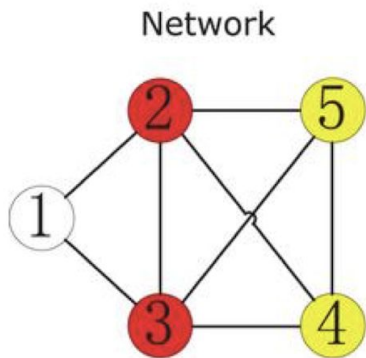


11/2

## 1. GNN - Graph

그래프란?  
Edge로  
연결된  
node들의  
집합, adj와  
feature  
matrix로  
표현됨

Vertex (Node)  
Node Feature Matrix



Adjacency matrix A

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

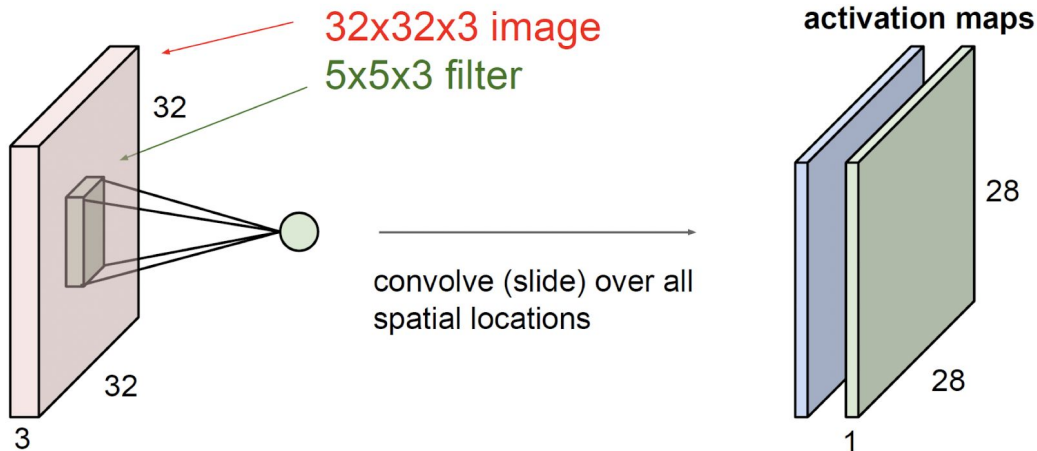
Edge  
Adjacency Matrix

Feature matrix A+I

	feature 4				
node 1	1	1	1	0	0
node 2	1	1	1	1	1
node 3	1	1	1	1	1
node 4	0	1	1	1	1
node 5	0	1	1	1	1

## 1. GNN - CNN에서의 Convolution

- 이미지에 필터를 씌우고, local의 값을 뽑아냄
- layer가 깊어질수록 더 많은, 고차원적인 정보를 뽑아낸다.
- 한 layer에서 같은 가중치를 공유하며, 필터를 슬라이드하면서 뽑아낸다.



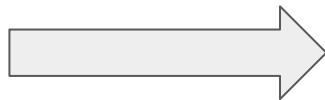
## 1. GNN - CNN

CNN의 원리가  
각 layer의 activation map의  
value를 update를 해서  
그 value가 이미지의 상태를  
결정한다.

GCN에 적용을 해보면, 노드의  
feature가 그래프의 상태를  
결정한다.

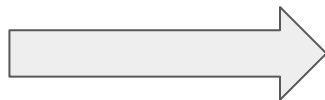
> 노드의 feature를 update를  
하면 같은 효과를 나타낼 수  
있다.

Activation map  
value update



이미지가  
무엇인지 등의  
상태를 결정

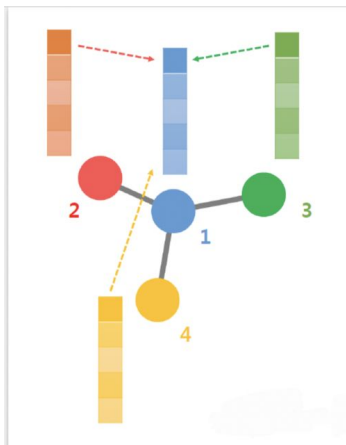
그래프의 노드  
feature



그래프의  
상태를  
결정한다는  
개념

>> 그래프의 노드 feature를 update 하면  
같은 효과를 나타낼 수 있다.

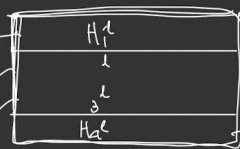
# 1. GNN - update



GNN에서 update 하는 법

$H_k^{(l)}$

+



이전 layer의 hidden state

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

같은 가중치 공유

$$\Rightarrow H_i^{(l+1)} = \sigma \left( \sum_{j \in N(i)} H_j^{(l)} W^{(l)} + b^{(l)} \right) \text{ 연결된 노드만}$$

→ 연결된 노드만 다 해주기 힘들다 → Adj 이용

# 1. GNN - update Adj matrix 이용

Adjacency Matrix A:

	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

대각이 0이면 자기 자신의 정보가 사라짐

**Feature Matrix H (n=5, feature=4):**

$H_0$
$H_1$
$H_2$
$H_3$
$H_4$

**Weight Matrix W (feature=4, filter=4):**

$w_1$	...
...	...
...	...
...	...

**Handwritten Note:** 64 (feature \* filter)

**Handwritten Note:** A·H·W 하면

1	1	1
---	---	---

feature
64

→

64
$H^{t+1}$

이전 노드와 연결된 0, 1, 4 번째 노드의 feature만 들먹 - 주변 정보 합쳐서 뽑음

**Handwritten Note:** 64 번째 low는 첫 번째 노드의 4 번째인 feature

각 노드는 4 번째 노드에 64 번째 feature를 씌움

같은 값을 공해니까 filter의 행이 같다.

새로운 layer는 근처 노드의 정보만 받아들이기

필터가 같고 근방 같은 노드를 공함 - CNN 특성

## 1. GNN - update Adj matrix 이용

결국 AHW를 하면 연결된 노드의 feature만 들어가고 주변 정보만 합쳐서 뽑는다.

update하는 과정이 convolution 연산과 같이 주변의 정보들을 이용하고, 같은 가중치를 한다는 특징이라서 GCN이라고 부른다

그리고 layer이 깊어질수록 모든 그래프를 cover할 수 있다.

## 2. GNN library

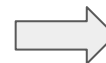
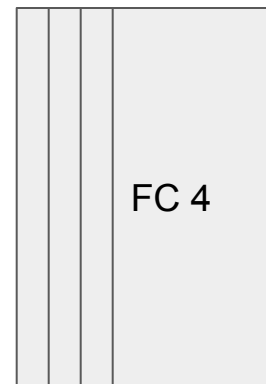
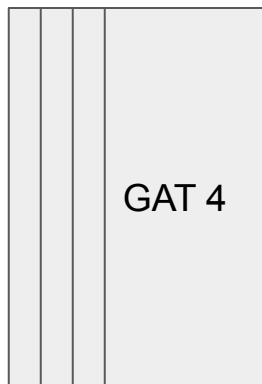
**PyG** (*PyTorch Geometric*) is a library built upon **PyTorch** to easily write and train Graph Neural Networks (GNNs) for a wide range of applications related to structured data.





## 2. GNN libray

```
GNN_Model(  
  (gconv1): ModuleList(  
    (0): GAT_gate(  
      (W): Linear(in_features=140, out_features=140, bias=True)  
      (gate): Linear(in_features=280, out_features=1, bias=True)  
      (leakyrelu): LeakyReLU(negative_slope=0.2)  
    )  
    (1): GAT_gate(  
      (W): Linear(in_features=140, out_features=140, bias=True)  
      (gate): Linear(in_features=280, out_features=1, bias=True)  
      (leakyrelu): LeakyReLU(negative_slope=0.2)  
    )  
    (2): GAT_gate(  
      (W): Linear(in_features=140, out_features=140, bias=True)  
      (gate): Linear(in_features=280, out_features=1, bias=True)  
      (leakyrelu): LeakyReLU(negative_slope=0.2)  
    )  
    (3): GAT_gate(  
      (W): Linear(in_features=140, out_features=140, bias=True)  
      (gate): Linear(in_features=280, out_features=1, bias=True)  
      (leakyrelu): LeakyReLU(negative_slope=0.2)  
    )  
  )  
  (FC): ModuleList(  
    (0): Linear(in_features=140, out_features=128, bias=True)  
    (1): Linear(in_features=128, out_features=128, bias=True)  
    (2): Linear(in_features=128, out_features=128, bias=True)  
    (3): Linear(in_features=128, out_features=1, bias=True)  
  )  
  (embede): Linear(in_features=56, out_features=140, bias=False)  
)
```



to classify whether the protein complex model is correct or incorrect

### 3. GNN\_DOVE 메일

Dear Jin,

Thank you so much for your interest in our work GNN-DOVE! I am happy to hear that you have tried it and successfully run it locally.

For training, the same questions have been widely asked by the users. Therefore, I have attached detailed instructions at [https://github.com/kiharalab/GNN\\_DOVE/issues/2](https://github.com/kiharalab/GNN_DOVE/issues/2) and [https://github.com/kiharalab/GNN\\_DOVE/issues/4](https://github.com/kiharalab/GNN_DOVE/issues/4). I hope that can help you fully understand the training process of GNN-DOVE.

If you have any further questions, feel free to email me.

Best,  
Xiao Wang

[https://github.com/kiharalab/GNN\\_DOVE/issues/2](https://github.com/kiharalab/GNN_DOVE/issues/2)