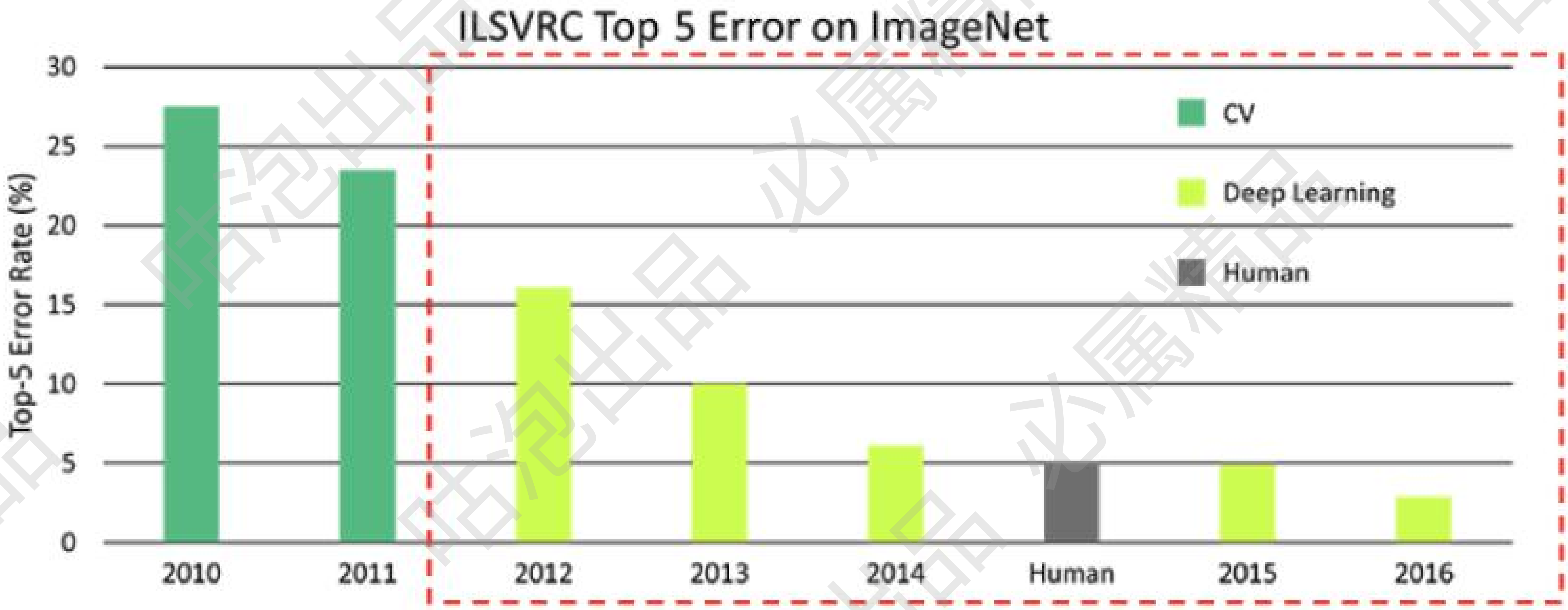
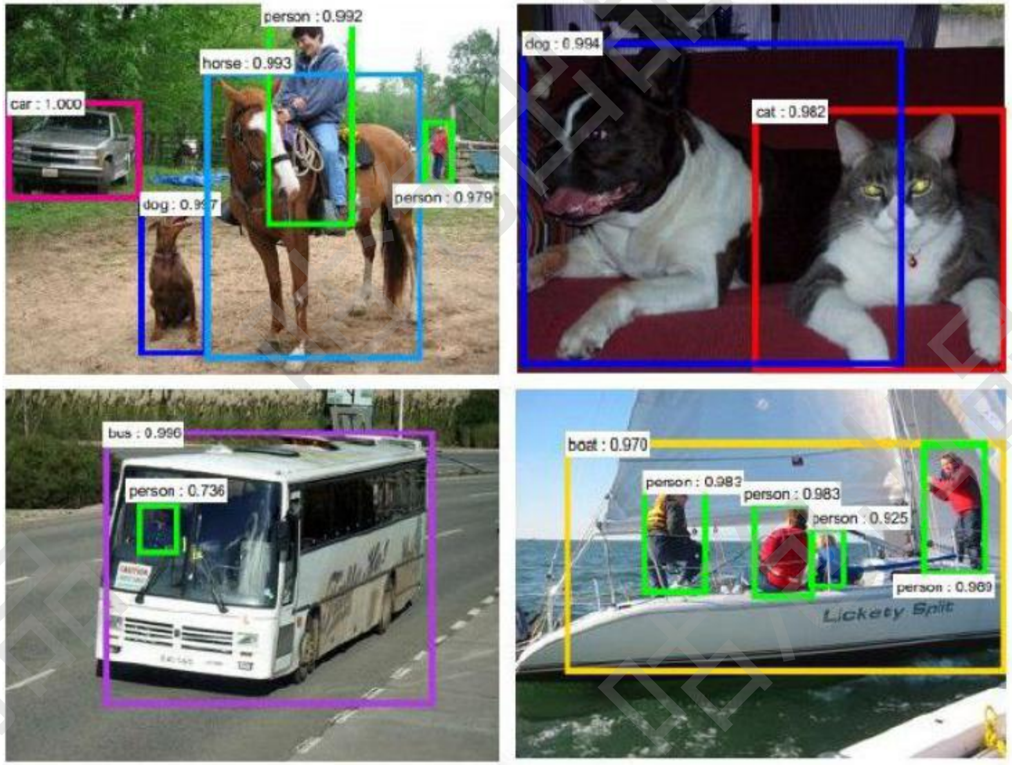
# 计算机视觉

- CV领域发展：



# 卷积神经网络：

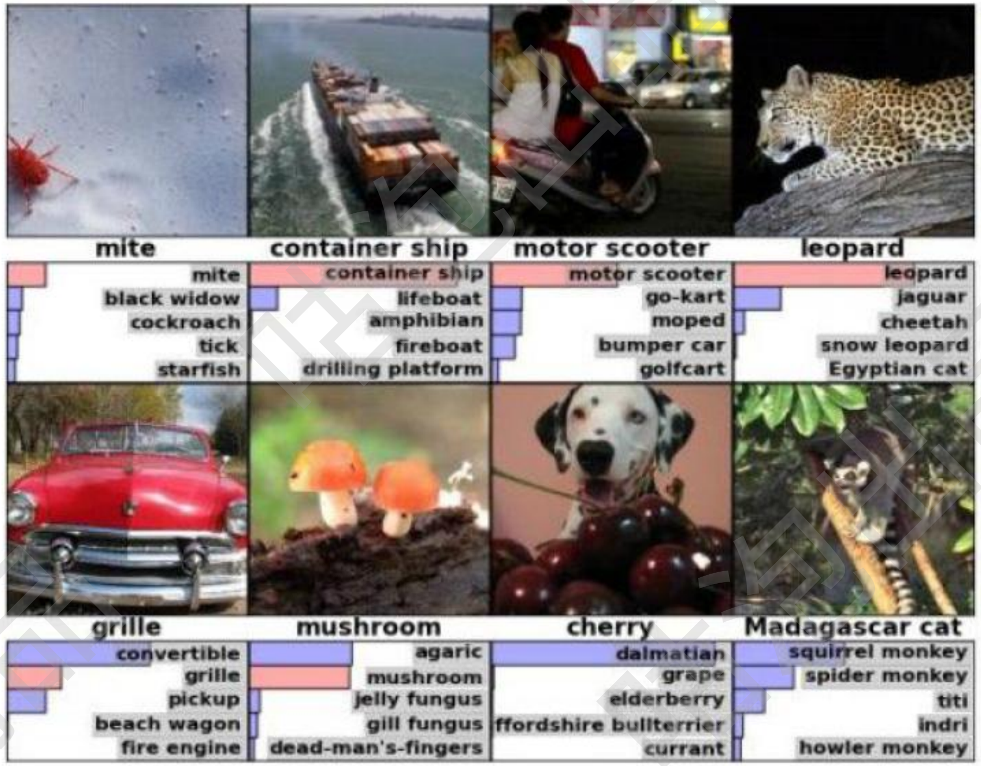
- 检测任务：



car : 1.000
horse : 0.993
person : 0.992
dog : 0.997
person : 0.979
dog : 0.994
cat : 0.982
bus : 0.996
person : 0.736
boat : 0.970
person : 0.983
person : 0.983
person : 0.925
person : 0.989
Lickety Split


# 卷积神经网络：

分类与检索：



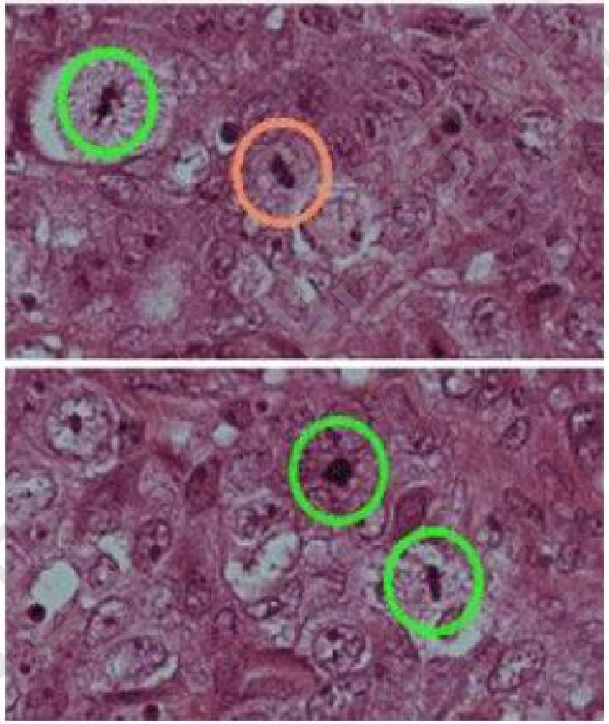

# 卷积神经网络：

- 超分辨率重构：

Inputs | Our result | SRGAN result | Original

# 卷积神经网络：

- 医学任务等：



# 卷积神经网络：

- 无人驾驶：

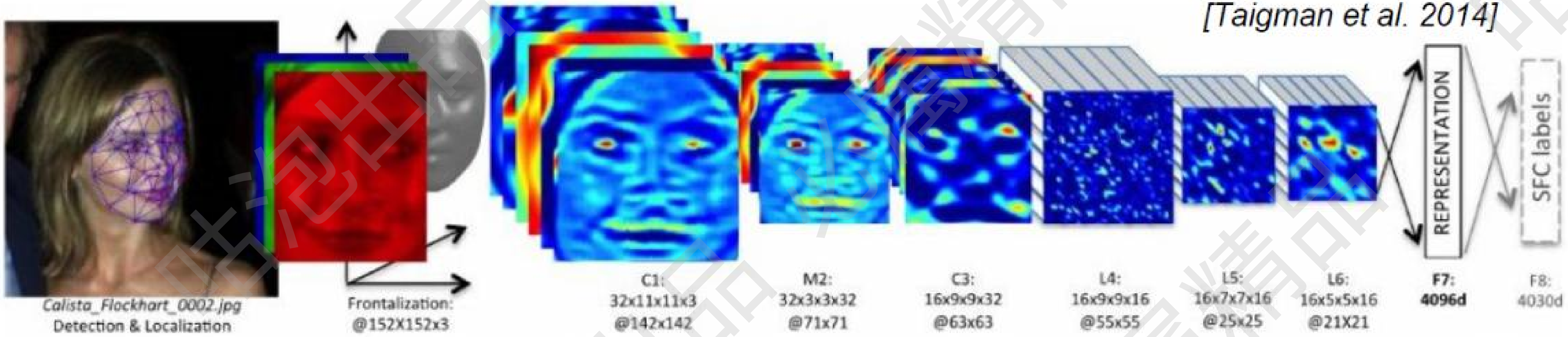


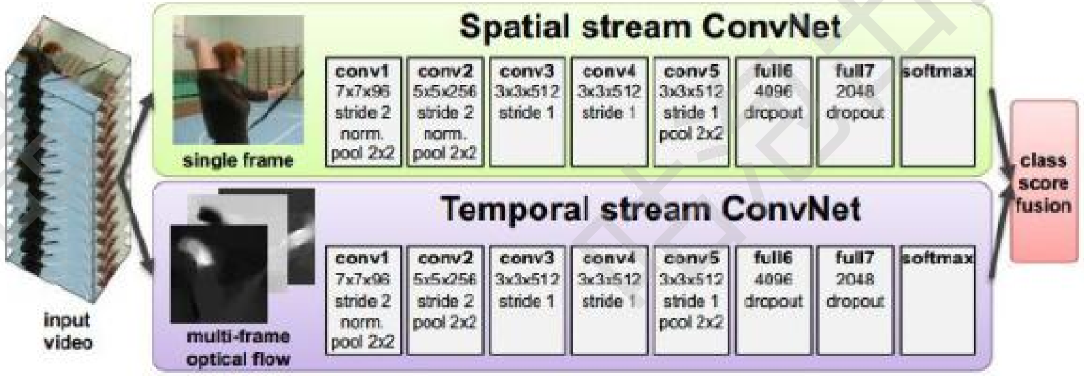
NVIDIA Tegra X1

# 卷积神经网络：

- ✓ 人脸识别：

[Taigman et al. 2014]



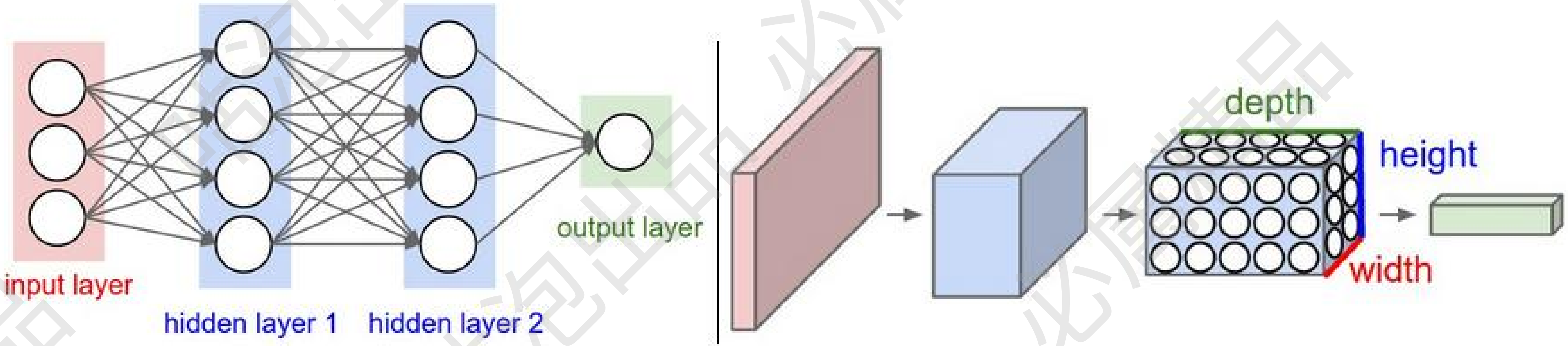




[Simonyan et al. 2014]

[Goodfellow 2014]

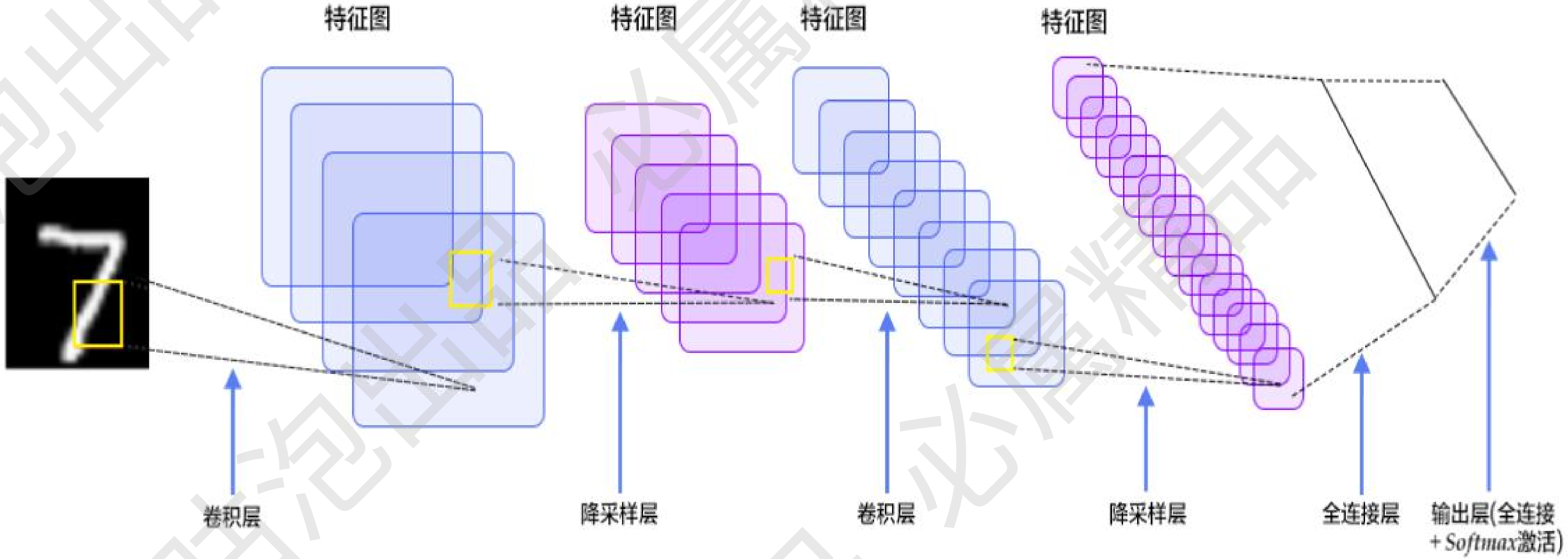
# 卷积神经网络：

- 卷积网络与传统网络的区别：



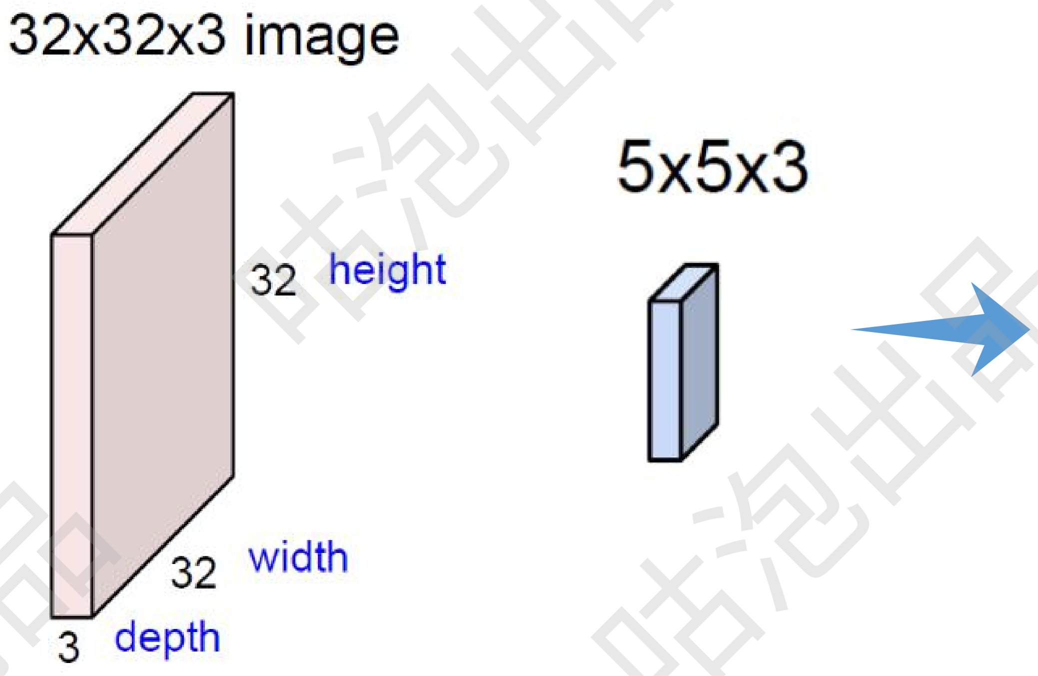
# 卷积神经网络：

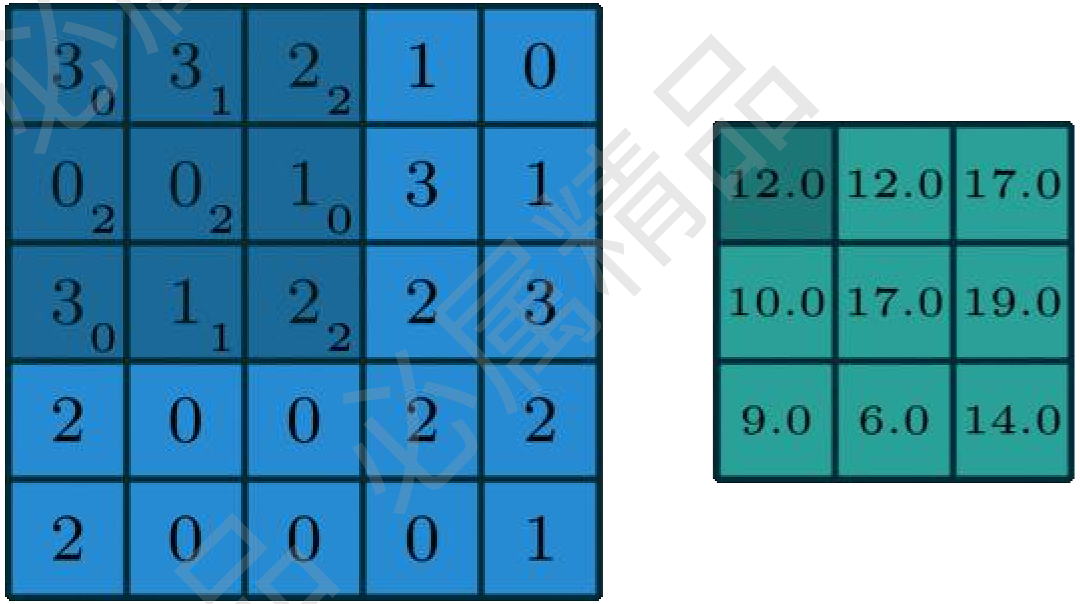
- 整体架构：
  - 输入层
  - 卷积层
  - 池化层
  - 全连接层



# 卷积神经网络：

- 卷积做了一件什么事?



# 卷积神经网络：

## 图像颜色通道

Original image (RGB)

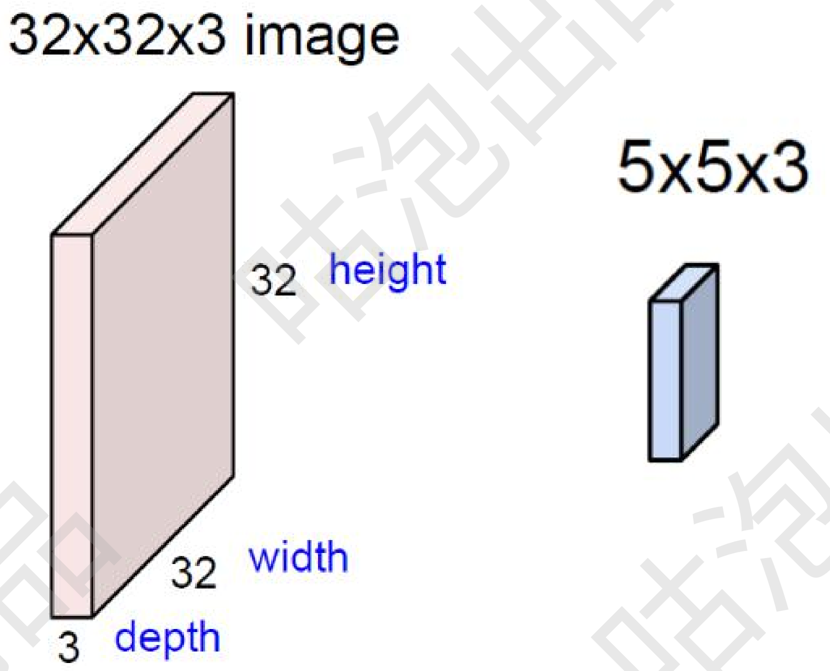
R channel

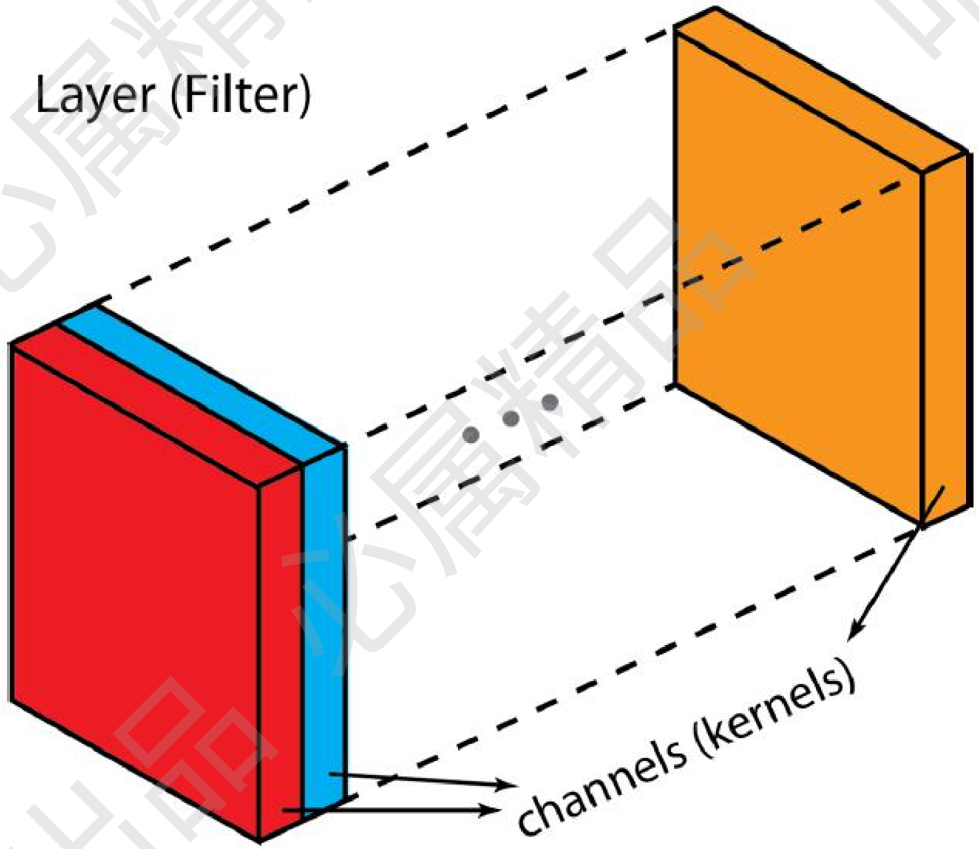
G channel

B channel

# 卷积神经网络：

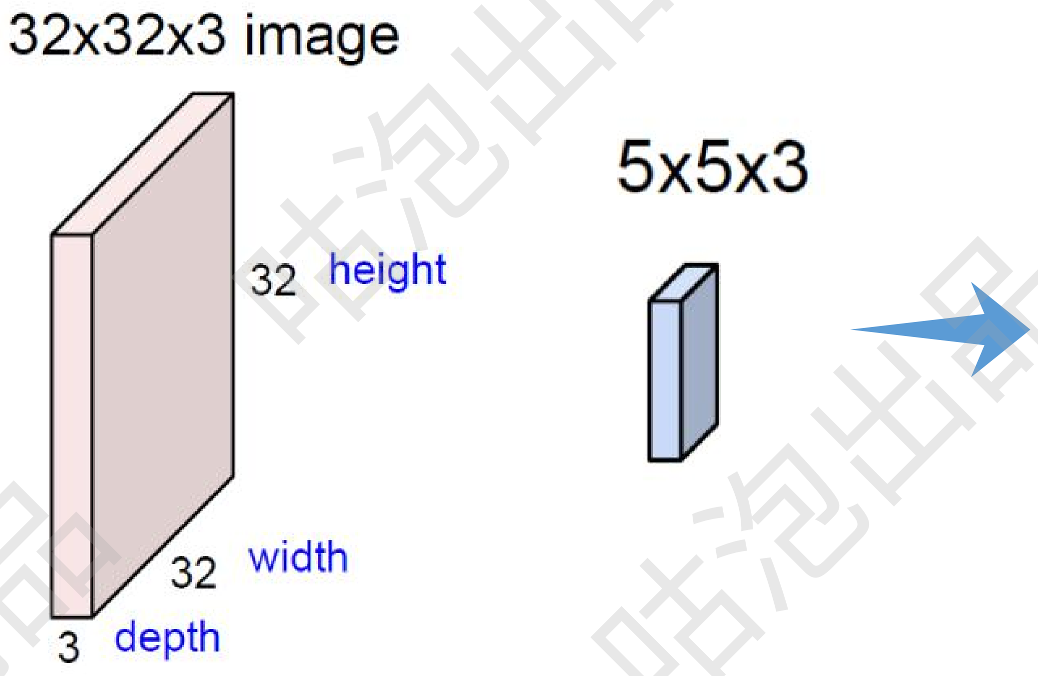
- 卷积做了一件什么事？

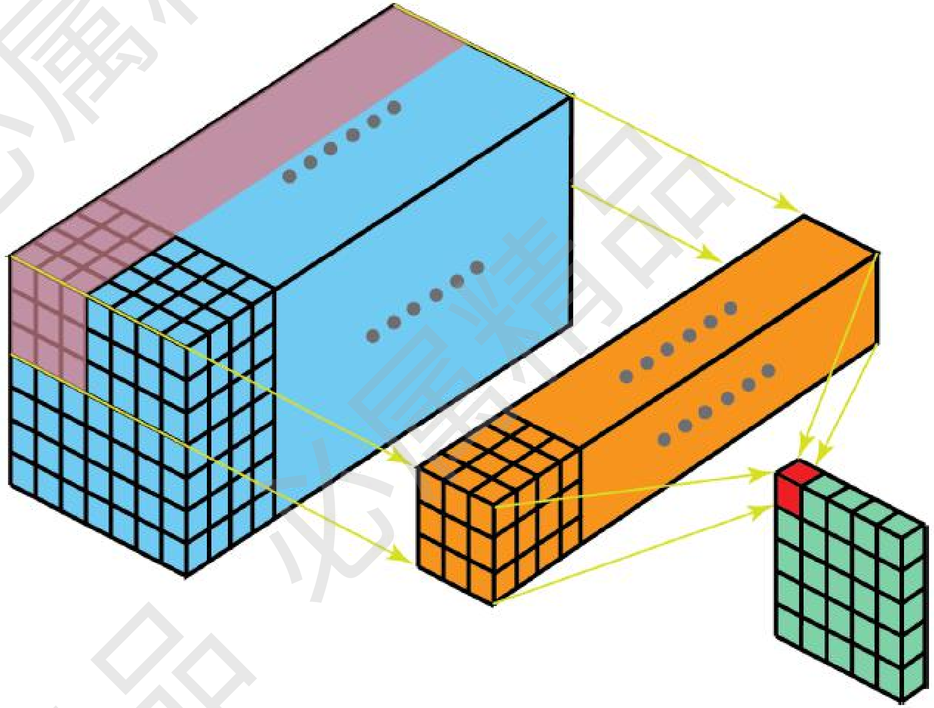


# 卷积神经网络：

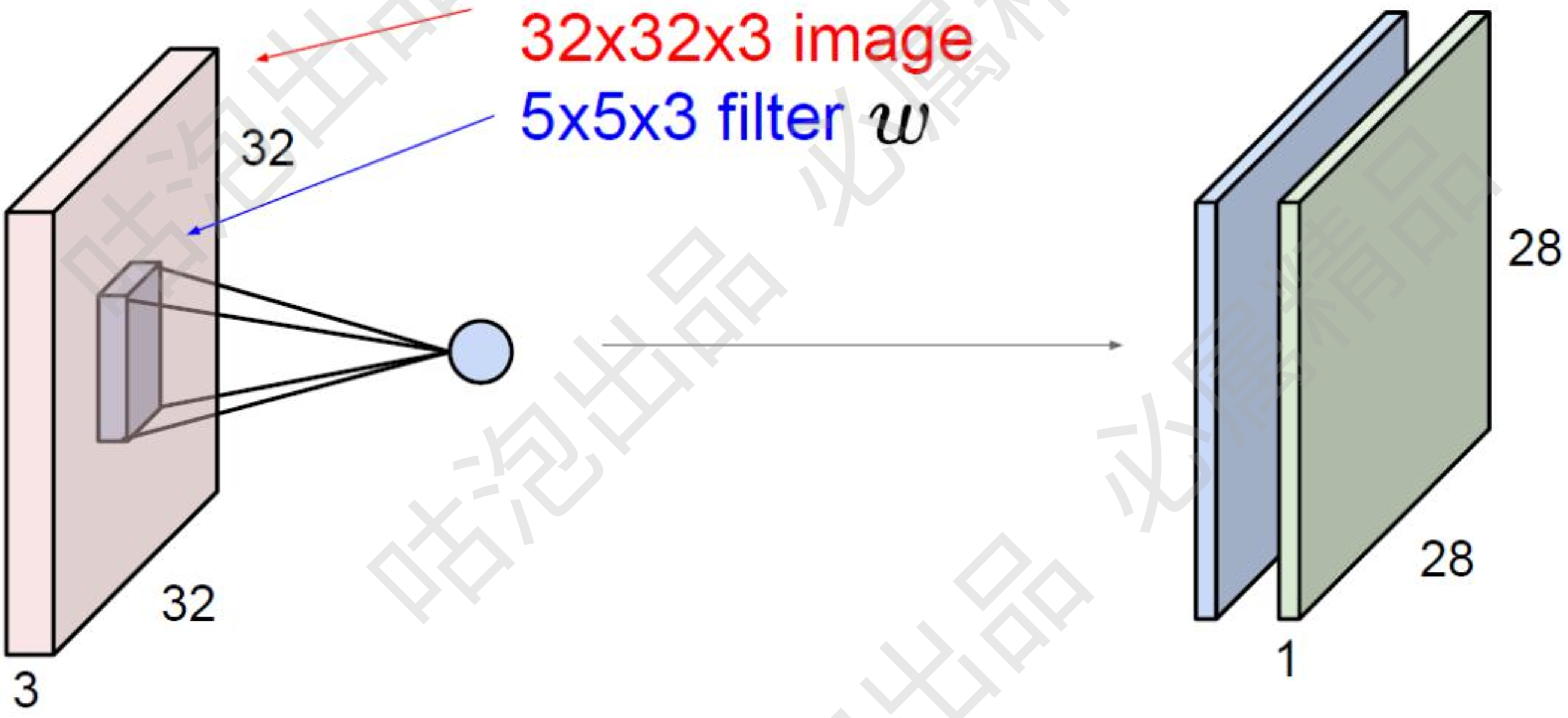
✓ 卷积做了一件什么事？



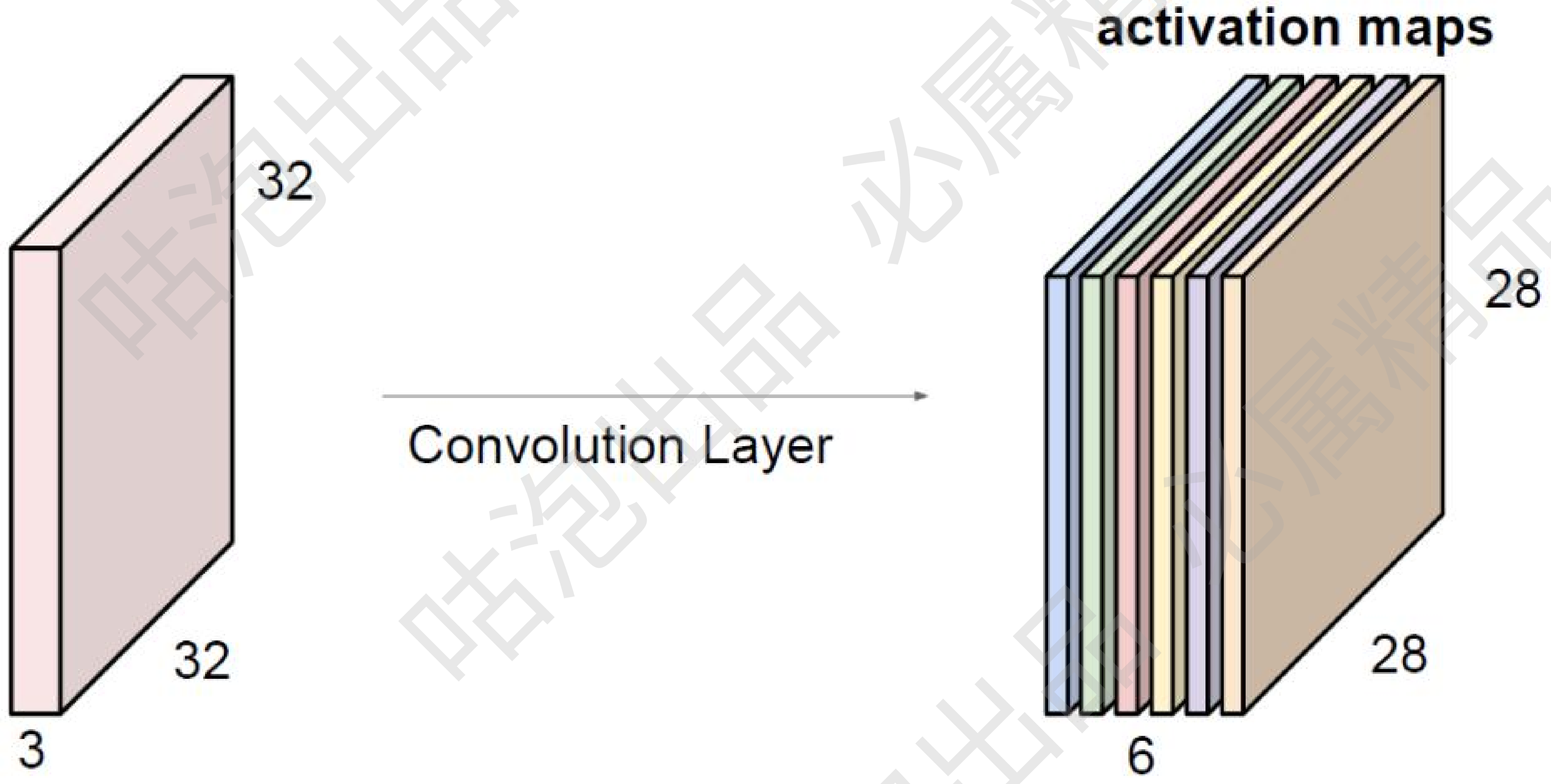
# 卷积神经网络：

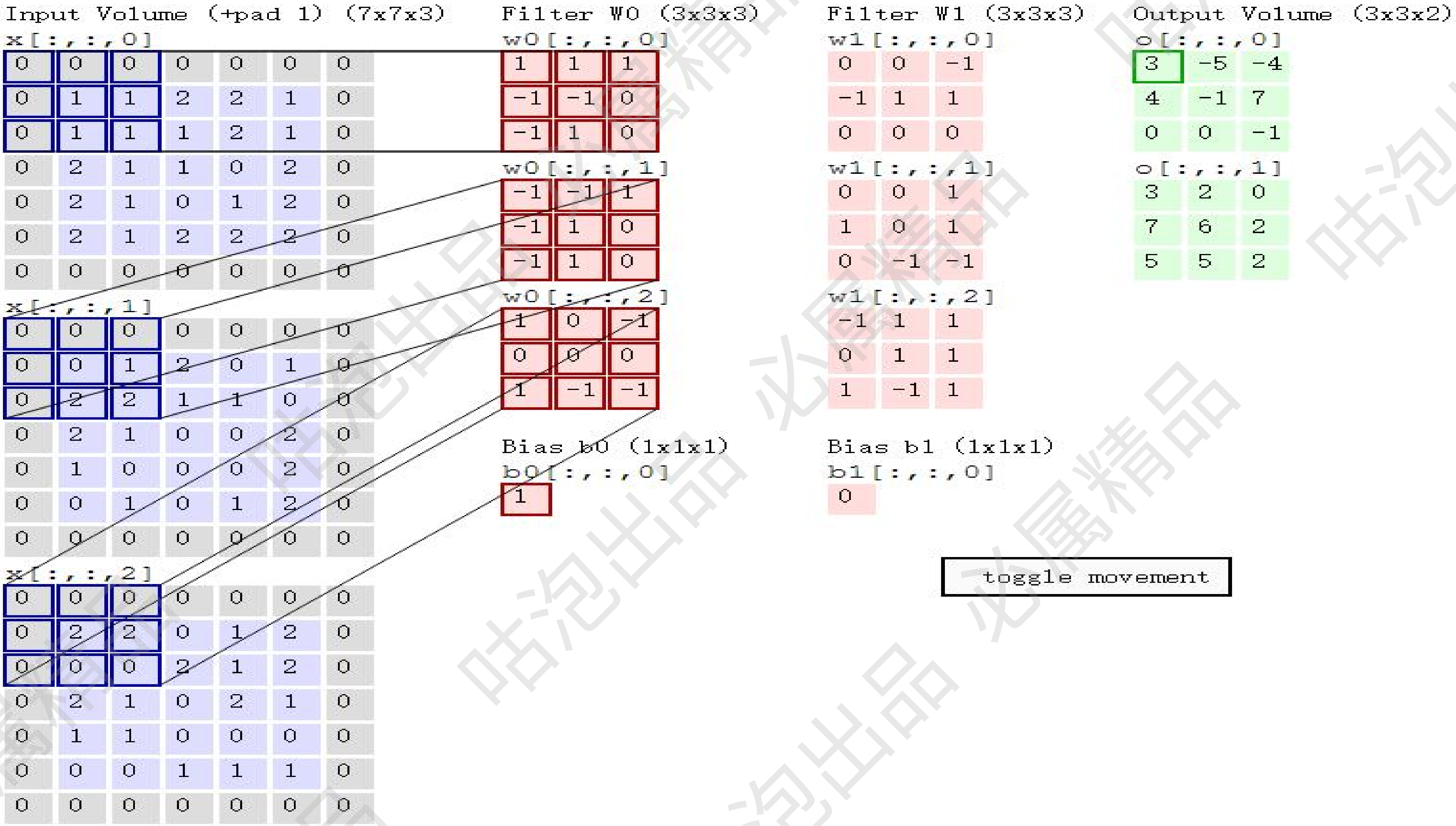
- 特征图个数

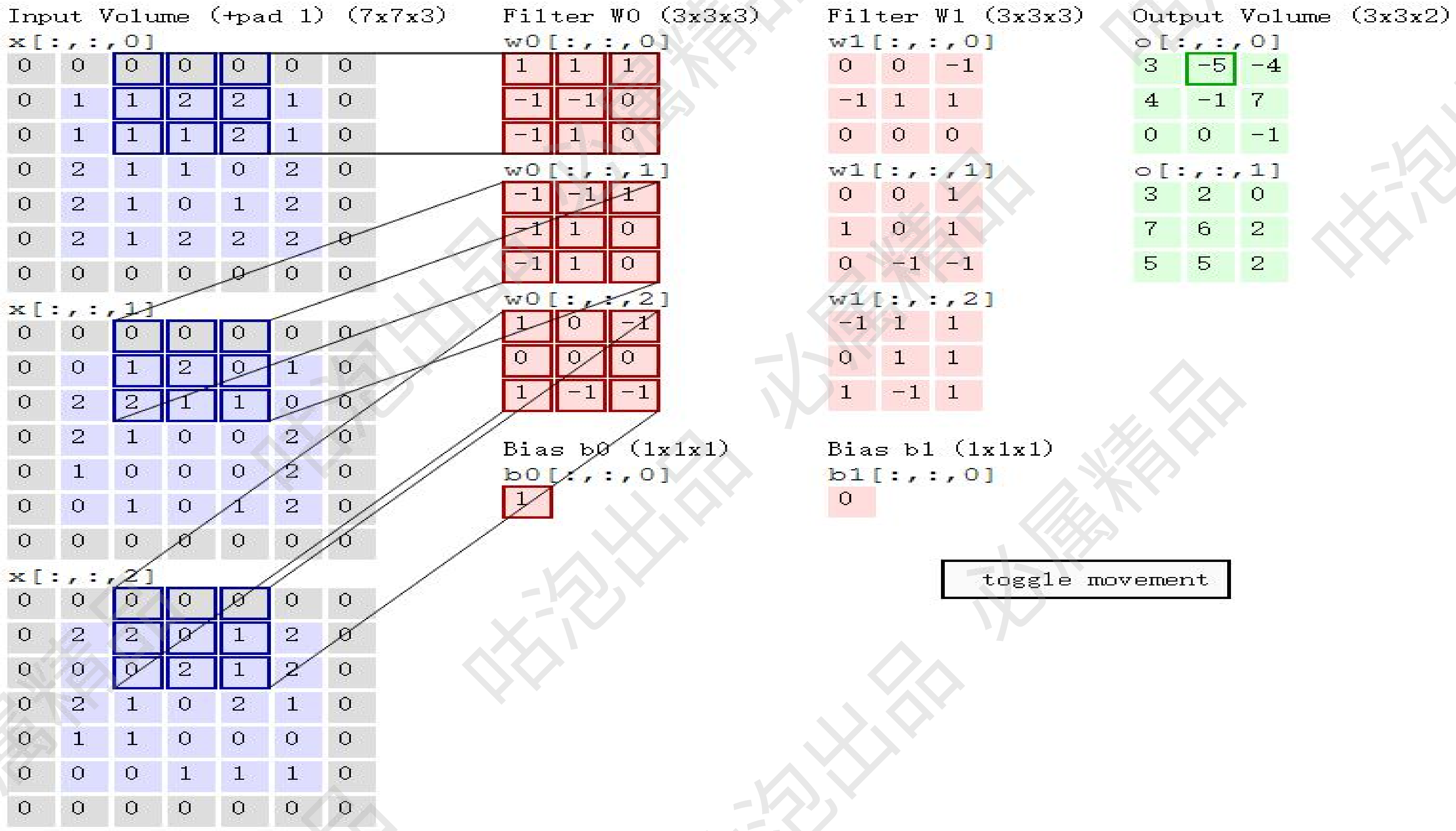


# 卷积神经网络：

- 特征图个数




Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 1 2 2 1 0
0 1 1 1 2 1 0
0 2 1 1 0 2 0
0 2 1 0 1 2 0
0 2 1 2 2 2 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 0 1 2 0 1 0
0 2 2 1 1 0 0
0 2 1 0 0 2 0
0 1 0 0 0 2 0
0 0 1 0 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 2 2 0 1 2 0
0 0 0 2 1 2 0
0 2 1 0 2 1 0
0 1 1 0 0 0 0
0 0 0 1 1 1 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
1 1 1
-1 -1 0
-1 1 0
w0[:,:,1]
-1 -1 1
-1 1 0
-1 1 0
w0[:,:,2]
1 0 -1
0 0 0
1 -1 -1
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
0 0 -1
-1 1 1
0 0 0
w1[:,:,1]
0 0 1
1 0 1
0 -1 -1
w1[:,:,2]
-1 1 1
0 1 1
1 -1 1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
3 -5 -4
4 -1 7
0 0 -1
o[:,:,1]
3 2 0
7 6 2
5 5 2
toggle movement



Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 1 2 2 1 0
0 1 1 1 2 1 0
0 2 1 1 0 2 0
0 2 1 0 1 2 0
0 2 1 2 2 2 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 0 1 2 0 1 0
0 2 2 1 1 0 0
0 2 1 0 0 2 0
0 1 0 0 0 2 0
0 0 1 0 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 2 2 0 1 2 0
0 0 0 2 1 2 0
0 2 1 0 2 1 0
0 1 1 0 0 0 0
0 0 0 1 1 1 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
1 1 1
-1 -1 0
-1 1 0
w0[:,:,1]
-1 -1 1
-1 1 0
-1 1 0
w0[:,:,2]
1 0 -1
0 0 0
1 -1 -1
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
0 0 -1
-1 1 1
0 0 0
w1[:,:,1]
0 0 1
1 0 1
0 -1 -1
w1[:,:,2]
-1 1 1
0 1 1
1 -1 1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
3 -5 -4
4 -1 7
0 0 -1
o[:,:,1]
3 2 0
7 6 2
5 5 2
toggle movement

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 2 | 2 | 1 | 0 |
| 0 | 1 | 1 | 1 | 2 | 1 | 0 |
| 0 | 2 | 1 | 1 | 0 | 2 | 0 |
| 0 | 2 | 1 | 0 | 1 | 2 | 0 |
| 0 | 2 | 1 | 2 | 2 | 2 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

x[:,:,1]

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 2 | 0 | 1 | 0 |
| 0 | 2 | 2 | 1 | 1 | 0 | 0 |
| 0 | 2 | 1 | 0 | 0 | 2 | 0 |
| 0 | 1 | 0 | 0 | 0 | 2 | 0 |
| 0 | 0 | 1 | 0 | 1 | 2 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

x[:,:,2]

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 2 | 0 | 1 | 2 | 0 |
| 0 | 0 | 0 | 2 | 1 | 2 | 0 |
| 0 | 2 | 1 | 0 | 2 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Filter W0 (3x3x3)

w0[:,:,0]

| | | |
|---|---|---|
| 1 | 1 | 1 |
| -1 | -1 | 0 |
| -1 | 1 | 0 |

w0[:,:,1]

| | | |
|---|---|---|
| -1 | -1 | 1 |
| -1 | 1 | 0 |
| -1 | 1 | 0 |

w0[:,:,2]

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 0 | 0 | 0 |
| 1 | -1 | -1 |

Bias b0 (1x1x1)

b0[:,:,0]

| |
|---|
| 1 |

Filter W1 (3x3x3)

w1[:,:,0]

| | | |
|---|---|---|
| 0 | 0 | -1 |
| -1 | 1 | 1 |
| 0 | 0 | 0 |

w1[:,:,1]

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 1 |
| 0 | -1 | -1 |

w1[:,:,2]

| | | |
|---|---|---|
| -1 | 1 | 1 |
| 0 | 1 | 1 |
| 1 | -1 | 1 |

Bias b1 (1x1x1)

b1[:,:,0]

| |
|---|
| 0 |

Output Volume (3x3x2)

o[:,:,0]

| | | |
|---|---|---|
| 3 | -5 | -4 |
| 4 | -1 | 7 |
| 0 | 0 | -1 |

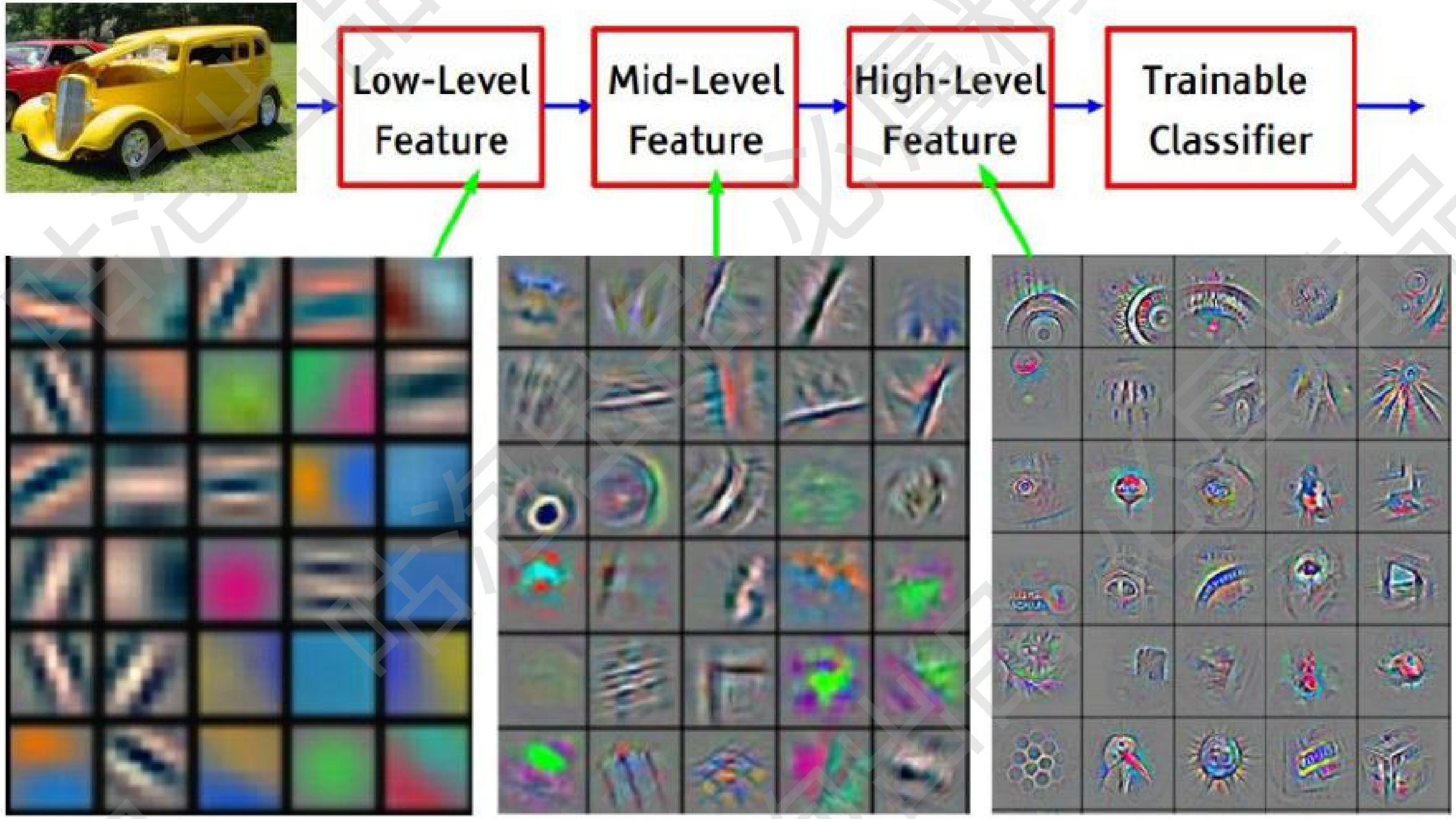
o[:,:,1]

| | | |
|---|---|---|
| 3 | 2 | 0 |
| 7 | 6 | 2 |
| 5 | 5 | 2 |

toggle movement

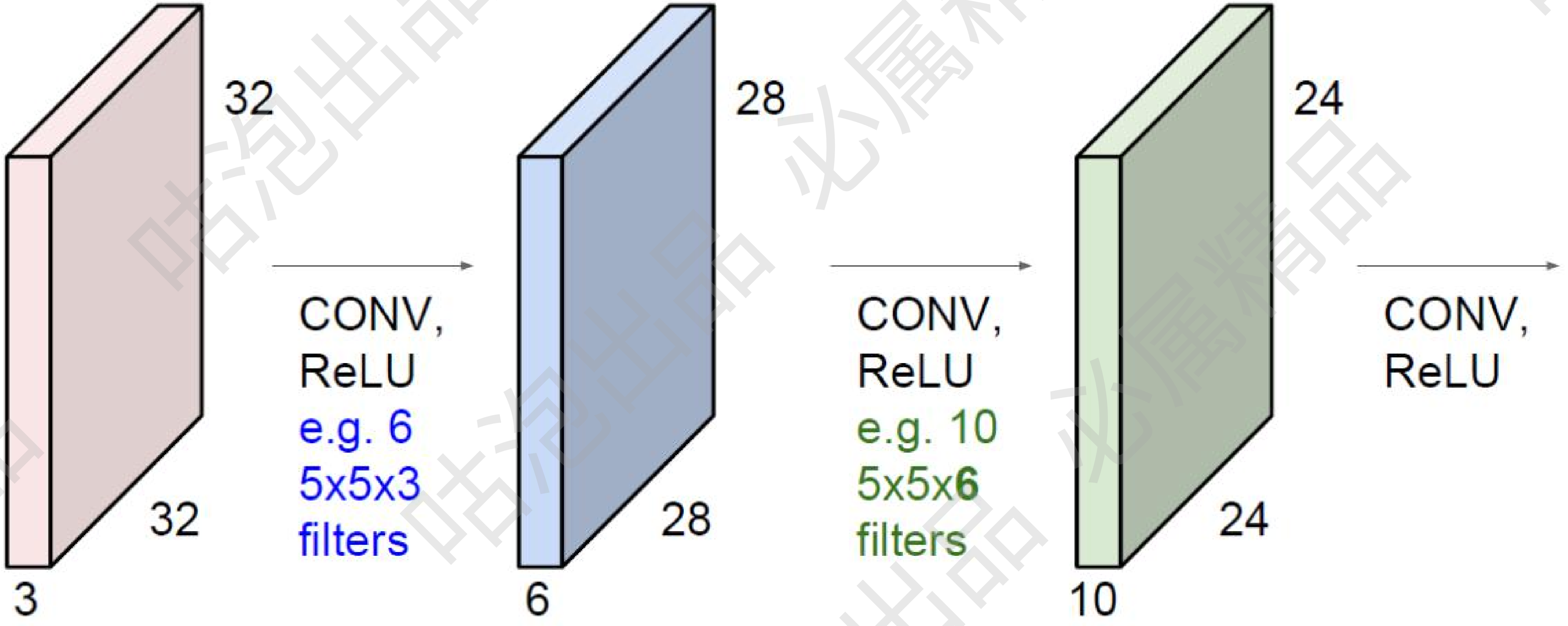
# 卷积神经网络：

- 只做一次卷积就可以了吗？



# 卷积神经网络：

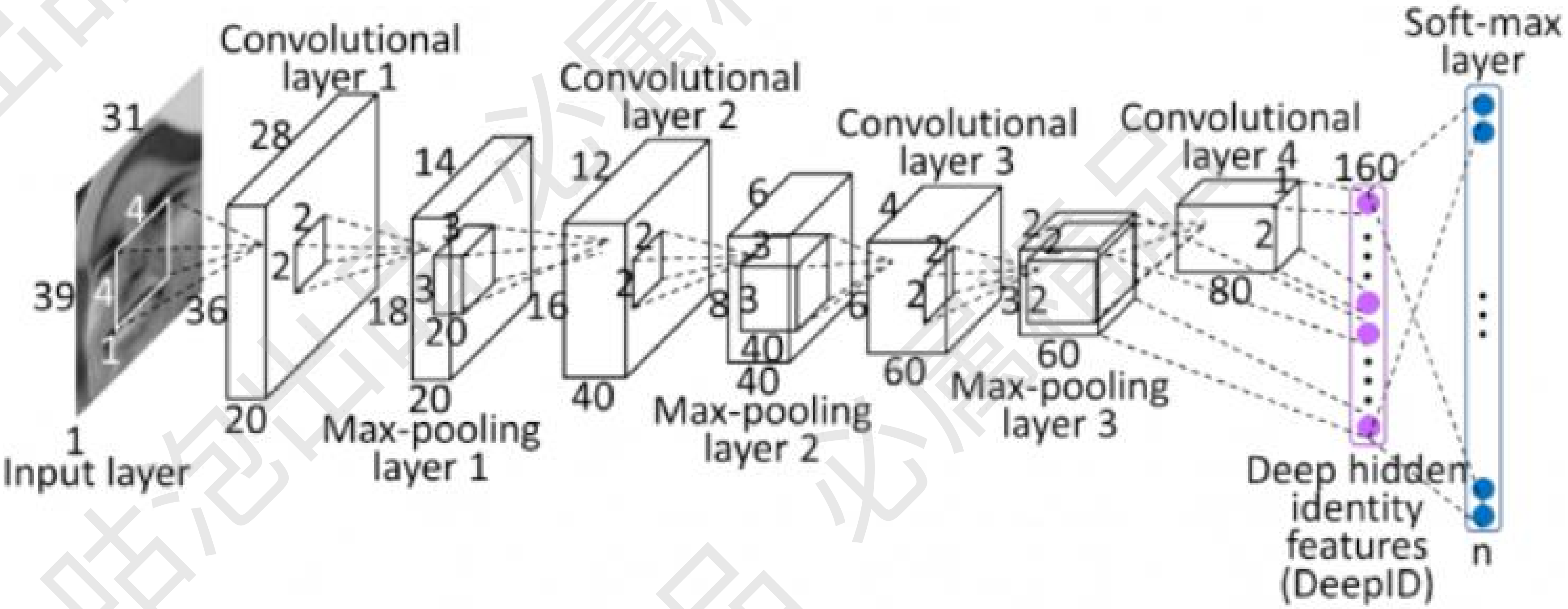
✓ 堆叠的卷积层



# 卷积神经网络：

✓ 卷积层涉及参数：

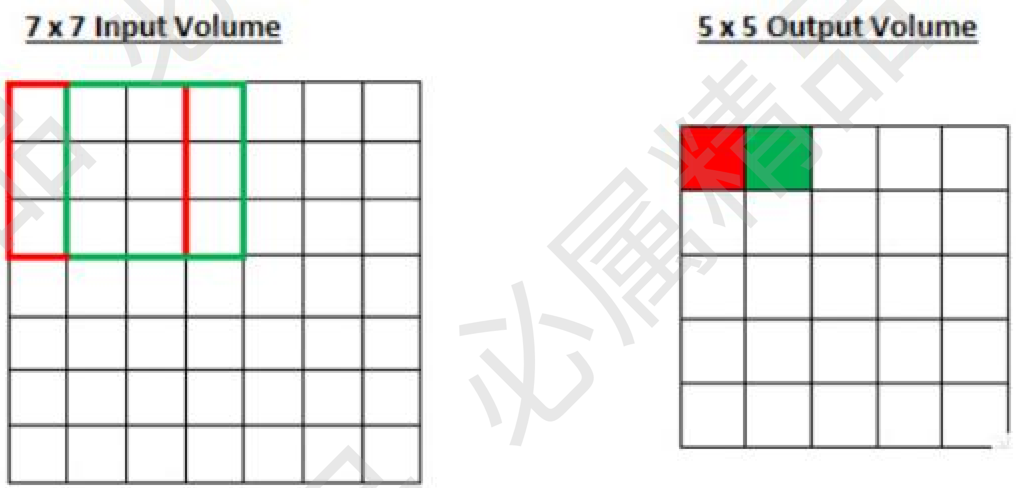
- 滑动窗口步长
- 卷积核尺寸
- 边缘填充
- 卷积核个数



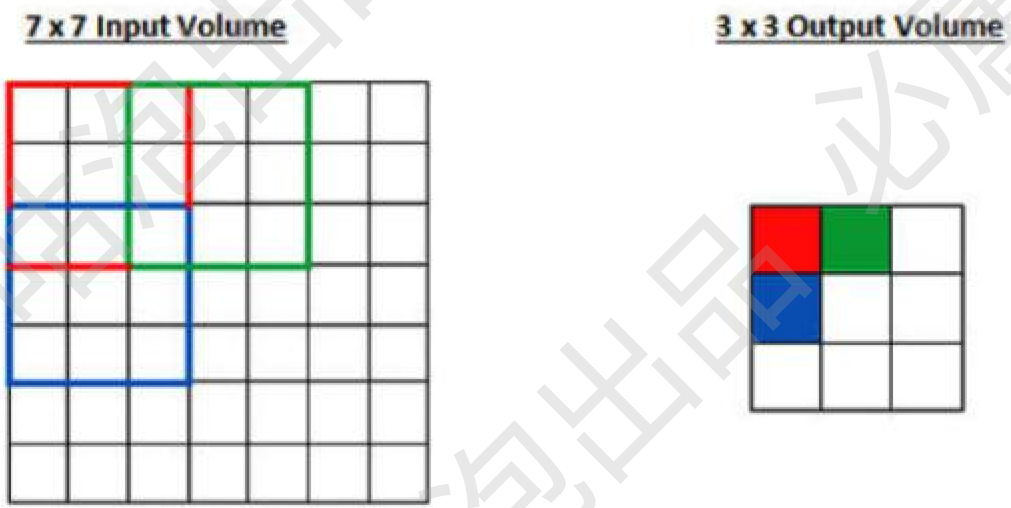
# 卷积神经网络：

✓ 步长：

✎ 步长为1的卷积：

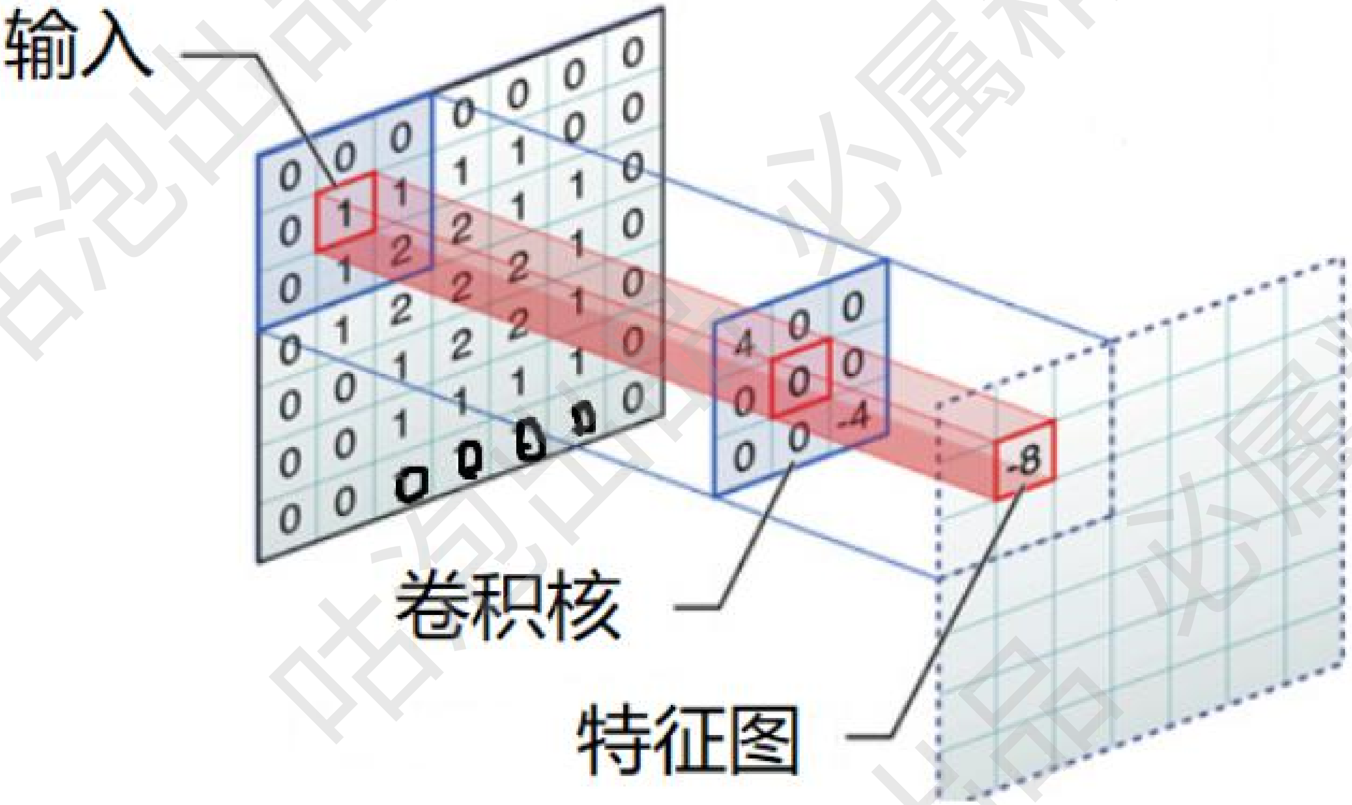



✎ 步长为2的卷积：



# 卷积神经网络：

- 边界填充：

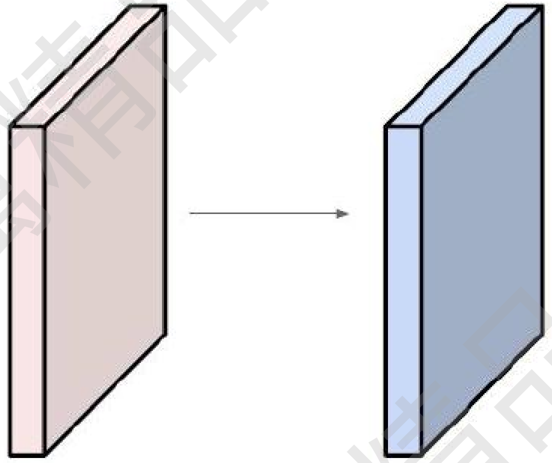


# 卷积神经网络：

卷积结果计算公式：

长度： $H_2 = \frac{H_1 - F_H + 2P}{S} + 1$

宽度： $W_2 = \frac{W_1 - F_W + 2P}{S} + 1$



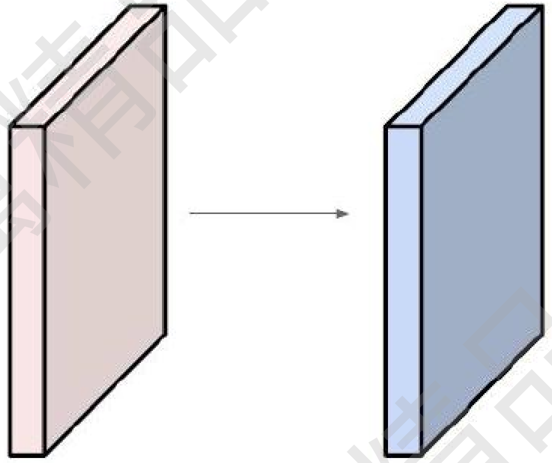
其中W1、H1表示输入的宽度、长度；W2、H2表示输出特征图的宽度、长度；F表示卷积核长和宽的大小；S表示滑动窗口的步长;P表示边界填充(加几圈0)。

# 卷积神经网络：

- 卷积结果计算公式：

- 长度： $H_2 = \frac{H_1 - F_H + 2P}{S} + 1$

- 宽度： $W_2 = \frac{W_1 - F_W + 2P}{S} + 1$

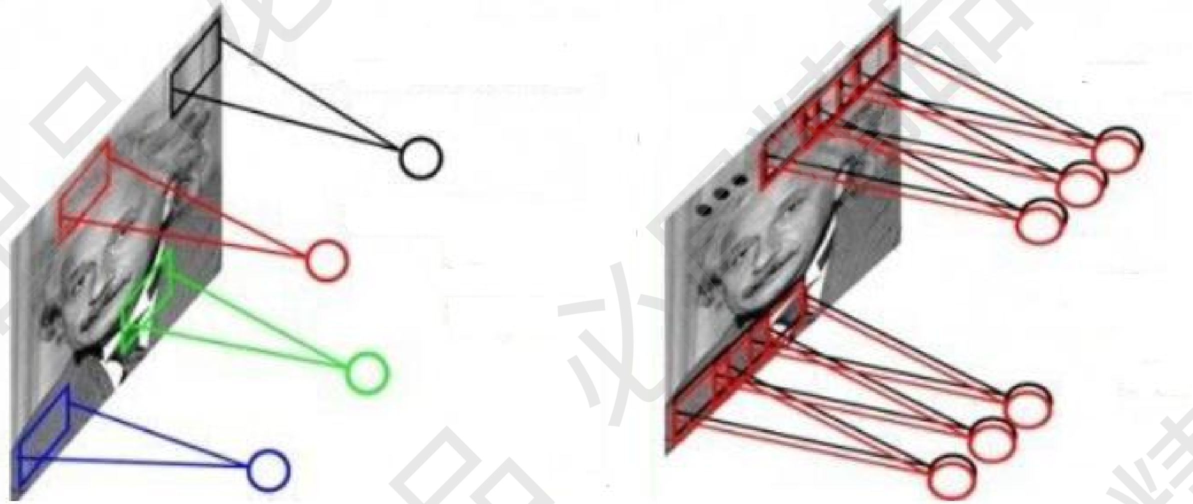


- 如果输入数据是32*32*3的图像，用10个5*5*3的filter来进行卷积操作，指定步长为1，边界填充为2，最终输入的规模为?

- (32-5+2*2)/1 + 1 = 32，所以输出规模为32*32*10，经过卷积操作后也可以保持特征图长度、宽度不变。

# 卷积神经网络：

- 卷积参数共享：

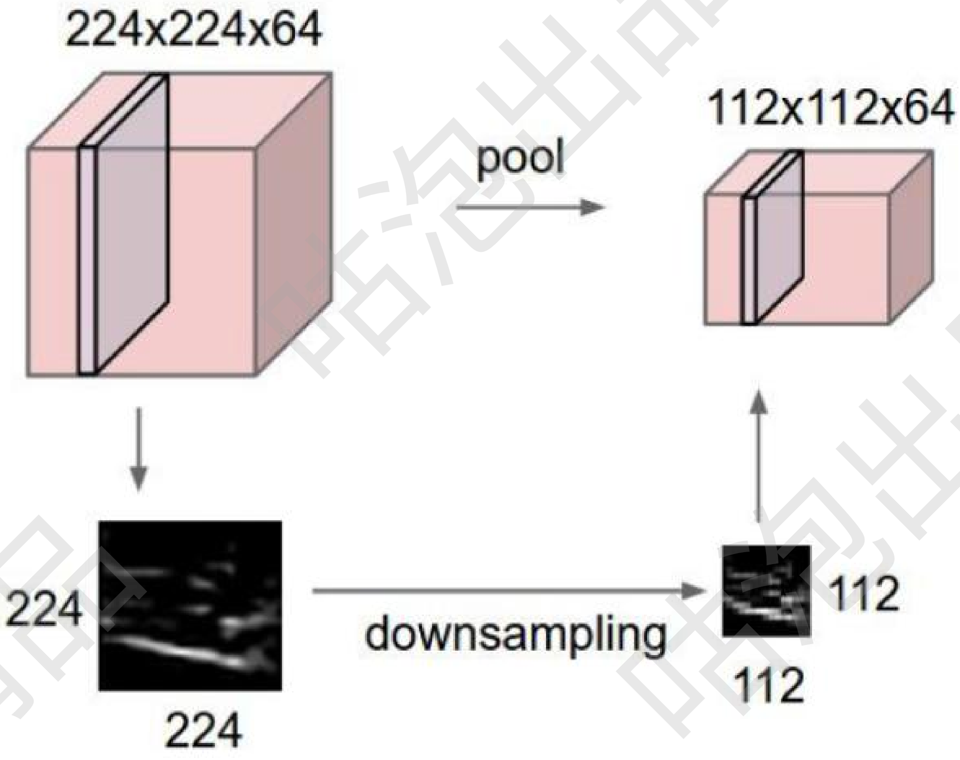


- 数据依旧是32*32*3的图像，继续用10个5*5*3的filter来进行卷积操作，所需的权重参数有多少个呢？

- 5*5*3 = 75，表示每一个卷积核只需要75个参数，此时有10个不同的卷积核，就需要10*75 = 750个卷积核参数，不要忘记还有b参数，每个卷积核都有一个对应的偏置参数，最终只需要750+10=760个权重参数。

# 卷积神经网络：

- 池化层：

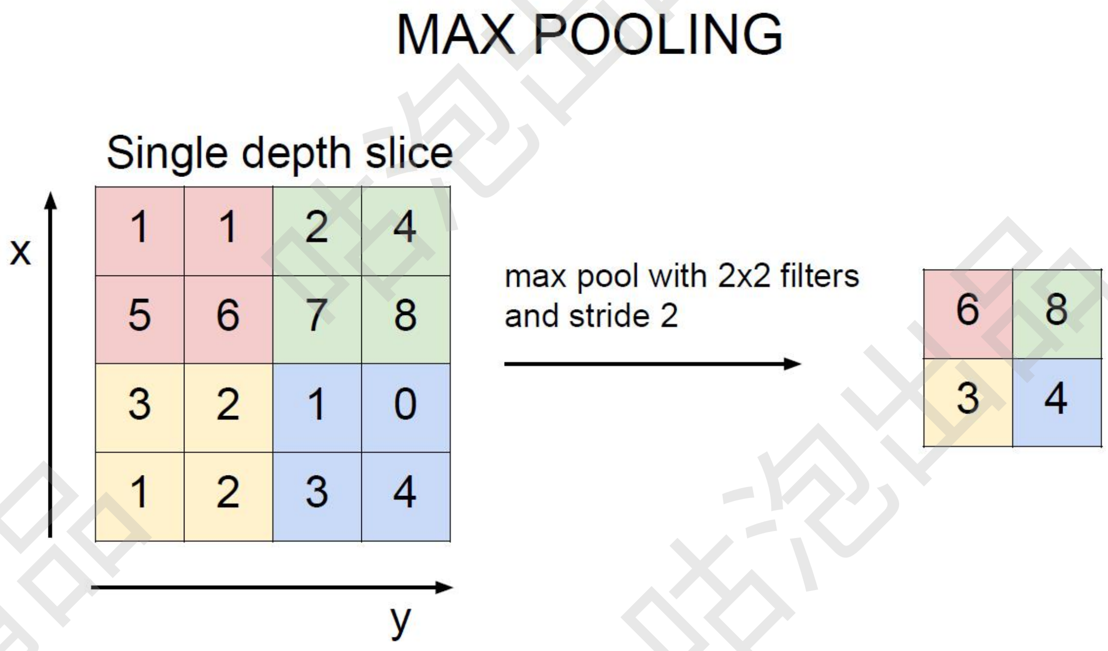



| | | | |
|---|---|---|---|
| 1 | 3 | 2 | 9 |
| 7 | 4 | 1 | 5 |
| 8 | 5 | 2 | 3 |
| 4 | 2 | 1 | 4 |

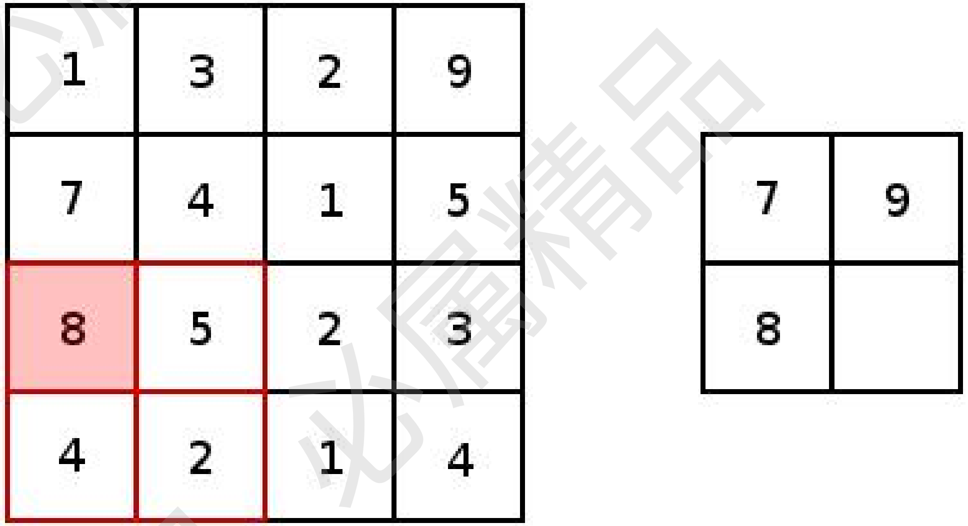
| | |
|---|---|
| 7 | 9 |
| 8 | |

# 卷积神经网络：

✓ 最大池化：

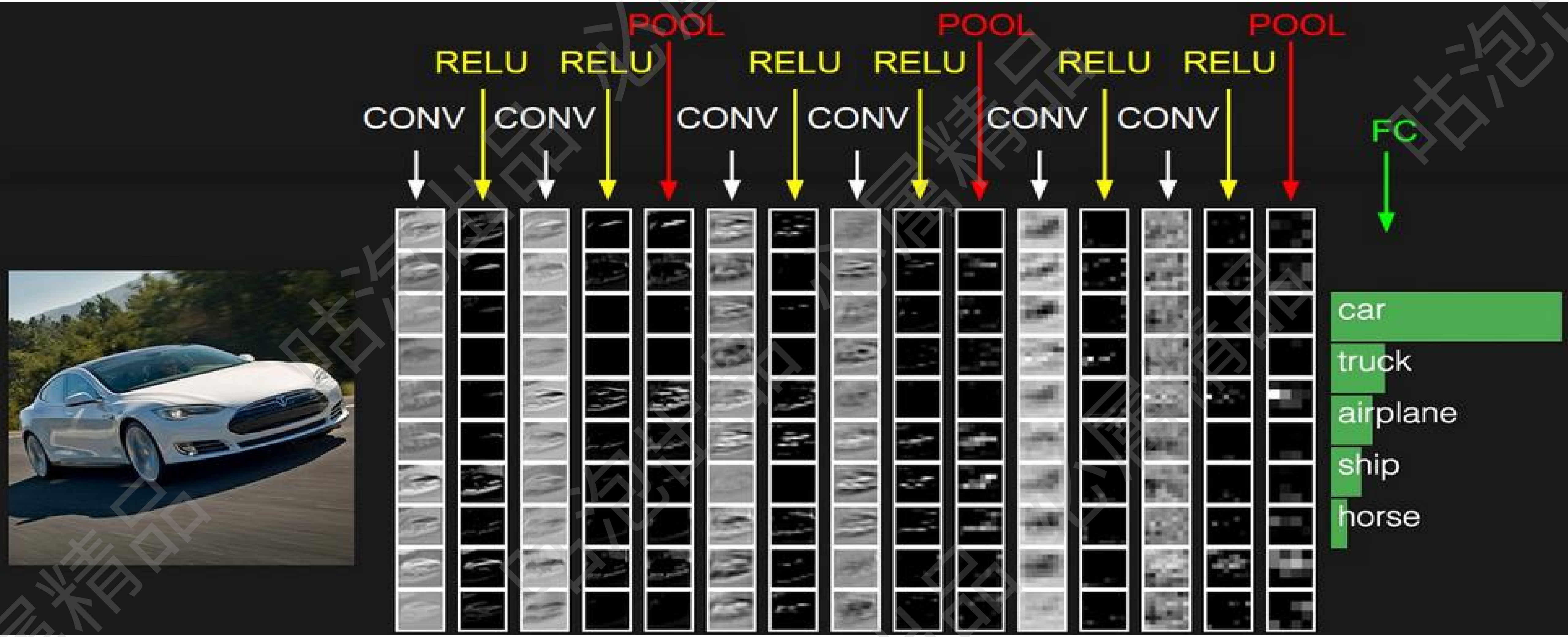


MAX POOLING
Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters and stride 2
6 8
3 4

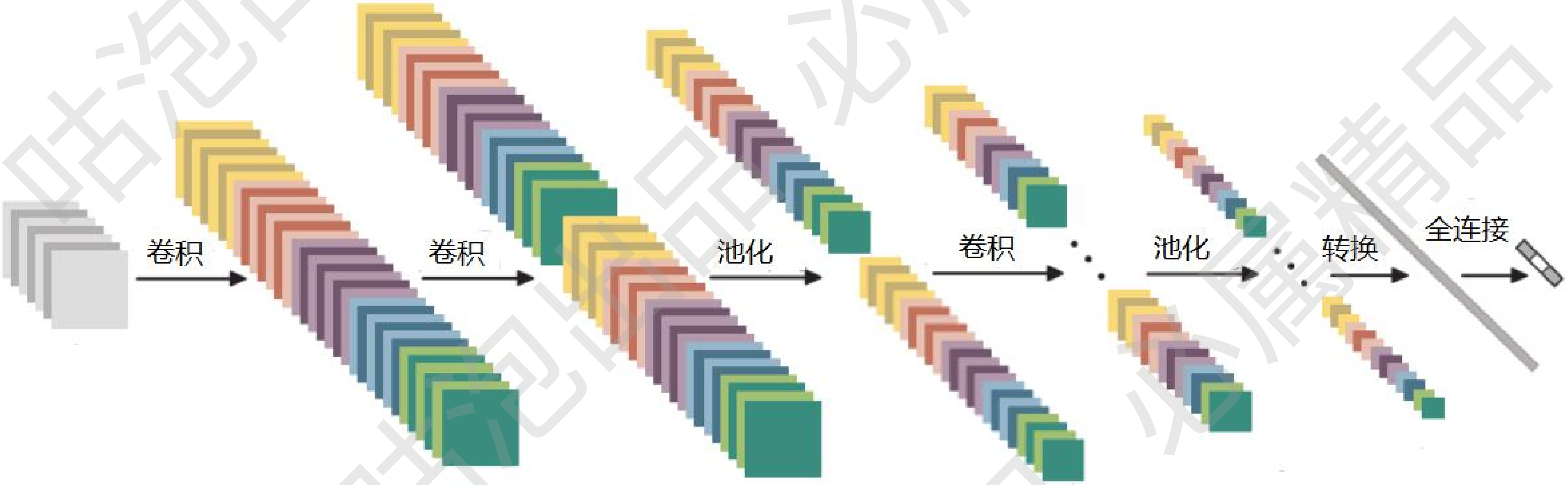
1 3 2 9
7 4 1 5
8 5 2 3
4 2 1 4
7 9
8

# 卷积神经网络：


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

# 卷积神经网络：

- 特征图变化：



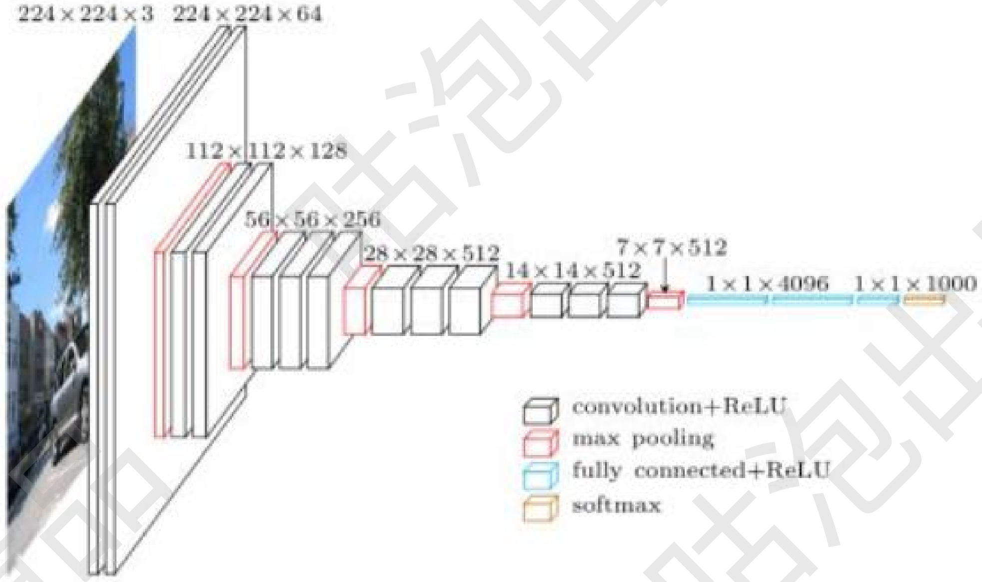
# 卷积神经网络：

✓ 经典网络-Alexnet：

[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

# 卷积神经网络：

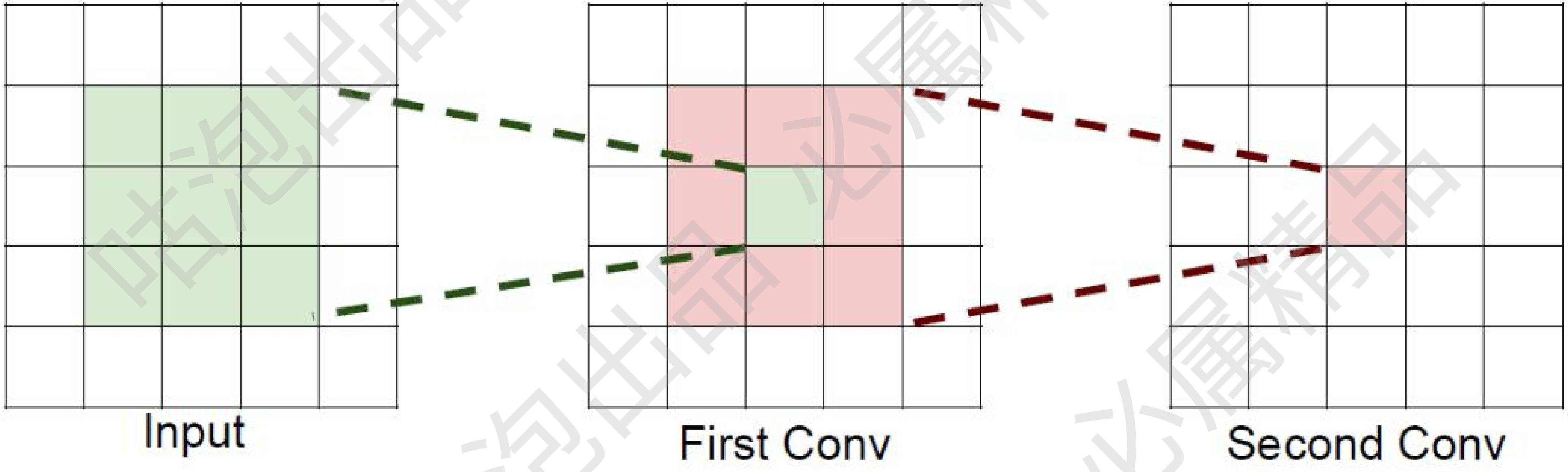
- 经典网络-Vgg：




| ConvNet Configuration | | | | | |
|---|---|---|---|---|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64<br>**LRN** | conv3-64<br>**conv3-64** | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128<br>**conv3-128** | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 |
| maxpool | | | | | |
| conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256<br>**conv1-256** | conv3-256<br>conv3-256<br>**conv3-256** | conv3-256<br>conv3-256<br>conv3-256<br>**conv3-256** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

# 卷积神经网络：

- 感受野：




- 如果堆叠3个3*3的卷积层，并且保持滑动窗口步长为1，其感受野就是7*7的了，这跟一个使用7*7卷积核的结果是一样的，那为什么非要堆叠3个小卷积呢？

# 卷积神经网络：

## 感受野

假设输入大小都是h*w*c，并且都使用c个卷积核(得到c个特征图)，可以来计算一下其各自所需参数：

一个7*7卷积核所需参数：

$= C \times (7 \times 7 \times C) = \mathbf{49\ C^2}$

3个3*3卷积核所需参数：

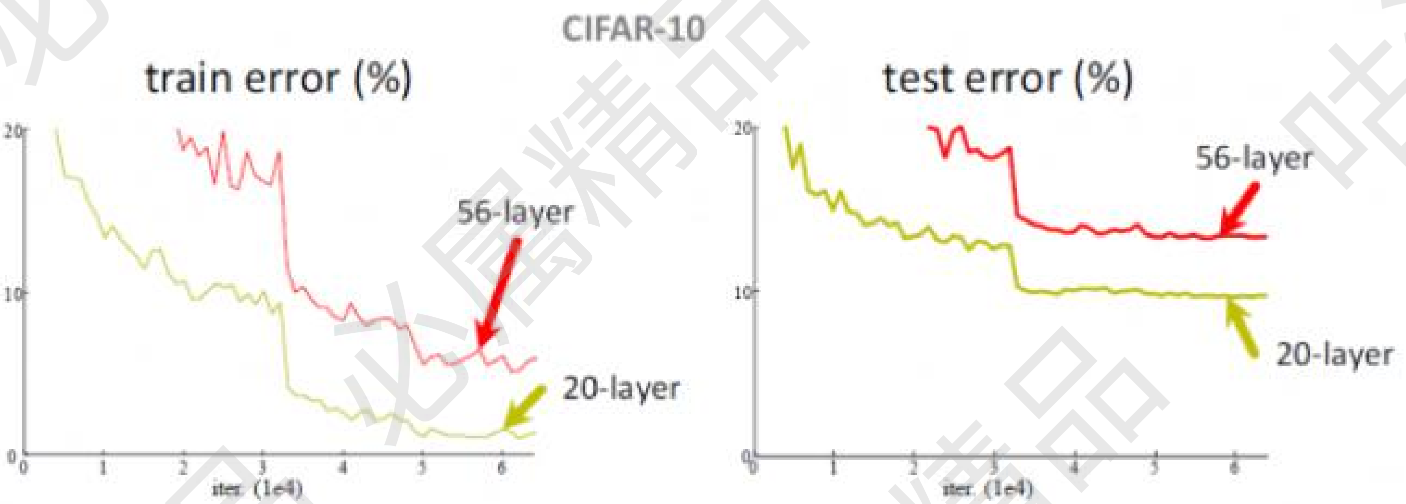
$= 3 \times C \times (3 \times 3 \times C) = \mathbf{27\ C^2}$

很明显，堆叠小的卷积核所需的参数更少一些，并且卷积过程越多，特征提取也会越细致，加入的非线性变换也随着增多，还不会增大权重参数个数，这就是VGG网络的基本出发点，用小的卷积核来完成体特征提取操作。

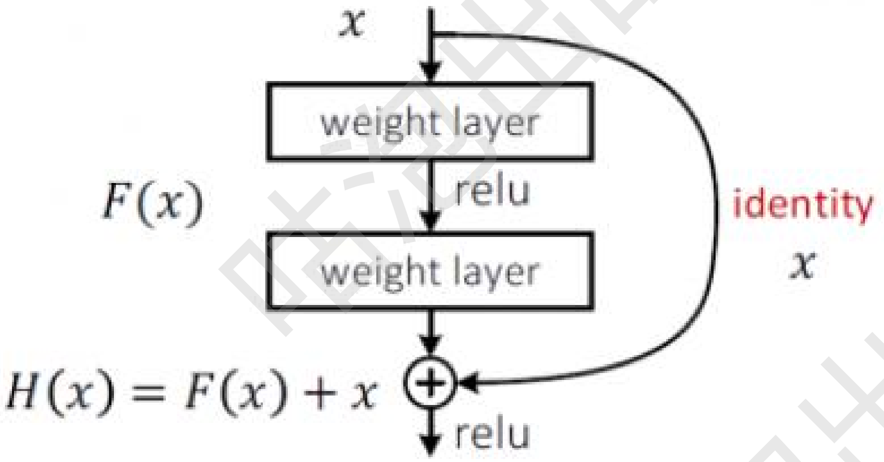
# 卷积神经网络：

## 经典网络-Resnet

### 深层网络遇到的问题：

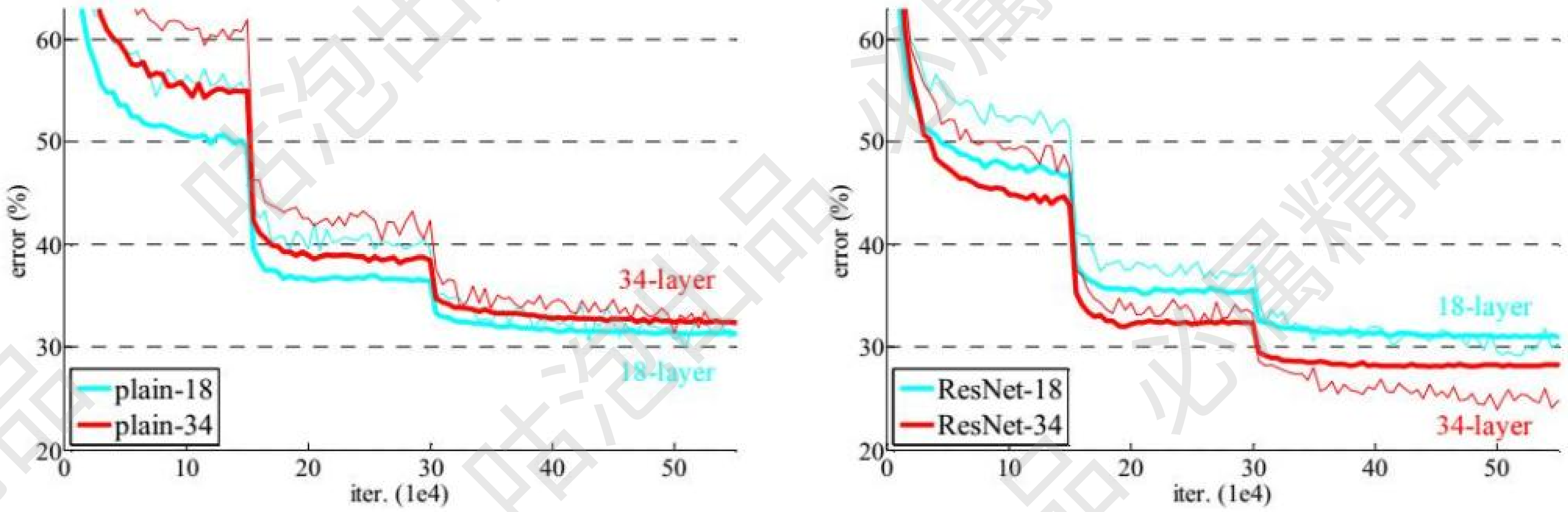



### 解决方案：



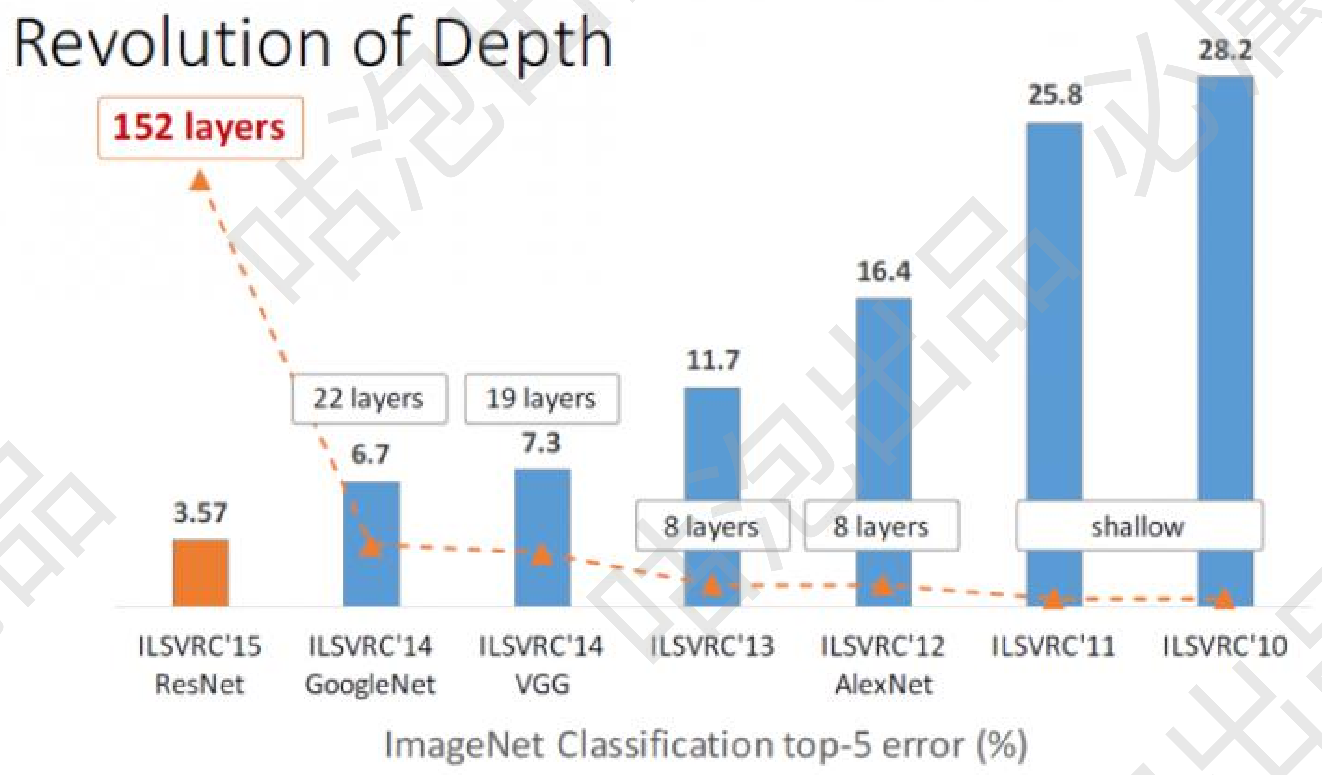
# 卷积神经网络：

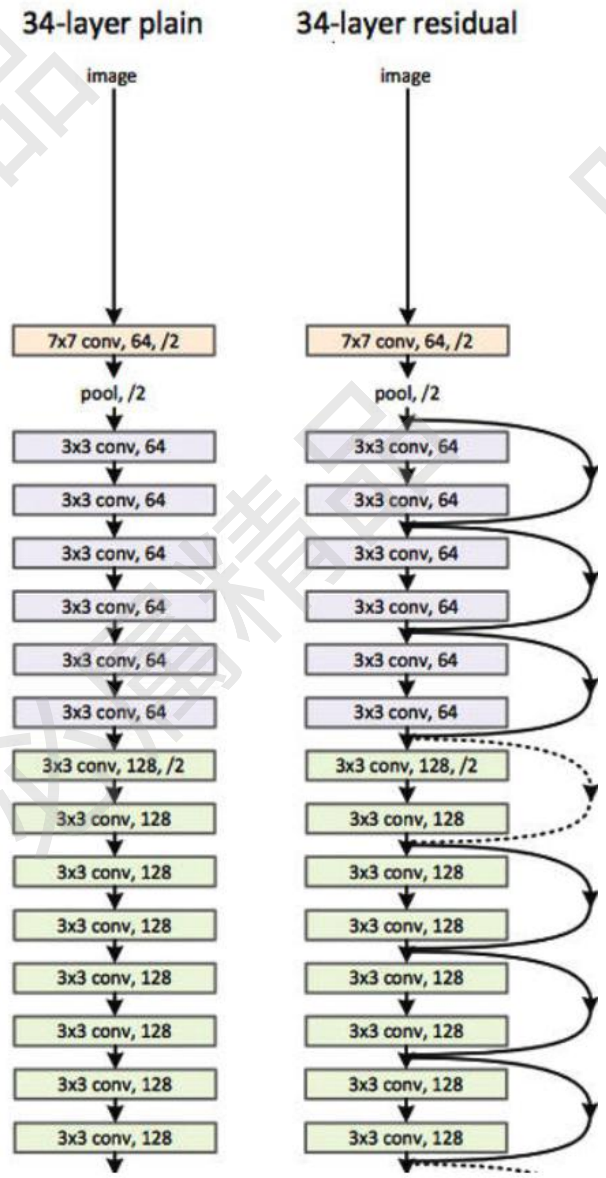
- 经典网络-Resnet



# 卷积神经网络：

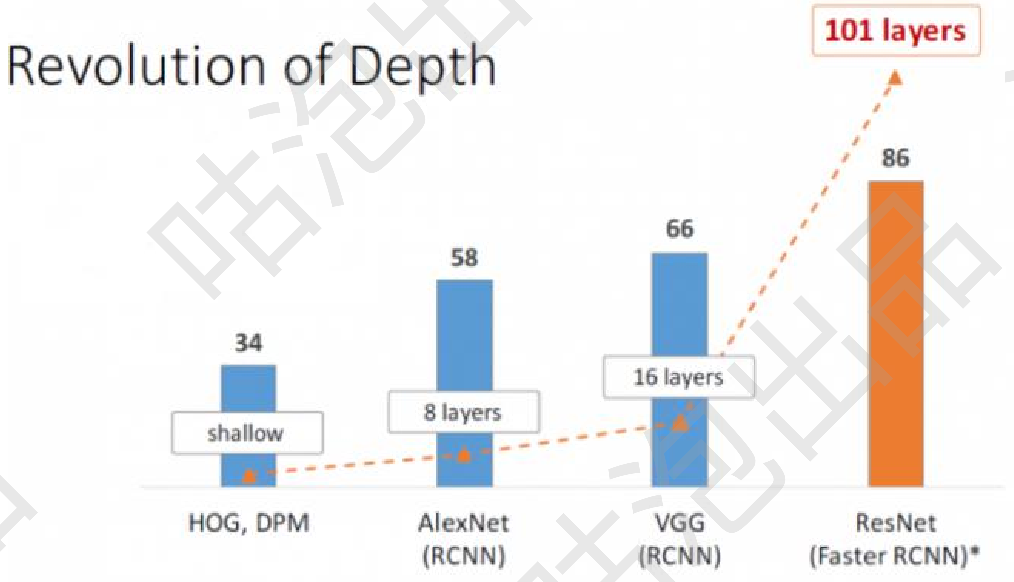
- 经典网络-Resnet



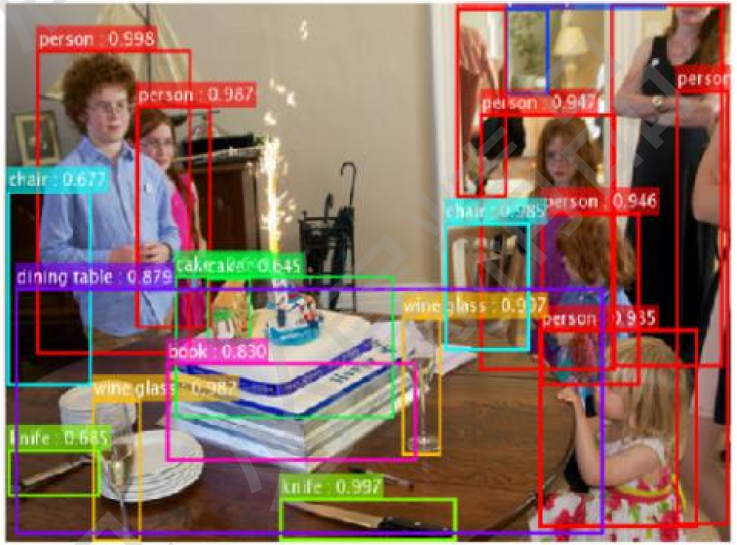



# 卷积神经网络：

## 经典网络-Resnet



PASCAL VOC 2007 **Object Detection** mAP (%)



ResNet's object detection result on COCO