大数据导论
Introduction to Big Data

大数据深度学习基础

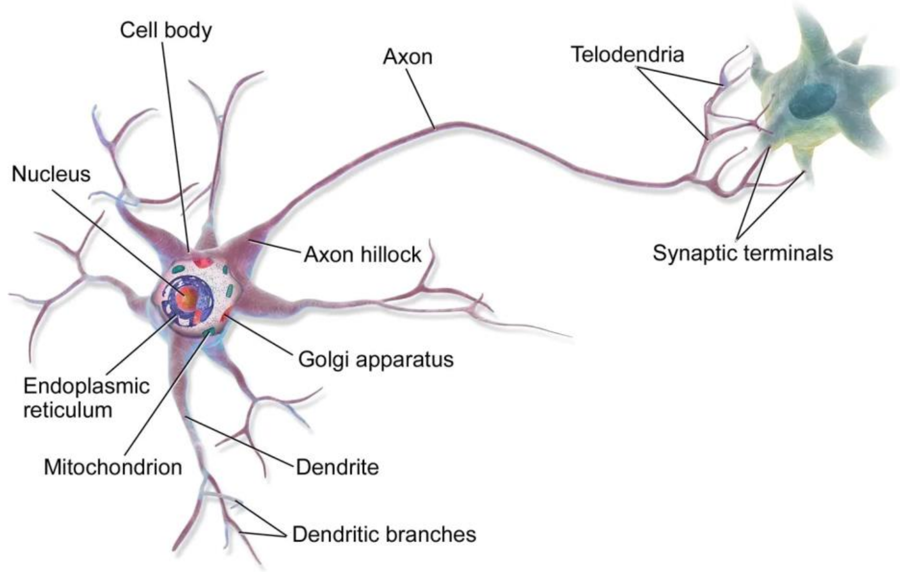
叶允明
计算机科学与技术学院
哈尔滨工业大学（深圳）

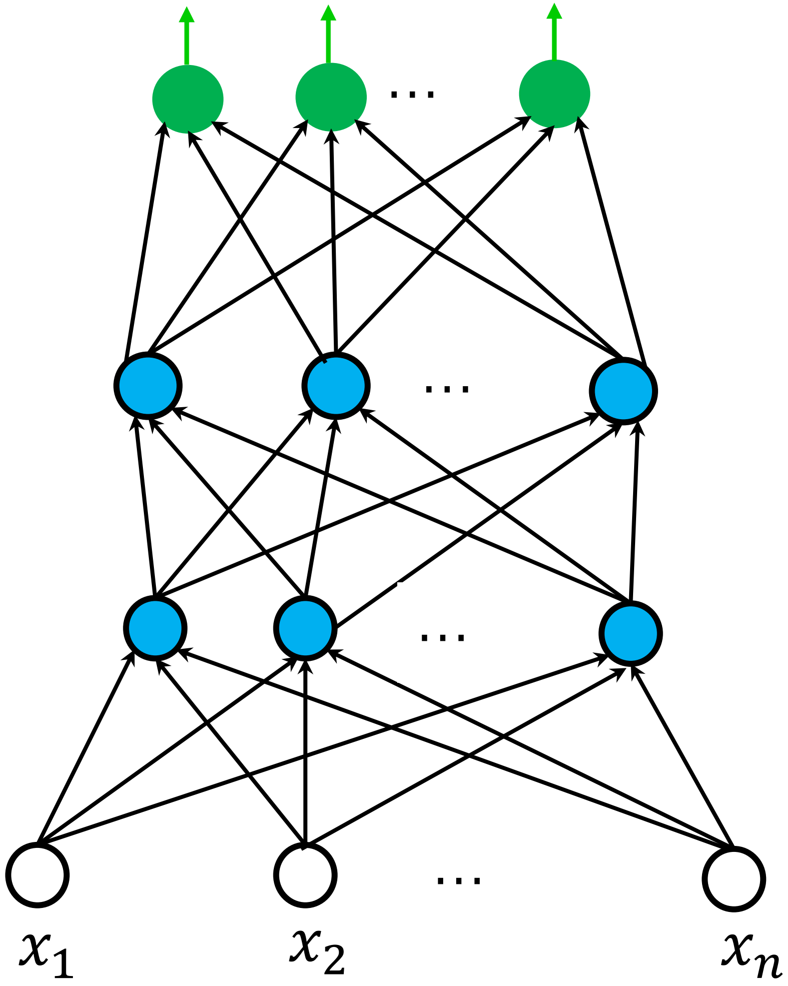
目录

- 神经网络与深度学习基础
- 如何学习出更好的神经网络

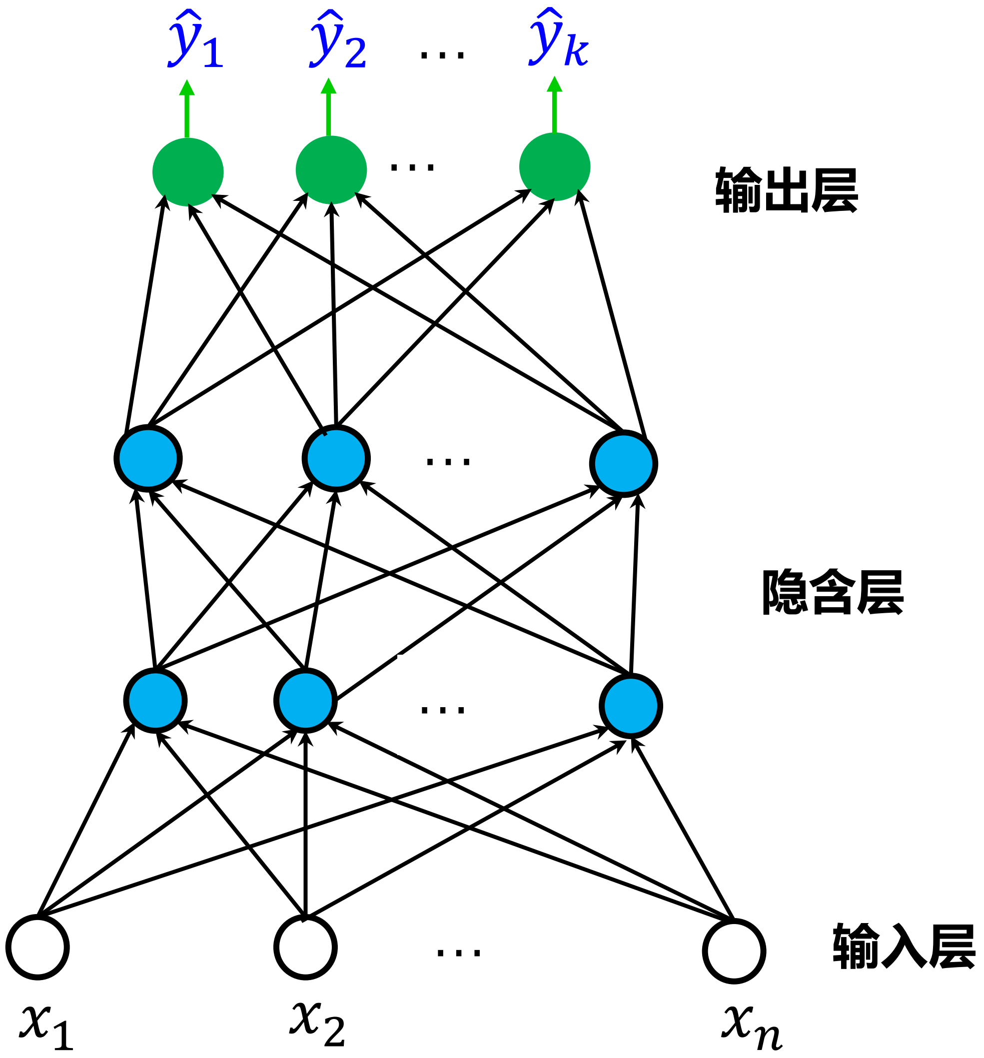
神经网络与深度学习基础

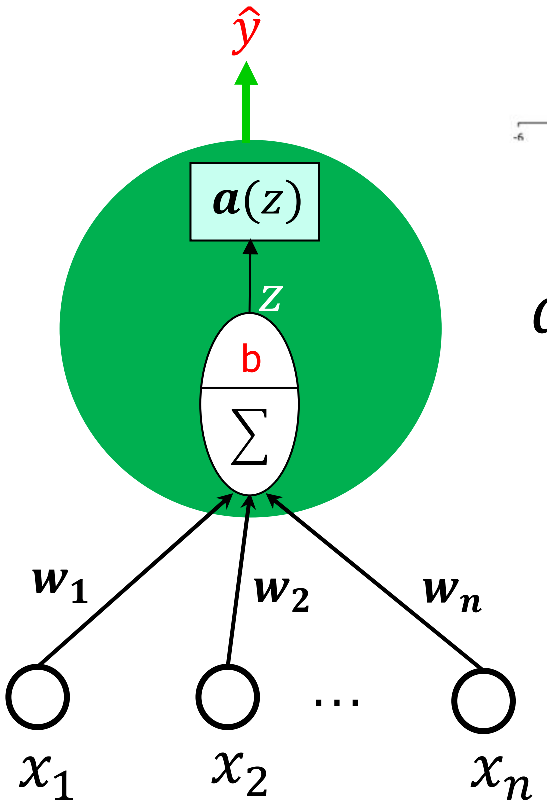
人工神经网络(Artificial Neural Network)

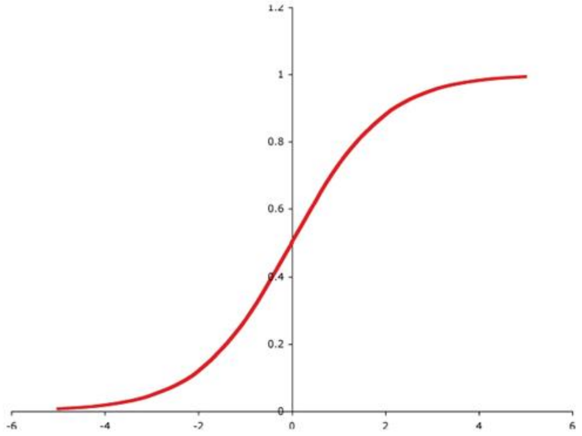




神经网络的常用结构





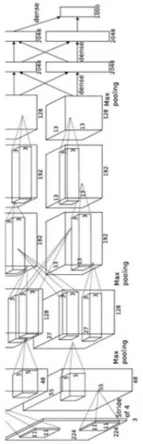


$$a(z) = \frac{1}{1 + e^{-z}}$$

深度学习：深度神经网络

8层

16.4%



AlexNet (2012)

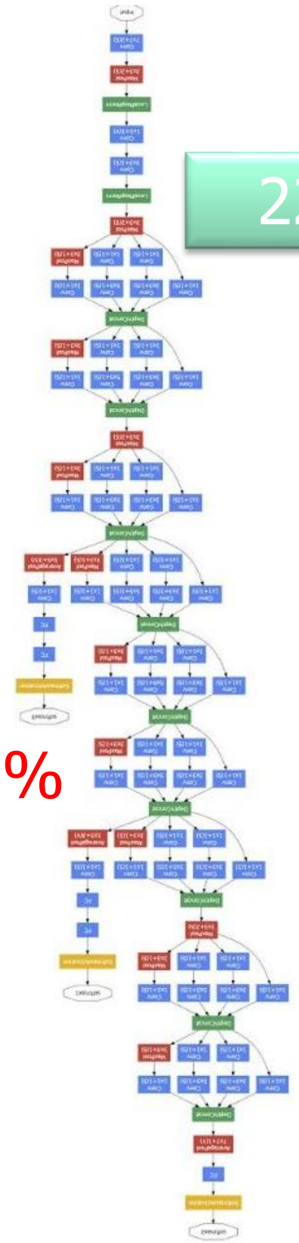
19 层

7.3%

VGG (2014)

22 层

6.7%



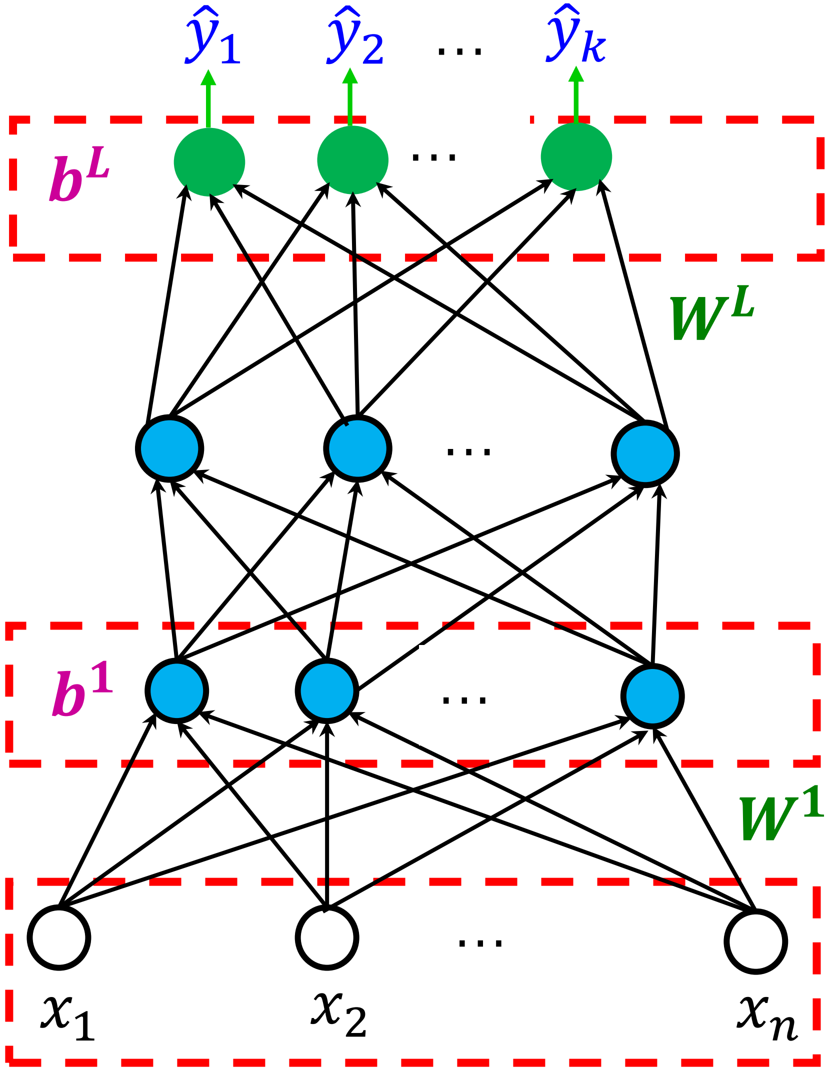
GoogleNet (2014)

……

据推测：**GPT-4在120层中总共包含了1.8万亿参数！**

全连接前馈神经网络的计算过程

$$a(z)=\sigma(z)=\frac{1}{1+e^{-z}}$$



$$\boldsymbol{a}^L=\boldsymbol{\sigma}(\boldsymbol{W}^L\,\boldsymbol{a}^{L-1}+\boldsymbol{b}^L)$$

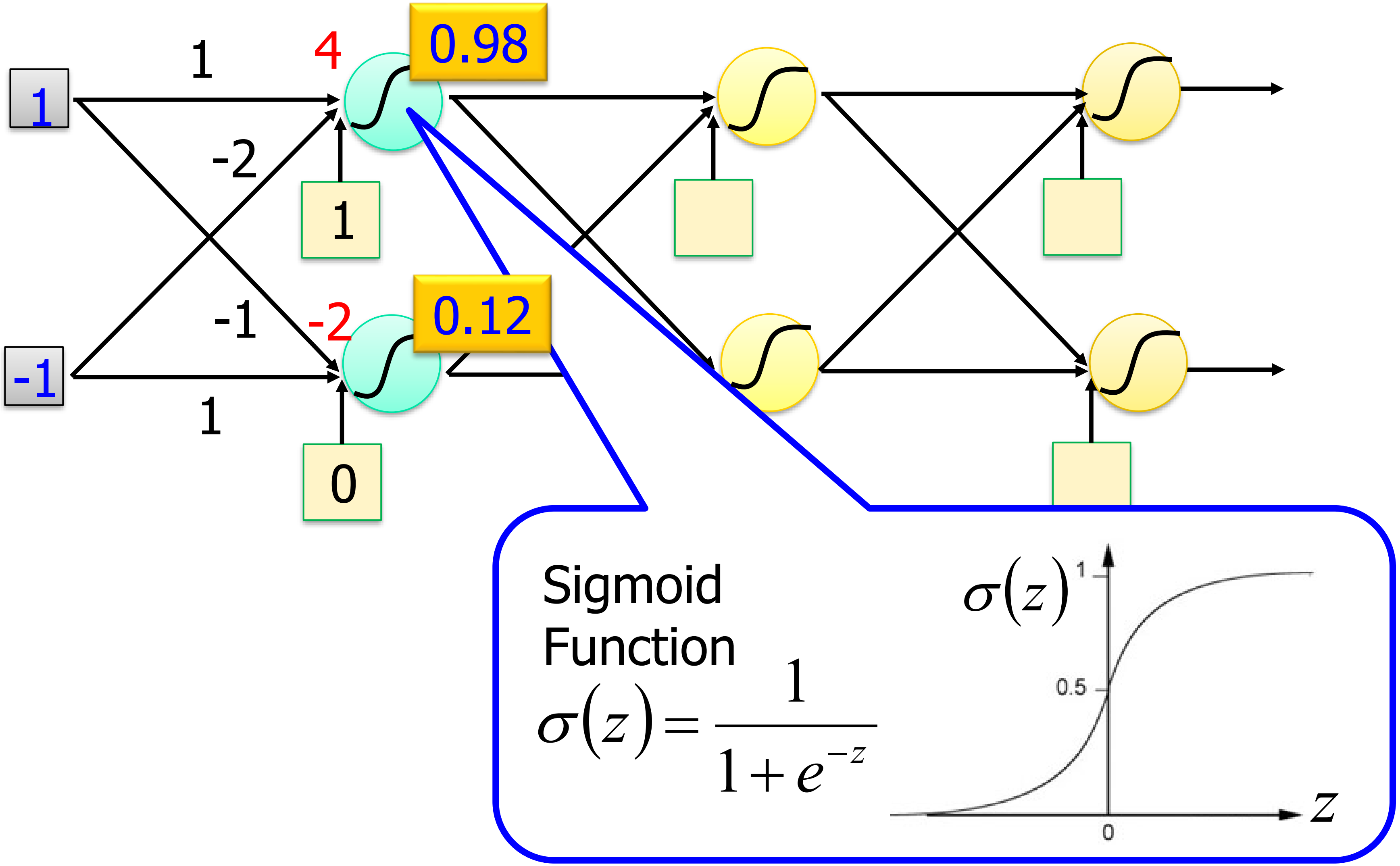
d^L is the total number of neurons of L layer

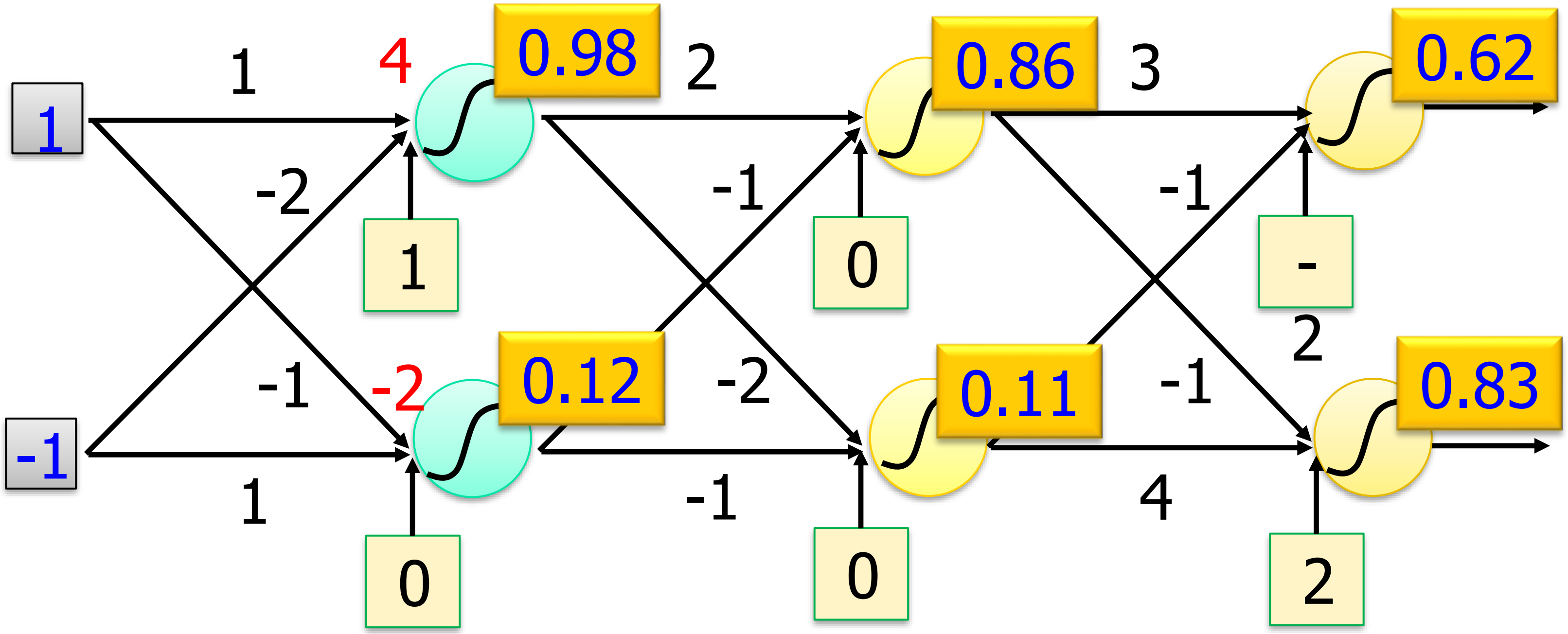
$$\boldsymbol{b}^1=\begin{bmatrix} b_1^1 \\ b_2^1 \\ \cdots \\ b_i^1 \\ \cdots \\ b_{d^1}^1 \end{bmatrix} \qquad \boldsymbol{w}_i^1=\begin{bmatrix} w_{i1}^1 \\ w_{i2}^1 \\ \cdots \\ w_{ij}^1 \\ \cdots \\ w_{id^0}^1 \end{bmatrix}$$

$$\boldsymbol{a}^1=\boldsymbol{\sigma}(\boldsymbol{W}^1\,\boldsymbol{a}^0+\boldsymbol{b}^1)$$

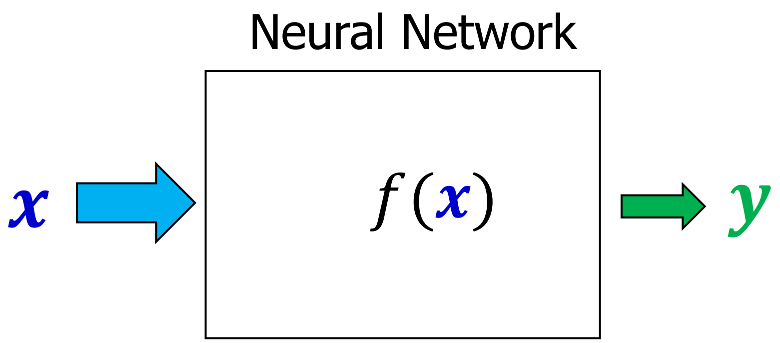
$$\boldsymbol{a}^0$$

$$\boldsymbol{W}^1=\begin{bmatrix} (\boldsymbol{w}_1^1)^T \\ (\boldsymbol{w}_2^1)^T \\ \cdots \\ (\boldsymbol{w}_i^1)^T \\ \cdots \\ (\boldsymbol{w}_{d^1}^1)^T \end{bmatrix}=\begin{bmatrix} w_{11}^1 & \cdots & w_{1d^0}^1 \\ \cdots & w_{ij}^1 & \cdots \\ w_{d^1d^0}^1 & \cdots & w_{d^1d^0}^1 \end{bmatrix}$$

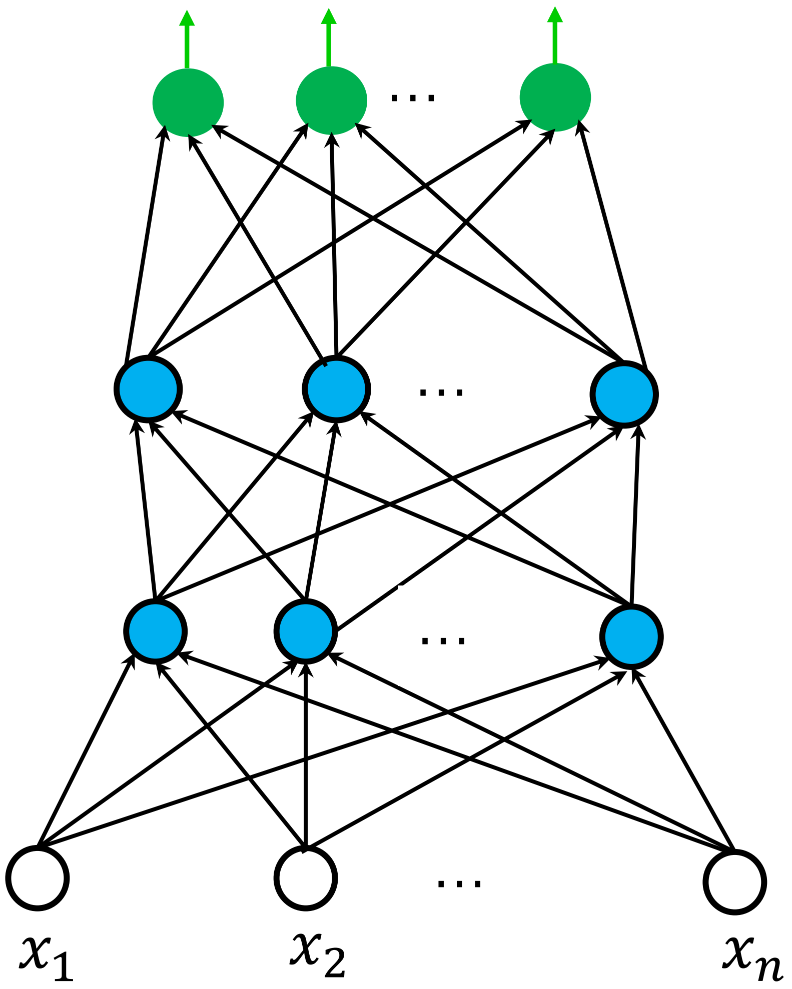

1
-1
1
-2
-1
1
4
-2
0.98
0.12
1
0
Sigmoid Function
$\sigma(z) = \frac{1}{1+e^{-z}}$
$\sigma(z)$
1
0.5
0
z


1
-1
1
-2
-1
1
4
-2
1
0
0.98
0.12
2
-1
-2
-1
0
0
0.86
0.11
3
-1
-1
4
-
2
2
0.62
0.83

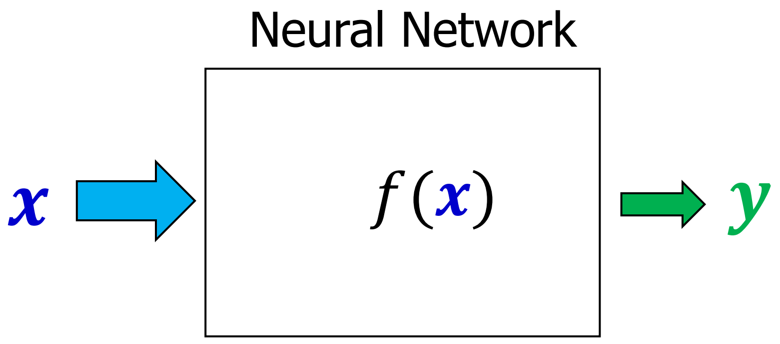
神经网络的本质



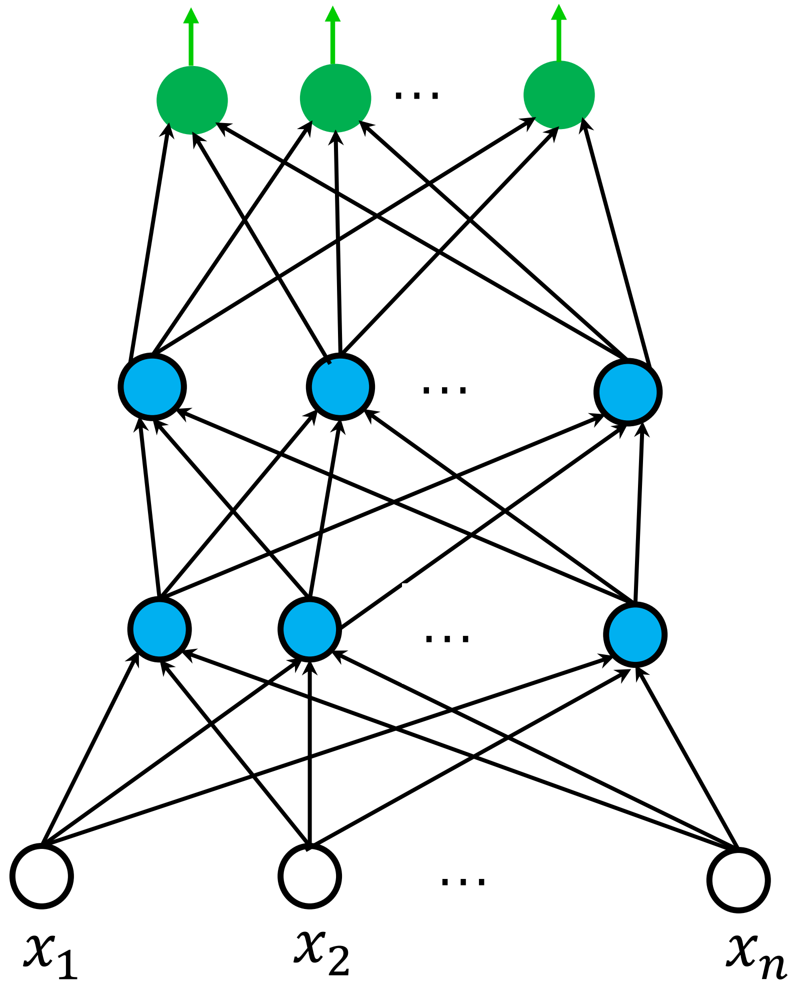
$$f(\boldsymbol{x}) = f_n(f_{n-1}(\dots f_2(f_1(\boldsymbol{x}))\dots))$$



神经网络能做什么？



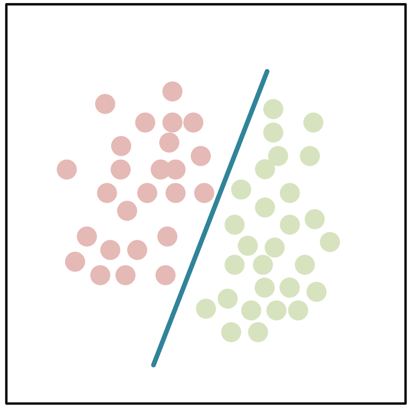
- 表征学习（Representation learning）
- 执行决策（Decision making）



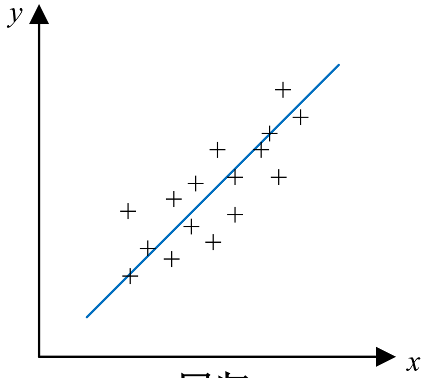
为什么神经网络具有普适的应用价值

- 事物的向量化表示具有通用性
- 各种应用任务都可转化为函数f(x)拟合问题

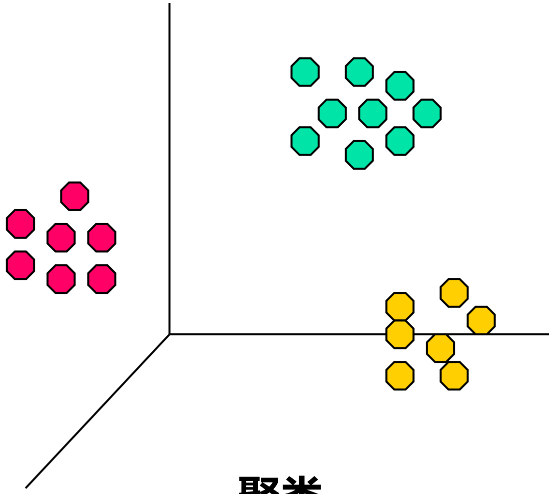
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



分类



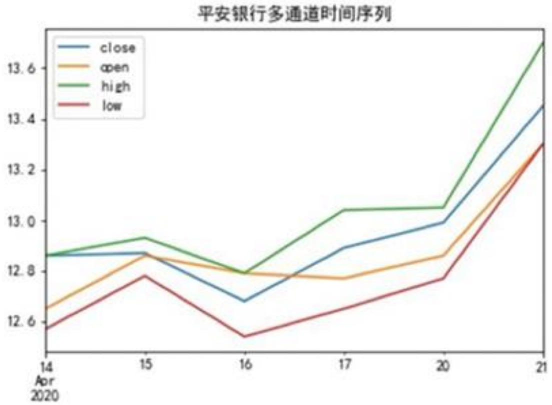
回归



聚类

据中央气象台消息，今年第11号台风“轩岚诺”（超强台风级）的中心今天（8月31日）早晨5点钟位于日本冲绳县那霸市偏东方向约340公里的西北太平洋洋面上，就是北纬26.1度、东经131.1度，中心附近最大风力17级以上（62米/秒），中心最低气压为915百帕，七级风圈半径220~230公里，十级风圈半径70公里，十二级风圈半径40公里。

预计，“轩岚诺”将以每小时30公里左右的速度向西偏南转西南方向移动，9月1~2日将在琉球群岛以东洋面停滞或回旋，而后转向北偏西方向移动，3日夜间移入东海东南部海面。未来4~5天“轩岚诺”将维持超强台风级的强度，最大强度可达17级以上（62~70米/秒）。



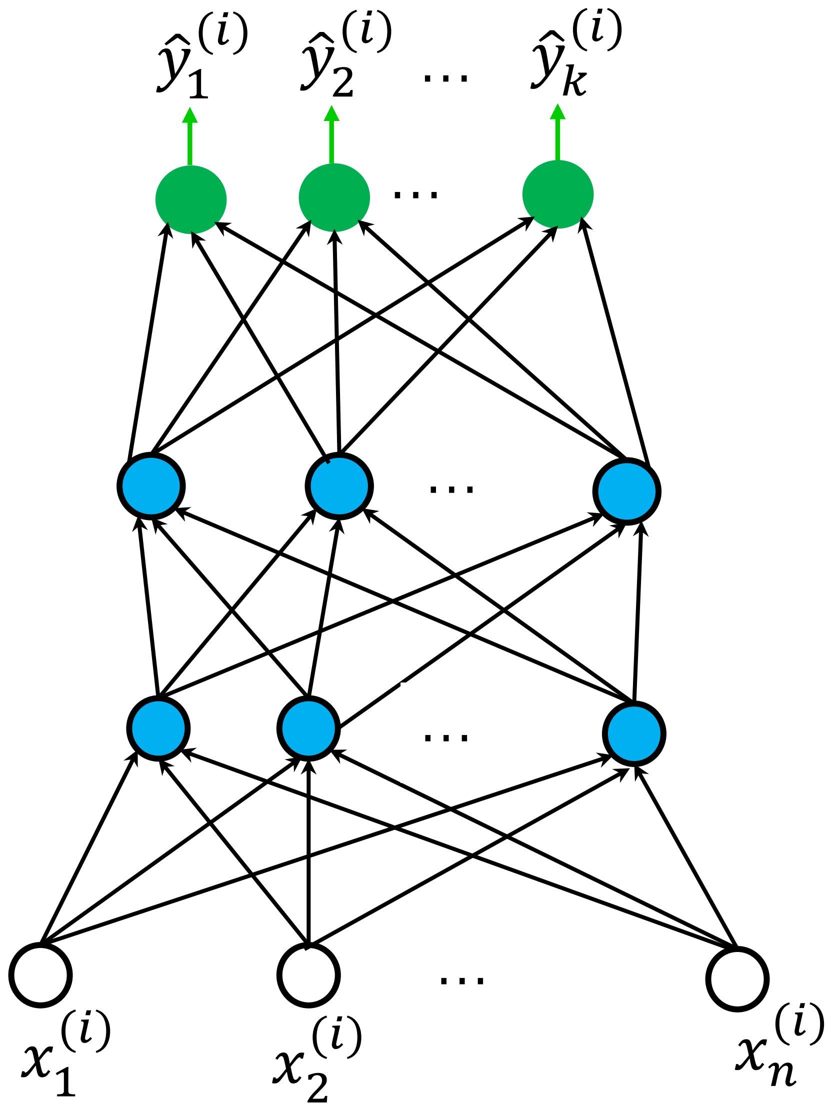
如何学习出神经网络模型： loss function

- Given $\mathbb{X} = \{\langle \boldsymbol{x}^{(1)}, y^{(1)} \rangle, \langle \boldsymbol{x}^{(2)}, y^{(2)} \rangle \dots, \langle \boldsymbol{x}^{(m)}, y^{(m)} \rangle\}$
- Translate $y^{(i)}$ to be a k-dim one-hot vector $\boldsymbol{y}^{(i)}$

$$\begin{bmatrix} \boldsymbol{y}^{(1)} \\ \boldsymbol{y}^{(2)} \\ \dots \\ \boldsymbol{y}^{(i)} \\ \dots \\ \boldsymbol{y}^{(m)} \end{bmatrix} \Longleftrightarrow \begin{bmatrix} \hat{\boldsymbol{y}}^{(1)} \\ \hat{\boldsymbol{y}}^{(2)} \\ \dots \\ \hat{\boldsymbol{y}}^{(i)} \\ \dots \\ \hat{\boldsymbol{y}}^{(m)} \end{bmatrix}$$

Minimize loss:

$$\boldsymbol{L(W,b)} = \frac{1}{2}\sum_{i=1}^{m} \left\| \boldsymbol{y}^{(i)} - \hat{\boldsymbol{y}}^{(i)} \right\|^2$$

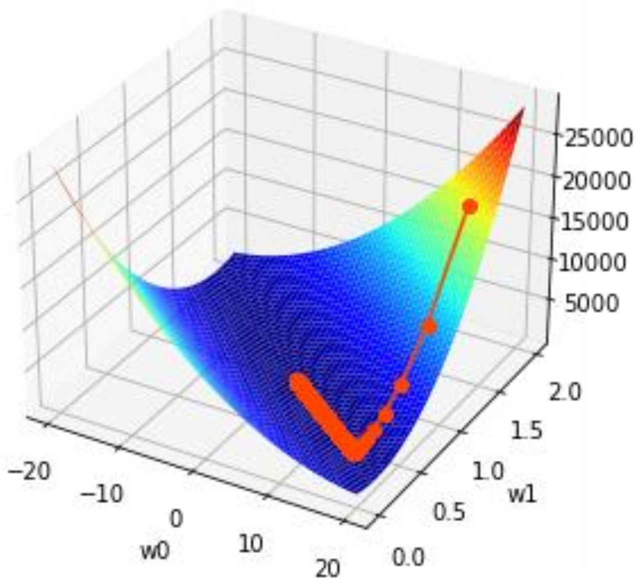


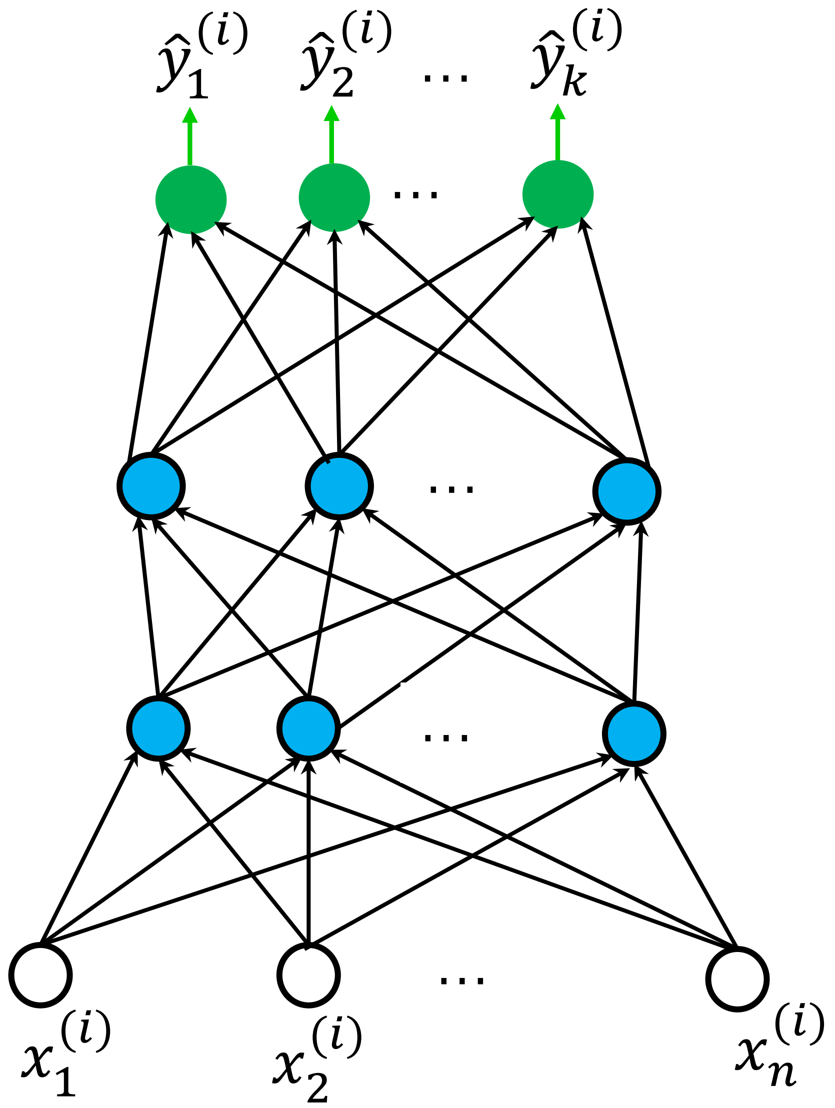
如何学习出神经网络模型： 优化问题

$$L(W,b)= \frac{1}{2}\sum_{i=1}^{m}\| y^{(i)} - \hat{y}^{(i)} \|^2$$

$$\hat{y}^{(i)} = f\left(x^{(i)}\right)$$

$$= \sigma(W^L \ \sigma(W^{L-1} \ldots \sigma(W^2\sigma(W^1 a^0 + b^1) + b^2) \ldots + b^L)$$



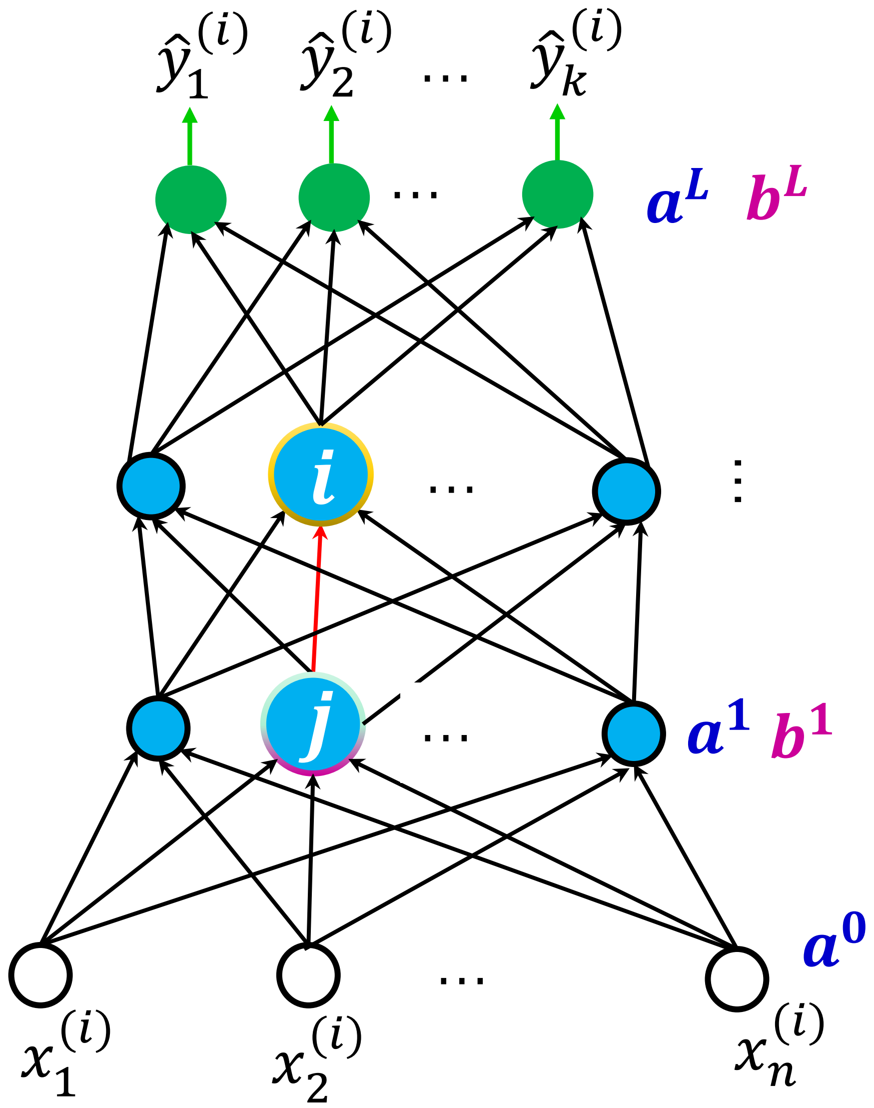


如何学习出神经网络模型：梯度下降法

$$L(W,b)=\frac{1}{2}\sum_{i=1}^{m}\left\| y^{(i)}-\hat{y}^{(i)}\right\|^2$$

$$\left.\begin{array}{l}\begin{bmatrix} w_{11}^1 & \dots & w_{1d^0}^1 \\ \dots & w_{ij}^1 & \dots \\ w_{d^1d^0}^1 & \dots & w_{d^1d^0}^1 \end{bmatrix} \cdots \\ \begin{bmatrix} b_1^1 \\ b_2^1 \\ \dots \\ b_i^1 \\ \dots \\ b_{d^1}^1 \end{bmatrix} \cdots \end{array}\right\} \nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_{11}^1} \\ \frac{\partial L}{\partial w_{12}^1} \\ \vdots \\ \frac{\partial L}{\partial w_{ij}^r} \\ \vdots \end{bmatrix}$$

$$w_{ij}^{r(new)}=w_{ij}^{r(old)}-\eta\frac{\partial L}{\partial w_{ij}^r}$$



$$L(W,b) = \frac{1}{2}\sum_{i=1}^{m}\left\| y^{(i)} - \hat{y}^{(i)} \right\|^2$$

$$a_i^r = a(z_i^r):$$

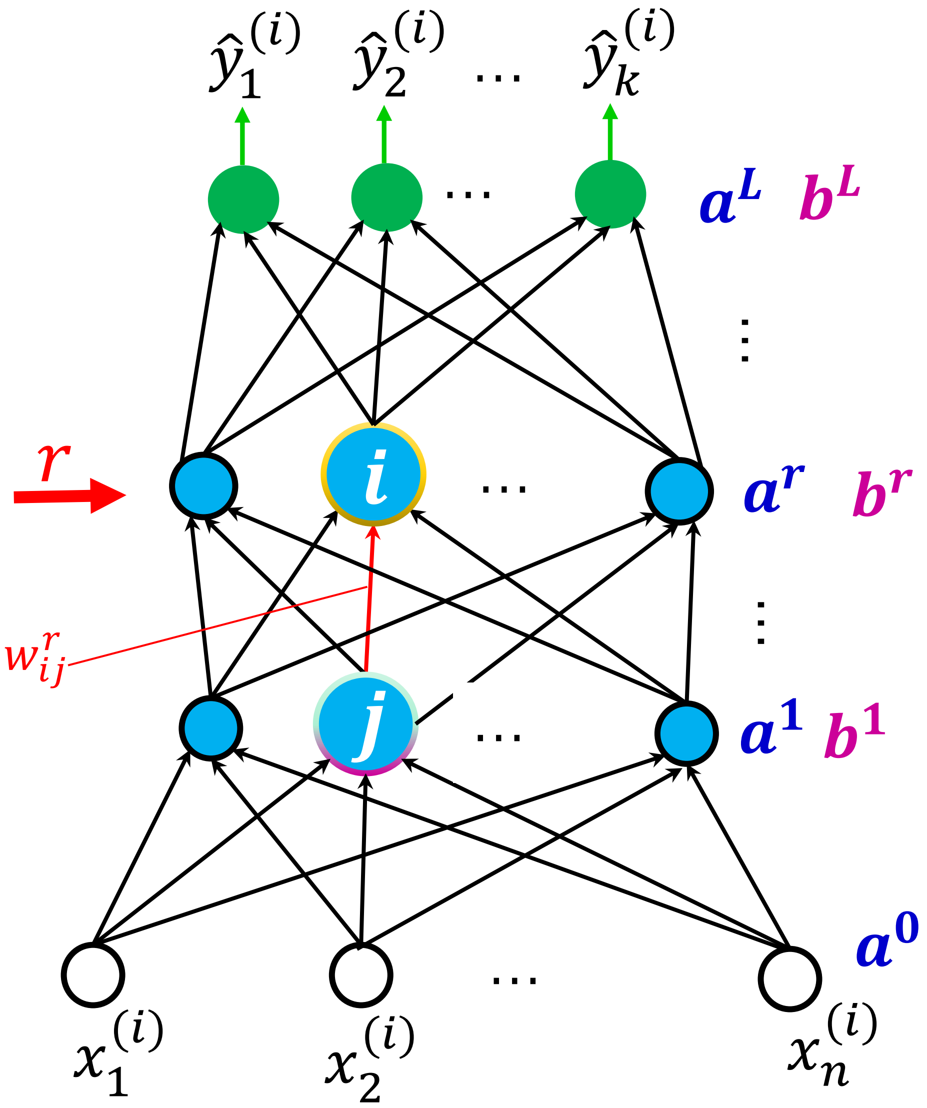
$$z_i^r = \sum_{j=1}^{d^{r-1}} w_{ij}^r \cdot a_j^{r-1} + b_i^r$$

$$\frac{\partial L}{\partial w_{ij}^r} = ?$$

$$\hat{y} = f(x)$$

$$a(W^L\, a(W^{L-1} \ldots a(W^r \ldots a(W^1 x + b^1) \ldots + b^r) \ldots + b^L)$$

$$w_{ij}^r \cdot a_j^{r-1} \Longrightarrow z_i^r \Longrightarrow a_i^r \quad \cdots \quad a^L = \hat{y} \Longrightarrow L(.)$$



Chain rule for derivation:

$$y = f(u), \qquad u = g(x)$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

BP（Back Propagation）算法: basic idea

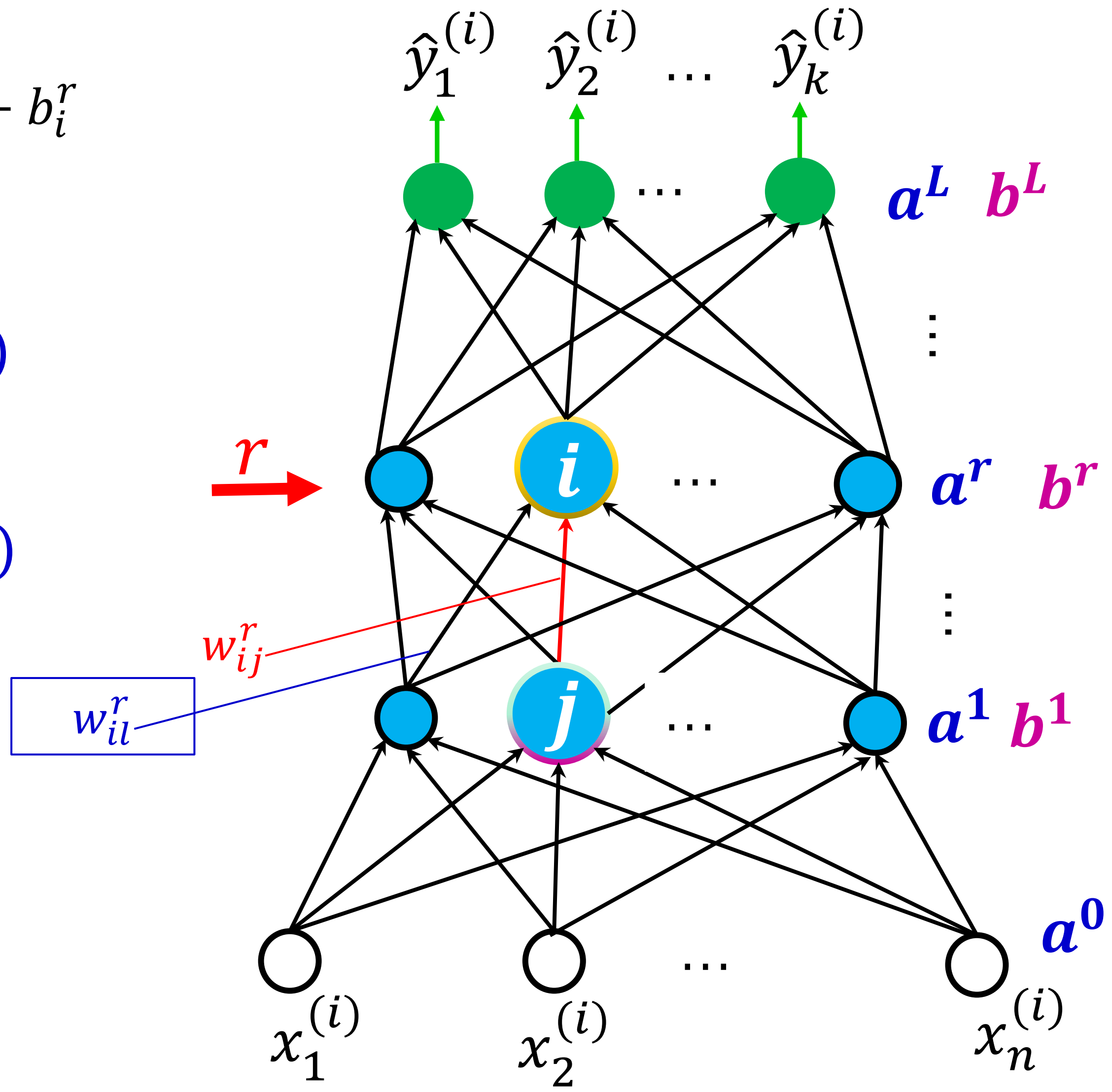
$$L(W,b)=\frac{1}{2}\sum_{i=1}^{m}\left\| y^{(i)}-\hat{y}^{(i)}\right\|^2 \qquad z_i^r=\sum_{j=1}^{d^{r-1}} w_{ij}^r\cdot a_j^{r-1}+b_i^r$$

$$w_{ij}^r\cdot a_j^{r-1} \Longrightarrow z_i^r \Longrightarrow a_i^r \quad \cdots \quad a^L=\hat{y} \Longrightarrow L(.)$$

$$w_{il}^r\cdot a_l^{r-1} \Longrightarrow z_i^r \Longrightarrow a_i^r \quad \cdots \quad a^L=\hat{y} \Longrightarrow L(.)$$

$$\frac{\partial L}{\partial w_{ij}^r}=\frac{\partial L}{\partial a^L}\cdots\cdots\frac{\partial a^{r+1}}{\partial a_i^r}\frac{\partial a_i^r}{\partial z_i^r}\frac{\partial z_i^r}{\partial w_{ij}^r}$$

$$\frac{\partial L}{\partial b_i^r}=\frac{\partial L}{\partial a^L}\cdots\cdots\frac{\partial a^{r+1}}{\partial a_i^r}\frac{\partial a_i^r}{\partial z_i^r}\frac{\partial z_i^r}{\partial b_i^r}$$



$$L(W,b) = \frac{1}{2}\sum_{i=1}^{m}\left\| y^{(i)} - \hat{y}^{(i)} \right\|^2$$

$$z_i^r = \sum_{j=1}^{d^{r-1}} w_{ij}^r \cdot a_j^{r-1} + b_i^r$$

$$\frac{\partial L}{\partial w_{ij}^r} = \underbrace{\frac{\partial L}{\partial a^L} \cdots\cdots \frac{\partial a^{r+1}}{\partial a_i^r}\frac{\partial a_i^r}{\partial z_i^r}}_{\delta_i^r = \frac{\partial L}{\partial z_i^r}} \frac{\partial z_i^r}{\partial w_{ij}^r}$$

$$\frac{\partial z_i^r}{\partial w_{ij}^r} = a_j^{r-1}$$

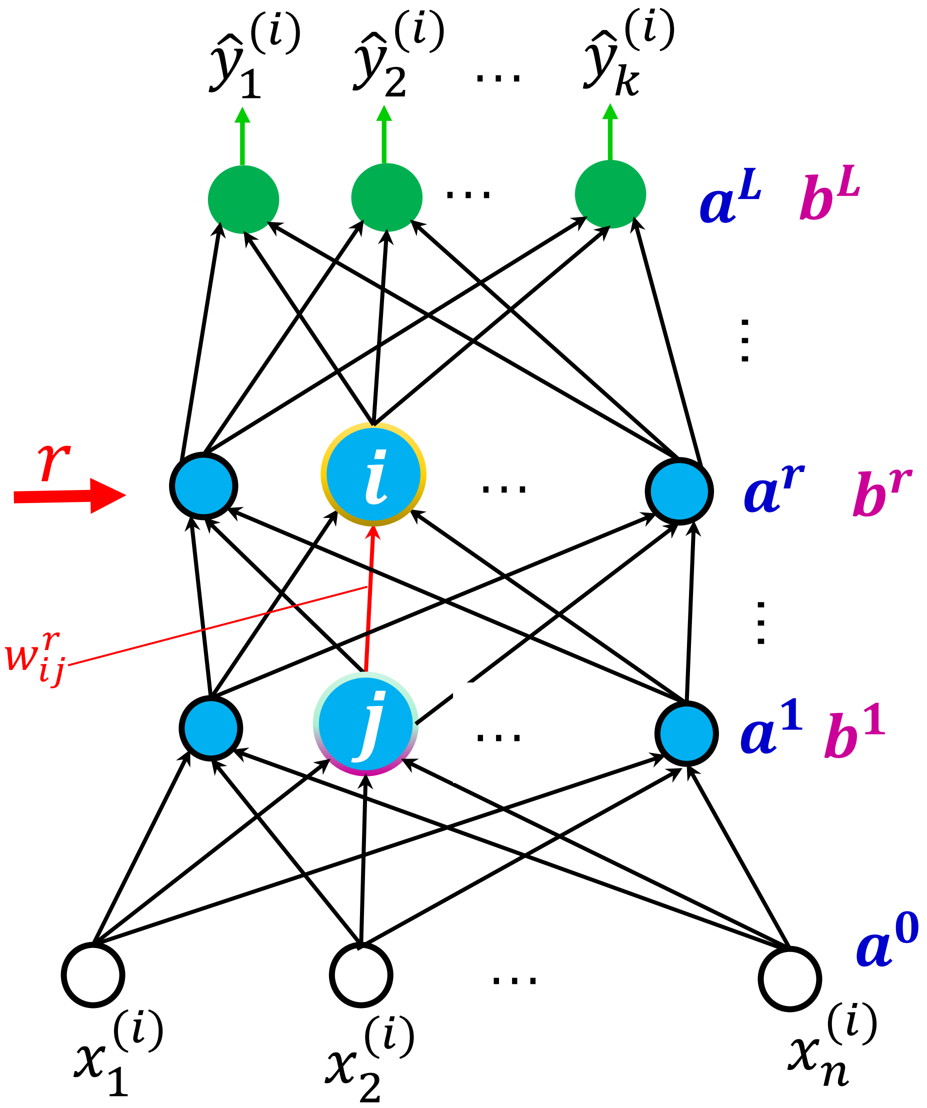
$$\delta_i^r = \frac{\partial L}{\partial z_i^r}$$

$$\frac{\partial a_i^r}{\partial z_i^r}:$$

a: $\sigma = \dfrac{1}{1+e^{-z}}$

$$\frac{\partial L}{\partial w_{ij}^r} = \delta_i^r \times a_j^{r-1}$$

$$\frac{\partial a_i^r}{\partial z_i^r} = a_i^r(1 - a_i^r)$$



$$L(W,b) = \frac{1}{2}\sum_{i=1}^{m}\| \mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)} \|^2 \qquad z_i^r = \sum_{j=1}^{d^{r-1}} w_{ij}^r \cdot a_j^{r-1} + b_i^r$$

$$\frac{\partial L}{\partial w_{ij}^r} = \underbrace{\frac{\partial L}{\partial \mathbf{a}^L} \cdots \cdots \frac{\partial \mathbf{a}^{r+1}}{\partial a_i^r} \frac{\partial a_i^r}{\partial z_i^r}}_{\delta_i^r = \frac{\partial L}{\partial z_i^r}} \frac{\partial z_i^r}{\partial w_{ij}^r} \qquad \frac{\partial z_i^r}{\partial w_{ij}^r} = a_j^{r-1}$$

$$\delta_i^r = \frac{\partial L}{\partial z_i^r}$$

$\frac{\partial a_i^r}{\partial z_i^r}$:

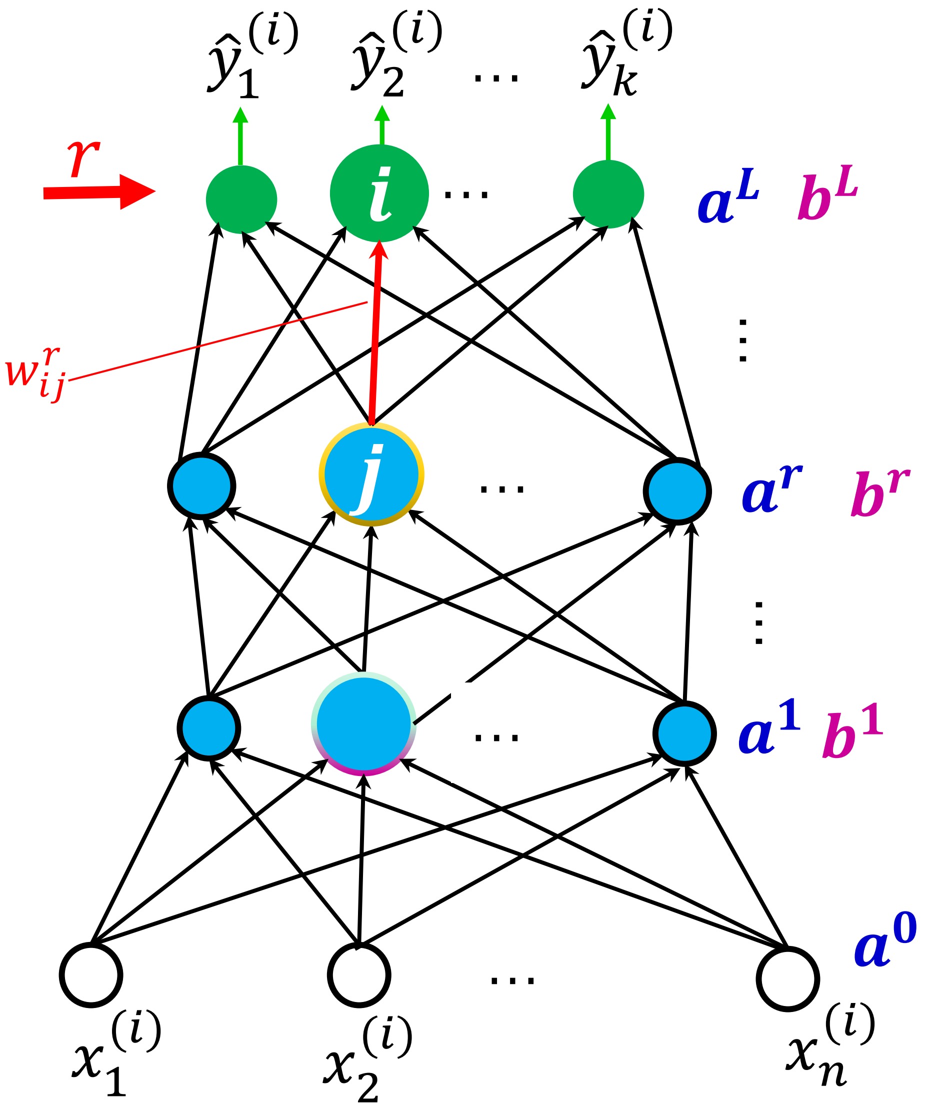
a: $\sigma = \frac{1}{1+e^{-z}}$

$$\frac{\partial a_i^r}{\partial z_i^r} = a_i^r(1 - a_i^r)$$

$$\frac{\partial L}{\partial w_{ij}^r} = \delta_i^r \times a_j^{r-1}$$

Output Layer($r = L$):

$$\delta_i^r = -(y_i - a_i^L)a_i^L(1 - a_i^L)$$



$$L(W,b) = \frac{1}{2}\sum_{i=1}^{m}\left\| y^{(i)} - \hat{y}^{(i)} \right\|^2 \qquad z_i^r = \sum_{j=1}^{d^{r-1}} w_{ij}^r \cdot a_j^{r-1} + b_i^r$$

$$\frac{\partial L}{\partial w_{ij}^r} = \underbrace{\frac{\partial L}{\partial a^L} \cdots\cdots \frac{\partial a^{r+1}}{\partial a_i^r} \frac{\partial a_i^r}{\partial z_i^r}}_{\delta_i^r = \frac{\partial L}{\partial z_i^r}} \frac{\partial z_i^r}{\partial w_{ij}^r}$$

$$\delta_i^r = \frac{\partial L}{\partial z_i^r}$$

$$\frac{\partial z_i^r}{\partial w_{ij}^r} = a_j^{r-1}$$

$\frac{\partial a_i^r}{\partial z_i^r}$:

a: $\sigma = \frac{1}{1+e^{-z}}$

$$\frac{\partial a_i^r}{\partial z_i^r} = a_i^r(1- a_i^r)$$

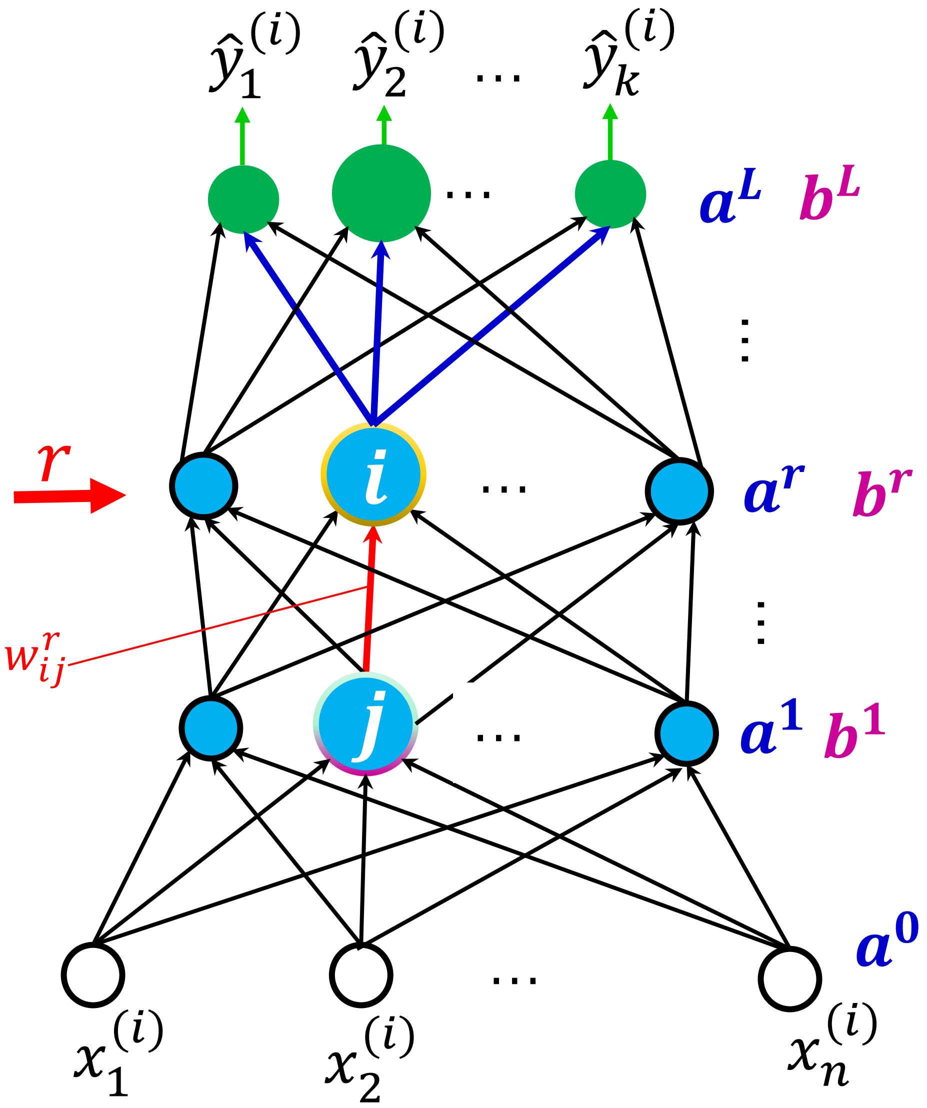
$$\frac{\partial L}{\partial w_{ij}^r} = \delta_i^r \times a_j^{r-1}$$

Output Layer($r = L$):

$$\delta_i^r = -(y_i - a_i^L)a_i^L(1 - a_i^L)$$

Hidden Layer ($r < L$):

$$\delta_i^r = a_i^r(1 - a_i^r) \sum_{s \in Next(i)} \delta_s^{r+1} w_{si}^{r+1}$$



批处理反向传播（BP with Batch Training）

$$L(W,b) = \frac{1}{2}\sum_{i=1}^{m}\| y^{(i)} - \hat{y}^{(i)} \|^2$$

- Given $\mathbb{X} = \{\langle x^{(1)}, y^{(1)}\rangle, \langle x^{(2)}, y^{(2)}\rangle \ldots, \langle x^{(m)}, y^{(m)}\rangle\}$, and hyper parameters
  - Translate $y^{(i)}$ to be a k-dim one-hot vector $y^{(i)}$ for classification task

$$\frac{\partial L}{\partial w_{ij}^{r}} = \delta_i^r \times a_j^{r-1}$$

1. Initialize network weights with small random values
2. **Do**

   ① Initialize Loss: $L(W,b)=0$

   $$\delta_i^r = -(y_i - a_i^L) a_i^L (1 - a_i^L)$$

   ② **For each sample $\langle x^{(N)}, y^{(N)}\rangle$**

   $$\delta_i^r = a_i^r (1 - a_i^r) \sum_{s \in Next(i)} \delta_s^{r+1} w_{si}^{r+1}$$

   a. **Compute the output of $\hat{y}^{(N)} = f(x^{(N)})$**

   b. **Update the loss: $L(W,b) = L(W,b) + \frac{1}{2}(y^{(N)} - \hat{y}^{(N)})^2$**

   ③ **Calculate the gradient with $L(W,b)$ and current weights $w_{ij}^{r(old)}$**

   ④ **Update the weights:** $w_{ij}^{r(new)} = w_{ij}^{r(old)} - \eta \frac{\partial L}{\partial w_{ij}^{r}}$

3. **Until stopping criteria satisfied**

随机梯度下降算法（BP with SGD）

$$L(W,b) = \frac{1}{2}\sum_{i=1}^{m} \| y^{(i)} - \hat{y}^{(i)} \|^2$$

- Given $\mathbb{X} = \{\langle x^{(1)}, y^{(1)}\rangle, \langle x^{(2)}, y^{(2)}\rangle \ldots, \langle x^{(m)}, y^{(m)}\rangle\}$, and hyper parameters
  - Translate $y^{(i)}$ to be a k-dim one-hot vector $y^{(i)}$ for classification task

$$\frac{\partial L}{\partial w_{ij}^r} = \delta_i^r \times a_j^{r-1}$$

1. Initialize network weights with small random values
2. **Do**

   ① **For each sample $\langle x^{(N)}, y^{(N)}\rangle$**

$$\delta_i^r = -(y_i - a_i^L) a_i^L (1 - a_i^L)$$

   a. **Compute the output of $\hat{y}^{(N)} = f(x^{(N)})$**

$$\delta_i^r = a_i^r (1 - a_i^r) \sum_{s \in Next(i)} \delta_s^{r+1} w_{si}^{r+1}$$

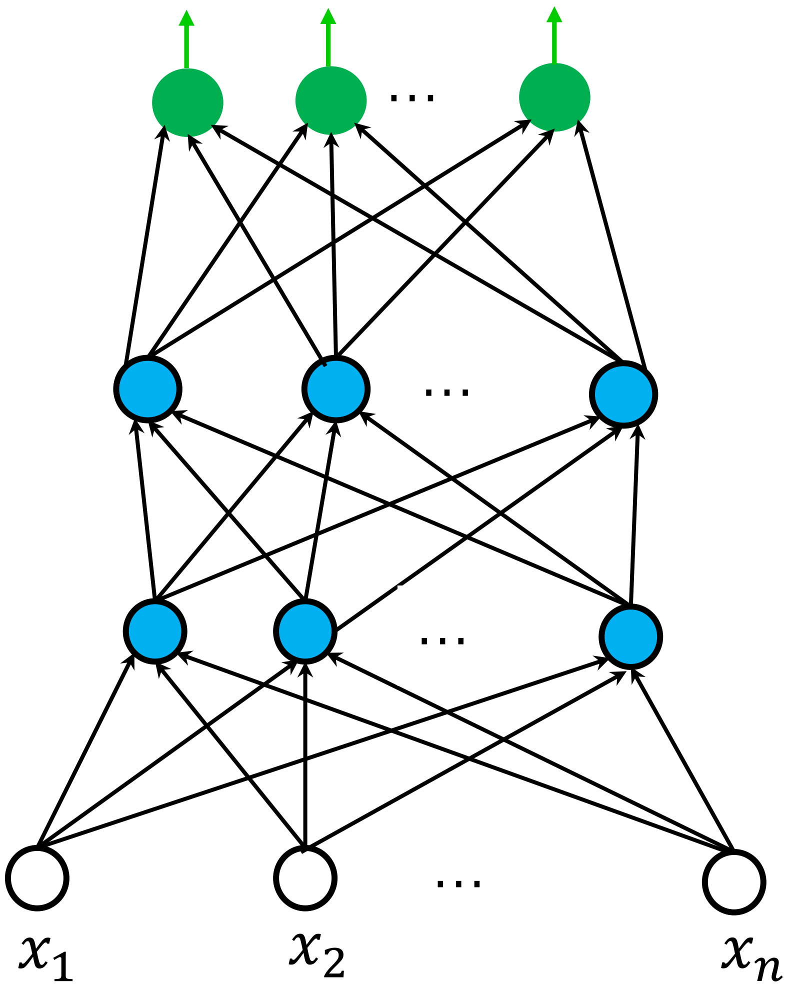
   b. **Compute the loss of $x^{(N)}$: $l(W,b) = \frac{1}{2}(y^{(N)} - \hat{y}^{(N)})^2$**

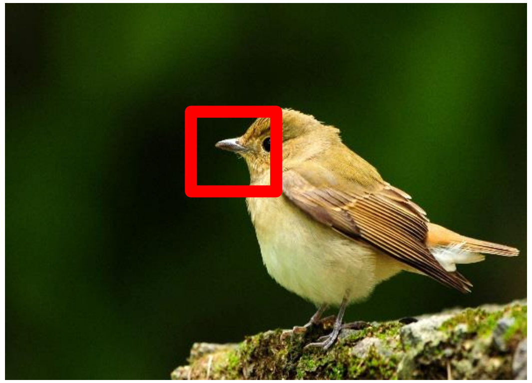
   c. **Calculate the gradient with $l(W,b)$ and current weights $w_{ij}^{r(old)}$**

   d. **Update the weights:** $w_{ij}^{r(new)} = w_{ij}^{r(old)} - \eta \frac{\partial L}{\partial w_{ij}^r}$

3. **Until stopping criteria satisfied**

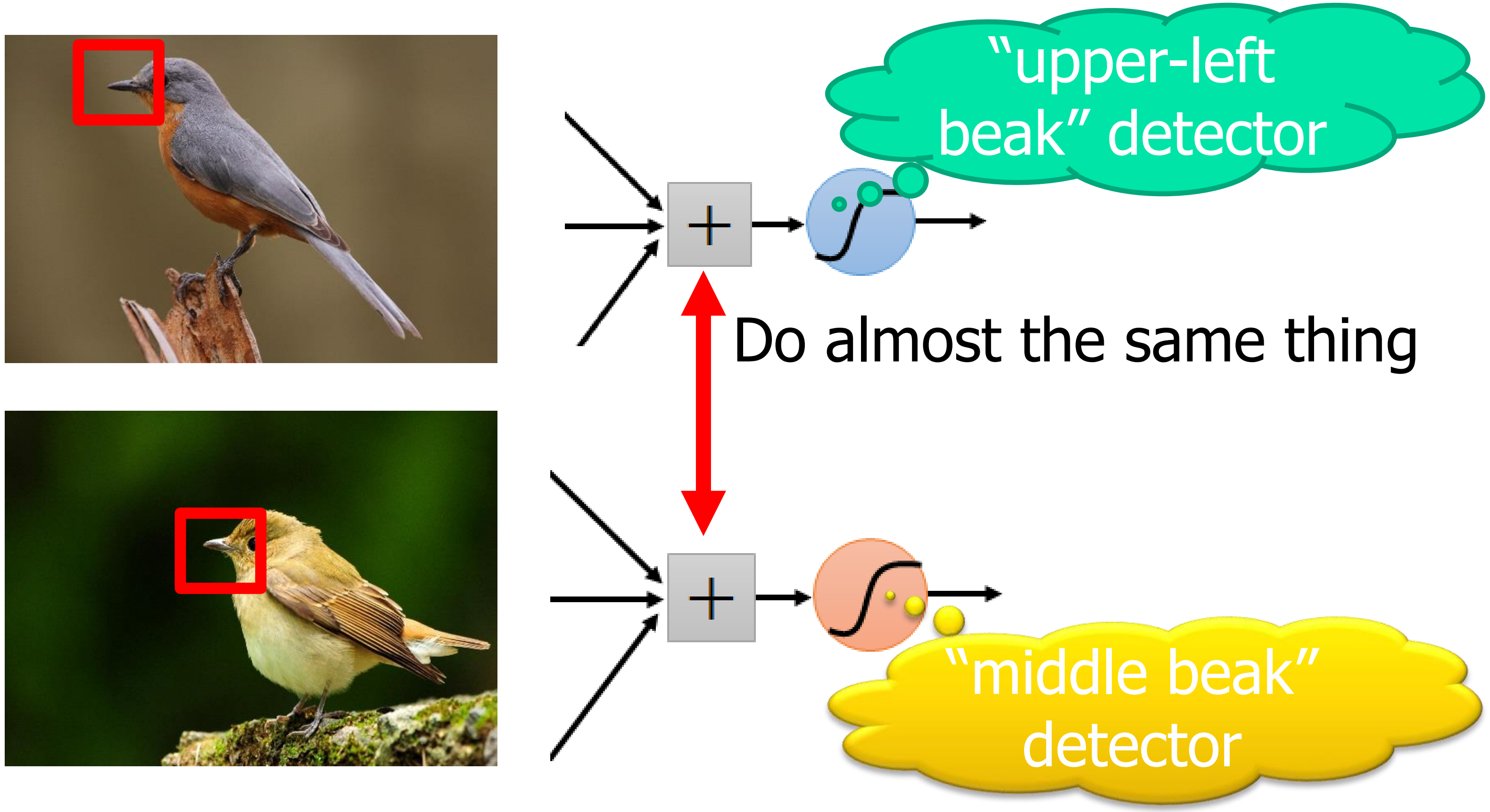
全连接网络的问题





卷积神经网络（CNN）：可挖掘局部模式

- The same patterns appear in different regions.



卷积神经网络（CNN）：可挖掘局部模式

- Subsampling the pixels will not change the object

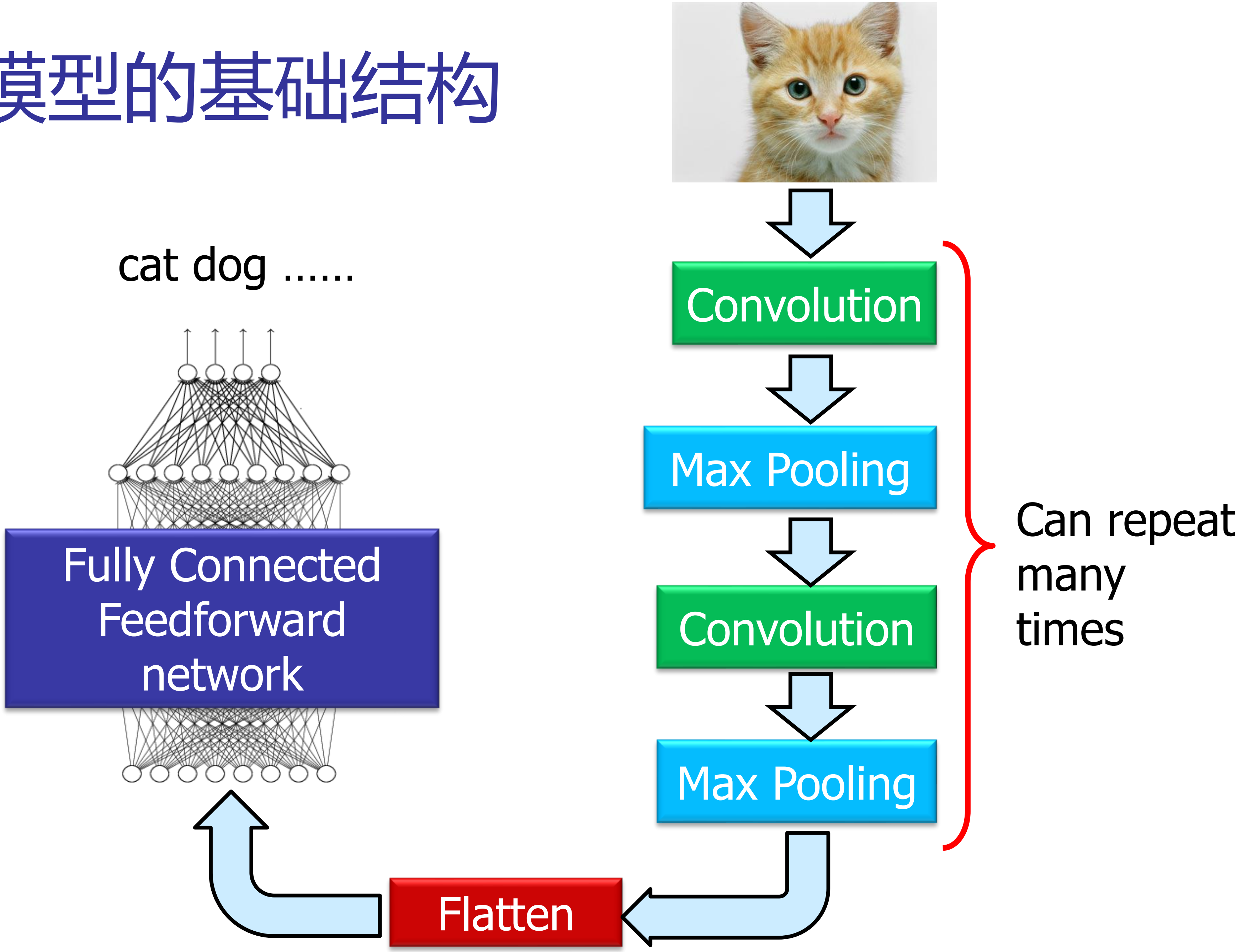
bird

bird

We can subsample the pixels to make image smaller

Less parameters for the network to process the image

CNN模型的基础结构



cat dog ……
Fully Connected Feedforward network
Convolution
Max Pooling
Convolution
Max Pooling
Can repeat many times
Flatten

CNN – 卷积（Convolution）

Those are the network parameters to be learned.

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

6 x 6 image

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

Filter 1

Matrix

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

Filter 2

Matrix

⋮ ⋮

Each filter detects a small pattern (3 x 3).

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

Filter 1

stride=1

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

6 x 6 image

3 -1

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

Filter 1

If stride=2

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

6 x 6 image

3 -3

We set stride=1 below

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

Filter 1

stride=1

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

6 x 6 image

$\boxed{3}$	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
$\boxed{3}$	-2	-2	-1

stride=1

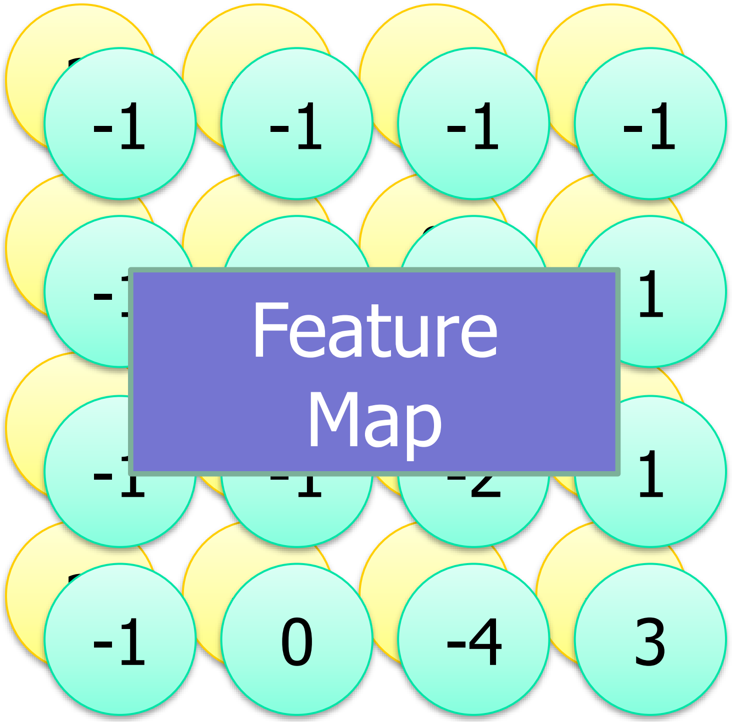
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

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

Filter 2

Do the same process for every filter



4 x 4 image

CNN – Colorful image

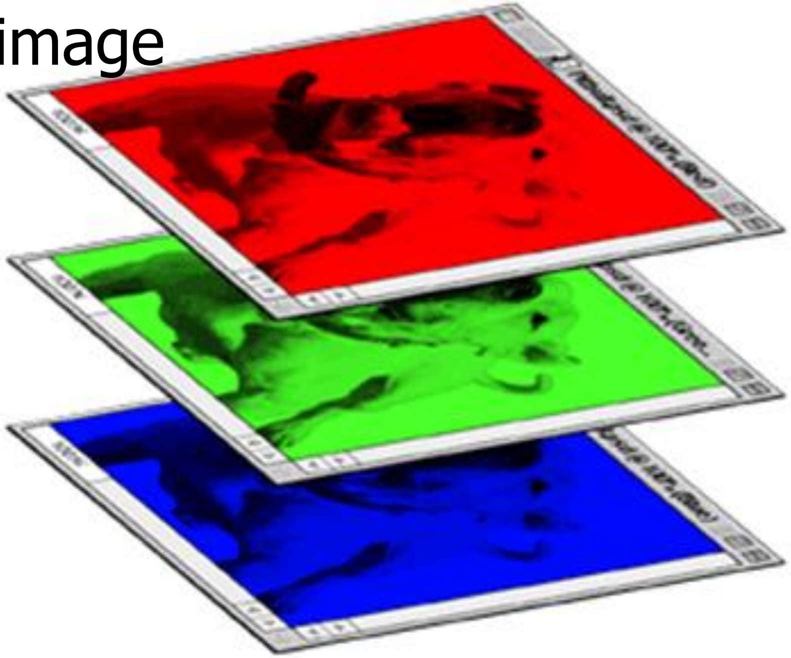
Filter 1

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

Filter 2

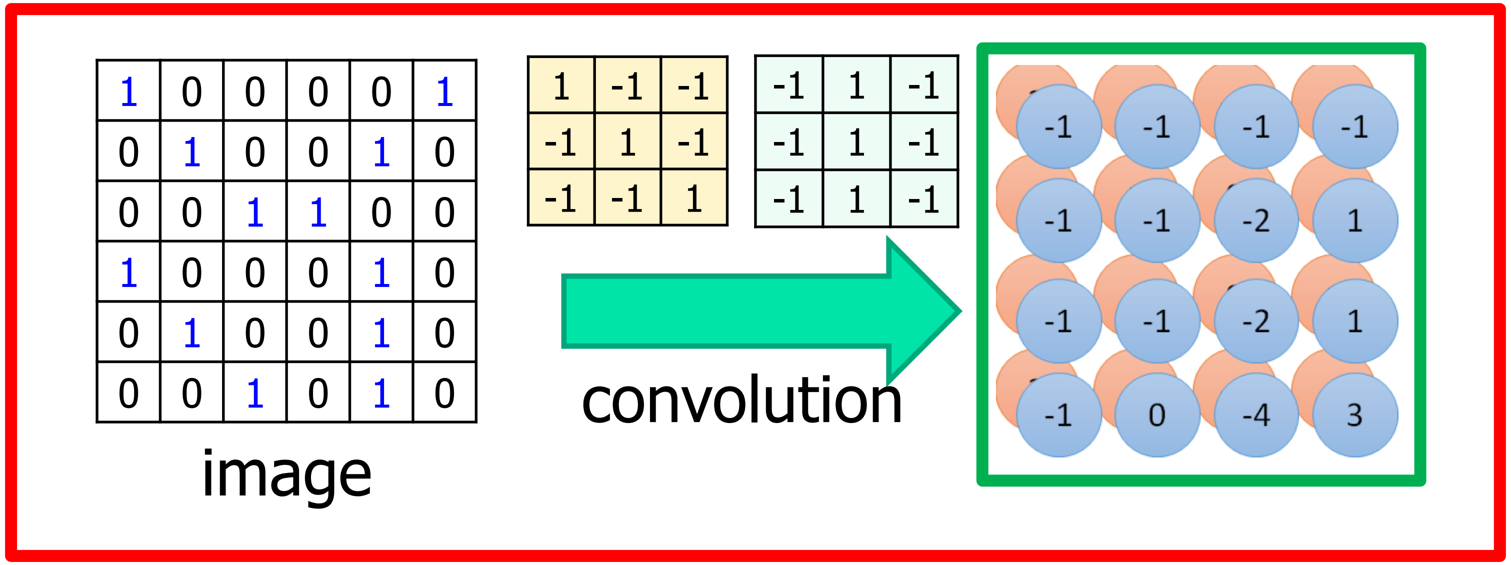
-1	1	-1
-1	1	-1
-1	1	-1

Colorful image

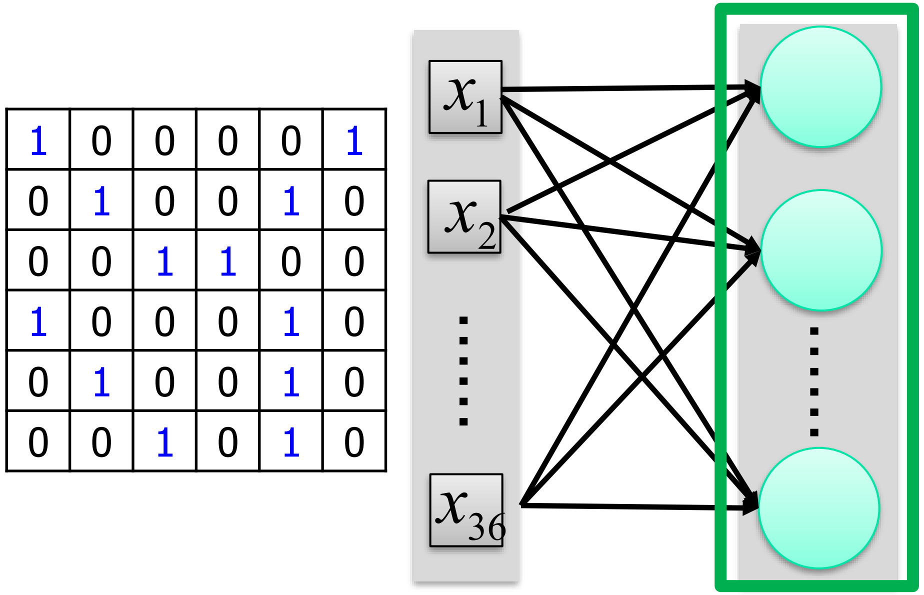


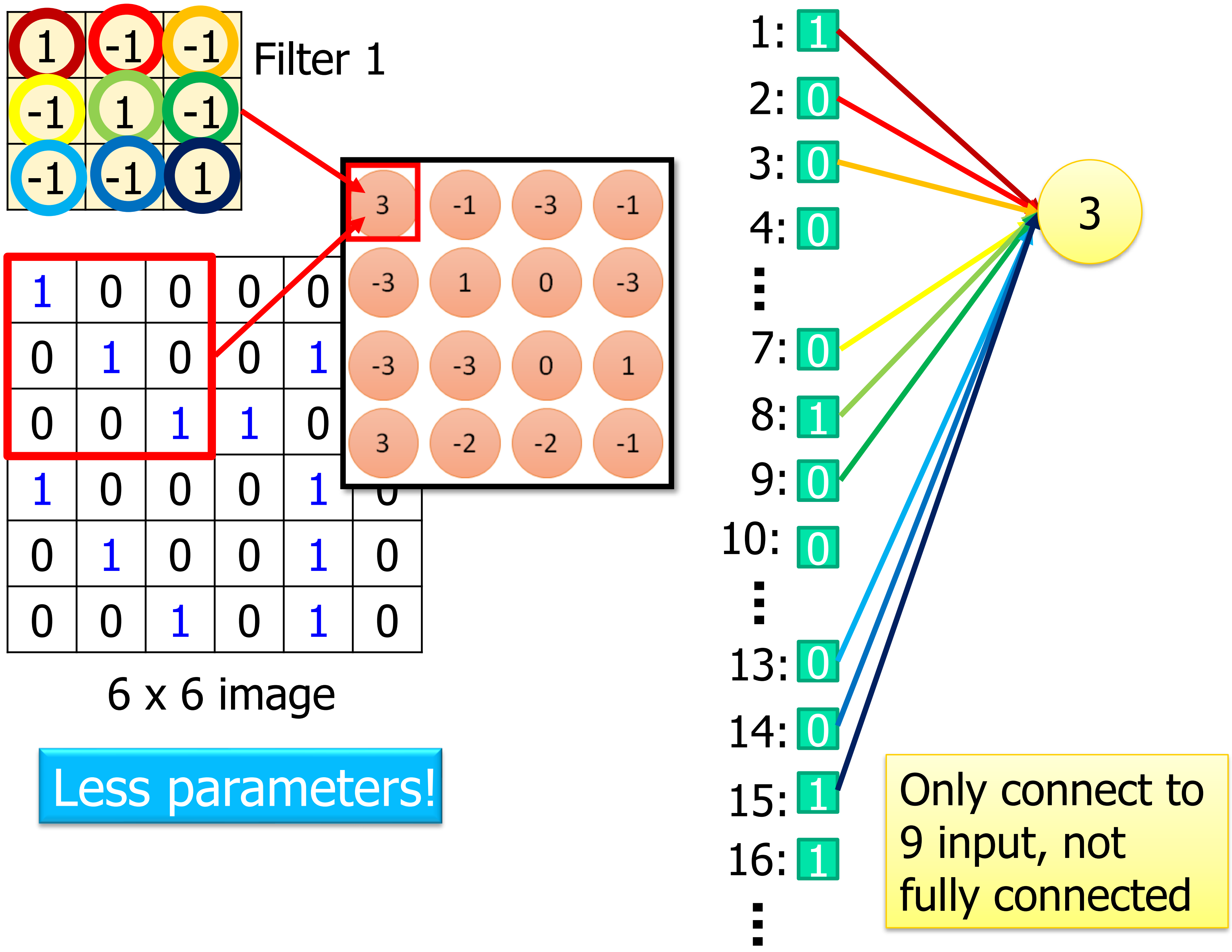
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

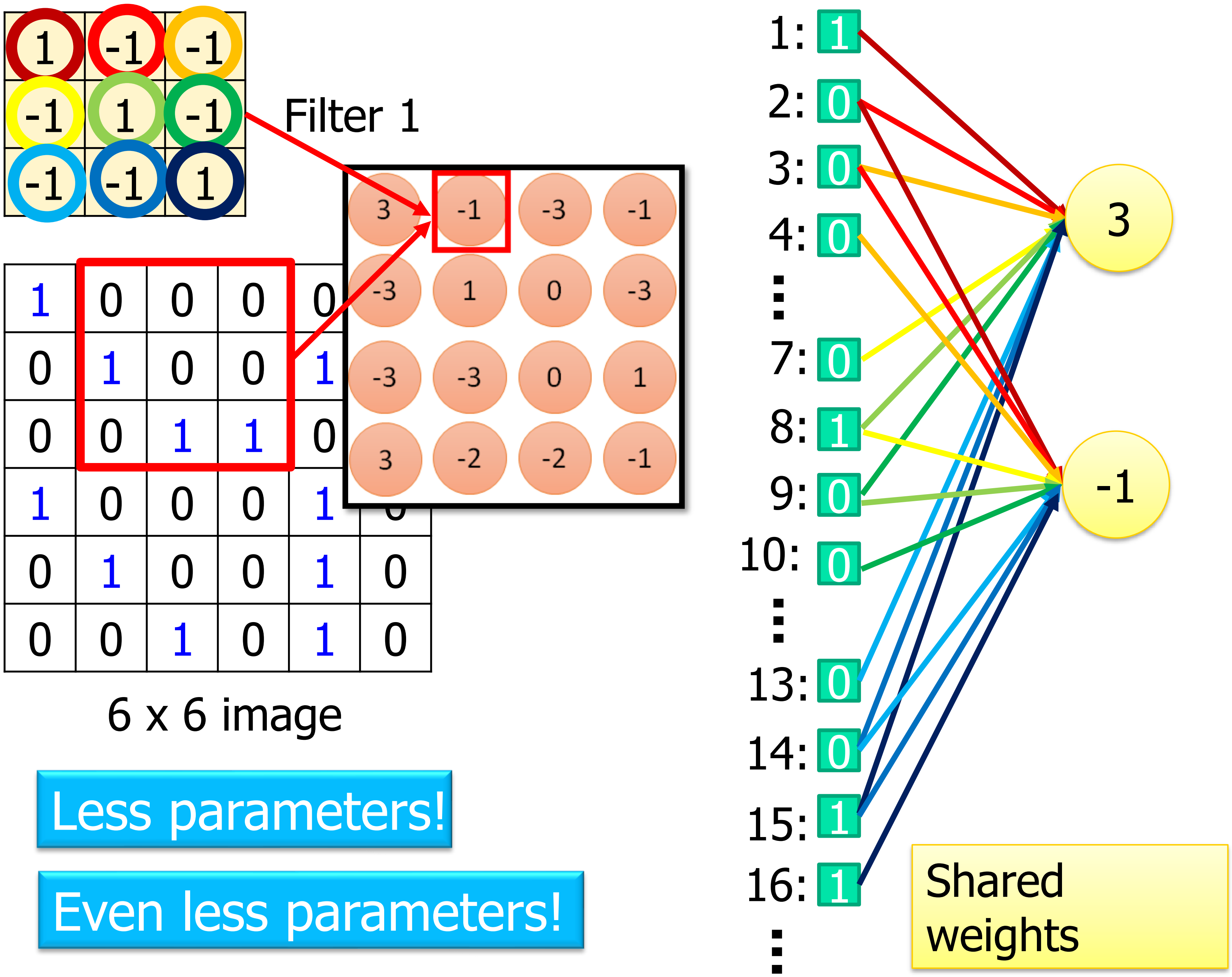
Convolution v.s. Fully Connected



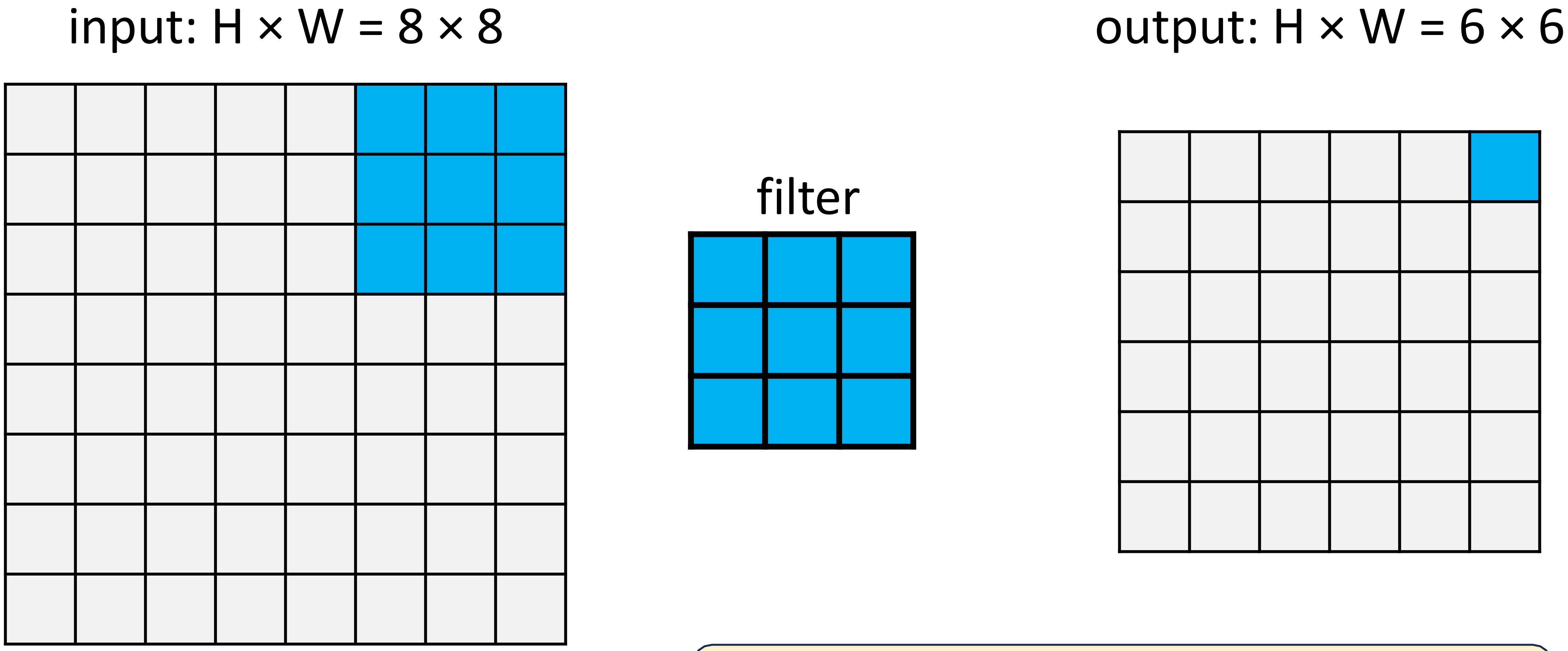
Fully-
connected




Filter 1
1 -1 -1
-1 1 -1
-1 -1 1
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
6 x 6 image
Less parameters!
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
Only connect to 9 input, not fully connected


1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
Less parameters!
Even less parameters!
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
-1
Shared weights

Convolution: 填充（padding）

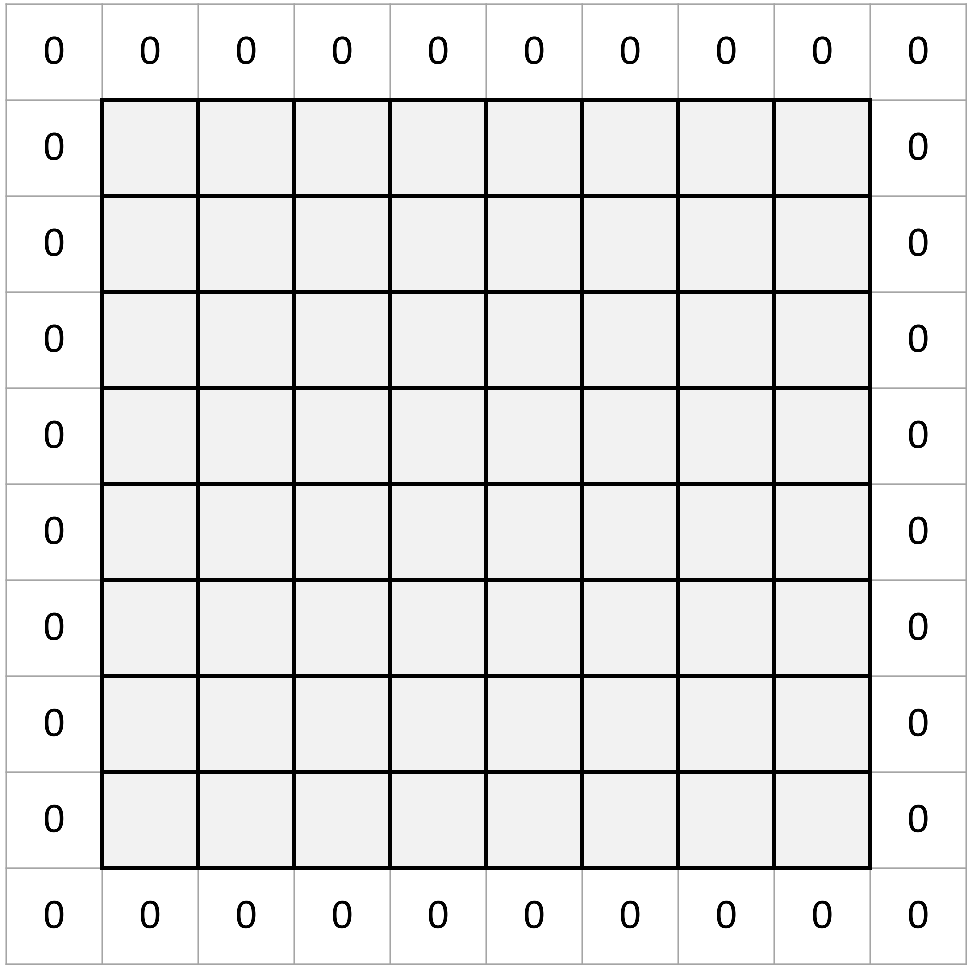


$$H_{out} = H_{in} - K_h + 1$$

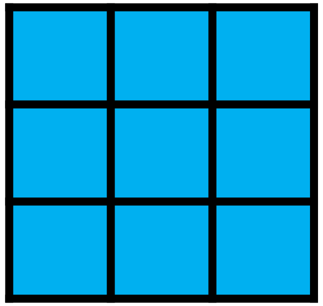
Convolution: 填充（padding）

- pad = ⌊kernel_size / 2⌋
- maintains feature map size

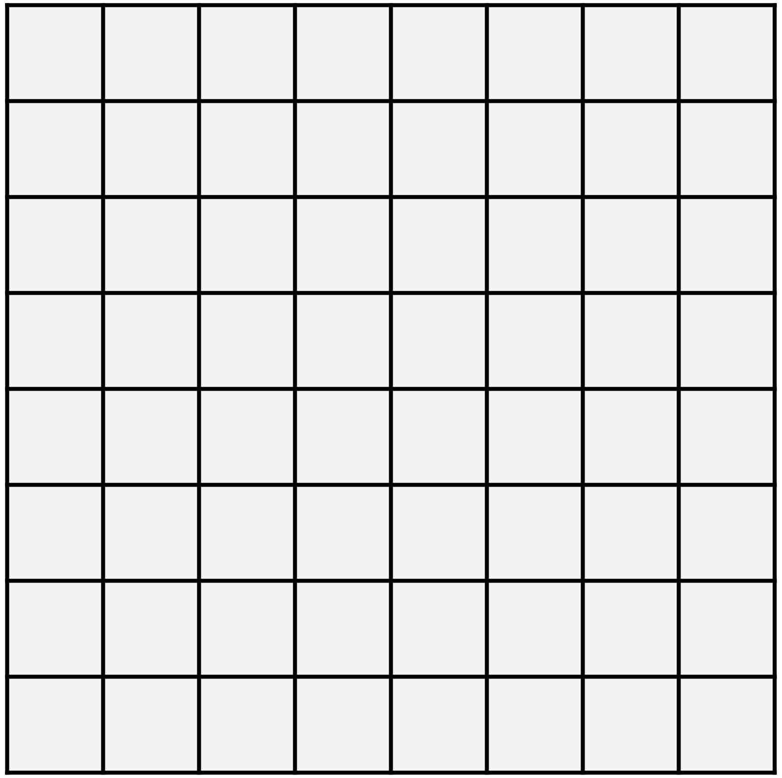
input: 8 × 8, + pad



filter

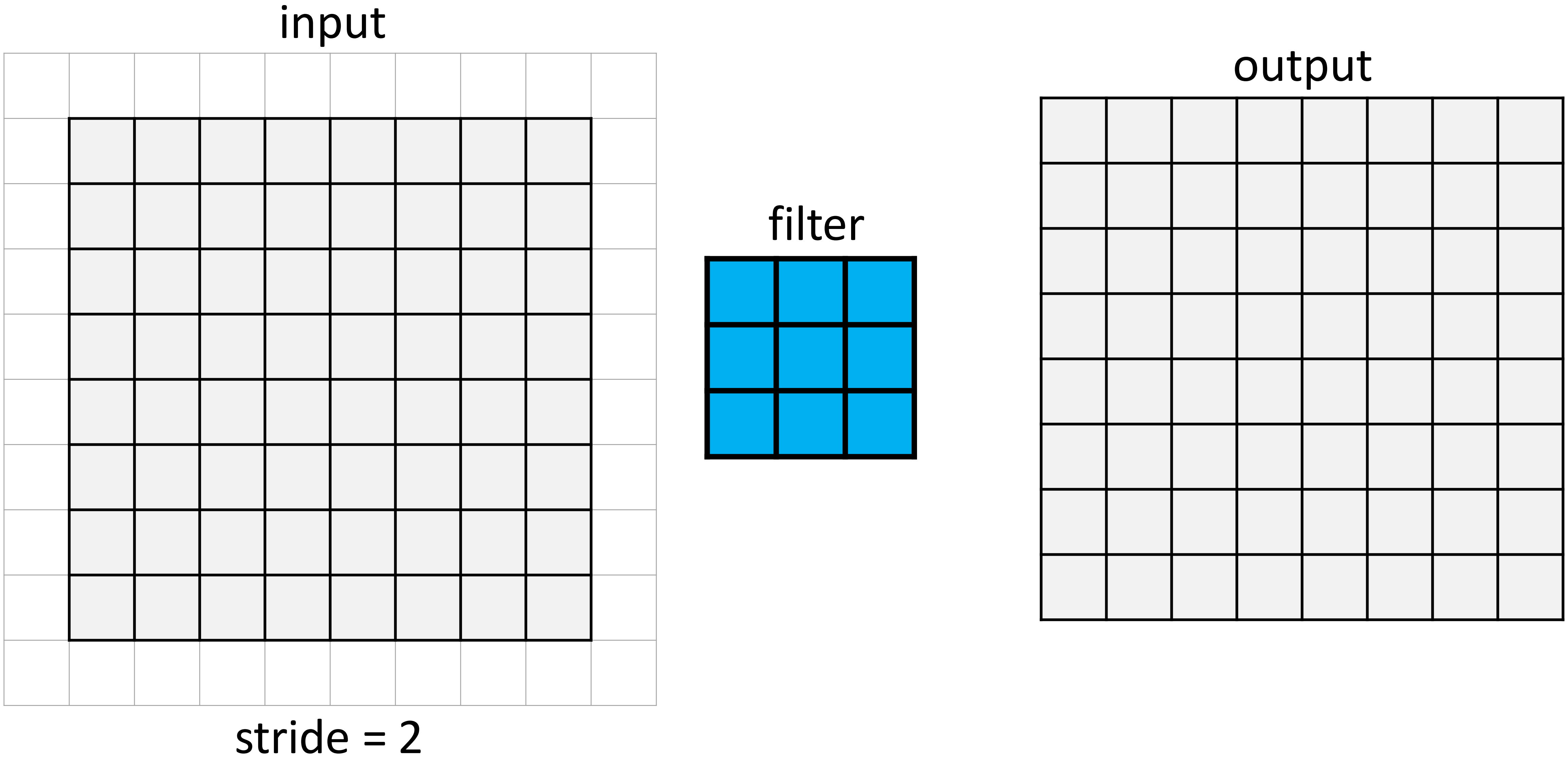


output: H × W = 8 × 8

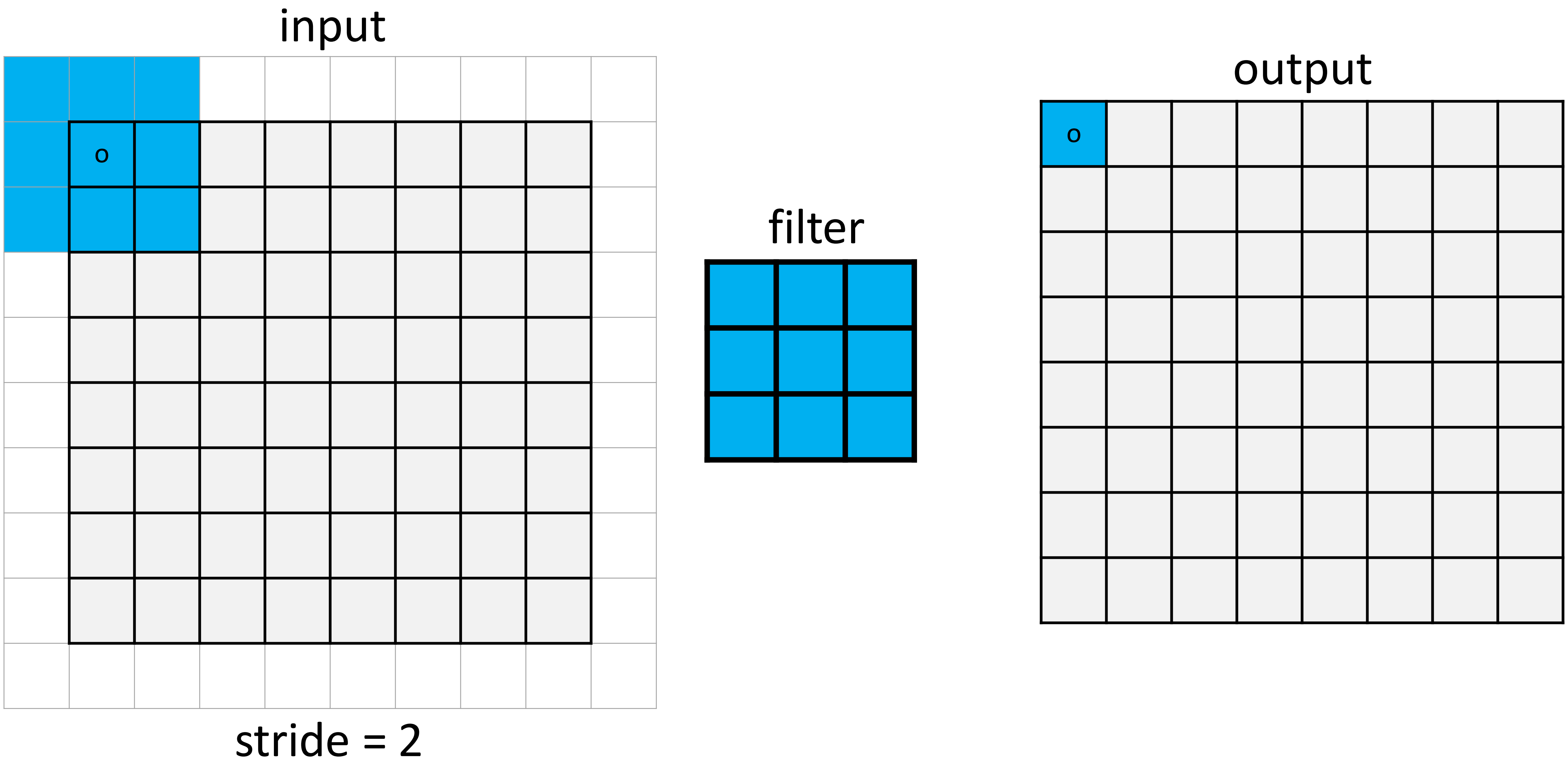


$$H_{out} = H_{in} + 2pad_h - K_h + 1$$

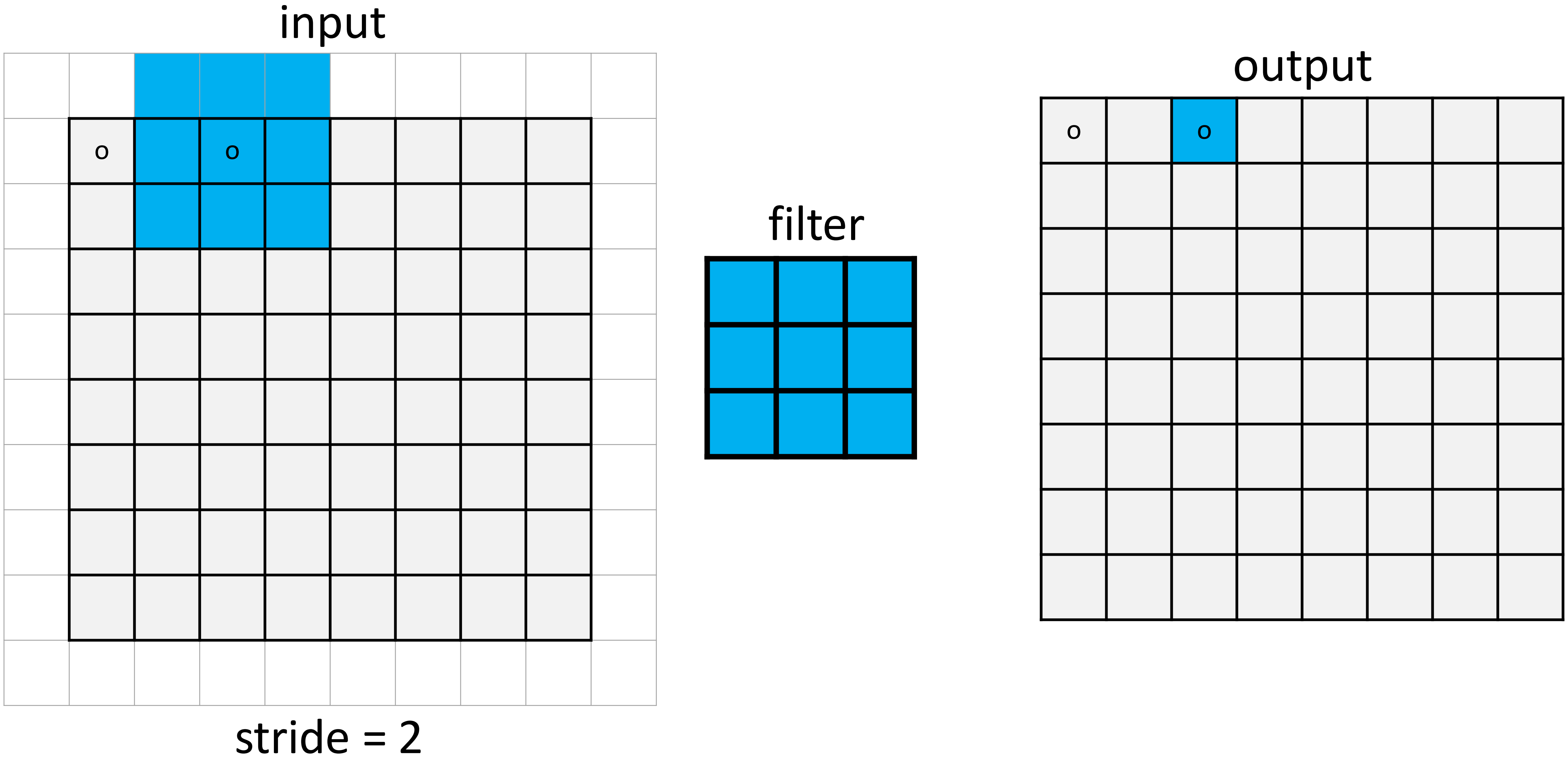
Convolution：步长（stride）



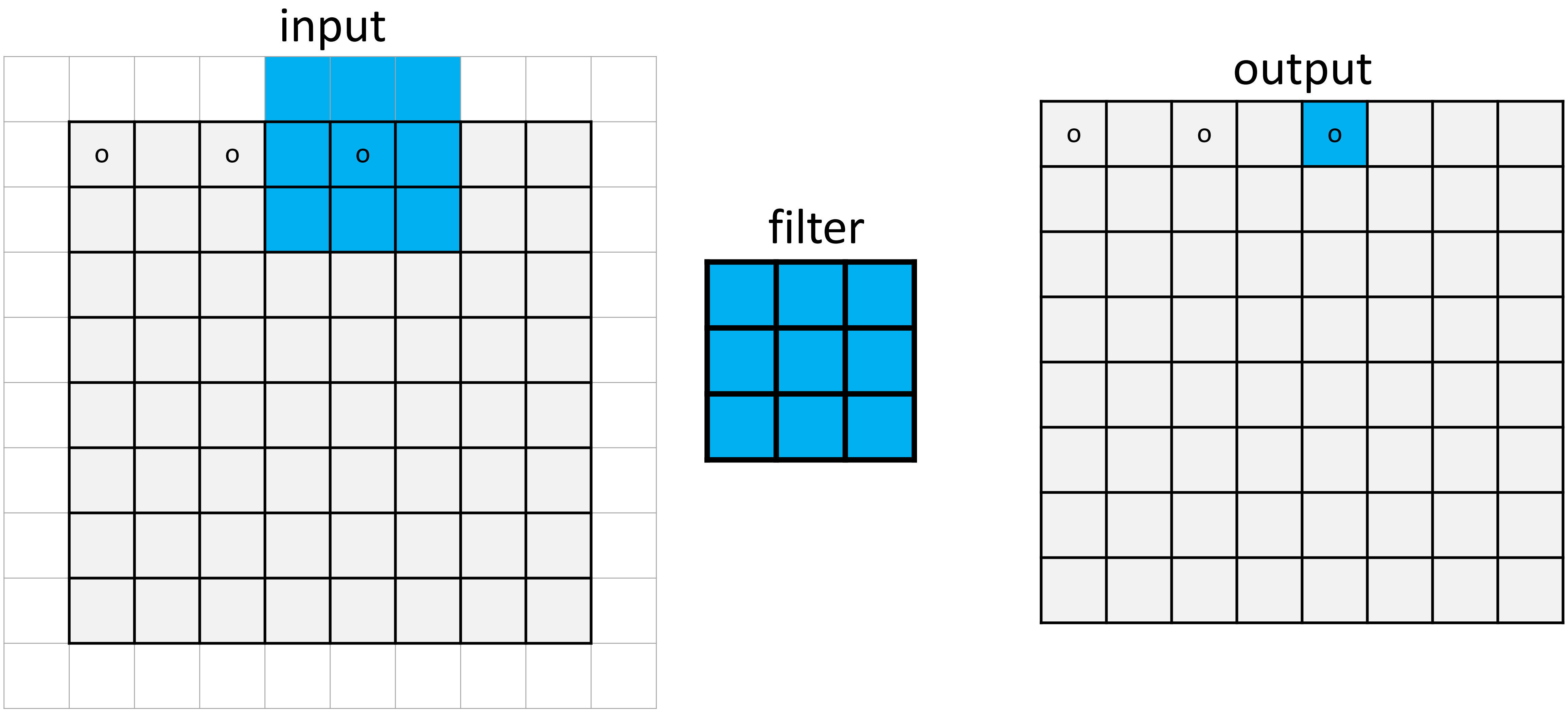
Convolution：步长（stride)



Convolution：步长（stride）

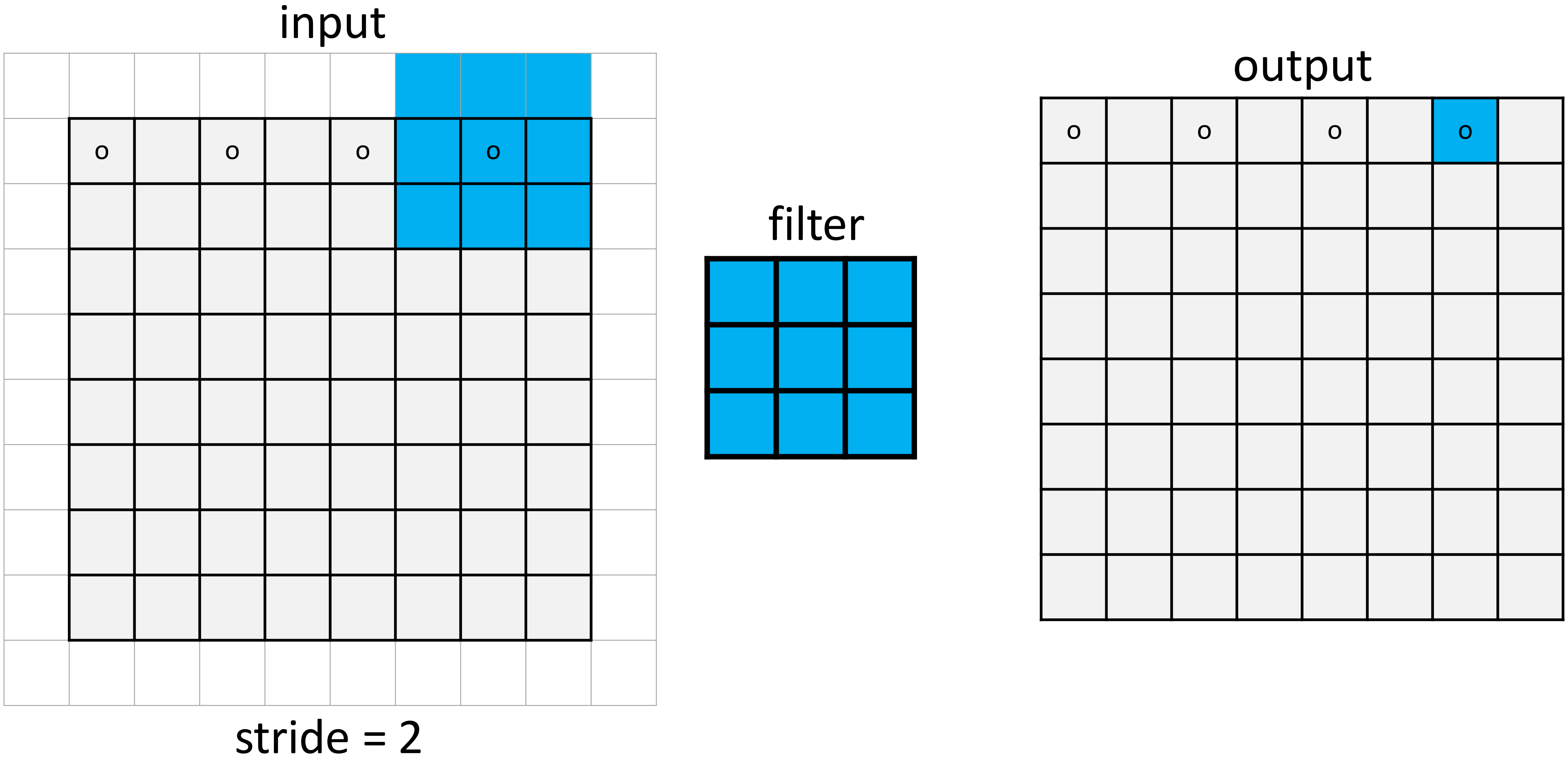


Convolution：步长（stride）

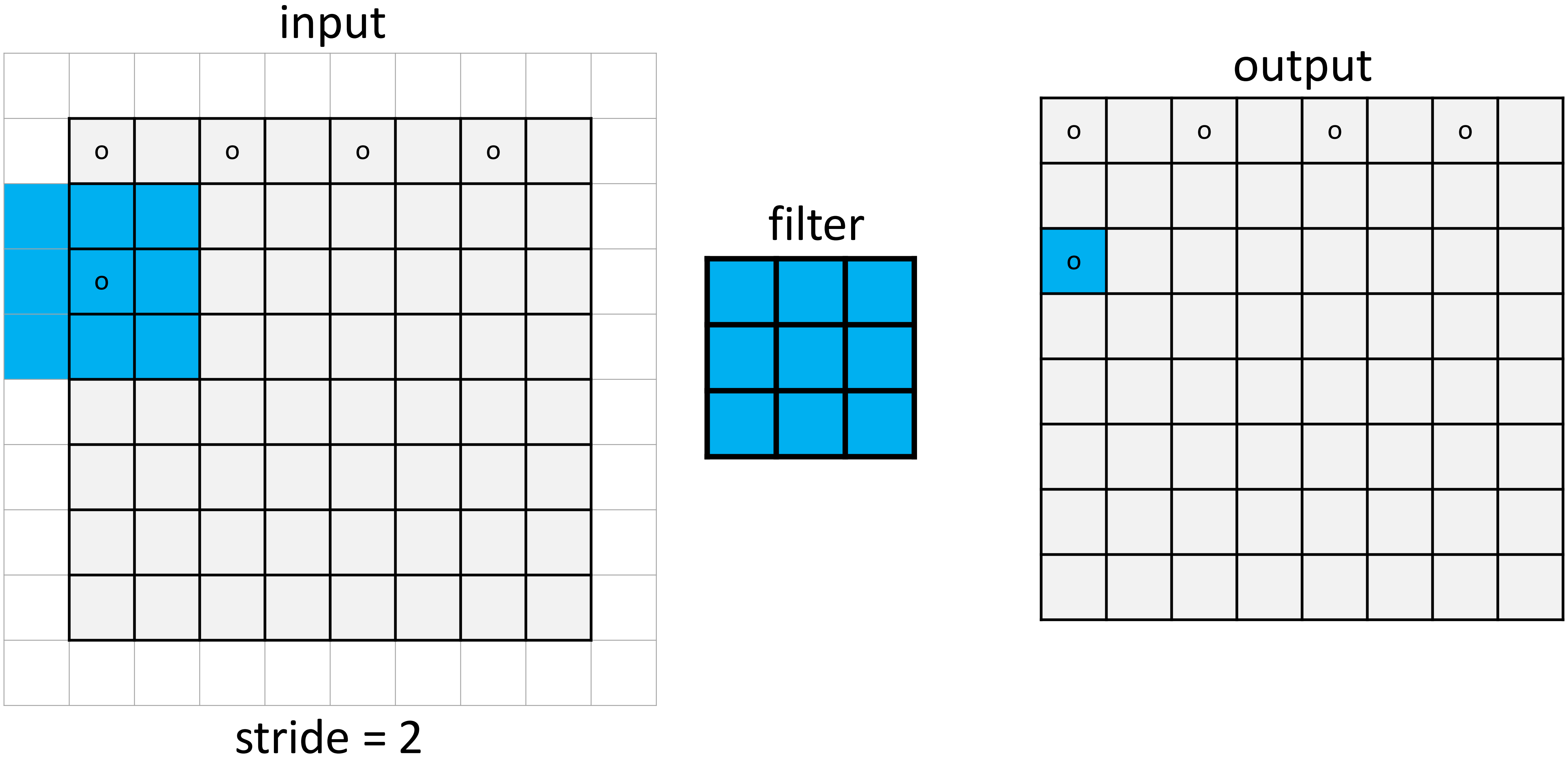


stride = 2

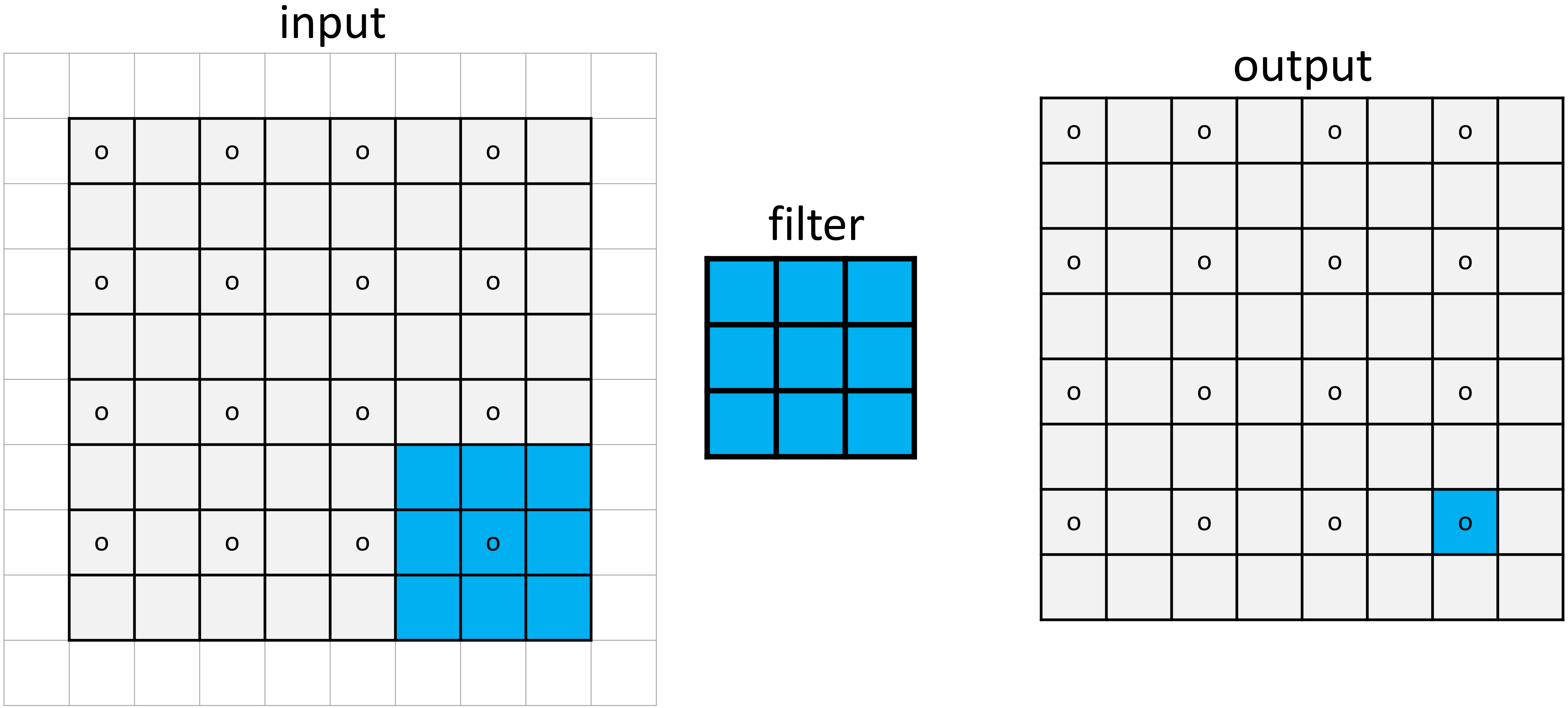
Convolution：步长（stride）



Convolution：步长（stride)



Convolution：步长（stride）

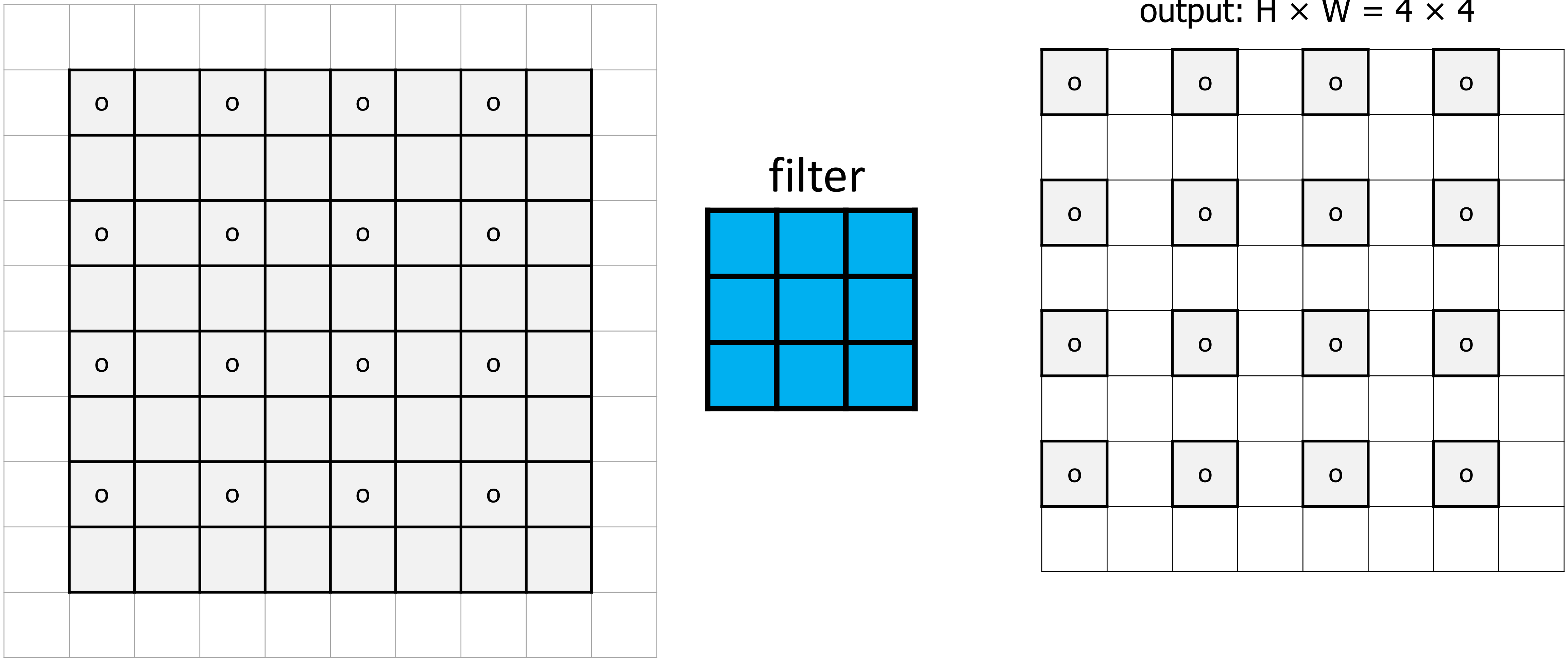


Convolution：步长（stride）

input

filter

output: H × W = 4 × 4

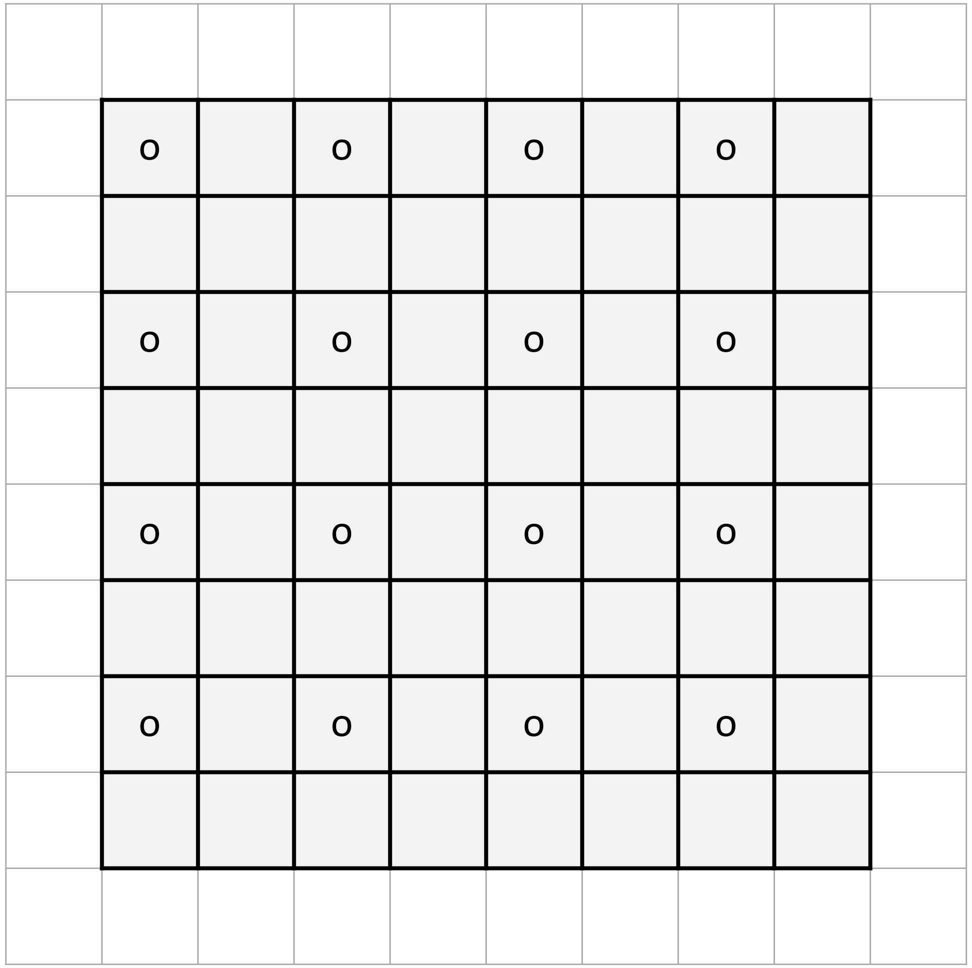


stride = 2

Convolution：步长（stride）

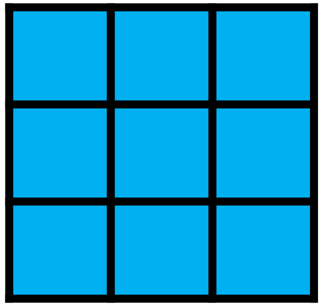
- reduces feature map size
- compress and abstract

input

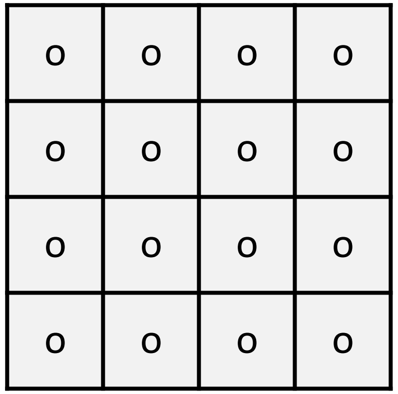


stride = 2

filter



output: H × W = 4 × 4



$$H_{out} = \lfloor (H_{in} + 2pad_h - K_h) / str \rfloor + 1$$

*rounding operation depends on libraries

CNN – Max Pooling

Filter 1

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

Filter 2

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

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

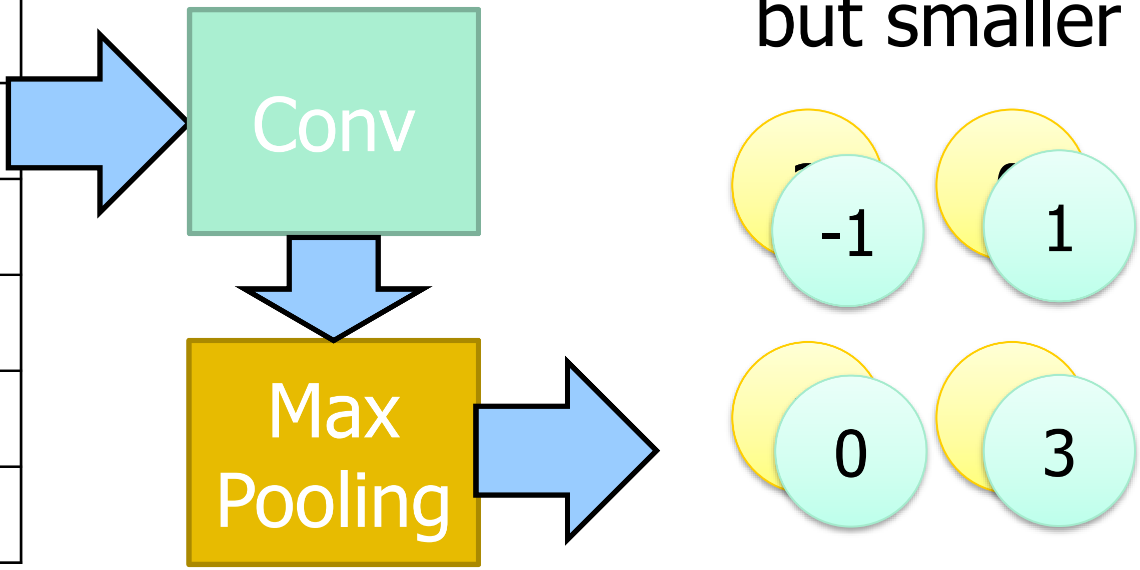
-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

CNN – Max Pooling

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

6 x 6 image

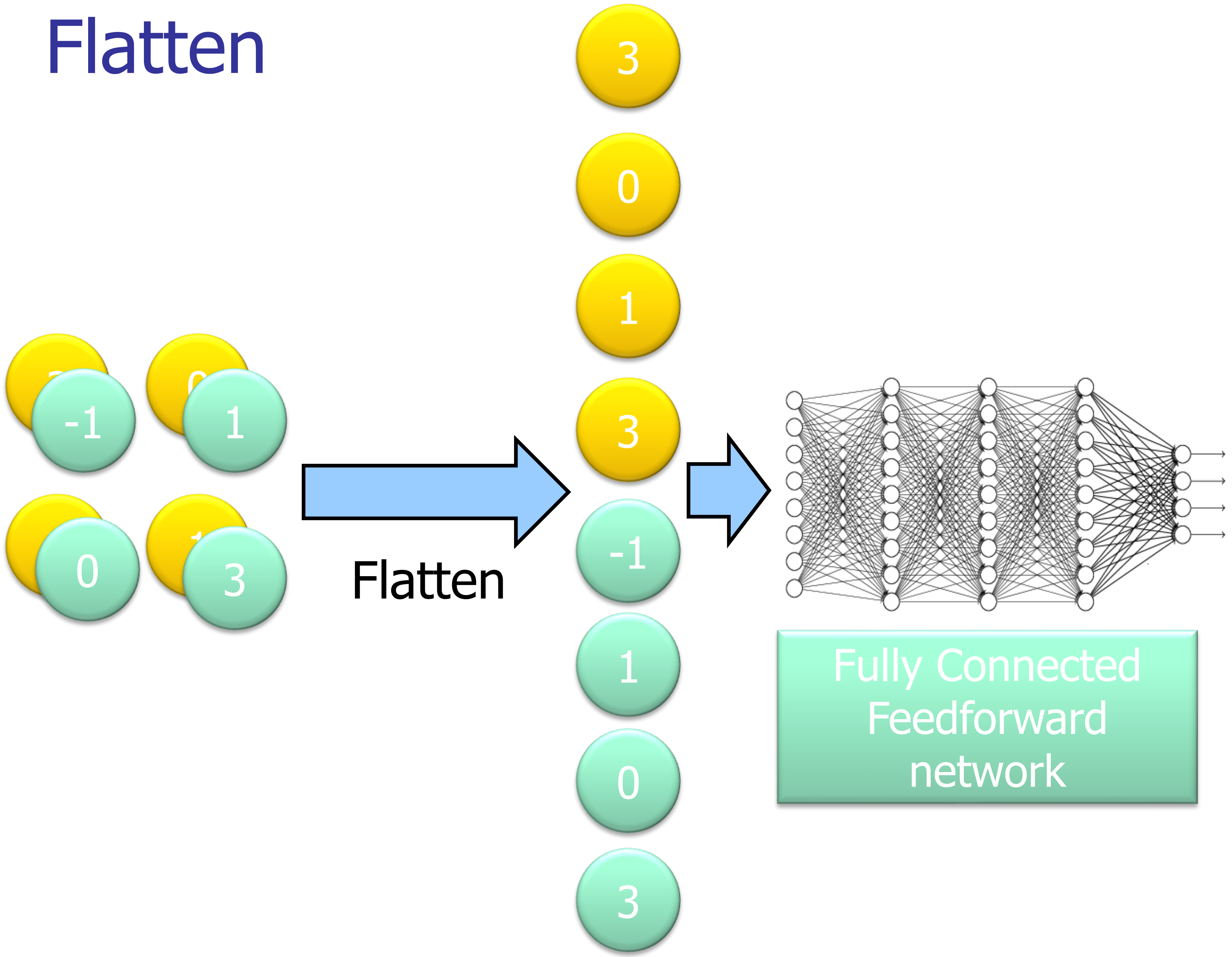
New image
but smaller



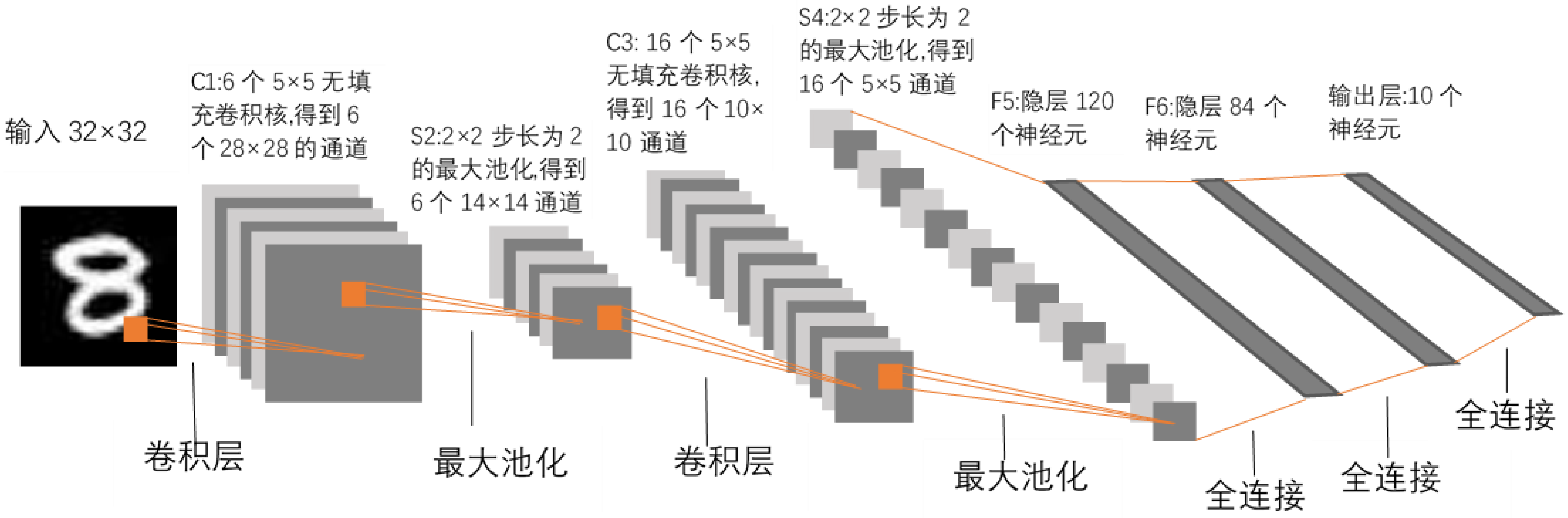
2 x 2 image

Each filter
is a channel

Flatten

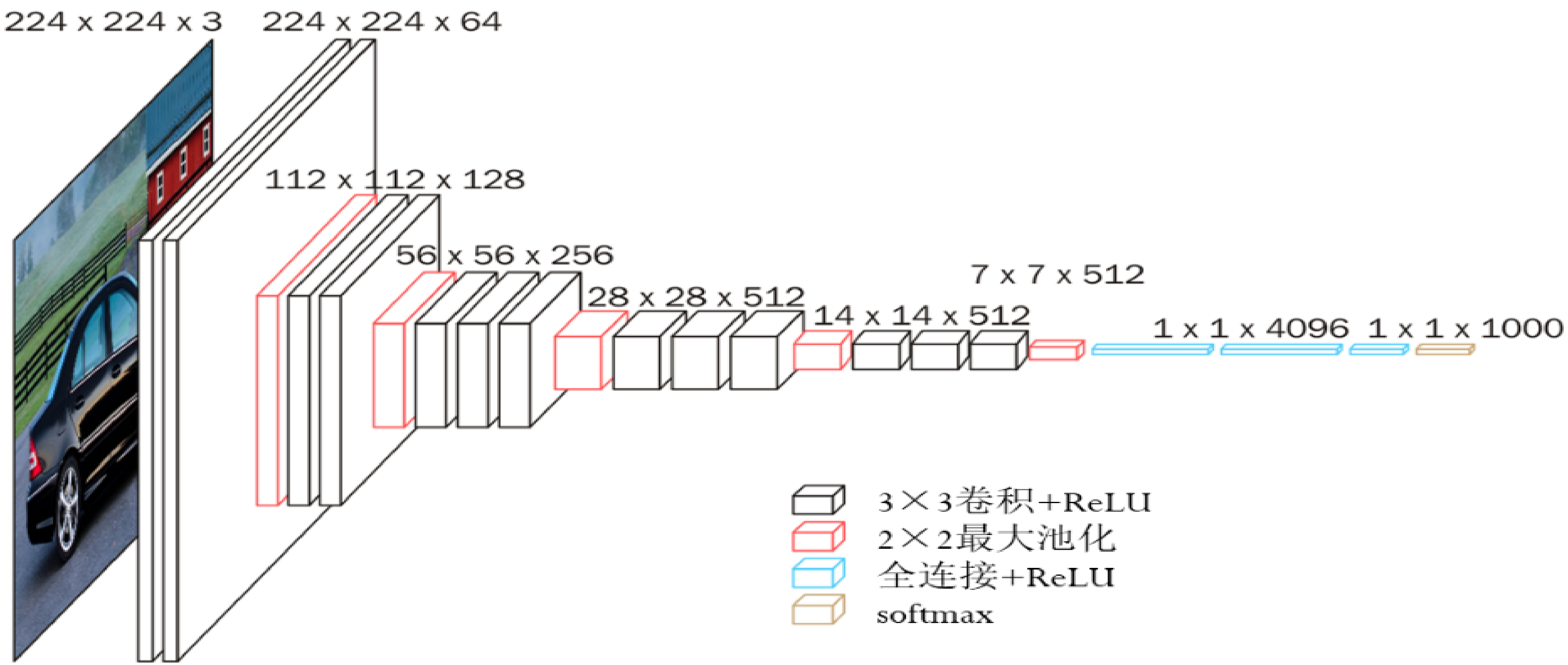

-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected Feedforward network

LeNet



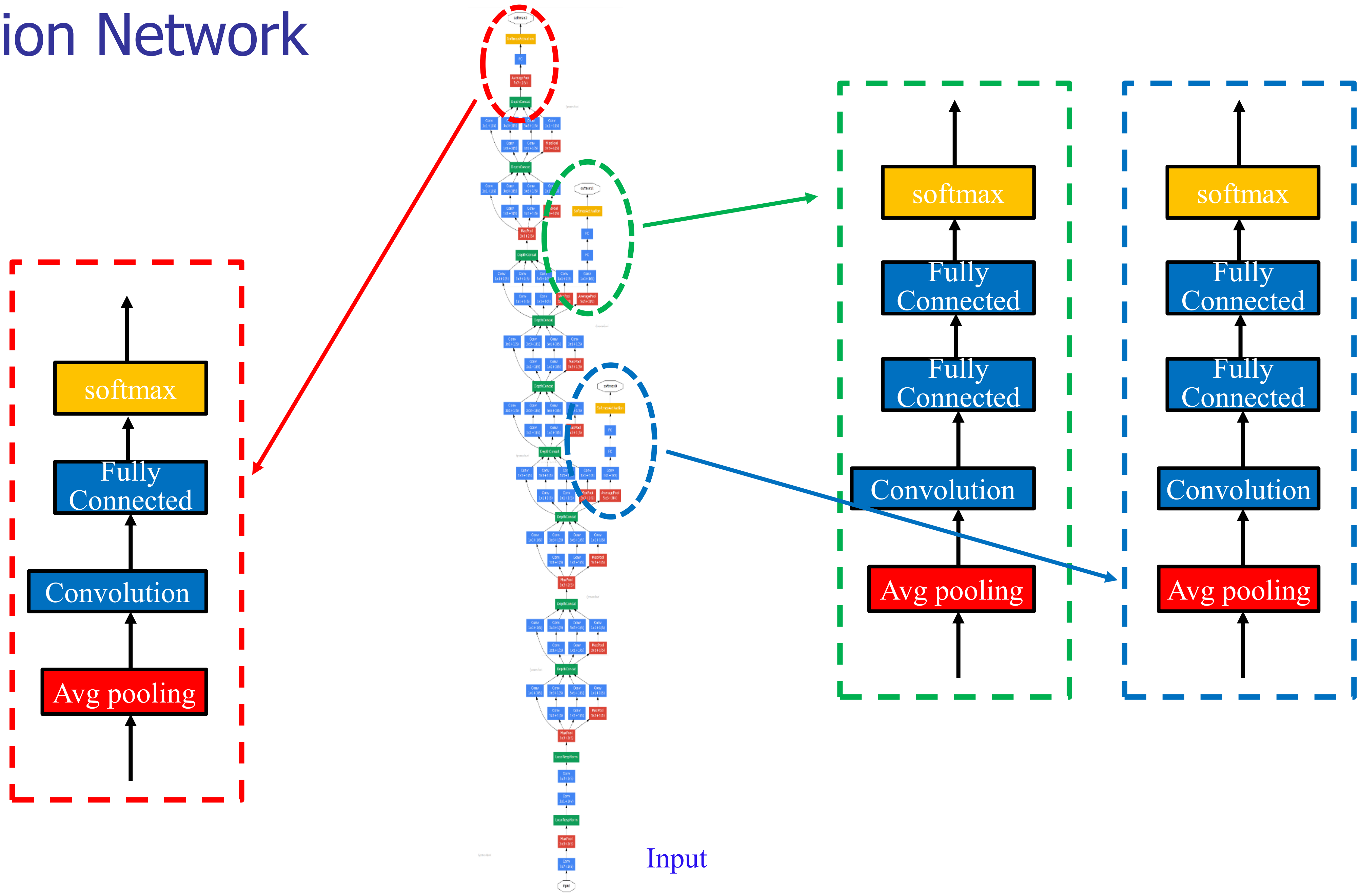
This slide and the following seven slides are from: https://mp.weixin.qq.com/s/sdrIgGOo6DLTGZ3uth6I0

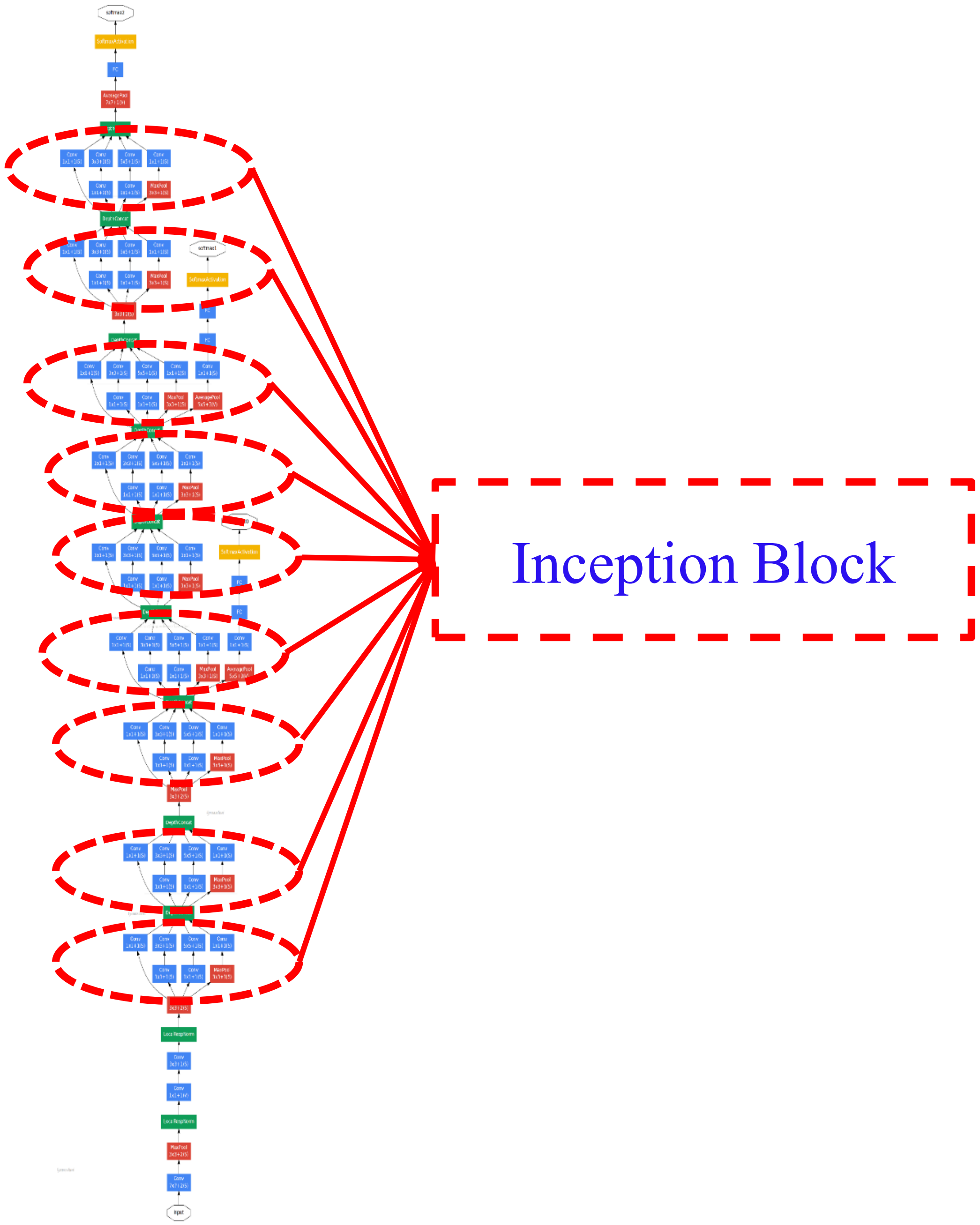
VGG-16



224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
3×3卷积+ReLU
2×2最大池化
全连接+ReLU
softmax

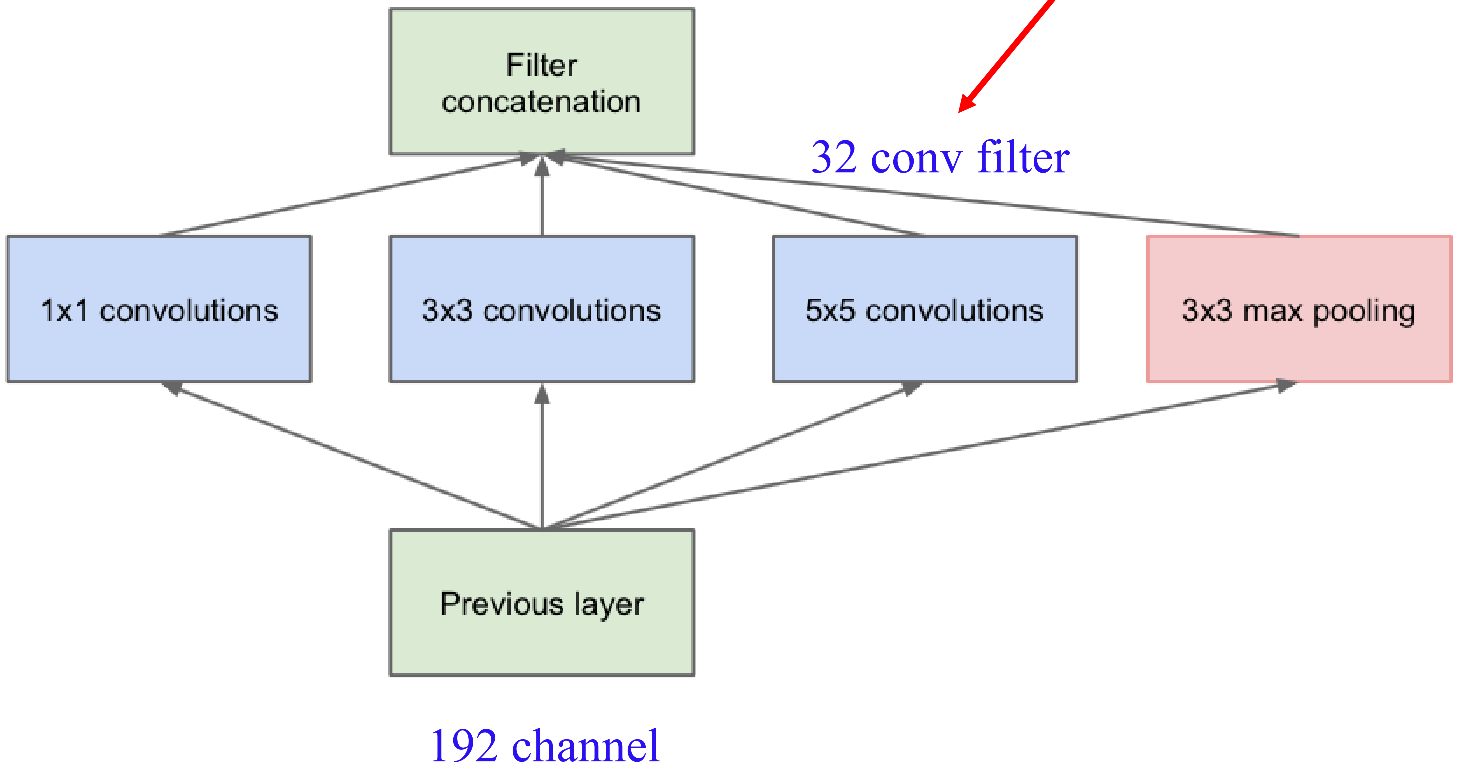
Inception Network




Inception Block

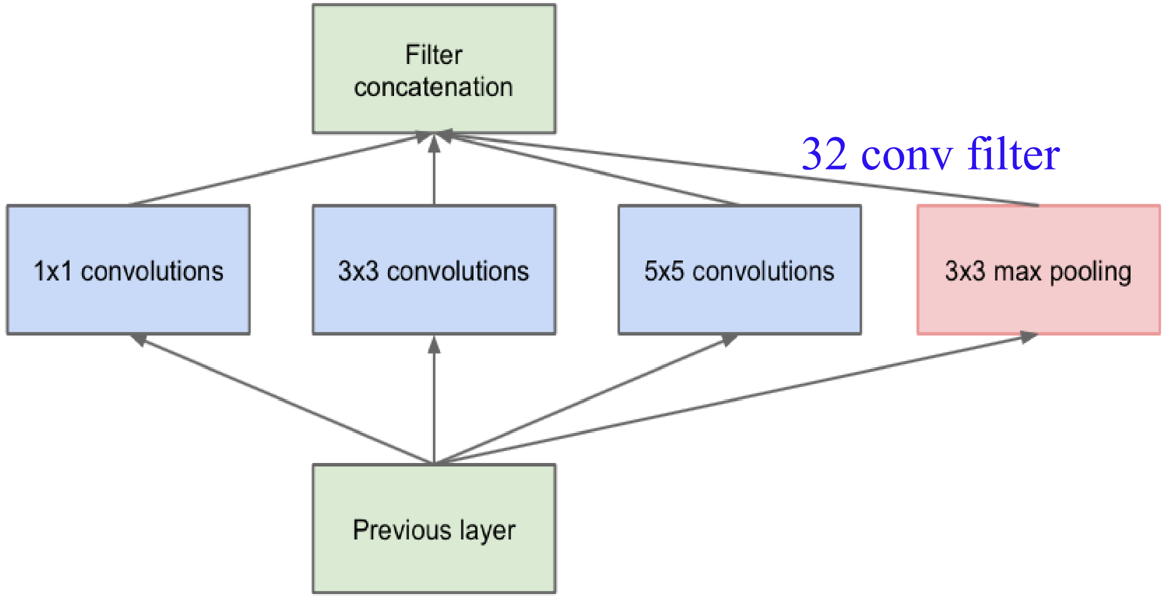
Inception Block

Parameter amount：
$(5\times5\times192+1)\times32=153{,}632$



Parameters：

$(5\times5\times192+1)\times32=153{,}632$



32 conv filter

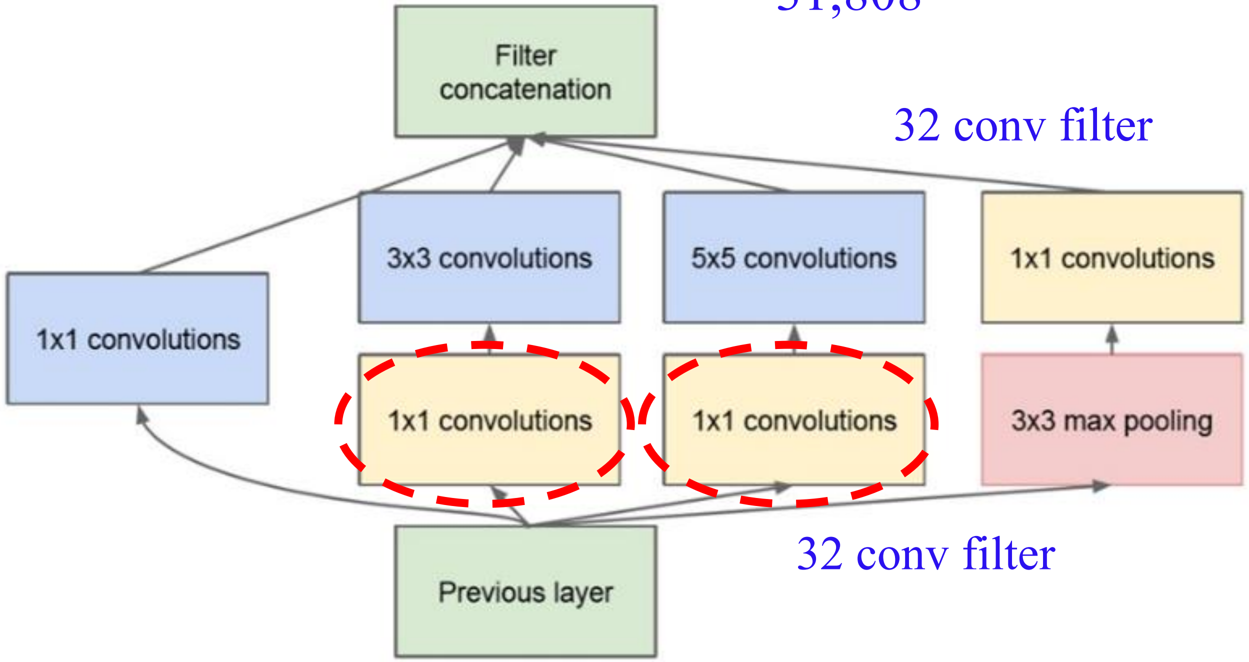
192 channel

Parameters：

$(1\times1\times192+1)\times32+$

$(5\times5\times32+1)\times32$

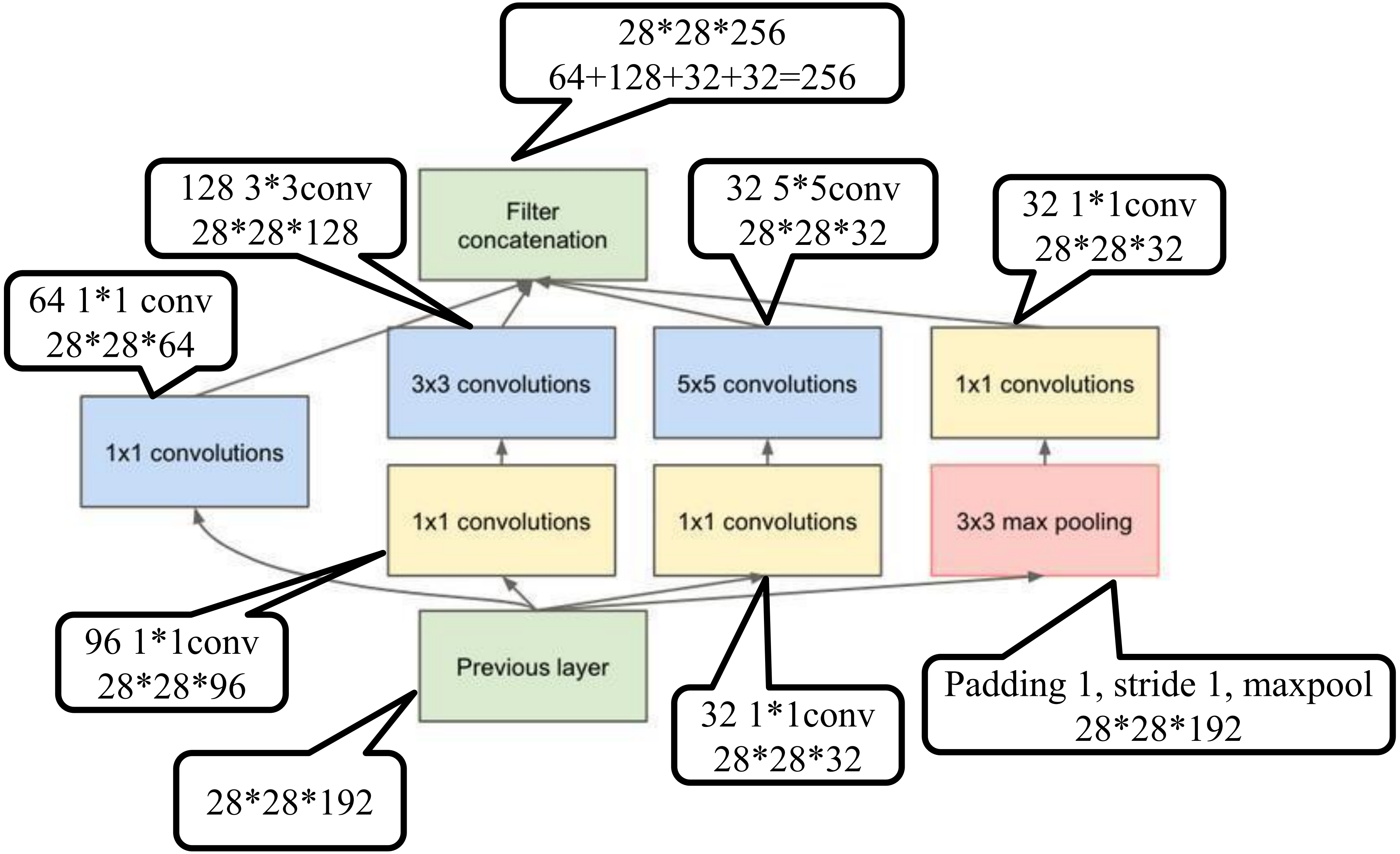
$=31{,}808$



32 conv filter

32 conv filter

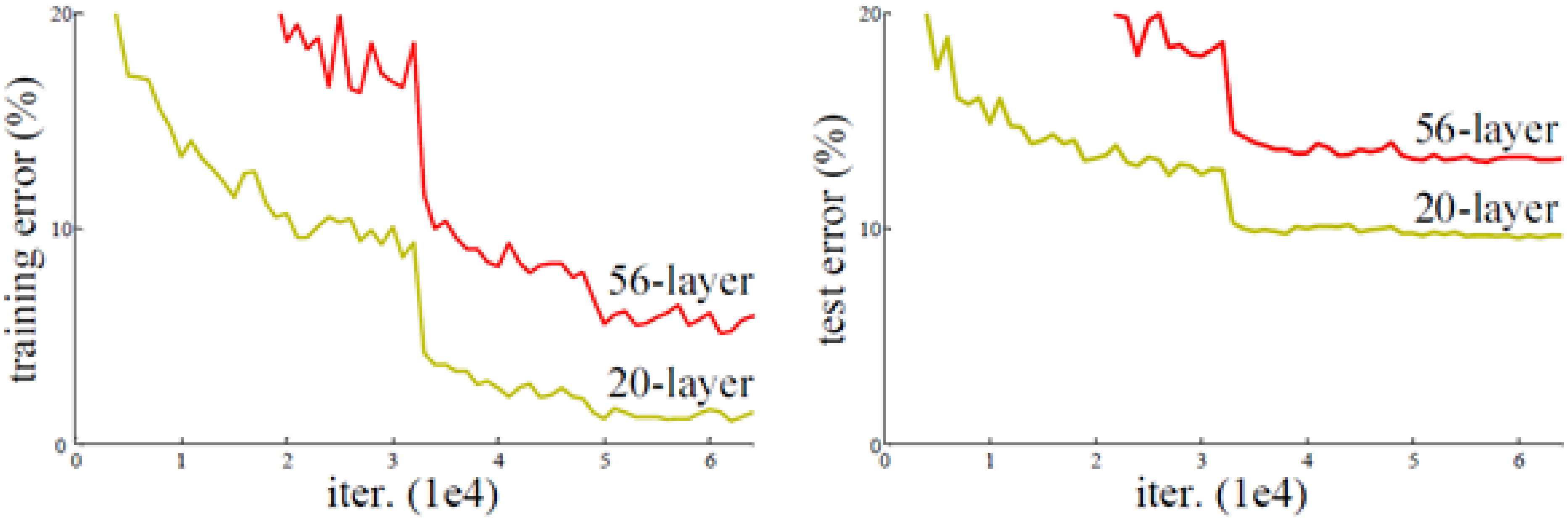
192 channel


28*28*256
64+128+32+32=256
Filter concatenation
128 3*3conv
28*28*128
32 5*5conv
28*28*32
32 1*1conv
28*28*32
64 1*1 conv
28*28*64
3x3 convolutions
5x5 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
3x3 max pooling
96 1*1conv
28*28*96
Previous layer
32 1*1conv
28*28*32
Padding 1, stride 1, maxpool
28*28*192
28*28*192

7×7 average pooling
Stride 1, no padding

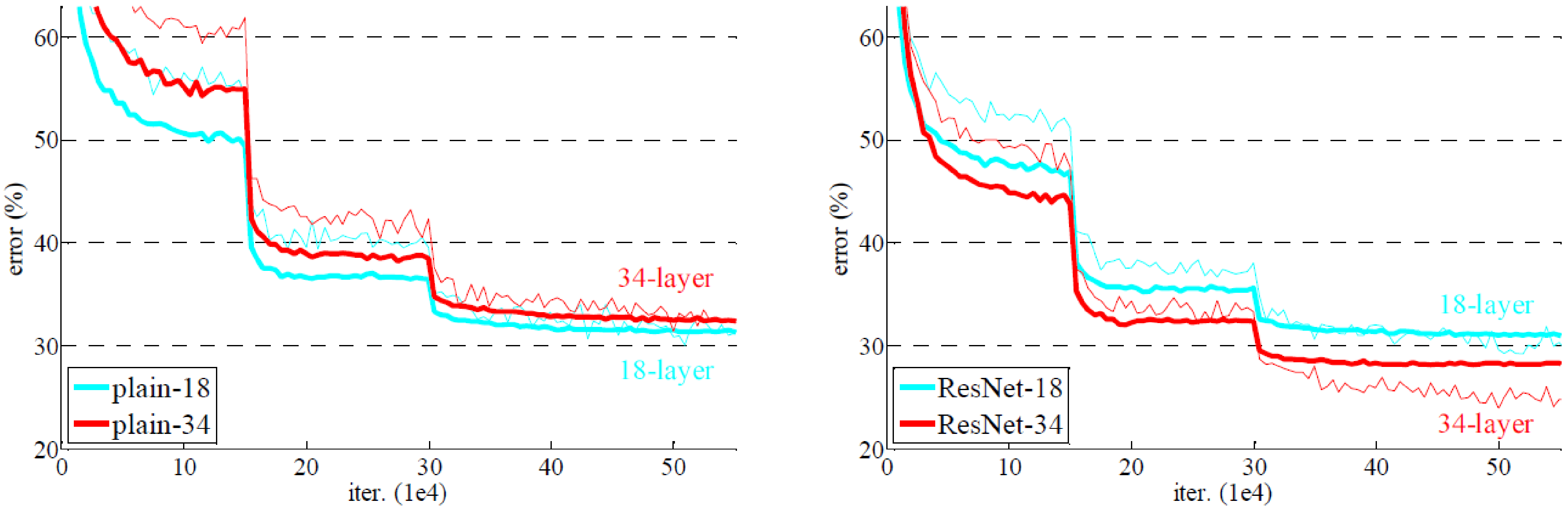
5×5 average pooling
Stride 3， no padding

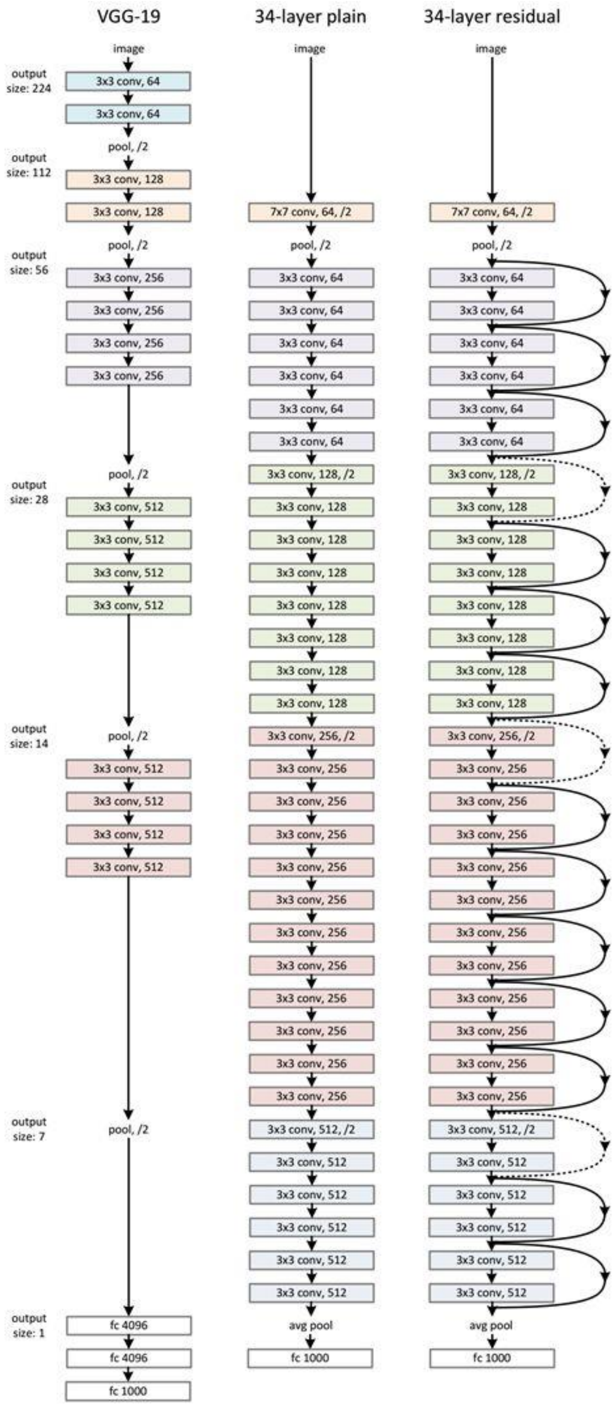
Degradation Problem in Deep Networks



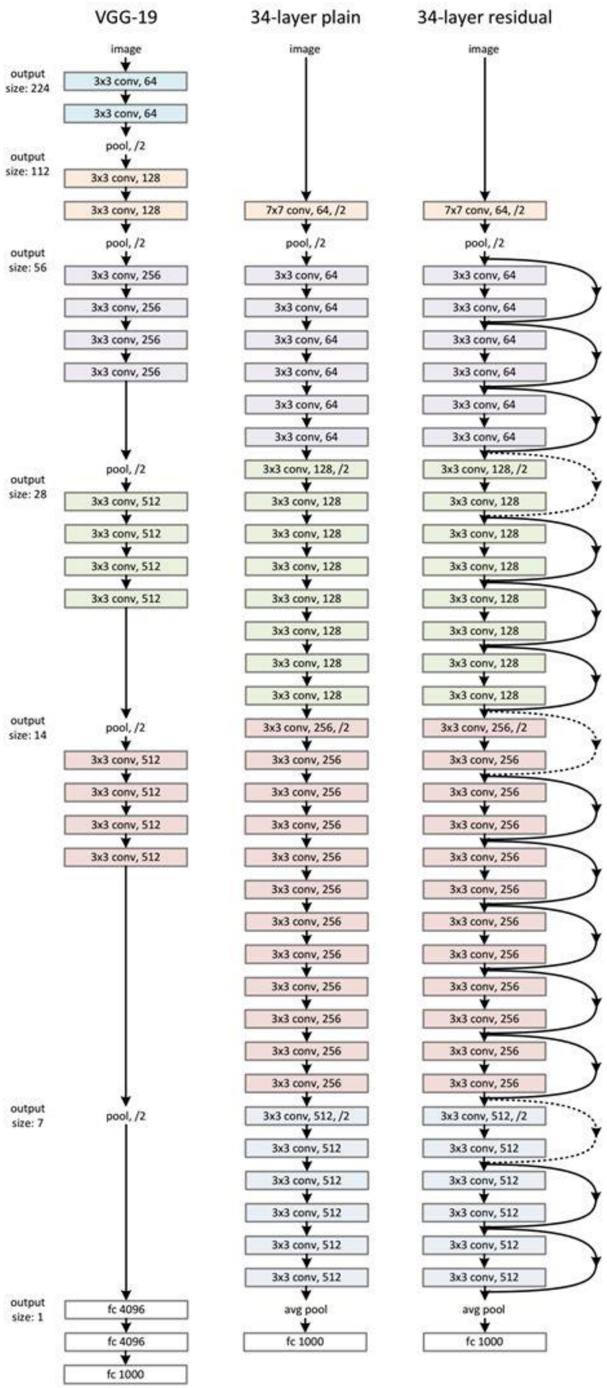
Deep Residual Learning

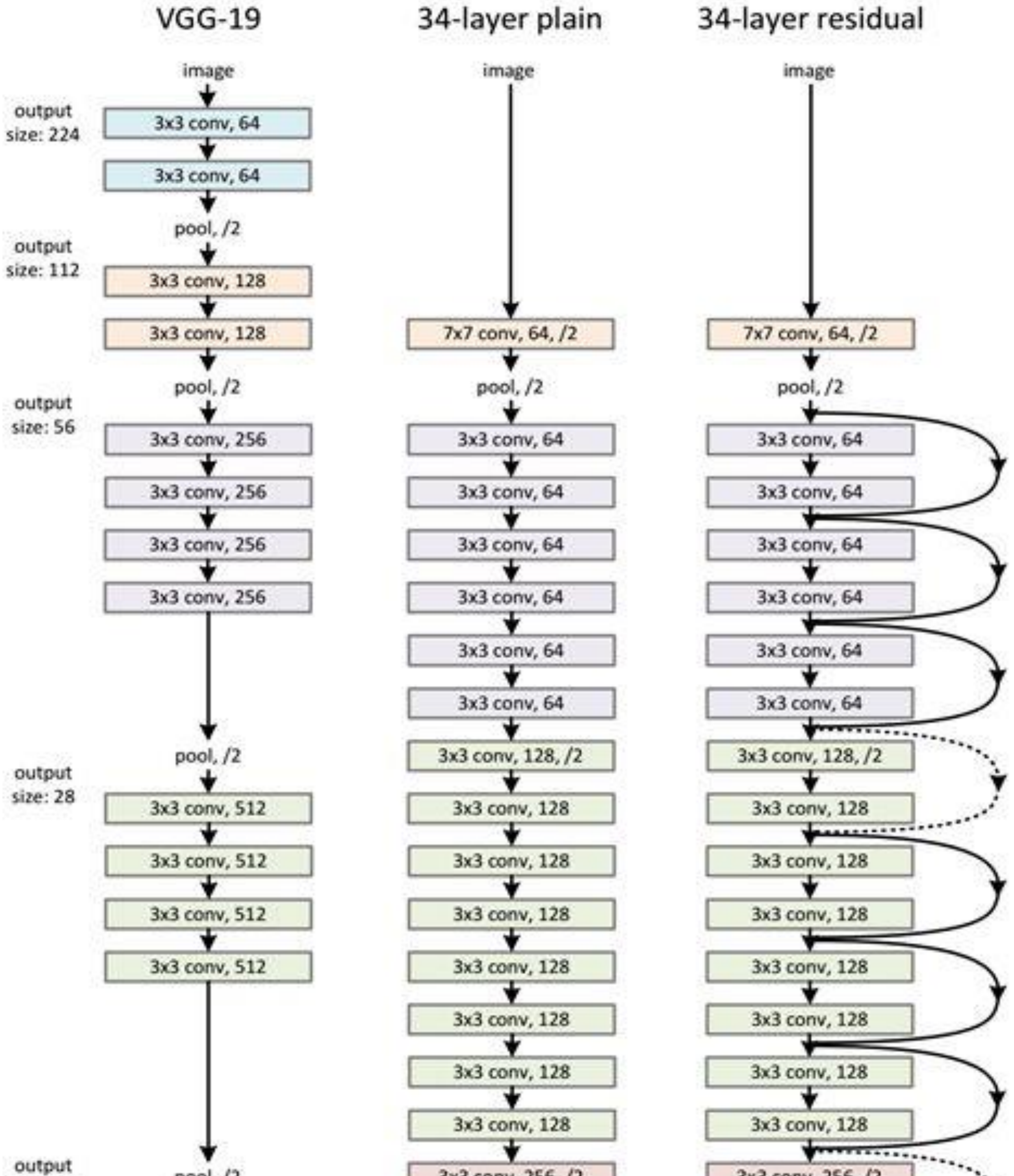
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR 2016.

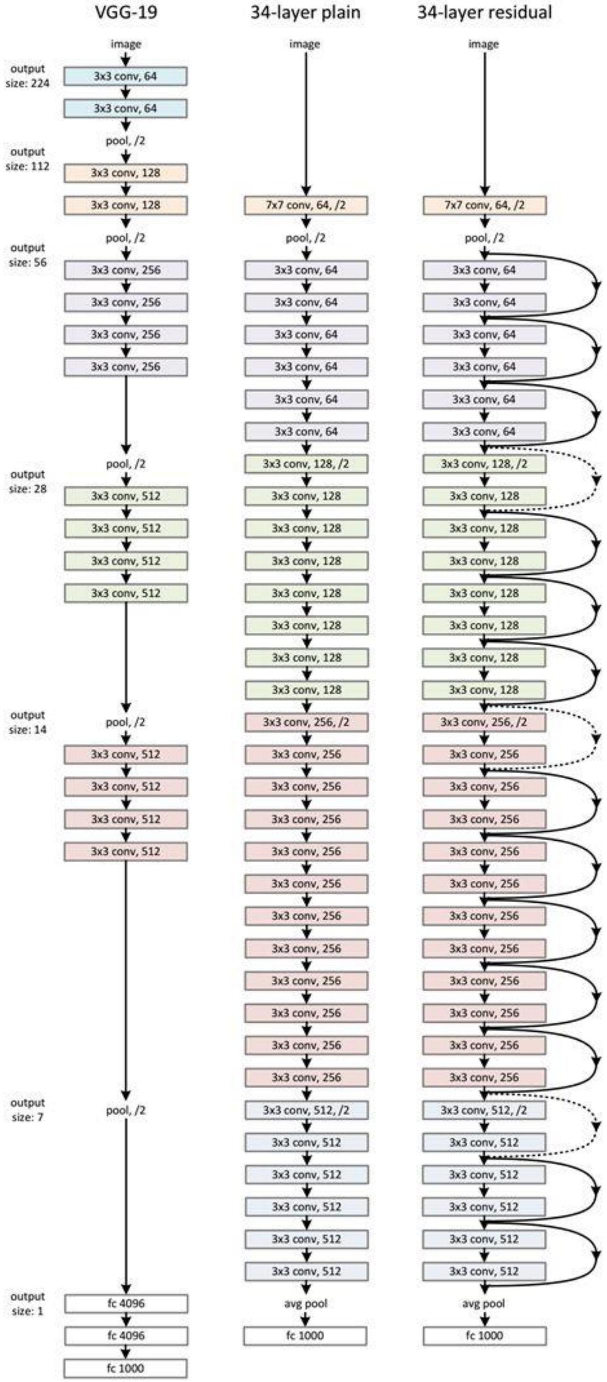


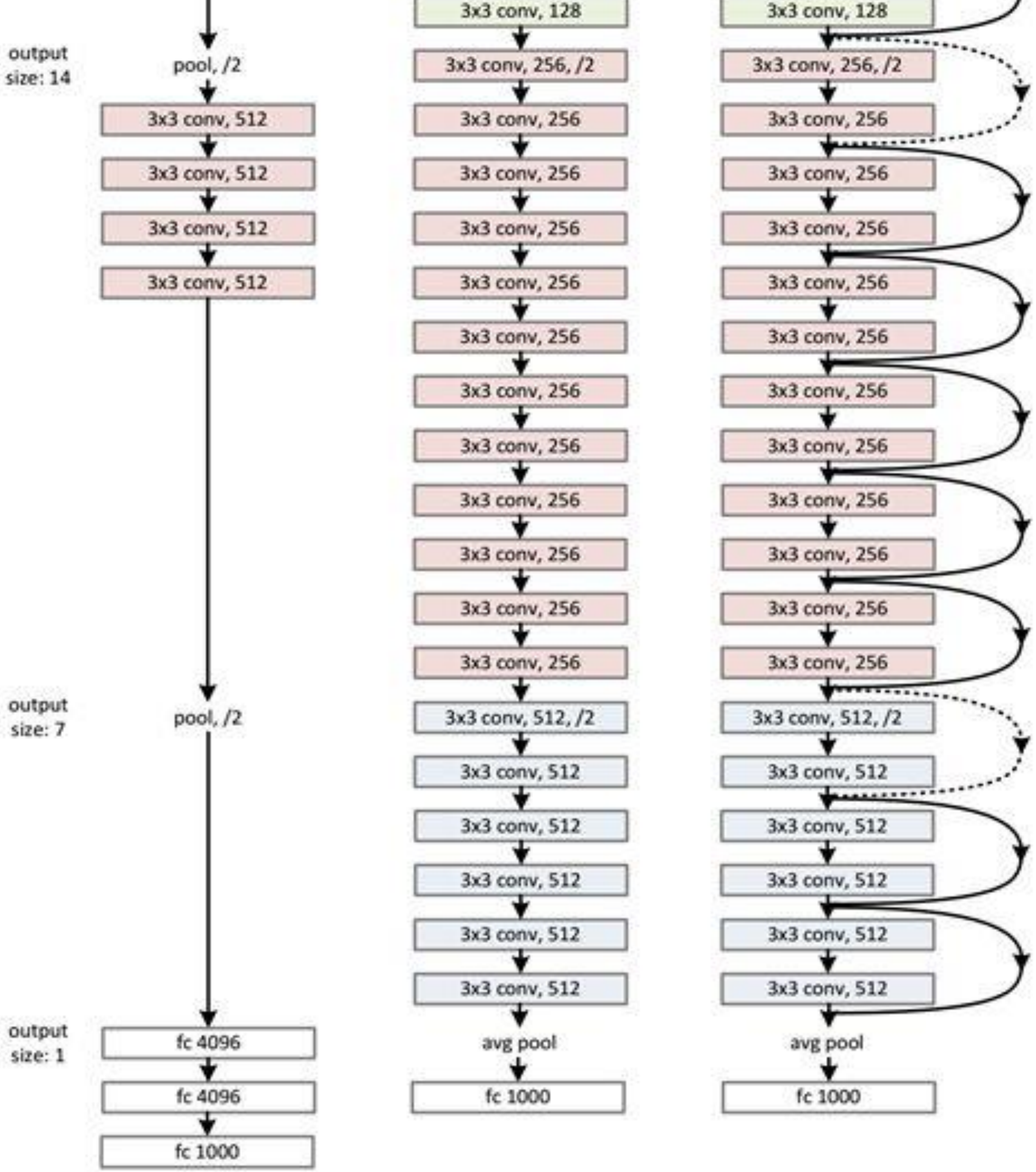


VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
7x7 conv, 64, /2
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 64
3x3 conv, 128, /2
output size: 28
3x3 conv, 512
3x3 conv, 128
3x3 conv, 256, /2
output size: 14
3x3 conv, 256
3x3 conv, 512, /2
output size: 7
output size: 1
fc 4096
fc 4096
fc 1000
avg pool

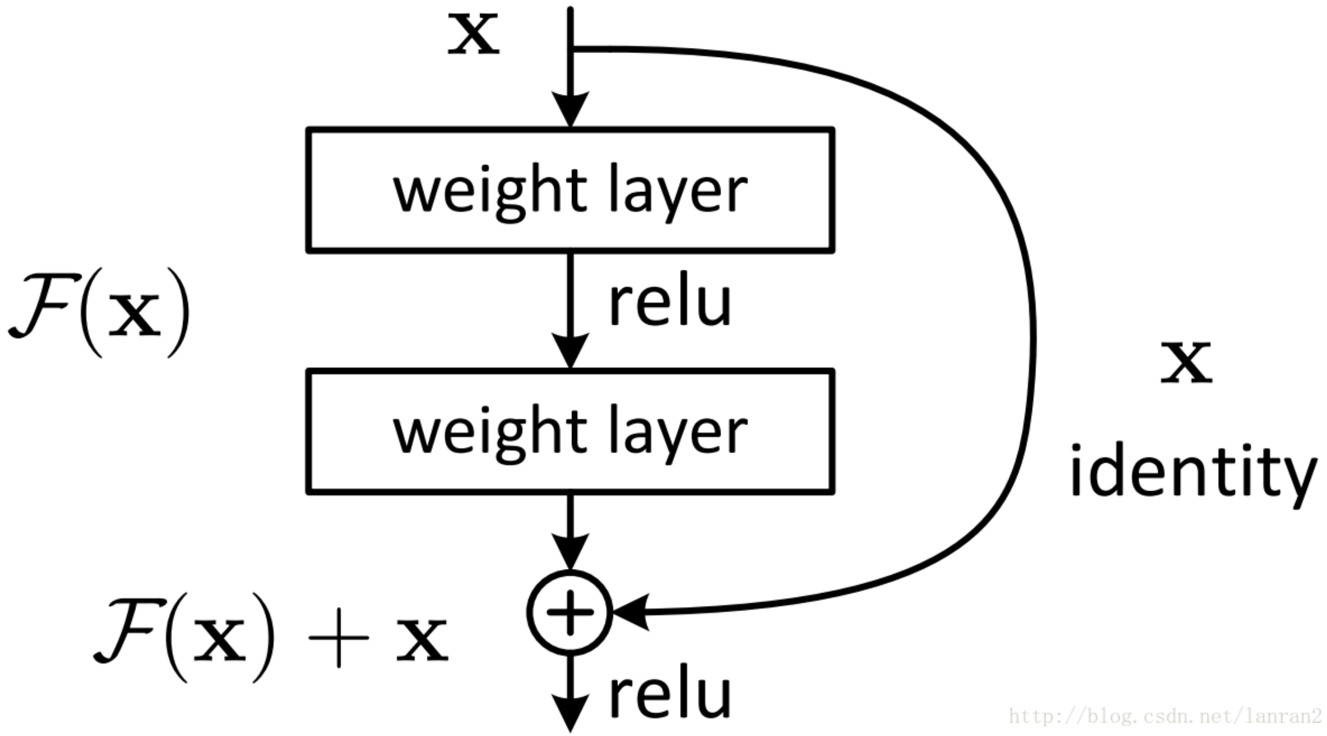

VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 14
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 7
output size: 1
fc 4096
fc 4096
fc 1000
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000

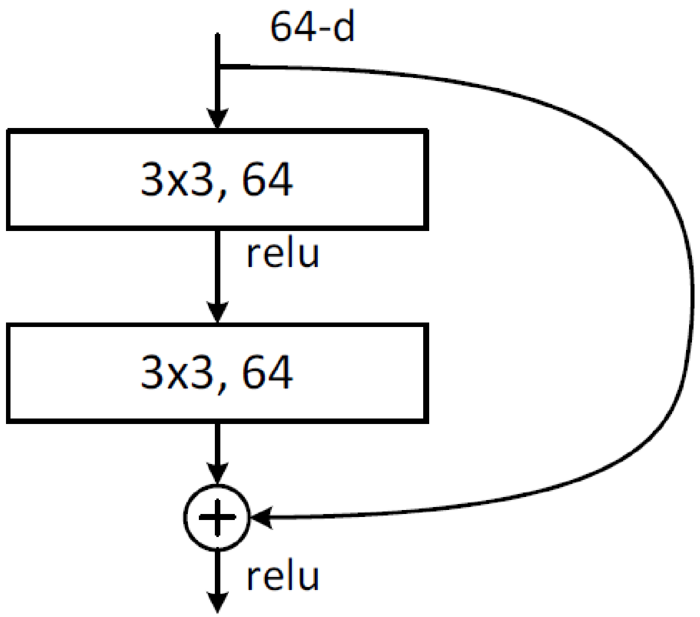

VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
7x7 conv, 64, /2
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 64
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 128, /2
3x3 conv, 128
output

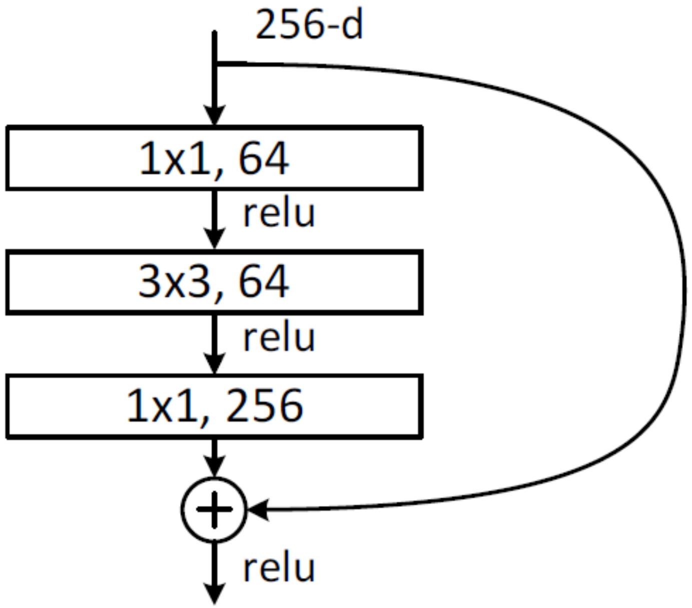

VGG-19
34-layer plain
34-layer residual
image
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1
3x3 conv, 64
pool, /2
3x3 conv, 128
3x3 conv, 256
3x3 conv, 512
7x7 conv, 64, /2
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
fc 4096
fc 1000
avg pool


output size: 14
pool, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512
output size: 7
3x3 conv, 512, /2
output size: 1
fc 4096
fc 1000
avg pool

Identity Mapping by Shortcuts







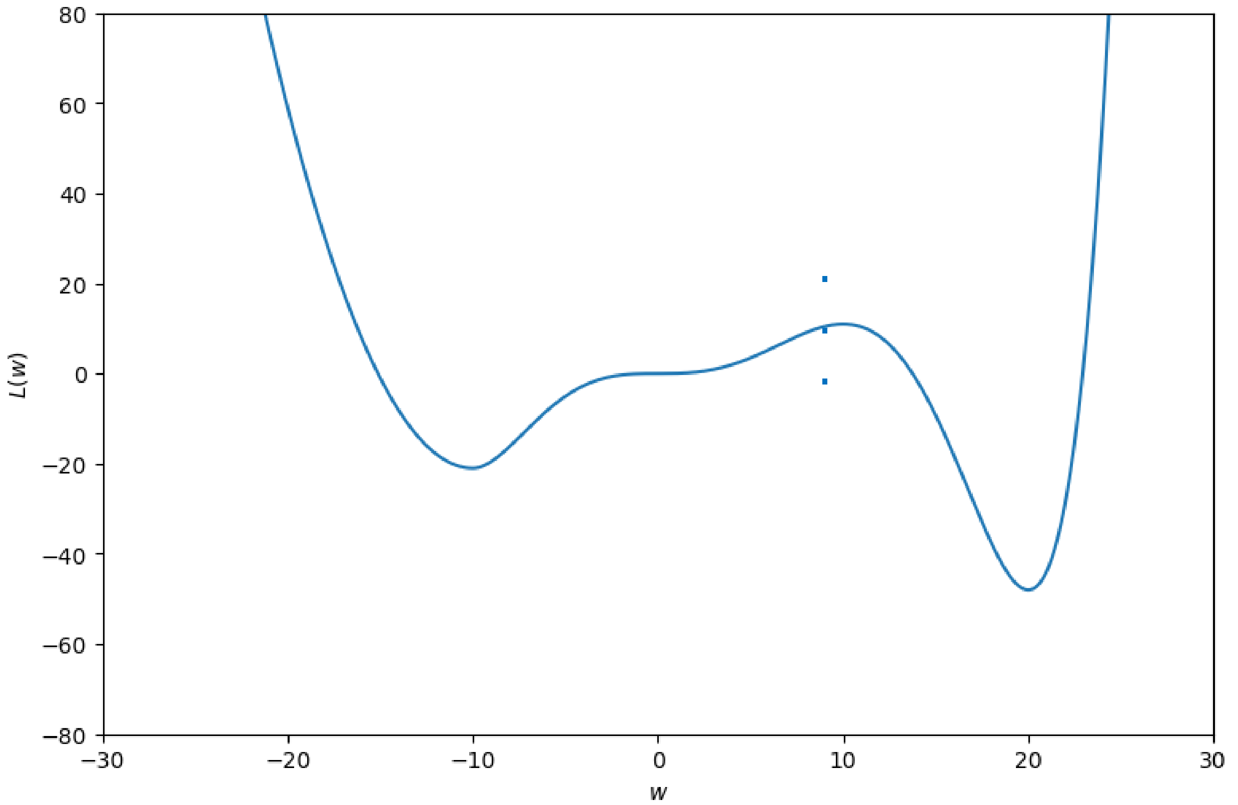
Some Techniques
for Learning Better Deep Networks

Techniques for Learning Better NNs

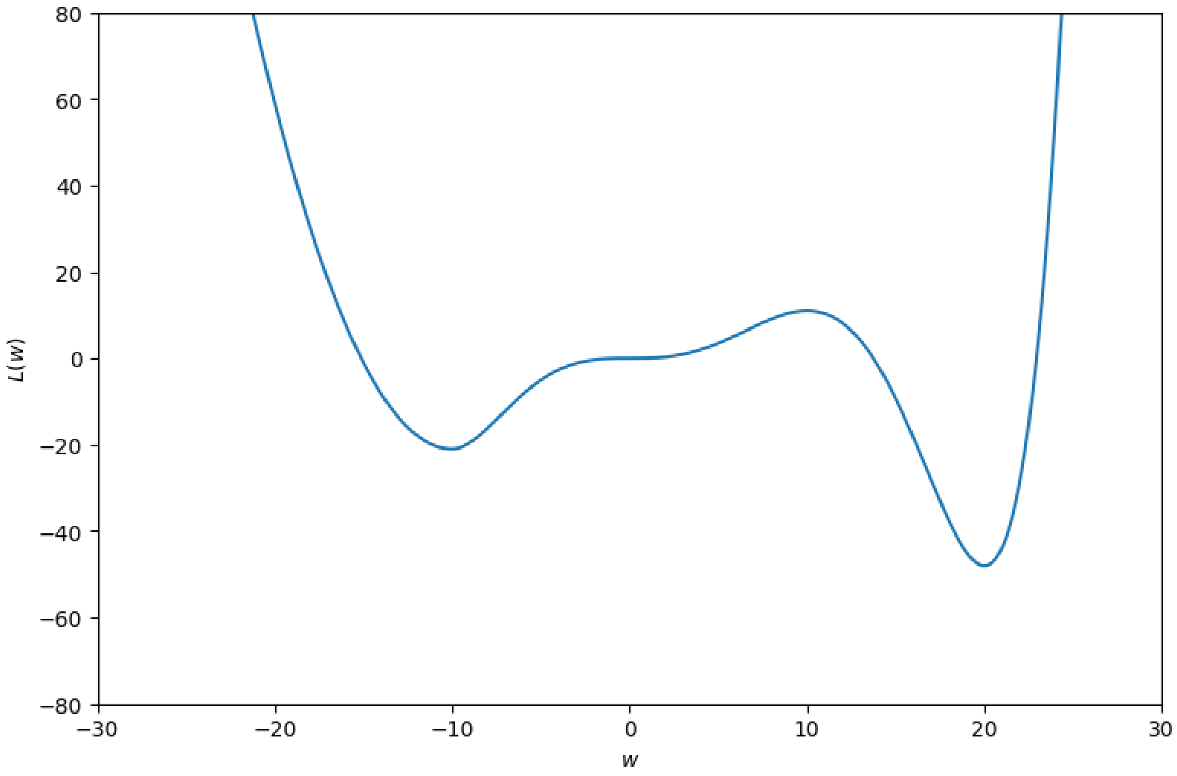
- Design good network structures
- Design good loss?
- Optimization techniques
- Others
  - Data preprocessing
  - Data augmentation

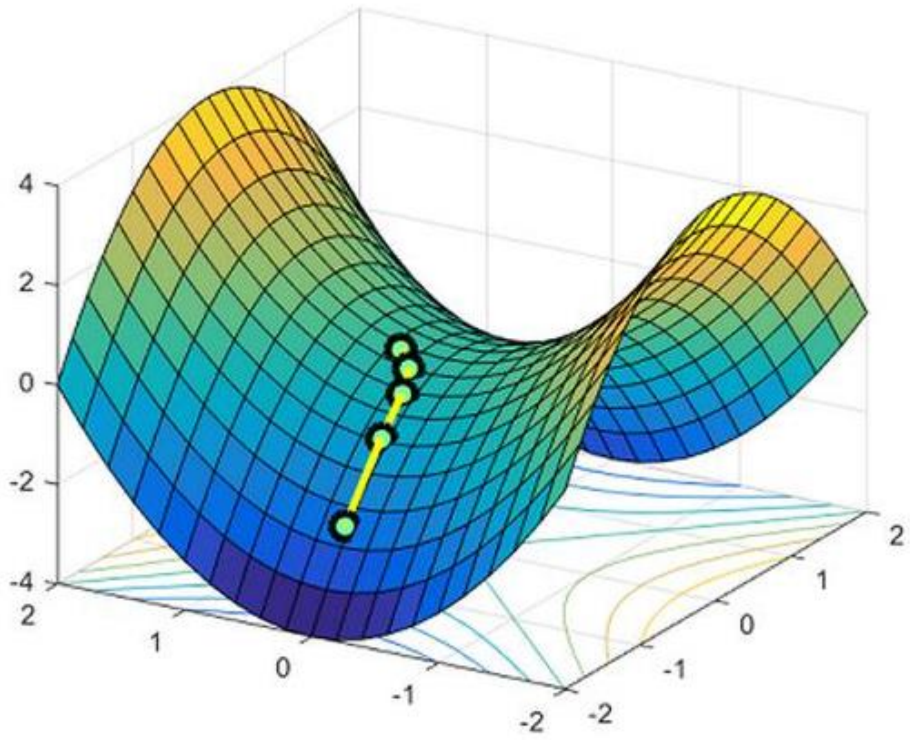
Remember Two Facts

- A NN is a complicated function
  - We can study it from function properties
- Optimizing a NN is usually a non-convex problem



Stationary Point and Saddle Points in High-dimensional Space





Flat Minima in NN

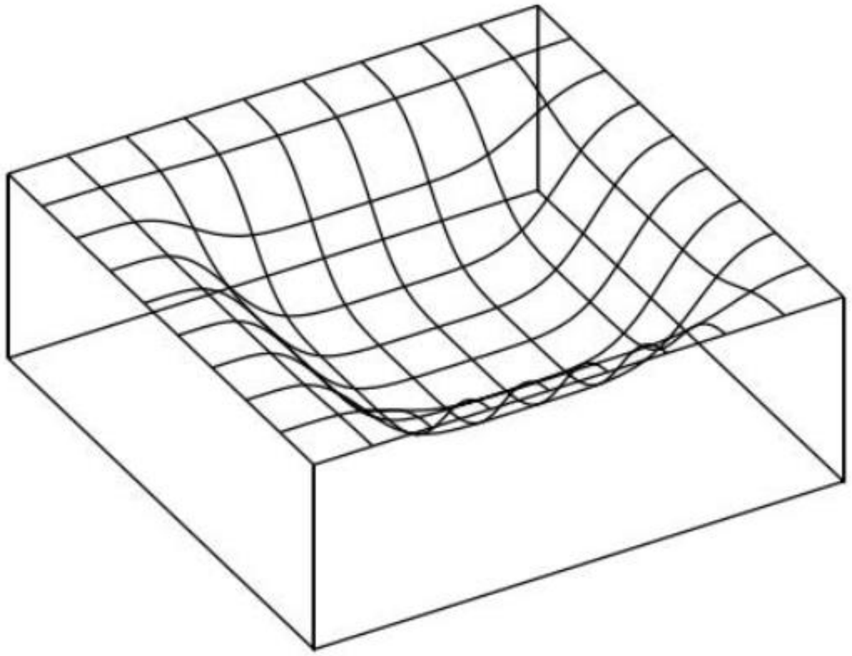


Figure 1: *Example of a "flat" minimum.*

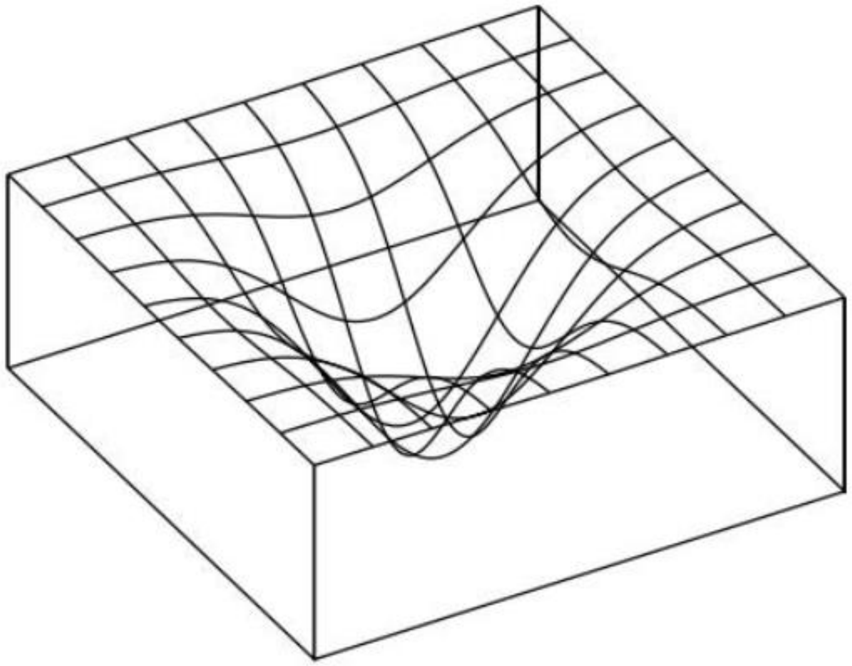
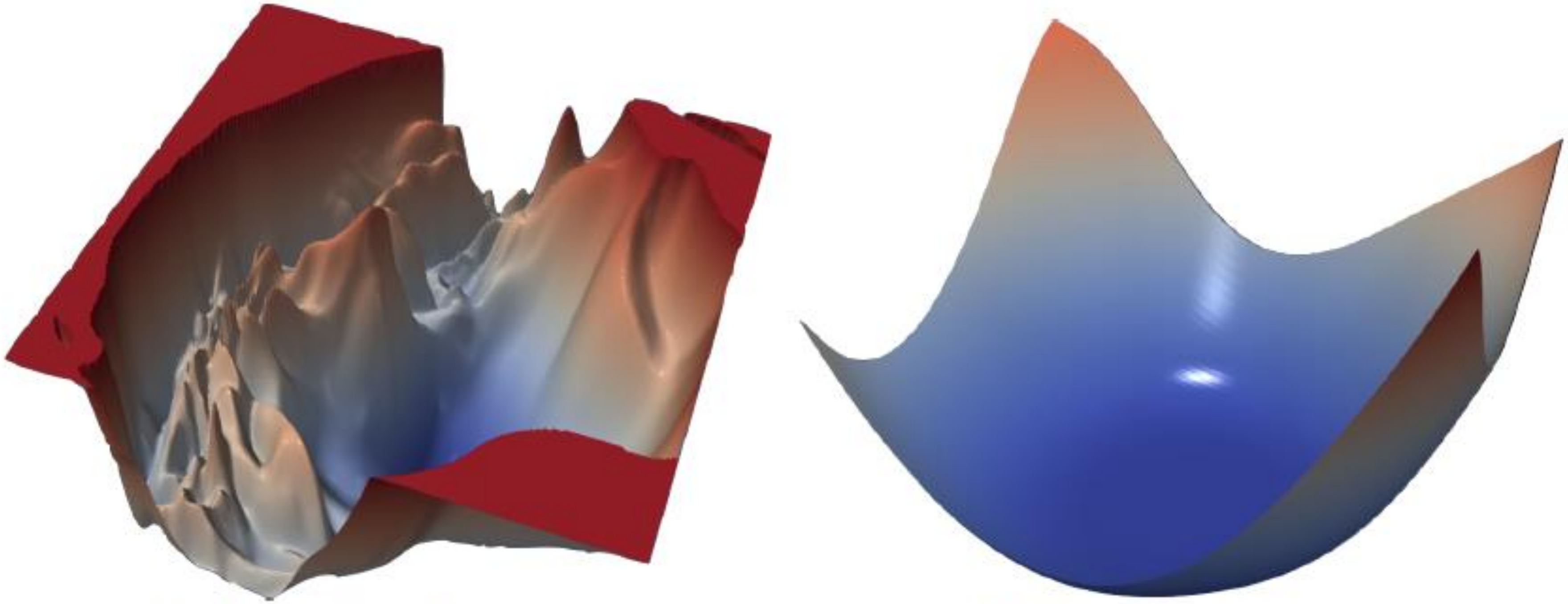


Figure 2: *Example of a "sharp" minimum.*

https://www.pianshen.com/article/47781237198/

Loss Landscape of NN



(a) ResNet-110, no skip connections (b) DenseNet, 121 layers

Figure 4: The loss surfaces of ResNet-110-noshort and DenseNet for CIFAR-10.

Visualizing the Loss Landscape of Neural Nets, NIPS 2018

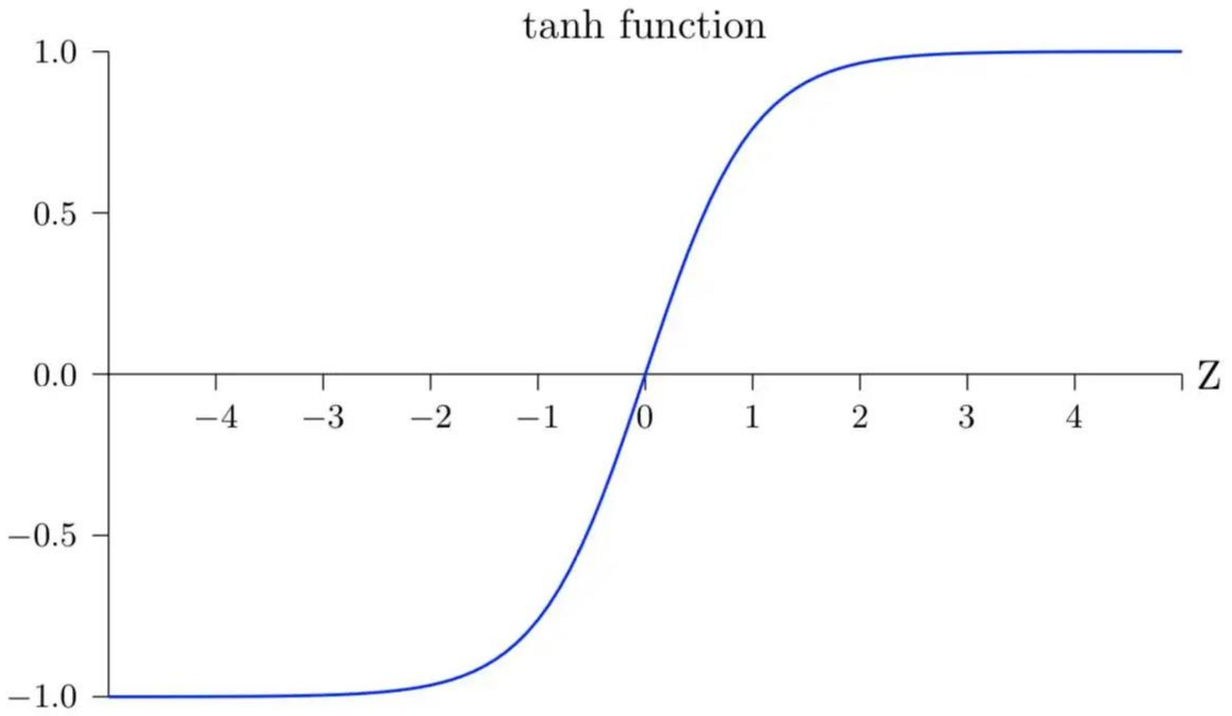
Network Design Issues

- How many layers?
- How many neurons in a layer?
- Connection structure:
  - other than fully-connected
- Activation Functions

Activation Function: Hyperbolic Tangent

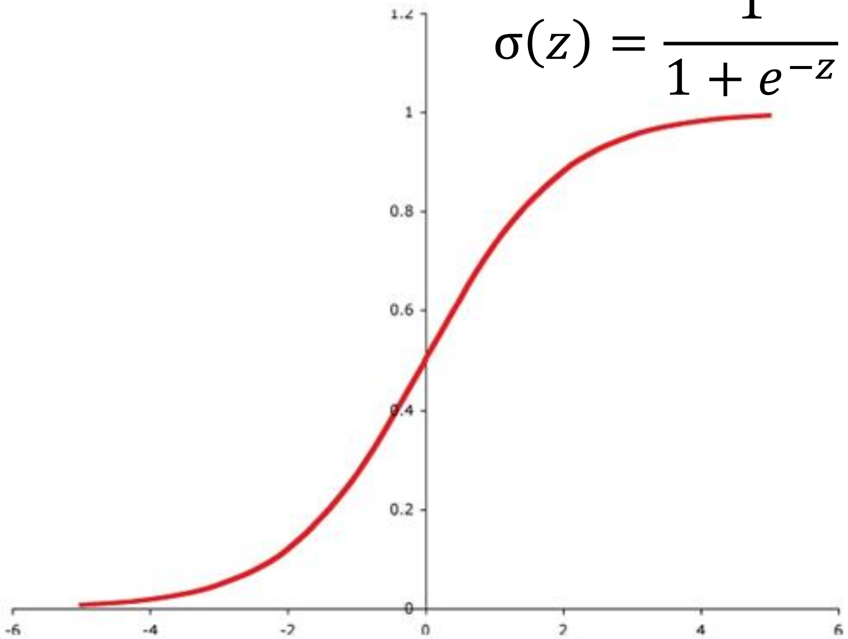
$$a(\mathrm{z}) = \tanh(z)$$

$$= \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

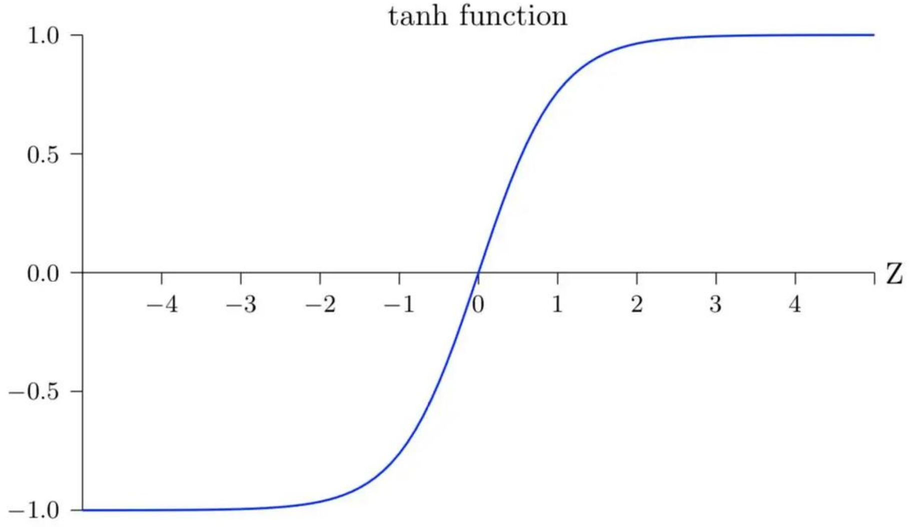


Gradient Vanishing Problem

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



$$\frac{\partial L}{\partial w_{ij}^{r}} = \frac{\partial L}{\partial \boldsymbol{a}^{L}} \cdots \cdots \frac{\partial \boldsymbol{a}^{r+1}}{\partial a_i^{r}} \frac{\partial a_i^{r}}{\partial z_i^{r}} \frac{\partial z_i^{r}}{\partial w_{ij}^{r}}$$

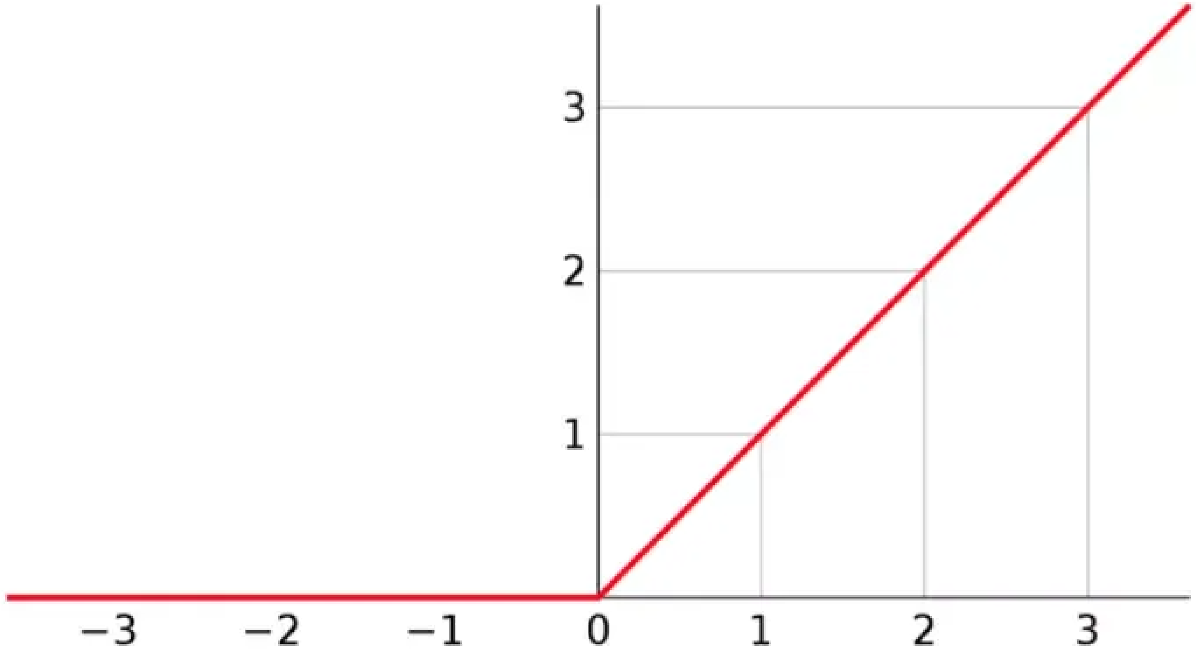


Sigmoid function are now often used as gate unit

Activation Function: ReLU

$$a(z) = \mathrm{ReLU}(z)$$

$$= \max(0, z)$$



Design Good Loss Functions

- Different tasks should have specific loss functions
  - MSE
  - Cross entropy
  - Focal loss
  - ……

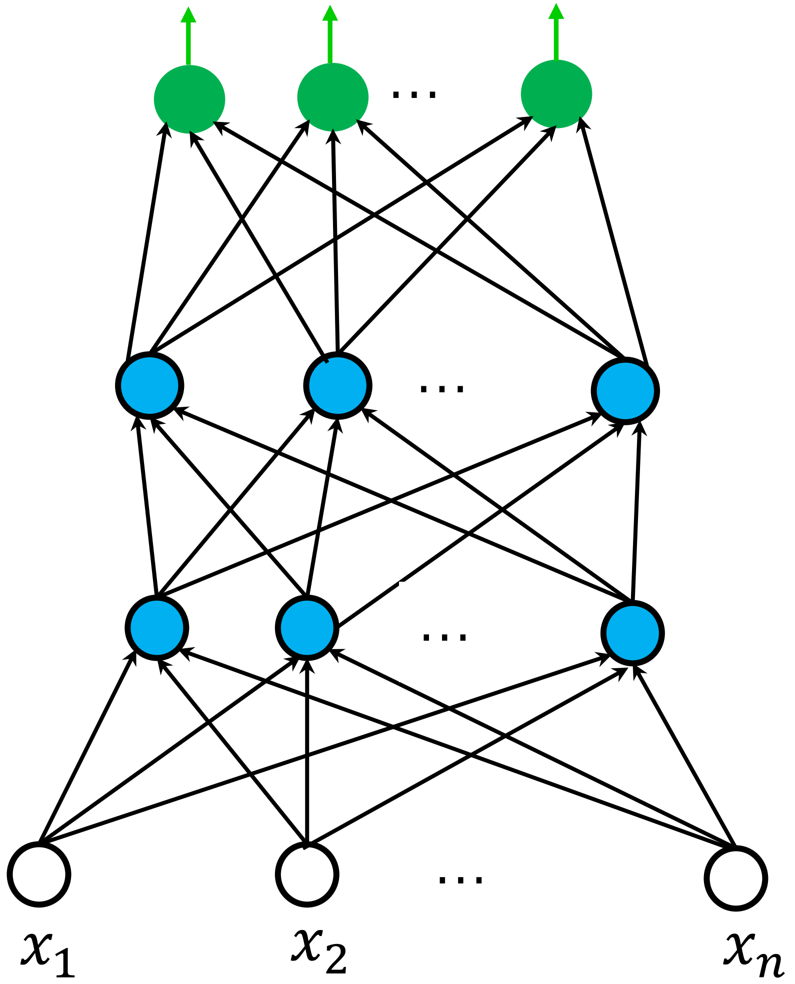
$$\boldsymbol{L(W)} = \frac{1}{2}\sum_{i=1}^{m} \| \boldsymbol{y}^{(i)} - \widehat{\boldsymbol{y}}^{(i)} \|^2$$

$$\boldsymbol{L(W)} = -\frac{1}{m}\sum_{i=1}^{m} \boldsymbol{y}^{(i)} \boldsymbol{ln}(\widehat{\boldsymbol{y}}^{(i)})$$

- Adding regularization terms: L1, L2, ……

Parameter Initialization Techniques

- Initialize weights with Small random numbers
  - Set bias to zero initially
- Layer-wise pretraining
- Pretrained model+finetune



Employ Mini-batch instead of SGD

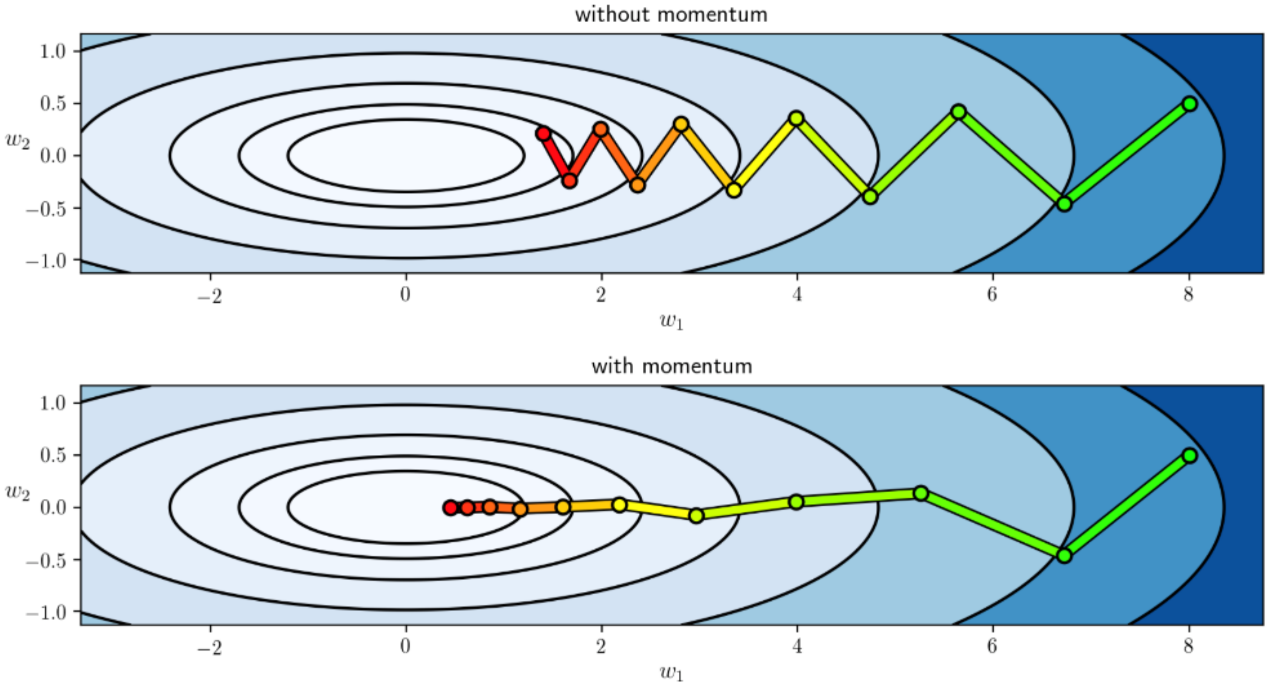
- Bias and Variance
- Small mini-batch usually don't change the bias expectation of SGD
- Mini-batch can fully exploit the GPU power
- Big mini-batch can have lower variance without big loss of bias expectation
  - Can have bigger learning rate
  - Correspondingly, with small mini-batch, learning rate should also be small
- Can set the learning rate linearly with mini-batch size

Gradient Updating Optimization

- The problem of gradient estimation error
- Momentum+

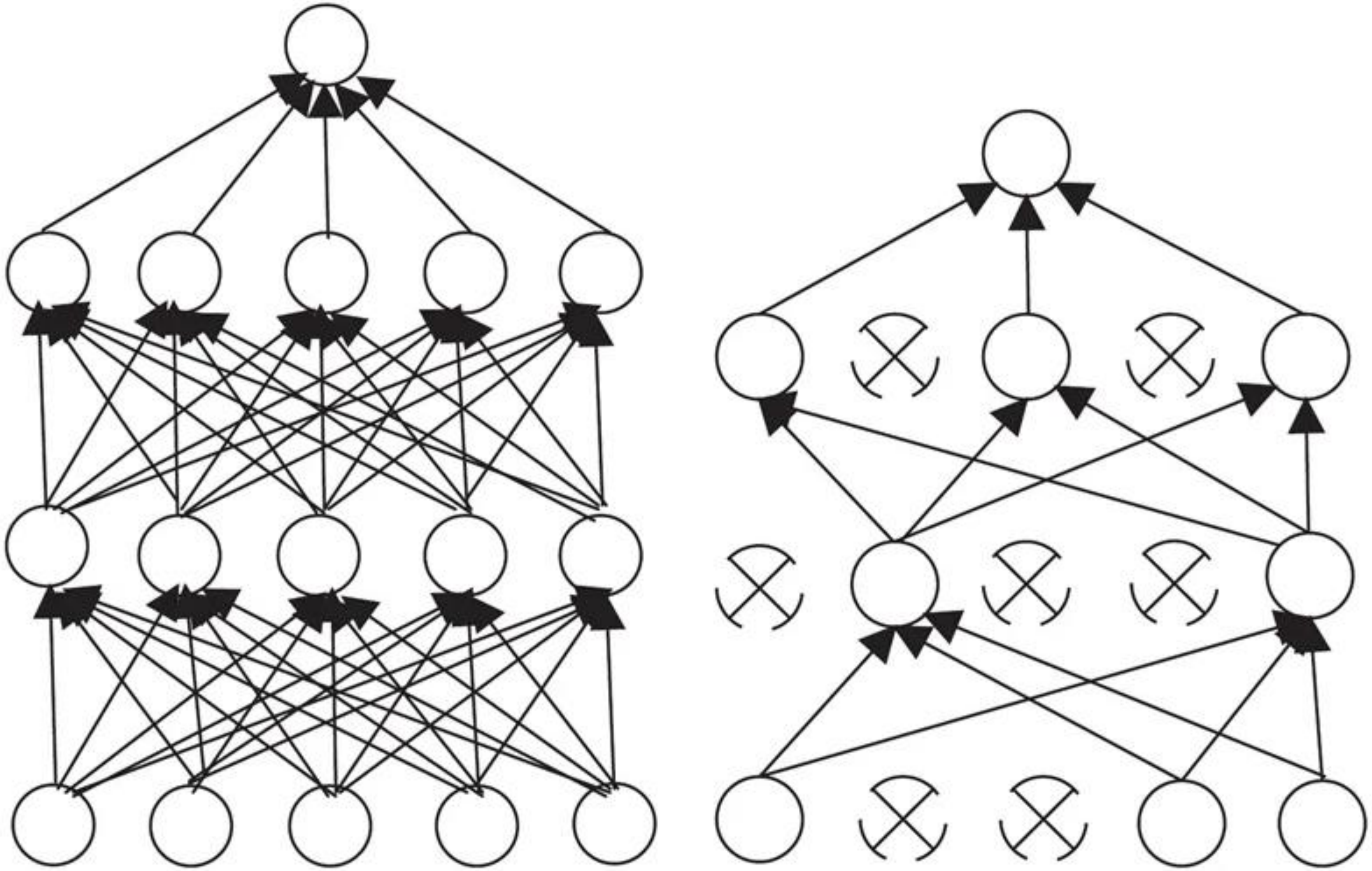
  Adaptive Learning Rate
- Adam

$$\Delta\theta_t = \rho\Delta\theta_{t-1} - \alpha\mathbf{g}_t$$



- Gradient clipping

Dropout



Other Classical Techniques

- Batch normalization
- Layer normalization
- Weight decay
- Early stopping
- Hyperparameter optimization
  - Grid search
  - Bayesian optimization

Acknowledgements

- Some text, figures and formulations are from WWW.
- This lecture is distributed for nonprofit purpose.

Thank You for Your Attention

Contact me at: yym@hit.edu.cn

Tel: 26033008, 13760196623

Address: Rm.1402, H# Building