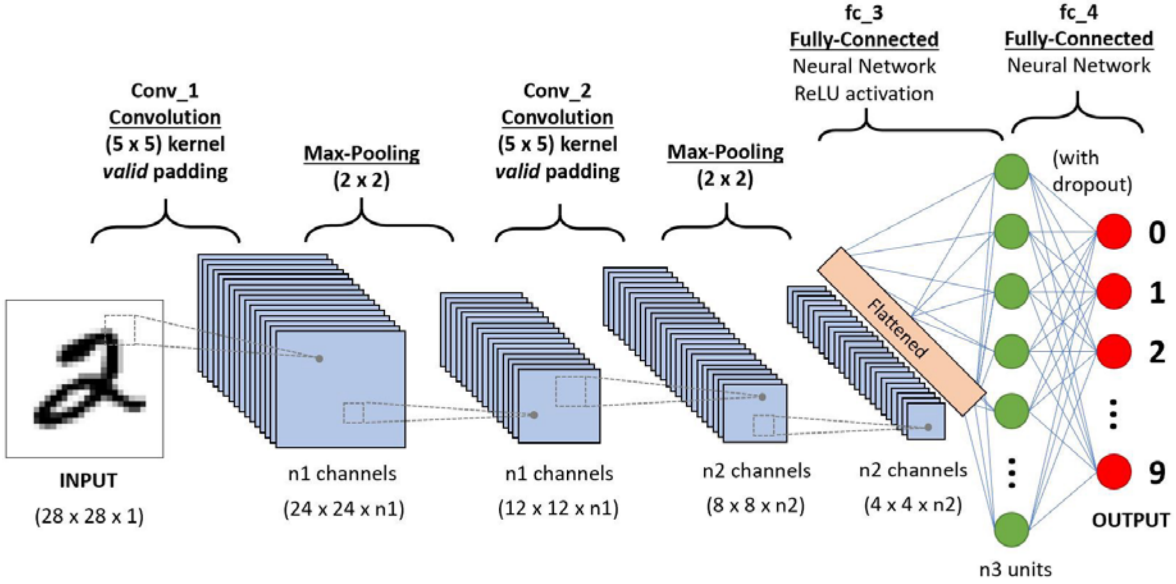
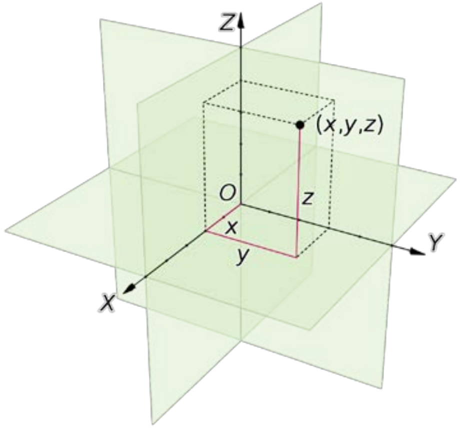
Crystal Graph Convolutional Neural Network(CGCNN)

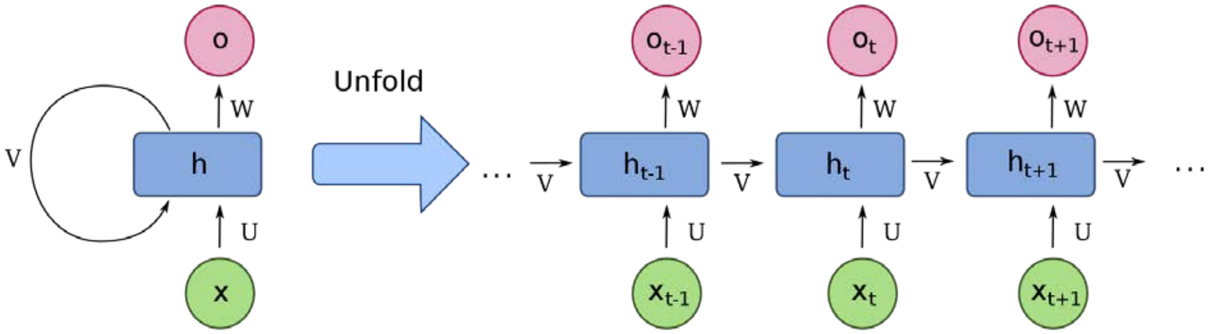
제일원리 전자구조계산 연구실
석사 과정 박형선

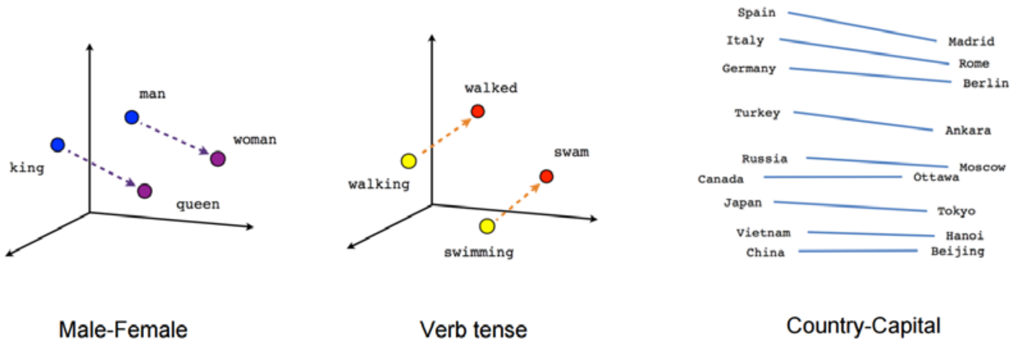
Typical data structure and neural network (Euclidean)



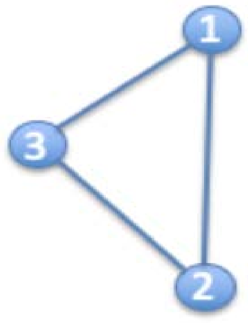


- Image processing -> CNN
- Sequence processing -> RNN
- Euclidean data structure



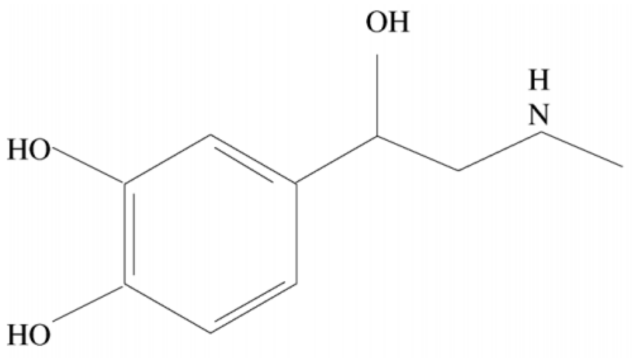


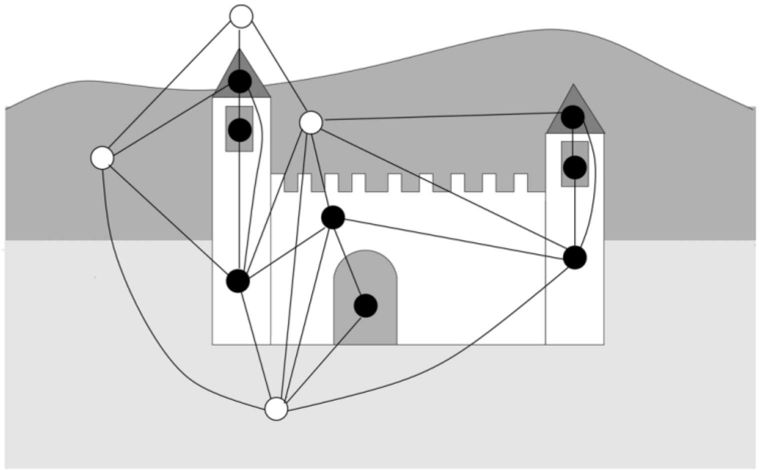
1. The Graph Neural Network Model (2009)

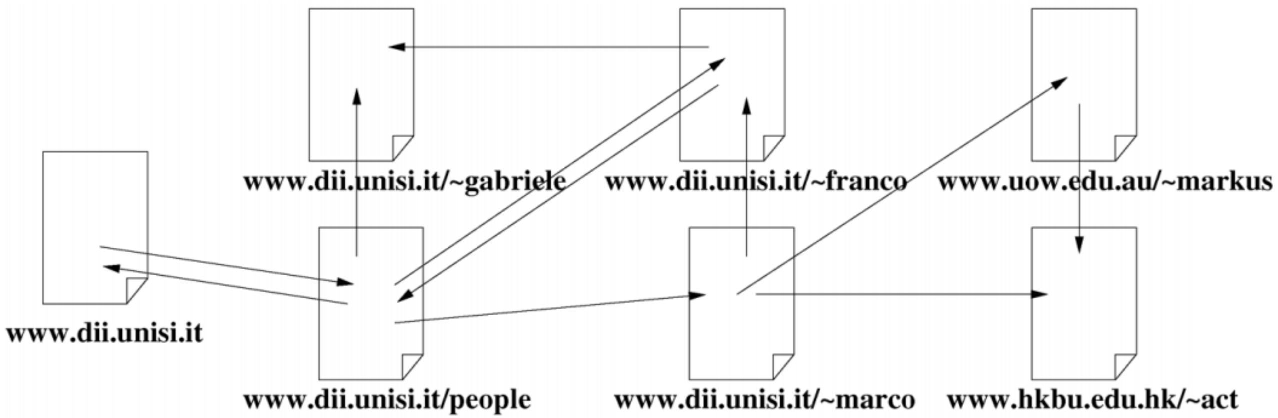


$G = (V, E)$

- Graph : Set of nodes and egdes
- Relational information between objects
- Directed/undirected
- Weighted/Unweighted
- Molecule structure, image, hyperlink







Notation and Principles

Theorem (Banach Fixed-Point Theorem). *Let (X, d) be a complete metric space and let $T : X \to X$ be a contraction mapping. Then T has a unique fixed point x' and for every $x \in X$ the sequence $T^k(x)$ with $k \to \infty$ converges to x'.*

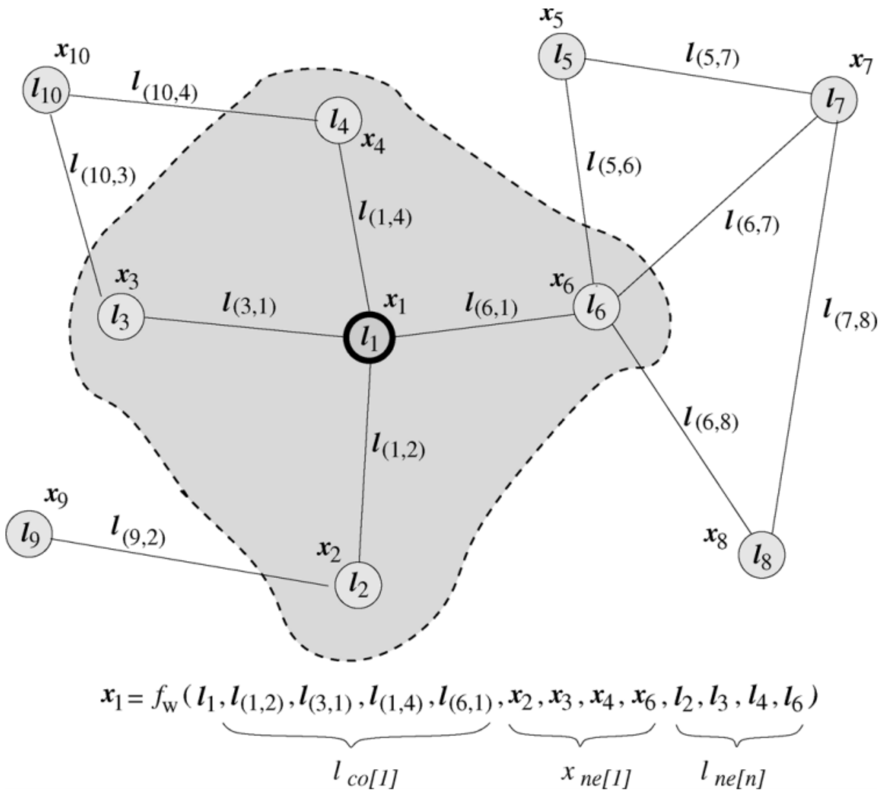


Fig. 2. Graph and the neighborhood of a node. The state $\boldsymbol{x}_1$ of the node 1 depends on the information contained in its neighborhood.

$$\mathcal{L} = \{(\boldsymbol{G}_i, n_{i,j}, \boldsymbol{t}_{i,j})|, \boldsymbol{G}_i = (\boldsymbol{N}_i, \boldsymbol{E}_i) \in \mathcal{G}; \\ n_{i,j} \in \boldsymbol{N}_i;\ \boldsymbol{t}_{i,j} \in \mathbb{R}^m,\ 1 \le i \le p, 1 \le j \le q_i\}$$

$$\boldsymbol{x}_n = f_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{l}_{\text{co}[n]}, \boldsymbol{x}_{\text{ne}[n]}, \boldsymbol{l}_{\text{ne}[n]}) \qquad \boldsymbol{x} = F_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l})$$

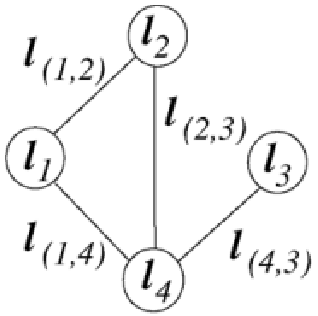
$$\boldsymbol{o}_n = g_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{l}_n) \qquad \boldsymbol{o} = G_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l}_{\boldsymbol{N}})$$

$$\mathbf{h}_v = f(\mathbf{x}_v, \mathbf{x}_{co[v]}, \mathbf{h}_{ne[v]}, \mathbf{x}_{ne[v]})$$

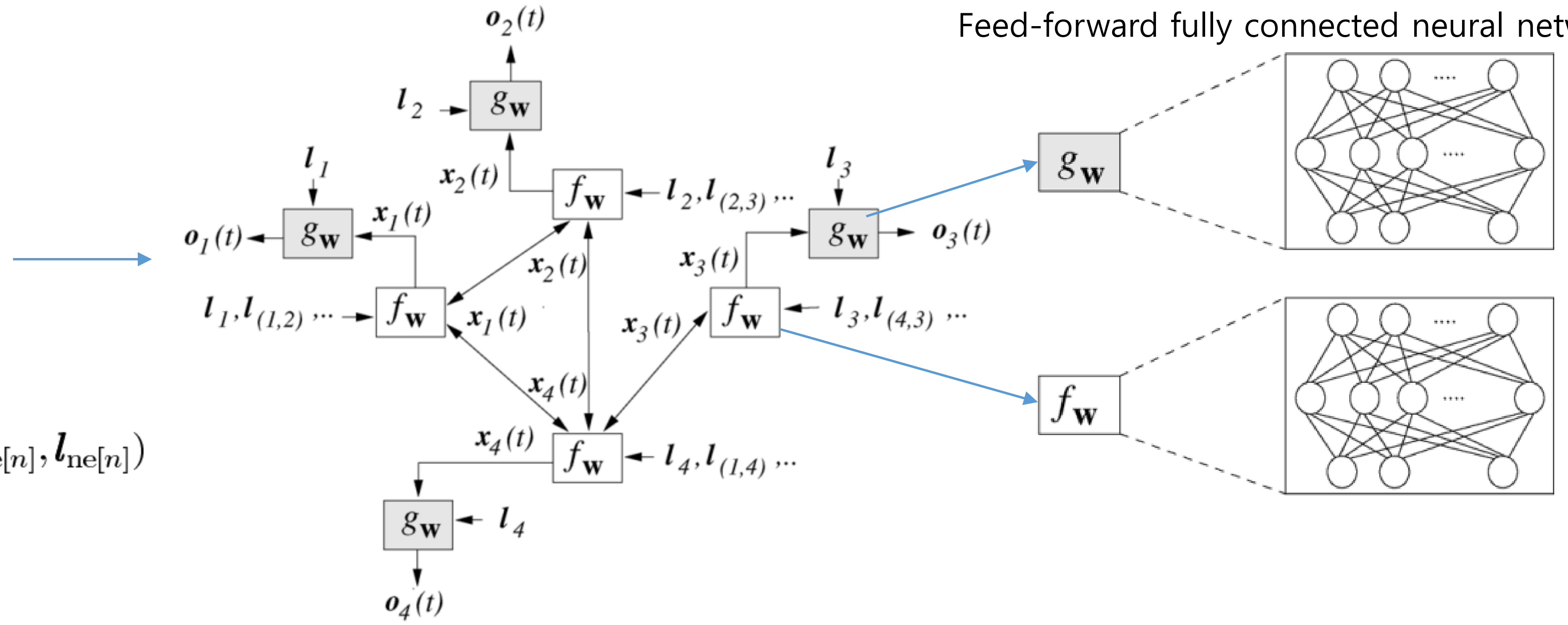
$$\mathbf{H}^{t+1} = F(\mathbf{H}^t, \mathbf{X})$$

- How to update the value of the nodes
- Local transition / output functions
- Global transition / output functions

Embedding network conversion

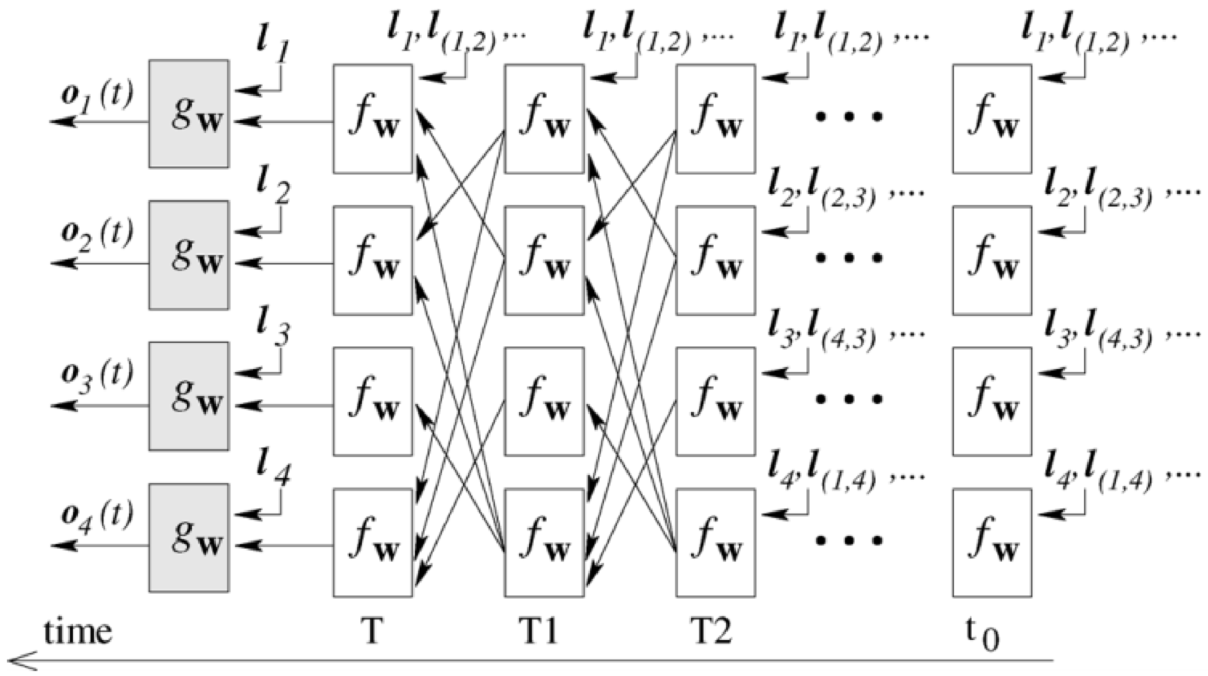


$$\boldsymbol{x}_n = f_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{l}_{\mathrm{co}[n]}, \boldsymbol{x}_{\mathrm{ne}[n]}, \boldsymbol{l}_{\mathrm{ne}[n]})$$
$$\boldsymbol{o}_n = g_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{l}_n)$$



- Structure conversion with parametric function f and g
- Objective : optimizing the parameters of the functions

Recurrent Neural Network(RNN)



- RecGNN
- Embedding block -> Recurrent Neural Network(RNN)
- Feedforward propagation can be applied
- Fixed point condition -> disturb stable representation training and various distribution of nodes

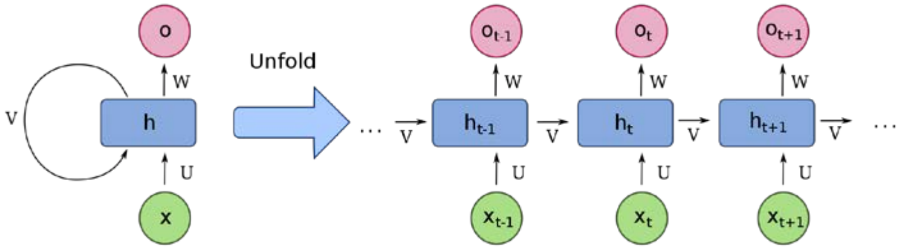
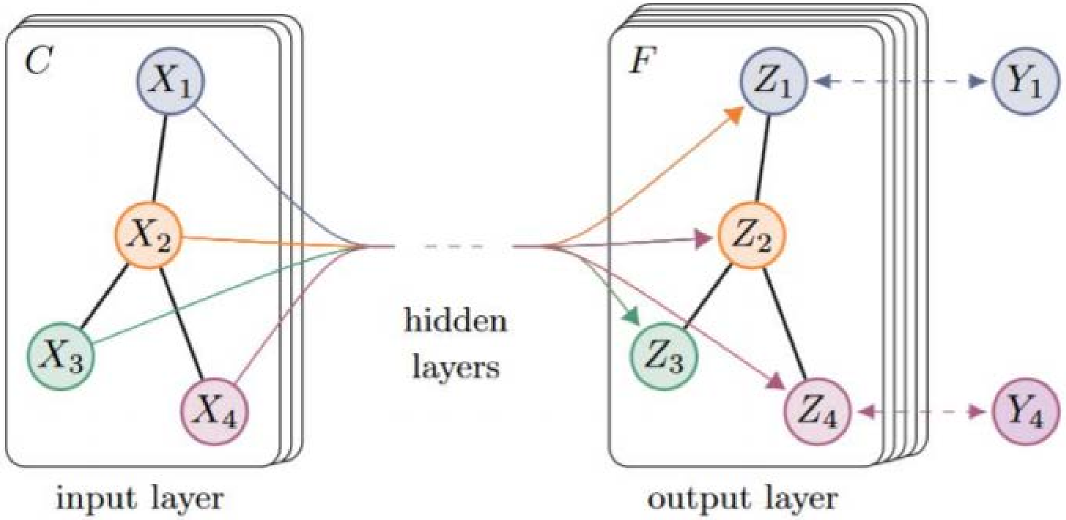
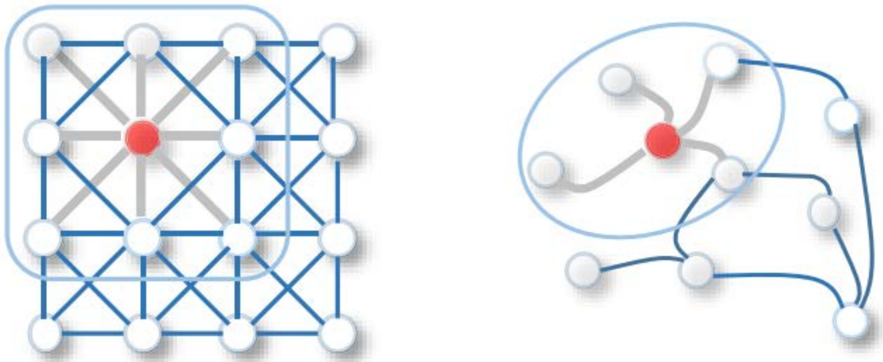
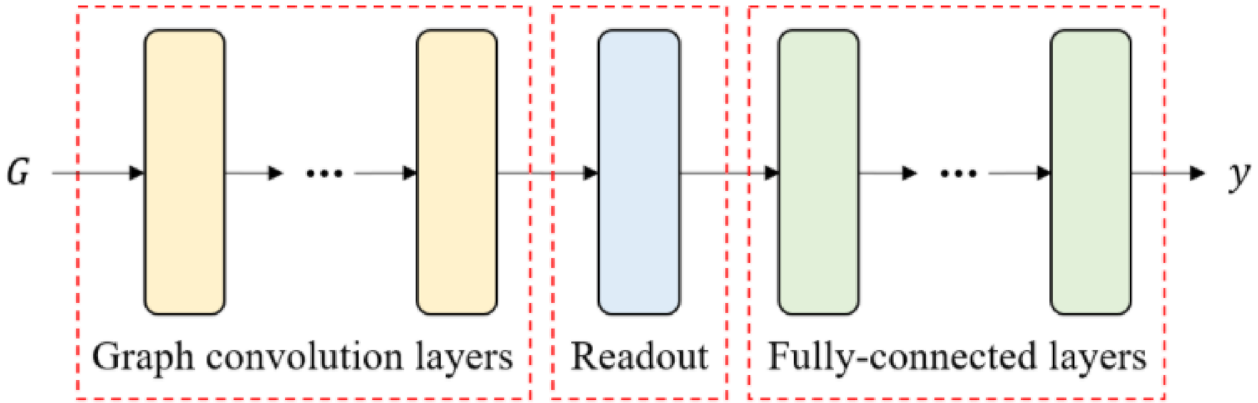


Fig. 3. Graph (on the top), the corresponding encoding network (in the middle), and the network obtained by unfolding the encoding network (at the bottom). The nodes (the circles) of the graph are replaced, in the encoding network, by units computing f_w and g_w (the squares). When f_w and g_w are implemented by feedforward neural networks, the encoding network is a recurrent neural network. In the unfolding network, each layer corresponds to a time instant and contains a copy of all the units of the encoding network. Connections between layers depend on encoding network connectivity.

2. Graph Convolution Network



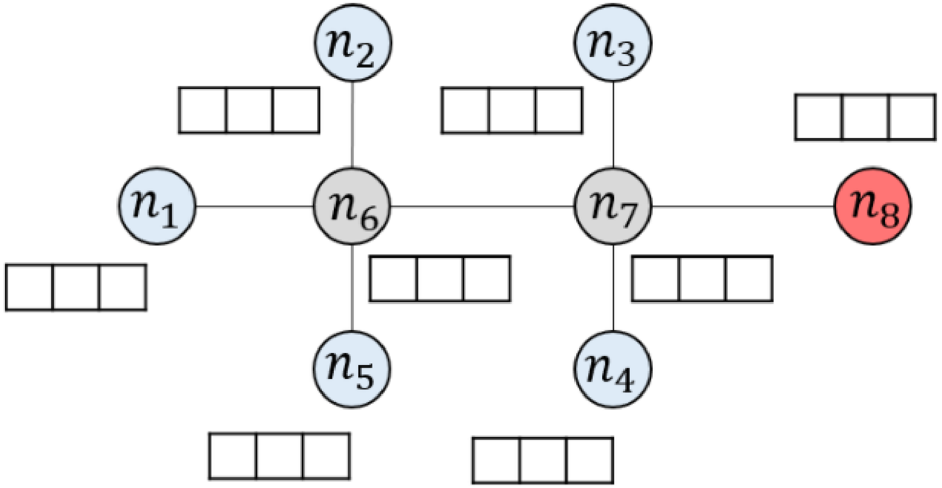




- Graph convolution : using the concept of spatial filter convolution in feed forward neural network
- Local node feature -> hidden state generation

Graph Convolution

- Adjacency matrix A, node feature matrix X, trainable weight matrix W, hidden matrix H, m = 2



$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$H = \psi(A, X) = \sigma(AXW)$$

$$A \in \mathbb{R}^{N\times N}$$

$$X \in \mathbb{R}^{N\times d}$$

$$W \in \mathbb{R}^{d\times m}$$

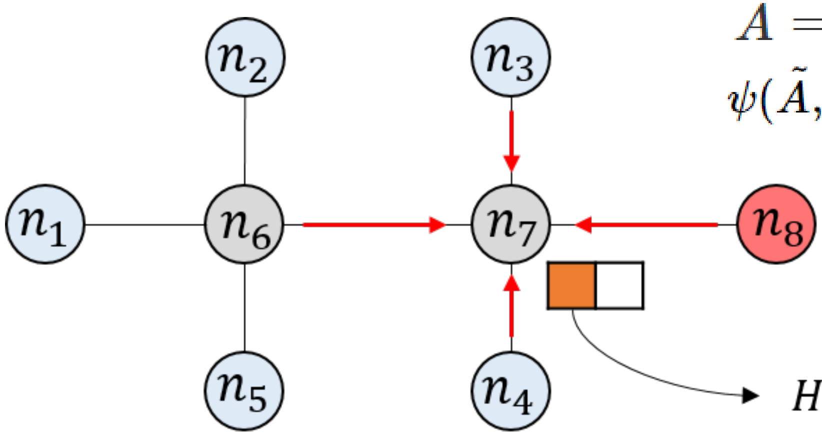
$$H \in \mathbb{R}^{N\times m}$$

$$S = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ \vdots & \vdots & \vdots \\ x_{81} & x_{82} & x_{83} \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{3} w_{i1}x_{1i} & \sum_{i=1}^{3} w_{i2}x_{1i} \\ \sum_{i=1}^{3} w_{i1}x_{2i} & \sum_{i=1}^{3} w_{i2}x_{2i} \\ \vdots & \vdots \\ \sum_{i=1}^{3} w_{i1}x_{8i} & \sum_{i=1}^{3} w_{i2}x_{8i} \end{bmatrix}$$

강원대학교 물리학과
Kangwon National University Department of Physics
박 형 선
전자구조계산 연구실

Graph Convolution

- Adjacency matrix A, node feature matrix X, trainable weight matrix W, hidden matrix H, m = 2



$$\tilde{A} = A + I$$

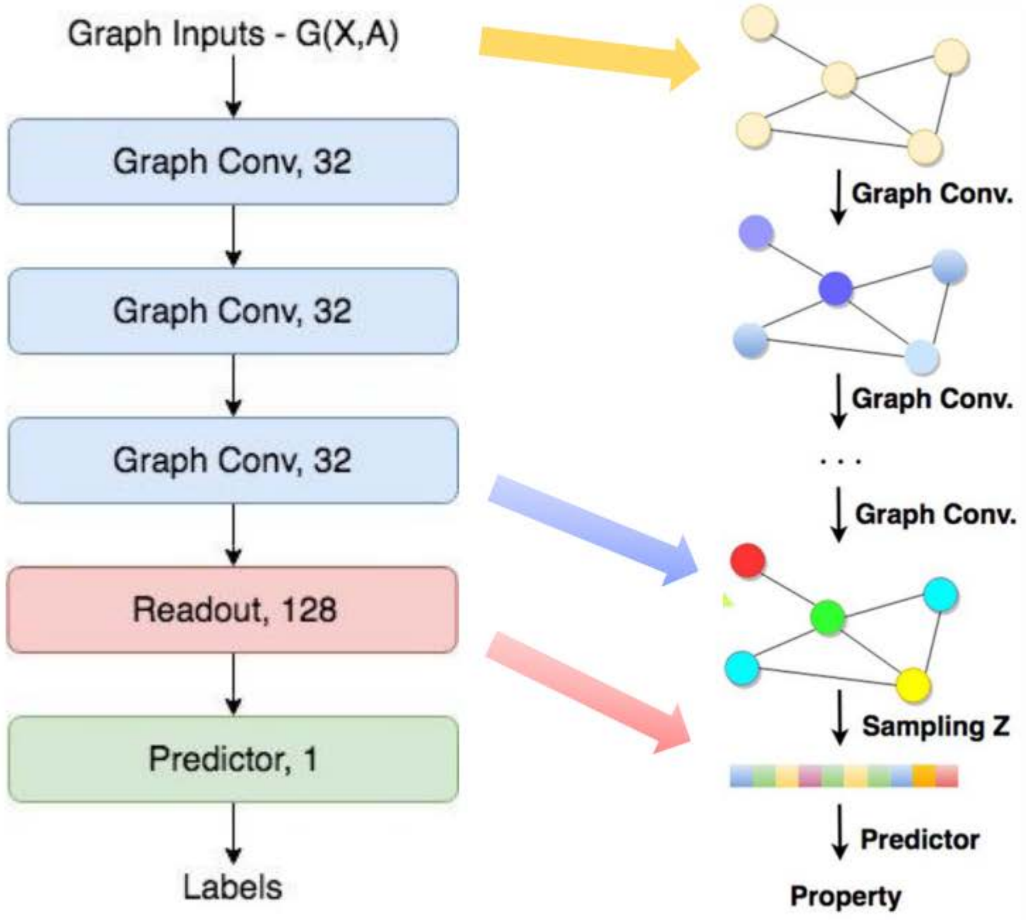
$$\psi(\tilde{A}, X) = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}XW)$$

$$H_{71} = \sigma(s_{31} + s_{41} + s_{61} + s_{81})$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$S = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ \vdots & \vdots & \vdots \\ x_{81} & x_{82} & x_{83} \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{3} w_{i1}x_{1i} & \sum_{i=1}^{3} w_{i2}x_{1i} \\ \sum_{i=1}^{3} w_{i1}x_{2i} & \sum_{i=1}^{3} w_{i2}x_{2i} \\ \vdots & \vdots \\ \sum_{i=1}^{3} w_{i1}x_{8i} & \sum_{i=1}^{3} w_{i2}x_{8i} \end{bmatrix}$$

Overall structure of GCN



Input node features, $\left\{H_i^{(0)}\right\}$

: Raw node information

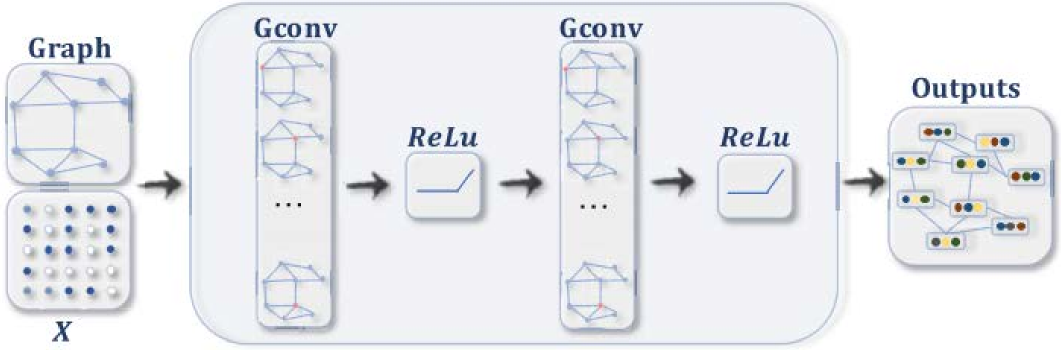
- A Massage Passing function: $m_v^{t+1} = \sum_{w\in N(v)} M_t(h_v^t, h_u^t, e_{uv})$
- A Node Update function: $h_v^{l+1} = U_t(h_v^t, m_v^{t+1})$
- A Readout function (for graph classification): $\hat{y} = R(\{h_v^T | v \in G\})$

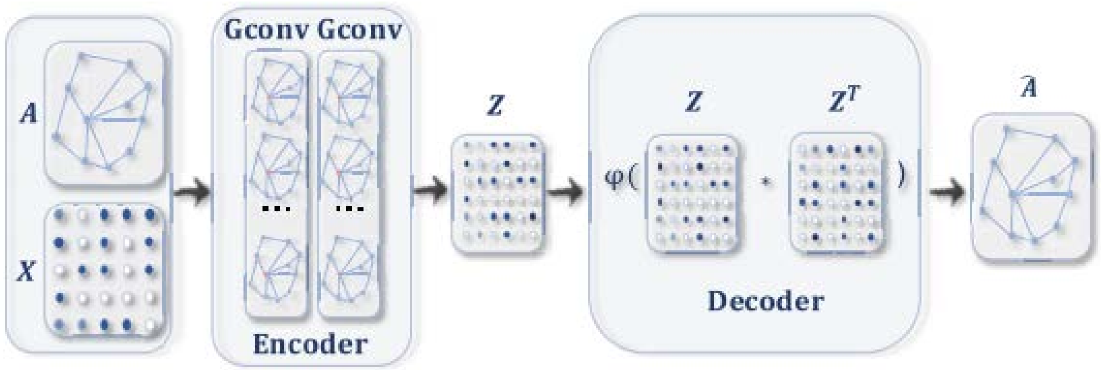
Final node states, $\left\{H_i^{(L)}\right\}$

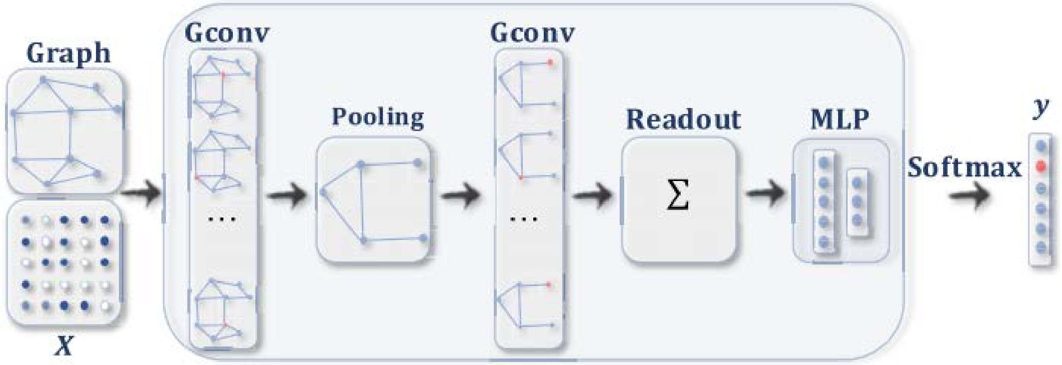
: How the NN recognizes the nodes

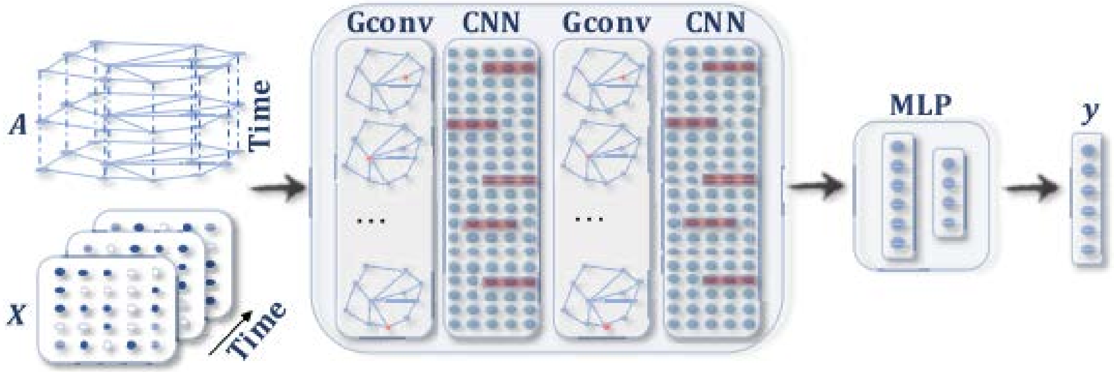
Graph features, Z

Overall structures of GCN by objectives

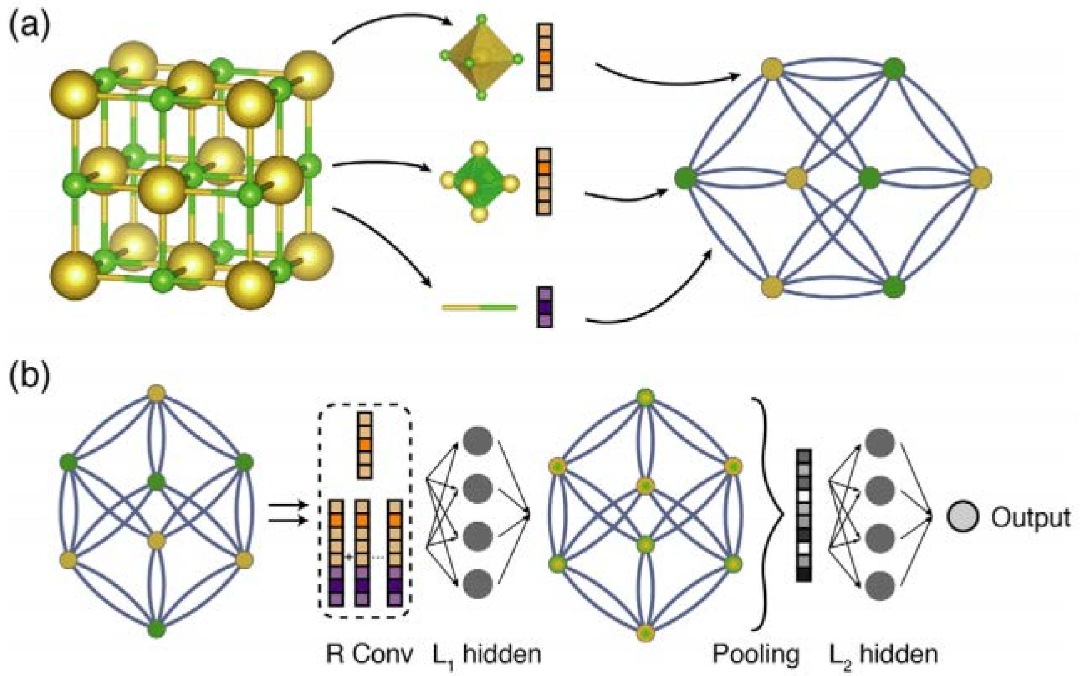








3. Main idea of the CGCNN paper



$$v_i^{(t+1)} = \mathrm{Conv}\left(v_i^{(t)}, v_j^{(t)}, u_{(i,j)_k}\right), \quad (i,j)_k \in \mathcal{G}.$$

$$v_c = \mathrm{Pool}(v_0^{(0)}, v_1^{(0)}, \ldots, v_N^{(0)}, \ldots, v_N^{(R)})$$

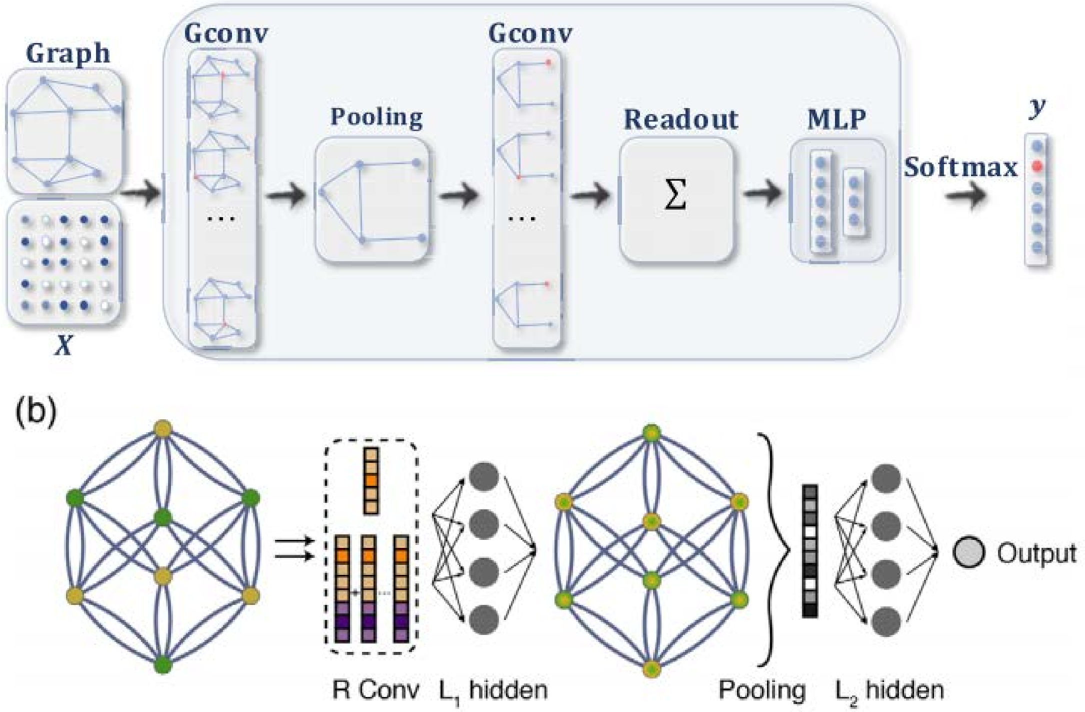
$$\min_{W} J(y, f(\mathcal{C}; W))$$

$$v_i^{(t+1)} = g\left[\left(\sum_{j,k} v_j^{(t)} \oplus u_{(i,j)_k}\right) W_c^{(t)} + v_i^{(t)} W_s^{(t)} + b^{(t)}\right]$$

$$v_i^{(t+1)} = v_i^{(t)} + \sum_{j,k} \sigma\left(z_{(i,j)_k}^{(t)} W_f^{(t)} + b_f^{(t)}\right) \odot g\left(z_{(i,j)_k}^{(t)} W_s^{(t)} + b_s^{(t)}\right),$$

- Molecular structure → transformation to graph structure
- Implementation of Graph convolution and pooling

3. Main idea of the CGCNN paper



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

$$\boldsymbol{v}_c = \mathrm{Pool}(\boldsymbol{v}_0^{(0)}, \boldsymbol{v}_1^{(0)}, \ldots, \boldsymbol{v}_N^{(0)}, \ldots, \boldsymbol{v}_N^{(R)})$$

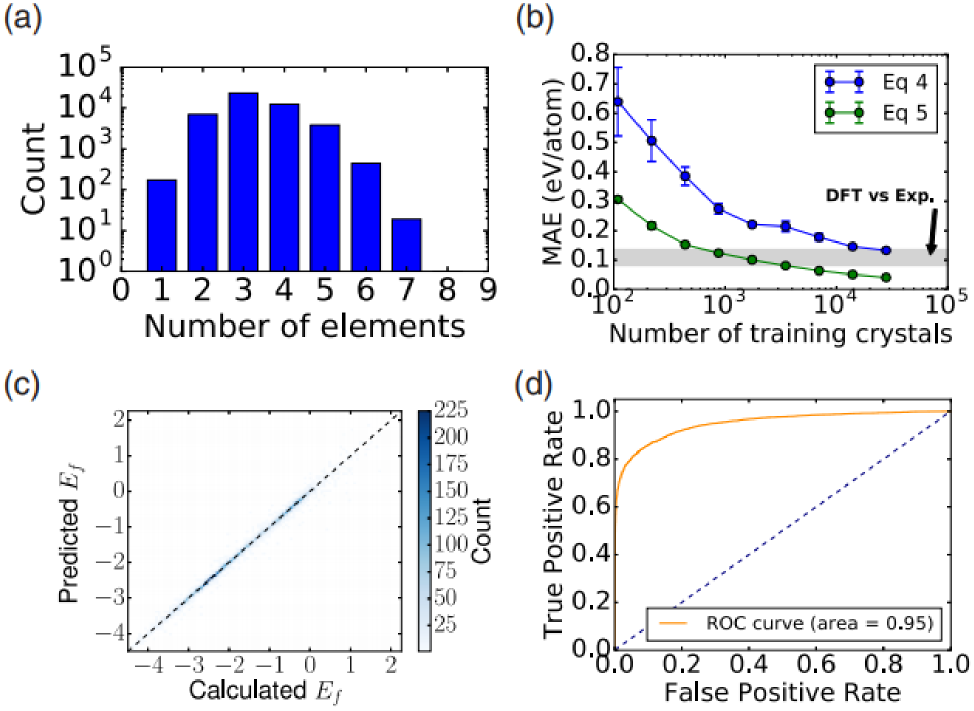
$$\min_{\boldsymbol{W}} J(y, f(\mathcal{C}; \boldsymbol{W}))$$

$$\boldsymbol{v}_i^{(t+1)} = g\left[\left(\sum_{j,k} \boldsymbol{v}_j^{(t)} \oplus \boldsymbol{u}_{(i,j)_k}\right)\boldsymbol{W}_c^{(t)} + \boldsymbol{v}_i^{(t)}\boldsymbol{W}_s^{(t)} + \boldsymbol{b}^{(t)}\right]$$

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

- Molecular structure → transformation to graph structure
- Implementation of Graph convolution and pooling

Performance



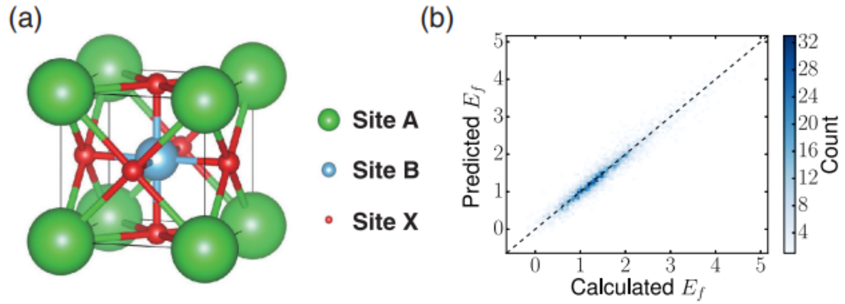


TABLE I. Summary of the prediction performance of seven different properties on test sets.

Property	# of train data	Unit	MAE_{model}	MAE_{DFT}
Formation energy	28 046	eV/atom	0.039	0.081–0.136 [28]
Absolute energy	28 046	eV/atom	0.072	⋯
Band gap	16 458	eV	0.388	0.6 [32]
Fermi energy	28 046	eV	0.363	⋯
Bulk moduli	2041	log(GPa)	0.054	0.050 [13]
Shear moduli	2041	log(GPa)	0.087	0.069 [13]
Poisson ratio	2041	⋯	0.030	⋯

4. Performance Test : energy per atom

```
new_crystal = data[cif_id][0]['structure']
new_crystal
```

```
Structure Summary
Lattice
    abc : 8.985792534516808 8.985792534516808 11.889019
 angles : 90.0 90.0 150.6870489721173
 volume : 469.9829181504342
      A : 2.273613 -8.693397 0.0
      B : 2.273613 8.693397 0.0
      C : 0.0 0.0 11.889019
PeriodicSite: Cs (2.2736, 4.3640, 2.9723) [0.2490, 0.7510, 0.2500]
PeriodicSite: Cs (2.2736, -4.3640, 8.9168) [0.7510, 0.2490, 0.7500]
PeriodicSite: Dy (0.0000, 0.0000, 5.9445) [0.0000, 0.0000, 0.5000]
PeriodicSite: Dy (0.0000, 0.0000, 0.0000) [0.0000, 0.0000, 0.0000]
PeriodicSite: Cd (2.2736, -0.6417, 2.9723) [0.5369, 0.4631, 0.2500]
PeriodicSite: Cd (2.2736, 0.6417, 8.9168) [0.4631, 0.5369, 0.7500]
PeriodicSite: Te (2.2736, 2.0830, 6.4966) [0.3802, 0.6198, 0.5464]
PeriodicSite: Te (2.2736, -2.0830, 5.3924) [0.6198, 0.3802, 0.4536]
PeriodicSite: Te (2.2736, -2.0830, 0.5521) [0.6198, 0.3802, 0.0464]
PeriodicSite: Te (2.2736, 2.0830, 11.3369) [0.3802, 0.6198, 0.9536]
PeriodicSite: Te (2.2736, -7.5590, 2.9723) [0.9348, 0.0652, 0.2500]
PeriodicSite: Te (2.2736, 7.5590, 8.9168) [0.0652, 0.9348, 0.7500]
```

- Material Project : DFT calculation open database
- Pymatgen(Python Material Genomics) - MPRester
- cif, xyz, gaussian file format -> structure object

Performance Test : energy per atom

```
 92%|█████████▎| 37/40 [13:07<00:27,  9.03s/it]Test: [0/4]    Time: 0.035 (0.035)    Loss: 0.0631 (0.0631)   MAE: 0.253 (0.253)
 * MAE 0.231
Epoch: [37][0/47]       Time: 0.201 (0.201)     Data: 0.035 (0.035)     Loss: 0.0322 (0.0322)   MAE: 0.229 (0.229)
Epoch: [37][10/47]      Time: 0.190 (0.194)     Data: 0.018 (0.020)     Loss: 0.0097 (0.0286)   MAE: 0.115 (0.199)
Epoch: [37][20/47]      Time: 0.186 (0.191)     Data: 0.017 (0.019)     Loss: 0.0085 (0.0233)   MAE: 0.098 (0.172)
Epoch: [37][30/47]      Time: 0.188 (0.191)     Data: 0.017 (0.019)     Loss: 0.0124 (0.0233)   MAE: 0.115 (0.170)
Epoch: [37][40/47]      Time: 0.191 (0.191)     Data: 0.018 (0.018)     Loss: 0.0240 (0.0217)   MAE: 0.135 (0.165)

 95%|█████████▌| 38/40 [13:16<00:18,  9.04s/it]Test: [0/4]    Time: 0.035 (0.035)    Loss: 0.0218 (0.0218)   MAE: 0.117 (0.117)
 * MAE 0.123
Epoch: [38][0/47]       Time: 0.207 (0.207)     Data: 0.037 (0.037)     Loss: 0.0102 (0.0102)   MAE: 0.110 (0.110)
Epoch: [38][10/47]      Time: 0.200 (0.196)     Data: 0.018 (0.020)     Loss: 0.0257 (0.0150)   MAE: 0.216 (0.139)
Epoch: [38][20/47]      Time: 0.192 (0.192)     Data: 0.017 (0.019)     Loss: 0.0226 (0.0178)   MAE: 0.197 (0.160)
Epoch: [38][30/47]      Time: 0.182 (0.191)     Data: 0.020 (0.019)     Loss: 0.0098 (0.0193)   MAE: 0.104 (0.162)
Epoch: [38][40/47]      Time: 0.185 (0.190)     Data: 0.017 (0.018)     Loss: 0.0400 (0.0197)   MAE: 0.128 (0.159)

 98%|█████████▊| 39/40 [13:25<00:09,  9.05s/it]Test: [0/4]    Time: 0.037 (0.037)    Loss: 0.0207 (0.0207)   MAE: 0.129 (0.129)
 * MAE 0.129
Epoch: [39][0/47]       Time: 0.212 (0.212)     Data: 0.037 (0.037)     Loss: 0.0113 (0.0113)   MAE: 0.104 (0.104)
Epoch: [39][10/47]      Time: 0.182 (0.193)     Data: 0.017 (0.020)     Loss: 0.0191 (0.0175)   MAE: 0.107 (0.145)
Epoch: [39][20/47]      Time: 0.184 (0.192)     Data: 0.017 (0.019)     Loss: 0.0102 (0.0154)   MAE: 0.104 (0.137)
Epoch: [39][30/47]      Time: 0.203 (0.191)     Data: 0.018 (0.019)     Loss: 0.0147 (0.0151)   MAE: 0.154 (0.135)
Epoch: [39][40/47]      Time: 0.196 (0.191)     Data: 0.018 (0.019)     Loss: 0.0104 (0.0152)   MAE: 0.126 (0.132)

100%|██████████| 40/40 [13:34<00:00, 20.37s/it]Test: [0/4]    Time: 0.034 (0.034)    Loss: 0.0206 (0.0206)   MAE: 0.124 (0.124)
 * MAE 0.127
```

감사합니다

Reference : The Graph Neural Network Model [IEEE]

A Comprehensive Survey on Graph Neural Networks [IEEE]

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS [arXiv]

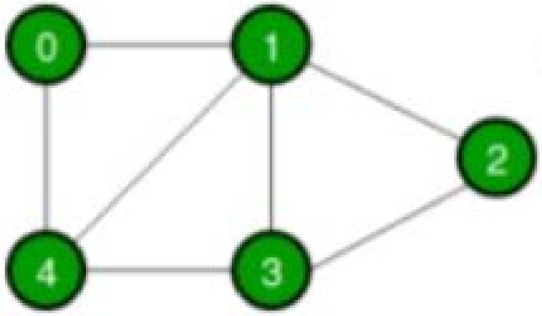
Graph Neural Networks A Review of Methods and Applications [arXiv]

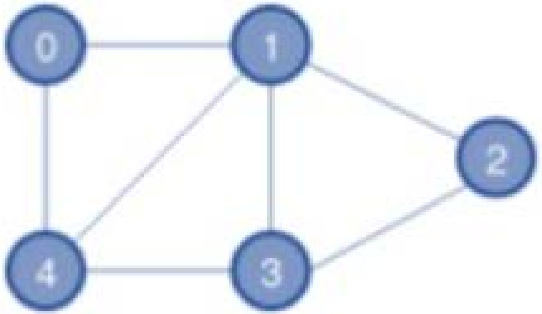
Supplementary

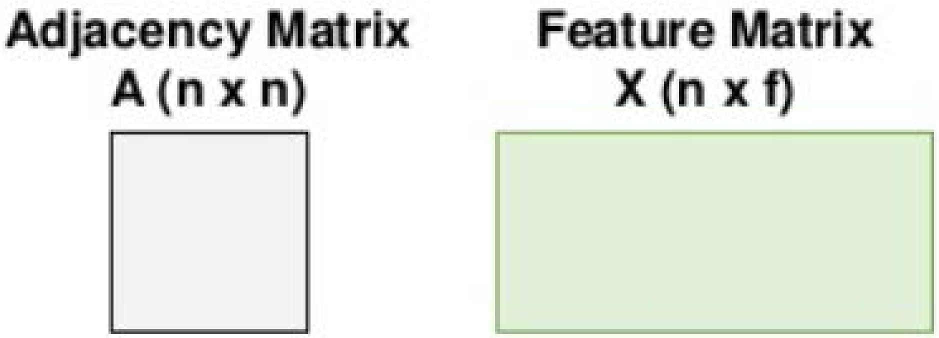
How to update hidden state

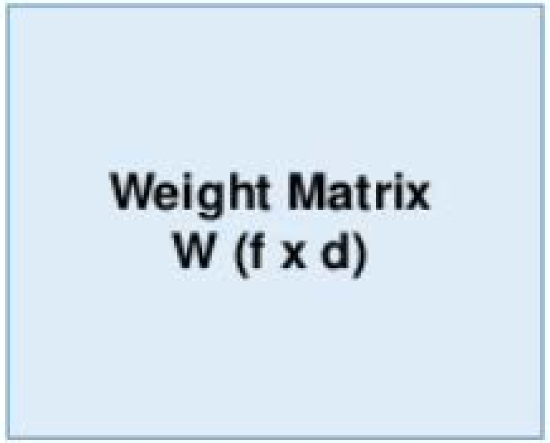
- How to update hidden states in GCN (Matrix 연산으로 이해하기)

$$H^{(l+1)} = \sigma\left(AH^{(l)}W^{(l)} + b^{(l)}\right)$$



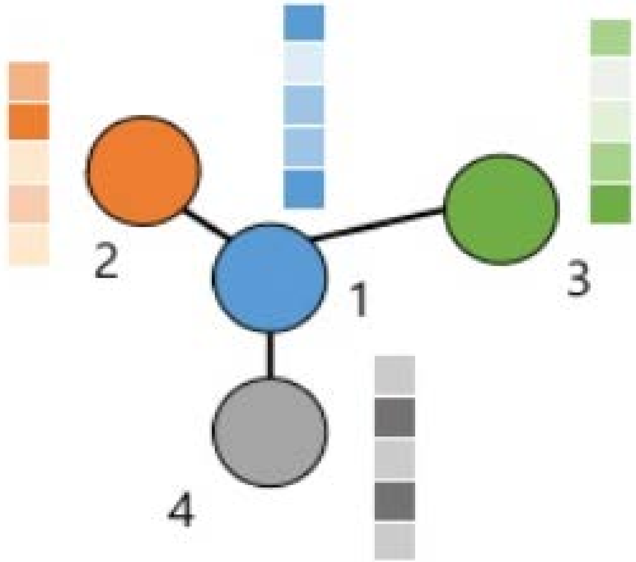






(참고) n: 노드의 수 / f: 특징 수 / d: 필터수 (depth)
결국 d가 다음 레이어의 f 값이 된다

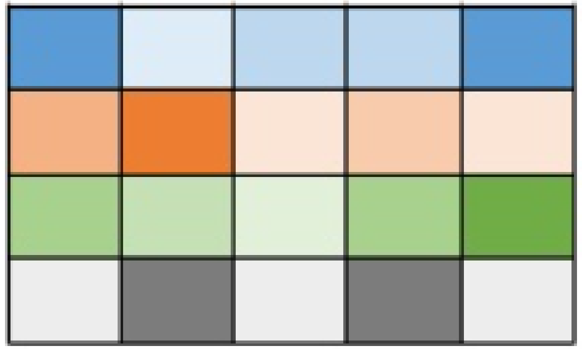
Adjacency matrix, Feature matrix



Adjacency Matrix **A** (4 x 4)

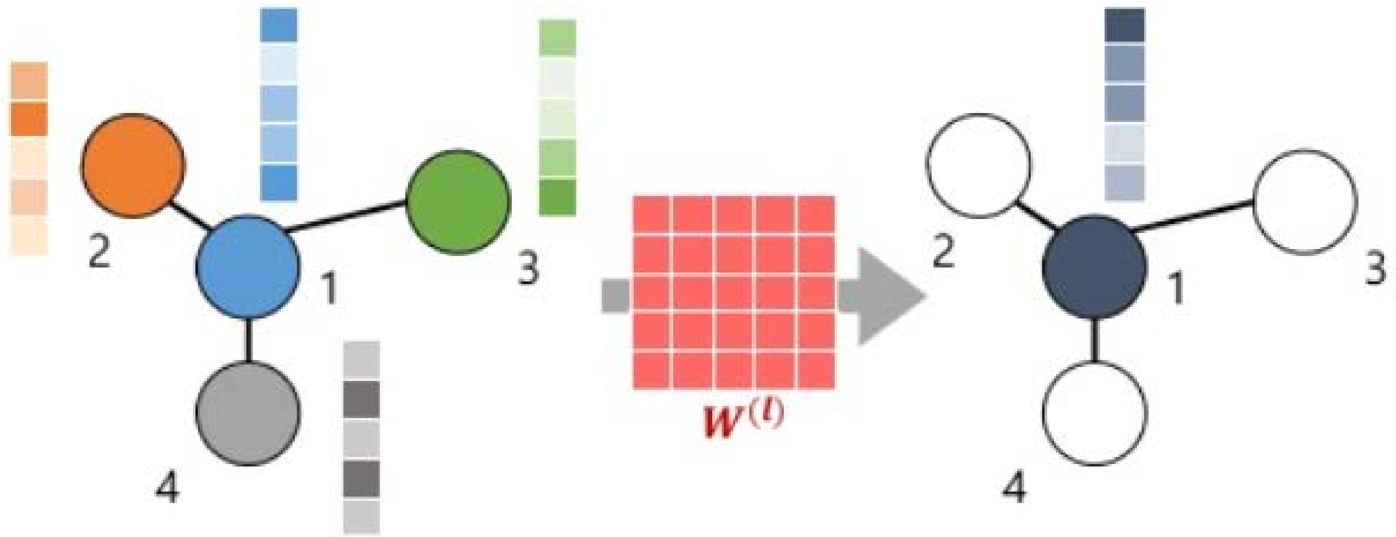
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1

Feature Matrix **X** (4 x 5)



Updating algorithm for hidden state

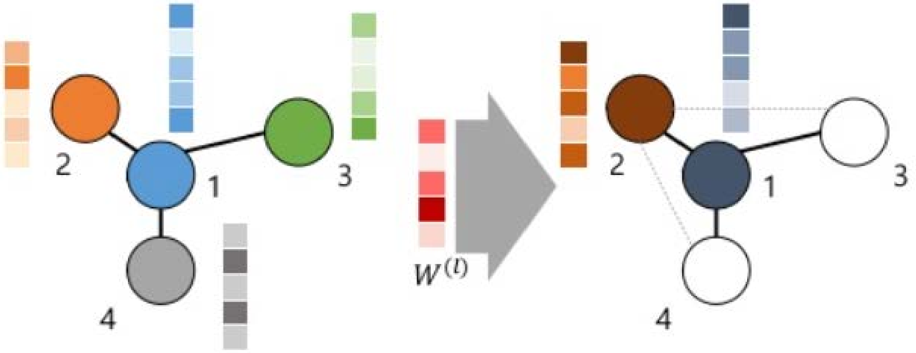
$$H_1^{(l+1)} = \sigma\left(H_1^{(l)}W^{(l)} + H_2^{(l)}W^{(l)} + H_3^{(l)}W^{(l)} + H_4^{(l)}W^{(l)} + b^{(l)}\right)$$



$$H_1^{(l+1)} = \sigma\left(H_1^{(l)}\mathbf{W}^{(l)} + H_2^{(l)}\mathbf{W}^{(l)} + H_3^{(l)}\mathbf{W}^{(l)} + H_4^{(l)}\mathbf{W}^{(l)} + b^{(l)}\right)$$

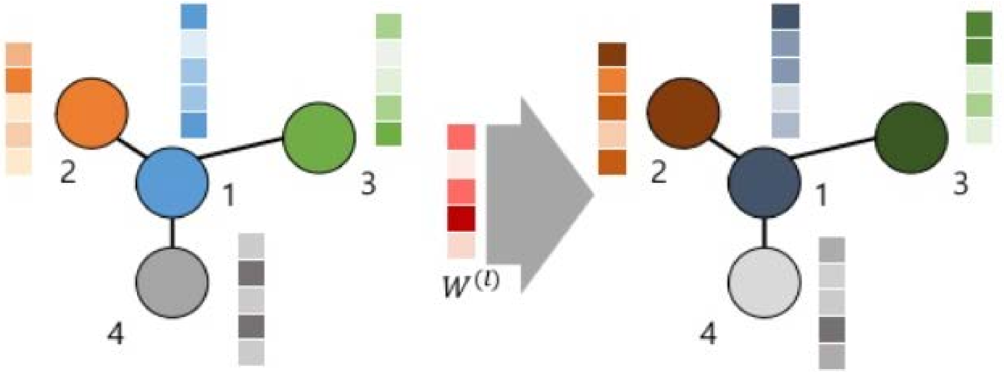
Weight Sharing

Updating examples

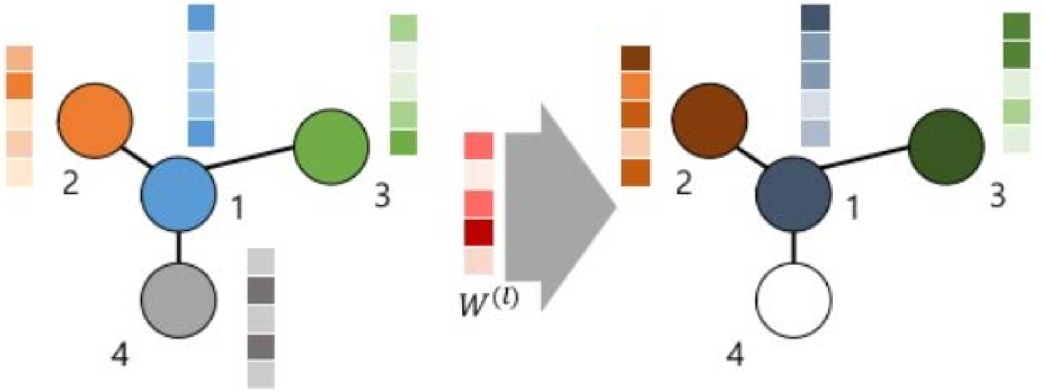


$$H_2^{(l+1)} = \sigma\left(H_1^{(l)}W^{(l)} + H_2^{(l)}W^{(l)} + \cancelto{0}{H_3^{(l)}}W^{(l)} + \cancelto{0}{H_4^{(l)}}W^{(l)} + b^{(l)}\right)$$

주변 노드 정보 활용

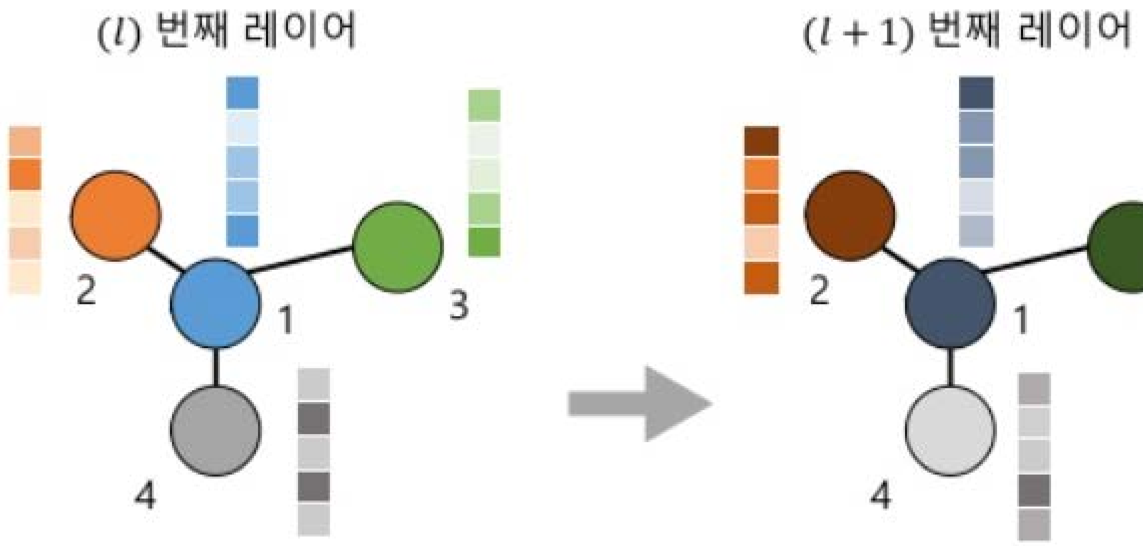


$$H_4^{(l+1)} = \sigma\left(H_1^{(l)}W^{(l)} + H_4^{(l)}W^{(l)} + b^{(l)}\right)$$



$$H_3^{(l+1)} = \sigma\left(H_1^{(l)}W^{(l)} + H_3^{(l)}W^{(l)} + b^{(l)}\right)$$

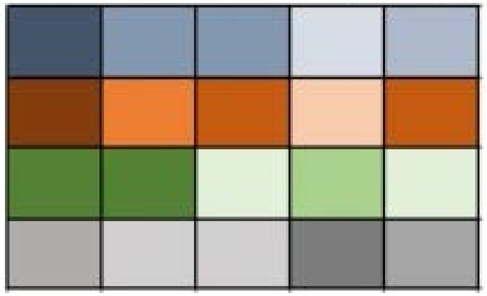
Updating examples



Adjacency Matrix **A** (4 x 4)

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

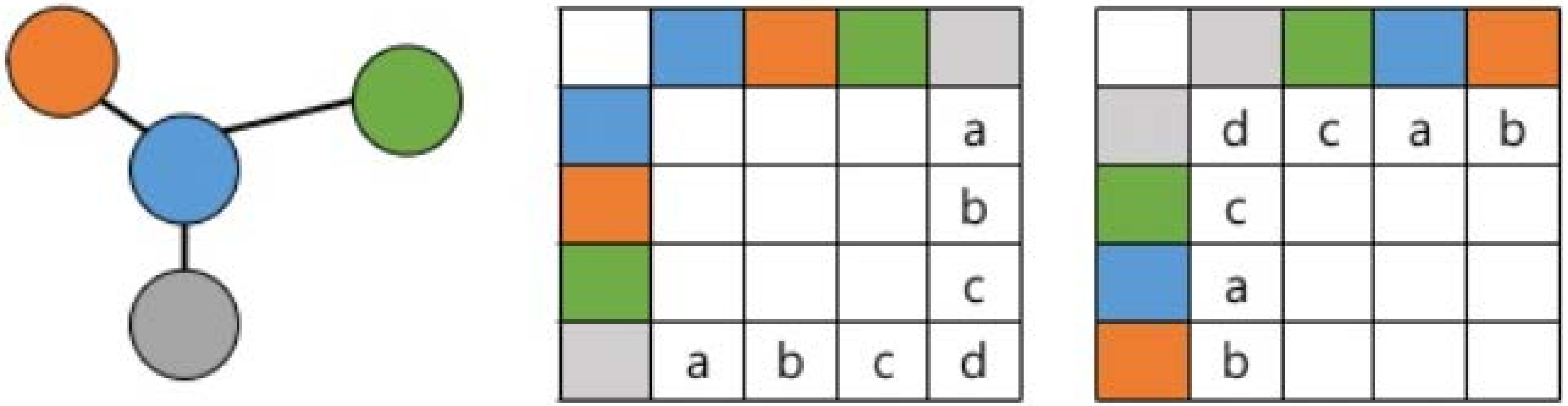
Feature Matrix **X** (4 x 5)



$$H_i^{(l+1)} = \sigma\left(\sum_{j\in N(i)} H_j^{(l)} W^{(l)} + b^{(l)}\right) \quad \text{or} \quad H^{(l+1)} = \sigma\left(AH^{(l)}W^{(l)} + b^{(l)}\right)$$

5. Readout layer

- Readout: Permutation Invariance



노드 순서에 따라 값의 변동이 있을 수 있다

Readout 방법 중 한가지: Node-wise summation

$$z_G = \sigma\left(\sum_{i \in G} MLP(H_i^{(l)})\right)$$