



Topics in AI (CPSC 532S): Multimodal Learning with Vision, Language and Sound



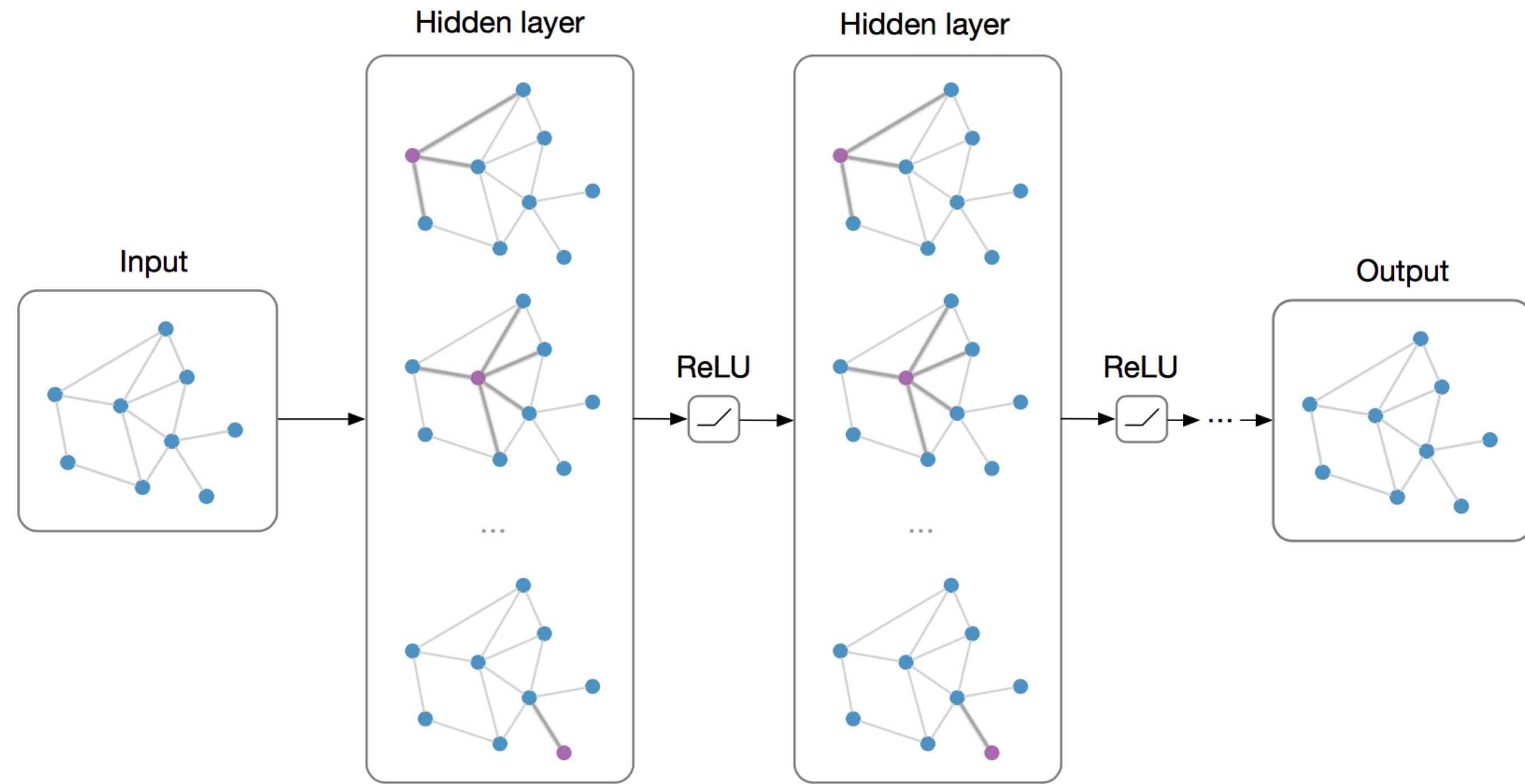
A decorative horizontal bar at the bottom of the slide, composed of four colored segments: light green, teal, light blue, and light purple.

Lecture 20: Graph Neural Networks (cont)

Logistics

Relatively **short** lecture today ... RL next class

Graph Neural Networks (GNNs)



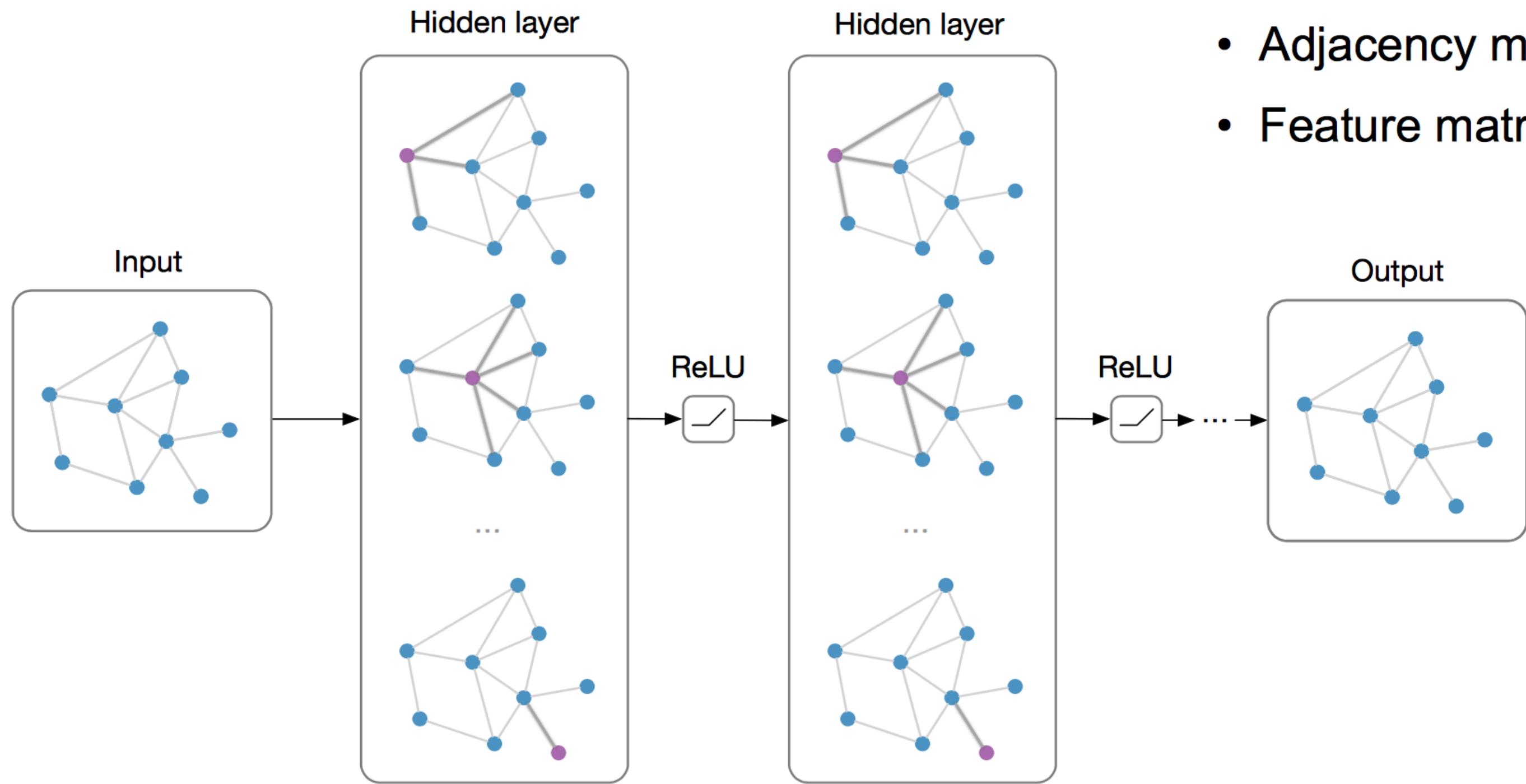
Main Idea: Pass messages between pairs of nodes and agglomerate

Alternative Interpretation: Pass messages between nodes to refine node (and possibly edge) representations

Graph Neural Networks (GNNs)

Notation: $\mathcal{G} = (\mathbf{A}, \mathbf{X})$

- Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$
- Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$

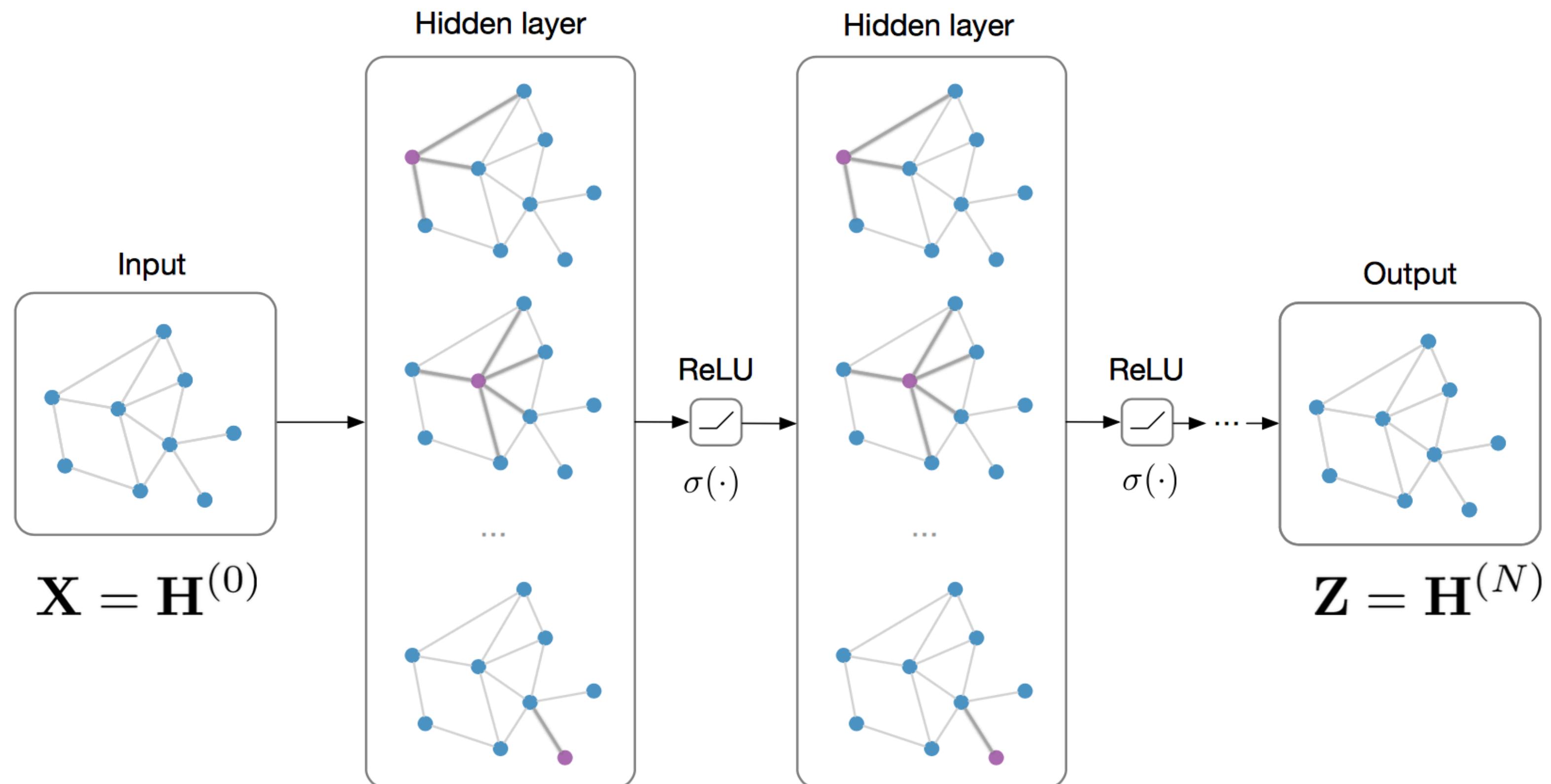


Main Idea: Pass messages between pairs of nodes and agglomerate

Alternative Interpretation: Pass messages between nodes to refine node (and possibly edge) representations

Classification and Link Prediction with GNNs / GCNs

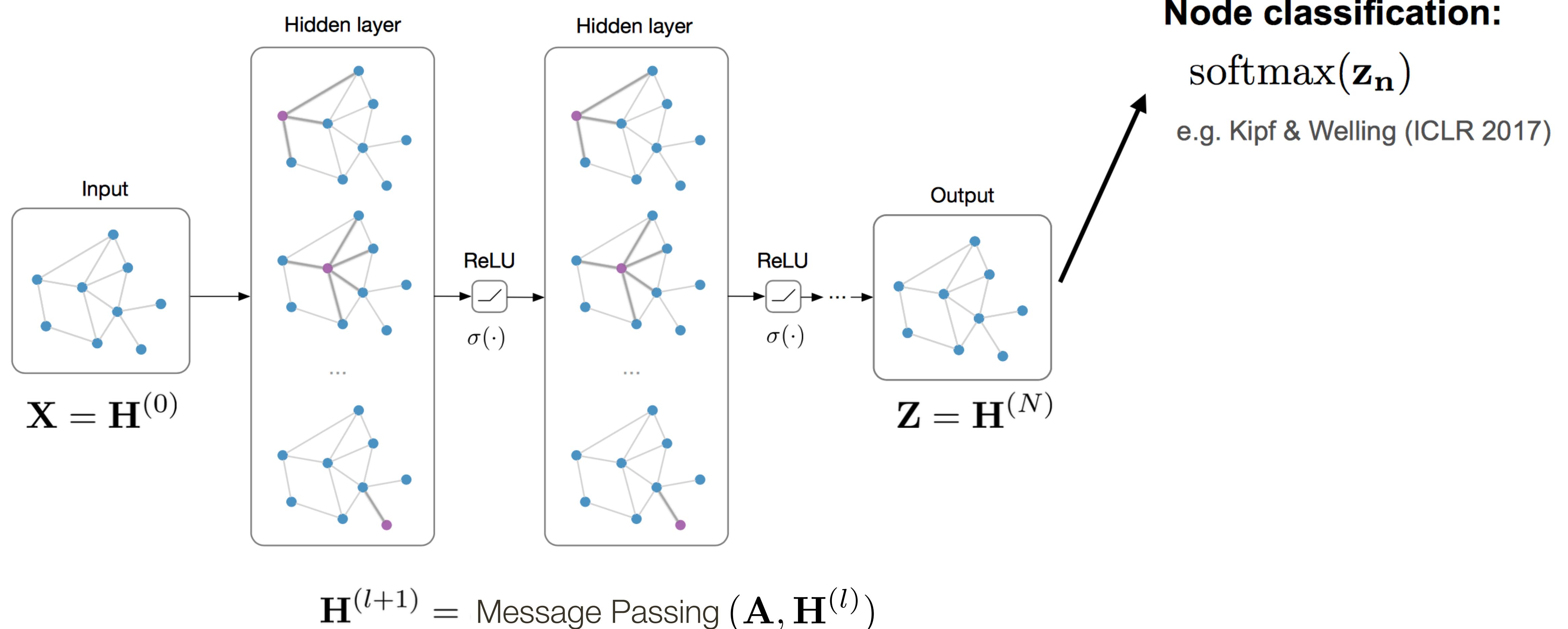
Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



$$\mathbf{H}^{(l+1)} = \text{Message Passing} (\mathbf{A}, \mathbf{H}^{(l)})$$

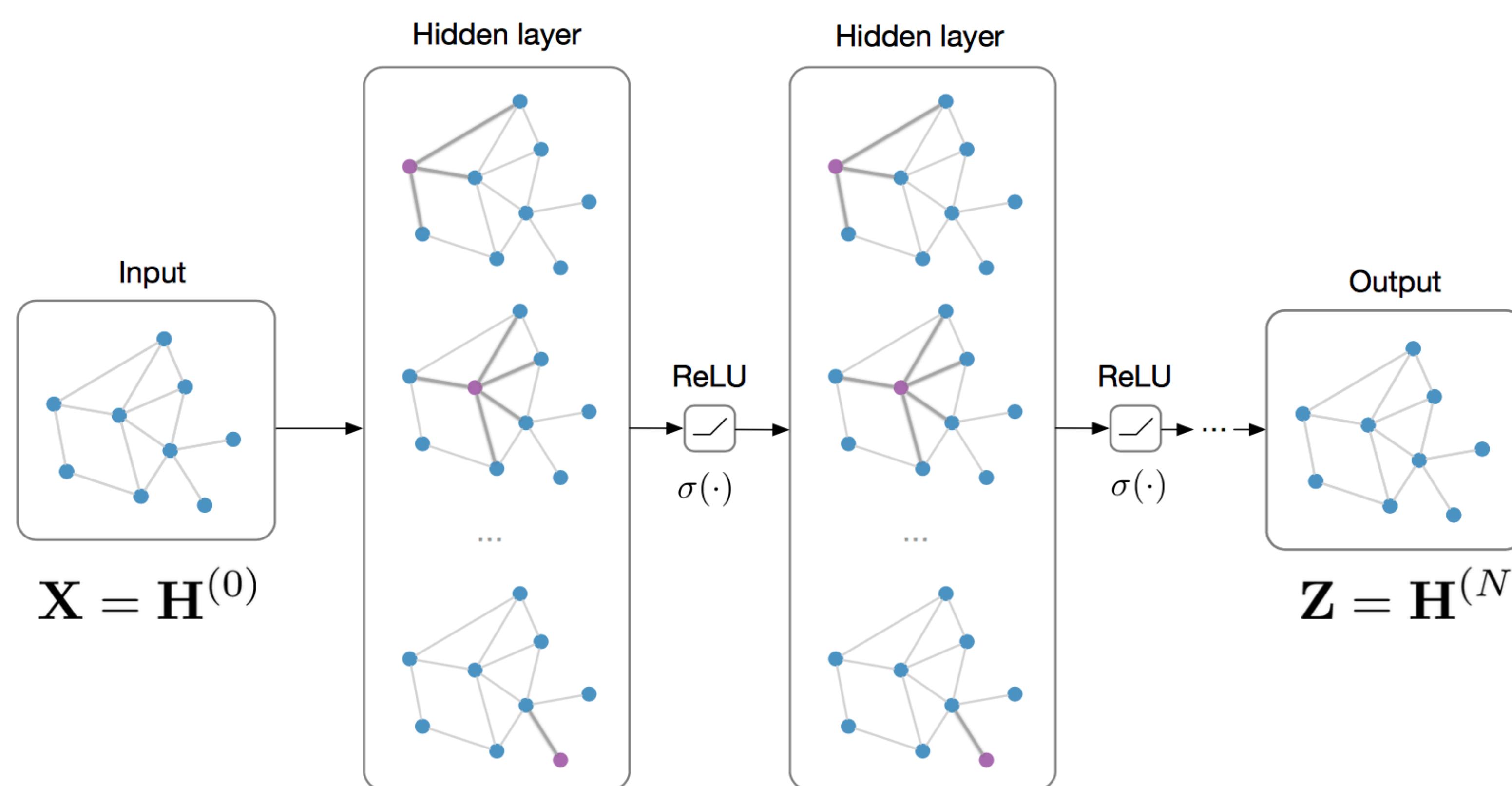
Classification and Link Prediction with GNNs / GCNs

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



Classification and Link Prediction with GNNs / GCNs

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



$$\mathbf{H}^{(l+1)} = \text{Message Passing} (\mathbf{A}, \mathbf{H}^{(l)})$$

Node classification:

$$\text{softmax}(\mathbf{z}_n)$$

e.g. Kipf & Welling (ICLR 2017)

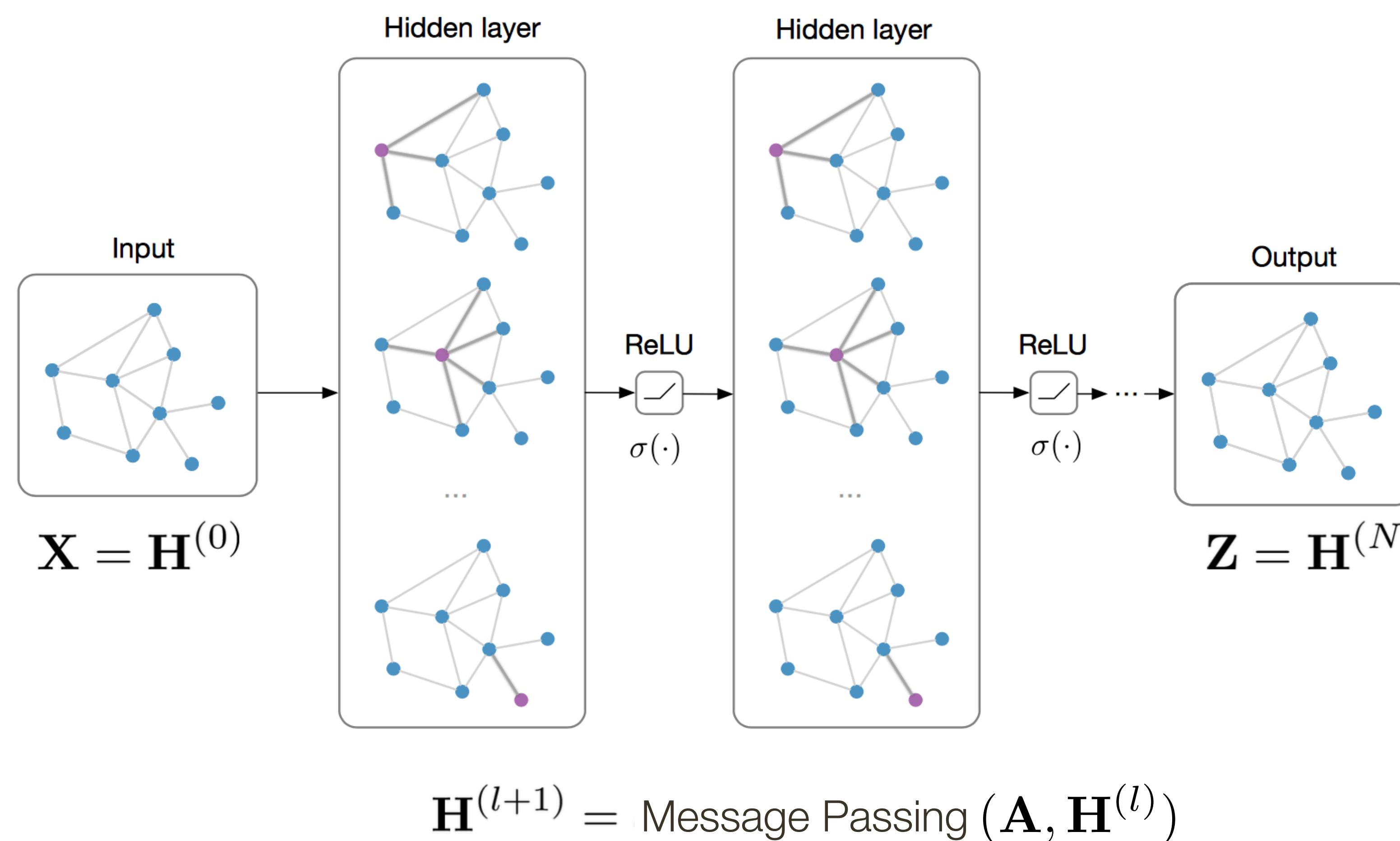
Graph classification:

$$\text{softmax}(\sum_n \mathbf{z}_n)$$

e.g. Duvenaud et al. (NIPS 2015)

Classification and Link Prediction with GNNs / GCNs

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



Node classification:

$$\text{softmax}(\mathbf{z}_n)$$

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

$$\text{softmax}(\sum_n \mathbf{z}_n)$$

e.g. Duvenaud et al. (NIPS 2015)

Link prediction:

$$p(A_{ij}) = \sigma(\mathbf{z}_i^T \mathbf{z}_j)$$

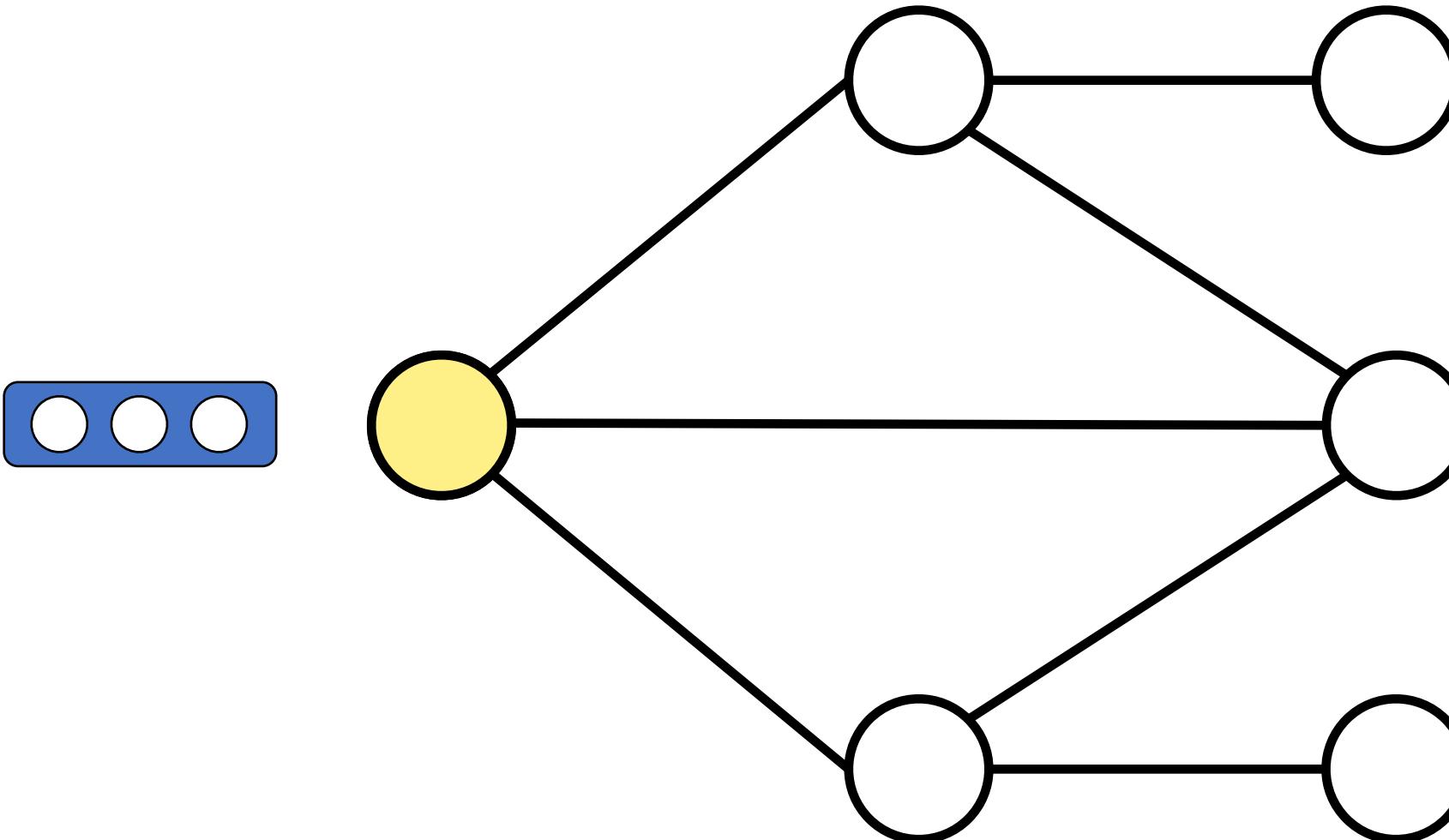
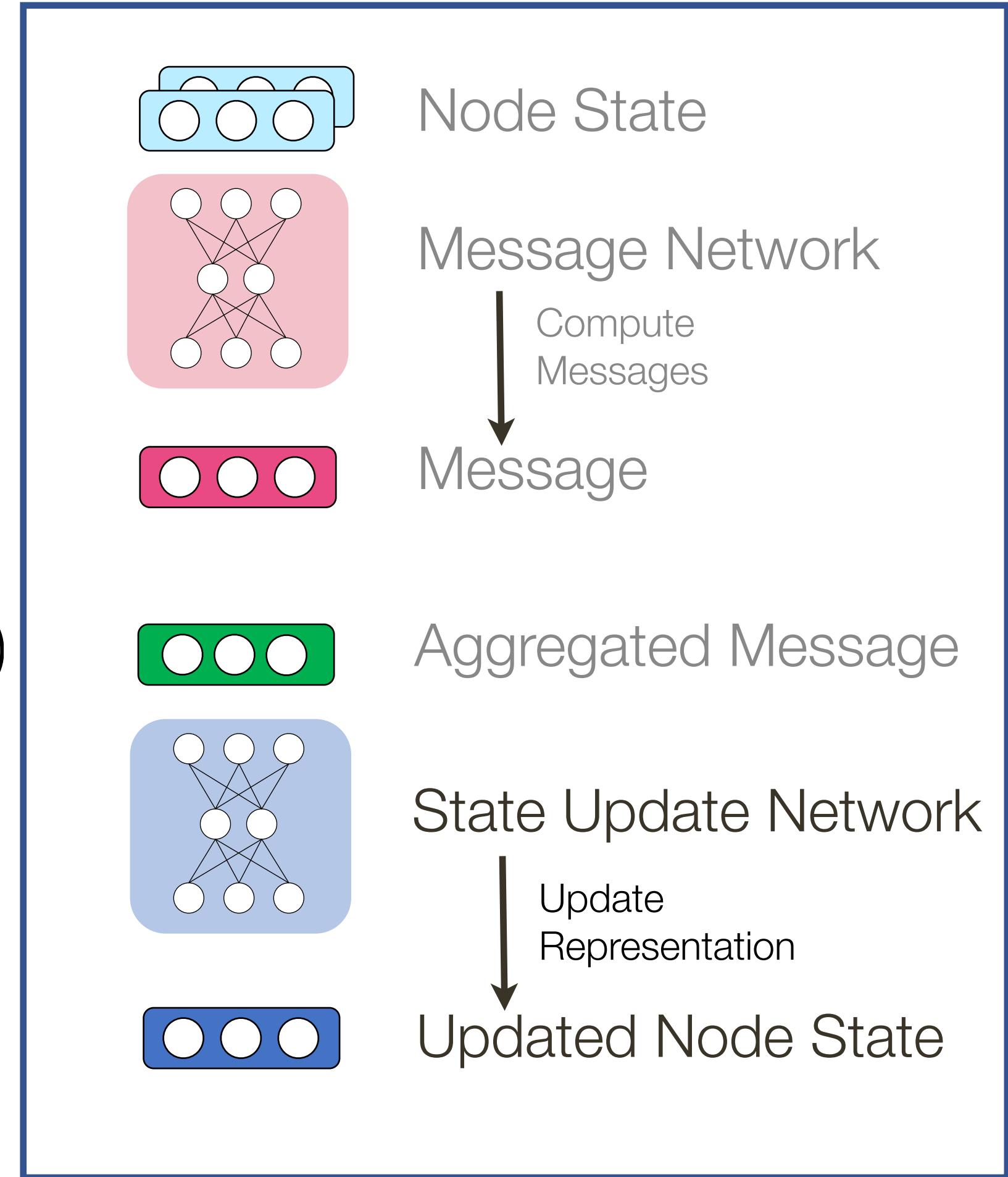
Kipf & Welling (NIPS BDL 2016)

“Graph Auto-Encoders”

Message Passing in GNNs

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

$$\begin{aligned} \mathbf{h}_i^t & \quad \mathbf{h}_j^t \\ \mathbf{m}_{ji}^t &= f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t) \\ \bar{\mathbf{m}}_i^t &= f_{\text{agg}} (\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}) \\ \mathbf{h}_i^{t+1} &= f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t) \end{aligned}$$



Note: We can do all updates in parallel! (but can also be serial)

GNN 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

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

GNN Instantiations

1. Compute Messages

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

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

GNN Instantiations

1. Compute Messages

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

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

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

GNN 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

GNN 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

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

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

GNN 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

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

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

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

GNN 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

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

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

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

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

GNN 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

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

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

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

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

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

GNN 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

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

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

GNN 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

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

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

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

GNN 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

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

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

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

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

GNN Readout

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

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

GNN Readout

1. Node Readout

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

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

GNN Readout

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

GNN Readout

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

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

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

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

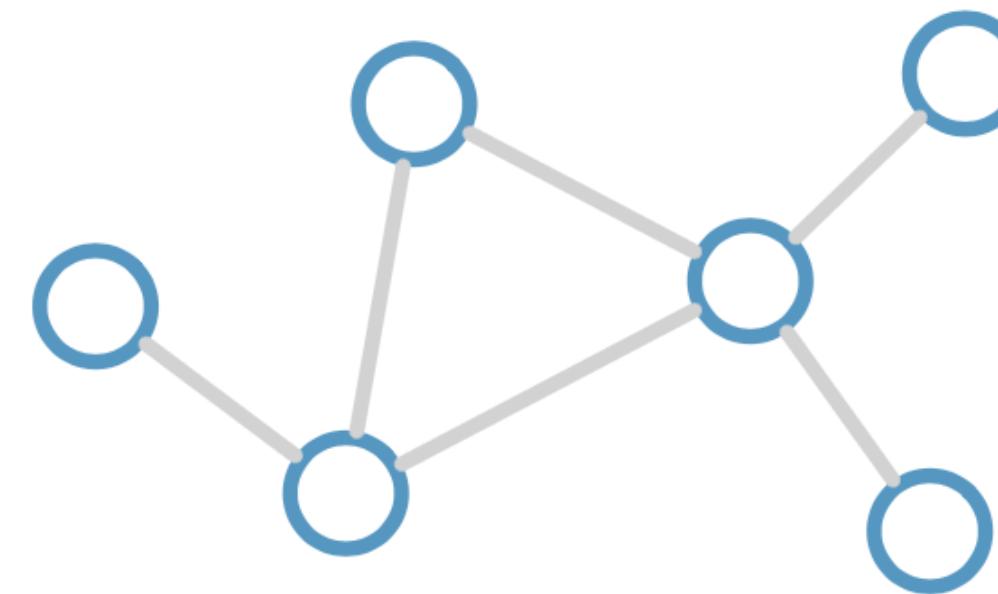
$$f_{\text{readout}}(\{\mathbf{h}_i^T\}, \mathbf{g}) = \sum_i \sigma(\text{MLP}_1(\mathbf{h}_i^T, \mathbf{g})) \text{MLP}_2(\mathbf{h}_i^T, \mathbf{g})$$

Graph Feature

Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

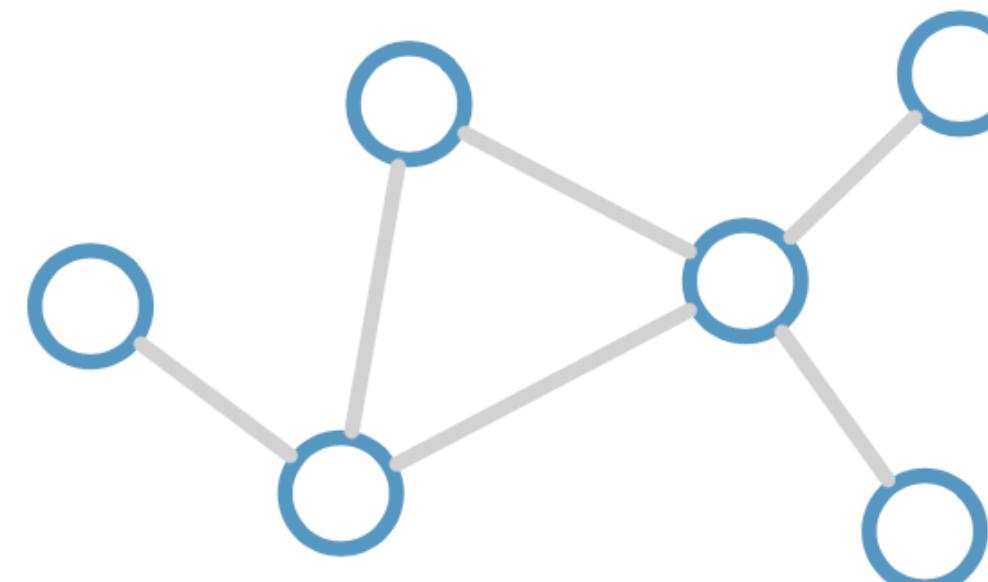
Consider this
undirected graph:



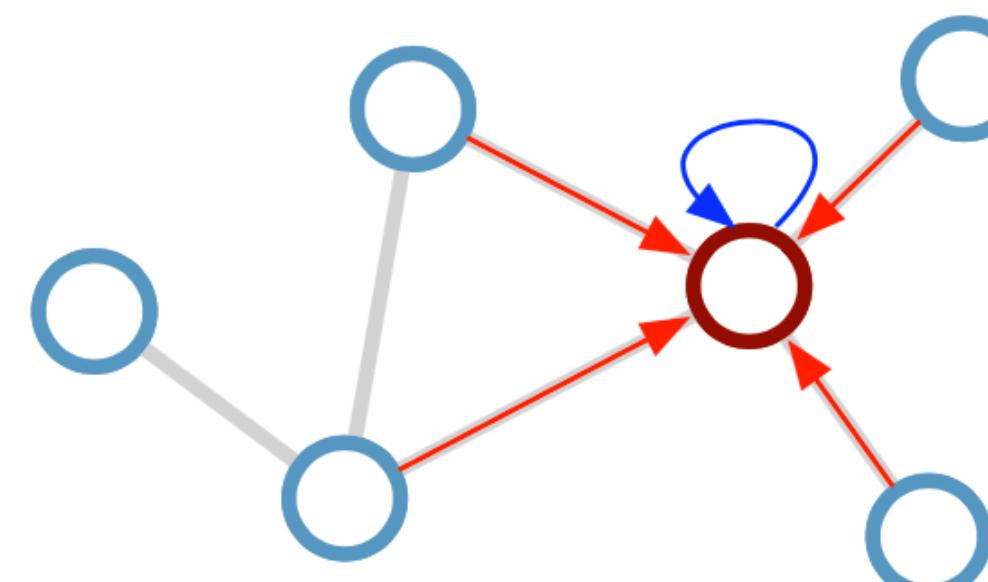
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this
undirected graph:



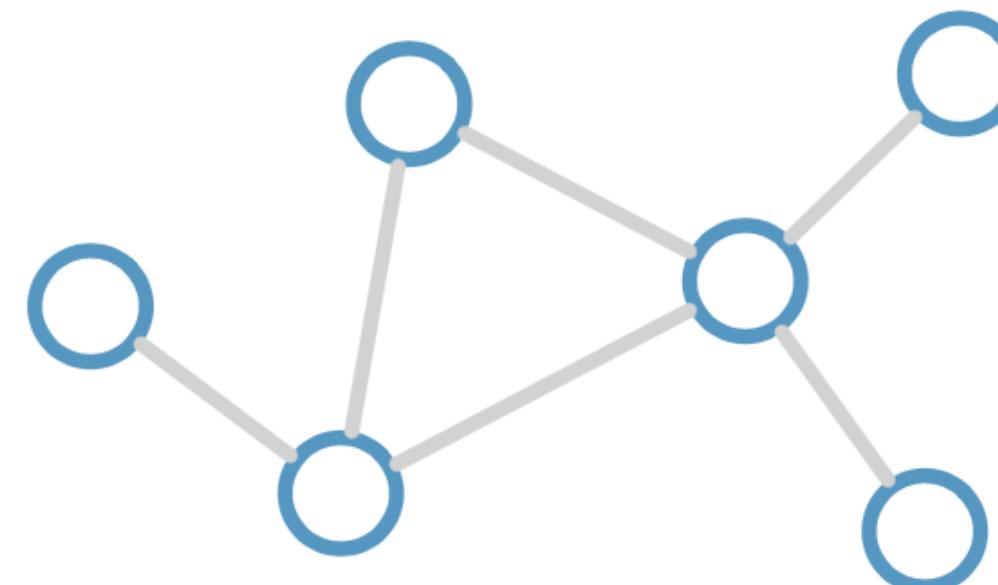
Calculate update
for node in red:



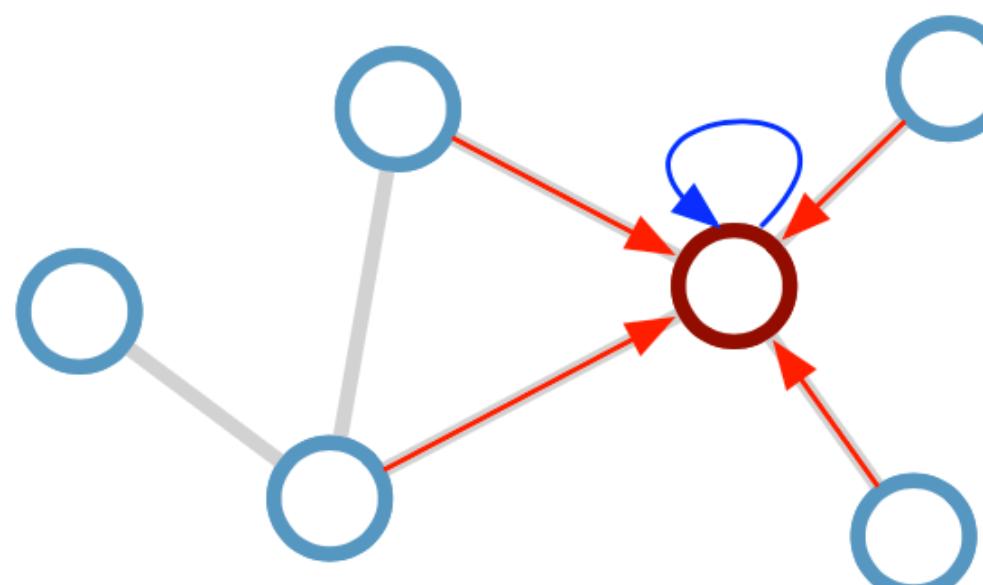
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this
undirected graph:



Calculate update
for node in red:



**Update
rule:**

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

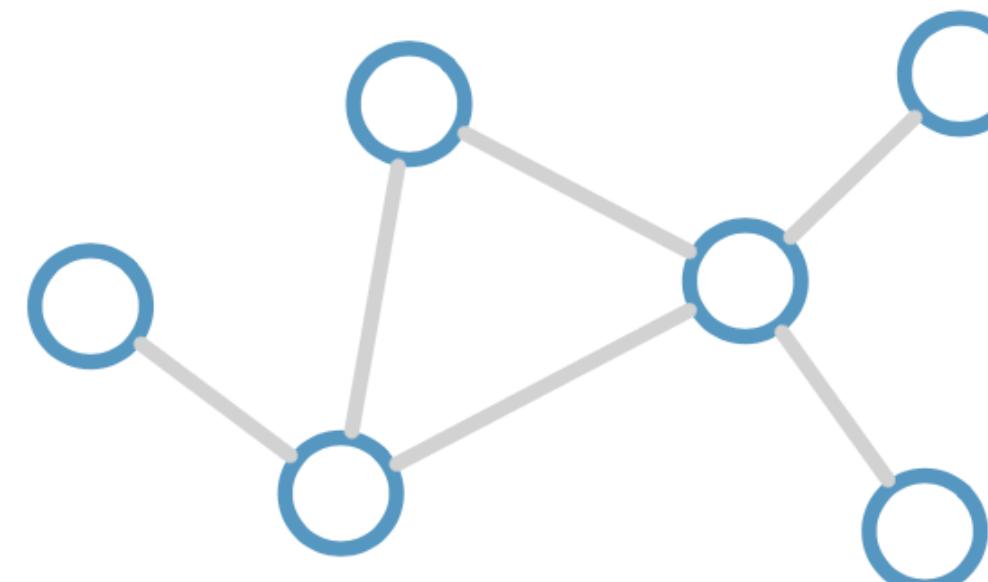
\mathcal{N}_i : neighbor indices

c_{ij} : norm. constant
(fixed/trainable)

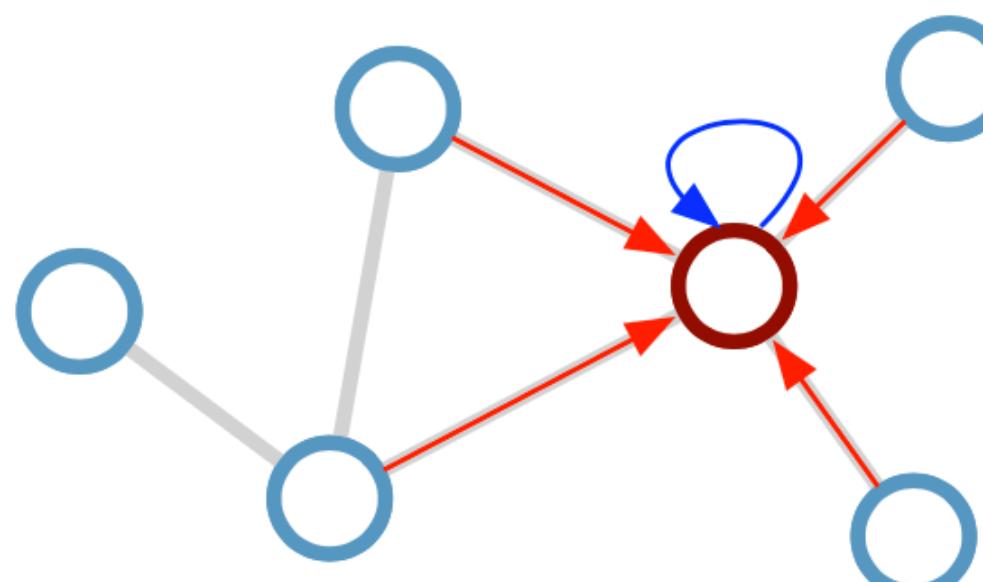
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

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

Compute Message

Aggregate Messages

Update Node Representation

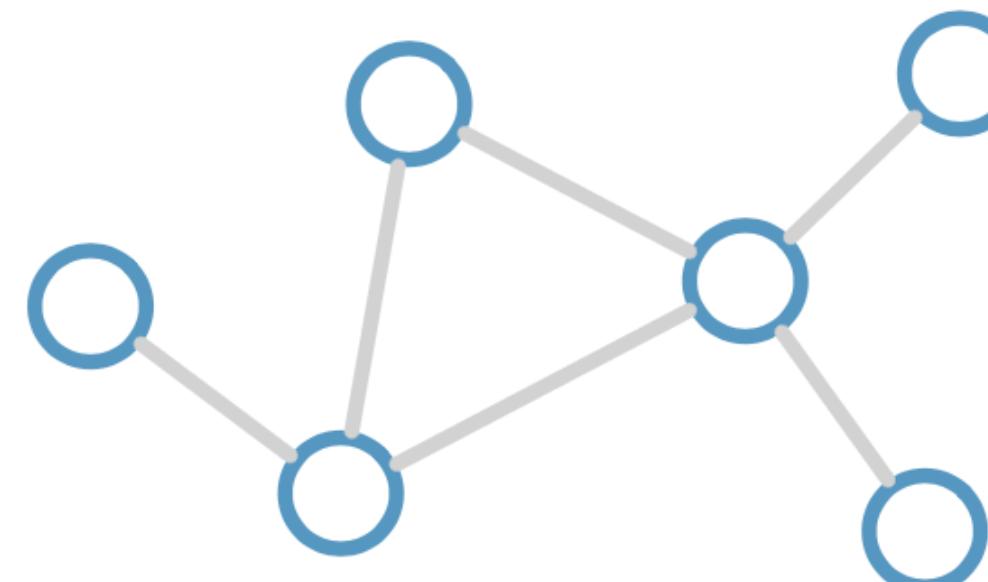
\mathcal{N}_i : neighbor indices

c_{ij} : norm. constant
(fixed/trainable)

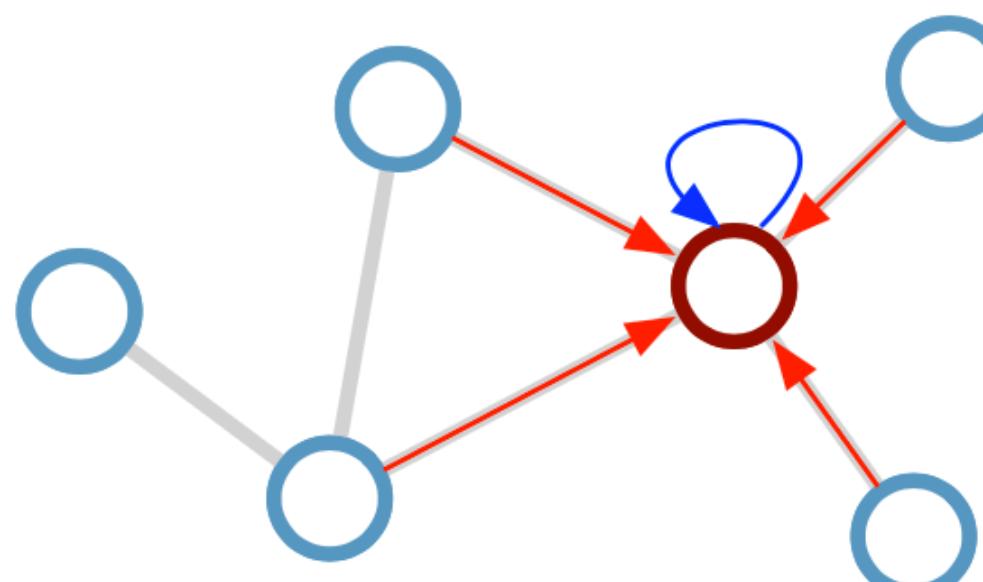
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this
undirected graph:



Calculate update
for node in red:



**Update
rule:**

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

Compute Message

Aggregate Messages

Update Node Representation

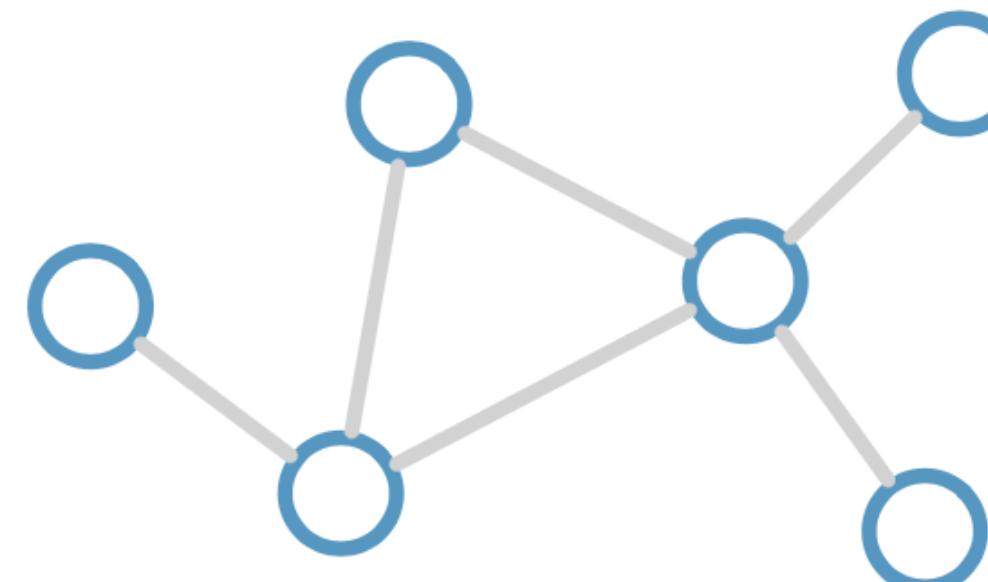
\mathcal{N}_i : neighbor indices

c_{ij} : norm. constant
(fixed/trainable)

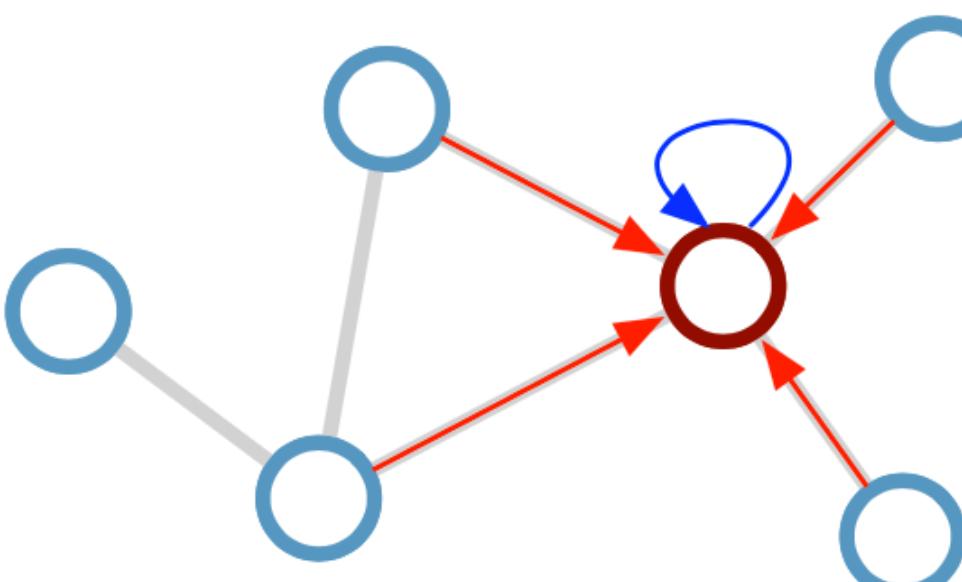
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

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

Compute Message

Aggregate Messages

Update Node Representation

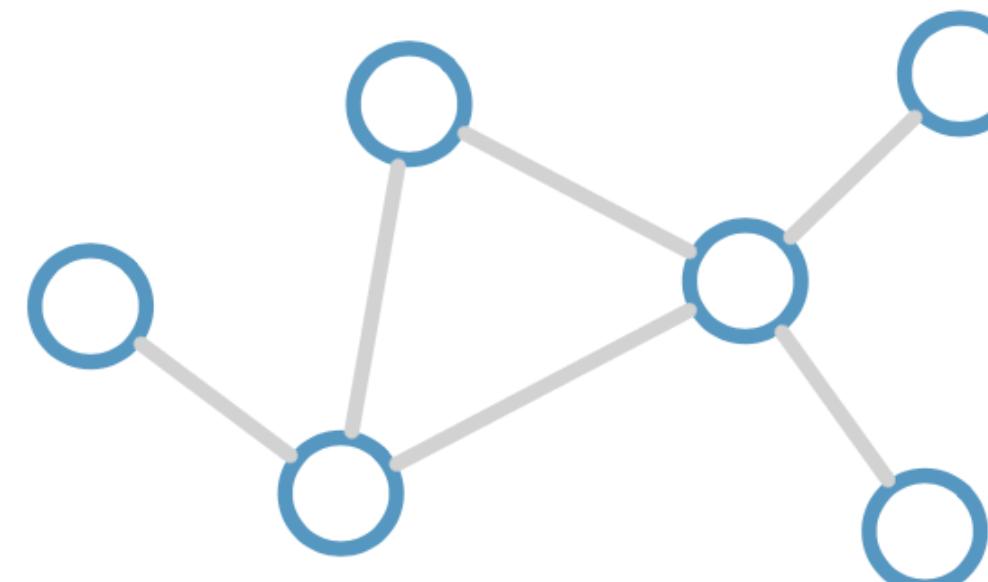
\mathcal{N}_i : neighbor indices

c_{ij} : norm. constant
(fixed/trainable)

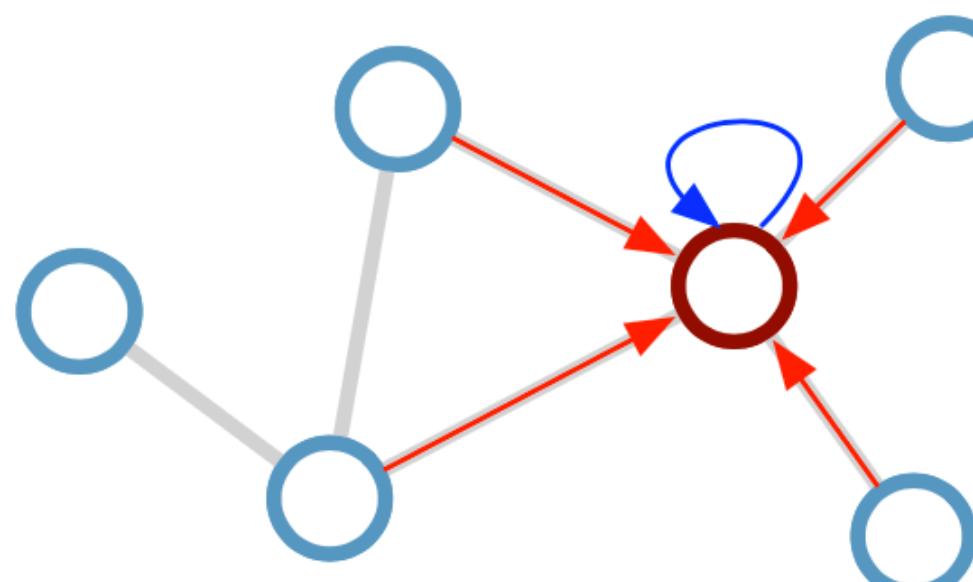
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this
undirected graph:



Calculate update
for node in red:



**Update
rule:**

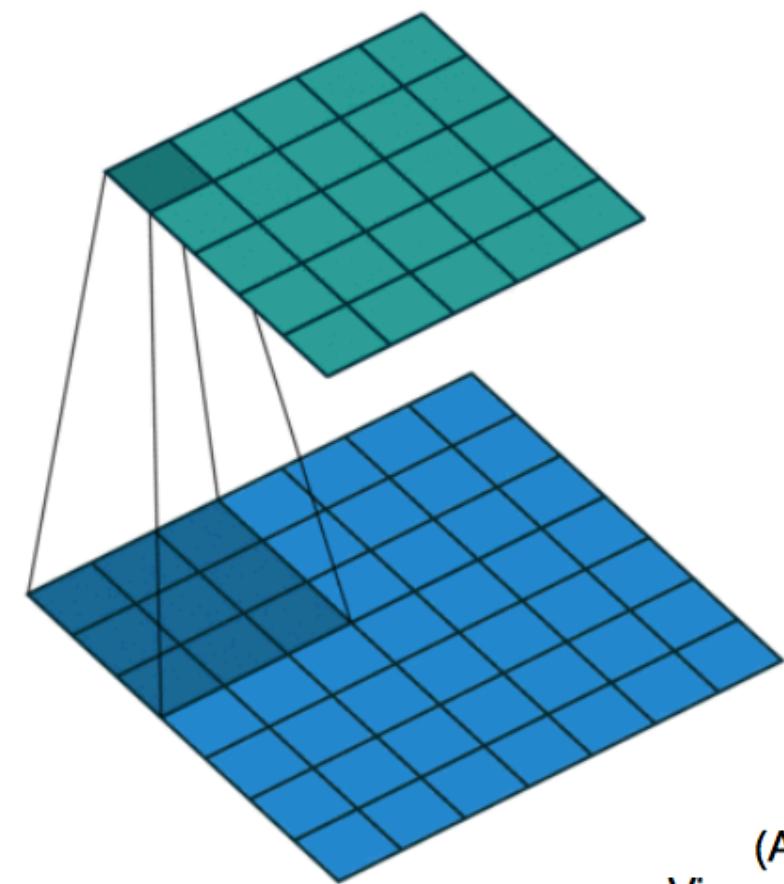
$$\mathbf{h}_i^{(l+1)} = \sigma \left(\mathbf{h}_i^{(l)} \mathbf{W}_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{h}_j^{(l)} \mathbf{W}_1^{(l)} \right)$$

\mathcal{N}_i : neighbor indices

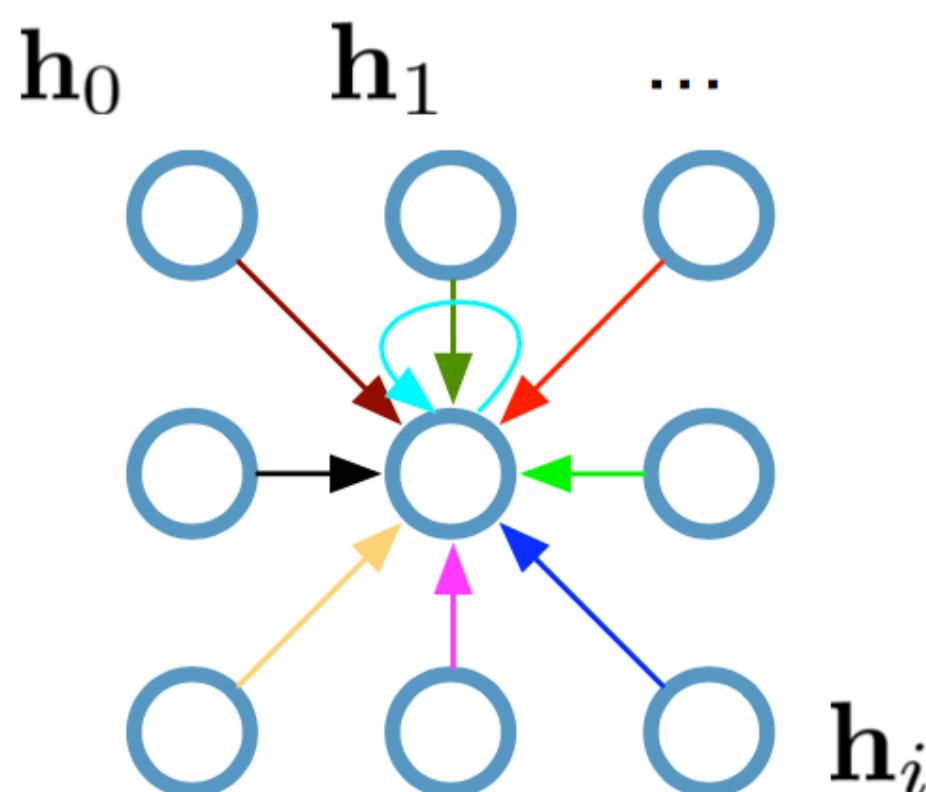
c_{ij} : norm. constant
(fixed/trainable)

Recap: Convolutional Neural Networks (CNNs) on Grids

**Single CNN layer
with 3x3 filter:**

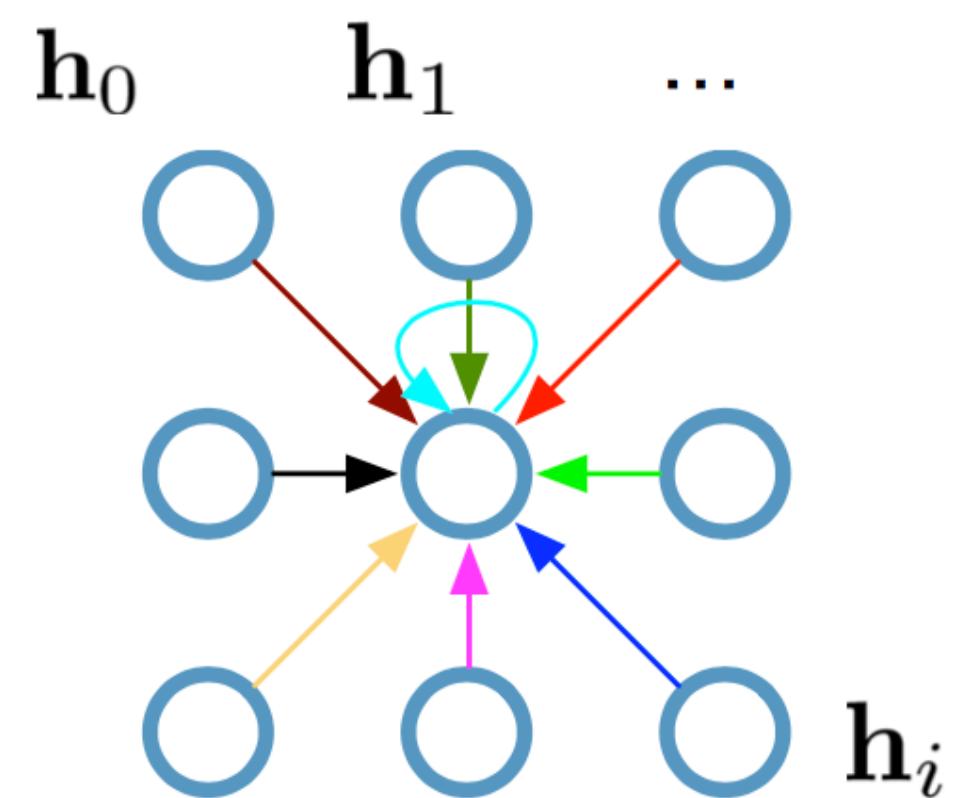
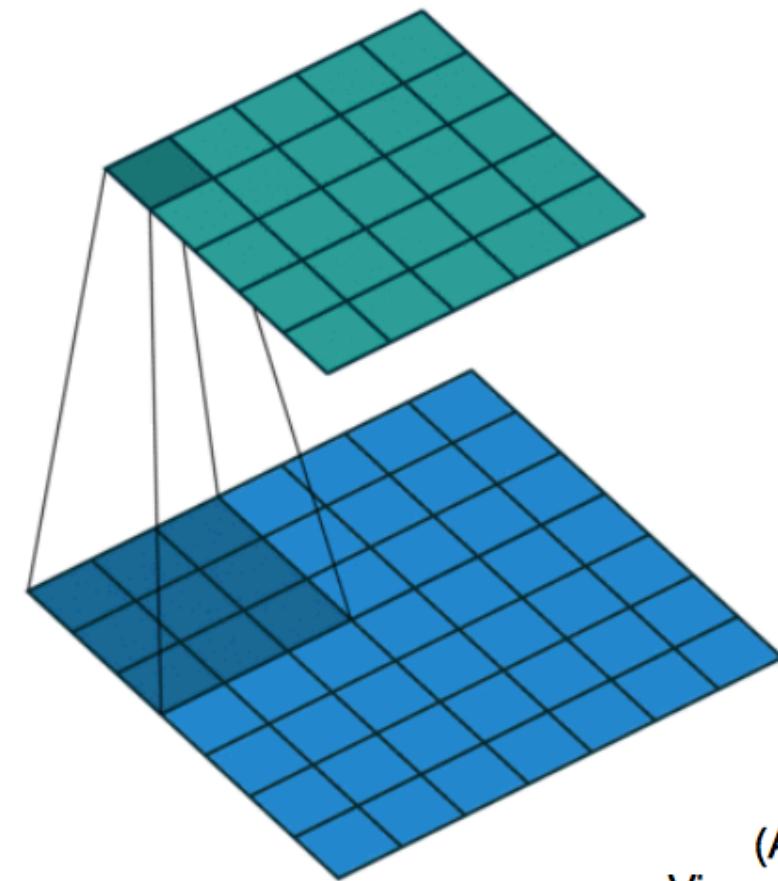


(Animation by
Vincent Dumoulin)



Recap: Convolutional Neural Networks (CNNs) on Grids

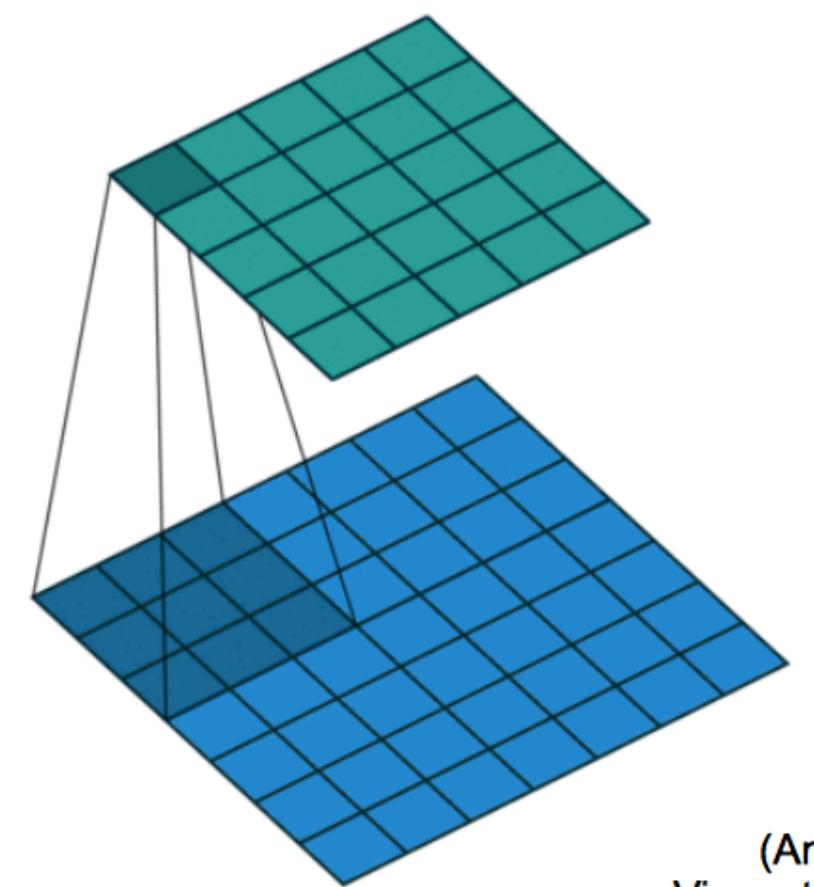
**Single CNN layer
with 3x3 filter:**



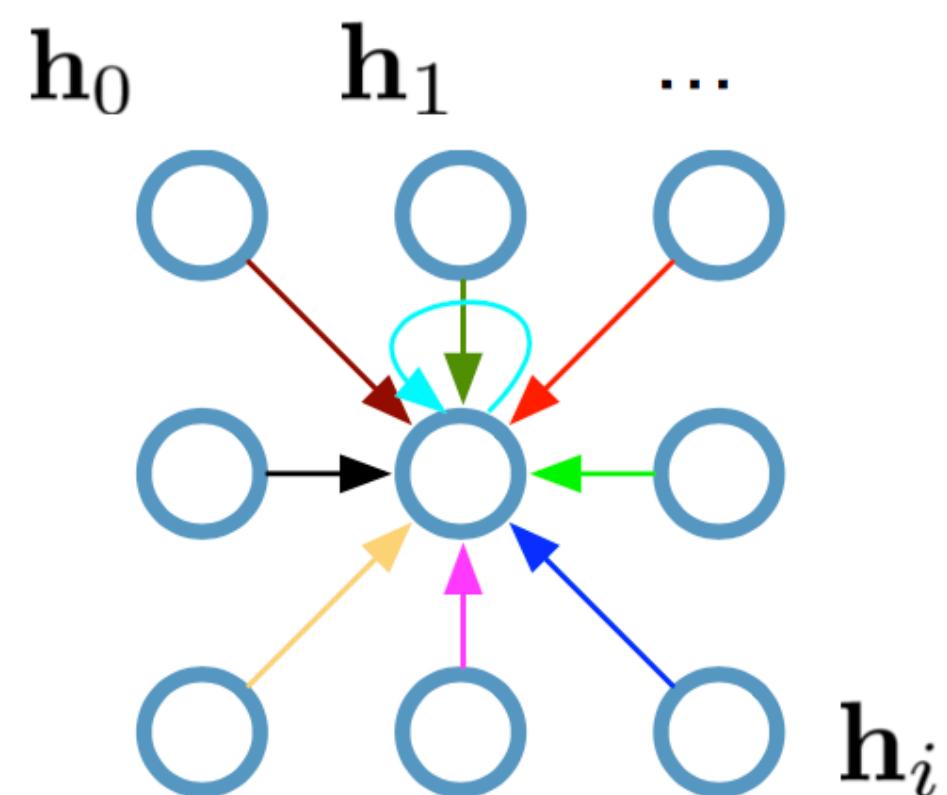
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Recap: Convolutional Neural Networks (CNNs) on Grids

**Single CNN layer
with 3x3 filter:**



(Animation by
Vincent Dumoulin)



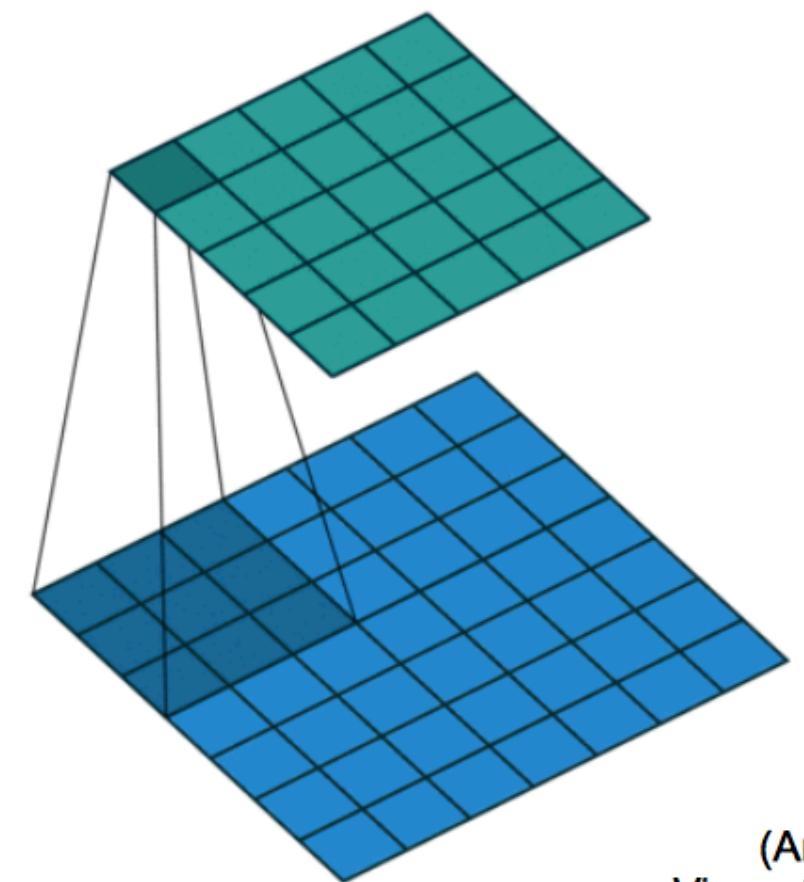
$h_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Update for a single pixel:

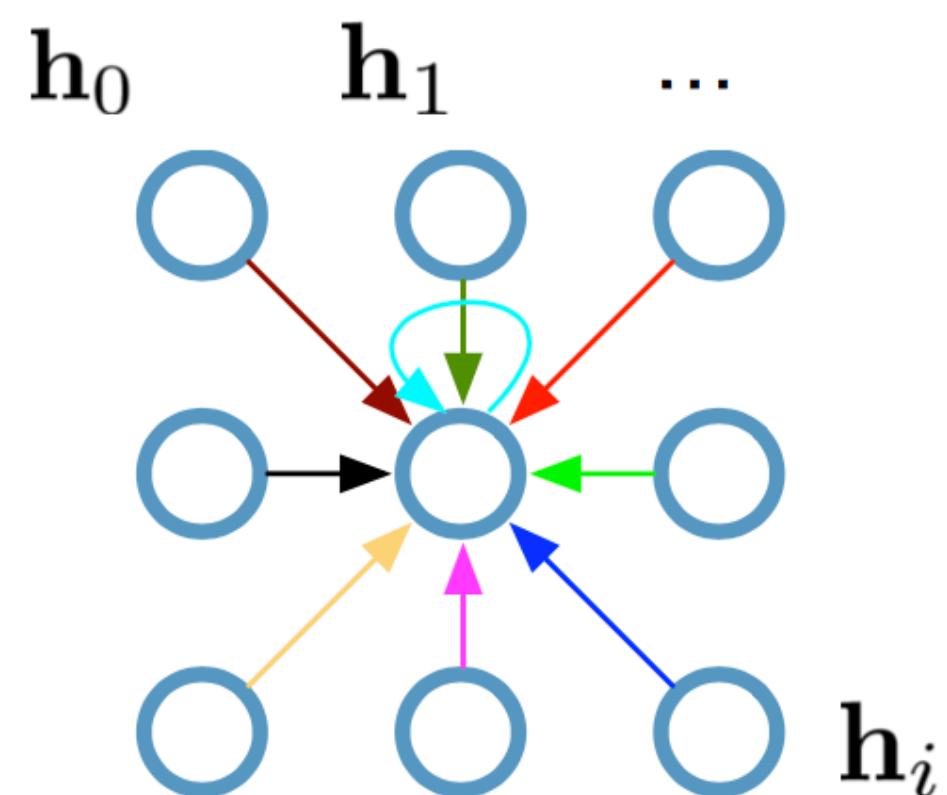
- Transform messages individually $W_i h_i$
- Add everything up $\sum_i W_i h_i$

Recap: Convolutional Neural Networks (CNNs) on Grids

**Single CNN layer
with 3x3 filter:**



(Animation by
Vincent Dumoulin)



$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Full update:

$$\mathbf{h}_4^{(l+1)} = \sigma \left(\mathbf{W}_0^{(l)} \mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)} \mathbf{h}_1^{(l)} + \dots + \mathbf{W}_8^{(l)} \mathbf{h}_8^{(l)} \right)$$

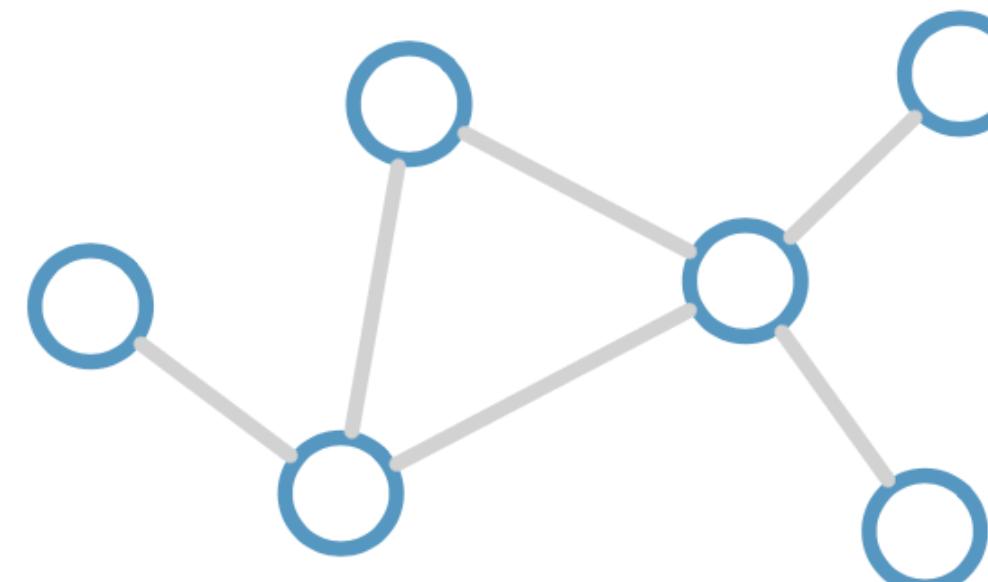
Update for a single pixel:

- Transform messages individually $\mathbf{W}_i \mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i \mathbf{h}_i$

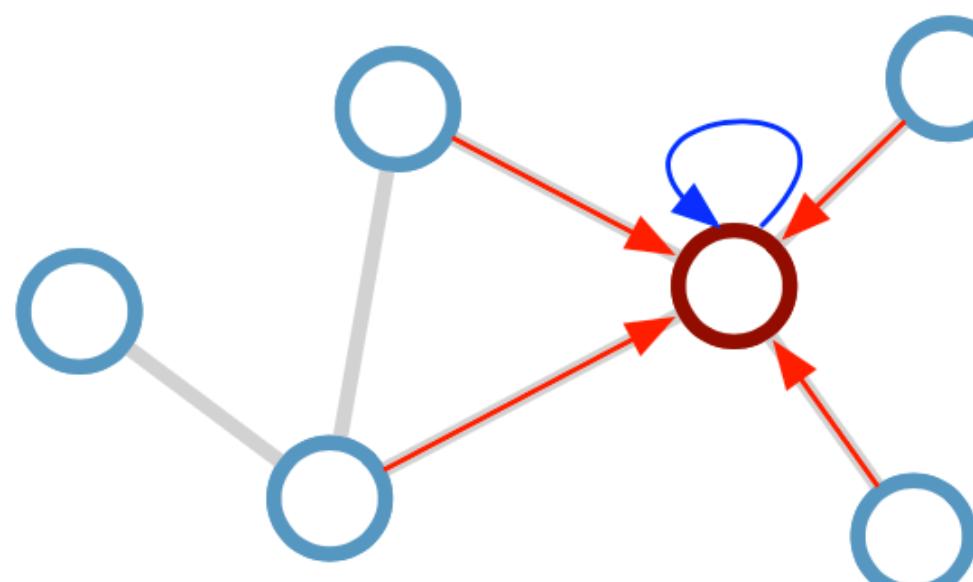
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this
undirected graph:



Calculate update
for node in red:



**Update
rule:**

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

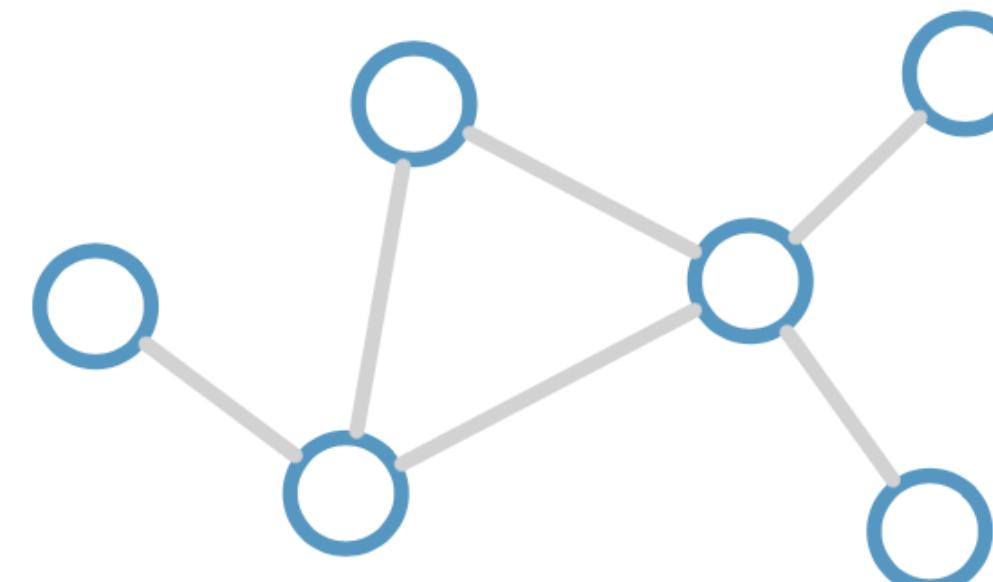
\mathcal{N}_i : neighbor indices

c_{ij} : norm. constant
(fixed/trainable)

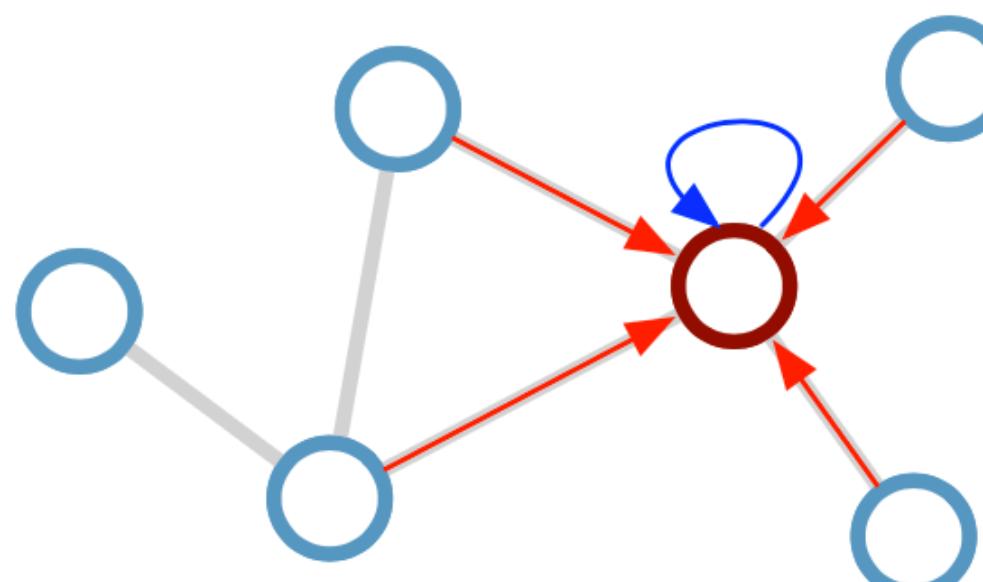
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



No self loops, or normalization:

$$\sigma(\mathbf{A} \mathbf{H} \mathbf{W}_1^{(l)})$$

Update rule:

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

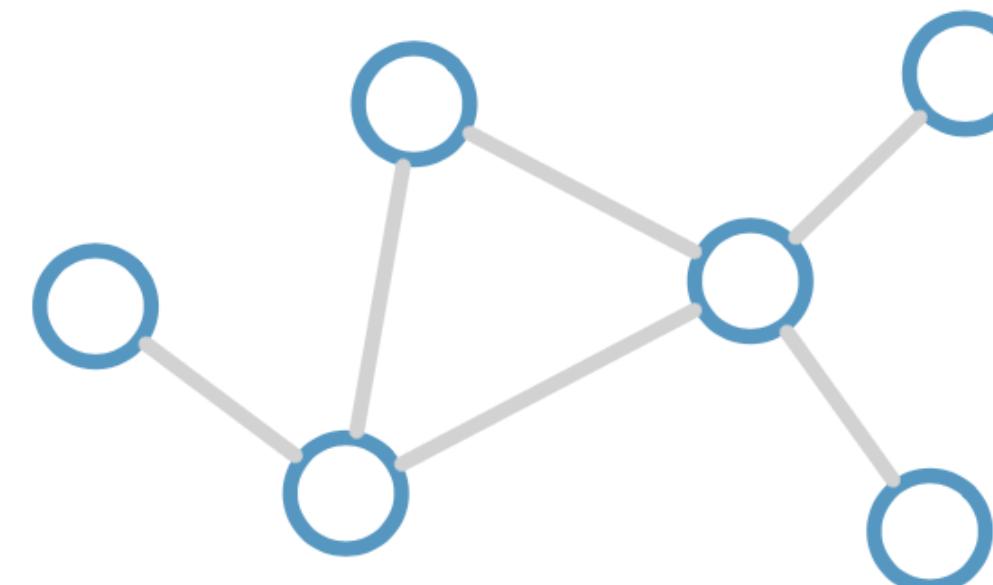
\mathcal{N}_i : neighbor indices

c_{ij} : norm. constant
(fixed/trainable)

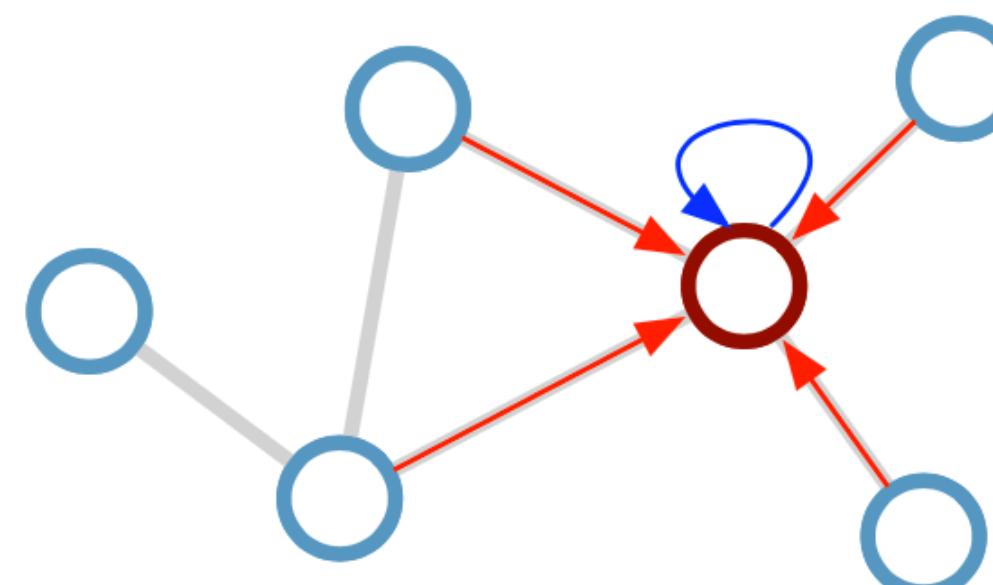
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



No self loops, or normalization:

$$\sigma(\mathbf{A} \mathbf{H} \mathbf{W}_1^{(l)})$$

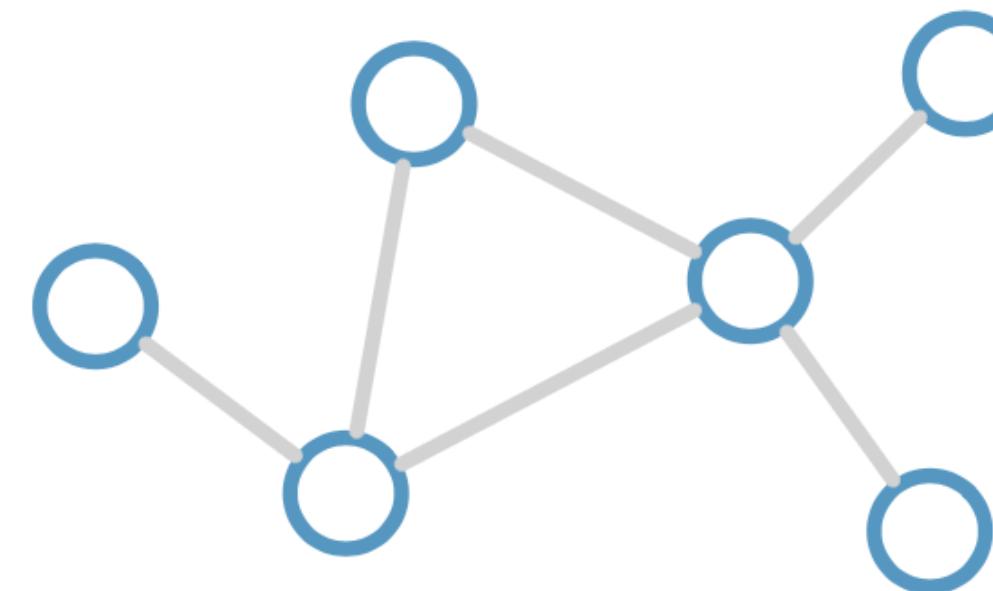
Update rule: $\mathbf{h}_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}_i} \mathbf{h}_j^{(l)} \mathbf{W}_1^{(l)} \right)$

\mathcal{N}_i : neighbor indices c_{ij} : norm. constant
(fixed/trainable)

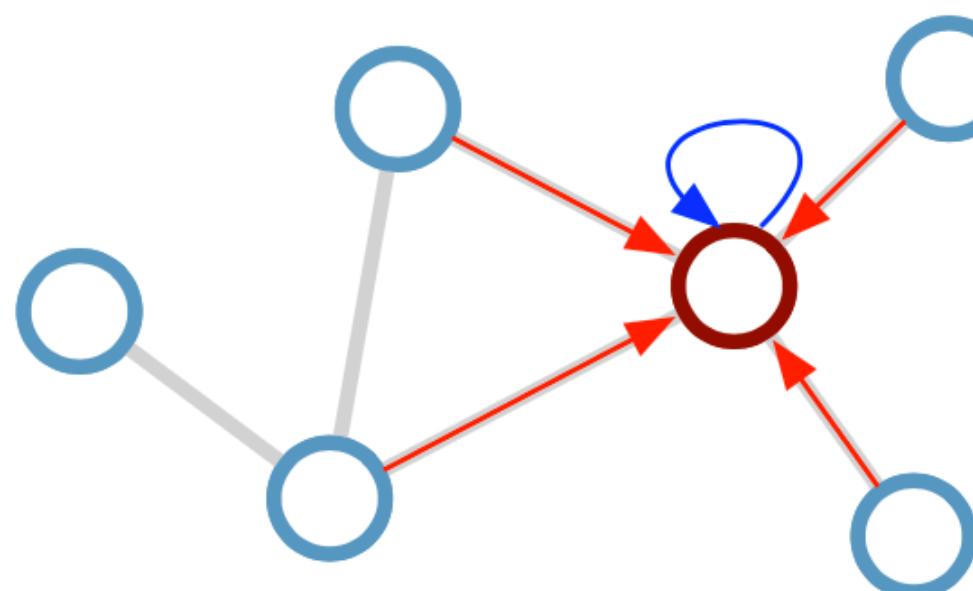
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



No self loops, or normalization:

$$\sigma(\mathbf{A}\mathbf{H}\mathbf{W}_1^{(l)})$$

+ self loops:

$$\sigma((\mathbf{A} + \mathbf{I})\mathbf{H}\mathbf{W}_1^{(l)})$$

Update rule:

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

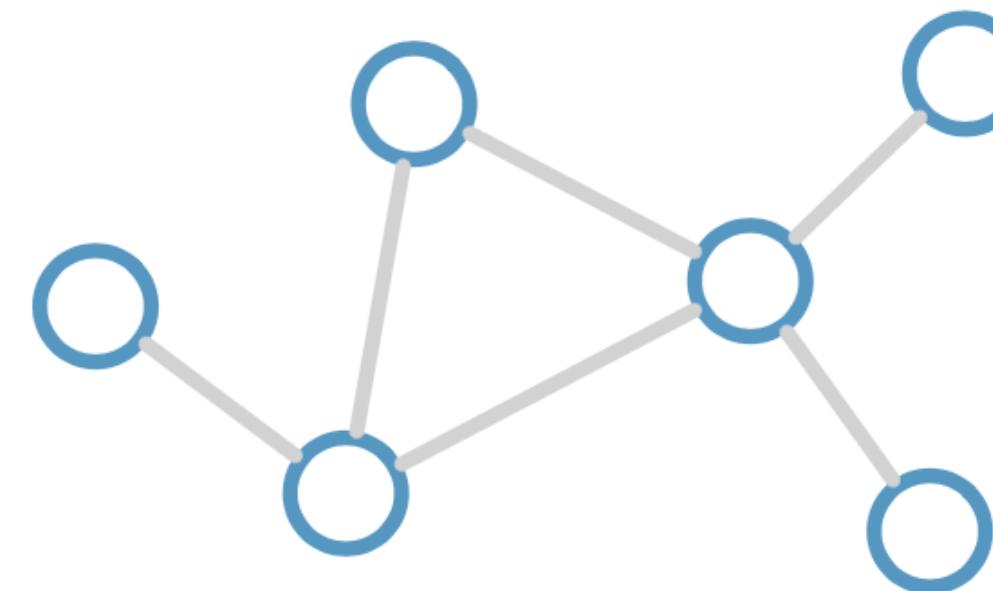
\mathcal{N}_i : neighbor indices

c_{ij} : norm. constant
(fixed/trainable)

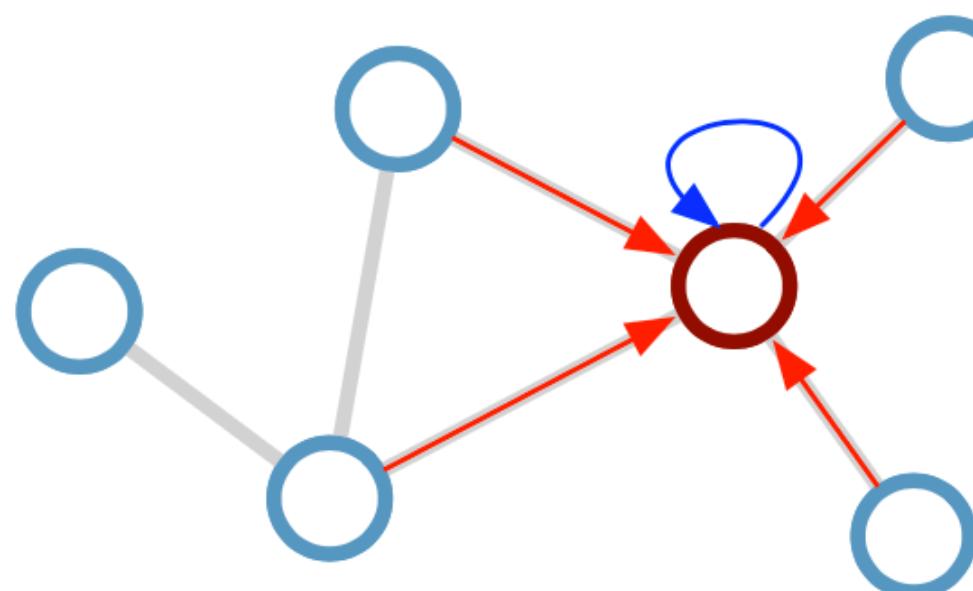
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

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

No self loops, or normalization:

$$\sigma(\mathbf{A} \mathbf{H} \mathbf{W}_1^{(l)})$$

+ self loops:

$$\sigma((\mathbf{A} + \mathbf{I}) \mathbf{H} \mathbf{W}_1^{(l)})$$

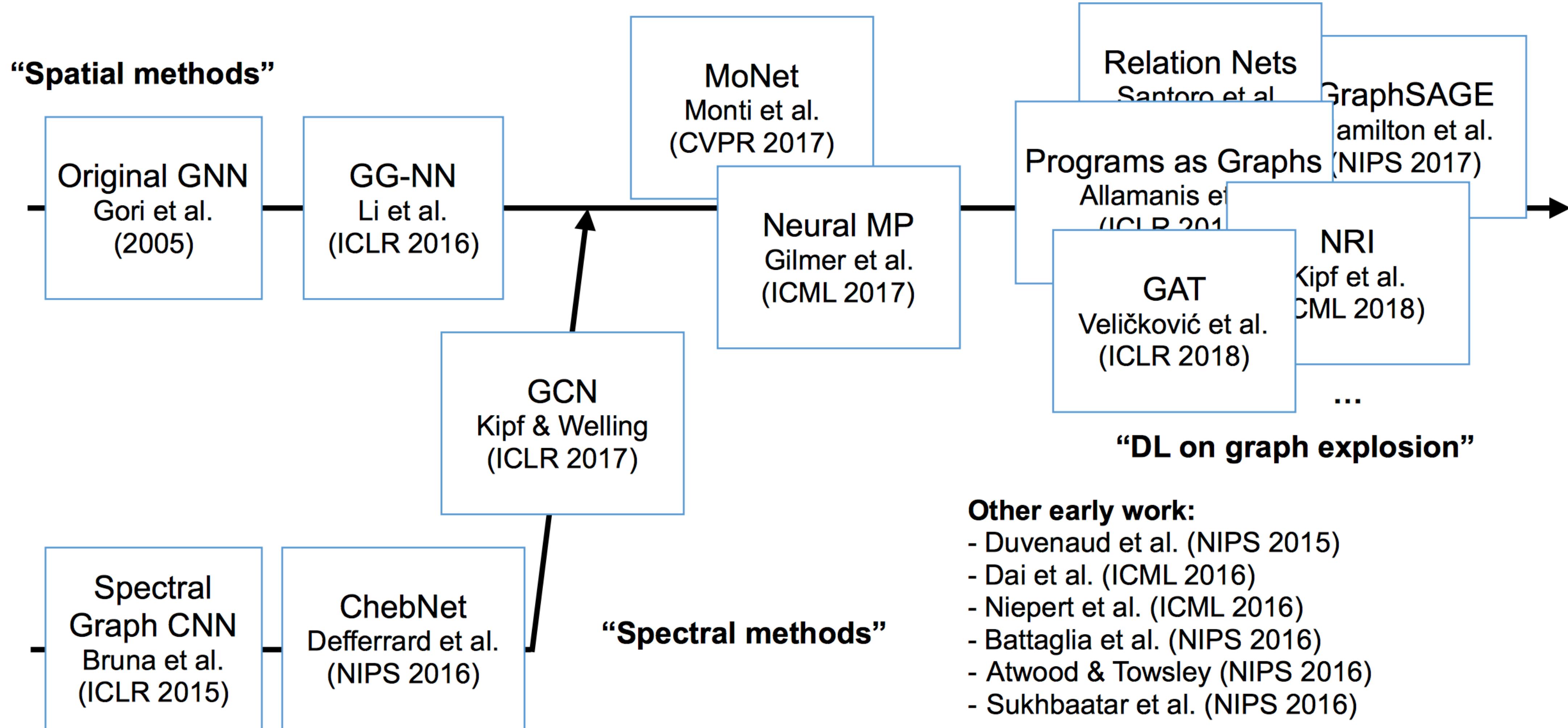
+ normalization:

$$\sigma([\mathbf{D}^{-\frac{1}{2}} (\mathbf{A} + \mathbf{I}) \mathbf{D}^{-\frac{1}{2}}] \mathbf{H} \mathbf{W}_1^{(l)})$$

\mathcal{N}_i : neighbor indices

c_{ij} : norm. constant
(fixed/trainable)

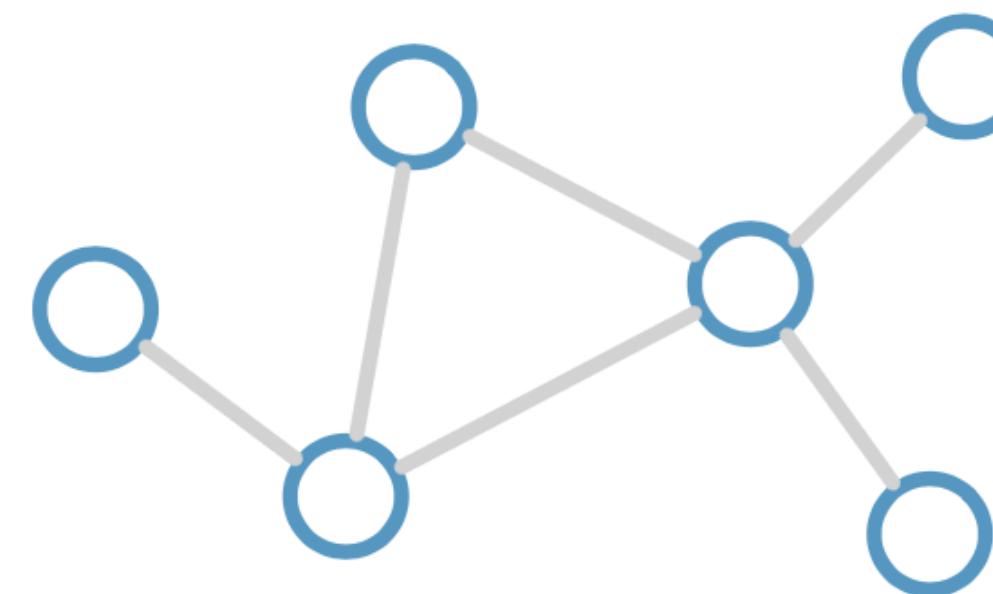
A Brief History of Graph Neural Nets



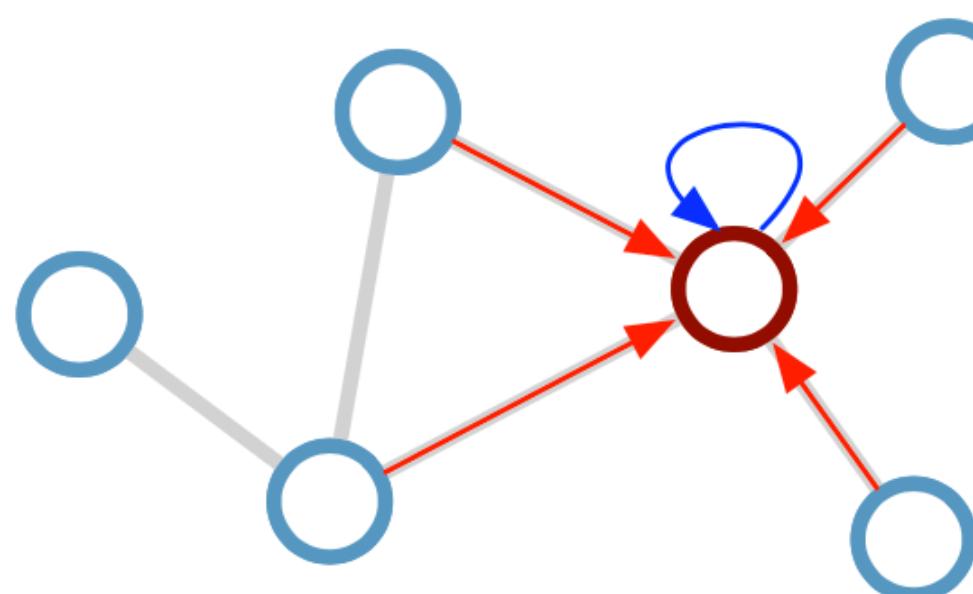
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

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

No self loops, or normalization:

$$\sigma(\mathbf{A} \mathbf{H} \mathbf{W}_1^{(l)})$$

+ self loops:

$$\sigma((\mathbf{A} + \mathbf{I}) \mathbf{H} \mathbf{W}_1^{(l)})$$

+ normalization:

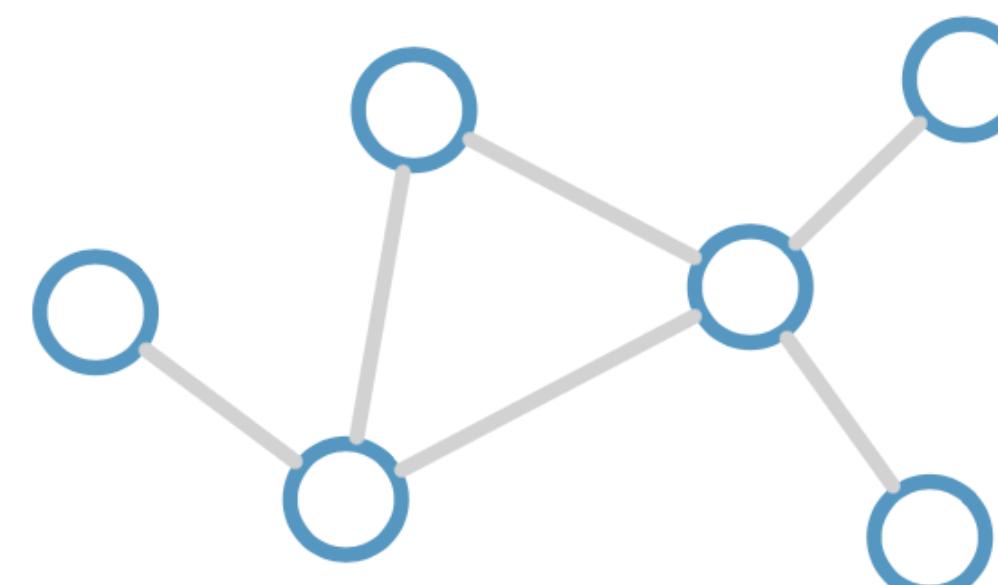
$$\sigma([\mathbf{D}^{-\frac{1}{2}} (\mathbf{A} + \mathbf{I}) \mathbf{D}^{-\frac{1}{2}}] \mathbf{H} \mathbf{W}_1^{(l)})$$

\mathcal{N}_i : neighbor indices

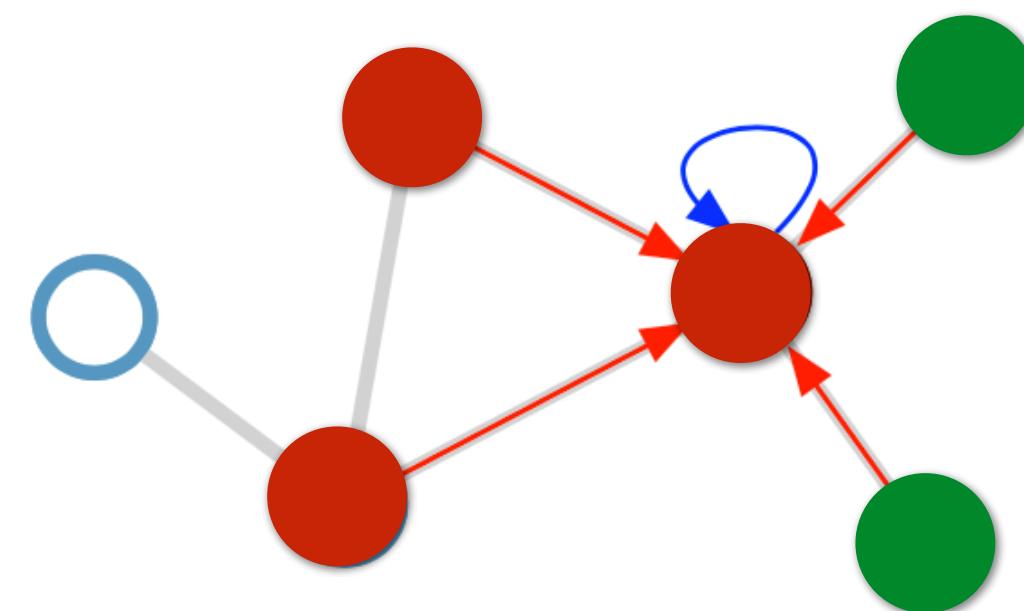
c_{ij} : norm. constant
(fixed/trainable)

GCN with **different** node types and feature dimensions

Consider this
undirected graph:



Calculate update
for node in red:



$$\mathbf{W}_0, \mathbf{W}_1 \in \mathbb{R}^{\mathbf{F} \times \mathbf{F}}$$

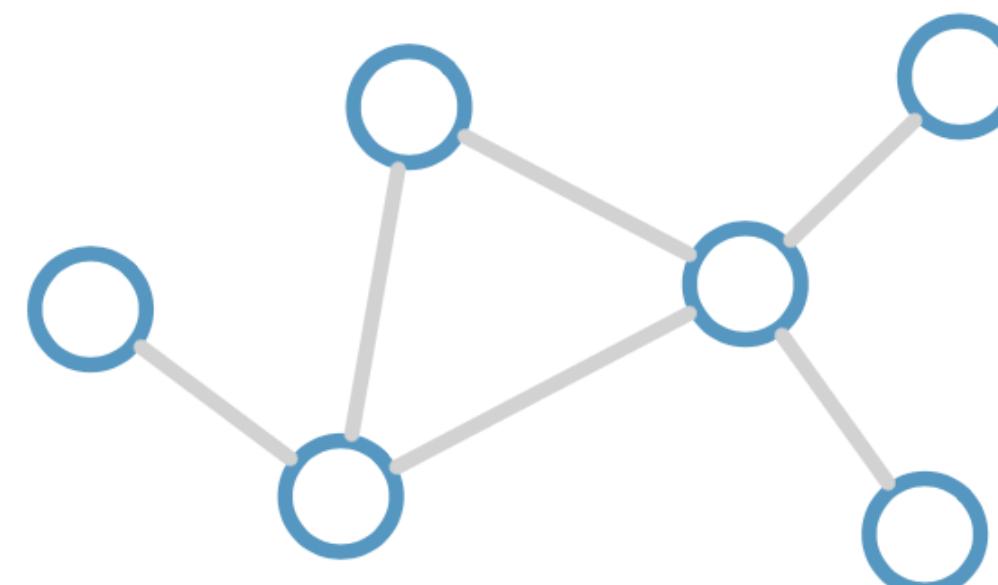
$$\mathbf{W}_2 \in \mathbb{R}^{\mathbf{F} \times \mathbf{F}'}$$

Update rule: $\mathbf{h}_i^{(l+1)} = \sigma \left(\mathbf{h}_i^{(l)} \mathbf{W}_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{h}_j^{(l)} \mathbf{W}_1^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{h}_j^{(l)} \mathbf{W}_2^{(l)} \right)$

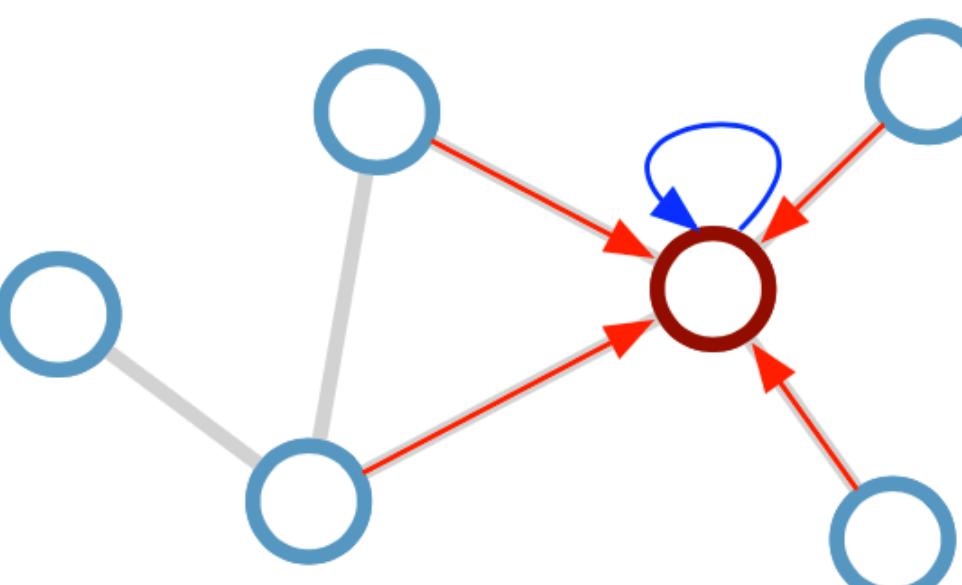
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

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

Desirable properties:

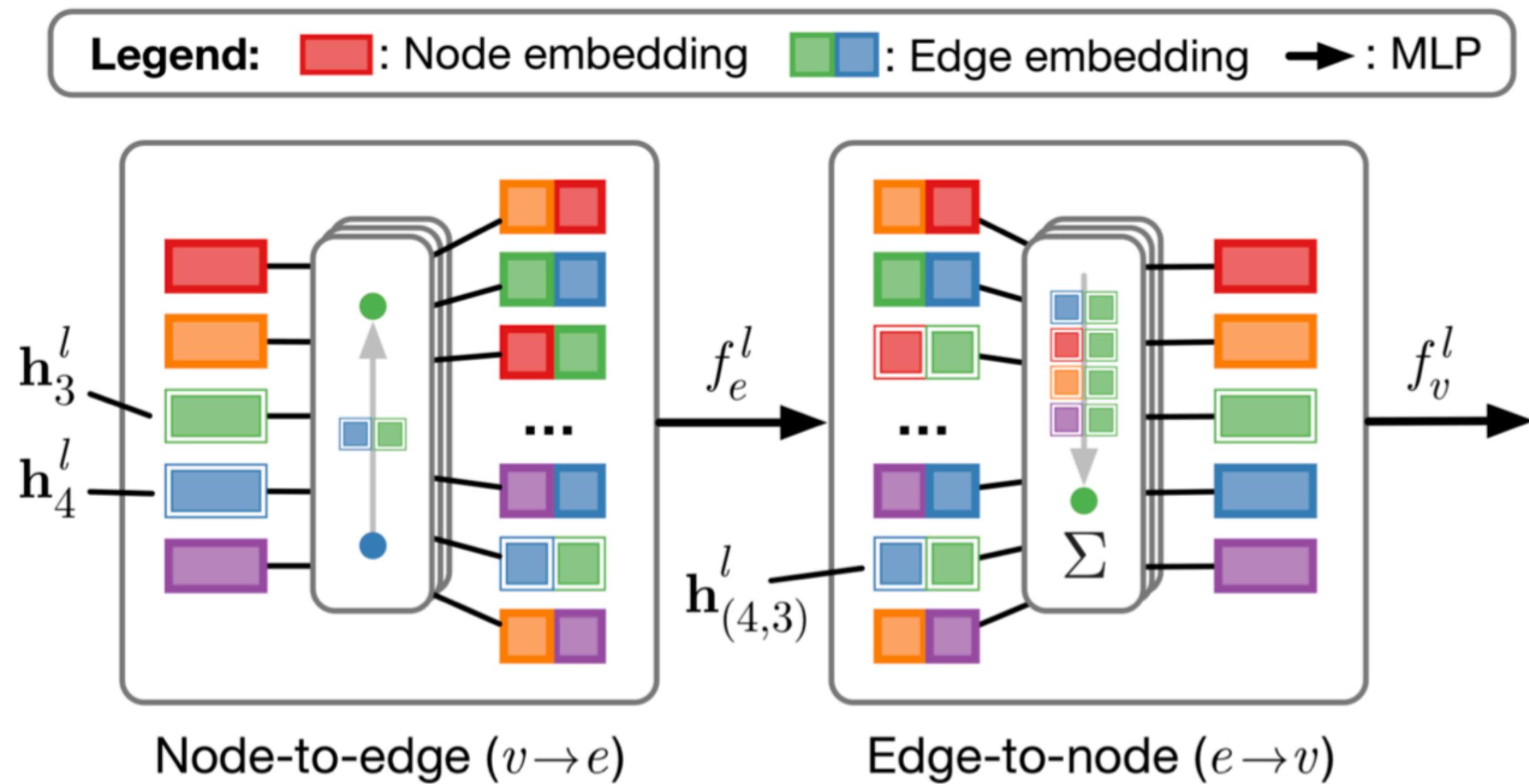
- Weight sharing over all locations
- Invariance to permutations
- Linear complexity $O(E)$
- Applicable both in transductive and inductive settings

\mathcal{N}_i : neighbor indices

c_{ij} : norm. constant
(fixed/trainable)

GNNs with Edge Embeddings

Battaglia et al. (NIPS 2016), Gilmer et al. (ICML 2017), Kipf et al. (ICML 2018)



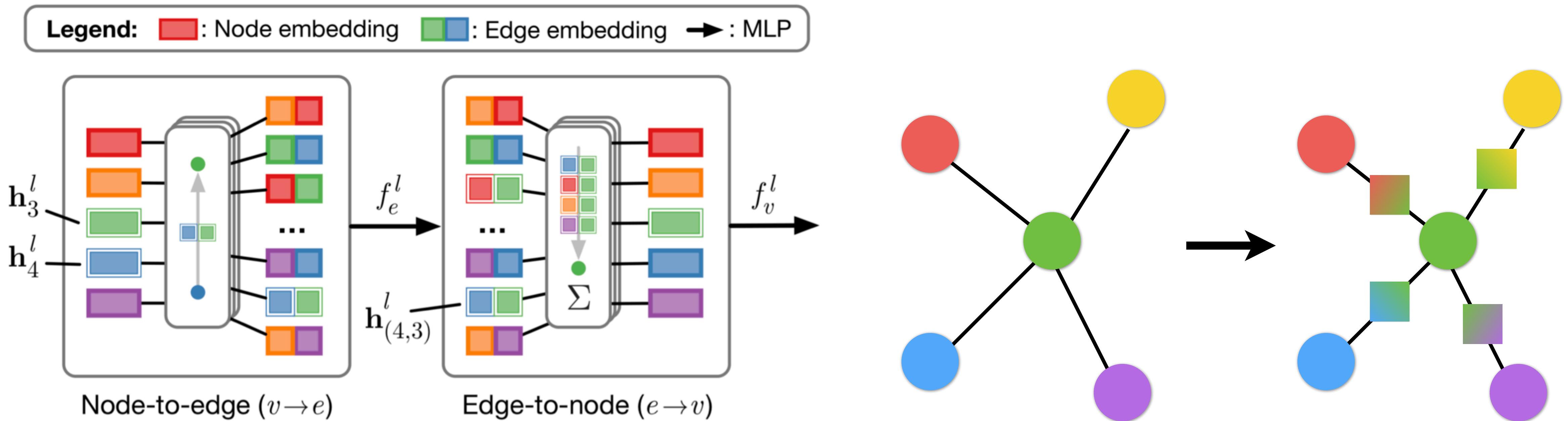
Note: The nodes and edges need not to have same dimensional representations; (the FC layers) will take care of this

Formally: $v \rightarrow e : \mathbf{h}_{(i,j)}^l = f_e^l([\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}])$

$$e \rightarrow v : \mathbf{h}_j^{l+1} = f_v^l([\sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j])$$

GNNs with Edge Embeddings

Battaglia et al. (NIPS 2016), Gilmer et al. (ICML 2017), Kipf et al. (ICML 2018)

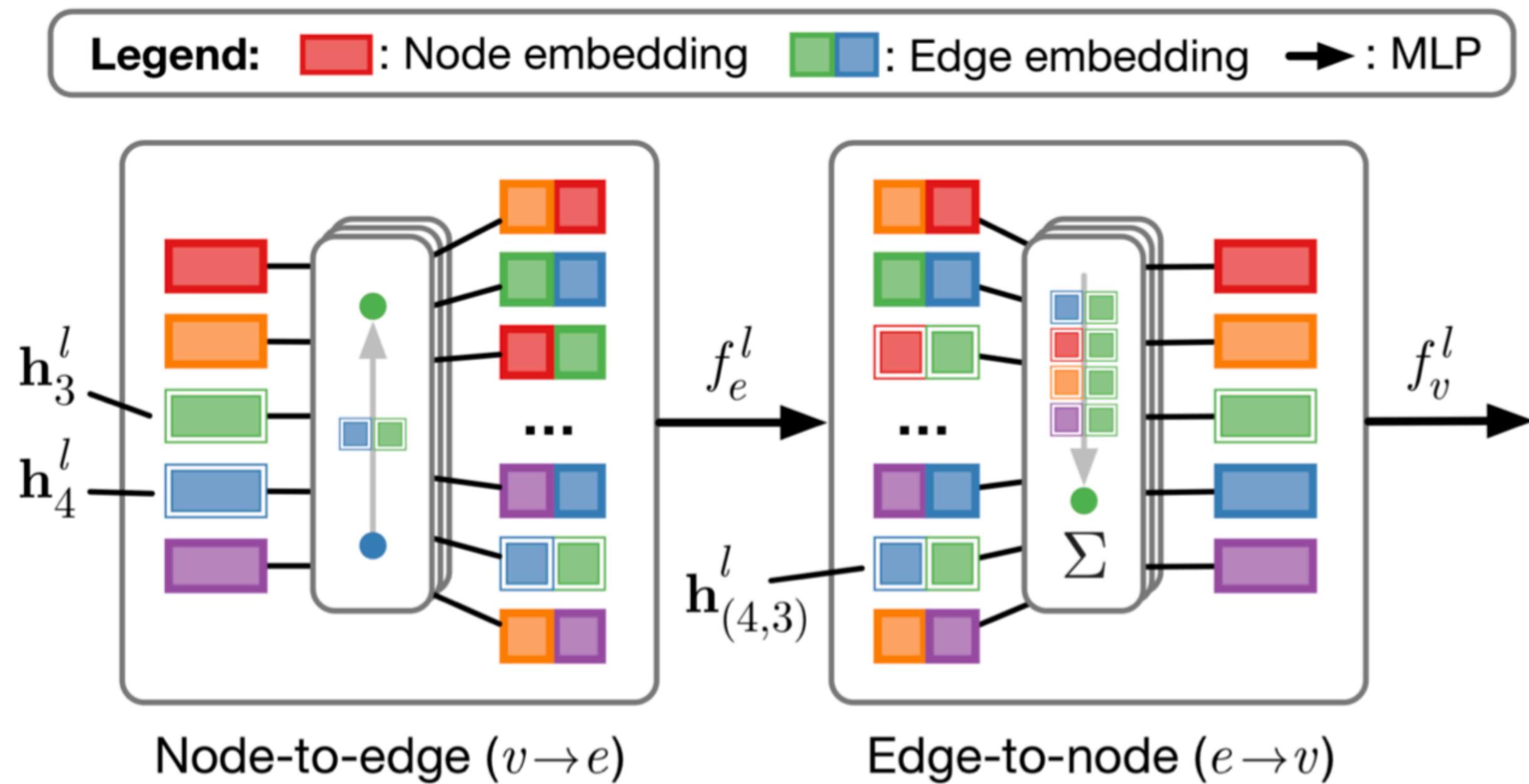


Formally: $v \rightarrow e : \mathbf{h}_{(i,j)}^l = f_e^l([\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}])$

$$e \rightarrow v : \mathbf{h}_j^{l+1} = f_v^l([\sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j])$$

GNNs with Edge Embeddings

Battaglia et al. (NIPS 2016), Gilmer et al. (ICML 2017), Kipf et al. (ICML 2018)



Pros:

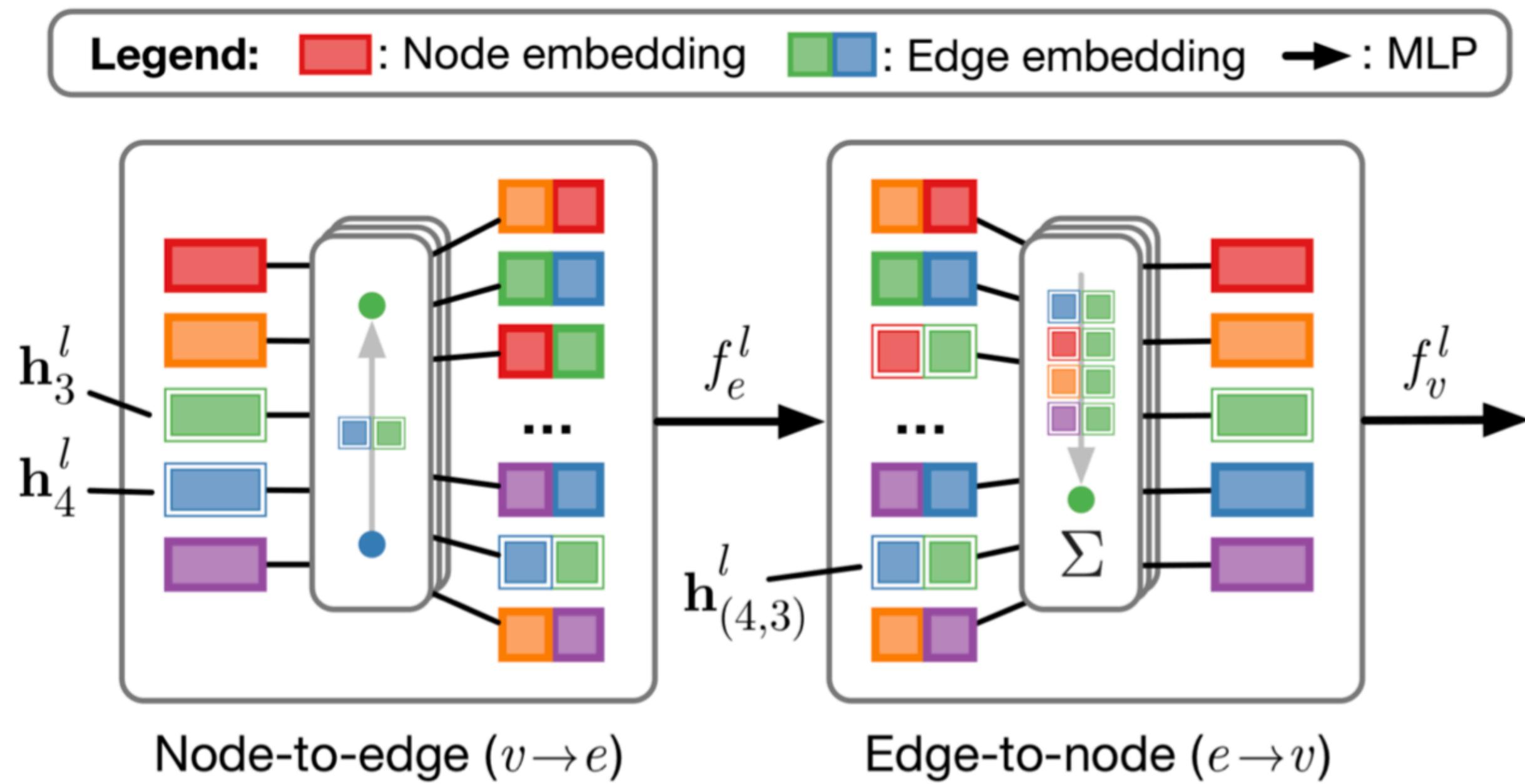
- Supports edge features
- More expressive than GCN
- As general as it gets (?)
- Supports sparse matrix ops

Formally: $v \rightarrow e : \mathbf{h}_{(i,j)}^l = f_e^l([\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}])$

$e \rightarrow v : \mathbf{h}_j^{l+1} = f_v^l([\sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j])$

GNNs with Edge Embeddings

Battaglia et al. (NIPS 2016), Gilmer et al. (ICML 2017), Kipf et al. (ICML 2018)



Formally: $v \rightarrow e : \mathbf{h}_{(i,j)}^l = f_e^l([\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}])$
 $e \rightarrow v : \mathbf{h}_j^{l+1} = f_v^l([\sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j])$

Pros:

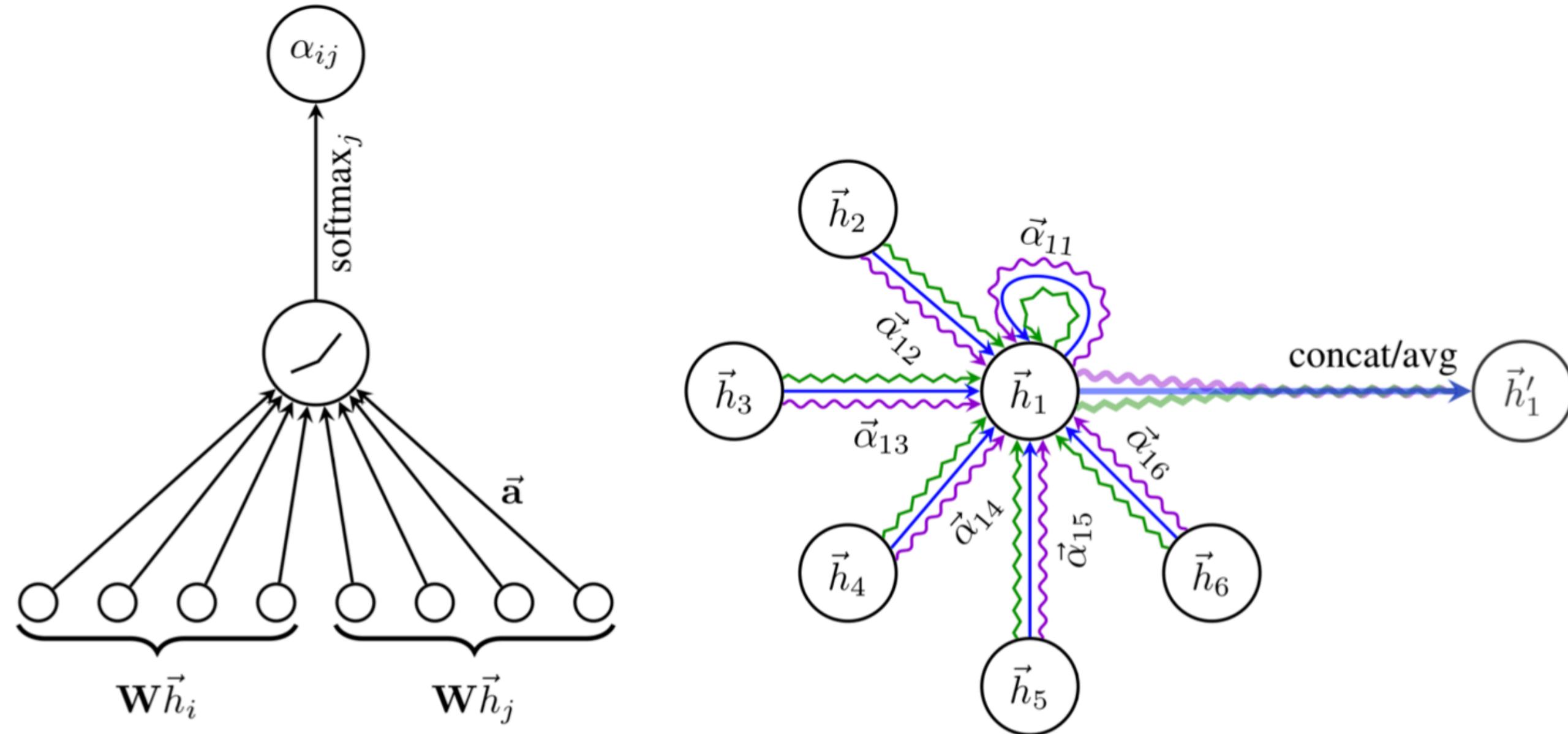
- Supports edge features
- More expressive than GCN
- As general as it gets (?)
- Supports sparse matrix ops

Cons:

- Need to store intermediate edge-based activations
- Difficult to implement with subsampling
→ In practice limited to small graphs

Graph Neural Networks (GNNs) with Attention

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)



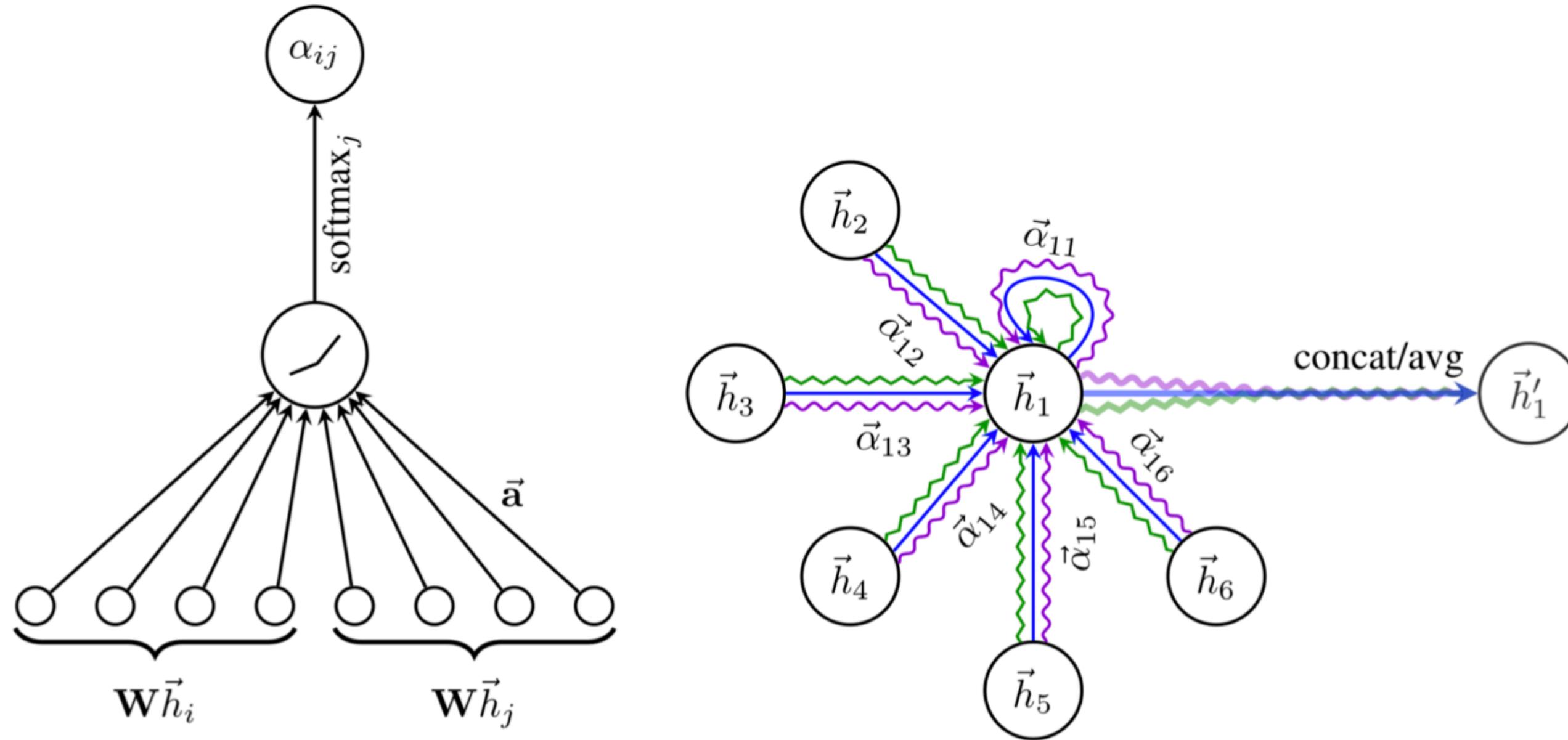
[Figure from Veličković et al. (ICLR 2018)]

$$\vec{h}'_i = \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right)$$

Multi-graph (k-type edges), a.k.a., can think of it as K attention heads

Graph Neural Networks (GNNs) with Attention

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)



[Figure from Veličković et al. (ICLR 2018)]

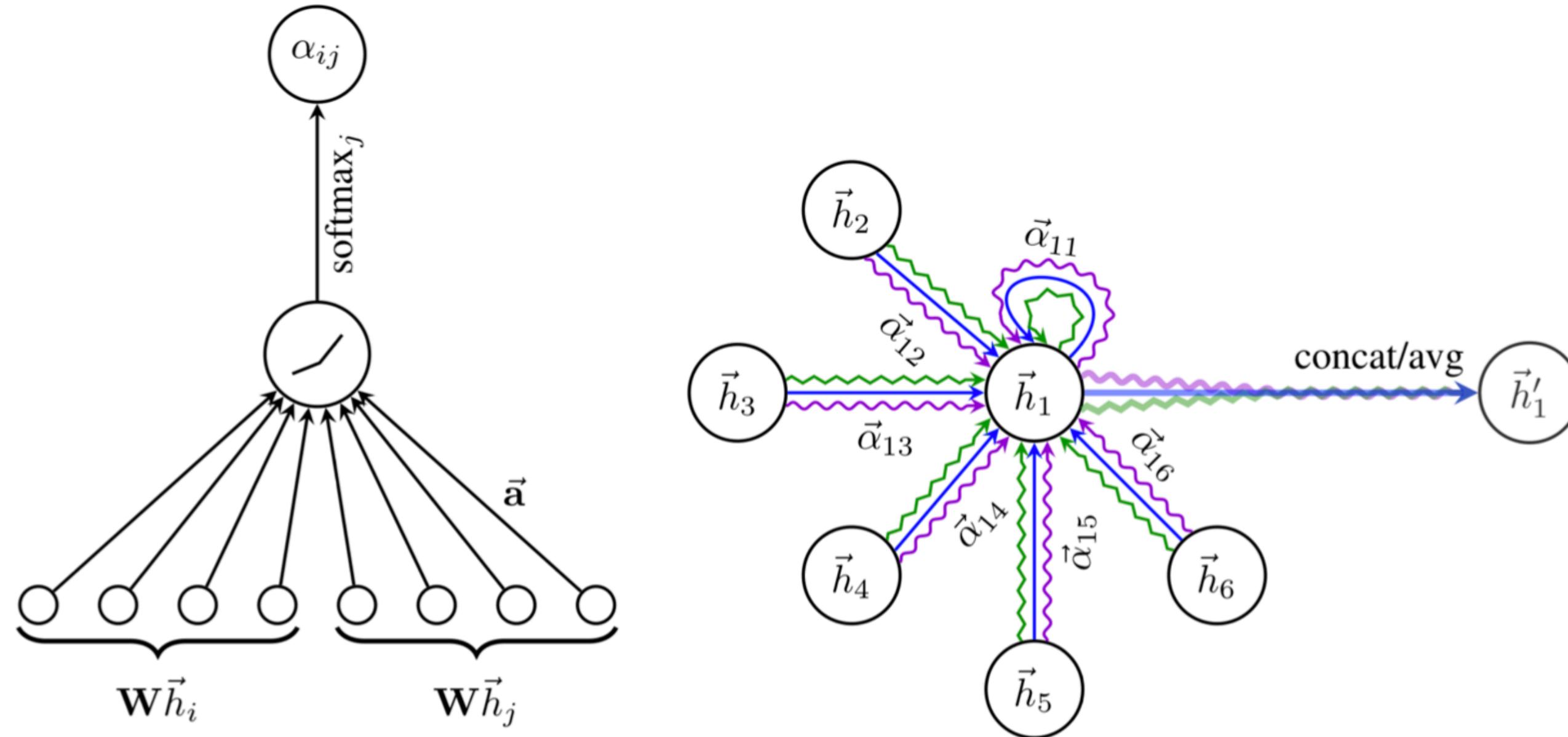
$$\vec{h}'_i = \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right)$$

$$\alpha_{ij} = \frac{\exp \left(\text{LeakyReLU} \left(\vec{\mathbf{a}}^T [\mathbf{W} \vec{h}_i \| \mathbf{W} \vec{h}_j] \right) \right)}{\sum_{k \in \mathcal{N}_i} \exp \left(\text{LeakyReLU} \left(\vec{\mathbf{a}}^T [\mathbf{W} \vec{h}_i \| \mathbf{W} \vec{h}_k] \right) \right)}$$

Multi-graph (k-type edges), a.k.a., can think of it as K attention heads

Graph Neural Networks (GNNs) with Attention

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)



[Figure from Veličković et al. (ICLR 2018)]

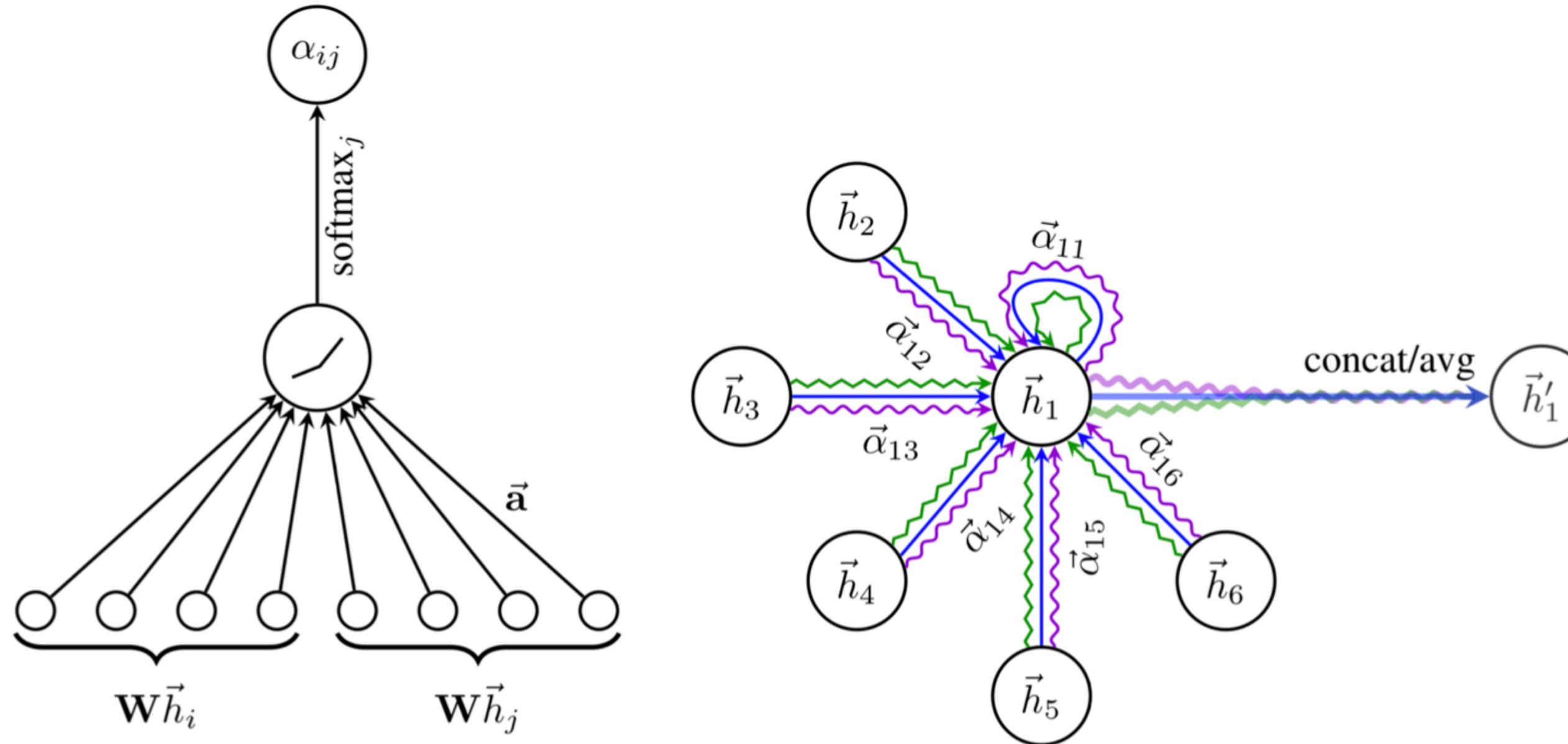
$$\vec{h}'_i = \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right)$$

$$\alpha_{ij} = \frac{\exp \left(\text{LeakyReLU} \left(\vec{a}^T [\mathbf{W} \vec{h}_i \| \mathbf{W} \vec{h}_j] \right) \right)}{\sum_{k \in \mathcal{N}_i} \exp \left(\text{LeakyReLU} \left(\vec{a}^T [\mathbf{W} \vec{h}_i \| \mathbf{W} \vec{h}_k] \right) \right)}$$

Multi-graph (k-type edges), a.k.a., can think of it as K attention heads

Graph Neural Networks (GNNs) with Attention

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)



[Figure from Veličković et al. (ICLR 2018)]

Pros:

- No need to store intermediate edge-based activation vectors (when using dot-product attn.)
- Slower than GCNs but faster than GNNs with edge embeddings

Cons:

- (Most likely) less expressive than GNNs with edge embeddings
- Can be more difficult to optimize

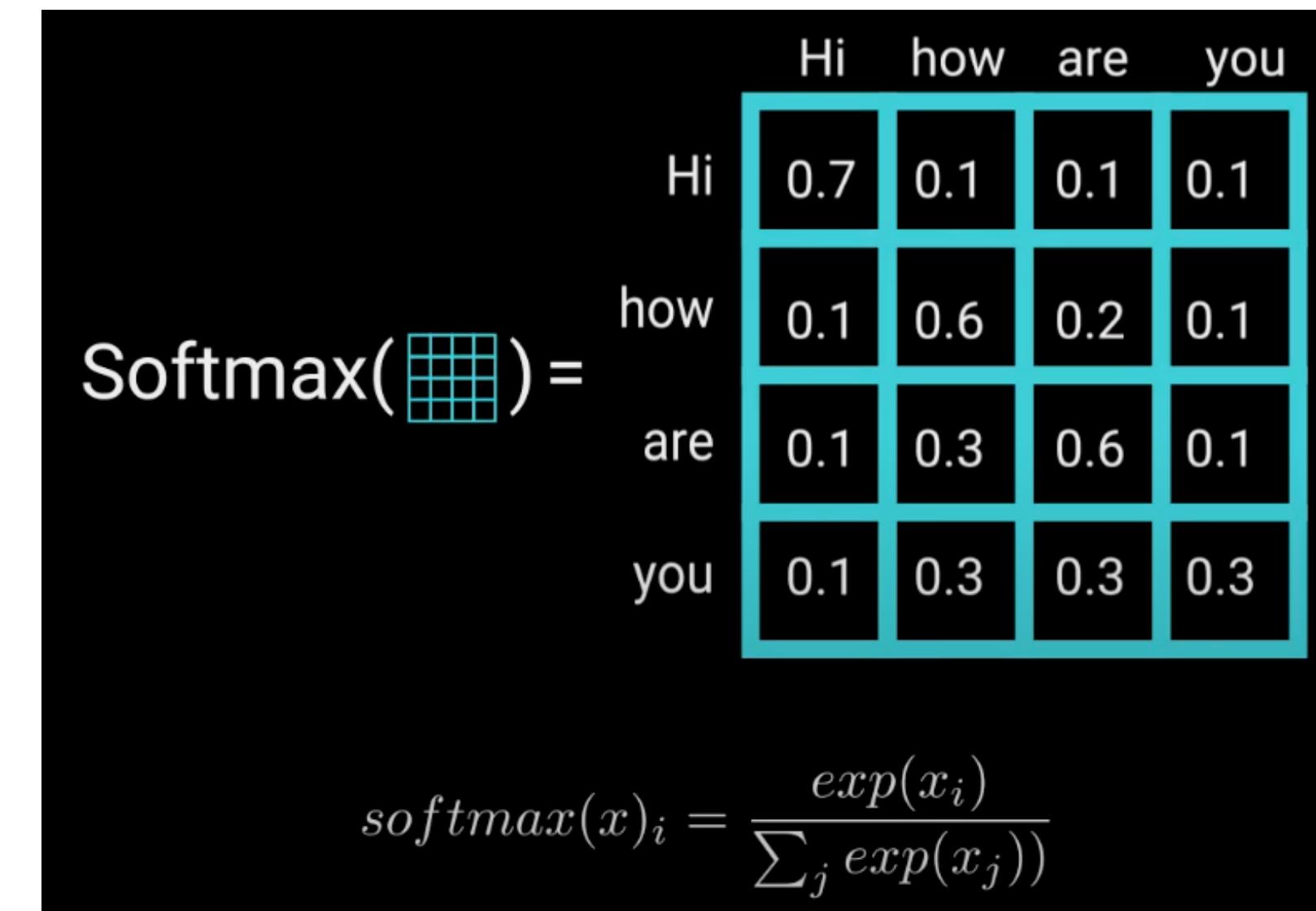
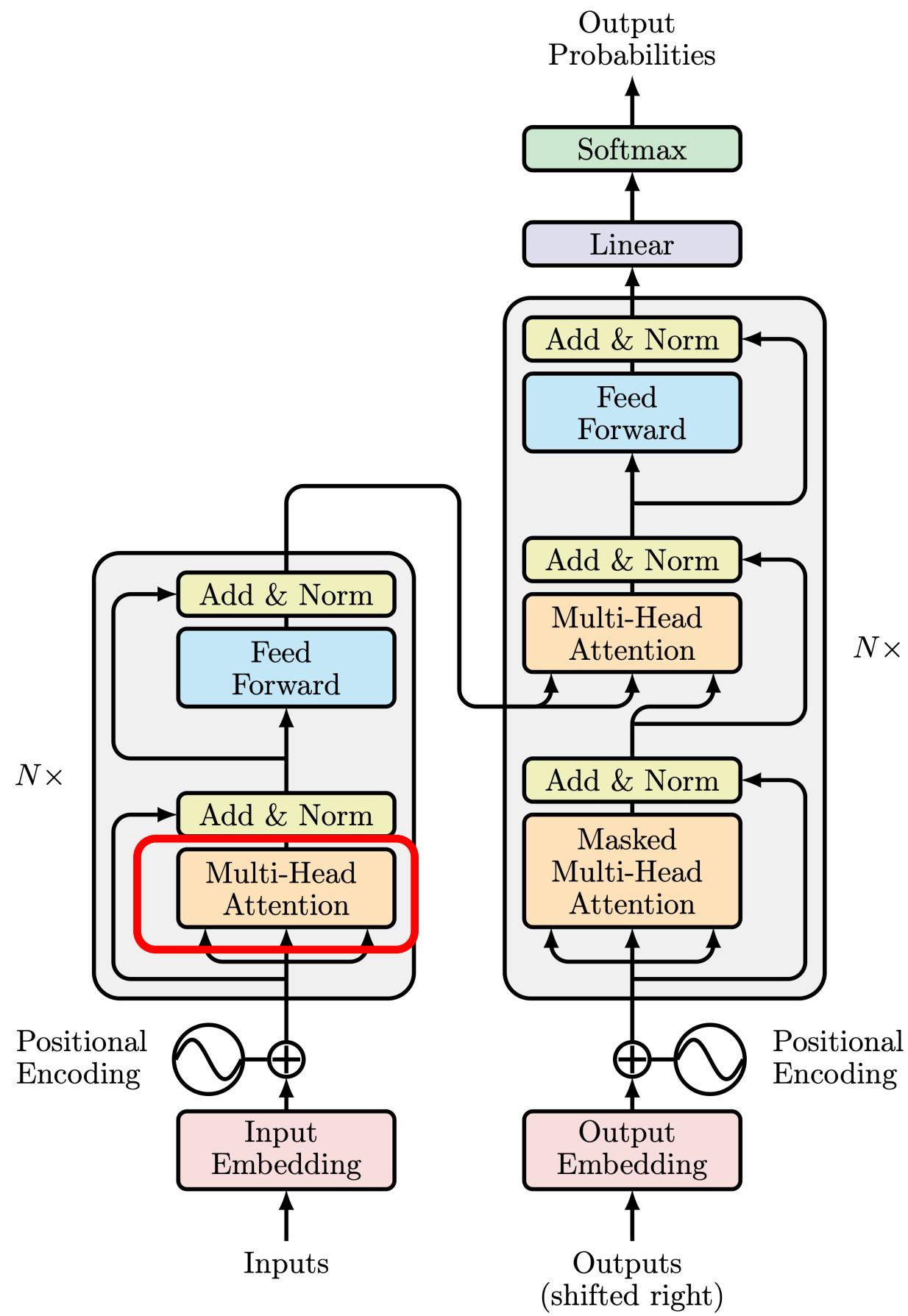
$$\vec{h}'_i = \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right)$$

$$\alpha_{ij} = \frac{\exp \left(\text{LeakyReLU} \left(\vec{a}^T [\mathbf{W} \vec{h}_i \| \mathbf{W} \vec{h}_j] \right) \right)}{\sum_{k \in \mathcal{N}_i} \exp \left(\text{LeakyReLU} \left(\vec{a}^T [\mathbf{W} \vec{h}_i \| \mathbf{W} \vec{h}_k] \right) \right)}$$

Multi-graph (k-type edges), a.k.a., can think of it as K attention heads

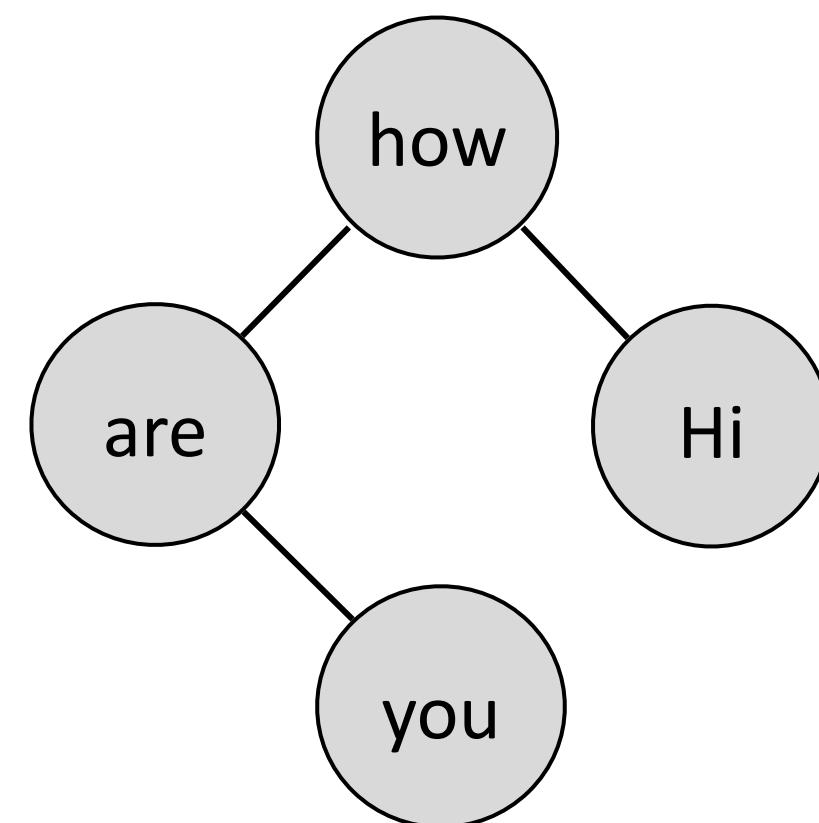
GNN Relationship to Transformers

- 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!



GNN Relationship to Transformers

- Apply the adjacency matrix as a mask to the attention and renormalize it, is like Graph Attention Networks (GAT) [10]
- Encoder 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

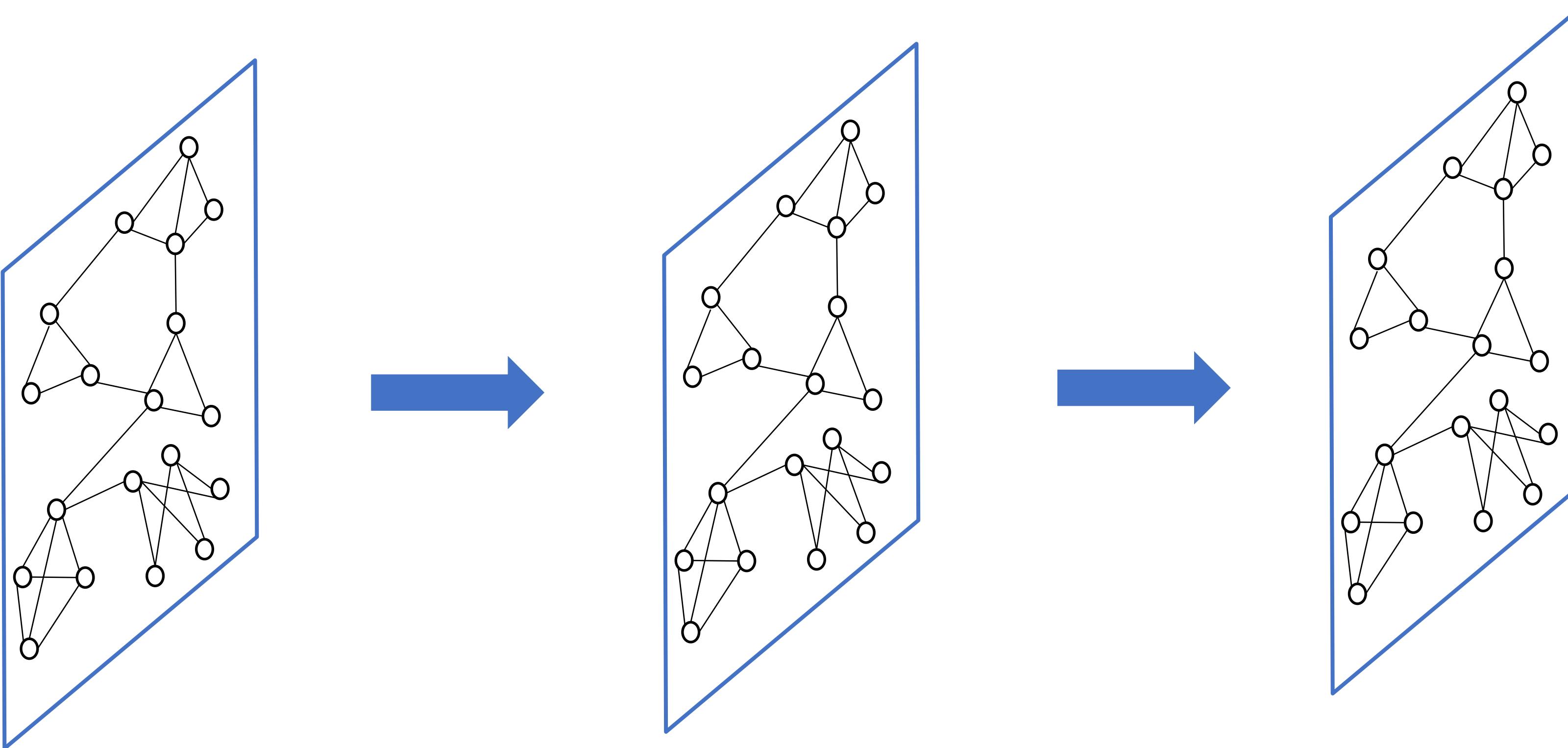
Softmax($\begin{matrix} \cdot & \cdot & \cdot & \cdot \end{matrix}$) =

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

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

So far ... we mainly focused on **graph filtering**

Goal: Refine node (or possibly) edge feature

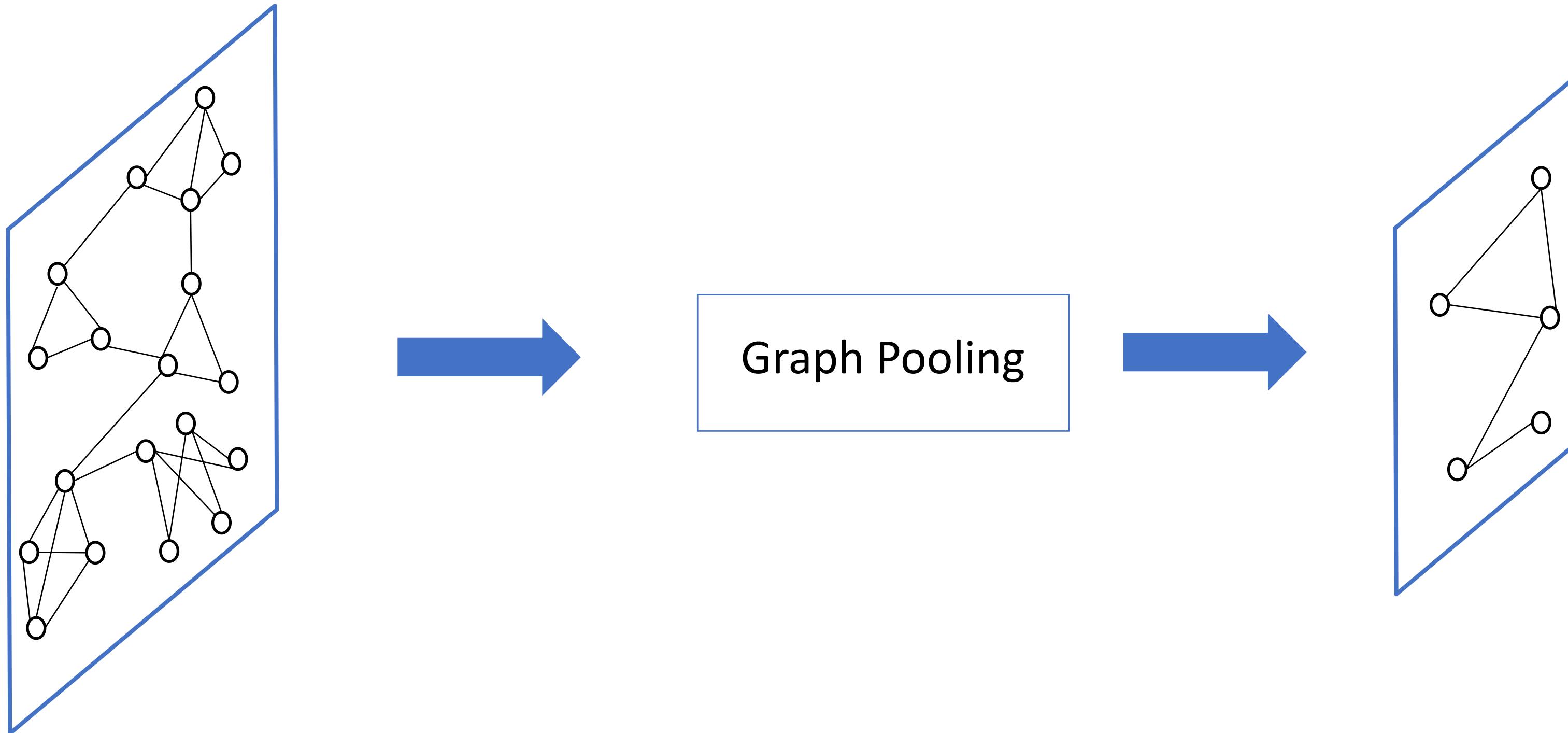


$$\mathbf{A} \in \{0, 1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A} \in \{0, 1\}^{n \times n}, \mathbf{X}_f \in \mathbb{R}^{n \times d}$$

We can also do **graph pooling**

Goal: Generate a smaller graph that captures original graphs information



$$\mathbf{A} \in \{0, 1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0, 1\}^{n_p \times n_p}, \mathbf{X}_p \in \mathbb{R}^{n \times d}, n_p < n$$

Global pooling vs. Graph Pooling

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

Global pooling vs. Graph Pooling

Issue: Global pooling over a (large) graph will lose information

Toy example: we use 1-dim node embeddings

Node embeddings for $G1 = \{-1, -2, 0, 1, 2\}$

Node embeddings for $G2 = \{-10, -20, 0, 10, 20\}$

Clearly $G1$ and $G2$ (have very different node embeddings)

Global pooling vs. Graph Pooling

Issue: Global pooling over a (large) graph will lose information

Toy example: we use 1-dim node embeddings

Node embeddings for $G1 = \{-1, -2, 0, 1, 2\}$

Node embeddings for $G2 = \{-10, -20, 0, 10, 20\}$

Clearly $G1$ and $G2$ (have very different node embeddings)

If we do global ReLU(Sum()) pooling:

Prediction for $G1 = 0$

Prediction for $G2 = 0$

We cannot differentiate $G1$ and $G2$

Global pooling vs. Graph Pooling

A solution: Let's aggregate all the node embeddings **hierarchically**

Toy example: We will aggregate via $\text{ReLU}(\text{Sum}())$. We first separately aggregate the first 2 nodes and last 3 nodes. Then we aggregate again to make the final prediction.

G_1 node embeddings: $\{-1, -2, 0, 1, 2\}$

Round 1: $\text{ReLU}(\text{Sum}(\{-1, -2\})) = 0$, $\text{ReLU}(\text{Sum}(\{0, 1, 2\})) = 3$

Round 2: $\text{ReLU}(\text{Sum}(\{0, 3\})) = 3$

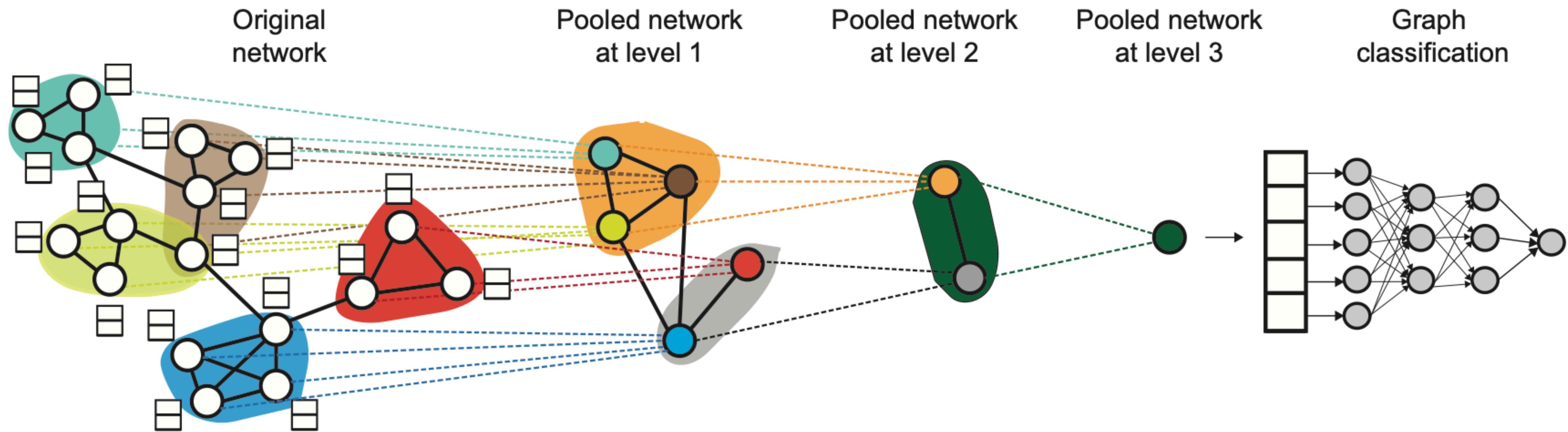
G_2 node embeddings: $\{-10, -20, 0, 10, 20\}$

Round 1: $\text{ReLU}(\text{Sum}(\{-10, -20\})) = 0$, $\text{ReLU}(\text{Sum}(\{0, 10, 20\})) = 30$

Round 2: $\text{ReLU}(\text{Sum}(\{0, 30\})) = 30$

Now we can tell the difference!

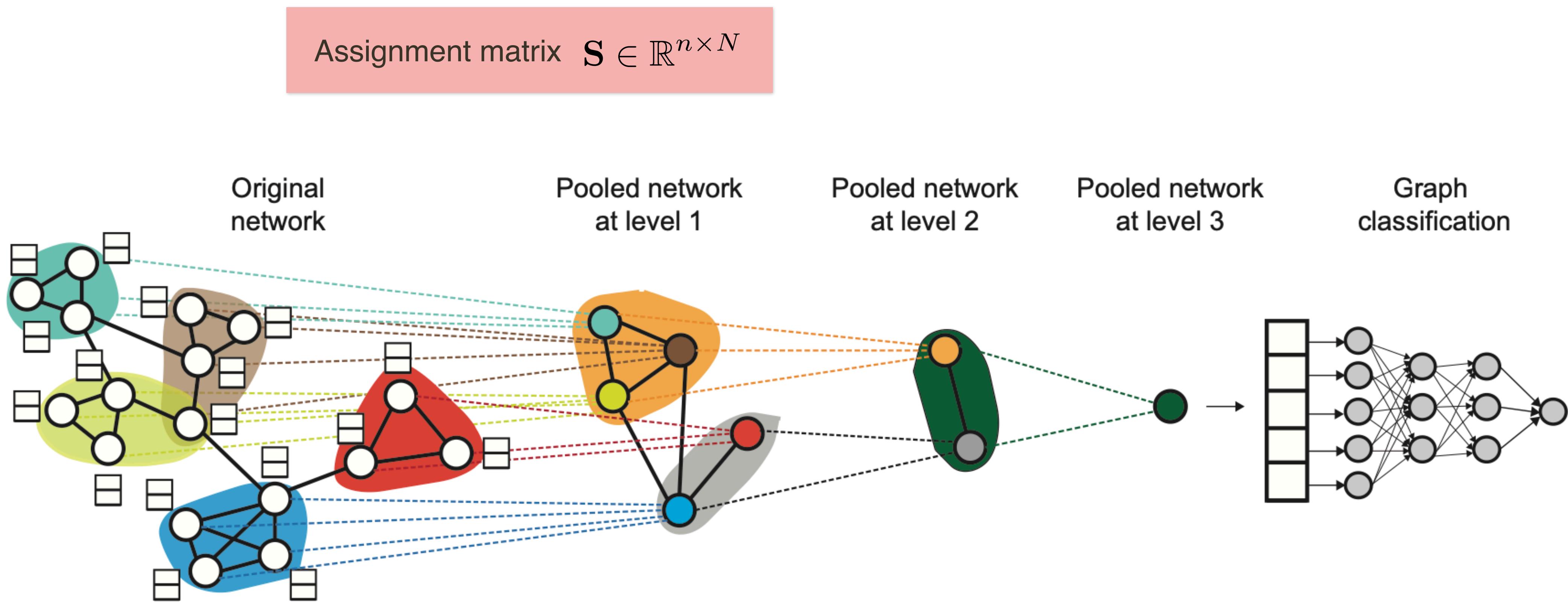
DiffPool – Differentiable Graph Pooling



Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$

Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$

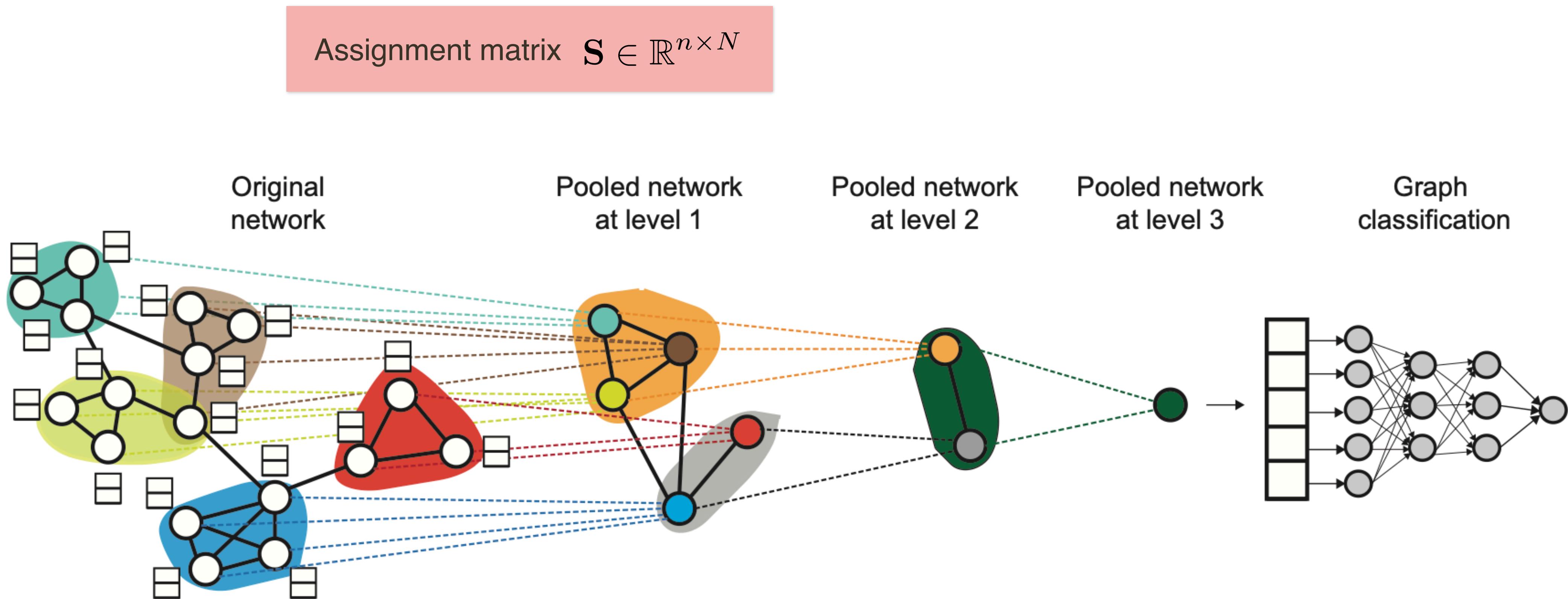
DiffPool – Differentiable Graph Pooling



Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$

Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$

DiffPool – Differentiable Graph Pooling



Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$

Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$

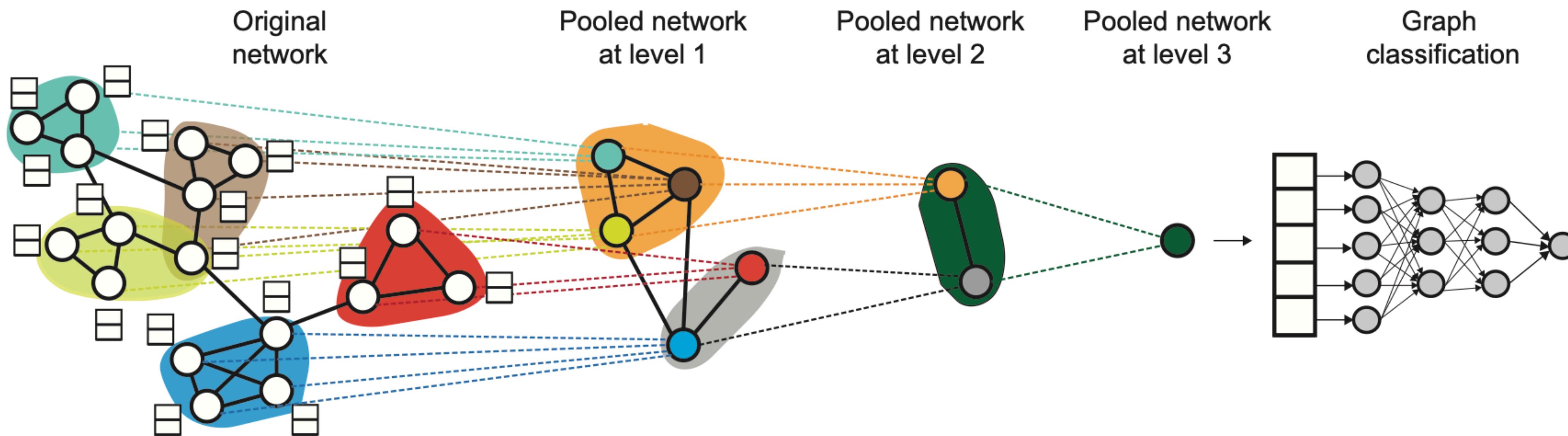
Coarsened adjacent matrix $\mathbf{A}' = \mathbf{S}\mathbf{A}\mathbf{S}^T \in \mathbb{R}^{n \times n}$

Coarsened feature matrix $\mathbf{X}' = \mathbf{S}\mathbf{X} \in \mathbb{R}^{n \times F}$

DiffPool – Differentiable Graph Pooling

Assignment matrix $\mathbf{S} \in \mathbb{R}^{n \times N}$

$$\mathbf{S} = \text{Softmax}(\mathbf{W} \cdot GNN_2(\mathbf{X}, \mathbf{A})); \mathbf{W} \in \mathbb{R}^{n \times N}$$



Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$

Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$

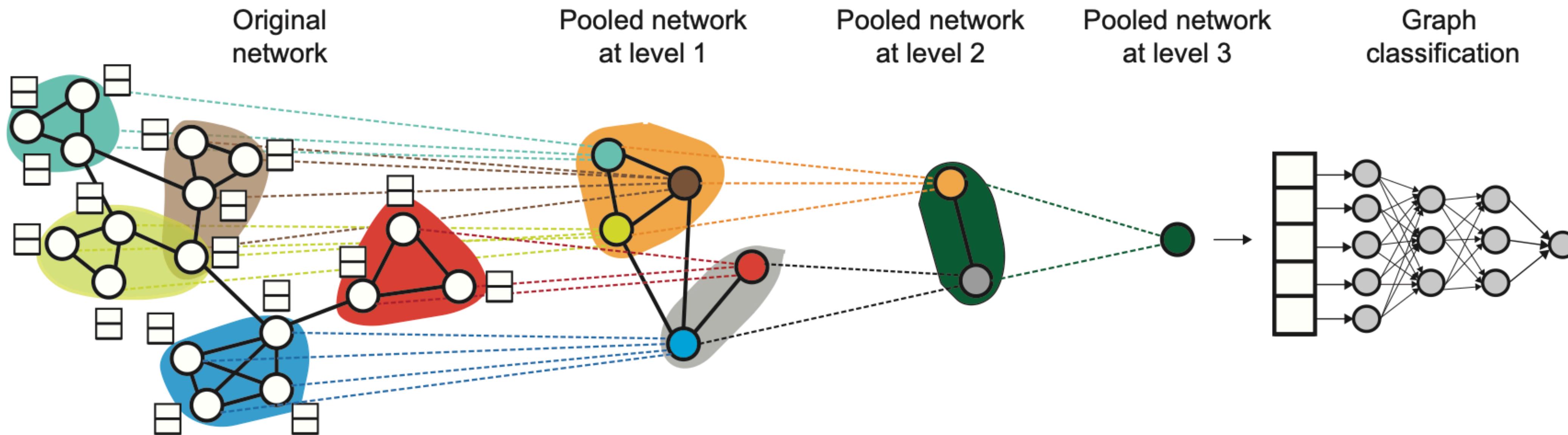
Coarsened adjacent matrix $\mathbf{A}' = \mathbf{S}\mathbf{A}\mathbf{S}^T \in \mathbb{R}^{n \times n}$

Coarsened feature matrix $\mathbf{X}' = \mathbf{S}\mathbf{X} \in \mathbb{R}^{n \times F}$

DiffPool – Differentiable Graph Pooling

Assignment matrix $\mathbf{S} \in \mathbb{R}^{n \times N}$

$$\mathbf{S} = \text{Softmax}(\mathbf{W} \cdot GNN_2(\mathbf{X}, \mathbf{A})); \mathbf{W} \in \mathbb{R}^{n \times N}$$



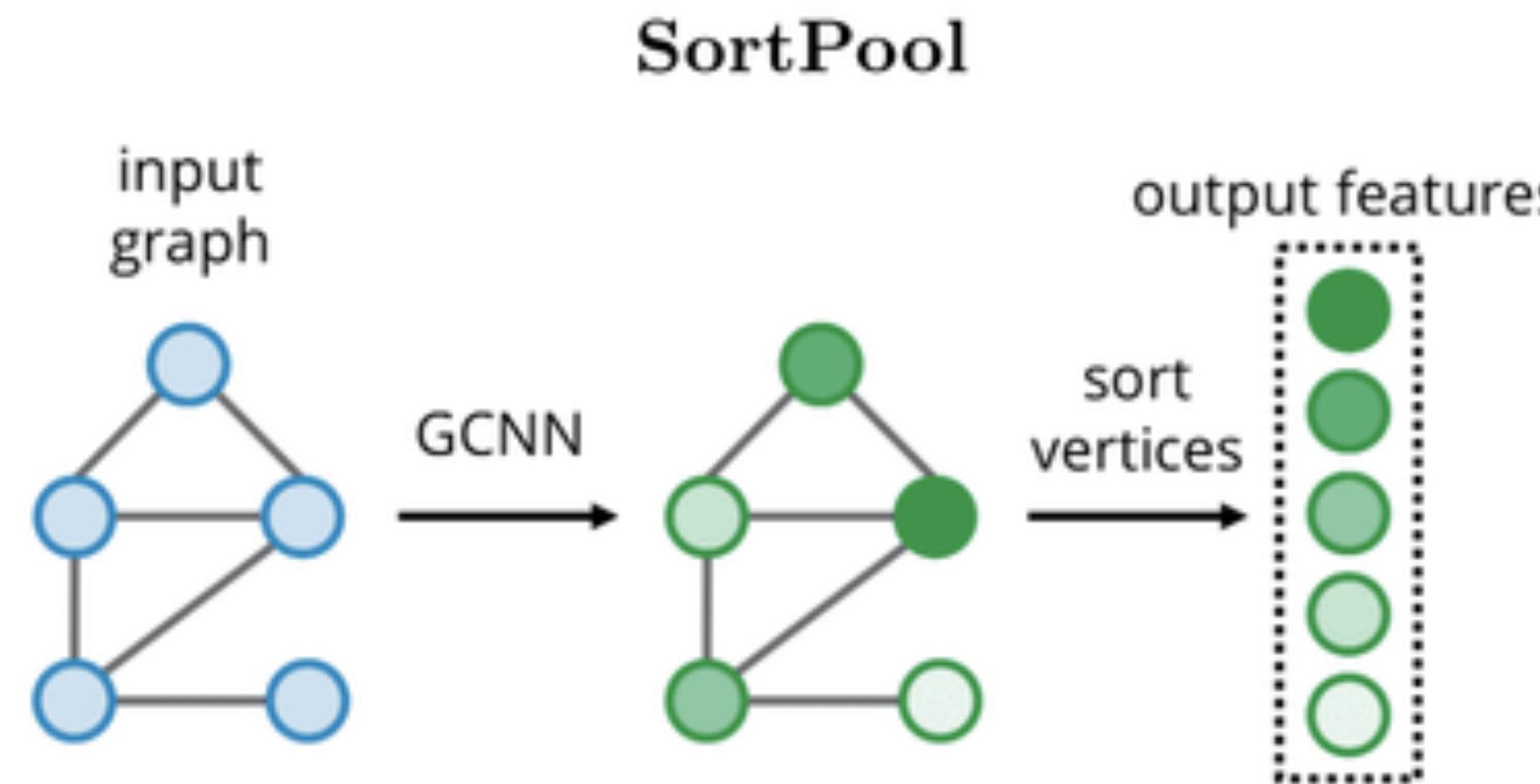
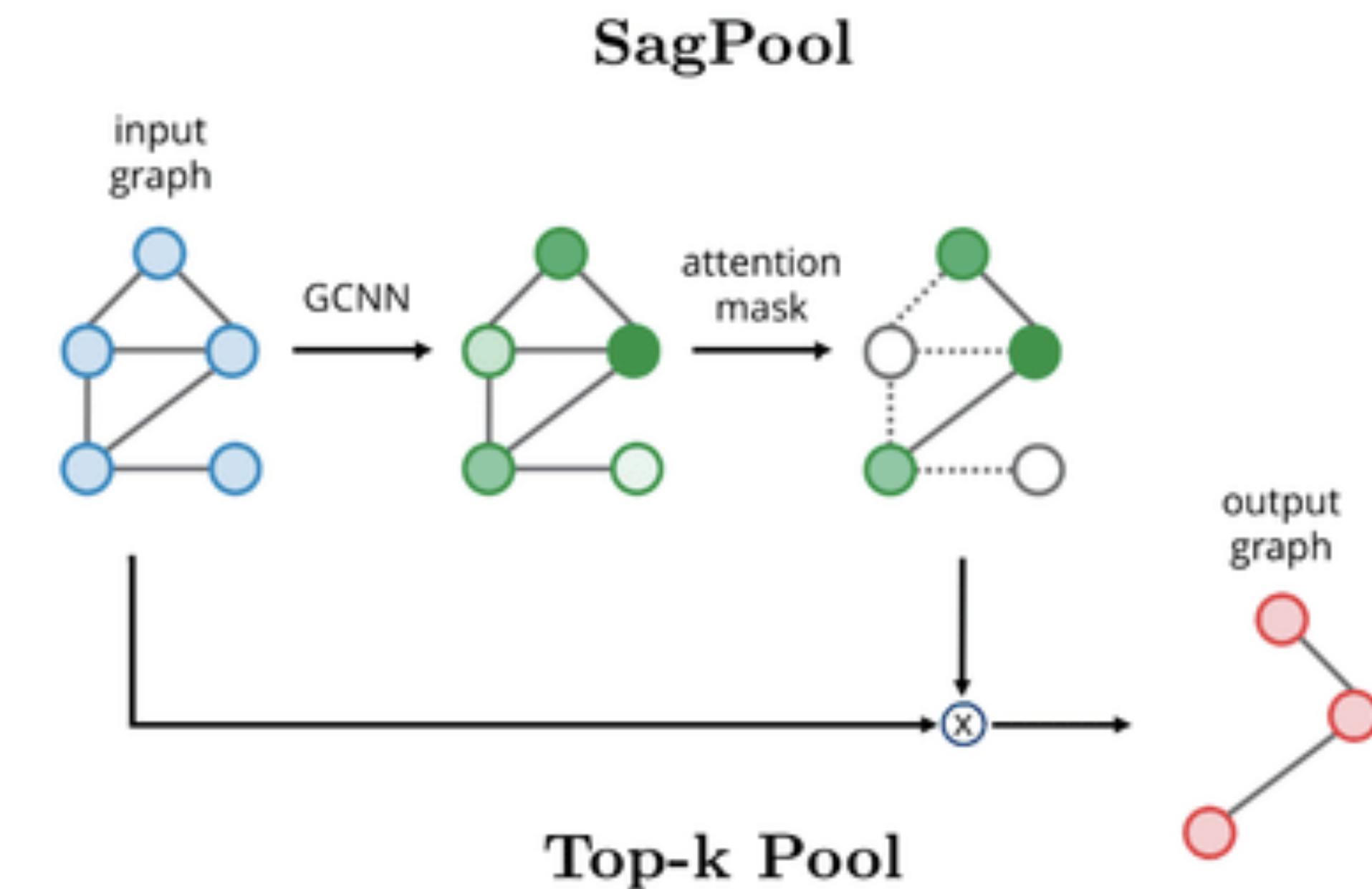
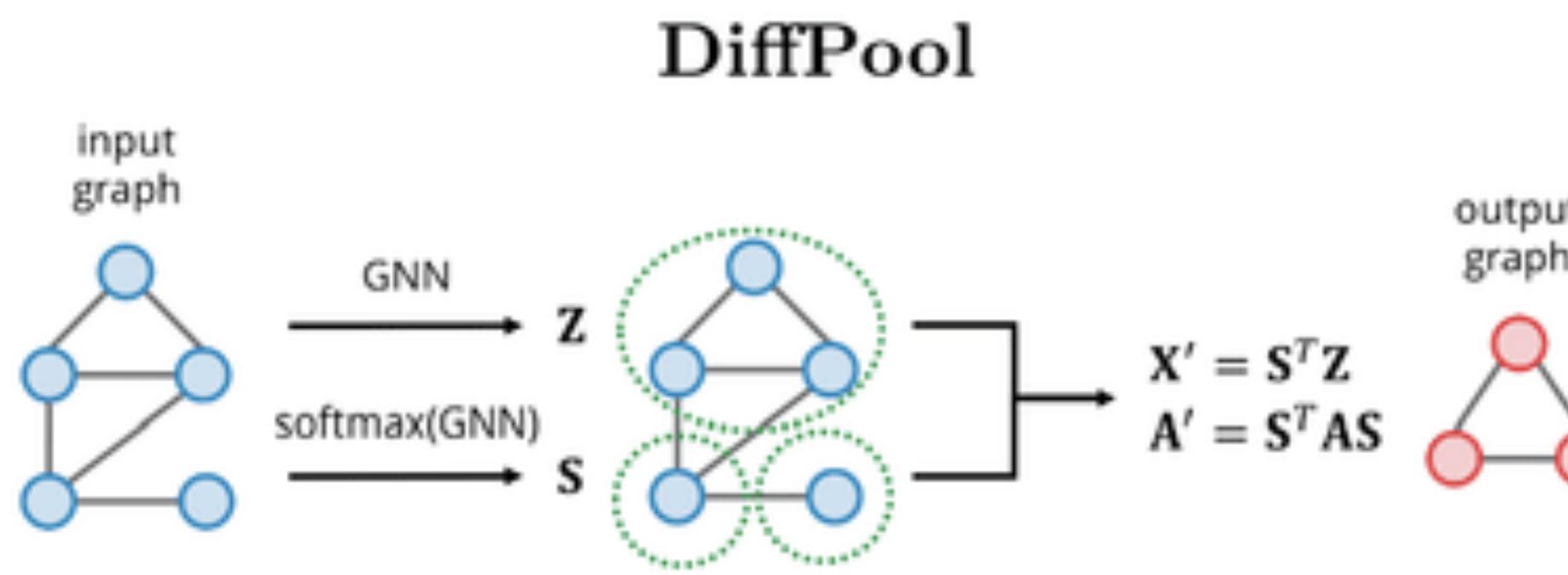
Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$

Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$

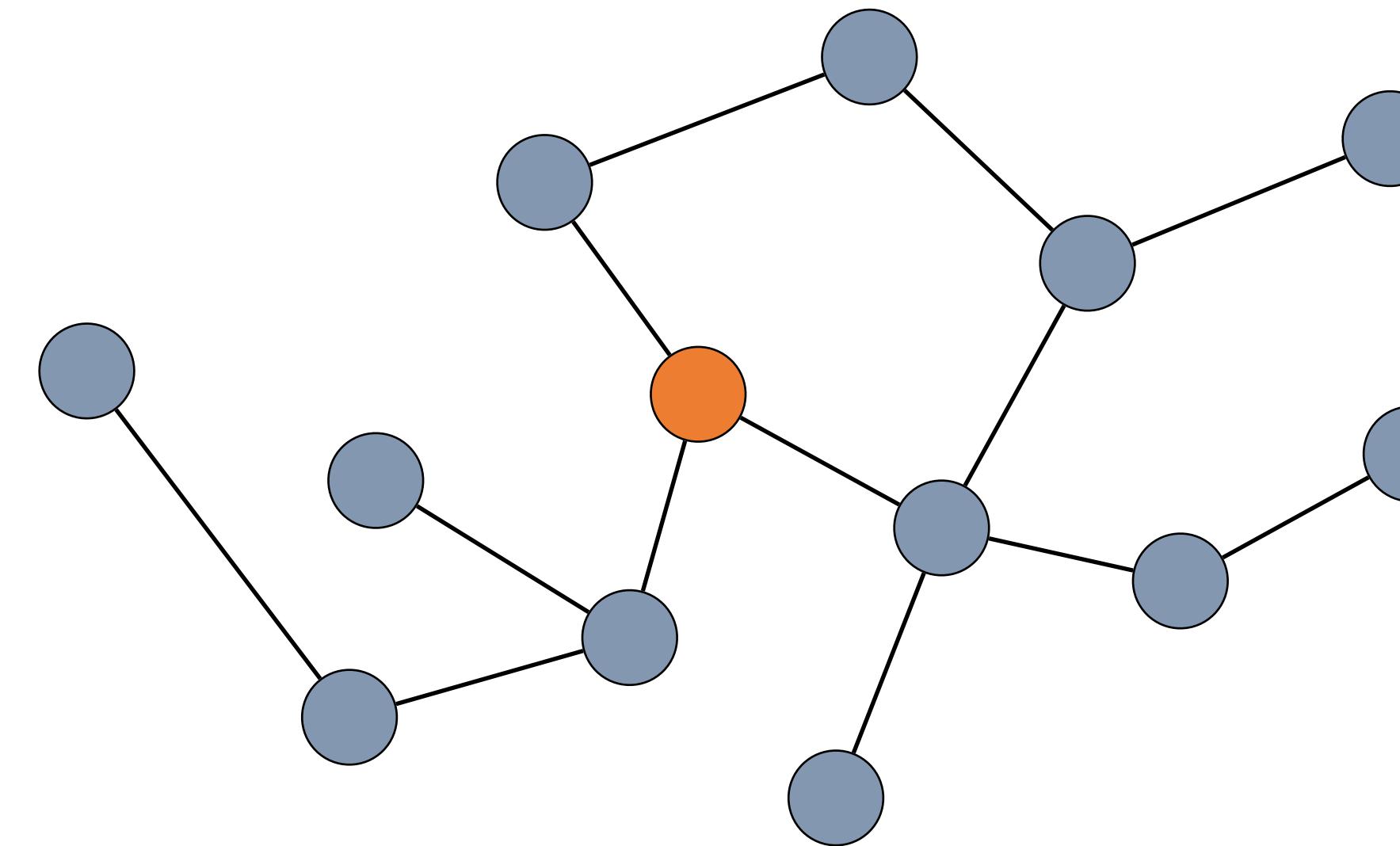
Coarsened adjacent matrix $\mathbf{A}' = \mathbf{S} \mathbf{A} \mathbf{S}^T \in \mathbb{R}^{n \times n}$

Coarsened feature matrix $\mathbf{X}' = \mathbf{S} \cdot GNN_1(\mathbf{X}, \mathbf{A})$

Graph Pooling

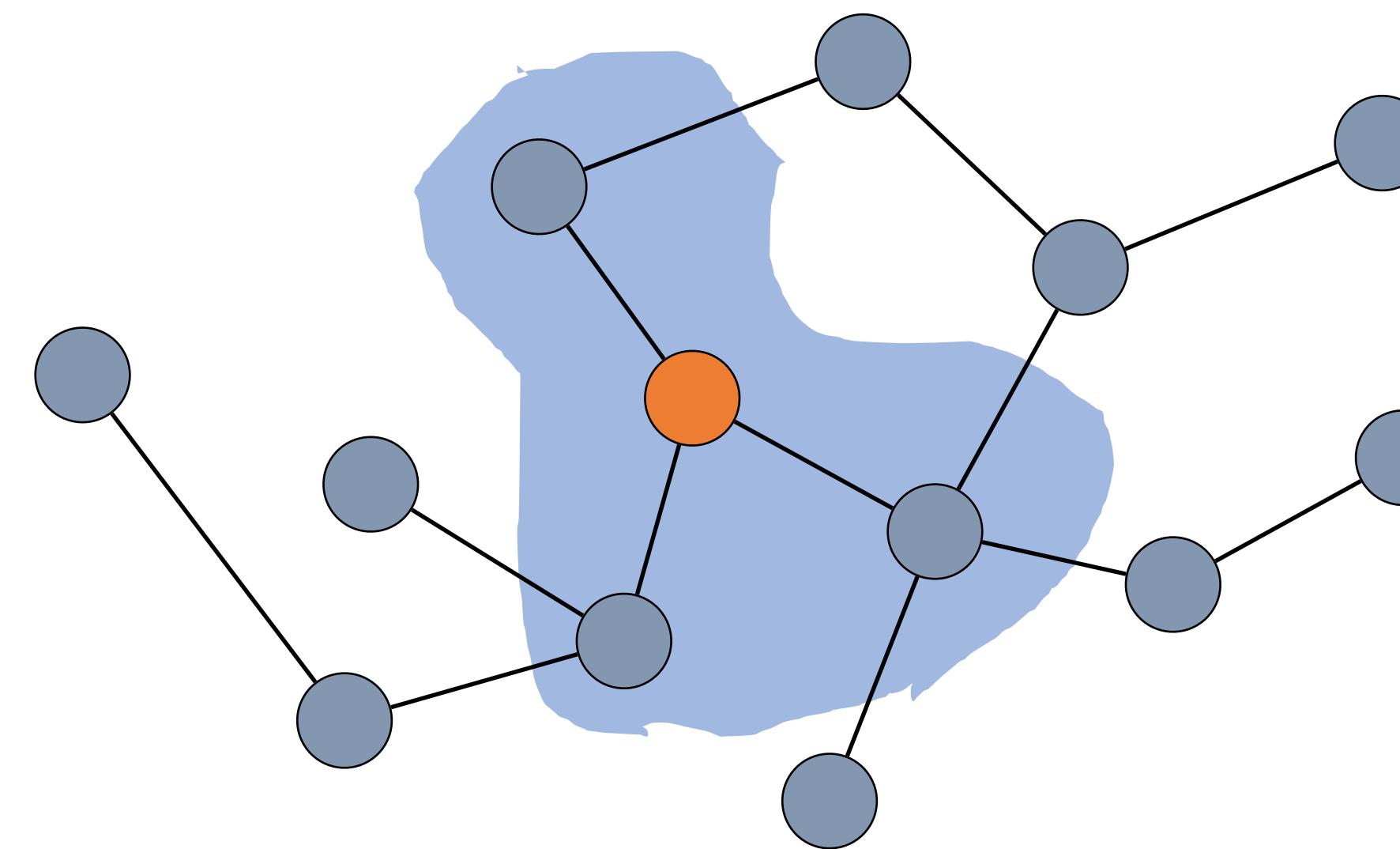


Deeper GNNs (more layers)



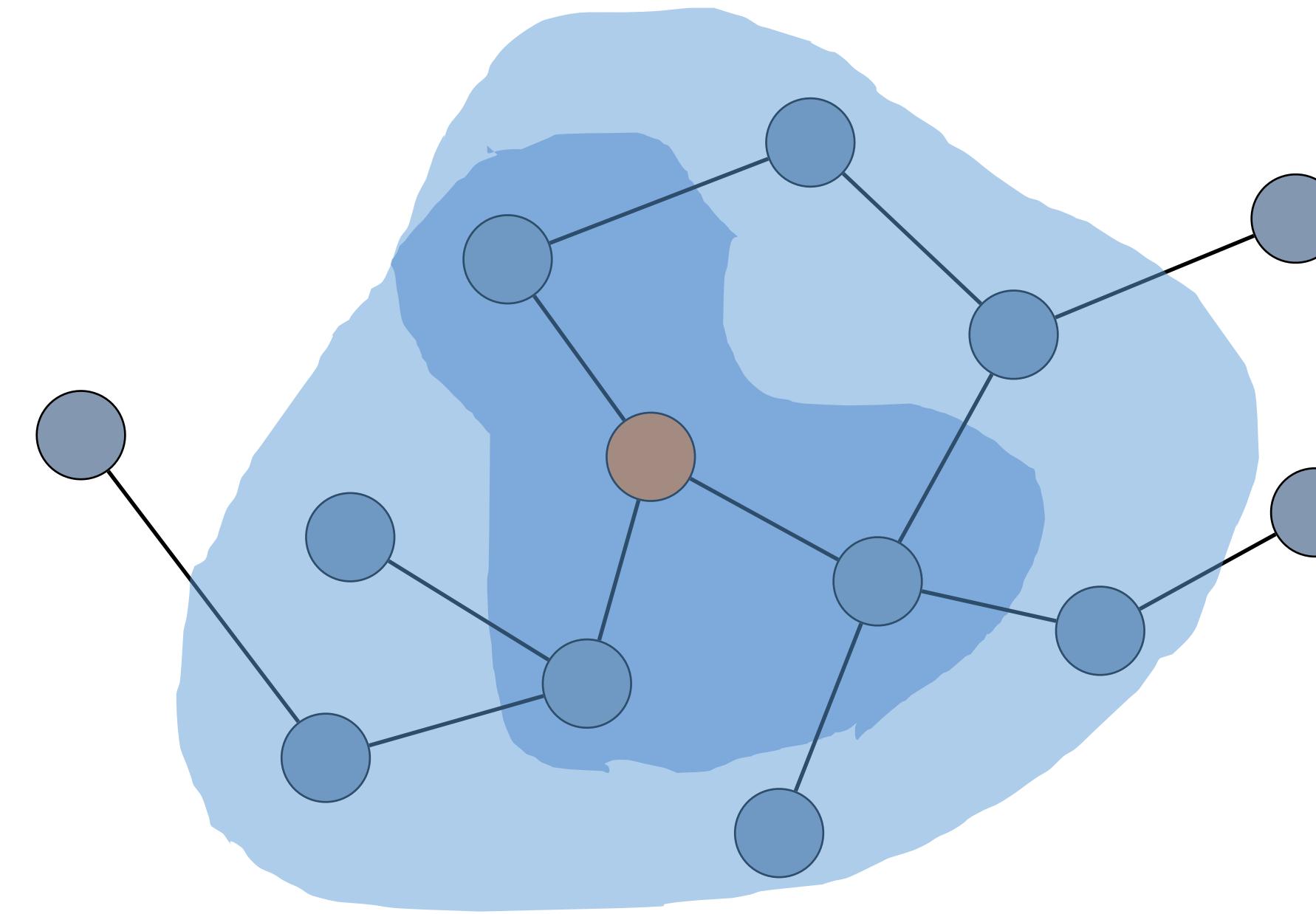
Depth in GNNs **increases receptive field** and **model capacity**

Deeper GNNs (more layers)



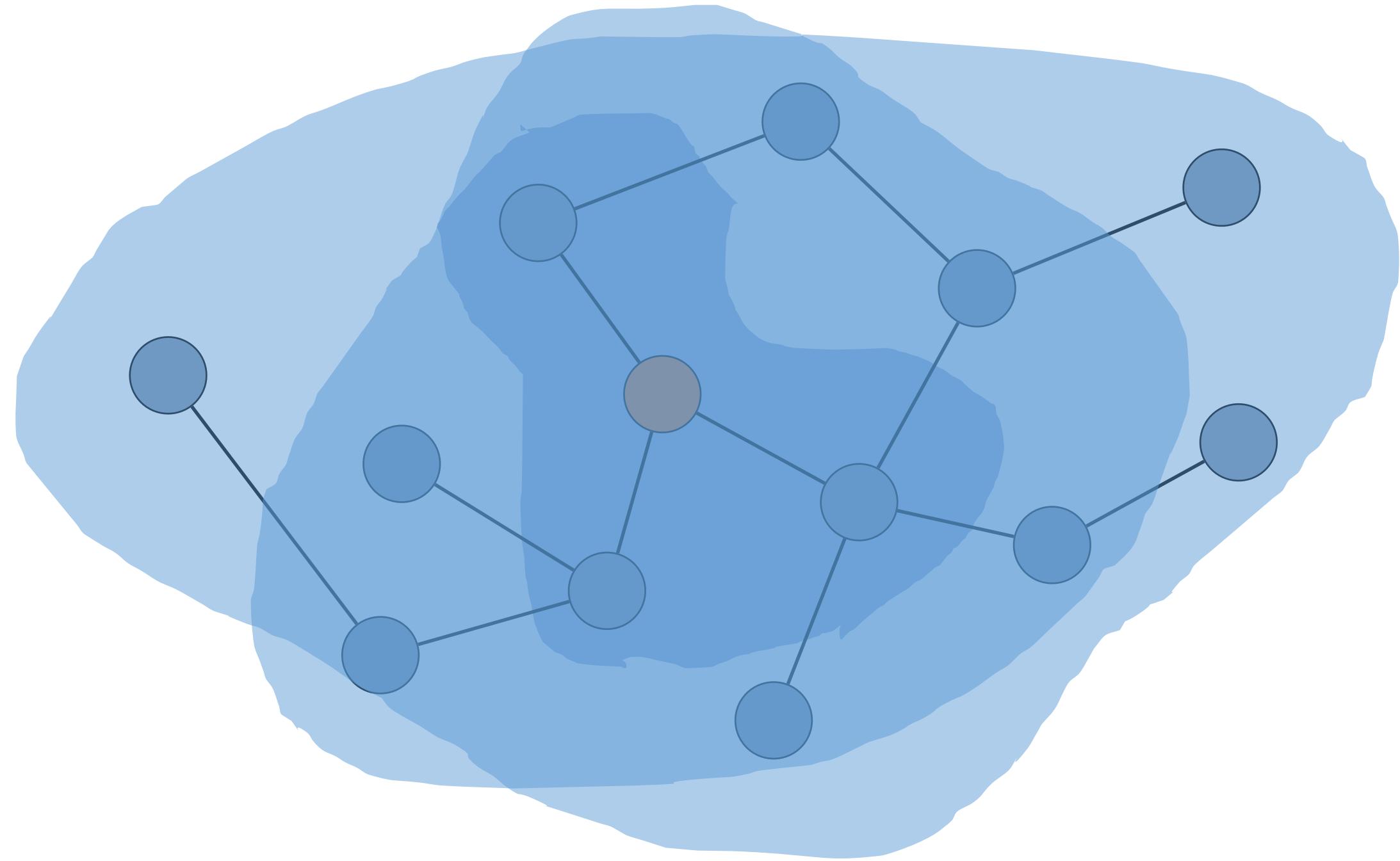
Depth in GNNs **increases receptive field** and **model capacity**

Deeper GNNs (more layers)



Depth in GNNs **increases receptive field** and **model capacity**

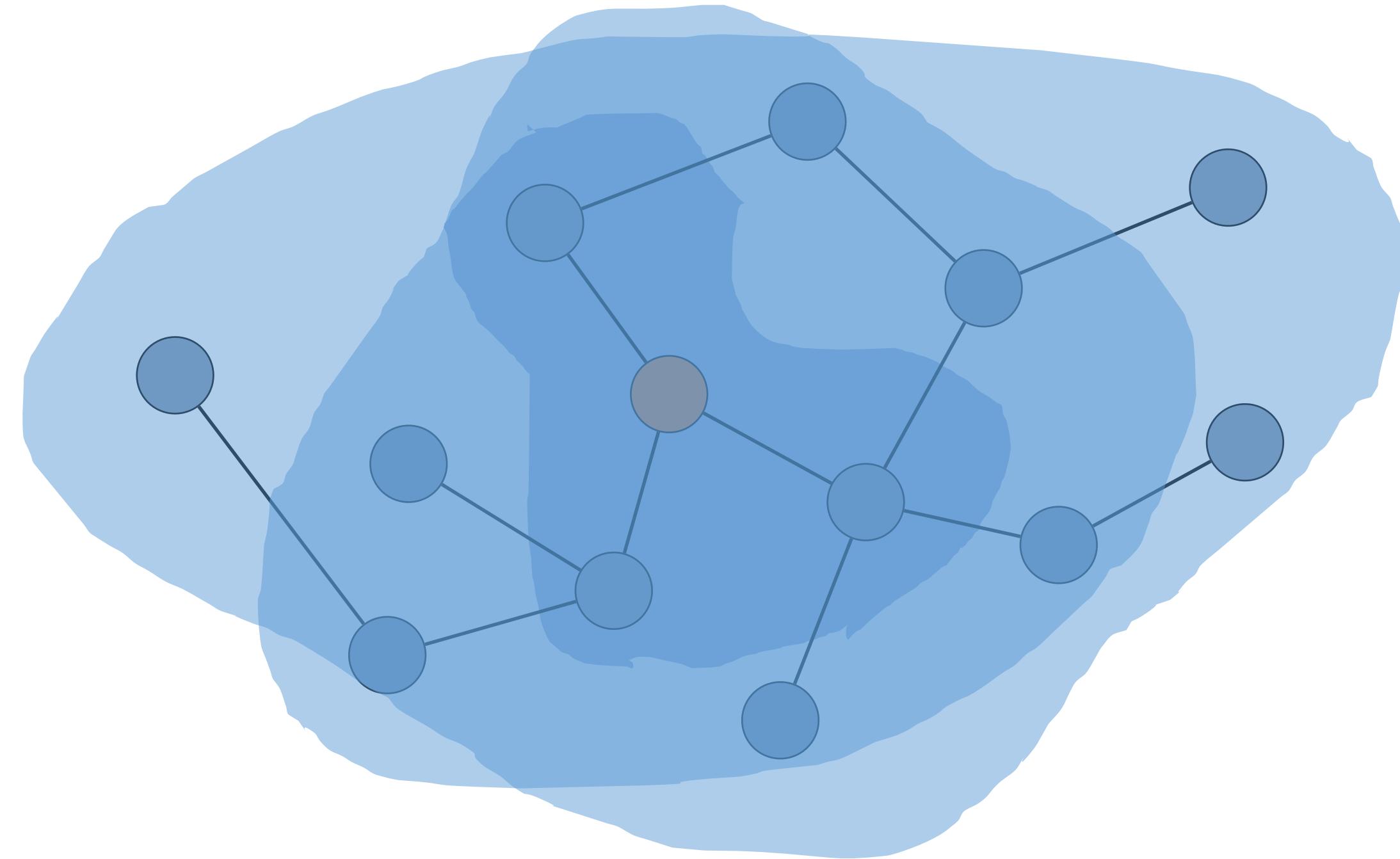
Deeper GNNs (more layers)



Depth in GNNs **increases receptive field** and **model capacity**

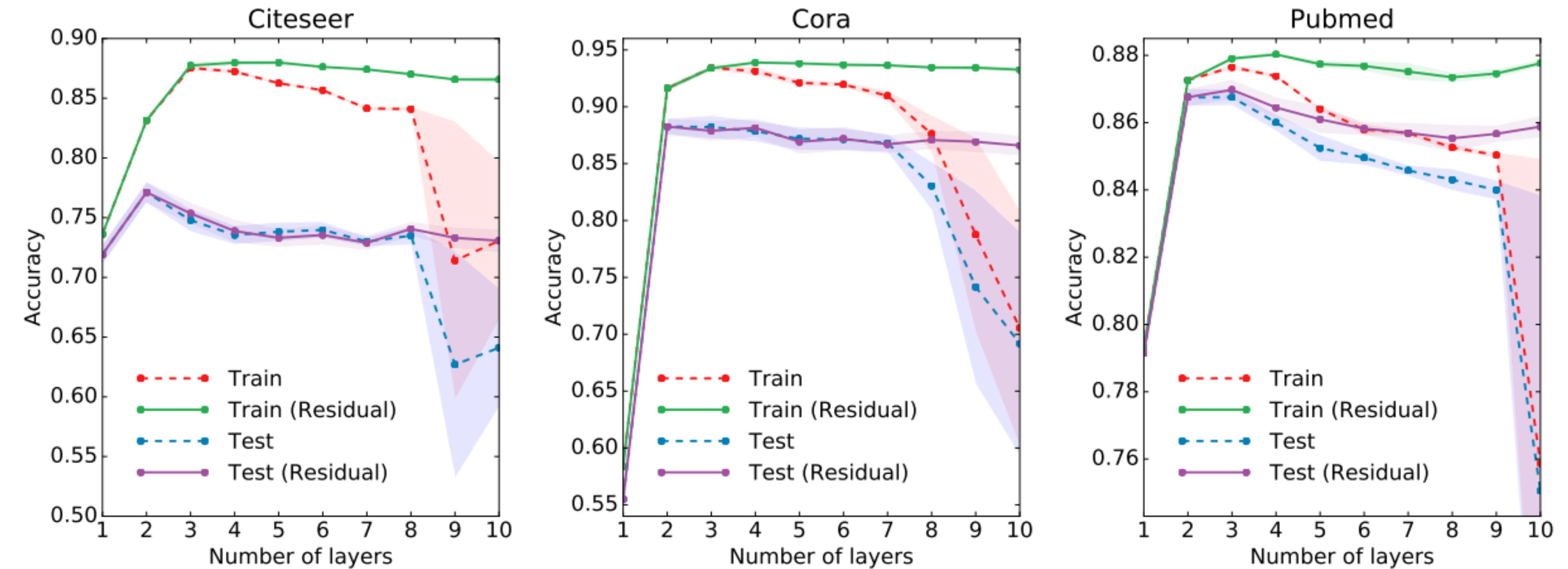
Deeper GNNs (more layers)

However, unlike in CNNs, deeper GNN models generally do not work well



Depth in GNNs **increases receptive field** and **model capacity**

Deeper GNNs (more layers)



Deeper GNNs (more layers) – Oversmoothing

Latent node vectors get closer to each other as the number of GCN layers increases, this makes it difficult to distinguish nodes in deeper GCNs

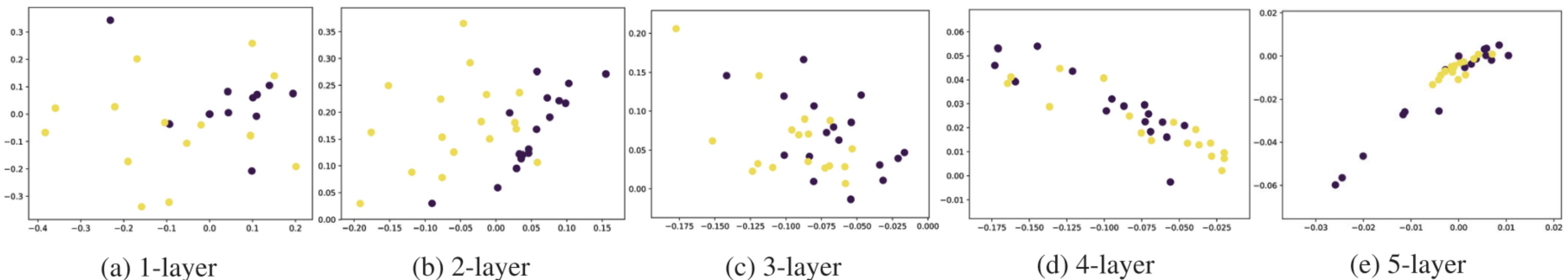


Figure 2: Vertex embeddings of Zachary's karate club network with GCNs with 1,2,3,4,5 layers.

Deeper GNNs (more layers) – Oversmoothing

Recall **GCN** propagation equation:

$$\sigma([\mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}] \mathbf{H} \mathbf{W}_1^{(l)})$$


Lets assume we have no non-linearity

$$\lim_{k \rightarrow \infty} \mathbf{M}^k \mathbf{v} \propto \mathbf{u}_1$$

$$\mathbf{M} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}$$


where \mathbf{u}_1 is an **eigenvector** corresponding to largest **eigenvalue** of \mathbf{M}

GNN Oversmoothing

Oversmoothing is theoretically proven

- Deeper GCN converge to a solution where connected nodes will have similar latent vectors
- Such convergence in GCN happens very quickly (exponential to the depth), regardless of the initial node vectors

Similar results can be derived for other generic “vanilla” GNNs

GNN Oversmoothing

Combining a proper normalizer and a residual node update formulation addresses oversmoothing (in many cases)

- **Normalization** (PairNorm) **PairNorm: Tackling Oversmoothing in GNNs.** ICLR 2020

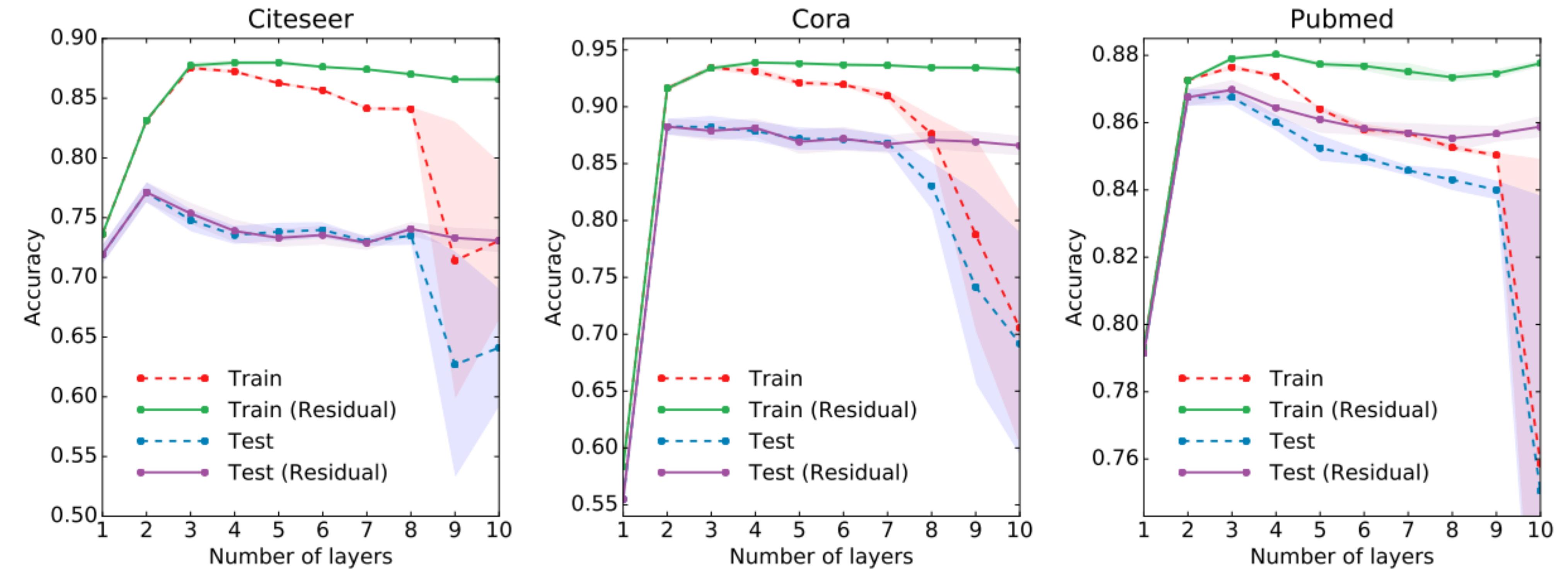
$$\tilde{\mathbf{x}}_i^c = \tilde{\mathbf{x}}_i - \frac{1}{n} \sum_{i=1}^n \tilde{\mathbf{x}}_i \quad (\text{Center})$$

$$\dot{\mathbf{x}}_i = s \cdot \frac{\tilde{\mathbf{x}}_i^c}{\sqrt{\frac{1}{n} \sum_{i=1}^n \|\tilde{\mathbf{x}}_i^c\|_2^2}} = s\sqrt{n} \cdot \frac{\tilde{\mathbf{x}}_i^c}{\sqrt{\|\tilde{\mathbf{X}}^c\|_F^2}} \quad (\text{Scale})$$

- **Residual** update

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

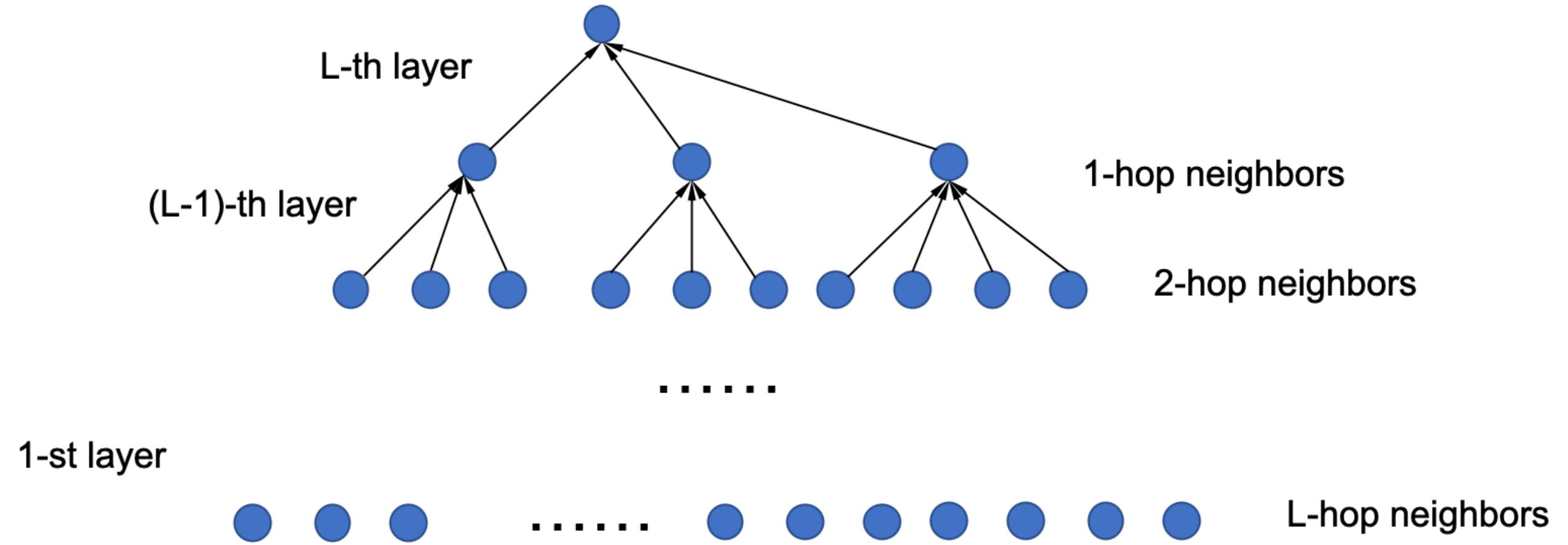
Deeper GNNs (more layers)



GNN Scalability

Node-wise **local neighborhood** sampling

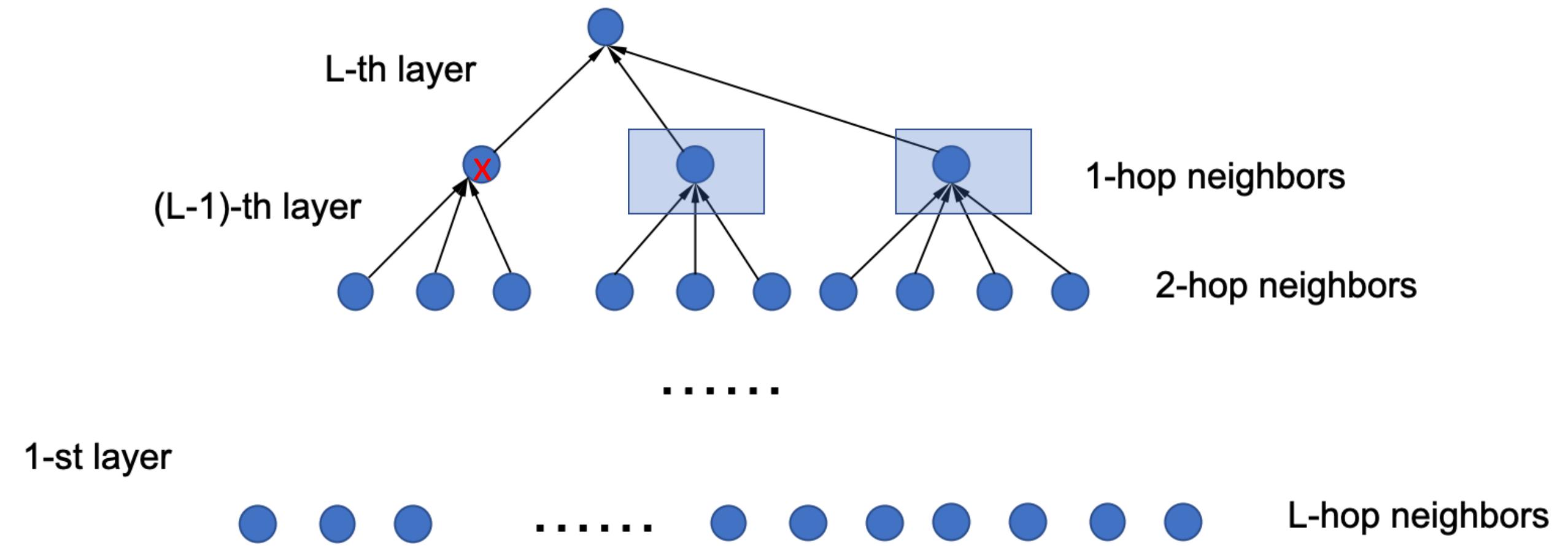
- Sample a set of neighborhoods instead of using the entire local neighborhood



GNN Scalability

Node-wise **local neighborhood** sampling

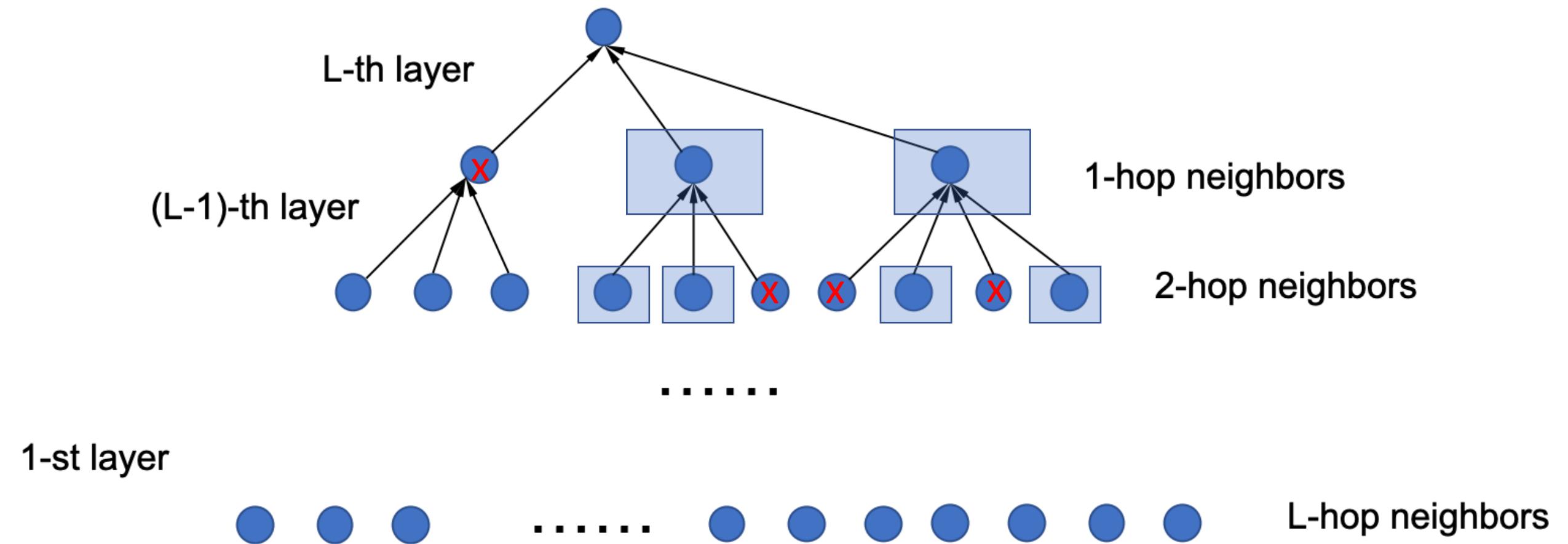
- Sample a set of neighborhoods instead of using the entire local neighborhood



GNN Scalability

Node-wise **local neighborhood** sampling

- Sample a set of neighborhoods instead of using the entire local neighborhood

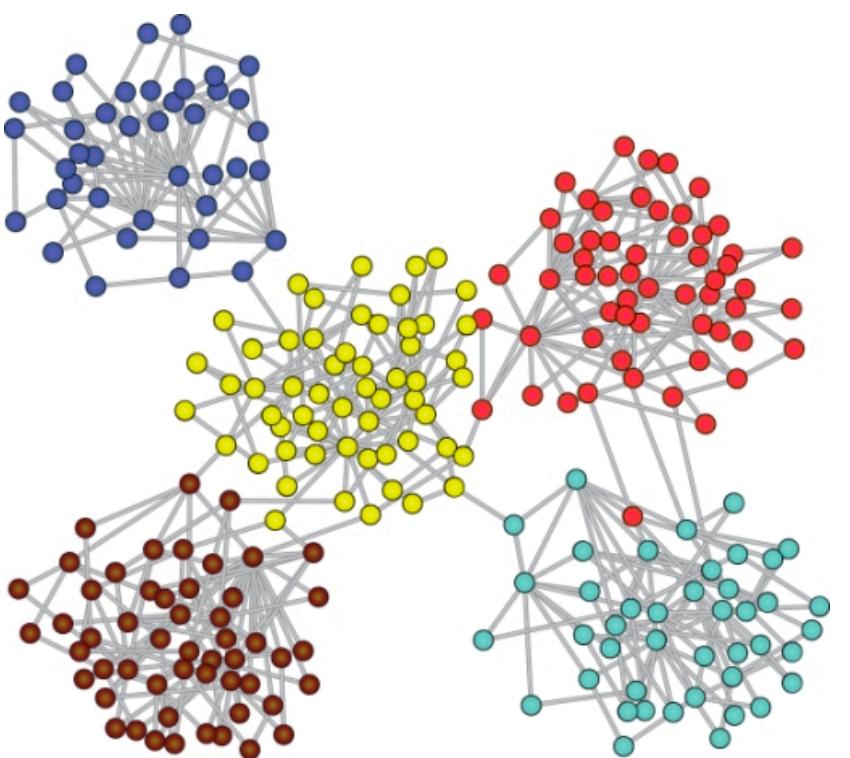


GNN Scalability

Node-wise **local neighborhood** sampling

- Sample a set of neighborhoods instead of using the entire local neighborhood

Subgraph-wise sampling



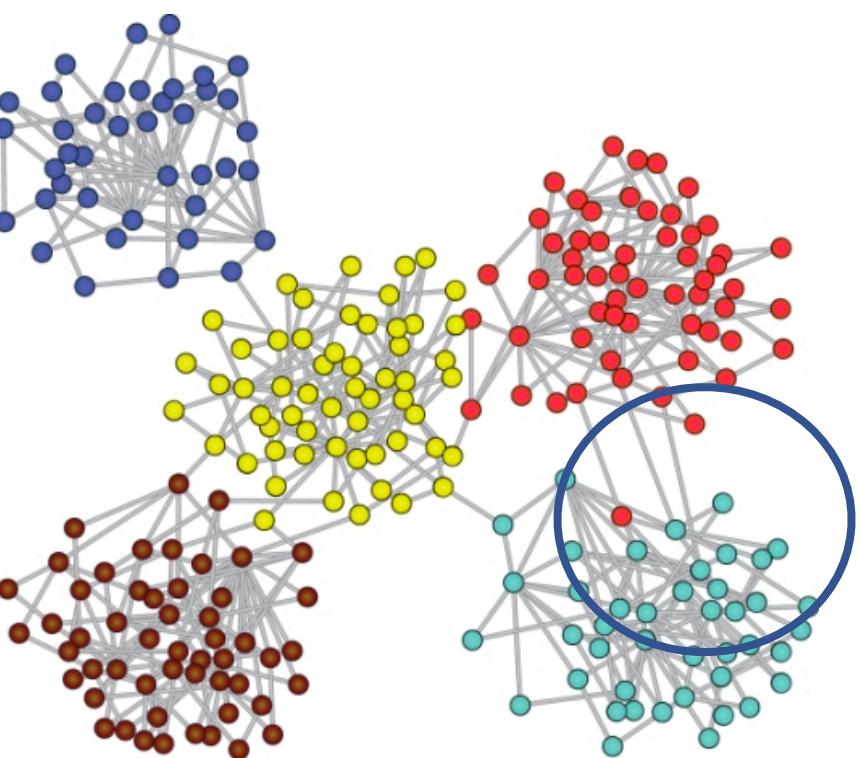
GNN Scalability

Node-wise **local neighborhood** sampling

- Sample a set of neighborhoods instead of using the entire local neighborhood

Subgraph-wise sampling

1. Select a subgraph

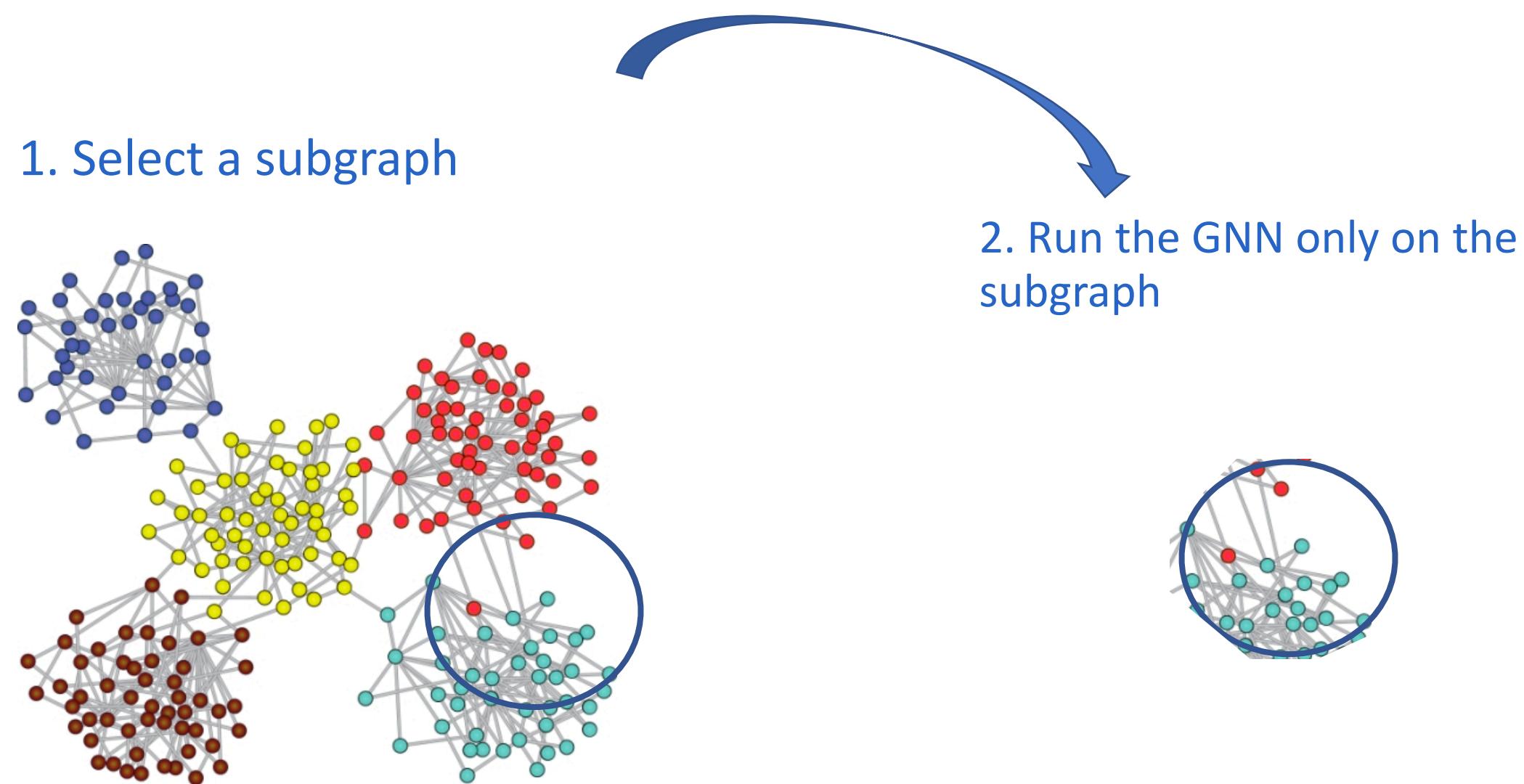


GNN Scalability

Node-wise **local neighborhood** sampling

- Sample a set of neighborhoods instead of using the entire local neighborhood

Subgraph-wise sampling

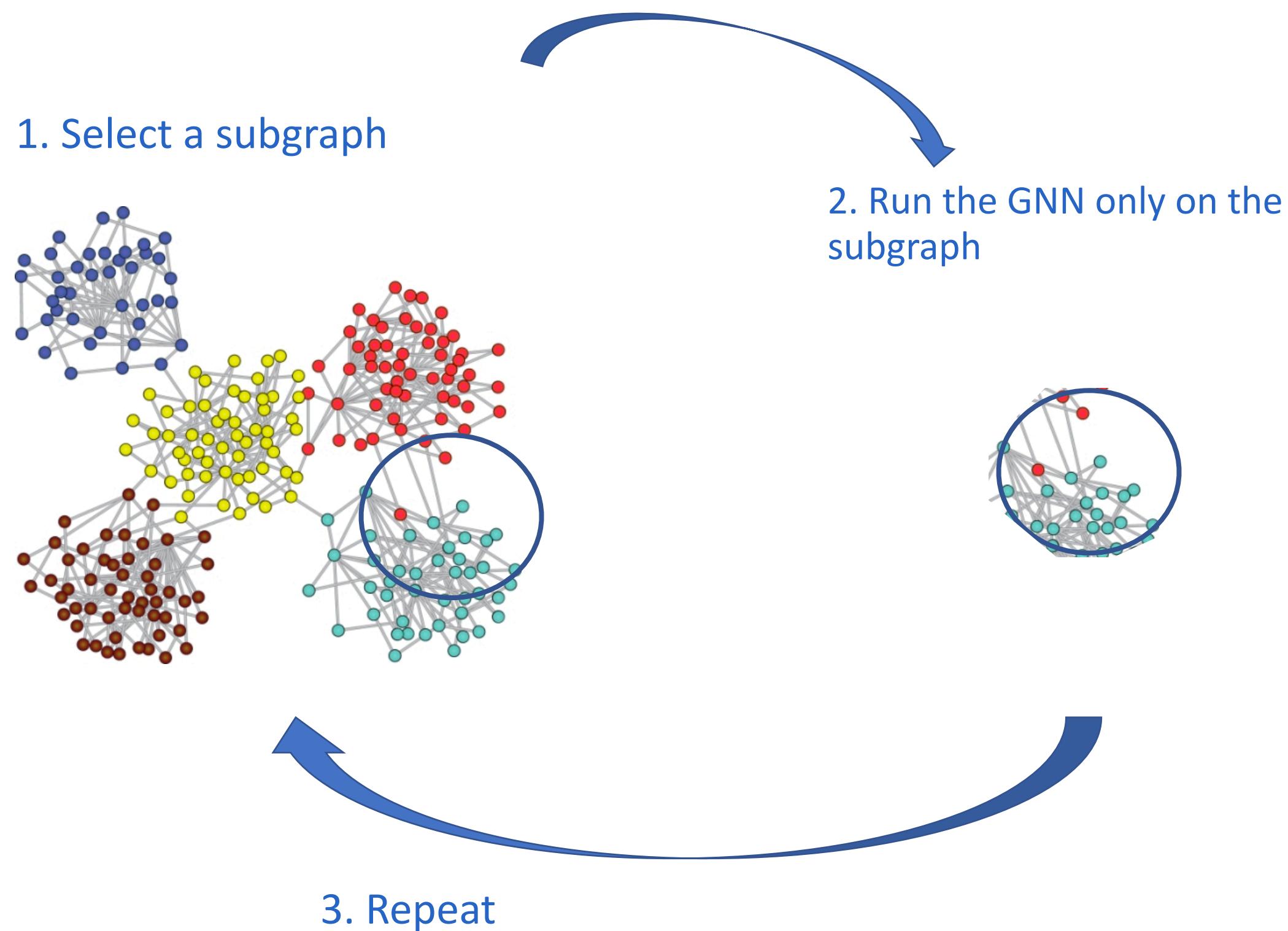


GNN Scalability

Node-wise **local neighborhood** sampling

- Sample a set of neighborhoods instead of using the entire local neighborhood

Subgraph-wise sampling



GNN Scalability

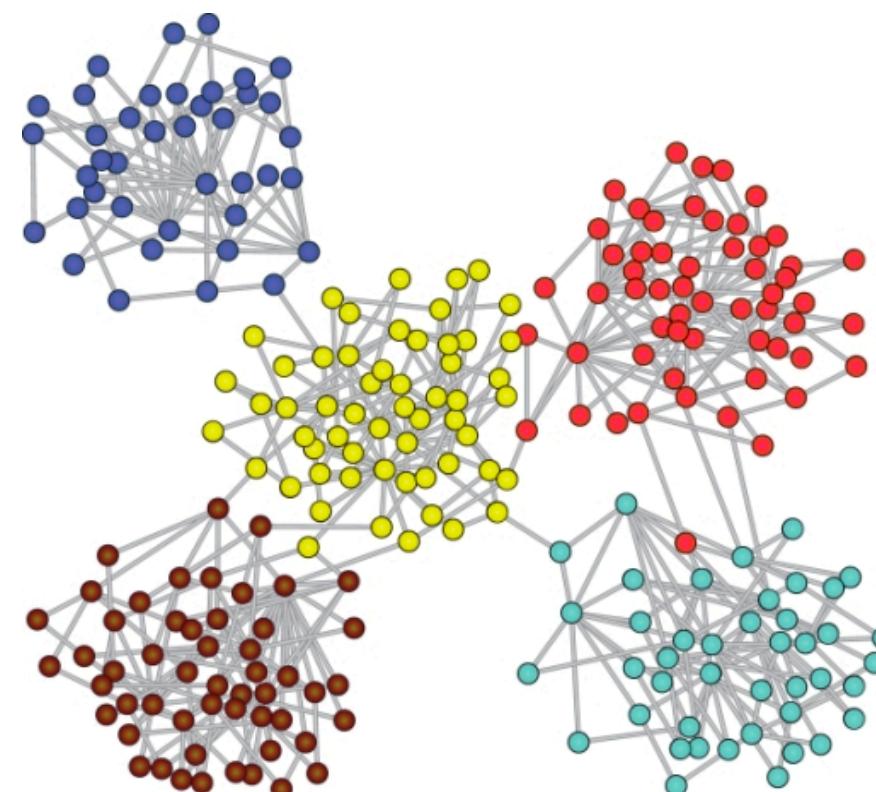
Node-wise **local neighborhood** sampling

- Sample a set of neighborhoods instead of using the entire local neighborhood

Subgraph-wise sampling

Clustering algorithms (**Cluster GCN**)

1. Run a clustering algorithm
2. Select subgraph/cluster
3. Run full-batch GNN on the small subgraph



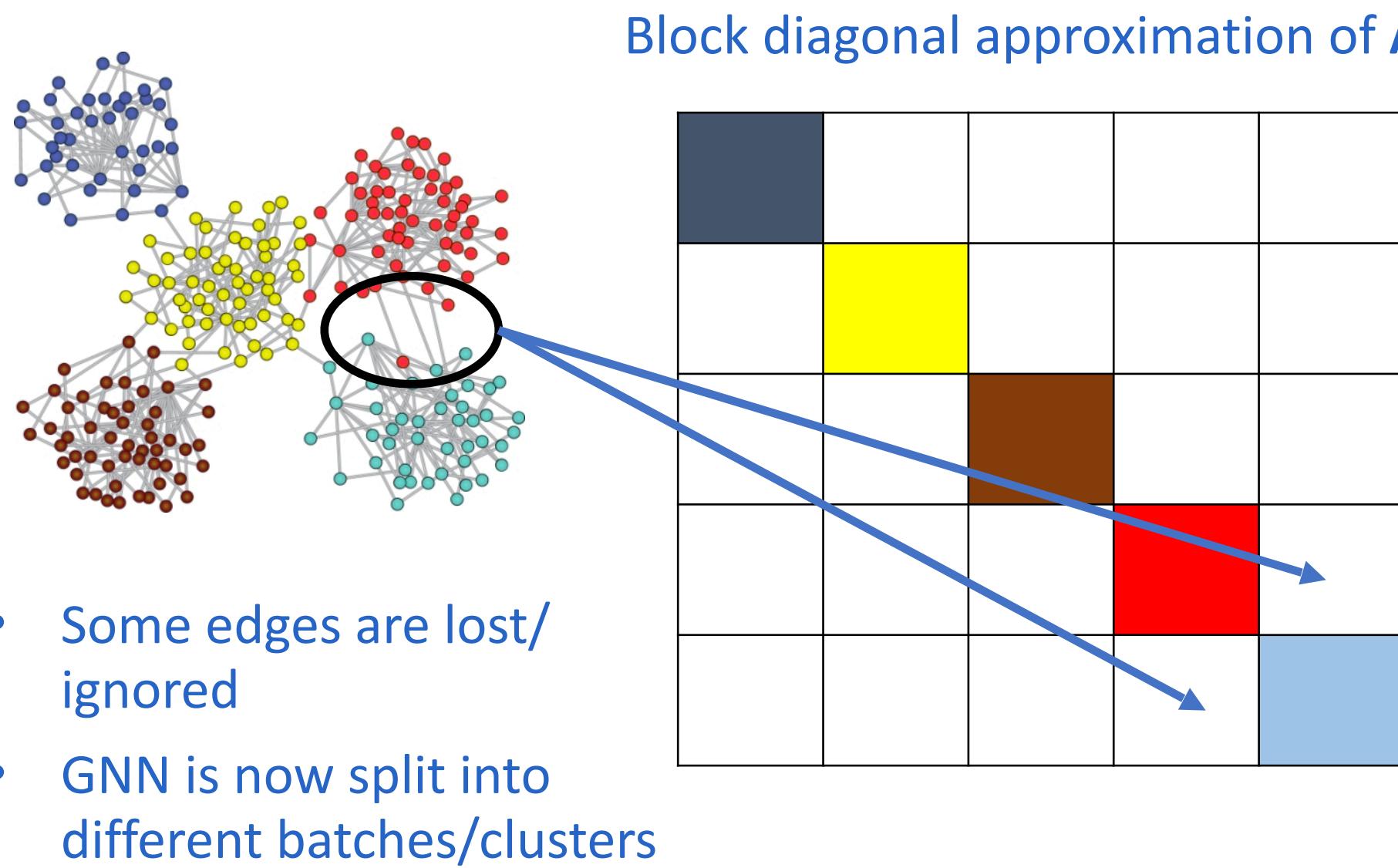
GNN Scalability

Node-wise **local neighborhood** sampling

- Sample a set of neighborhoods instead of using the entire local neighborhood

Subgraph-wise sampling

Clustering algorithms (**Cluster GCN**)



GNN Scalability

Node-wise **local neighborhood** sampling

- Sample a set of neighborhoods instead of using the entire local neighborhood

Subgraph-wise sampling

Clustering algorithms (**Cluster GCN**)

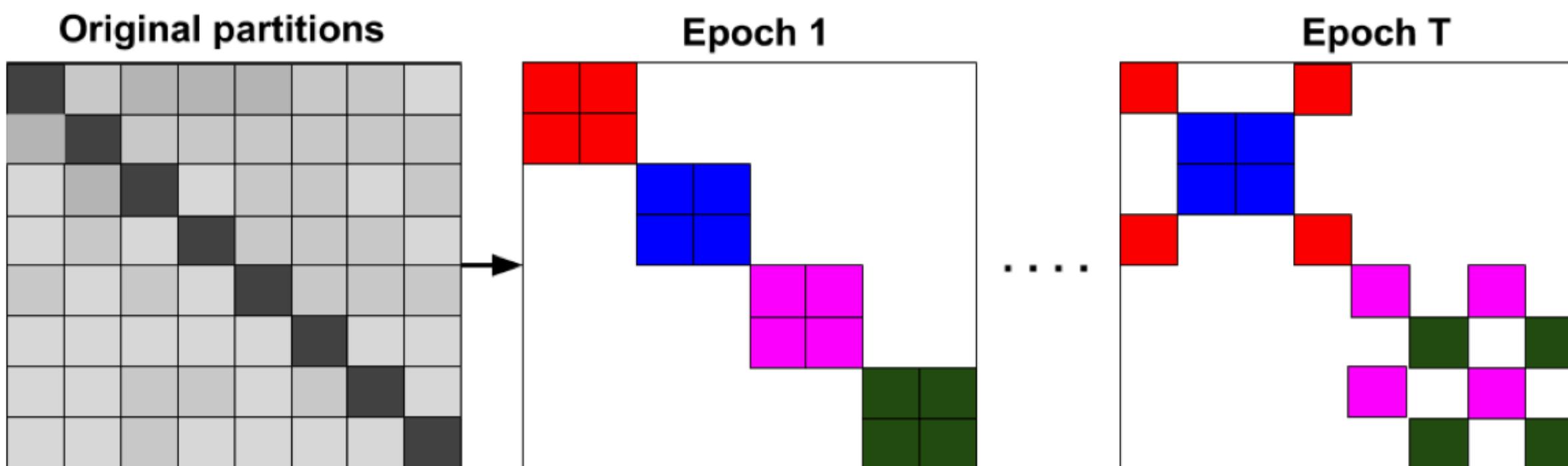
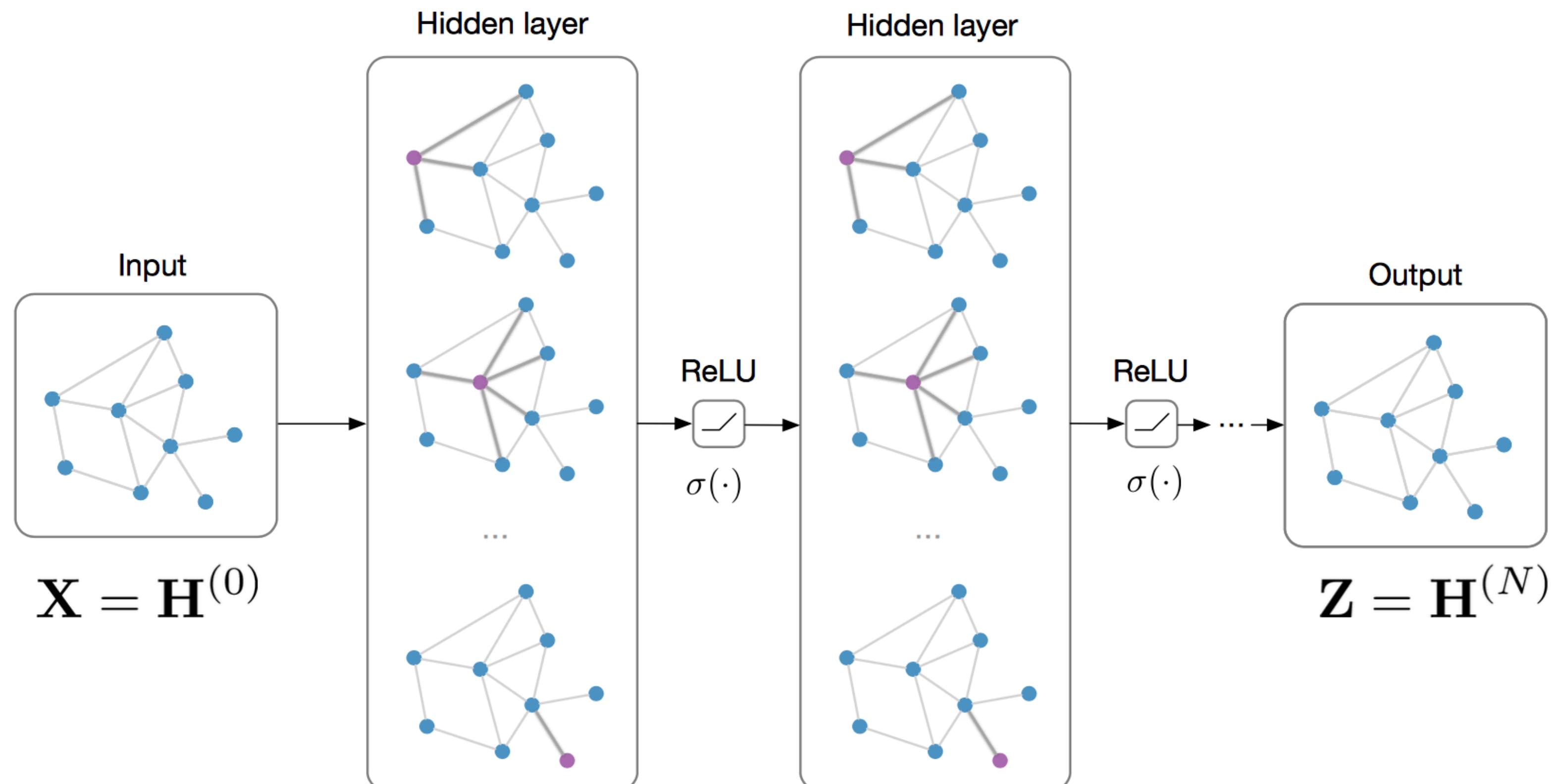


Figure 3: The proposed stochastic multiple partitions scheme. In each epoch, we randomly sample q clusters ($q = 2$ is used in this example) and their between-cluster links to form a new batch. Same color blocks are in the same batch.

How do we use GNN / GCN for real
problems?

Classification and Link Prediction with GNNs / GCNs

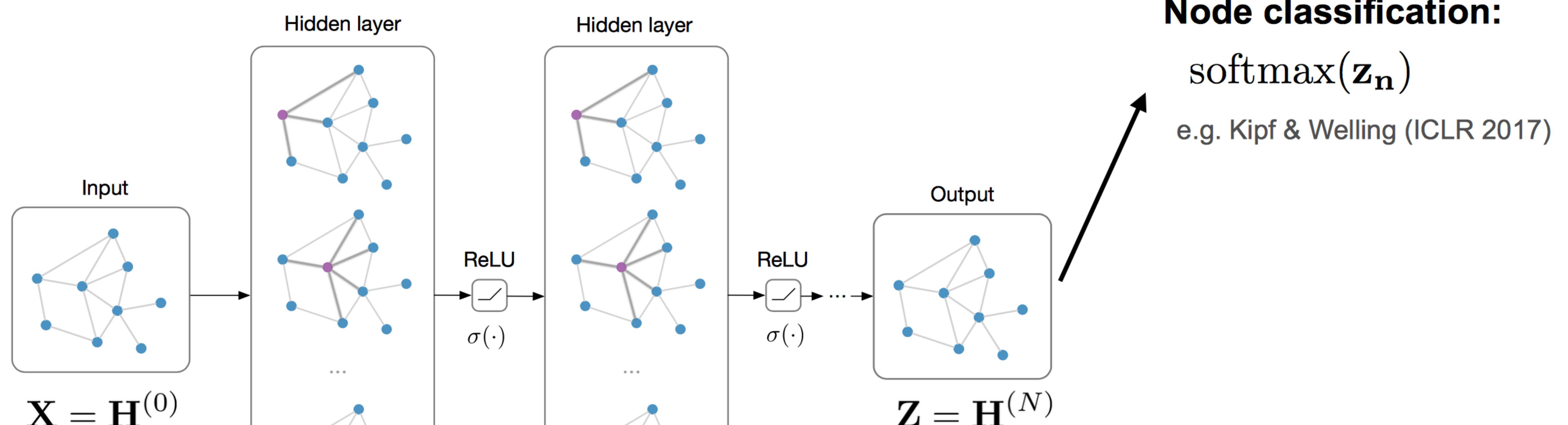
Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



$$\mathbf{H}^{(l+1)} = \sigma \left(\hat{\mathbf{A}} \mathbf{H}^{(l)} \mathbf{W}^{(l)} \right)$$

Classification and Link Prediction with GNNs / GCNs

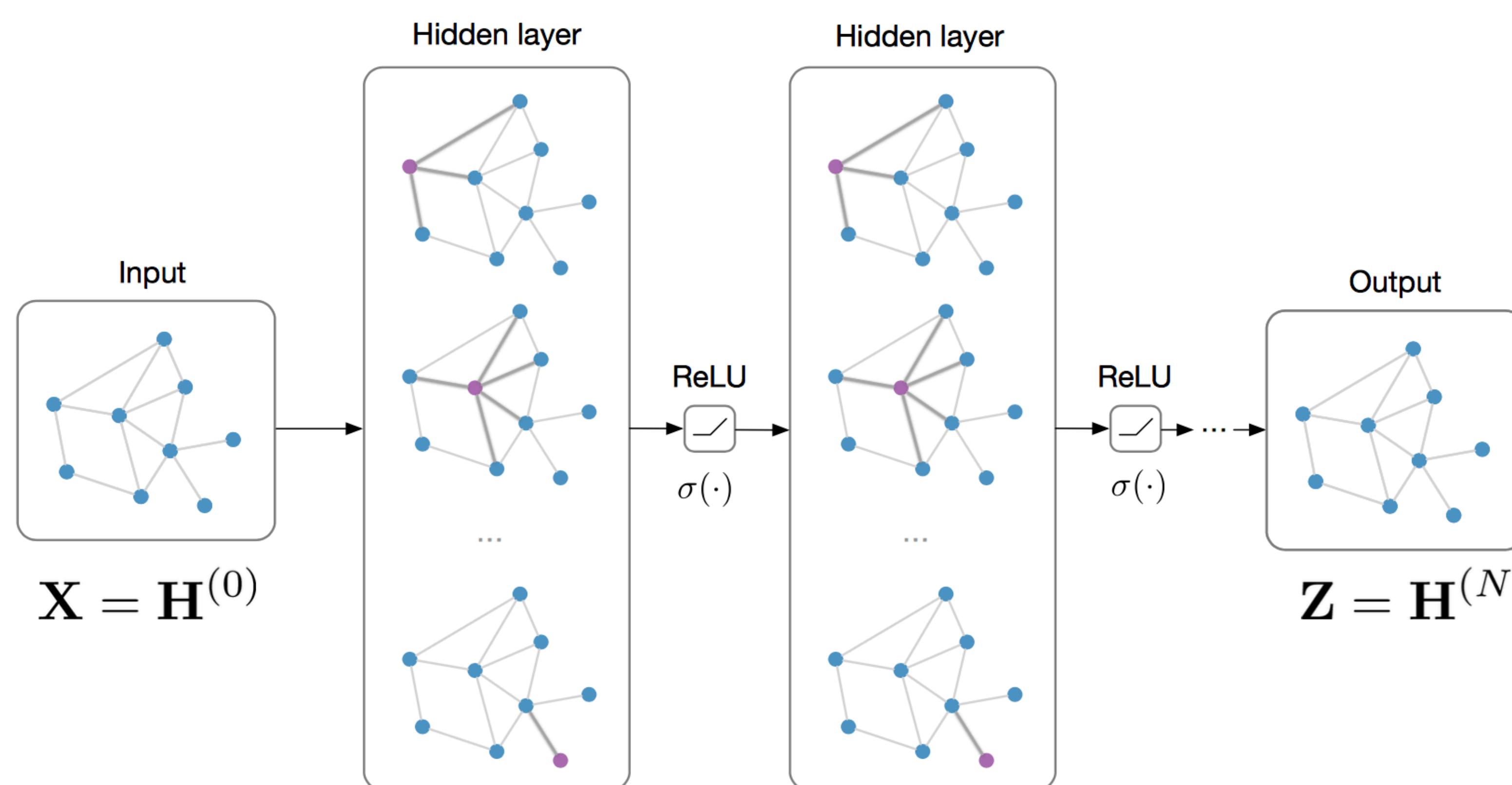
Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



$$\mathbf{H}^{(l+1)} = \sigma \left(\hat{\mathbf{A}} \mathbf{H}^{(l)} \mathbf{W}^{(l)} \right)$$

Classification and Link Prediction with GNNs / GCNs

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



$$\mathbf{H}^{(l+1)} = \sigma \left(\hat{\mathbf{A}} \mathbf{H}^{(l)} \mathbf{W}^{(l)} \right)$$

Node classification:

$$\text{softmax}(\mathbf{z}_n)$$

e.g. Kipf & Welling (ICLR 2017)

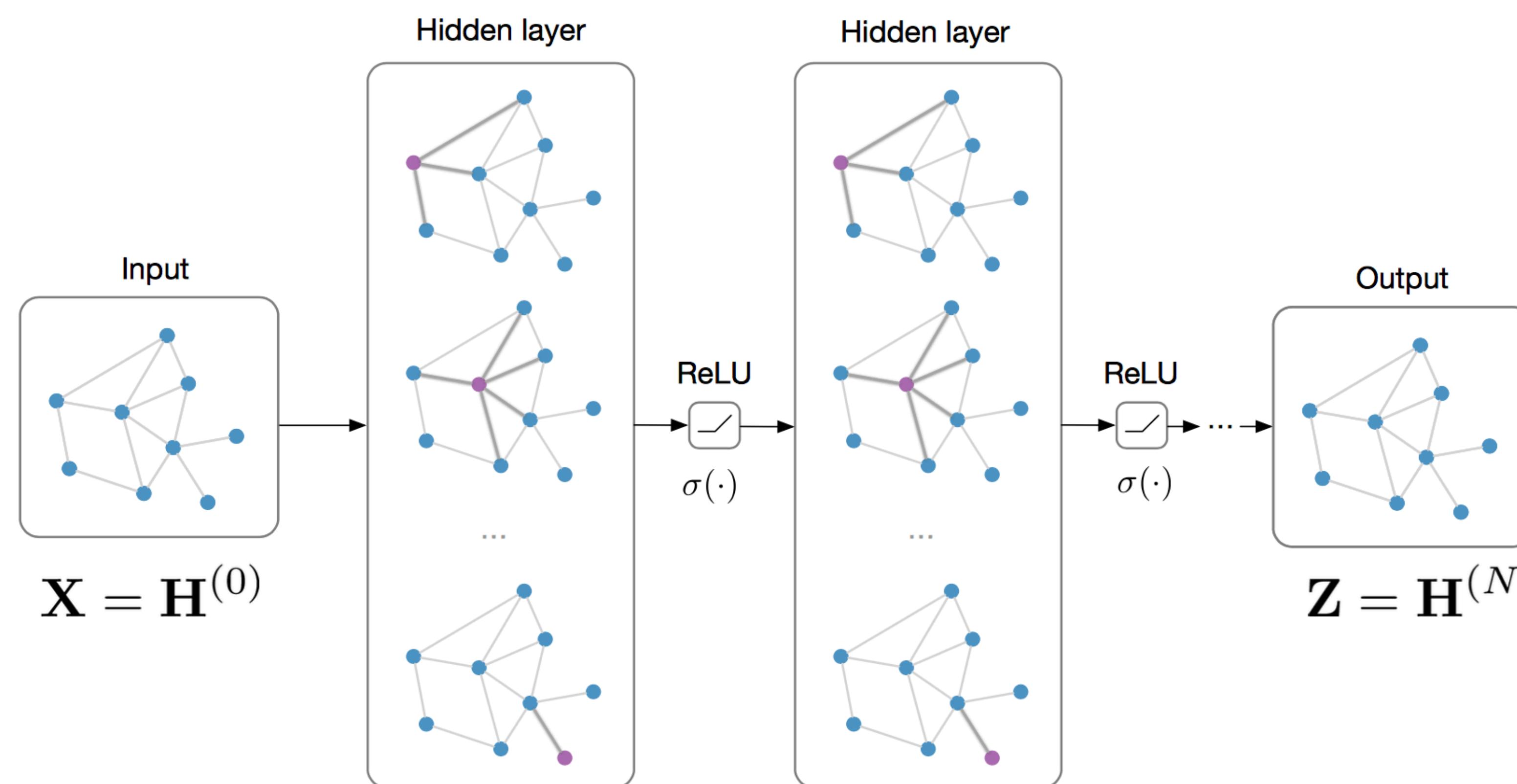
Graph classification:

$$\text{softmax}(\sum_n \mathbf{z}_n)$$

e.g. Duvenaud et al. (NIPS 2015)

Classification and Link Prediction with GNNs / GCNs

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



$$\mathbf{H}^{(l+1)} = \sigma \left(\hat{\mathbf{A}} \mathbf{H}^{(l)} \mathbf{W}^{(l)} \right)$$

Node classification:

$$\text{softmax}(\mathbf{z}_n)$$

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

$$\text{softmax}(\sum_n \mathbf{z}_n)$$

e.g. Duvenaud et al. (NIPS 2015)

Link prediction:

$$p(A_{ij}) = \sigma(\mathbf{z}_i^T \mathbf{z}_j)$$

Kipf & Welling (NIPS BDL 2016)

“Graph Auto-Encoders”

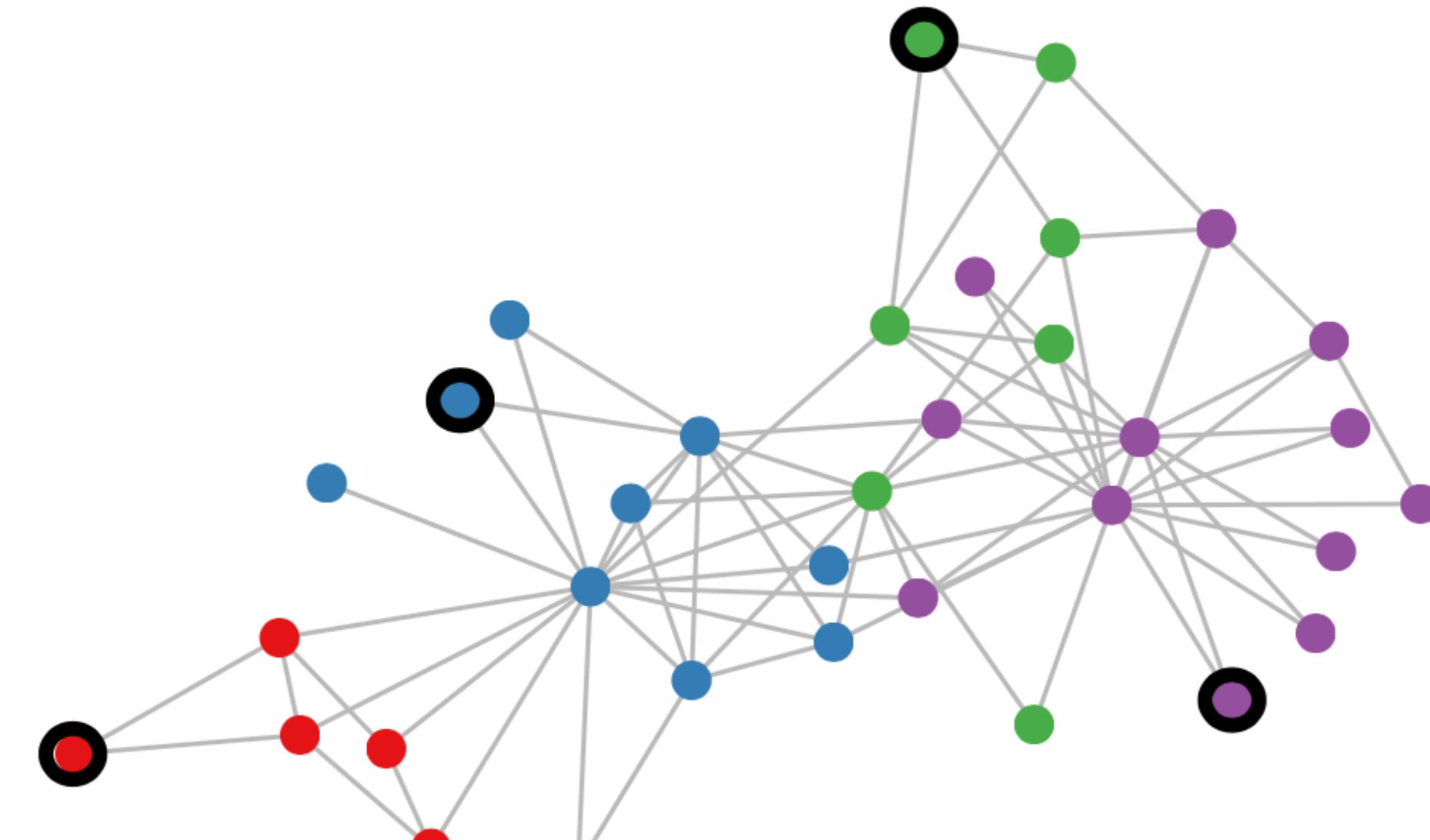
Semi-supervised Classification on Graphs

Setting:

Some nodes are labeled (black circle)
All other nodes are unlabeled

Task:

Predict node label of unlabeled nodes



Semi-supervised Classification on Graphs

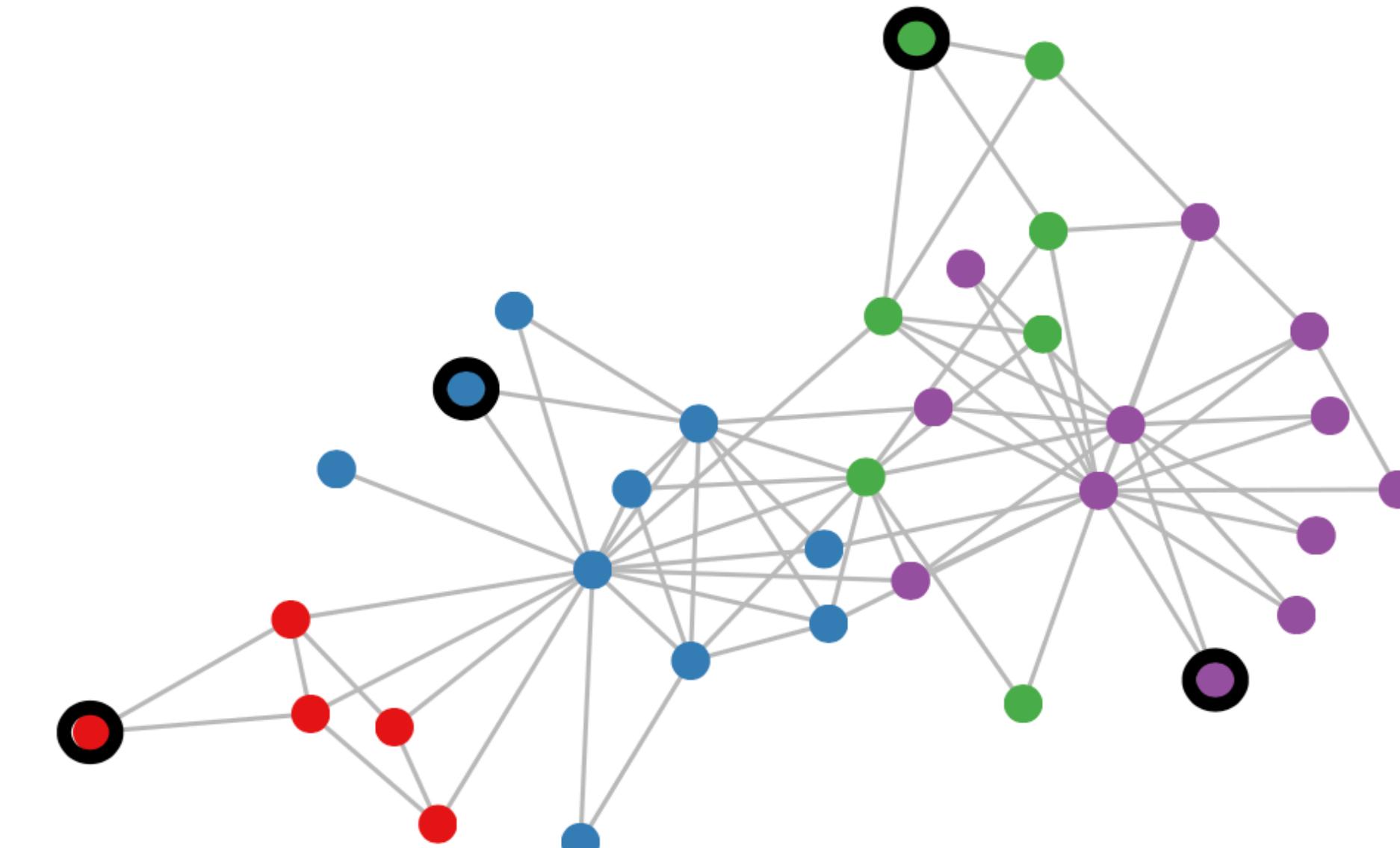
Setting:

Some nodes are labeled (black circle)

All other nodes are unlabeled

Task:

Predict node label of unlabeled nodes



Evaluate loss on labeled nodes only:

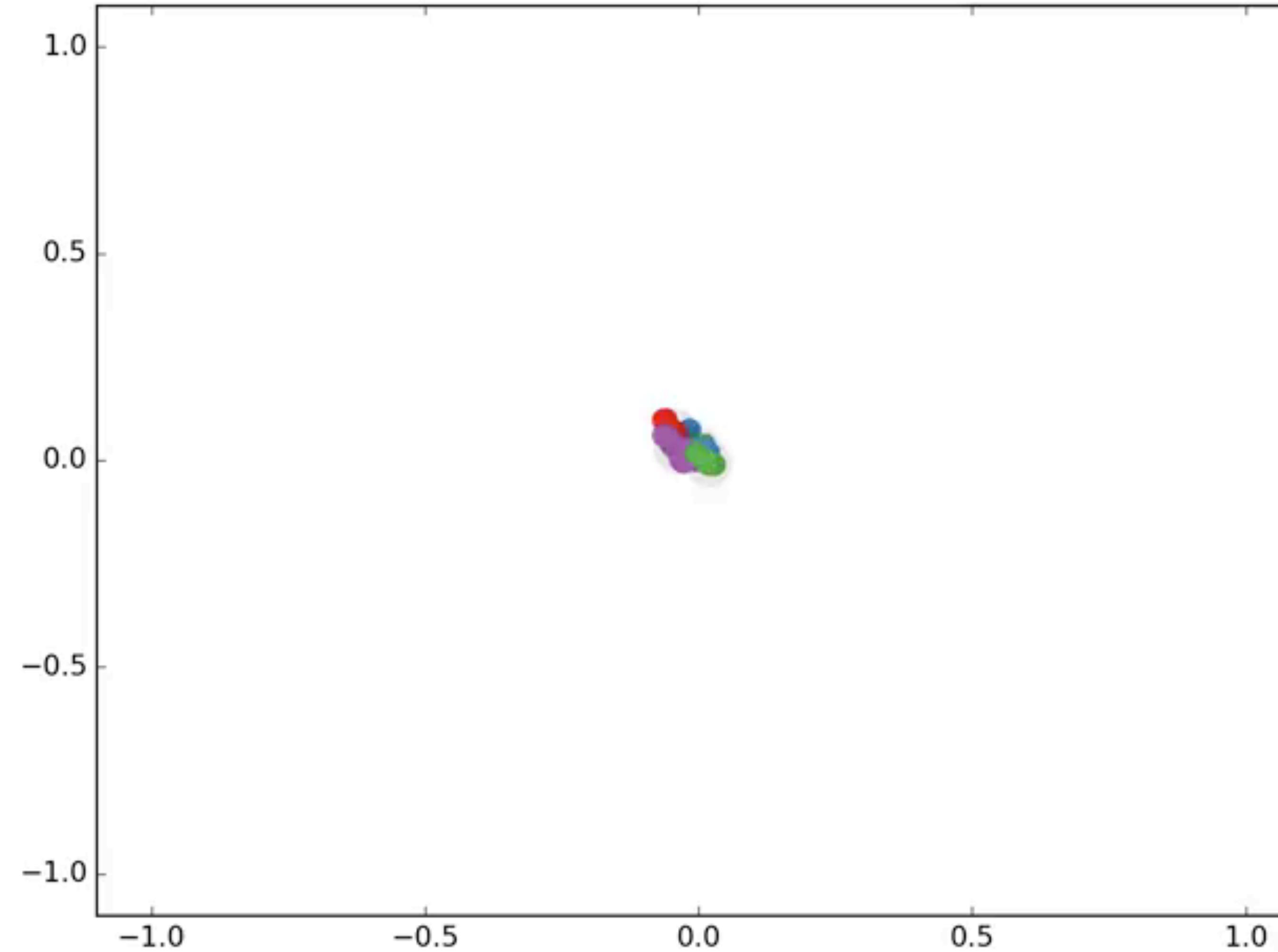
$$\mathcal{L} = - \sum_{l \in \mathcal{Y}_L} \sum_{f=1}^F Y_{lf} \ln Z_{lf}$$

\mathcal{Y}_L set of labeled node indices

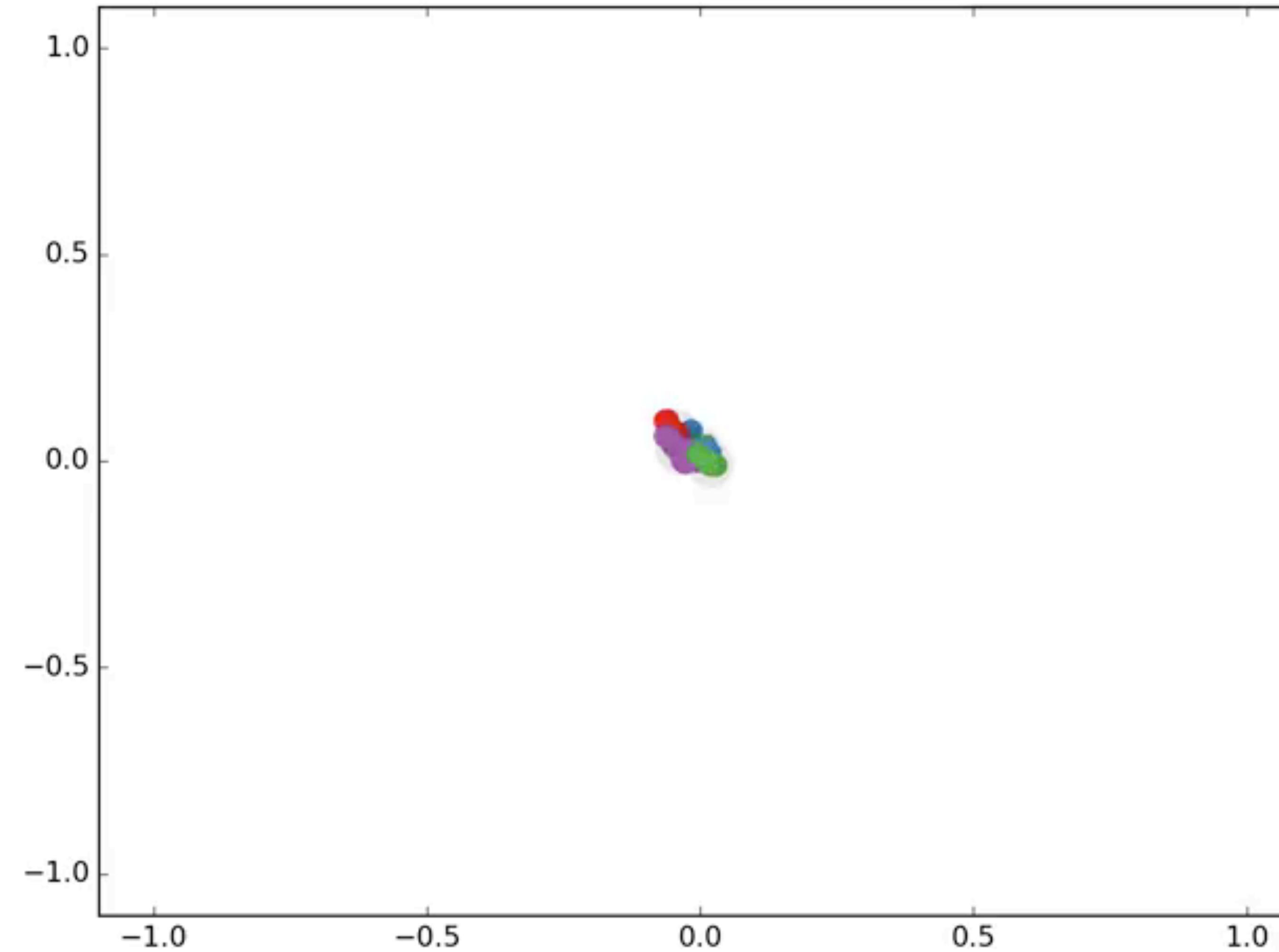
\mathbf{Y} label matrix

\mathbf{Z} GCN output (after softmax)

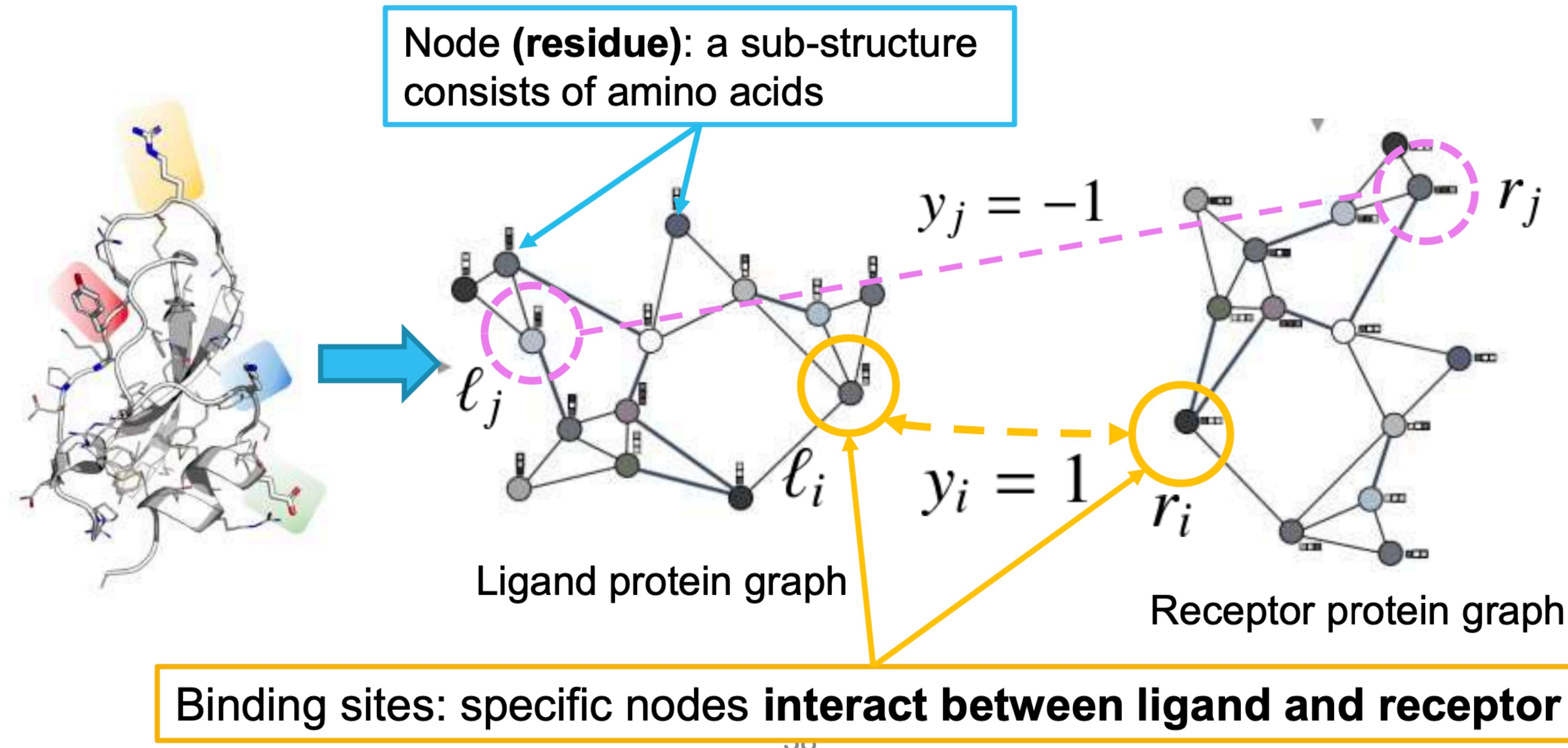
Semi-supervised Classification on Graphs



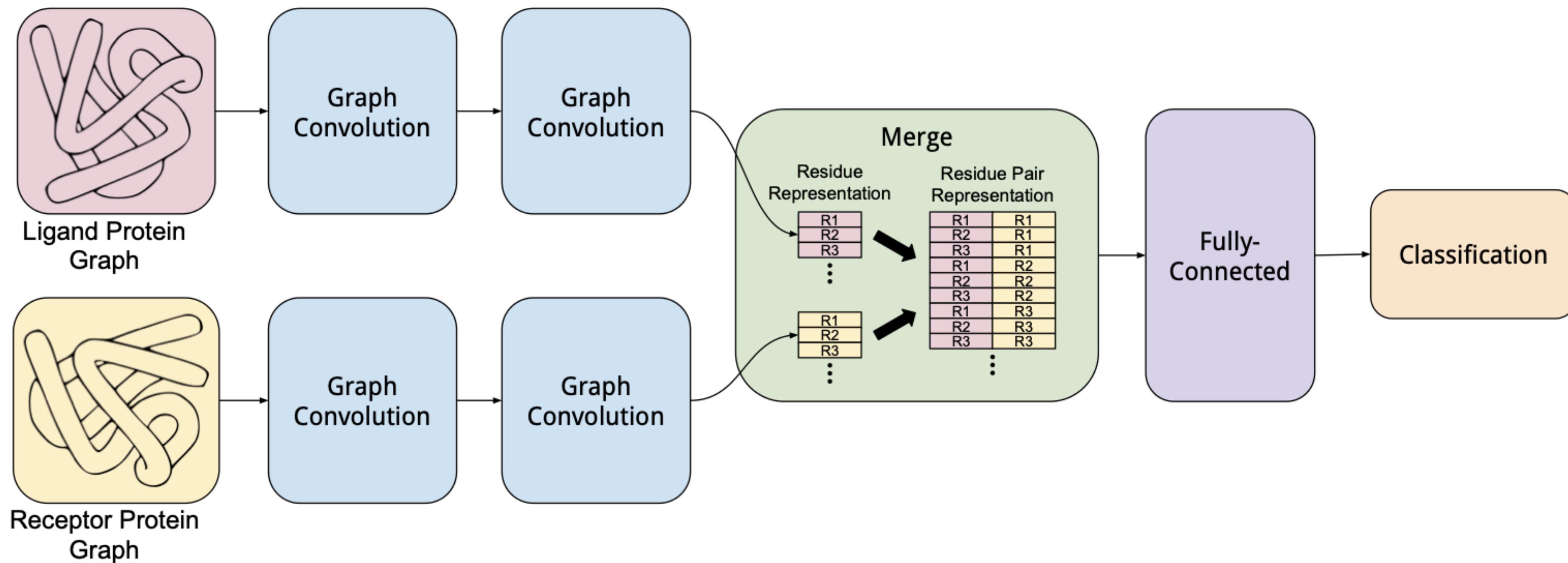
Semi-supervised Classification on Graphs



Protein Interface Prediction



Protein Interface Prediction



Protein Interface Prediction

Node features only:
$$z_i = \sigma\left(W^C x_i + \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} W^N x_j + b\right)$$

Node features + **edge** features:
$$z_i = \sigma\left(W^C x_i + \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} W^N x_j + \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} W^E A_{ij} + b\right)$$

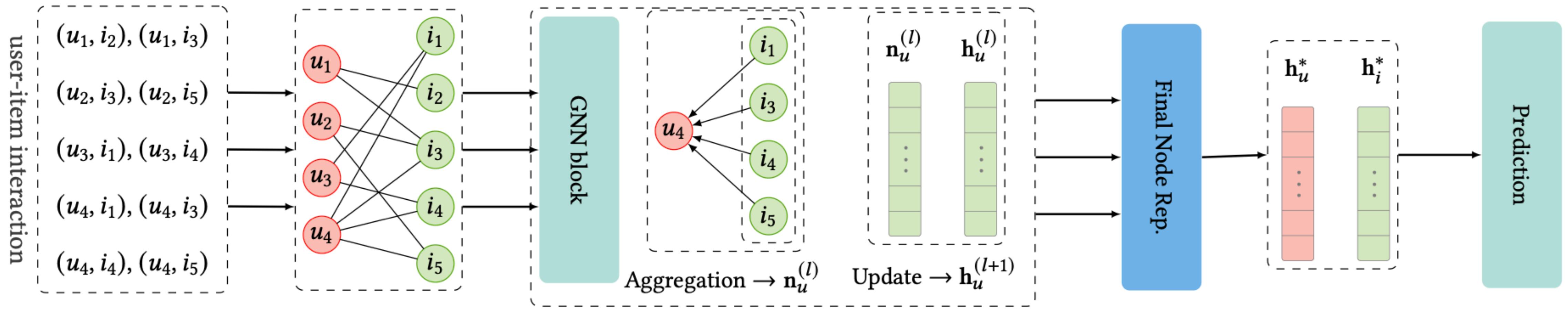
Node features + **edge** features:
$$z_i = \sigma\left(W^C x_i + \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} W_j^N x_j + \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} W_j^E A_{ij} + b\right)$$

Neighbor nodes j are ordered by distances from the node i

Protein Interface Prediction

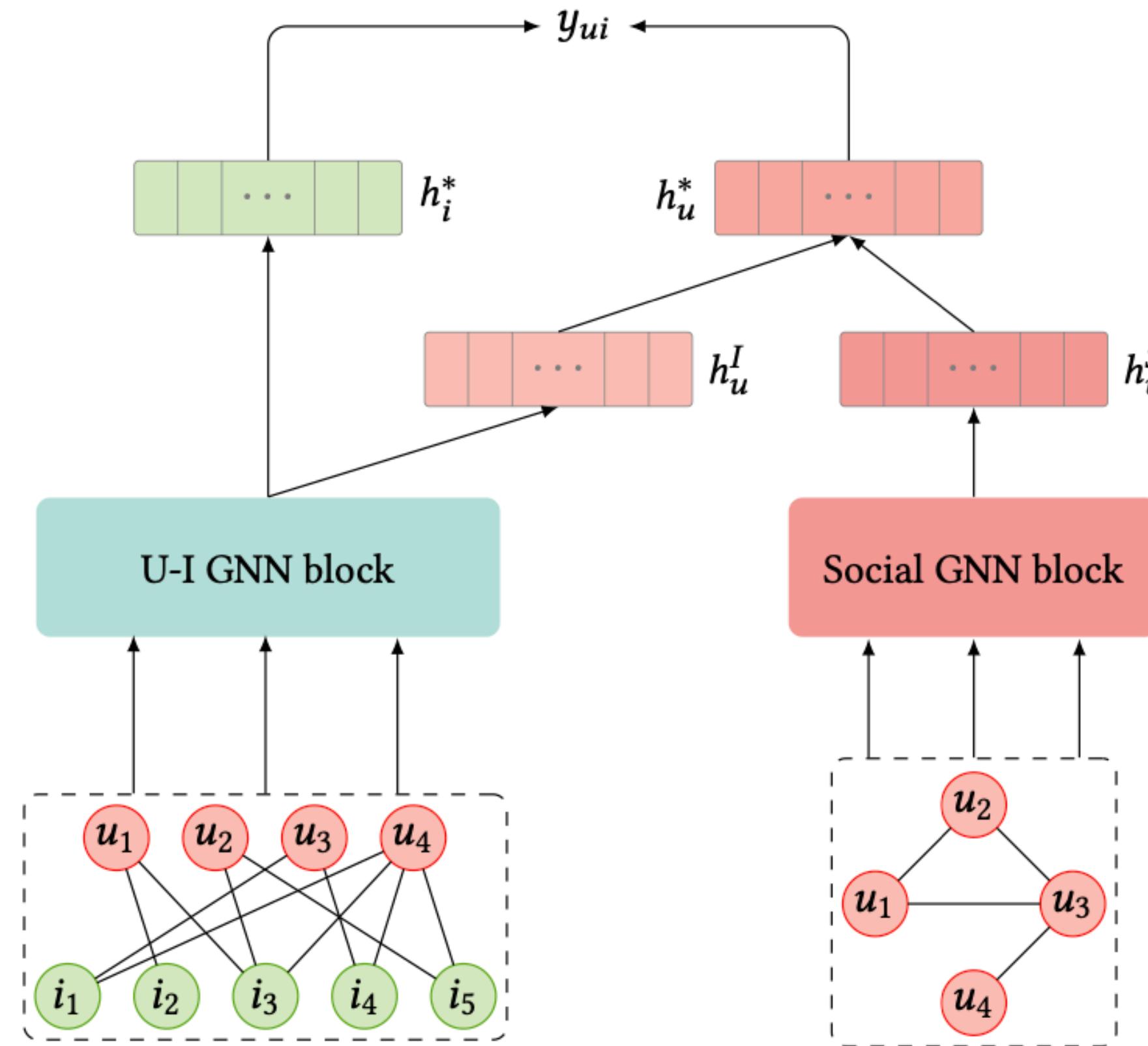
Method	Convolutional Layers			
	1	2	3	4
No Convolution	0.812 (0.007)	0.810 (0.006)	0.808 (0.006)	0.796 (0.006)
Diffusion (DCNN) (2 hops) [5]	0.790 (0.014)	–	–	–
Diffusion (DCNN) (5 hops) [5]	0.828 (0.018)	–	–	–
Single Weight Matrix (MFN [9])	0.865 (0.007)	0.871 (0.013)	0.873 (0.017)	0.869 (0.017)
Node Average (Equation 1)	0.864 (0.007)	0.882 (0.007)	0.891 (0.005)	0.889 (0.005)
Node and Edge Average (Equation 2)	0.876 (0.005)	0.898 (0.005)	0.895 (0.006)	0.889 (0.007)
DTNN [21]	0.867 (0.007)	0.880 (0.007)	0.882 (0.008)	0.873 (0.012)
Order Dependent (Equation 3)	0.854 (0.004)	0.873 (0.005)	0.891 (0.004)	0.889 (0.008)

Recommender Systems

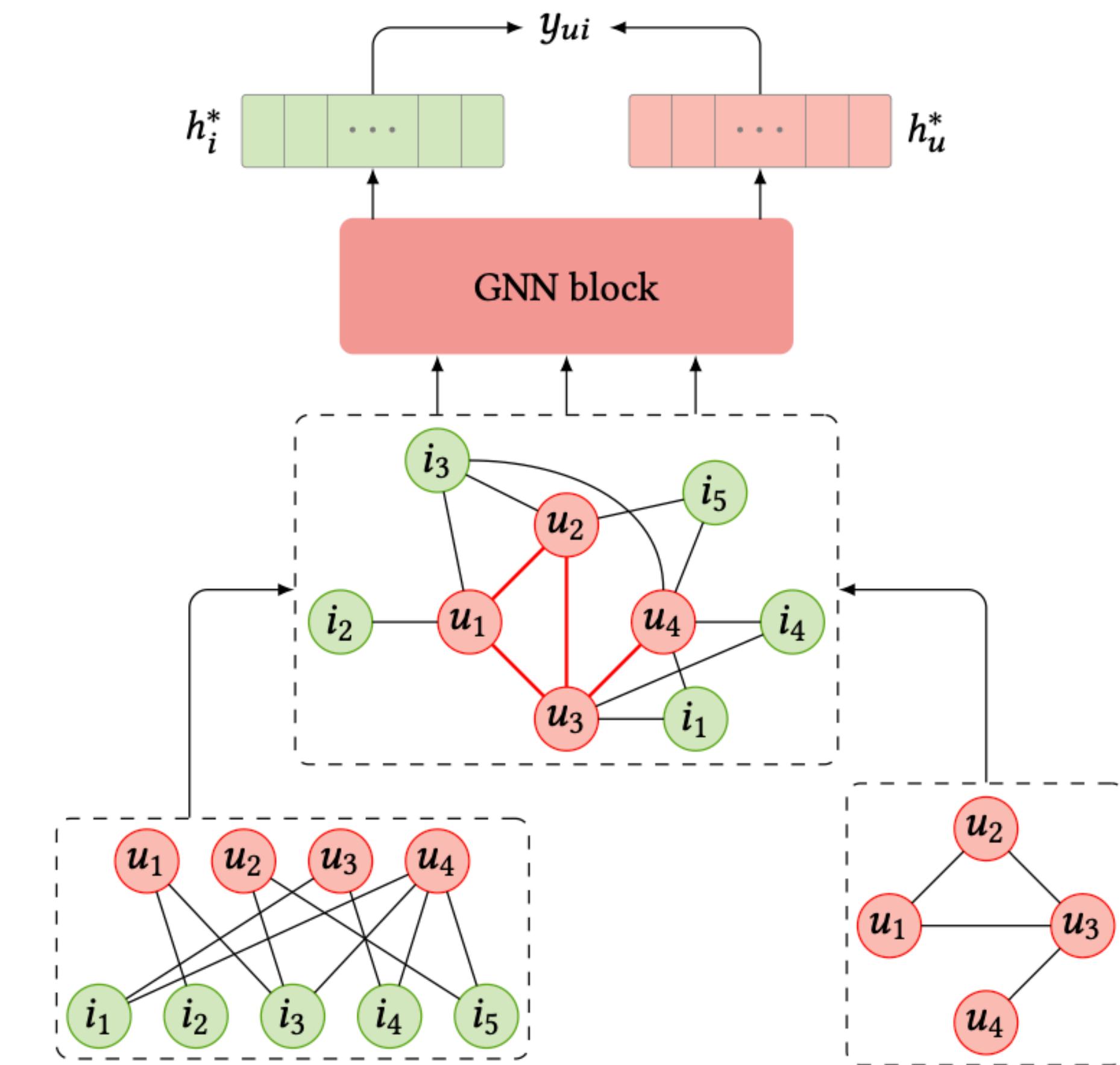


Typical framework of GNN in user-item collaborative filtering

Recommender Systems



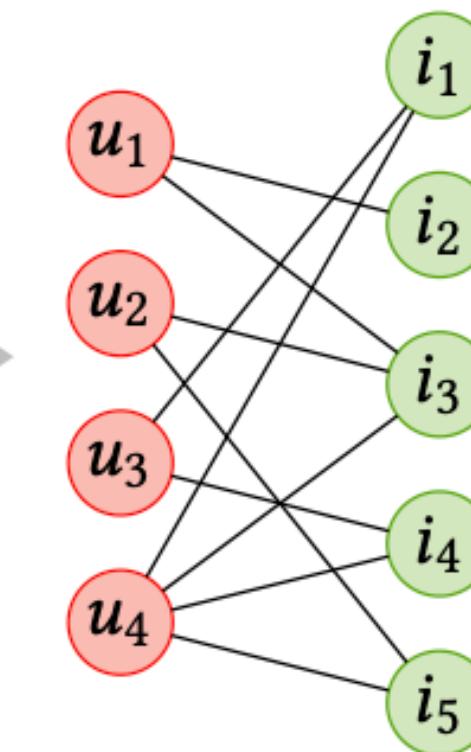
(a) The framework of GNN on the bipartite graph and social network graph separately.



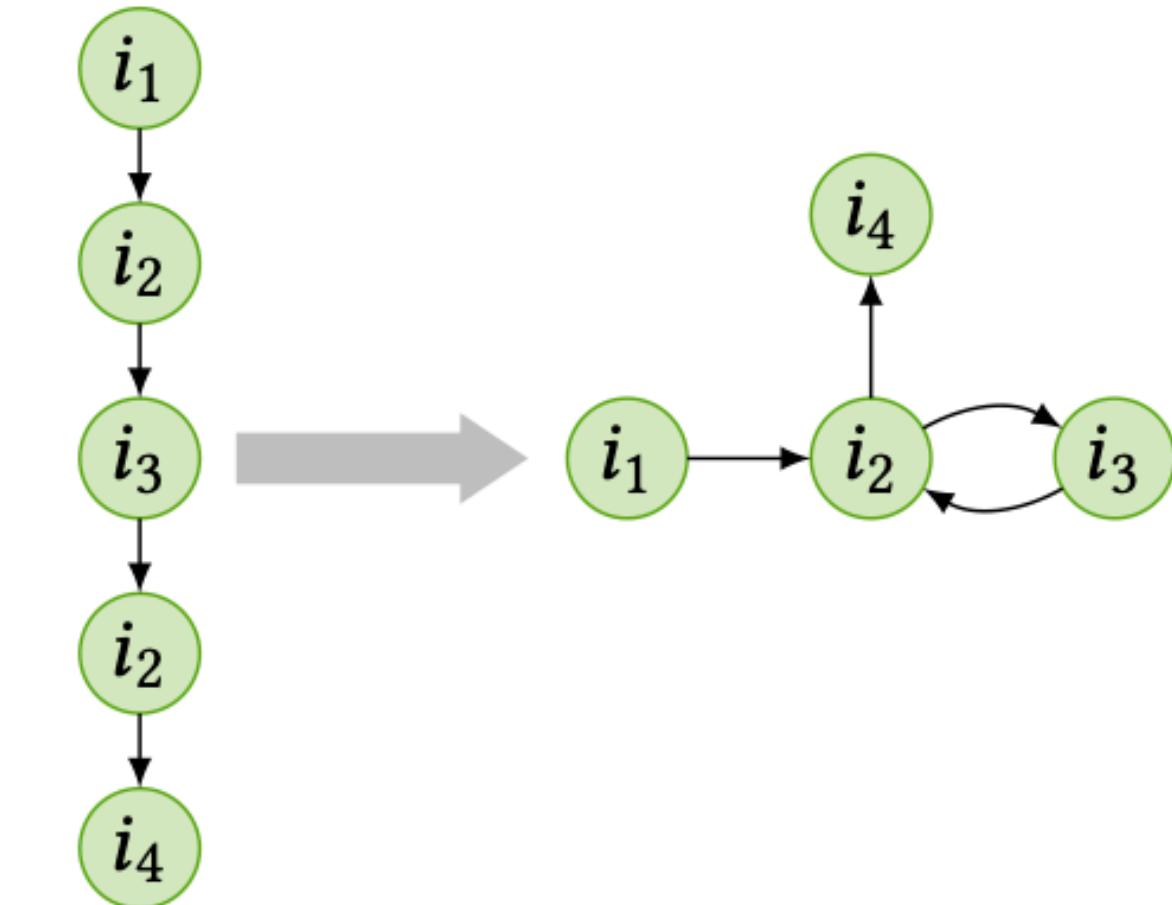
(b) The framework of GNN on the unified graph of user-item interactions and social network.

Recommender Systems

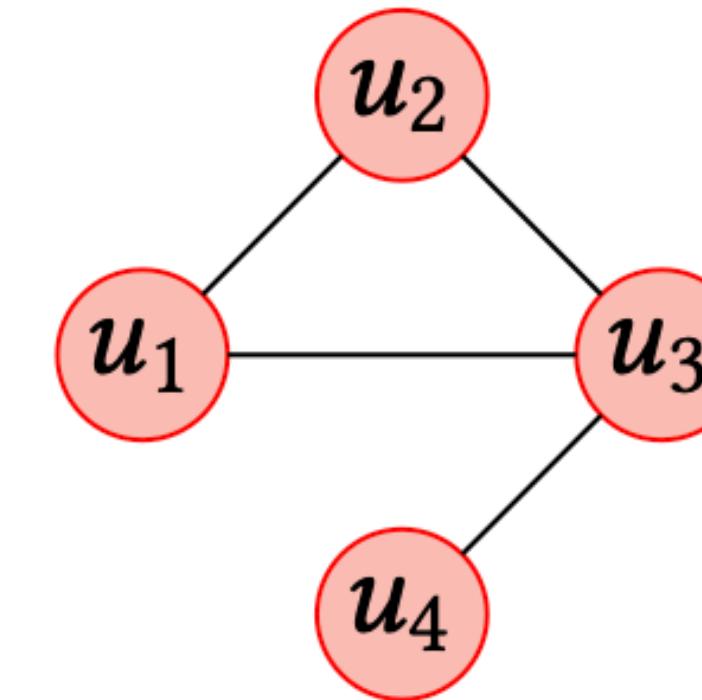
	i_1	i_2	i_3	i_4	i_5
u_1	0	1	1	0	0
u_2	0	0	1	0	1
u_3	1	0	0	1	0
u_4	1	0	1	1	1



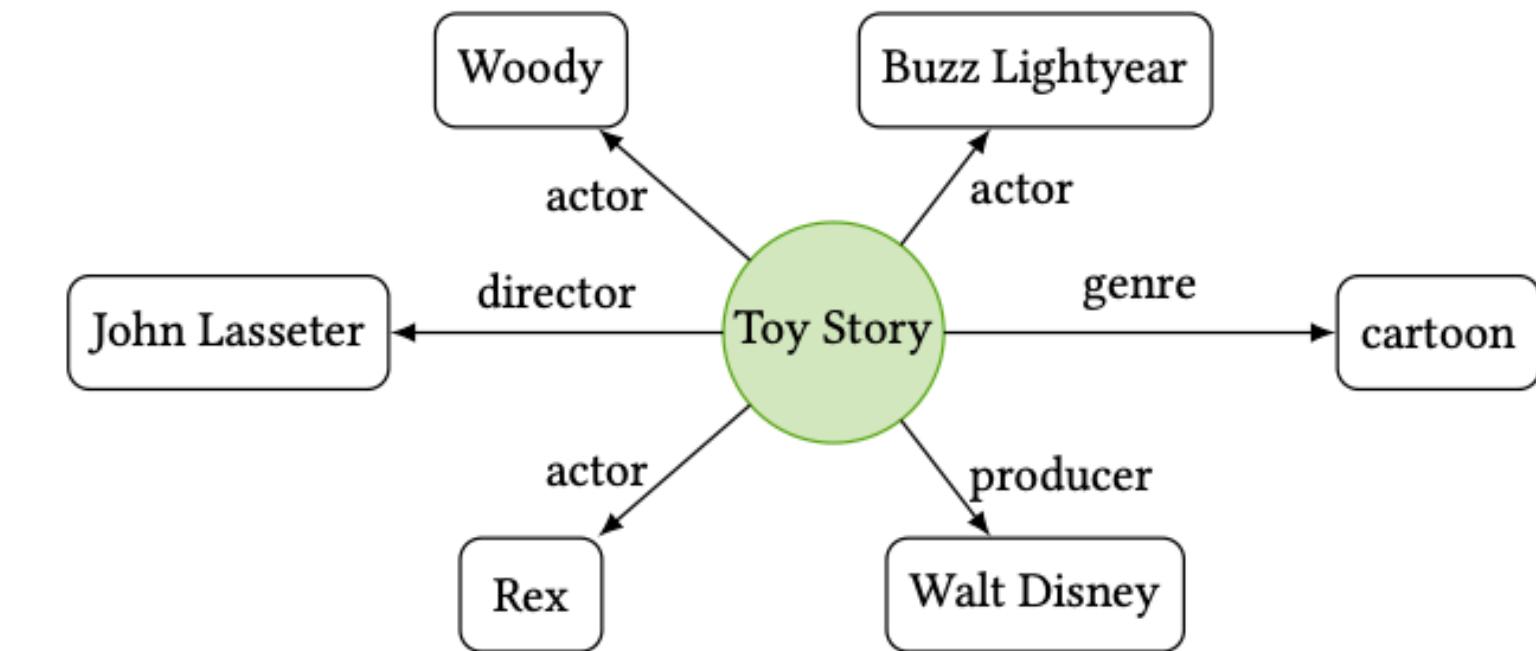
(a) User-item bipartite graph.



(b) Sequence graph.



(c) Social relationship between users.

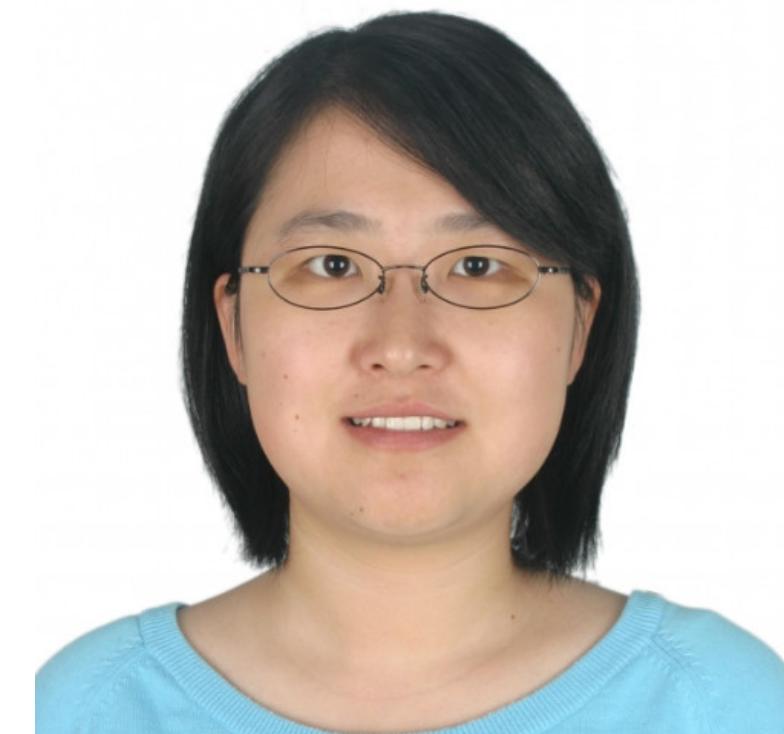


(d) Knowledge graph

G³raphGround: Graph-based Language Grounding



Mohit Bajaj



Lanjun Wang

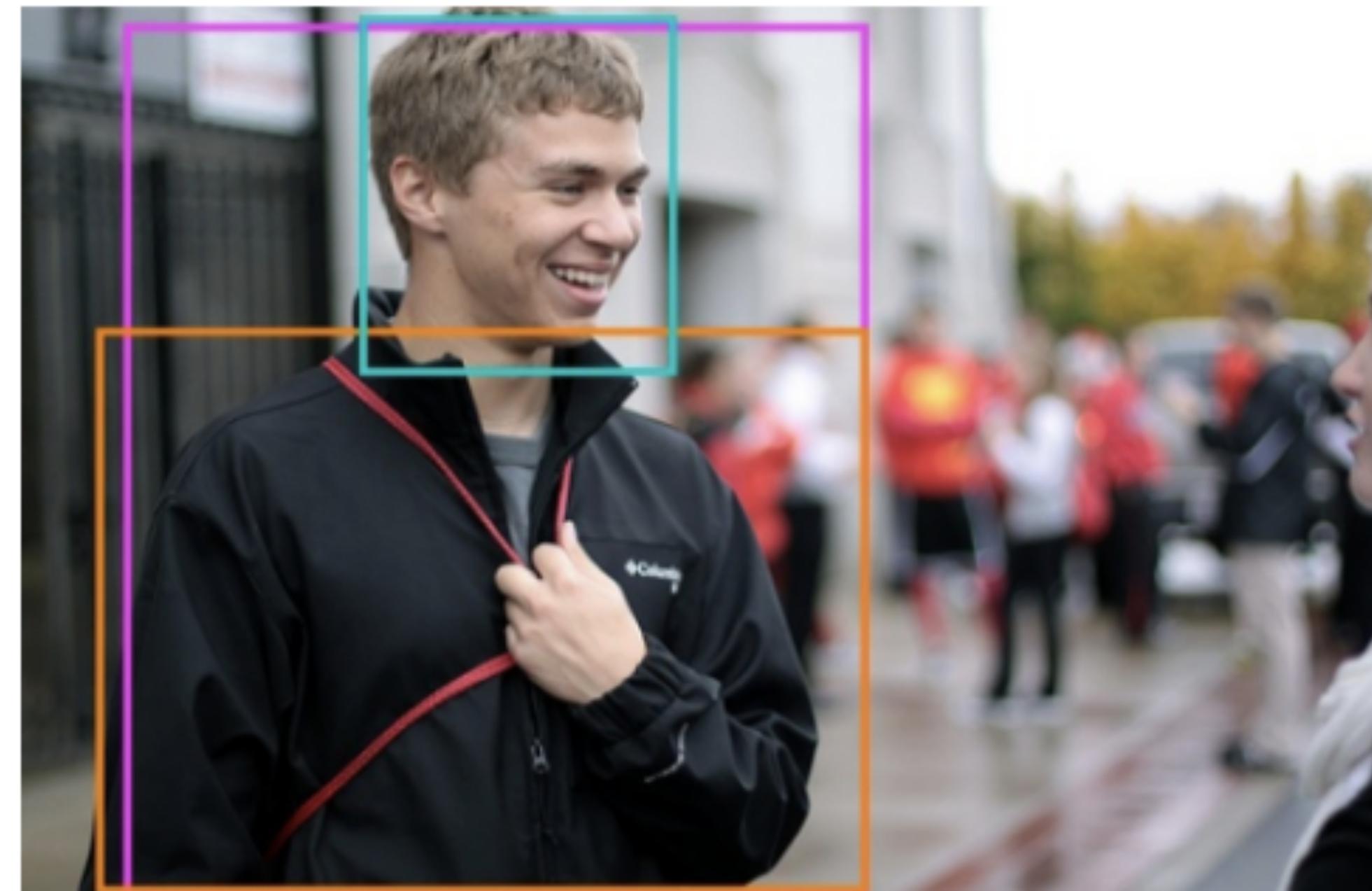


Leonid Sigal



Image Grounding: Beyond Object Detection

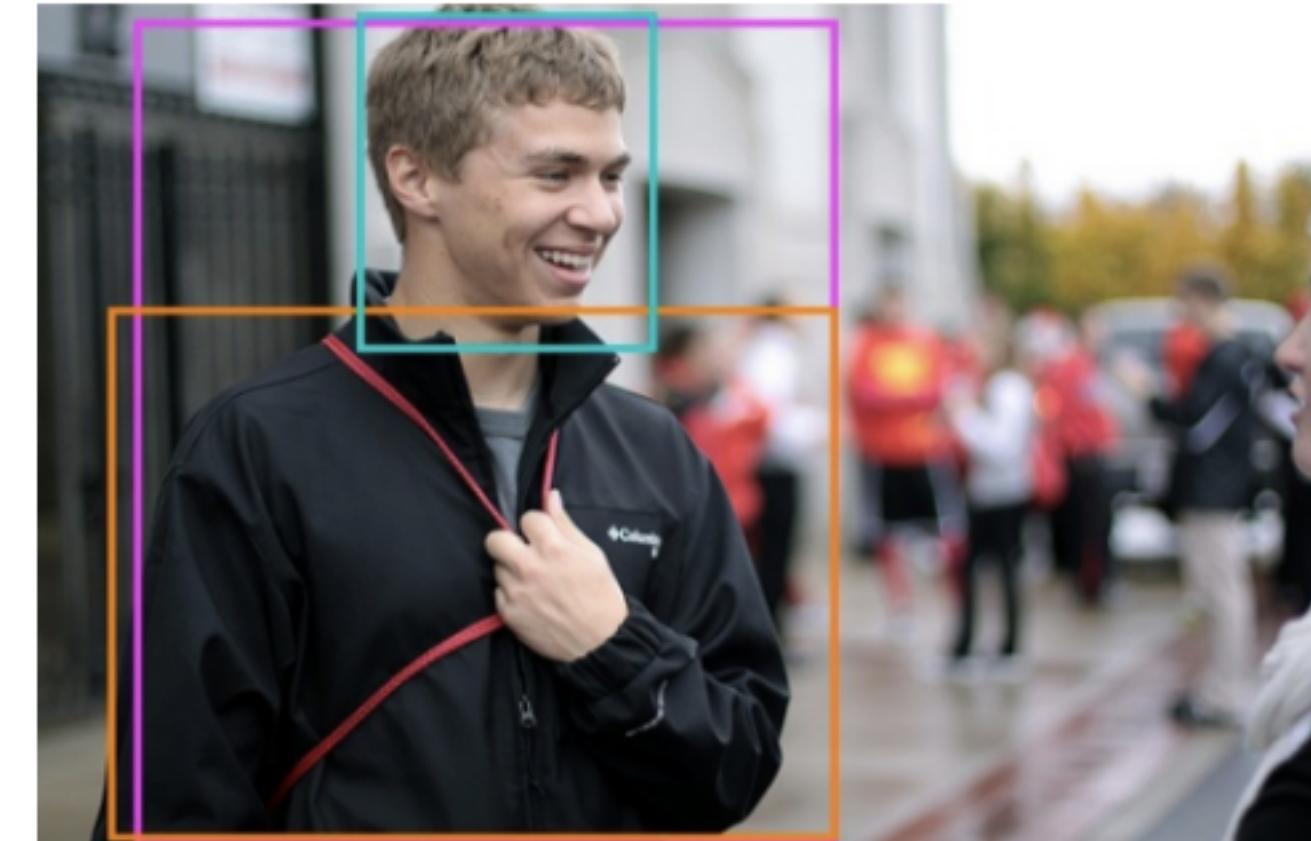
Given the **image** and one or more **natural language phrases**, locate regions that correspond to those phrases.



A man wearing a black-jacket has
a smile on his face.

Image Grounding: Beyond Object Detection

Given the **image** and one or more **natural language phrases**, locate regions that correspond to those phrases.

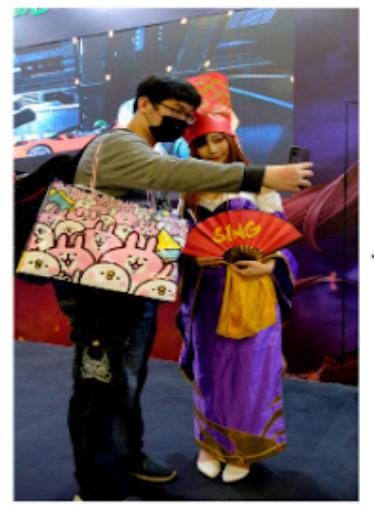


A man wearing a black-jacket has a smile on his face.

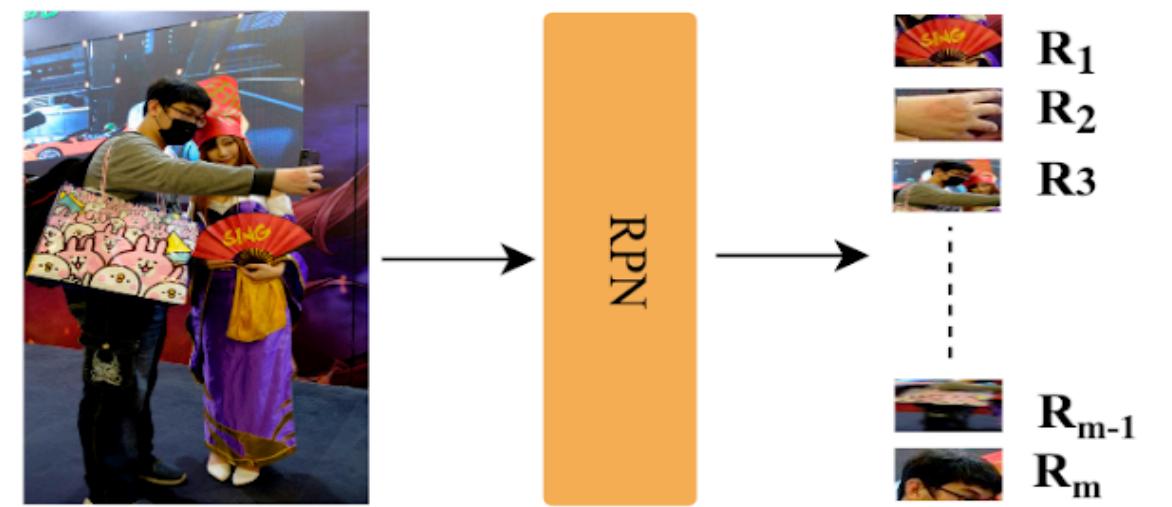
Fundamental task for **image / video understanding**

- Helps improve performance on other tasks (e.g., image captioning, VQA)

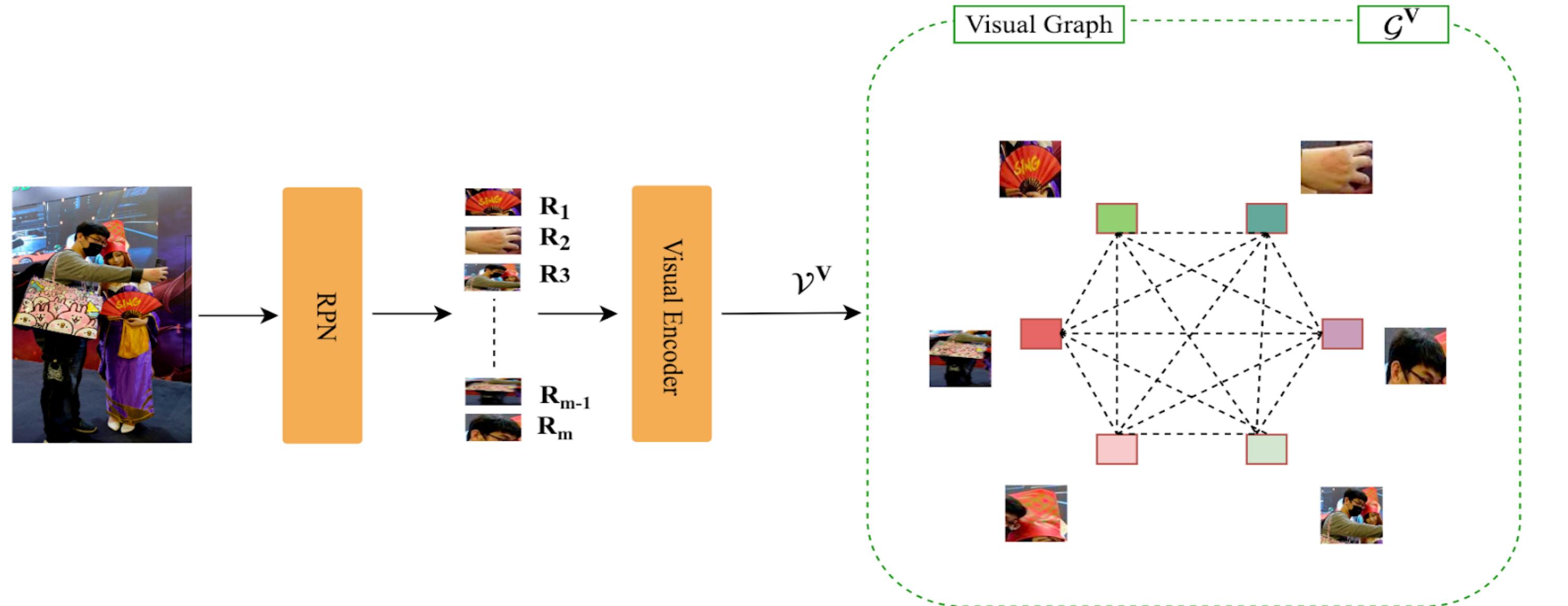
Proposed Architecture



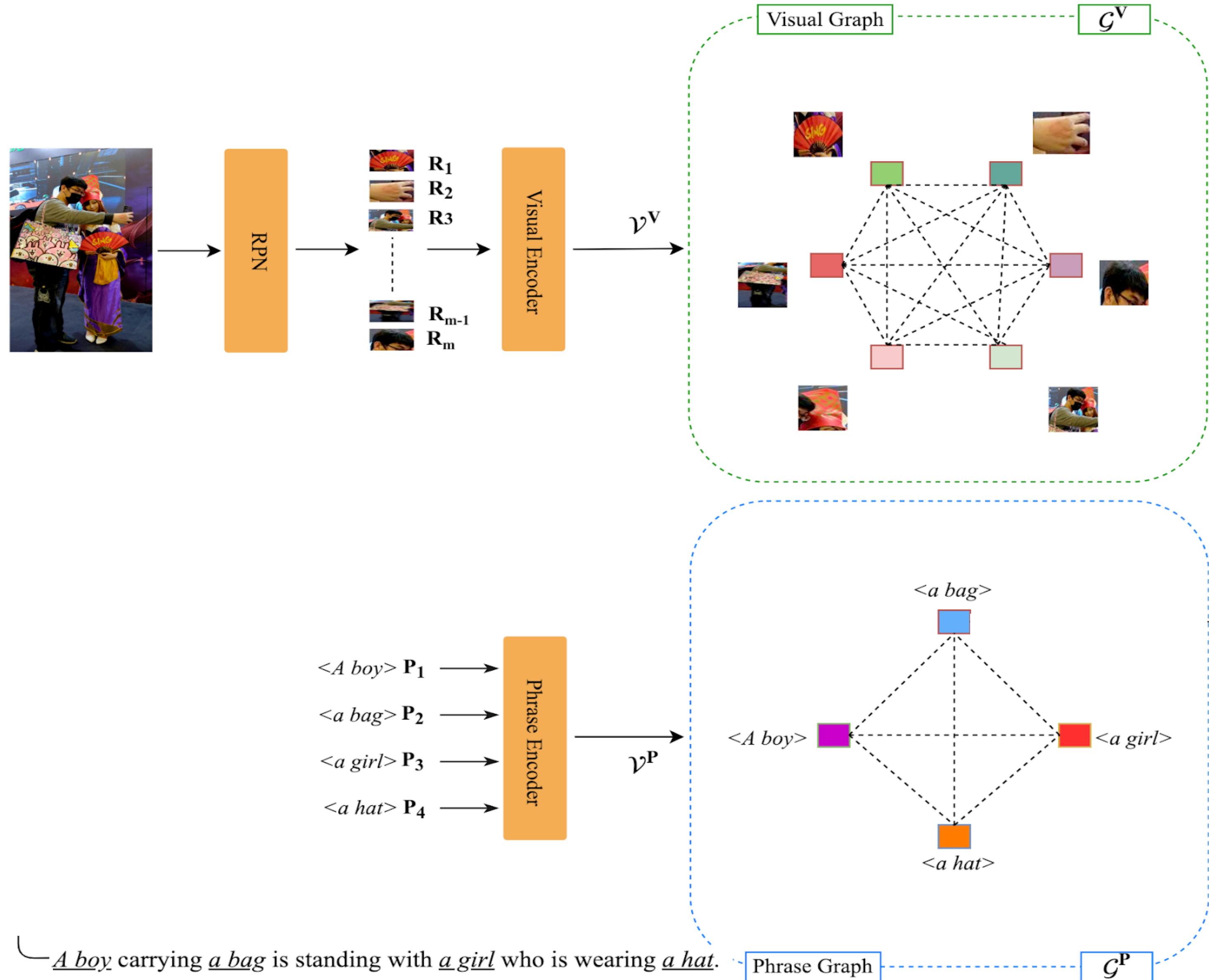
Proposed Architecture



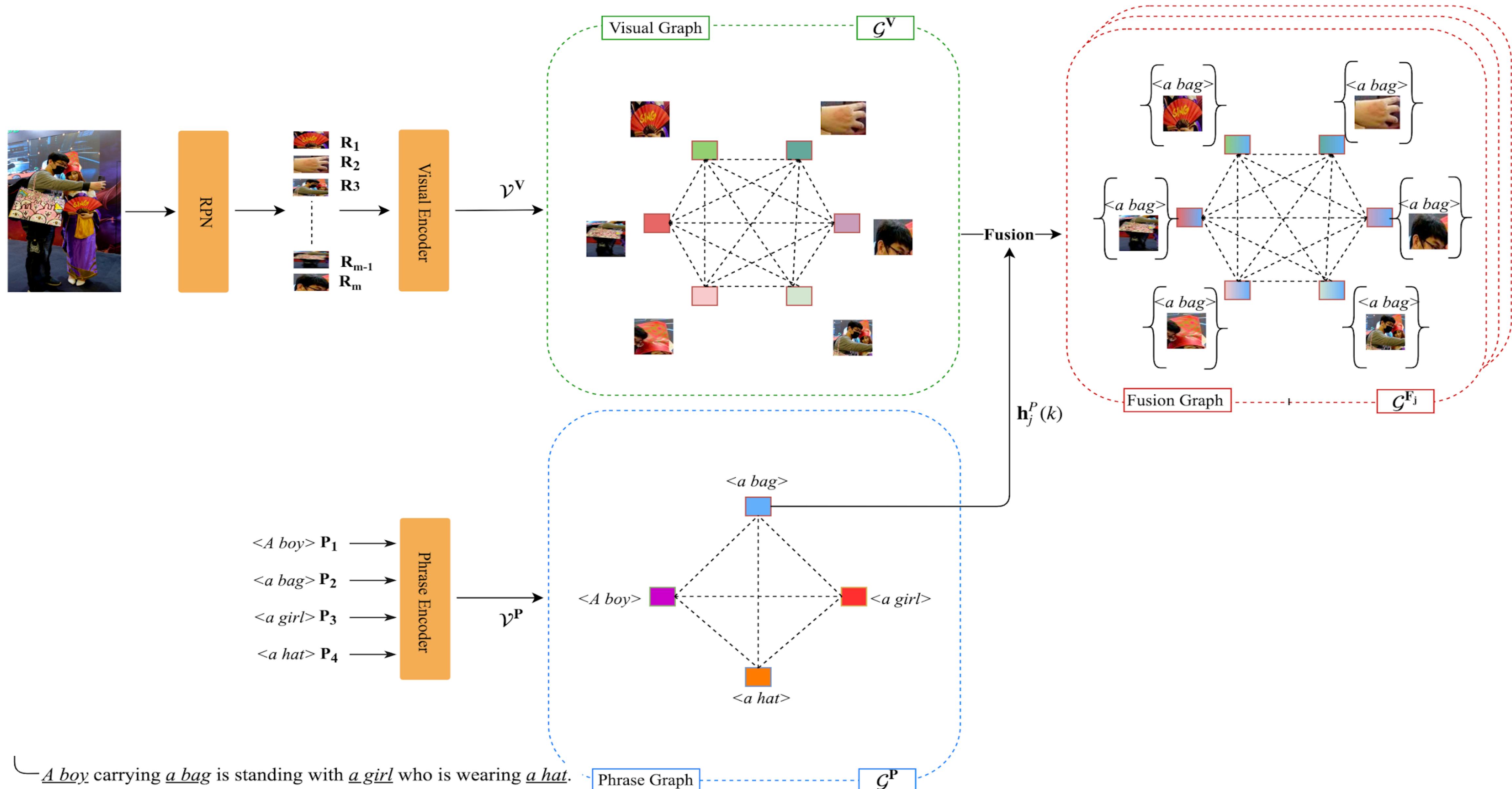
Proposed Architecture



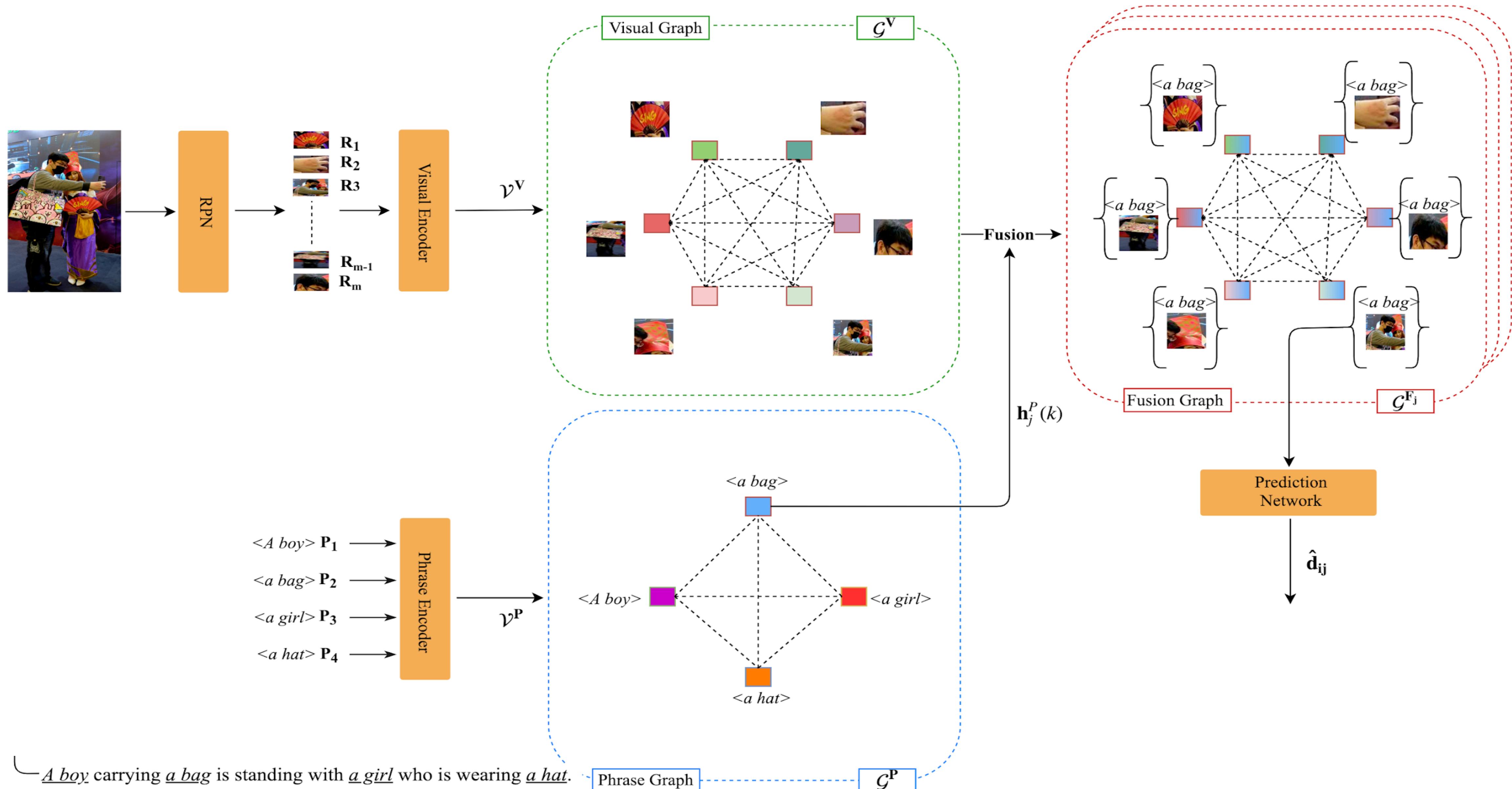
Proposed Architecture



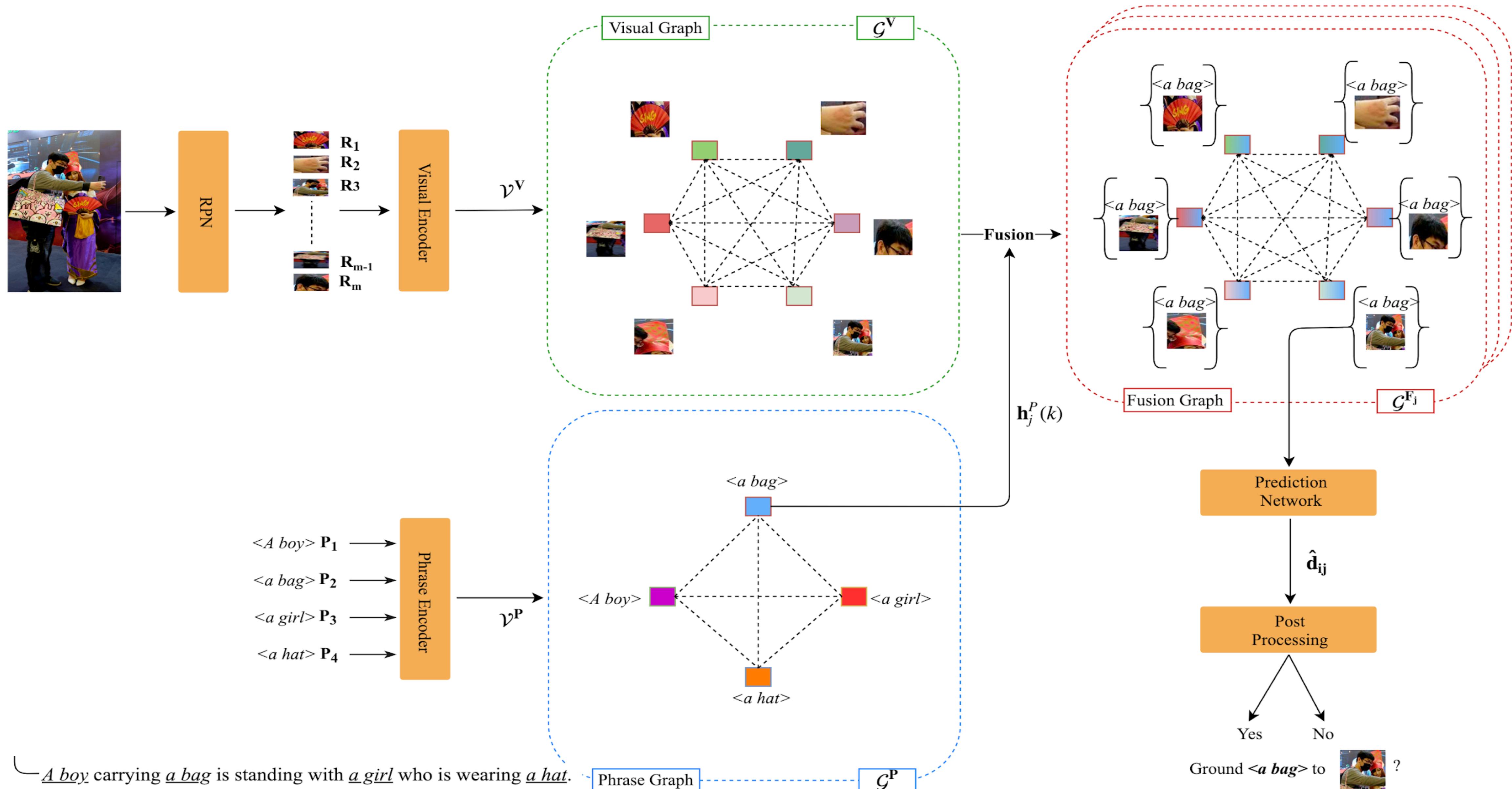
Proposed Architecture



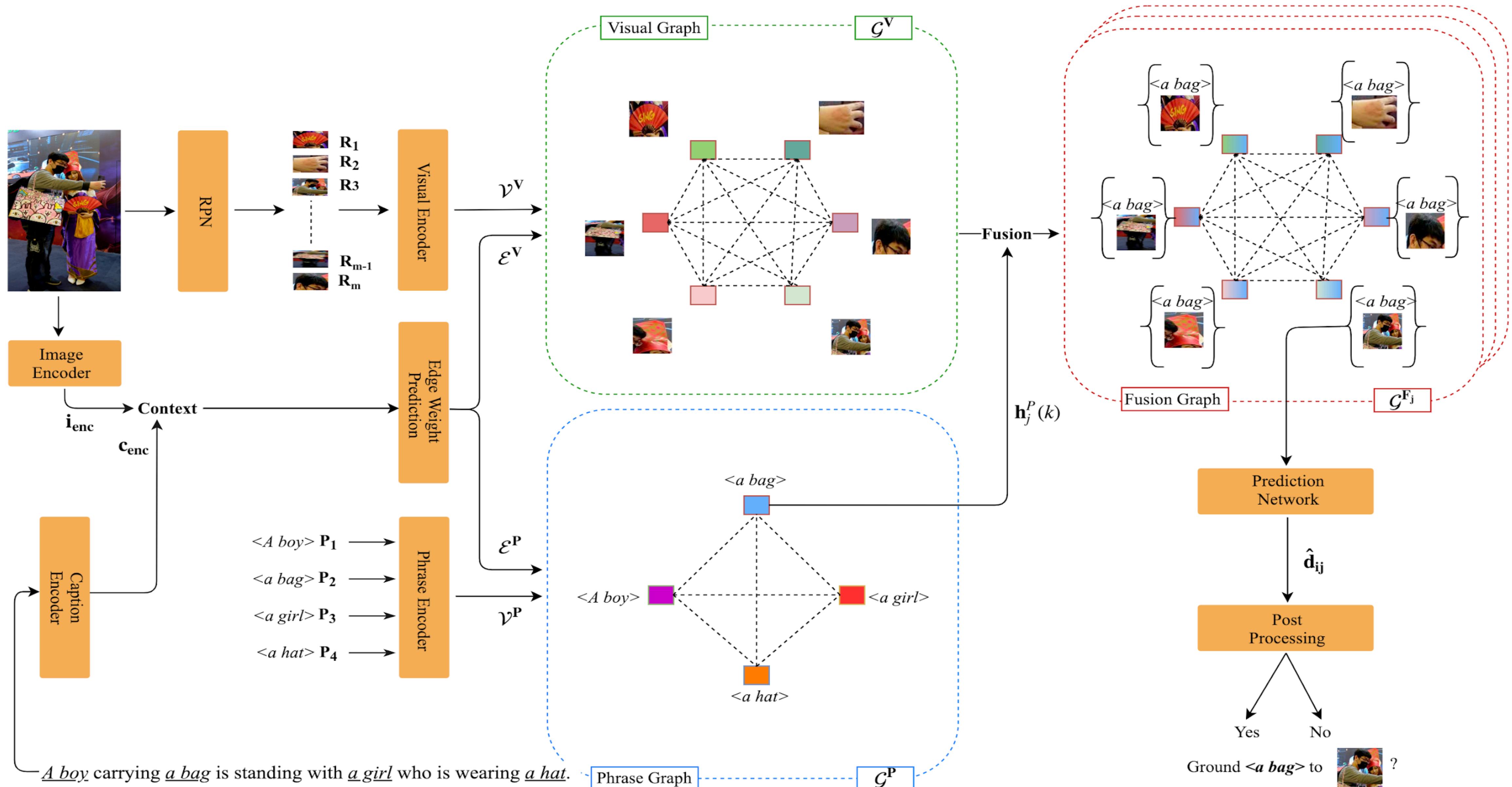
Proposed Architecture



Proposed Architecture



Proposed Architecture



Experiments

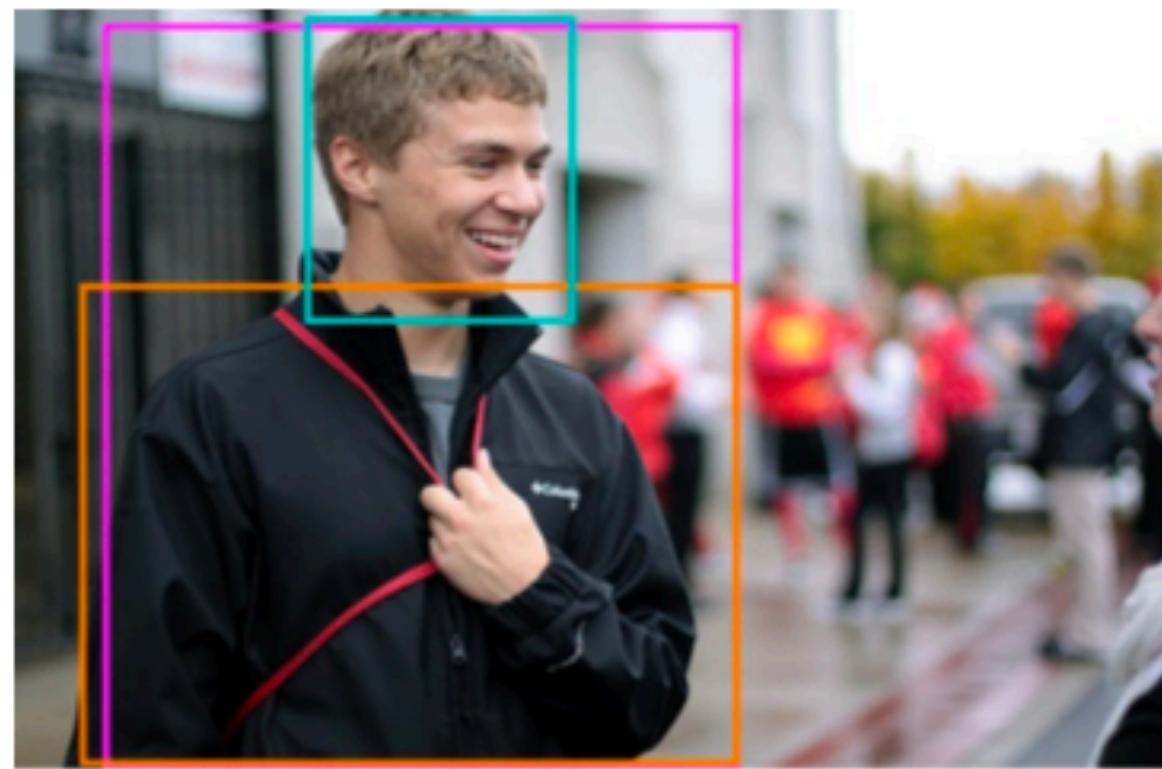
Datasets

- **Flickr30K Entities:** (mostly noun) Phrases parsed from image captions
- **ReferIt Game:** Unambiguous single phrases

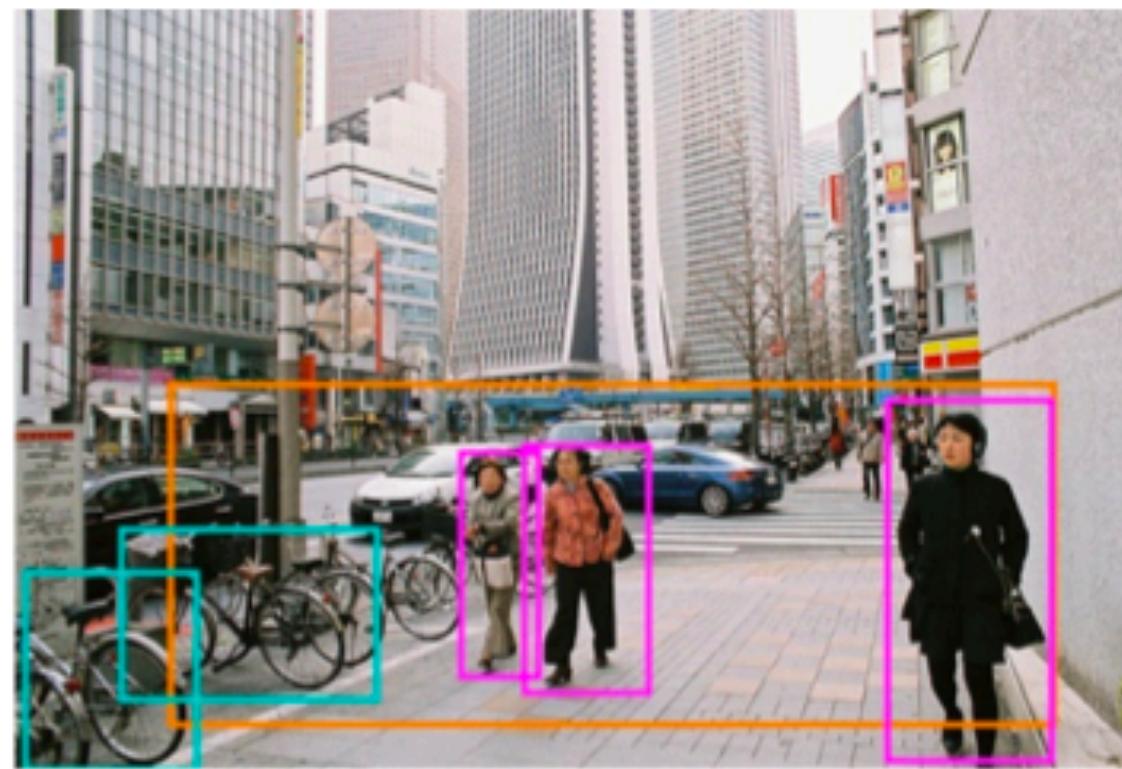
Evaluation

- Ratio of correctly grounded phrases to the total phrases

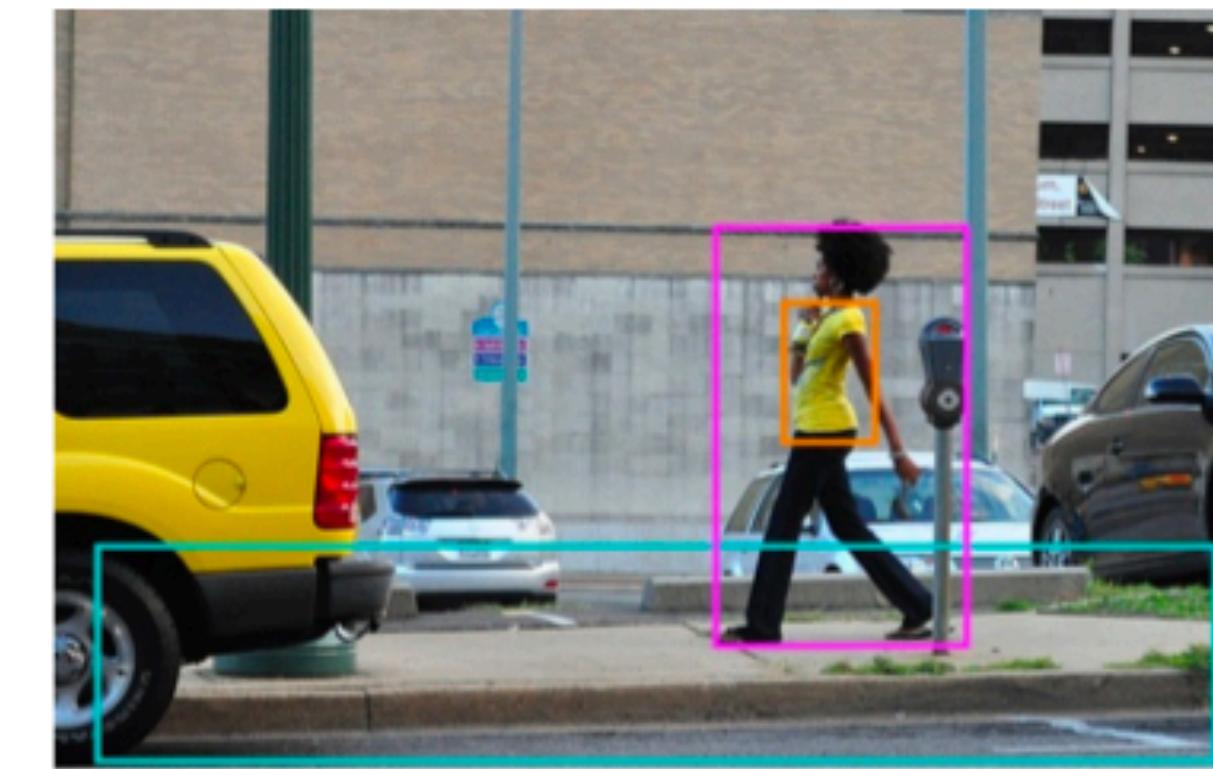
Qualitative Results: Flickr30K



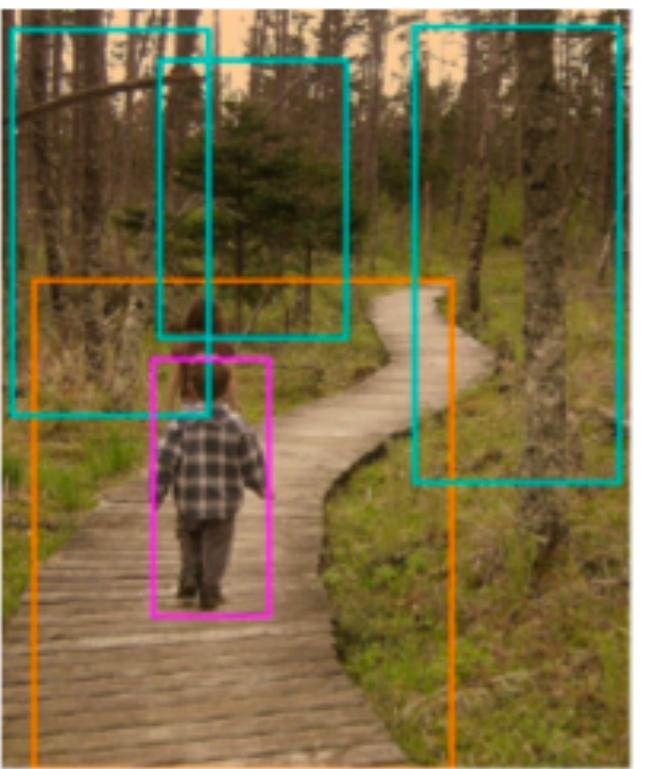
(a) A man wearing a black-jacket has a smile on his face.



(b) People are walking on the street, with bikes parked up to the left of the picture.



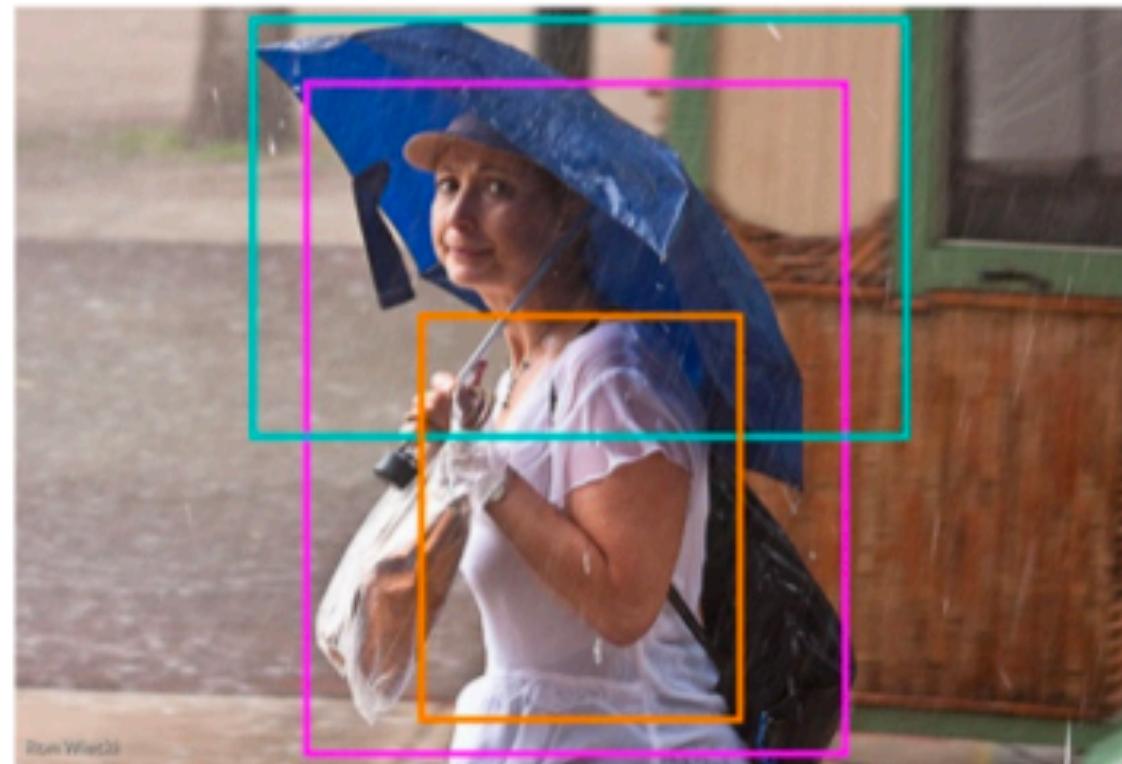
(c) A woman in a yellow shirt is walking down the sidewalk.



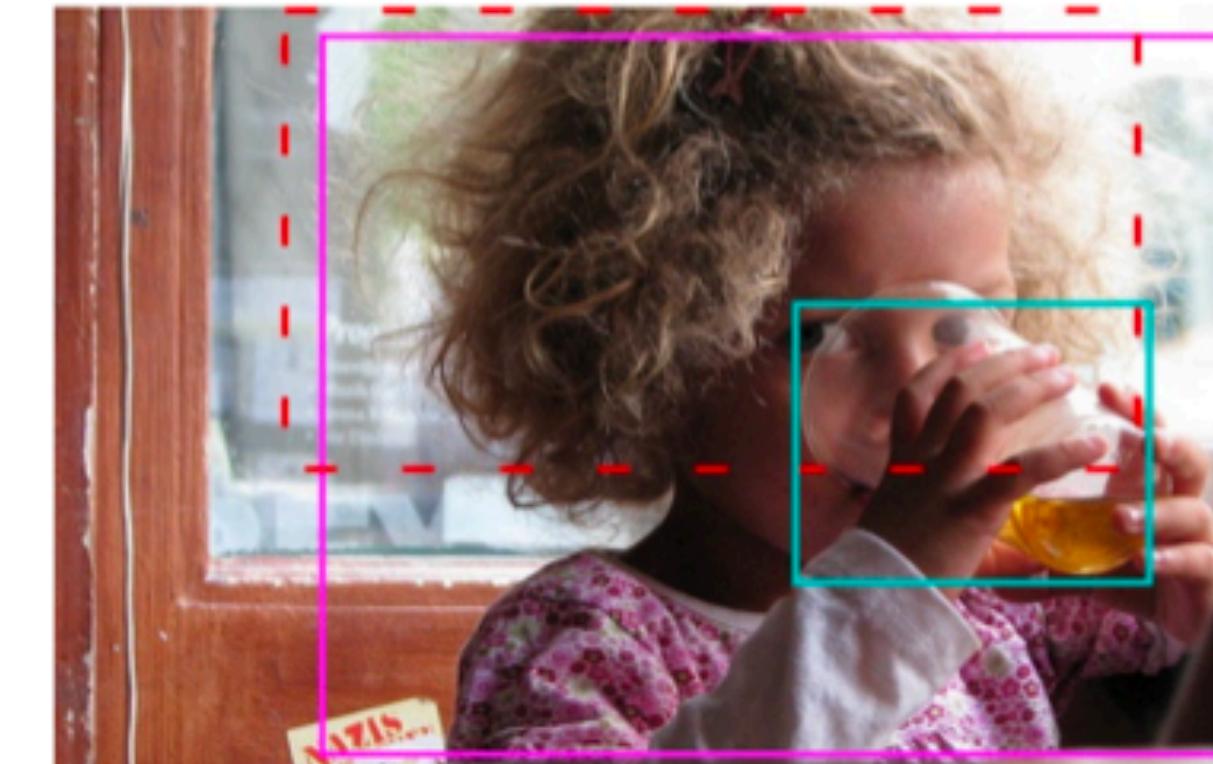
(d) A young boy is walking on wooden path in the middle of trees.



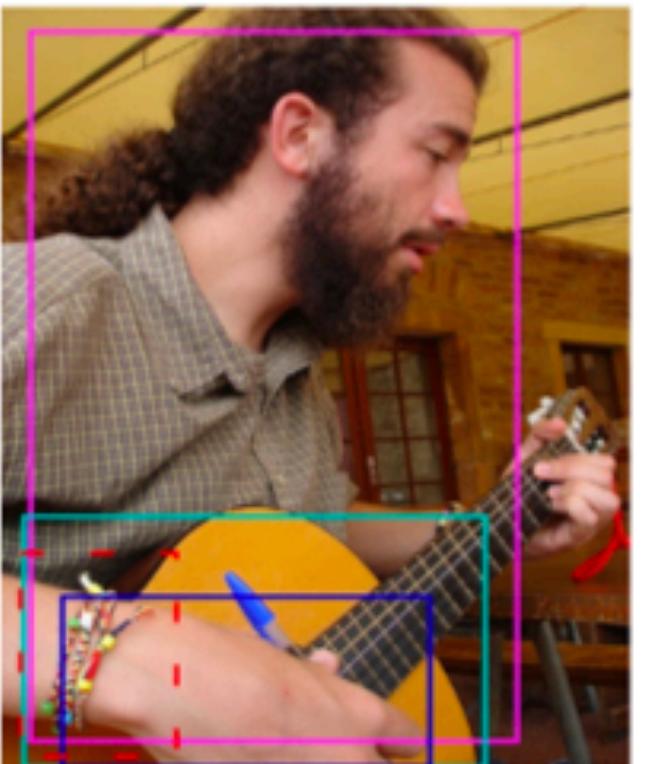
(e) Two women in colorful clothing are dancing inside a circle of other women.



(f) Lady wearing white shirt with blue umbrella in the rain.



(g) Young girl with curly hair is drinking out of a plastic cup.



(h) The bearded man keeps his blue Bic pen in hand while he plays the guitar.

Quantitative Results

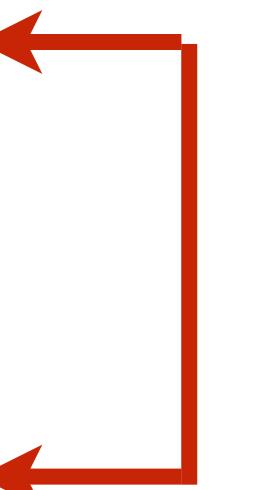
Flickr30k Entities:

Method	Accuracy
SMPL [27]	42.08
NonlinearSP [26]	43.89
GroundeR [23]	47.81
MCB [7]	48.69
RtP [21]	50.89
Similarity Network [25]	51.05
IGOP [34]	53.97
SPC+PPC [20]	55.49
SS+QRN (VGGdet) [4]	55.99
CITE [19]	59.27
SeqGROUND	61.60
CITE [19] (finetuned)	61.89
QRC Net [4] (finetuned)	65.14
G³RAPHGROUND++	66.67

Quantitative Results

Flickr30k Entities:

Method	Accuracy
SMPL [27]	42.08
NonlinearSP [26]	43.89
GroundeR [23]	47.81
MCB [7]	48.69
RtP [21]	50.89
Similarity Network [25]	51.05
IGOP [34]	53.97
SPC+PPC [20]	55.49
SS+QRN (VGGdet) [4]	55.99
CITE [19]	59.27
SeqGROUND	61.60
CITE [19] (finetuned)	61.89
QRC Net [4] (finetuned)	65.14
G³RAPHGROUND++	66.67



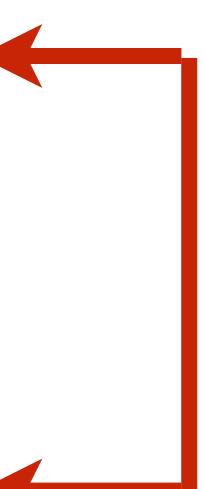
Quantitative Results

Flickr30k Entities:

Method	Accuracy
SMPL [27]	42.08
NonlinearSP [26]	43.89
GroundeR [23]	47.81
MCB [7]	48.69
RtP [21]	50.89
Similarity Network [25]	51.05
IGOP [34]	53.97
SPC+PPC [20]	55.49
SS+QRN (VGGdet) [4]	55.99
CITE [19]	59.27
SeqGROUND	61.60
CITE [19] (finetuned)	61.89
QRC Net [4] (finetuned)	65.14
G³RAPHGROUND++	66.67

ReferIt Game:

Method	Accuracy
SCRC [9]	17.93
MCB + Reg + Spatial [3]	26.54
GroundeR + Spatial [23]	26.93
Similarity Network + Spatial [25]	31.26
CGRE [17]	31.85
MNN + Reg + Spatial [3]	32.21
EB+QRN (VGGcls-SPAT) [4]	32.21
CITE [19]	34.13
IGOP [34]	34.70
QRC Net [4] (finetuned)	44.07
G³RAPHGROUND++	44.91



Ablation

Method	Flickr30k	ReferIt
GG - VisualG - FusionG	56.32	32.89
GG - VisualG	62.23	38.82
GG - FusionG	59.13	36.54
GG - PhraseG	60.82	38.12
GGFusionBase	60.41	38.65
GG - ImageContext	62.32	40.92
GG - PhraseContext	62.73	<i>n.a.</i>
G ³ RAPHGROUND (GG)	63.65	41.79
G³RAPHGROUND++	66.67	44.91

Ablation

Method	Flickr30k	ReferIt
GG - VisualG - FusionG	56.32	32.89
GG - VisualG	62.23	38.82
GG - FusionG	59.13	36.54
GG - PhraseG	60.82	38.12
GGFusionBase	60.41	38.65
GG - ImageContext	62.32	40.92
GG - PhraseContext	62.73	<i>n.a.</i>
G³RAPHGROUND (GG)	63.65	41.79
G³RAPHGROUND++	66.67	44.91



Visualizing Graph Attention



(a) A young boy is looking at a man painted in all gold.



(b) A man is checking his blue sneakers next to two men having a conversation.



(c) A brown dog jumps high on a field of grass.



(d) A woman stands in a field near a car and looks through binoculars.



THE UNIVERSITY OF BRITISH COLUMBIA

Energy-Based Learning for Scene Graph Generation



+++

Mohammed Suhail

Scene Graphs:

A **graph** based data structure for semantically representing image content

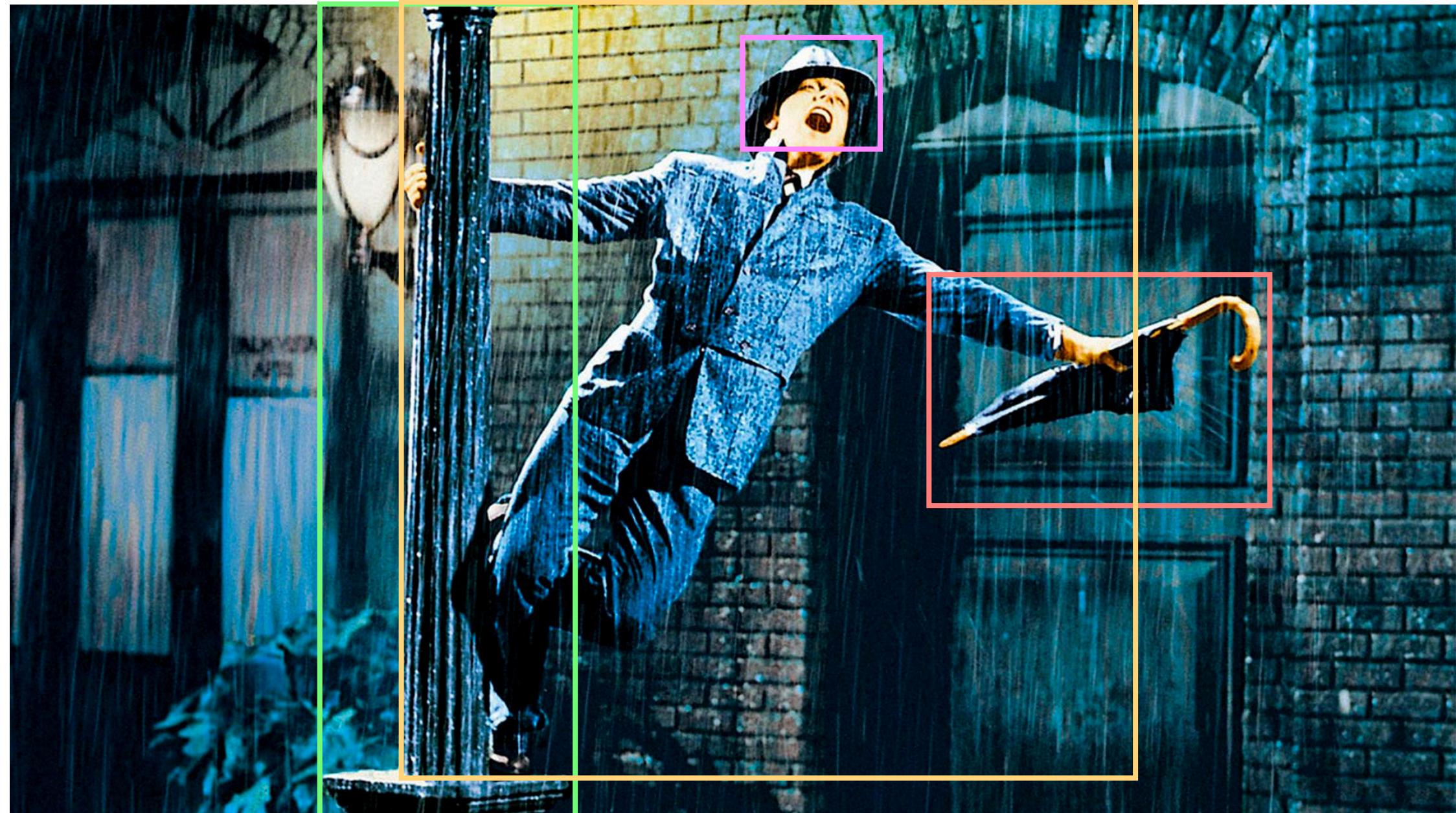
Scene Graphs



Scene Graphs

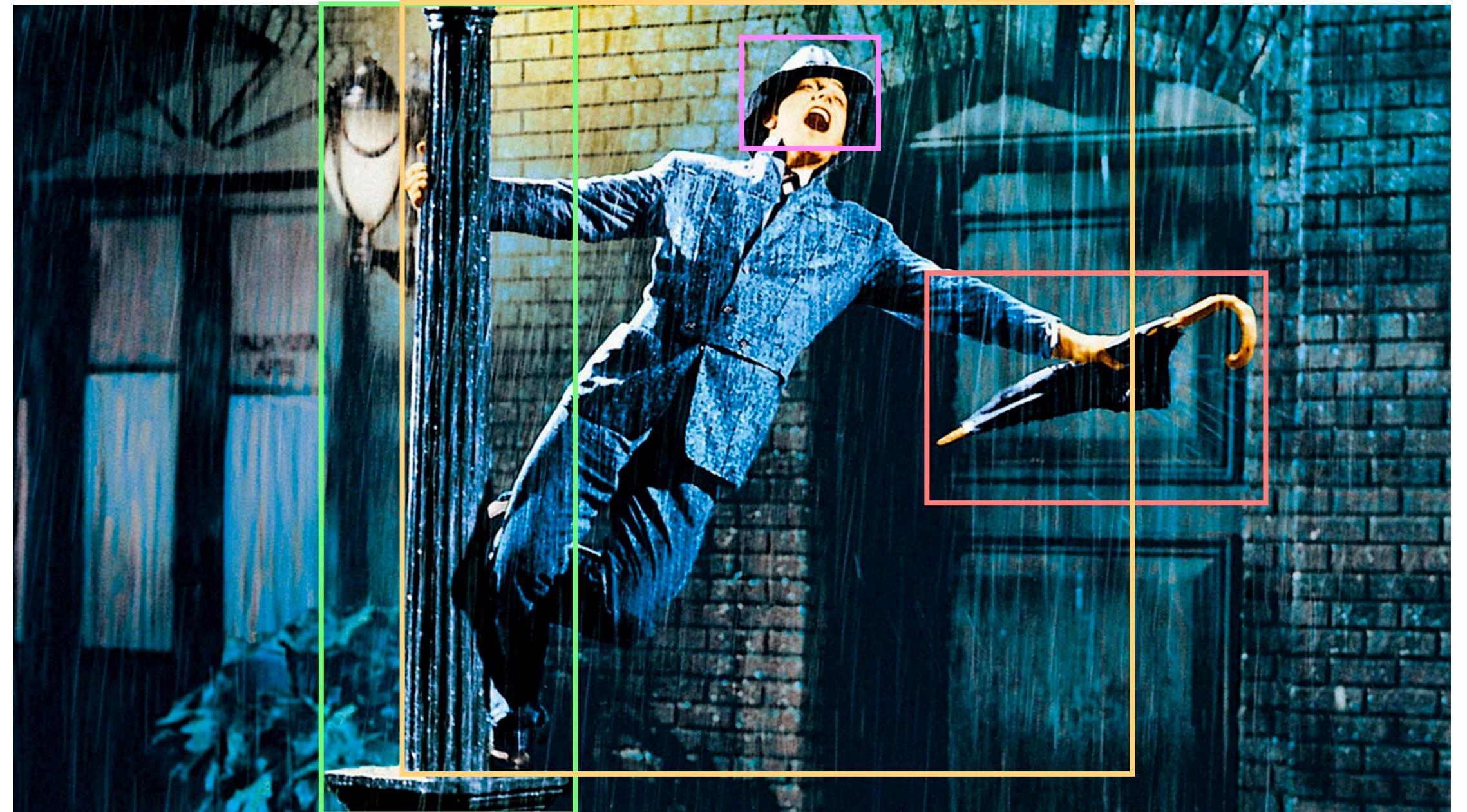


Scene Graphs

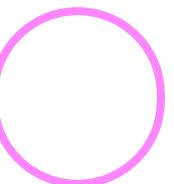


-  → Hat
-  → Umbrella
-  → Lamp post
-  → Person

Scene Graphs



Hat



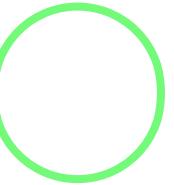
Person



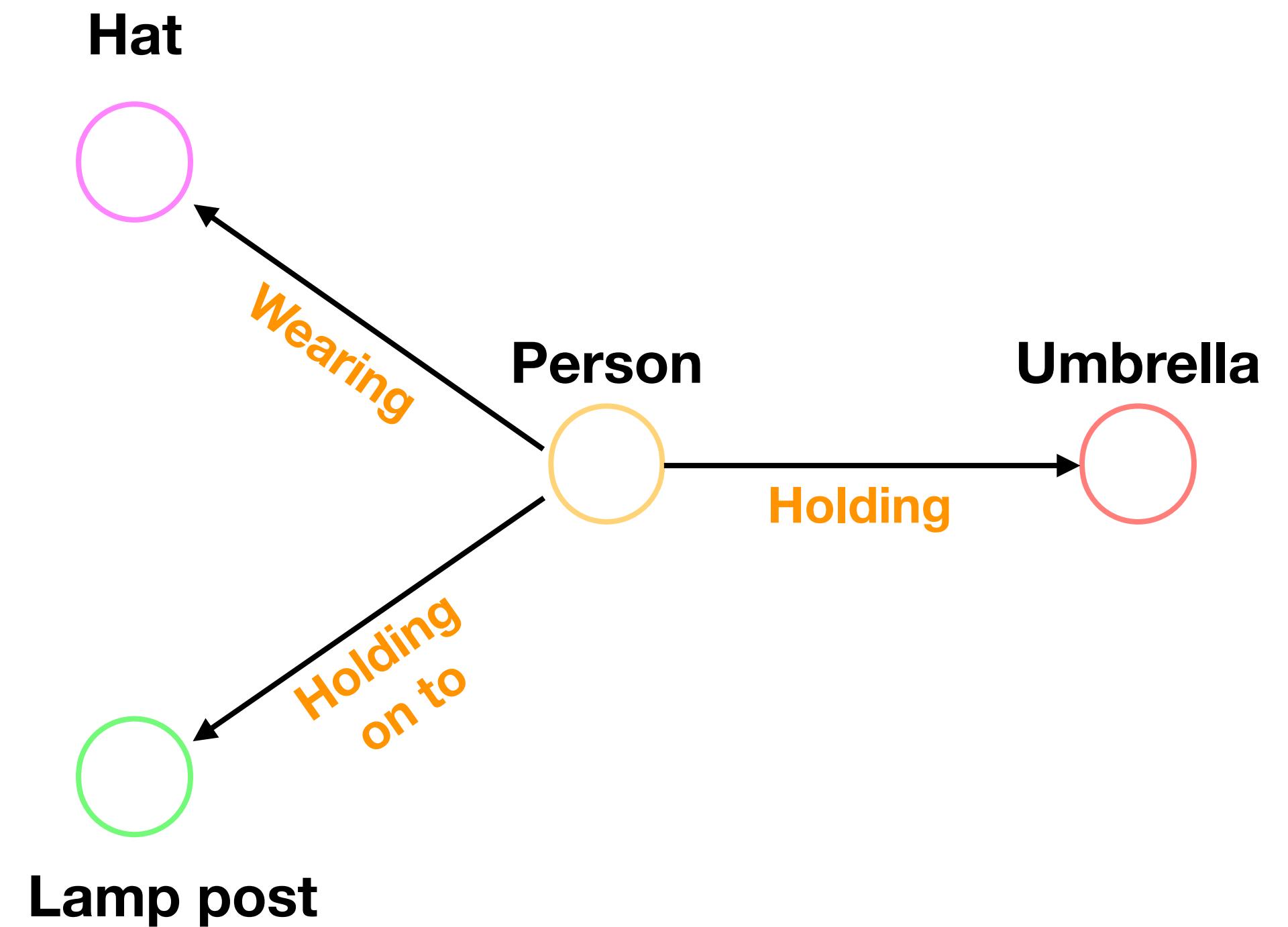
Umbrella



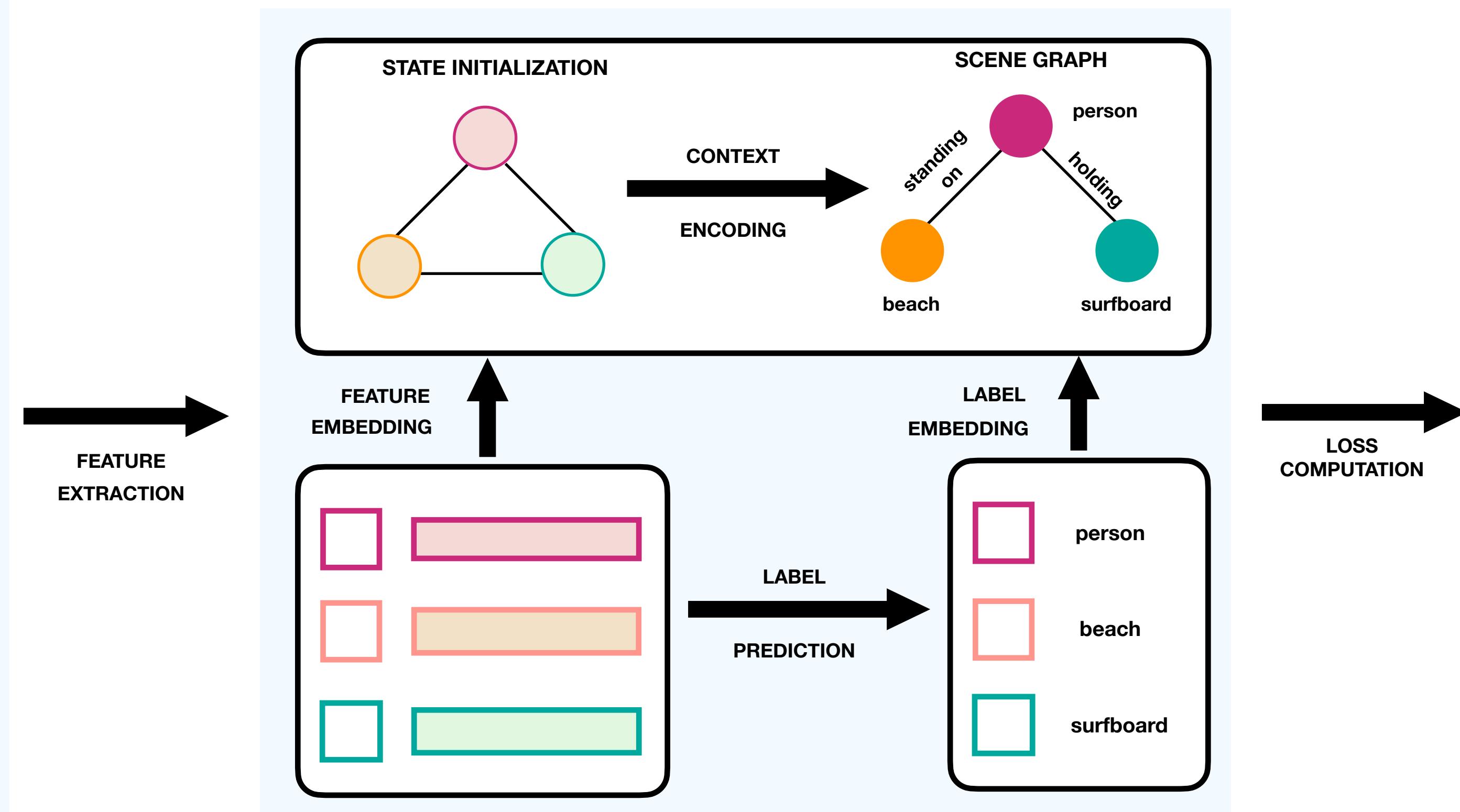
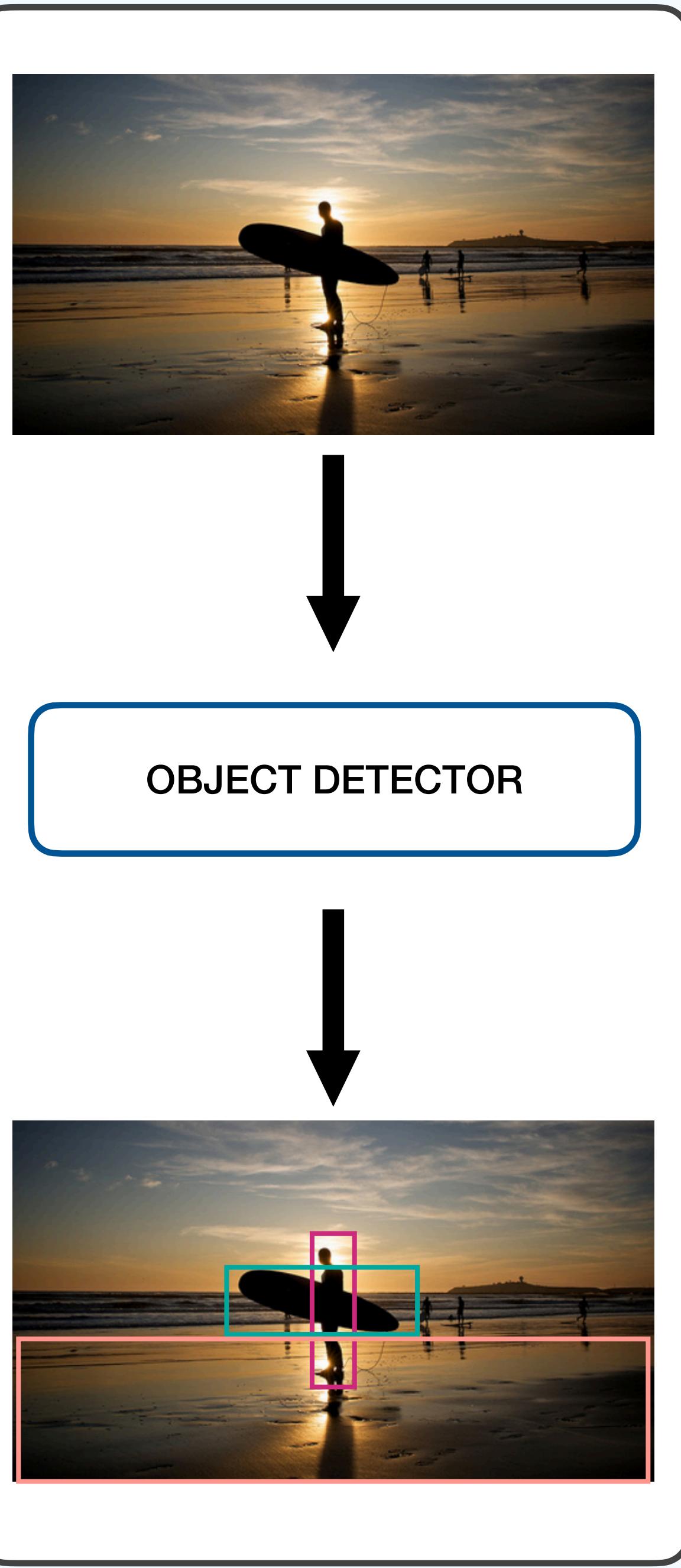
Lamp post

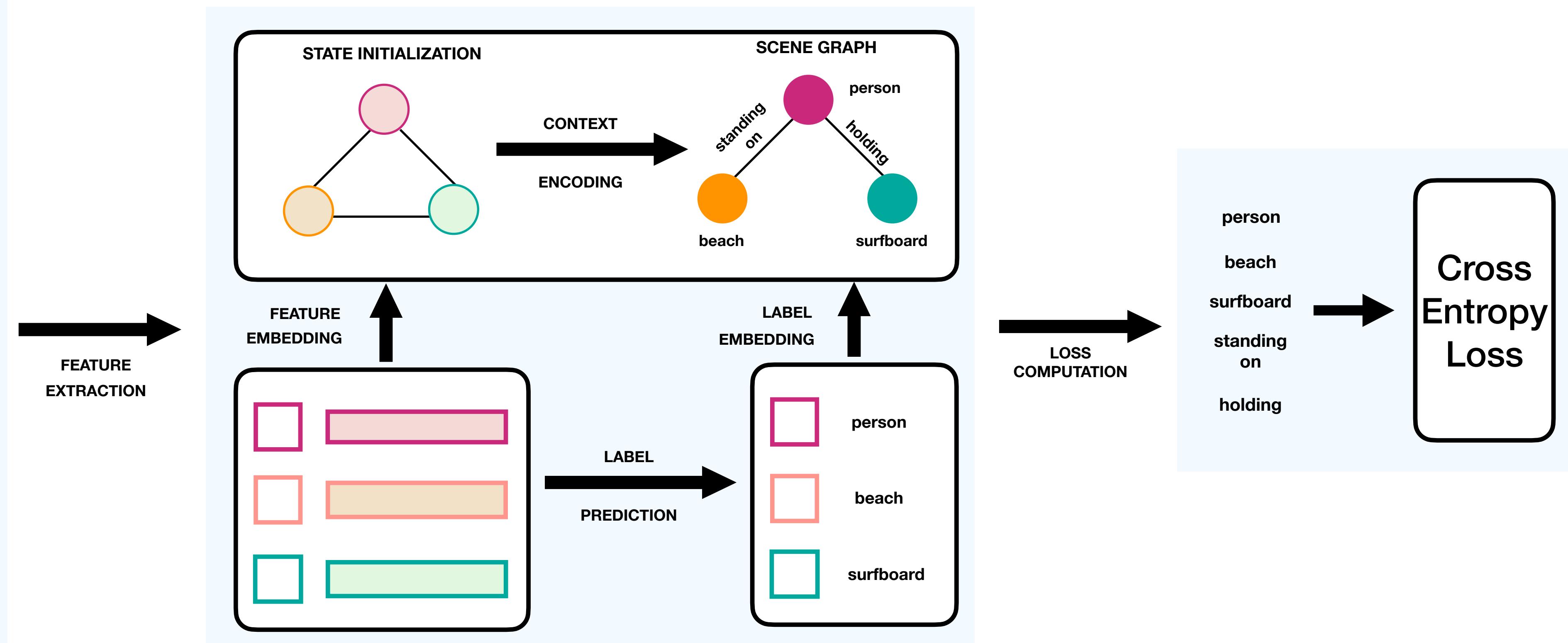
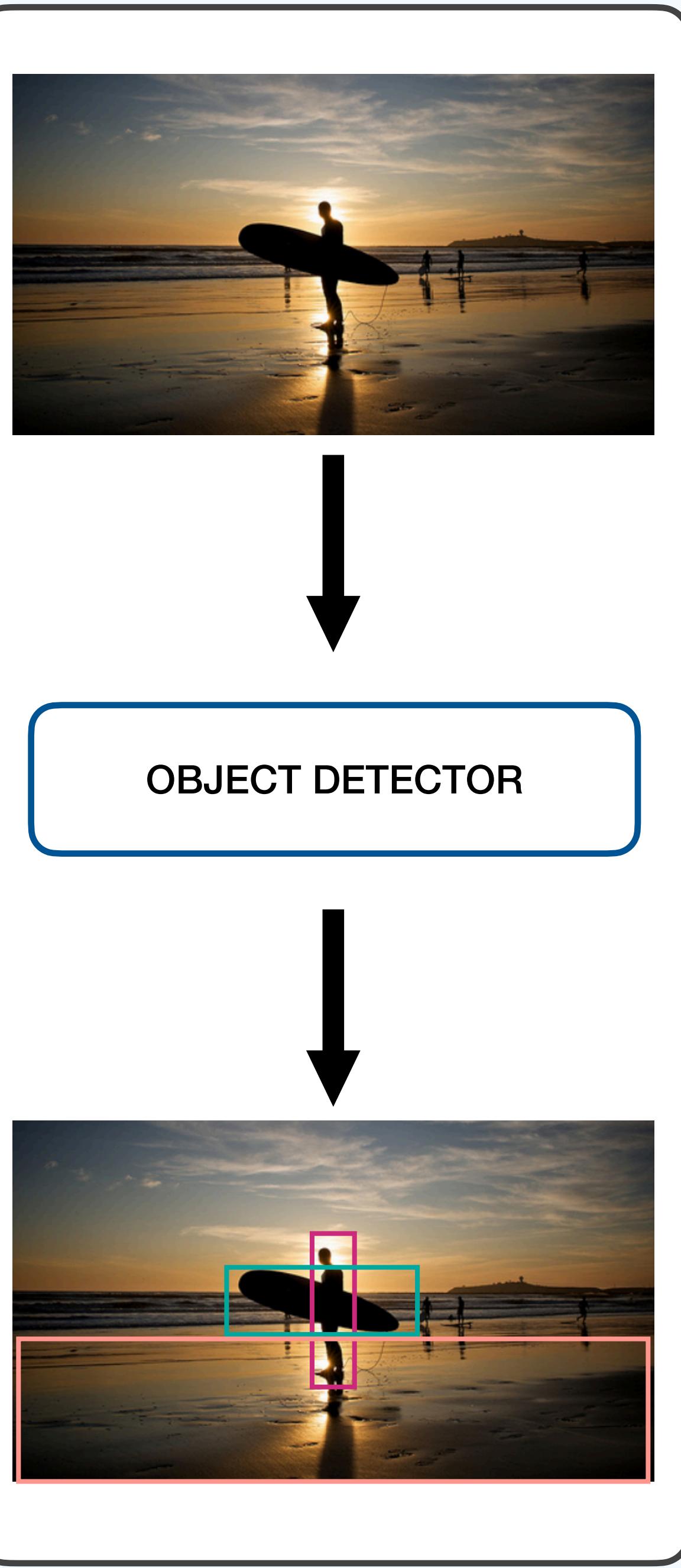


Scene Graphs

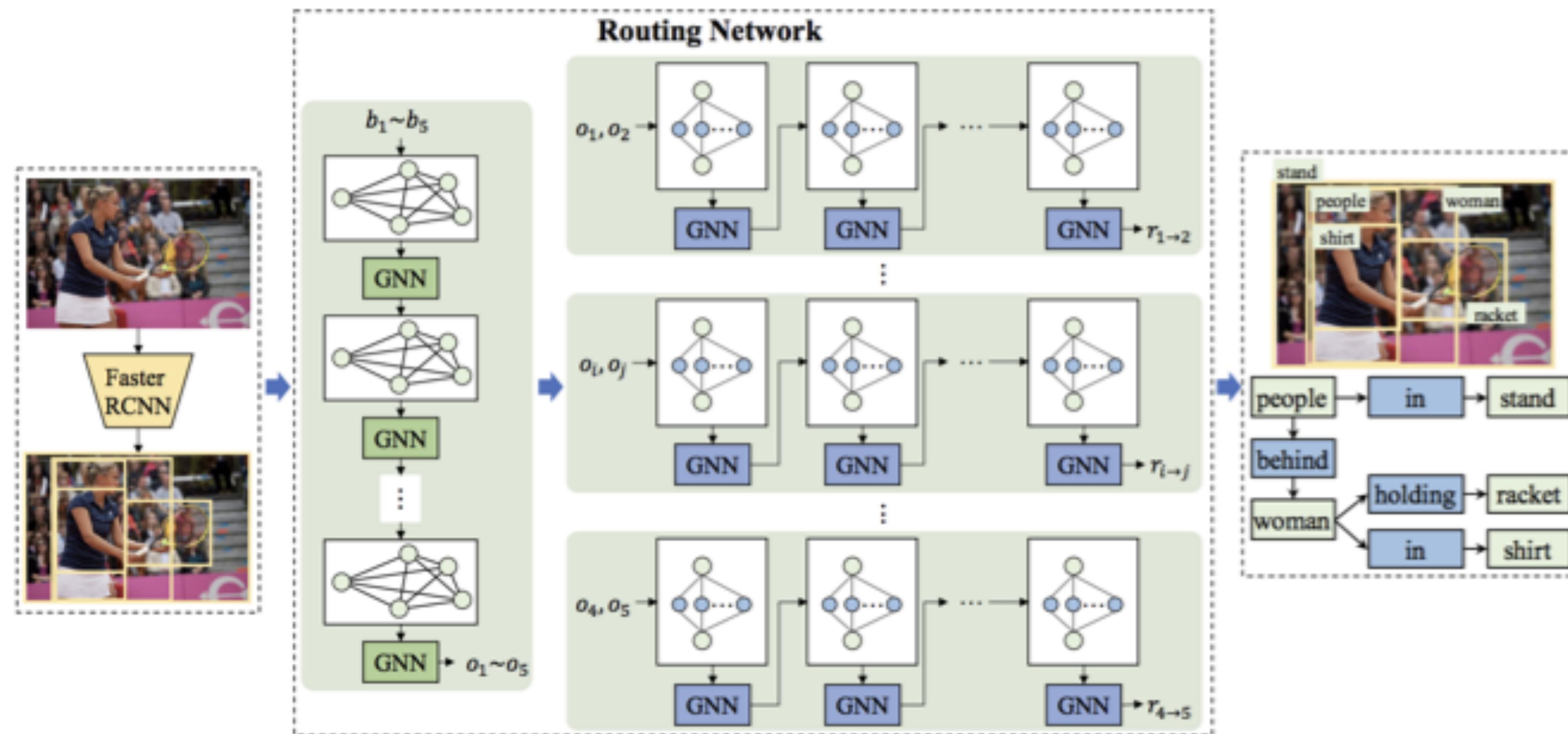


Scene Graph Generation Pipeline

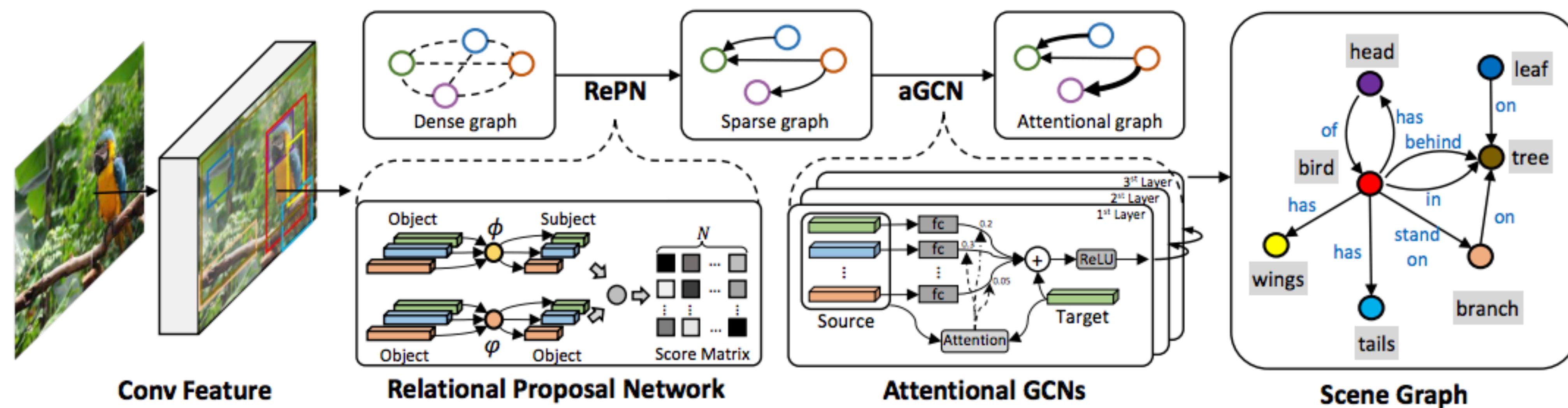




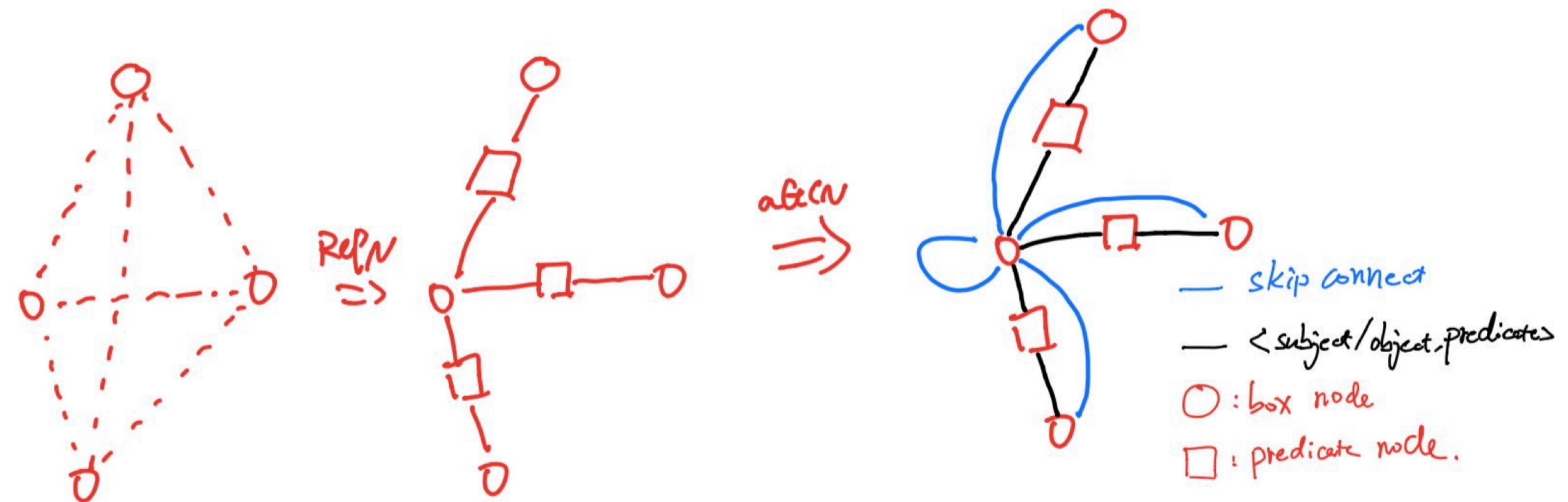
KERN Architecture



Graph RCNN

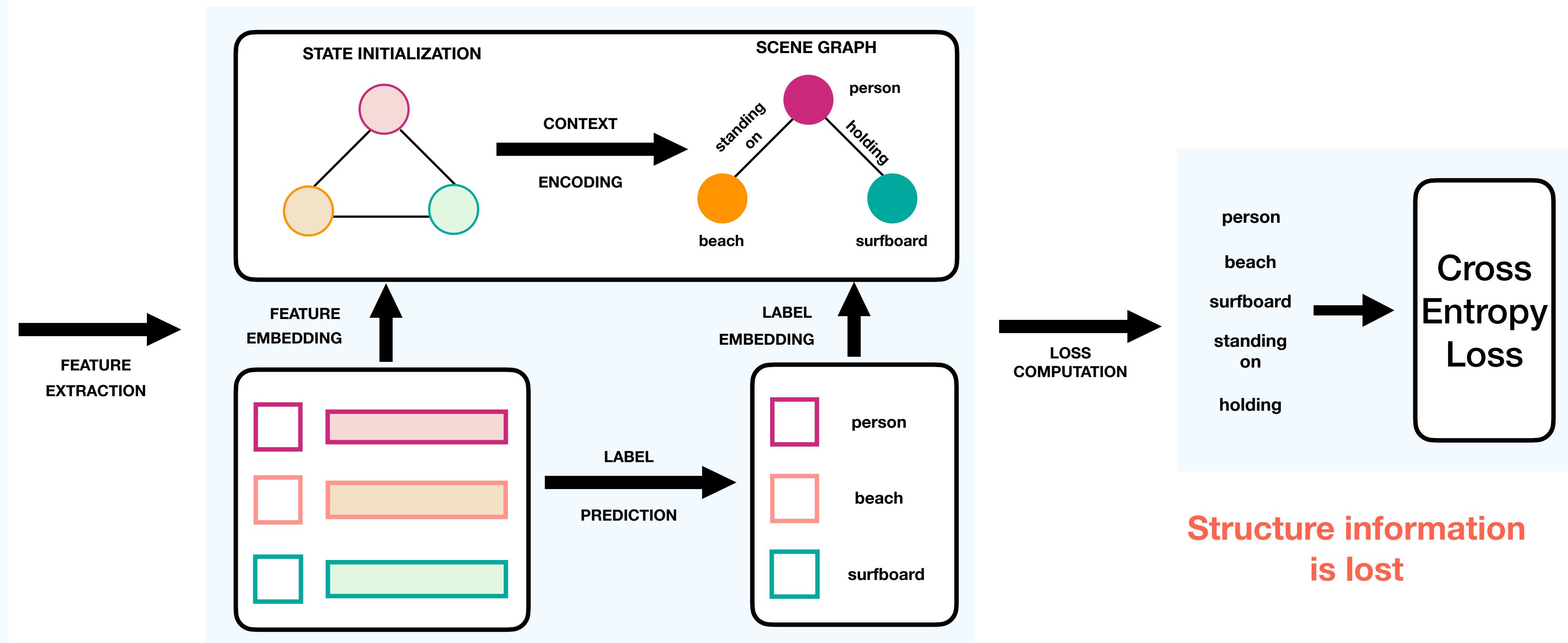
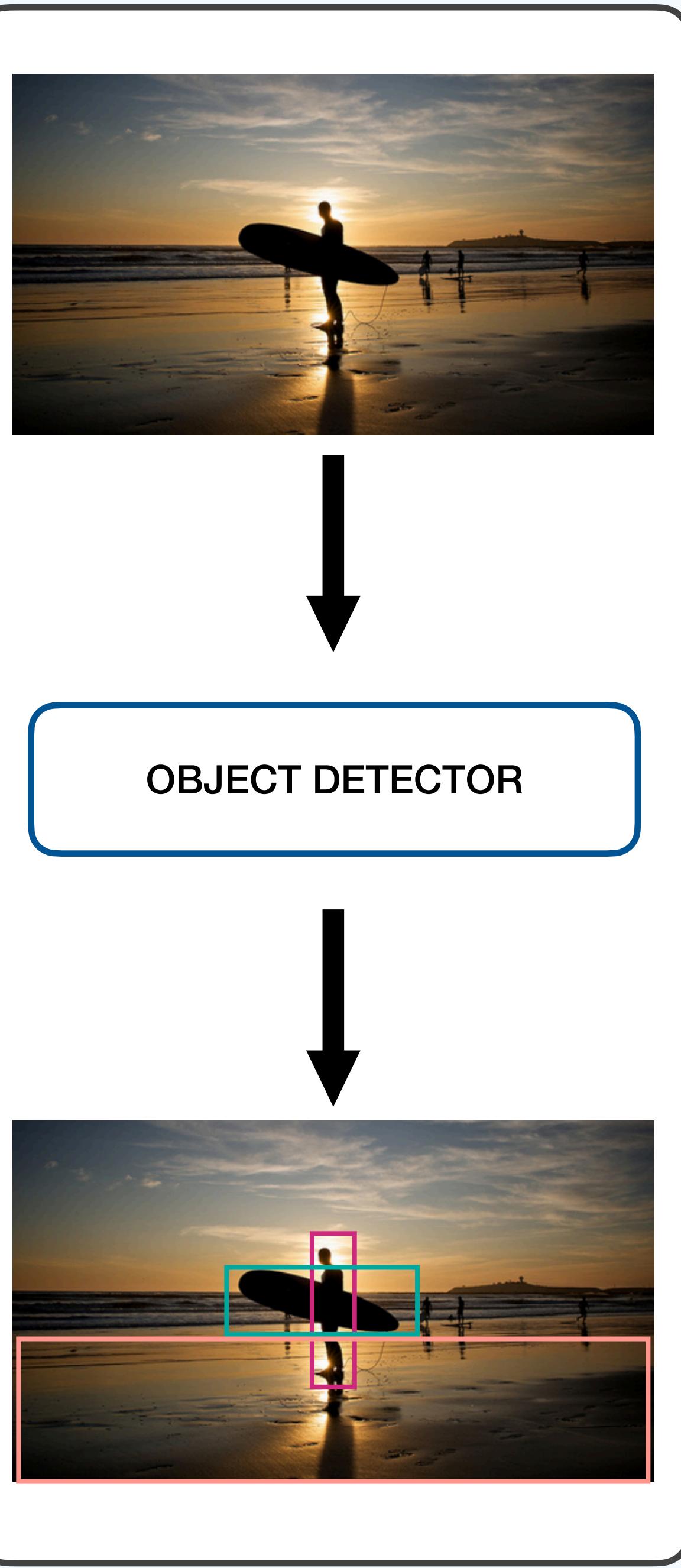


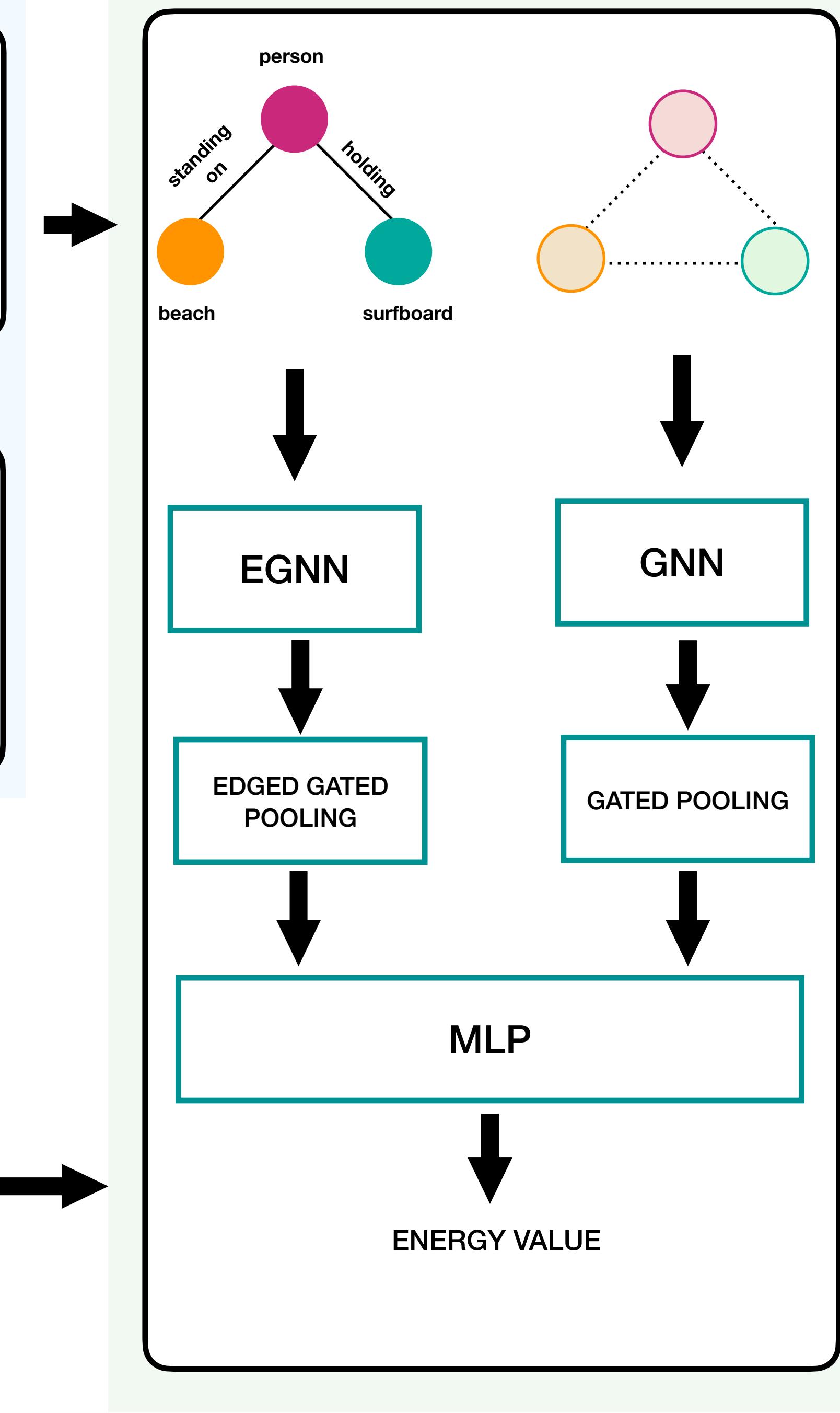
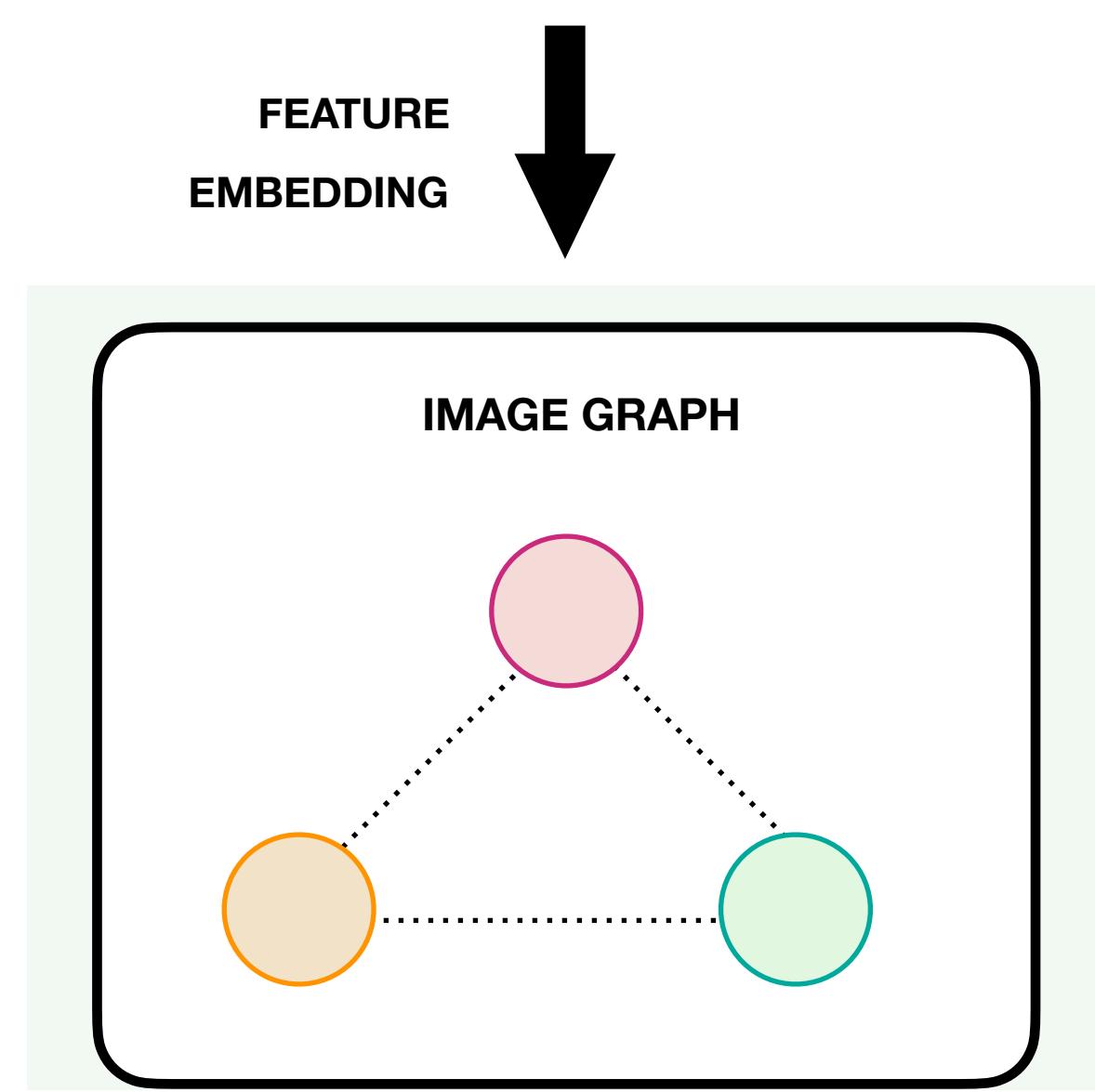
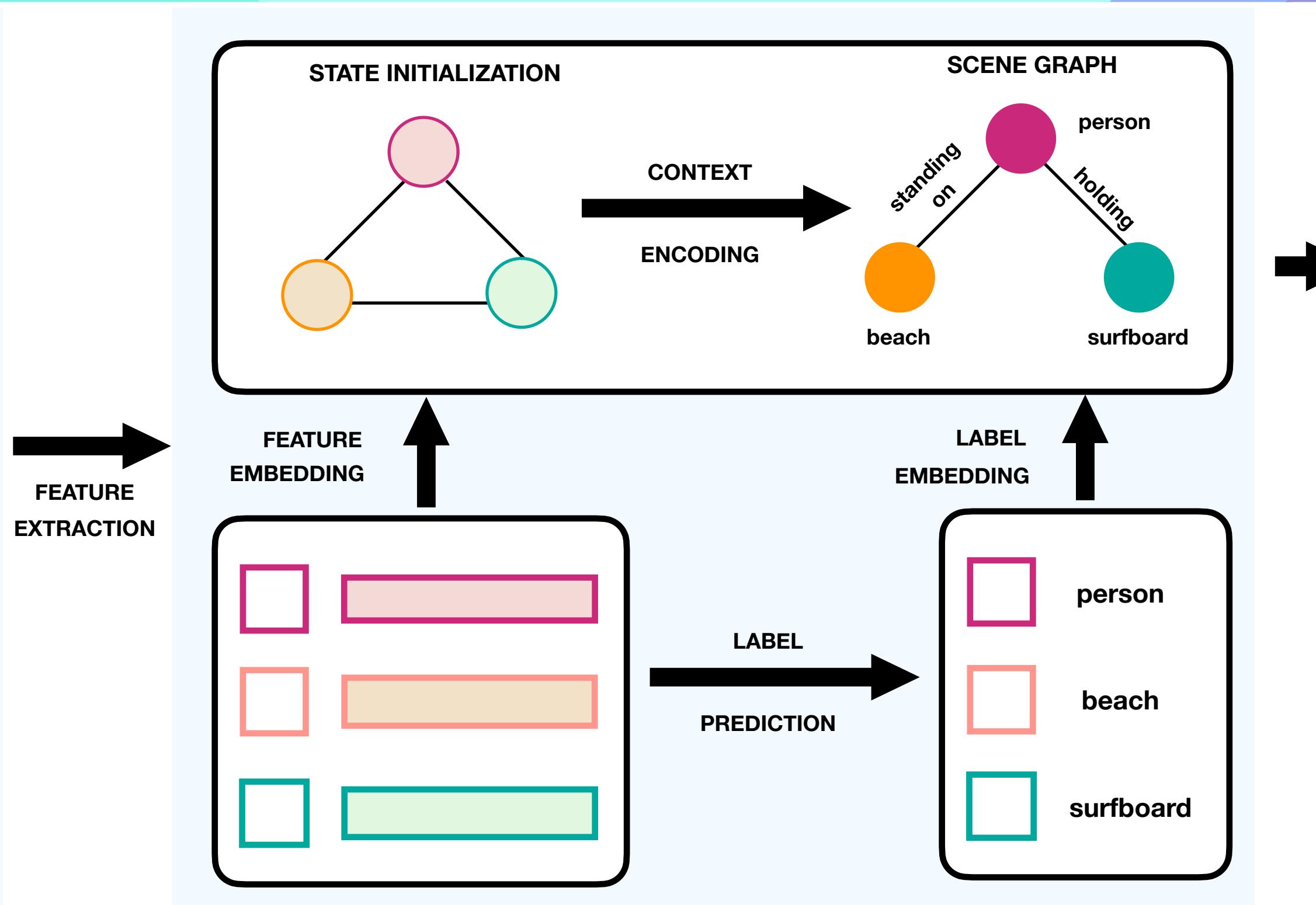
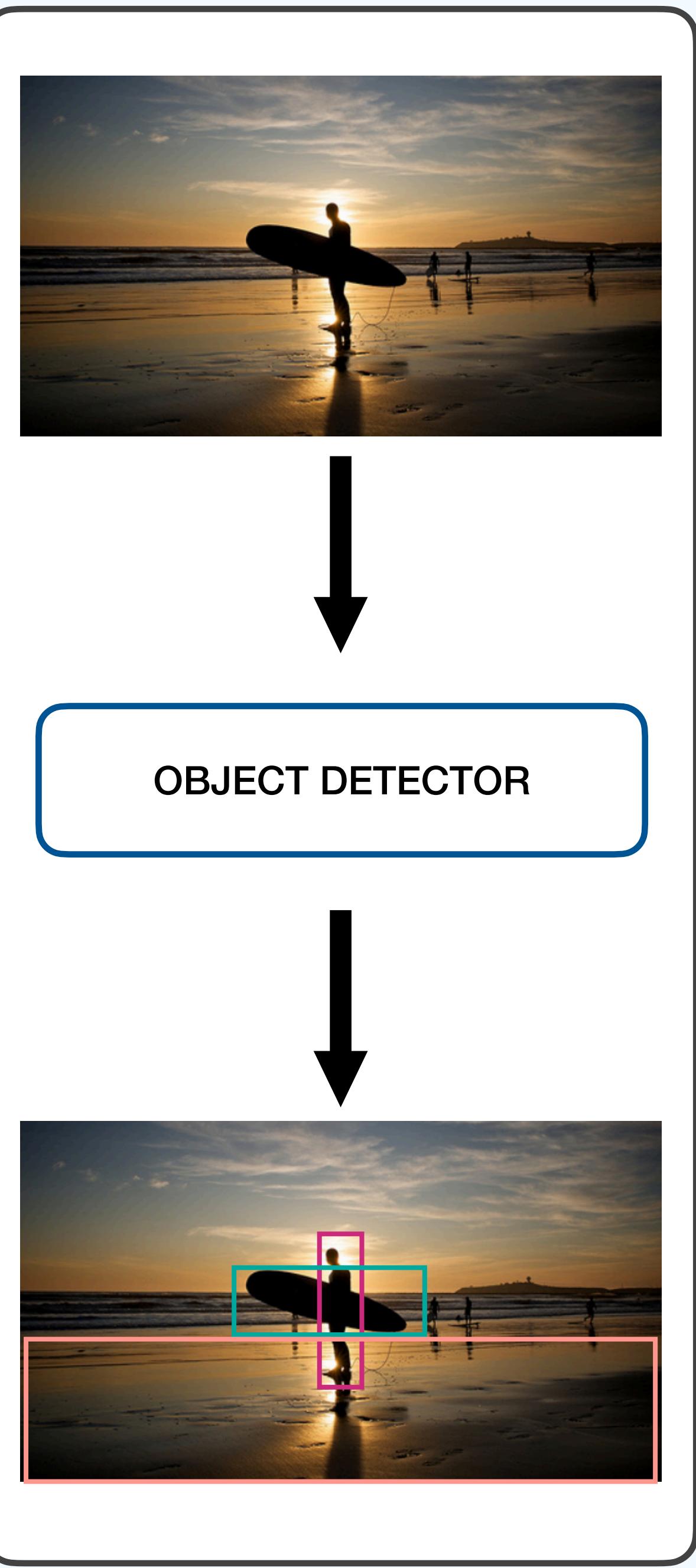
Graph RCNN



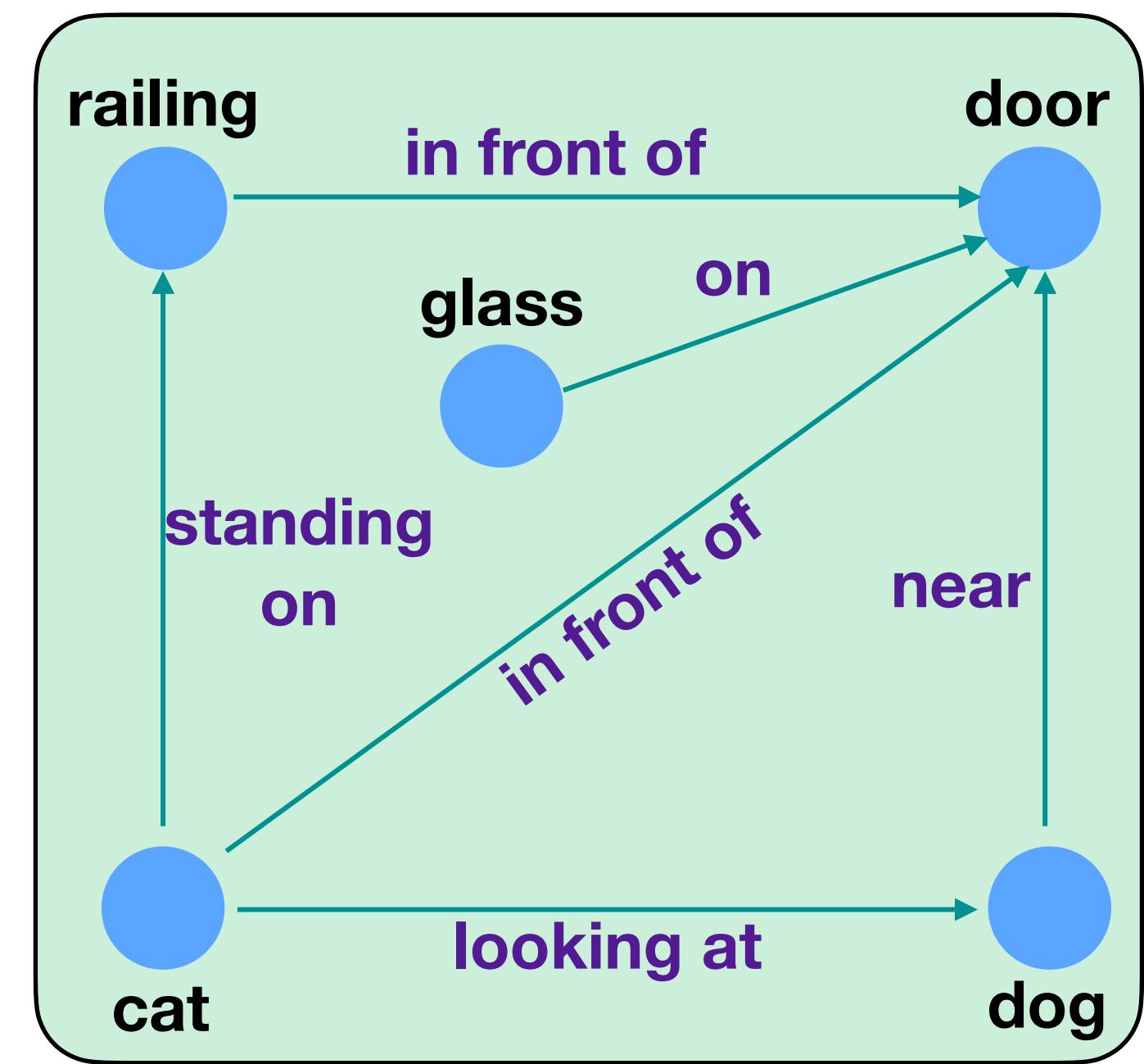
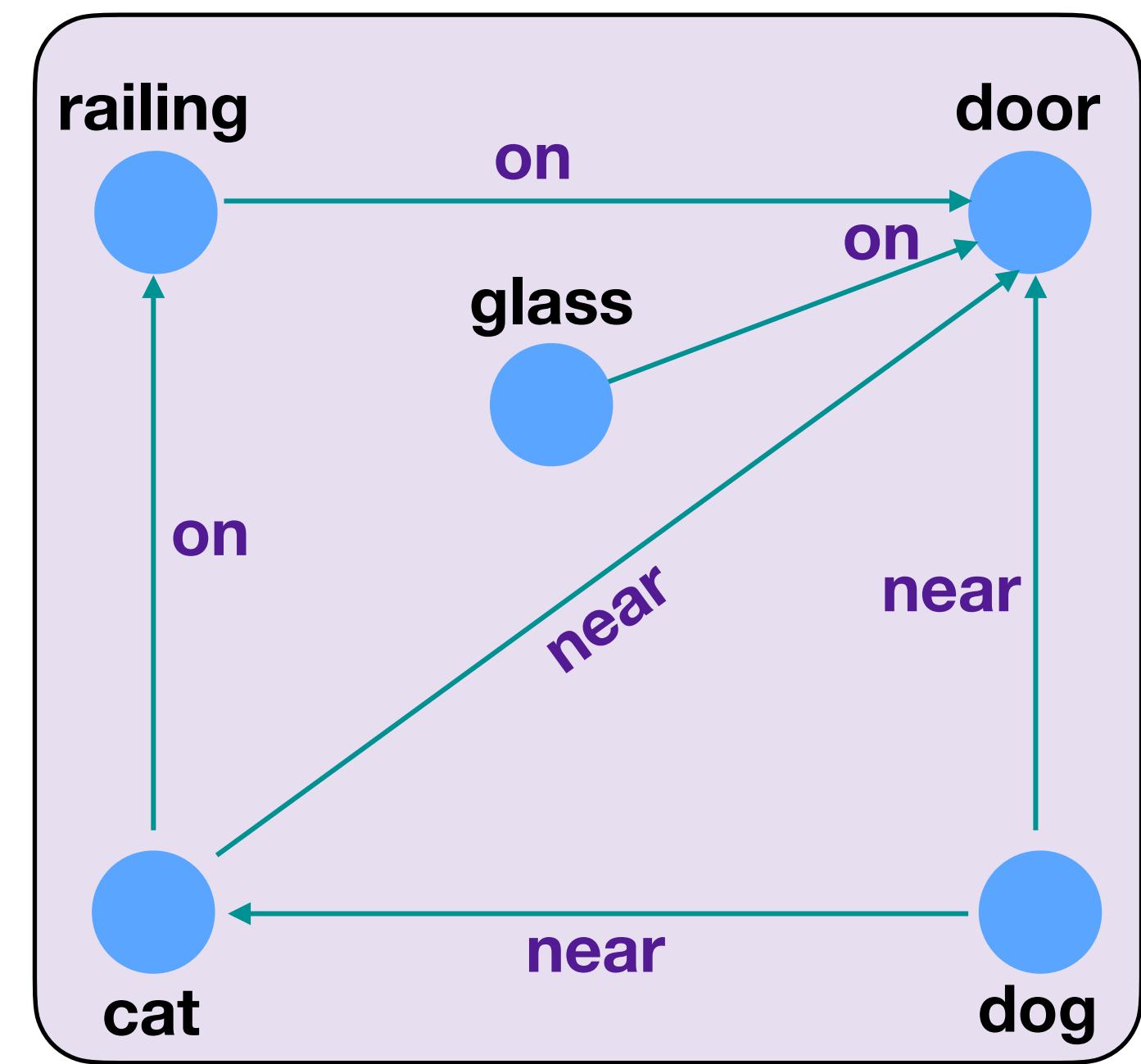
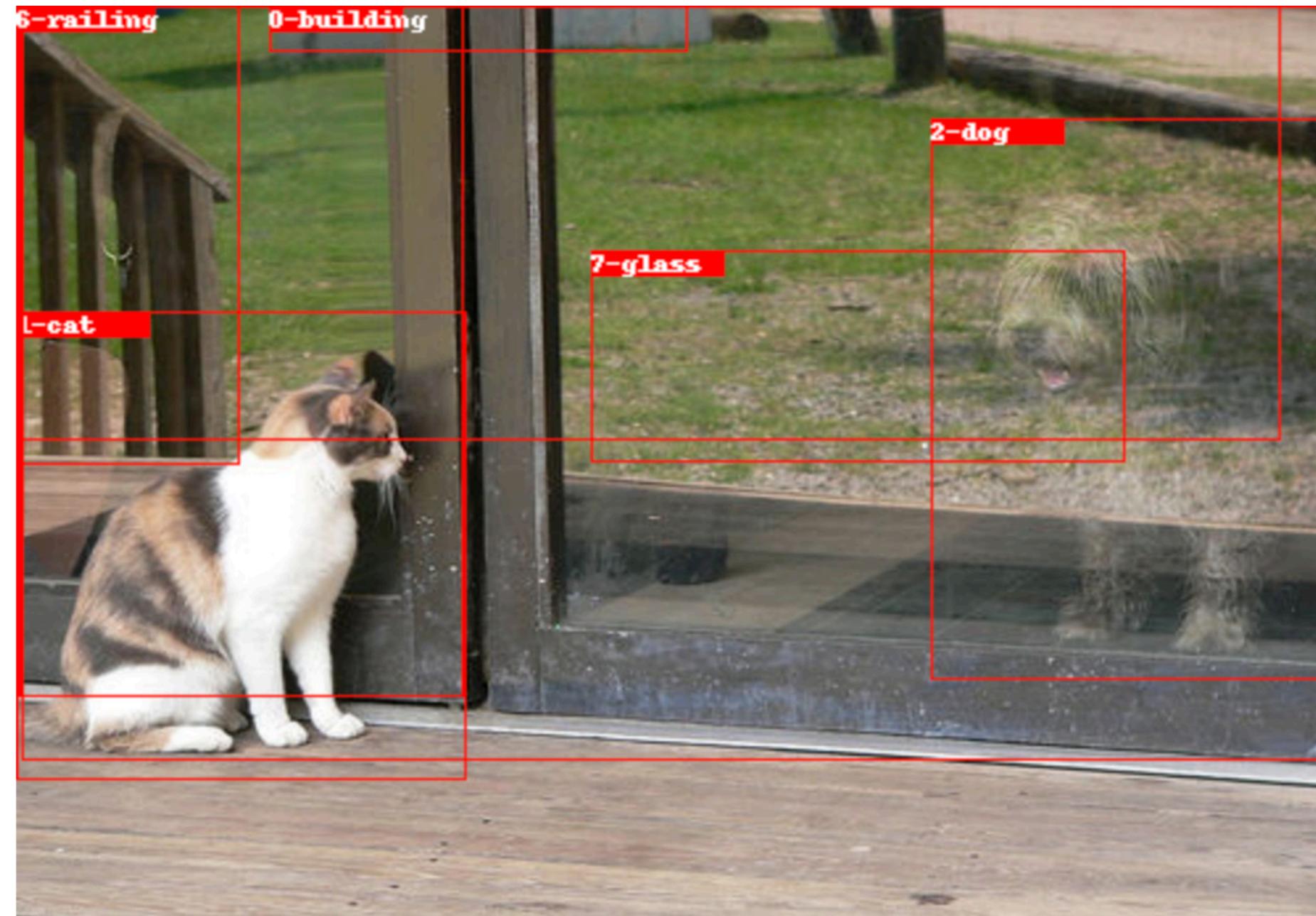
$$z_i^o = \sigma(\underbrace{W^{\text{skip}} Z^o \alpha^{\text{skip}}}_{\text{Message from Other Objects}} + \underbrace{W^{sr} Z^r \alpha^{sr} + W^{or} Z^r \alpha^{or}}_{\text{Messages from Neighboring Relationships}})$$

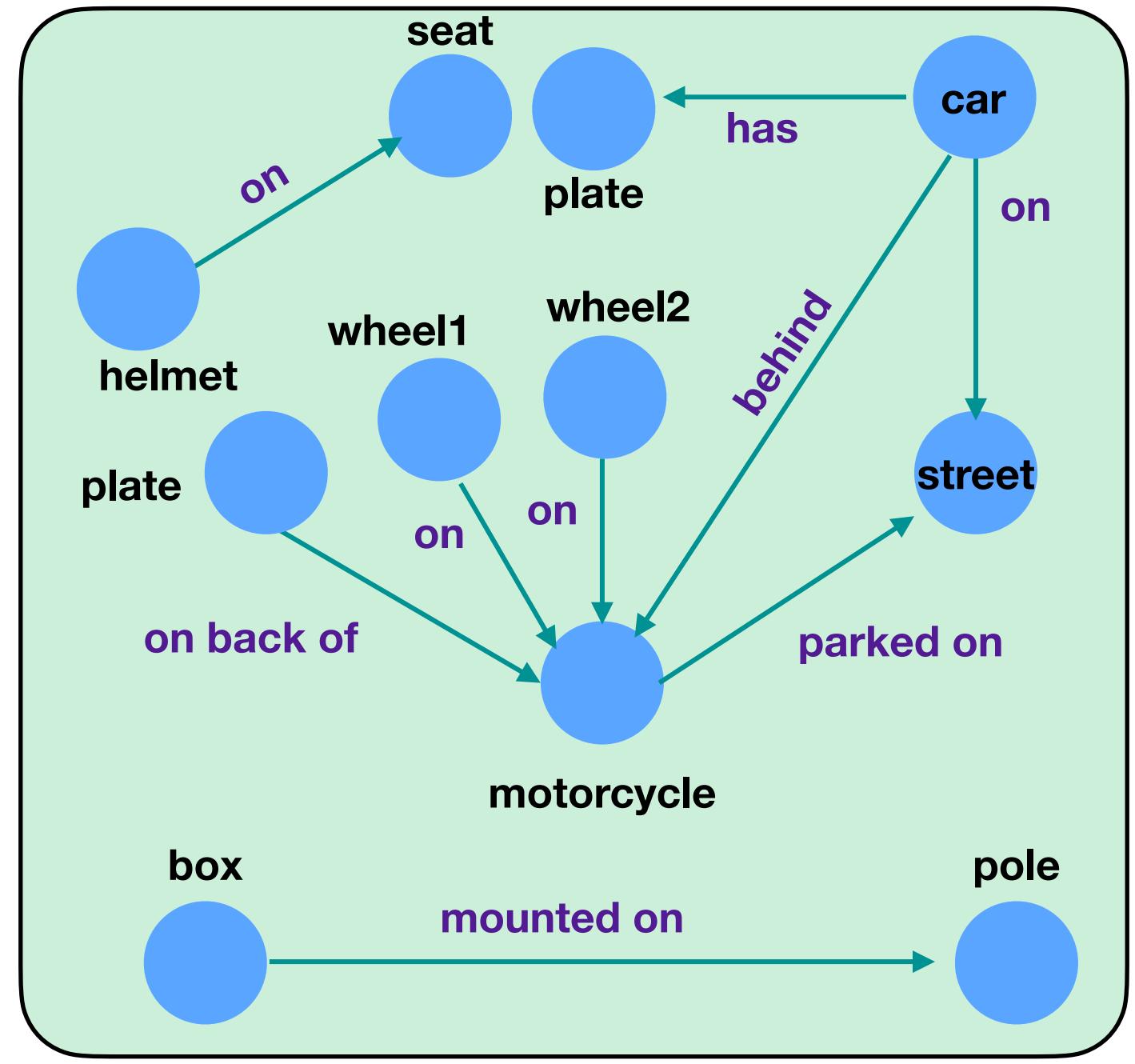
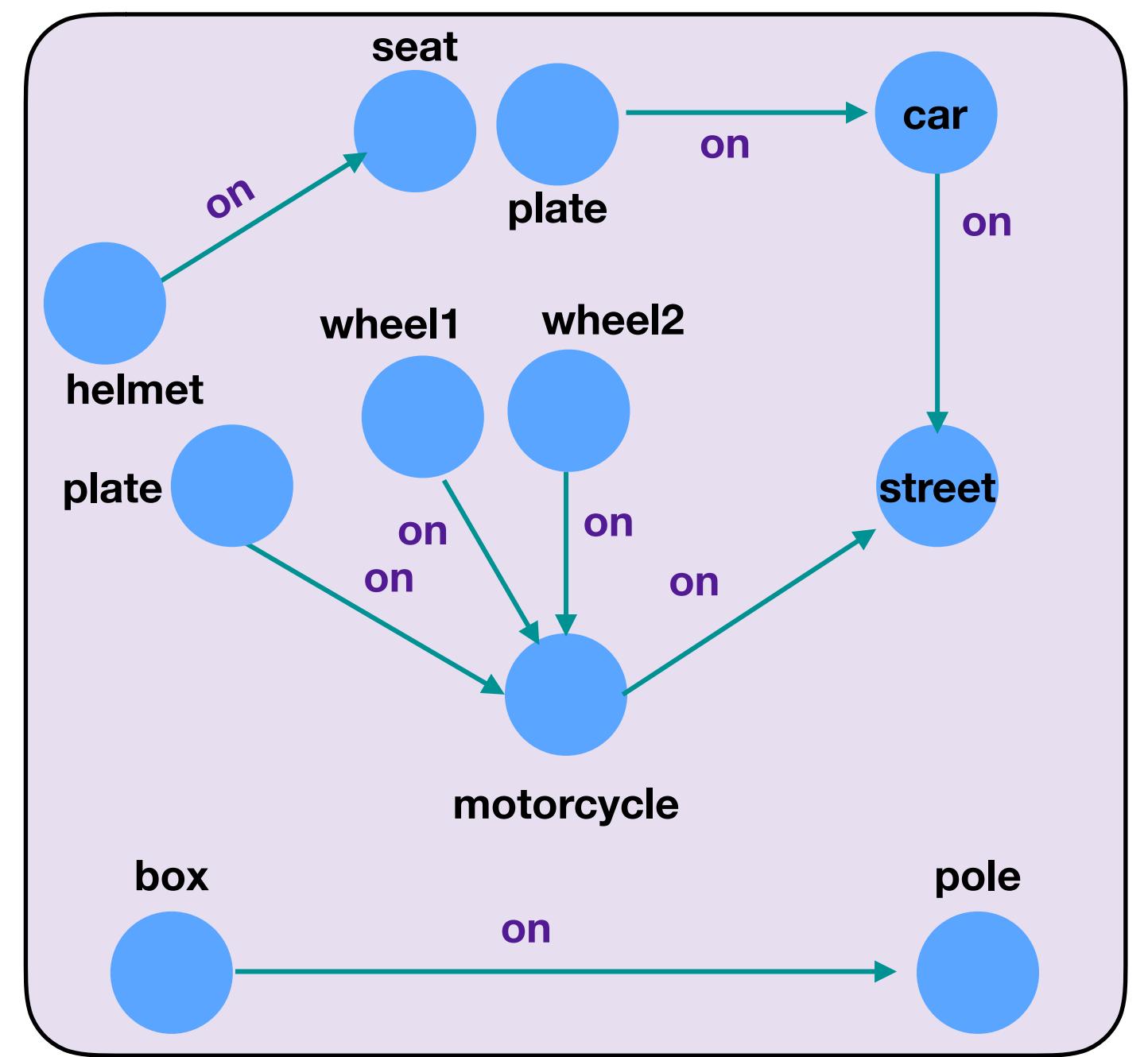
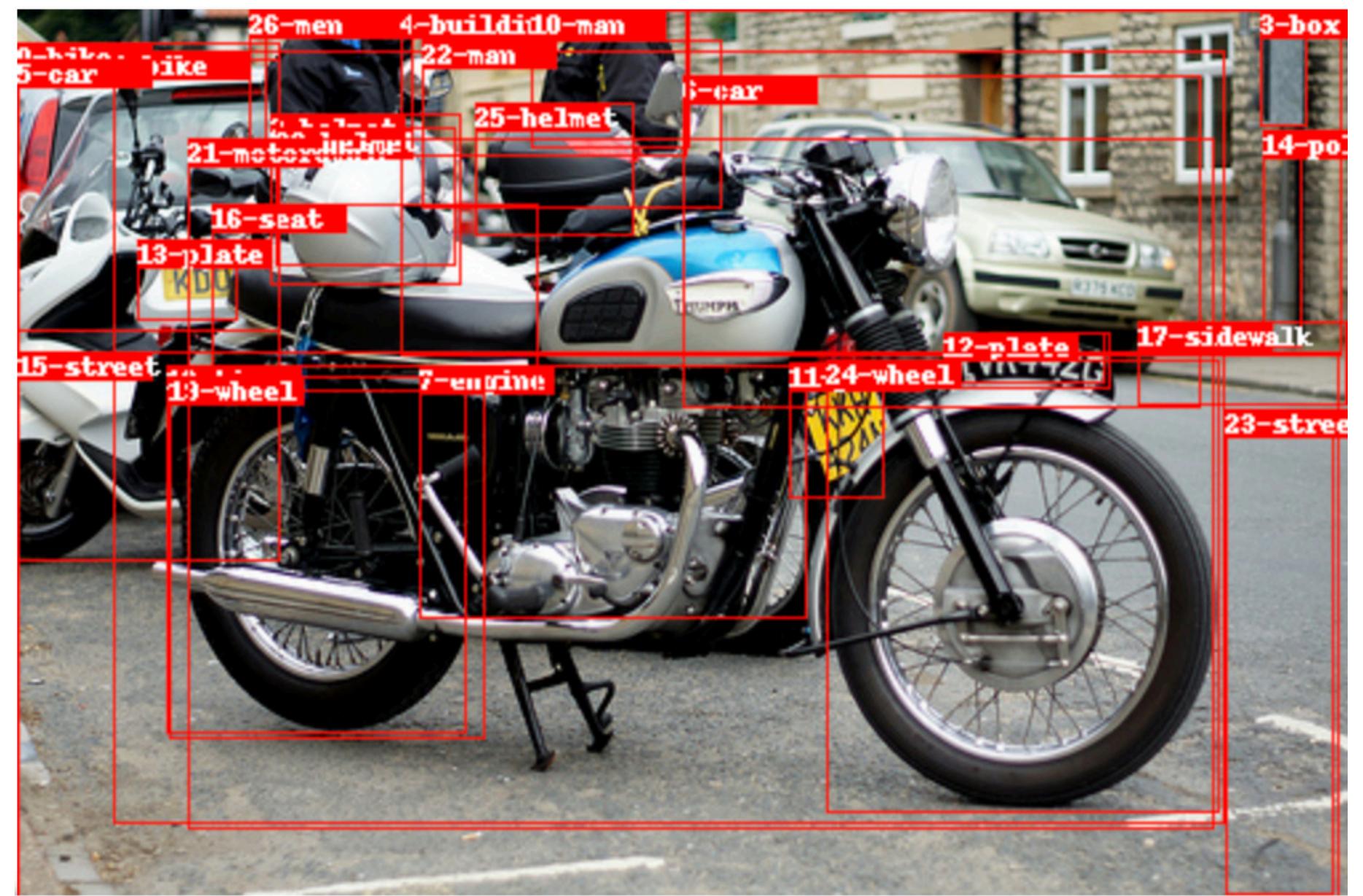
$$z_i^r = \sigma(z_i^r + \underbrace{W^{rs} Z^o \alpha^{rs} + W^{ro} Z^o \alpha^{ro}}_{\text{Messages from Neighboring Objects}}).$$

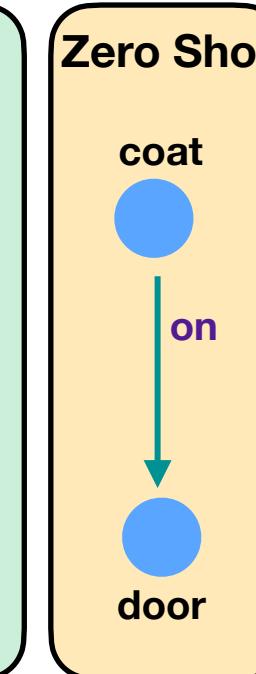
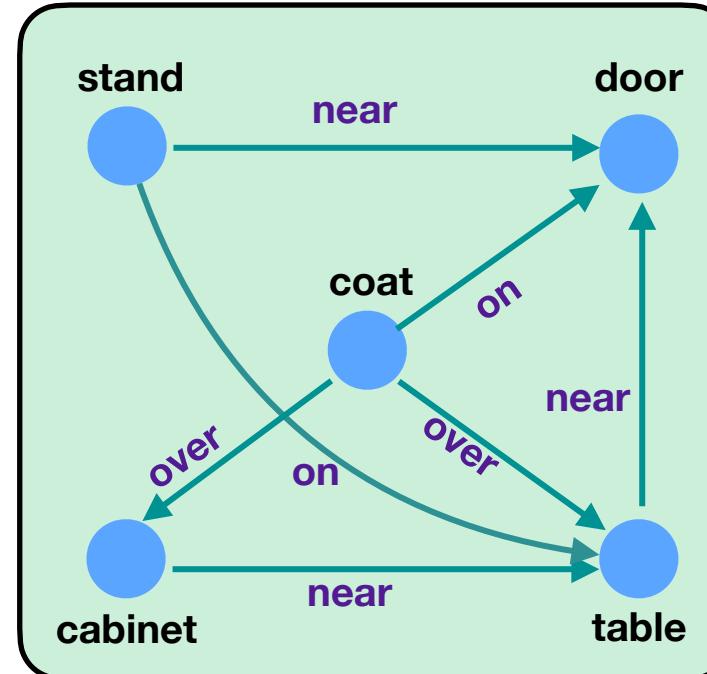
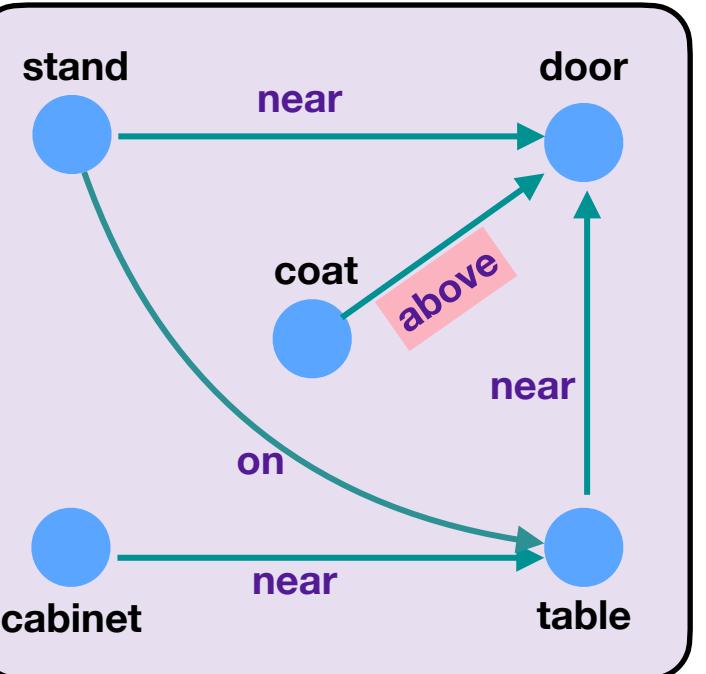
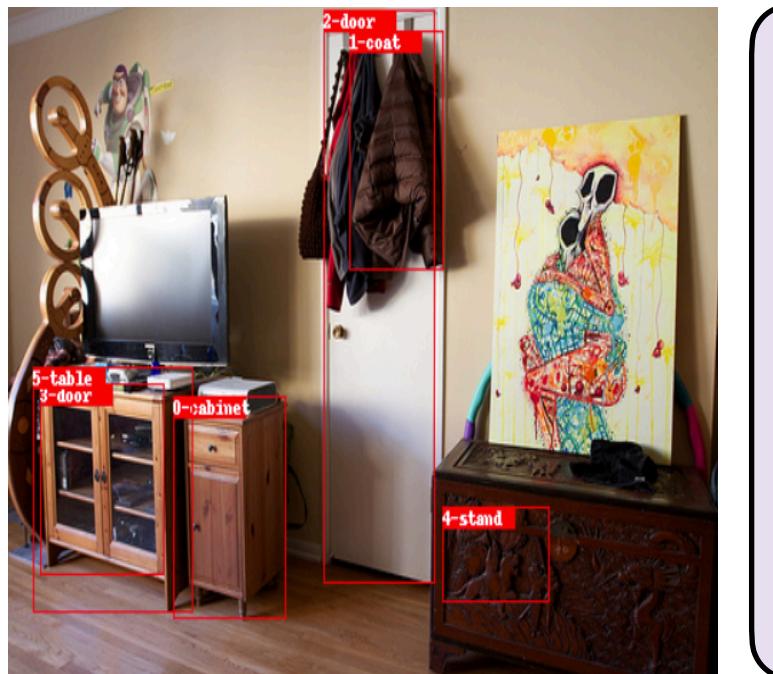
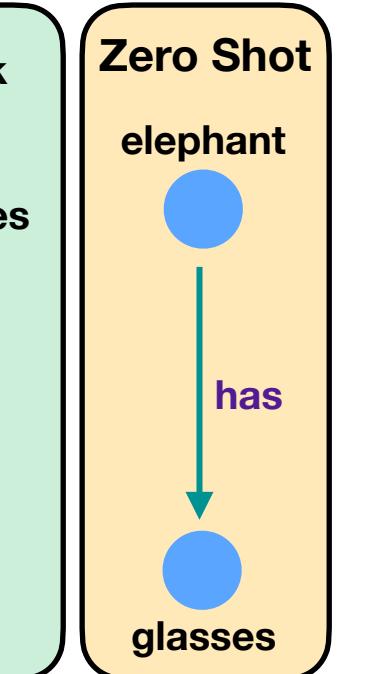
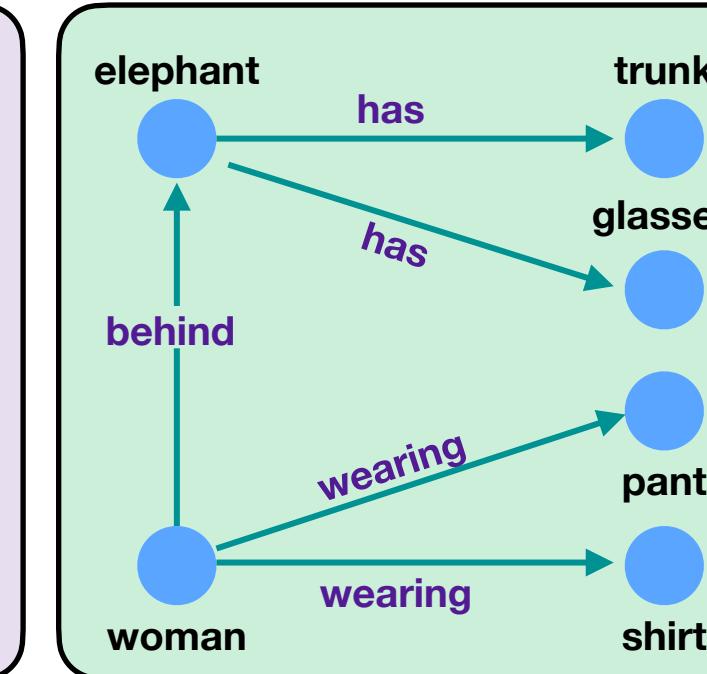
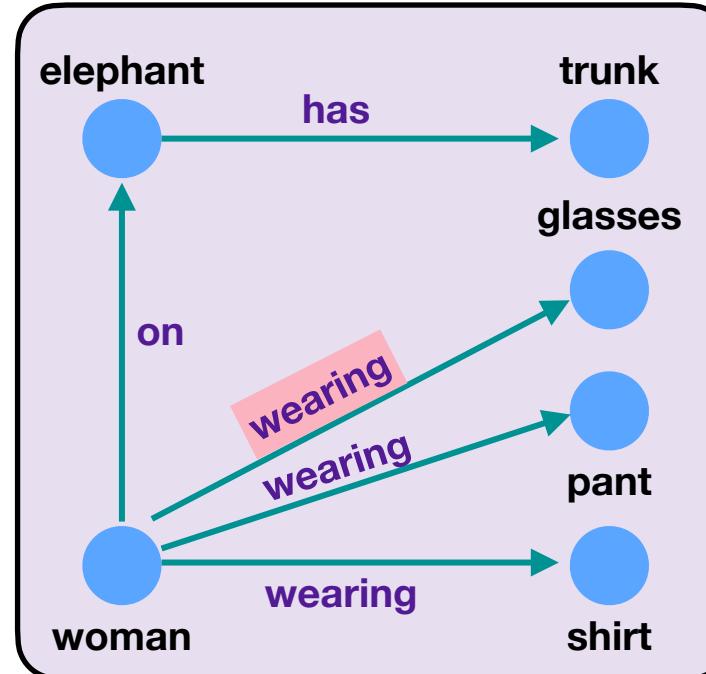
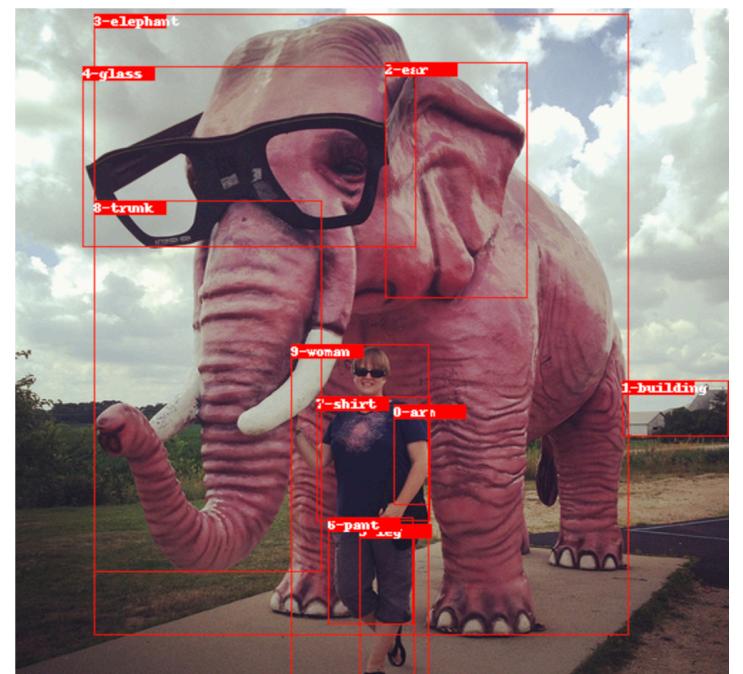
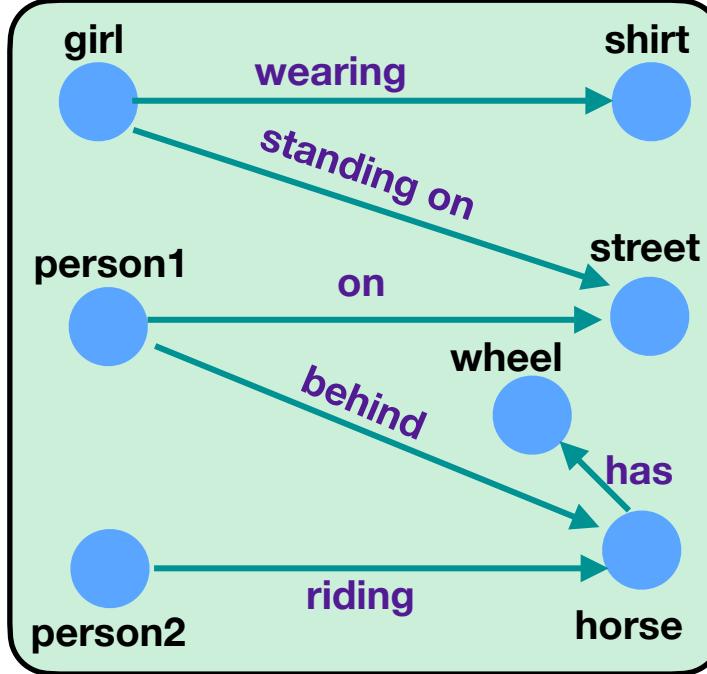
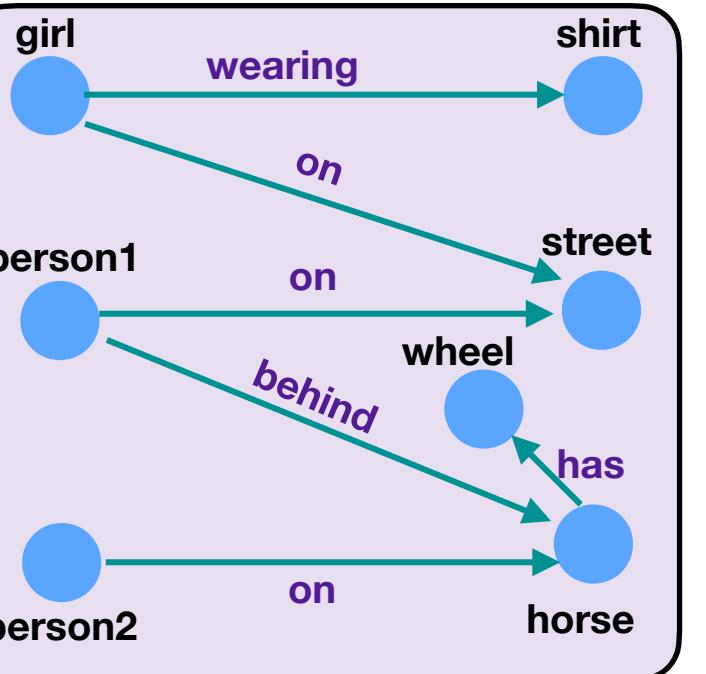
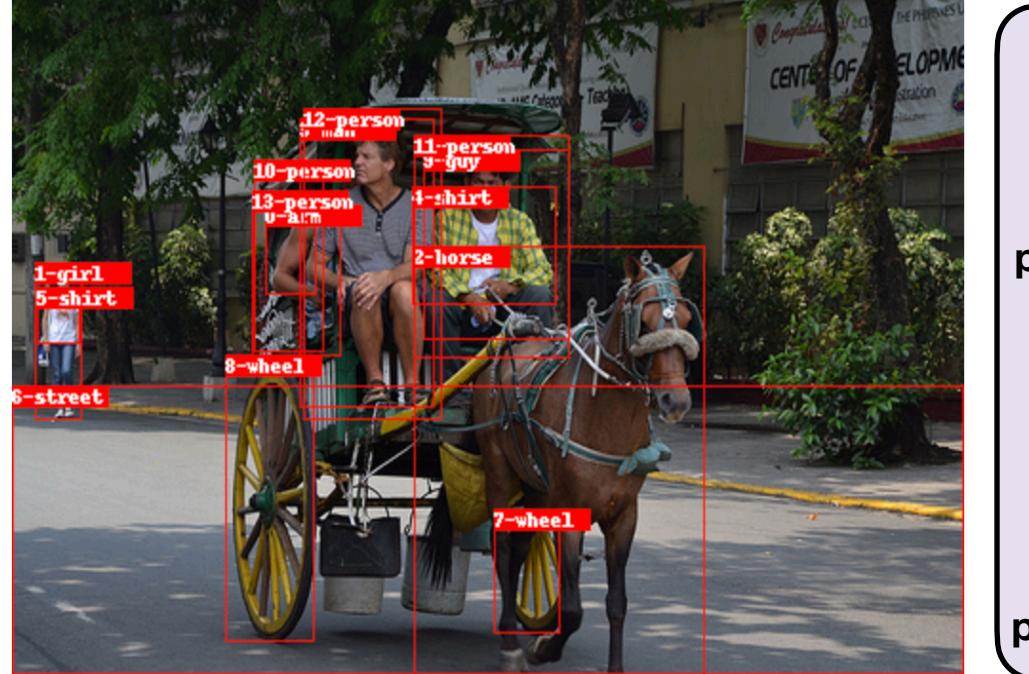
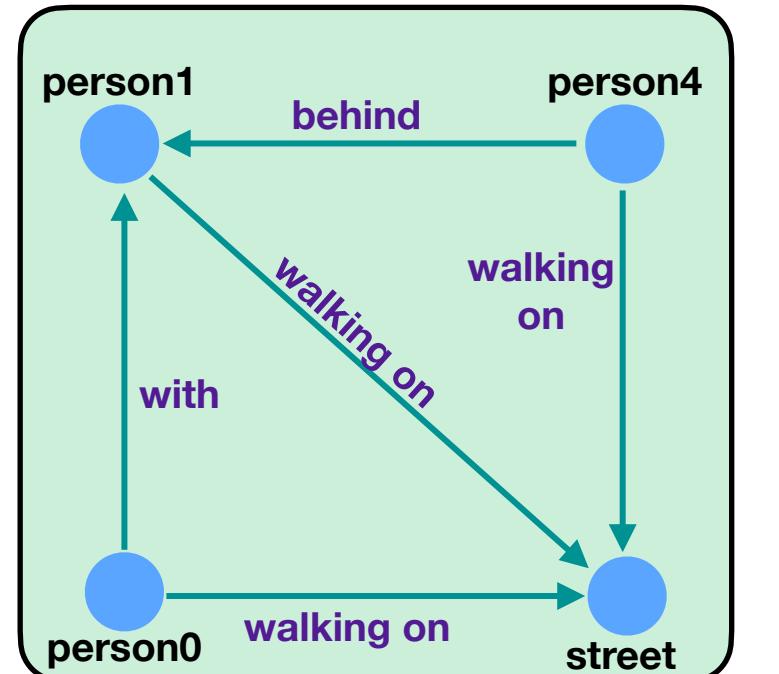
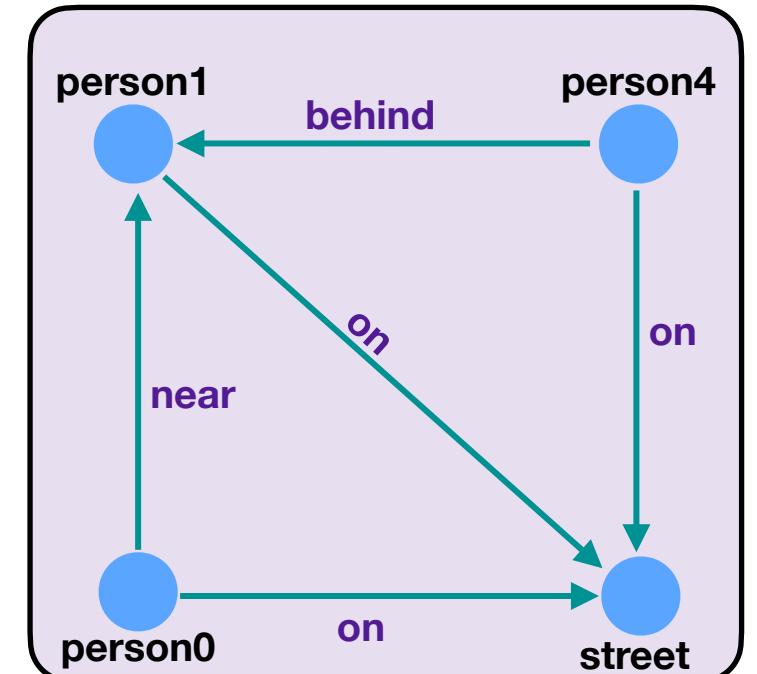
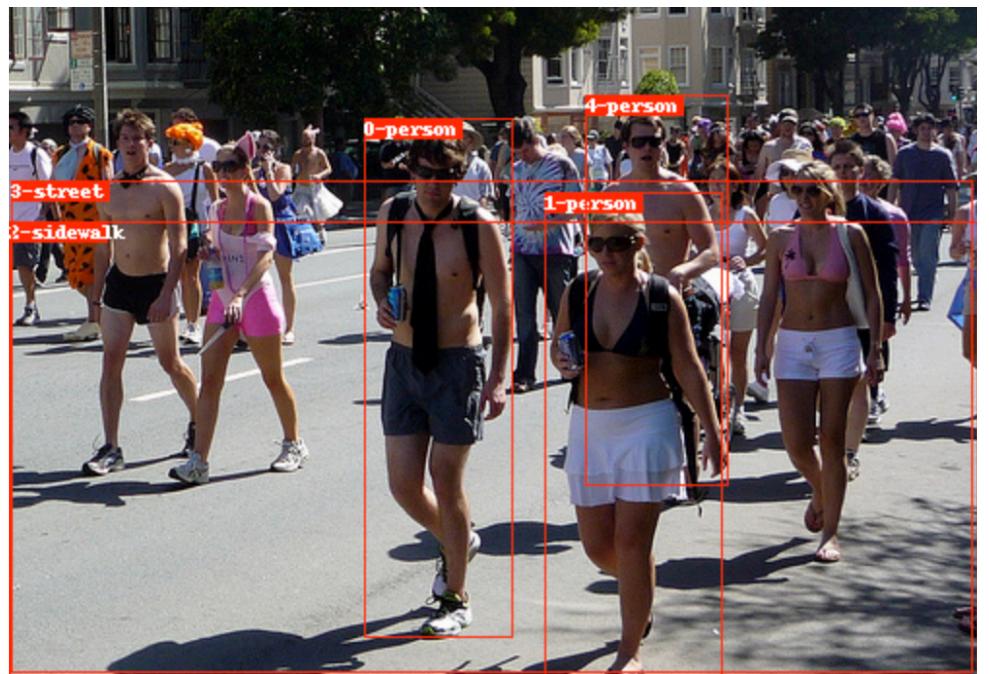
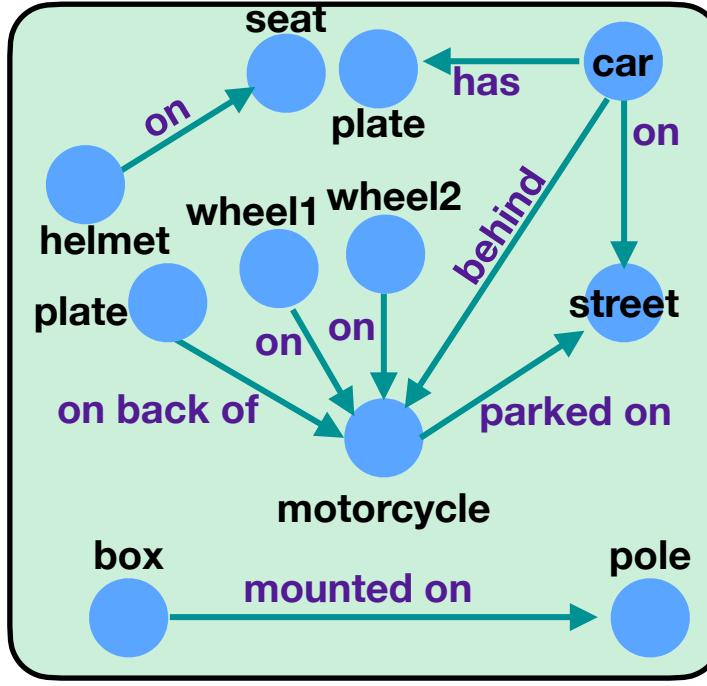
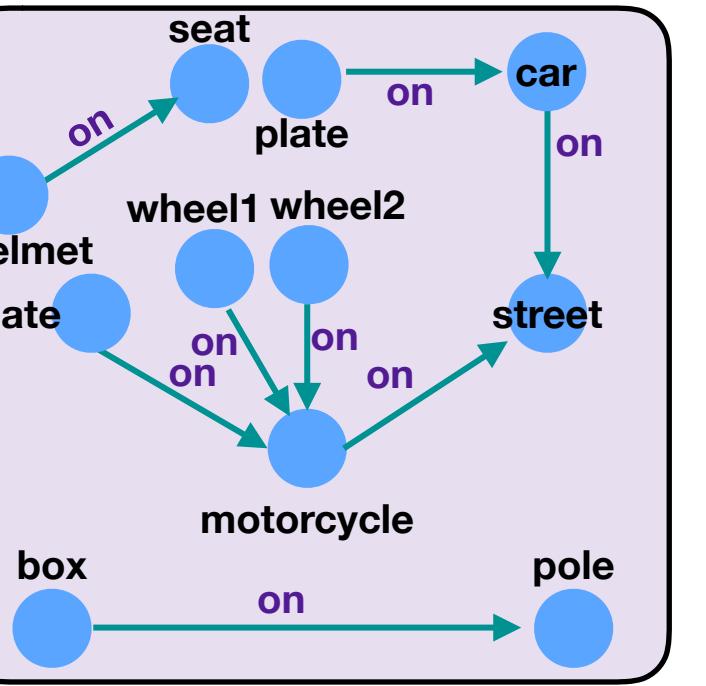
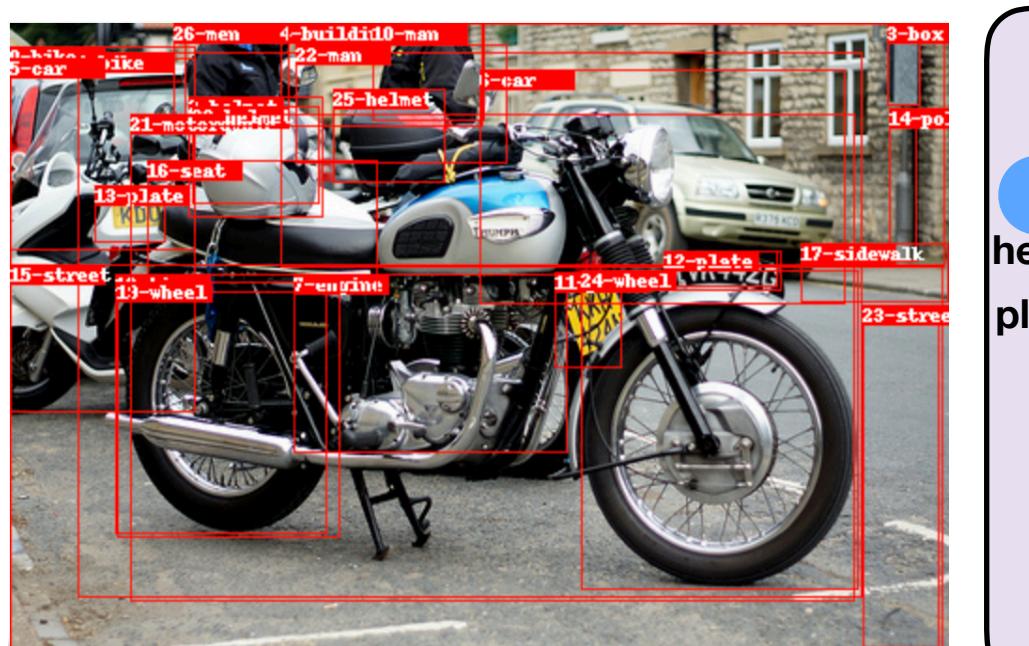
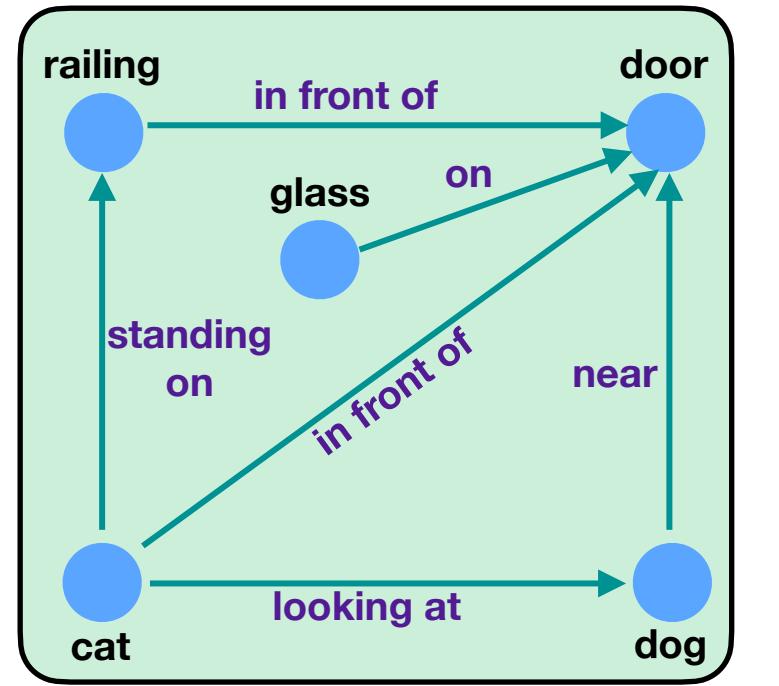
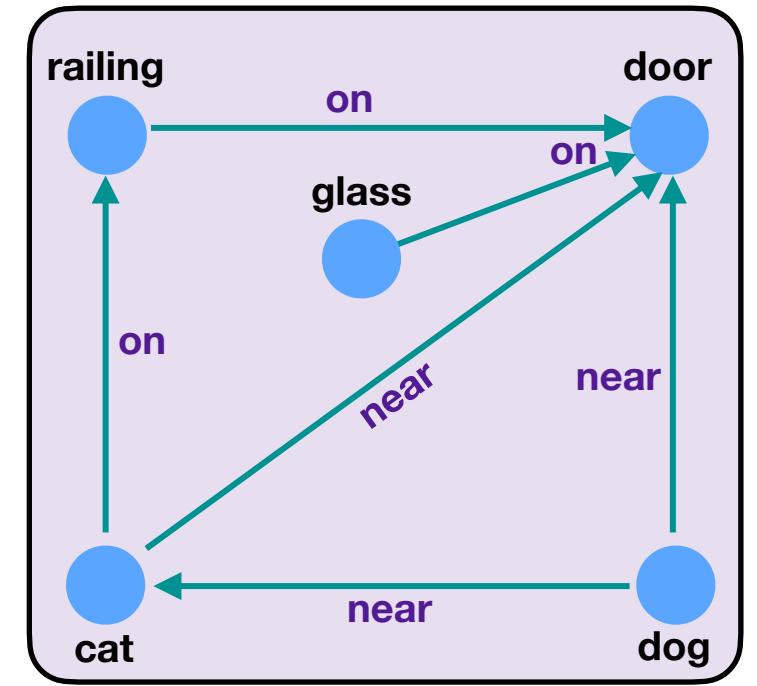
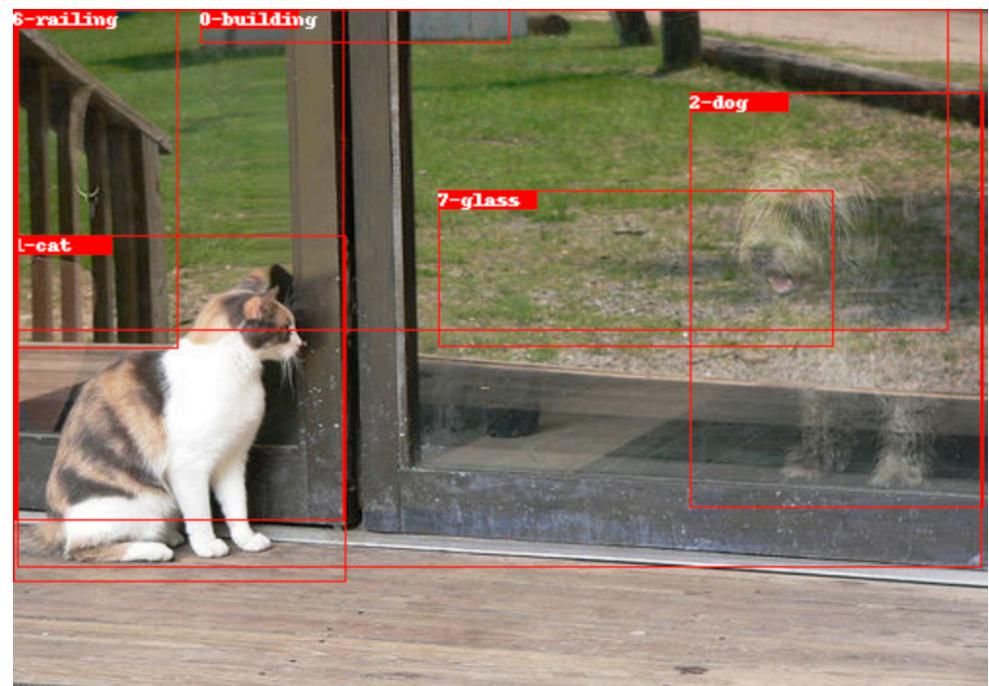




Visualizations



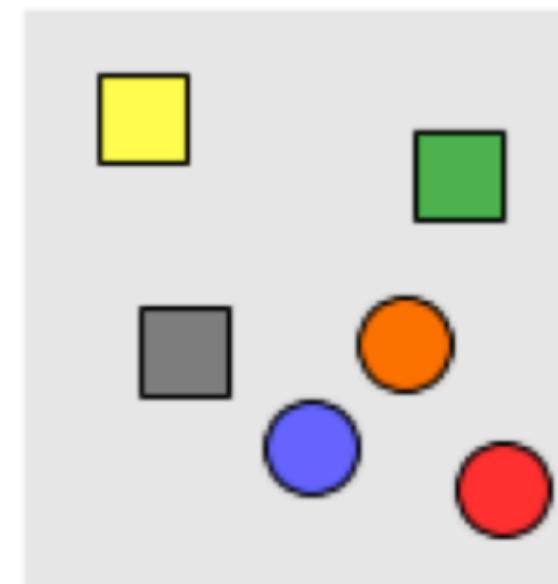




Conclusions

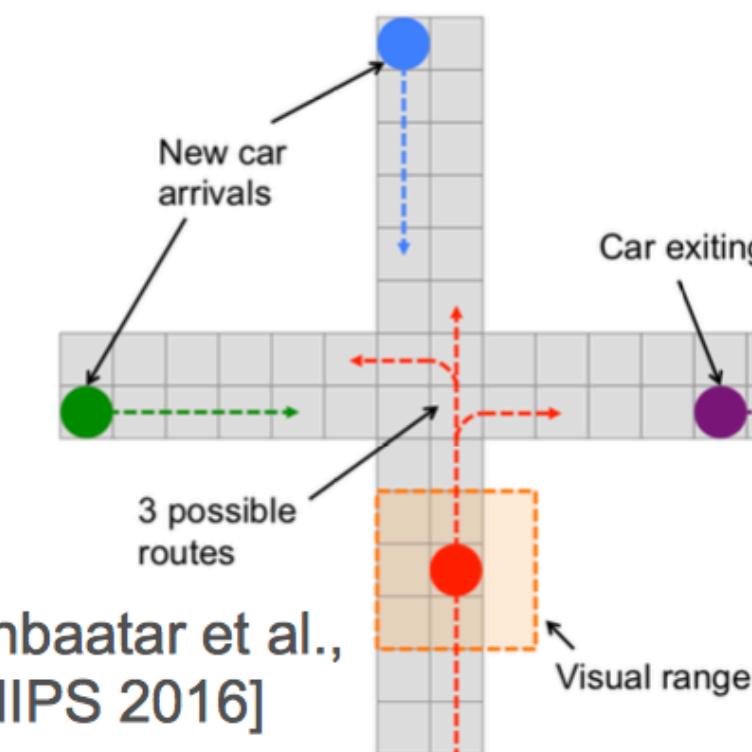
- Deep learning on graphs works and is very effective!
- Exciting area: lots of new applications and extensions (hard to keep up)

Relational reasoning



[Santoro et al., NIPS 2017]

Multi-Agent RL



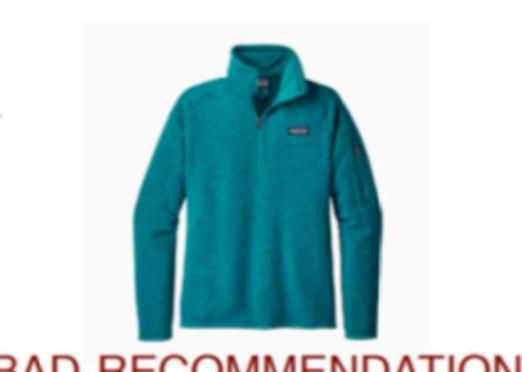
[Sukhbaatar et al.,
NIPS 2016]

GCN for recommendation on 16 billion edge graph!



Source pin

[Leskovec lab, Stanford]



Open problems:

- Theory
- Scalable, stable generative models
- Learning on large, evolving data
- Multi-modal and cross-model learning (e.g., sequence2graph)

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*.
- [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*.