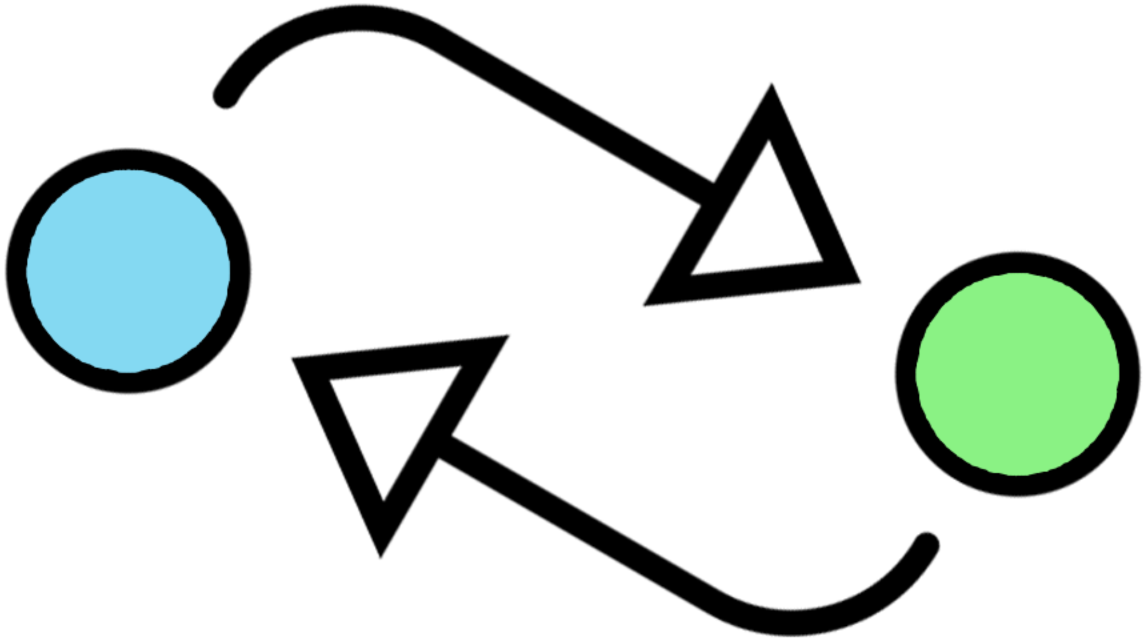


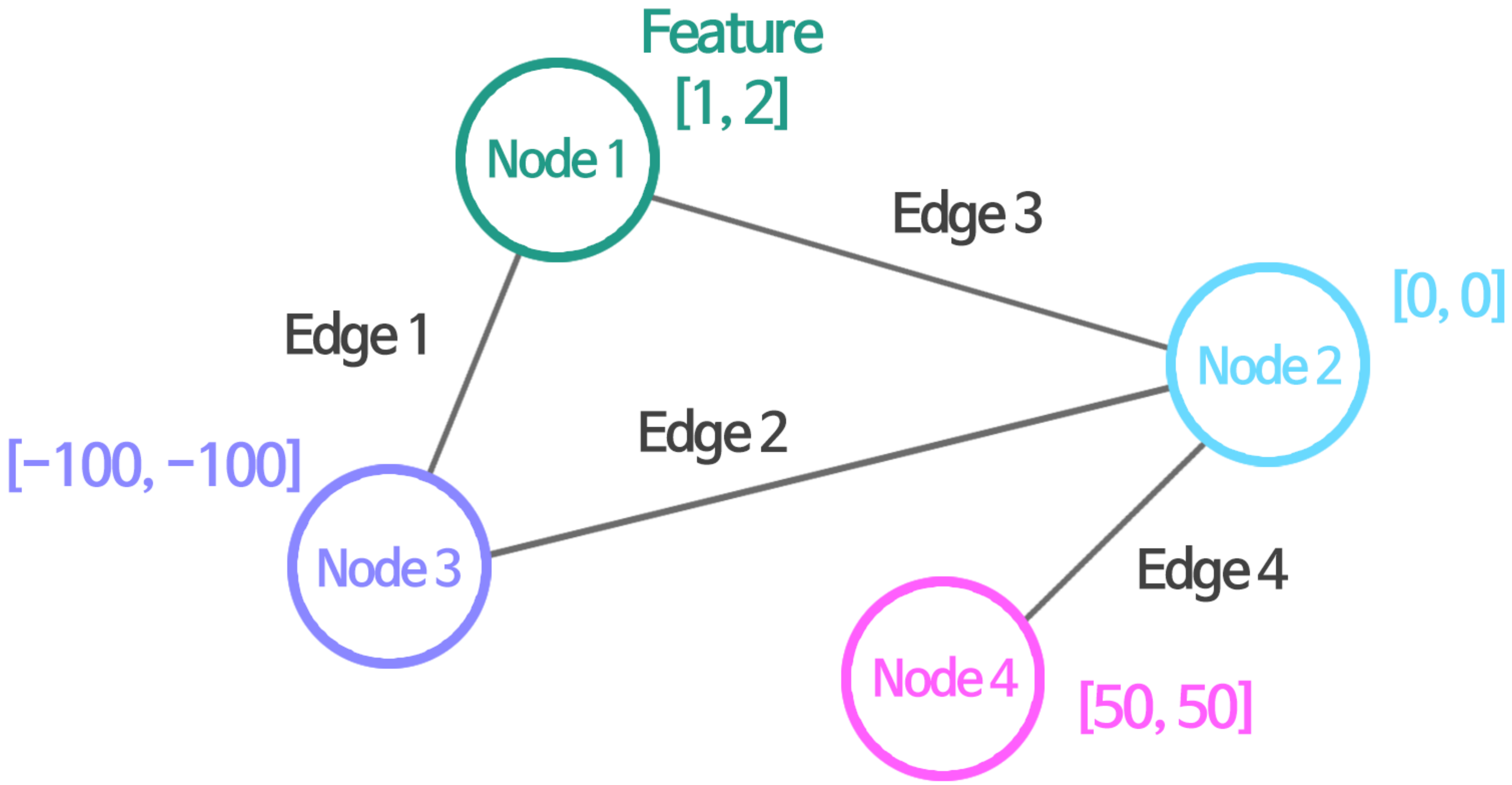
DGNN

Skeleton-Based Action Recognition with Directed Graph Neural Networks

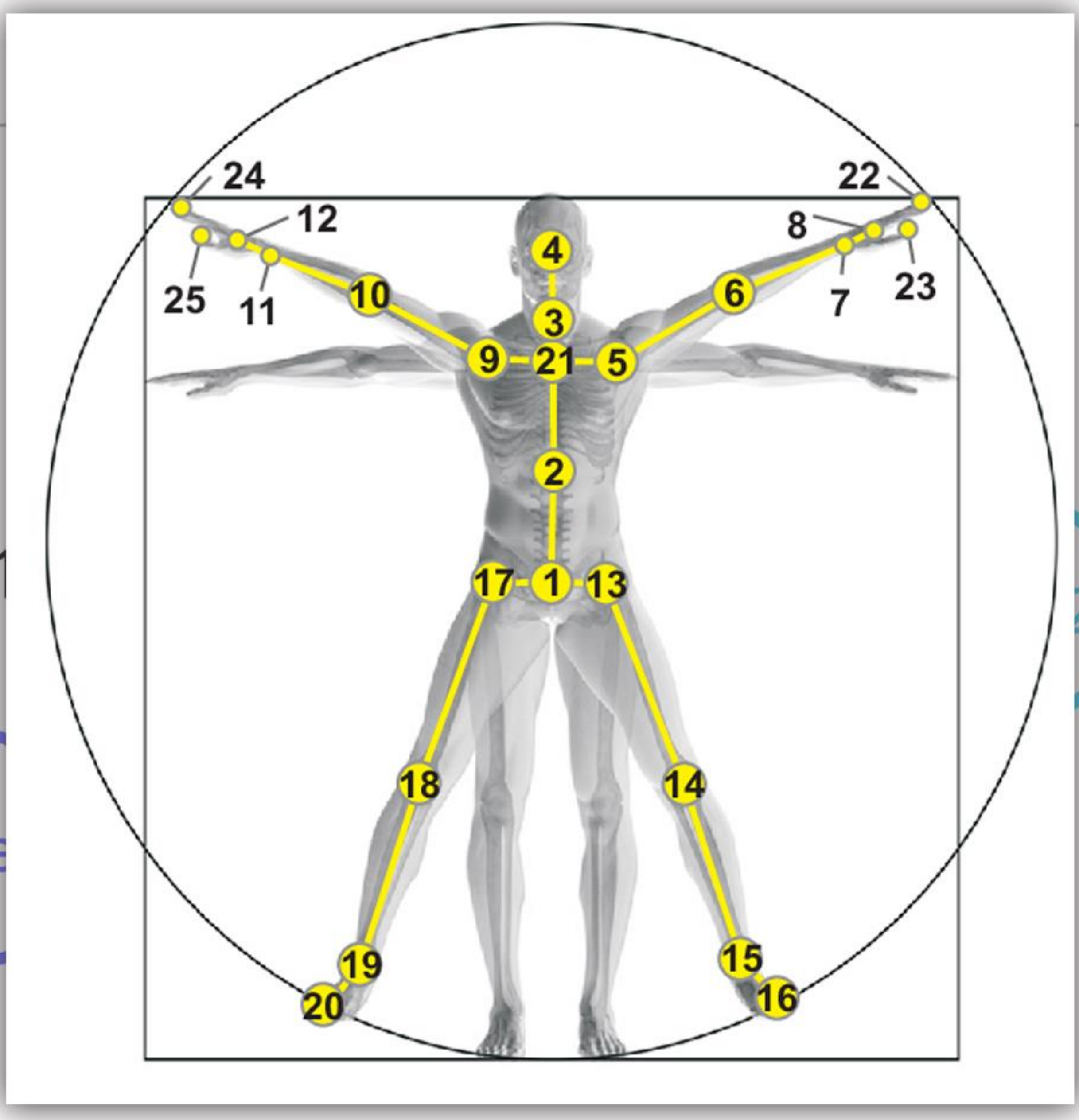
MOON SUNGWON

2020.02.11

Intro


Feature
[1, 2]
Node 1
Edge 3
[0, 0]
Node 2
Edge 1
Edge 2
[−100, −100]
Node 3
Edge 4
Node 4
[50, 50]

Intro



Edge 1

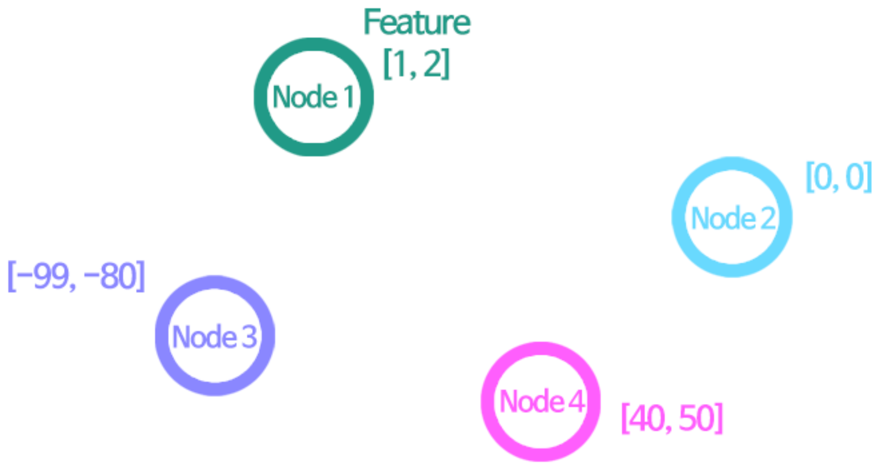
[0, 0]

2

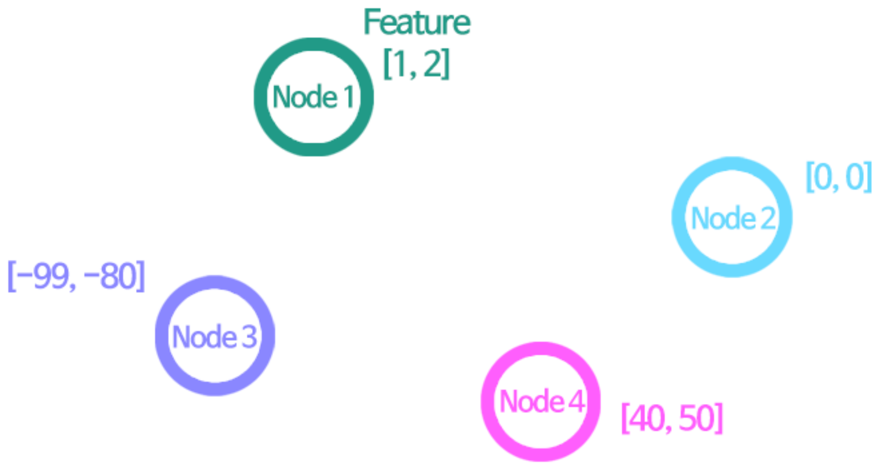
[−100, −100]

Node

Intro



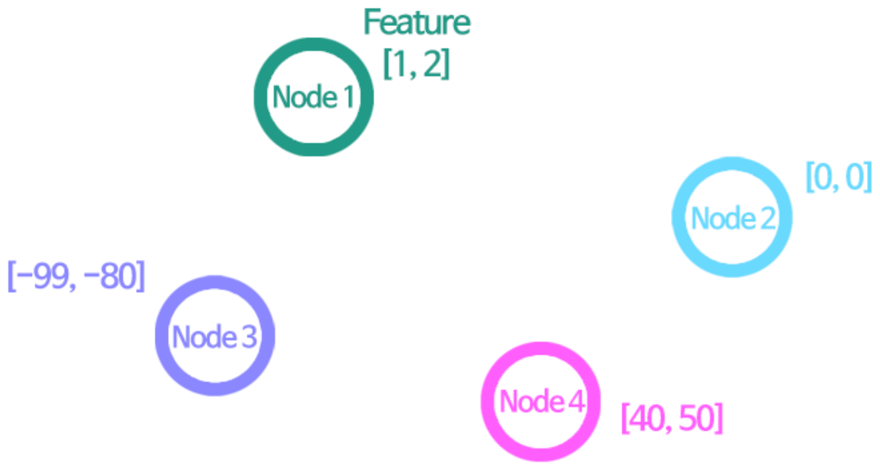
Intro



Feature

$$\begin{bmatrix} 1 & 2 \\ 0 & 0 \\ -99 & -80 \\ 40 & 50 \end{bmatrix}$$

Intro



Adjacent
(Conntection)

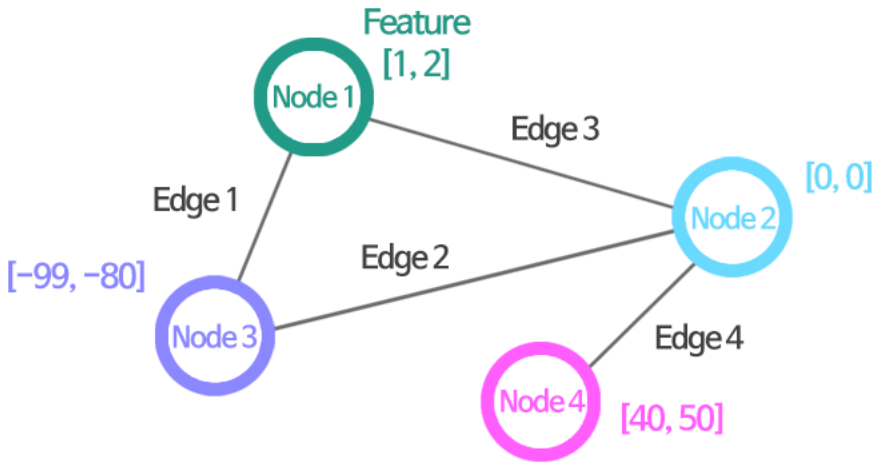
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Self-loop

Feature

$$\begin{bmatrix} 1 & 2 \\ 0 & 0 \\ -99 & -80 \\ 40 & 50 \end{bmatrix}$$

Intro



Adjacent
(Conntection)

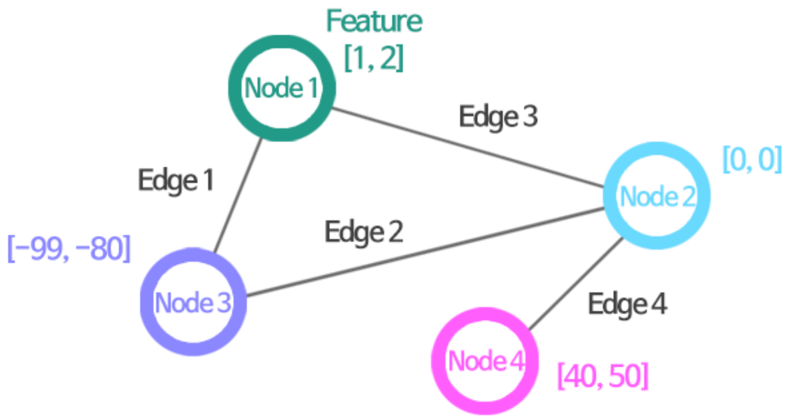
$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Feature

$$\begin{bmatrix} 1 & 2 \\ 0 & 0 \\ -99 & -80 \\ 40 & 50 \end{bmatrix}$$

Intro

$$
\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 0 & 0 \\ -99 & -80 \\ 40 & 50 \end{bmatrix}
$$



$$
= \begin{bmatrix} 1+0+(-99) & 2+0+(-80) \\ 1+0+(-99)+40 & 2+0+(-80)+50 \\ 1+0+(-99) & 2+0+(-80) \\ 0+40 & 0+50 \end{bmatrix}
$$

연결된 노드의
영향을 받는다!

Intro

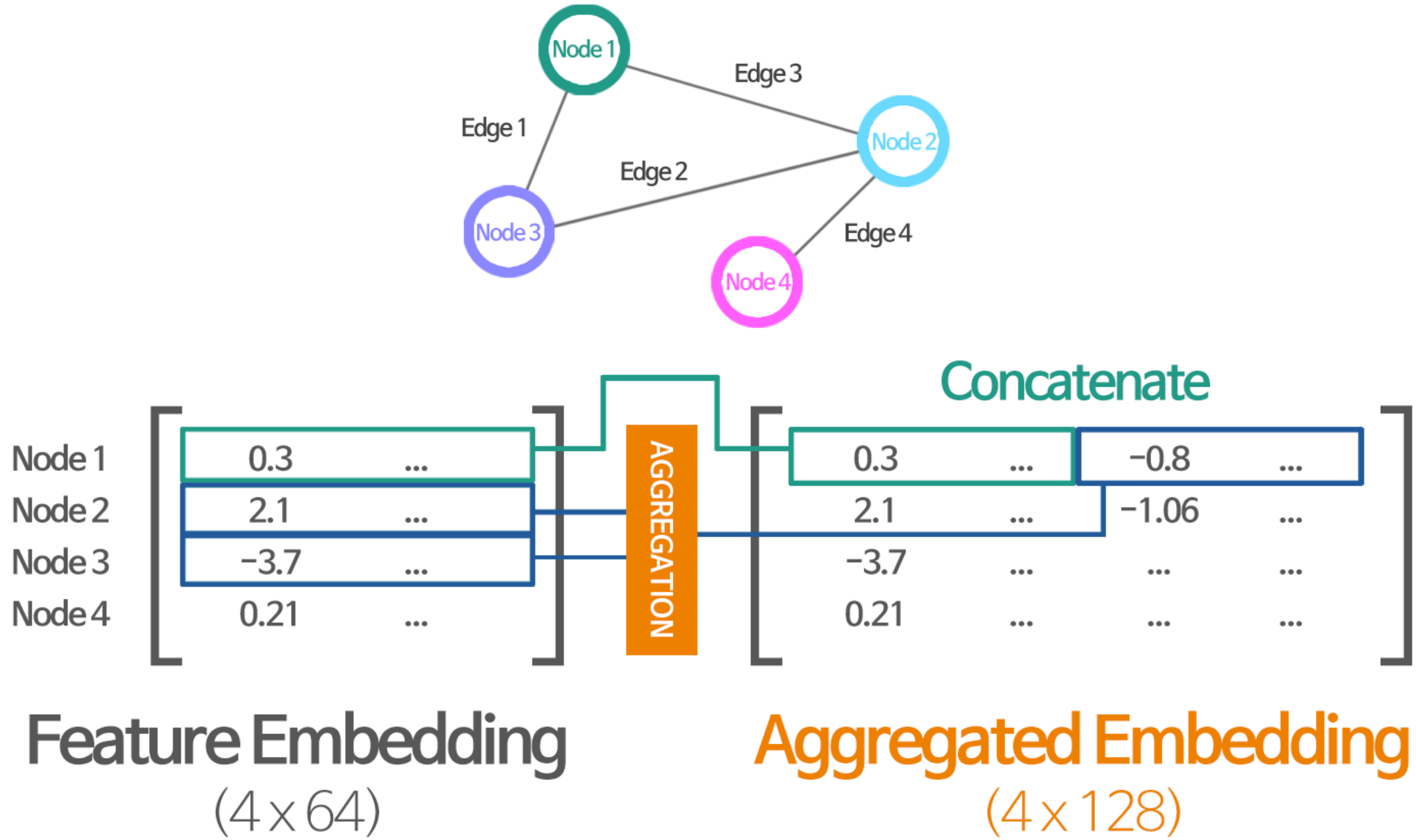
Feature Embedding
(4 x 64)

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 0 & 0 \\ -99 & -80 \\ 40 & 50 \end{bmatrix} \times \text{Weight}$$

Weight
(2 x 64)

Feature의 추상화
(Graph Convolution)

Intro



Node 1
Node 2
Node 3
Node 4
Edge 1
Edge 2
Edge 3
Edge 4
Concatenate
Node 1
Node 2
Node 3
Node 4
0.3 ...
2.1 ...
-3.7 ...
0.21 ...
AGGREGATION
0.3 ... -0.8 ...
2.1 ... -1.06 ...
-3.7 ... ...
0.21 ... ...
Feature Embedding
(4 x 64)
Aggregated Embedding
(4 x 128)

Intro

Normalization

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

Activation

Graph Convolution

Linear + ReLU　　　　Linear + ReLU

[4 x 2] → [4 x 64] → [4 x 128] → . . . → READ OUT → [1 x 512]

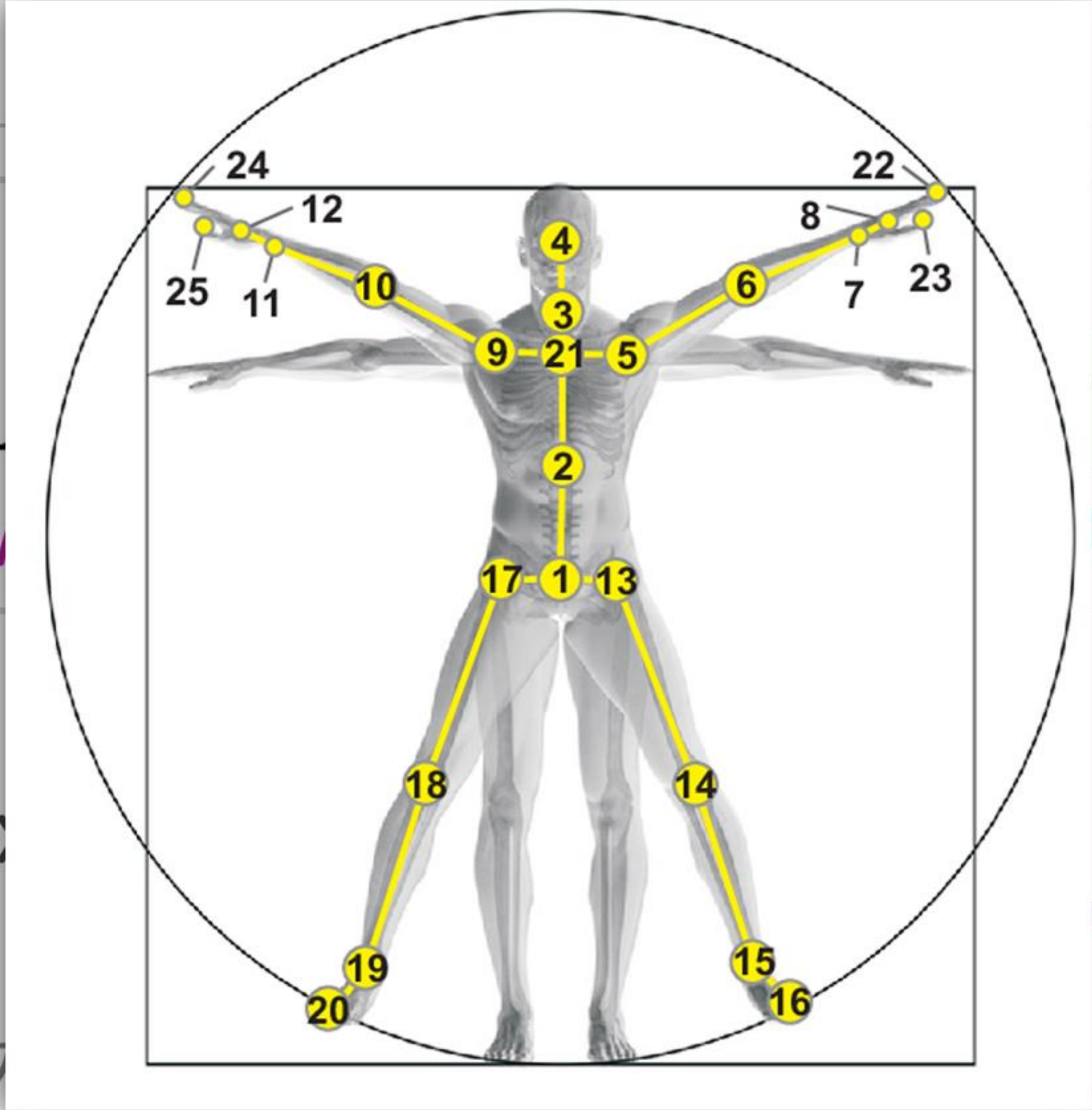
Aggregation　　　　(모든 Feature의 평균)

= 4개의 2차원 Node를 1개의 512차원 Feature로 표현!

Intro

$\psi(\tilde{A},$ ... $)$

Convolution



Linear + ReLU

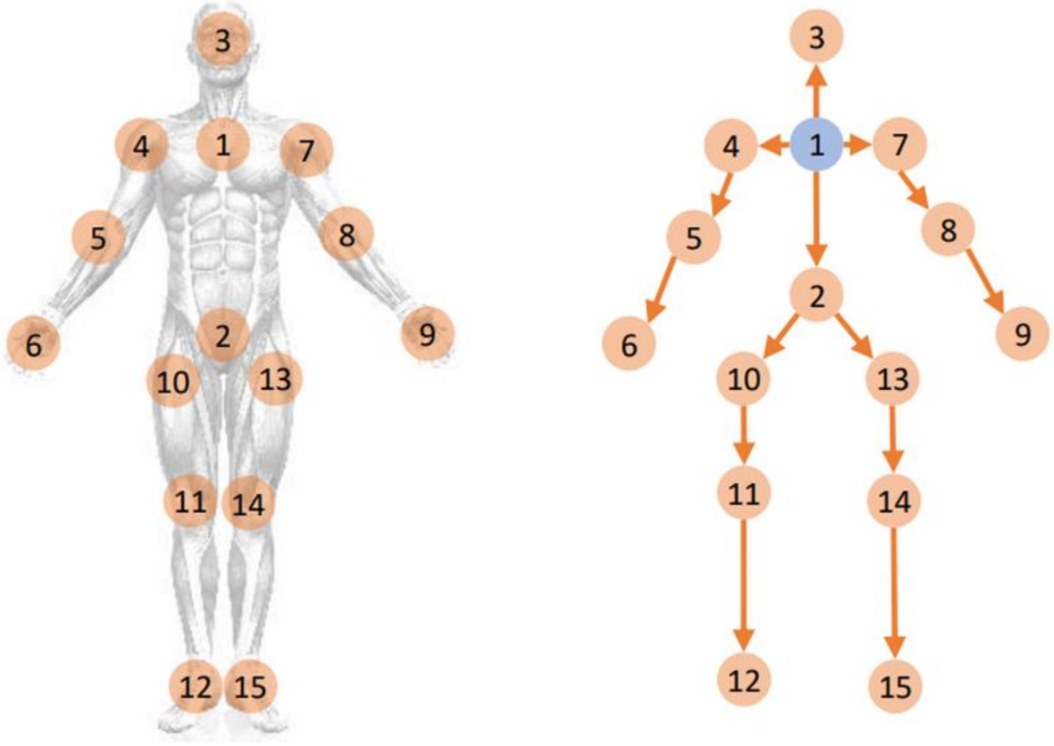
[4 x 2] → [4 ... OUT → [1 x 512]

...re의 평균)

= 4... 표현!

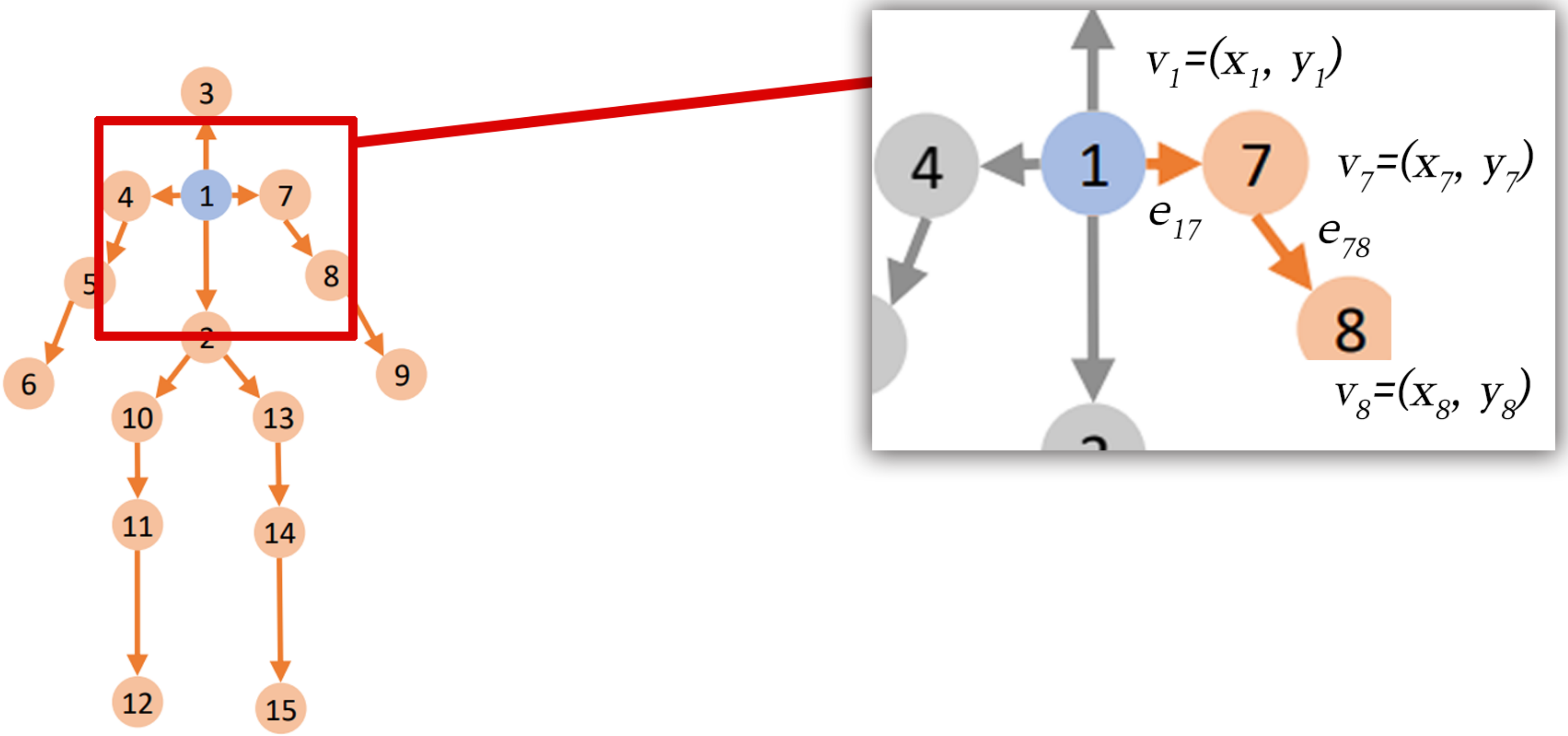
→ 25개의 Human Keypoint를 1개의 512차원 Feature로 표현!

Idea

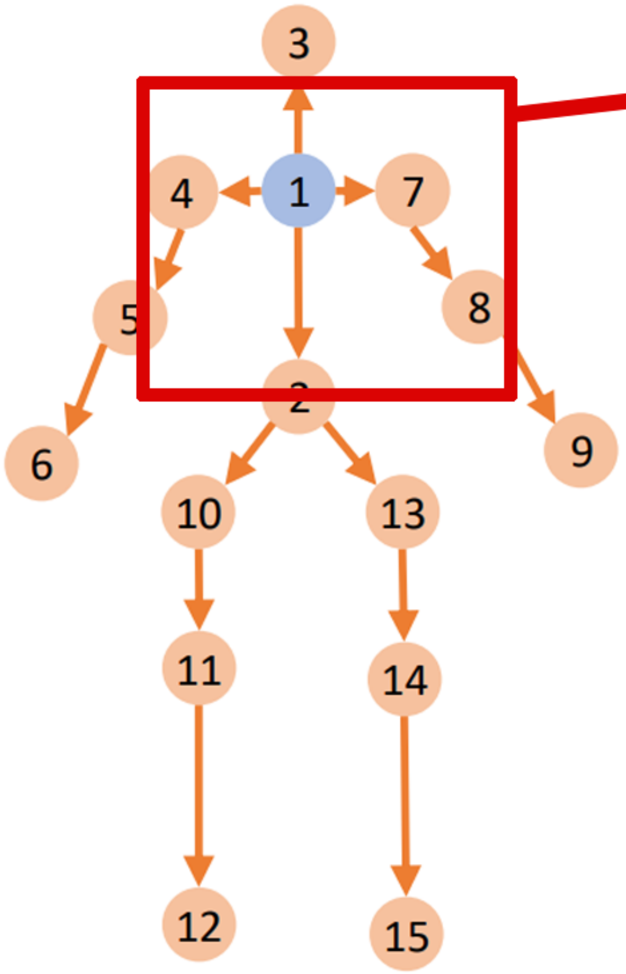


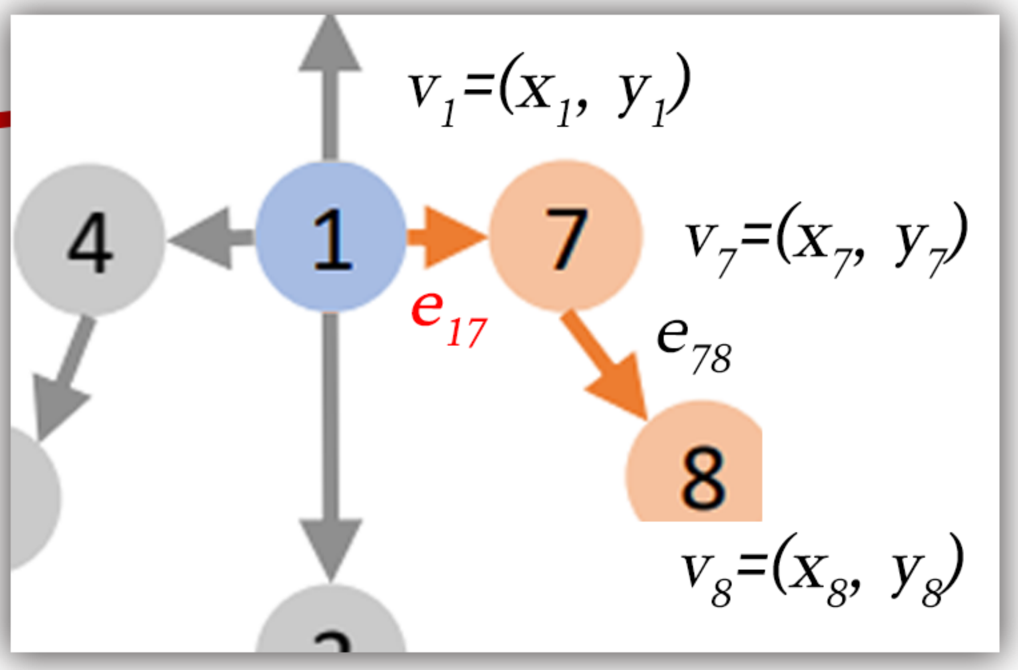
"Human Graph에 Direction을 부여해보자!"

Method



Method





$$e_{17} = (x_1 - x_7, y_1 - y_7)$$

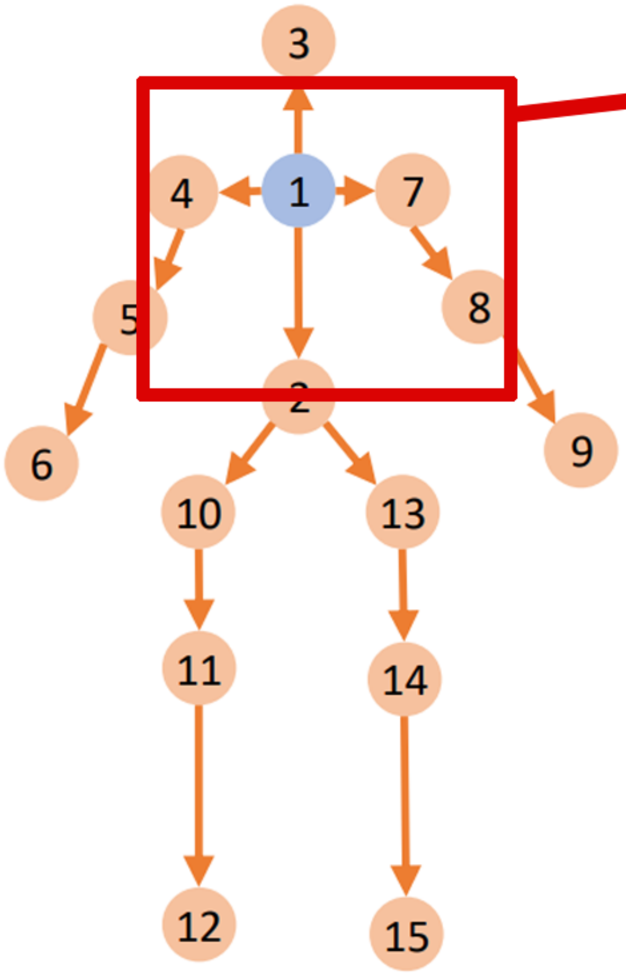
Source Vertex

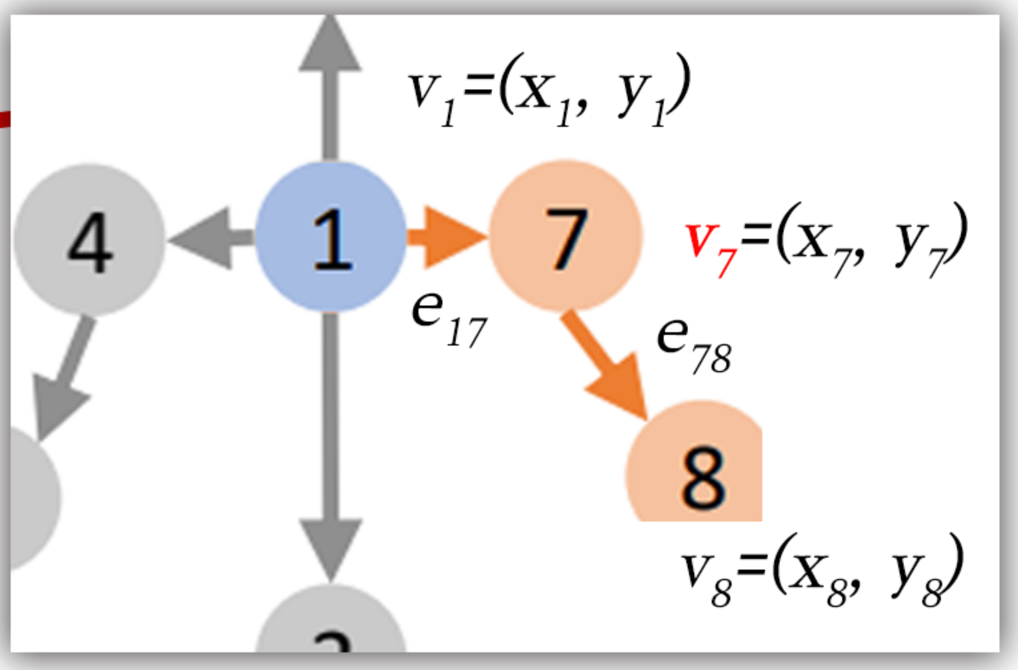
$$V_{17}^{S} = \{ v_1 \}$$

Target Vertex

$$V_{17}^{T} = \{ v_7 \}$$

Method





Incoming Edge

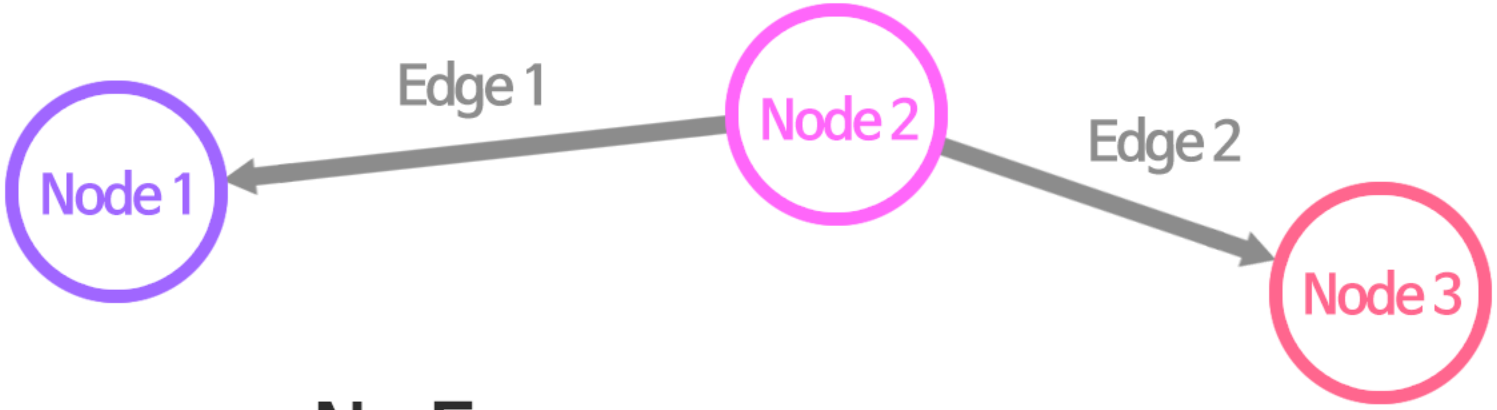
$e_7^- = \{ e_{17} \}$

Outgoing Edge

$e_7^+ = \{ e_{78} \}$

여러 개의 Edge를 포함할 수 있다. $\mathcal{E}_i^-$ $\mathcal{E}_i^+$ 로 표기

Method



$$N \times E$$

$$\begin{array}{c} \text{Node 1} \\ \text{Node 2} \\ \text{Node 3} \end{array} \begin{bmatrix} -1 & 0 \\ 1 & 1 \\ 0 & -1 \end{bmatrix} \Rightarrow \underbrace{\begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}}_{\text{Outgoing}} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_{\text{Incoming}}$$

$$\begin{array}{cc} \text{Edge 1} & \text{Edge 2} \end{array}$$

Method *(Updating Function)*

$$\begin{array}{c|c} & N \times E \\ \text{Node 1} & \\ \text{Node 2} & \\ \text{Node 3} & \end{array} \begin{bmatrix} -1 & 0 \\ 1 & 1 \\ 0 & -1 \end{bmatrix} \rightarrow \underset{\text{Outgoing}}{\begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}} \underset{\text{Incoming}}{\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}$$

Edge 1 Edge 2

Skeleton Data

= [Time x Joint x Feature]

= [TxNxC]

Edge Shape = $[T \times N_e \times C]$

Vertex Shape = $[T \times N_v \times C]$

Incidence Shape = $[N_v \times N_e]$

(Outgoing / Incoming)

Calculate Vertex

1) $[C \times T \times N_v]$ (Reshape Vertex)

2) $[C \times T \times N_e]$ (Reshape Edge)

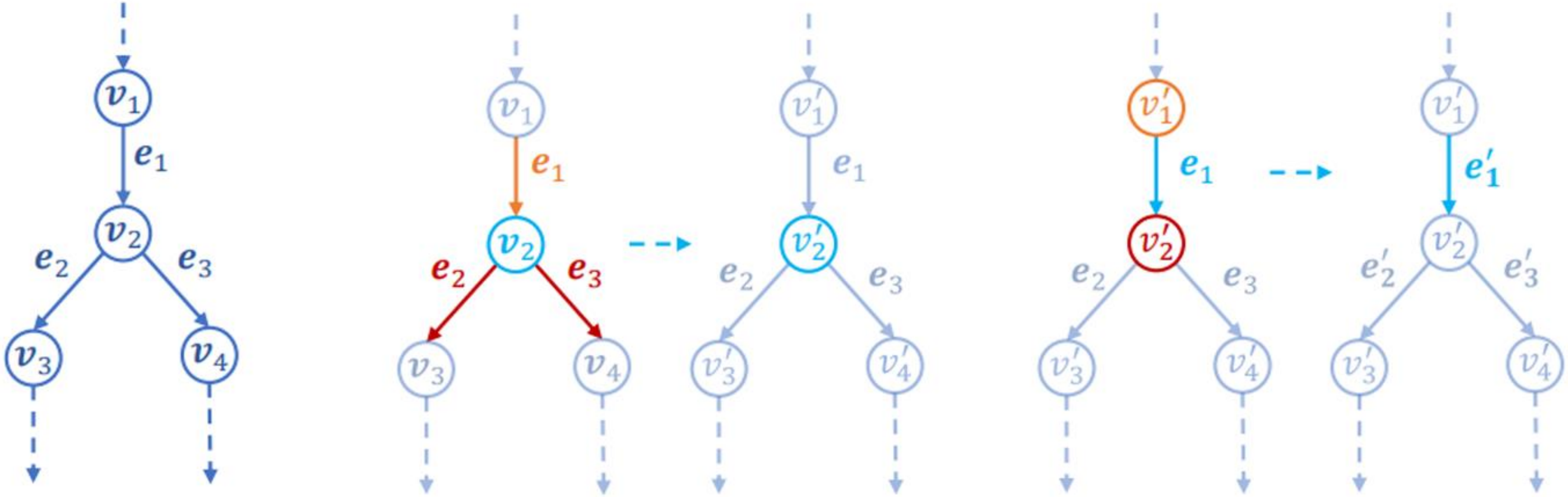
→ $[C \times T \times N_v]$ (Matmul with Outgoing^T)

3) $[C \times T \times N_e]$ (Reshape Edge)

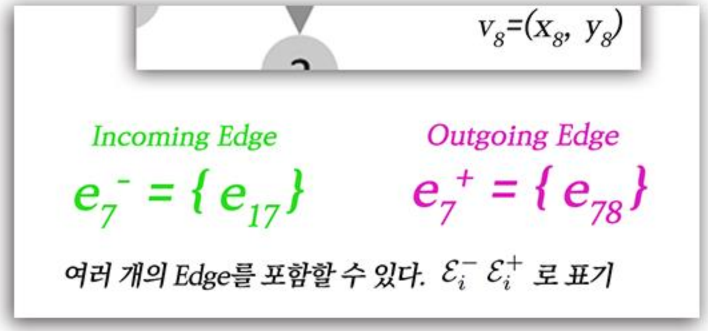
→ $[C \times T \times N_v]$ (Matmul with Incoming^T)

$$f_v' = Linear(Concat([\ 1), 2), 3)\ ]))$$

Method



(a) Original graph (b) Vertex updating (c) Edge updating



1) $\bar{\mathbf{e}}_i^- = g^{\mathbf{e}^-}(\mathcal{E}_i^-)$

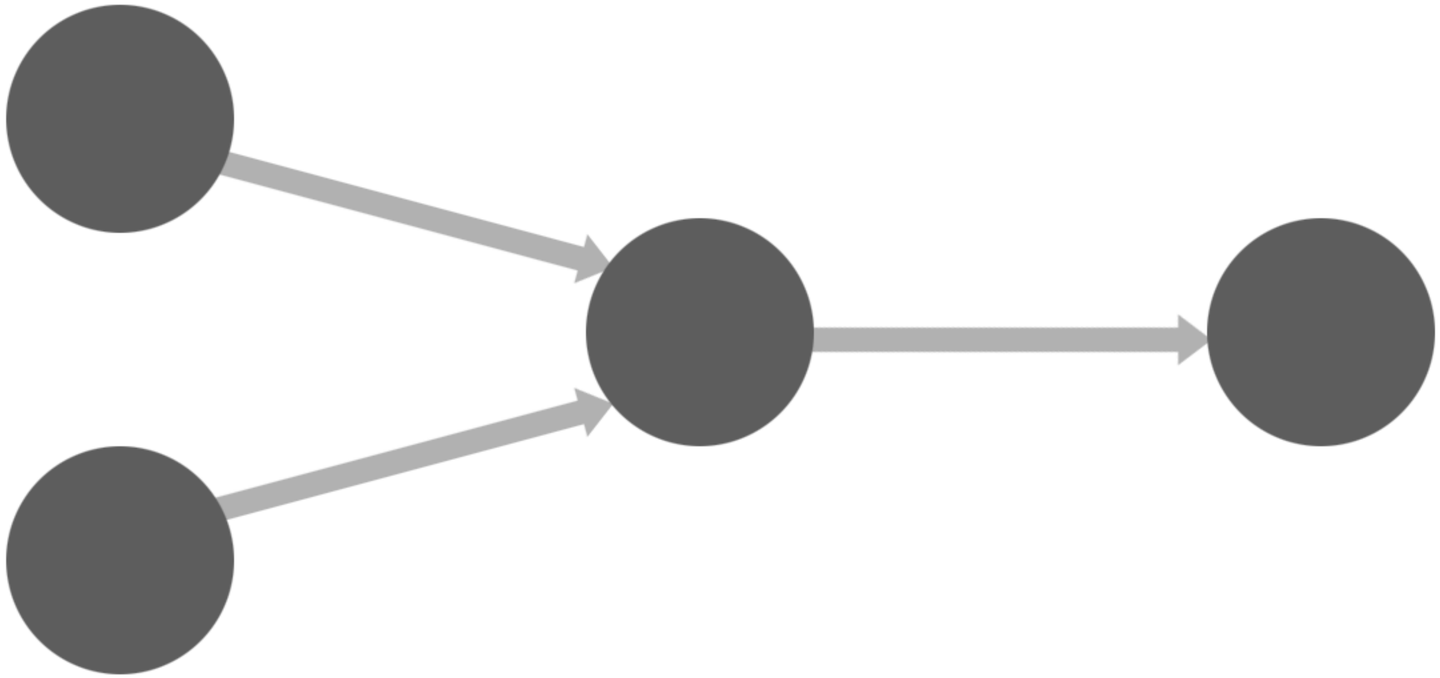
2) $\bar{\mathbf{e}}_i^+ = g^{\mathbf{e}^+}(\mathcal{E}_i^+)$

3) $\mathbf{v}'_i = h^{\mathbf{v}}([\mathbf{v}_i, \bar{\mathbf{e}}_j^-, \bar{\mathbf{e}}_j^+])$

4) $\mathbf{e}'_j = h^{\mathbf{e}}([\mathbf{e}_j, \mathbf{v}_j^{s\prime}, \mathbf{v}_j^{t\prime}])$

Aggregation: **Average Pooling**

Method



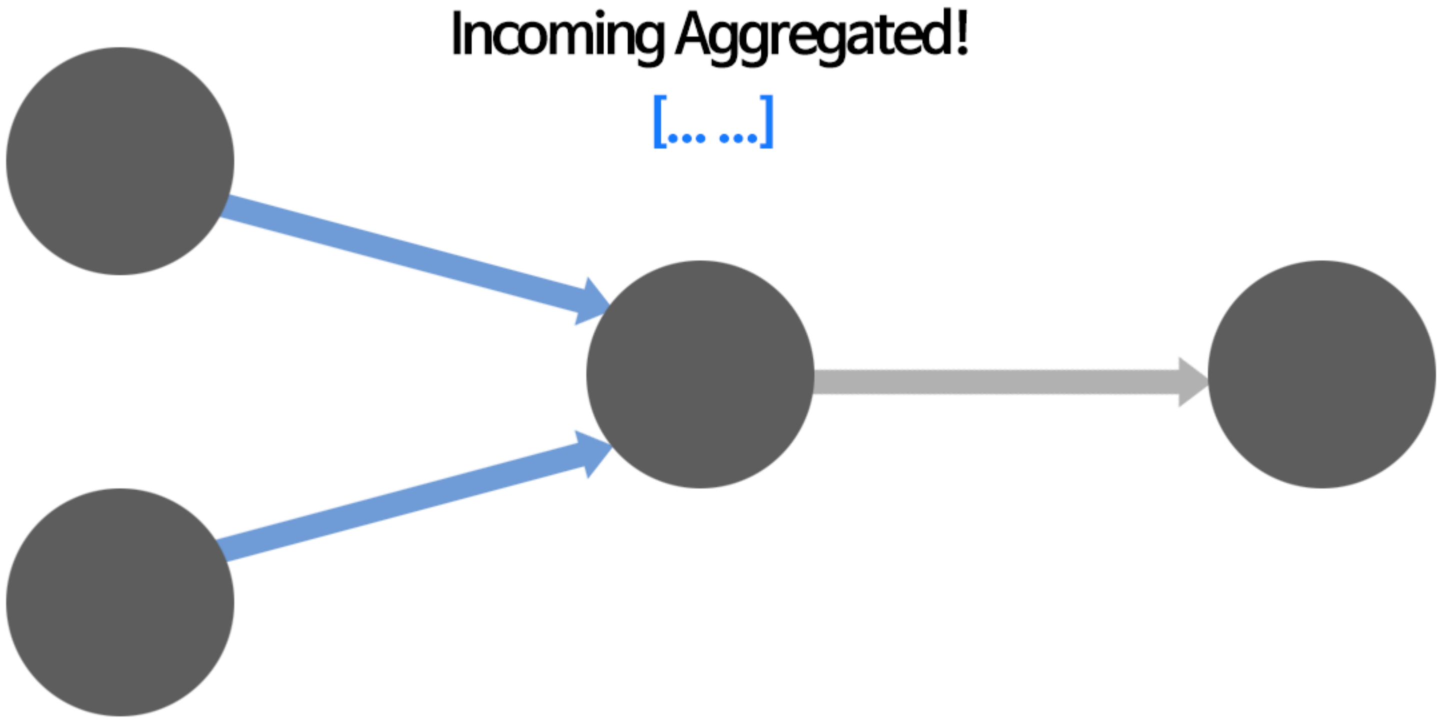
1) $\bar{\mathbf{e}}_i^- = g^{\mathbf{e}^-}(\mathcal{E}_i^-)$

2) $\bar{\mathbf{e}}_i^+ = g^{\mathbf{e}^+}(\mathcal{E}_i^+)$

3) $\mathbf{v}_i' = h^{\mathbf{v}}([\mathbf{v}_i, \bar{\mathbf{e}}_j^-, \bar{\mathbf{e}}_j^+])$

4) $\mathbf{e}_j' = h^{\mathbf{e}}([\mathbf{e}_j, \mathbf{v}_j^{s\prime}, \mathbf{v}_j^{t\prime}])$

Method



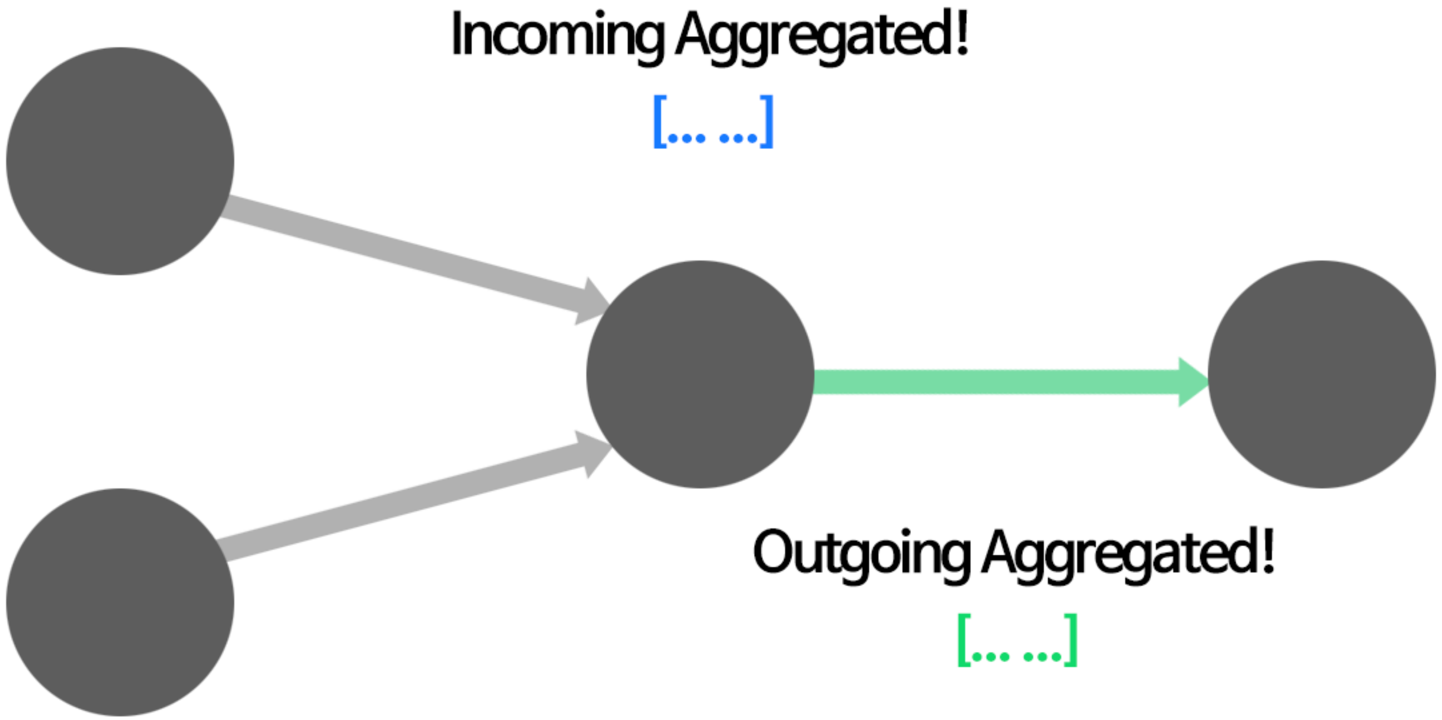
1) $\boxed{\bar{\mathbf{e}}_i^- = g^{\mathbf{e}^-}(\mathcal{E}_i^-)}$

2) $\bar{\mathbf{e}}_i^+ = g^{\mathbf{e}^+}(\mathcal{E}_i^+)$

3) $\mathbf{v}'_i = h^{\mathbf{v}}([\mathbf{v}_i, \bar{\mathbf{e}}_j^-, \bar{\mathbf{e}}_j^+])$

4) $\mathbf{e}'_j = h^{\mathbf{e}}([\mathbf{e}_j, \mathbf{v}_j^{s\prime}, \mathbf{v}_j^{t\prime}])$

Method



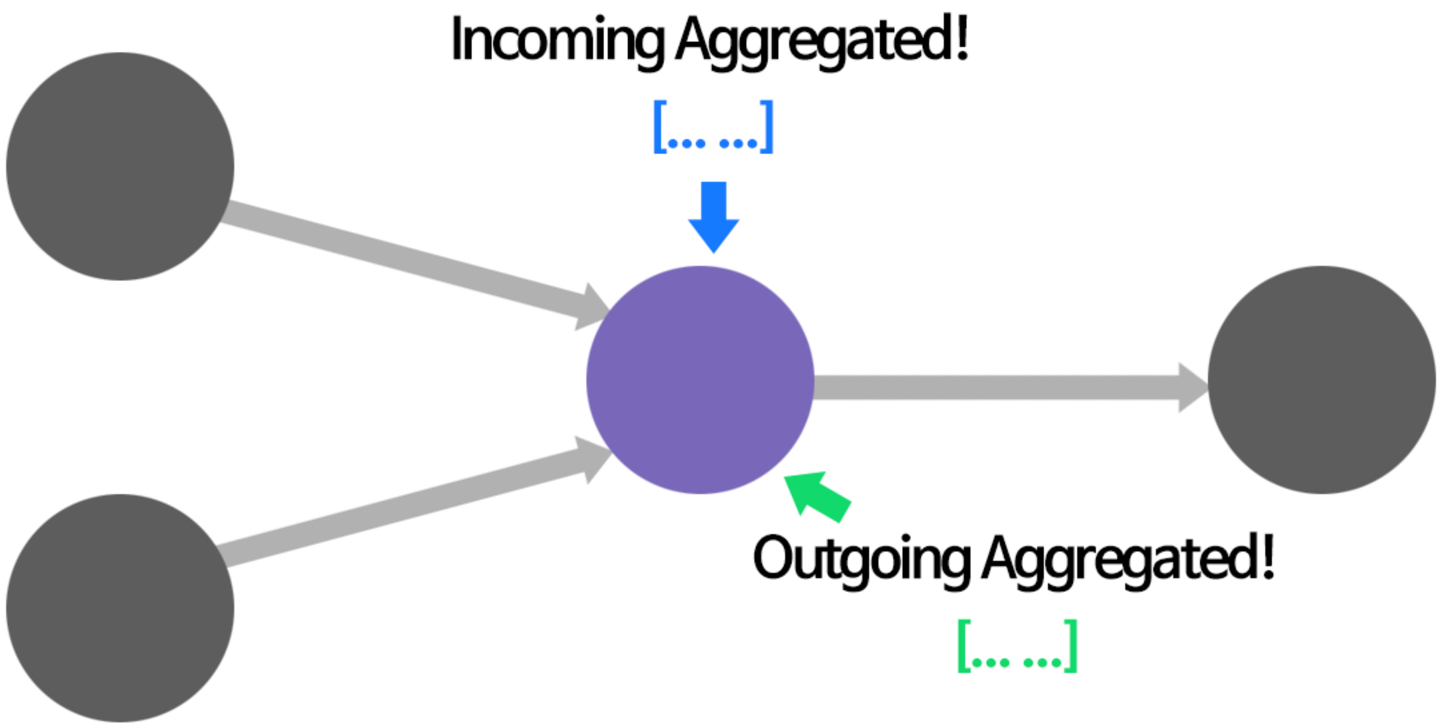
1) $\bar{\mathbf{e}}_i^- = g^{\mathbf{e}^-}(\mathcal{E}_i^-)$

2) $\boxed{\bar{\mathbf{e}}_i^+ = g^{\mathbf{e}^+}(\mathcal{E}_i^+)}$

3) $\mathbf{v}_i' = h^{\mathbf{v}}([\mathbf{v}_i, \bar{\mathbf{e}}_j^-, \bar{\mathbf{e}}_j^+])$

4) $\mathbf{e}_j' = h^{\mathbf{e}}([\mathbf{e}_j, \mathbf{v}_j^{s\prime}, \mathbf{v}_j^{t\prime}])$

Method



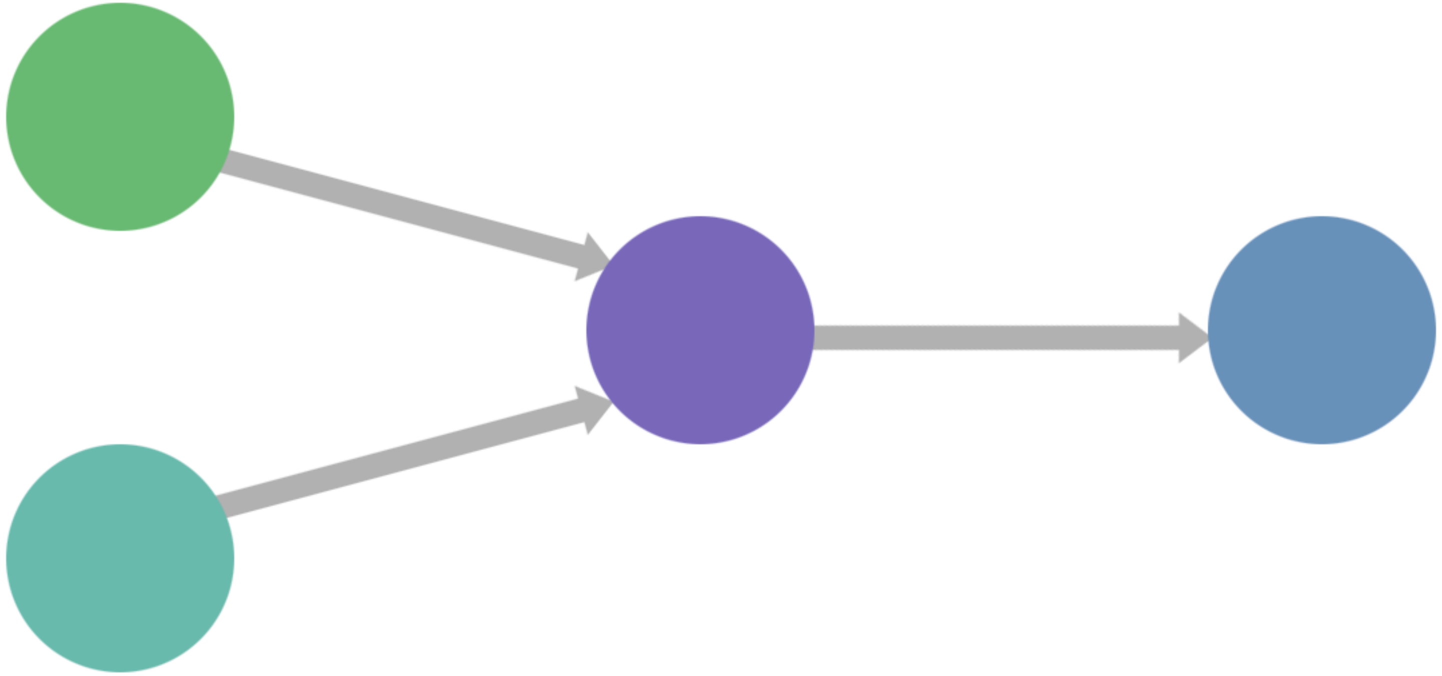
1) $\bar{\mathbf{e}}_i^- = g^{\mathbf{e}^-}(\mathcal{E}_i^-)$

2) $\bar{\mathbf{e}}_i^+ = g^{\mathbf{e}^+}(\mathcal{E}_i^+)$

3) $\boxed{\mathbf{v}_i' = h^{\mathbf{v}}([\mathbf{v}_i, \bar{\mathbf{e}}_j^-, \bar{\mathbf{e}}_j^+])}$

4) $\mathbf{e}_j' = h^{\mathbf{e}}([\mathbf{e}_j, \mathbf{v}_j^{s\prime}, \mathbf{v}_j^{t\prime}])$

Method



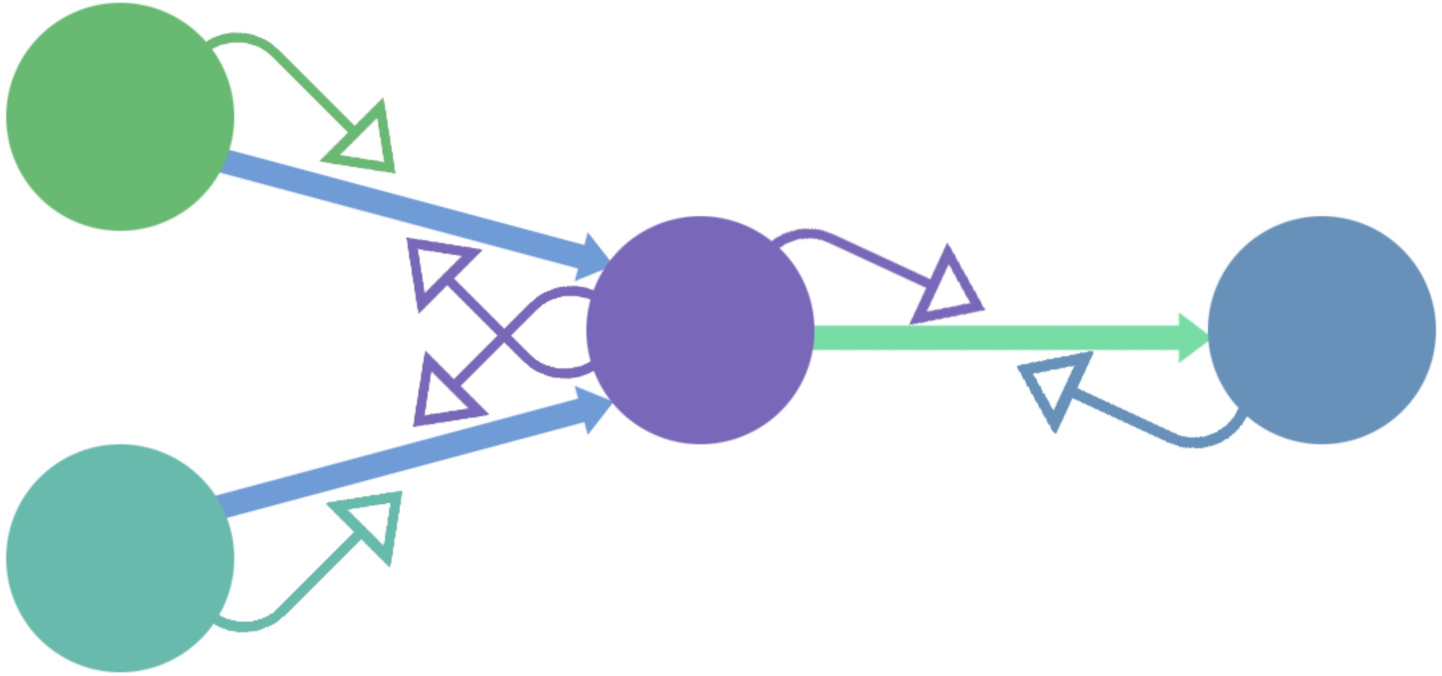
1) $\bar{\mathbf{e}}_i^- = g^{\mathbf{e}^-}(\mathcal{E}_i^-)$

2) $\bar{\mathbf{e}}_i^+ = g^{\mathbf{e}^+}(\mathcal{E}_i^+)$

3) $\boxed{\mathbf{v}_i' = h^{\mathbf{v}}([\mathbf{v}_i, \bar{\mathbf{e}}_j^-, \bar{\mathbf{e}}_j^+])}$

4) $\mathbf{e}_j' = h^{\mathbf{e}}([\mathbf{e}_j, \mathbf{v}_j^{s\prime}, \mathbf{v}_j^{t\prime}])$

Method



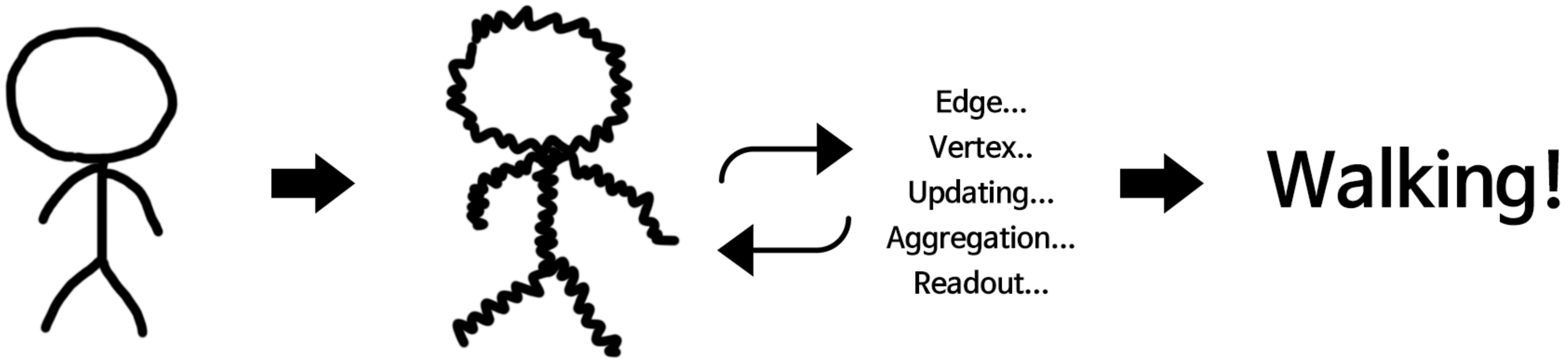
1) $\bar{\mathbf{e}}_i^- = g^{\mathbf{e}^-}(\mathcal{E}_i^-)$

2) $\bar{\mathbf{e}}_i^+ = g^{\mathbf{e}^+}(\mathcal{E}_i^+)$

3) $\mathbf{v}_i' = h^{\mathbf{v}}([\mathbf{v}_i, \bar{\mathbf{e}}_j^-, \bar{\mathbf{e}}_j^+])$

4) $\boxed{\mathbf{e}_j' = h^{\mathbf{e}}([\mathbf{e}_j, \mathbf{v}_j^{s\prime}, \mathbf{v}_j^{t\prime}])}$

Method Summary



Moreover...

Graph Construct *(Connection / Adjacent)* 의 유연성을 위해

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 \\ p_6 & p_7 & p_8 & p_9 & p_a \\ p_b & p_c & p_d & p_e & p_f \\ p_g & p_h & p_i & p_j & p_k \\ p_l & p_n & p_m & p_o & p_p \end{bmatrix}$$

A ***P***

(Trainable Parameters)

두 행렬을 활용, 새로운 Connection을 구축

Method	A	PA	$P+A$	P_0	P_{10}
Accuracy	94.4	95.0	95.3	95.2	95.5

Moreover...

시간적 정보를 추가적으로 활용하기 위해

Skeleton Data

= [Time x Joint x Feature]

= [T x N x C]

→ **[C x T x N]** (Reshape) 에 C@(9, 1) Convolution 2D 진행

= Time축을 Window Size 9로 집약시키는 효과

Moreover...

동적 정보를 활용하기 위해

Joint(Vertex)와 Bone(Edge)에 대해 Motion 정보 생성

$$\mathbf{m}_{\mathbf{v}_t} = \mathbf{v}_{t+1} - \mathbf{v}_t$$

$$\mathbf{m}_{\mathbf{e}_t} = \mathbf{e}_{t+1} - \mathbf{e}_t$$

Method	NTU(cv)	NTU(cs)	SK(t1)	SK(t5)
Spatial	95.5	89.2	36.1	58.7
Motion	93.8	86.8	31.8	54.8
Fusion	96.1	89.9	36.9	59.6

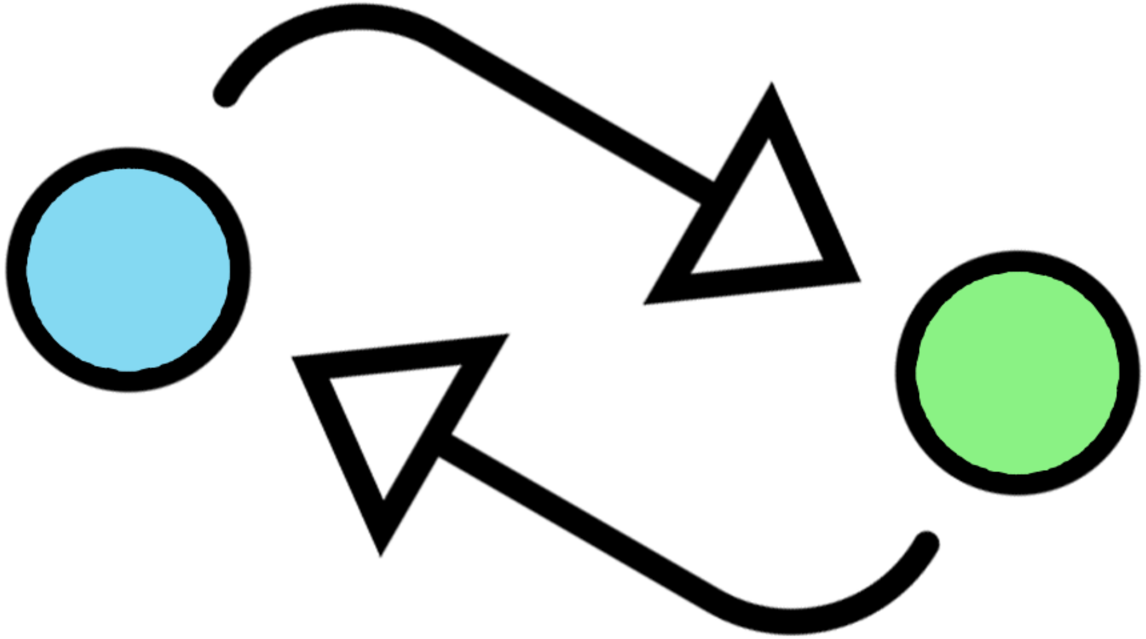
두 데이터(정확히는 넷)에 대해

학습 후 앙상블!

Result

NTU-RGBD

Method	CS	CV
Lie Group [30]	50.1	82.8
HBRNN [7]	59.1	64.0
Deep LSTM [25]	60.7	67.3
ST-LSTM [20]	69.2	77.7
STA-LSTM [28]	73.4	81.2
VA-LSTM [37]	79.2	87.7
ARRN-LSTM [18]	80.7	88.8
TCN [14]	74.3	83.1
Clips+CNN+MTLN [13]	79.6	84.8
Synthesized CNN [21]	80.0	87.2
3scale ResNet152 [16]	85.0	92.3
ST-GCN [34]	81.5	88.3
DPRL+GCNN [29]	83.5	89.8
DGNN (ours)	89.9	96.1



END