tinyML® Research Symposium

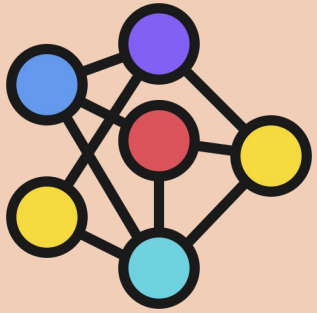
Enabling Ultra-low Power Machine Learning at the Edge

April 22, 2024

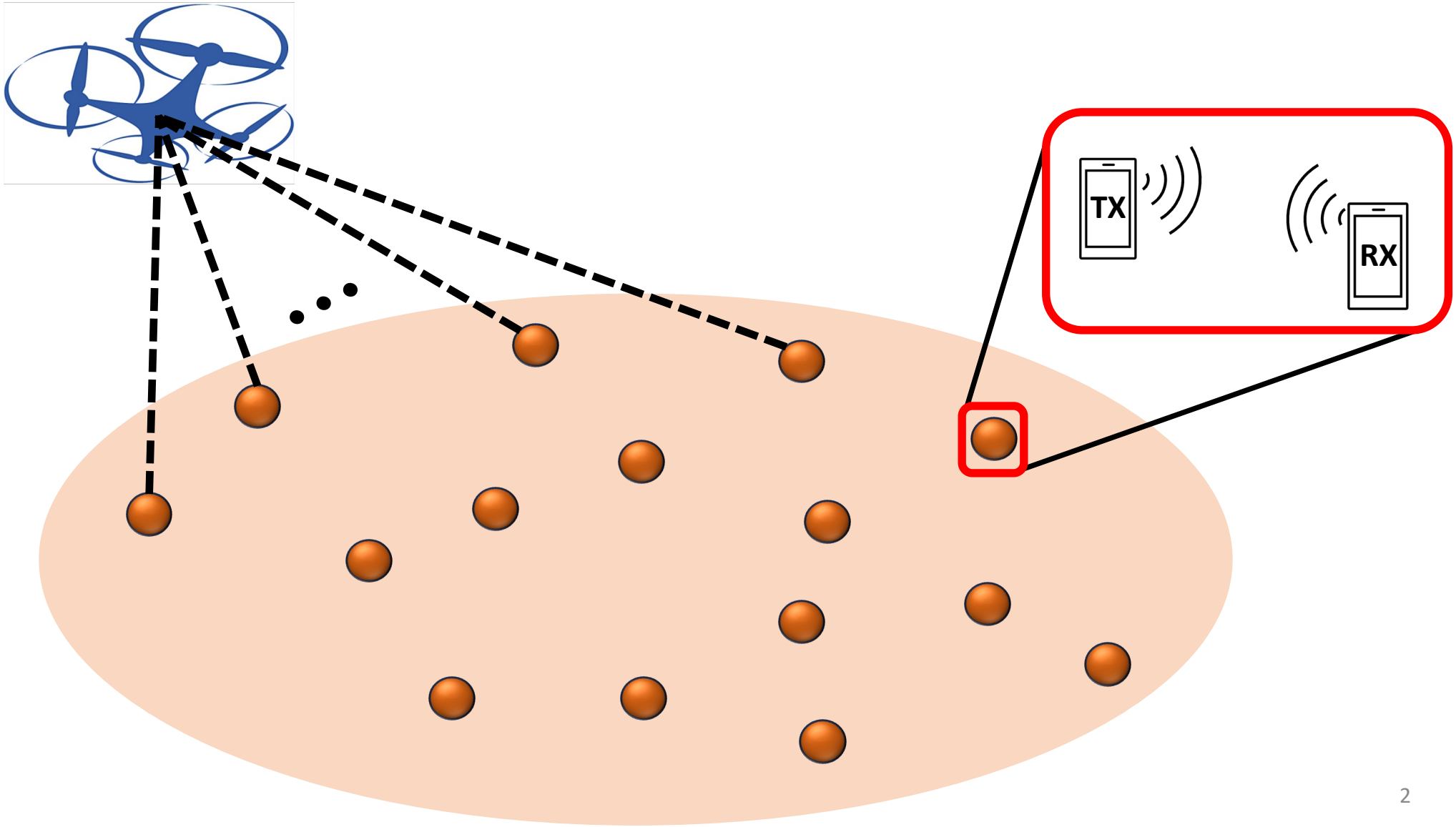
www.tinyML.org

Tiny Graph Neural Networks for Radio Resource Management

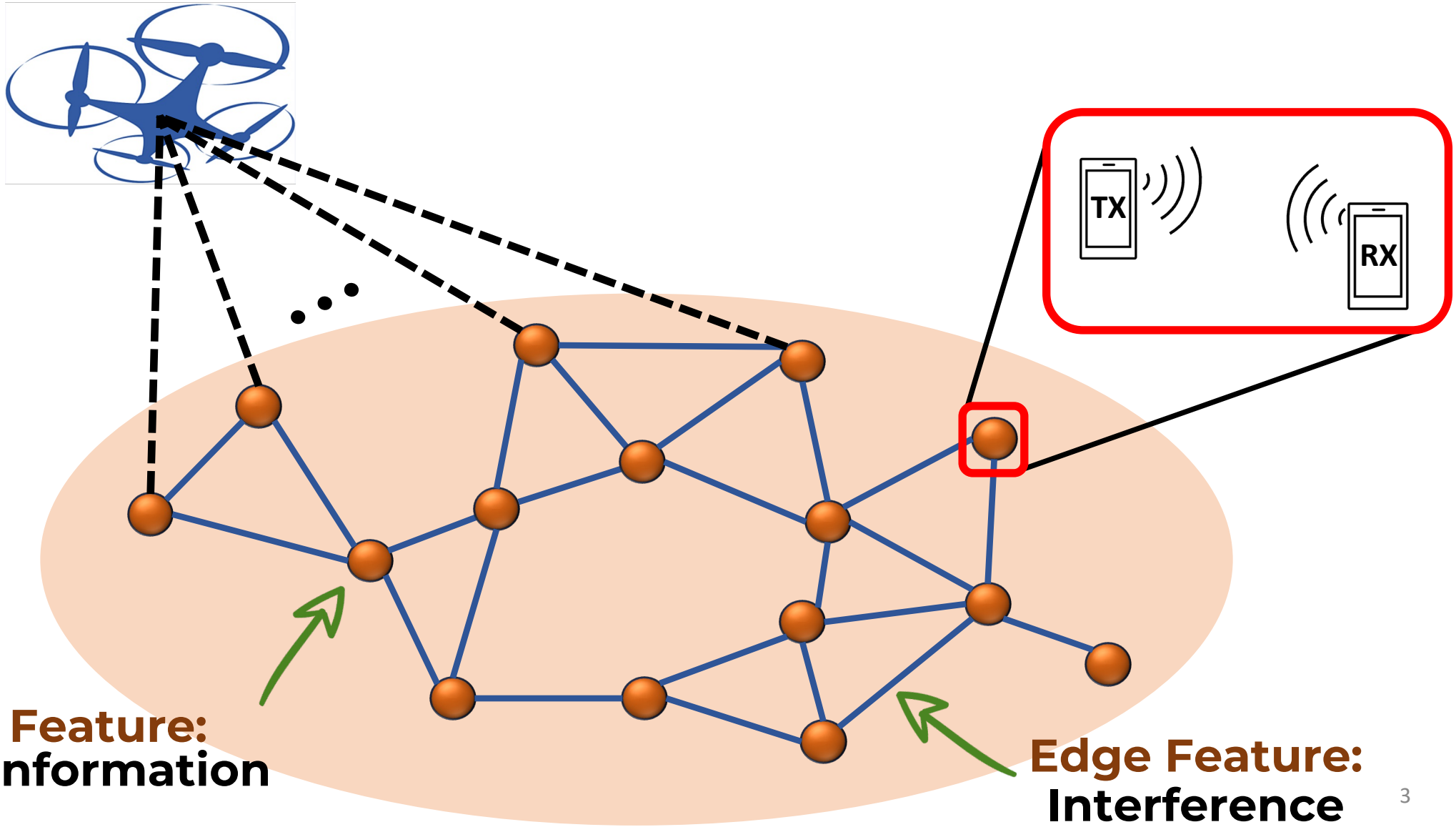
Ahmad Ghasemi and Hossein Pishro-Nik



Radio Resource Management (RRM)



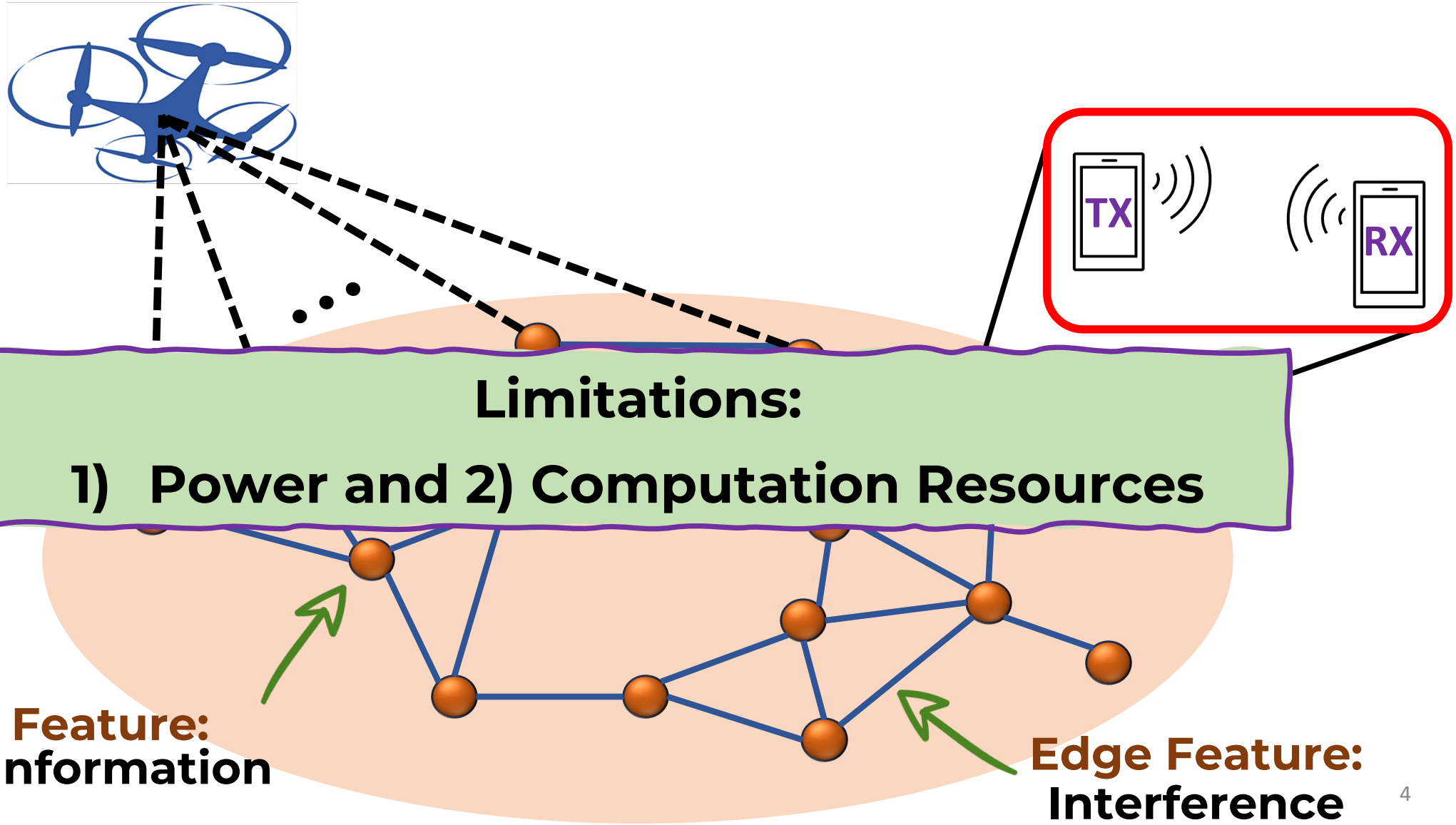
GNN based RRM



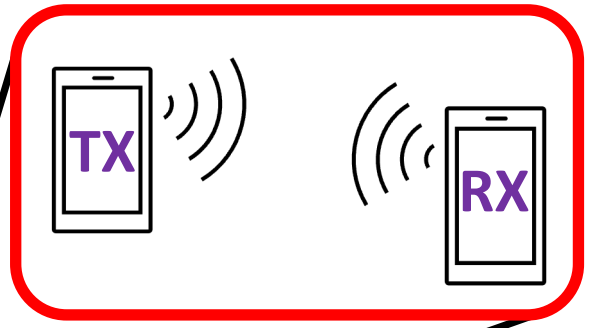
Vertex Feature:
Channel Information

Edge Feature:
Interference

GNN based RRM



GNN based RRM



Limitations:

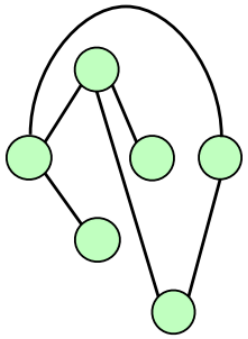
1) Power and 2) Computation Resources

➔ Smaller Models

Vertex Feature:
Channel Information

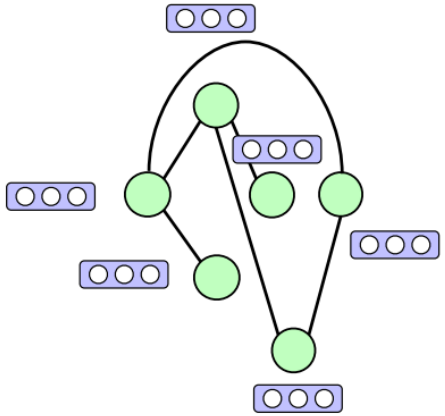
Edge Feature:
Interference

Message Passing GNN



Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

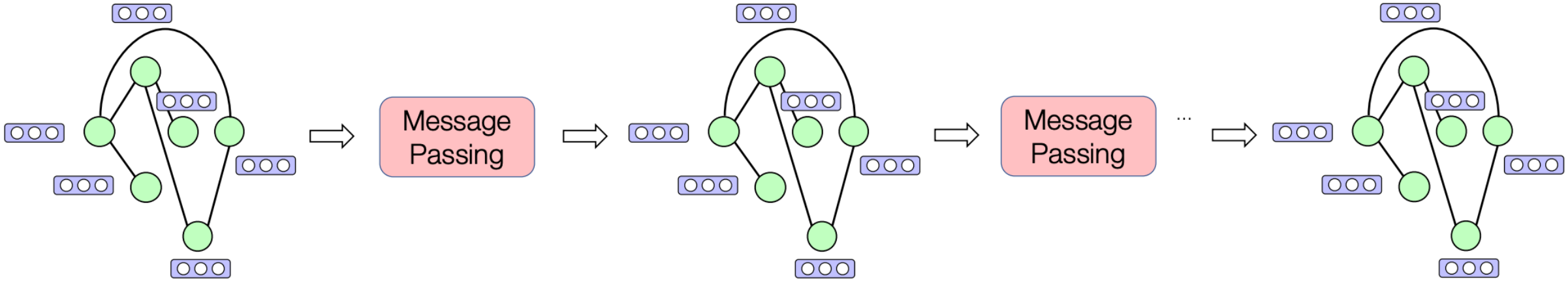
Message Passing GNN



Input Encoding

Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

Message Passing GNN

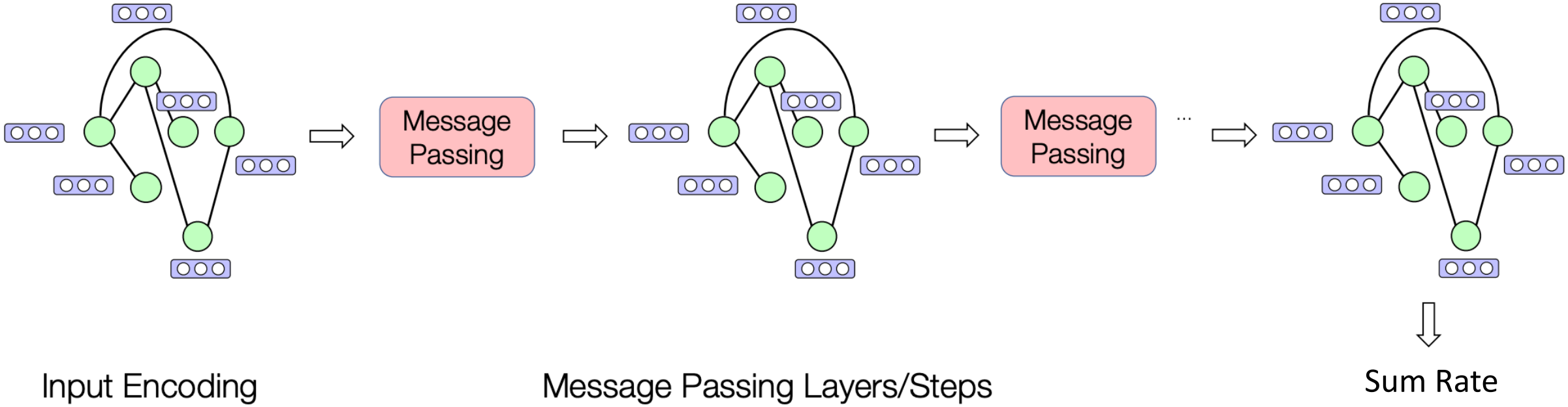


Input Encoding

Message Passing Layers/Steps

Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

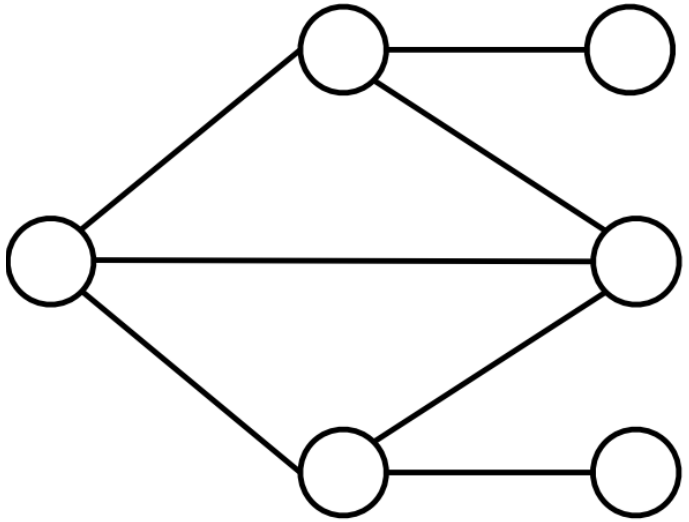
Message Passing GNN



Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

Massage Passing in MPGNN

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

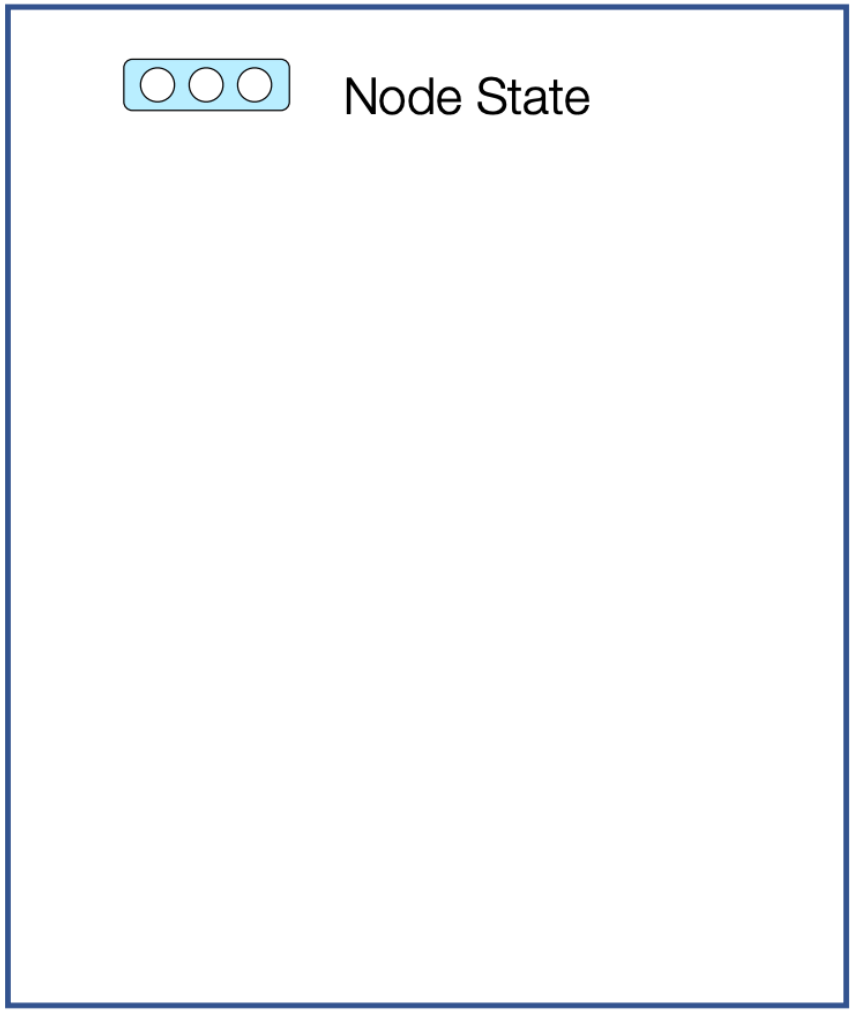


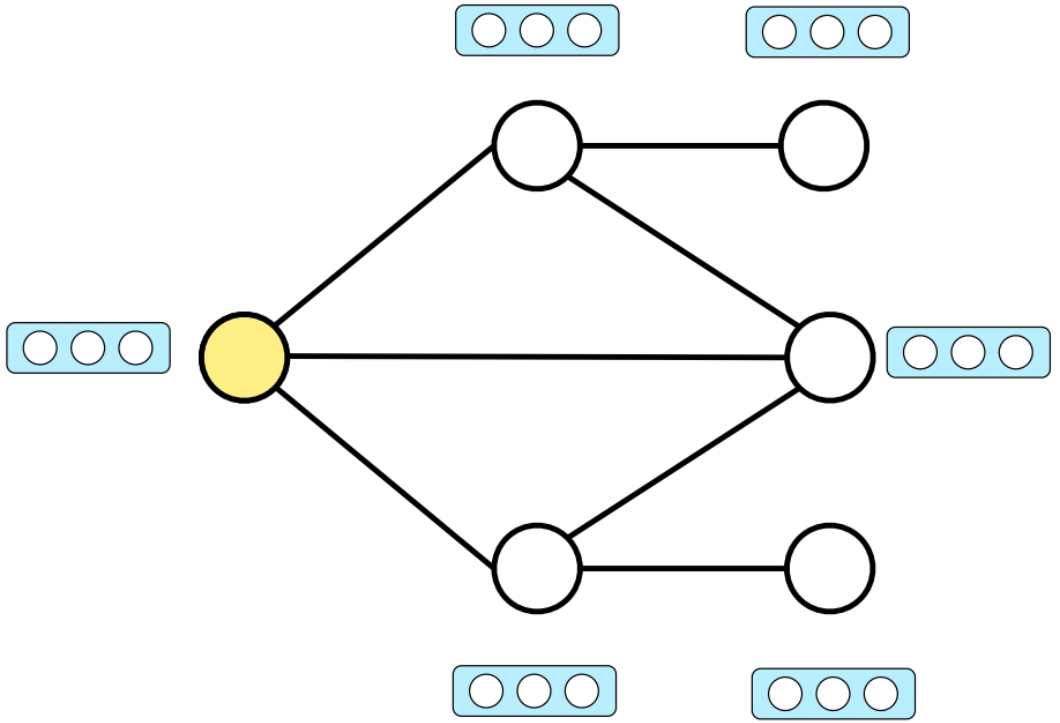
Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

Massage Passing in MPGNN

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

$\mathbf{h}_i^t$



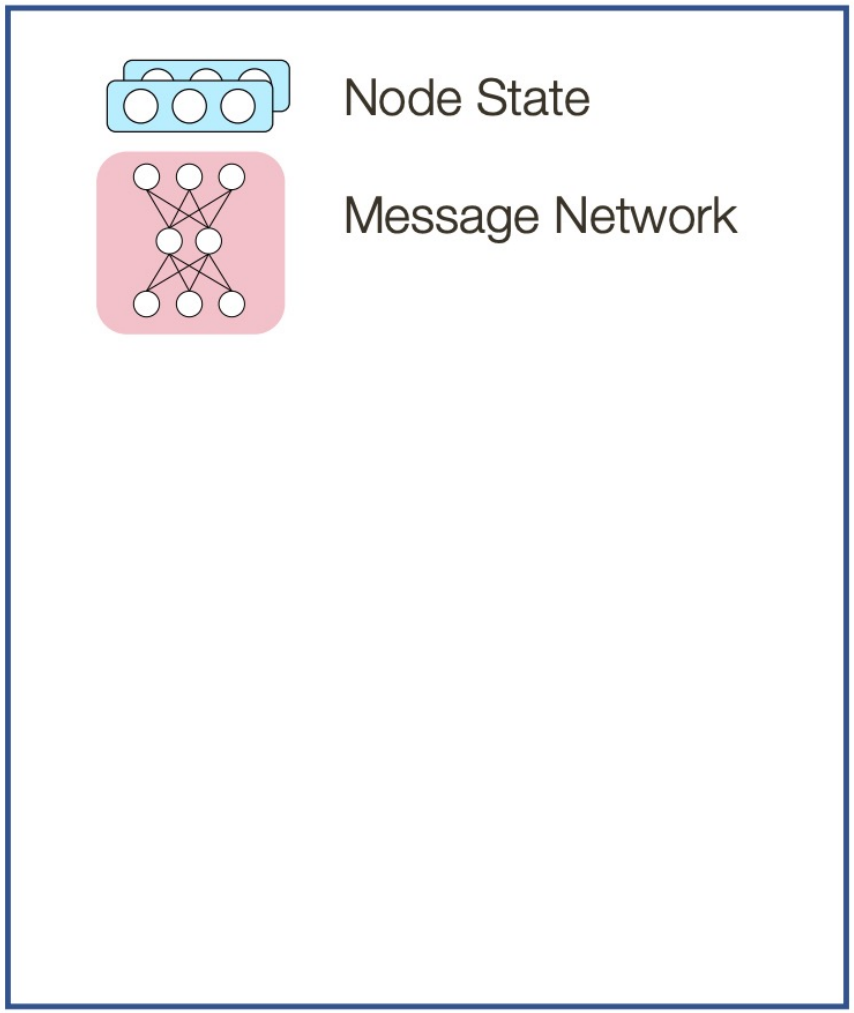


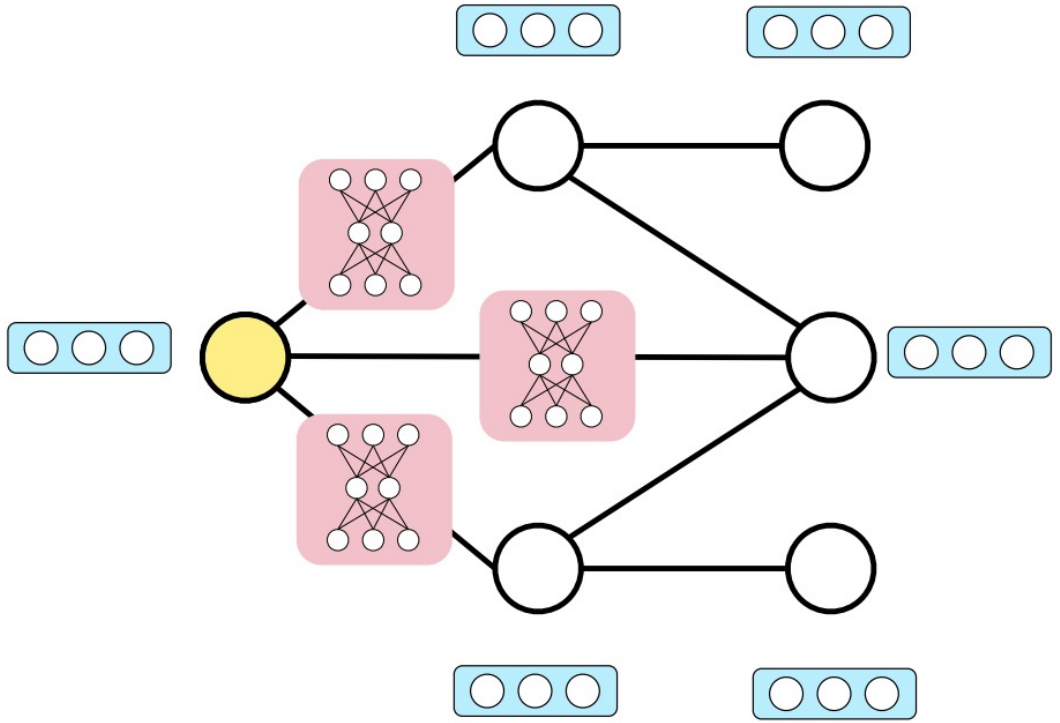
Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

Massage Passing in MPGNN

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

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





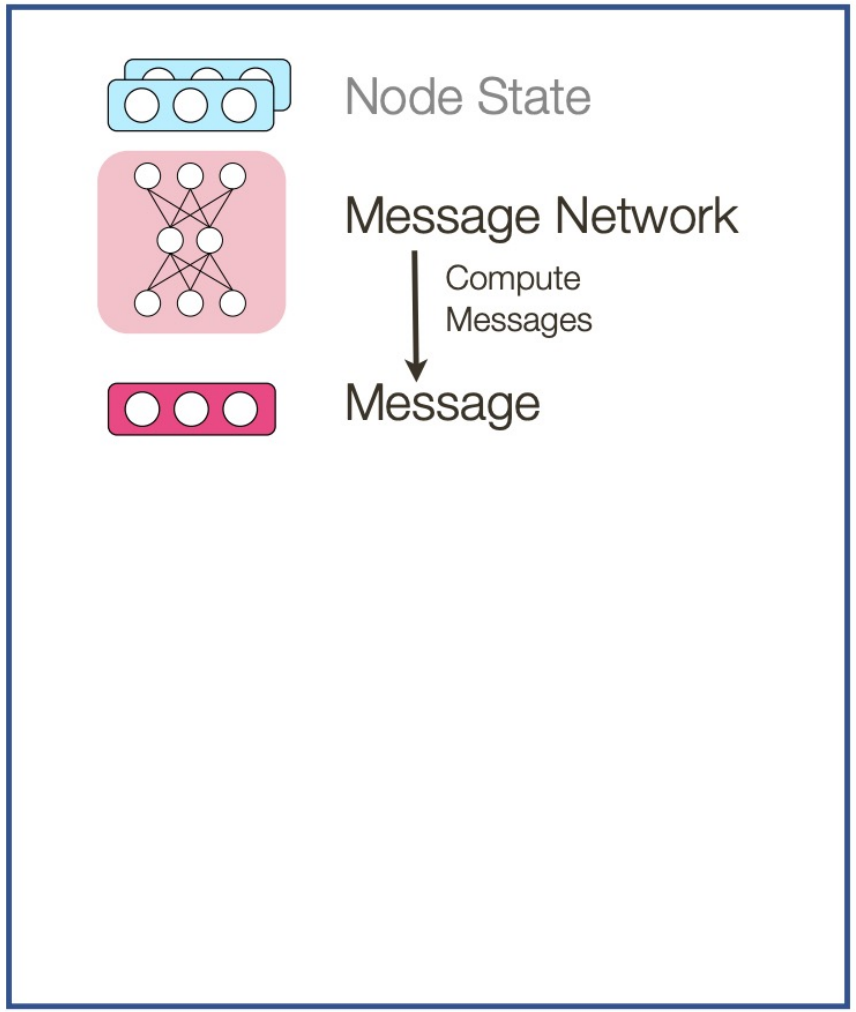
Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

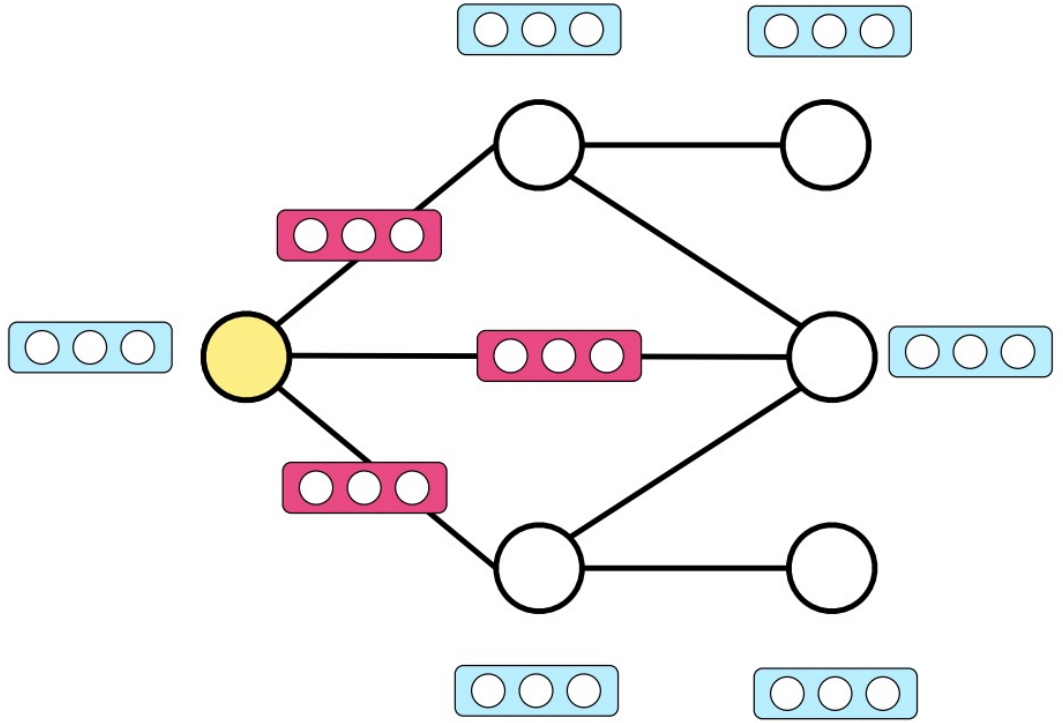
Massage Passing in MPGNN

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

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

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





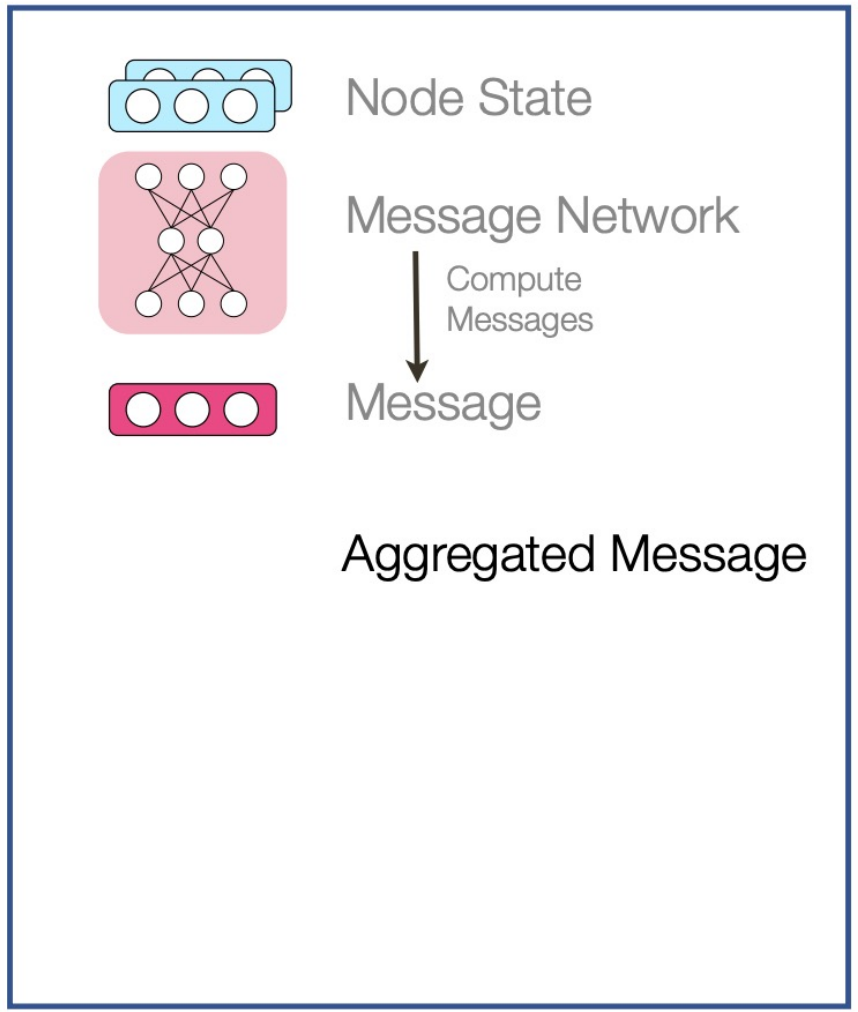
Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

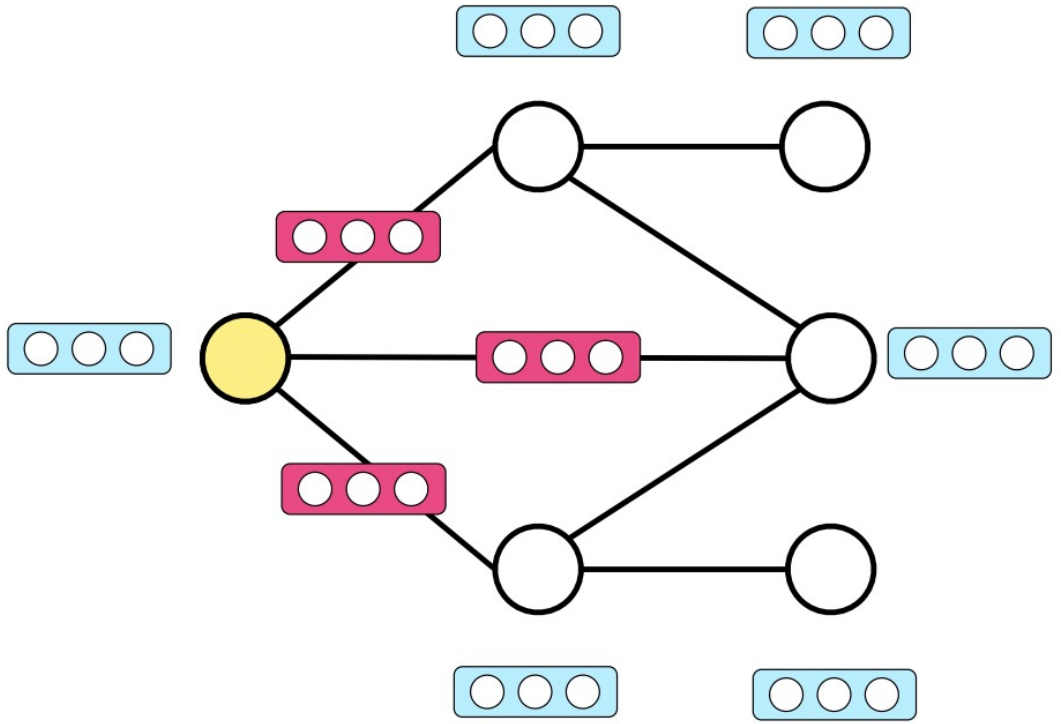
Massage Passing in MPGNN

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

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

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





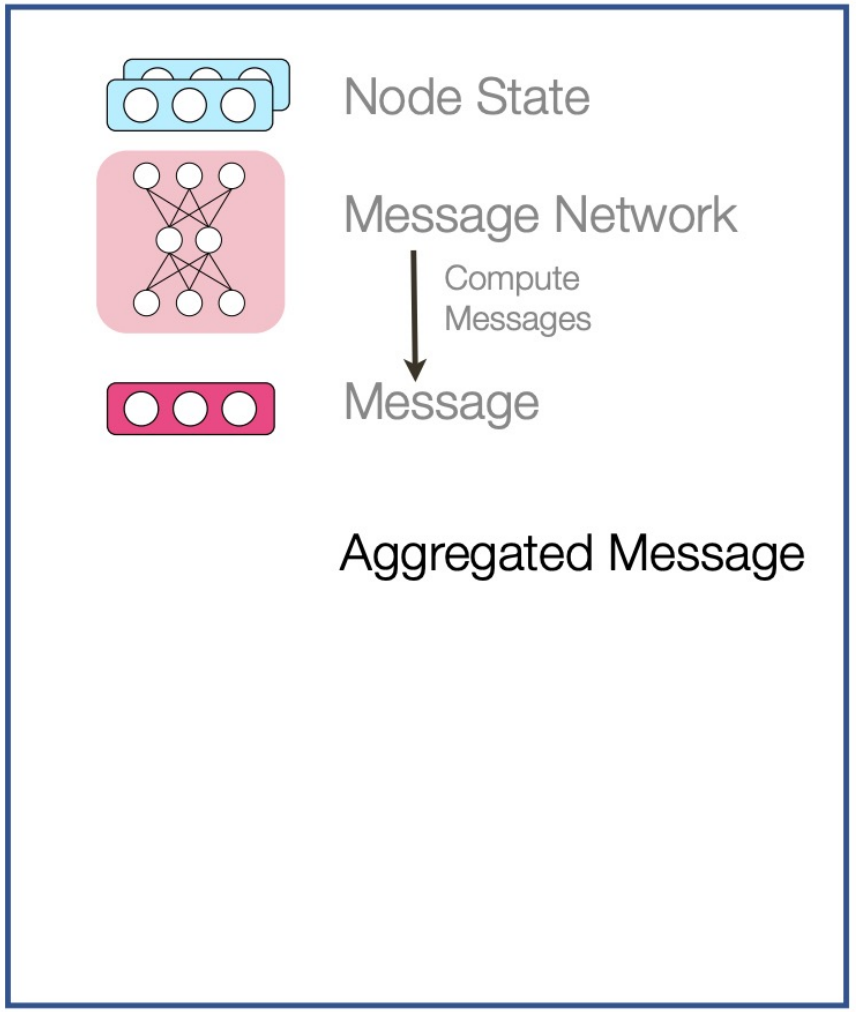
Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

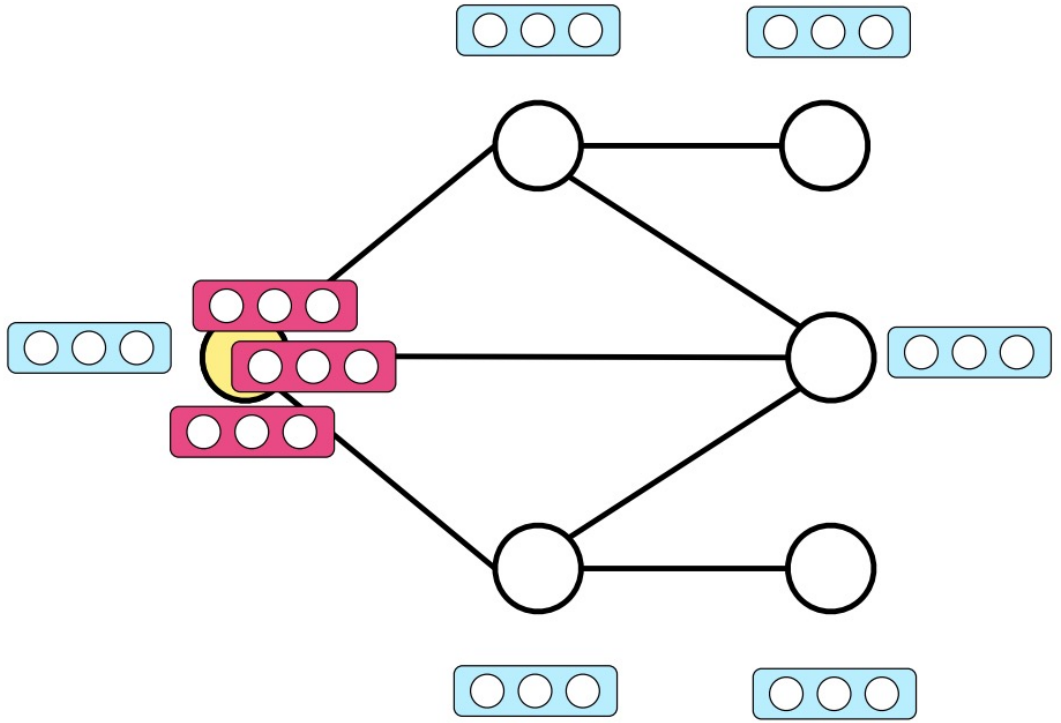
Massage Passing in MPGNN

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

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

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





Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

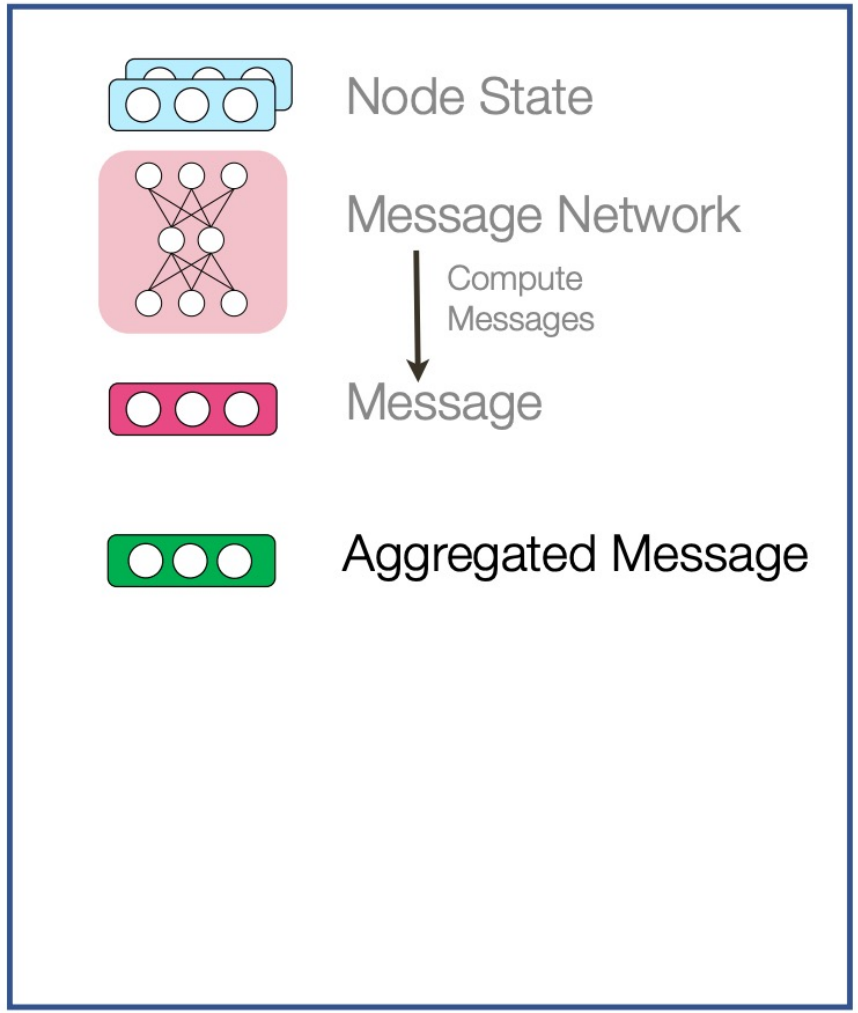
Massage Passing in MPGNN

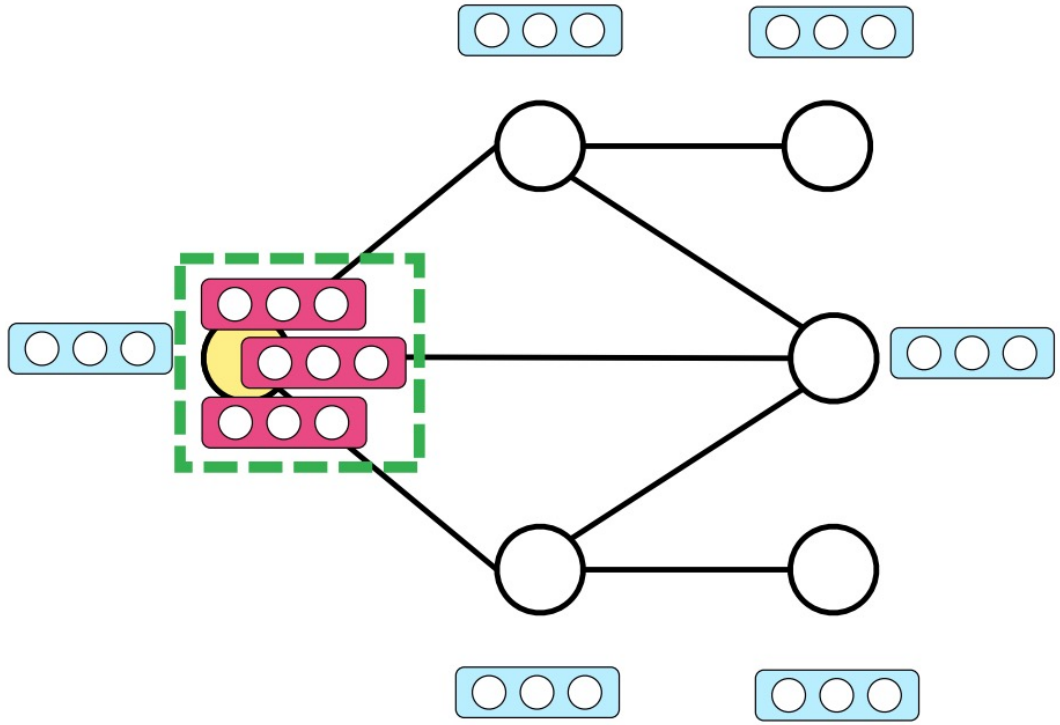
(t+1)-th message passing step/layer

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

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

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





Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

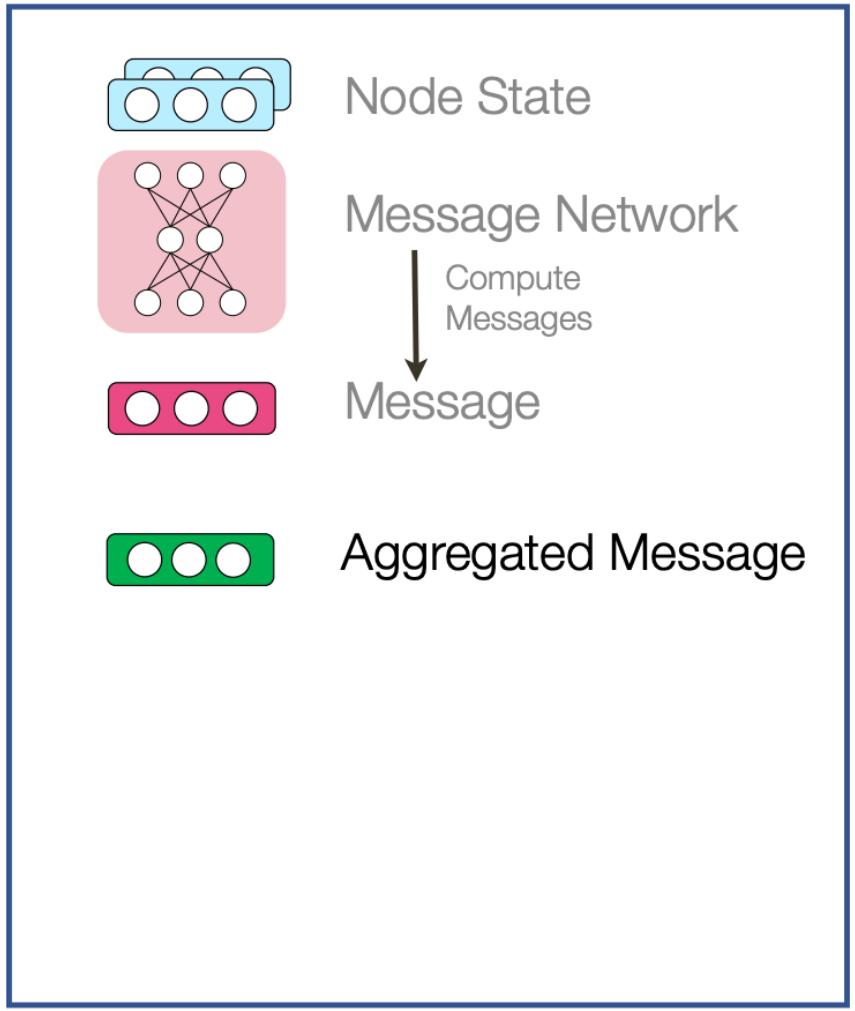
Massage Passing in MPGNN

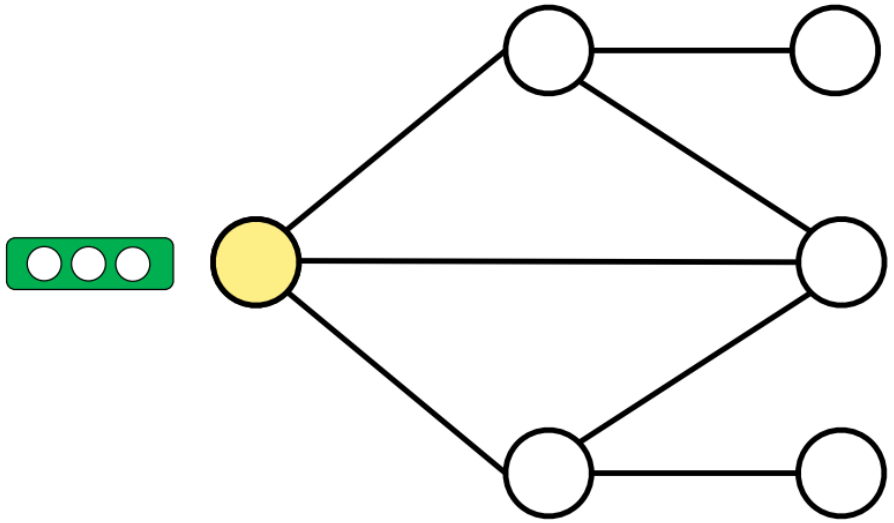
(t+1)-th message passing step/layer

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

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

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





Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

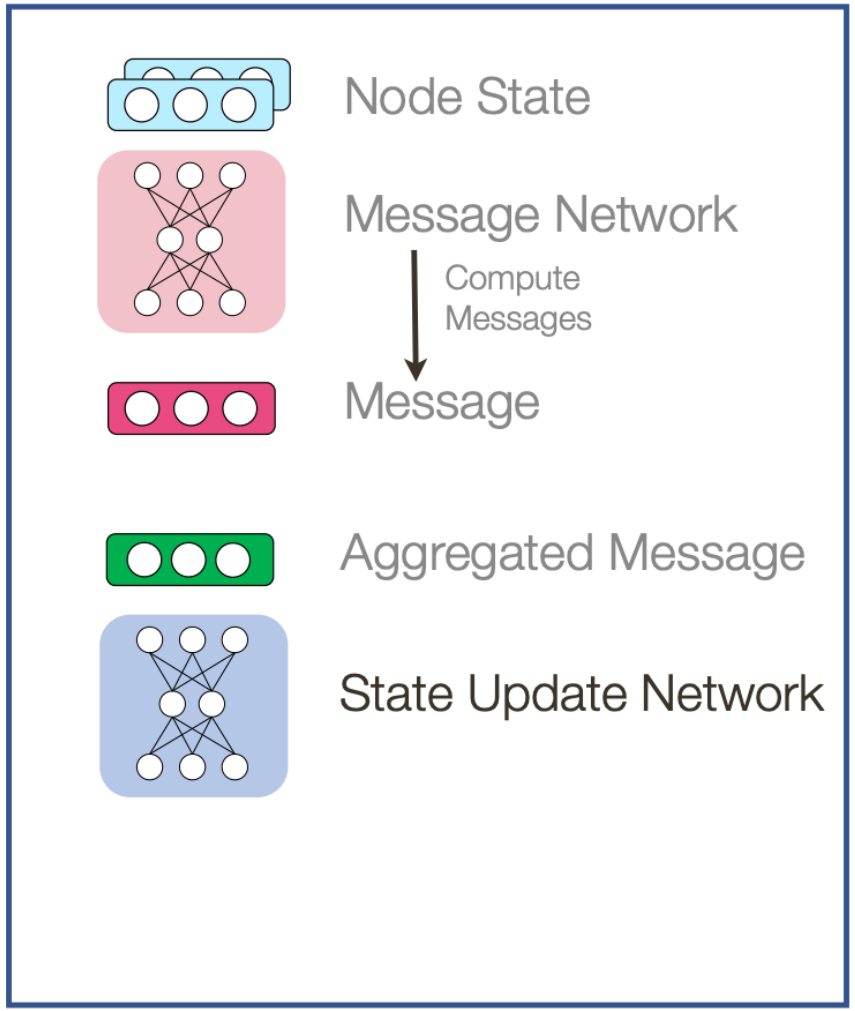
Massage Passing in MPGNN

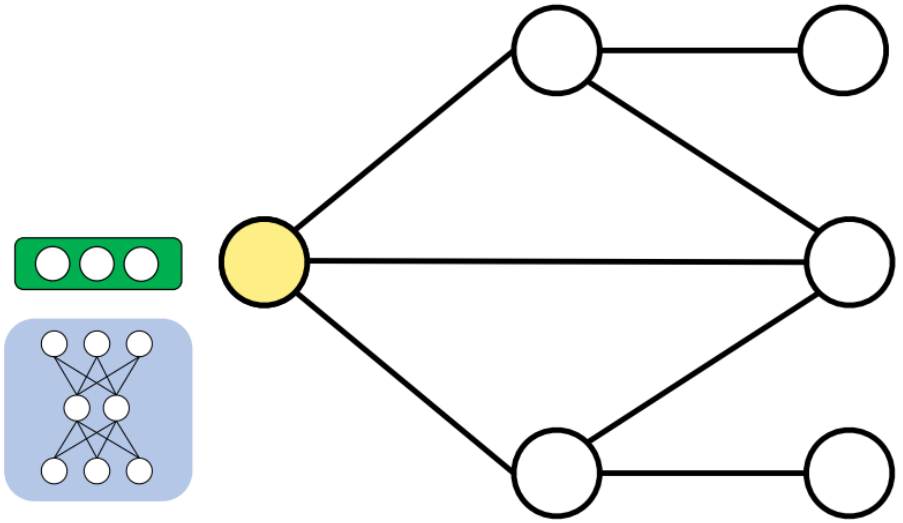
(t+1)-th message passing step/layer

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

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

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





Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

Massage Passing in MPGNN

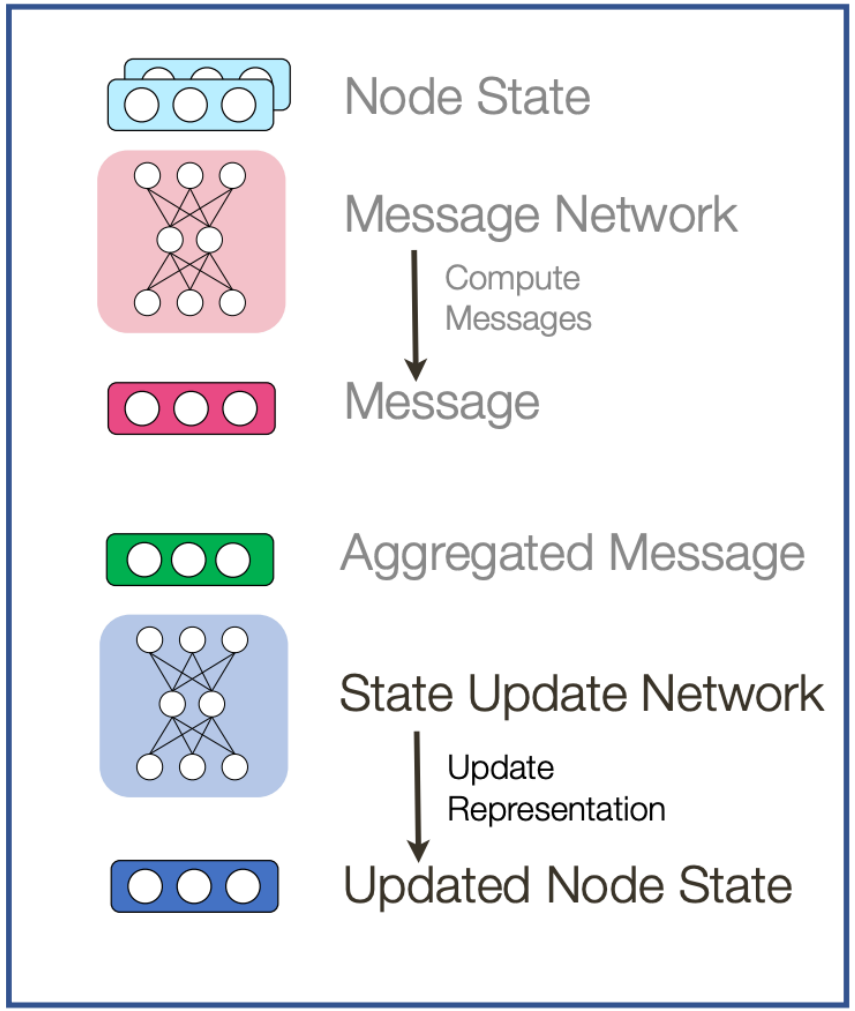
(t+1)-th message passing step/layer

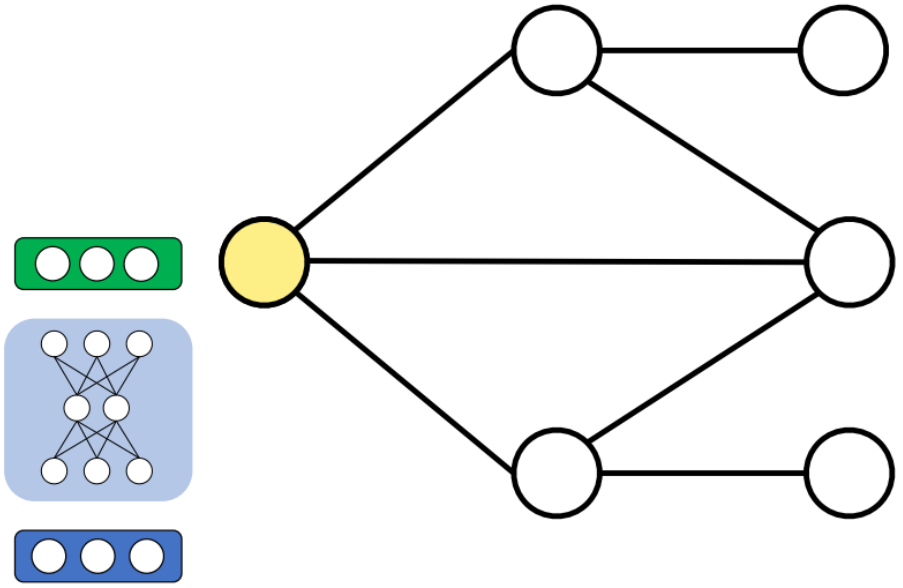
$\mathbf{h}_i^t \quad \mathbf{h}_j^t$

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

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

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





Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

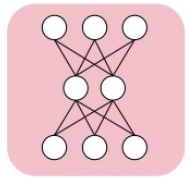
Massage Passing in MPGNN

1. Compute Messages

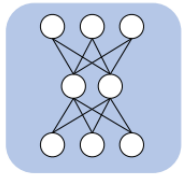
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{e}_{ji})$$

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

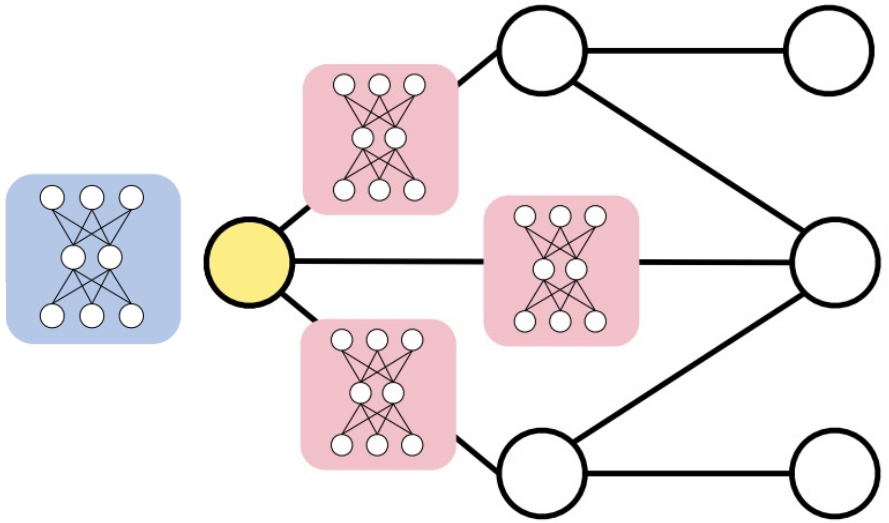
Edge Feature



Message Network



State Update Network



Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

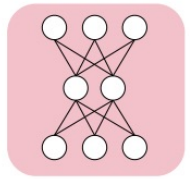
Massage Passing in MPGNN

1. Compute Messages

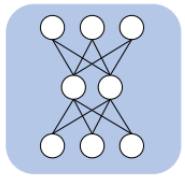
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{e}_{ji})$$

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

Edge Feature



Message Network

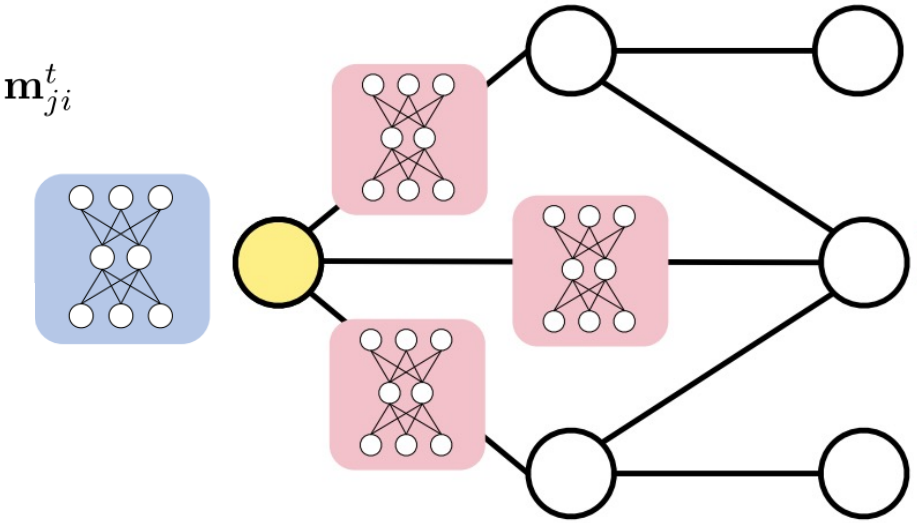


State Update Network

2. Aggregate Messages

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

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



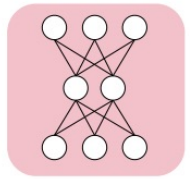
Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

Massage Passing in MPGNN

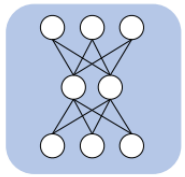
1. Compute Messages

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

Edge Feature



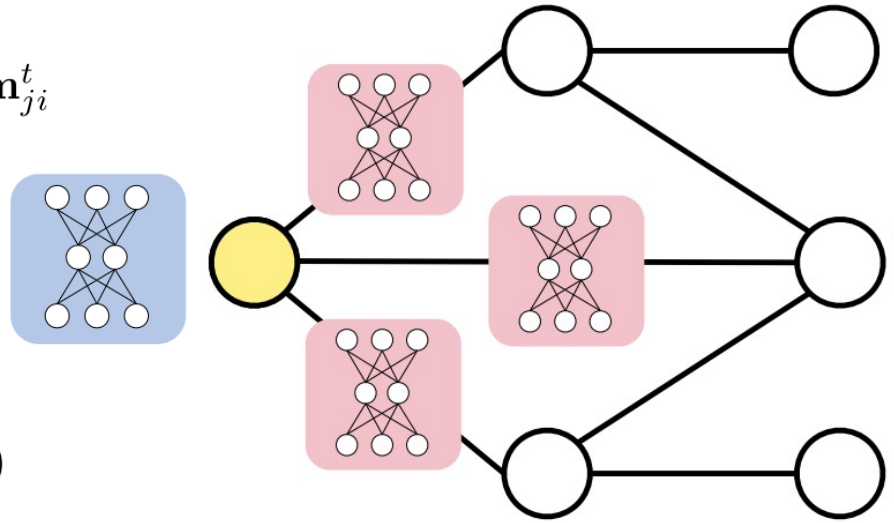
Message Network



State Update Network

2. Aggregate Messages

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



3. Update Node Representations

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

Slide Credit: https://www.cs.ubc.ca/~lsigal/532S_2022W1/Lecture19.pdf

Massage Passing in MPGNN

1. Compute Messages

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

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

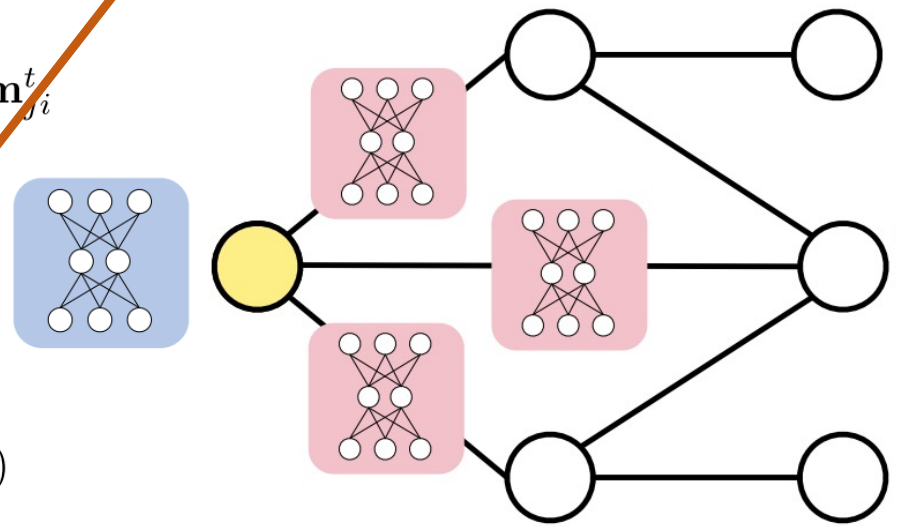
Edge Feature

Message Network

State Update Network

2. Aggregate Messages

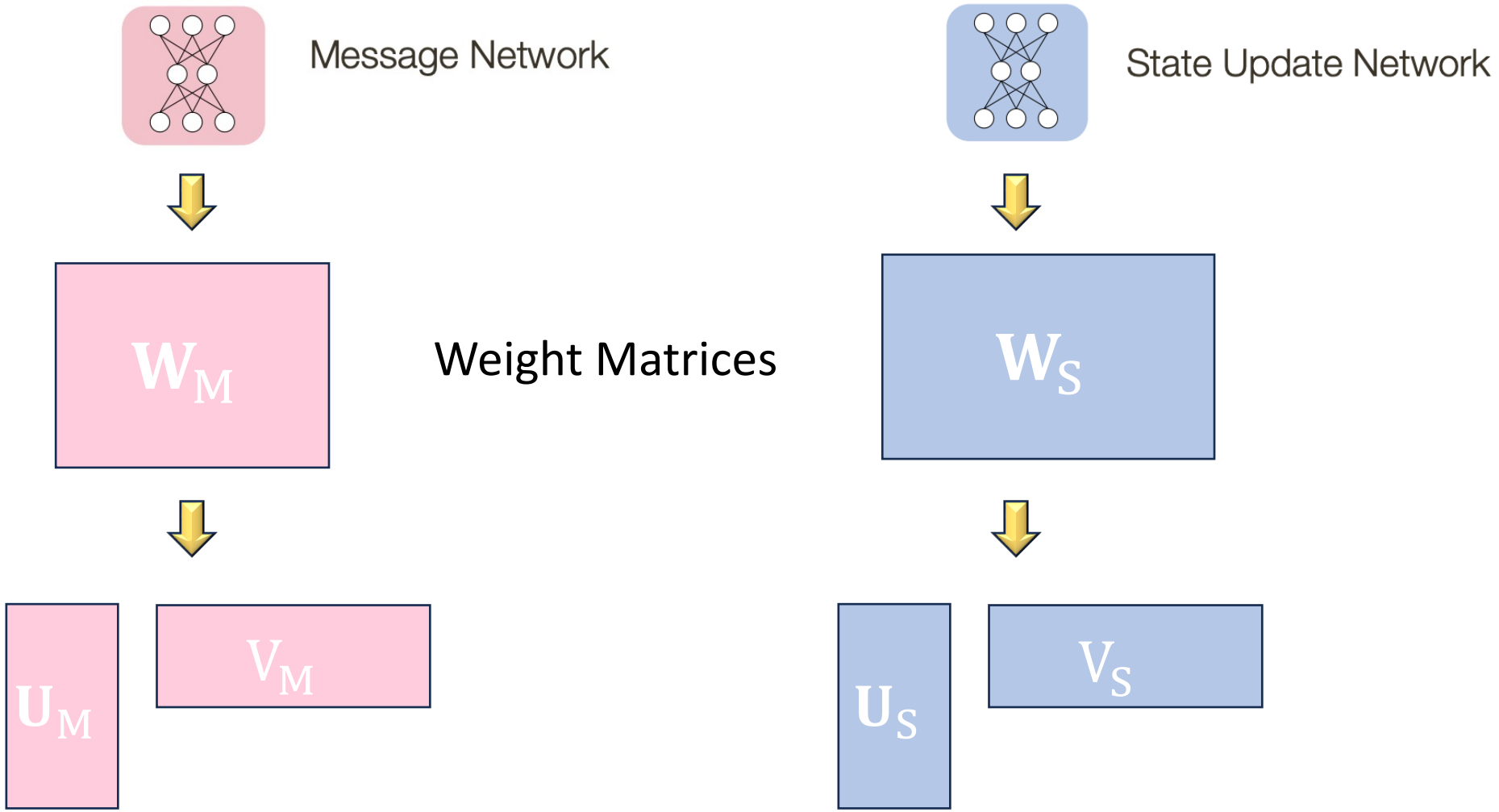
$$\bar{\mathbf{m}}_i^t = f_{\text{agg}}\left(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}\right) \qquad f_{\text{agg}}\left(\{\mathbf{m}_{ji}^t | j \in \mathcal{N}_i\}\right) = \max_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t$$



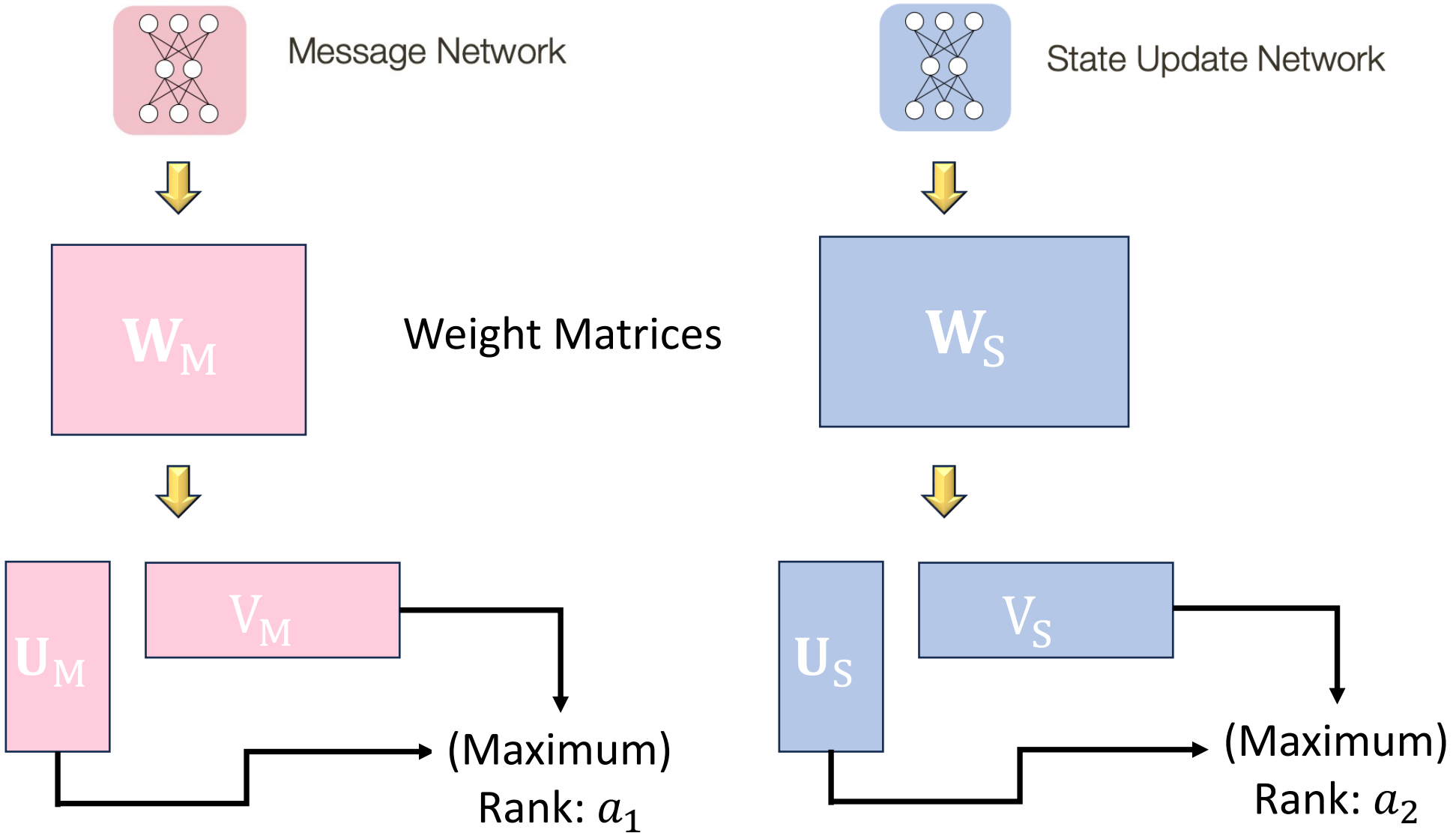
3. Update Node Representations

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

Proposed Approach: Low-Rank Linear Layer



Proposed Approach: Low-Rank Linear Layer



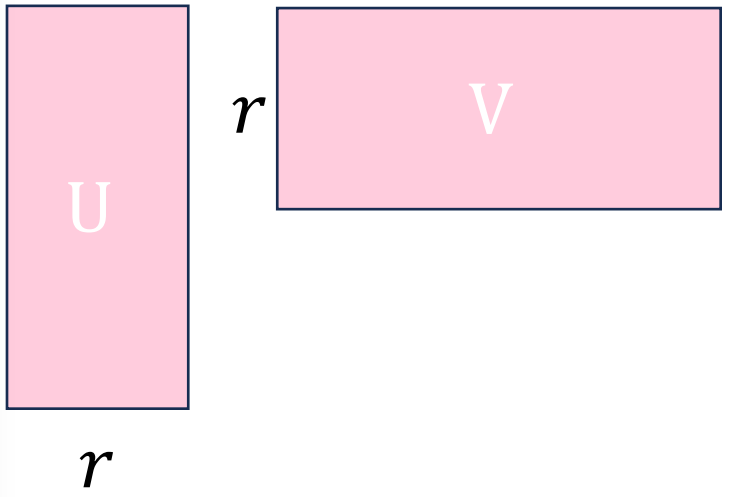
Proposed Approach: Low-Rank Linear Layer

Algorithm 1 Low-Rank Linear Layer in MPGNN

Require: Input features $\mathbf{X} \in \mathbb{R}^{N \times d_{\text{in}}}$, rank r

Ensure: Output features $\mathbf{Y} \in \mathbb{R}^{N \times d_{\text{out}}}$

1: Initialize $\mathbf{U} \in \mathbb{R}^{d_{\text{in}} \times r}$, $\mathbf{V} \in \mathbb{R}^{r \times d_{\text{out}}}$
2: **for** each layer in the MPGNN **do**
3: Compute $\mathbf{Y} \leftarrow \mathbf{X}(\mathbf{UV}) + \mathbf{b}$
4: Apply activation function (e.g., ReLU) to Y
5: $\mathbf{X} \leftarrow \mathbf{Y}$ ▷ Feed to next layer
6: **end for**
7: **return** Y



Proposed Approach: Low-Rank Linear Layer

Algorithm 1 Low-Rank Linear Layer in MPGNN

Require: Input features $\mathbf{X} \in \mathbb{R}^{N \times d_{\text{in}}}$, rank r

Ensure: Output features $\mathbf{Y} \in \mathbb{R}^{N \times d_{\text{out}}}$

1: Initialize $\mathbf{U} \in \mathbb{R}$

2: **for** each layer

3: Compute $\mathbf{Y} \leftarrow \mathbf{X}(\mathbf{UV}) + \mathbf{b}$

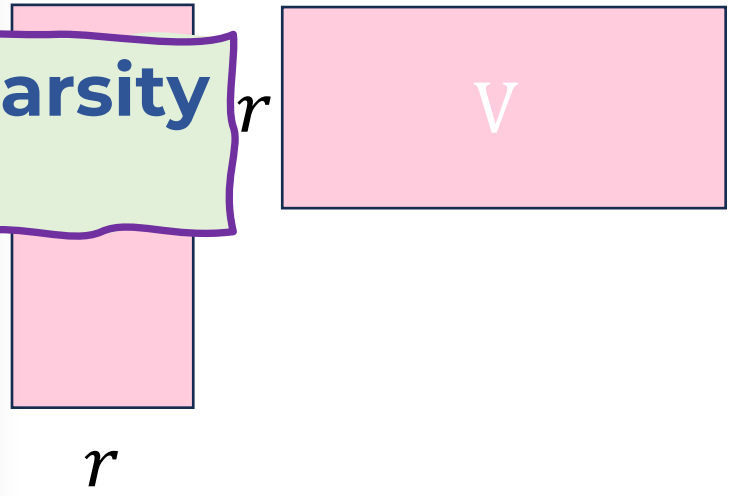
4: Apply activation function (e.g., ReLU) to $\mathbf{Y}$

5: $\mathbf{X} \leftarrow \mathbf{Y}$ ▷ Feed to next layer

6: **end for**

7: **return** $\mathbf{Y}$

We can do this because of the Sparsity of the Weight Matrices.



Proposed Approach: Low-Rank Linear Layer

Advantages

- No fine-tuning/retraining required.

Proposed Approach: Low-Rank Linear Layer

Advantages

- No fine-tuning/retraining required.
- Prevent overfitting by regularizing the model

Proposed Approach: Low-Rank Linear Layer

Advantages

- No fine-tuning/retraining required.
- Prevent overfitting by regularizing the model
- Reduce computational complexity ➔ faster training and inference.

Results

Message and Update State Nets: two fully-connected

 # Transceiver pairs = 50

 # Training samples = 2000

 # Testing samples = 500

Evaluation metric = Sum Rate

Results

Model Size Ratio: $\frac{\text{Original size}}{\text{Low rank size}}$

a1 \ a2	4	16	32	64	128	256
4	58.82	22.42	12.29	6.45	3.31	1.68
16	25.87	15.10	9.71	5.66	3.09	1.62
32	14.81	10.51	7.58	4.87	2.84	1.55
64	7.98	6.54	5.27	3.80	2.44	1.42

a_1: (Maximum) rank for Message Network

a_2: new rank for State Update Network.

Results

Sum Rate Ratio: $\frac{\text{Low rank sum rate}}{\text{Original sum rate}}$

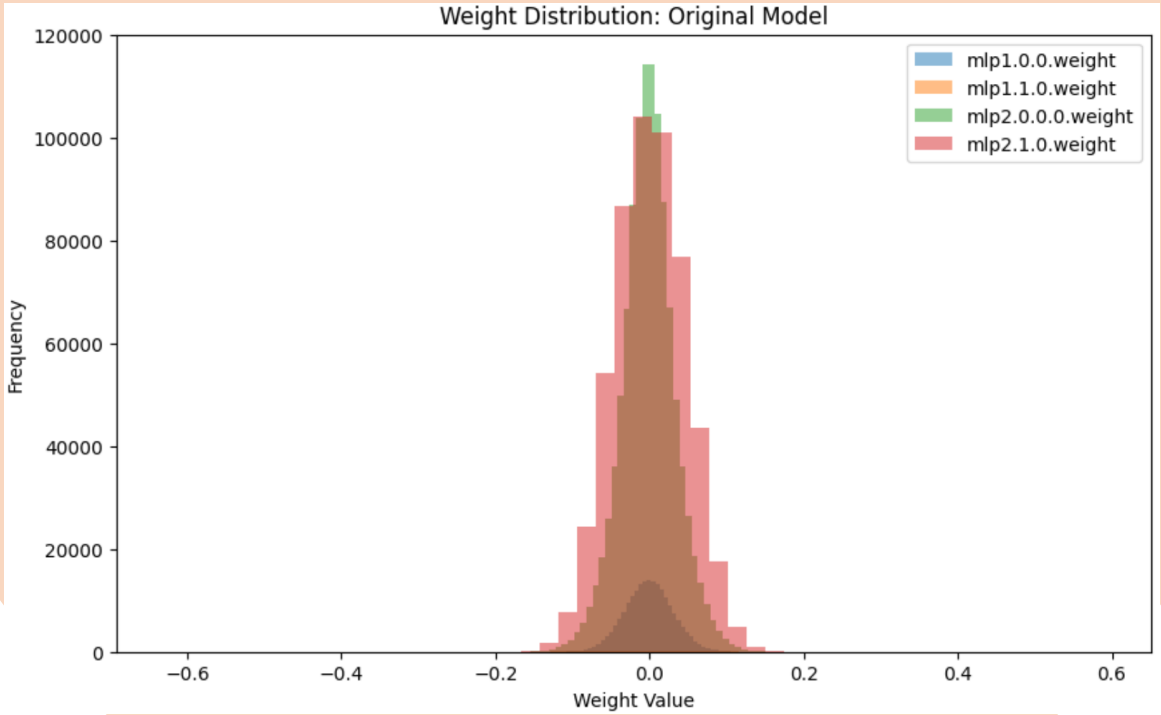
a1 \ a2	4	16	32	64	128	256	512
4	0.762	0.819	0.596	0.739	0.723	0.57	0.765
16	0.971	0.603	0.896	0.99	0.99	0.838	0.819
32	0.743	0.603	0.652	0.714	0.67	0.64	0.99
64	0.80	0.71	0.904	0.790	0.844	0.884	0.853

a_1: (Maximum) rank for Message Network

a_2: new rank for State Update Network.

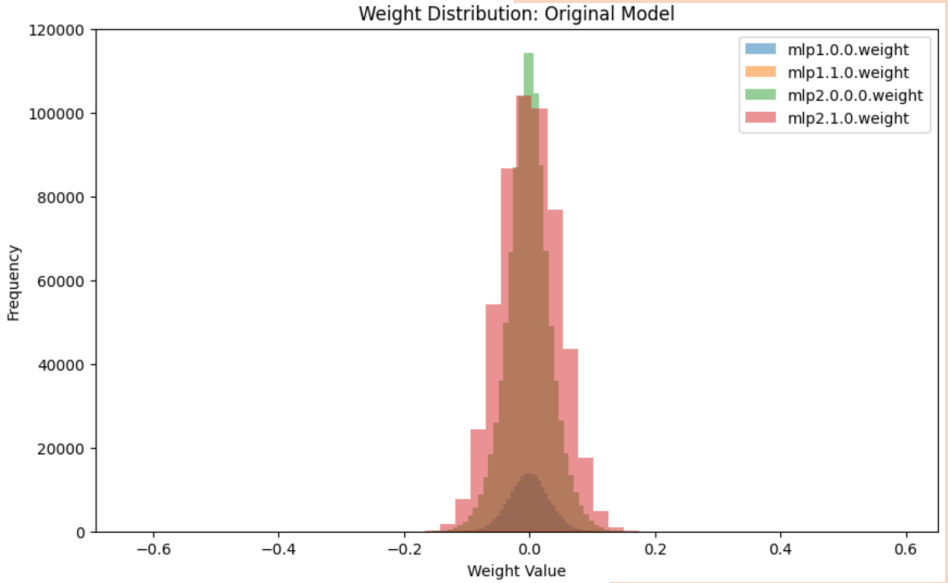
Results

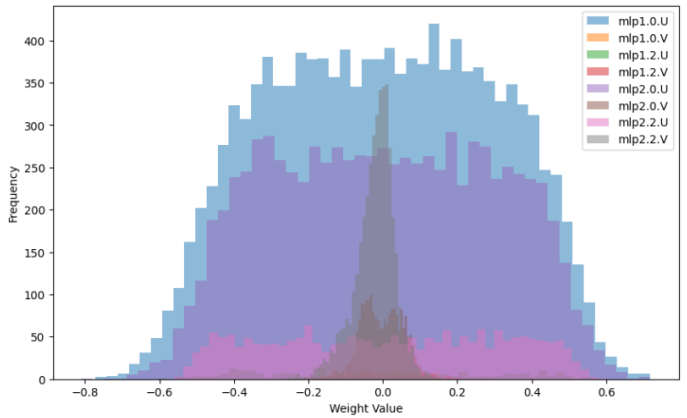
Weight Distribution



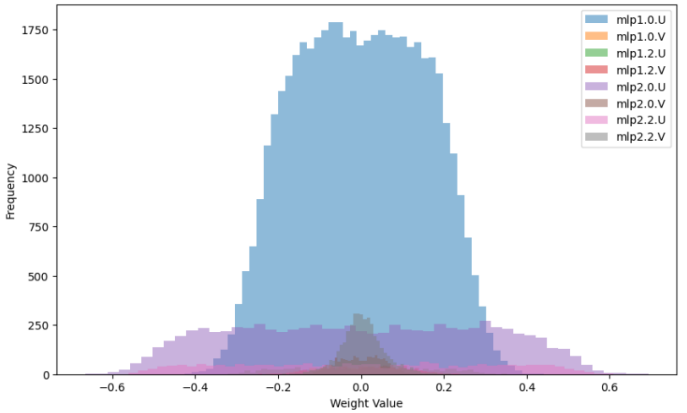
Results

Weight Distribution

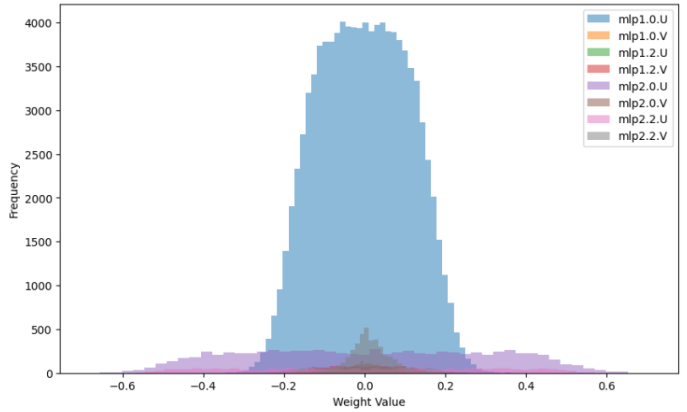




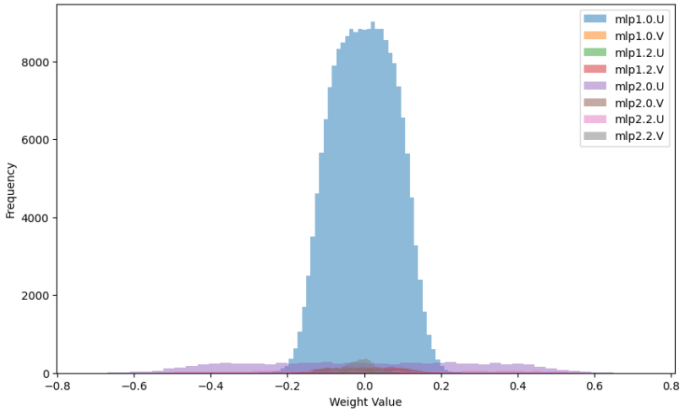
(a) $a_1 = 4$ and $a_2 = 4$



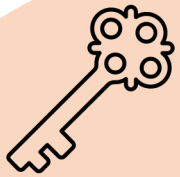
(b) $a_1 = 16$ and $a_2 = 4$



(c) $a_1 = 32$ and $a_2 = 4$



(d) $a_1 = 64$ and $a_2 = 4$



Key Takeaways

✓ GNN-based RRM

✓ Proposed Low-Rank GNN

✓ Reduces the model size 60X

✓ Keep the performance (At best $\leq 1\%$ loss)

My Webpage QR Code ☺

Acknowledgement

- This work was partially supported by NSF under Grant CNS-2150832.

Copyright Notice

www.tinyml.org