

# tinyML<sup>®</sup> Research Symposium

*Enabling Ultra-low Power Machine Learning at the Edge*

April 22, 2024



[www.tinyML.org](http://www.tinyML.org)

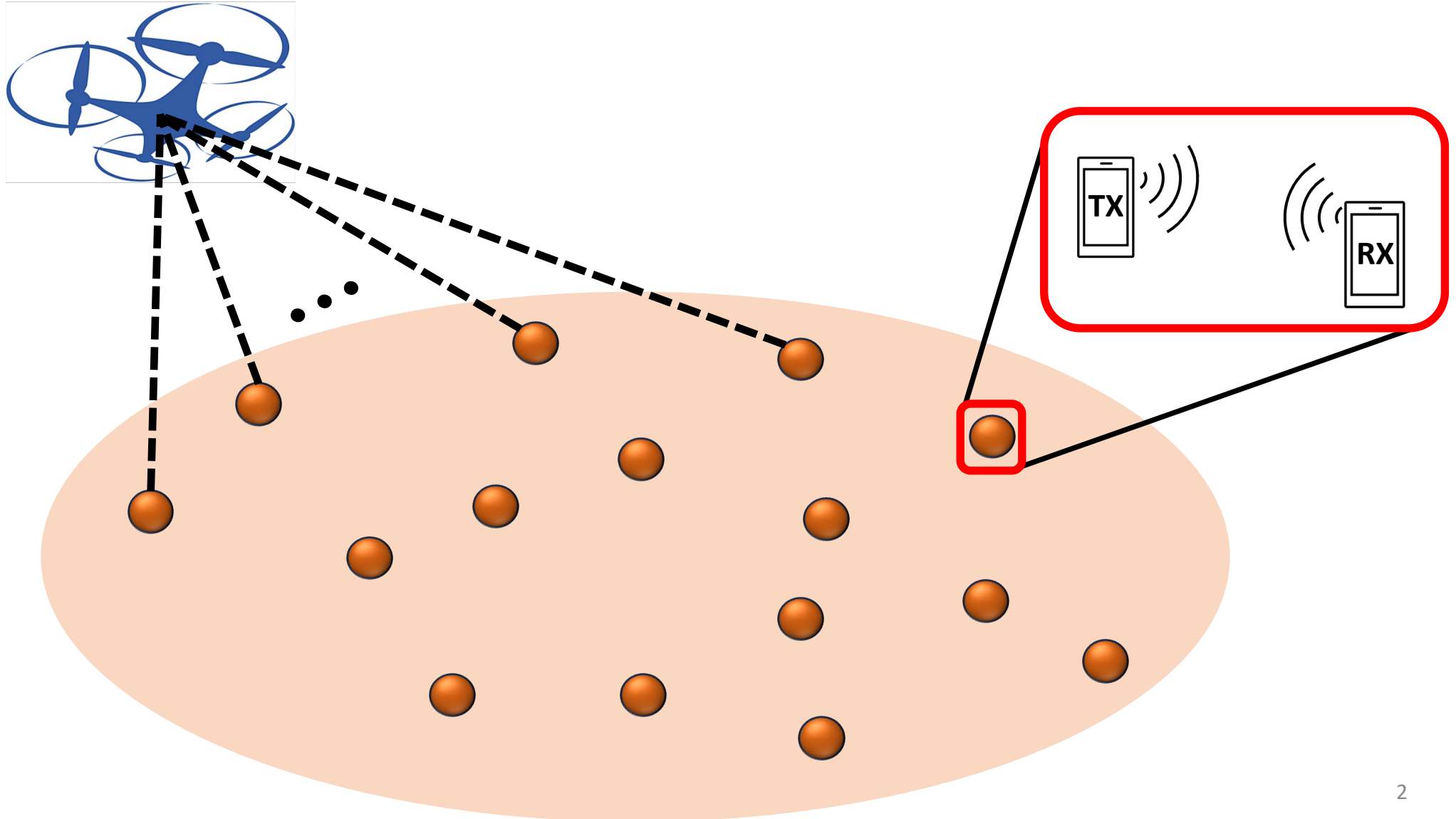


# Tiny Graph Neural Networks for Radio Resource Management

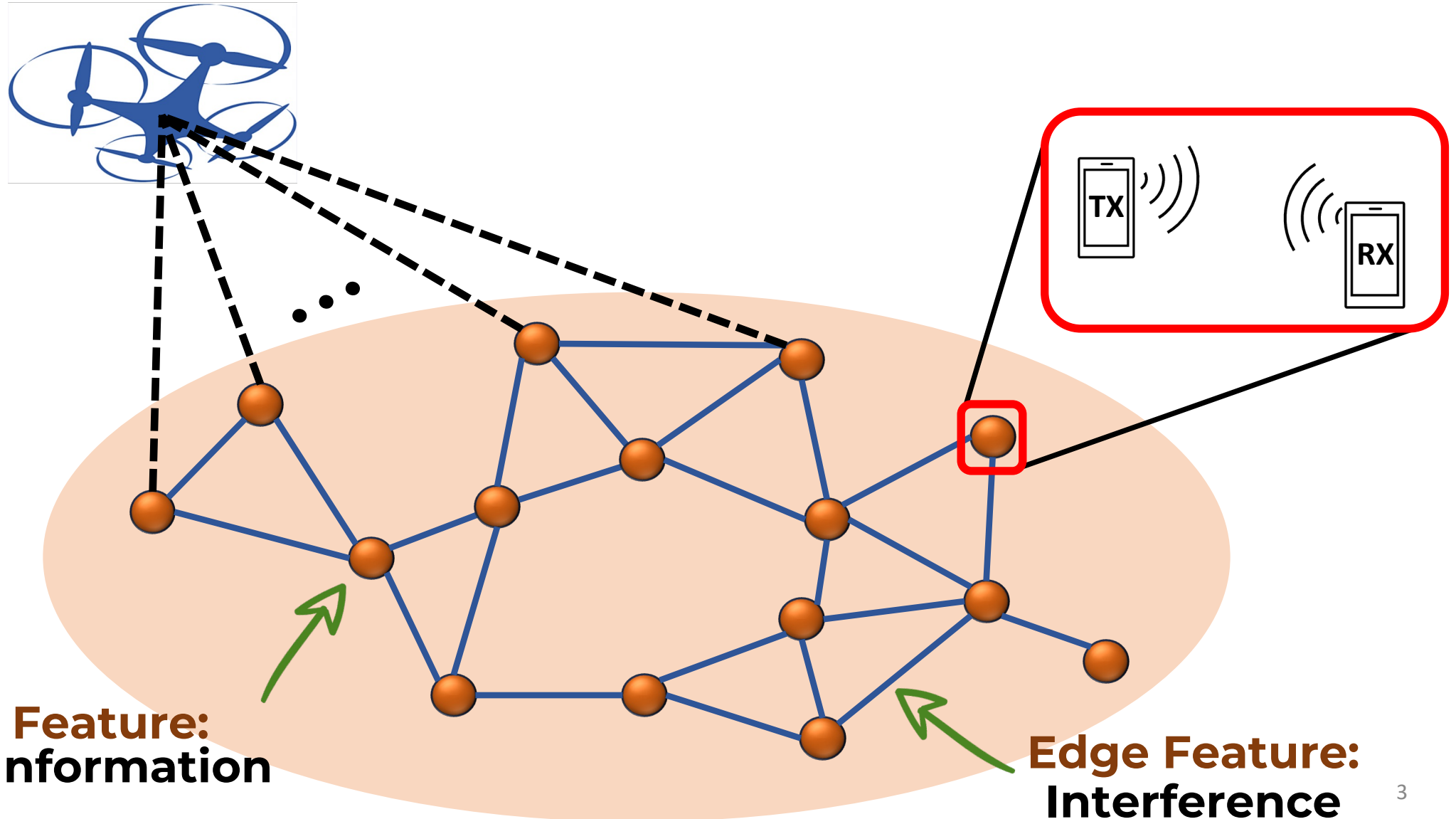
**Ahmad Ghasemi and Hossein Pishro-Nik**



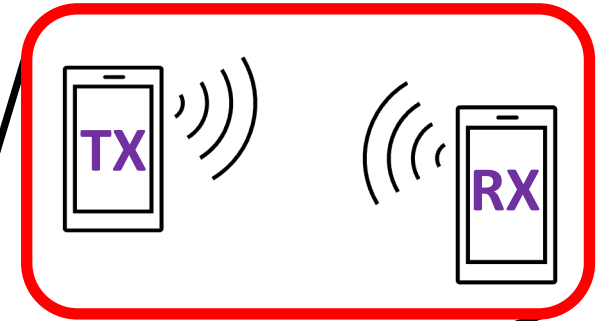
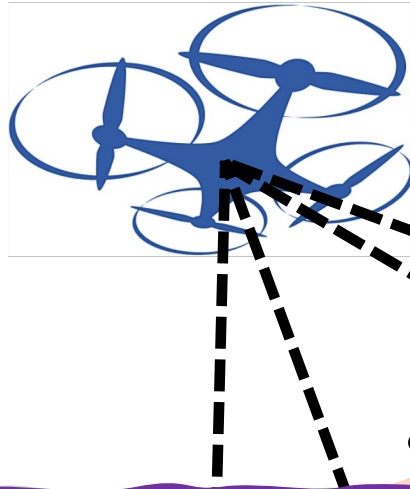
# Radio Resource Management (RRM)



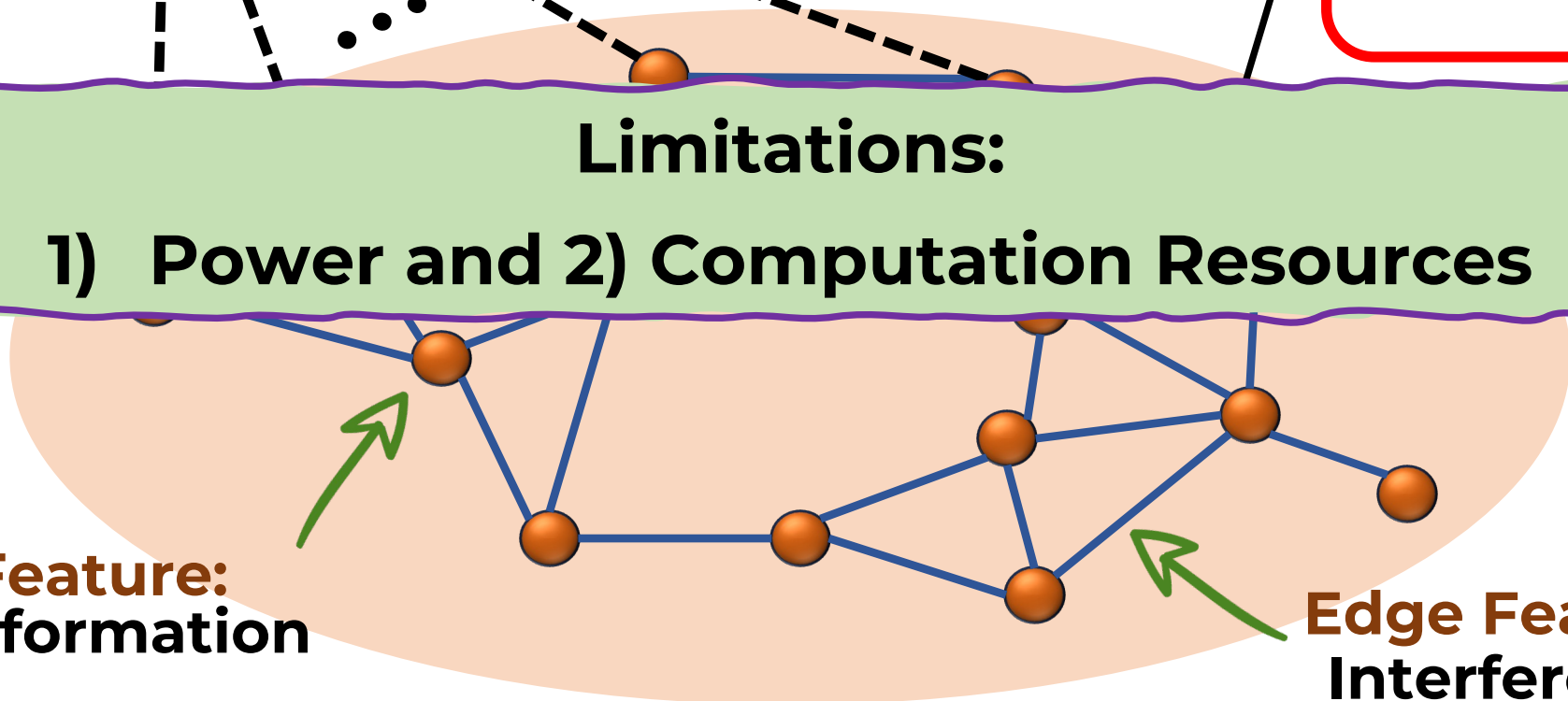
# GNN based RRM



# GNN based RRM



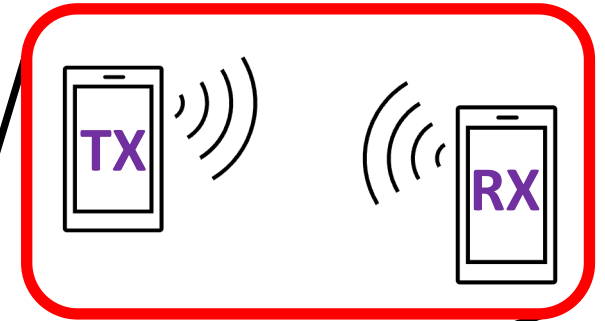
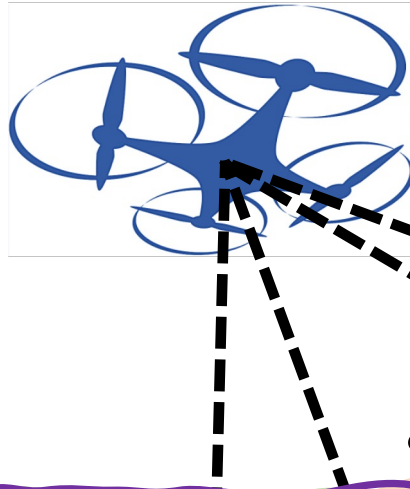
**Limitations:**  
1) Power and 2) Computation Resources



**Vertex Feature:**  
Channel Information

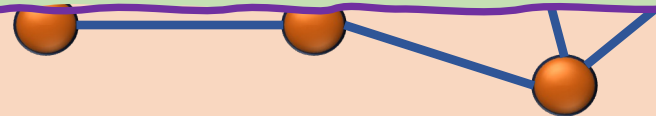
**Edge Feature:**  
Interference

# GNN based RRM



**Limitations:**  
1) Power and 2) Computation Resources  
**→ Smaller Models**

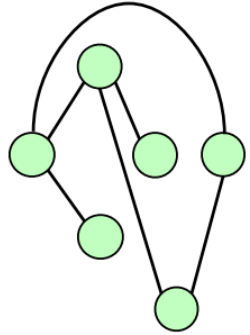
**Vertex Feature:**  
**Channel Information**



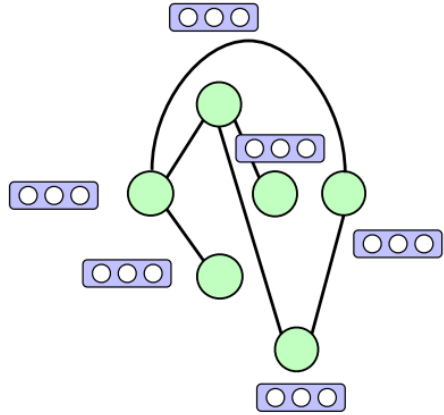
**Edge Feature:**  
**Interference**



# Message Passing GNN



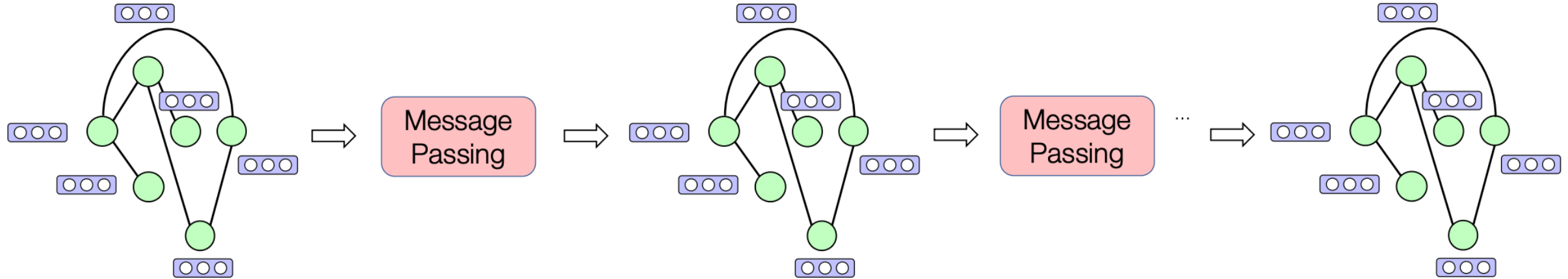
# Message Passing GNN



Input Encoding



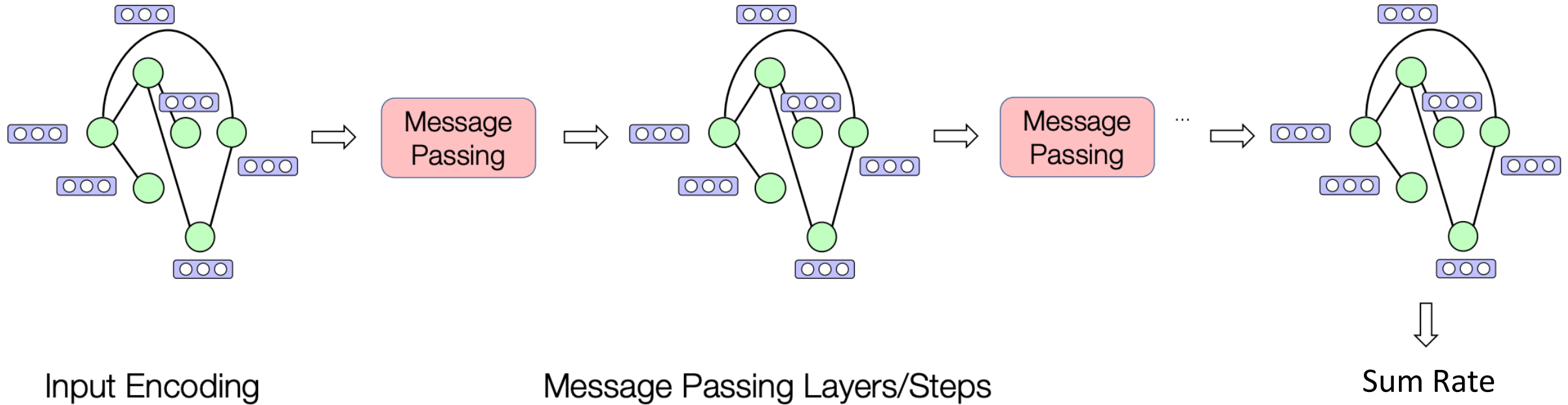
# Message Passing GNN



Input Encoding

Message Passing Layers/Steps

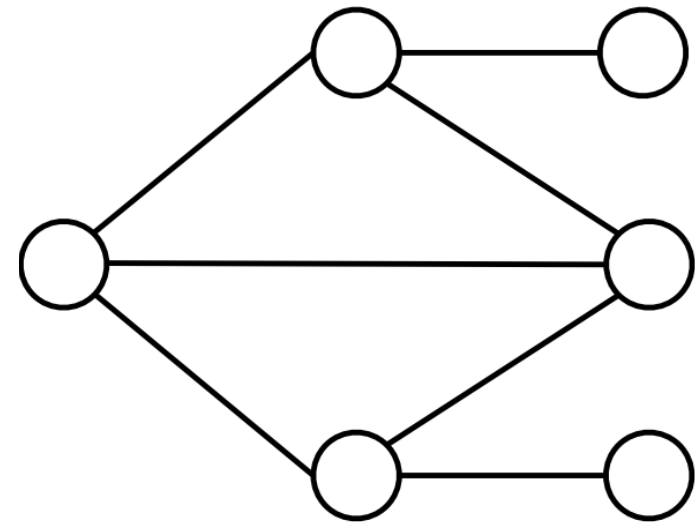
# Message Passing GNN





# Message Passing in MPGNN

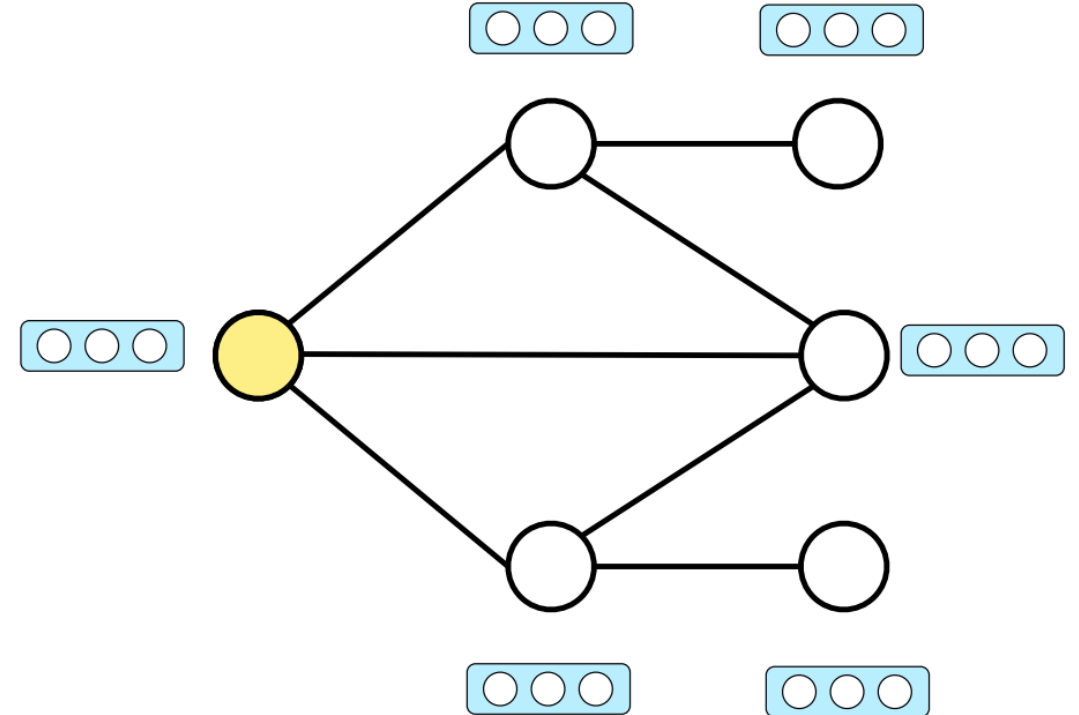
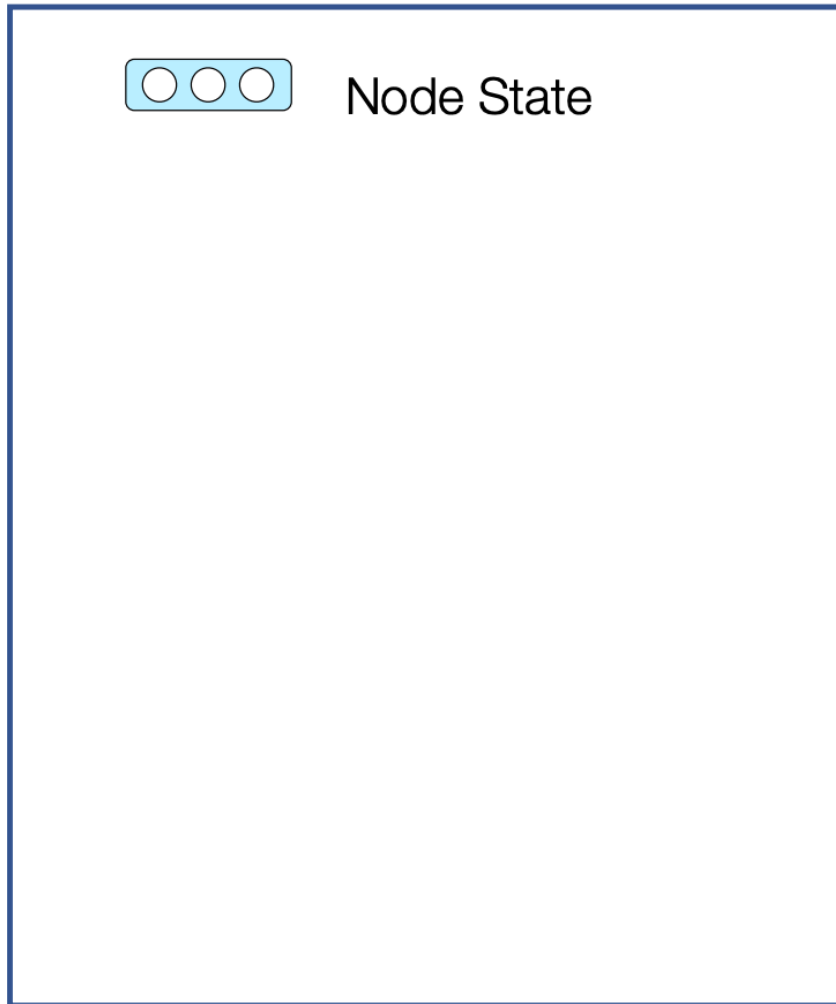
( $t+1$ )-th message passing step/layer



# Message Passing in MPGNN

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

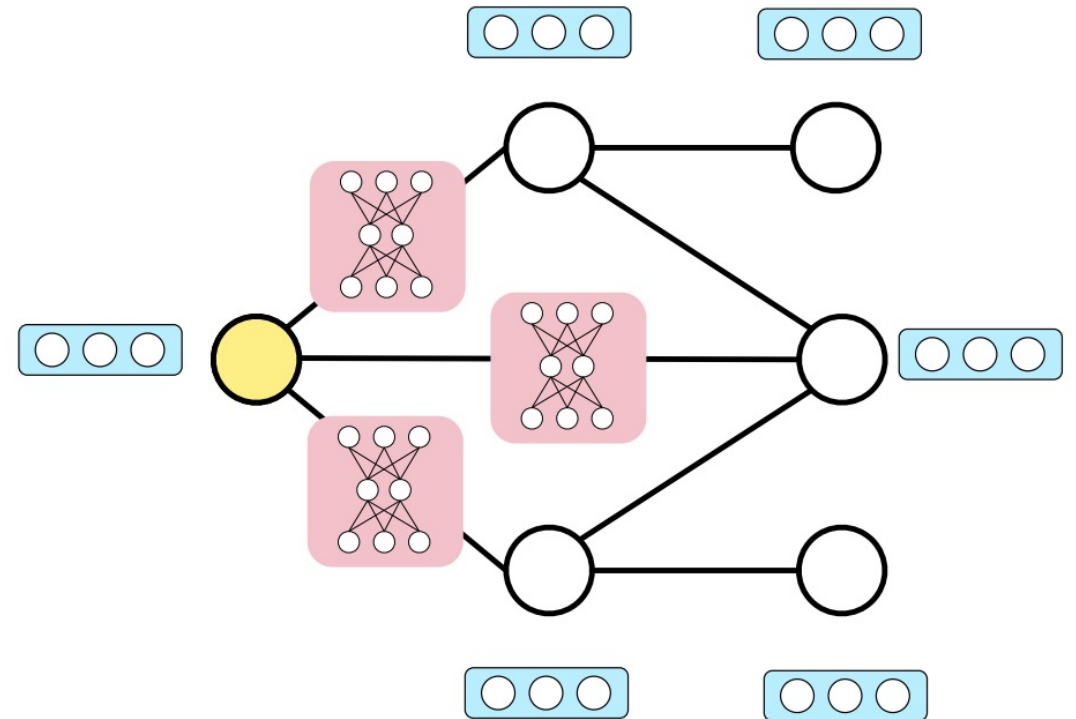
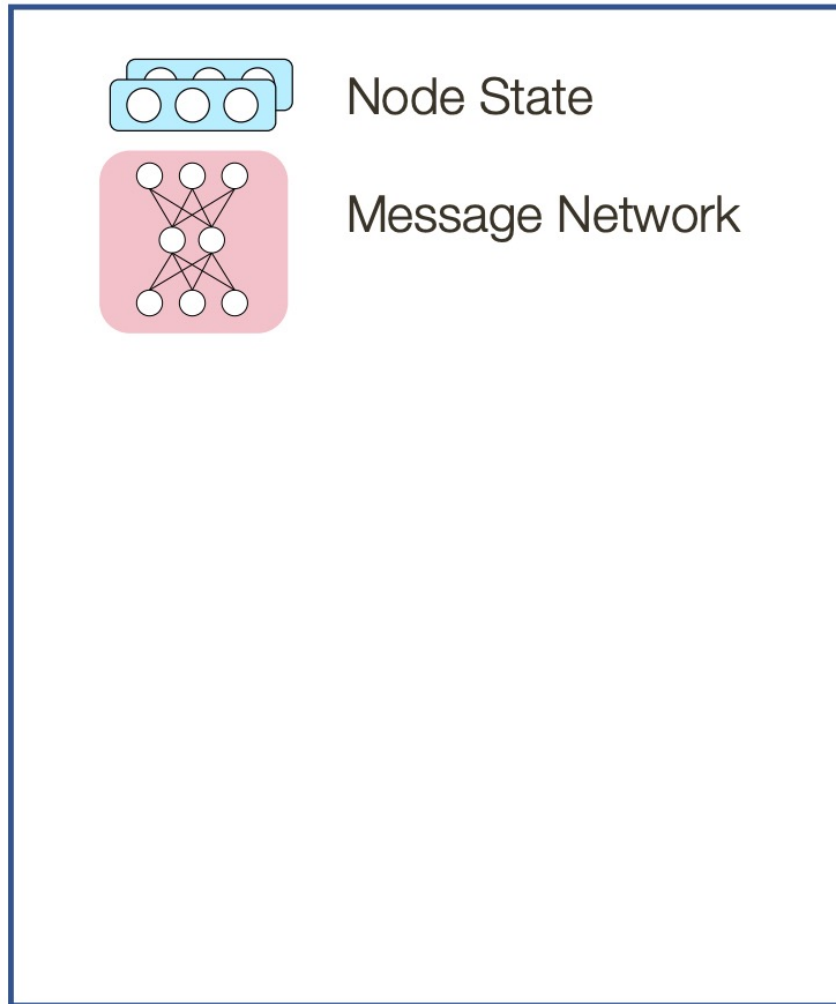
$\mathbf{h}_i^t$



# Message Passing in MPGNN

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

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

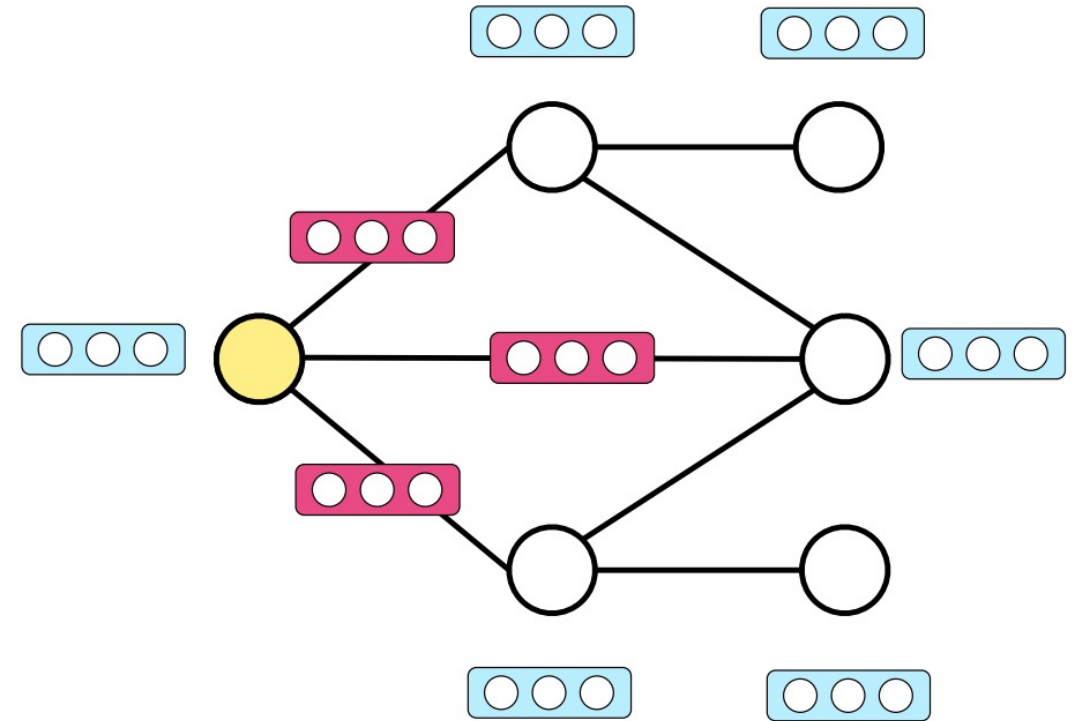
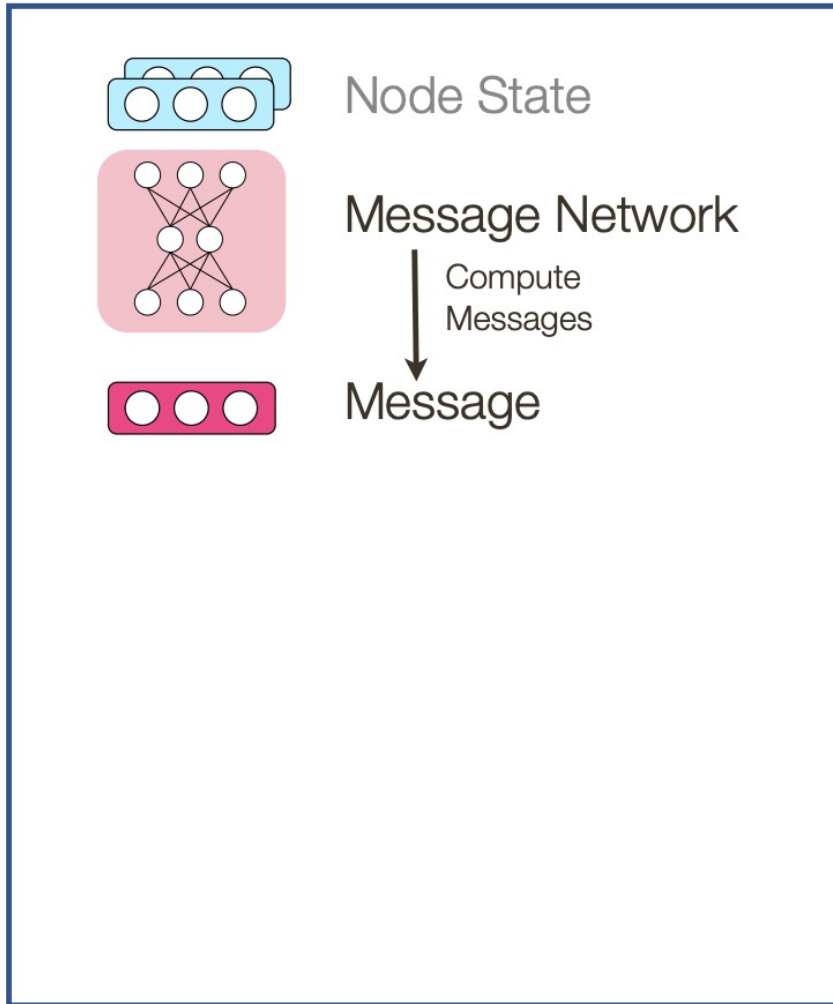


# Message Passing in MPGNN

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

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

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

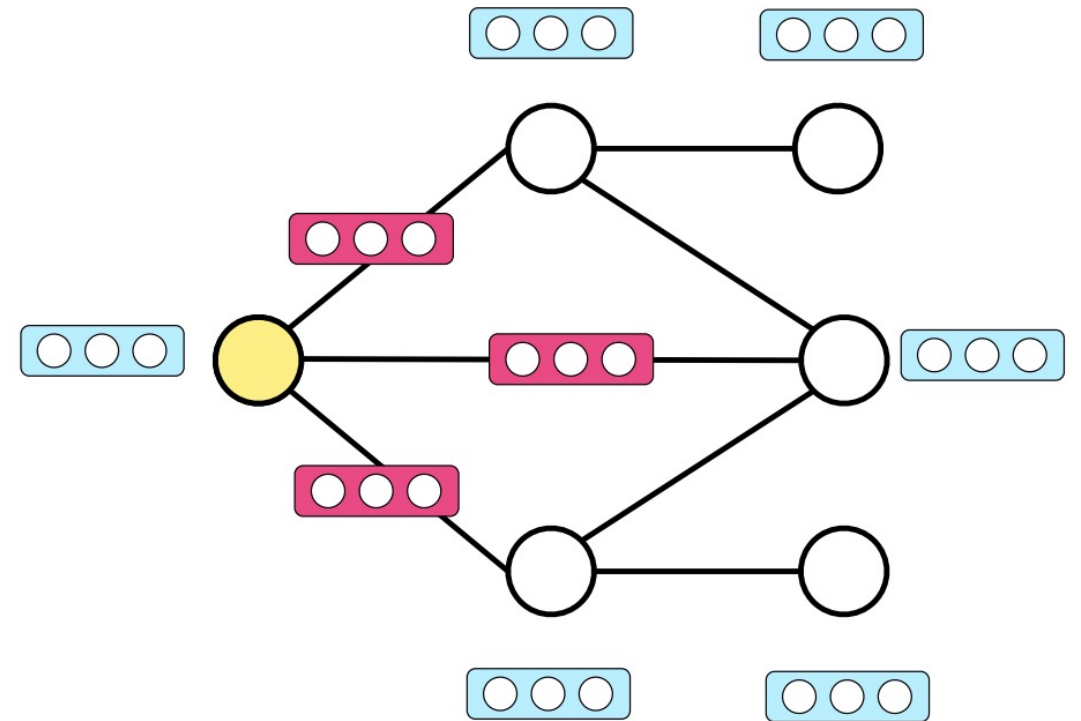
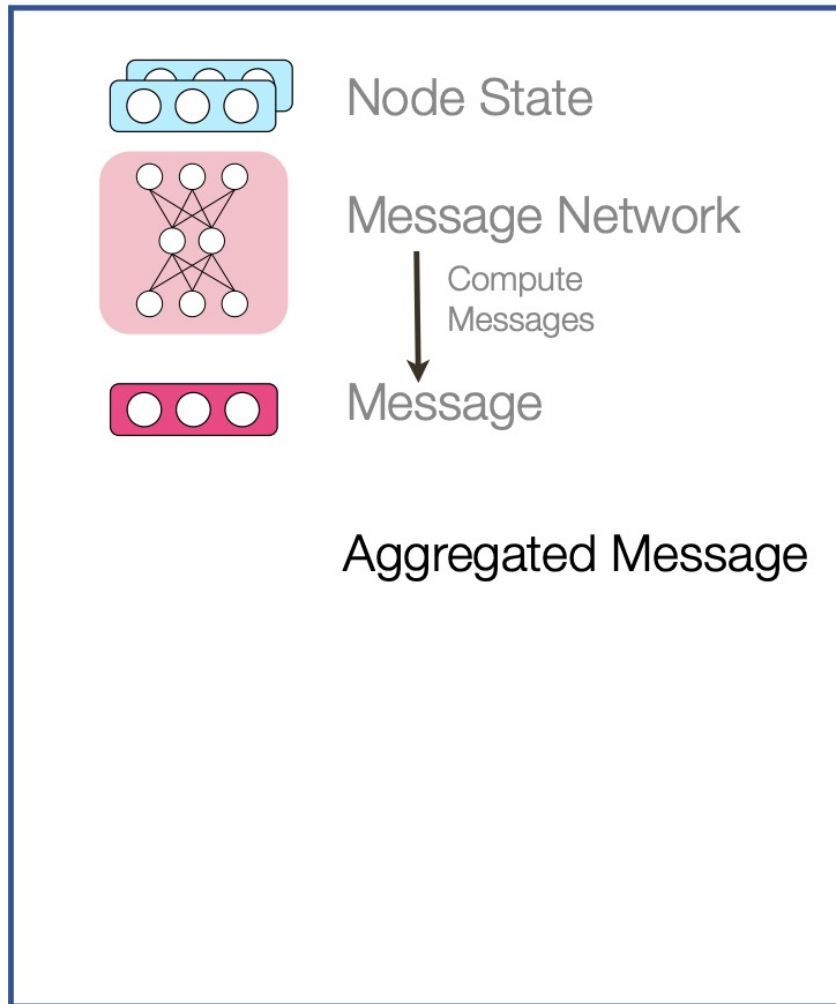


# Message Passing in MPGNN

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

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

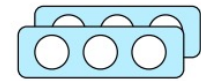
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{h}_i^t)$$



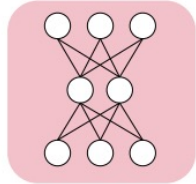
# Message Passing in MPGNN

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

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

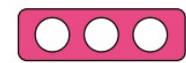


Node State



Message Network

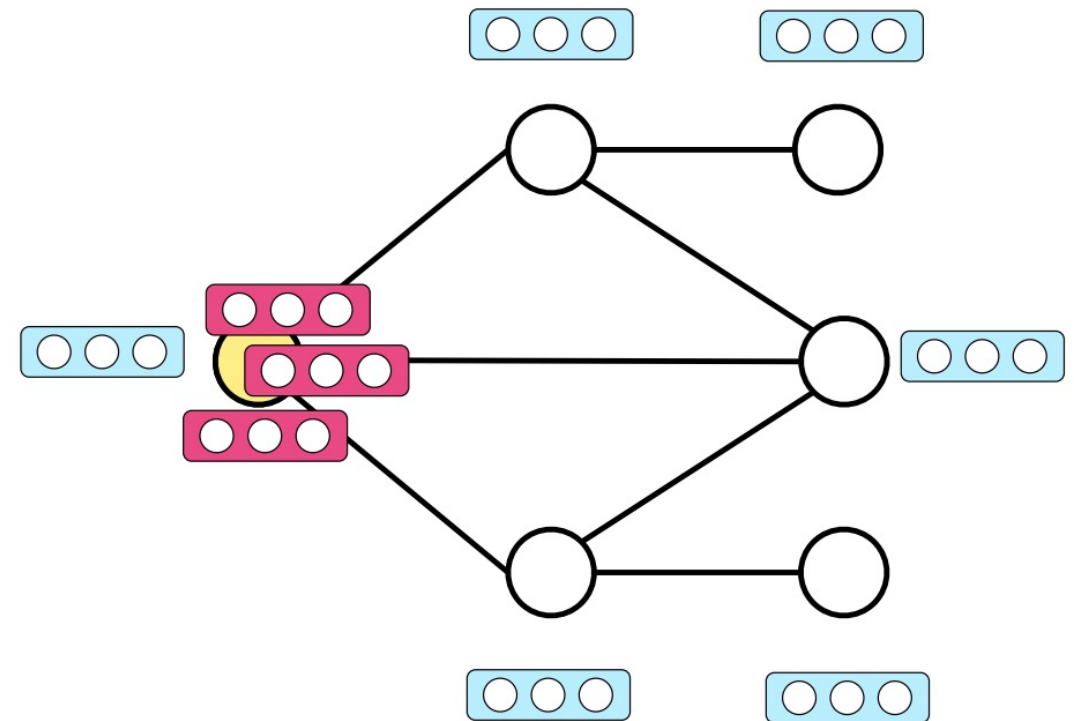
↓ Compute Messages



Message

Aggregated Message

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





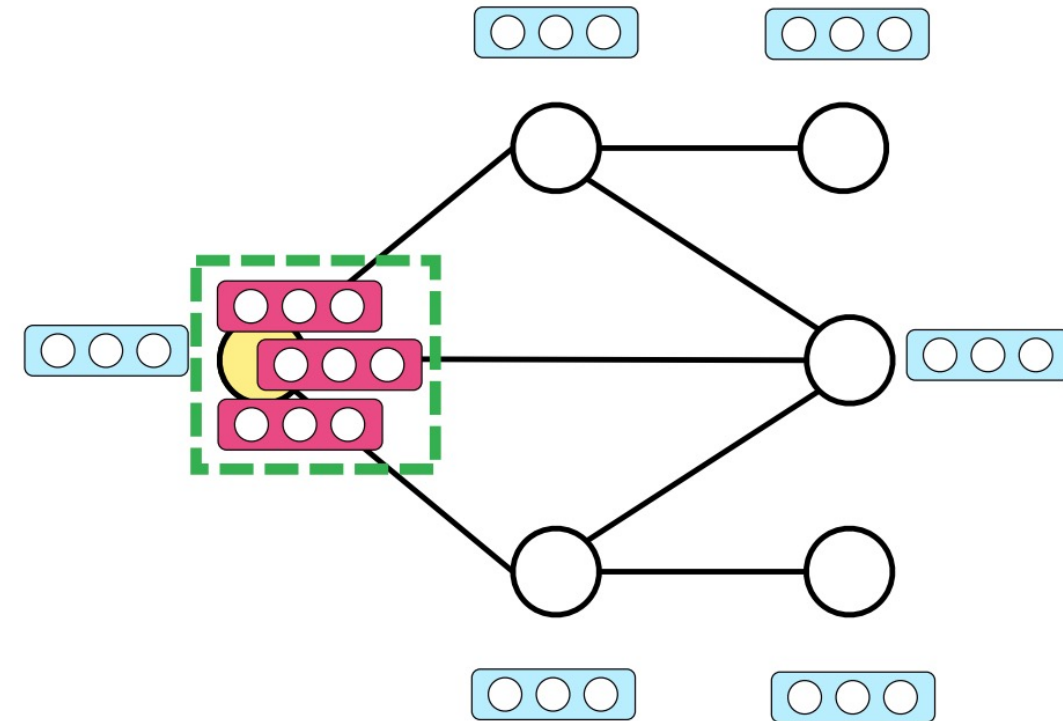
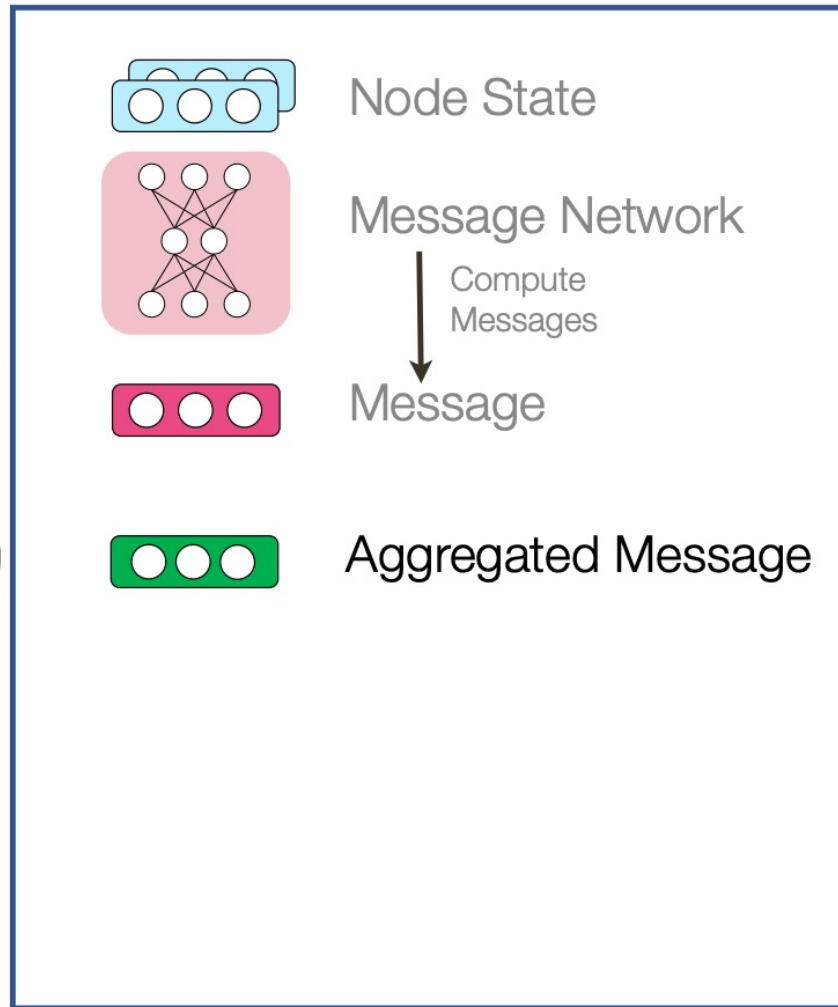
# Message Passing in MPGNN

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

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

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

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



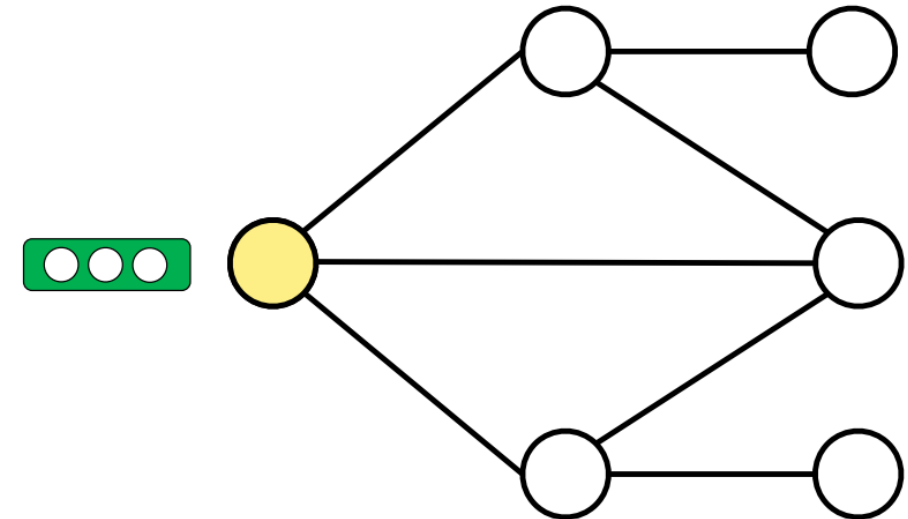
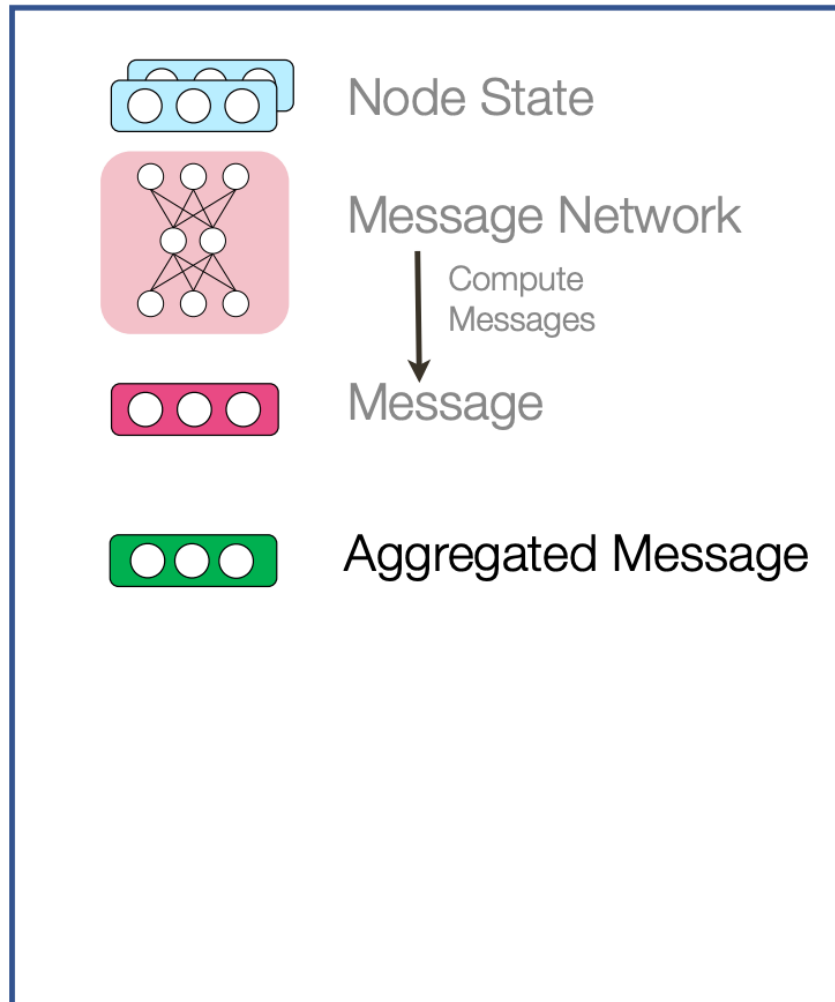
# Message Passing in MPGNN

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

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

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

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



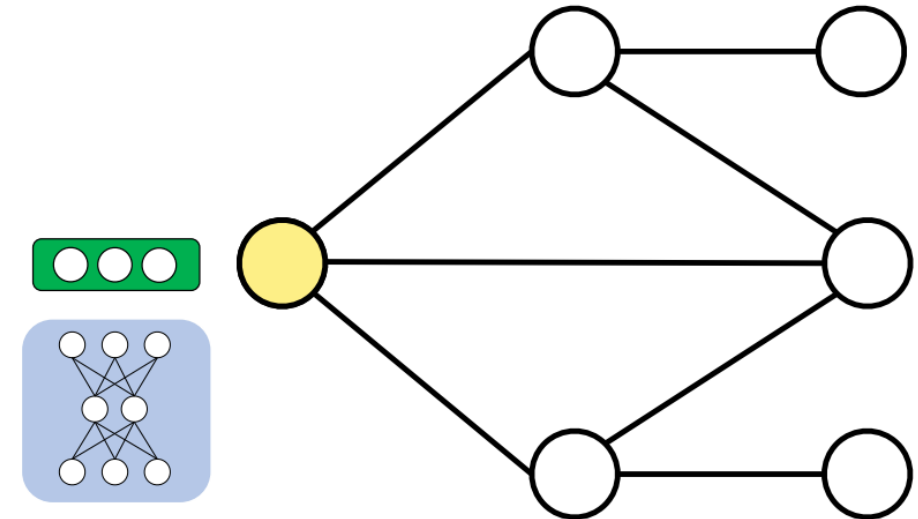
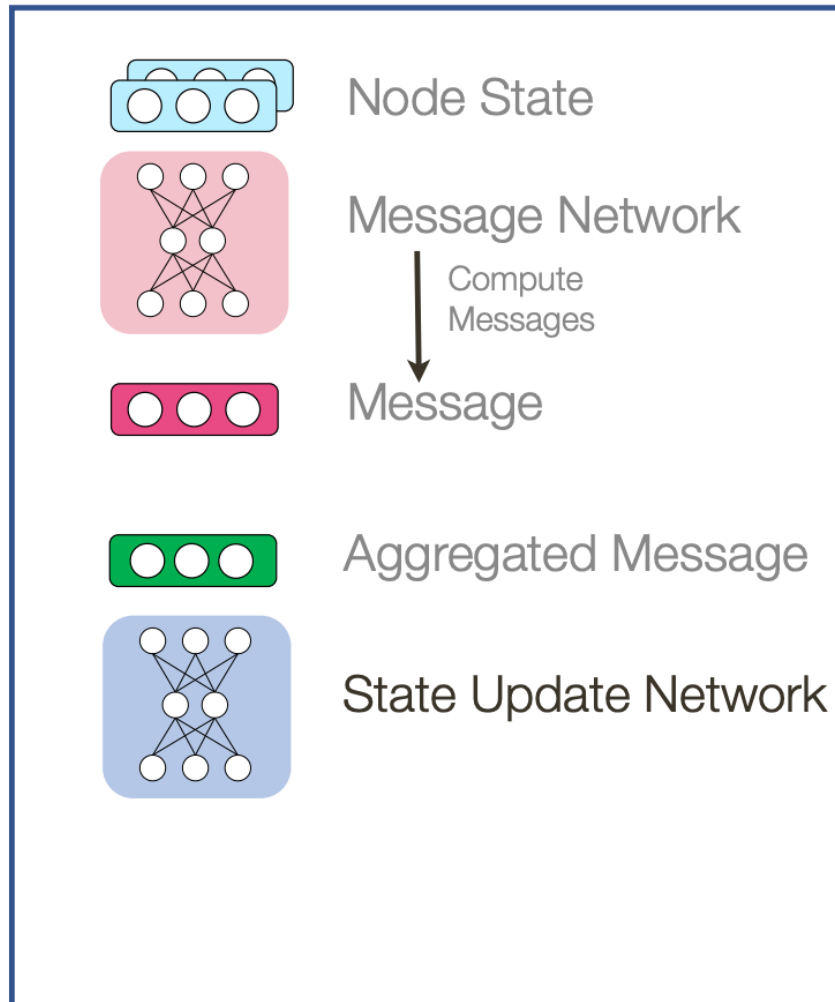
# Message Passing in MPGNN

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

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

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

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



# Message Passing in MPGNN

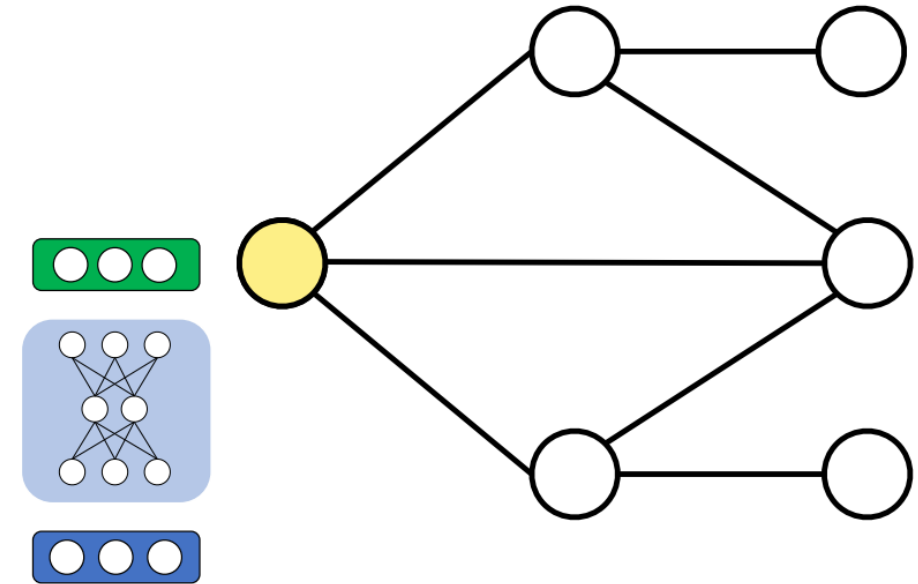
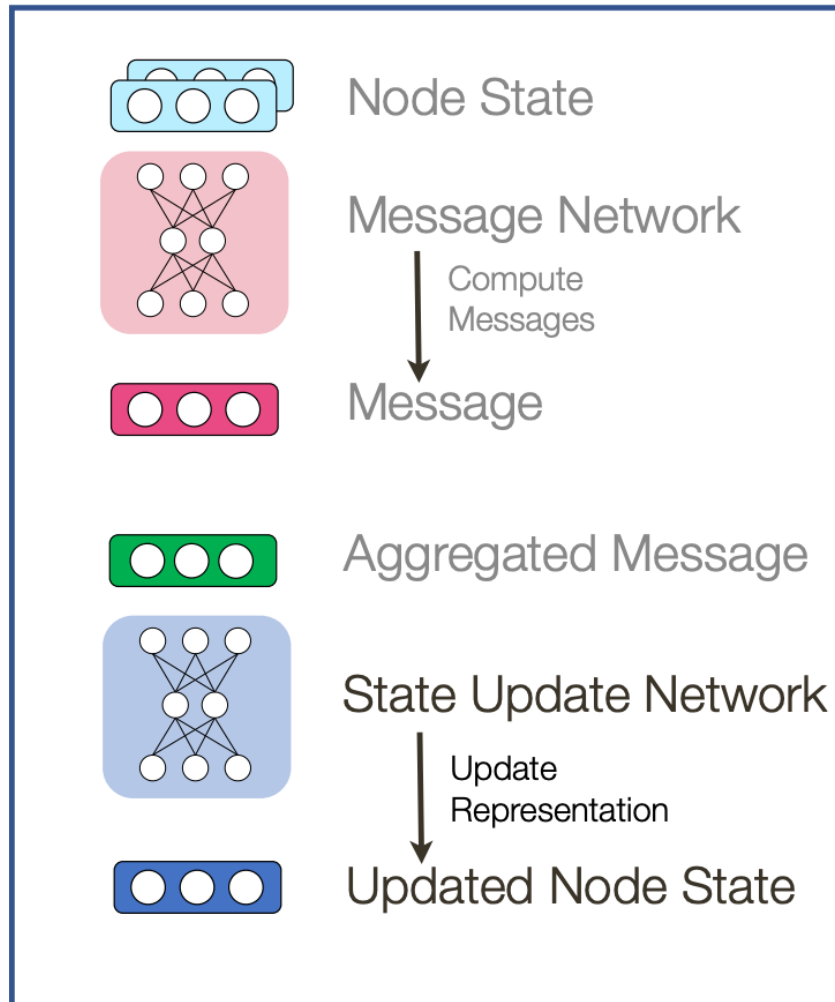
(t+1)-th message passing step/layer

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

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

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

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



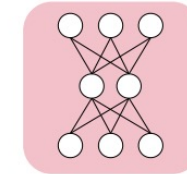
# Message Passing in MPGNN

## 1. Compute Messages

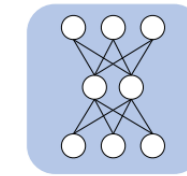
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{e}_{ji})$$

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

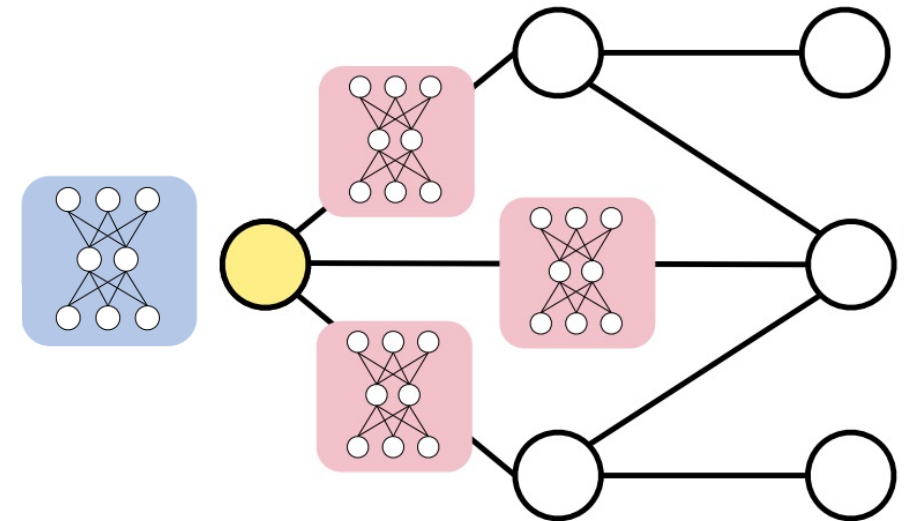
Edge Feature



Message Network



State Update Network



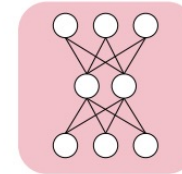
# Message Passing in MPGNN

## 1. Compute Messages

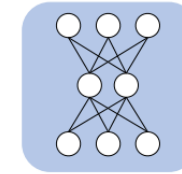
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{e}_{ji})$$

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

Edge Feature



Message Network

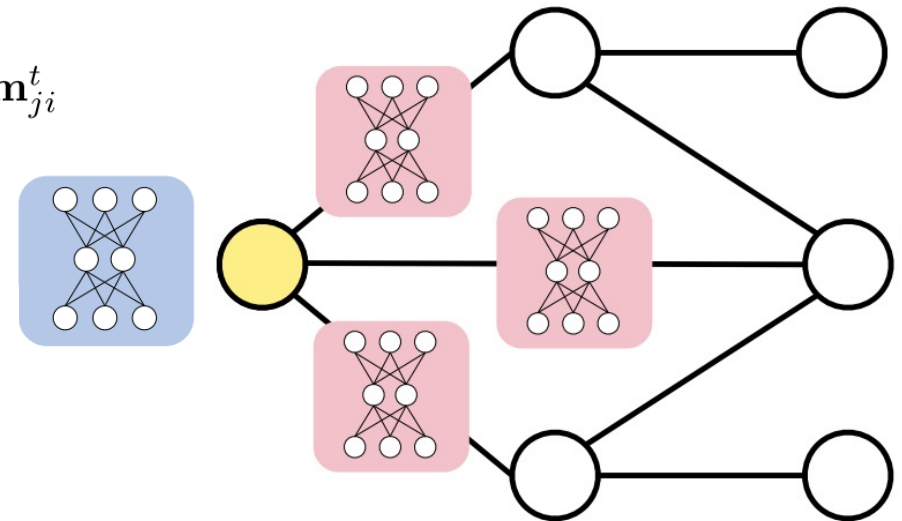


State Update Network

## 2. Aggregate Messages

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

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



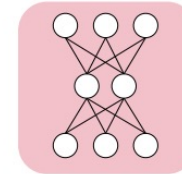
# Message Passing in MPGNN

## 1. Compute Messages

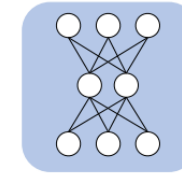
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{e}_{ji})$$

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

Edge Feature



Message Network

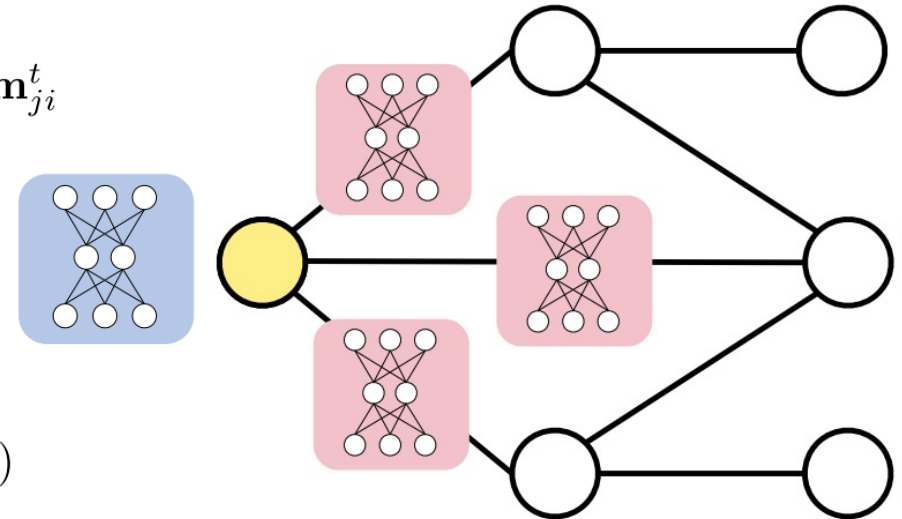


State Update Network

## 2. Aggregate Messages

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

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



## 3. Update Node Representations

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

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

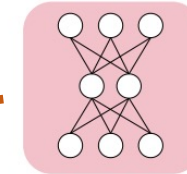
# Message Passing in MPGNN

## 1. Compute Messages

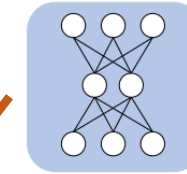
$$\mathbf{m}_{ji}^t = f_{\text{msg}}(\mathbf{h}_j^t, \mathbf{e}_{ji})$$

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

Edge Feature



Message Network

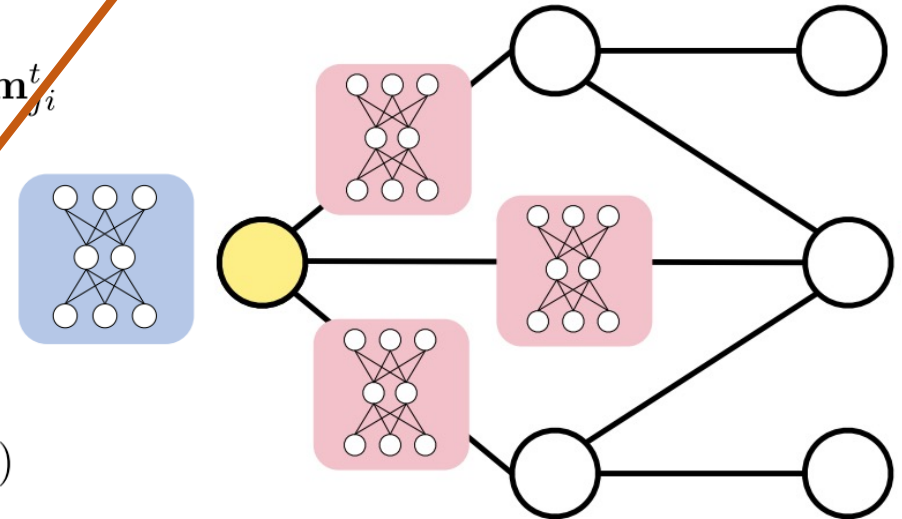


State Update Network

## 2. Aggregate Messages

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

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



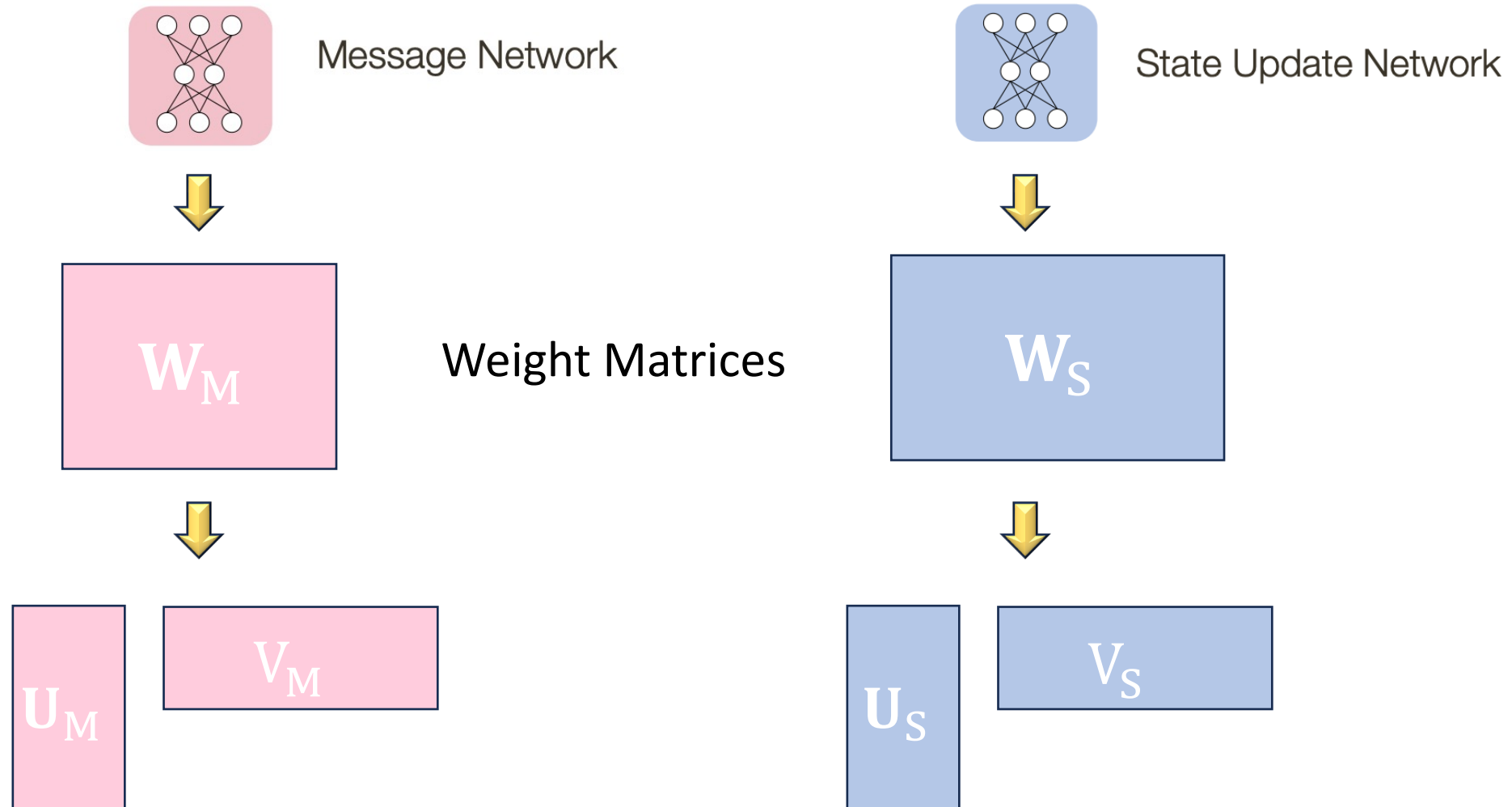
## 3. Update Node Representations

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

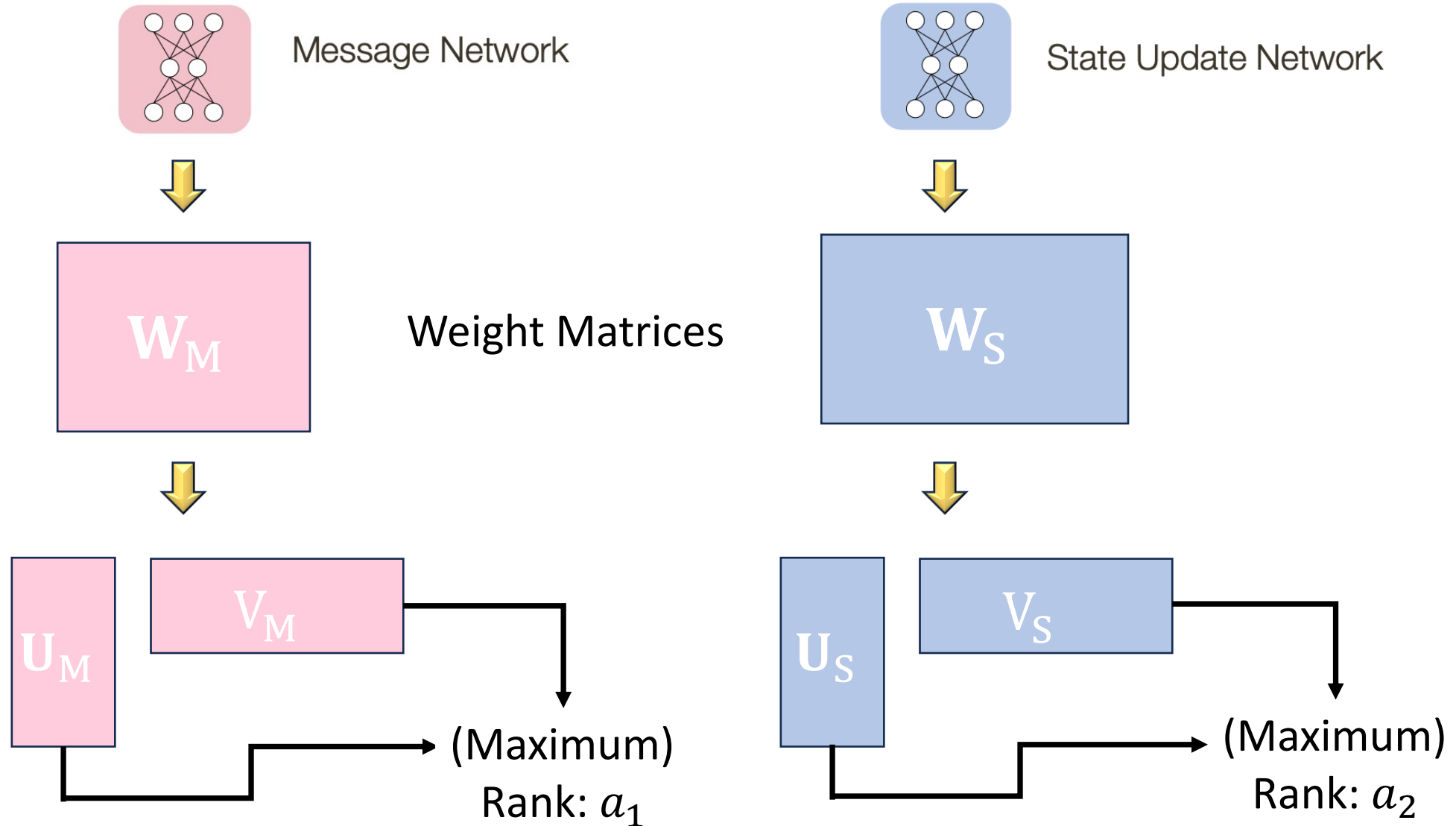
$$f_{\text{update}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t) = \text{MLP}([\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t])$$



# 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**

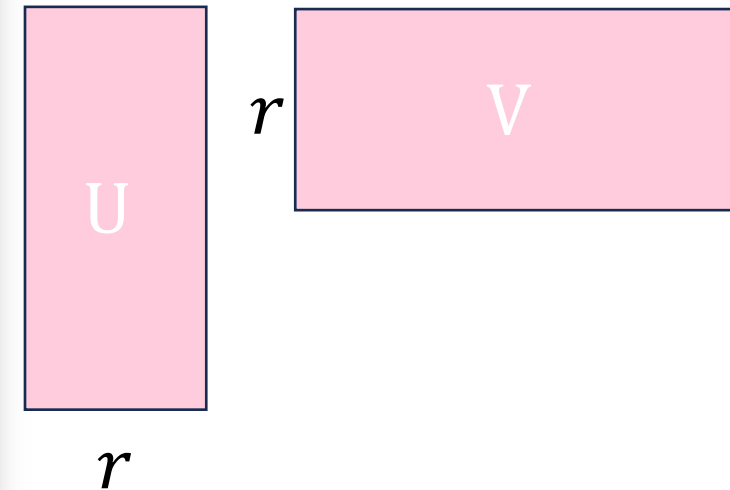
3:     Compute  $\mathbf{Y} \leftarrow \mathbf{X}(\mathbf{UV}) + \mathbf{b}$

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

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

6: **end for**

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





# Proposed Approach: Low-Rank Linear Layer

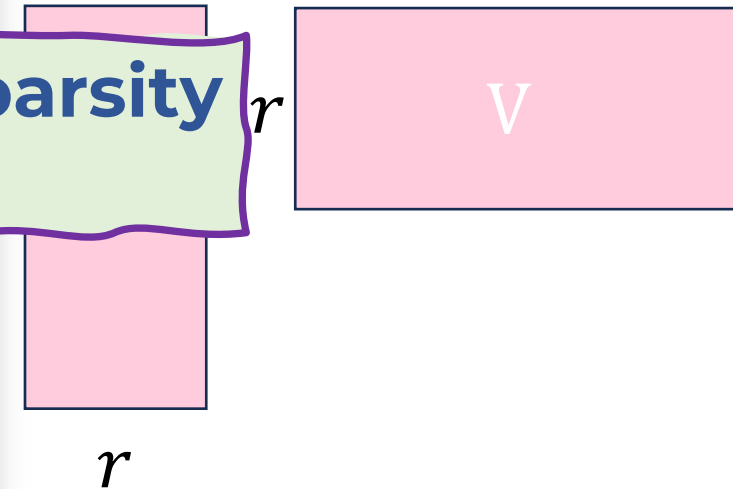
## Algorithm 1 Low-Rank Linear Layer in MPGNN

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

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

- 1: Initialize  $U \in \mathbb{R}^{N \times r}$
- 2: **for** each layer
- 3:     Compute  $Y \leftarrow X(UV) + b$
- 4:     Apply activation function (e.g., ReLU) to  $Y$
- 5:      $X \leftarrow Y$                              ▸ Feed to next layer
- 6: **end for**
- 7: **return**  $Y$

**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:  $\frac{\text{Original size}}{\text{Low rank size}}$

$a_1 \backslash a_2$	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:  $\frac{\text{Low rank sum rate}}{\text{Original sum rate}}$

$a_1 \backslash a_2$	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

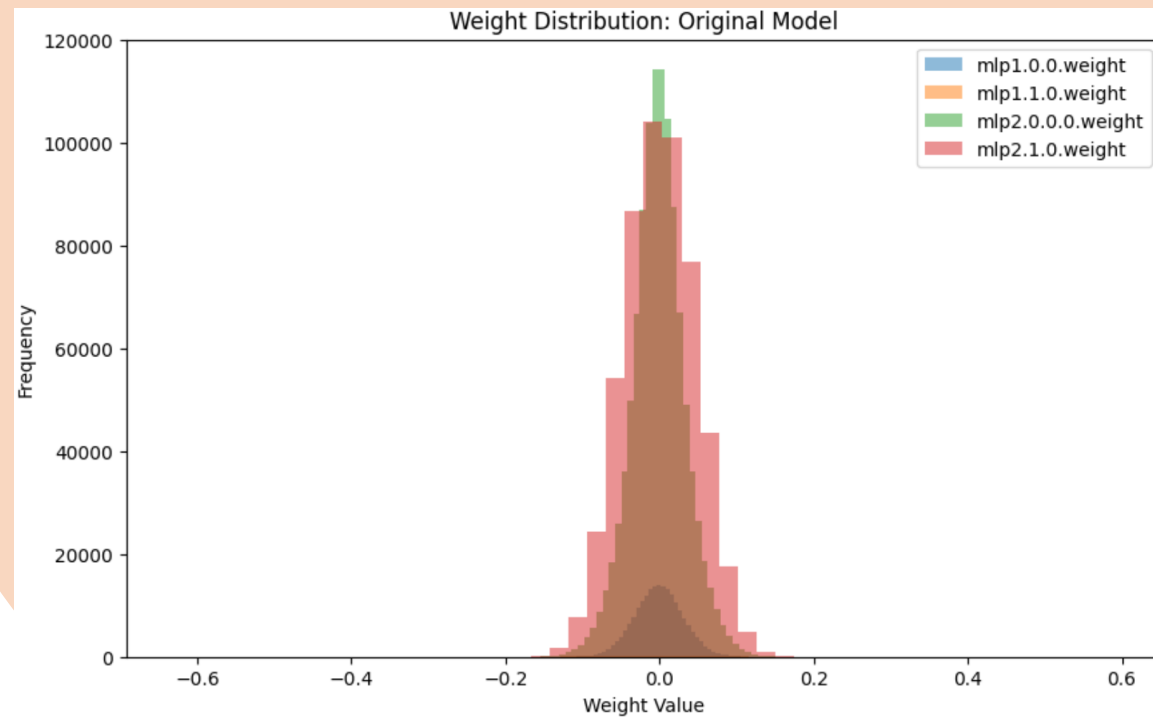
$a_1$ : (Maximum) rank for Message Network

$a_2$ : new rank for State Update Network.



# Results

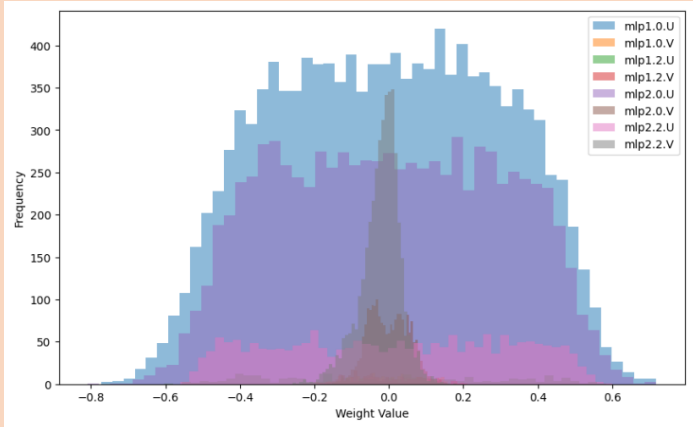
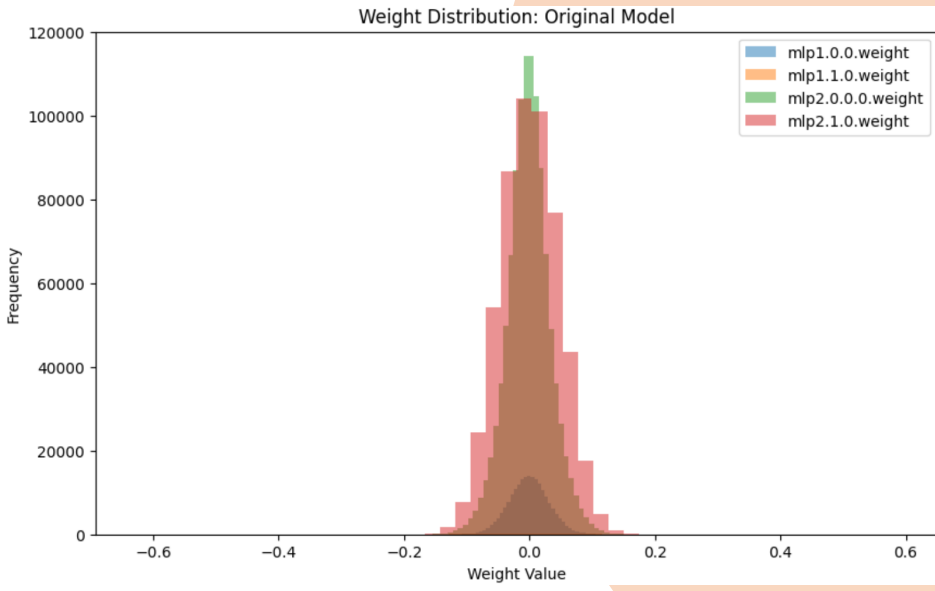
## Weight Distribution



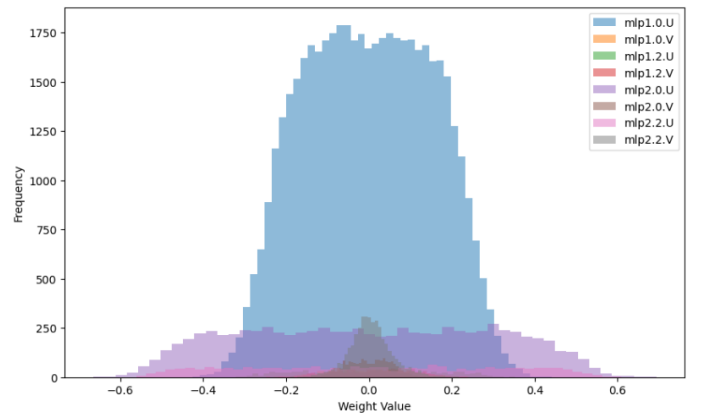


# Results

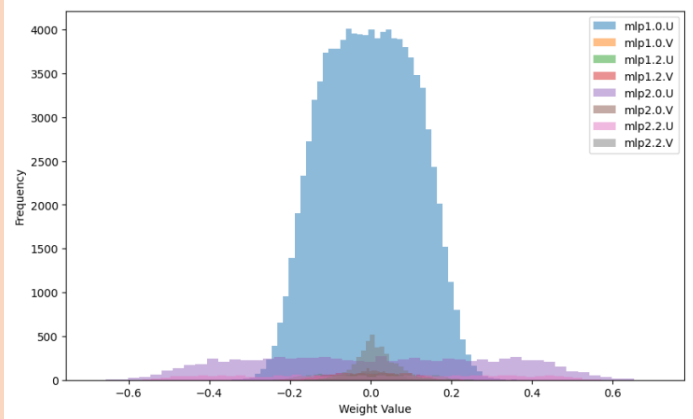
## Weight Distribution



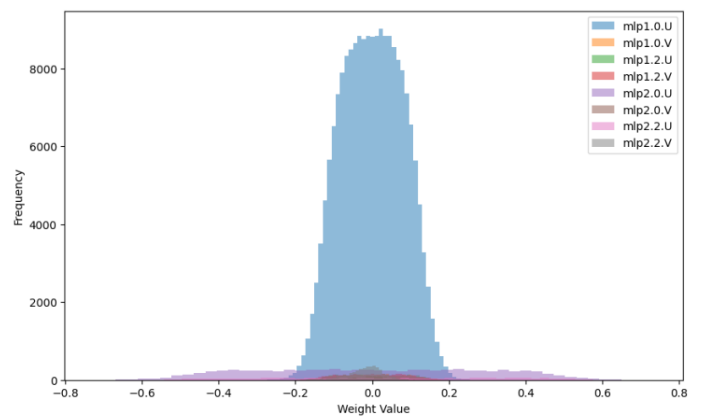
(a)  $a_1 = 4$  and  $a_2 = 4$



(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

This presentation in this publication was presented at the tinyML<sup>®</sup> Research Symposium 2024. The content reflects the opinion of the author(s) and their respective companies. The inclusion of presentations in this publication does not constitute an endorsement by tinyML Foundation or the sponsors.

There is no copyright protection claimed by this publication. However, each presentation is the work of the authors and their respective companies and may contain copyrighted material. As such, it is strongly encouraged that any use reflect proper acknowledgement to the appropriate source. Any questions regarding the use of any materials presented should be directed to the author(s) or their companies.

tinyML is a registered trademark of the tinyML Foundation.

**[www.tinyml.org](http://www.tinyml.org)**