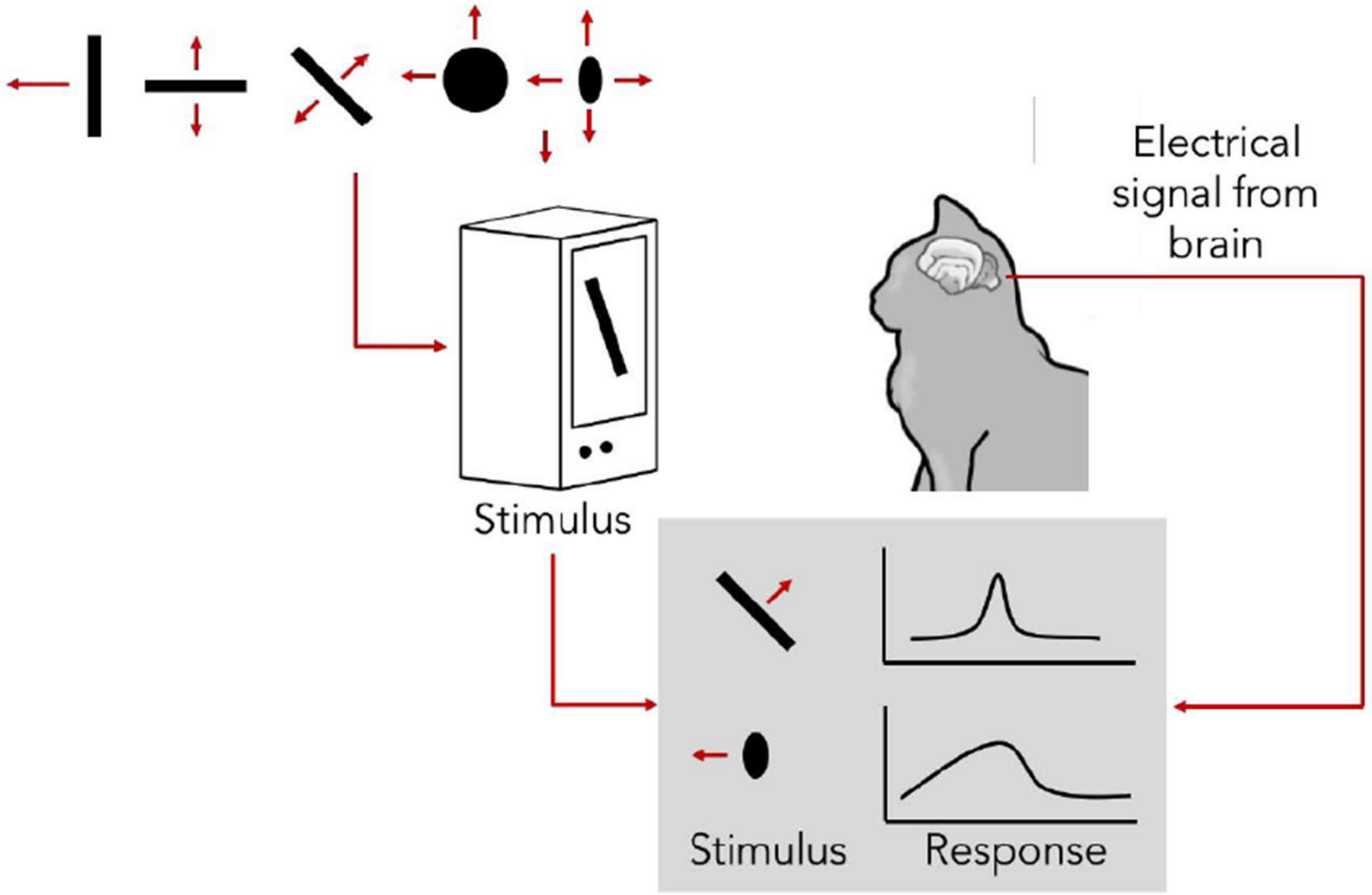
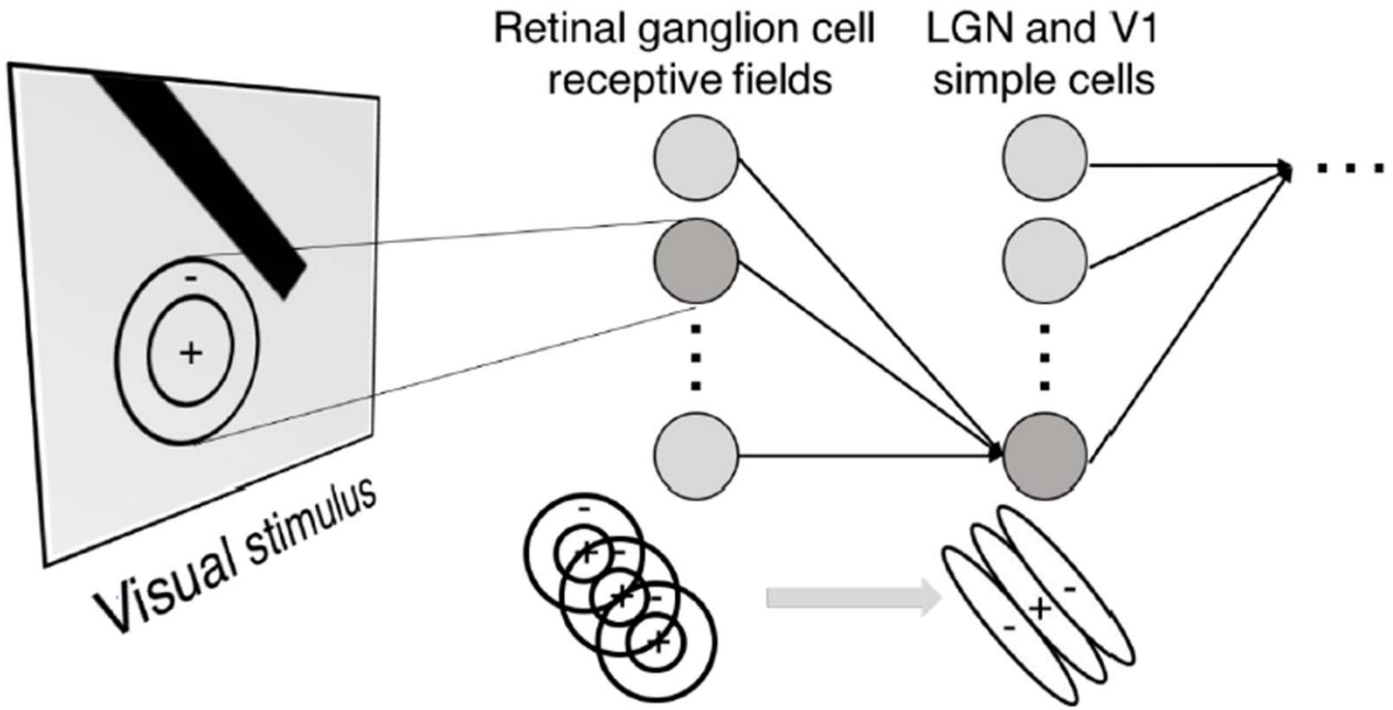
Convolutional neural network

How human recognize an image?



Stanford, cs231n 2017

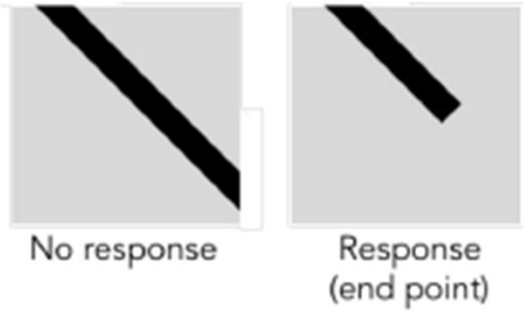
Hierarchical organization



-Simple cells:
Response to light orientation

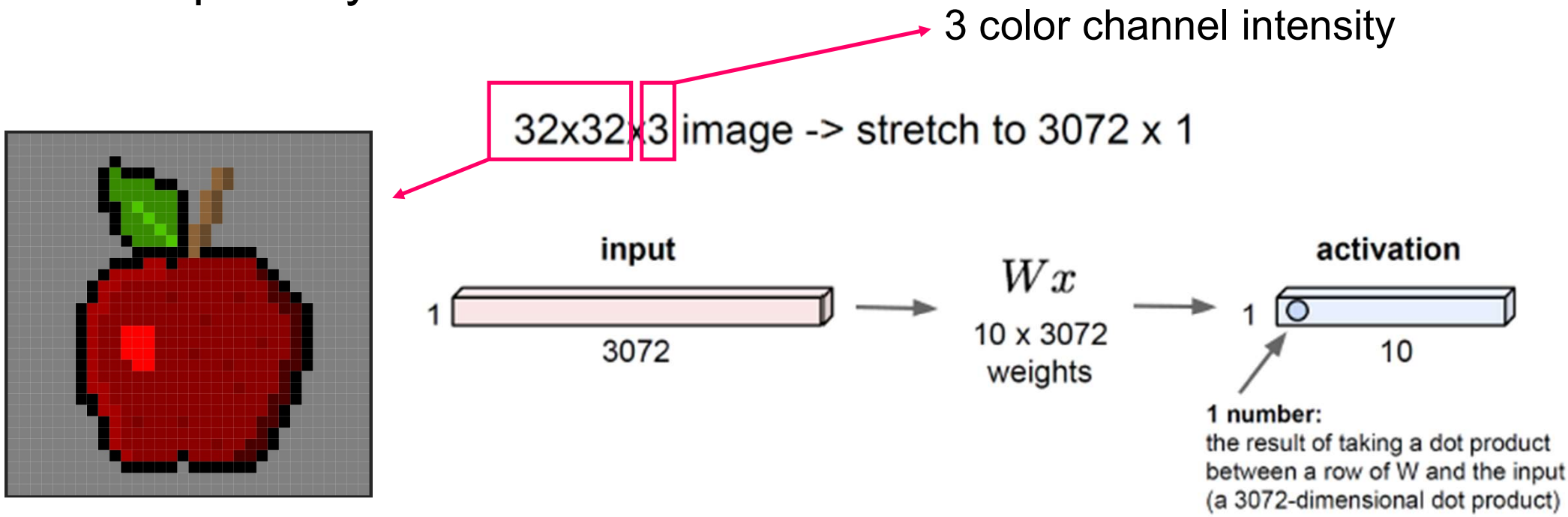
-Complex cells:
Orientation & movement

-Hypercomplex cells:
Response to movement with an end point



-Each neuron has a different role

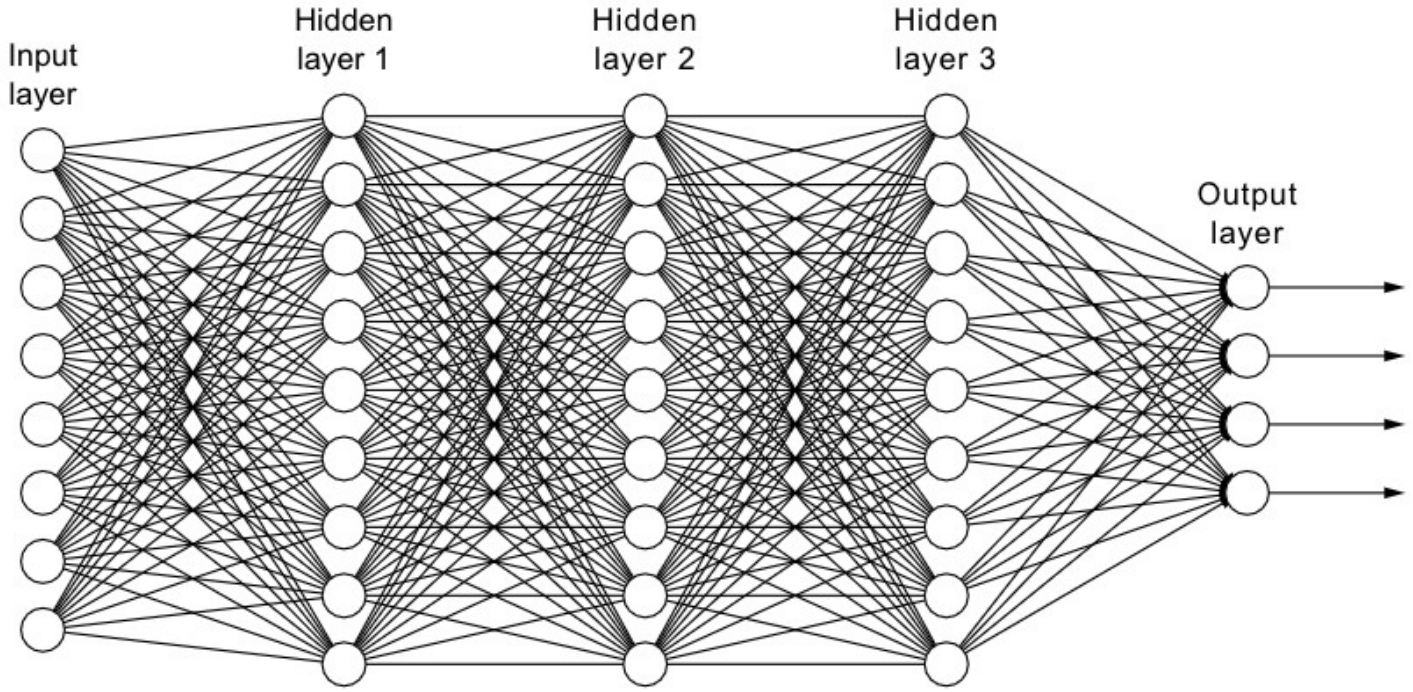
MLP / simple fully connected later



-10 filter

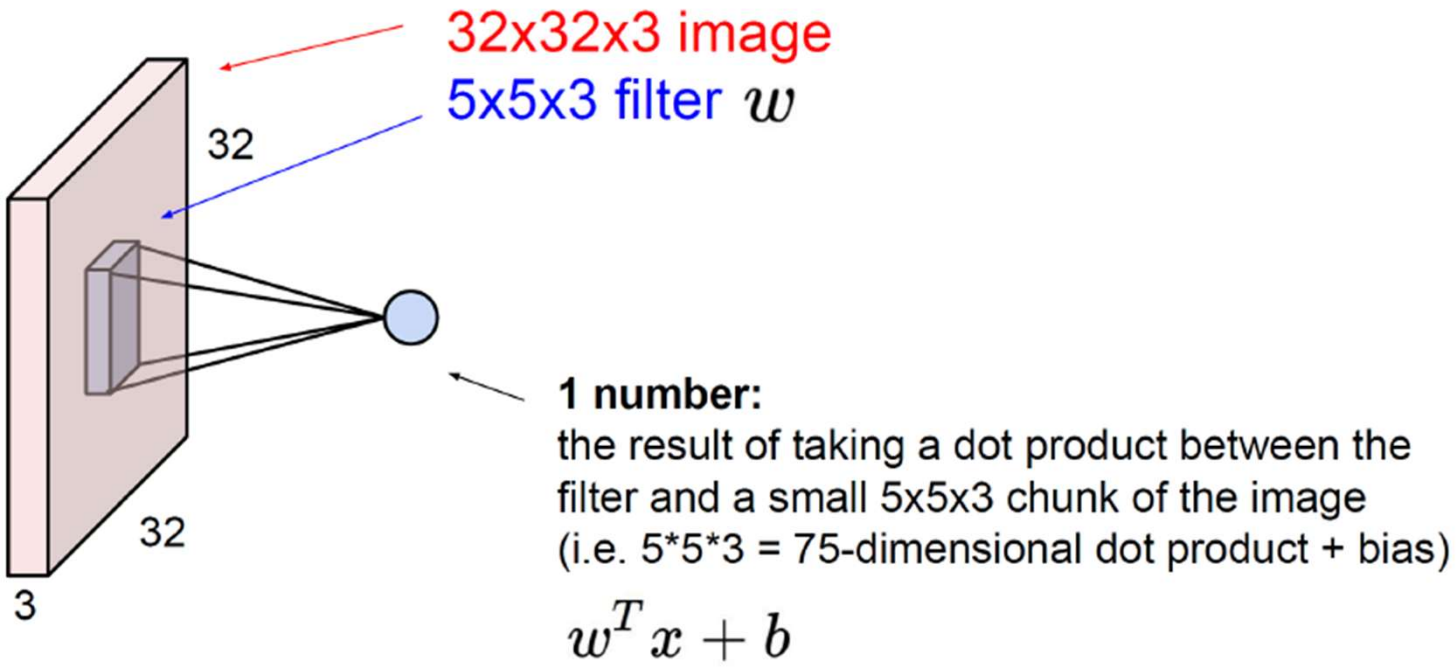
→ 10 row * 1-3072 → dot product → 1-10 activation layer

Explosion of model parameters

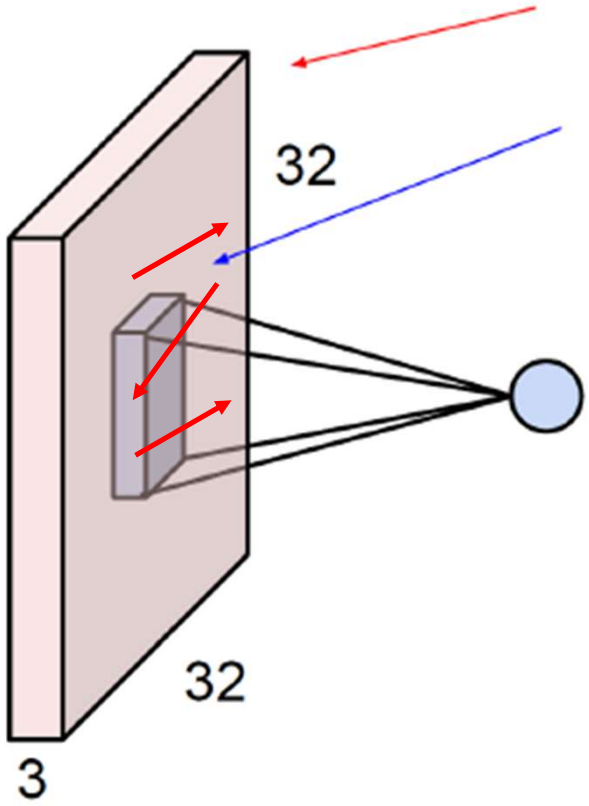


-Layer1: Node [n] → Layer2: Node [m] (all pairwise) → n*m
-Multiple layers → n*n*n*n …
→ Too many parameters to learn
→ Even some of them might be meaningless

Convolutional neural network

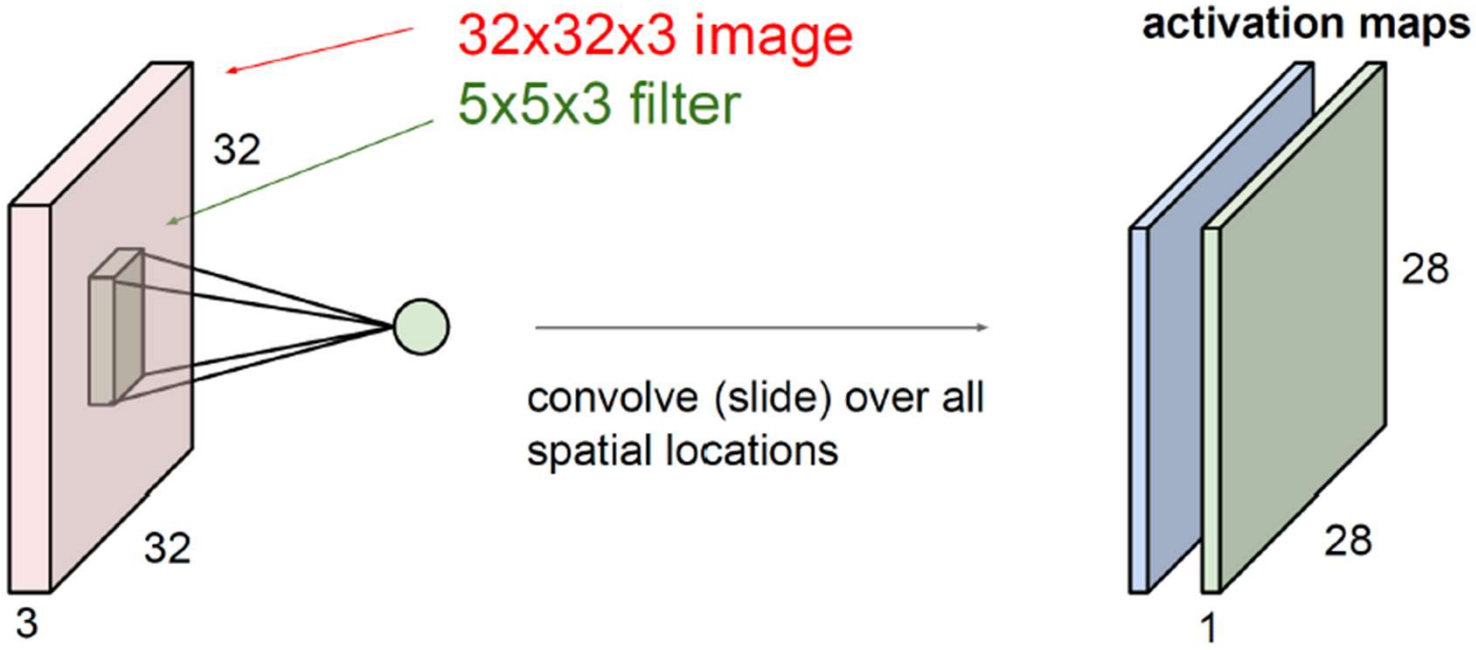


Convolutional neural network



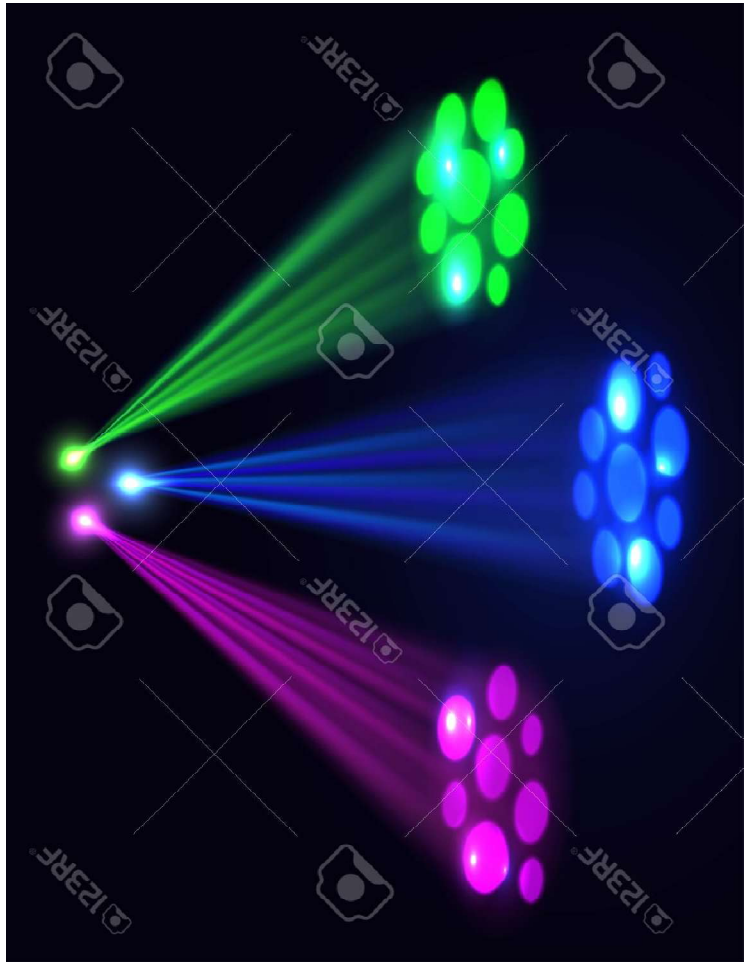
-Chunk filter Sliding → compress the information

Convolutional neural network



-32 X 32 → 28 X 28 (1 activation map)

Convolutional neural network

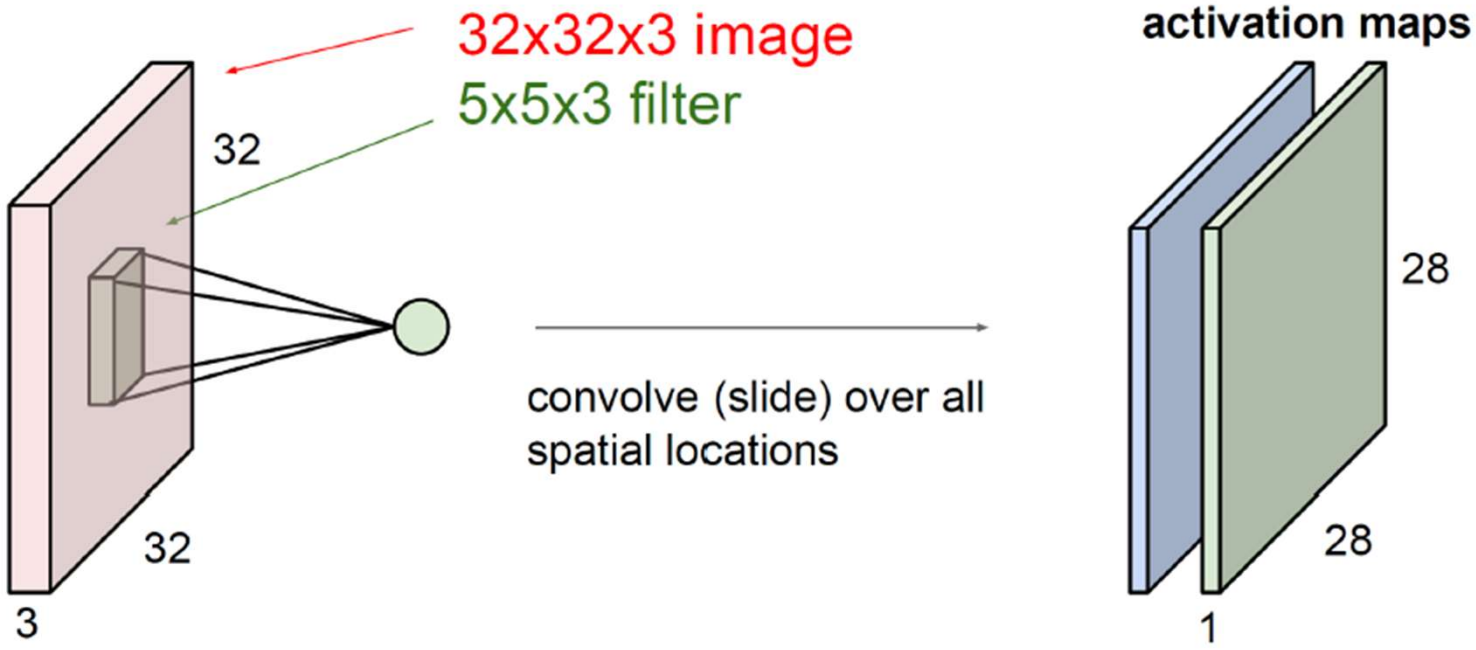


→ Only green neuron can be activated

→ Only blue neuron can be activated

→ Only magenta neuron can be activated

Convolutional neural network

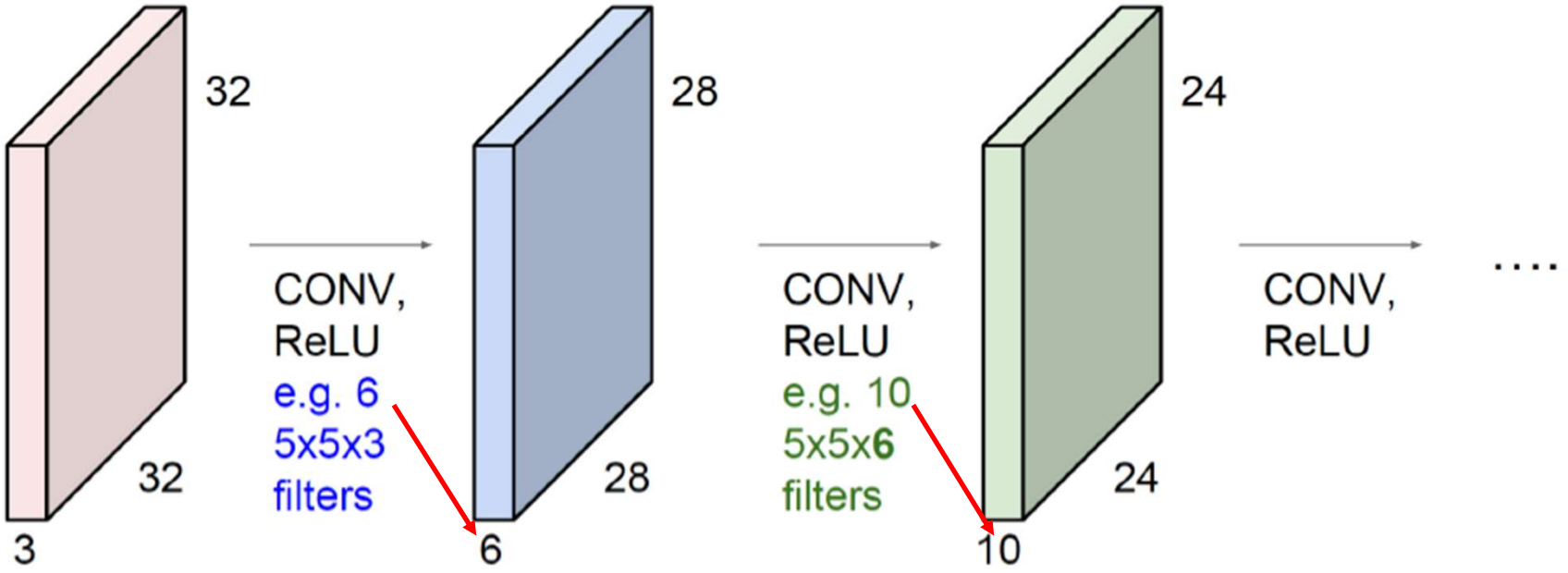


-Artificial filter → so that only some of the neuron can pass the information

Ex: blue filter → only blue neuron can pass the information → 1 activation map

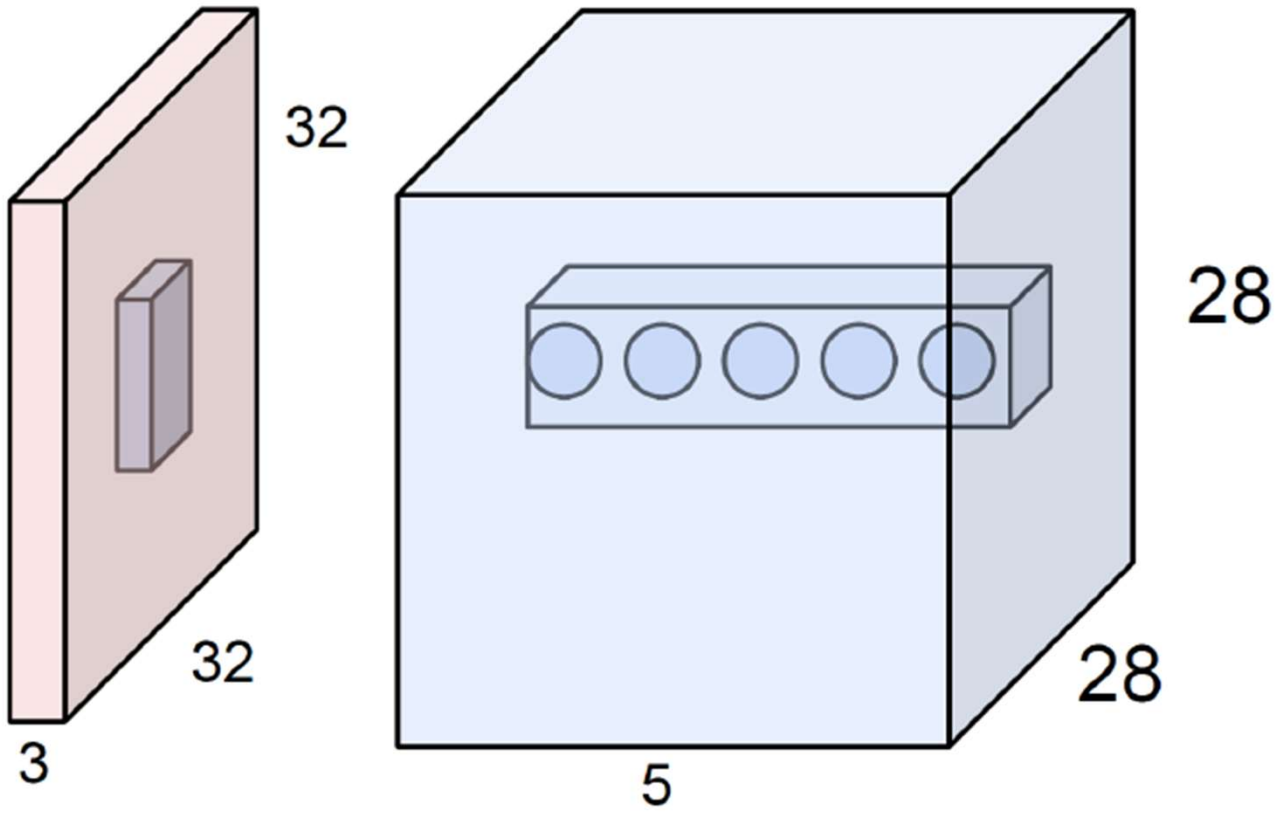
green filter → …

Convolutional neural network

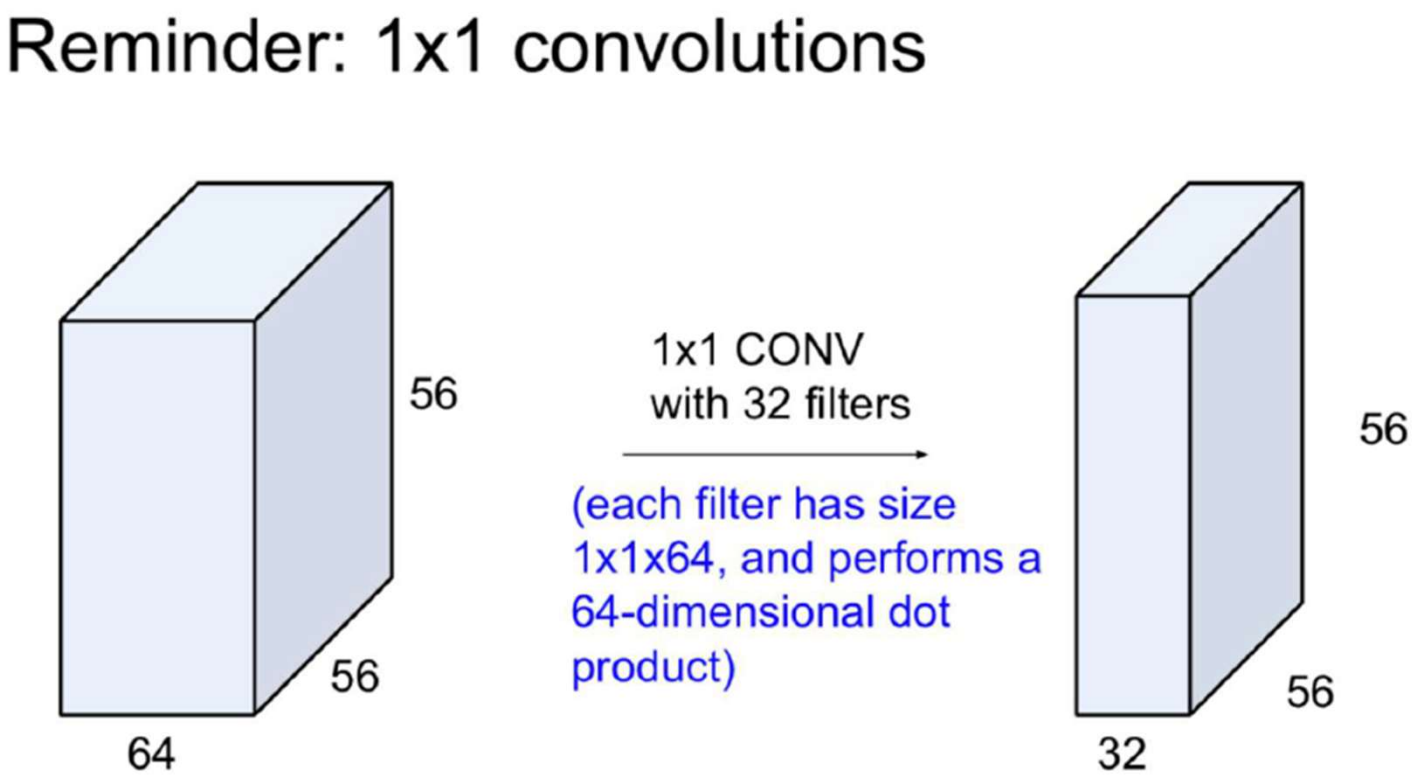


-# of activation map → number of x in the next layer

Convolutional neural network



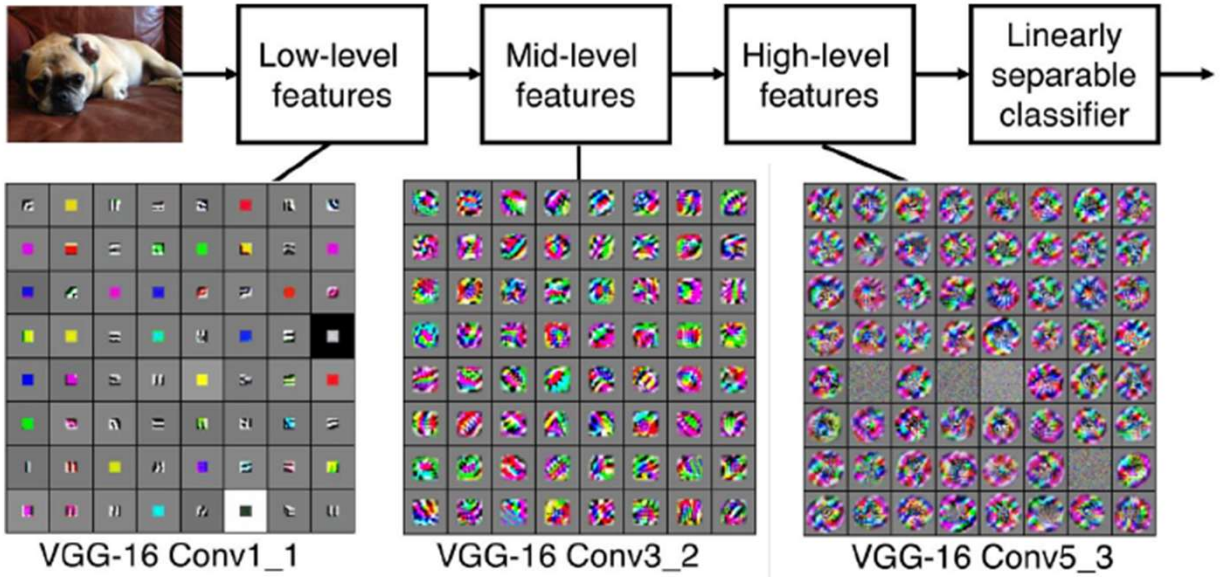
Convolutional neural network

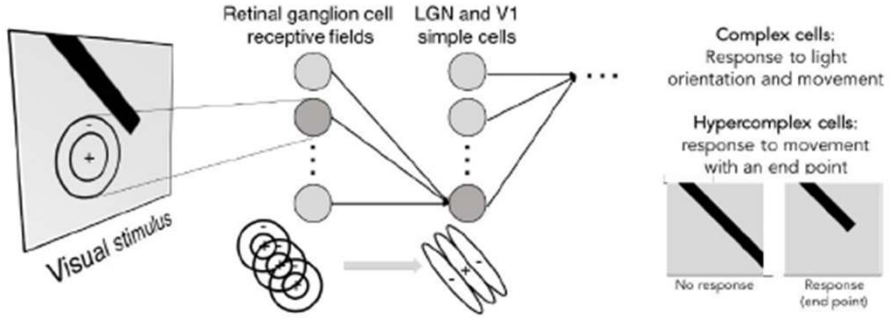


-1X1 filter with a certain depth

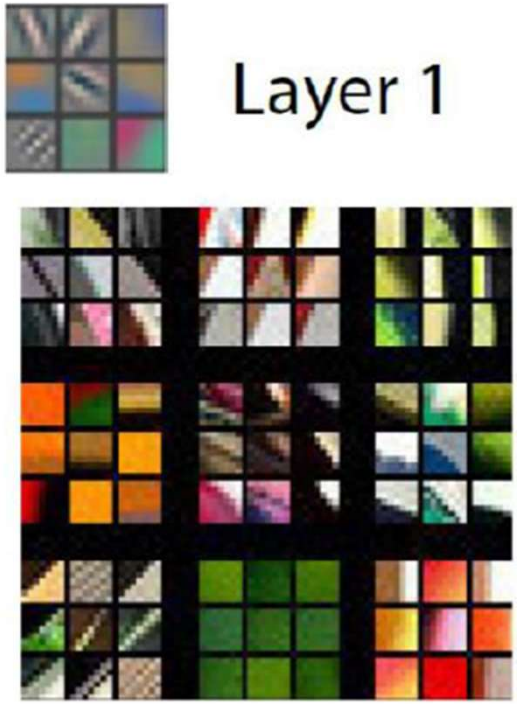
→ can modulate the output layer without changing the dimension

Convolutional neural network

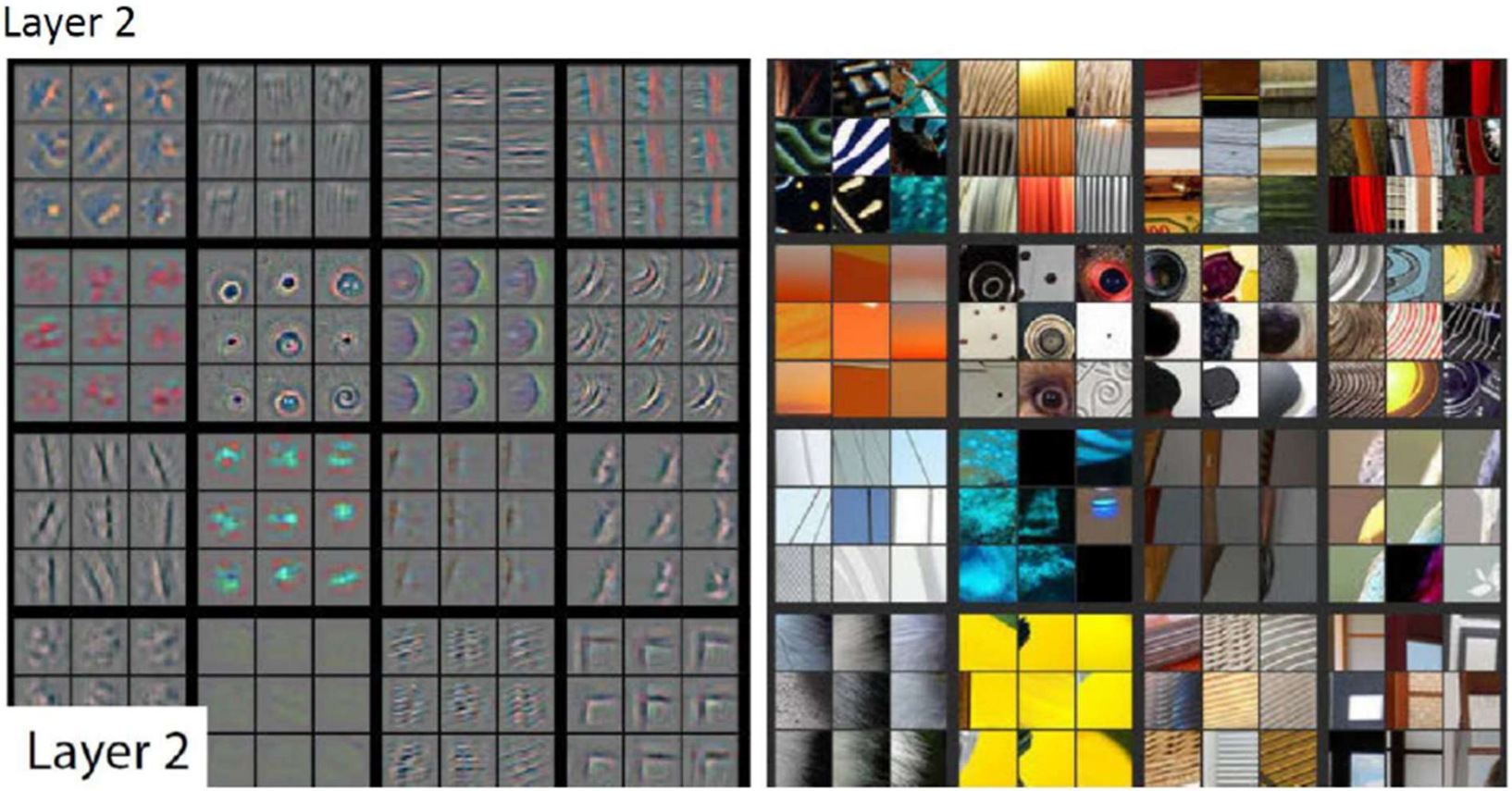




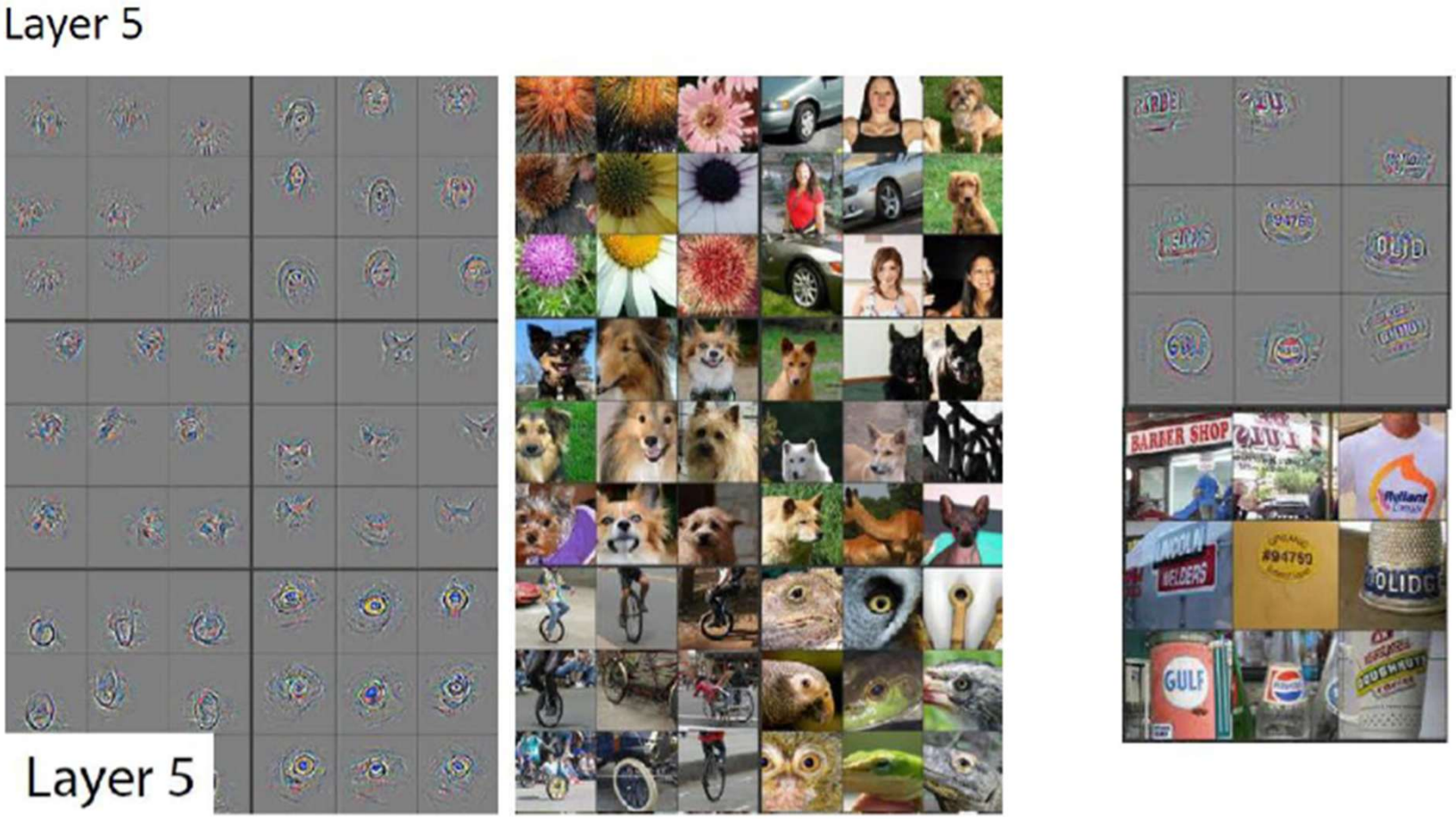
Convolutional neural network



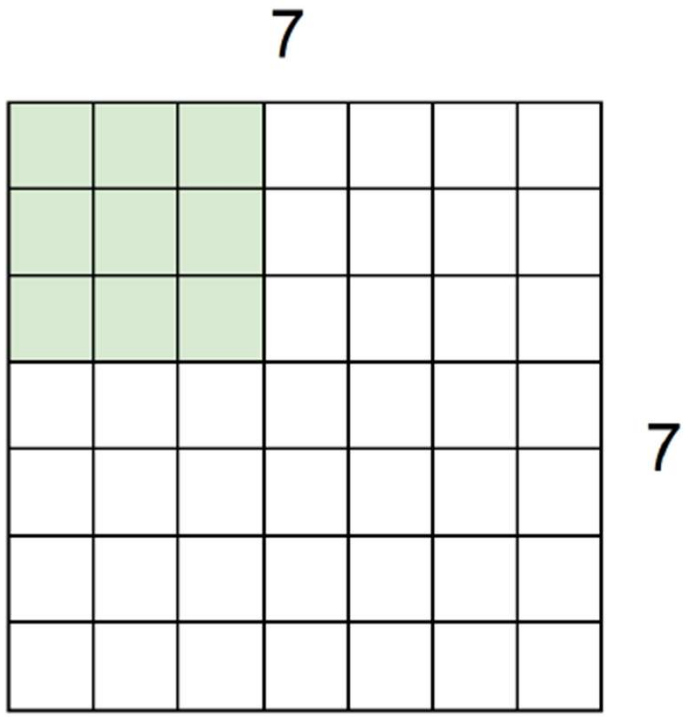
Convolutional neural network



Convolutional neural network

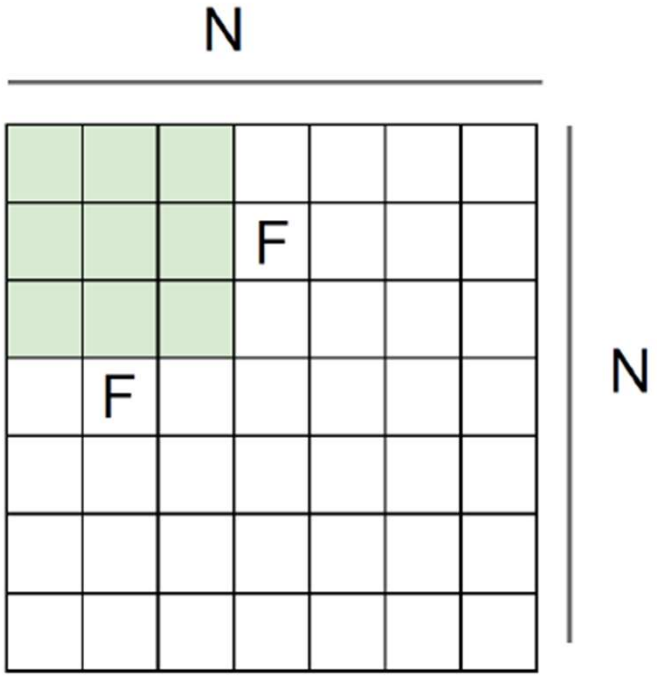


Calculating the dimension



7x7 input (spatially)
assume 3x3 filter

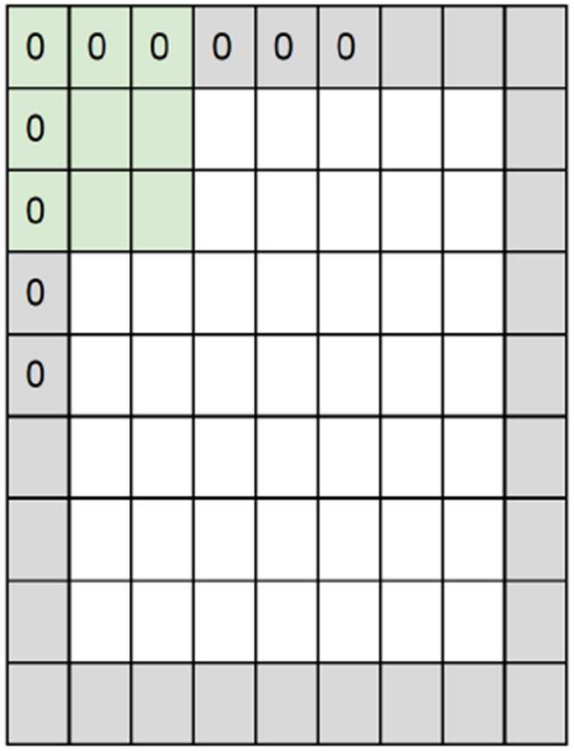
Calculating the dimension



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33

Zero padding



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

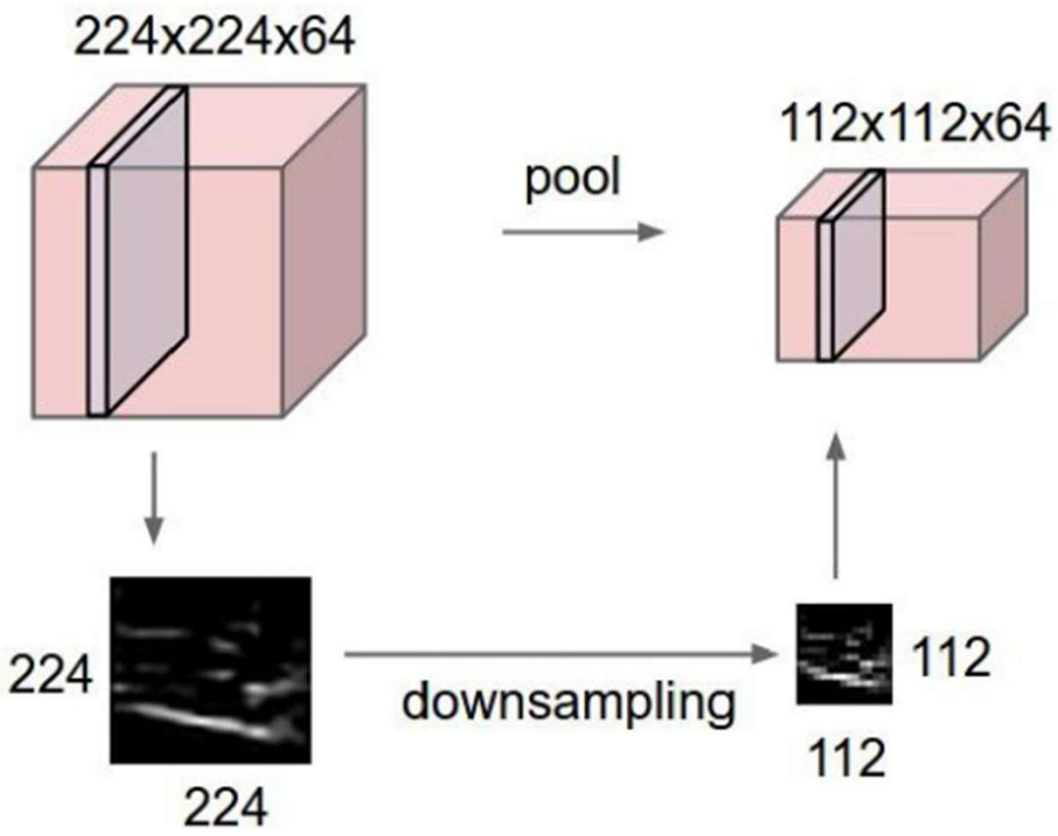
7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

-Often the boundary is not very neat or discrete
-Compatibility between new source of data

Calculating the dimension

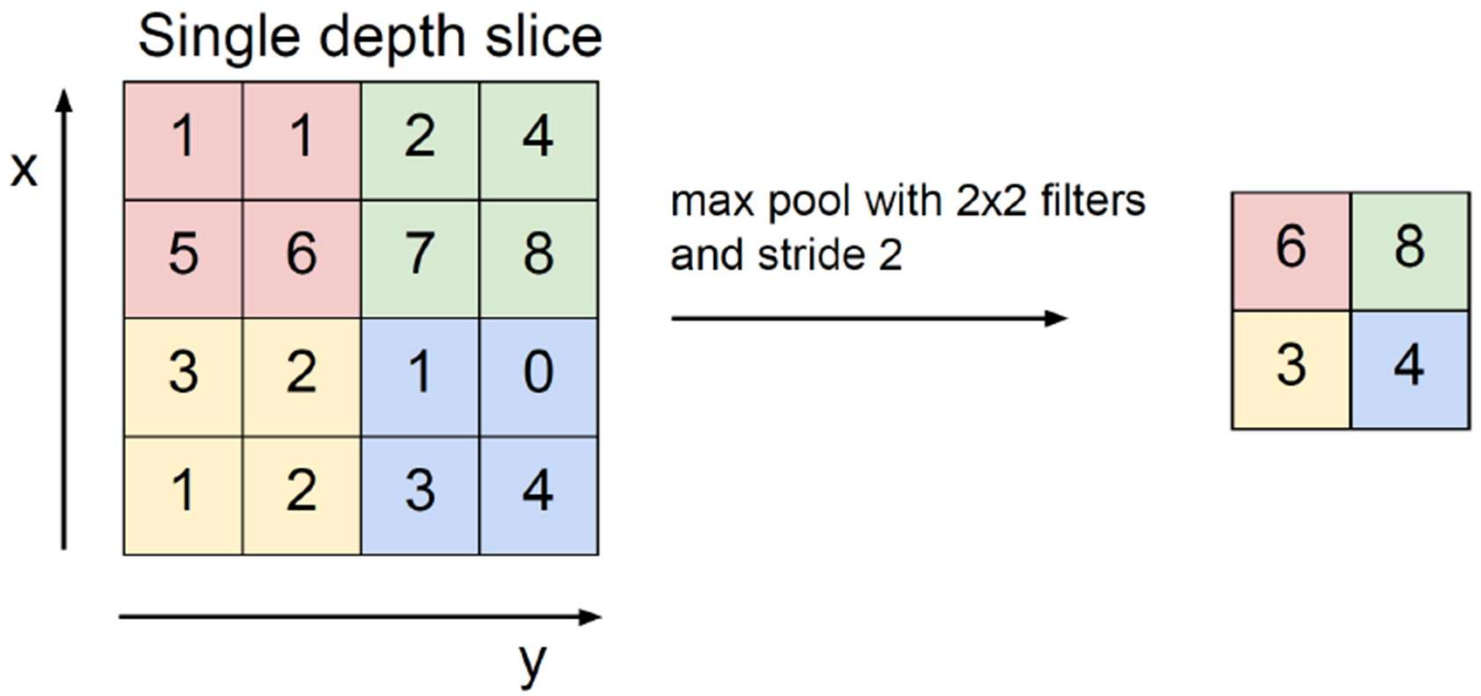
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$

Pooling layer



-Down sampling → reduce computational cost
(Usually averaging)

Max pooling


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

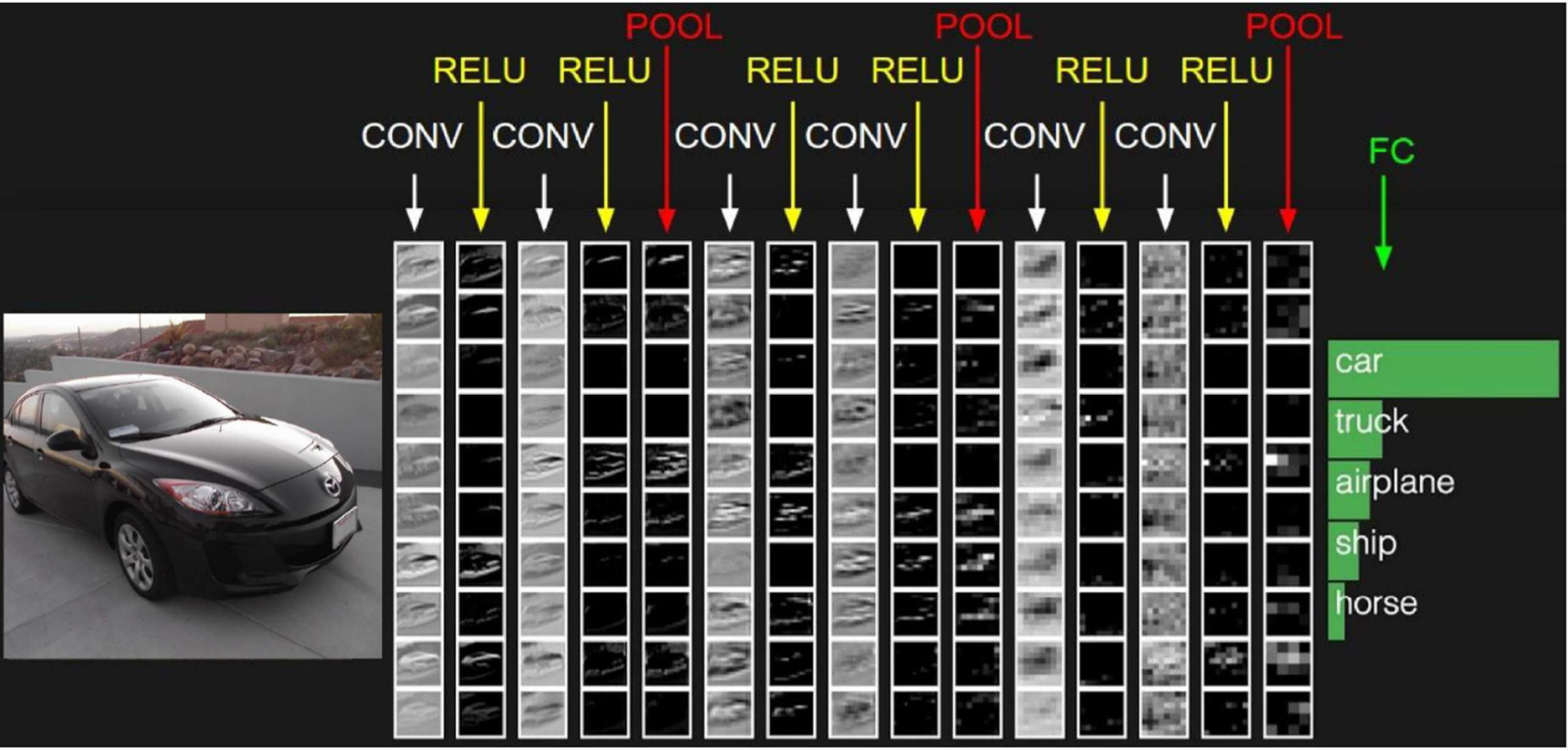
FC: Fully connected layer

→ MLP

→ Usually, MLP classify really well

→ After CNN, FC can be used for classifier

Convolutional neural network



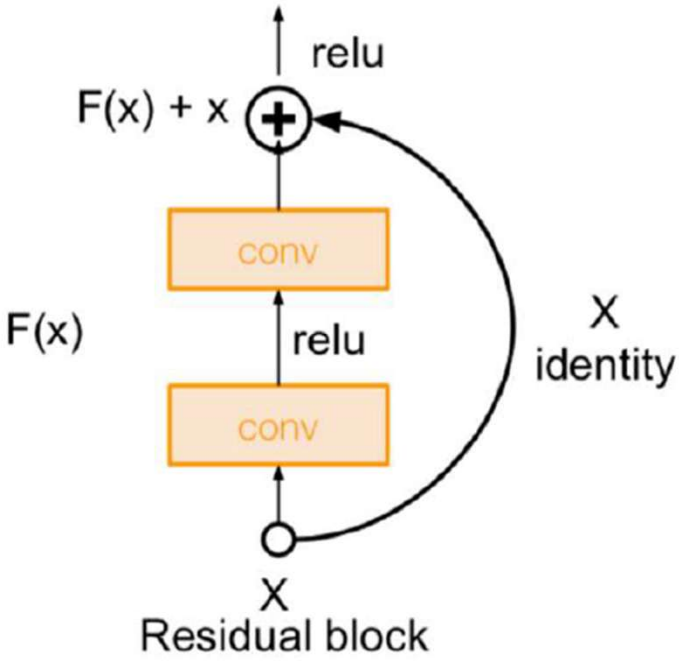
Stanford, cs231n 2017

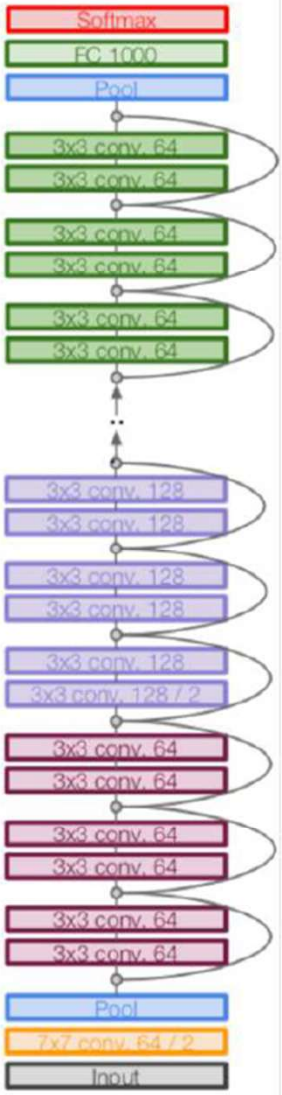
Case Study: ResNet

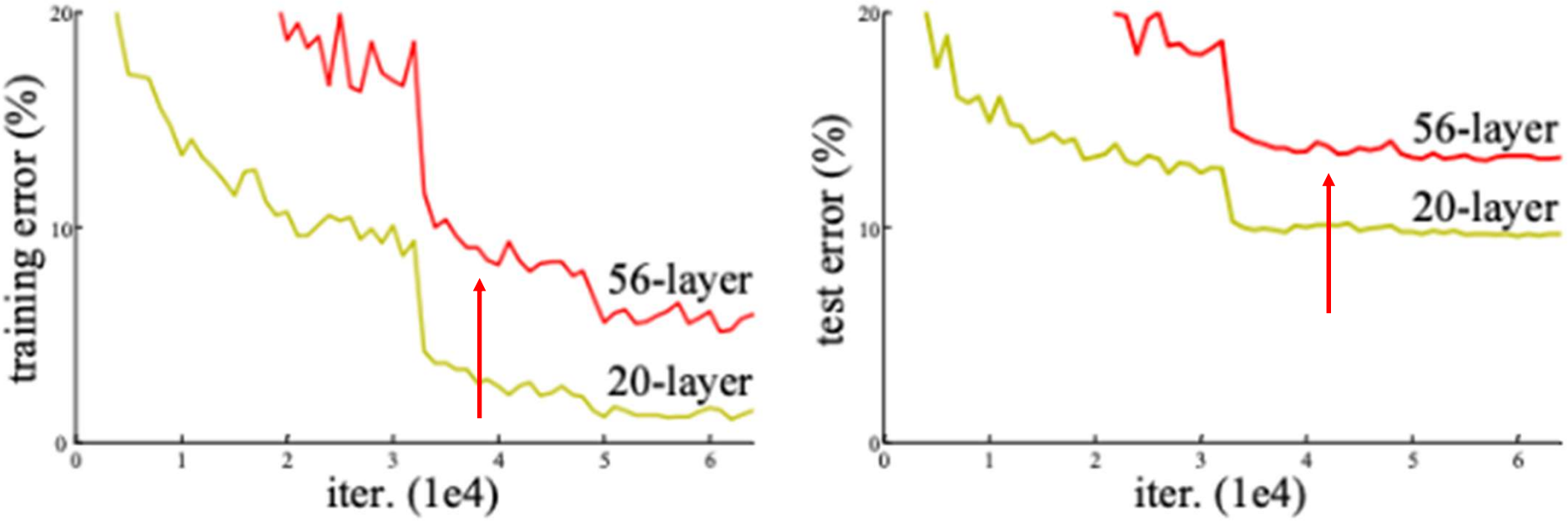
[He et al., 2015]

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!

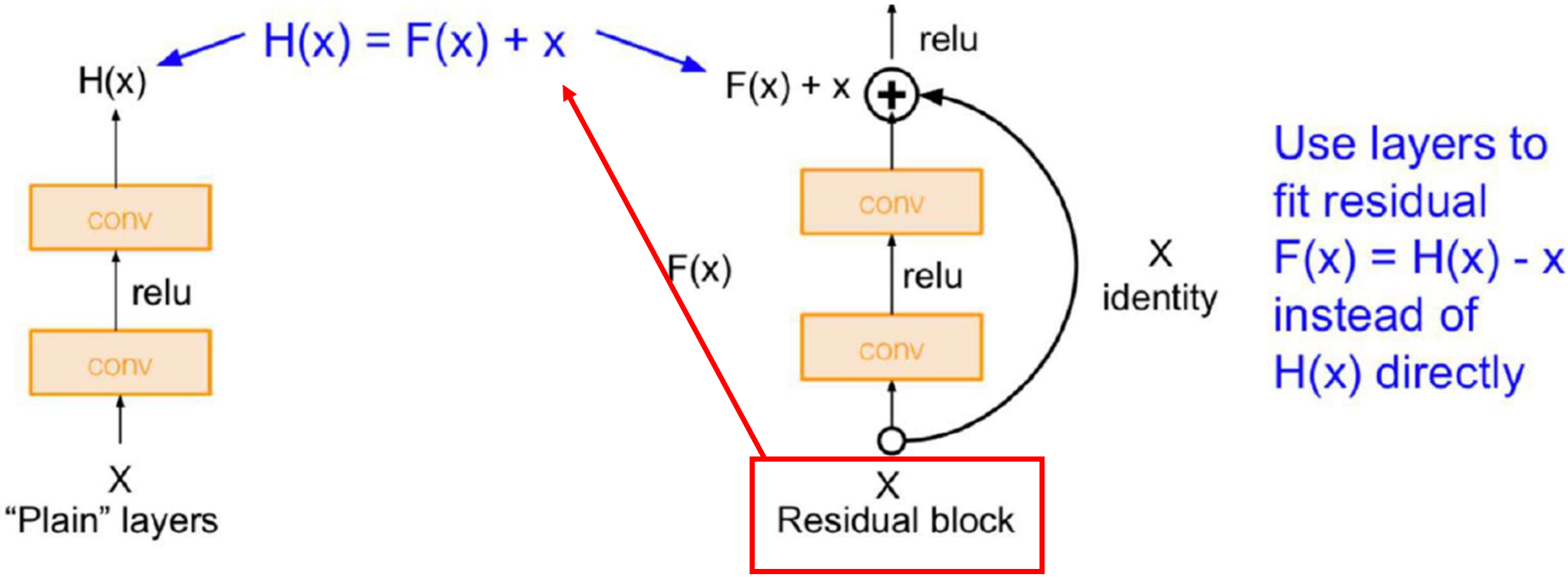






Deeper → No benefit for improving “error”

Residual block



*Deeper: hard to learn everything, keep what you've learned before

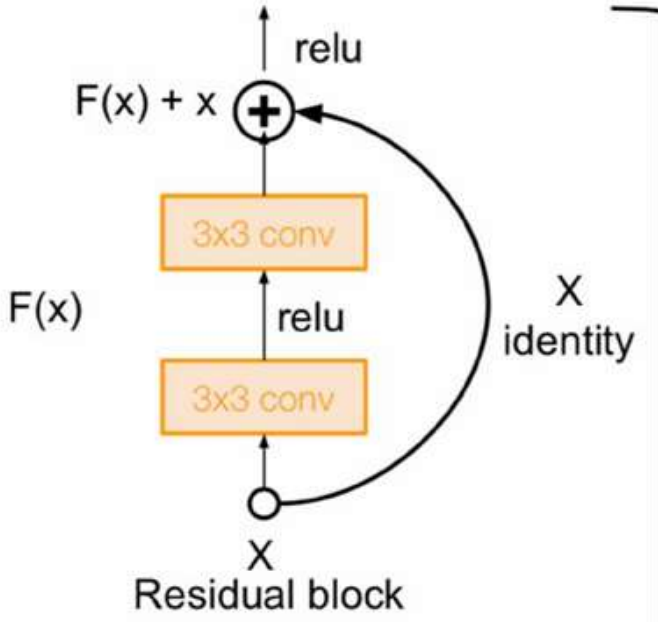
-N-layer input: output (x) from the previous layer
-N-layer output: H(x) → start to learn from what you've learned before (x)
-Training: F(x): H(x) – x

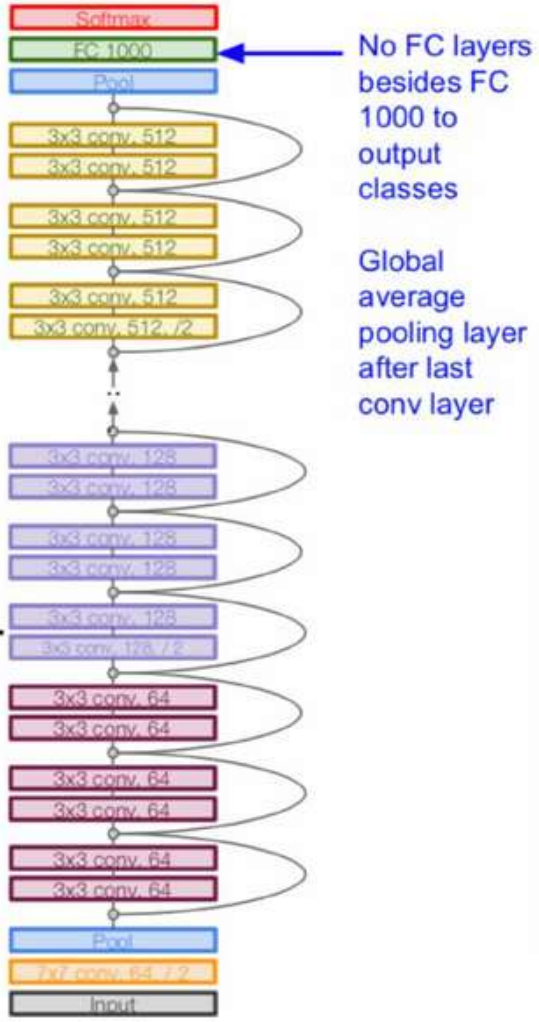
Case Study: ResNet

[He et al., 2015]

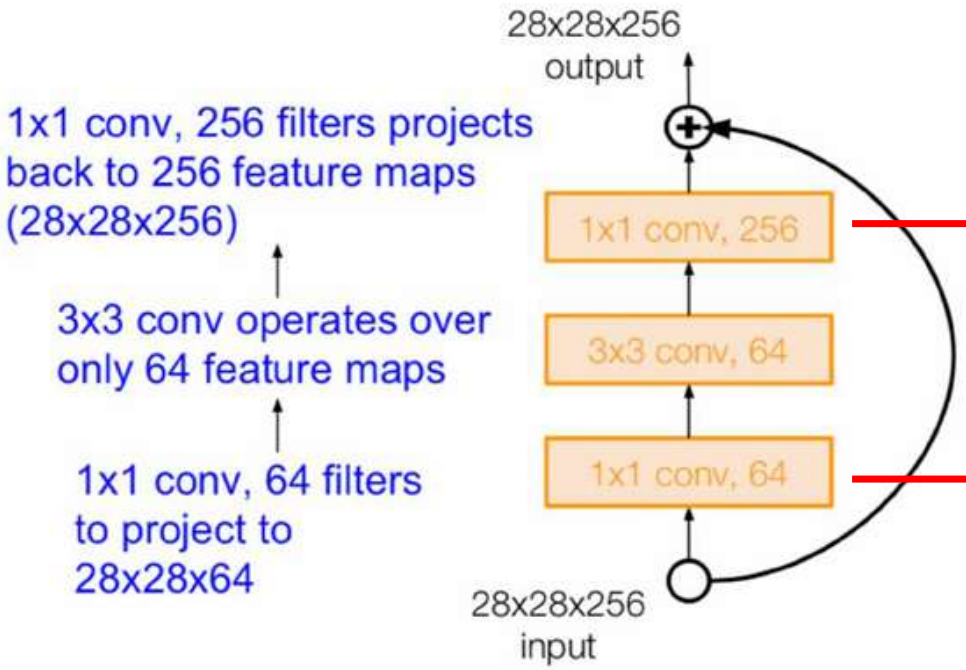
Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)





No pooling, instead stride 2

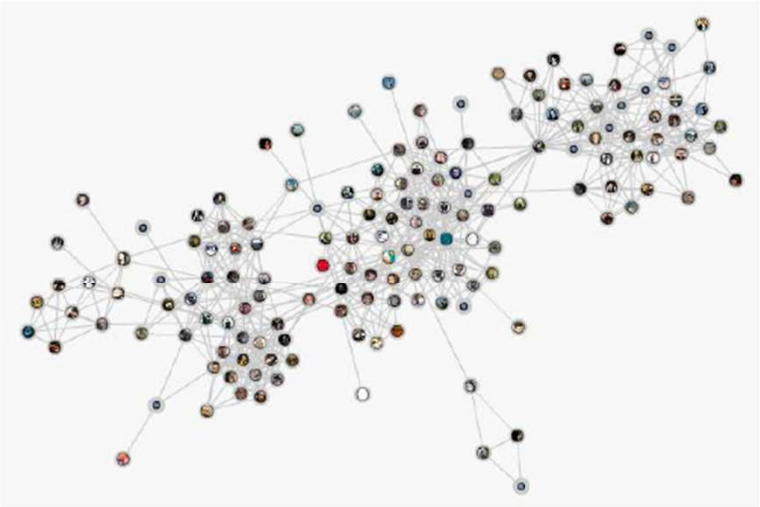


Return 256 dimesion

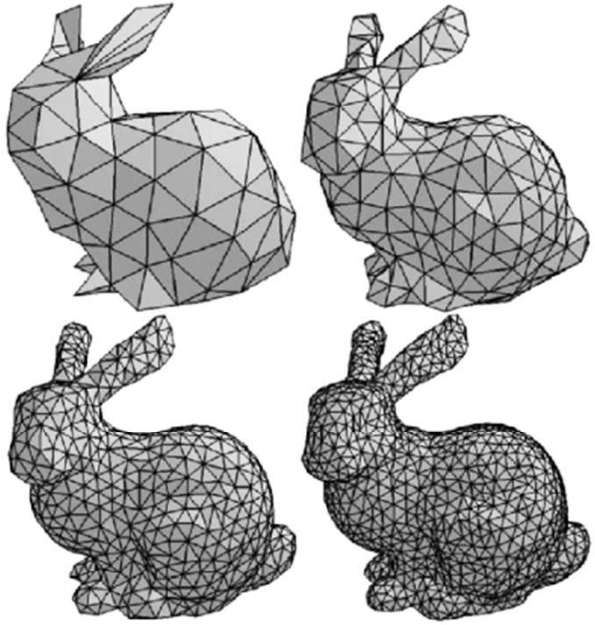
1X1 bottleneck layer
→ reduce the computational cost

GCN

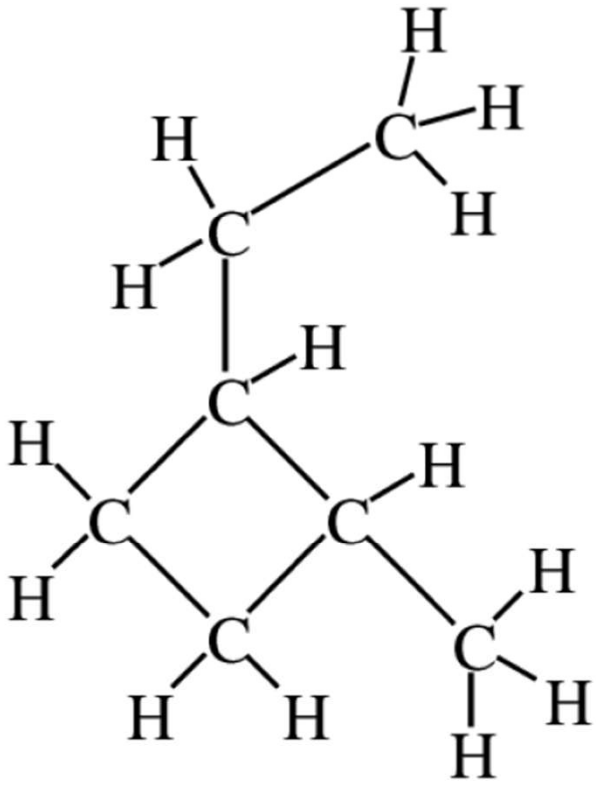
Graph structure



Social Graph

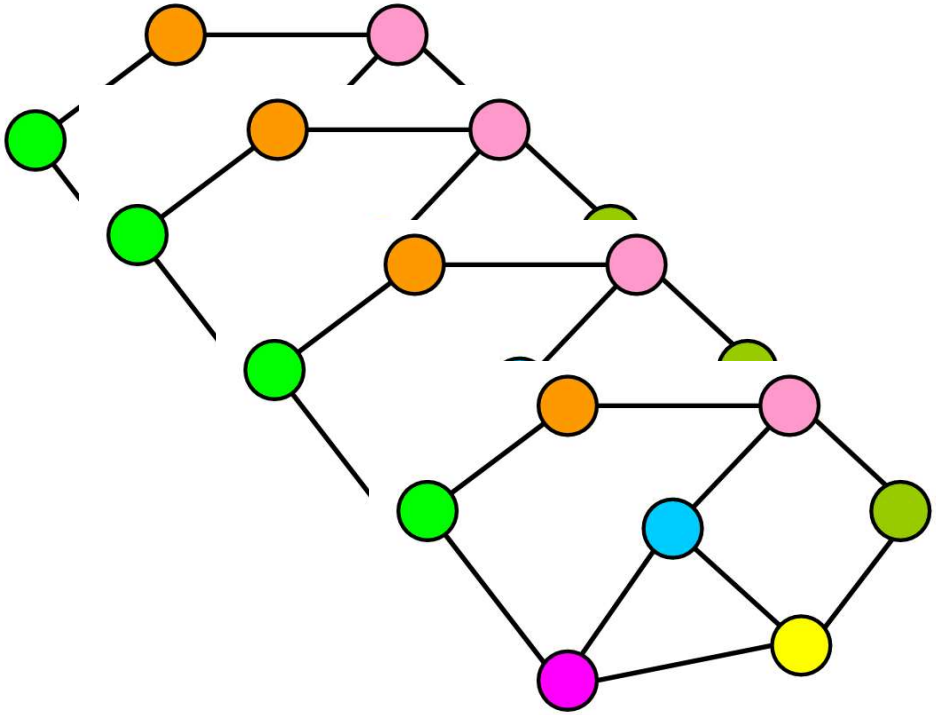


3D Mesh

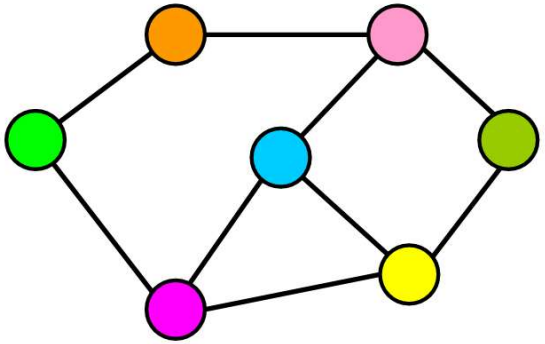


Molecular Graph

Graph structure

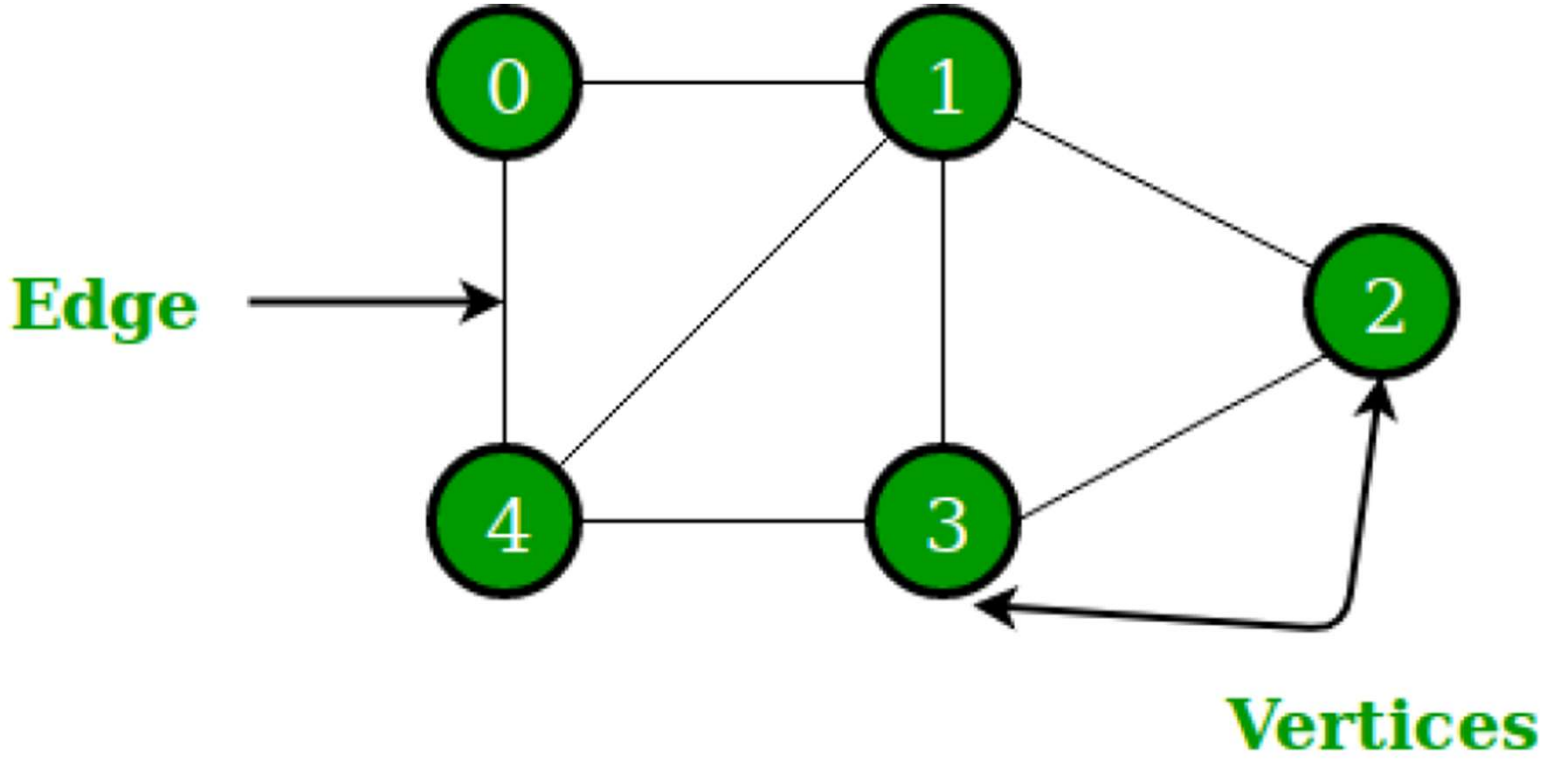


Training



Query → classification

Graph structure



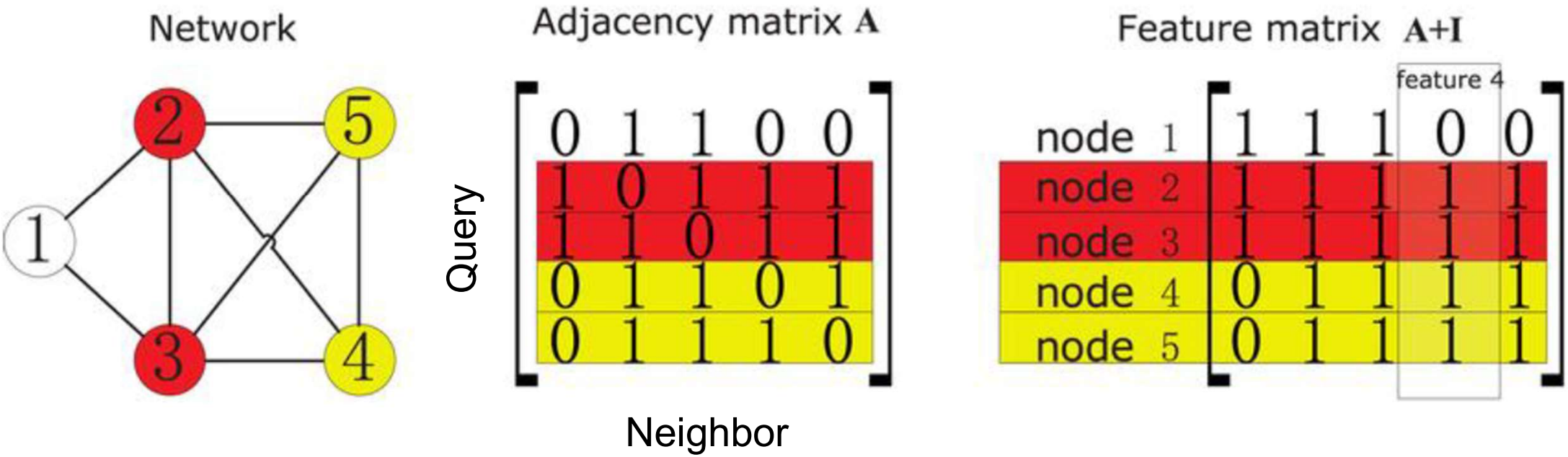
Vertex (Node)

Edge

Graph structure

Vertex (Node)
Node Feature Matrix

Edge
Adjacency Matrix

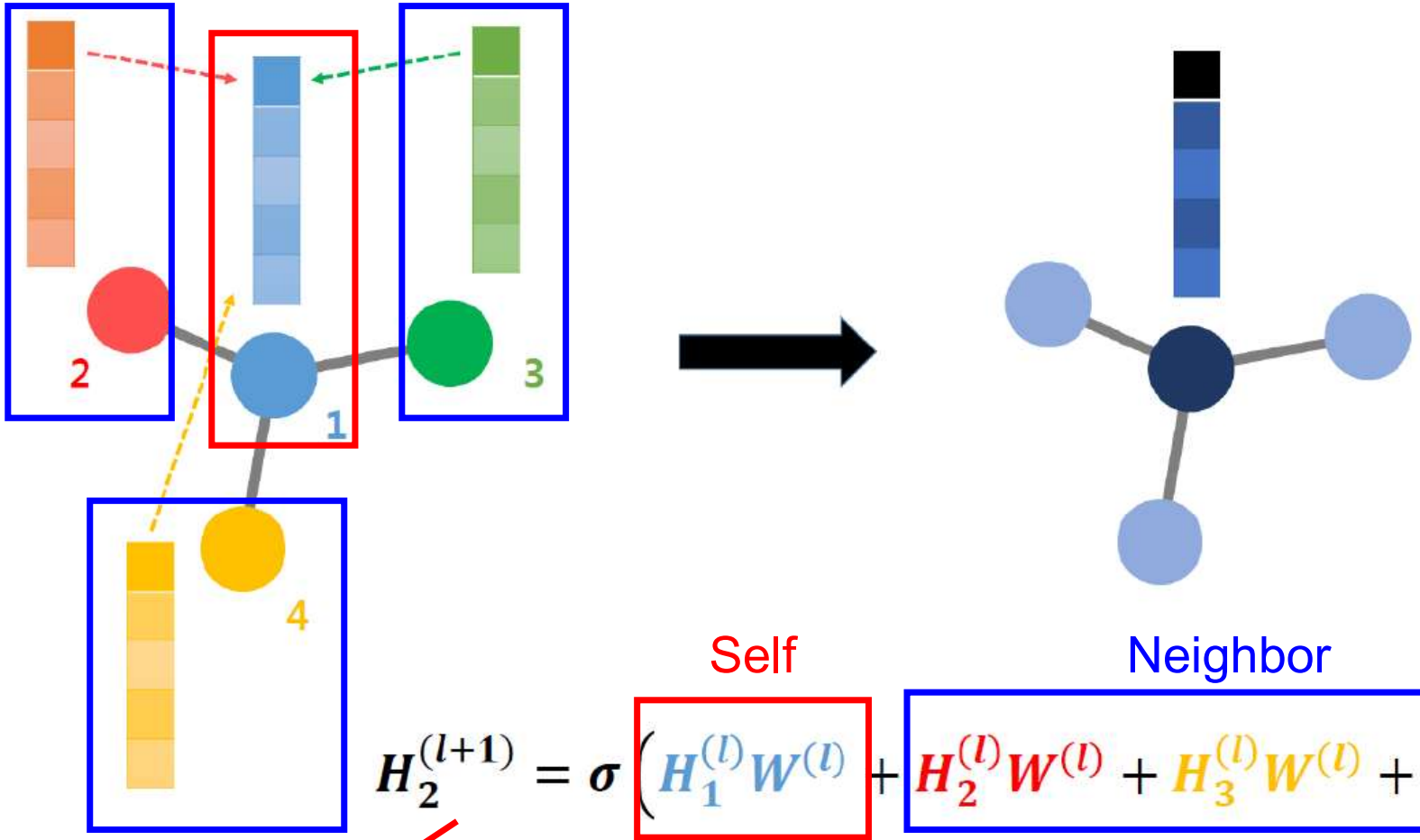


Graph convolutional network

CNN updates values in activation map in each layer.
Values of activation map determine the state of image.

Values of each **node feature** determine the **state** of graph.
→ Make each layer of network update values of each node feature

Graph convolutional network



Self

Neighbor

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

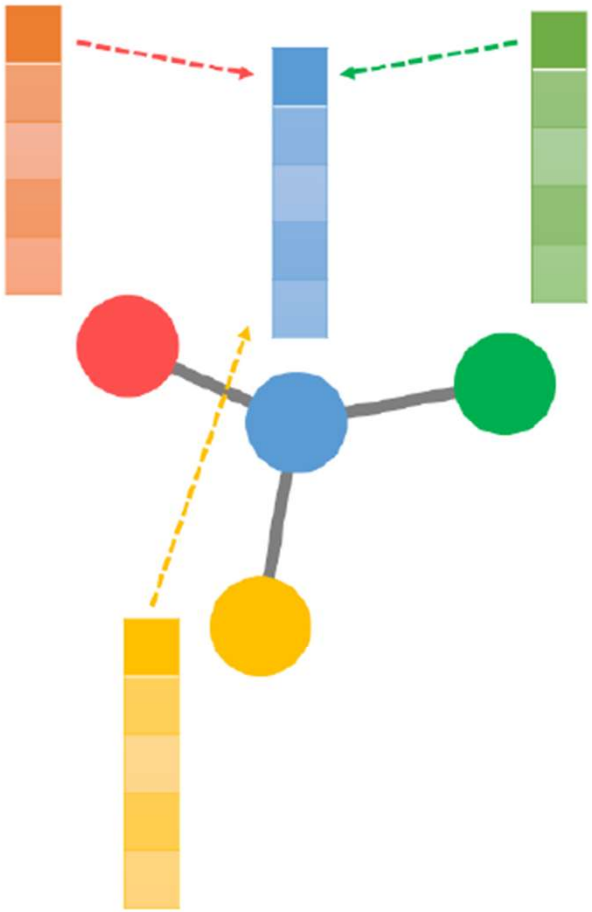
Node feature matrix

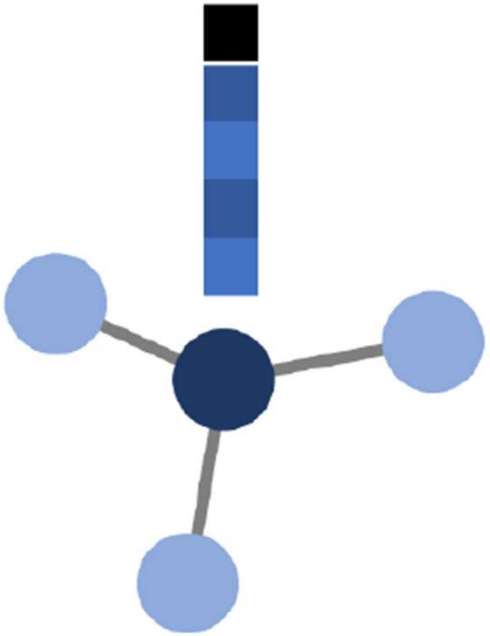
$$H_i^{(l+1)} = \sigma\left(\sum_{j \in N(i)} H_j^{(l)}W^{(l)} + b^{(l)}\right)$$

Layer

Every nodes

Graph convolutional network





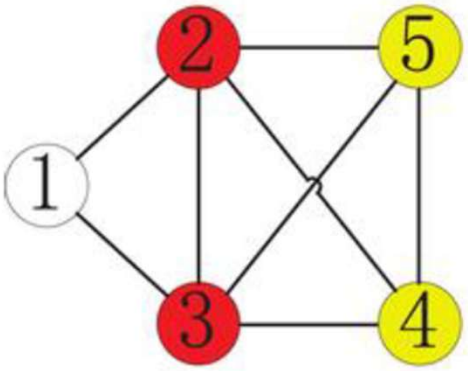
$$H^{(l+1)} = \sigma\left(AH^{(l)}W^{(l)} + b^{(l)}\right)$$

learnable parameters are shared

Graph convolutional network

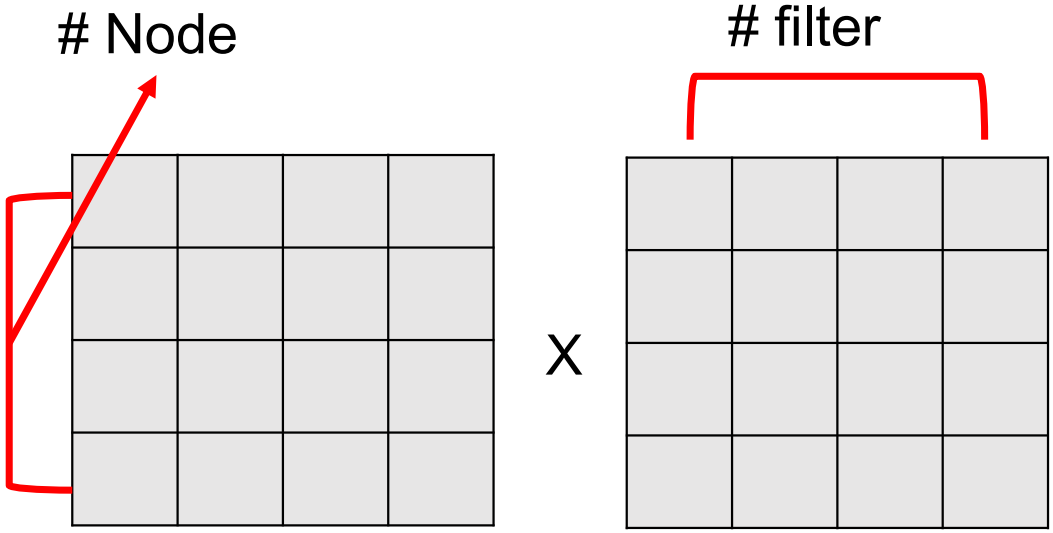
$$H^{(l+1)} = \sigma\left(AH^{(l)}W^{(l)} + b^{(l)}\right)$$

Network



Adjacency matrix A

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



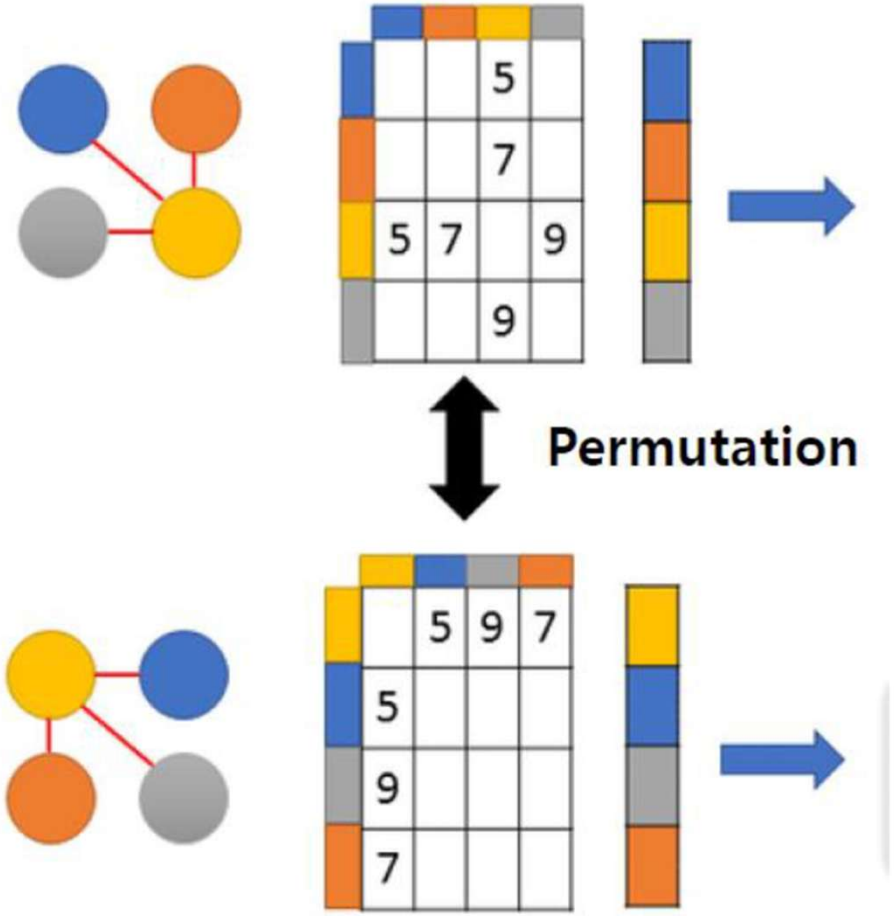
H: feature matrix

W: weight matrix

→ Only takes the weight from the direct neighbors

→ New node feature matrix (H) = #node X #filter

Readout layer (or Permutation invariance)



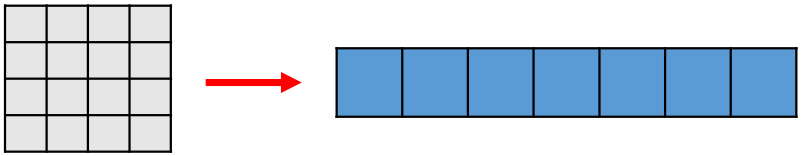
Mathematically proved

Node-wise summation

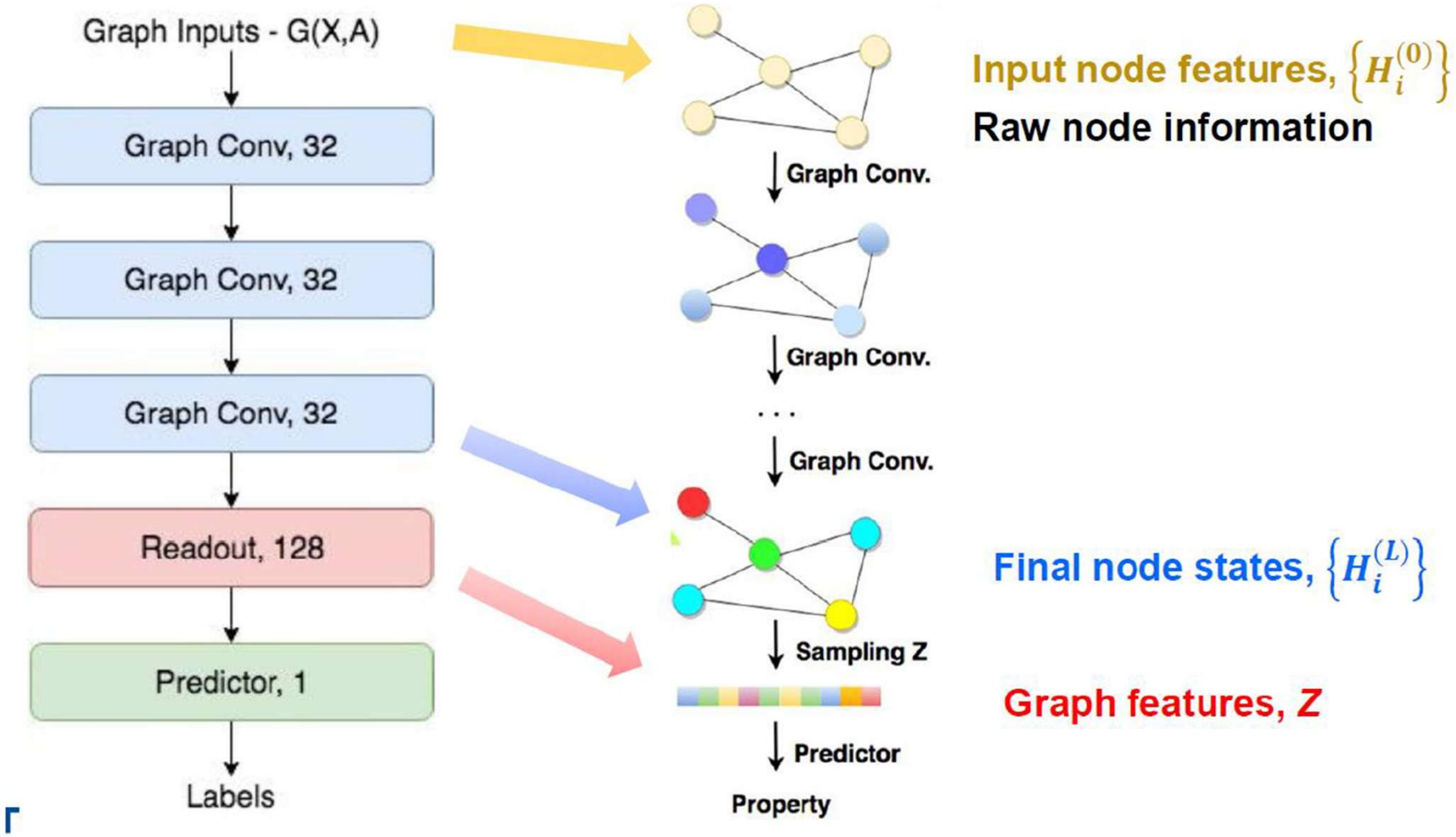
$$z_G = \tau\left(\sum_{i \in G} MLP\left(H_i^{(L)}\right)\right)$$

-The result should be same while the order of the nodes change (same graph)

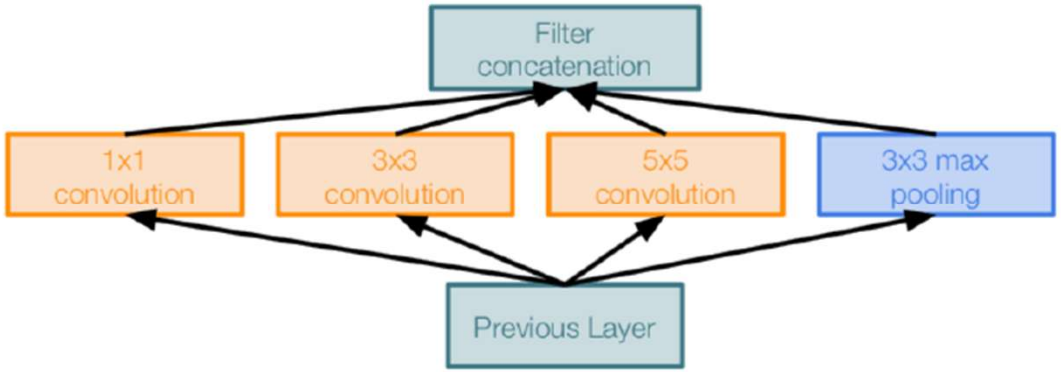
→H (feature matrix) → MLP
→ 1 linear vector



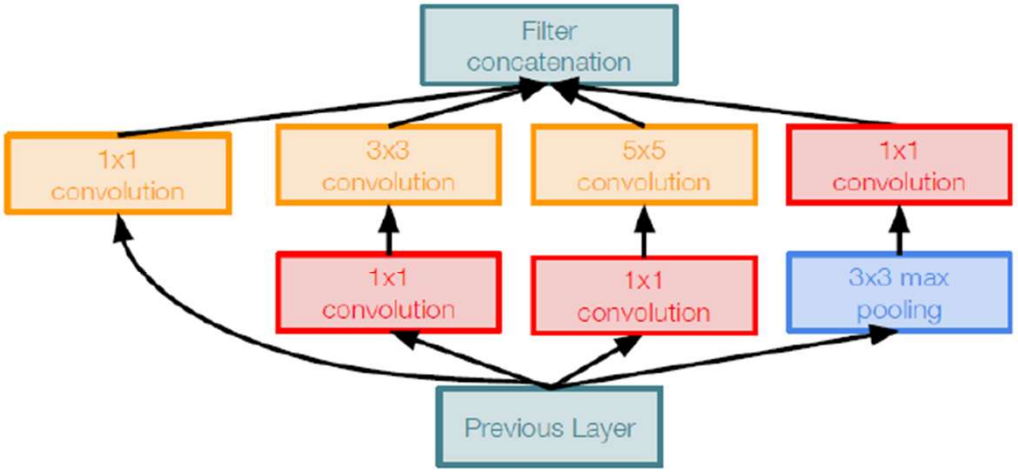
Graph convolutional network



Inception



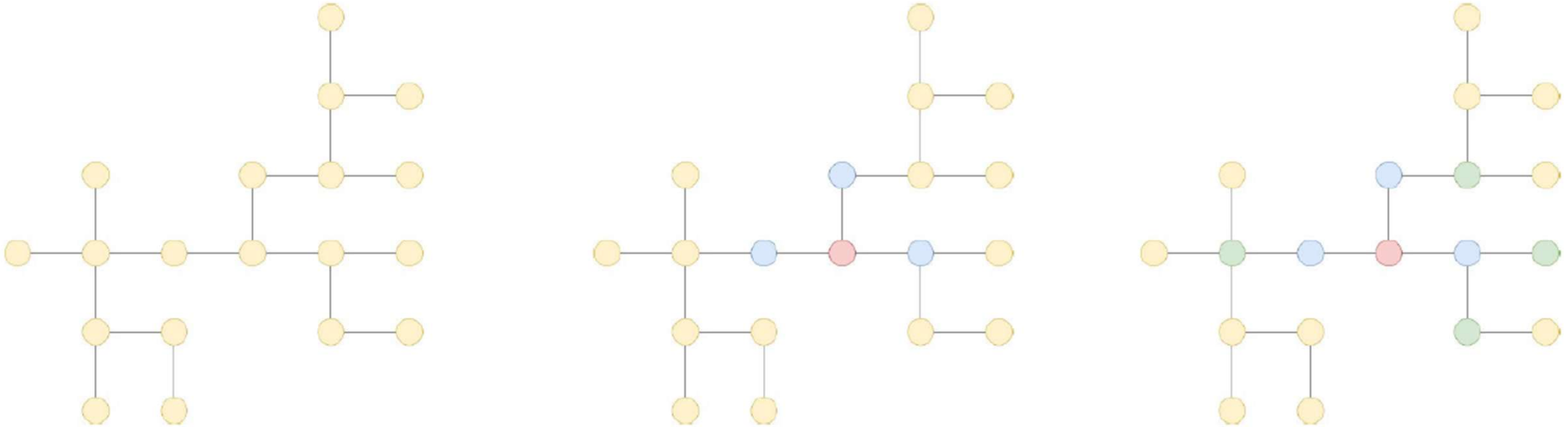
Naive Inception module



Inception module with dimension reduction

-Different filtering, pooling, … → concatenate later → depth increase
+ α

Inception



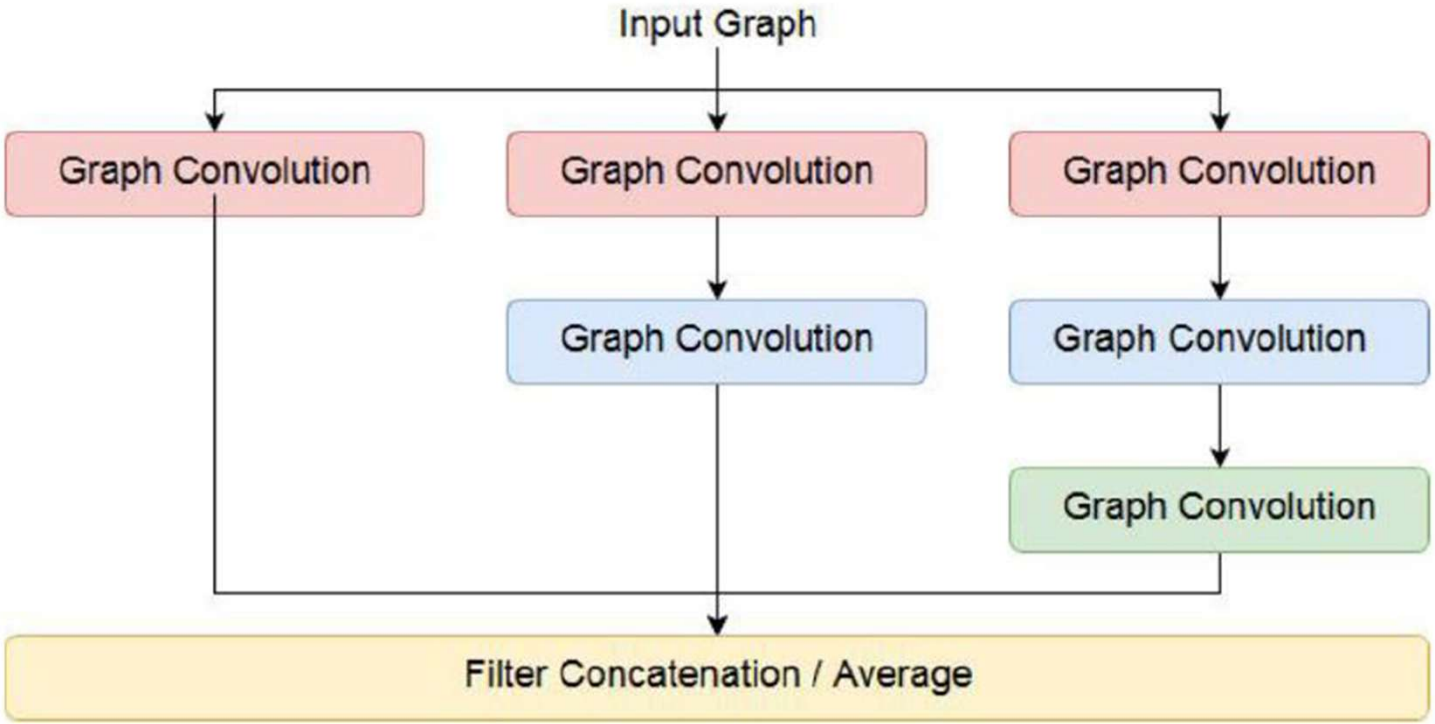
$$H^{(1)} = \sigma\left(AH^{(0)}W^{(0)}\right)$$

Information from direct neighbors

$$H^{(2)} = \sigma\left(AH^{(1)}W^{(1)}\right)$$

Information from the second direct neighbors

Inception



- Make network **wider**
- Avoid **vanishing gradient**

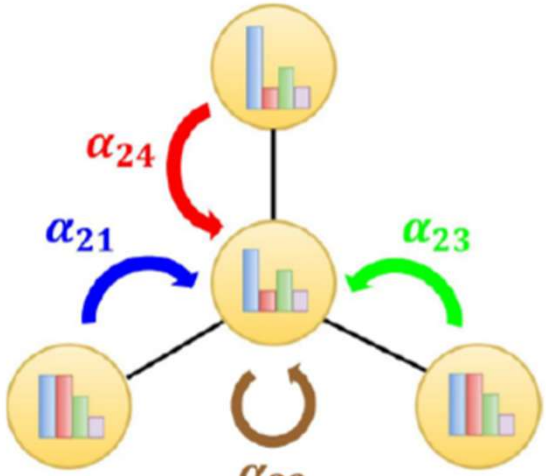
Math: GCN(i) * GCN(j)

Graph attention network

Vanilla GCN updates information of neighbor atoms **with same importance**.

$$H^{(l+1)} = \sigma\left(\sum_{j \in N(i)} H_j^{(l)} W^{(l)}\right)$$

Attention mechanism enables it to update nodes **with different importance**



$$H^{(l+1)} = \sigma\left(\sum_{j \in N(i)} \alpha_{ij} H_j^{(l)} W^{(l)}\right)$$

Ryu, Seongok, Jaechang Lim, Seung Hwan Hong and Woo Youn Kim. "Deeply learning molecular structure-property relationships using attention- and gate-augmented graph convolutional network." *arXiv preprint arXiv:1805.10988* (2018).

-Use different W from each neighbor → a!