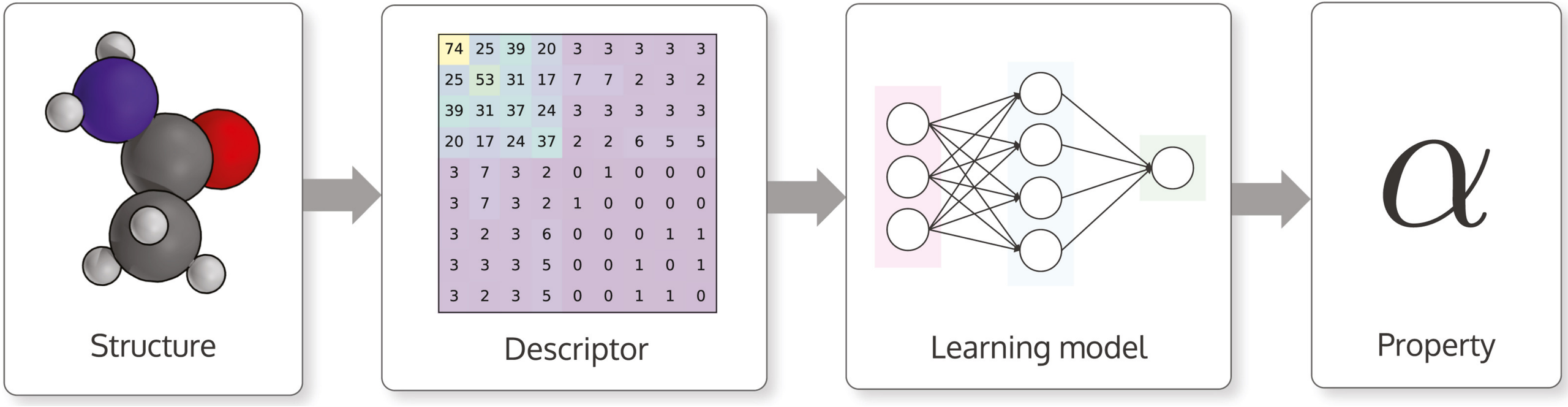
Lecture #8: Graph neural networks for materials science

Slides by Artem Dembitskiy

Goals/Agenda

- Know how to represent an atomic structure as a graph
- Learn about fundamentals of graph neural networks (GNNs)
- Crystal graph convolutional neural network overview

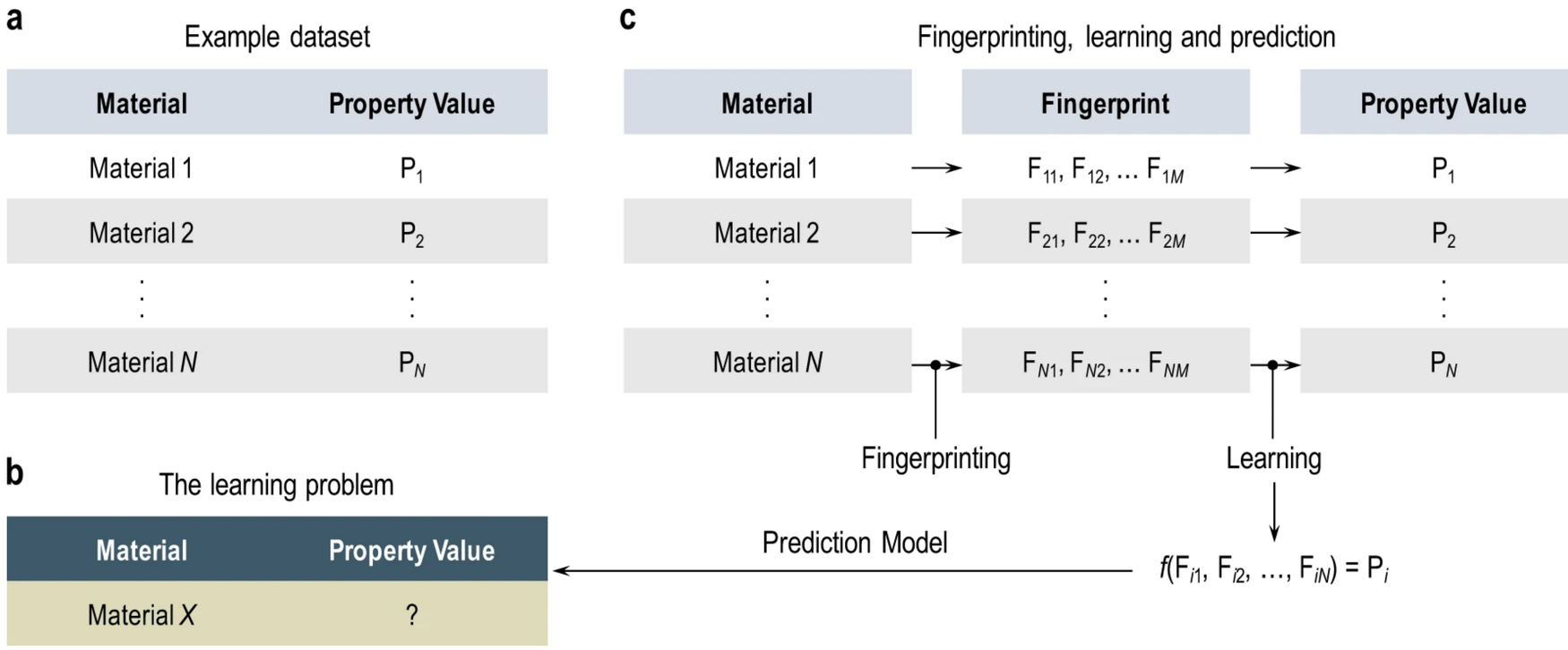

74 25 39 20 3 3 3 3 3
25 53 31 17 7 7 2 3 2
39 31 37 24 3 3 3 3 3
20 17 24 37 2 2 6 5 5
3 7 3 2 0 1 0 0 0
3 7 3 2 1 0 0 0 0
3 2 3 6 0 0 0 1 1
3 3 3 5 0 0 1 0 1
3 2 3 5 0 0 1 1 0
Structure
Descriptor
Learning model
Property

How to represent an atomic structure in a machine readable format?

Previously on

- Aggregation of element features
- Aggregation of structure features
- Feature engineering
- ...

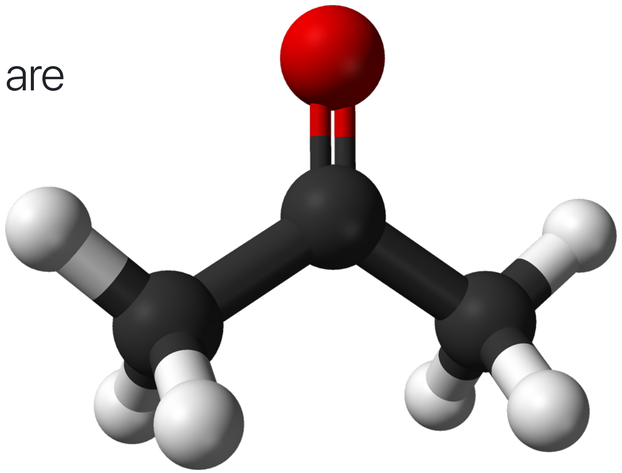
i.e., we use tables to train models

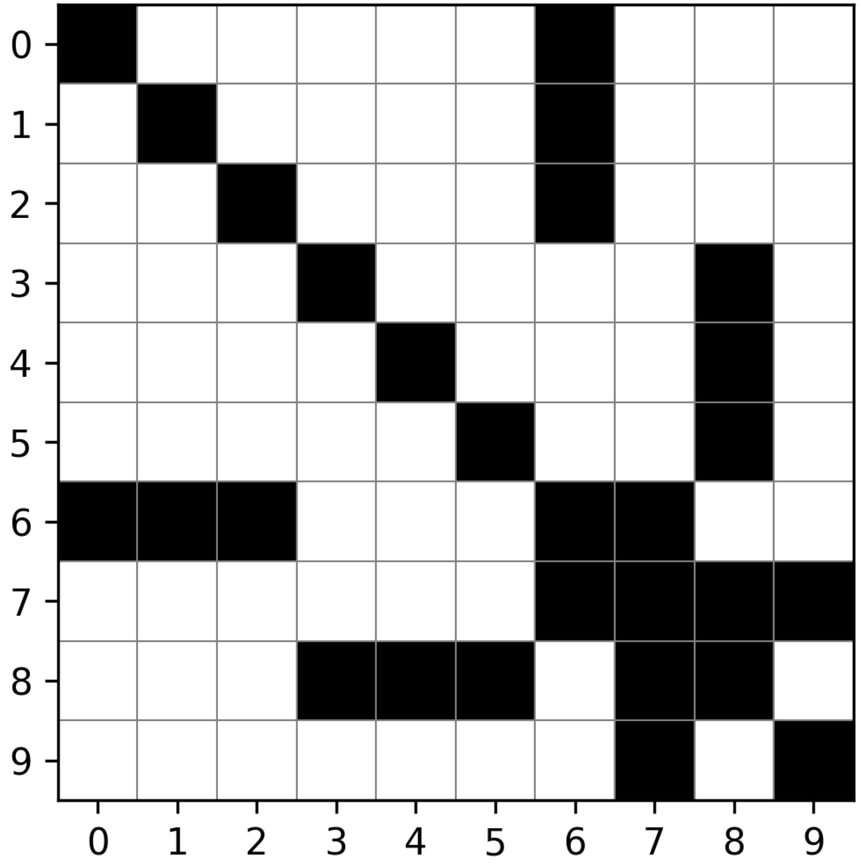


Adjacency matrix

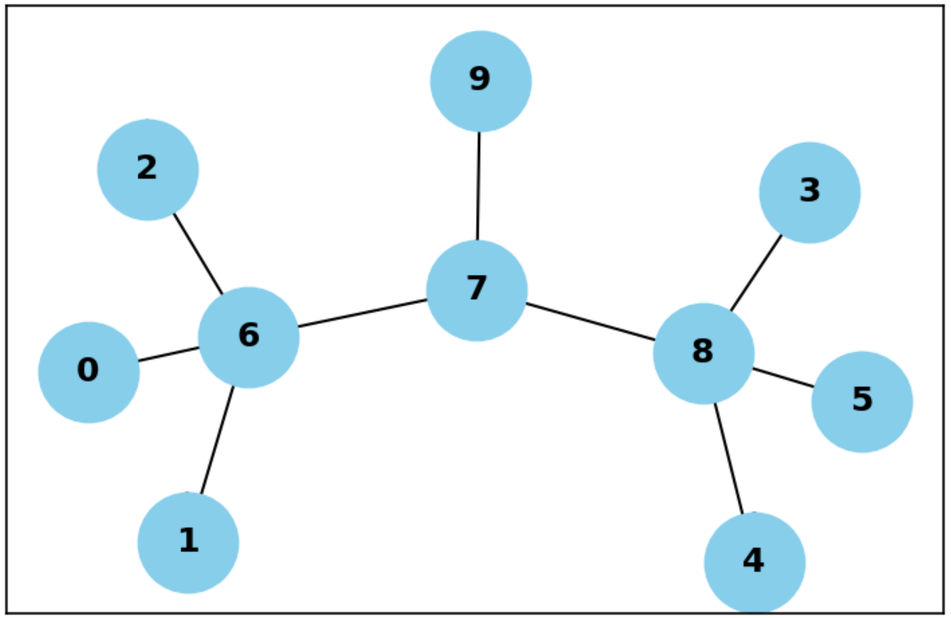
- Tells you whether pairs of atoms are adjacent or not
- Represents a finite graph
- Sparse representation
  - A lot of zeros
  - N atoms, d average neighbours

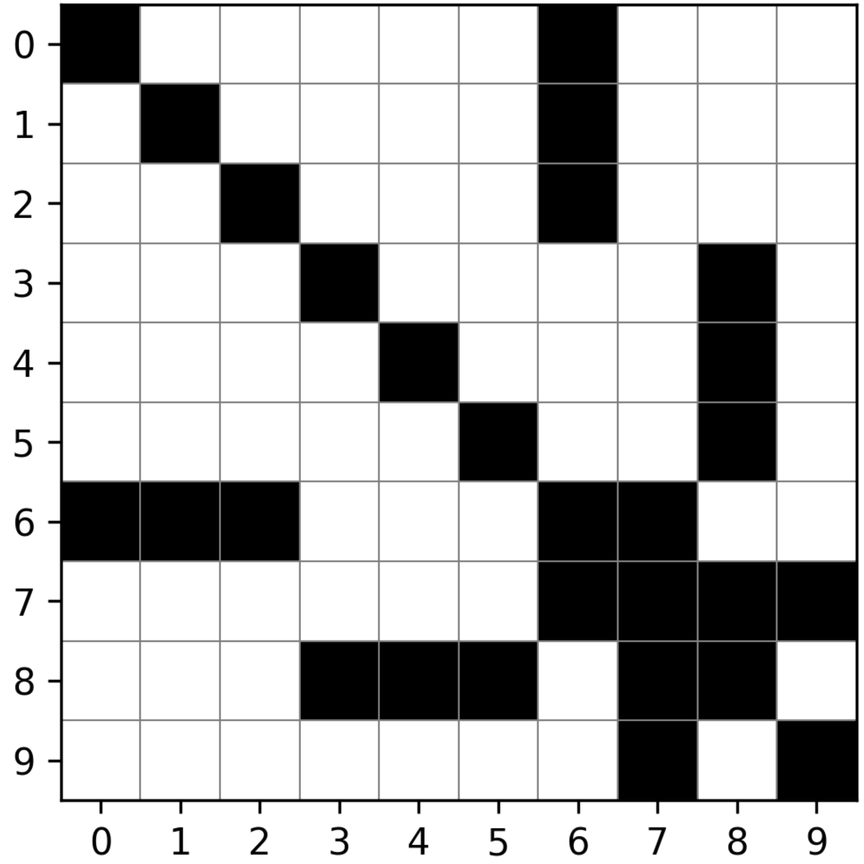
    -> N^2 - d x N zeros





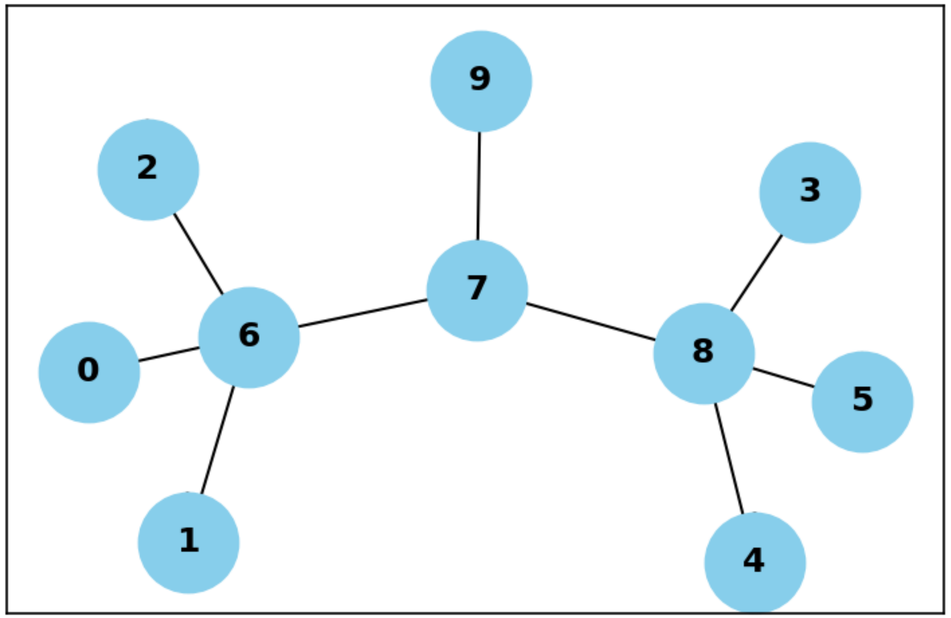
These are equivalent but differently structured





Graph representation

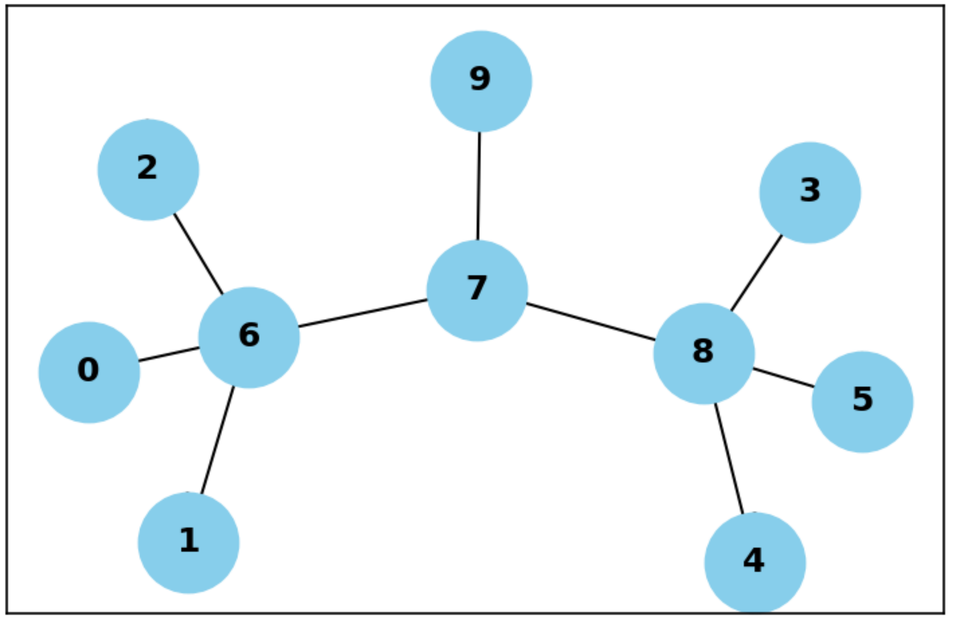
- Nodes = Atoms
- Edges = Bonds
- G = (*E*, *V*) - graph
  - *V* - set of nodes v_i
  - *E* - set of edges $\{e_{ij}\}$ between the nodes



Graph representation

We convert coordinates and atomic numbers $\{\boldsymbol{R}, Z\}$ into graph (E, V)

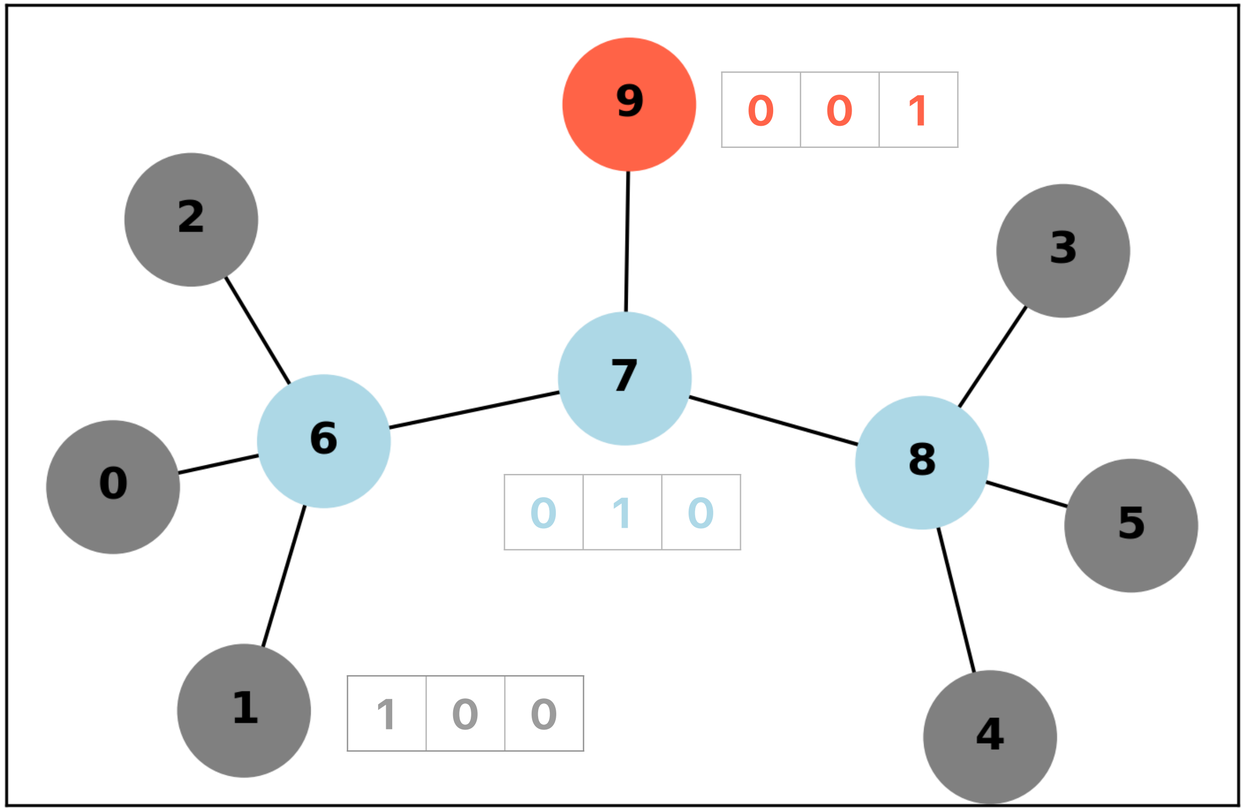
How to distinguish different atoms?



Hydrogen	1	0	0
Carbon	0	1	0
Oxygen	0	0	1

Node encoding

- Naive way - scalar features
  - atomic weight
  - ionic radii
  - number of valence electrons, etc
- Optimal way
  - One-Hot encoding

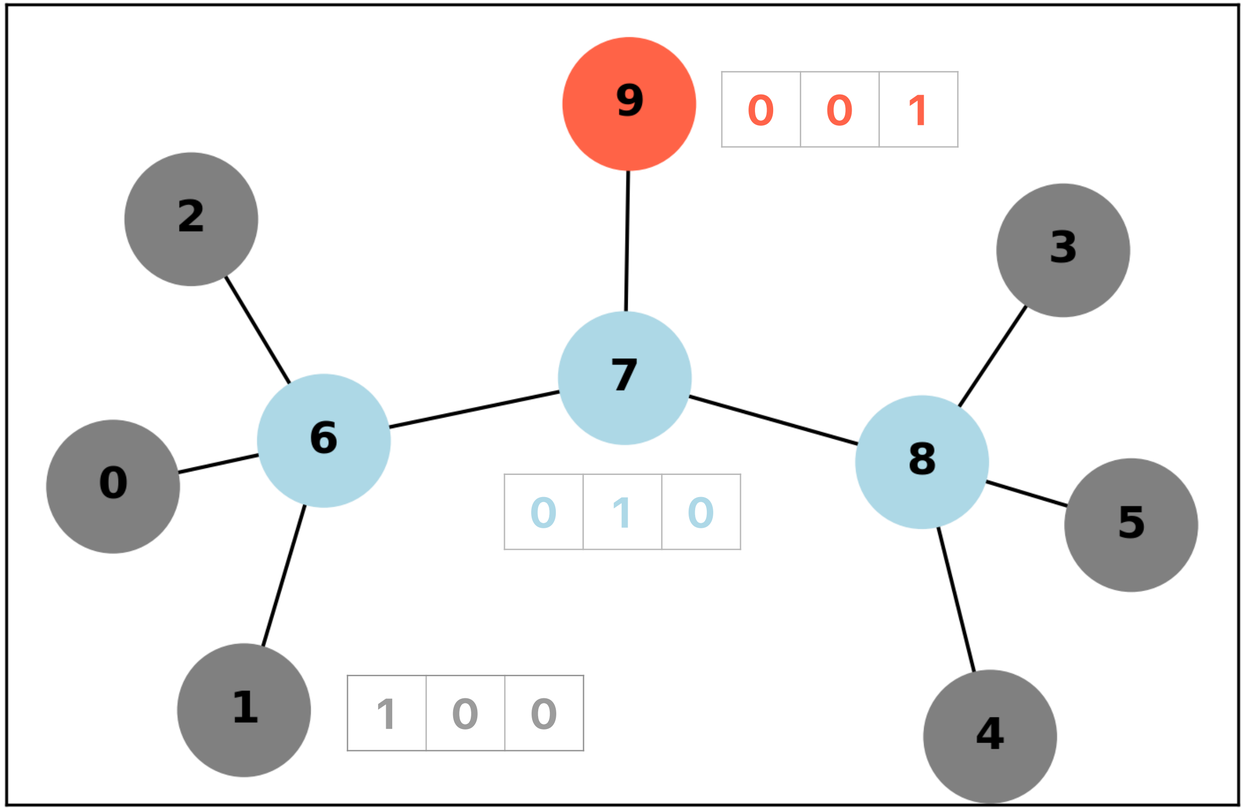


Hydrogen	1	0	0
Carbon	0	1	0
Oxygen	0	0	1

Graph representation

We convert coordinates and atomic numbers $\{\boldsymbol{R}, Z\}$ into graph (E, V)

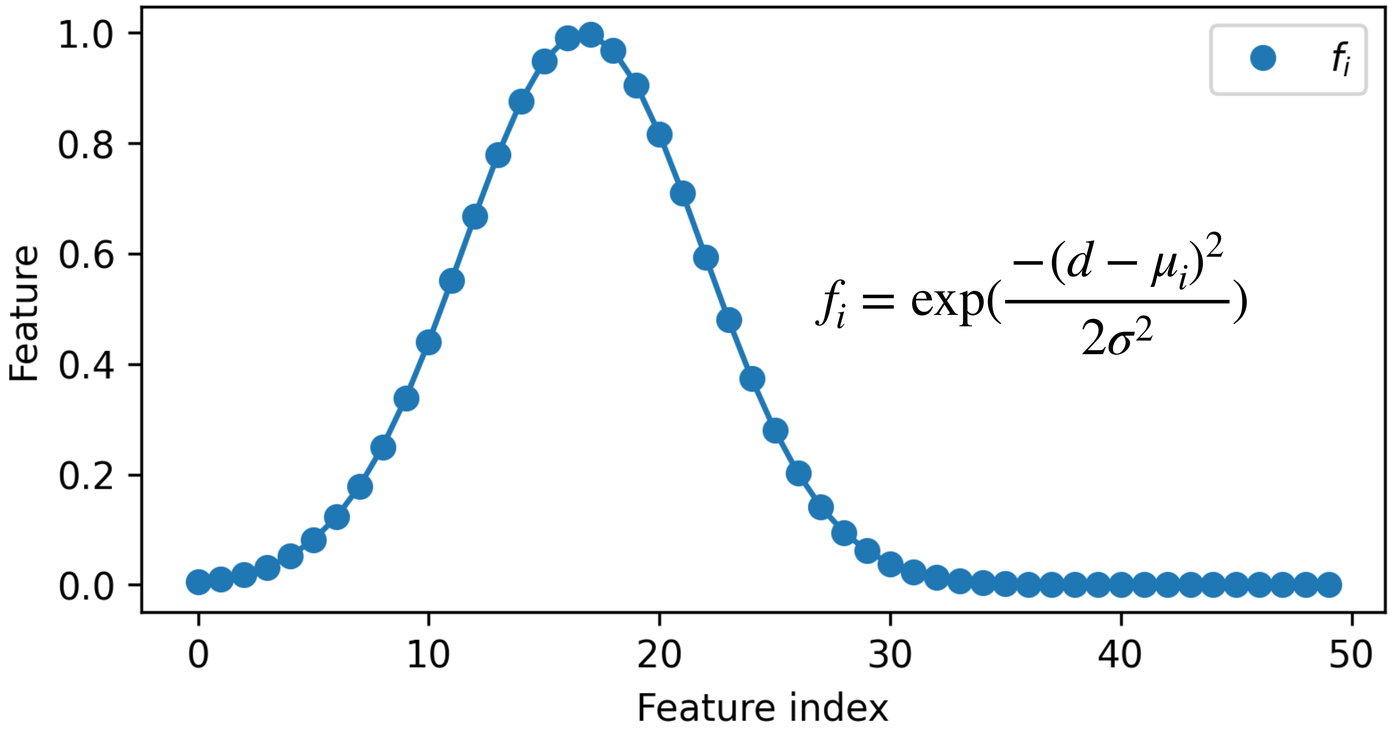
How to distinguish different bonds?



Edge encoding

- Naive way - scalar features
  - Bond length
  - Bond ionicity
- Optimal way
  - Bond distance expansion
    - e.g. gaussian expansion Edge

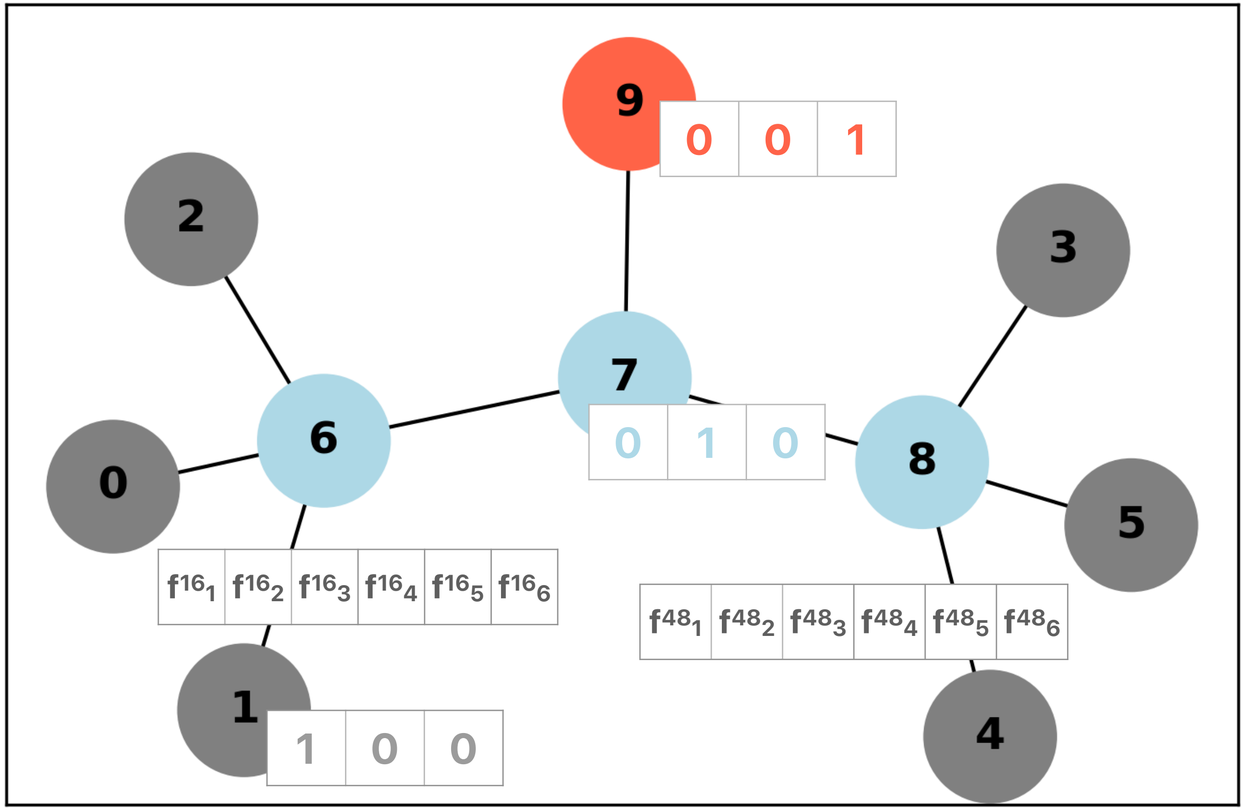
e_{ij} feature = $[f_1^{ij}, \ldots, f_n^{ij}]$



Graph representation

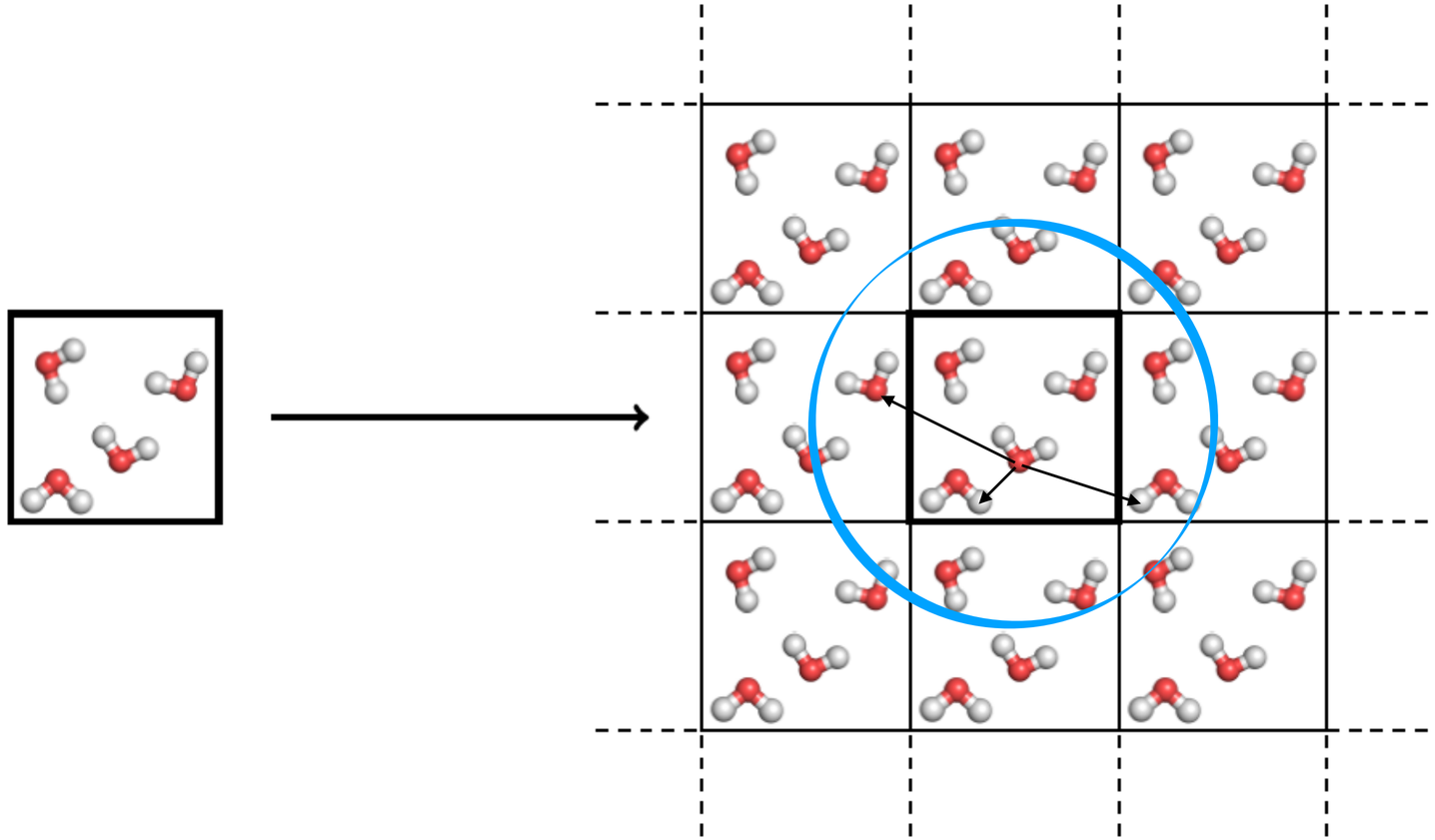
- We convert coordinates and atomic numbers $\{\boldsymbol{R}, Z\}$ into graph (E, V)
- Where each node and edge have their attributes (features)
- Note:
  - There are two edge directions: {i,j} and {j,i}
  - If direction matters, the graph is directed and e_{ij} != e_{ji}

Hydrogen	1	0	0
Carbon	0	1	0
Oxygen	0	0	1



How to deal with PBC in crystals?

- Add buffer layer (i.e. create at least 3x3x3 supercell)
- Collect neighbours for the central unit cell within r_cut
- That's it

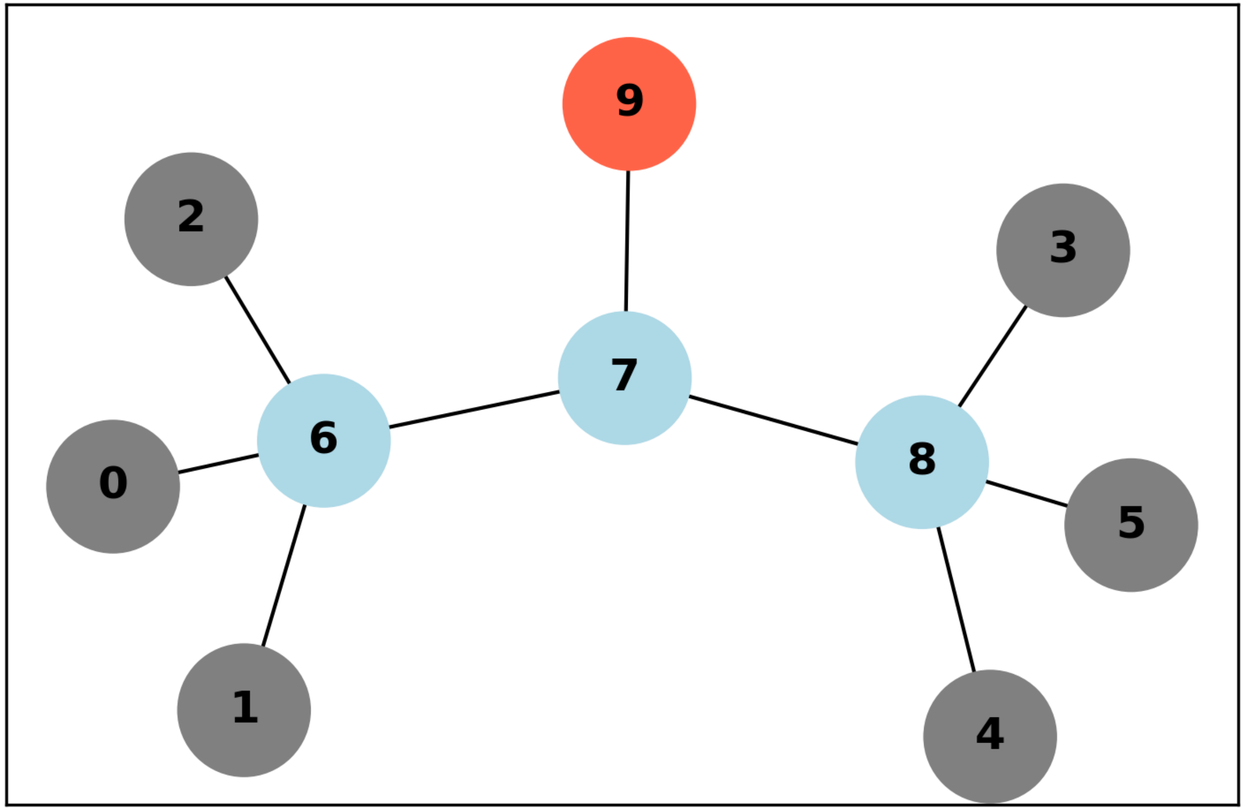


Message passing

0	0	1
0	1	0
1	0	0

Message passing

- We want neighbours to exchange the information
- For that we sent message between them
- And update state of each node



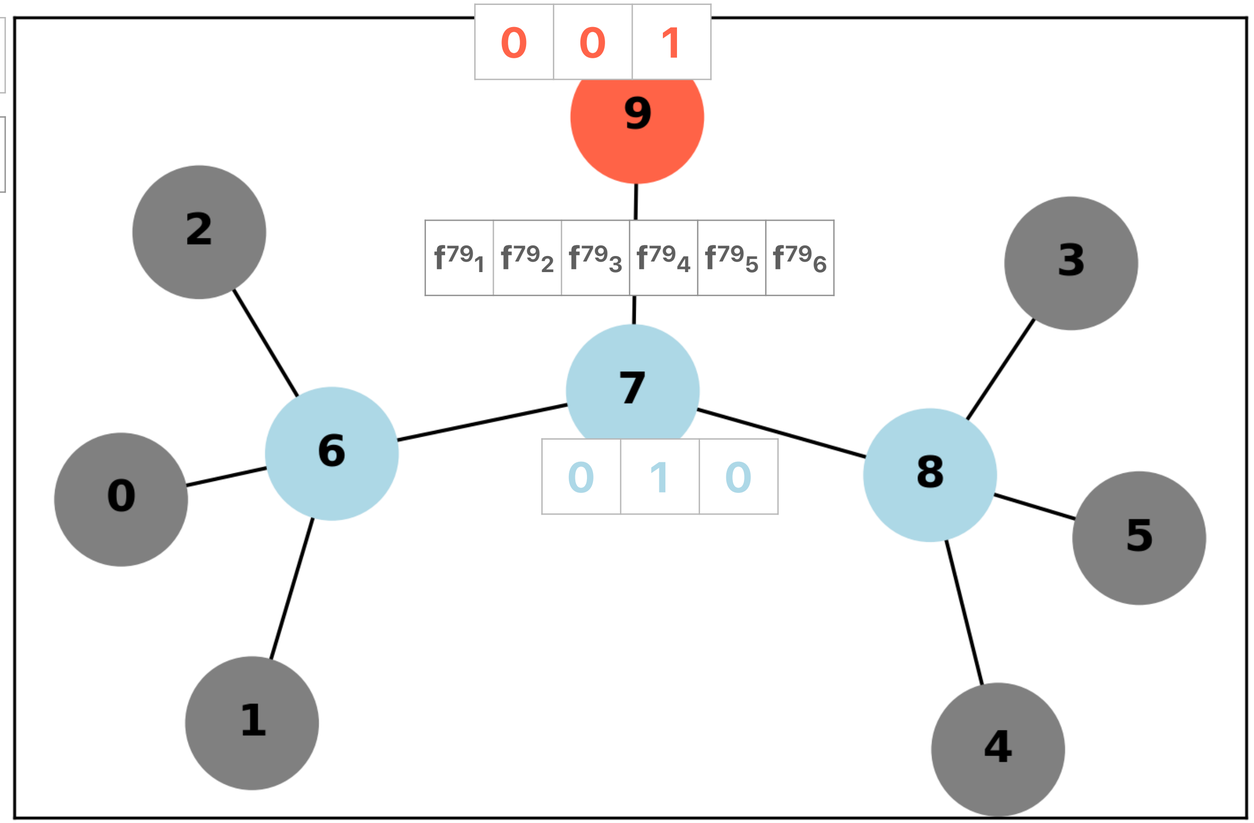
0	0	1
0	1	0
1	0	0

Message from 7th to 9th node

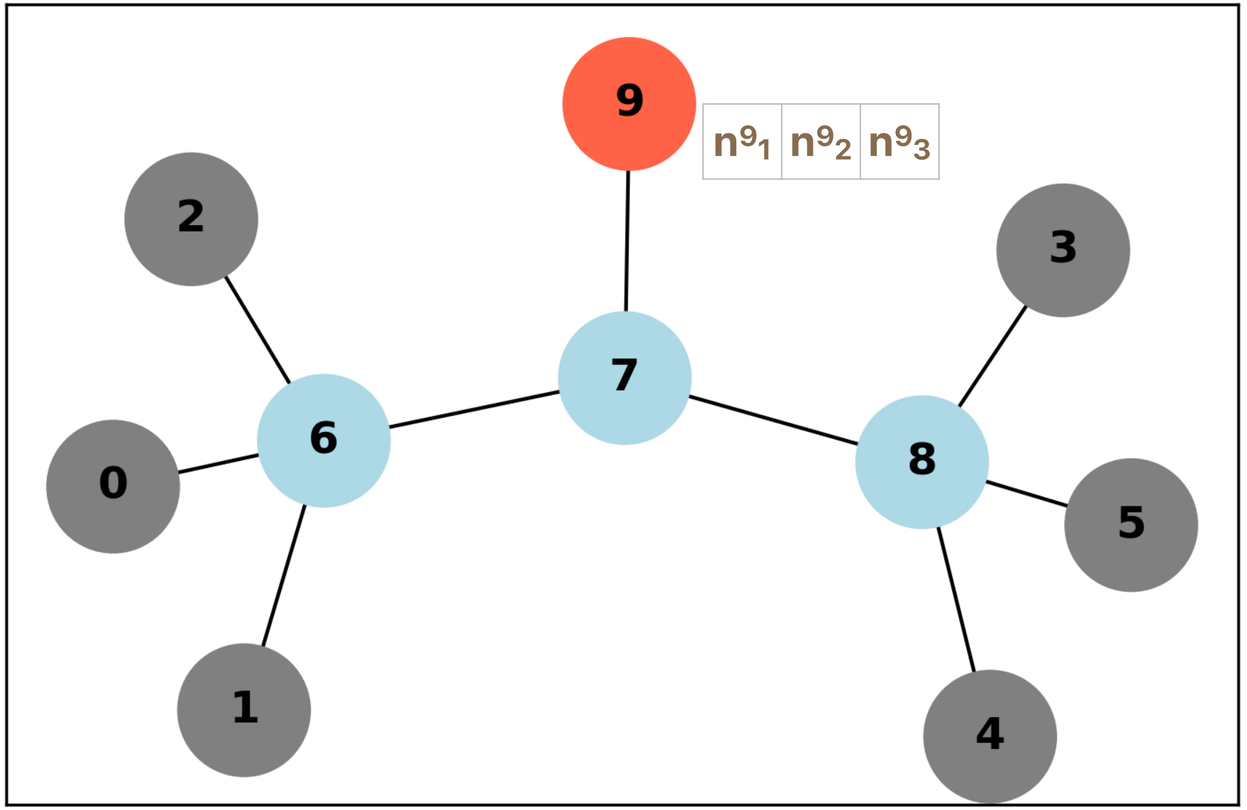
– Concatenation of node and edge features

m =

0	1	0	f^{79}_1	f^{79}_2	f^{79}_3	f^{79}_4	f^{79}_5	f^{79}_6



0	0	1
0	1	0
1	0	0



9th node update

- Input vector: concatenated source, target, and edge features

0	0	1	+= MLP(	0	0	1	0	1	0	f^{79}_1	f^{79}_2	f^{79}_3	f^{79}_4	f^{79}_5	f^{79}_6	)

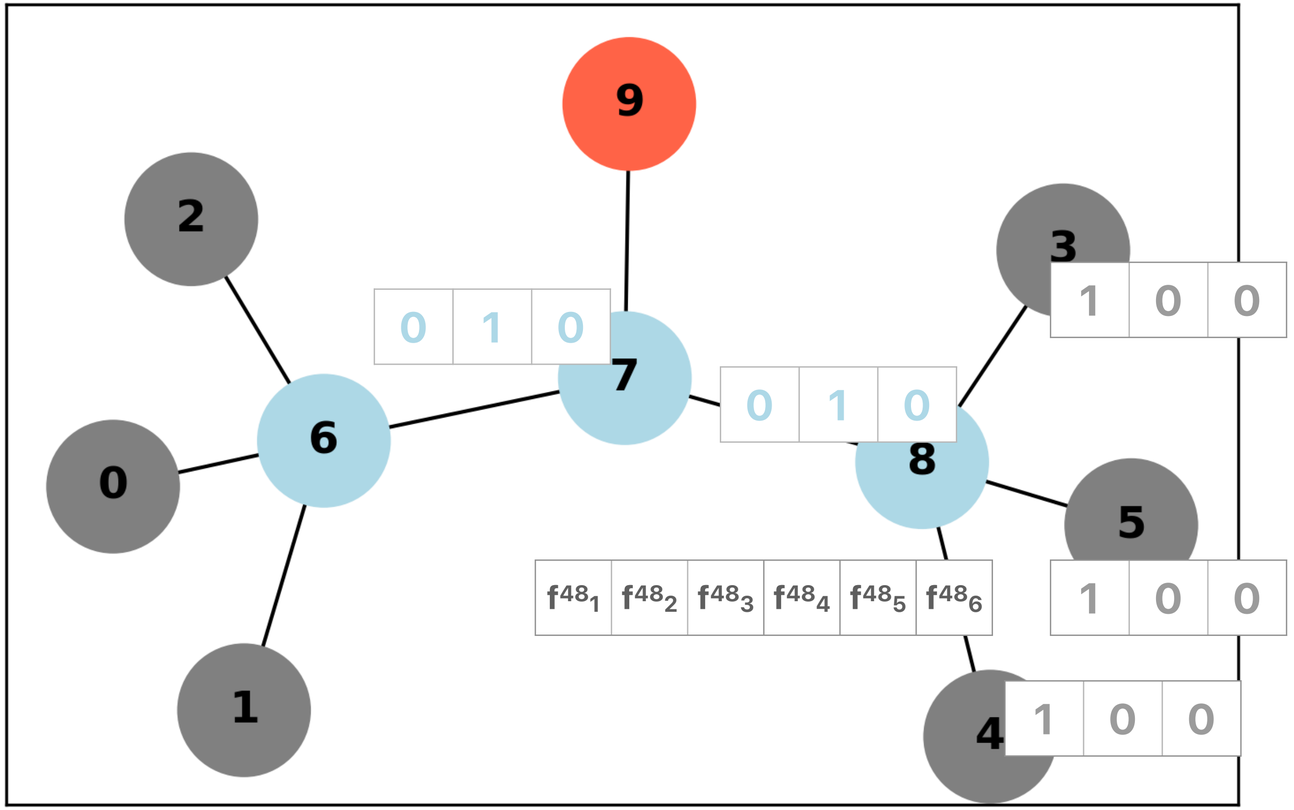
9th node new state

n^9_1	n^9_2	n^9_3

Message from 8th node neighbours

- Concatenation of node and edge features
- Summation (or averaging) of the per-pair messages

0	0	1
0	1	0
1	0	0



m =

1	0	0	f^{48}_1	f^{48}_2	f^{48}_3	f^{48}_4	f^{48}_5	f^{48}_6
+								
1	0	0	f^{58}_1	f^{58}_2	f^{58}_3	f^{58}_4	f^{58}_5	f^{58}_6
+								
1	0	0	f^{38}_1	f^{38}_2	f^{38}_3	f^{38}_4	f^{38}_5	f^{38}_6
+								
0	1	0	f^{78}_1	f^{78}_2	f^{78}_3	f^{78}_4	f^{78}_5	f^{78}_6

Message from 8th node

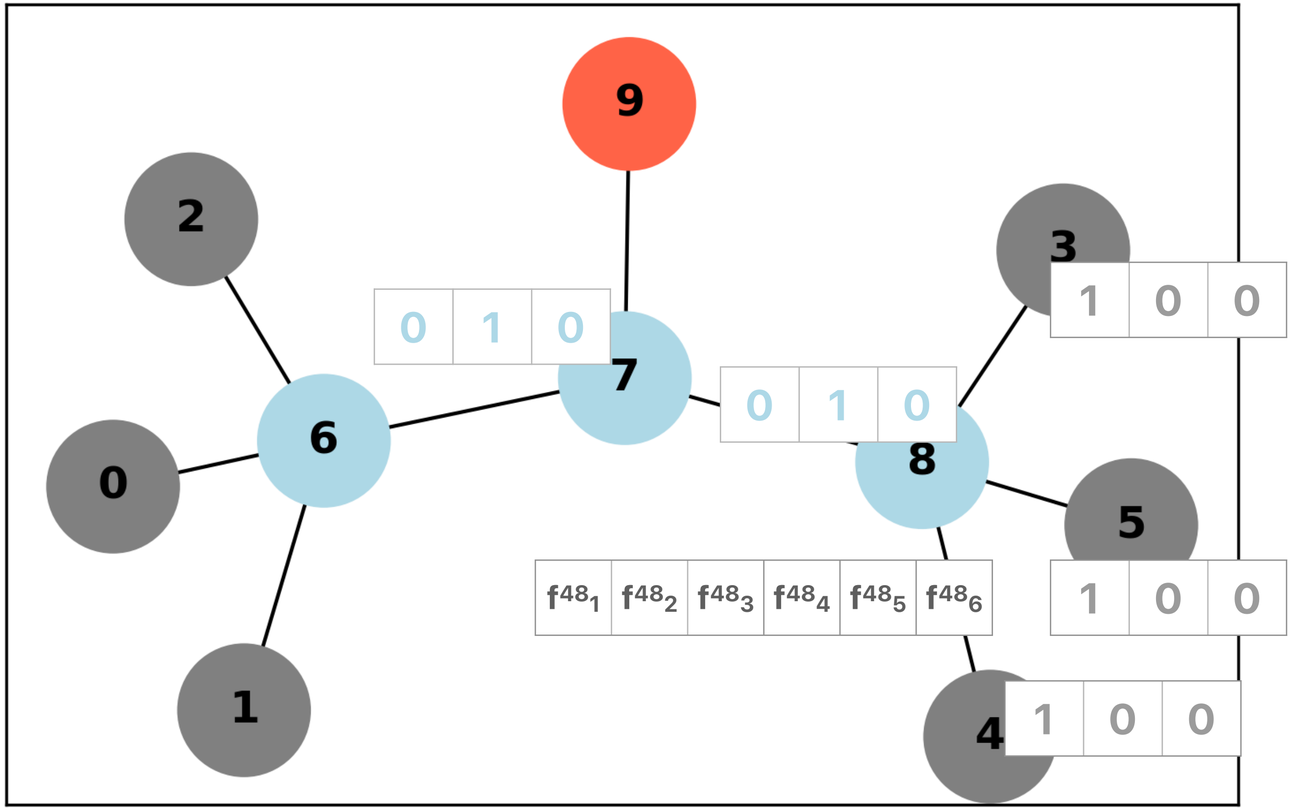
neighbours

- Concatenation of node and edge features
- Summation (or averaging) of the per-pair messages

0	0	1

0	1	0

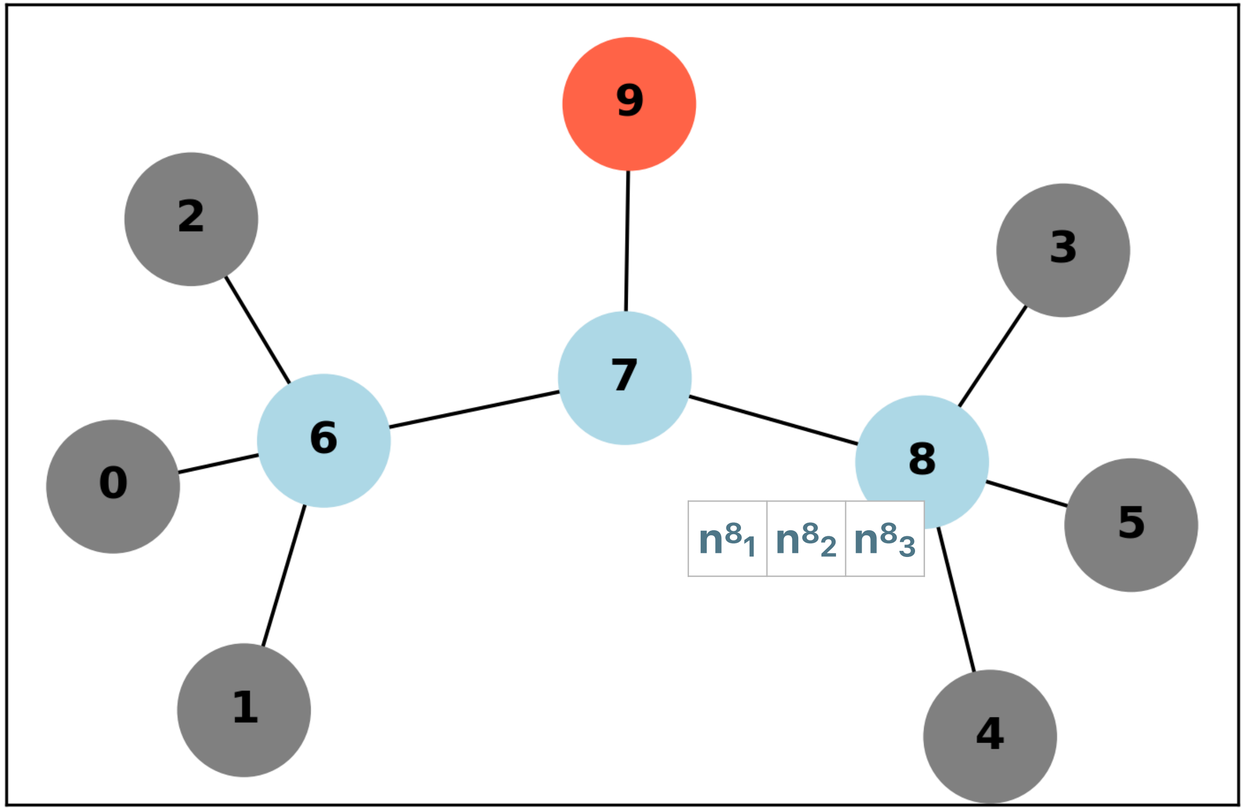
1	0	0



m =

3	1	0	f_1	f_2	f_3	f_4	f_5	f_6

0	0	1
0	1	0
1	0	0



8th node update

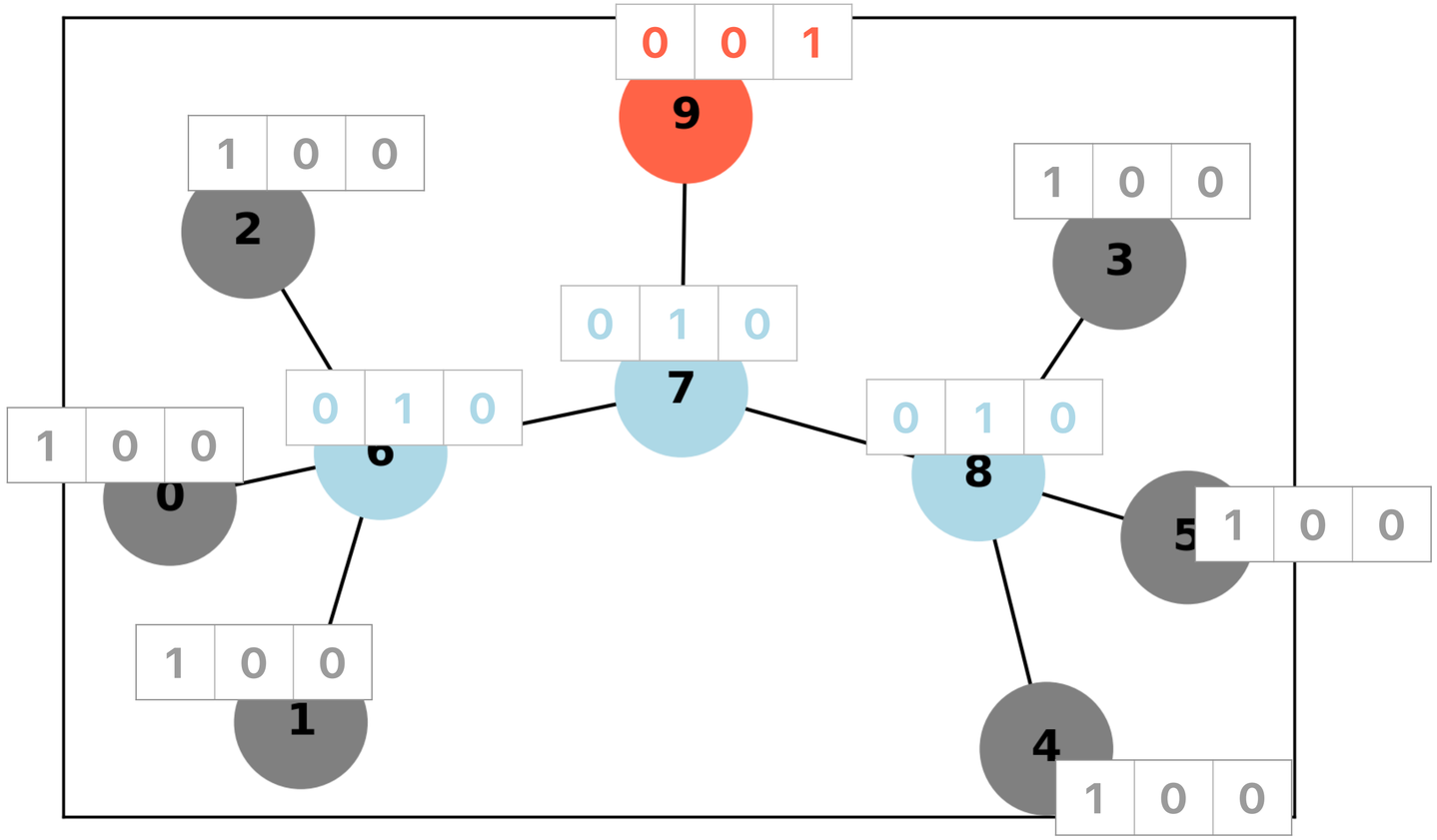
0	1	0	+=	MLP(	3	1	0	0	1	0	f_1	f_2	f_3	f_4	f_5	f_6	)

8th node new state

n^8_1	n^8_2	n^8_3

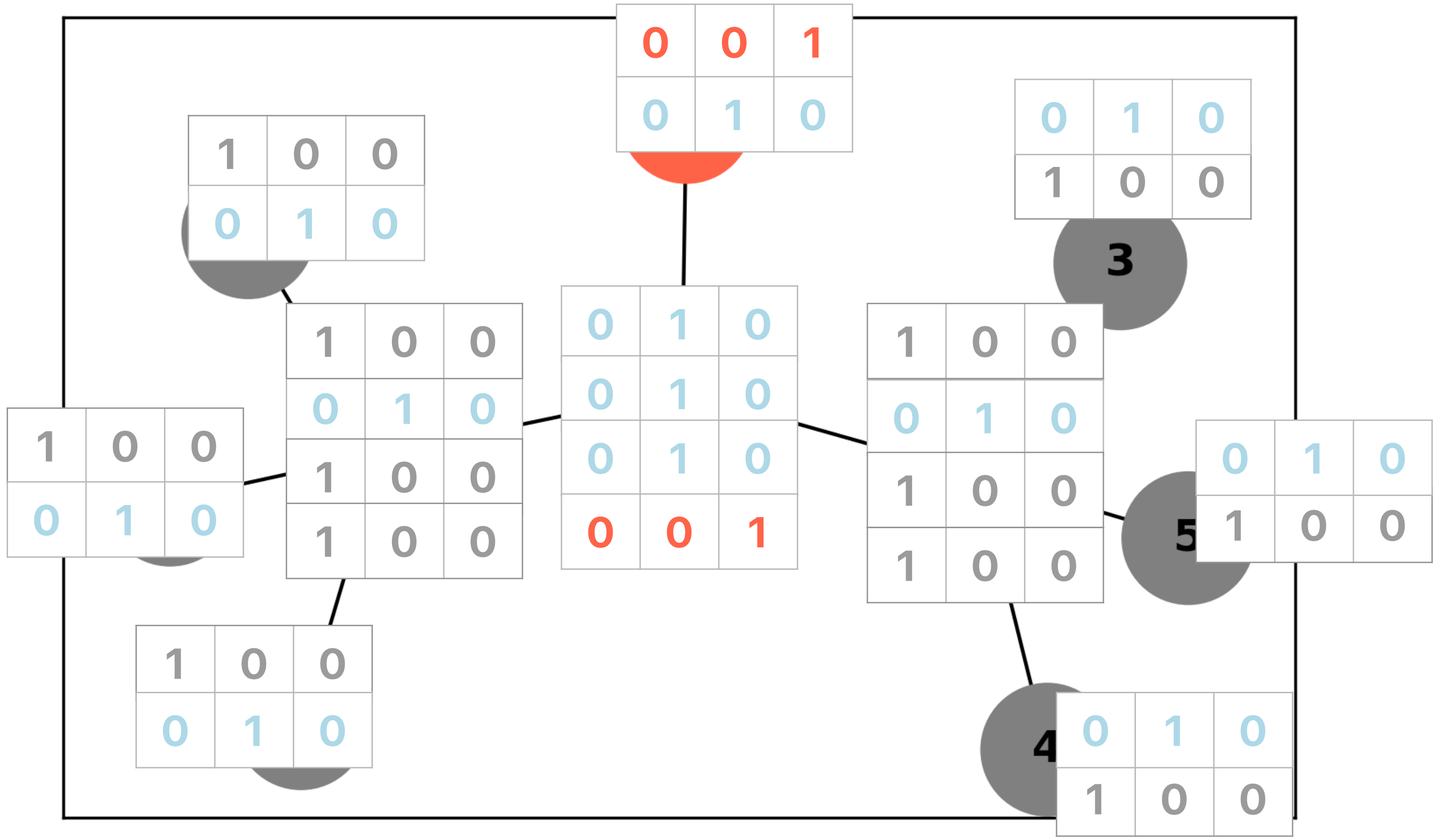
Before massage passing

Note: edge features are not shown

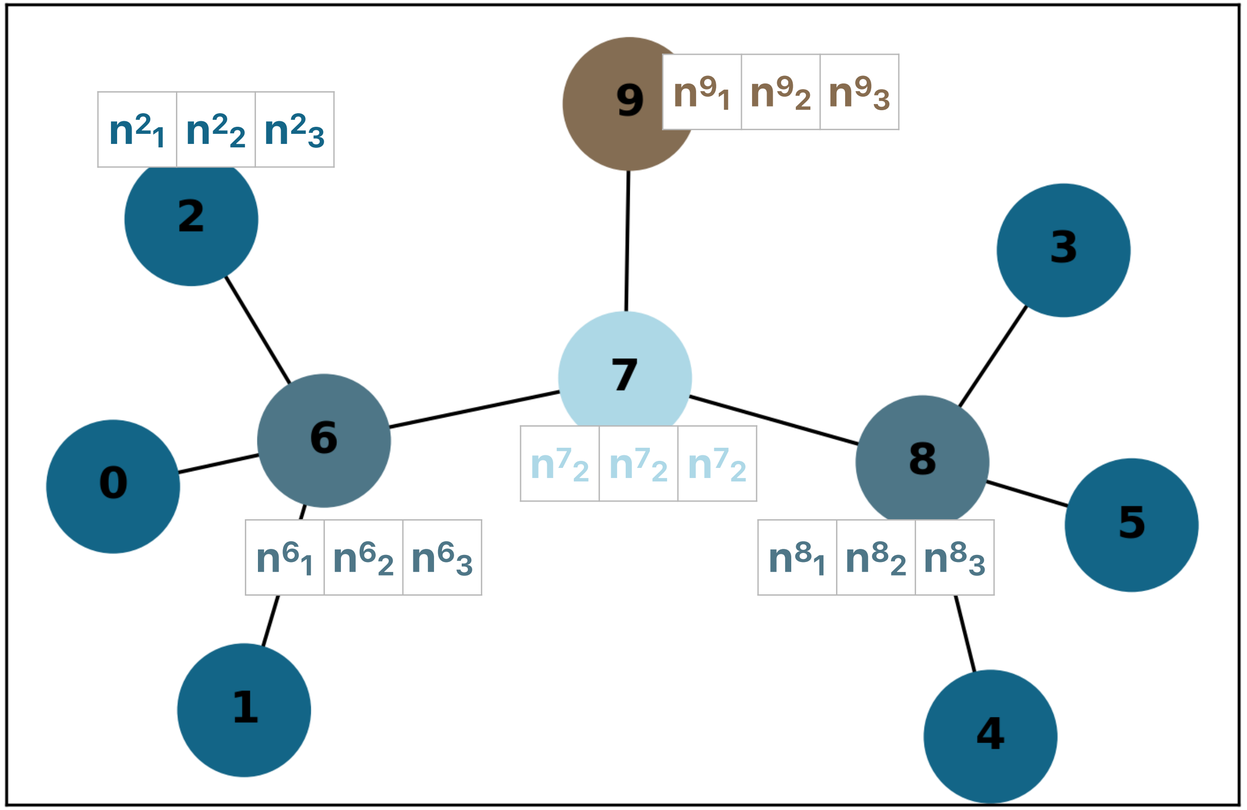


After message passing

Note: edge features are not shown



After message passing

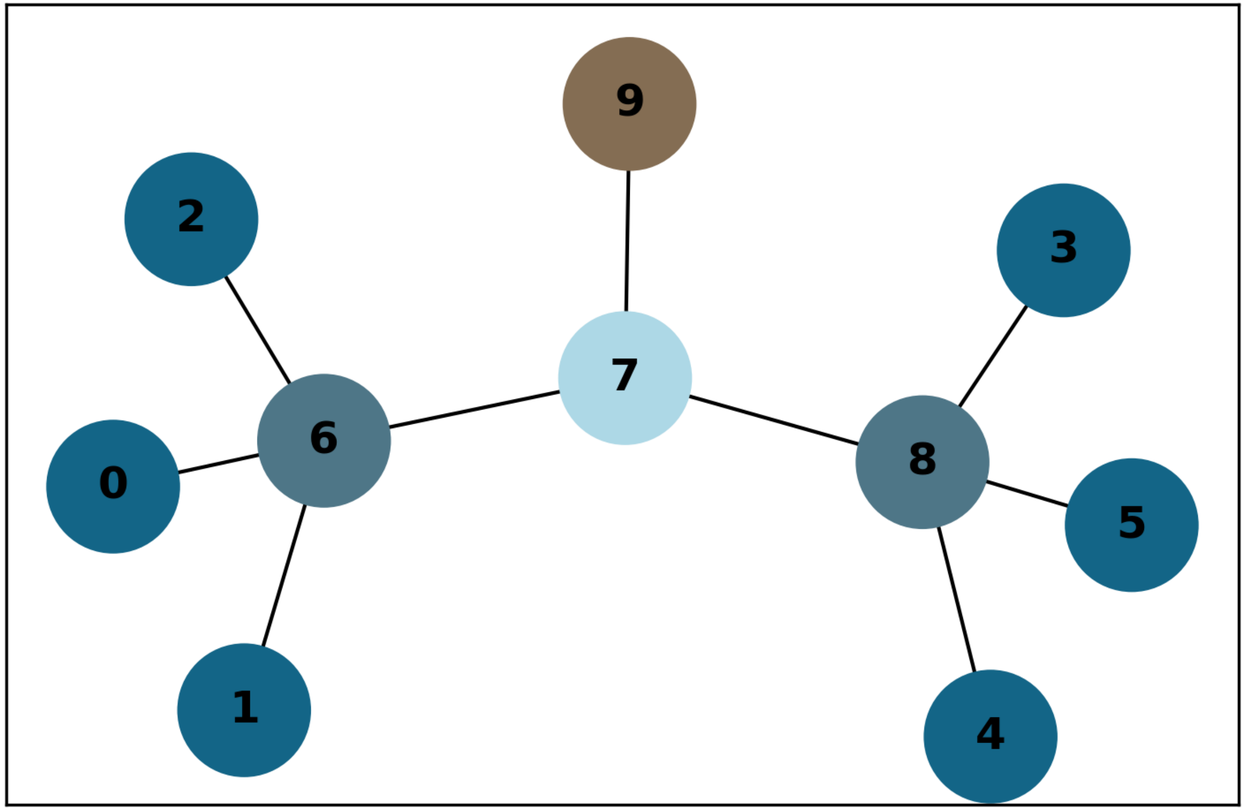


How to get a single value prediction for the entire graph?

Global pooling (readout) layer

- Pool updated node attributes into a single vector
- Using permutation invariant operation (e.g., min, max, mean, sum)

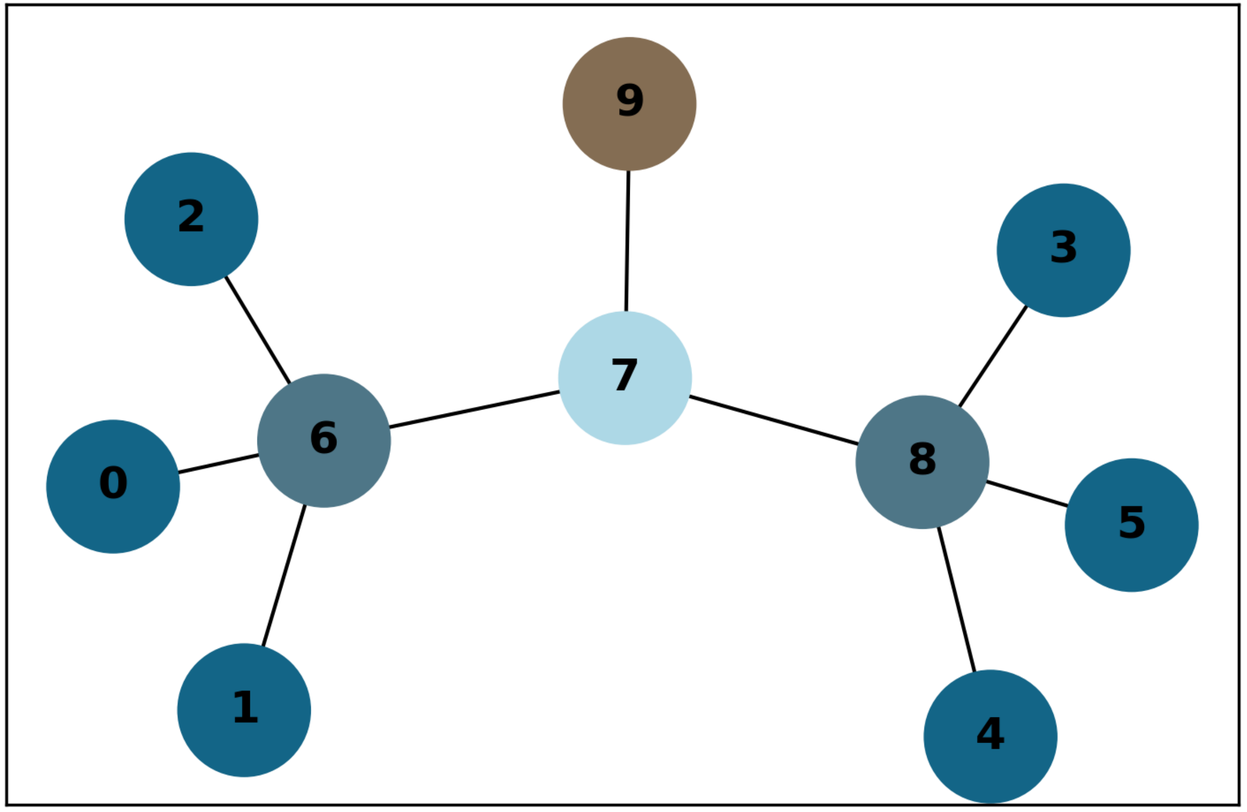
$$h = \frac{1}{N}\sum \mathbf{n_i} = \frac{1}{N} \cdot \begin{matrix} \mathbf{n^1_1} & \mathbf{n^1_2} & \mathbf{n^1_3} \\ + \\ \dots \\ + \\ \mathbf{n^9_1} & \mathbf{n^9_2} & \mathbf{n^9_3} \end{matrix}$$



Global pooling (readout) layer

- Pool updated node attributes into a single vector
- Using permutation invariant operation (e.g., min, max, mean, sum)

$$h = \frac{1}{N}\sum \mathbf{n_i} = \boxed{h_1}\boxed{h_2}\boxed{h_3}$$



Permutation invariance?

- Change in order of atoms should not effect the model prediction

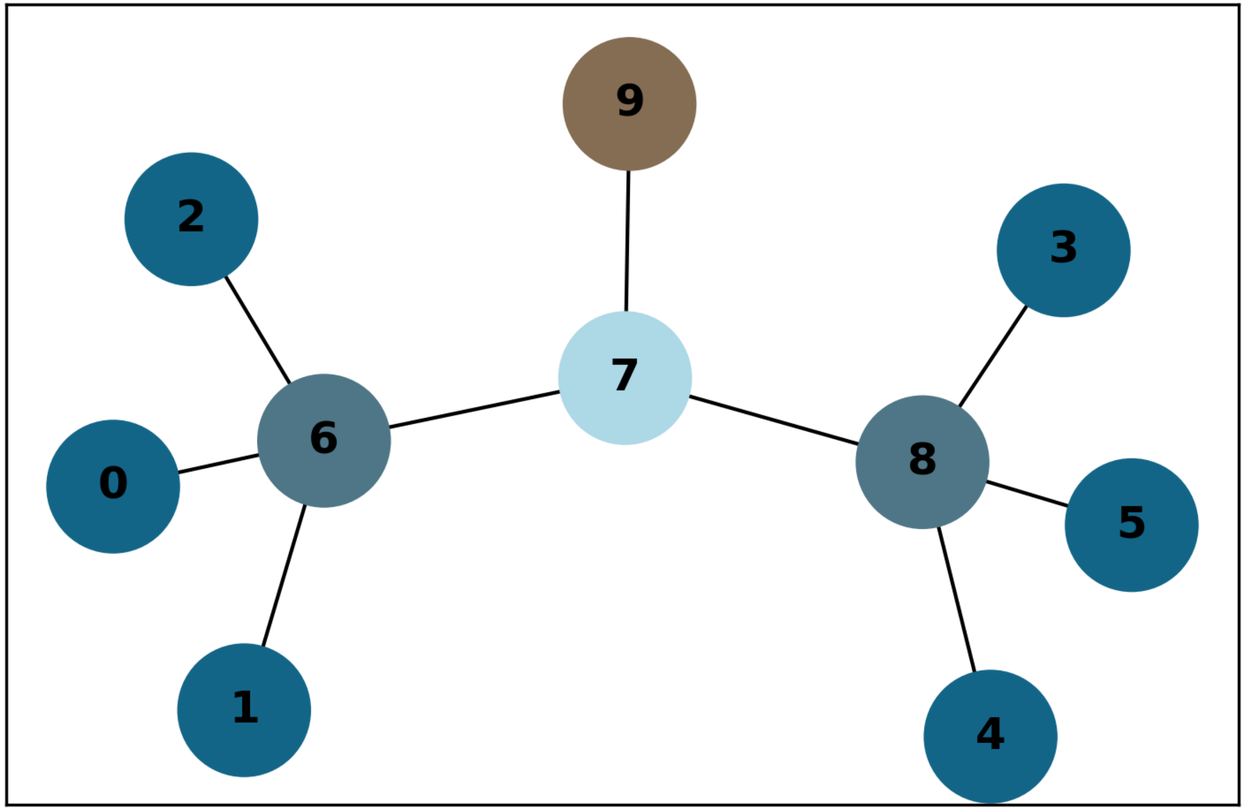
We also want translation and rotation invariance

- Rotating or translating atomic position should not effect the model prediction

$$h = \frac{1}{N}\sum \mathbf{n_i} = \boxed{\mathbf{h_1}}\boxed{\mathbf{h_2}}\boxed{\mathbf{h_3}}$$

Final prediction

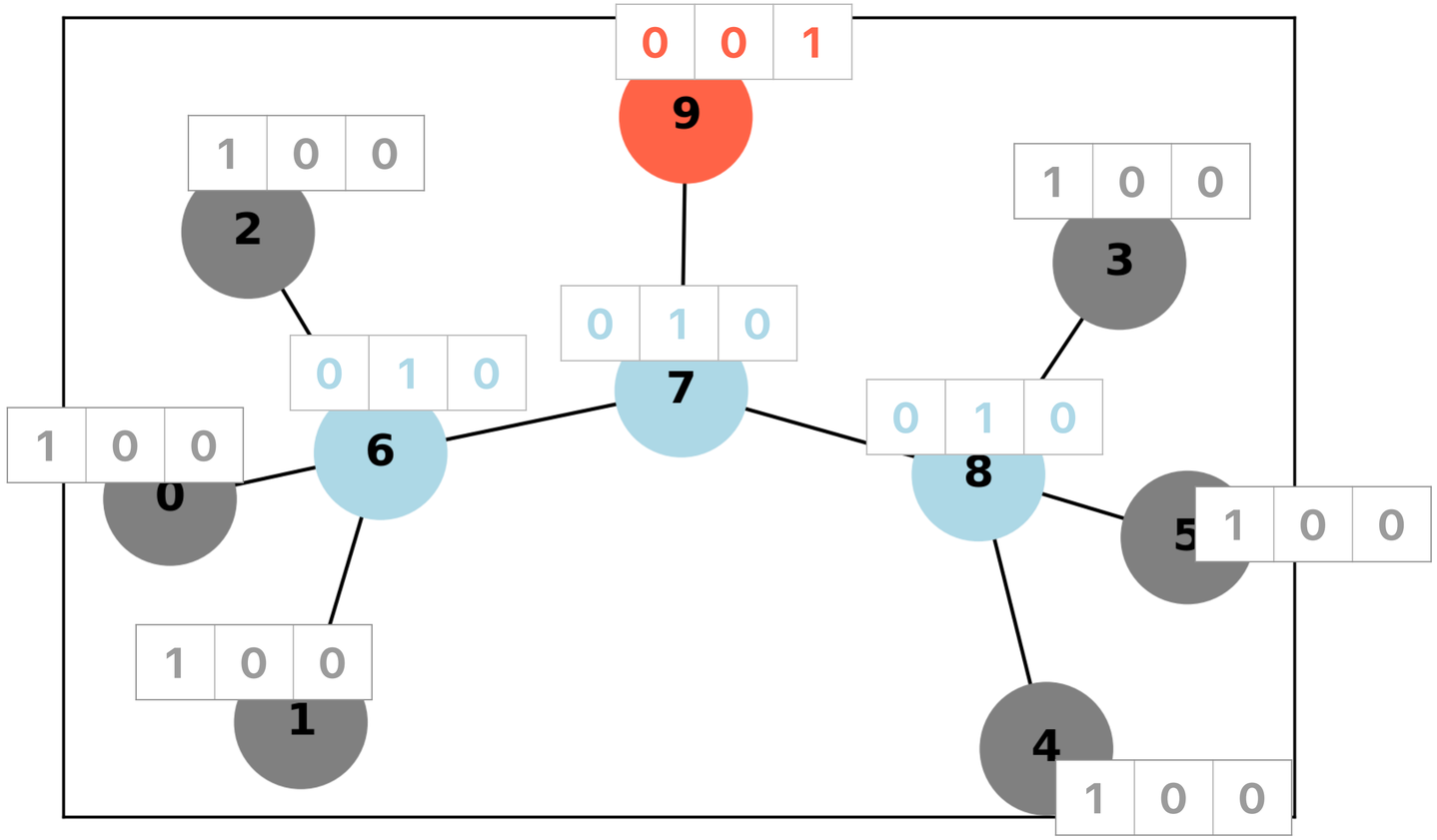
- MLP($\boxed{\mathbf{h_1}}\boxed{\mathbf{h_2}}\boxed{\mathbf{h_3}}$)
  - e.g., MLP(3 -> 16 -> 1)
  - 3 - input vector size
  - 16 - hidden layer size
  - 1 - output size



Perceptive field

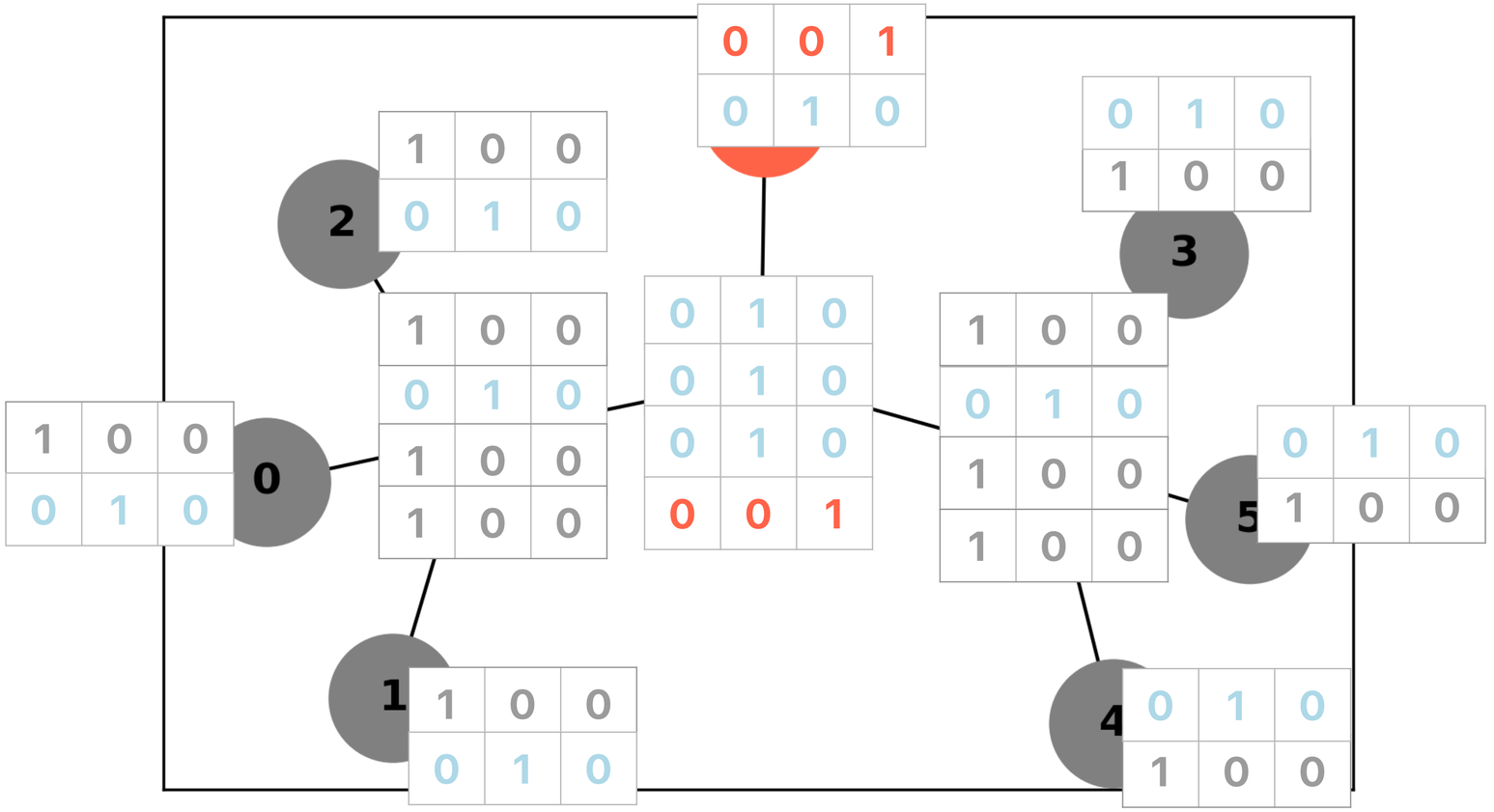
In the example

- We updated node attributes only ones
- In principle we can do it unlimited number of times



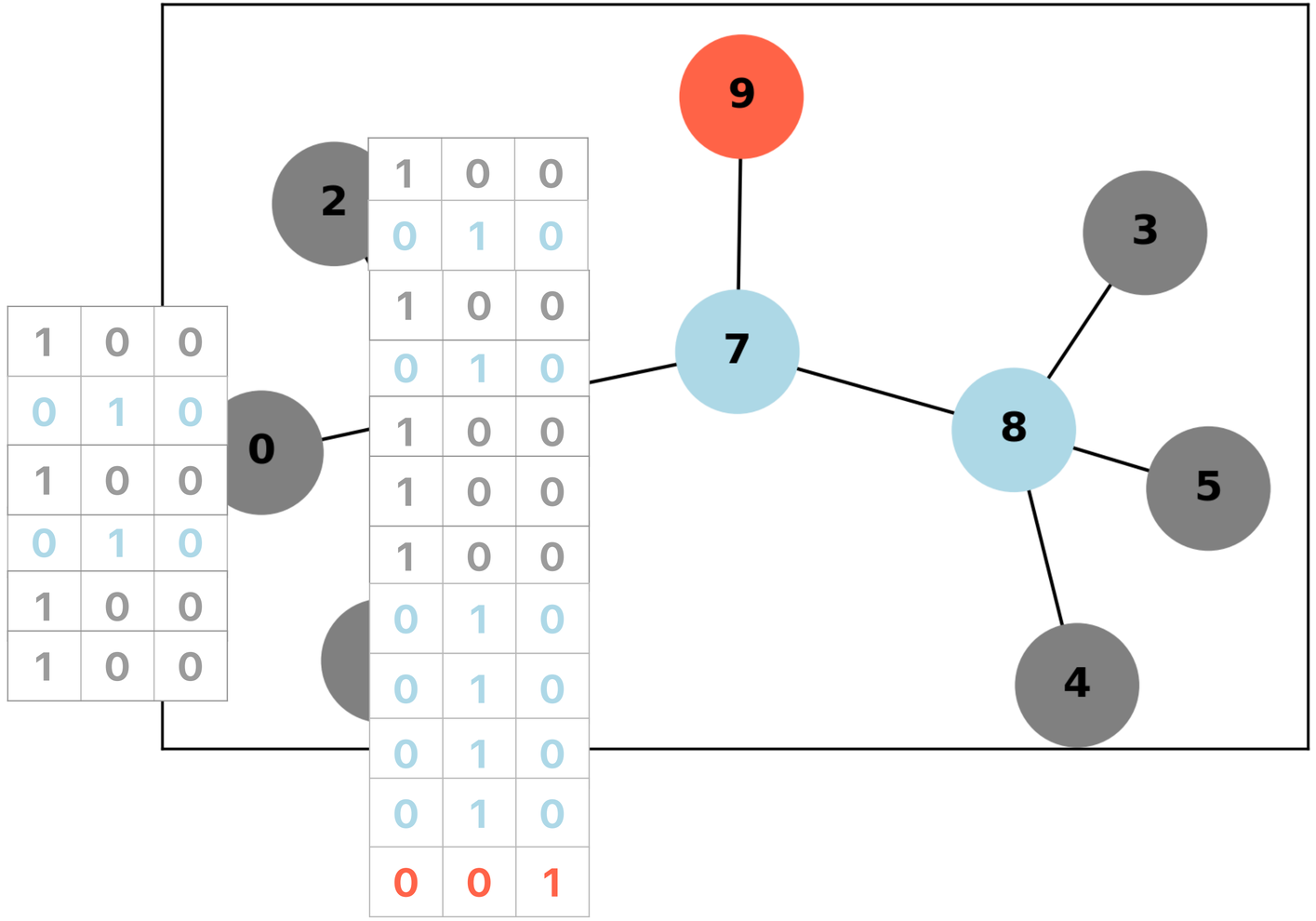
Consider 0th node

- After the first update
  - It knows about 6th node



Consider 0th node

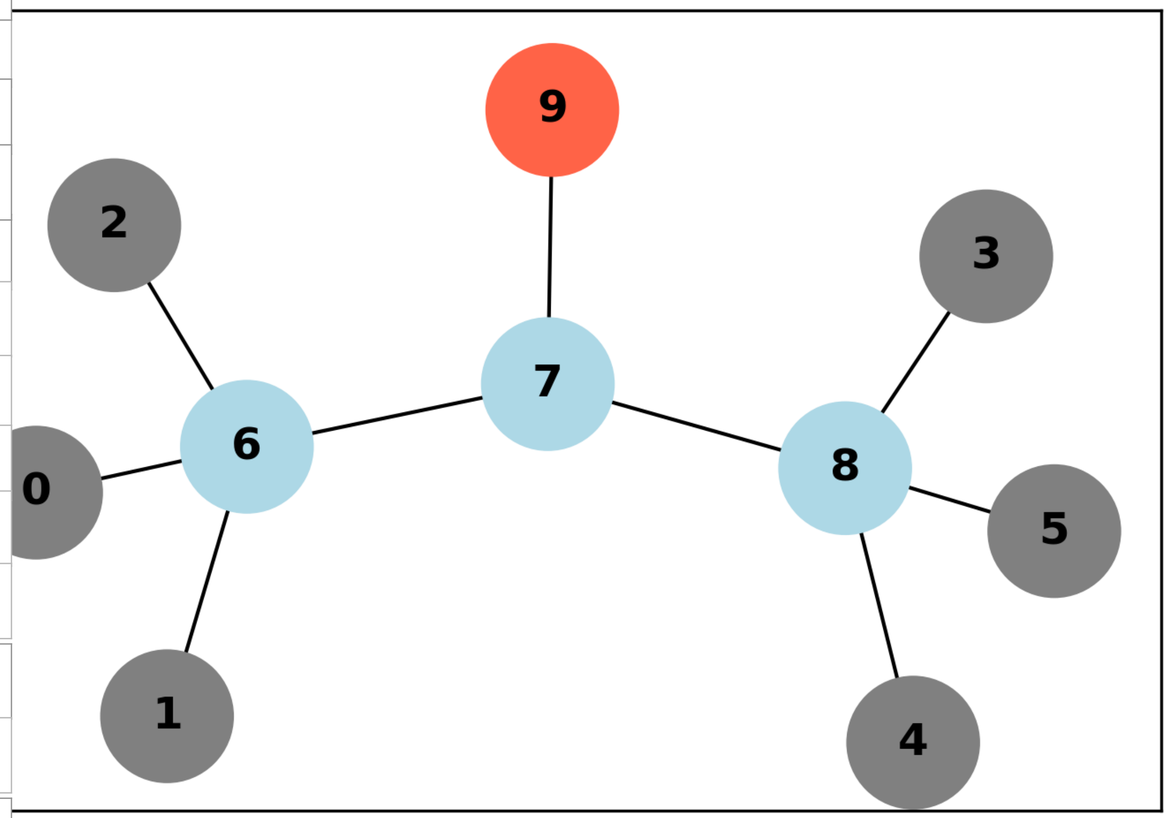
- After the first update
  - It knows about 6th node
- After the second update
  - It knows about 1th and 2th nodes



Consider 0th node

- After the first update
  - It knows about 6th node
- After the second update
  - It knows about 6th, 1th and 2th nodes
- After the third update
  - It knows about 6th, 1th, 2th, 7th, 8th, and 9th nodes

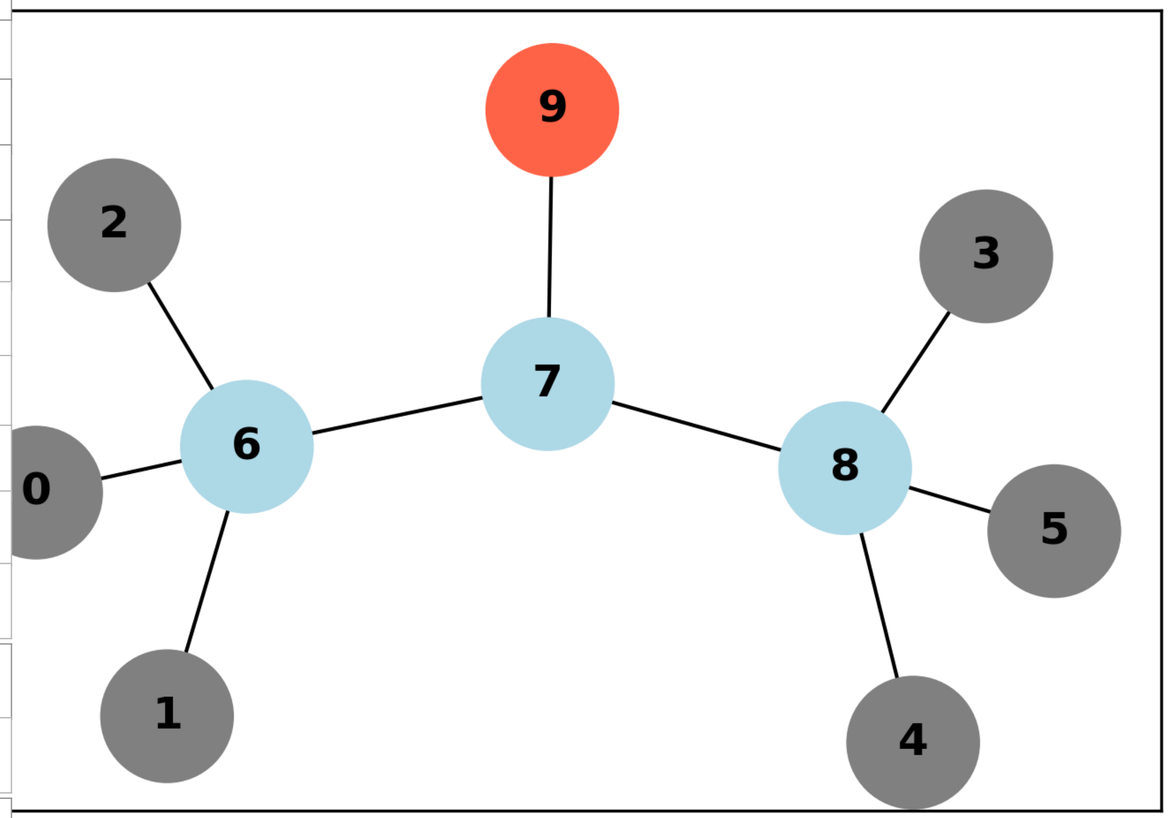
1	0	0
0	1	0
1	0	0
0	1	0
1	0	0
1	0	0
1	0	0
0	1	0
0	1	0
0	1	0
0	1	0
0	0	1
1	0	0
0	1	0
1	0	0
0	1	0
1	0	0
1	0	0



Consider 0th node

- With each message-passing step, a node's perceptive field increases
- First NN -> NN of first NN -> and so on

1	0	0
0	1	0
1	0	0
0	1	0
1	0	0
1	0	0
1	0	0
0	1	0
0	1	0
0	1	0
0	1	0
0	0	1
1	0	0
0	1	0
1	0	0
0	1	0
1	0	0
1	0	0



GNNs challenges

- Oversmoothing - many message passing layers
  - All nodes converge to the same average representation
  - The GNN can no longer distinguish local structural roles
- Expressivity limit - different structures may have the same graph
  - Graphs don’t about about angles

Overcoming GNNs challenges

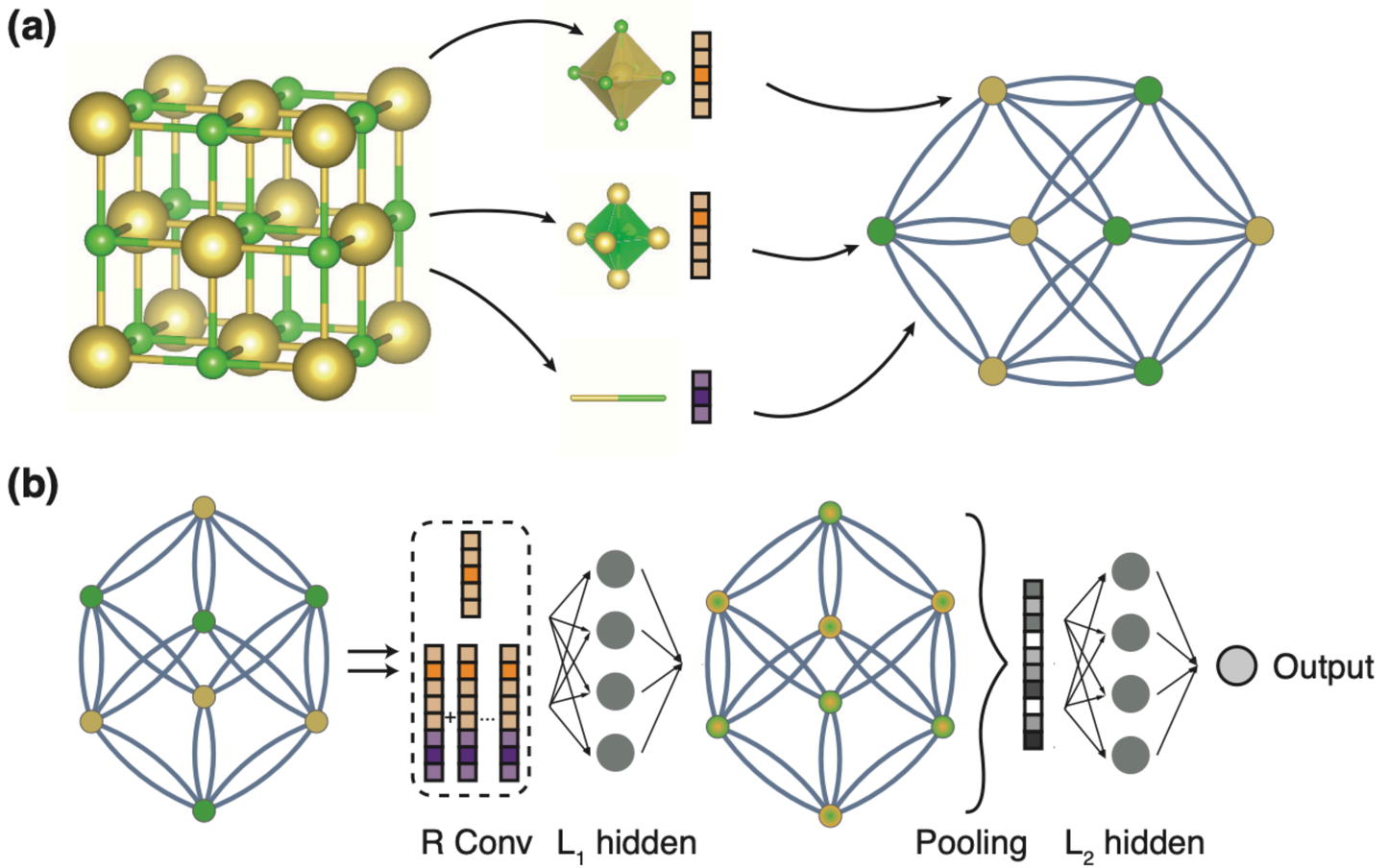
- Oversmoothing - use 2-4 message passing layers
- Expressivity limit - introduce higher order interactions (e.g., three-body terms)

Graph batching

Graphs cannot be stacked similarly to tabular data

- Several graphs are combined in a giant graph with multiple isolated components
- Since graphs are disconnected, message passing works as intended
- During post-processing, the graph batch index is used to aggregate information from nodes that belong to individual graphs

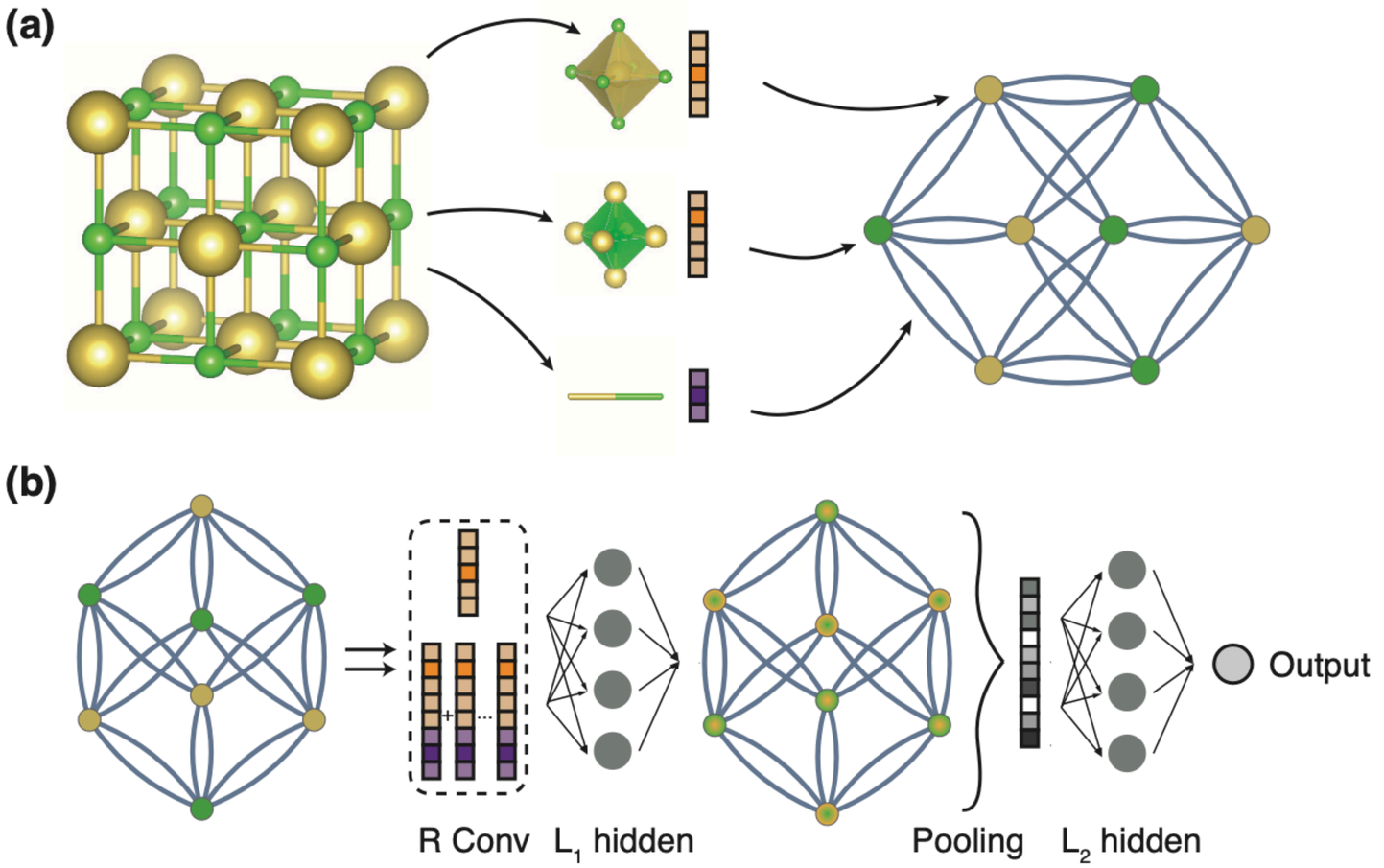
Example: Crystal graph convolutional neural network



Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties

Pipeline

- Convert structure to graph
- Update node states (convolution)
- Apply pooling
- Apply post-processing layer (regression of classification)



Pipeline

- Convert structure to graph
- Update node states (convolution)
- Apply pooling
- Apply post-processing layer (regression of classification)

$$\boldsymbol{v}_i^{(t+1)} = \text{Conv}\left(\boldsymbol{v}_i^{(t)}, \boldsymbol{v}_j^{(t)}, \boldsymbol{u}_{(i,j)_k}\right), (i,j)_k \in \mathcal{G}$$

$$\boldsymbol{v_c} = \text{Pool}\left(\boldsymbol{v}_0^{(0)}, \boldsymbol{v}_1^{(0)}, ..., \boldsymbol{v}_N^{(0)}, ..., \boldsymbol{v}_N^{(R)}\right)$$

Pipeline

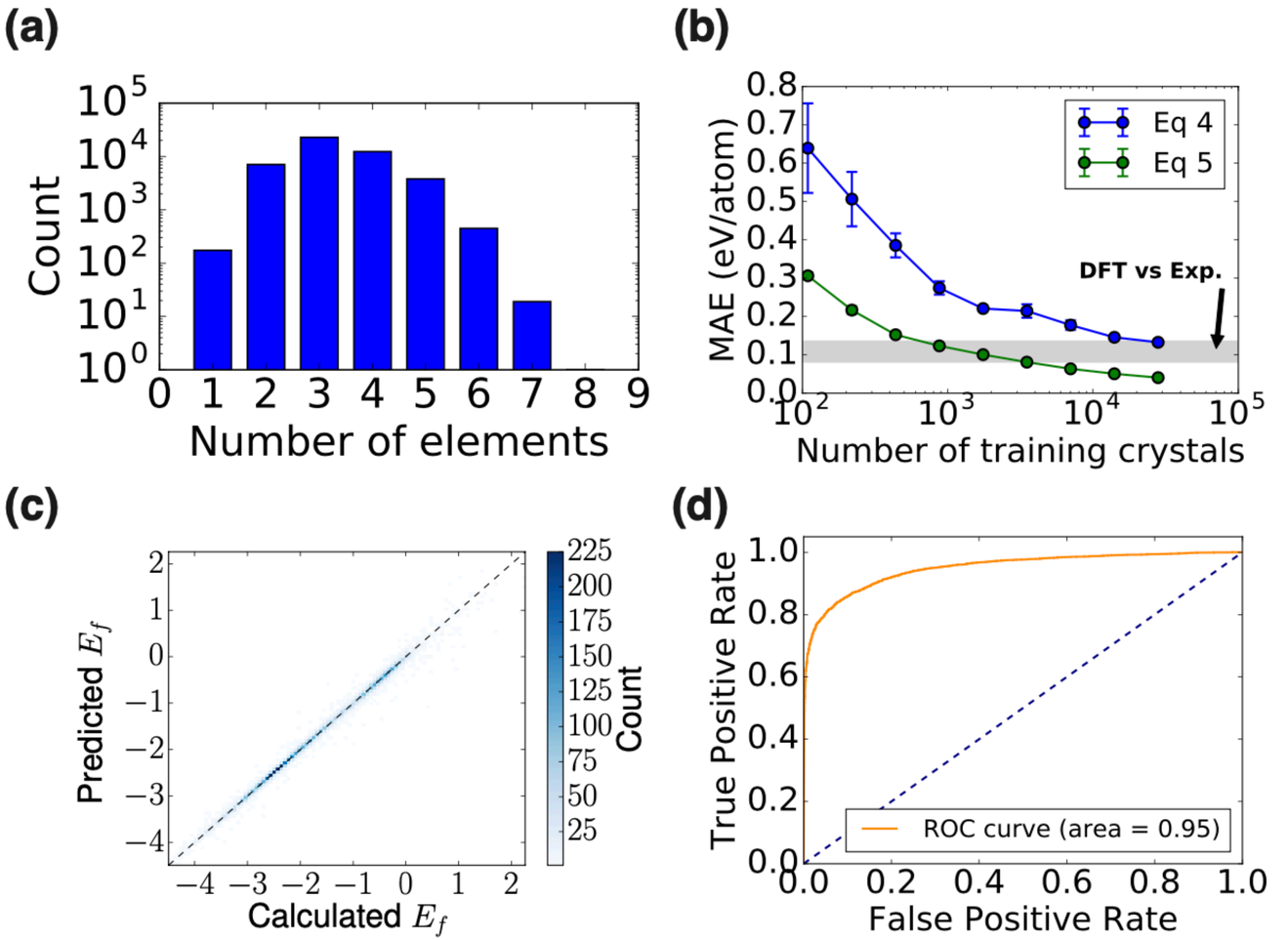
- Convert structure to graph
- Update node states
- Apply pooling
- Apply post-processing layer (regression of classification)

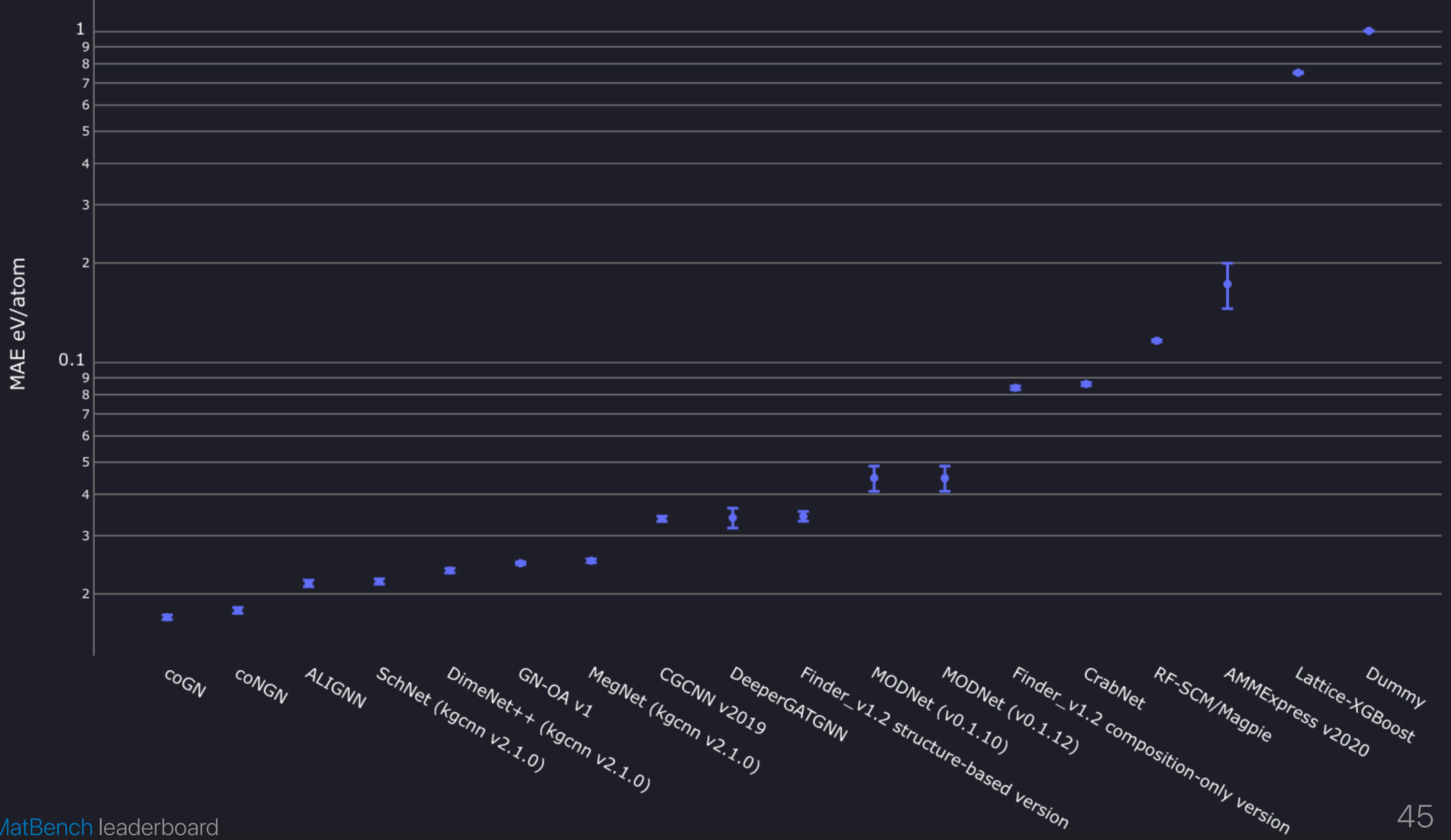
$$\boldsymbol{z}_{(i,j)_k}^{(t)} = \boldsymbol{v}_i^{(t)} \oplus \boldsymbol{v}_j^{(t)} \oplus \boldsymbol{u}_{(i,j)_k}$$

$$\boldsymbol{v}_i^{(t+1)} = \boldsymbol{v}_i^{(t)} + \sum_{j,k} \sigma(\boldsymbol{z}_{(i,j)_k}^{(t)} \boldsymbol{W}_f^{(t)} + \boldsymbol{b}_f^{(t)}) \odot g(\boldsymbol{z}_{(i,j)_k}^{(t)} \boldsymbol{W}_s^{(t)} + \boldsymbol{b}_s^{(t)})$$

$$\boldsymbol{v}_c = \text{Pool}\left(\boldsymbol{v}_0^{(0)}, \boldsymbol{v}_1^{(0)}, ..., \boldsymbol{v}_N^{(0)}, ..., \boldsymbol{v}_N^{(R)}\right)$$

Property	# of train data	Unit	$\text{MAE}_{\text{model}}$	MAE_{DFT}
Formation energy	28046	eV/atom	0.039	0.081–0.136[18]
Absolute energy	28046	eV/atom	0.072	–
Band gap	16458	eV	0.388	0.6[23]
Fermi energy	28046	eV	0.363	–
Bulk moduli	2041	log(GPa)	0.054	0.050[24]
Shear moduli	2041	log(GPa)	0.087	0.069[24]
Poisson ratio	2041	–	0.030	–





MatBench leaderboard

Take home message

- Graph is an intuitive representation of an atomistic structure
- GNNs is a powerful tool for predicting materials properties
- Be aware of GNNs limitations