

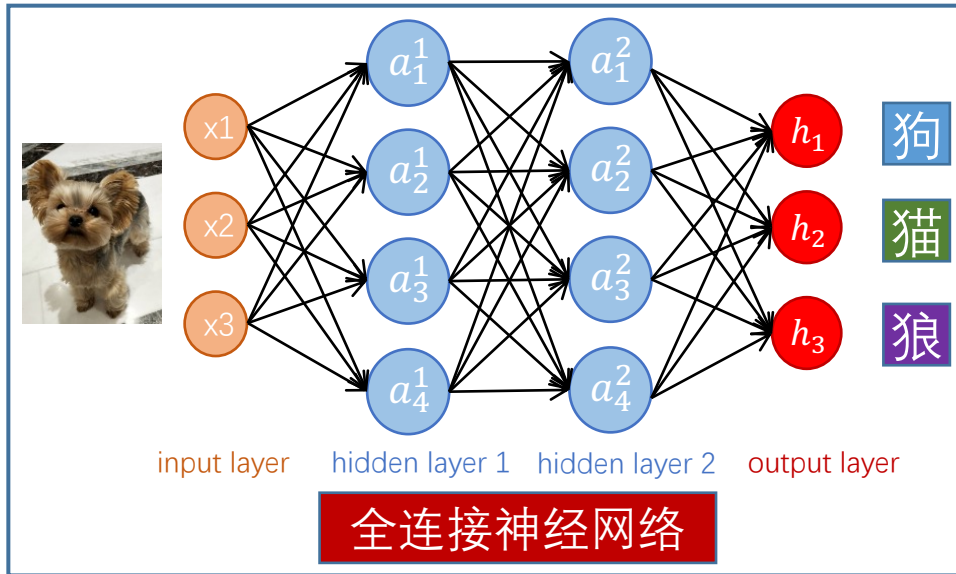
# 计算机视觉

# Computer Vision

Lecture 5: 卷积神经网络



# L04：神经网络和反向传播



反向传播

$$\frac{\partial L}{\partial W^l} = \frac{\partial L}{\partial h} \frac{\partial h}{\partial W^l}$$

$$\frac{\partial L}{\partial W^{l-1}} = \frac{\partial L}{\partial h} \frac{\partial h}{\partial a^l} \frac{\partial a^l}{\partial W^{l-1}}$$

$$\frac{\partial L}{\partial W^{l-2}} = \frac{\partial L}{\partial h} \frac{\partial h}{\partial a^l} \frac{\partial a^l}{\partial a^{l-1}} \frac{\partial a^{l-1}}{\partial W^{l-2}}$$

⋮

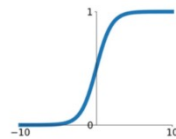
Chain rule

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial h} \frac{\partial h}{\partial a^l} \frac{\partial a^l}{\partial a^{l-1}} \frac{\partial a^{l-1}}{\partial a^{l-2}} \cdots \cdots \frac{\partial a^2}{\partial W^1}$$

激活函数

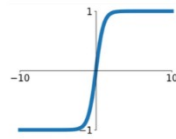
**Sigmoid**

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



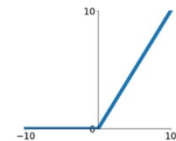
**tanh**

$$\tanh(x)$$



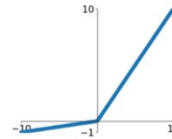
**ReLU**

$$\max(0, x)$$



**Leaky ReLU**

$$\max(0.1x, x)$$

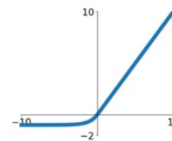


**Maxout**

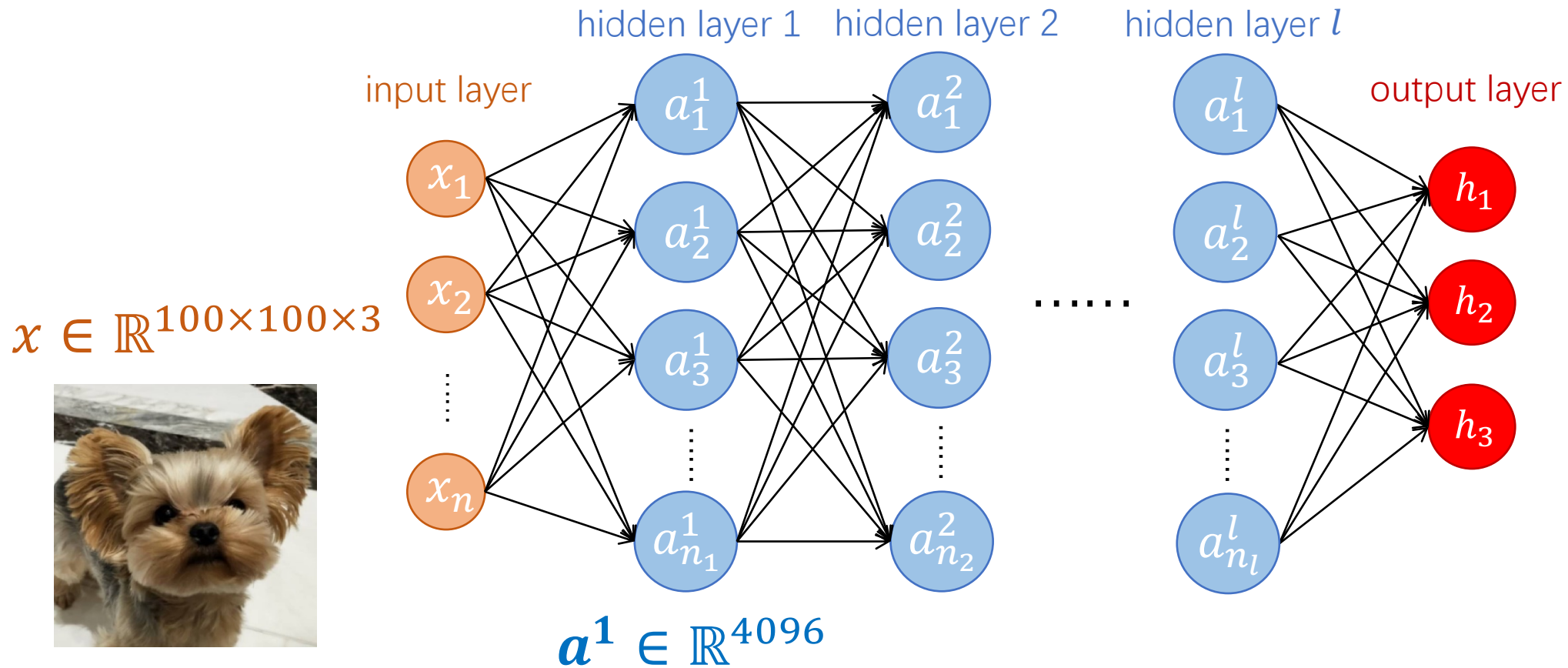
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

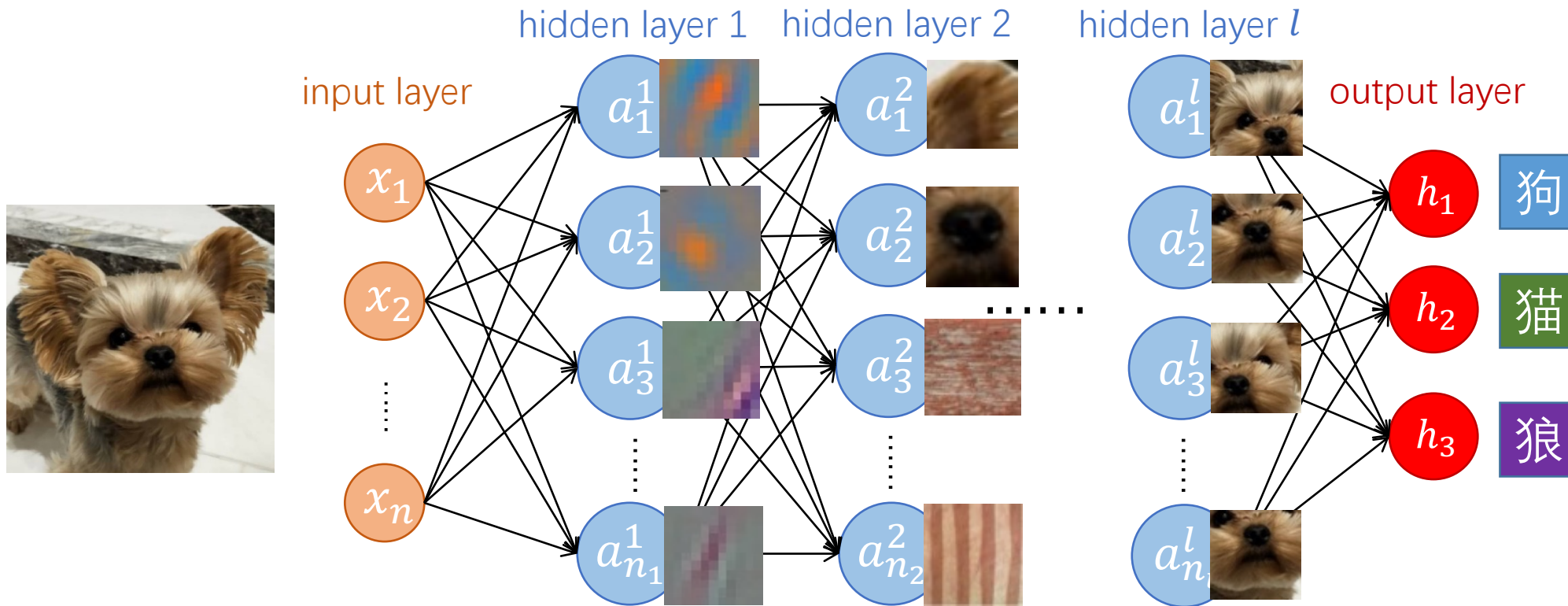


# 全连接神经网络 (FCNN) 的问题



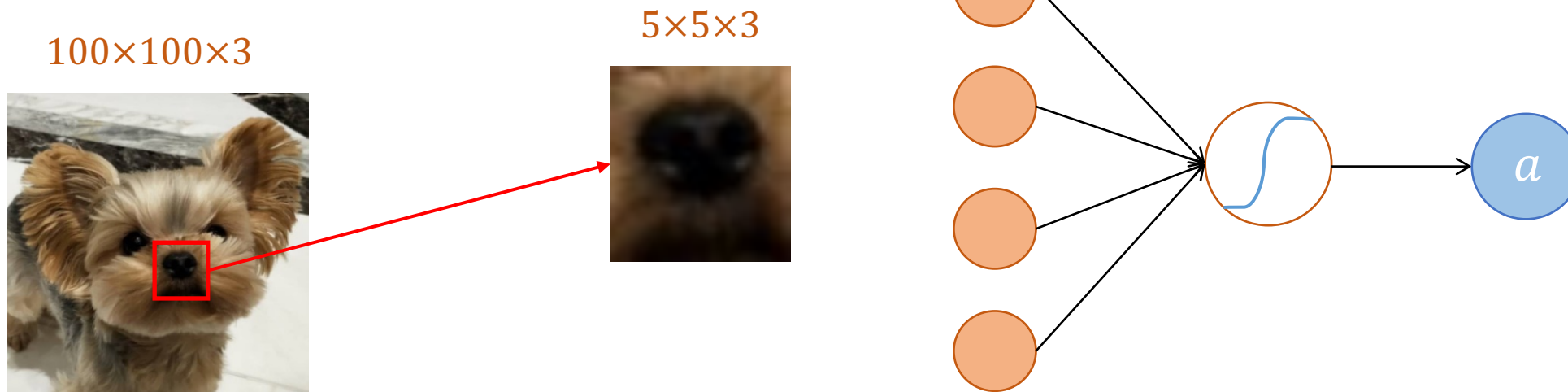
第一层参数个数：  $30000 \times 4096 \approx 1.2$  亿！！

# 观察FCNN



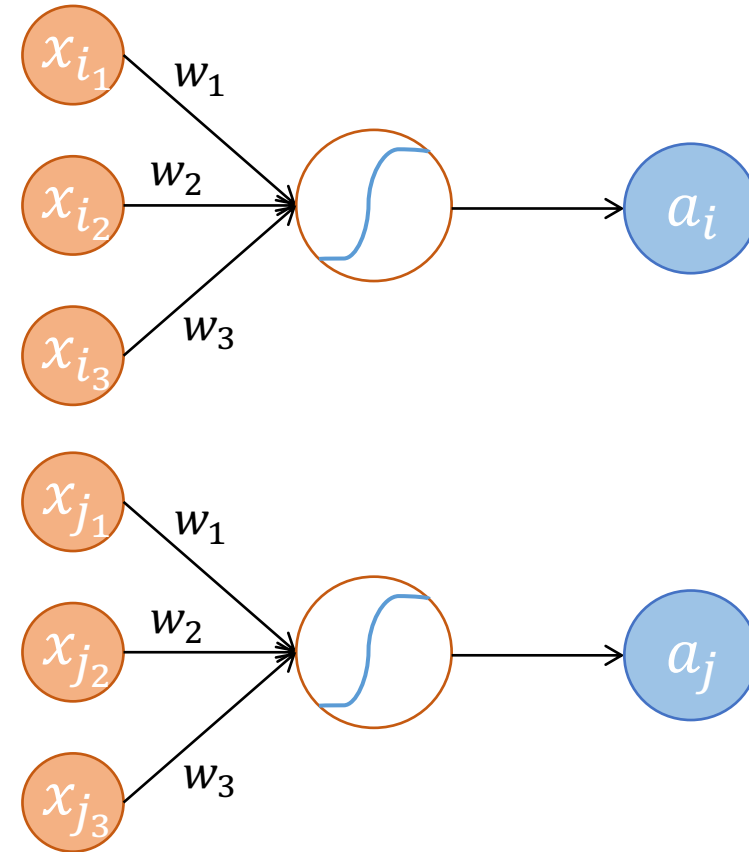
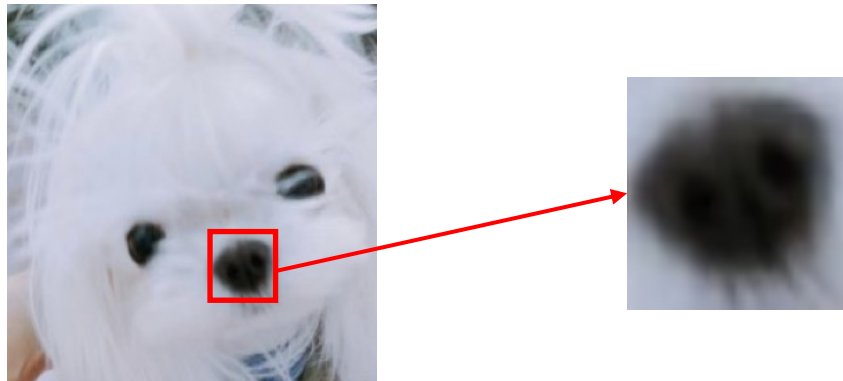
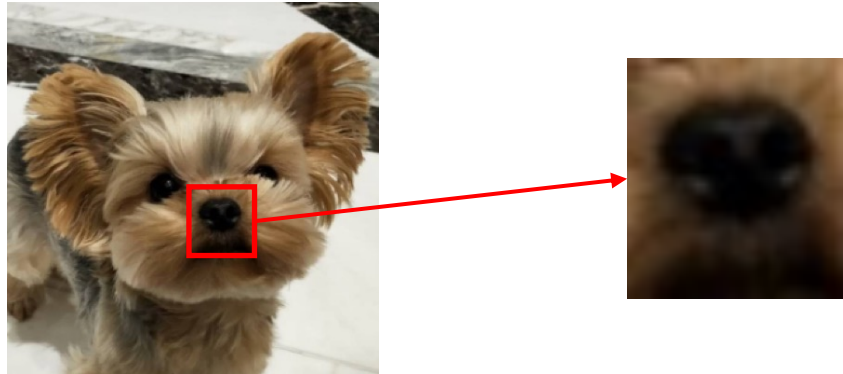
- ✓ 每个neural只关注某类比原图小很多的局部pattern
- ✓ 相似的局部pattern会出现在图片的不同部分
- ✓ 对像素做采样之后，图像的内容不会发生改变

# 局部pattern



每个神经元捕获局部pattern, 减少权重参数

# 相似pattern, 不同区域



功能相似的神经元可以共享权重参数

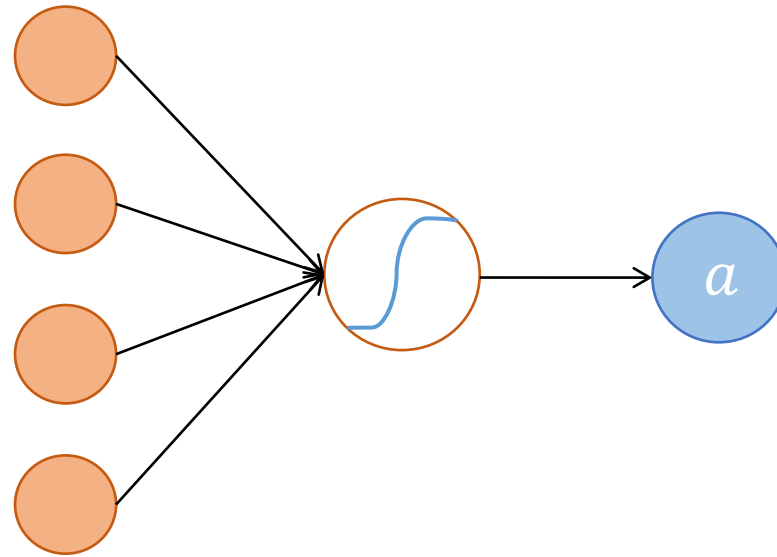
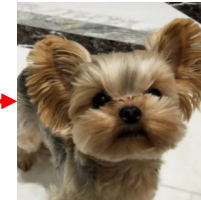
# 像素采样

$100 \times 100 \times 3$



downsampling

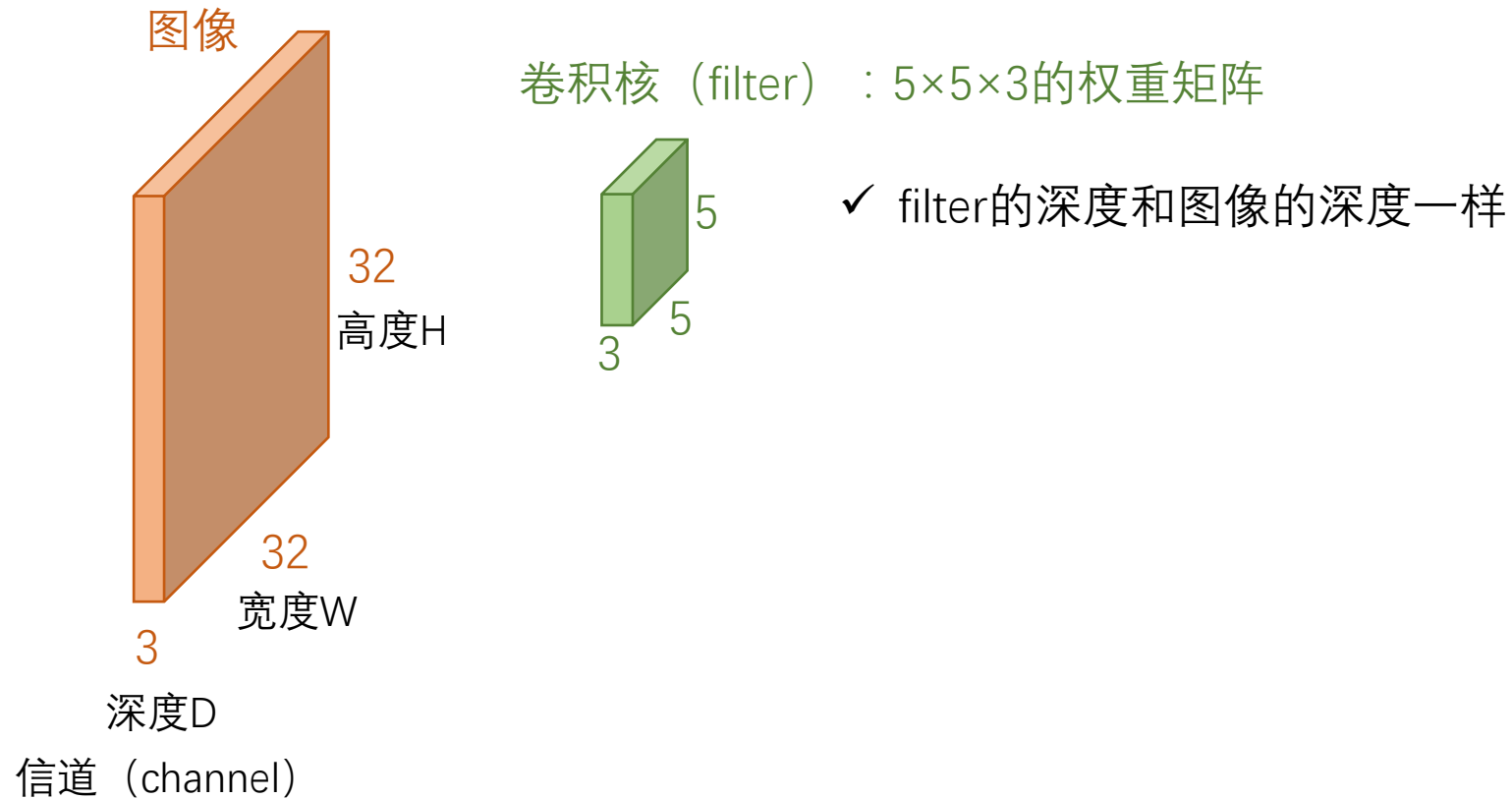
$50 \times 50 \times 3$



进一步减少参数数量

# 卷积神经网络

- Convolutional Neural Networks

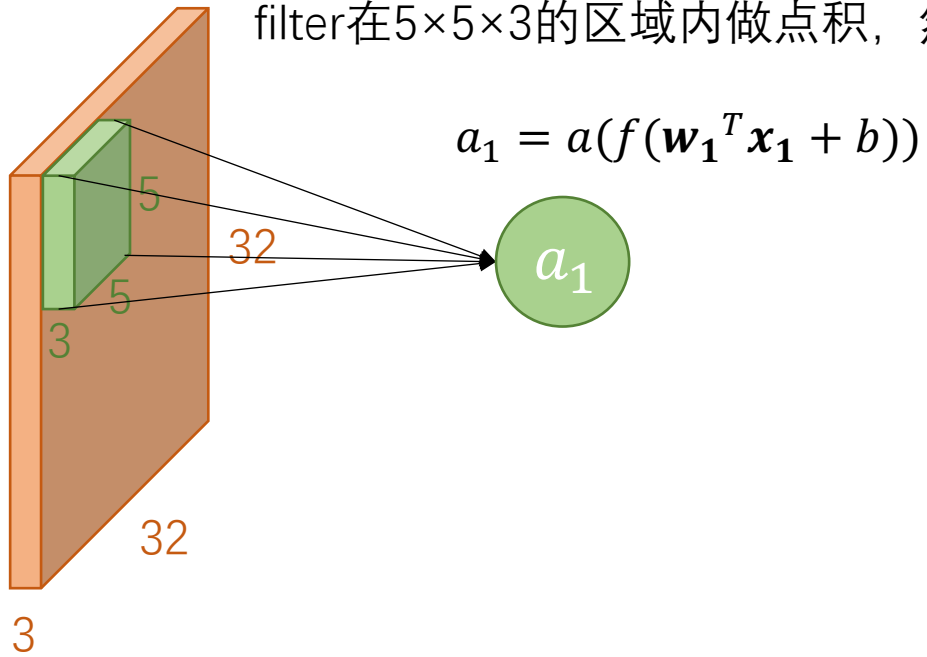




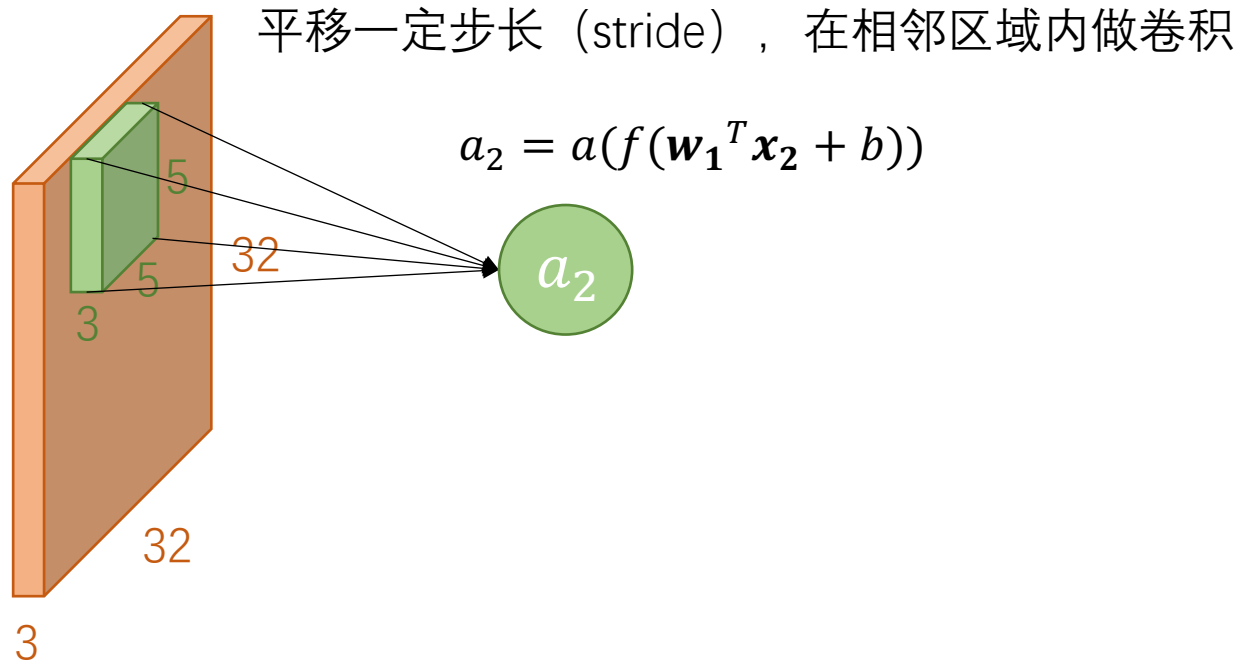
# 卷积神经网络

## 卷积

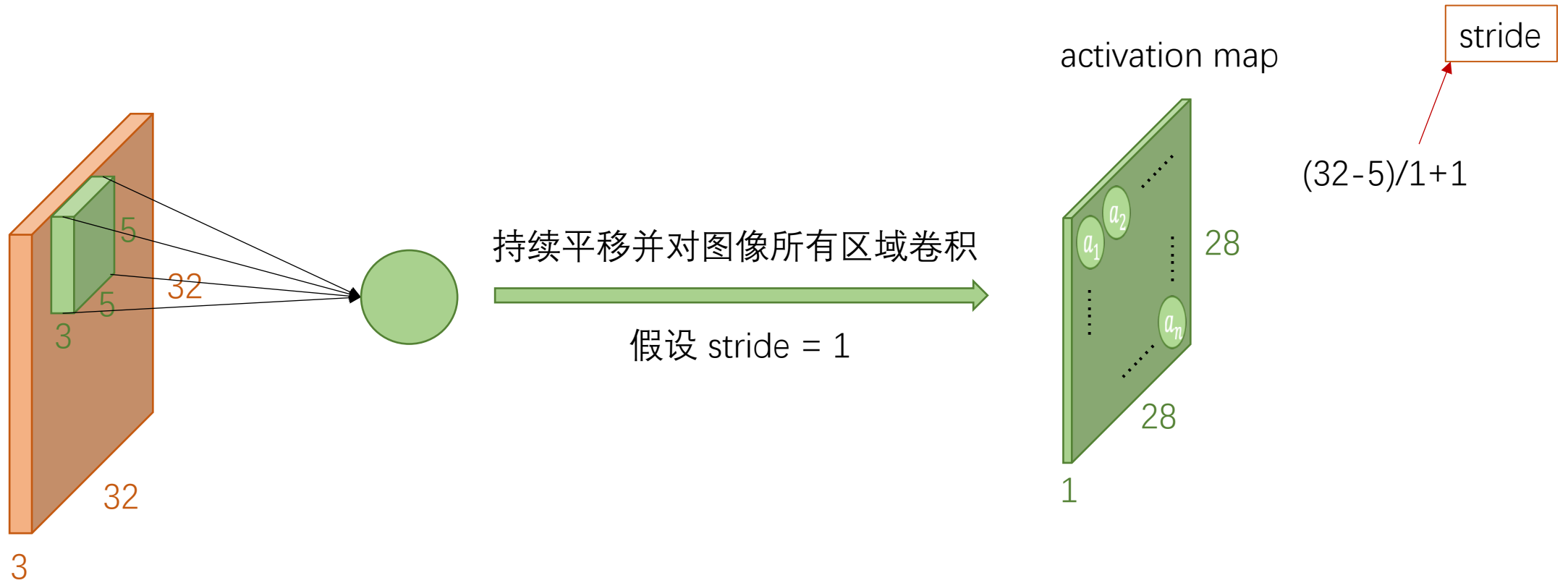
filter在 $5 \times 5 \times 3$ 的区域内做点积，然后激活（一般用ReLU）



# 卷积神经网络

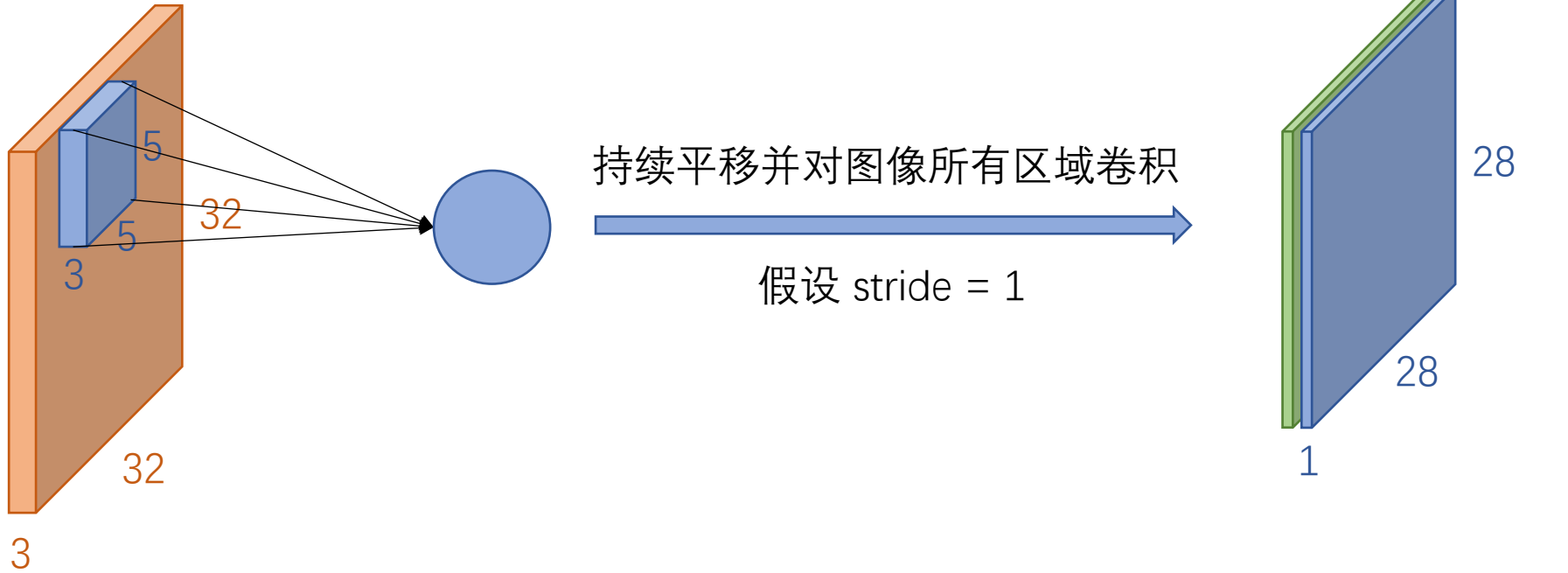


# 卷积神经网络



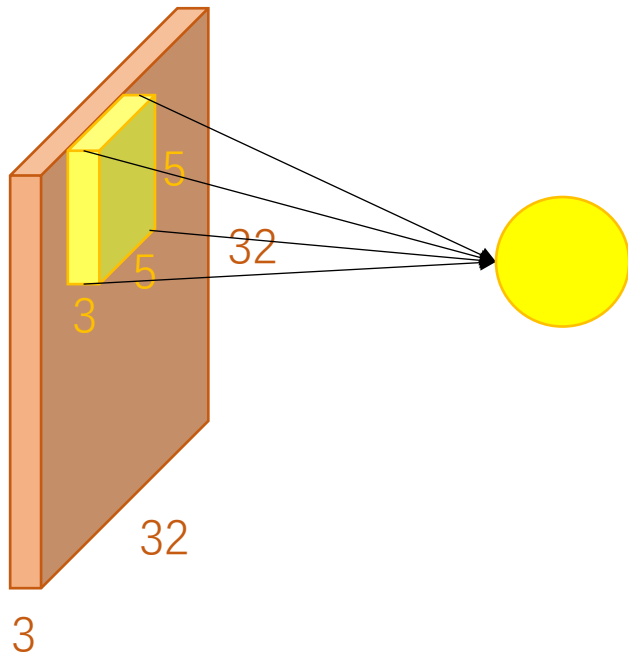
# 卷积神经网络

第二个filter：第二组 $5 \times 5 \times 3$ 的权重矩阵

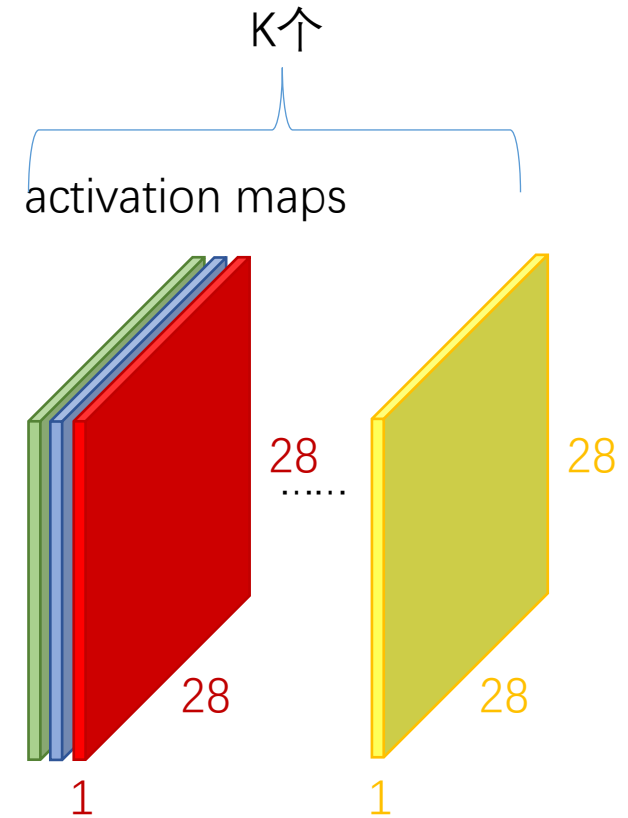


# 卷积神经网络

K个filter：K组 $5 \times 5 \times 3$ 的权重矩阵

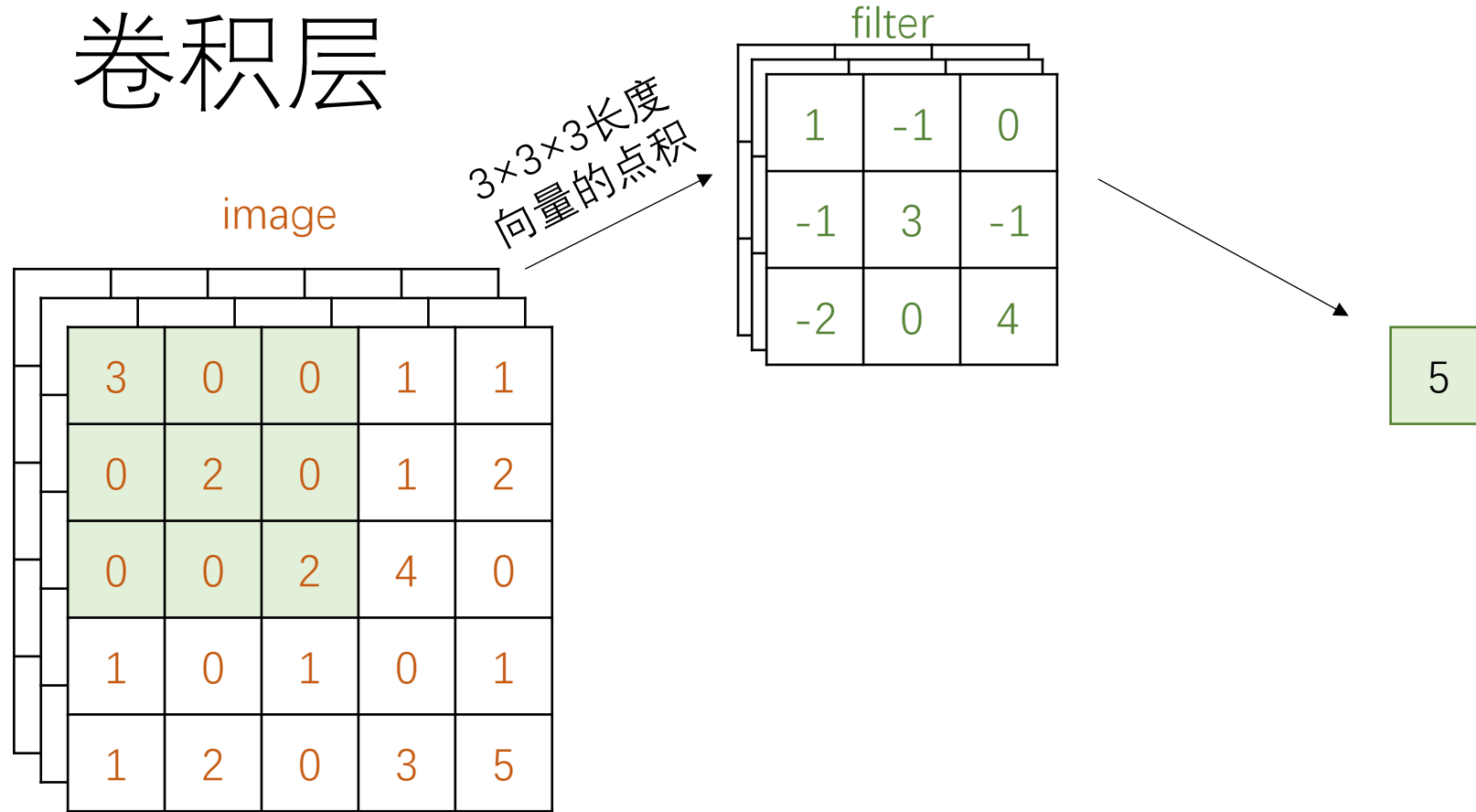


持续平移并对图像所有区域卷积  
假设 stride = 1

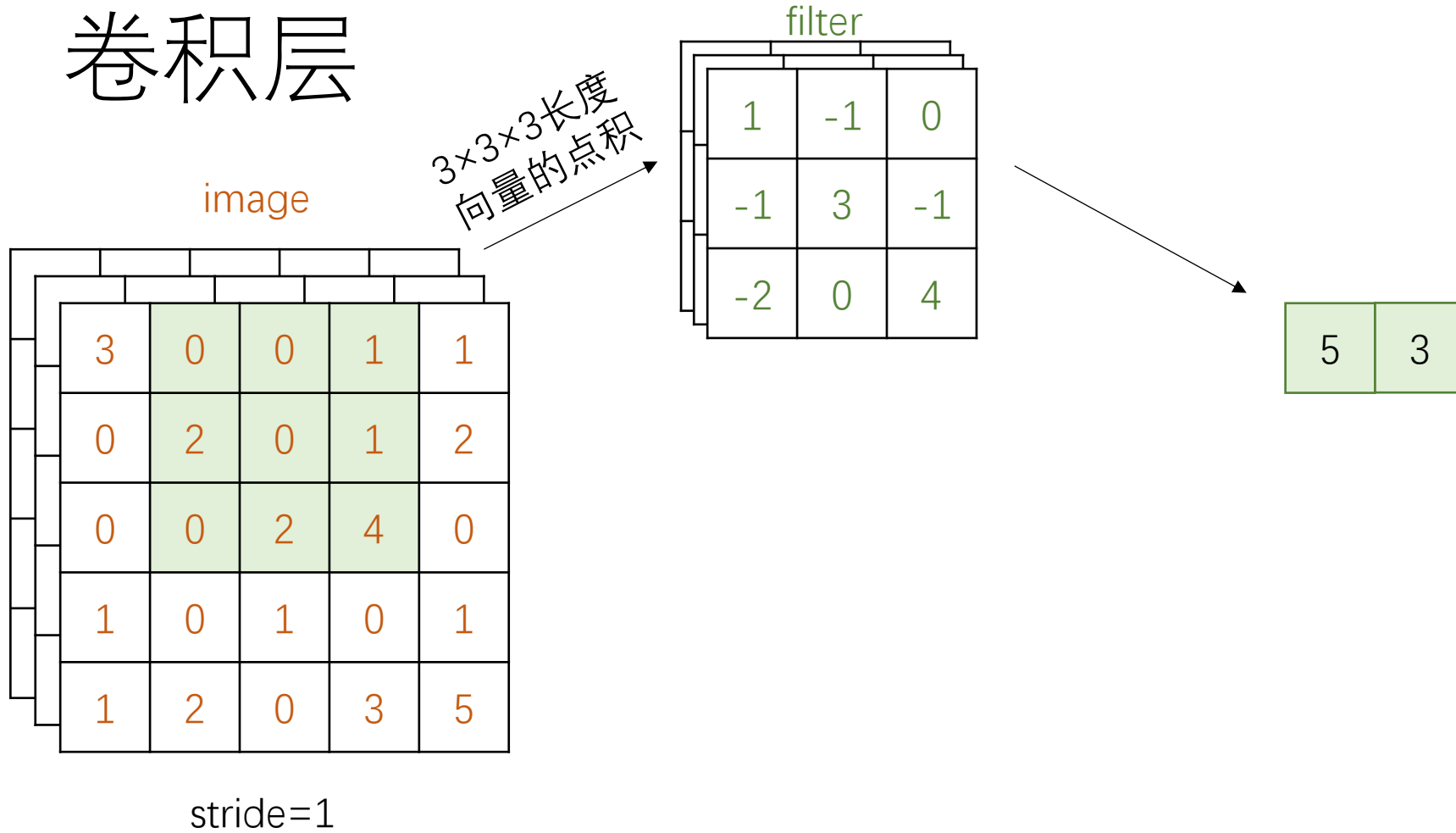


✓ activation maps大小为 $28 \times 28 \times K$   
(即activation maps的个数等于filter的个数)

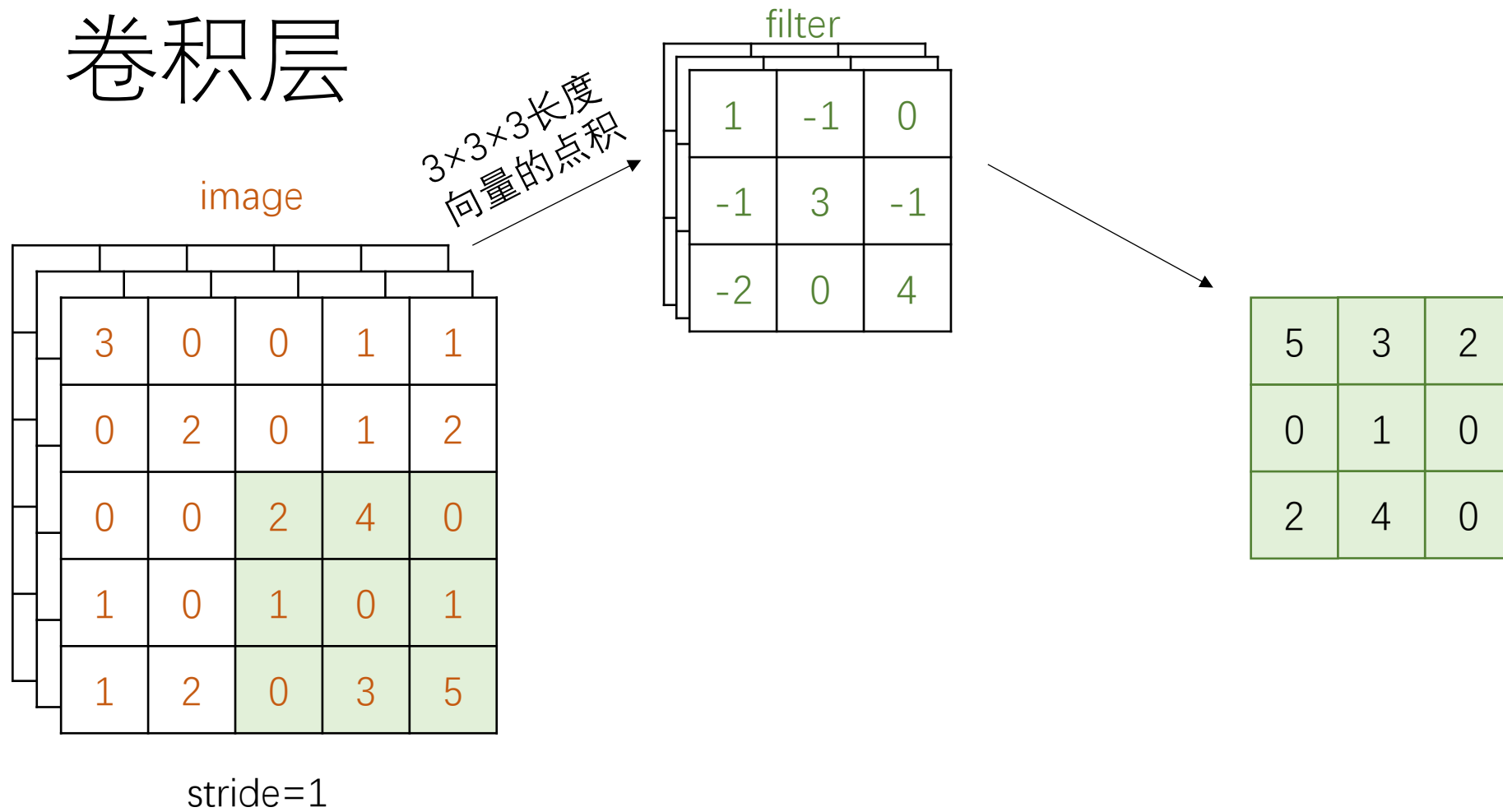
# 卷积层



# 卷积层

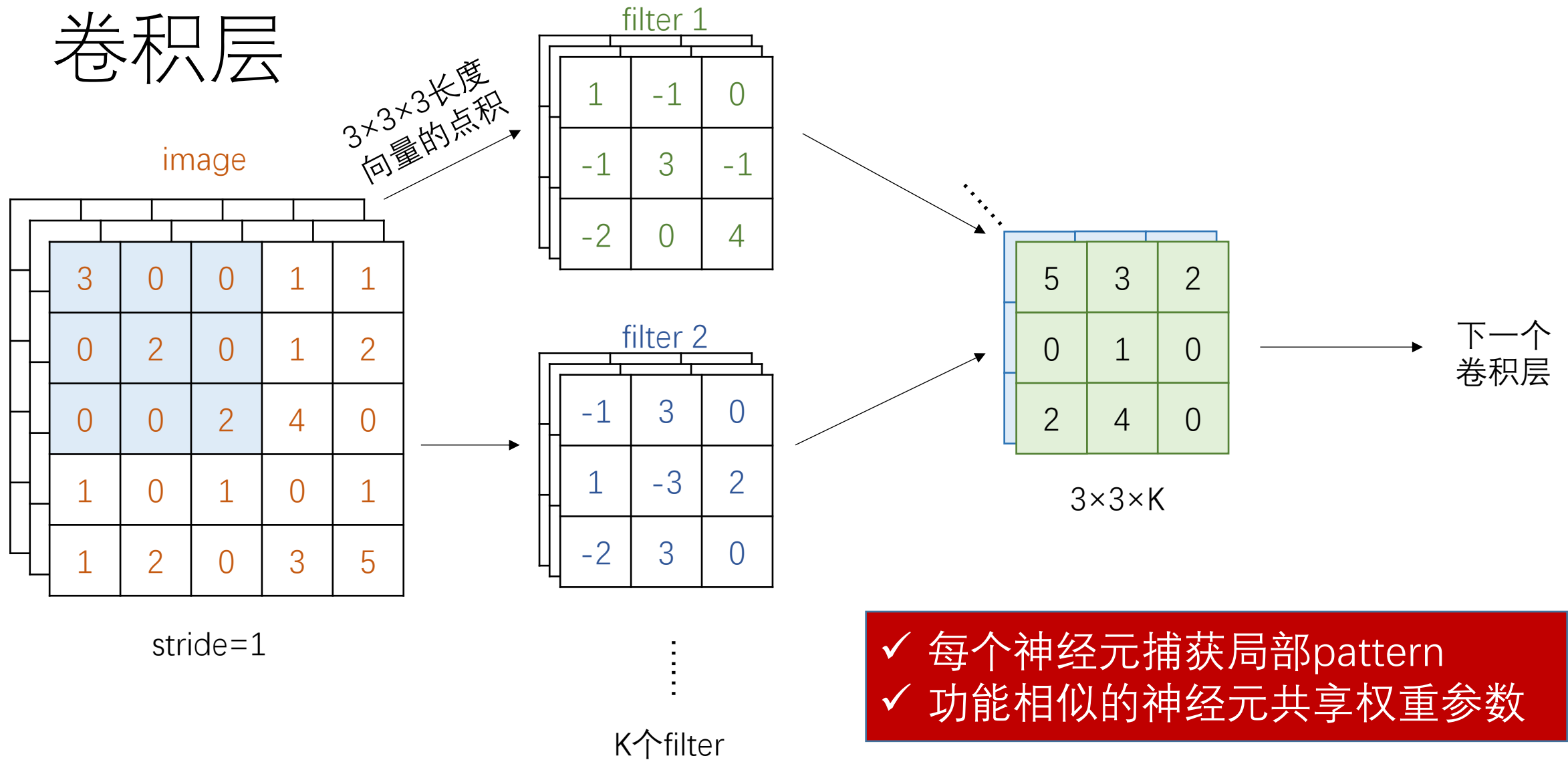


# 卷积层

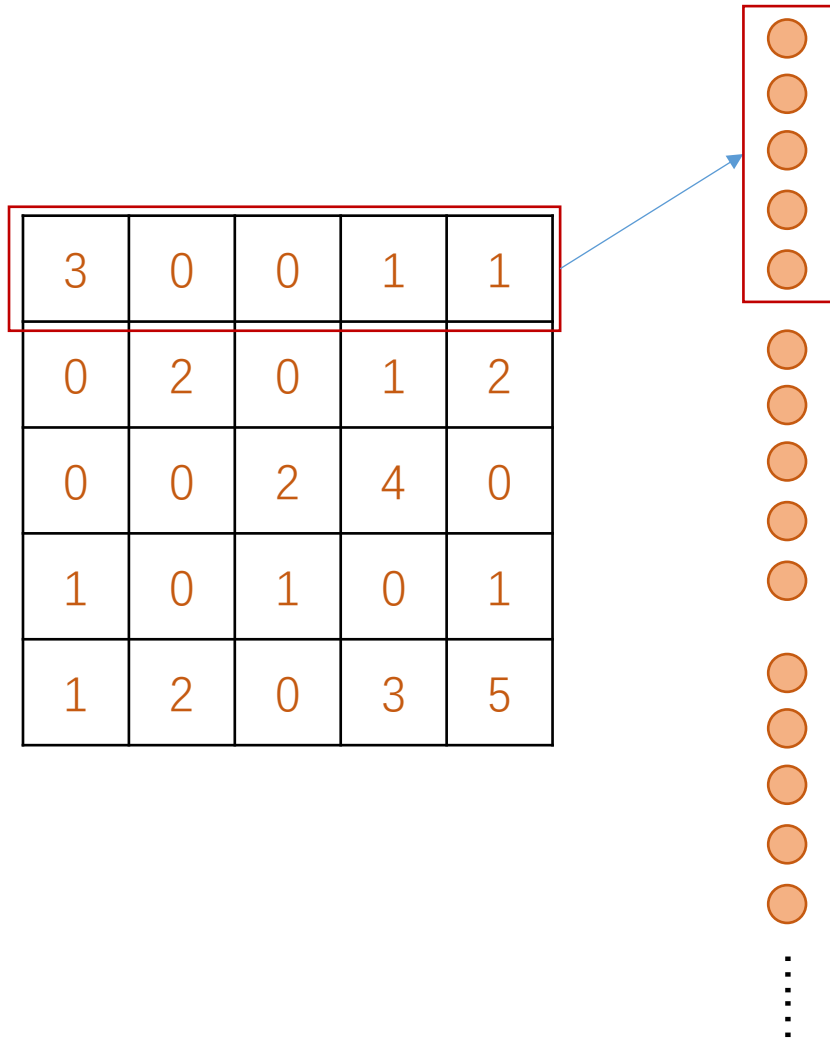




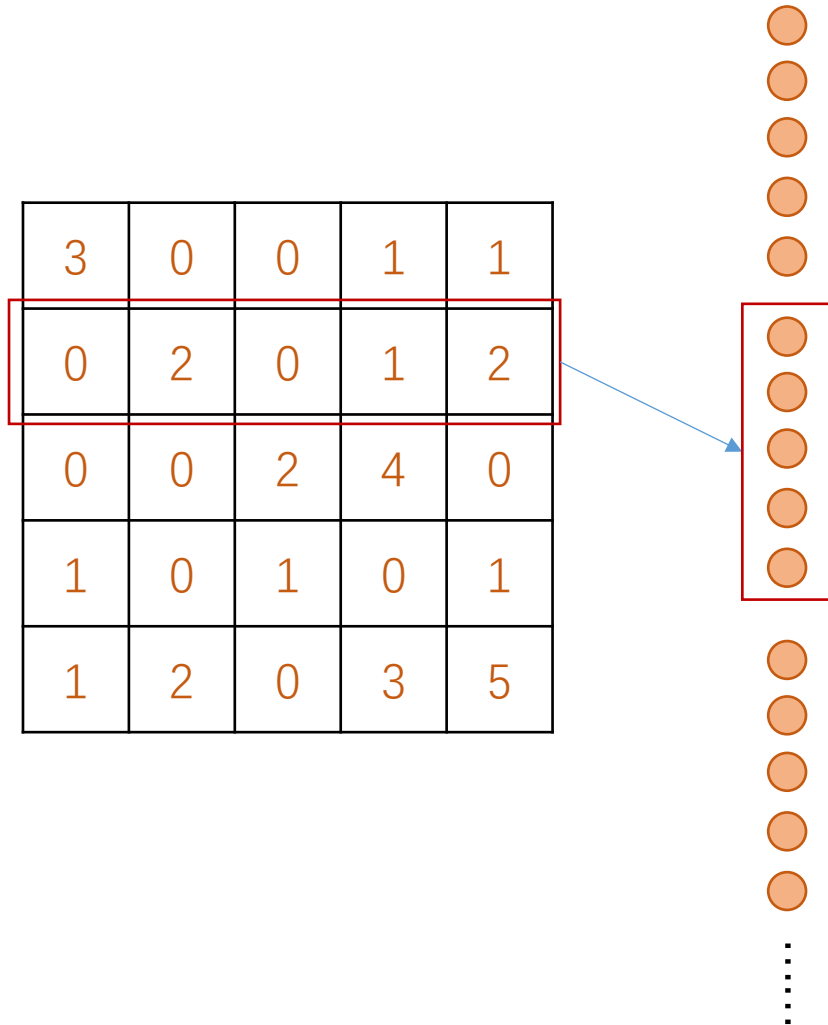
# 卷积层



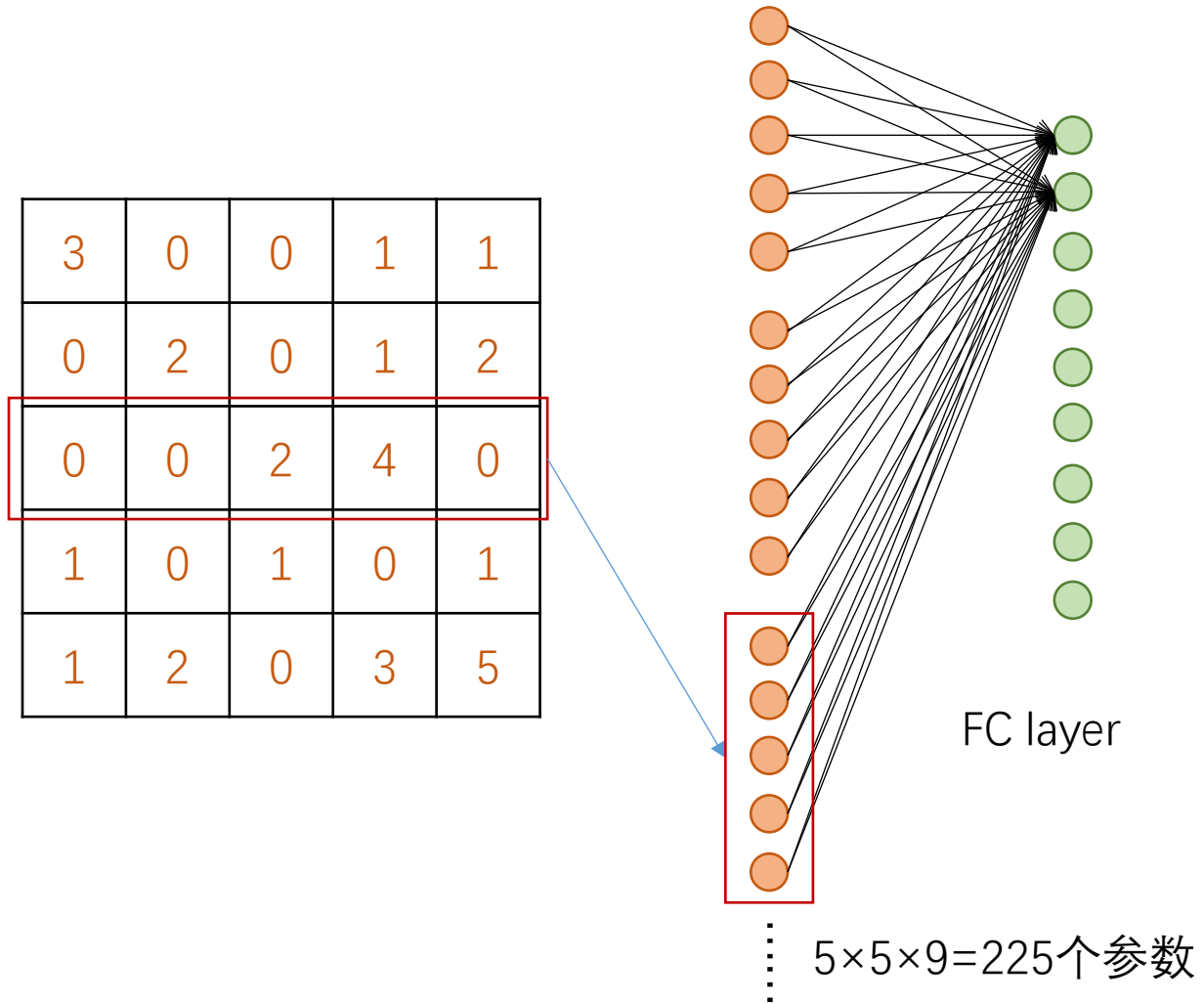
# FC层 Vs 卷积层



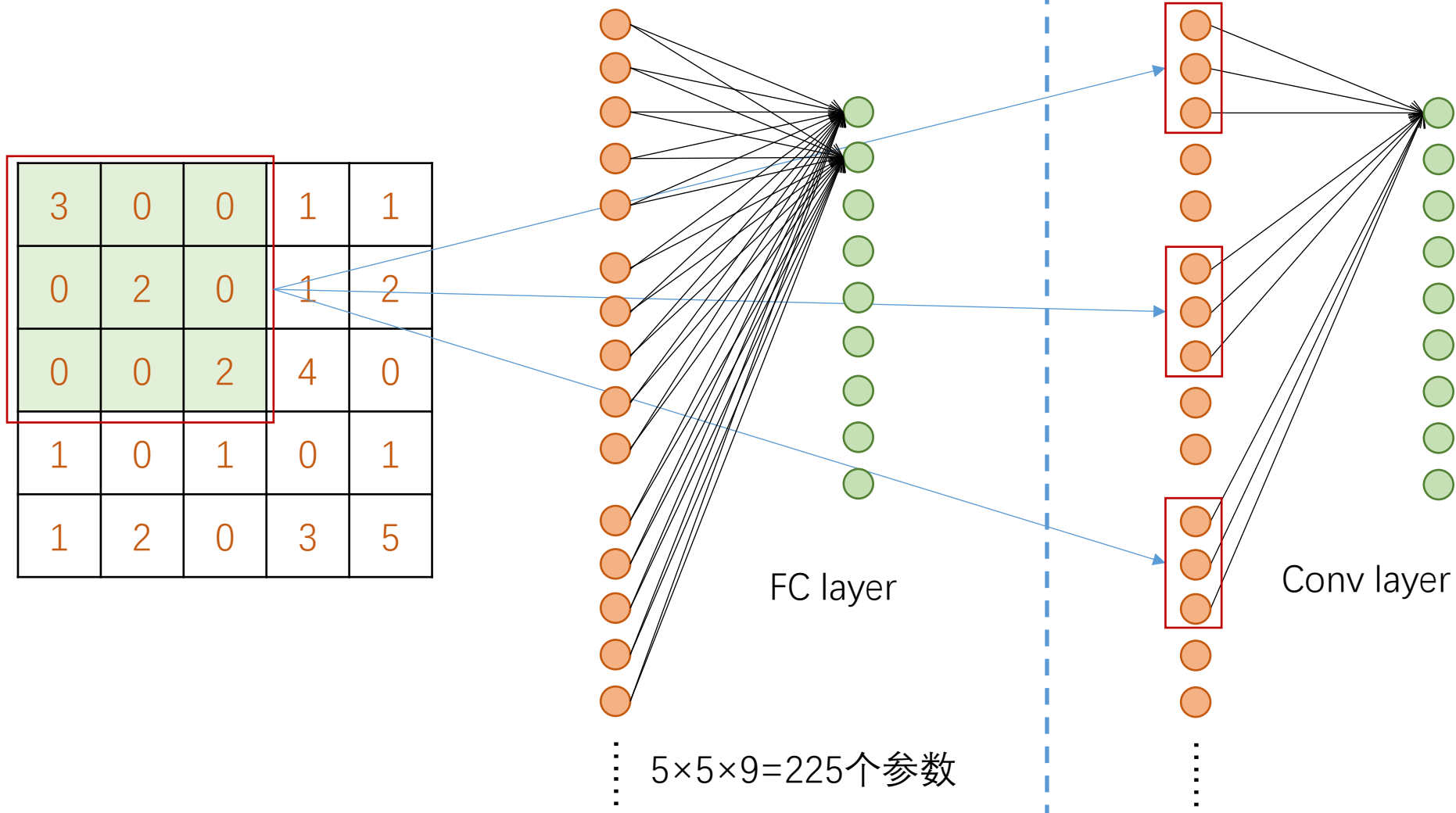
# FC层 Vs 卷积层



# FC层 Vs 卷积层

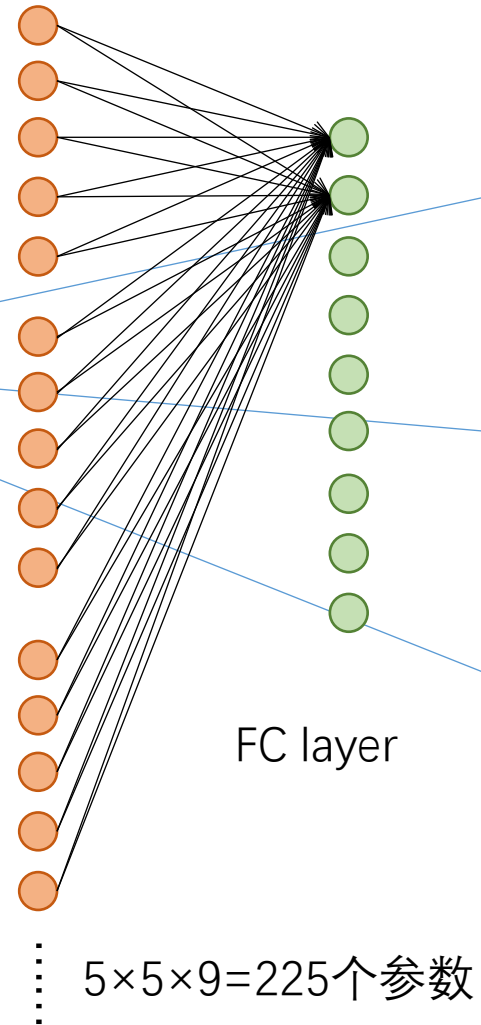


# FC层 Vs 卷积层

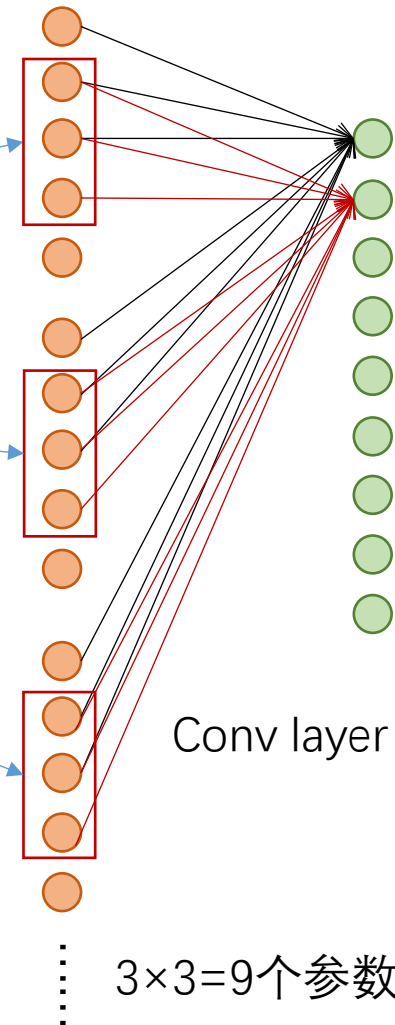


# FC层 Vs 卷积层

3	0	0	1	1
0	2	0	1	2
0	0	2	4	0
1	0	1	0	1
1	2	0	3	5

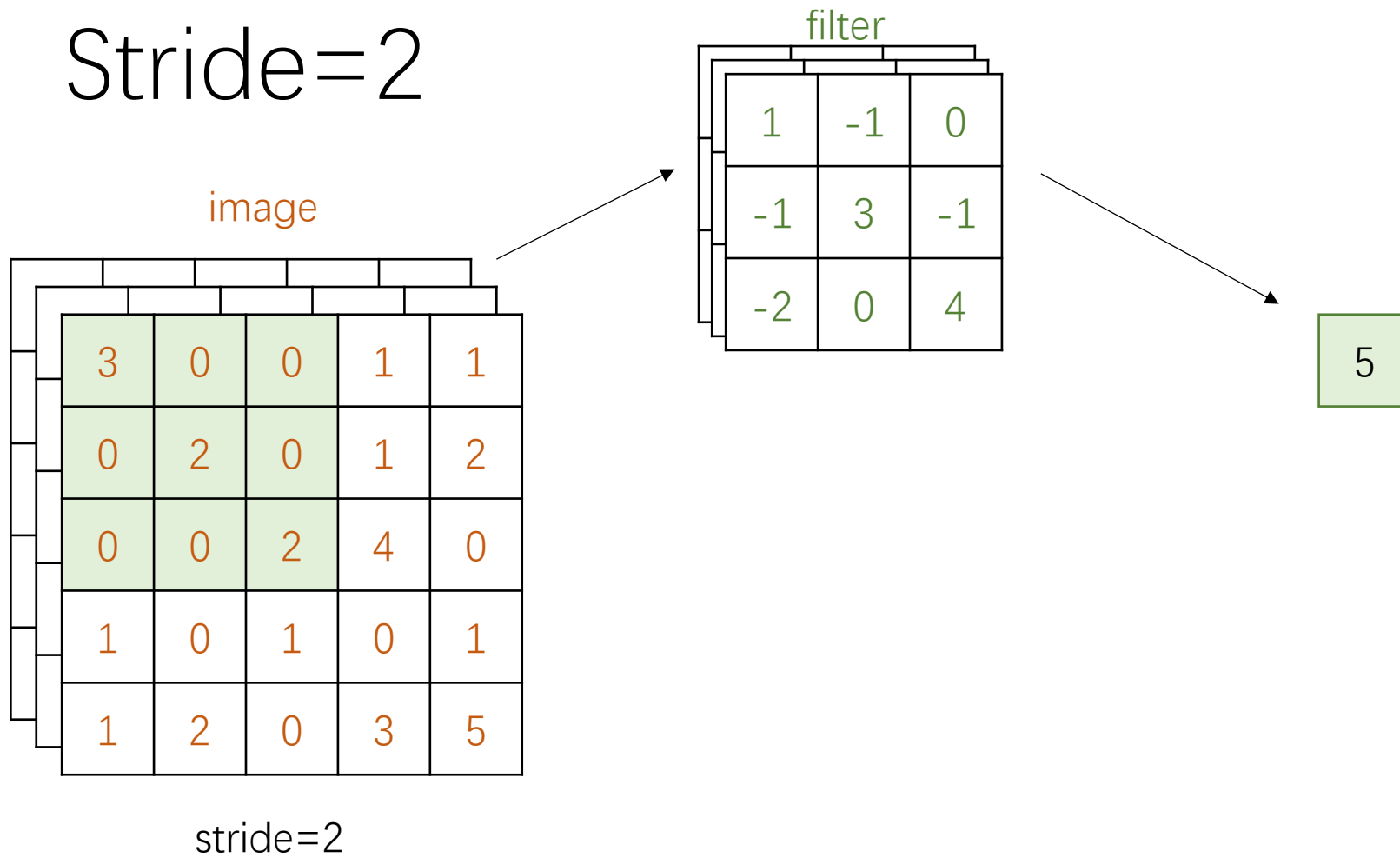


连接局部input → 减少参数数量  
权重共享, 进一步减少参数数量

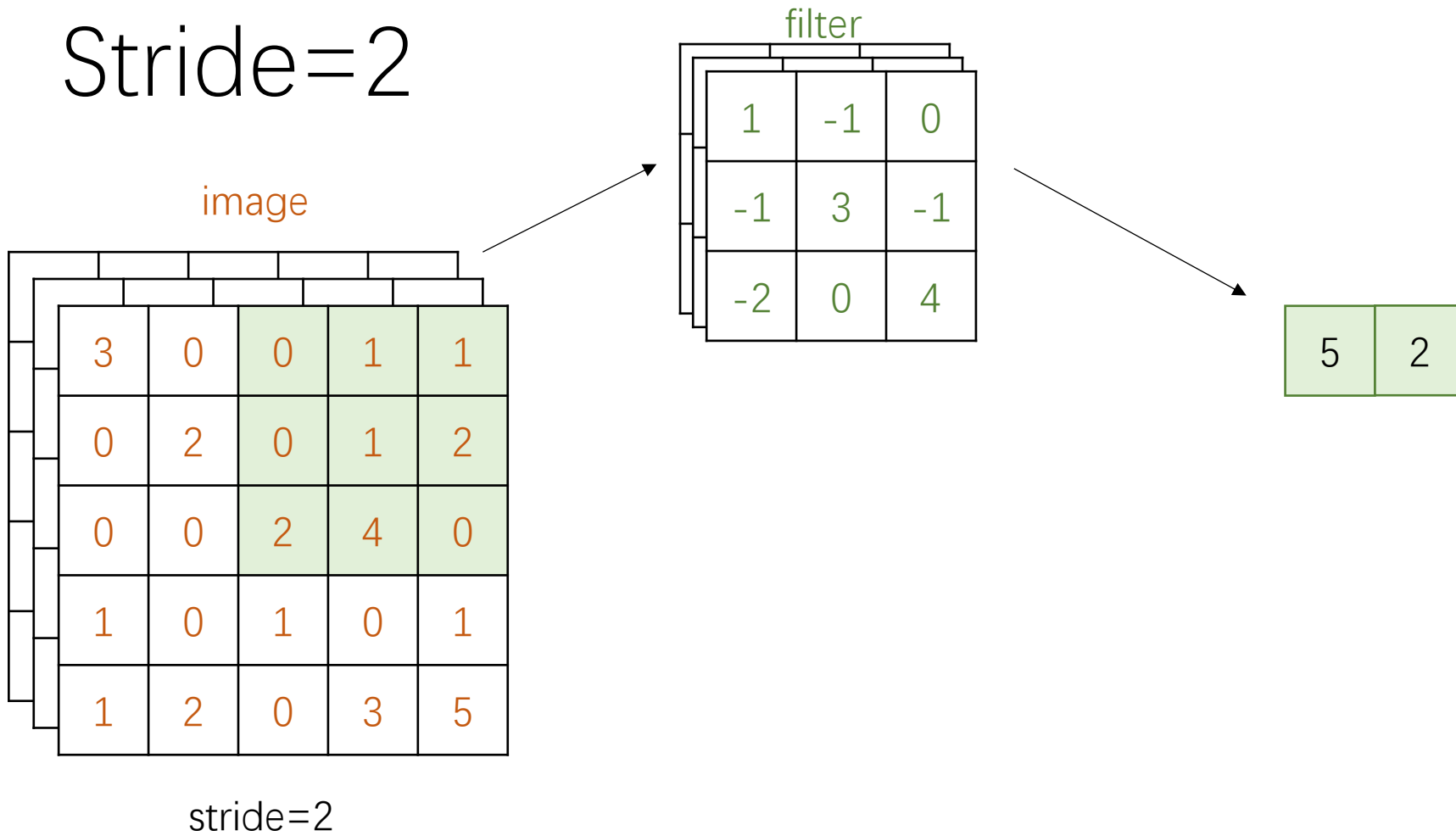


红黑权重相同 (同一个filter)

# Stride=2

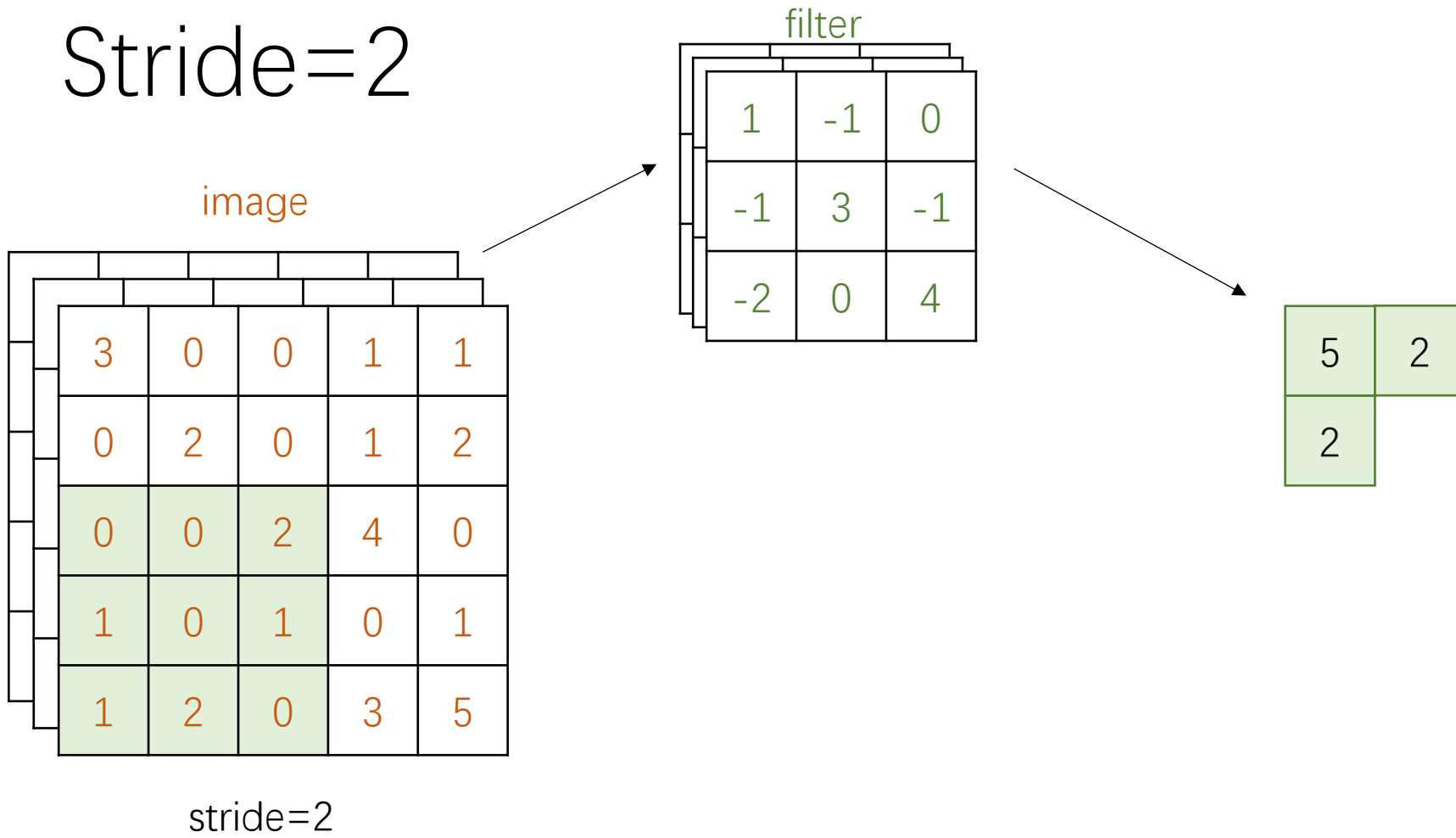


# Stride=2

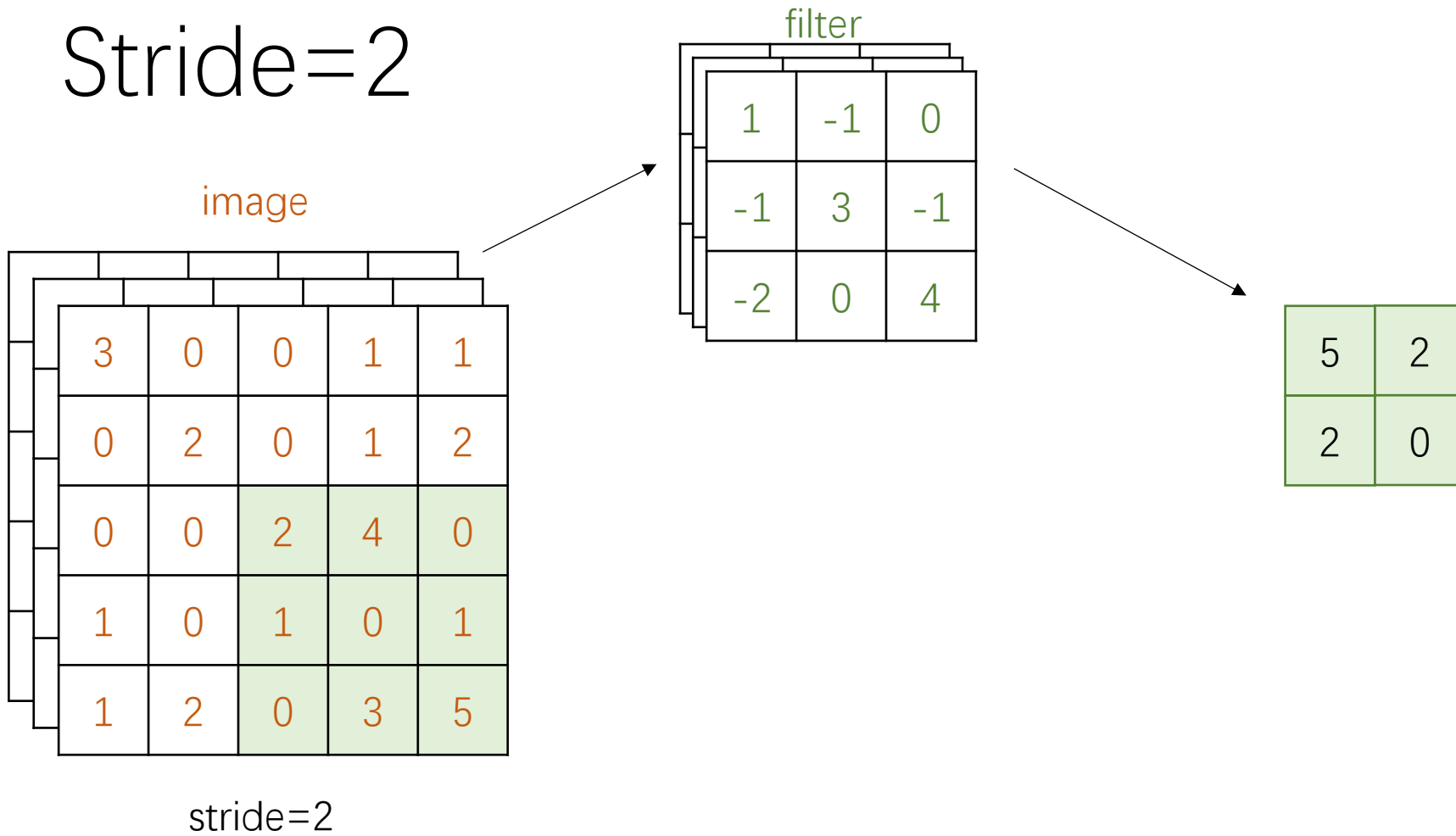




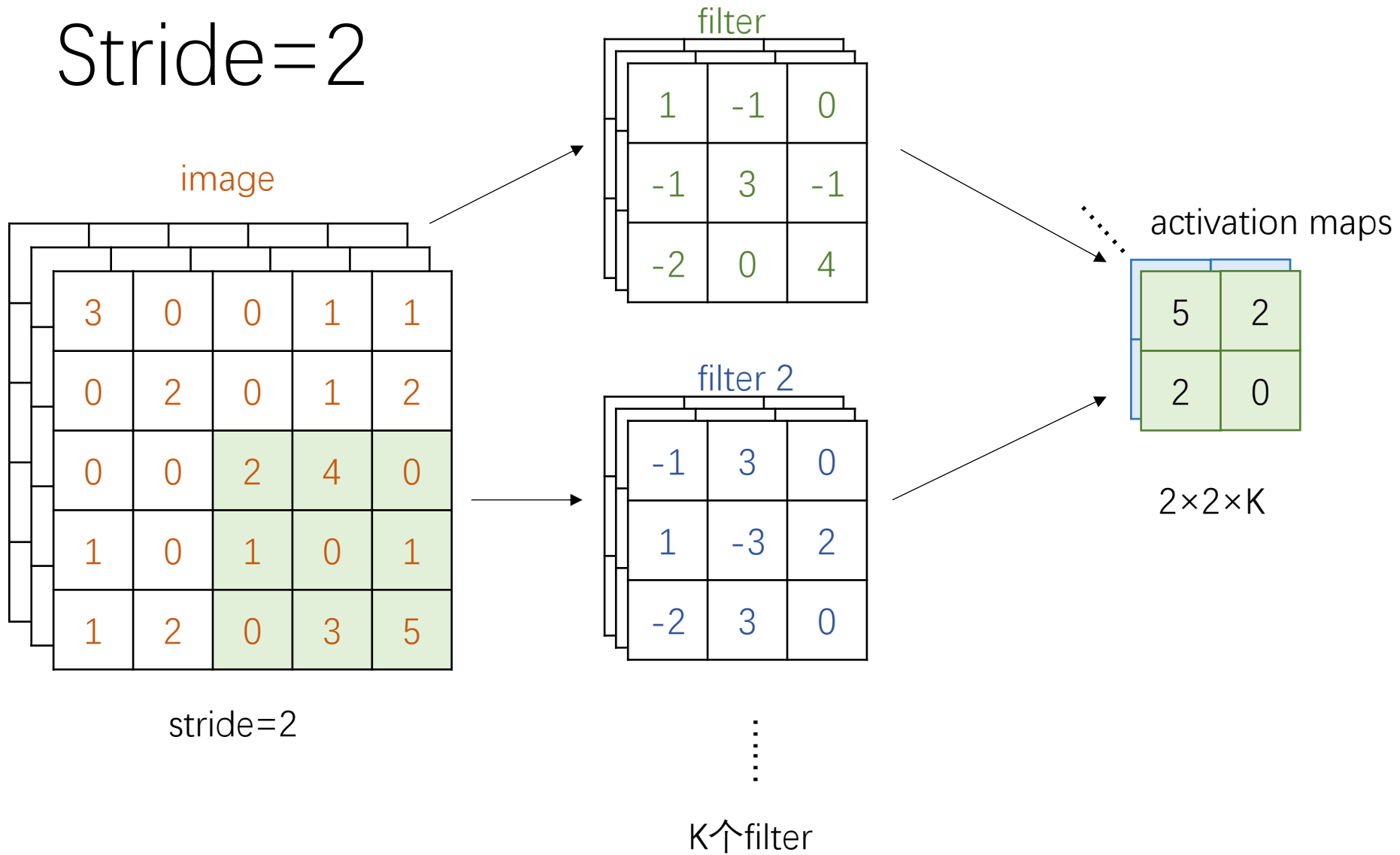
# Stride=2



# Stride=2



# Stride=2



# Zero Padding

filter无法捕获图像边缘的pattern  
✓ 用0对图像边缘进行填充

0	0	0	0	0	0	0
0	3	0	0	1	1	0
0	0	2	0	1	2	0
0	0	0	2	4	0	0
0	1	0	1	0	1	0
0	1	2	0	3	5	0
0	0	0	0	0	0	0

# Zero Padding

filter无法捕获图像边缘的pattern  
✓ 用0对图像边缘进行填充

0	0	0	0	0	0	0
0	3	0	0	1	1	0
0	0	2	0	1	2	0
0	0	0	2	4	0	0
0	1	0	1	0	1	0
0	1	2	0	3	5	0
0	0	0	0	0	0	0

# Zero Padding

filter无法捕获图像边缘的pattern  
✓ 用0对图像边缘进行填充

0	0	0	0	0	0	0
0	3	0	0	1	1	0
0	0	2	0	1	2	0
0	0	0	2	4	0	0
0	1	0	1	0	1	0
0	1	2	0	3	5	0
0	0	0	0	0	0	0

# Zero Padding

filter无法捕获图像边缘的pattern  
✓ 用0对图像边缘进行填充

0	0	0	0	0	0	0
0	3	0	0	1	1	0
0	0	2	0	1	2	0
0	0	0	2	4	0	0
0	1	0	1	0	1	0
0	1	2	0	3	5	0
0	0	0	0	0	0	0

# Zero Padding

filter无法捕获图像边缘的pattern  
✓ 用0对图像边缘进行填充

0	0	0	0	0	0	0
0	3	0	0	1	1	0
0	0	2	0	1	2	0
0	0	0	2	4	0	0
0	1	0	1	0	1	0
0	1	2	0	3	5	0
0	0	0	0	0	0	0

image :  $N \times N$

filter :  $F \times F$

stride : 1

padding : 1

activation map :  $(N+1 \times 2 - F)/1 + 1$

image :  $5 \times 5$

filter :  $3 \times 3$

stride : 1

padding : 1

activation map :  $(5+1 \times 2 - 3)/1 + 1, 5 \times 5$



# Zero Padding

filter无法捕获图像边缘的pattern  
 ✓ 用0对图像边缘进行填充

0	0	0	0	0	0	0
0	3	0	0	1	1	0
0	0	2	0	1	2	0
0	0	0	2	4	0	0
0	1	0	1	0	1	0
0	1	2	0	3	5	0
0	0	0	0	0	0	0

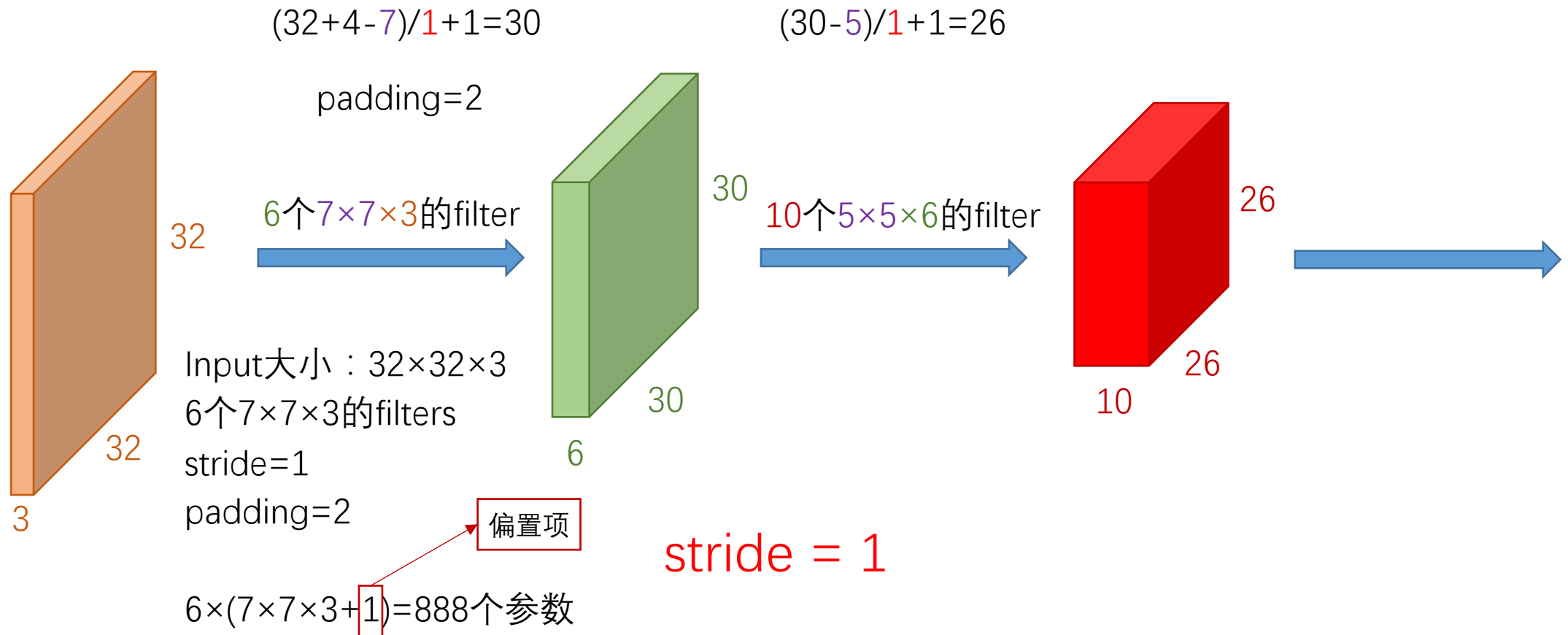
image :  $N \times N$                       filter :  $F \times F$   
 stride : 1                              padding : 1

activation map :  $(N+1 \times 2 - F) / 1 + 1$

现实模型中常常会在某些卷积层使用stride=1,  
 filter=  $F \times F$ , zero-padding= $(F-1)/2$ , 来保持卷  
 积后activation map大小不变（在pooling层减  
 小map)

$F=3 \rightarrow$  padding大小为1  
 $F=5 \rightarrow$  padding大小为2  
 $F=7 \rightarrow$  padding大小为3

# Activation Maps



# 卷积层小结

- 当前层输入： $W_1 \times H_1 \times D_1$
- 设置4个超参数：

- ✓ filter个数 $K$ ，大小 $F$
- ✓ stride大小 $S$
- ✓ zero padding数量 $P$

常用设置：

$K=32, 64, 128, 256, \dots$

$F=3, S=1, P=1$

$F=5, S=1, P=1$ 或 $2$

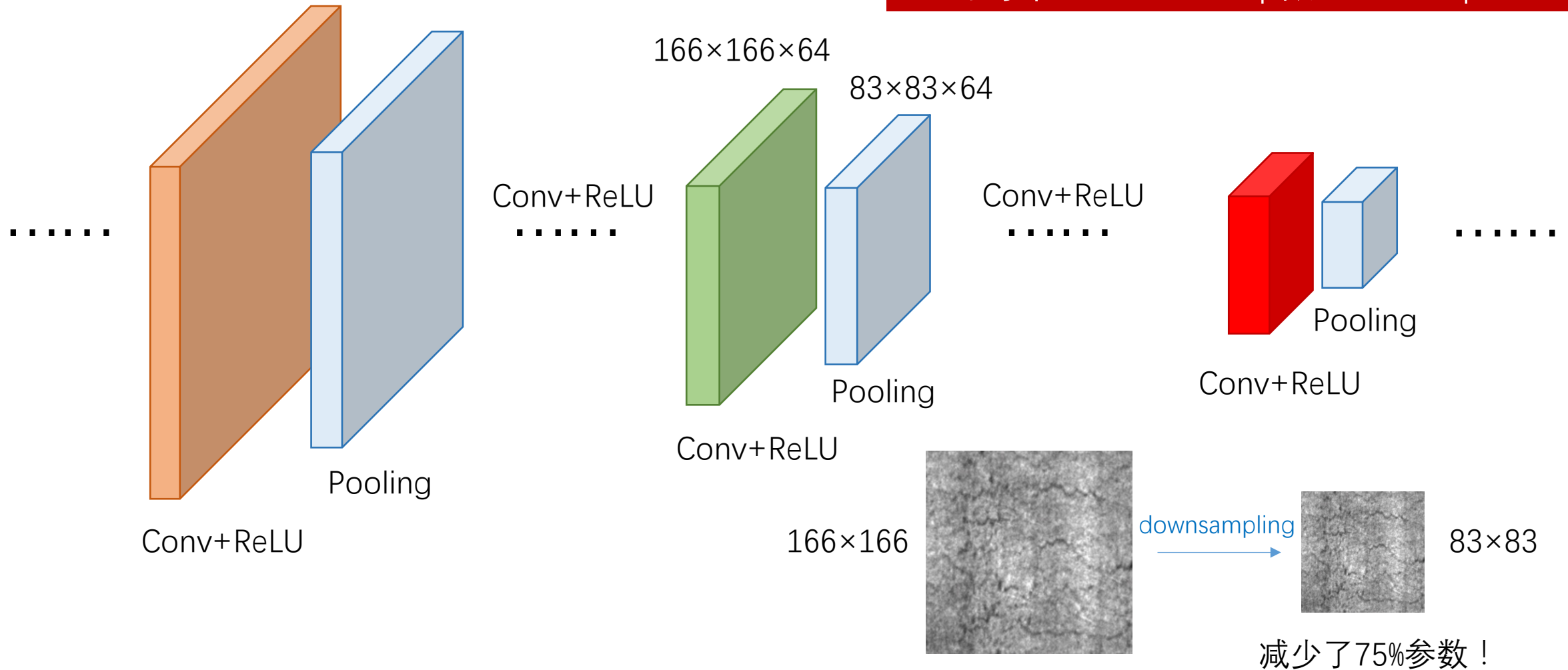
$F=5, S=2, P=\text{满足整除}$

$F=1, S=1, P=0$

- 引入 $K$ 个 $F \times F \times D_1$  filter，共计 $F \times F \times D_1 \times K$ 个权重参数和 $K$ 个偏置项
- 生成大小为 $W_2 \times H_2 \times D_2$ 的activation maps：
  - ✓  $W_2 = \frac{W_1 - F + 2P}{S} + 1$
  - ✓  $H_2 = \frac{H_1 - F + 2P}{S} + 1$
  - ✓  $D_2 = K$
- 第 $d$ 个activation map是第 $d$ 个filter与输入层做卷积的结果（每个卷积都加上第 $d$ 个偏置项）

# 池化层

- ✓ 每隔几个Conv+ReLU层增加一层Pooling
- ✓ 对每个activation map做downsample



# 最大池化 (Max Pooling)

单个activation map

3	0	0	4
0	2	5	1
3	8	2	4
1	0	1	0

# 最大池化 (Max Pooling)

单个activation map

3	0	0	4
0	2	5	1
3	8	2	4
1	0	1	0

2×2的filter

stride=2

3	5
8	4

- ✓ 也可以做sum或者mean等
- ✓ 池化区域可以不重叠，也可以重叠

进一步减少参数数量

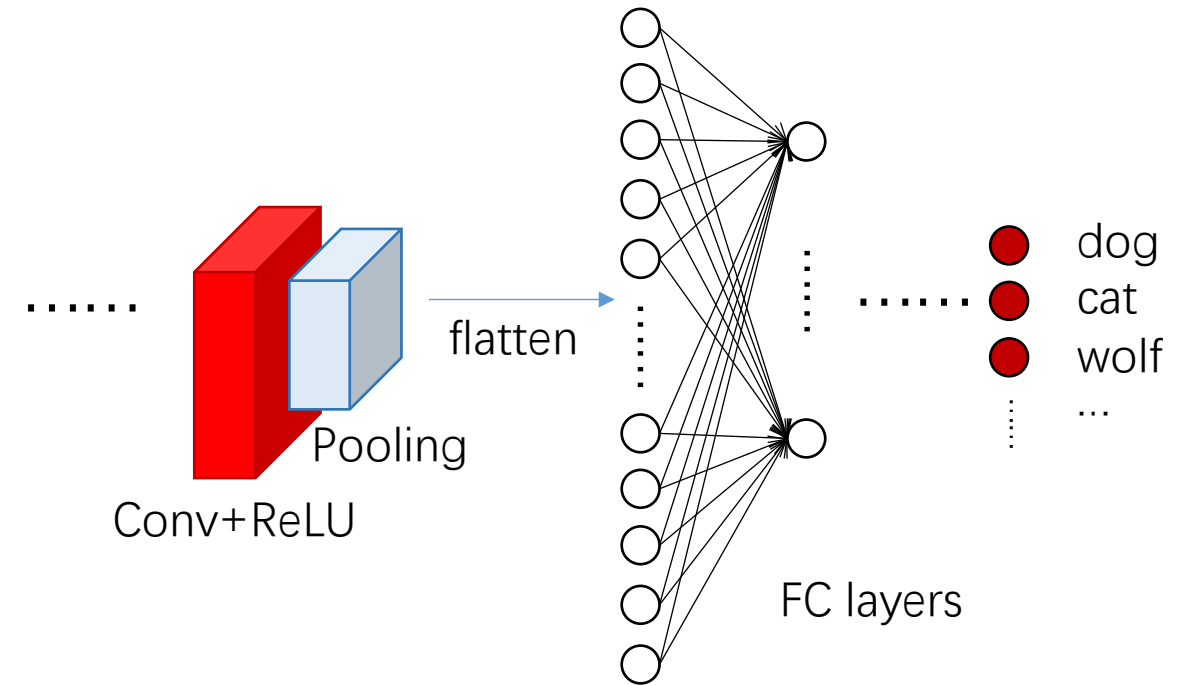
# 池化层小结

- Conv+ReLU层输出： $W_1 \times H_1 \times D_1$
- 设置2个超参数：
  - ✓ filter大小 $F$
  - ✓ stride大小 $S$
- 生成大小为 $W_2 \times H_2 \times D_2$ 的feature maps：
  - ✓  $W_2 = \frac{W_1 - F}{S} + 1$
  - ✓  $H_2 = \frac{H_1 - F}{S} + 1$
  - ✓  $D_2 = D_1$

✓ 只需要一个没有参数的filter  
✓ 一般不做padding

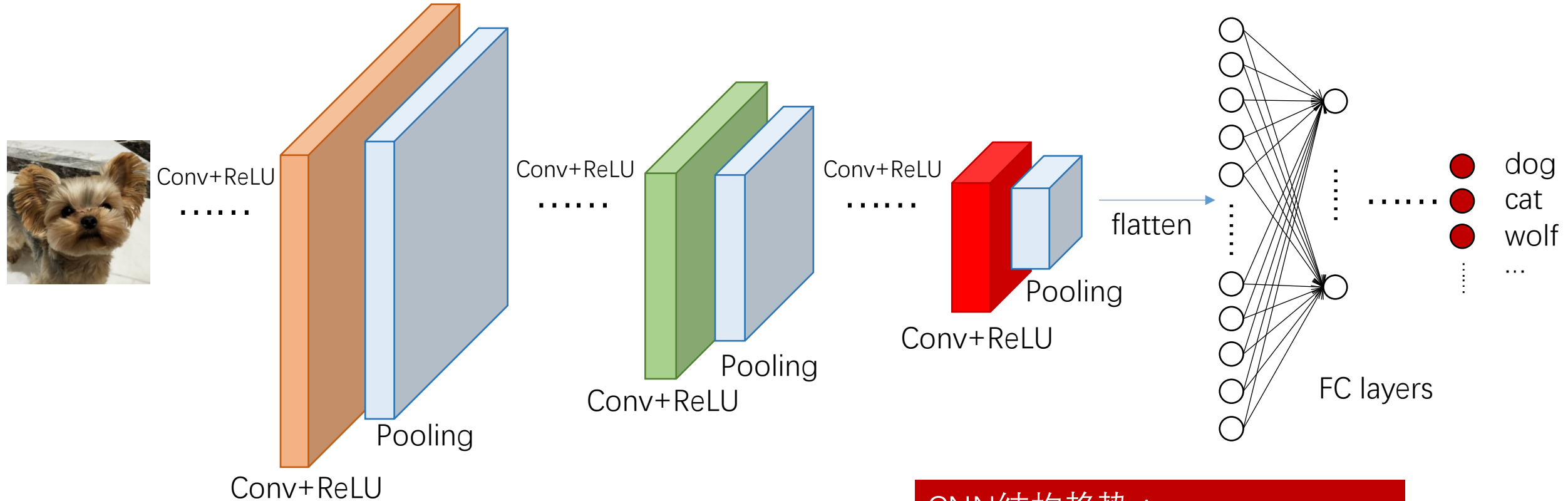
常用设置：  
F=2, S=2  
F=3, S=2

# FC层





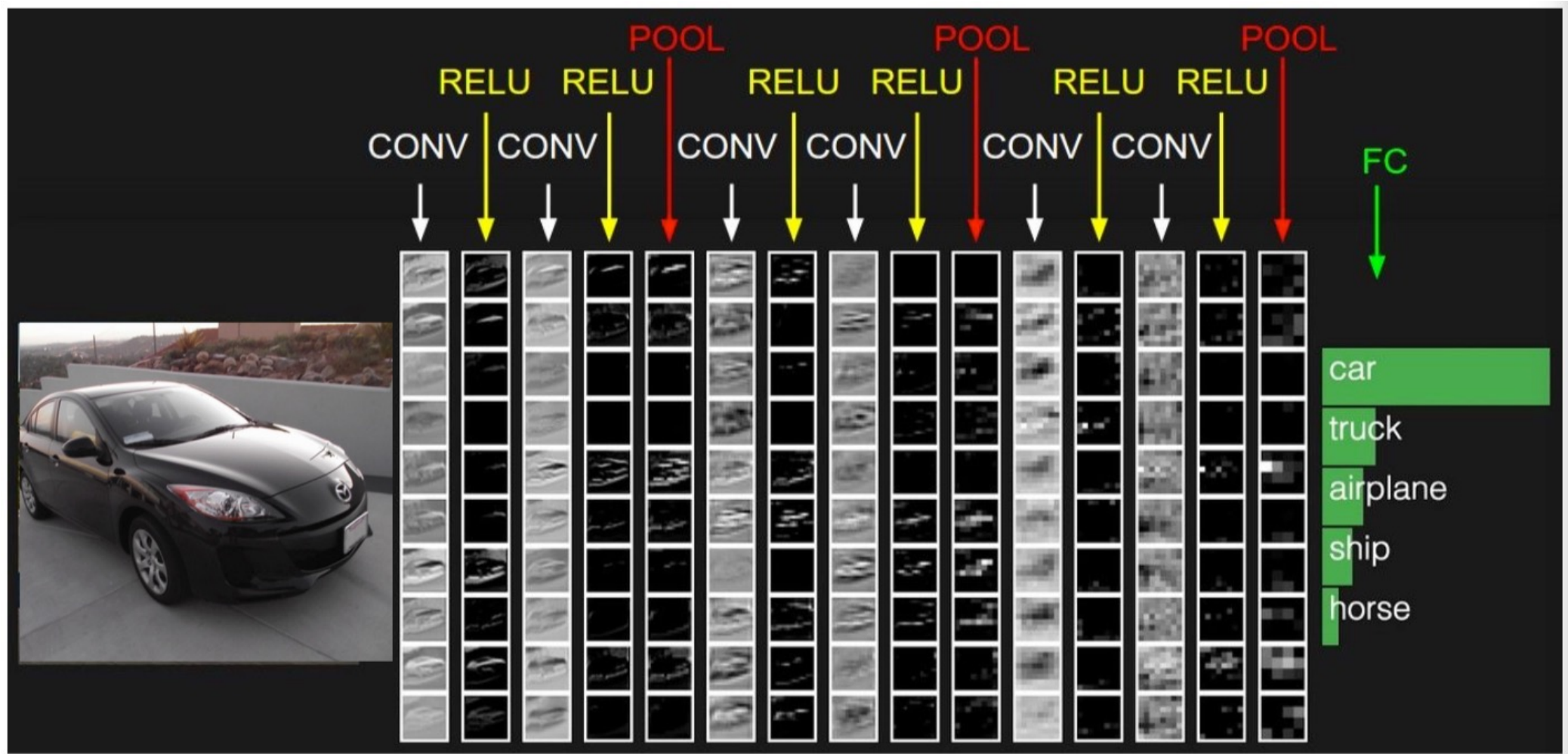
# 完整的CNN



CNN结构趋势：

- ✓ 使用较小的filter，加深网络
- ✓ 移除Pooling层和FC层

# 完整的CNN



# 反向传播：池化层（Max pooling）

forward pass阶段记  
住input最大的位置

3	0	0	4
0	2	5	1
3	8	2	4
1	0	1	0

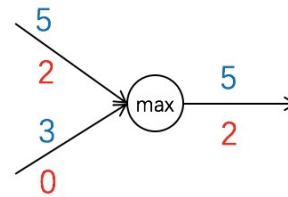
将上游梯度回传到  
input最大的位置



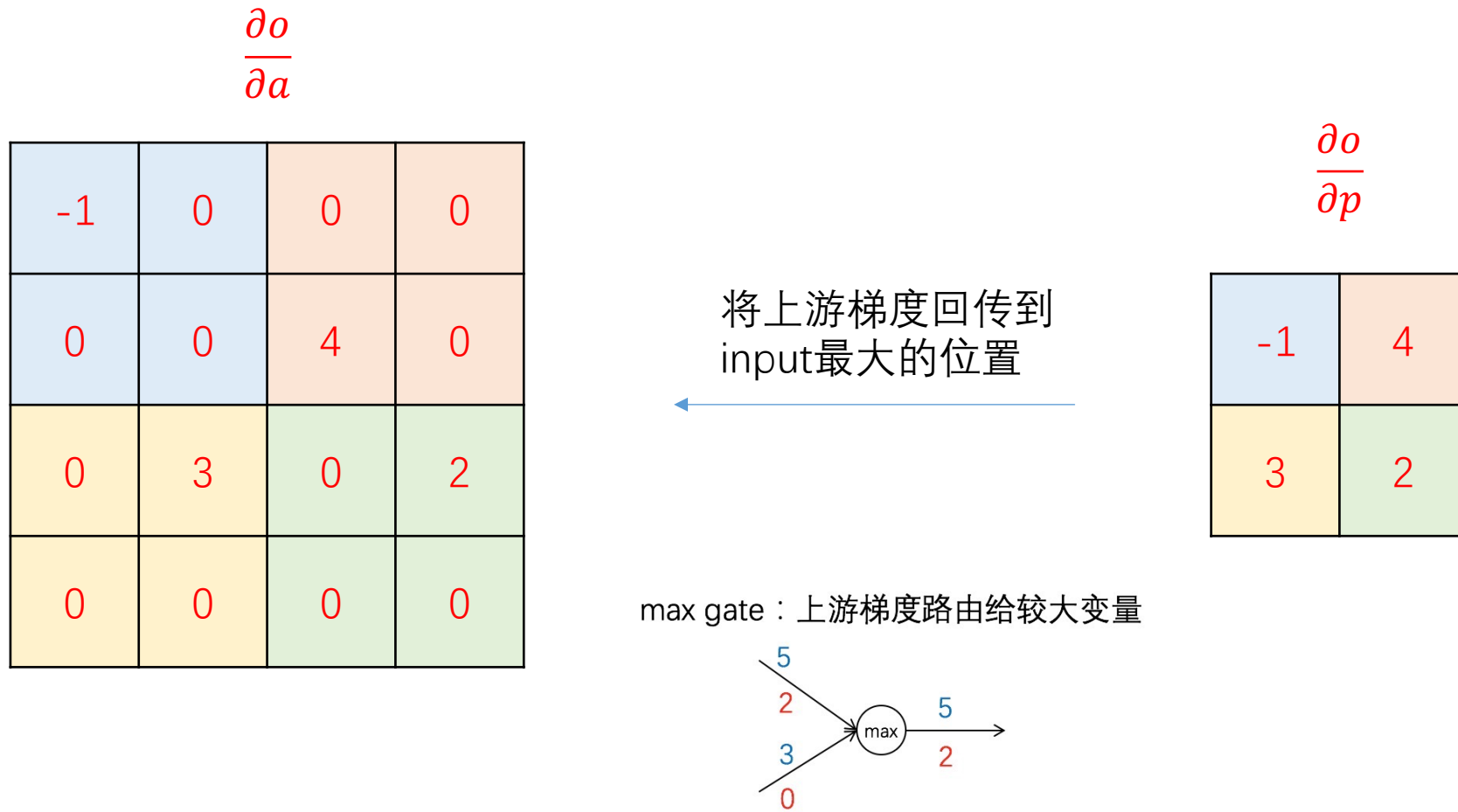
$$\frac{\partial o}{\partial p}$$

-1	4
3	2

max gate：上游梯度路由给较大变量



# 反向传播：池化层（Max pooling）



# 反向传播：激活层 (ReLU)

$$y = W^T X$$

3	-2	-2	4
-3	2	5	1
3	8	2	4
1	-4	1	-2

- ✓ input>0, 回传梯度
- ✓ 否则, 回传0



$$\frac{\partial o}{\partial a}$$

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

# 反向传播：激活层 (ReLU)

$$\frac{\partial o}{\partial y}$$

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

- ✓ input > 0, 回传梯度
- ✓ 否则, 回传0

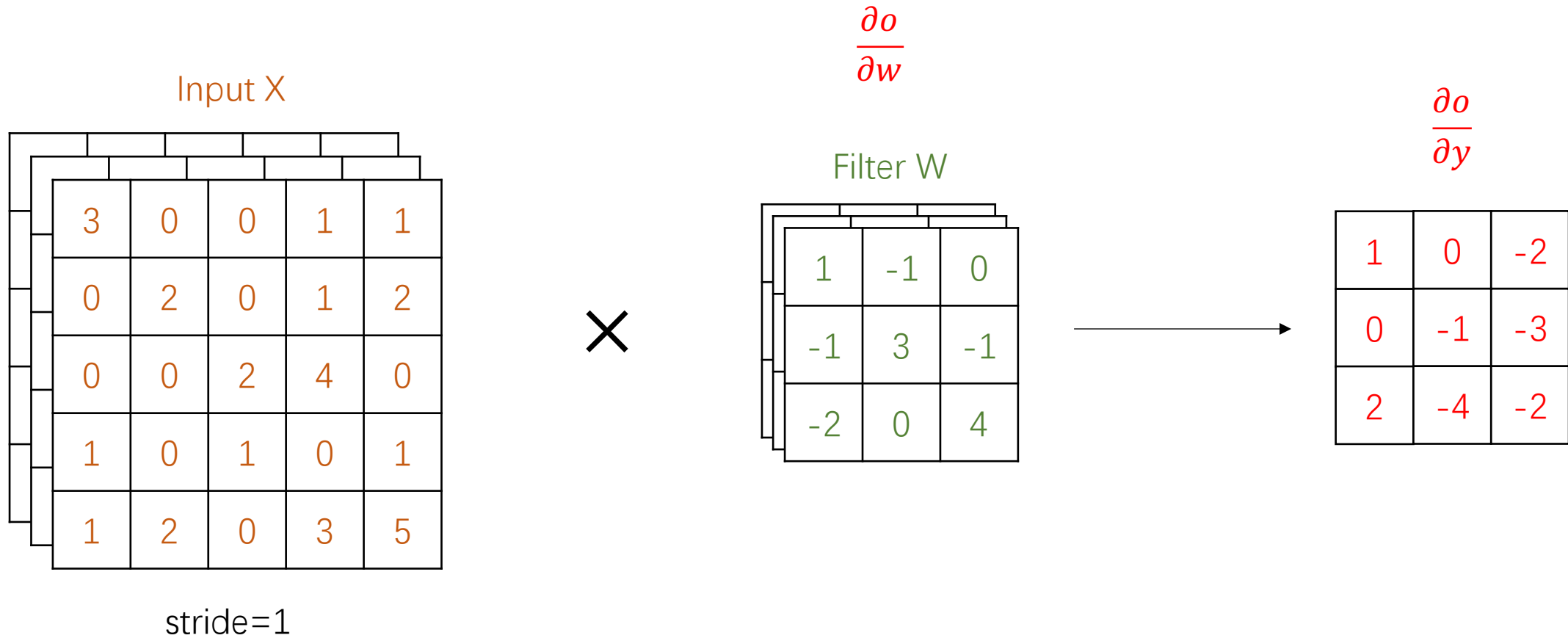


$$\frac{\partial o}{\partial a}$$

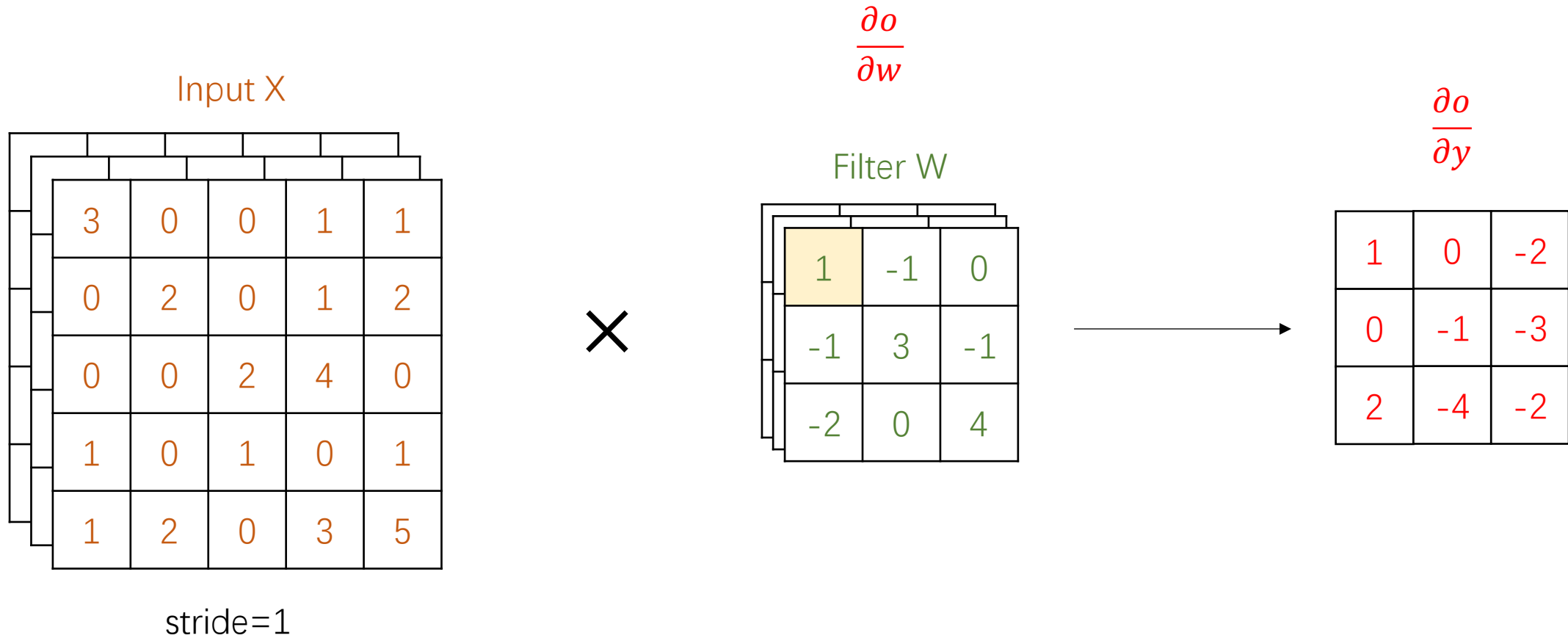
-1	0	0	0
0	0	4	0
0	3	0	2
0	0	0	0

1. 回传梯度矩阵  $\frac{\partial o}{\partial a}$
2. Input ≤ 0 处置0

# 反向传播：卷积层（不含激活）

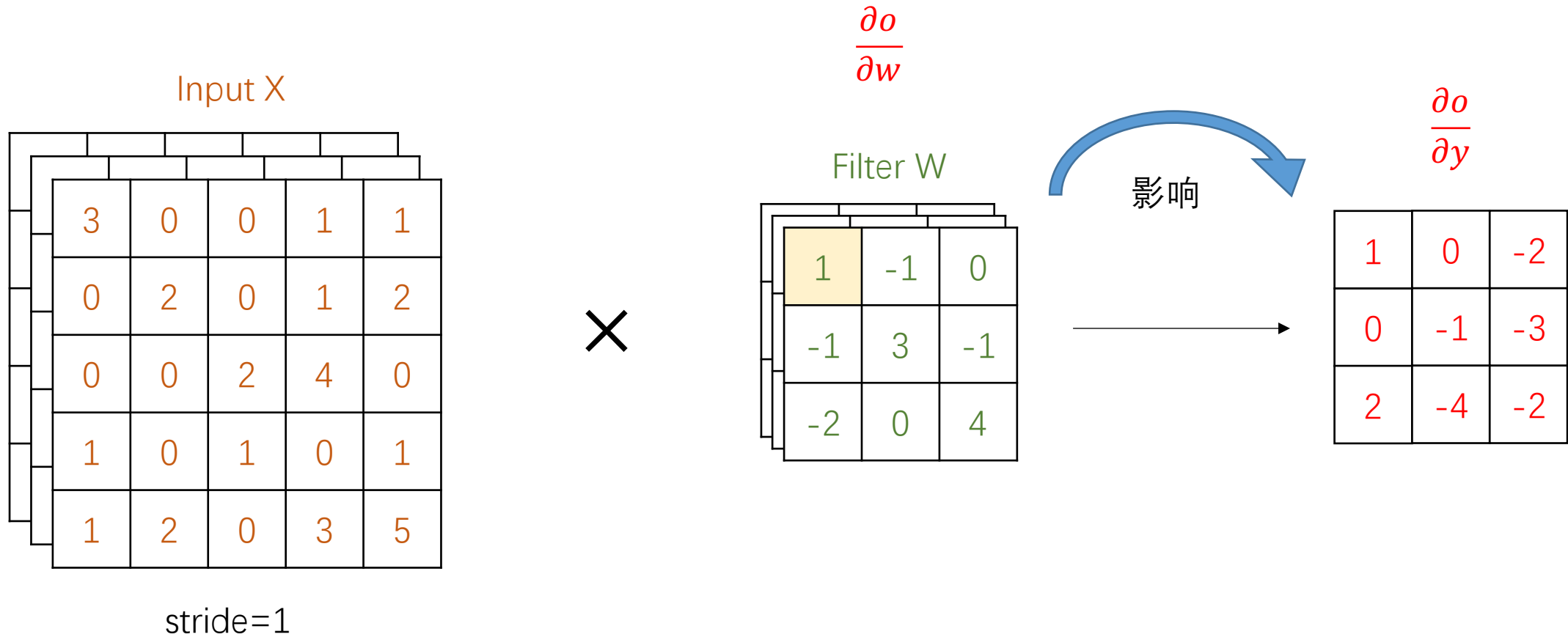


# 反向传播：卷积层（不含激活）

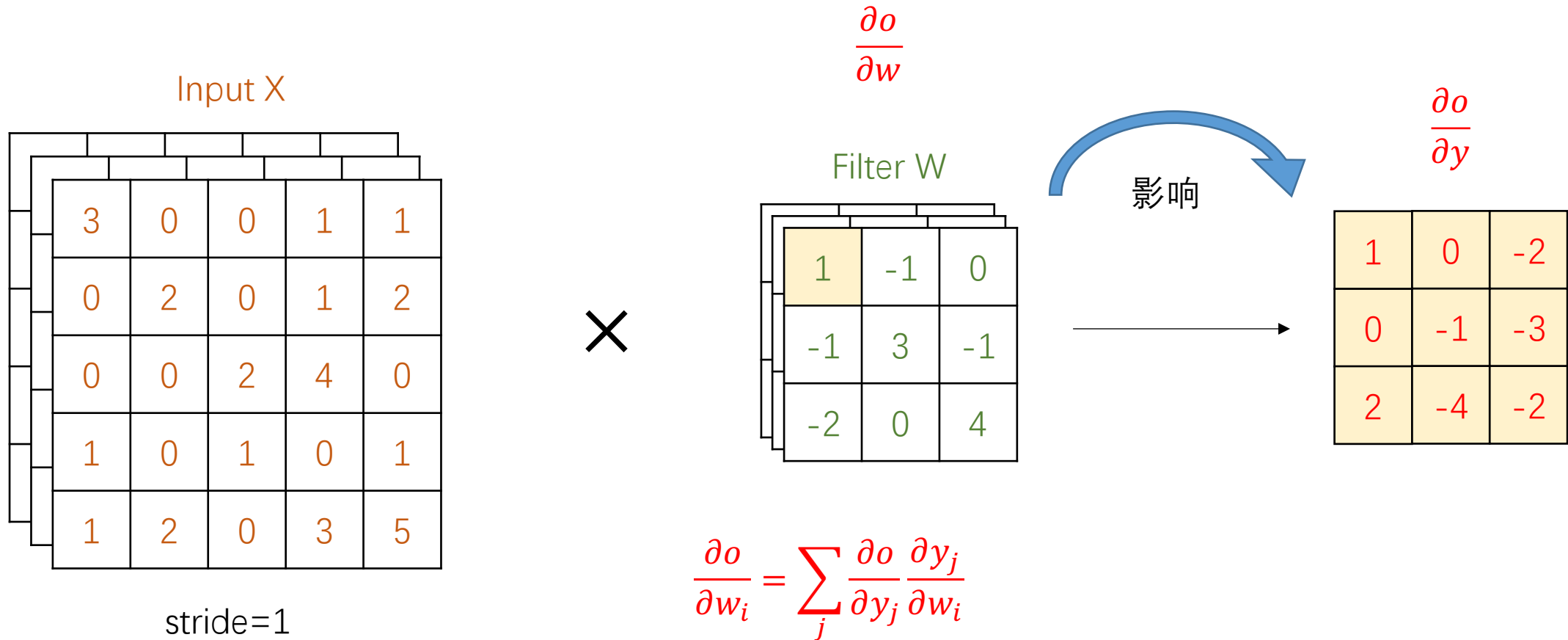




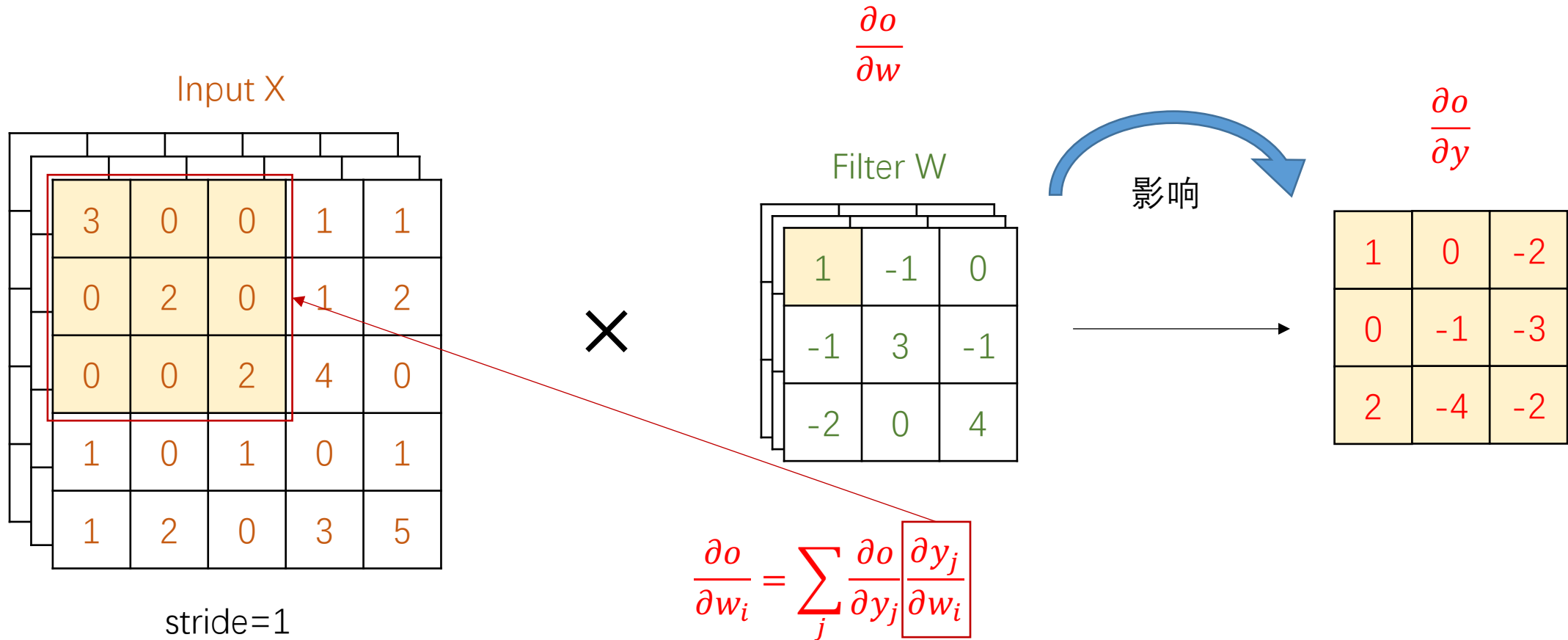
# 反向传播：卷积层（不含激活）



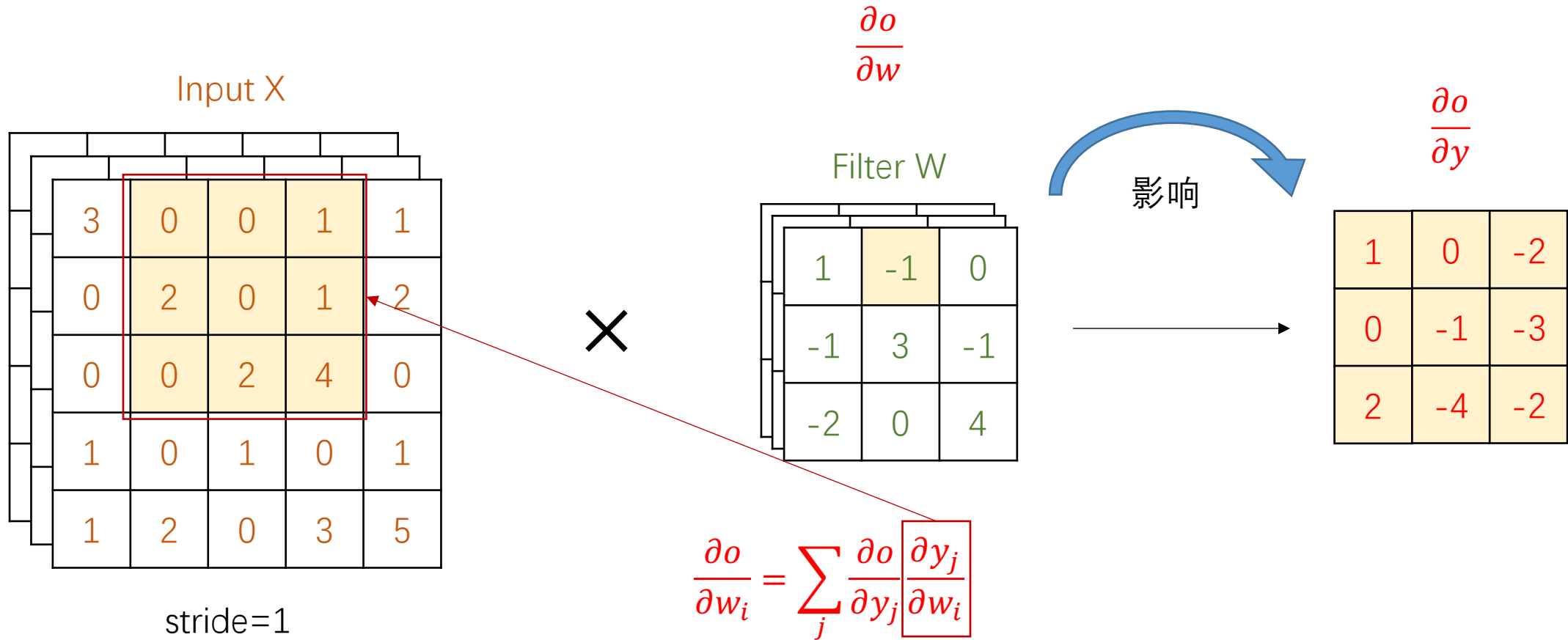
# 反向传播：卷积层（不含激活）



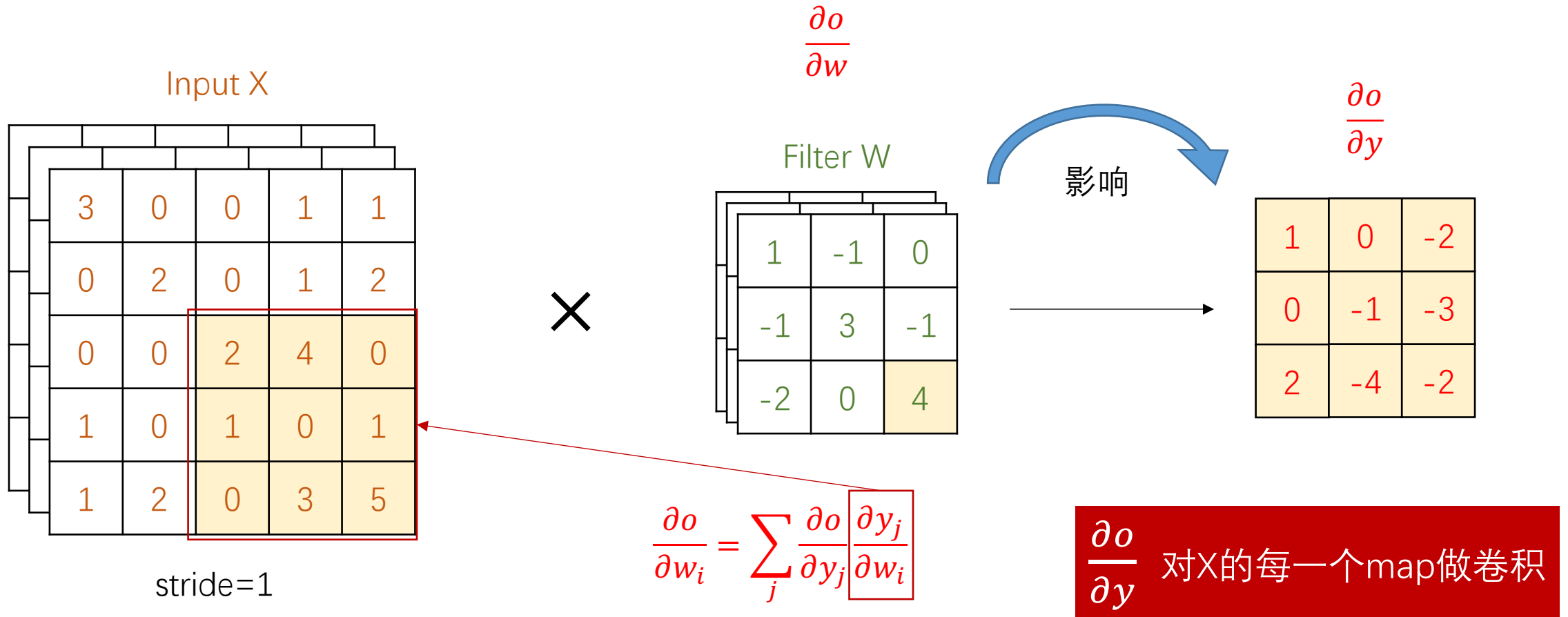
# 反向传播：卷积层（不含激活）



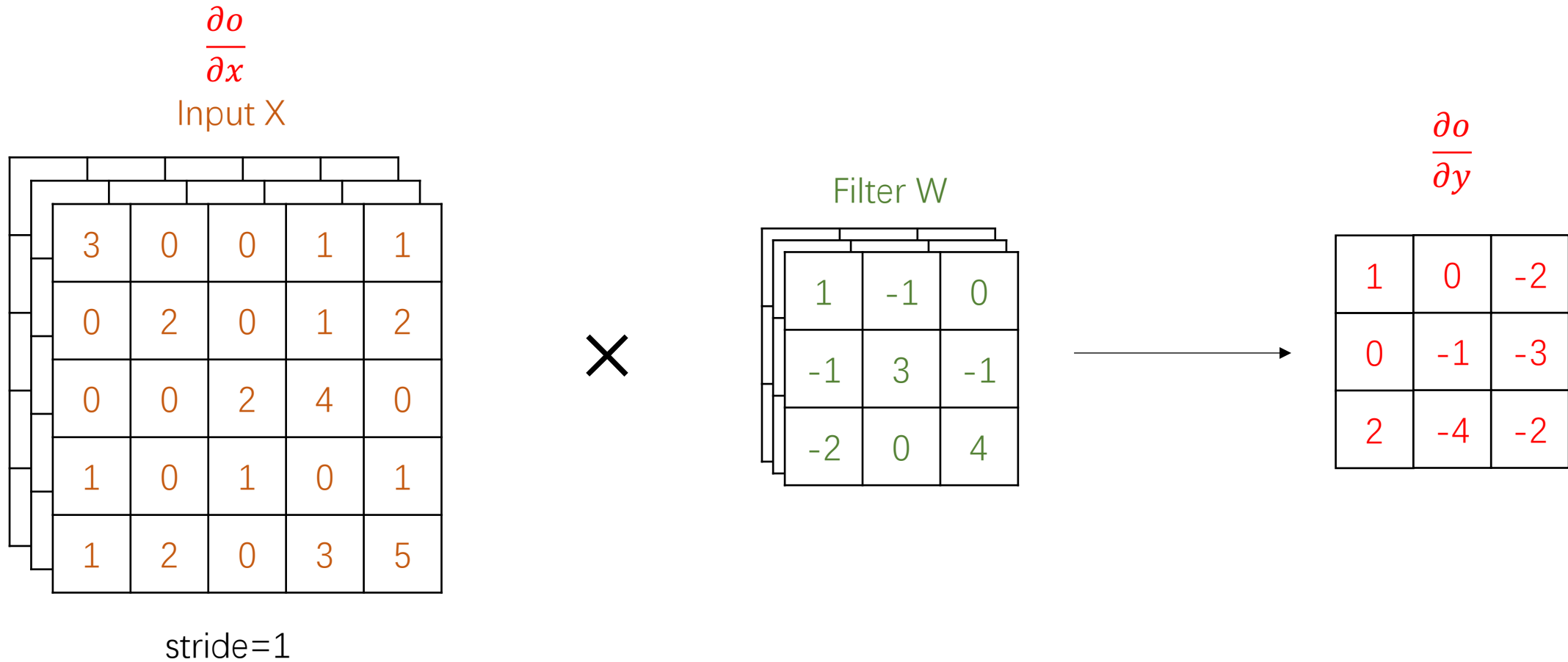
# 反向传播：卷积层（不含激活）



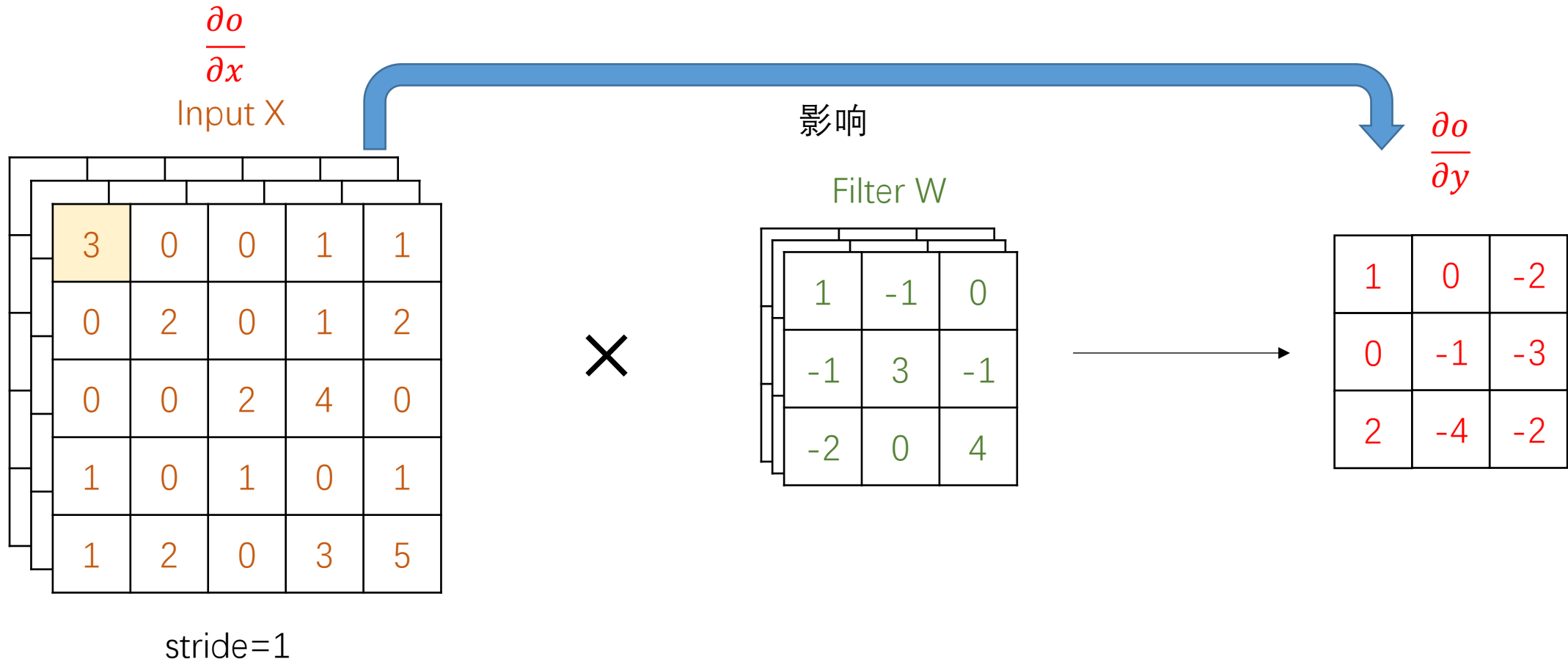
# 反向传播：卷积层（不含激活）



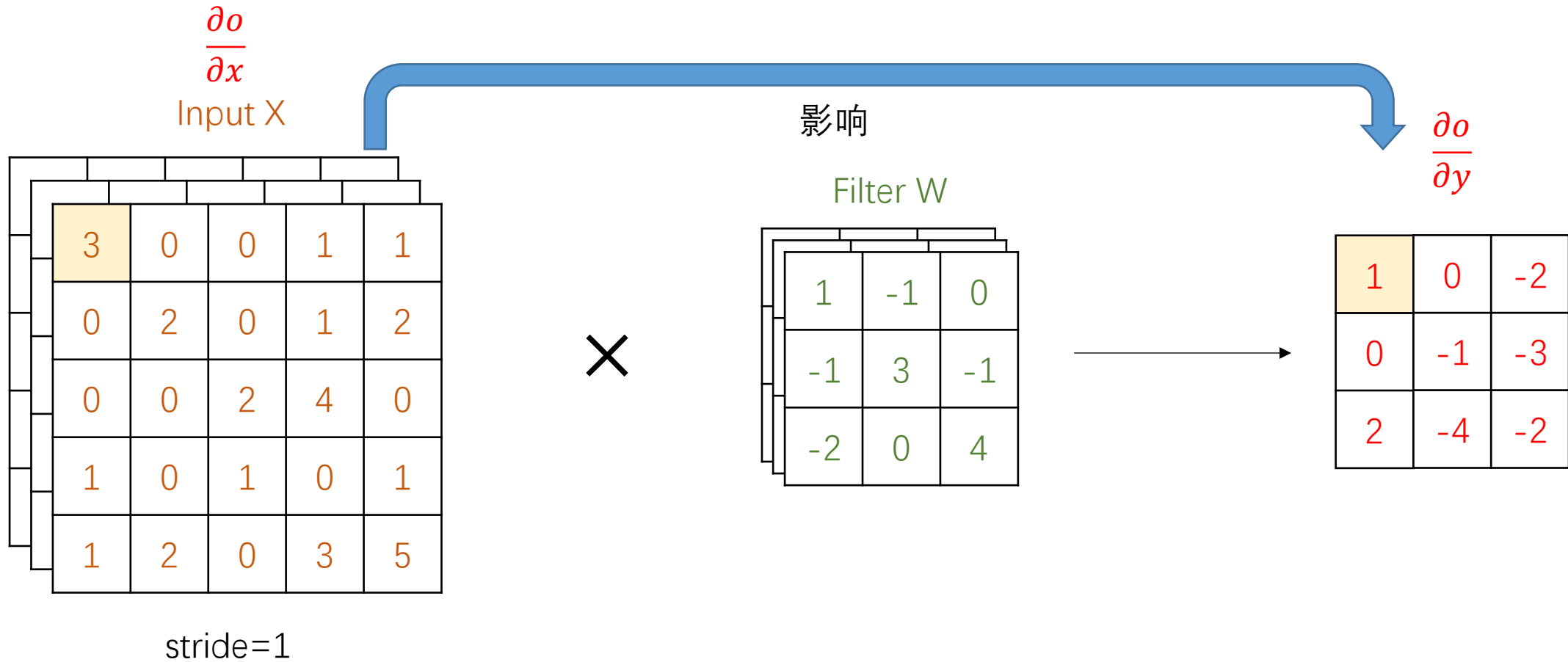
# 反向传播：卷积层（不含激活）



# 反向传播：卷积层（不含激活）

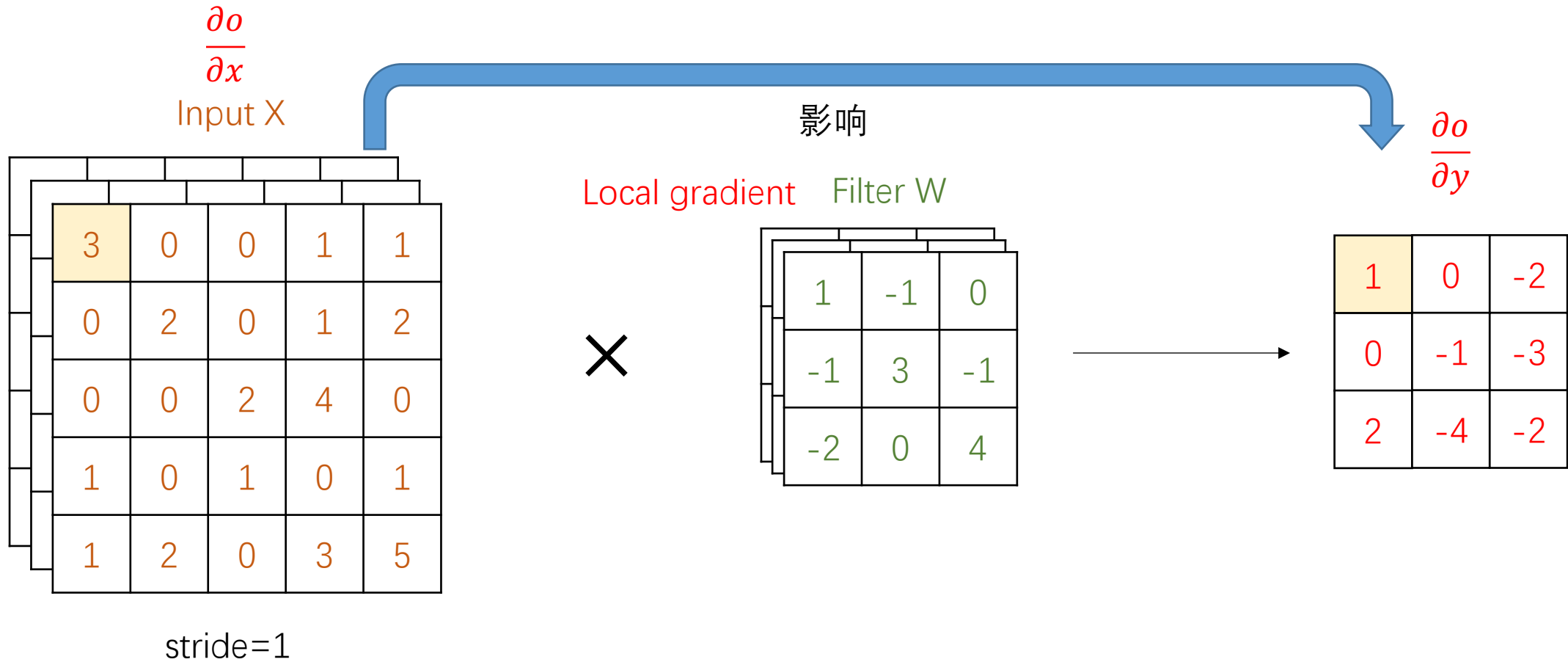


# 反向传播：卷积层（不含激活）

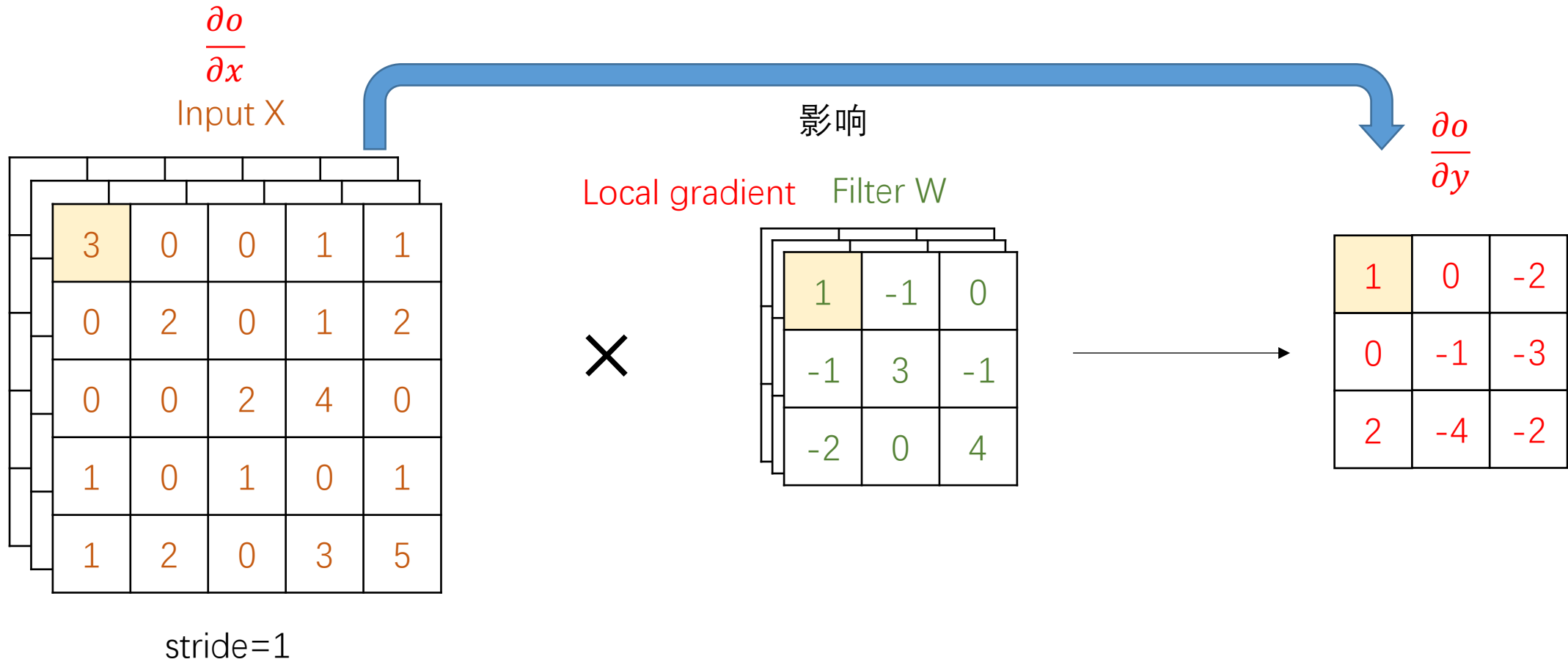




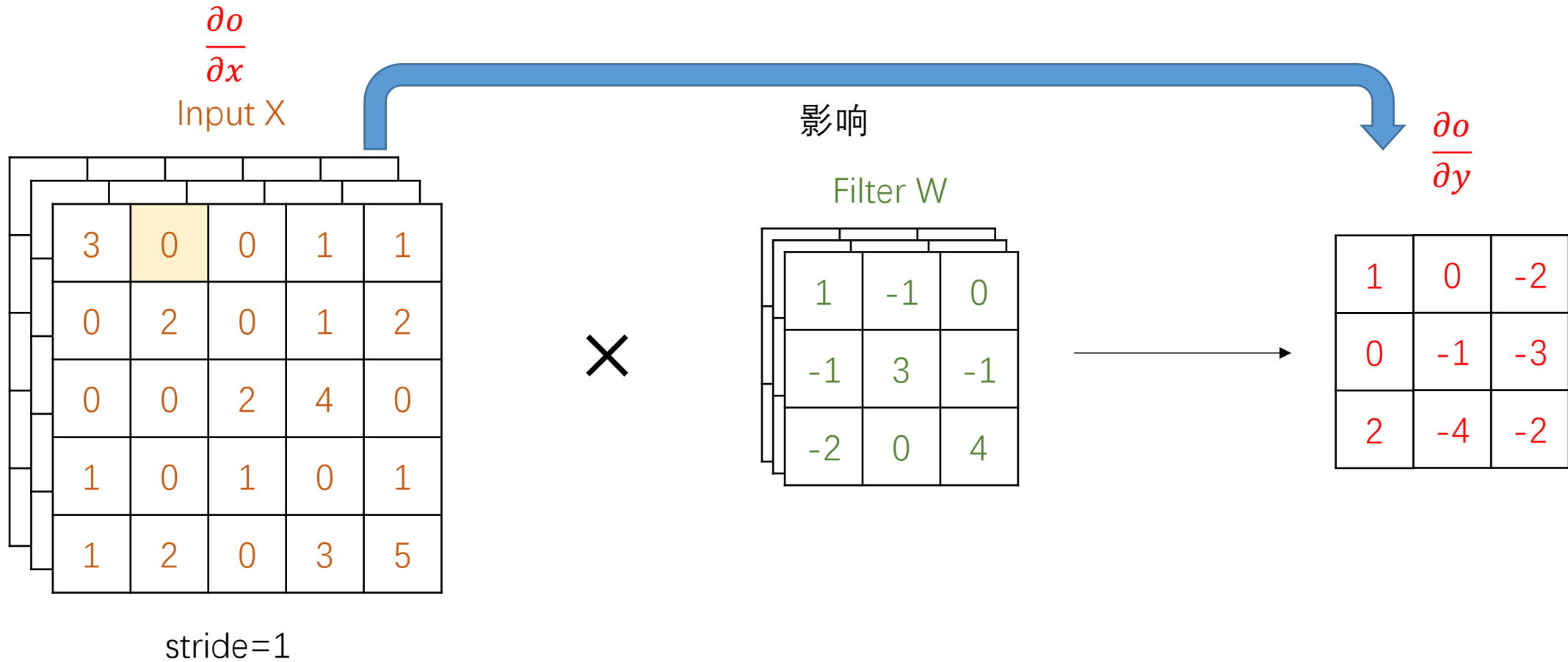
# 反向传播：卷积层（不含激活）



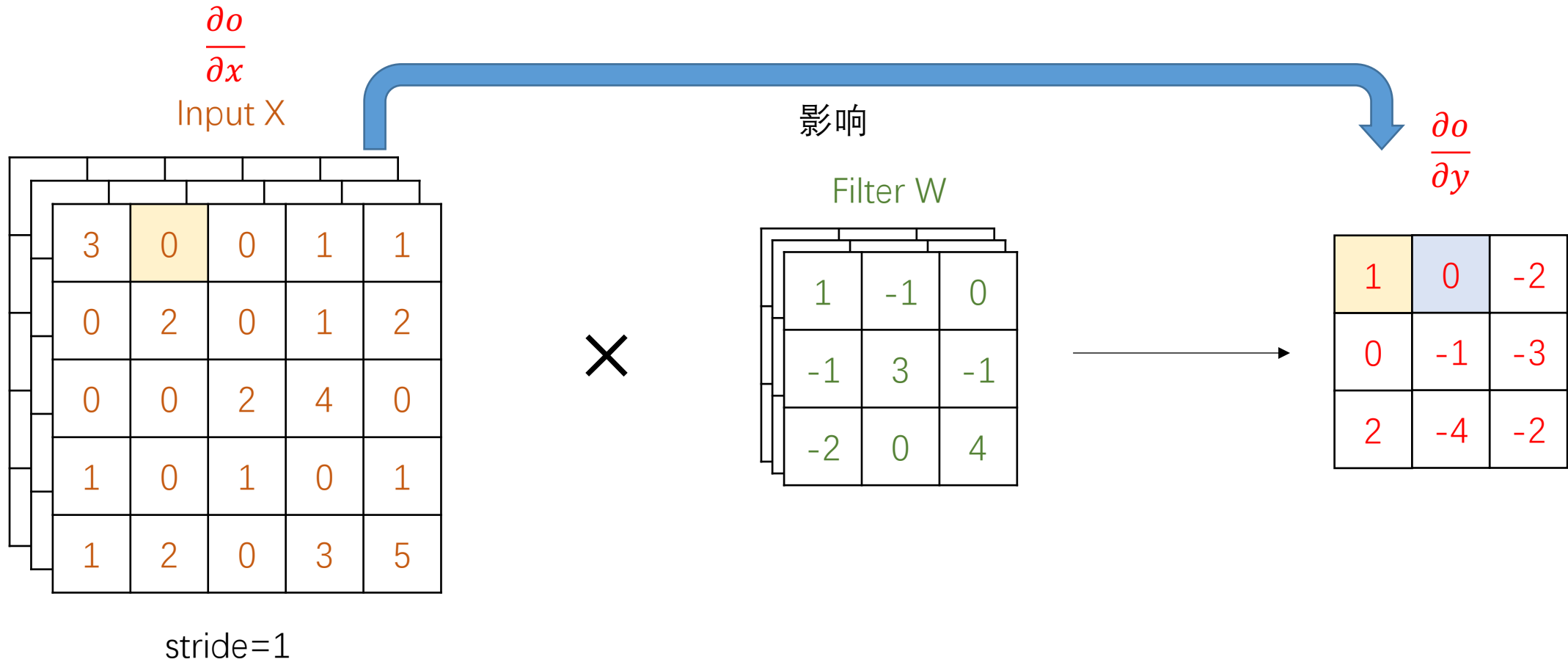
# 反向传播：卷积层（不含激活）



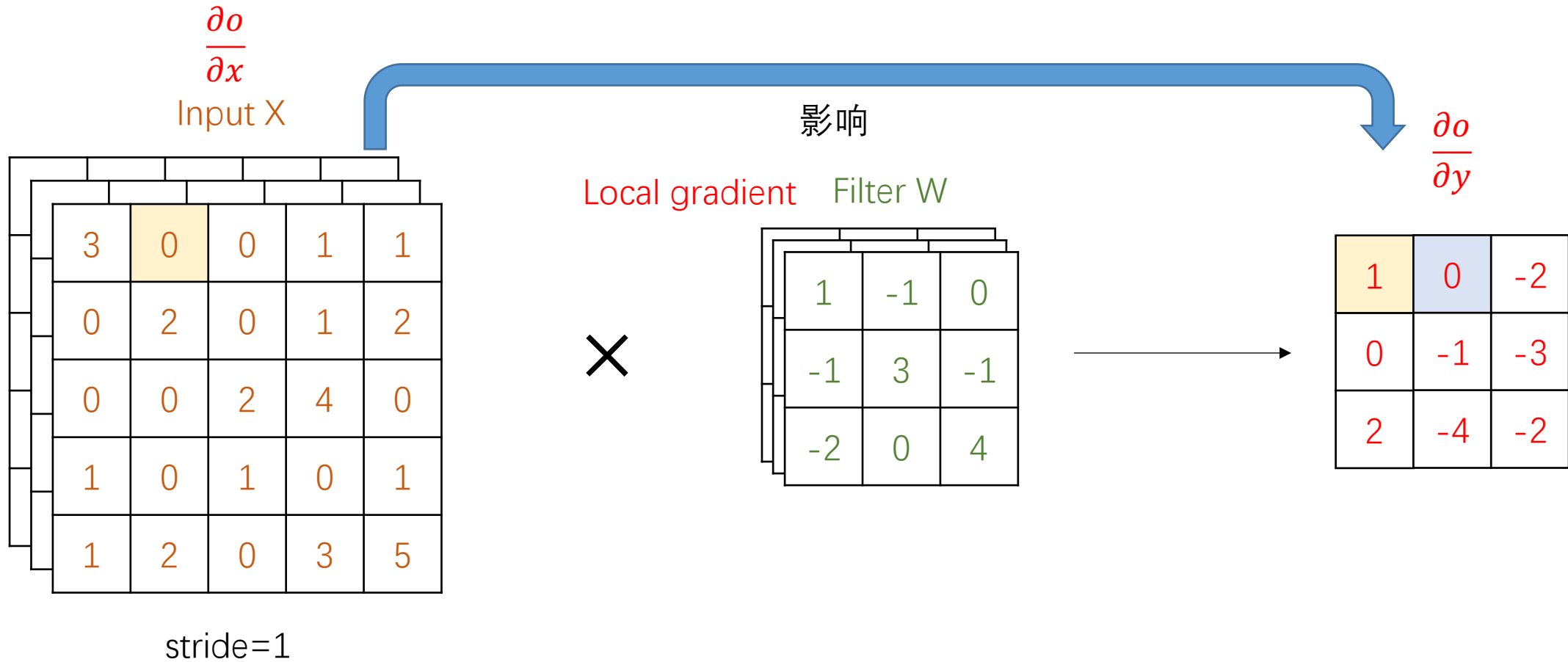
# 反向传播：卷积层（不含激活）



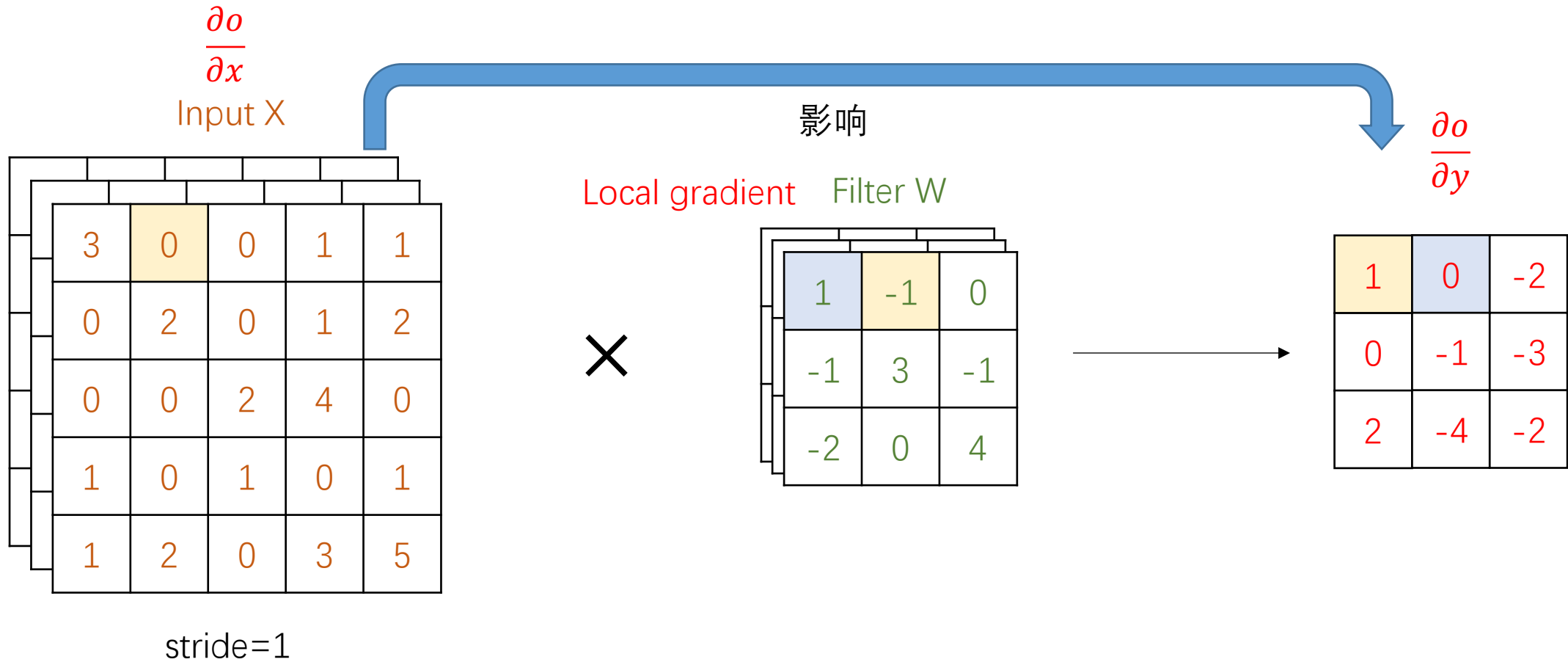
# 反向传播：卷积层（不含激活）



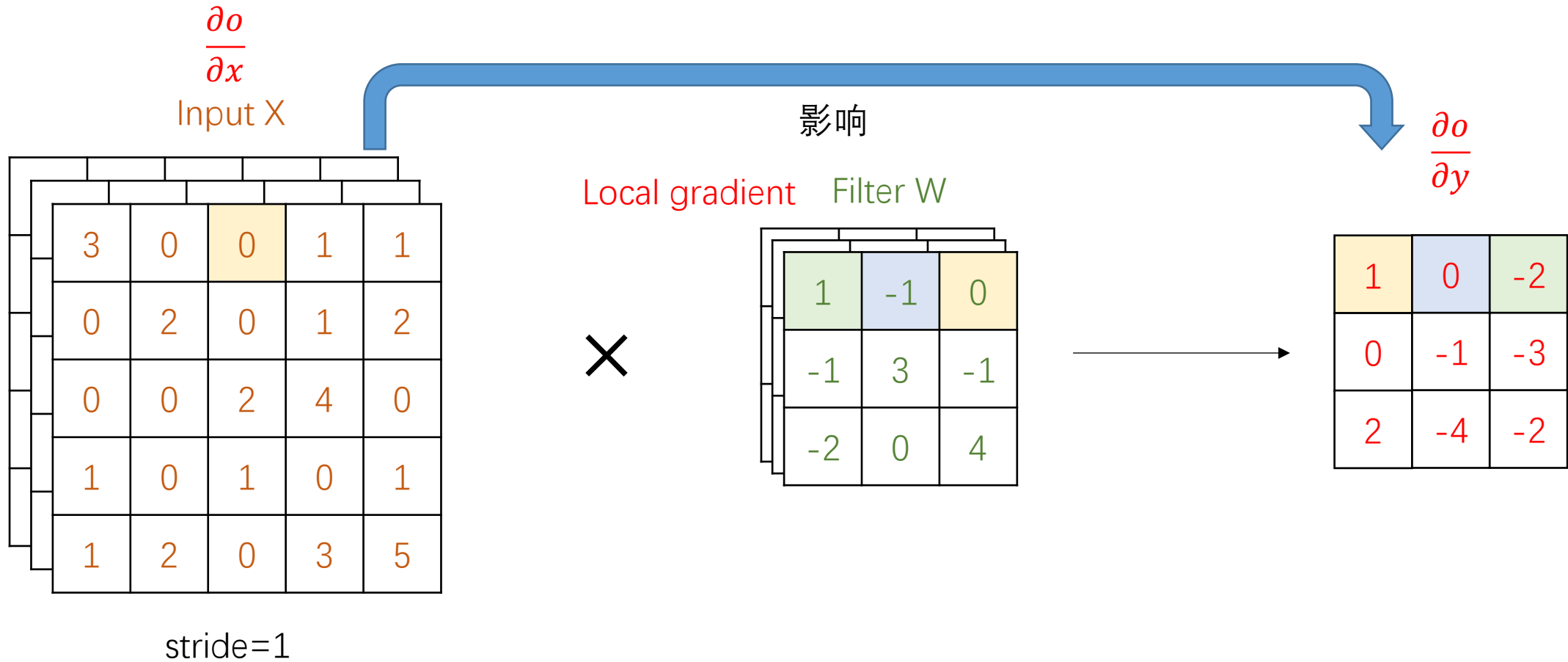
# 反向传播：卷积层（不含激活）



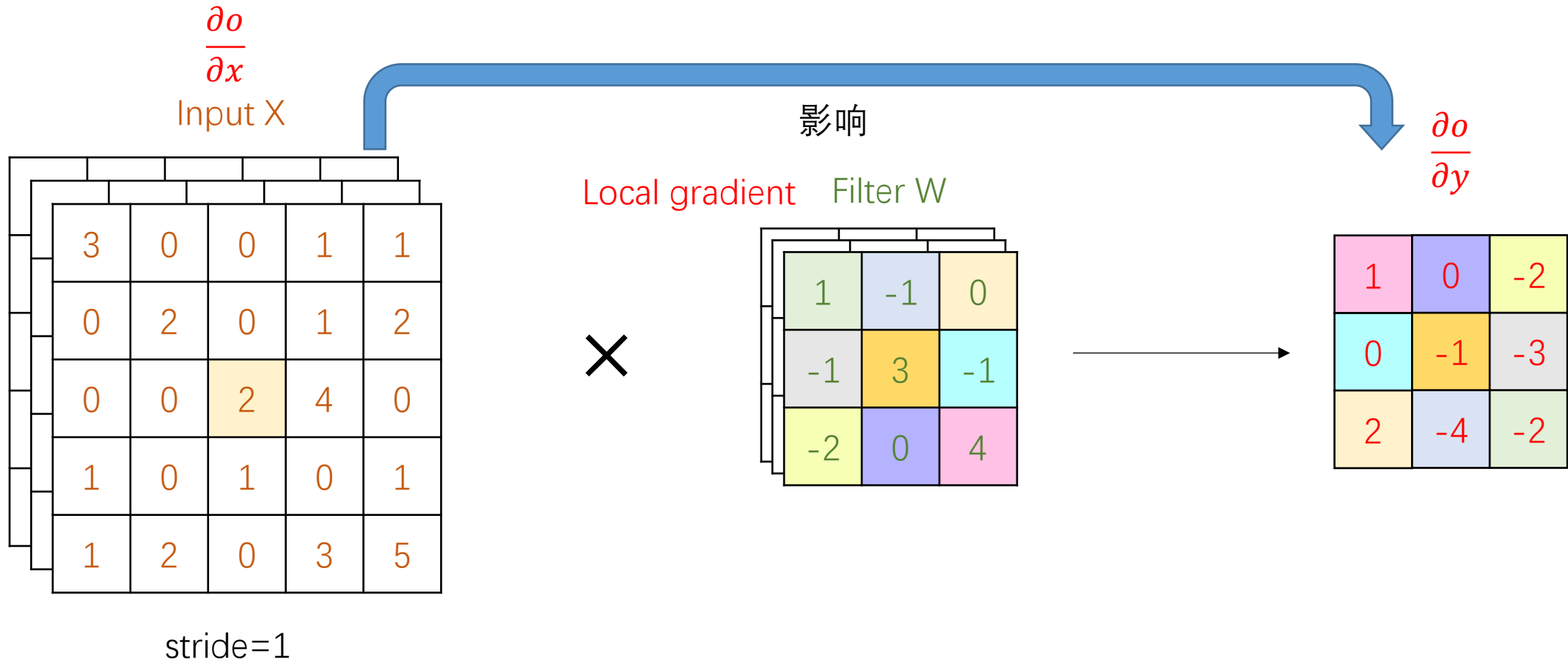
# 反向传播：卷积层（不含激活）



# 反向传播：卷积层（不含激活）

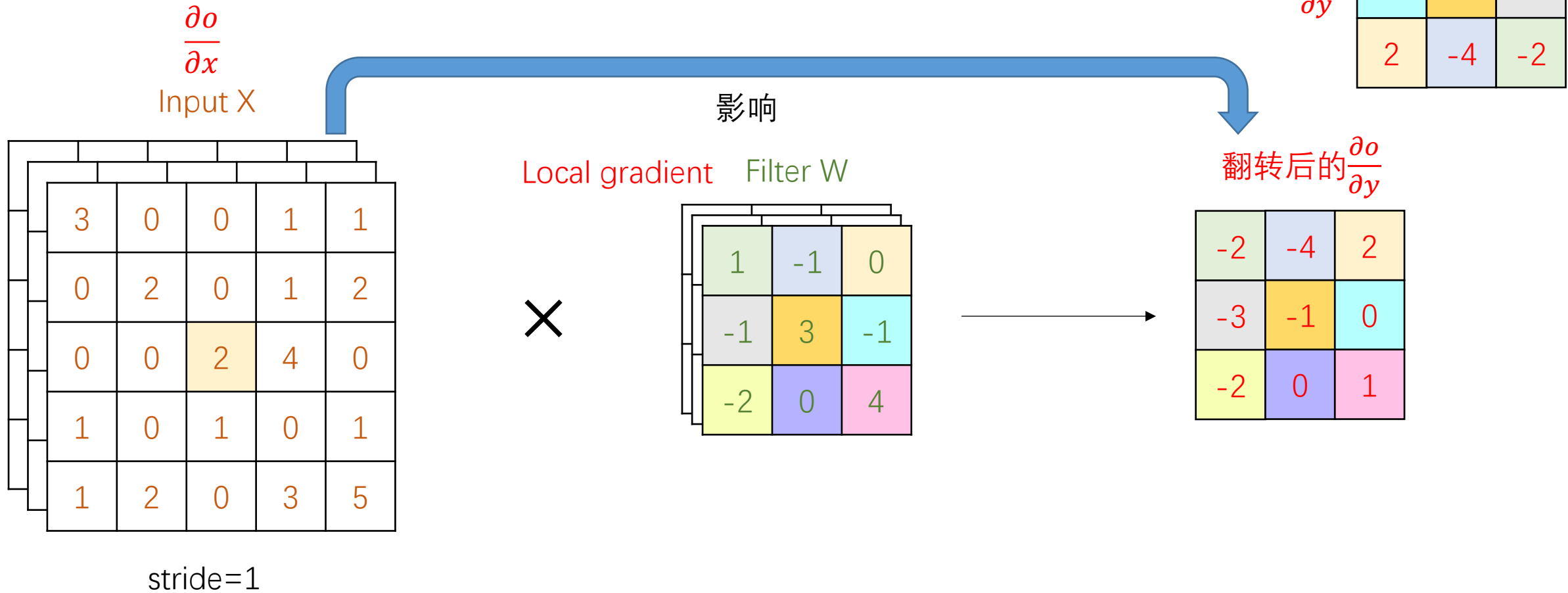


# 反向传播：卷积层（不含激活）

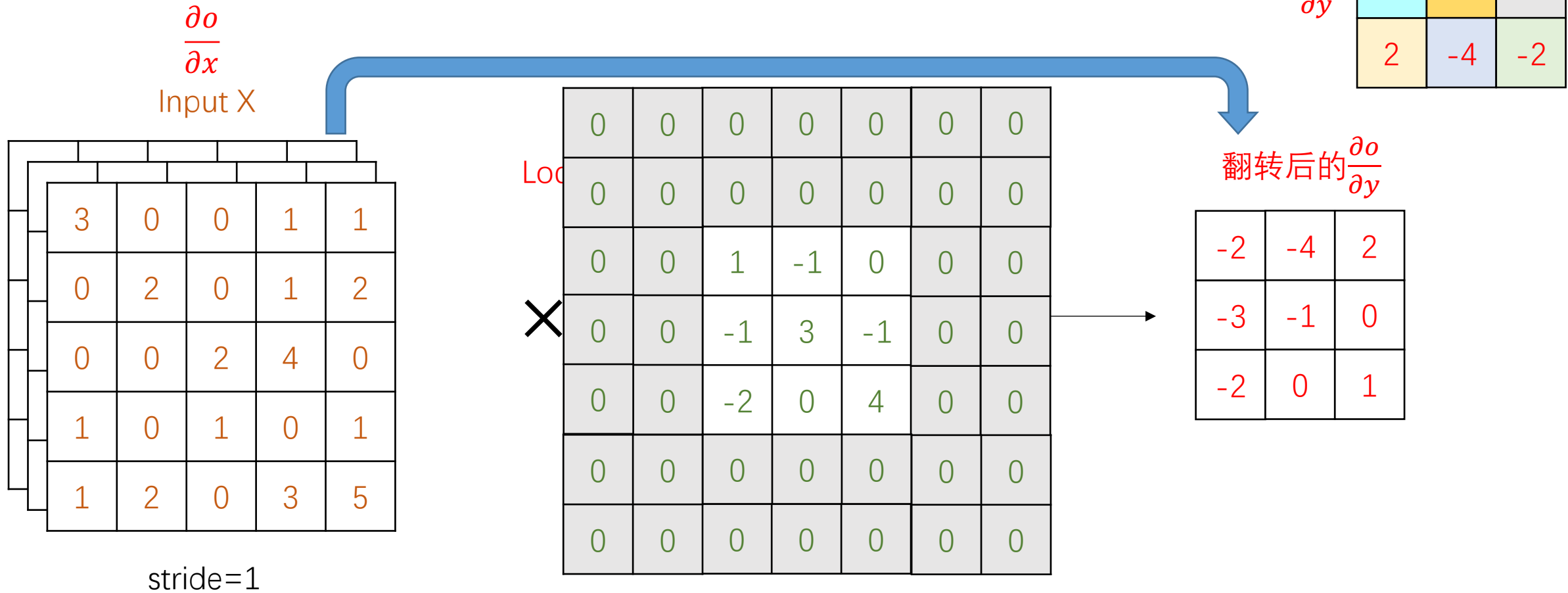




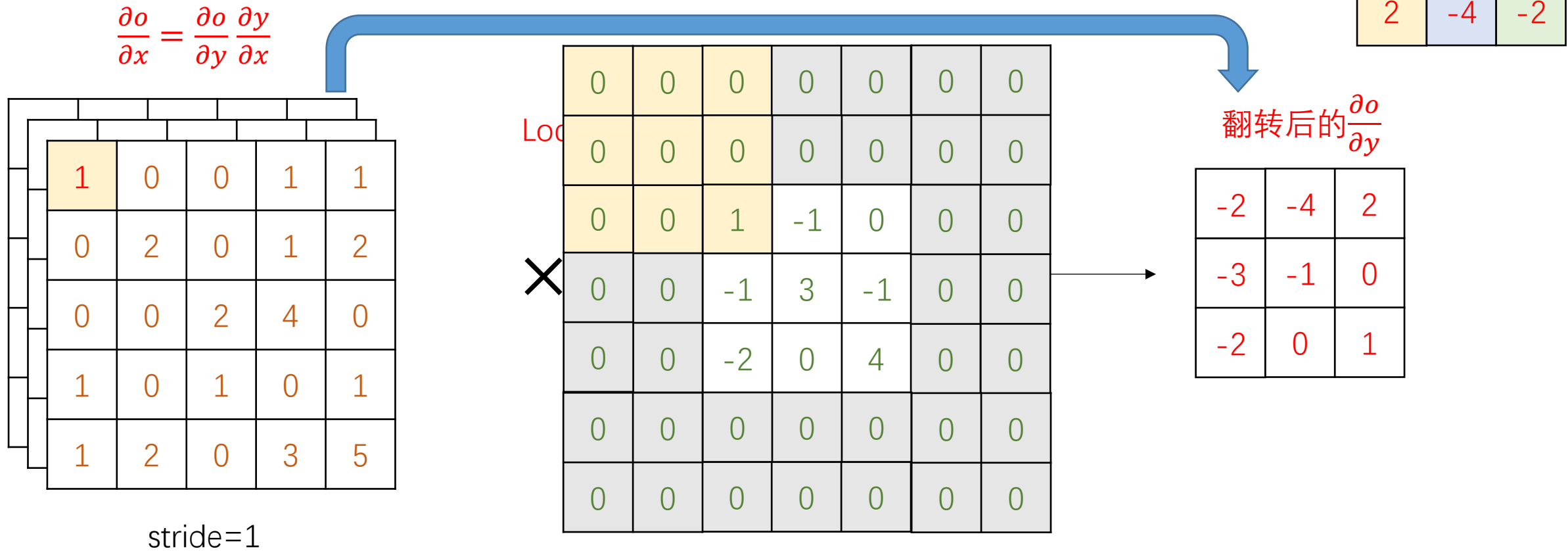
# 反向传播：卷积层（不含激活）



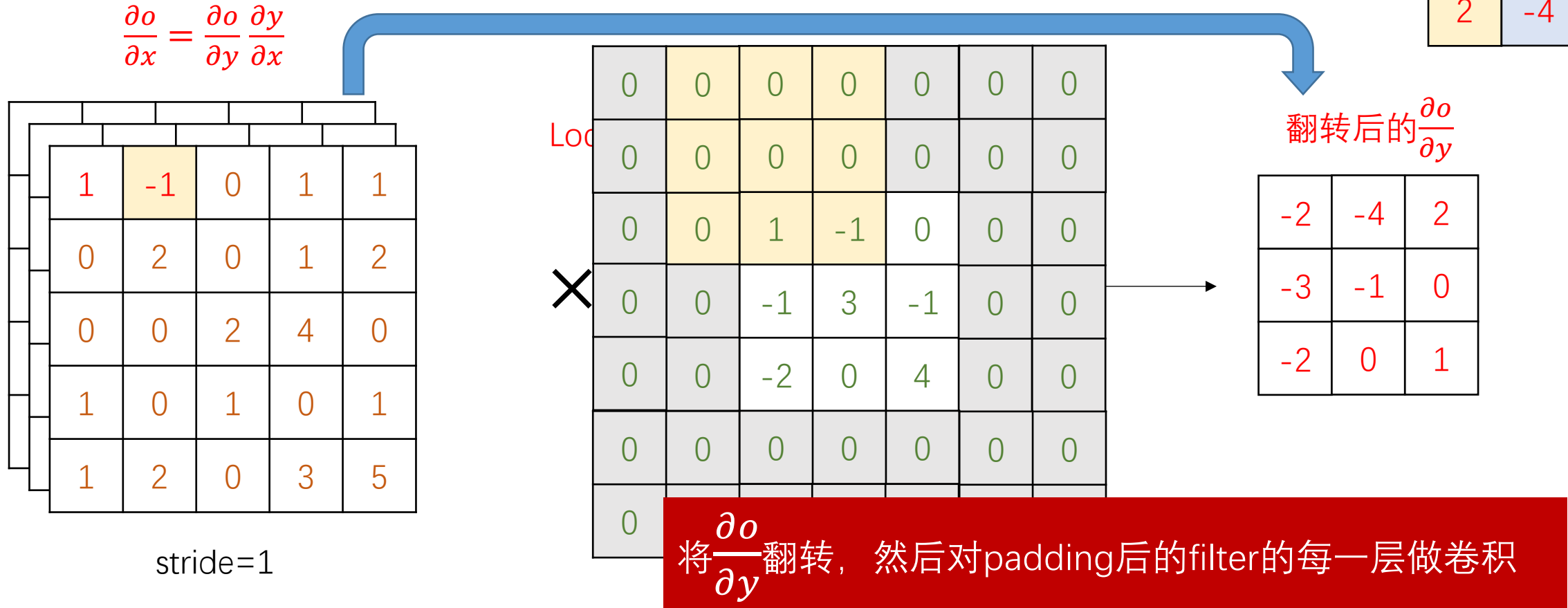
# 反向传播：卷积层（不含激活）



# 反向传播：卷积层（不含激活）



# 反向传播：卷积层（不含激活）



# Assignment 2

```
def conv_forward_naive(x, w, b, conv_param):
    """
    A naive implementation of the forward pass for a convolutional layer.

    The input consists of N data points, each with C channels, height H and
    width W. We convolve each input with F different filters, where each filter
    spans all C channels and has height HH and width WW.

    Input:
    - x: Input data of shape (N, C, H, W)
    - w: Filter weights of shape (F, C, HH, WW)
    - b: Biases, of shape (F,)
    - conv_param: A dictionary with the following keys:
        - 'stride': The number of pixels between adjacent receptive fields in the
            horizontal and vertical directions.
        - 'pad': The number of pixels that will be used to zero-pad the input.

    During padding, 'pad' zeros should be placed symmetrically (i.e equally on both sides)
    along the height and width axes of the input. Be careful not to modify the original
    input x directly.

    Returns a tuple of:
    - out: Output data, of shape (N, F, H', W') where H' and W' are given by
        H' = 1 + (H + 2 * pad - HH) / stride
        W' = 1 + (W + 2 * pad - WW) / stride
    - cache: (x, w, b, conv_param)
    """
    out = None
    #####
    # TODO: Implement the convolutional forward pass. #
    # Hint: you can use the function np.pad for padding. #
    #####
    # *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

    pass

    # *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
    #####
    # END OF YOUR CODE #
    #####
    cache = (x, w, b, conv_param)
    return out, cache
```

```
def conv_backward_naive(dout, cache):
    """
    A naive implementation of the backward pass for a convolutional layer.

    Inputs:
    - dout: Upstream derivatives.
    - cache: A tuple of (x, w, b, conv_param) as in conv_forward_naive

    Returns a tuple of:
    - dx: Gradient with respect to x
    - dw: Gradient with respect to w
    - db: Gradient with respect to b
    """
    dx, dw, db = None, None, None
    #####
    # TODO: Implement the convolutional backward pass. #
    #####
    # *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

    pass

    # *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
    #####
    # END OF YOUR CODE #
    #####
    return dx, dw, db
```

# CNN学了什么？

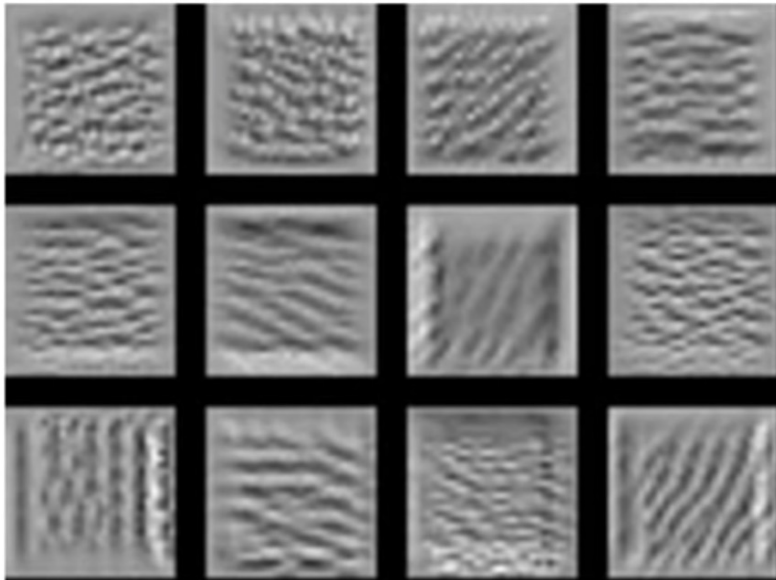
假设每个卷积层activation map  $a^k \in \mathbb{R}^{15 \times 15}$

定义activation degree:  $ad(a^k) \in \sum a_{ij}$

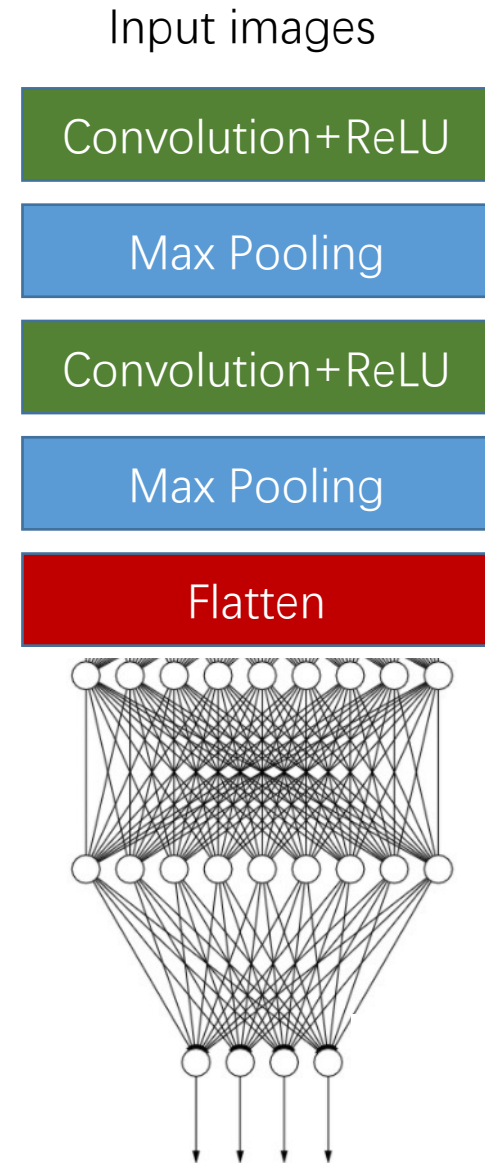
$\forall a^k$ , 寻找:  $x^k = \arg \max_x ad(a^k)$

梯度上升

MNIST  
其中12个 $x^k$



$a^k$   
 $32 \times 15 \times 15$



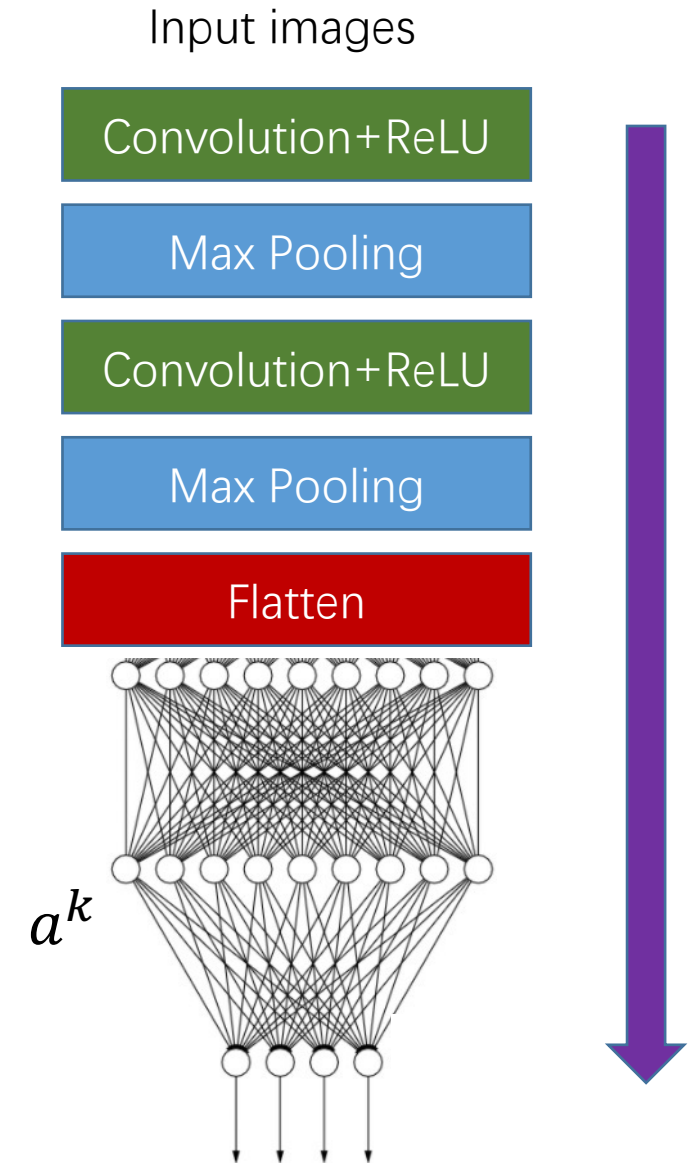
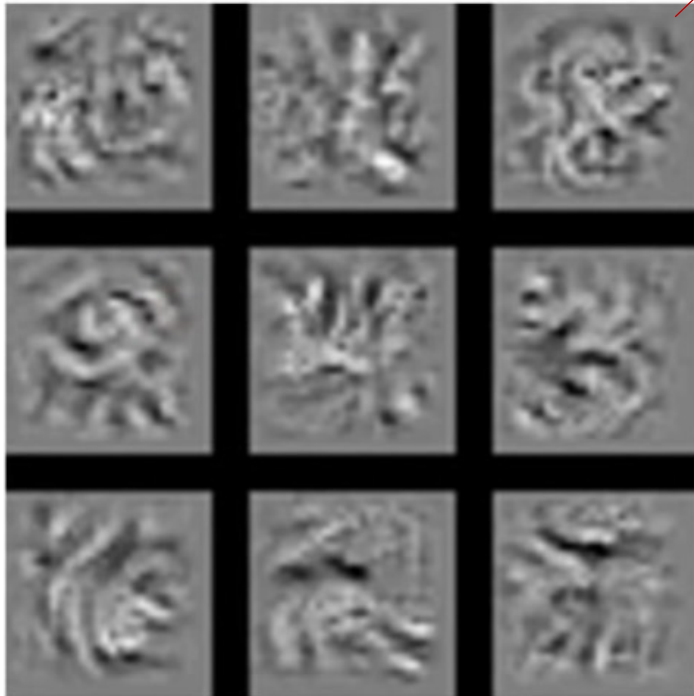
# CNN学了什么？

同理，当 $a^k$ 为FC layer的一个神经元

$\forall a^k$ ，寻找： $x^k = \arg \max_x a^k$

整个图片  
的信息

MNIST  
其中9个 $x^k$



# 小结

- 卷积神经网络
  - ✓ Conv+ReLU
  - ✓ Max pooling
- 各层的反向传播
- 理解CNN



# L06

- CNN的应用
- 神经网络训练细节
  - ✓ 激活函数
  - ✓ 数据预处理
  - ✓ 参数初始化
  - ✓ Batch Normalization