This notes not for newbie
(Must just know how but know why)

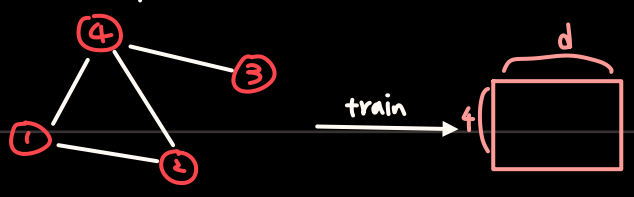
Assump you know : DL for CV (NN · CNN) · optimization (GD · SGD · mini-SGD) · LSTM · RNN · Attention

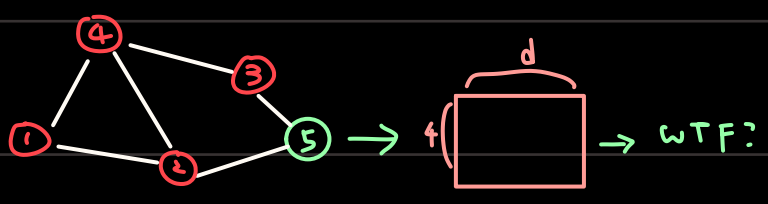
Deep Learning For Graphs

Weak of node embedding :

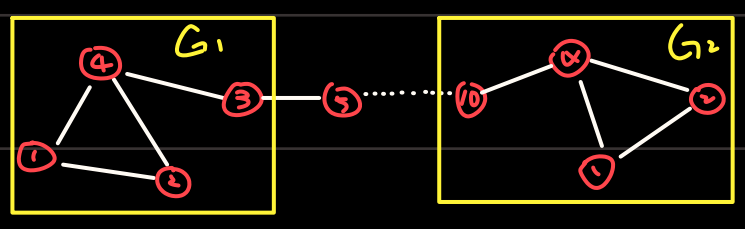
(1)

lookup-table = limitation



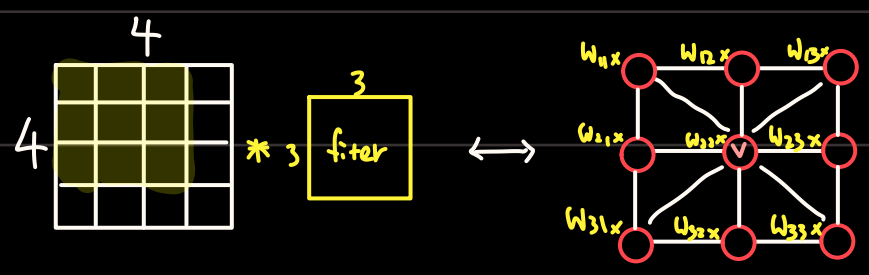


(2) Structure similarity : in space they are not close in big graph

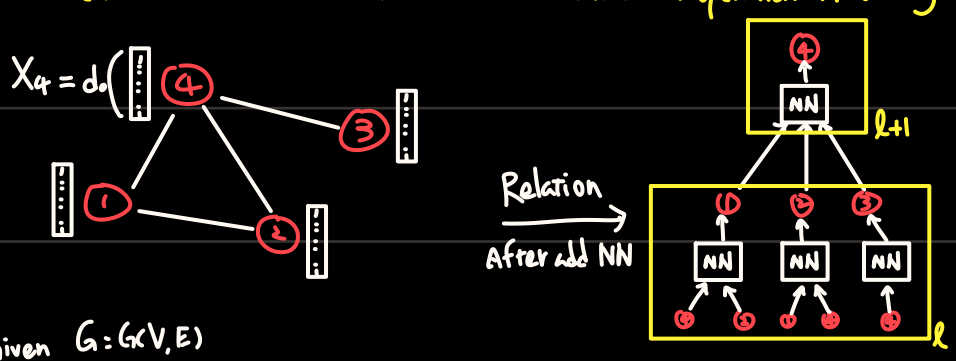


(Def) Aggregate from Neighbors (Basic Graph NN : NN4Graph)

Image is a tidy graph → CNN use pixel itself and its neighbors to transform a new pixel
(node) (3x3 filter) (feature mapping)



How about we use a shared weight Matrix to all the graph?
(Since in CNN use 1 filter to do all convolution operation in a single layer)



Given $G: G(V, E)$

(Forward) :

For each node $v \in V$ with node feature $h_v^{(0)} = x_v = [\,\cdots\,]$ do

$\{ h_v^{(l+1)} = \sigma\left(\left[W^{(l)} \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|}\right] + \left[B^{(l)} \cdot h_v^{(l)}\right]\right), \forall l \in \{0, \ldots L-1\} \}$ where shared weights : $\{W^{(l)}\}_{l=0}^{L-1}$ · $\{B^{(l)}\}_{l=0}^{L-1}$

$\rightarrow z_v = h_v^{(L)} \in R^d$ = final node embedding

(for loop will take too much time complexity to do single forward)

Matrix Formulation:

Let $H^{(l)} = [h_1^{(l)} \cdots h_{|V|}^{(l)}]^T \longrightarrow \sum_{u \in N(v)} h_u^{(l)} = A_v \cdot H^{(l)}$, where A_v is no.v row in adjacency matrix

Let D be a dingnal matrix where $D_{v,v} = Deg(v) = |N(v)| \longrightarrow D^{-1}_{v,v} = \frac{1}{|N(v)|}$

$\rightarrow \sum_{u \in N(v)} \frac{h_u^{(l)}}{N(v)} \quad \forall v \in V \Rightarrow D^{-1} \cdot A \cdot H^{(l)}$ where $D^{|V| \times |V|}$, $A^{|V| \times |V|}$, $H^{|V| \times d'}$, $d' \in \{d_0, \ldots d\}$

$\rightarrow$ $\begin{bmatrix} \frac{1}{|N(1)|} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{N(|V|)} \end{bmatrix} \left(\begin{bmatrix} Adj \in \{0,1\}^{|V| \times |V|} \end{bmatrix} \begin{bmatrix} h_1^{(l)} \\ \vdots \\ h_{|V|}^{(l)} \end{bmatrix} \right)$ Neighbors

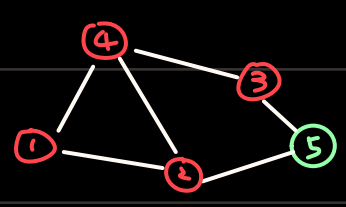
$\rightarrow h_v^{(l+1)} = \sigma(W^{(l)} \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|} + B^{(l)} \cdot h_v^{(l)}) \Rightarrow H^{(l+1)} = \sigma(D^{-1}A \cdot H^{(l)} \cdot W^{(l)^T} + H^{(l)} \cdot B^{(l)^T})$

$\rightarrow Z = \begin{pmatrix} z_1 \\ \vdots \\ z_{|V|} \end{pmatrix} = H^{(L)} \in R^{d \times |V|}$

train $\rightarrow$ Unsupervised : $\min_{W,B} \| Adj - \frac{Z^T \cdot Z}{\|Z\| \|Z\|} \|_F^2$ (row wise norm)

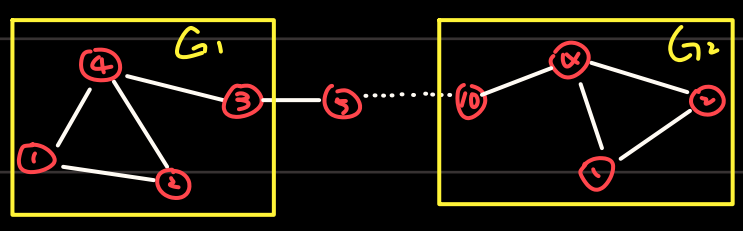
train $\rightarrow$ Supervised : Node classification (Use linear layer + sigmoid then do backprop to update weights)

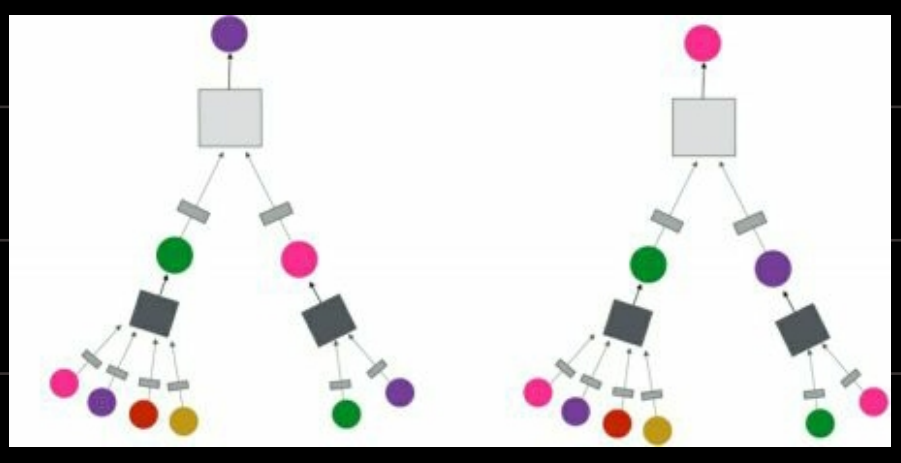
Previous limition discussion:

① 

$X_5 = [\ \ ]\ d_0 = h_5^{(0)} \rightarrow h_v^{(l+1)} = \sigma(W^{(l)} \sum_{u \in N(v)} \frac{h_u}{|N(v)|} + B^{(l)} \cdot h_v^{(l)}) \cdots \rightarrow h_5^{(L)}$ (solved)

$W^{(l)} \cdot \begin{bmatrix} h_2^{(l)} \\ h_3^{(l)} \end{bmatrix} \cdot \frac{1}{2} + B^{(l)} \cdot h_5^{(l)}$

② 



(solved)

Need to define

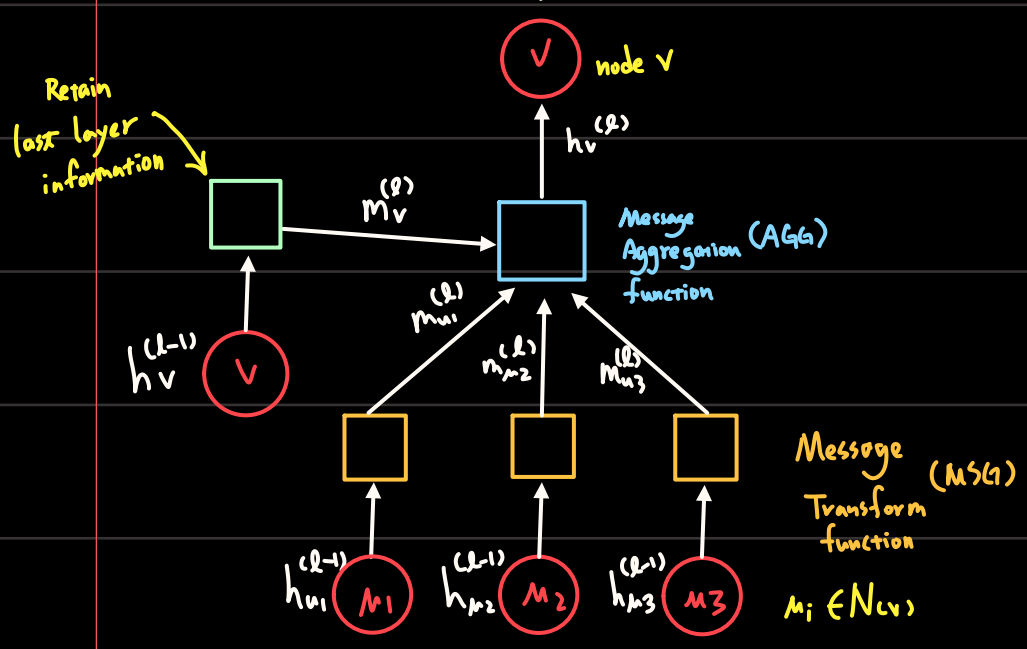
① A GNN layer = Message + Aggregation (GCN · GraphSage · GAT ... differ these 2 components)
3 classical GNN

② How layer connect ?

③ Graph augmentation · Structure manipulation

④ Task: How to train

Def Define A Single Layer = MSG + AGG (General Single Forward Formulation)



① Message Transformation

$m_u^{(l)} = MSG^{(l)}(h_u^{(l-1)})$

↓

Many Choices. e.g. $W^{(l)} \cdot h_u^{(l-1)}$ + Activation (or not)

② Message Aggregation

$h_v^{(l)} = AGG(\{m_u^{(l)}, u \in N(v)\})$ + Activation (or not)

↓

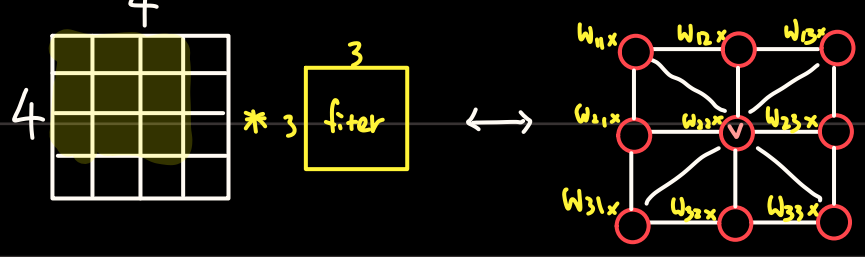
e.g. Sum · Mean · Max

Issue (similiar to ResNet · DenseNet): Add $h_v^{(l-1)}$ into message passing operation)

① $m_v^{(l)} = B^{(l)} \cdot h_v^{(l-1)}$

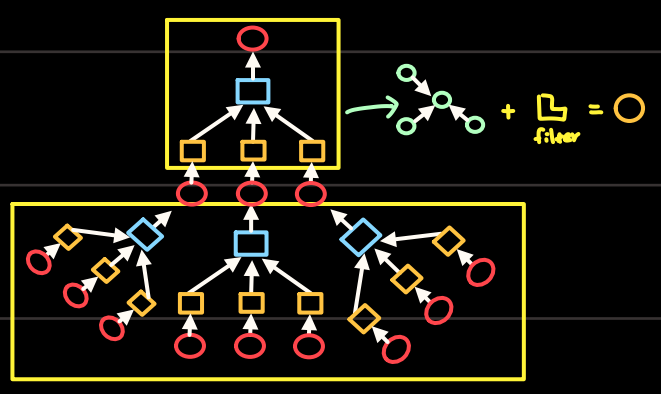
② $h_v^{(l)} = Concat(AGG(\{m_u^{(l)}, u \in N(v)\}), m_v^{(l)})$
(or Add)

Def Graph Convolutional Network (Spatial-Based Convolutional)



Origin Basic Operator: $h_v^{(l)} = \sigma(W^{(l)} \sum_{u \in N(v)} \frac{h_u^{(l-1)}}{|N(v)|} + B^{(l)} \cdot h_v^{(l-1)})$
(NN4Graph)
Mean

(Not like classic CNN)



Message + Aggregation: $h_v^{(l)} = \sigma(\sum_{u \in N(v)} W^{(l)} \frac{h_u^{(l-1)}}{|N(v)|} + B^{(l)} \cdot h_v^{(l-1)})$
AGG MSG

$\forall v \in V$: $\begin{bmatrix} h_1^{(l+1)} \\ \vdots \\ h_{N1}^{(l+1)} \end{bmatrix} = \sigma\left(\begin{bmatrix} \frac{1}{|N(v)|} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{|N(v)|} \end{bmatrix} \begin{bmatrix} W^{(l)} \end{bmatrix} \begin{bmatrix} Adj \in \{0,1\}^{n \times n} \end{bmatrix} + \begin{bmatrix} B^{(l)} \end{bmatrix}\right) \begin{bmatrix} h_1^{(l)} \\ \vdots \\ h_{N1}^{(l)} \end{bmatrix}$

MSG = Linear + Normalize Neighbors $= \frac{W^{(l)} \cdot h_u^{(l-1)}}{|N(v)|}$, $\forall u \in N(v)$

AGG = Sum + Activation $= \sigma(Sum(\{m_u^{(l)}, u \in N(v)\}))$

Def Graph SAGE (similar to DenseNet : Conv(concat([a1, a2, ...])))

$$h_v^{(l)} = \sigma\left(W^{(l)} \cdot \text{Concat}\left(\left[h_v^{(l-1)}, \text{AGG}\left(\{h_m^{(l-1)}, m \in N(v)\}\right)\right]\right)\right)$$

Stage I AGG choices :

① $\sum_{m \in N(v)} \frac{h_m^{(l-1)}}{|N(v)|}$ MSG

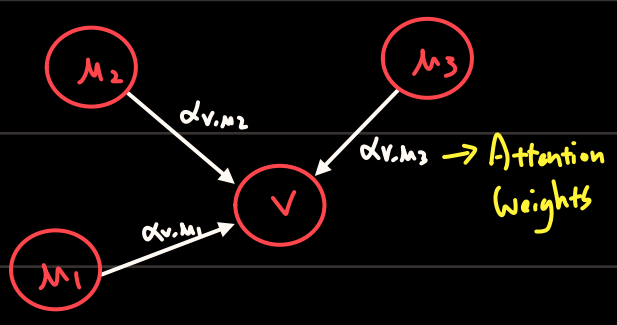
② or Max / Mean (pooling) $\left(\{\text{MSG}\ \text{MLP}(h_m^{(l-1)}), \forall m \in N(v)\}\right)$

③ Consider order : $\text{LSTM}([h_m^{(l-1)}, \forall m \in \pi(N(v))])$ No MSG

Stage II AGG : $\sigma(W^{(l)} \cdot \text{Concat}(h_v^{(l-1)}, m_{m \in N(v)}^{(l)}))$

Add l_2-Normalization when pass : $h_v^{(l)} = \frac{h_v^{(l)}}{\|h_v^{(l)}\|_2}$ $\forall v, \forall l$

Def Graph Attention Networks



Which neighbors should node v need to pay attention?

(If ignore attention $\alpha_{v,m_1} = \alpha_{v,m_2} = \alpha_{v,m_3}$)

$$h_v^{(l)} = \sigma\left(\sum_{m \in N(v)} \alpha_{v,m} W^{(l)} h_m^{(l-1)}\right) \quad \text{(Concept like weighted sum)}$$

Attention Coefficient e_{vu} :

e_{vu} = the importance of $m_m^{(l)}$ passing to v

$= a(W^{(l)} h_v^{(l-1)}, W^{(l)} h_m^{(l-1)})$, where a is a function which has many choises

a(·) choices : dot-product · concat + linear ....

Weight : $\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in N(v)} \exp(e_{vk})}$ (softmax)

Multi-Head Attention (From Transformer Structure)

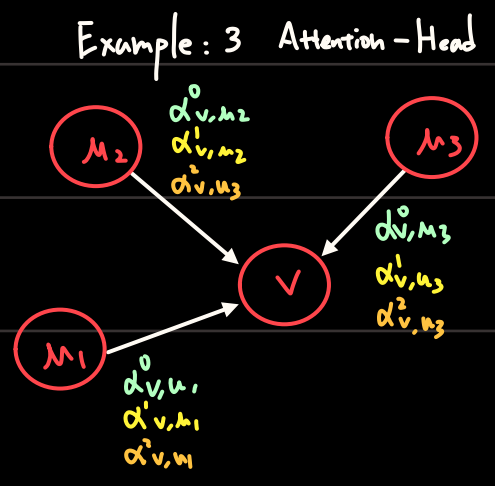
$$h_v^{(l)[0]} = \sigma\left(\sum_{m \in N(v)} \alpha_{v,m}^0 W^{(l)} h_m^{(l-1)}\right)$$

$$h_v^{(l)} = \text{AGG}\left(h_v^{(l)[1]} = \sigma\left(\sum_{m \in N(v)} \alpha_{v,m}^1 W^{(l)} h_m^{(l-1)}\right)\right)$$

Sum · Concat

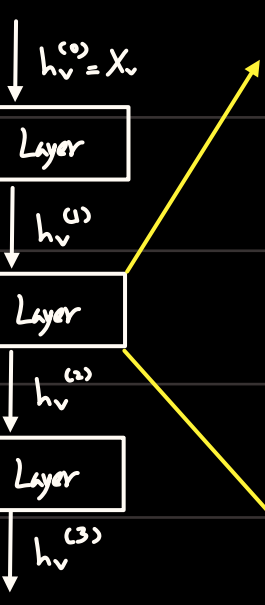
$$h_v^{(l)[2]} = \sigma\left(\sum_{m \in N(v)} \alpha_{v,u}^2 W^{(l)} h_m^{(l-1)}\right)$$

Example : 3 Attention-Head



Stacking GNN Layers

Construct Sequentially →



Layer Structure

$W^{(l)} \cdot h_u^{(l-1)} + b^{(l)}$

$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in N(v) \cup \{v\}} \exp(e_{vk})}$, $e_{vu} = a(W^{(l)} \cdot h_v^{(l-1)}, W^{(l)} \cdot h_u^{(l-1)})$

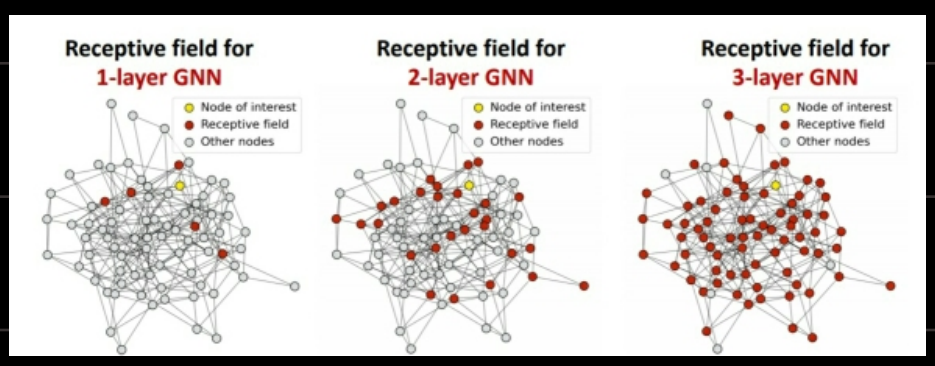
$\sum_{u \in N(v)}$, $\frac{\sum_{u \in N(v)}}{|N(v)|}$, max

The Over-Smoothing Problem: Stack too many layers → all embeddings converge to the same value (or very similar)

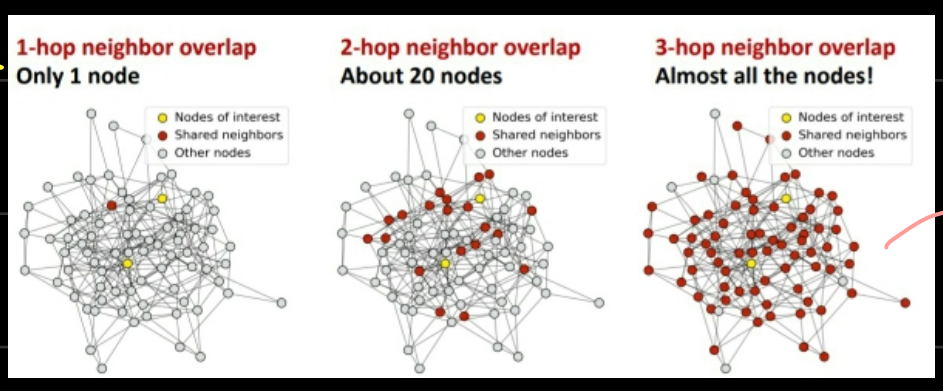
(similar to gradient vanishment) (all NN collect same information)

Receptive Field: the set of nodes that determine the embedding of a node of interest

(e.g. K-layer GNN: each node has a receptive field of K-hop neighborhood)



Compare 2 nodes



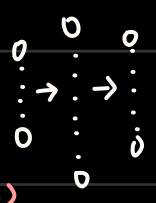
2 yellow nodes' receptive field almost the same (almost use same nodes to represent feature)

OverCome:

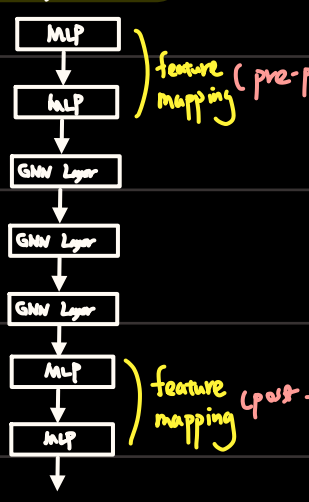
① Analyze the necessary receptive field (e.g. compute the diameter of the graph)

② Set num of GNN Layer L to be a bit more than receptive field

③ How to enchance expression with small layer GNN? → use MLP:



④ Add Layer that do not pass messages:



⑤ Residual Structure: (mixture message path)