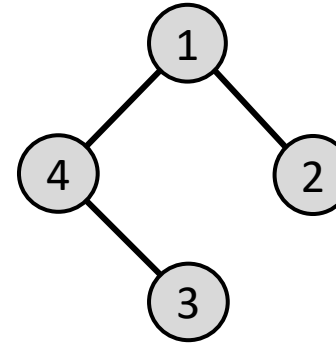
$$E = [(1,2), (1,4), (4,3)]$$

\Rightarrow

$$V' = [2,1,3,4]$$

\Rightarrow

$$E' = [(2,1), (2,4), (4,3)]$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

Permute Rows

	1	2	3	4
1	0	1	0	0
2	1	0	0	0
3	0	0	1	0
4	0	0	0	1

Permutation Matrix

Permute Columns

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Original Adj Matrix

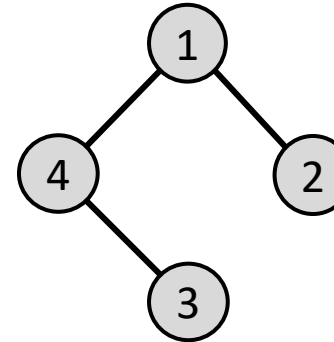
	1	2	3	4
1	0	1	0	0
2	1	0	0	0
3	0	0	1	0
4	0	0	0	1

Transposed
Permutation Matrix

Deep Learning for Graphs

Permutation

$$\begin{aligned}
 V = [1,2,3,4] & \Rightarrow V' = [2,1,3,4] \\
 E = [(1,2), (1,4), (4,3)] & \Rightarrow E' = [(2,1), (2,4), (4,3)]
 \end{aligned}$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

Permute Rows

	1	2	3	4
1	0	1	0	0
2	1	0	0	0
3	0	0	1	0
4	0	0	0	1

Permutation Matrix

Permute Columns

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Original Adj Matrix

	1	2	3	4
1	0	1	0	0
2	1	0	0	0
3	0	0	1	0
4	0	0	0	1

Transposed
Permutation Matrix

=

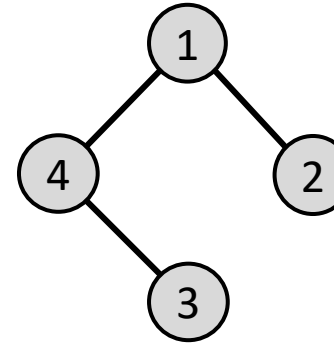
	1	2	3	4
1	0	1	0	0
2	1	0	0	1
3	0	0	0	1
4	0	1	1	0

Permuted Adj Matrix

Deep Learning for Graphs

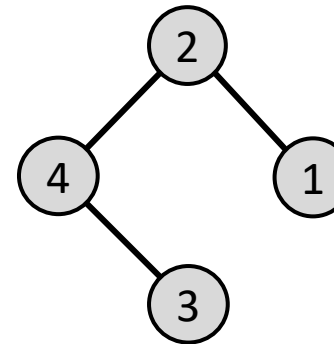
Permutation

$$\begin{aligned} V = [1,2,3,4] & \Rightarrow V' = [2,1,3,4] \\ E = [(1,2), (1,4), (4,3)] & \Rightarrow E' = [(2,1), (2,4), (4,3)] \end{aligned}$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$



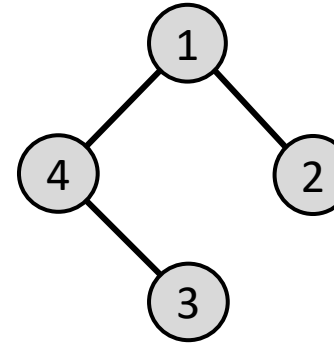
	1	2	3	4
1	0	1	0	0
2	1	0	0	1
3	0	0	0	1
4	0	1	1	0

$$V' = [2,1,3,4], E' = [(2,1), (2,4), (4,3)]$$

Deep Learning for Graphs

Permutation

$$\begin{aligned} V = [1,2,3,4] & \Rightarrow V' = [2,1,3,4] \\ E = [(1,2), (1,4), (4,3)] & \Rightarrow E' = [(2,1), (2,4), (4,3)] \end{aligned}$$



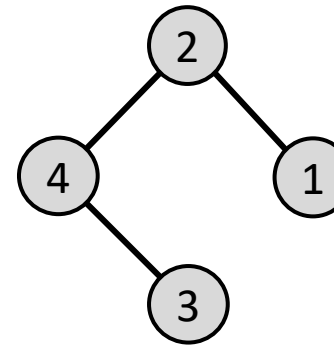
	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

Graph Isomorphism:

A bijection f between the vertex sets of $G1$ and $G2$ such that any two vertices u and v of $G1$ are adjacent **iff** $f(u)$ and $f(v)$ are adjacent in $G2$.

$$PA_1P^T = A_2$$



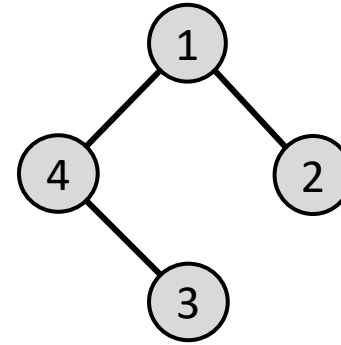
	1	2	3	4
1	0	1	0	0
2	1	0	0	1
3	0	0	0	1
4	0	1	1	0

$$V' = [2,1,3,4], E' = [(2,1), (2,4), (4,3)]$$

Deep Learning for Graphs

Permutation

$$\begin{aligned} V = [1,2,3,4] & \Rightarrow V' = [4,3,2,1] \\ E = [(1,2), (1,4), (4,3)] & \Rightarrow E' = [(4,3), (4,1), (1,2)] \end{aligned}$$



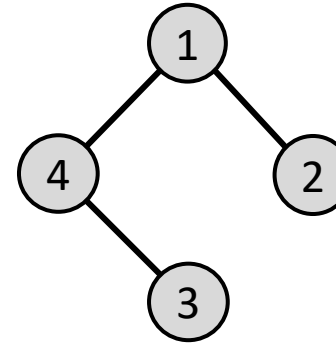
	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

Deep Learning for Graphs

Permutation

$$\begin{aligned} V = [1,2,3,4] & \Rightarrow V' = [4,3,2,1] \\ E = [(1,2), (1,4), (4,3)] & \Rightarrow E' = [(4,3), (4,1), (1,2)] \end{aligned}$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Original Adj Matrix

Deep Learning for Graphs

Permutation

$$V = [1,2,3,4]$$

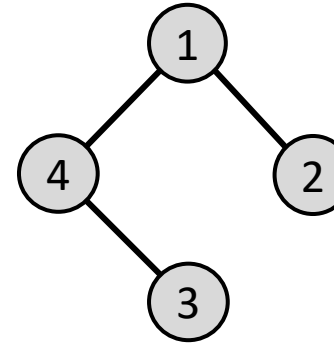
$$E = [(1,2), (1,4), (4,3)]$$

=>

$$V' = [4,3,2,1]$$

=>

$$E' = [(4,3), (4,1), (1,2)]$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

Permute Rows

	1	2	3	4
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0

Permutation Matrix

Permute Columns

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Original Adj Matrix

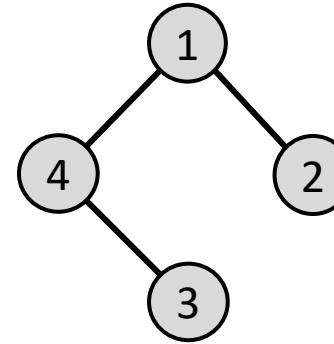
	1	2	3	4
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0

Transposed
Permutation Matrix

Deep Learning for Graphs

Permutation

$$\begin{aligned} V = [1,2,3,4] & \Rightarrow V' = [4,3,2,1] \\ E = [(1,2), (1,4), (4,3)] & \Rightarrow E' = [(4,3), (4,1), (1,2)] \end{aligned}$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

Permute Rows

	1	2	3	4
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0

Permutation Matrix

Permute Columns

	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Original Adj Matrix

	1	2	3	4
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0

Transposed
Permutation Matrix

=

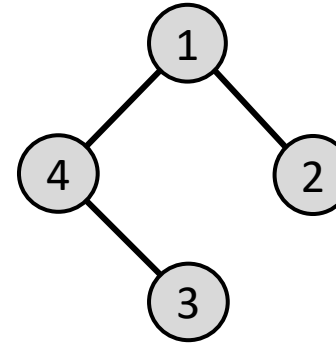
	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

Permuted Adj Matrix

Deep Learning for Graphs

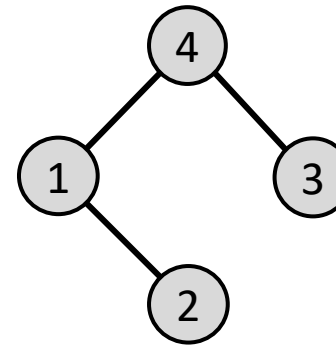
Permutation

$$\begin{aligned} V = [1,2,3,4] & \Rightarrow V' = [4,3,2,1] \\ E = [(1,2), (1,4), (4,3)] & \Rightarrow E' = [(4,3), (4,1), (1,2)] \end{aligned}$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$



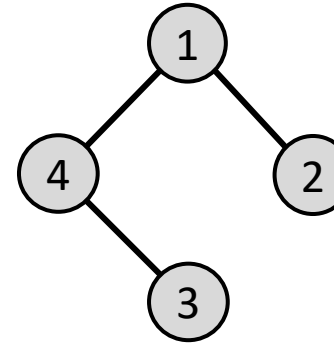
	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V' = [4,3,2,1], E' = [(4,3), (4,1), (1,2)]$$

Deep Learning for Graphs

Permutation

$$\begin{aligned}
 V = [1,2,3,4] & \Rightarrow V' = [4,3,2,1] \\
 E = [(1,2), (1,4), (4,3)] & \Rightarrow E' = [(4,3), (4,1), (1,2)]
 \end{aligned}$$



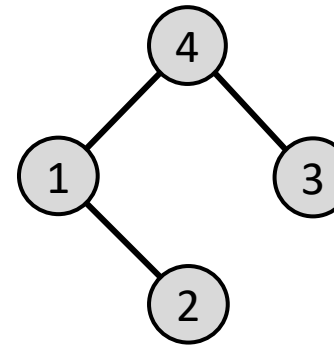
	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V = [1,2,3,4], E = [(1,2), (1,4), (4,3)]$$

Graph Automorphism:

A permutation σ of the vertex set V , such that the pair of vertices (u,v) form an edge **iff** the pair $(\sigma(u),\sigma(v))$ also form an edge.

$$PAP^T = A$$



	1	2	3	4
1	0	1	0	1
2	1	0	0	0
3	0	0	0	1
4	1	0	1	0

$$V' = [4,3,2,1], E' = [(4,3), (4,1), (1,2)]$$

Deep Learning for Graphs

Permutation Invariance & Equivariance

Graph Data (A, X) , Model $f(A, X)$

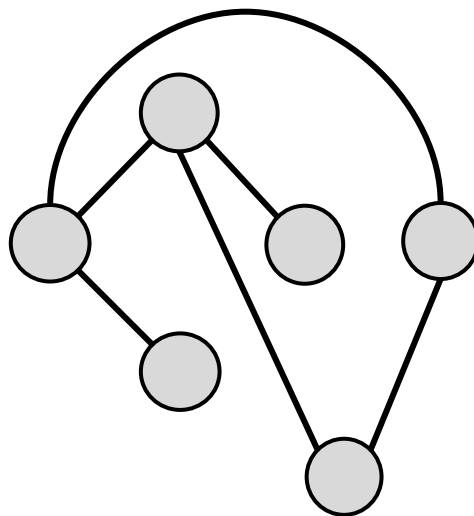
Invariance: $f(PAP^\top, PX) = f(A, X)$

Equivariance: $f(PAP^\top, PX) = Pf(A, X)$

Deep Learning for Graphs

Key Challenges:

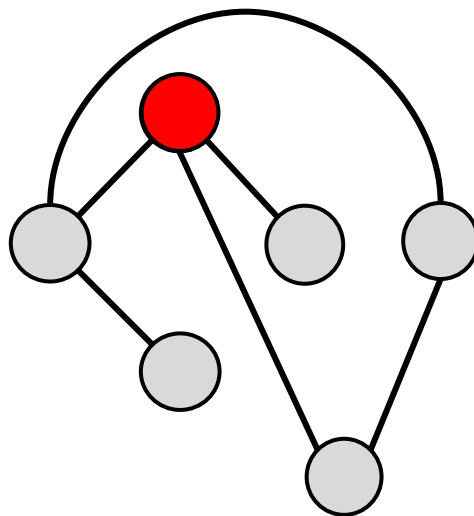
- **Unordered Neighbors**



Deep Learning for Graphs

Key Challenges:

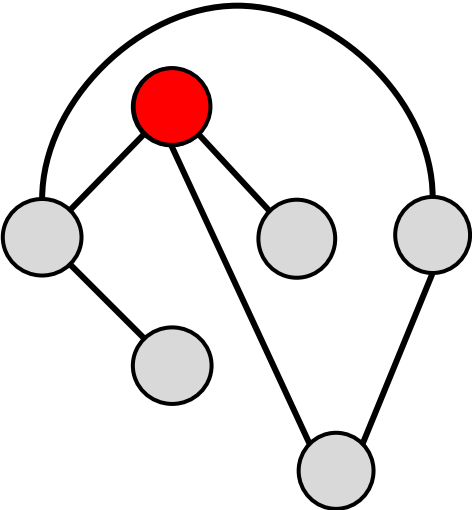
- **Unordered Neighbors**



Deep Learning for Graphs

Key Challenges:

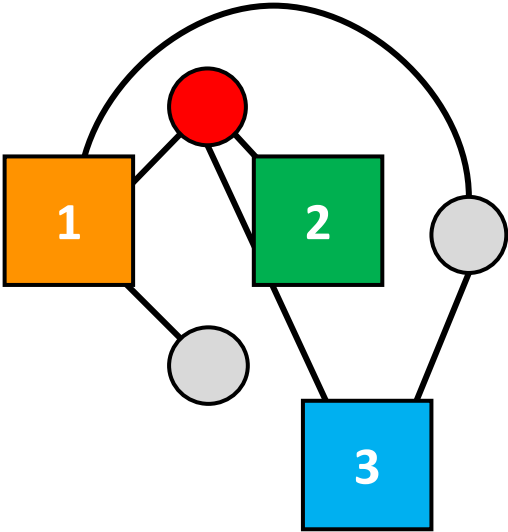
- **Unordered Neighbors**



Deep Learning for Graphs

Key Challenges:

- **Unordered Neighbors**



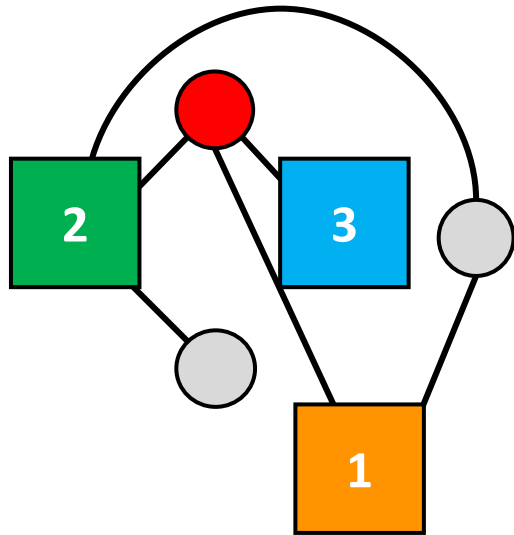
Option 1



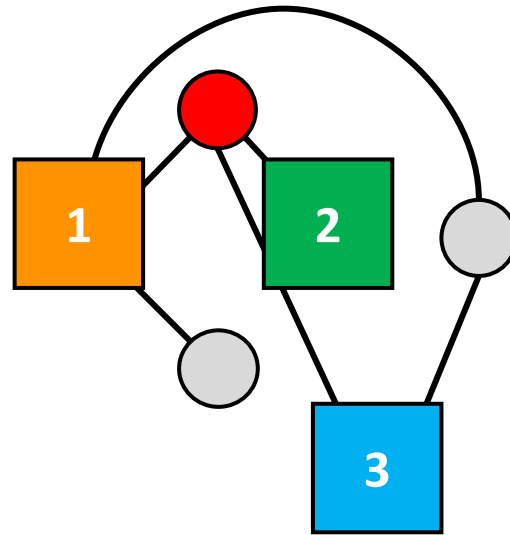
Deep Learning for Graphs

Key Challenges:

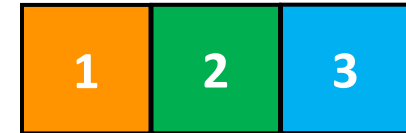
- **Unordered Neighbors**



Option 2



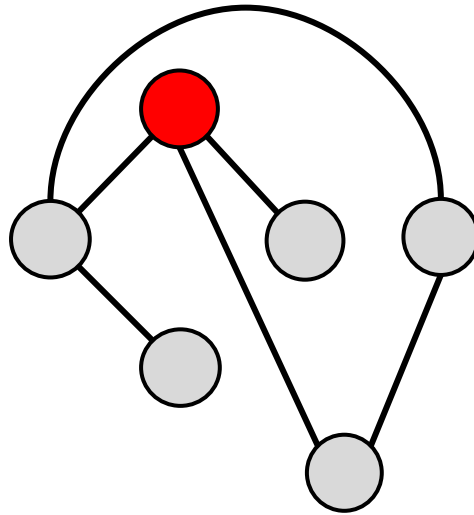
Option 1



Deep Learning for Graphs

Key Challenges:

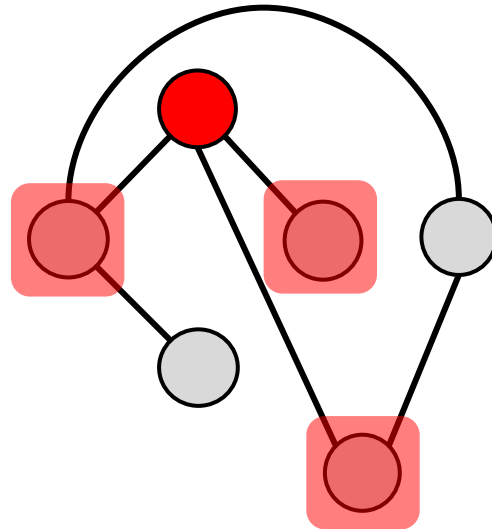
- Unordered Neighbors
- **Varying Neighborhood Sizes**



Deep Learning for Graphs

Key Challenges:

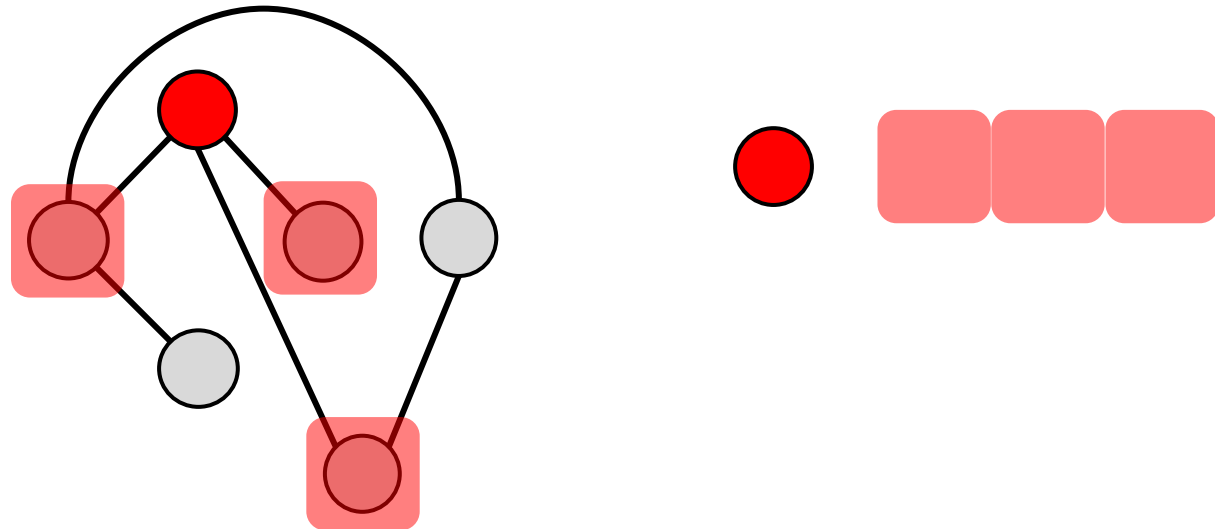
- Unordered Neighbors
- **Varying Neighborhood Sizes**



Deep Learning for Graphs

Key Challenges:

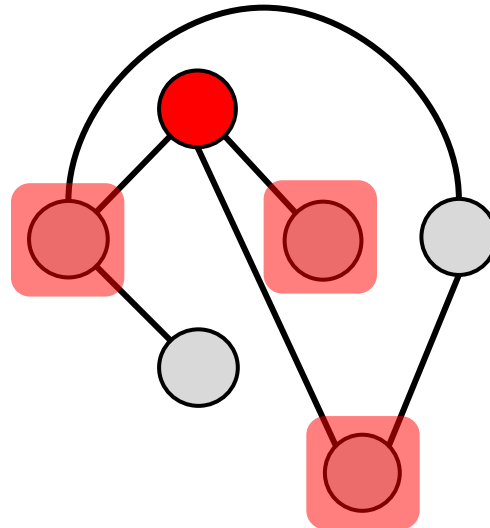
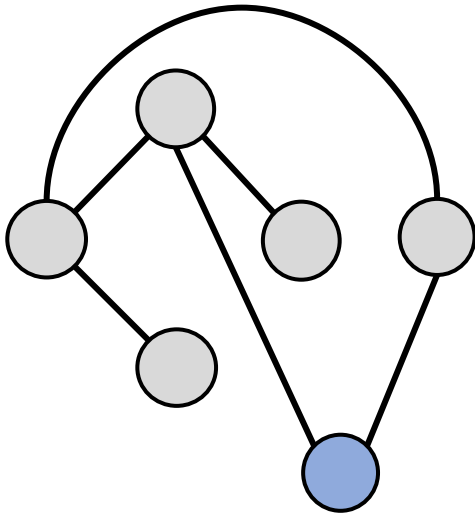
- Unordered Neighbors
- **Varying Neighborhood Sizes**



Deep Learning for Graphs

Key Challenges:

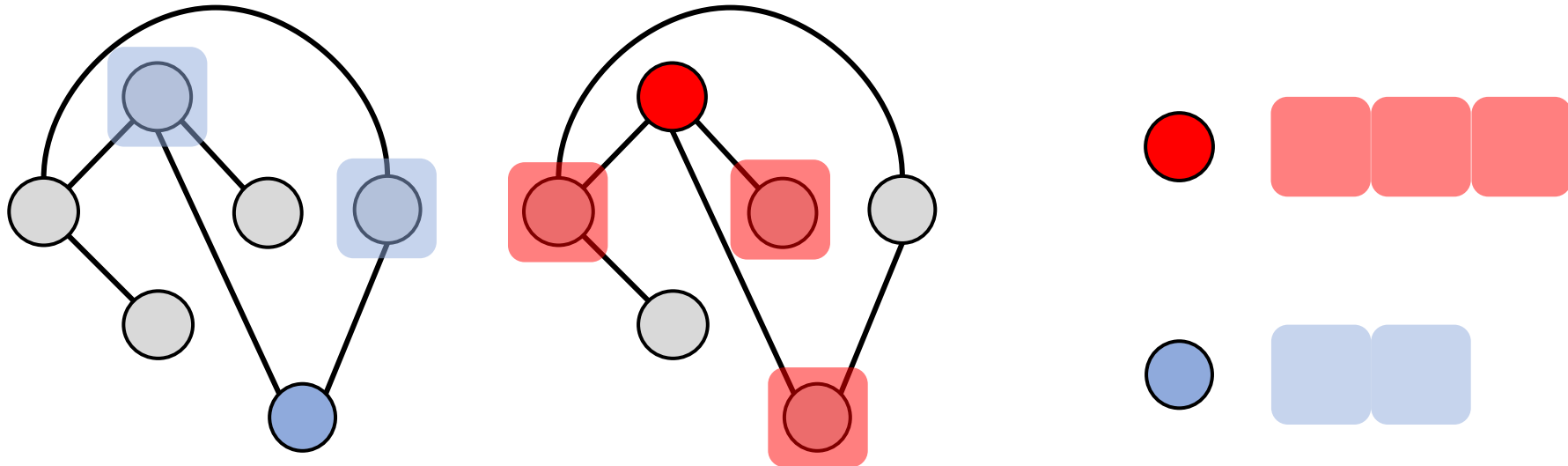
- Unordered Neighbors
- **Varying Neighborhood Sizes**



Deep Learning for Graphs

Key Challenges:

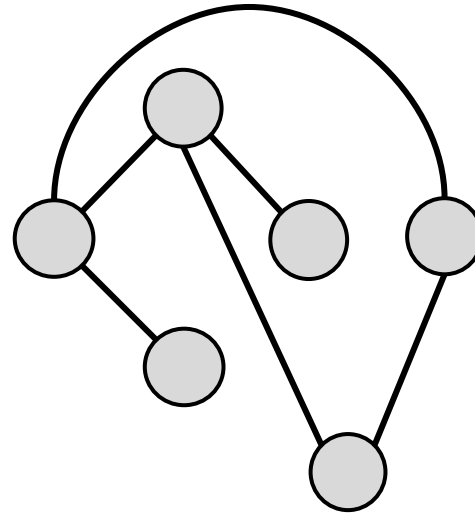
- Unordered Neighbors
- **Varying Neighborhood Sizes**



Deep Learning for Graphs

Key Challenges:

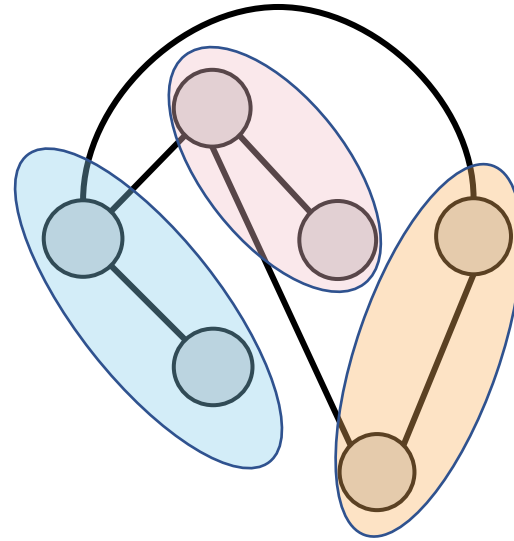
- Unordered Neighbors
- Varying Neighborhood Sizes
- **Varying Graph Partitions**



Deep Learning for Graphs

Key Challenges:

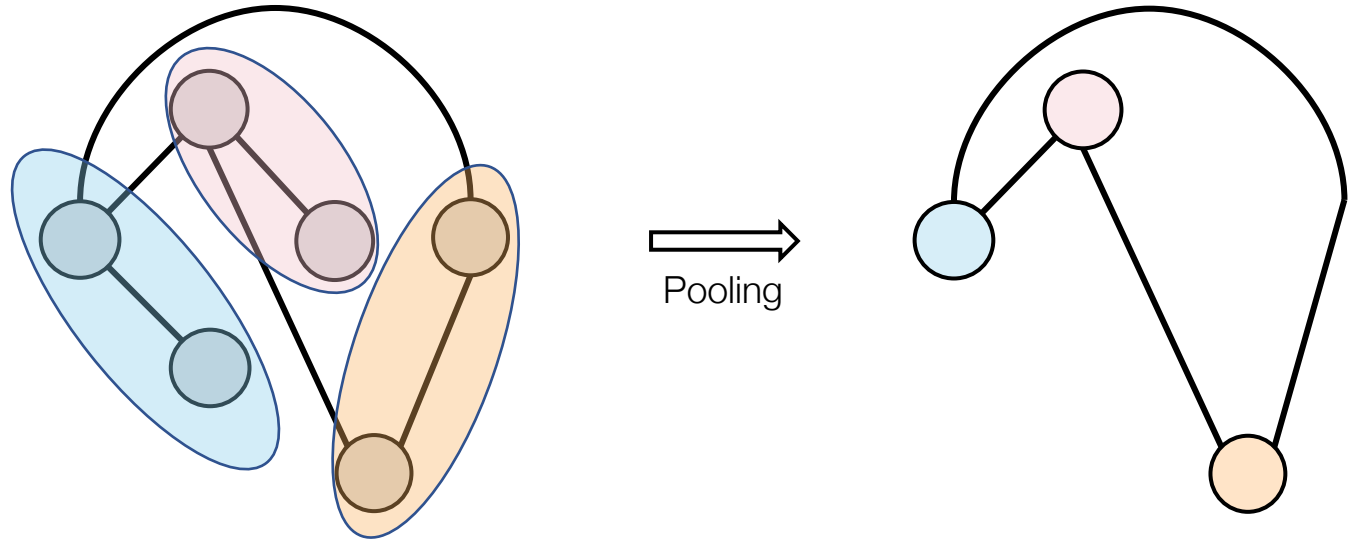
- Unordered Neighbors
- Varying Neighborhood Sizes
- **Varying Graph Partitions**



Deep Learning for Graphs

Key Challenges:

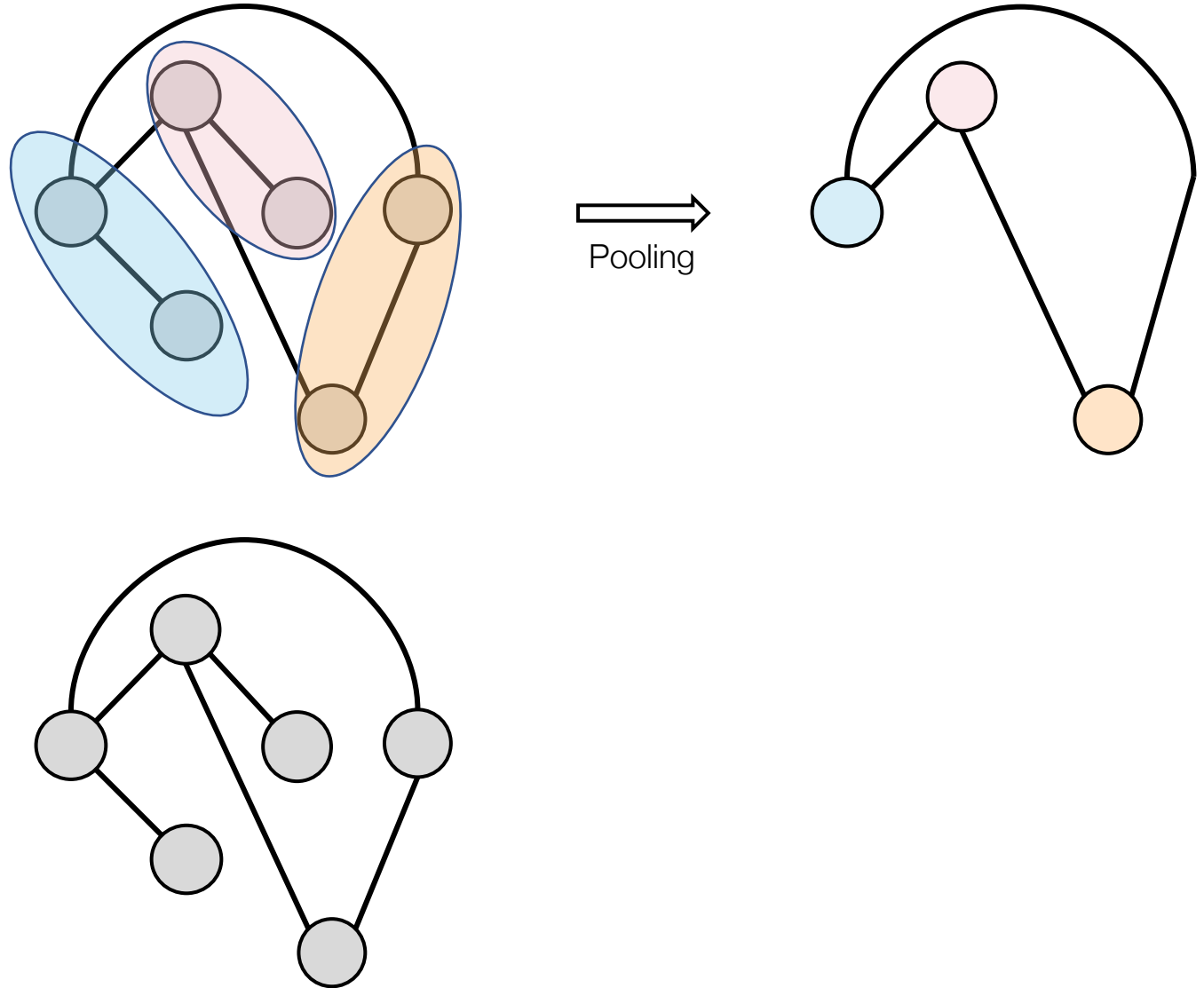
- Unordered Neighbors
- Varying Neighborhood Sizes
- **Varying Graph Partitions**



Deep Learning for Graphs

Key Challenges:

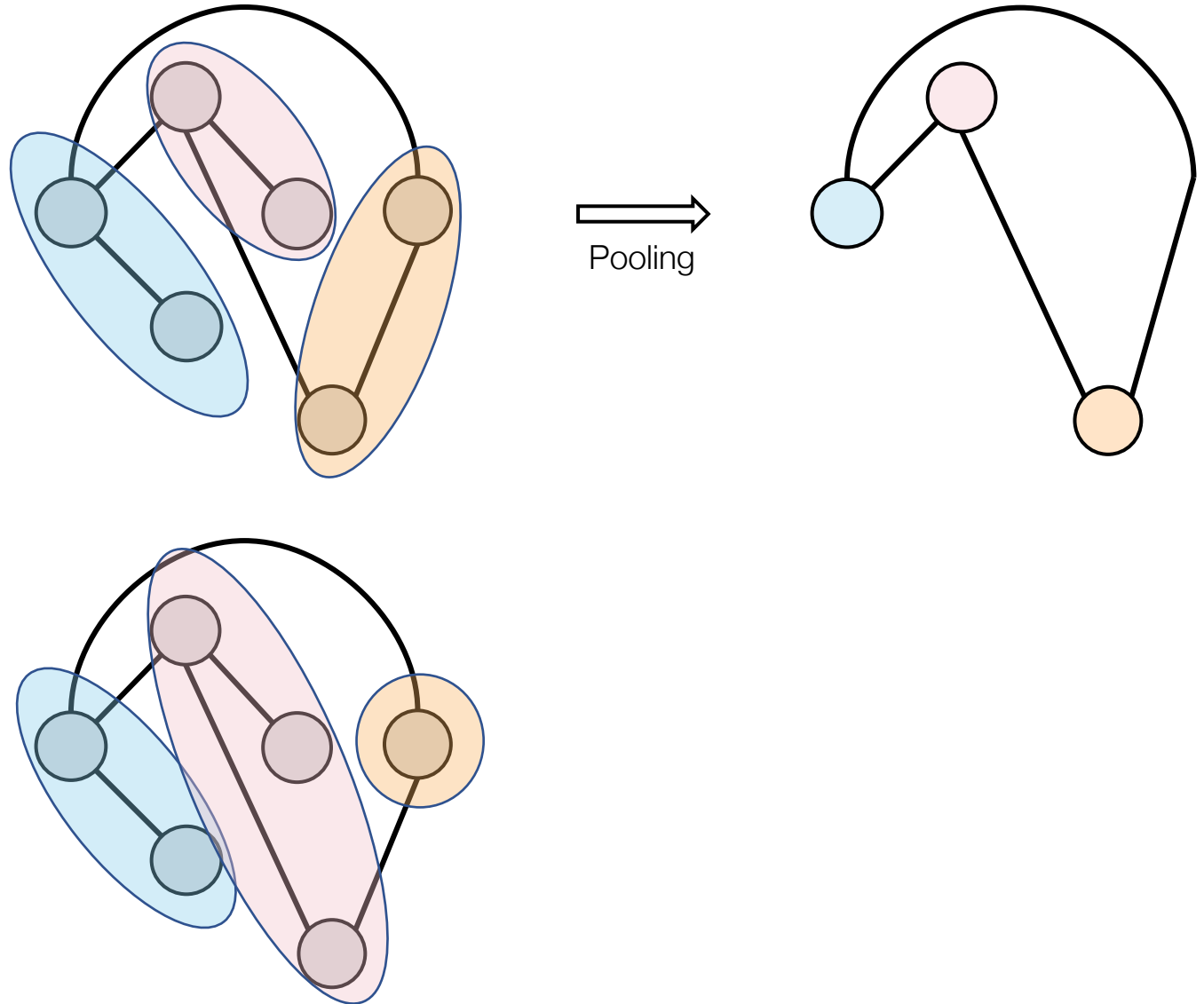
- Unordered Neighbors
- Varying Neighborhood Sizes
- **Varying Graph Partitions**



Deep Learning for Graphs

Key Challenges:

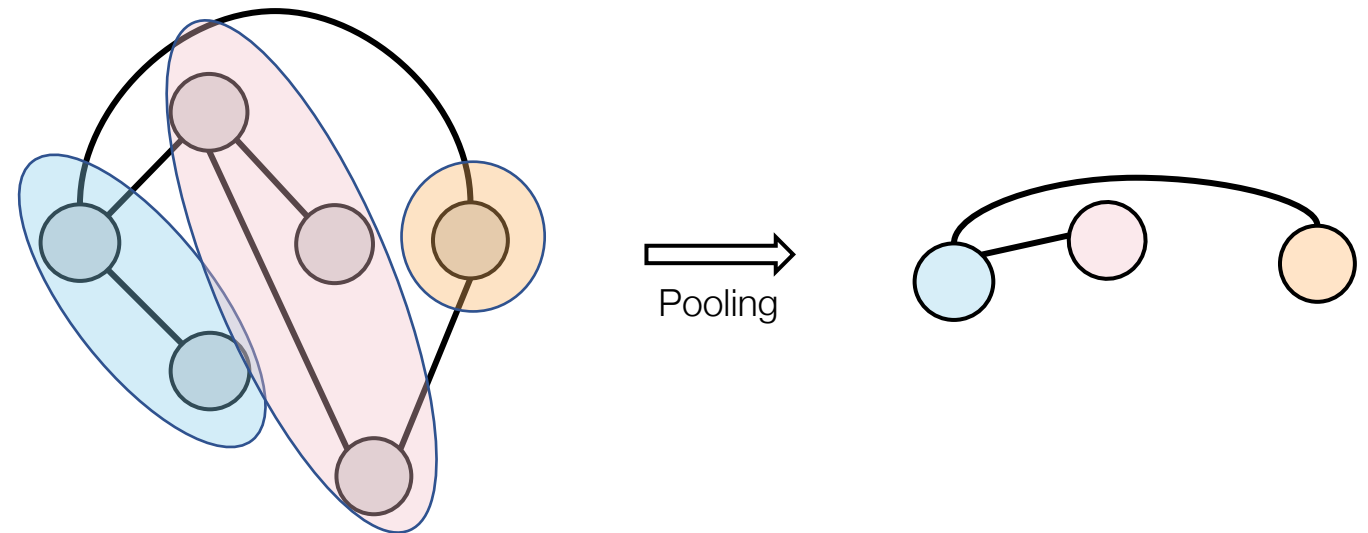
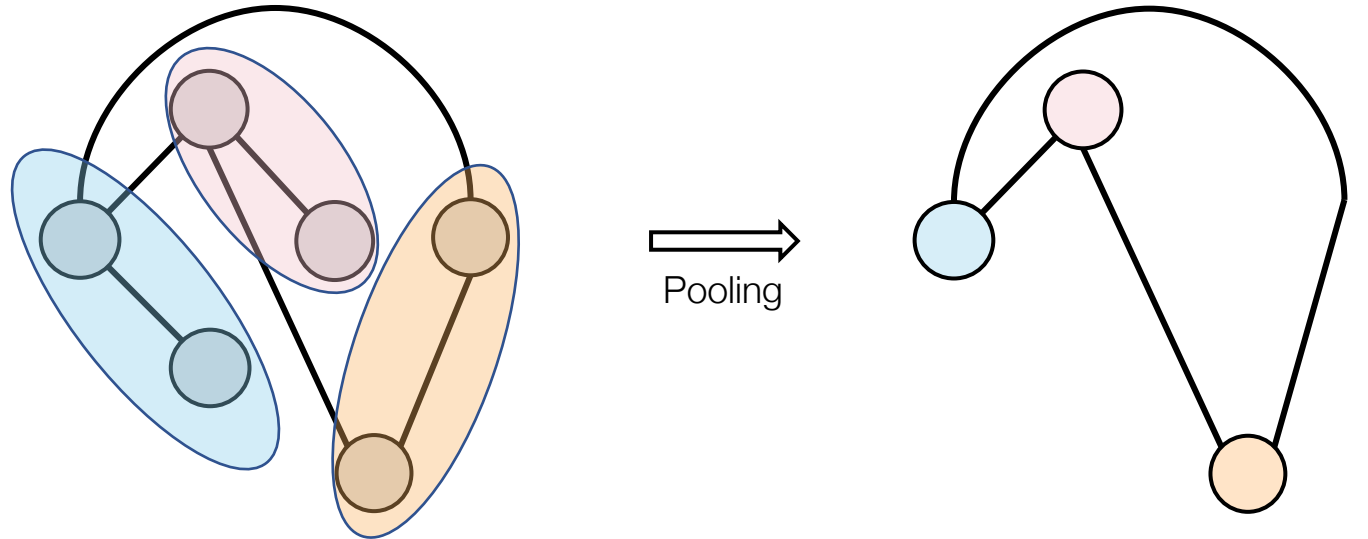
- Unordered Neighbors
- Varying Neighborhood Sizes
- **Varying Graph Partitions**



Deep Learning for Graphs

Key Challenges:

- Unordered Neighbors
- Varying Neighborhood Sizes
- **Varying Graph Partitions**



Deep Learning for Graphs

Graph Neural Networks (GNNs)

- Neural networks that can process general graph structured data

Deep Learning for Graphs

Graph Neural Networks (GNNs)

- Neural networks that can process general graph structured data
- First proposed in 2008 [1] and dates back to Recursive Neural Networks (mainly processing trees) in 90s [2]
- In fact, Boltzmann Machines [3] (fully connected graphs with binary units) in 80s can be viewed as GNNs

Deep Learning for Graphs

Graph Neural Networks (GNNs)

- Neural networks that can process general graph structured data
- First proposed in 2008 [1] and dates back to Recursive Neural Networks (mainly processing trees) in 90s [2]
- In fact, Boltzmann Machines [3] (fully connected graphs with binary units) in 80s can be viewed as GNNs
- Most of GNNs (if not all) can be incorporated by the **Message Passing** paradigm

Deep Learning for Graphs

Graph Neural Networks (GNNs)

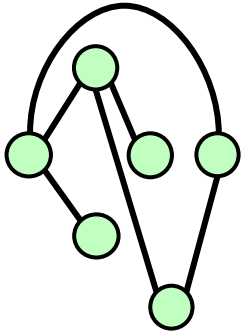
- Neural networks that can process general graph structured data
- First proposed in 2008 [1] and dates back to Recursive Neural Networks (mainly processing trees) in 90s [2]
- In fact, Boltzmann Machines [3] (fully connected graphs with binary units) in 80s can be viewed as GNNs
- Most of GNNs (if not all) can be incorporated by the **Message Passing** paradigm
- GNNs have been independently studied in signal processing community under **Graph Signal Processing**

Deep Learning for Graphs

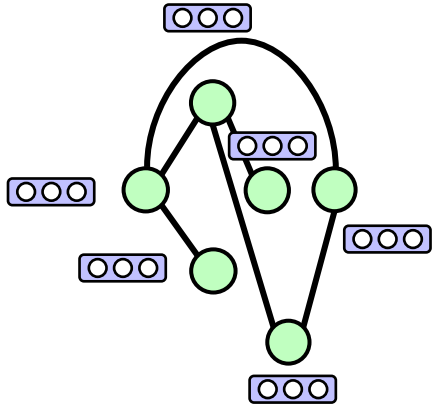
Graph Neural Networks (GNNs)

- Neural networks that can process general graph structured data
- First proposed in 2008 [1] and dates back to Recursive Neural Networks (mainly processing trees) in 90s [2]
- In fact, Boltzmann Machines [3] (fully connected graphs with binary units) in 80s can be viewed as GNNs
- Most of GNNs (if not all) can be incorporated by the **Message Passing** paradigm
- GNNs have been independently studied in signal processing community under **Graph Signal Processing**
- The study of GNNs and other related models are also called **Geometric Deep Learning**

Graph Neural Networks (GNNs)

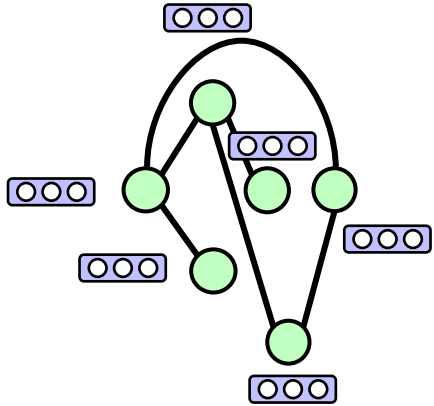


Graph Neural Networks (GNNs)



Input Encoding

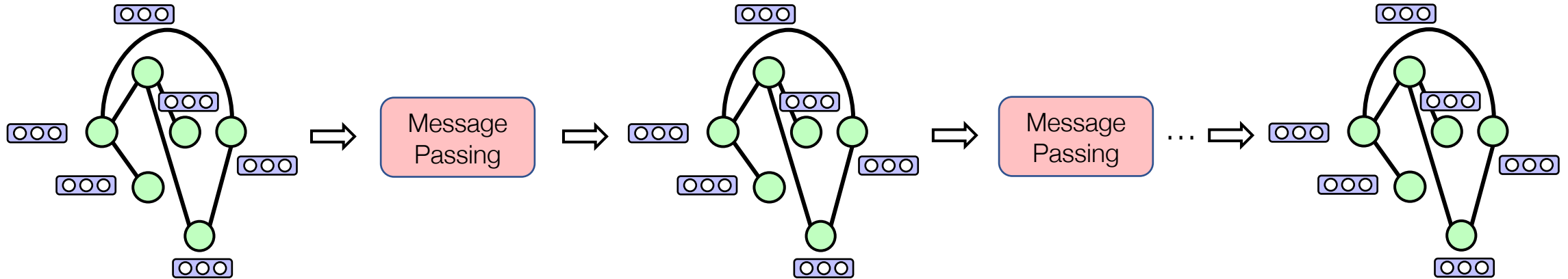
Graph Neural Networks (GNNs)



Input Encoding

1. Node Feature
 - *If it is unavailable, use 1-of-K, random, index/size encoding of node index)*
2. Edge Feature
 - *Feed it to message network*
3. Graph Feature
 - *Treat it as a super node in your graph*
 - *Feed graph feature to readout layer*

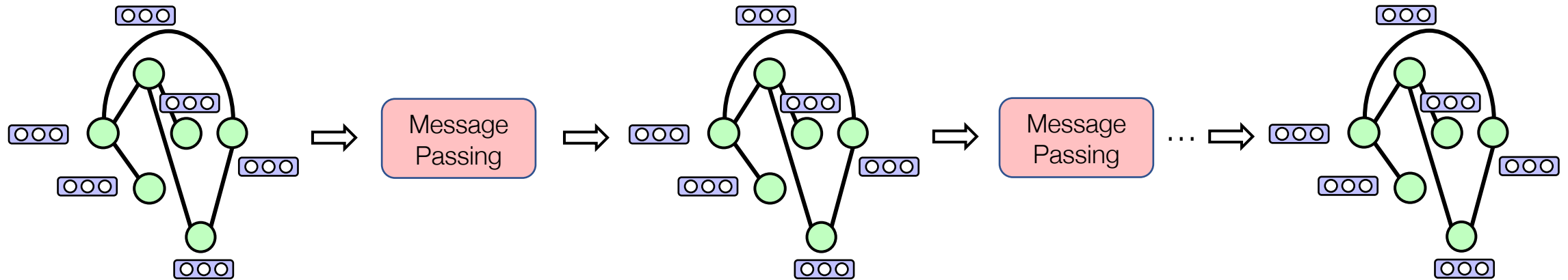
Graph Neural Networks (GNNs)



Input Encoding

Message Passing Layers/Steps

Graph Neural Networks (GNNs)

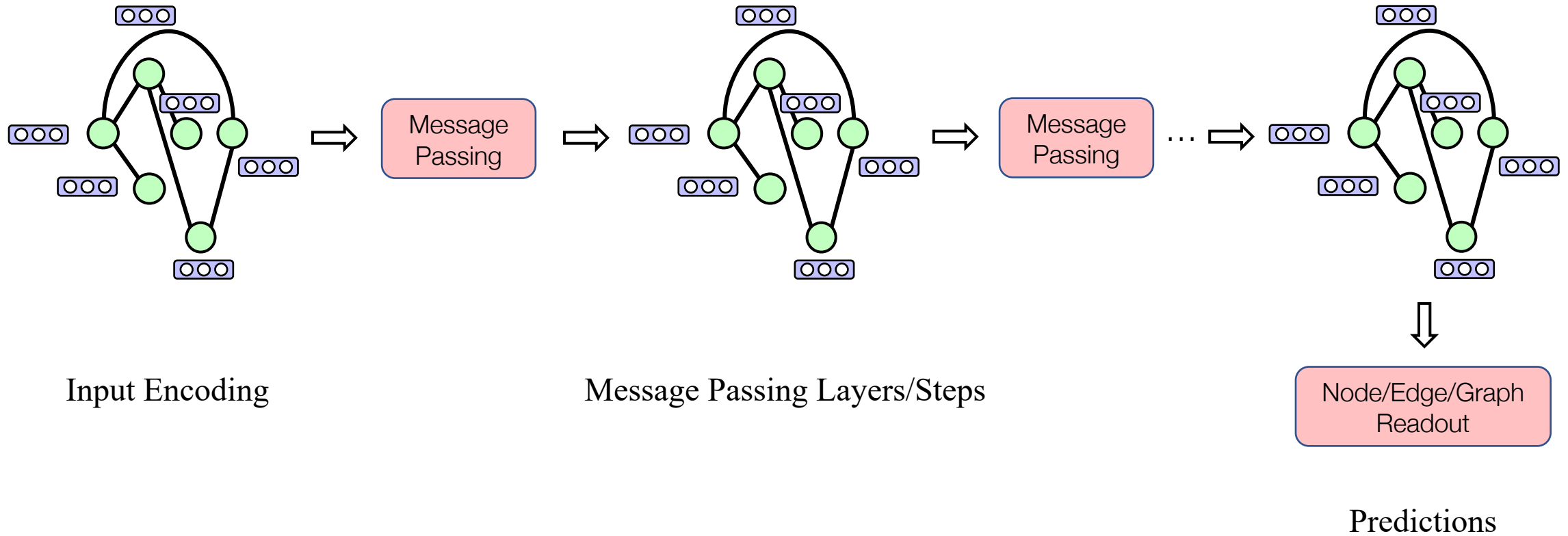


Input Encoding

Message Passing Layers/Steps

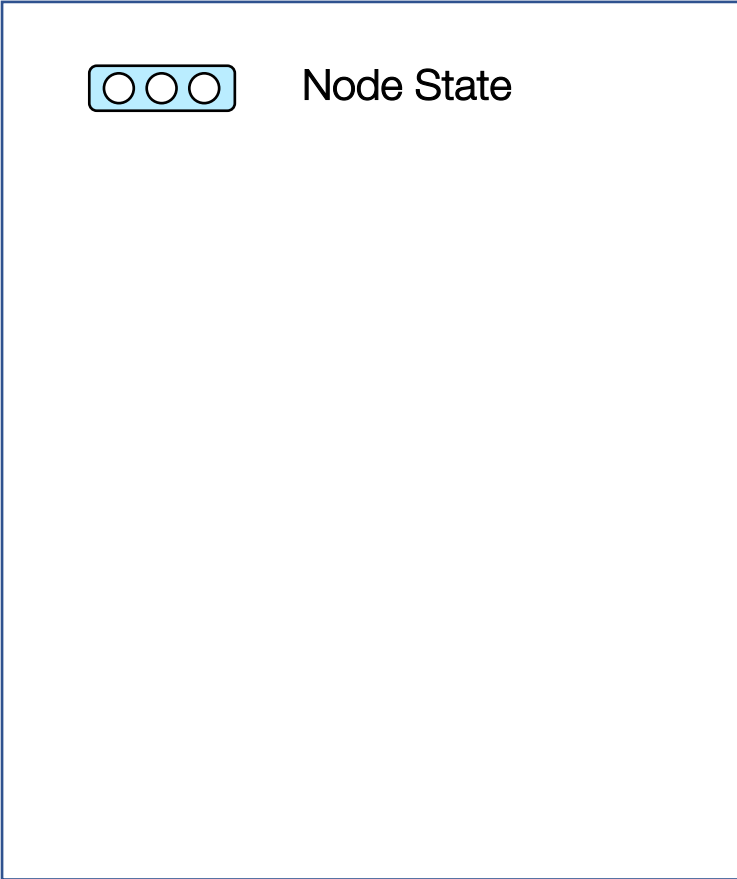
Steps: share message passing module (Recurrent Networks)
Layers: do not share message passing module (Feedforward Networks)

Graph Neural Networks (GNNs)

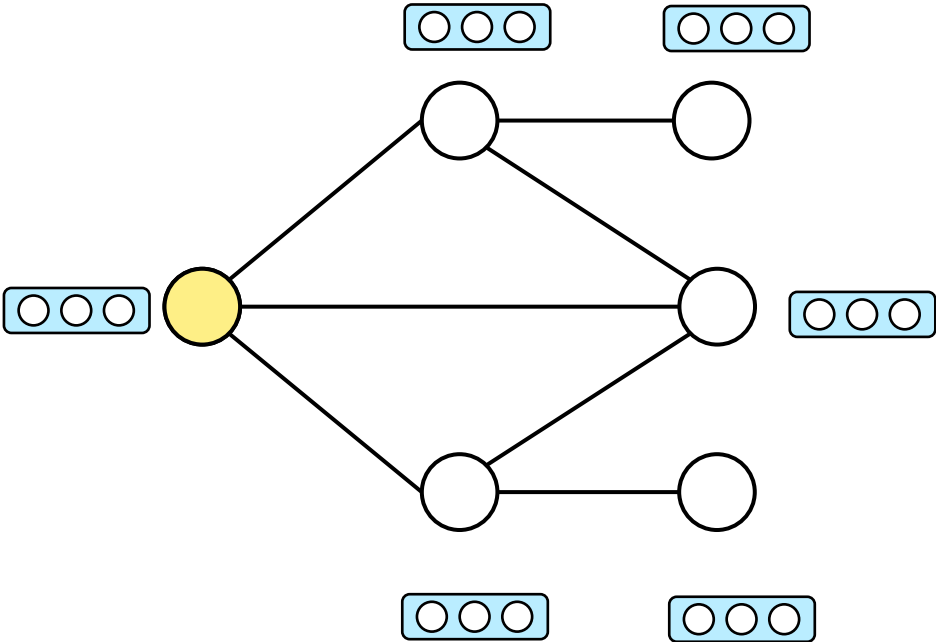


Message Passing in GNNs

\mathbf{h}_i^t

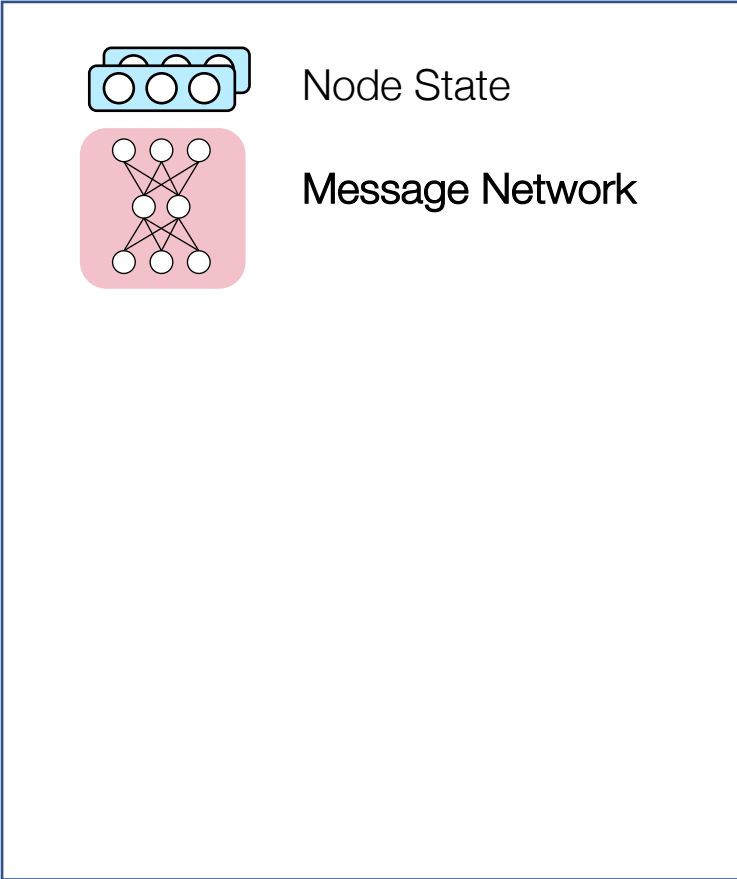


(t+1)-th message passing step/layer

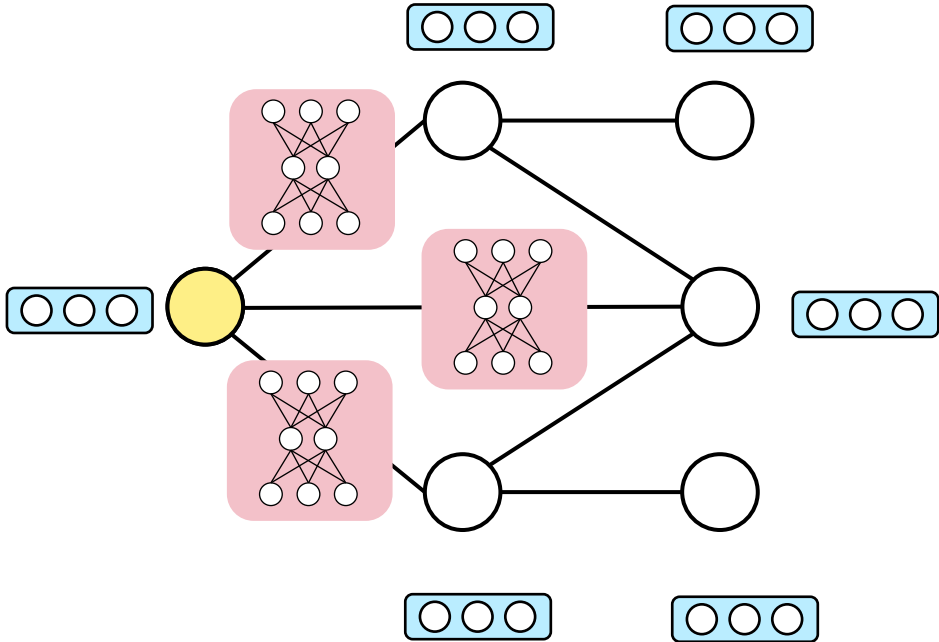


Message Passing in GNNs

\mathbf{h}_i^t \mathbf{h}_j^t

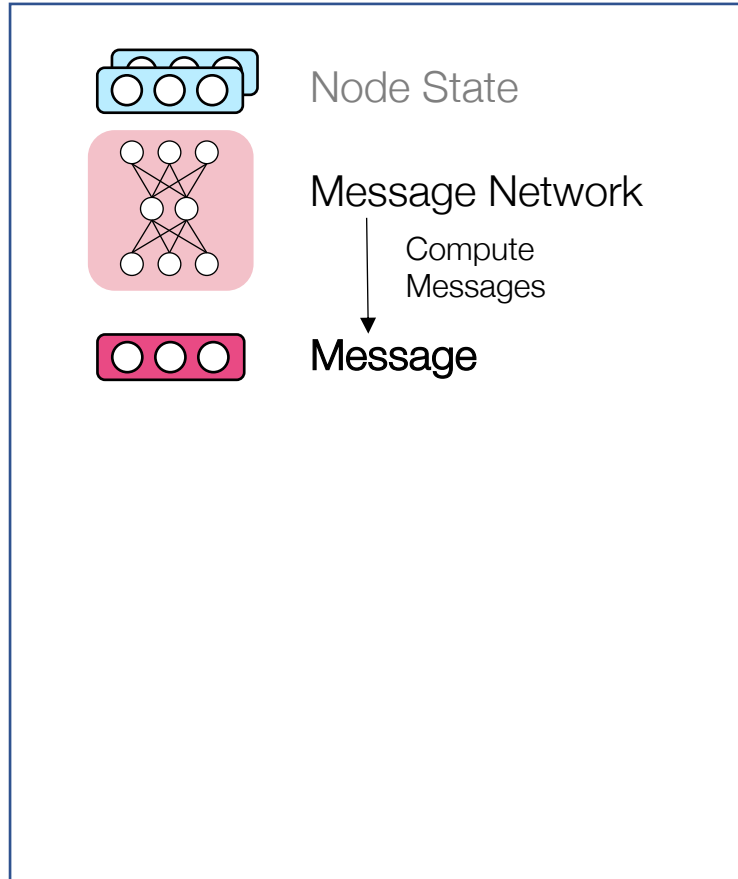


(t+1)-th message passing step/layer

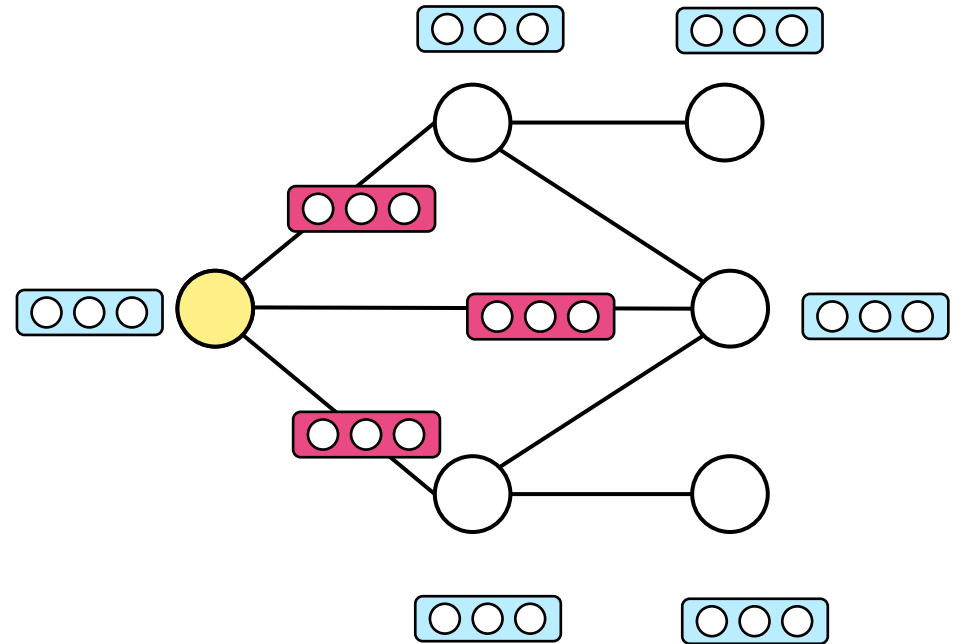


Message Passing in GNNs

$$\mathbf{h}_i^t \quad \mathbf{h}_j^t$$
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$



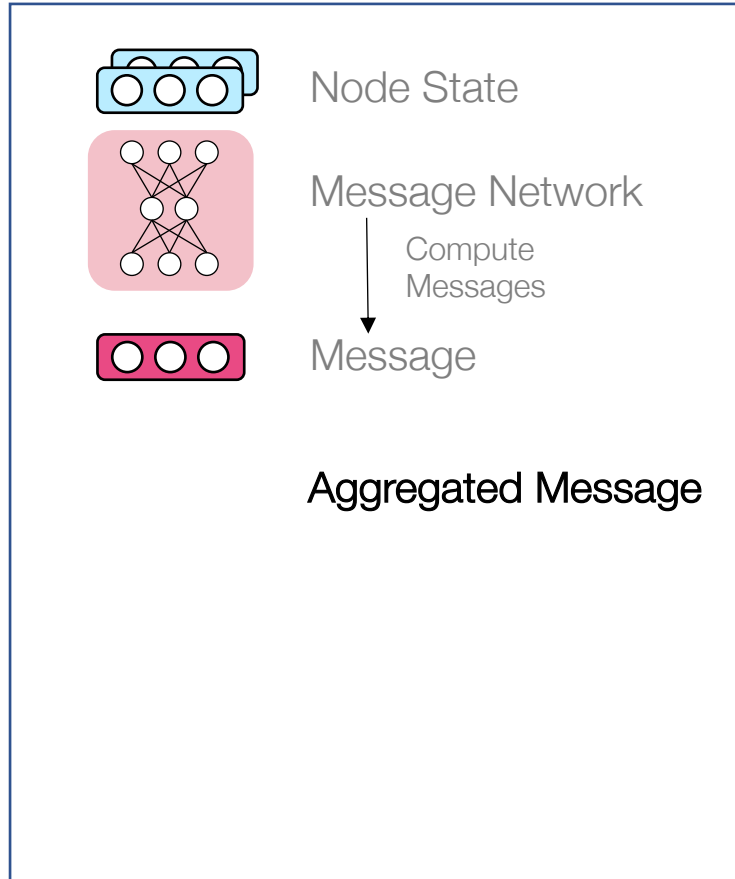
(t+1)-th message passing step/layer



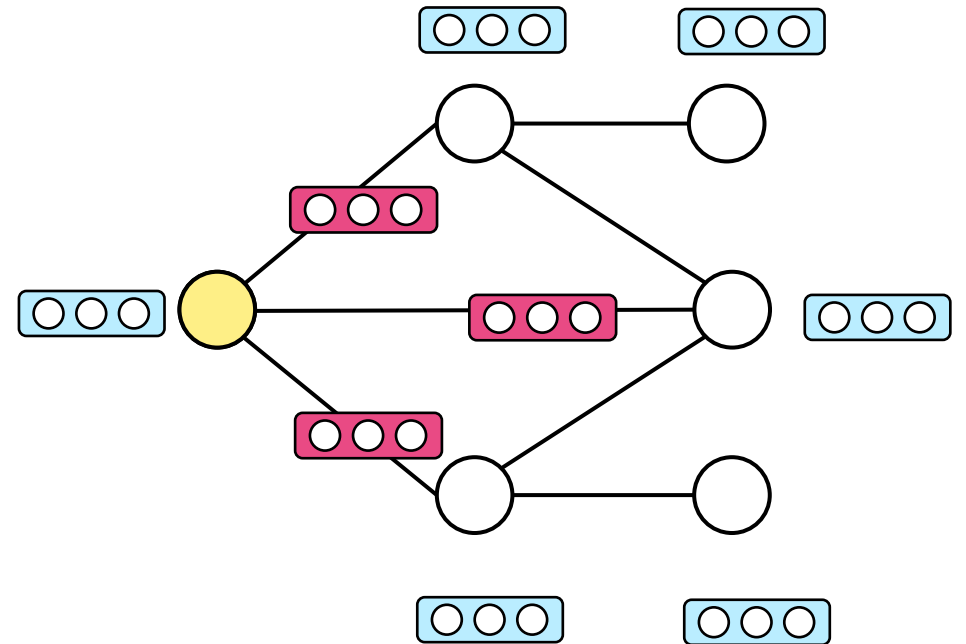
Message Passing in GNNs

\mathbf{h}_i^t \mathbf{h}_j^t

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$



(t+1)-th message passing step/layer

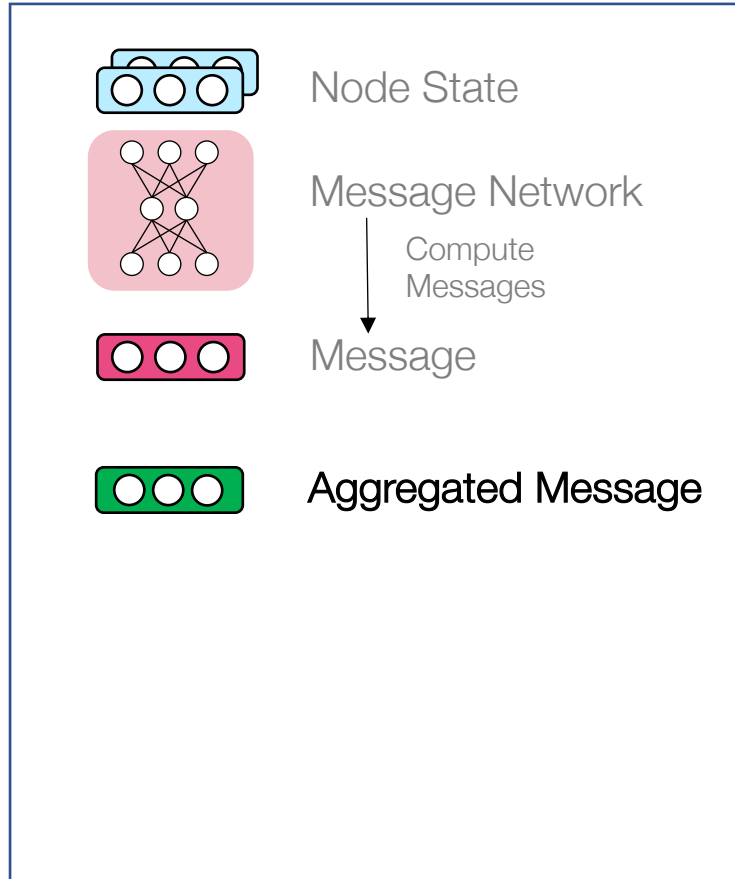


Message Passing in GNNs

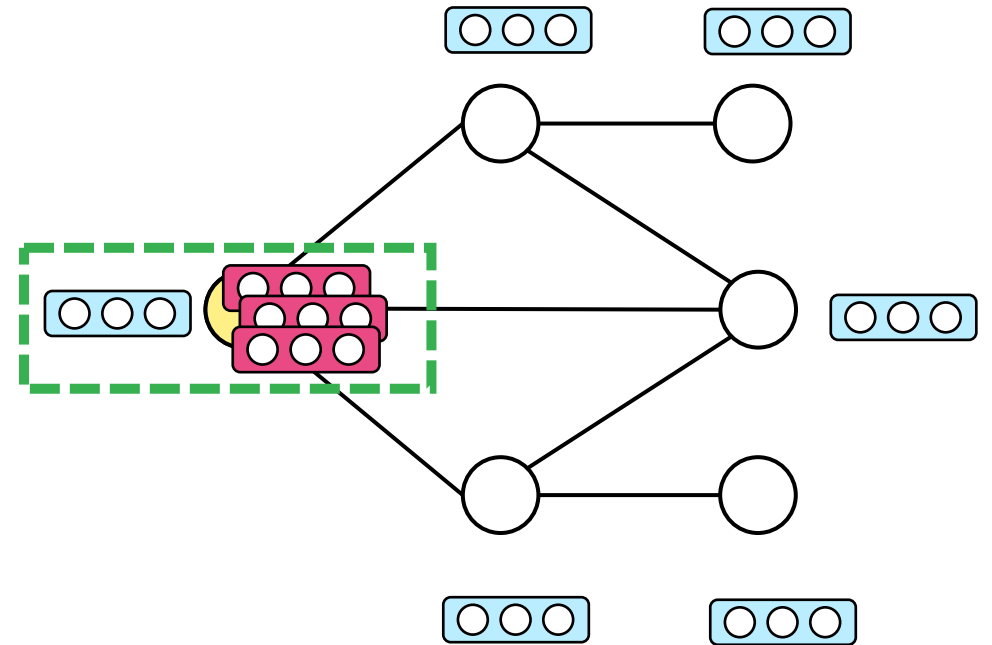
\mathbf{h}_i^t \mathbf{h}_j^t

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$



(t+1)-th message passing step/layer

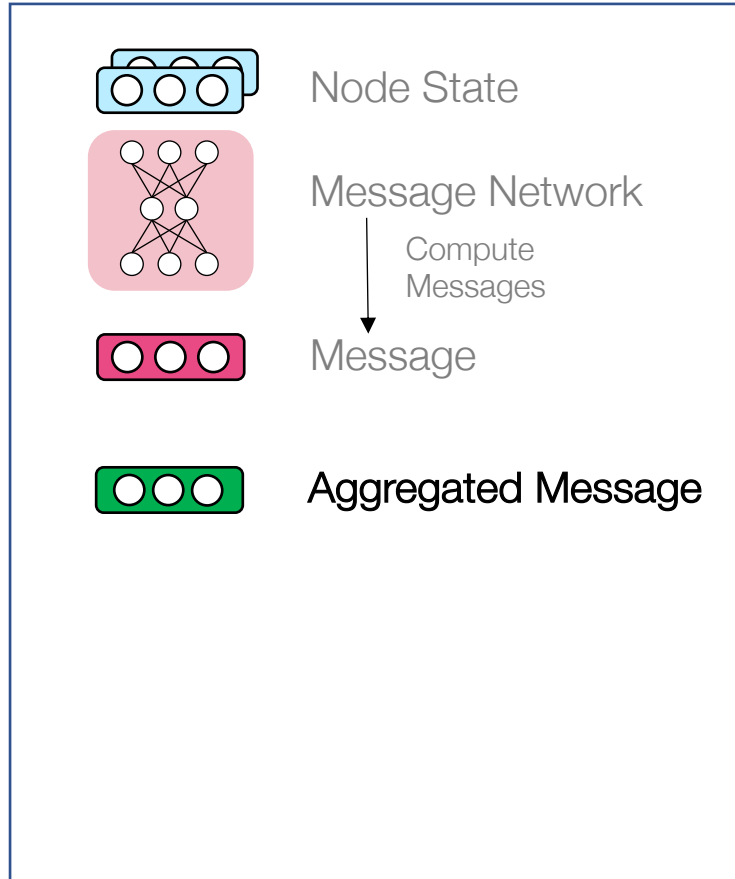


Message Passing in GNNs

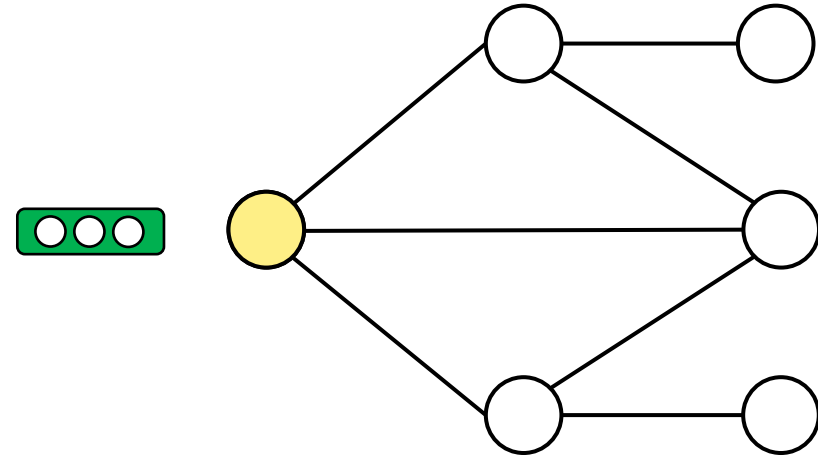
\mathbf{h}_i^t \mathbf{h}_j^t

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$



(t+1)-th message passing step/layer

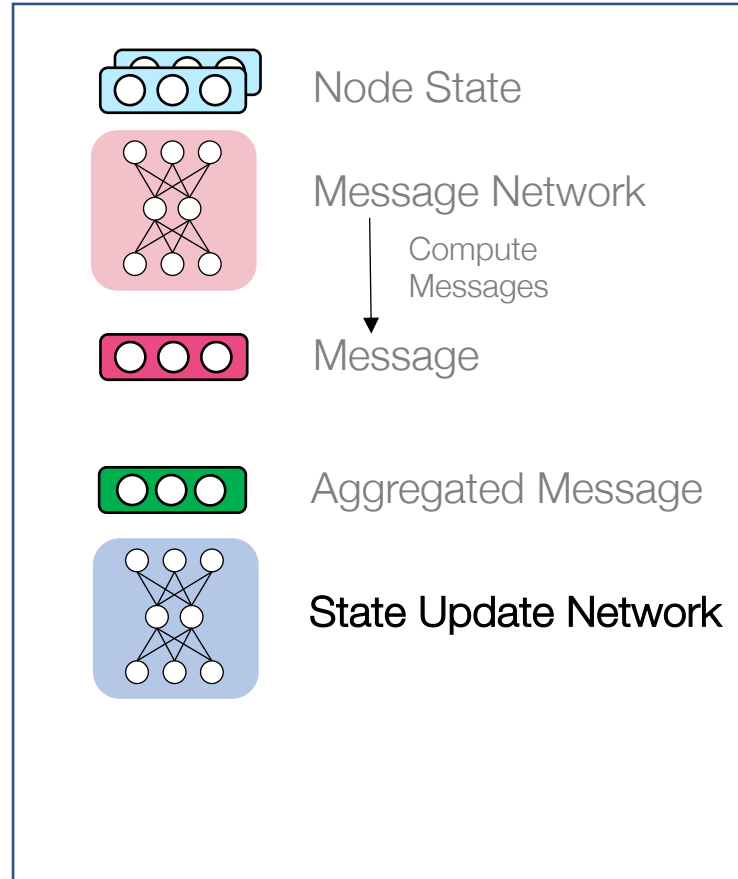


Message Passing in GNNs

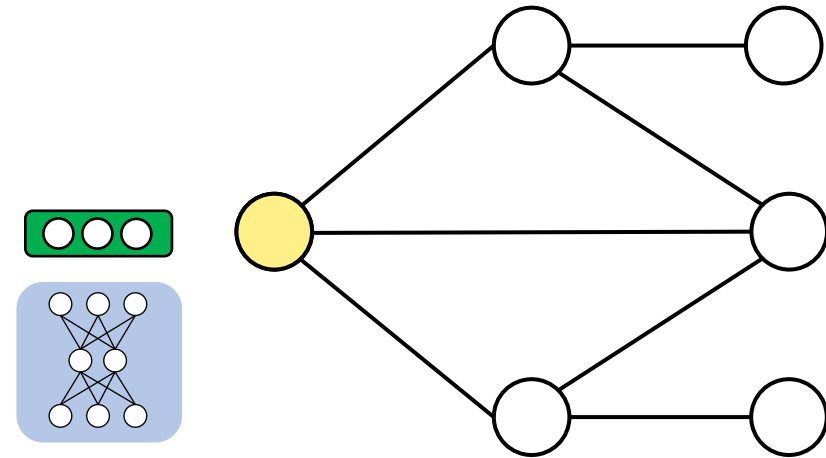
$$\mathbf{h}_i^t \quad \mathbf{h}_j^t$$

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$



(t+1)-th message passing step/layer



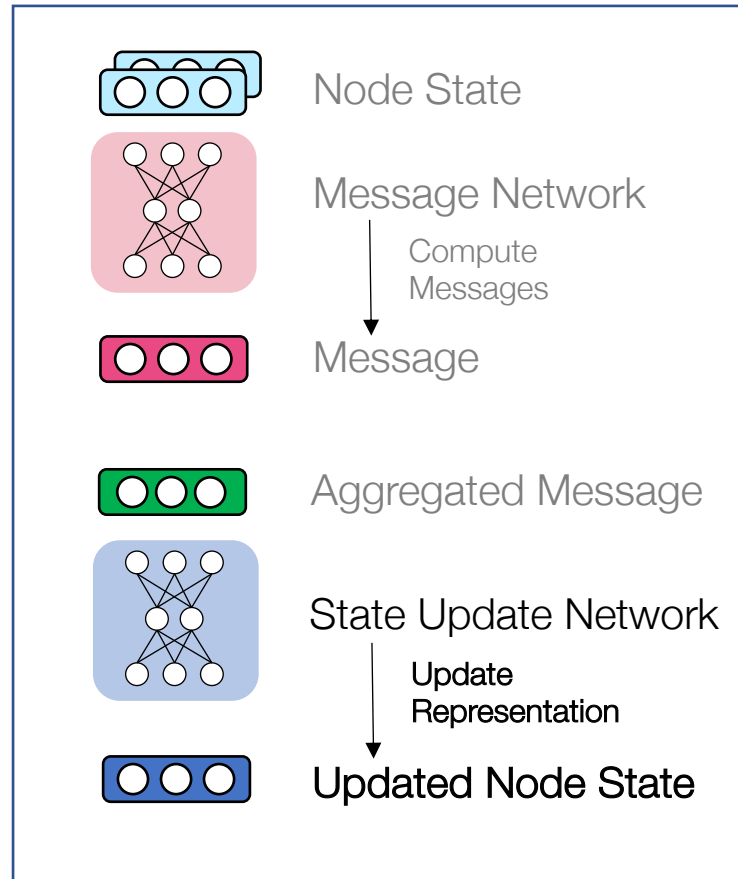
Message Passing in GNNs

$$\mathbf{h}_i^t \quad \mathbf{h}_j^t$$

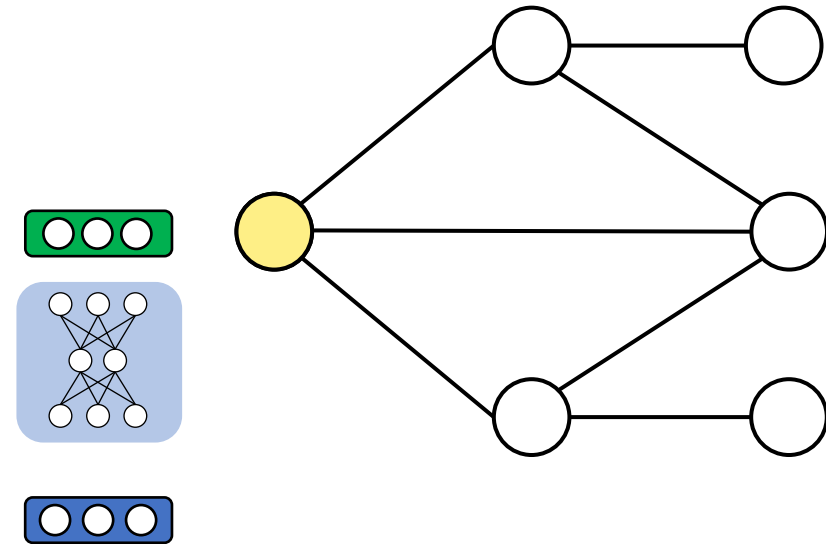
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

$$\mathbf{h}_i^{t+1} = f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$



(t+1)-th message passing step/layer



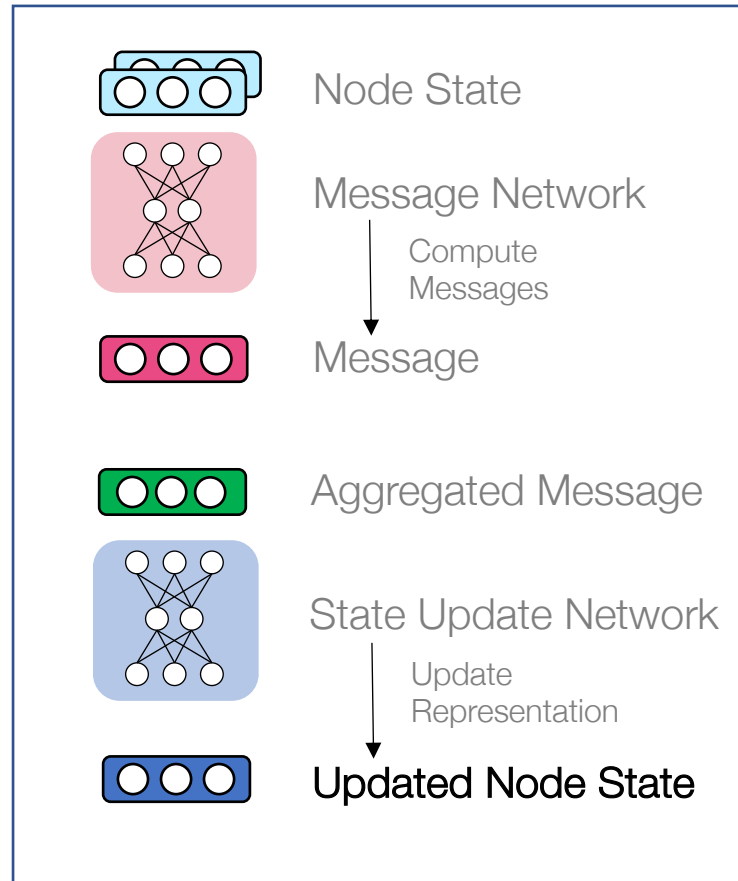
Message Passing in GNNs

$$\mathbf{h}_i^t \quad \mathbf{h}_j^t$$

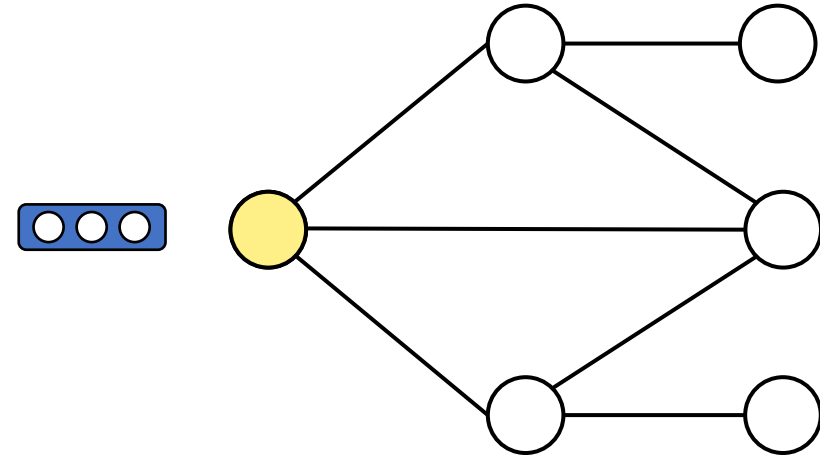
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

$$\mathbf{h}_i^{t+1} = f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$



(t+1)-th message passing step/layer



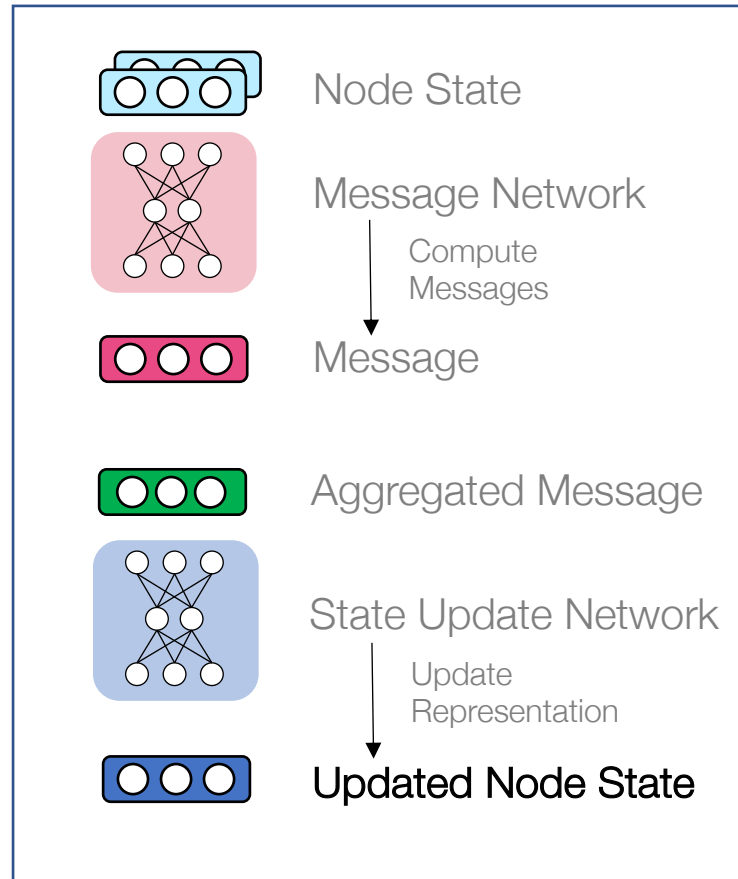
Message Passing in GNNs

$$\mathbf{h}_i^t \quad \mathbf{h}_j^t$$

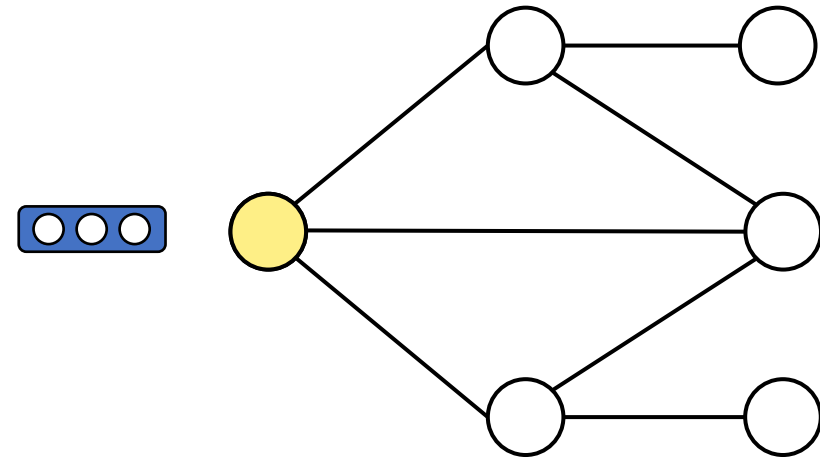
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

$$\mathbf{h}_i^{t+1} = f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$



(t+1)-th message passing step/layer



- **Parallel Schedule!**

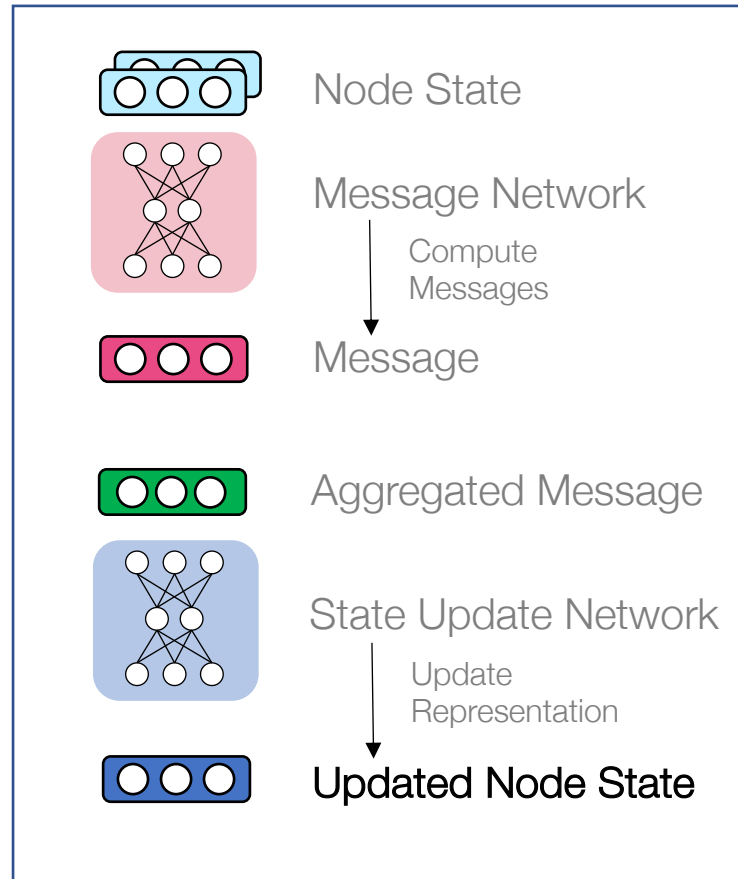
Message Passing in GNNs

$$\mathbf{h}_i^t \quad \mathbf{h}_j^t$$

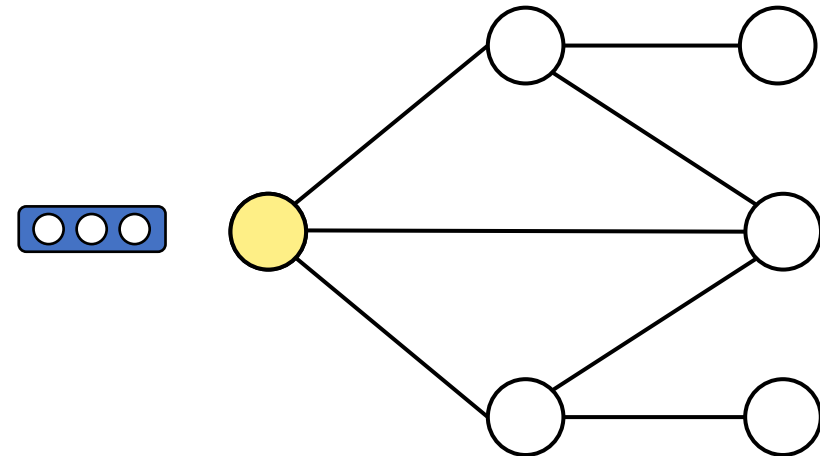
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

$$\mathbf{h}_i^{t+1} = f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$



(t+1)-th message passing step/layer



- **Parallel Schedule!**
- **Other schedules [8] are possible and could improve performance in certain tasks!**

Message Passing in GNNs

Instantiations:

1. Compute Messages

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

2. Aggregate Messages

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

3. Update Node Representations

$$\mathbf{h}_i^{t+1} = f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$

Message Passing in GNNs

Instantiations:

1. **Compute Messages**

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t]) \quad [4]$$

Message Passing in GNNs

Instantiations:

1. Compute Messages

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t]) \quad [4]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \mathbf{h}_j^t \quad [5]$$

Message Passing in GNNs

Instantiations:

1. Compute Messages

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t]) \quad [4]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \mathbf{h}_j^t \quad [5]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}]) \quad [4]$$

Message Passing in GNNs

Instantiations:

1. Compute Messages

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t]) \quad [4]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \mathbf{h}_j^t \quad [5]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}]) \quad [4]$$

Edge Feature

Message Passing in GNNs

Instantiations:

1. Compute Messages

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t]) \quad [4]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \mathbf{h}_j^t \quad [5]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}]) \quad [4]$$

Edge Feature

2. Aggregate Messages

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [4,5,7]$$

Message Passing in GNNs

Instantiations:

1. Compute Messages

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t]) \quad [4]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \mathbf{h}_j^t \quad [5]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}]) \quad [4]$$

Edge Feature

2. Aggregate Messages

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [4,5,7]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [6]$$

Message Passing in GNNs

Instantiations:

1. Compute Messages

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t]) \quad [4]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \mathbf{h}_j^t \quad [5]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}]) \quad [4]$$

Edge Feature

2. Aggregate Messages

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [4,5,7]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [6]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \max_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [6]$$

Message Passing in GNNs

Instantiations:

1. Compute Messages

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t]) \quad [4]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \mathbf{h}_j^t \quad [5]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}]) \quad [4]$$

Edge Feature

2. Aggregate Messages

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [4,5,7]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [6]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \max_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [6]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \text{LSTM}([\mathbf{m}_{ji}^t | j \in \mathcal{N}_i]) \quad [6]$$

Message Passing in GNNs

Instantiations:

1. Compute Messages

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t]) \quad [4]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \mathbf{h}_j^t \quad [5]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}]) \quad [4]$$

Edge Feature

2. Aggregate Messages

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [4,5,7]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [6]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \max_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [6]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \text{LSTM}([\mathbf{m}_{ji}^t | j \in \mathcal{N}_i]) \quad [6]$$

3. Update Node Representations

$$\mathbf{h}_i^{t+1} = f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$

$$f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t) = \text{GRU}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t) \quad [4,7]$$

Message Passing in GNNs

Instantiations:

1. Compute Messages

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t]) \quad [4]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \mathbf{h}_j^t \quad [5]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}]) \quad [4]$$

Edge Feature

2. Aggregate Messages

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [4,5,7]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [6]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \max_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [6]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \text{LSTM}([\mathbf{m}_{ji}^t | j \in \mathcal{N}_i]) \quad [6]$$

3. Update Node Representations

$$\mathbf{h}_i^{t+1} = f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$

$$f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t) = \text{GRU}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t) \quad [4,7]$$

$$f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t) = \text{MLP}_1(\mathbf{h}_i^t) + \text{MLP}_2(\bar{\mathbf{m}}_i^t) \quad [5]$$

Message Passing in GNNs

Instantiations:

1. Compute Messages

$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t]) \quad [4]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) = \mathbf{h}_j^t \quad [5]$$

$$f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}) = \text{MLP}([\mathbf{h}_j^t, \mathbf{h}_i^t, \mathbf{e}_{ji}]) \quad [4]$$

Edge Feature

2. Aggregate Messages

$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\})$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [4,5,7]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [6]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \max_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t \quad [6]$$

$$f_{\text{agg}}(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) = \text{LSTM}([\mathbf{m}_{ji}^t | j \in \mathcal{N}_i]) \quad [6]$$

3. Update Node Representations

$$\mathbf{h}_i^{t+1} = f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t)$$

$$f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t) = \text{GRU}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t) \quad [4,7]$$

$$f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t) = \text{MLP}_1(\mathbf{h}_i^t) + \text{MLP}_2(\bar{\mathbf{m}}_i^t) \quad [5]$$

$$f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t) = \text{MLP}([\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t]) \quad [6]$$

Readout in GNNs

Instantiations:

1. Node Readout

$$\mathbf{y}_i = f_{\text{readout}}(\mathbf{h}_i^T)$$

2. Edge Readout

$$\mathbf{y}_{ij} = f_{\text{readout}}(\mathbf{h}_i^T, \mathbf{h}_j^T)$$

3. Graph Readout

$$\mathbf{y} = f_{\text{readout}}(\{\mathbf{h}_i^T\})$$

Readout in GNNs

Instantiations:

1. Node Readout

$$\mathbf{y}_i = f_{\text{readout}}(\mathbf{h}_i^T)$$

$$f_{\text{readout}}(\mathbf{h}_i^T) = \text{MLP}(\mathbf{h}_i^T)$$

Readout in GNNs

Instantiations:

1. Node Readout

$$\mathbf{y}_i = f_{\text{readout}}(\mathbf{h}_i^T)$$

$$f_{\text{readout}}(\mathbf{h}_i^T) = \text{MLP}(\mathbf{h}_i^T)$$

2. Edge Readout

$$\mathbf{y}_{ij} = f_{\text{readout}}(\mathbf{h}_i^T, \mathbf{h}_j^T)$$

$$f_{\text{readout}}(\mathbf{h}_i^T, \mathbf{h}_j^T) = \text{MLP}([\mathbf{h}_i^T, \mathbf{h}_j^T])$$

$$f_{\text{readout}}(\mathbf{h}_i^T, \mathbf{h}_j^T, e_{ij}) = \text{MLP}([\mathbf{h}_i^T, \mathbf{h}_j^T, e_{ij}])$$

Edge Feature

Readout in GNNs

Instantiations:

1. Node Readout

$$\mathbf{y}_i = f_{\text{readout}}(\mathbf{h}_i^T)$$

$$f_{\text{readout}}(\mathbf{h}_i^T) = \text{MLP}(\mathbf{h}_i^T)$$

2. Edge Readout

$$\mathbf{y}_{ij} = f_{\text{readout}}(\mathbf{h}_i^T, \mathbf{h}_j^T)$$

$$f_{\text{readout}}(\mathbf{h}_i^T, \mathbf{h}_j^T) = \text{MLP}([\mathbf{h}_i^T, \mathbf{h}_j^T])$$

$$f_{\text{readout}}(\mathbf{h}_i^T, \mathbf{h}_j^T, e_{ij}) = \text{MLP}([\mathbf{h}_i^T, \mathbf{h}_j^T, e_{ij}])$$

Edge Feature

3. Graph Readout

$$\mathbf{y} = f_{\text{readout}}(\{\mathbf{h}_i^T\})$$

$$f_{\text{readout}}(\{\mathbf{h}_i^T\}) = \sum_i \sigma(\text{MLP}_1(\mathbf{h}_i^T)) \text{MLP}_2(\mathbf{h}_i^T)$$

$$f_{\text{readout}}(\{\mathbf{h}_i^T\}, \mathbf{g}) = \sum_i \sigma(\text{MLP}_1(\mathbf{h}_i^T, \mathbf{g})) \text{MLP}_2(\mathbf{h}_i^T, \mathbf{g})$$

Graph Feature

Implementations

1. *Although graph could be very sparse, we should maximally exploit dense operators since they are efficient on GPUs!*
2. *Parallel message passing is very GPU friendly!*

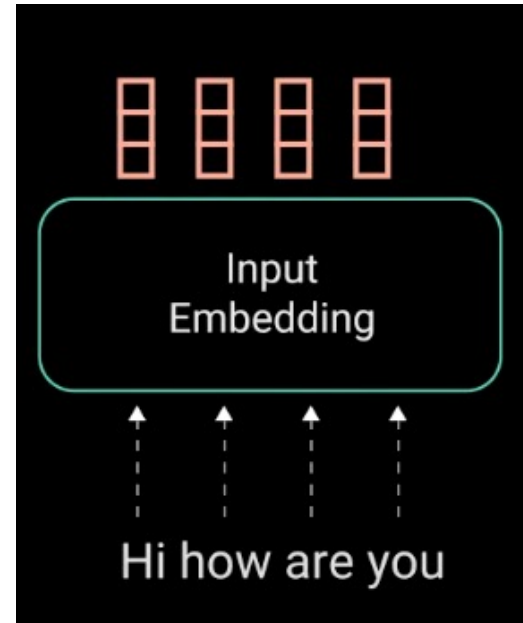
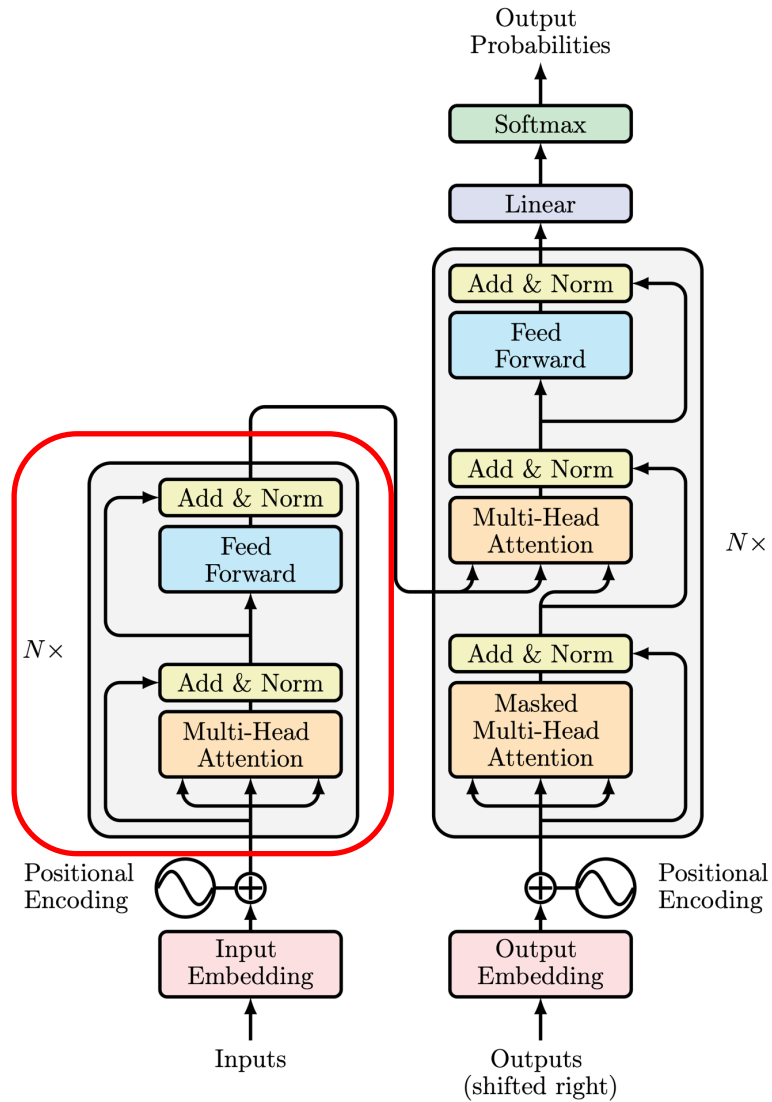
Implementations

1. *Although graph could be very sparse, we should maximally exploit dense operators since they are efficient on GPUs!*
2. *Parallel message passing is very GPU friendly!*

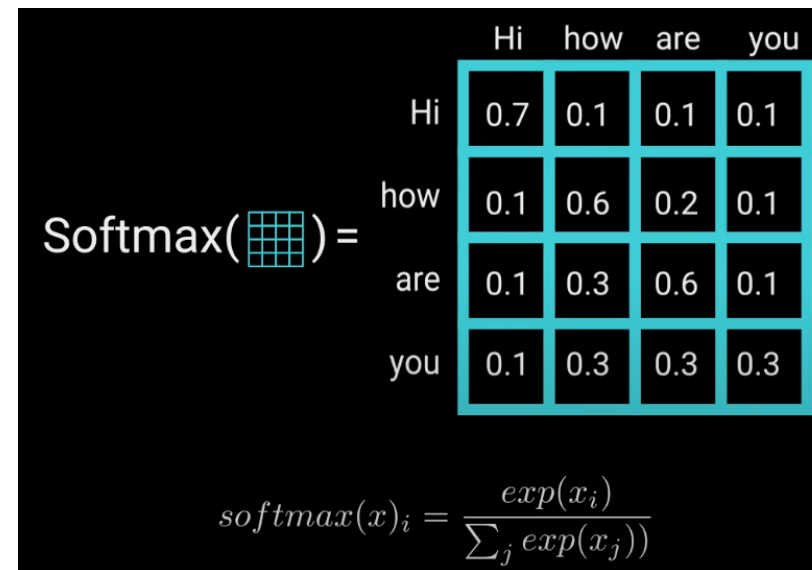
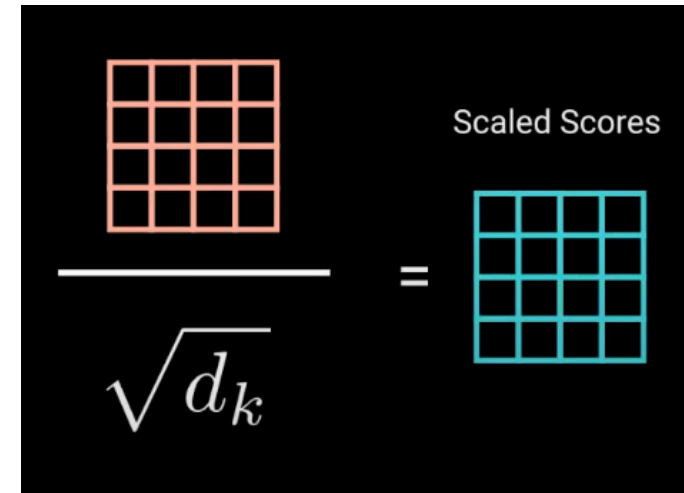
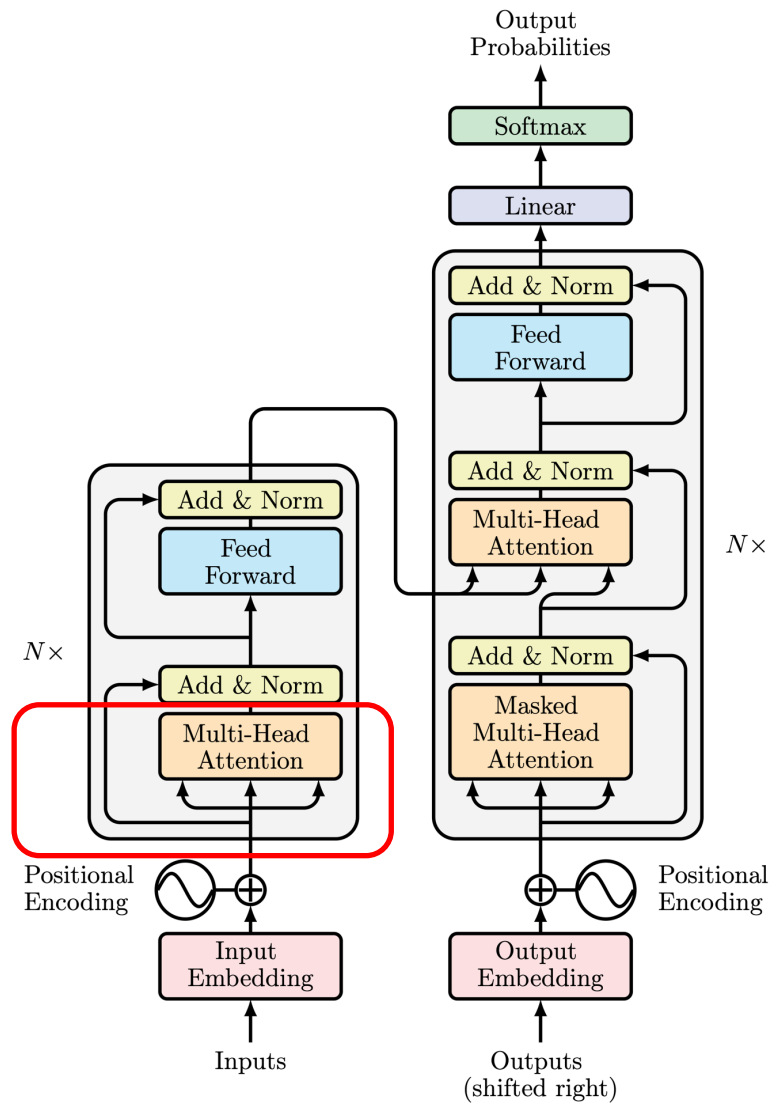
Tips:

- Use adjacency list representation
- Compute messages for all edges in parallel
- Compute aggregations for all nodes in parallel
- Compute updates for all nodes in parallel

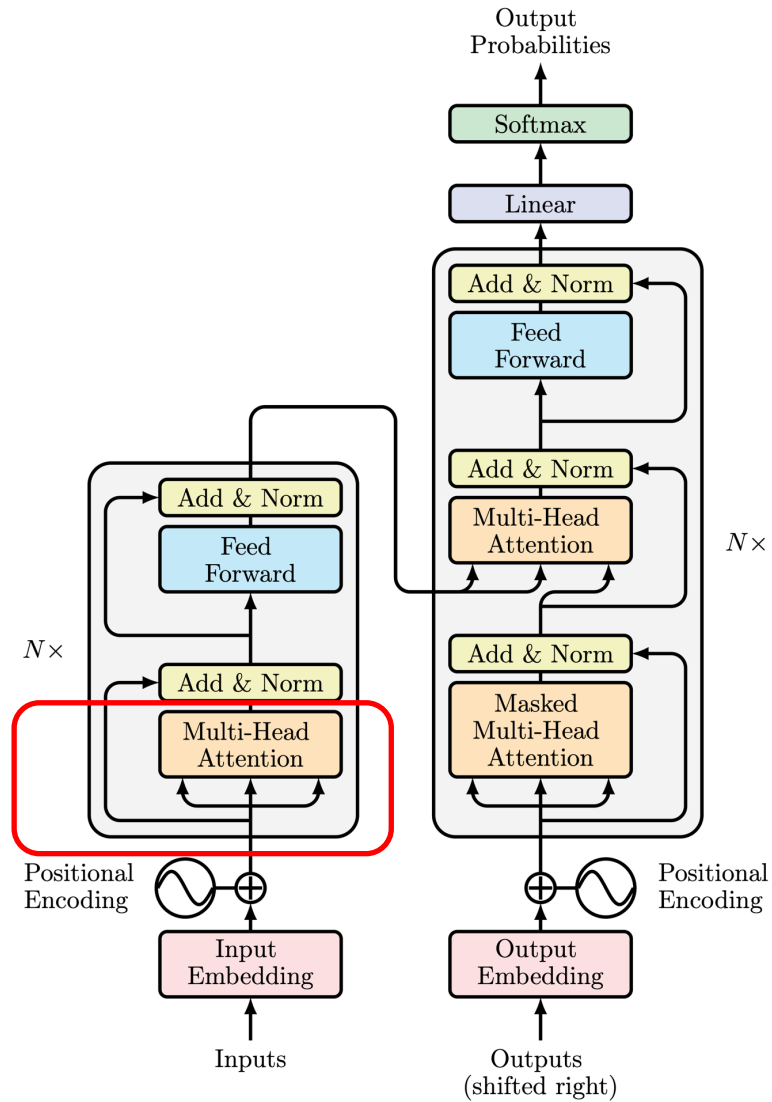
Relationships with Transformer



Relationships with Transformer



Relationships with Transformer



- Attention can be viewed as the weighted adjacency matrix of a fully connected graph!

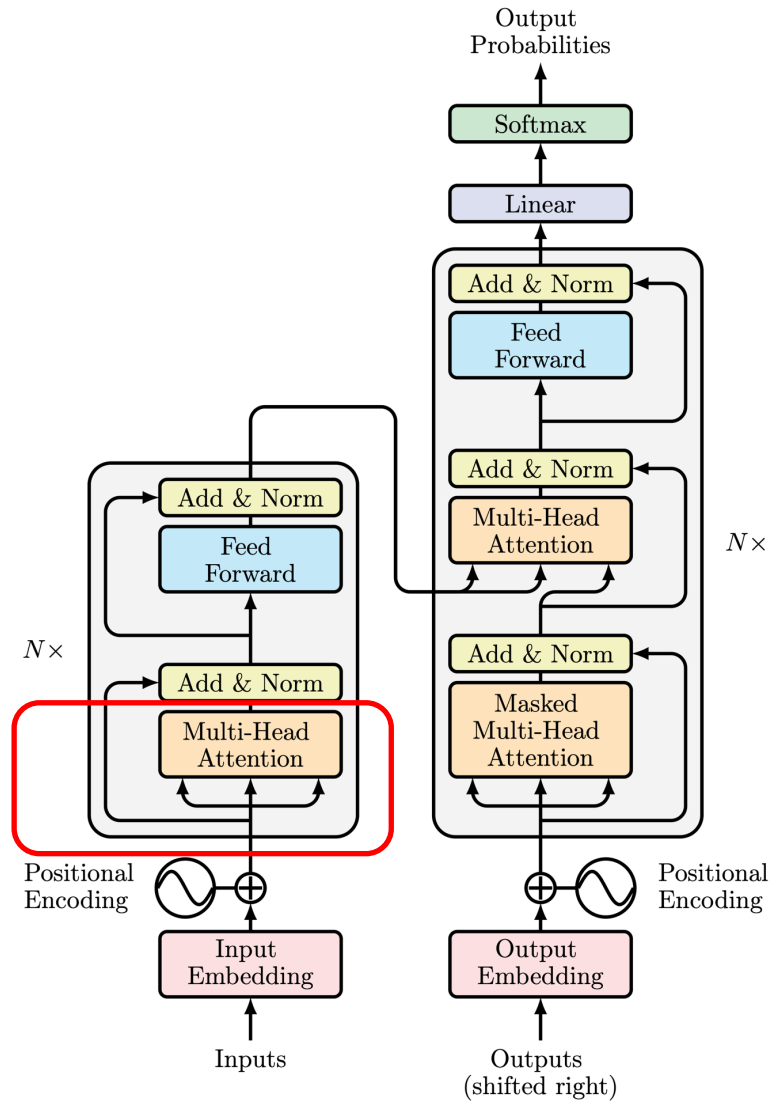
Softmax(

	Hi	how	are	you
Hi	0.7	0.1	0.1	0.1
how	0.1	0.6	0.2	0.1
are	0.1	0.3	0.6	0.1
you	0.1	0.3	0.3	0.3

) =

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Relationships with Transformer



- Attention can be viewed as the weighted adjacency matrix of a fully connected graph!
- Transformers (esp. encoder) can be viewed as GNNs applied to fully connected graphs!

Softmax(

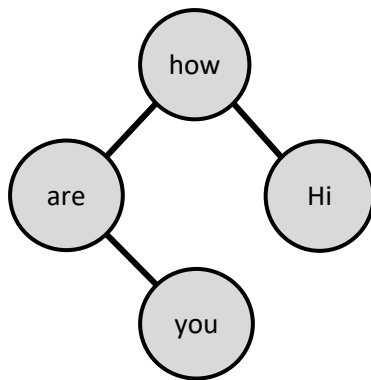
	Hi	how	are	you
Hi	0.7	0.1	0.1	0.1
how	0.1	0.6	0.2	0.1
are	0.1	0.3	0.6	0.1
you	0.1	0.3	0.3	0.3

) =

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Encode Graph Structures in Transformers

- Apply the adjacency matrix as a mask to the attention and renormalize it, like Graph Attention Networks (GAT) [10]
- Encode connectivities/distances as bias of the attention [11]



	Hi	how	are	you
Hi	0	1	0	1
how	1	0	0	0
are	0	0	0	1
you	1	0	1	0

$\text{Softmax}(\begin{matrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{matrix}) =$

	Hi	how	are	you
Hi	0.7	0.1	0.1	0.1
how	0.1	0.6	0.2	0.1
are	0.1	0.3	0.6	0.1
you	0.1	0.3	0.3	0.3

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

References

- [1] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M. and Monfardini, G., 2008. The graph neural network model. *IEEE transactions on neural networks*, 20(1), pp.61-80.
- [2] Goller, C. and Kuchler, A., 1996, June. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96)* (Vol. 1, pp. 347-352). IEEE.
- [3] Ackley, D.H., Hinton, G.E. and Sejnowski, T.J., 1985. A learning algorithm for Boltzmann machines. *Cognitive science*, 9(1), pp.147-169.
- [4] Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O. and Dahl, G.E., 2017, July. Neural message passing for quantum chemistry. In *International conference on machine learning* (pp. 1263-1272). PMLR.
- [5] Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G. and Grohe, M., 2019, July. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 4602-4609).
- [6] Hamilton, W.L., Ying, R. and Leskovec, J., 2017, December. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 1025-1035).
- [7] Li, Y., Tarlow, D., Brockschmidt, M. and Zemel, R., 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.
- [8] Liao, R., Brockschmidt, M., Tarlow, D., Gaunt, A.L., Urtasun, R. and Zemel, R., 2018. Graph partition neural networks for semi-supervised classification. *arXiv preprint arXiv:1803.06272*.

References

- [9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).
- [10] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. and Bengio, Y., 2017. Graph attention networks. arXiv preprint arXiv:1710.10903.
- [11] Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y. and Liu, T.Y., 2021. Do Transformers Really Perform Bad for Graph Representation?. arXiv preprint arXiv:2106.05234.

Questions?