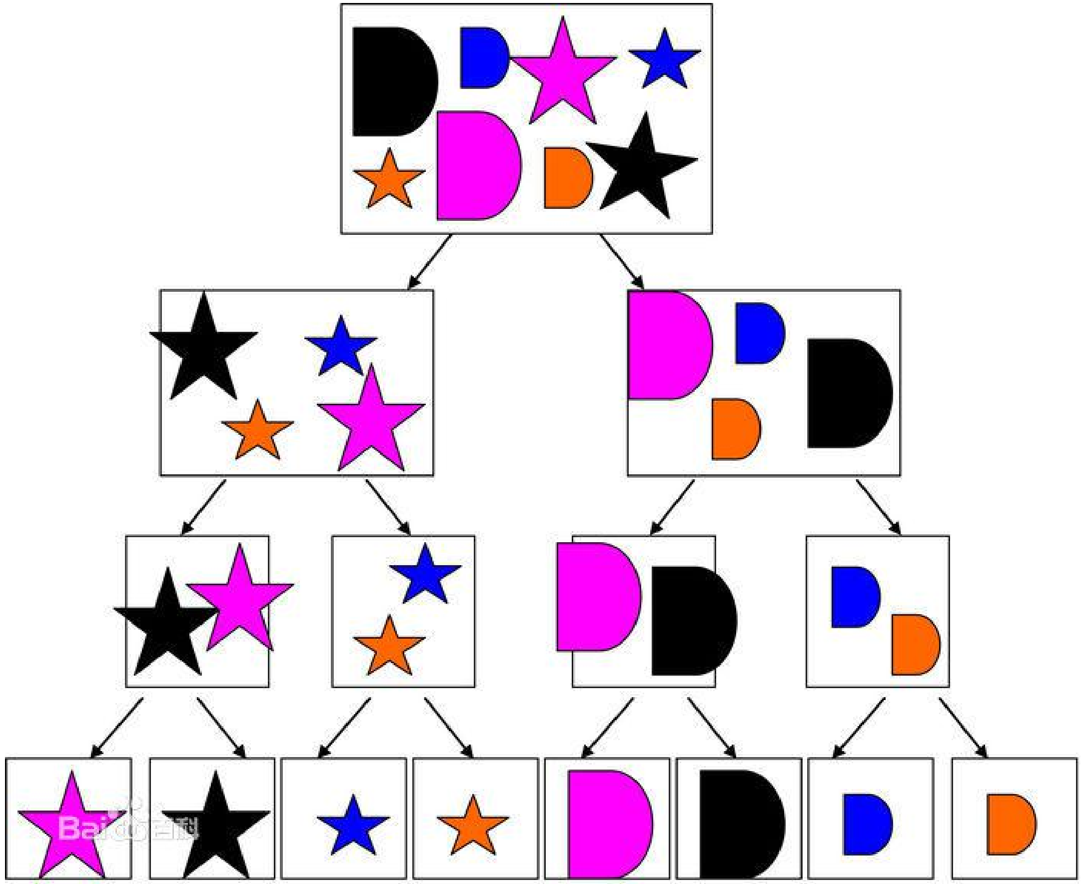
卷积神经网络

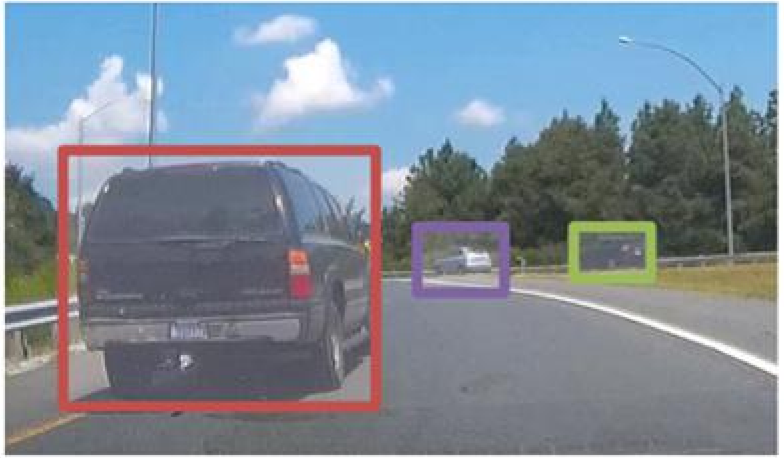
常见的视觉任务

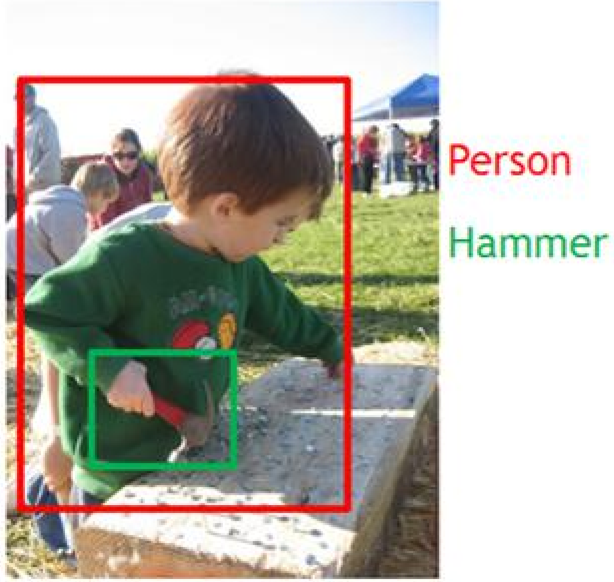
- 图像分类
- 目标检测
- 语义分割/实例分割
- 场景文字识别
- 图像生成
- 人体关键点检测
- 视频分类
- 度量学习



常见的视觉任务

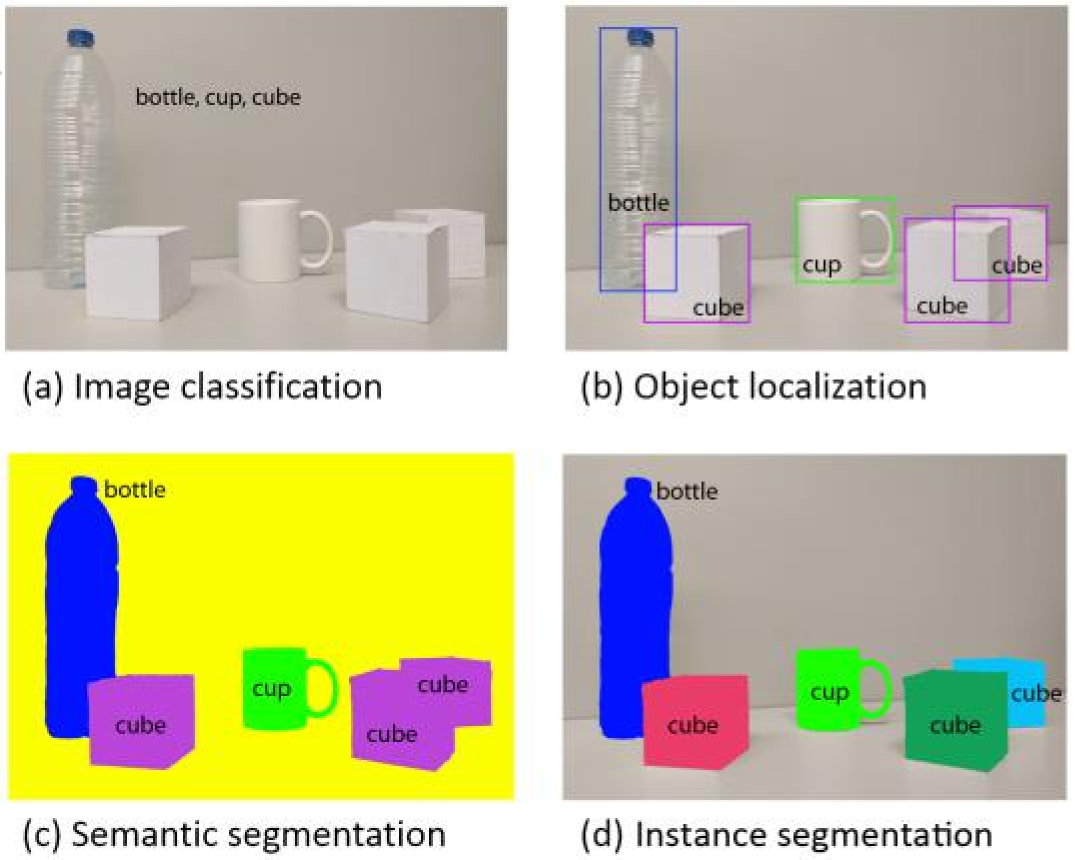
- 图像分类
- 目标检测
- 语义分割/实例分割
- 场景文字识别
- 图像生成
- 人体关键点检测
- 视频分类
- 度量学习





常见的视觉任务

- 图像分类
- 目标检测
- 语义分割/实例分割
- 场景文字识别
- 图像生成
- 人体关键点检测
- 视频分类
- 度量学习

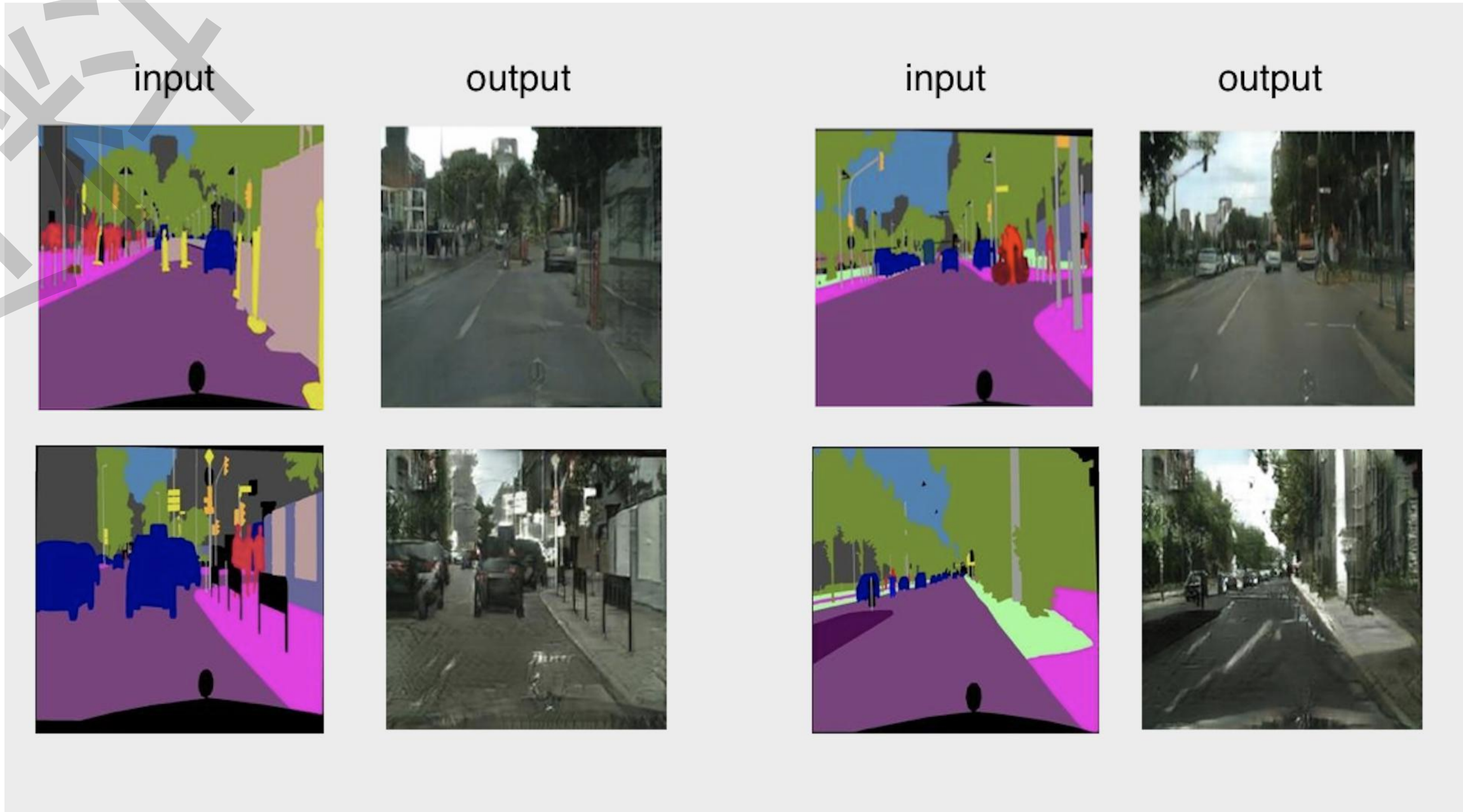


常见的视觉任务

- 图像分类
- 目标检测
- 语义分割/实例分割
- 场景文字识别
- 图像生成
- 人体关键点检测
- 视频分类
- 度量学习

常见的视觉任务

- 图像分类
- 目标检测
- 语义分割/实例分割
- 场景文字识别
- 图像生成
- 人体关键点检测
- 视频分类
- 度量学习



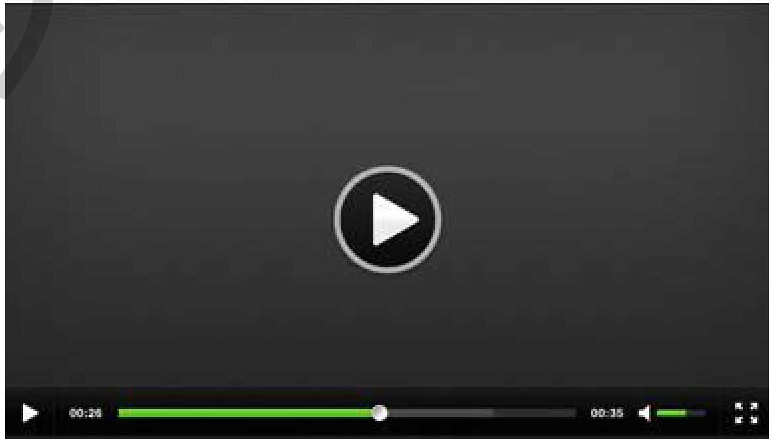
常见的视觉任务

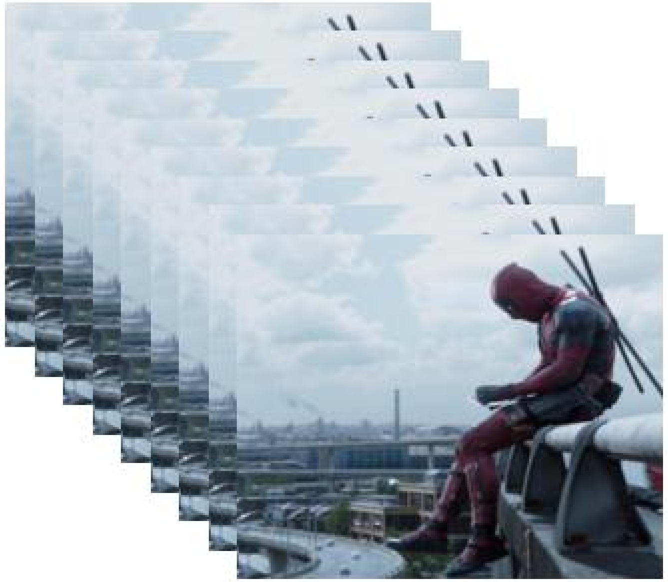
- 图像分类
- 目标检测
- 语义分割/实例分割
- 场景文字识别
- 图像生成
- 人体关键点检测
- 视频分类
- 度量学习

常见的视觉任务

百度智能云

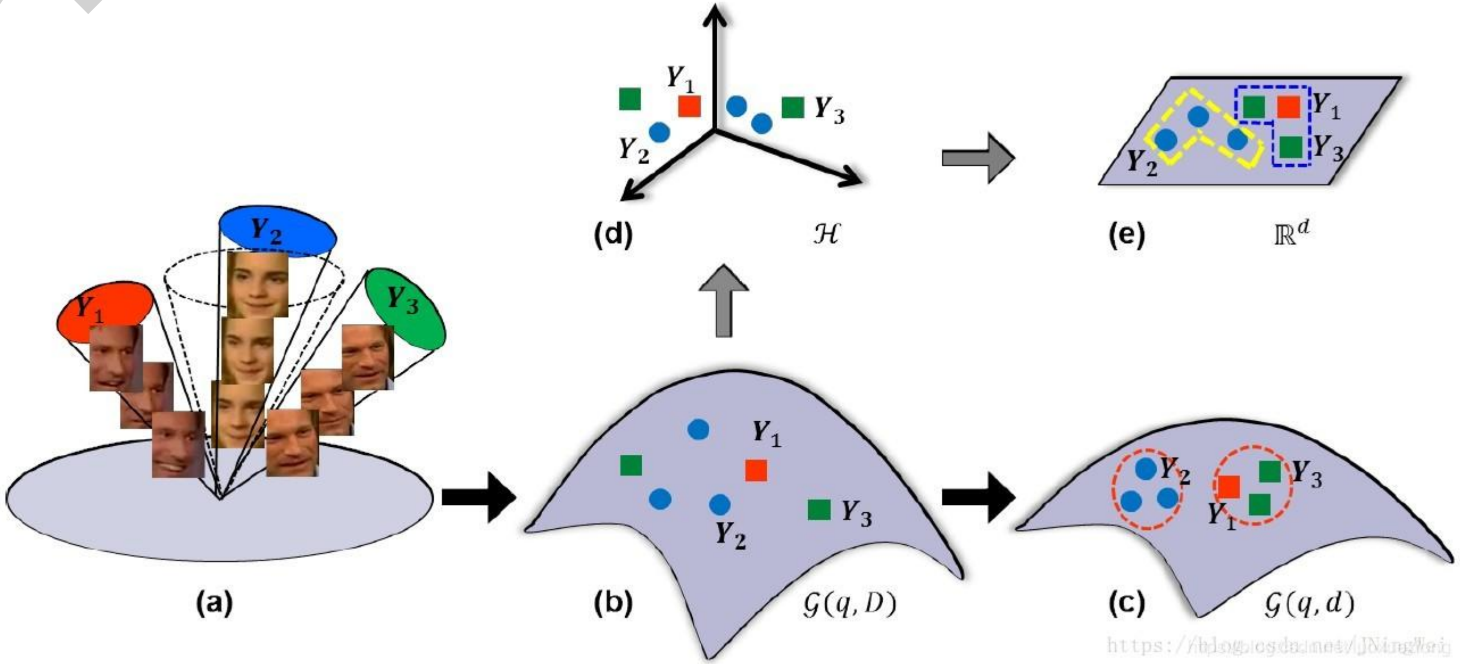
- 图像分类
- 目标检测
- 语义分割/实例分割
- 场景文字识别
- 图像生成
- 人体关键点检测
- 视频分类
- 度量学习





常见的视觉任务

- 图像分类
- 目标检测
- 语义分割/实例分割
- 场景文字识别
- 图像生成
- 人体关键点检测
- 视频分类
- 度量学习



使用卷积神经网络的原因和效果

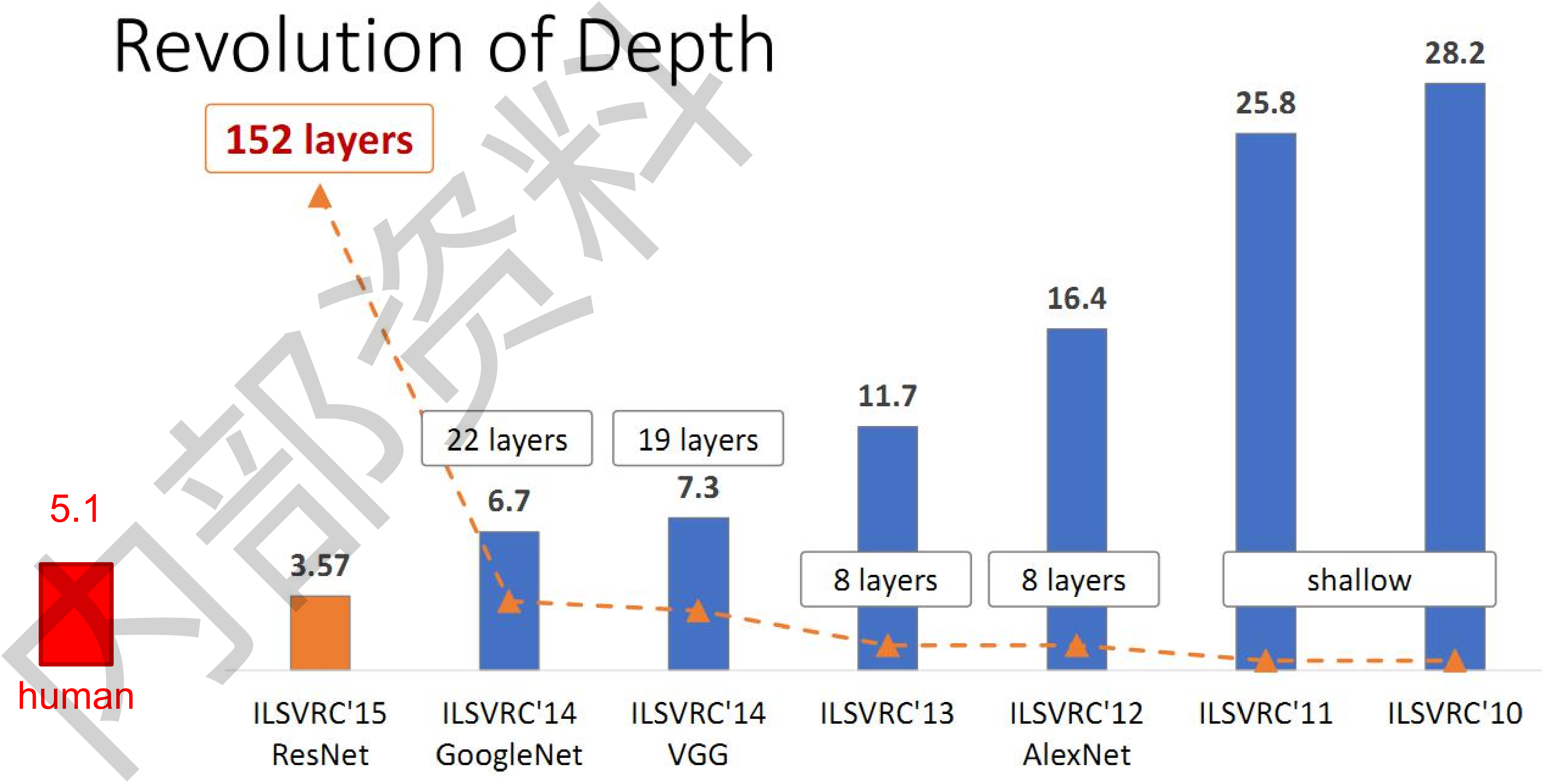
图像分类的样例

Response


```
{
    "log_id": "7275343294157017980",
    "result_num": 5,
    "result": [
        {
            "score": 0.860755,
            "root": "植物-果实/种子",
            "baike_info": [],
            "keyword": "蒲公英图片"
        },
        {
            "score": 0.669319,
            "root": "自然风景-天空",
            "keyword": "天空"
        },
        {
            "score": 0.488238,
            "root": "植物-菊科",
            "keyword": "蒲公英"
        },
        {
            "score": 0.294248,
```

图像识别技术的发展

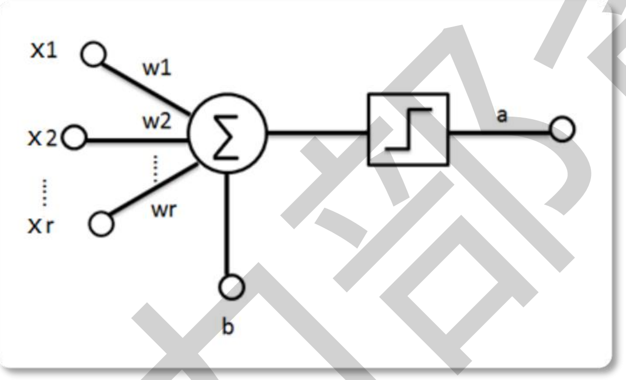
百度智能云



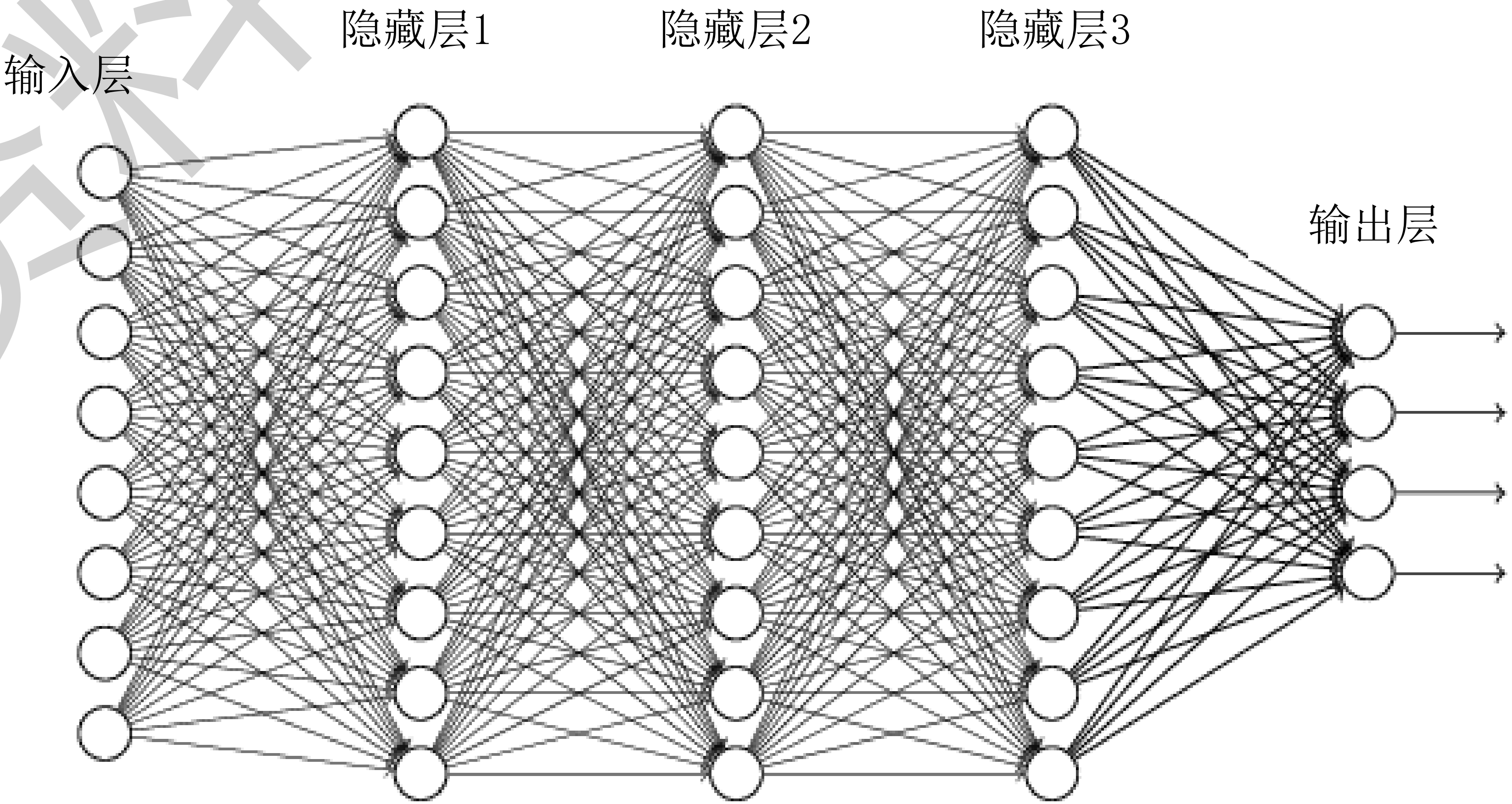
ImageNet 图像分类错误率（top-5）

回忆一下神经网络结构

来一张神经网络：



单个神经元

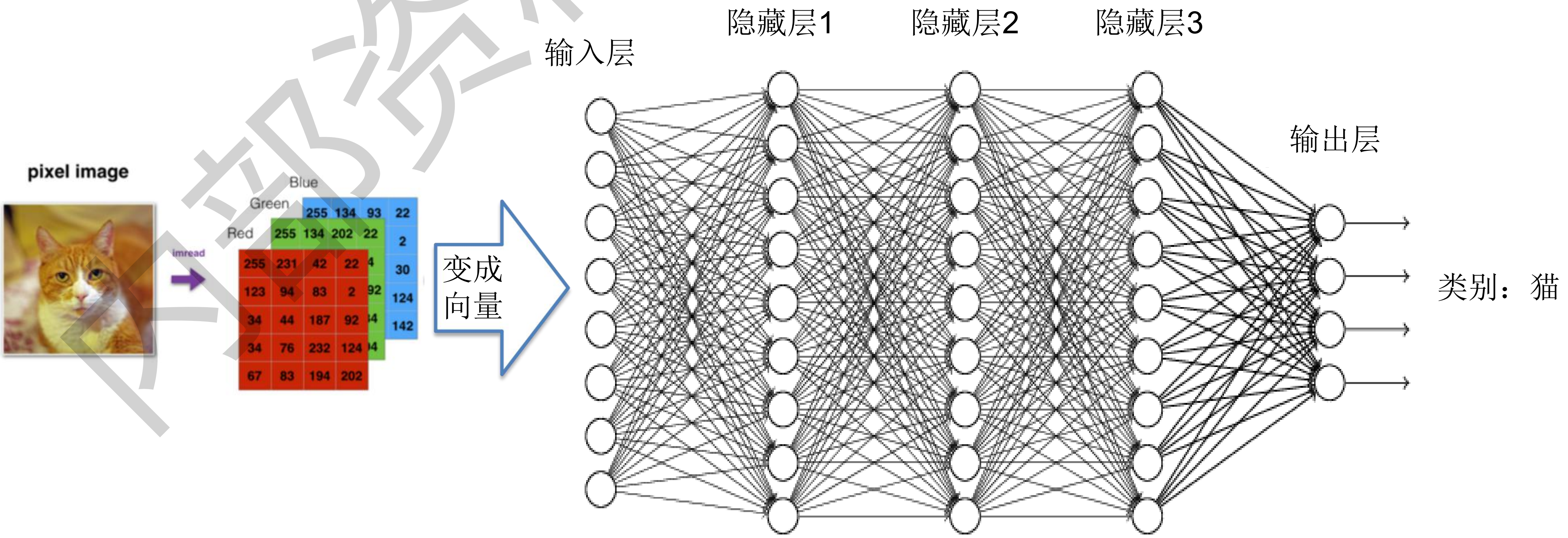


卷积神经网络与计算机视觉

- 视觉类任务大部分都需要使用 CNN 技术

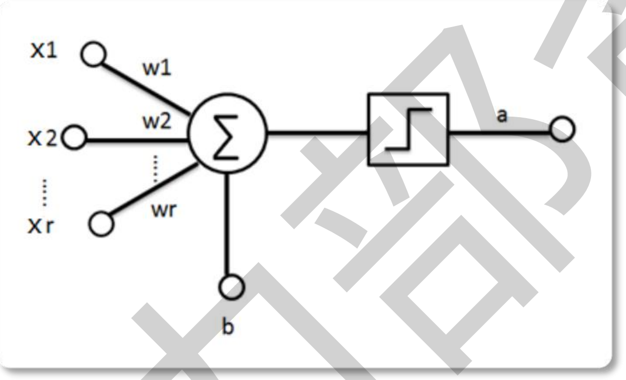
思考：为什么必须使用 CNN 呢？直接使用 DNN 可以吗？

维度灾难



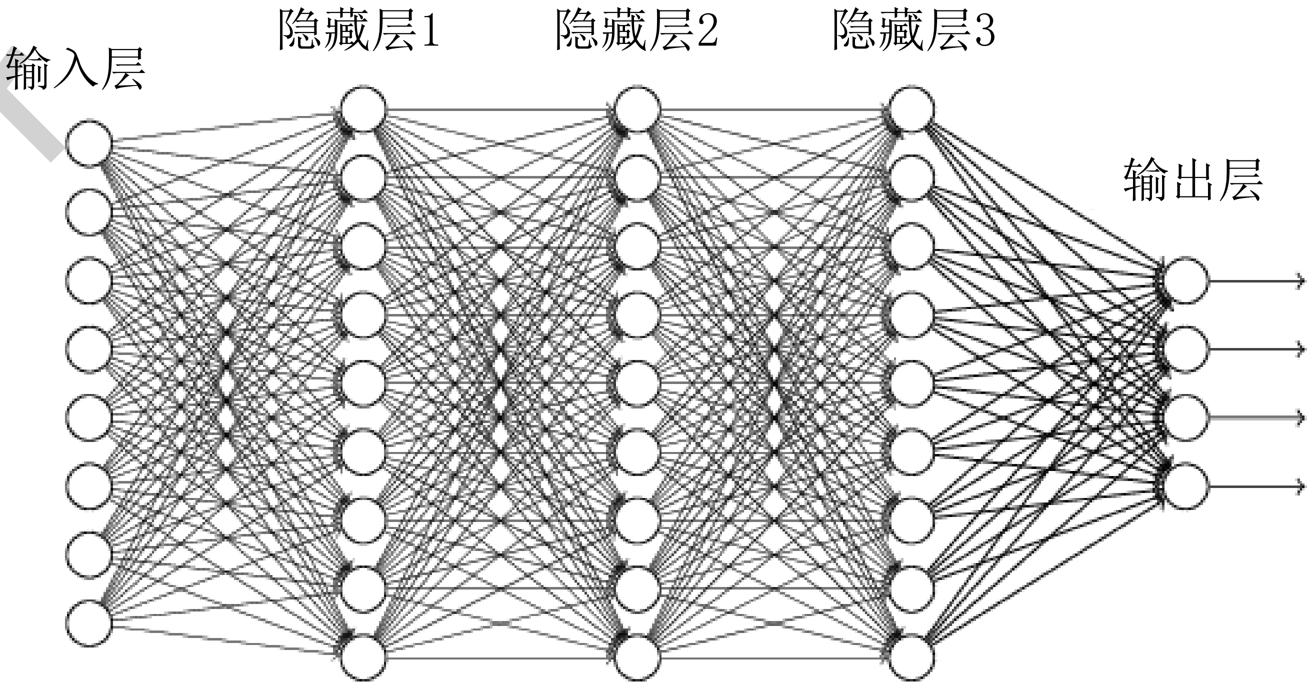
全连接神经网络：维度灾难

- 以200x200图像为例
  - 全连接网络
    - 输入层：40000维
    - 隐藏层：400000 神经元
    - 参数数目：160亿参数



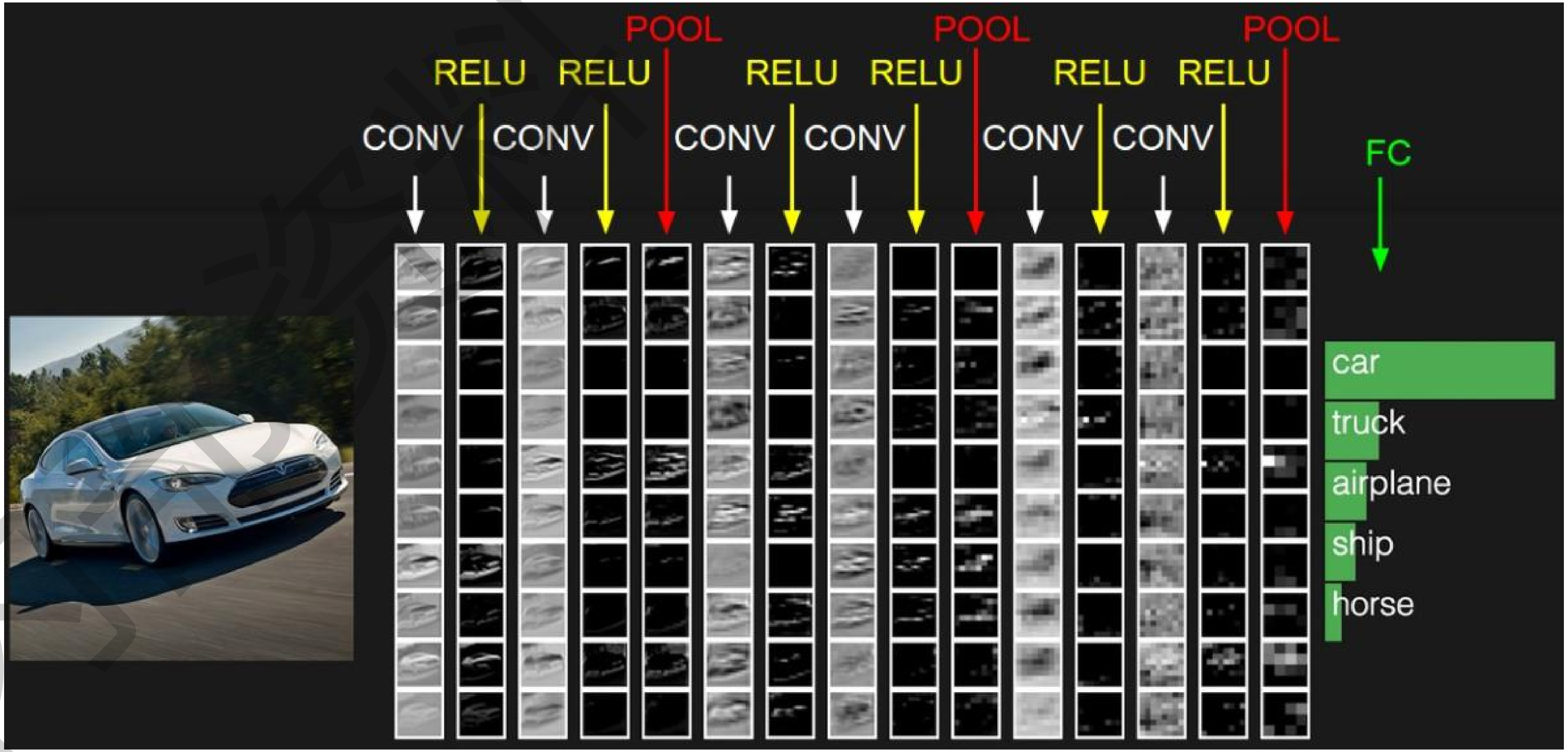
单个神经元

f(**wx**+b)



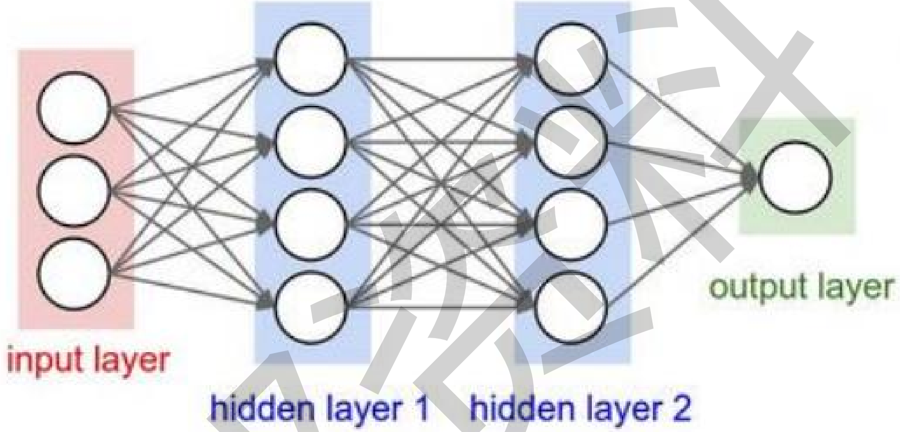
内存消耗巨大
计算量巨大
训练时间过长

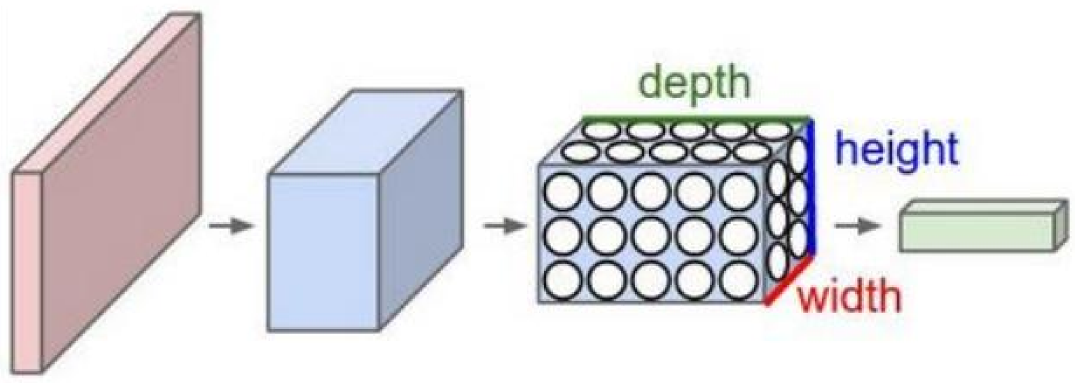
来个感性认识：CNN网络整体结构



使用卷积操作提取特征

CNN网络整体结构





[INPUT - CONV - RELU - POOL - FC]

输入层

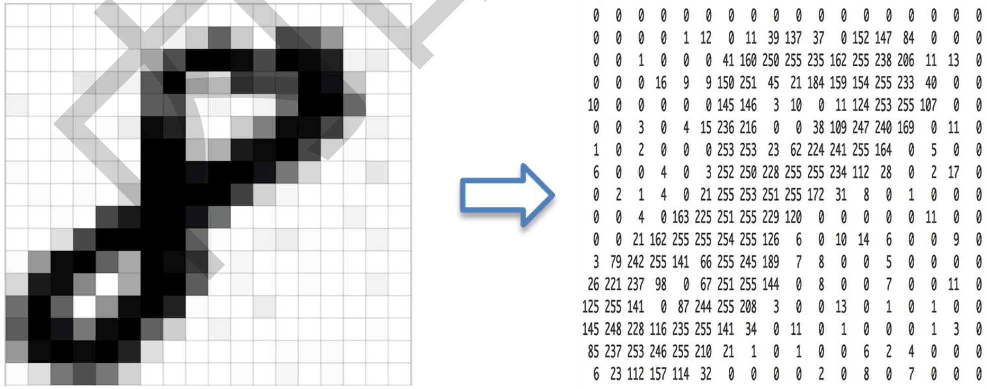
卷积层

激活函数

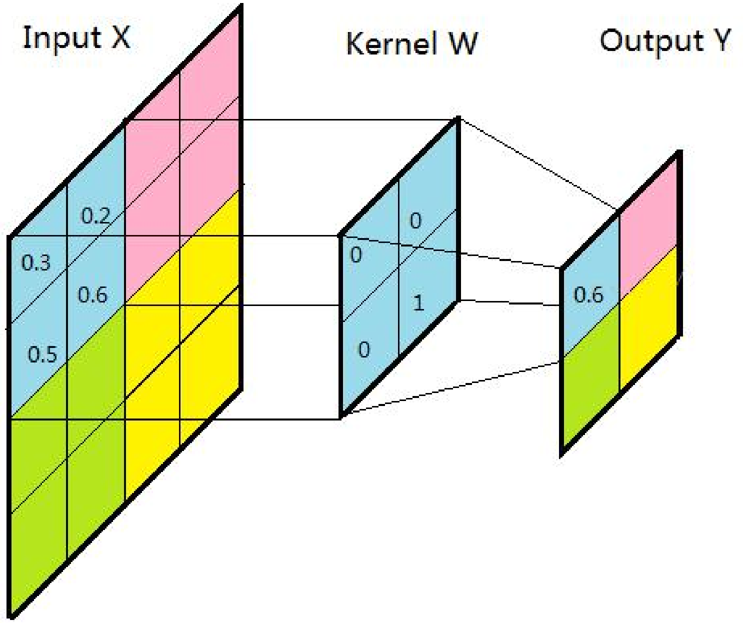
池化层

全连接层

Conv层（卷积层）

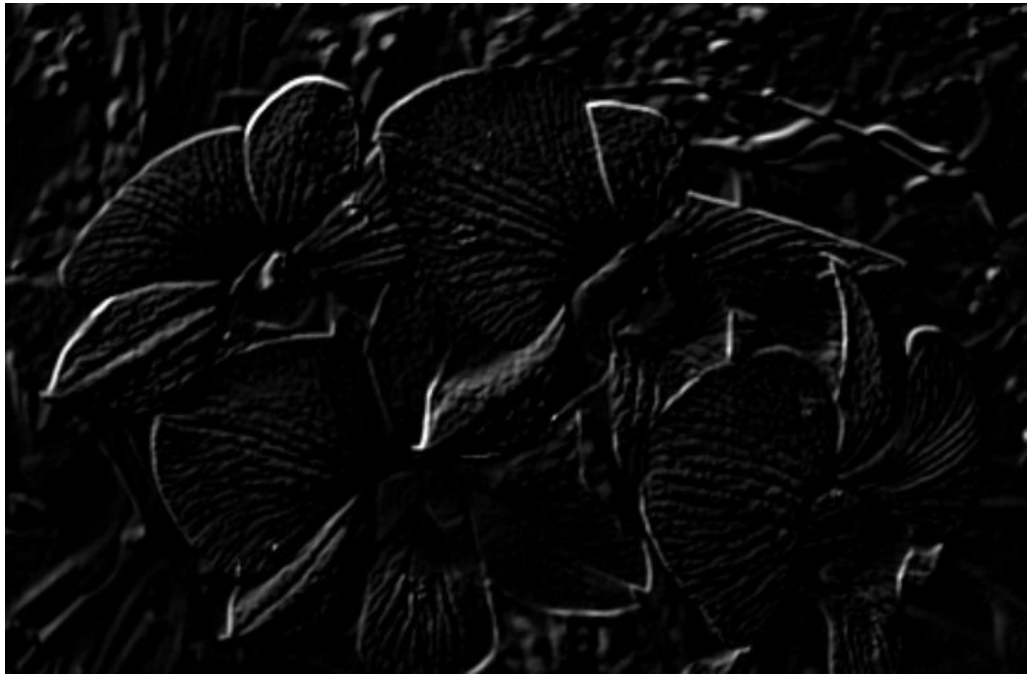


二维卷积适用于图片存储矩阵



敲黑板：
二维卷积核相当于计算窗口
在窗口内做两个向量的内积

图片经卷积后效果



原图（灰度图）

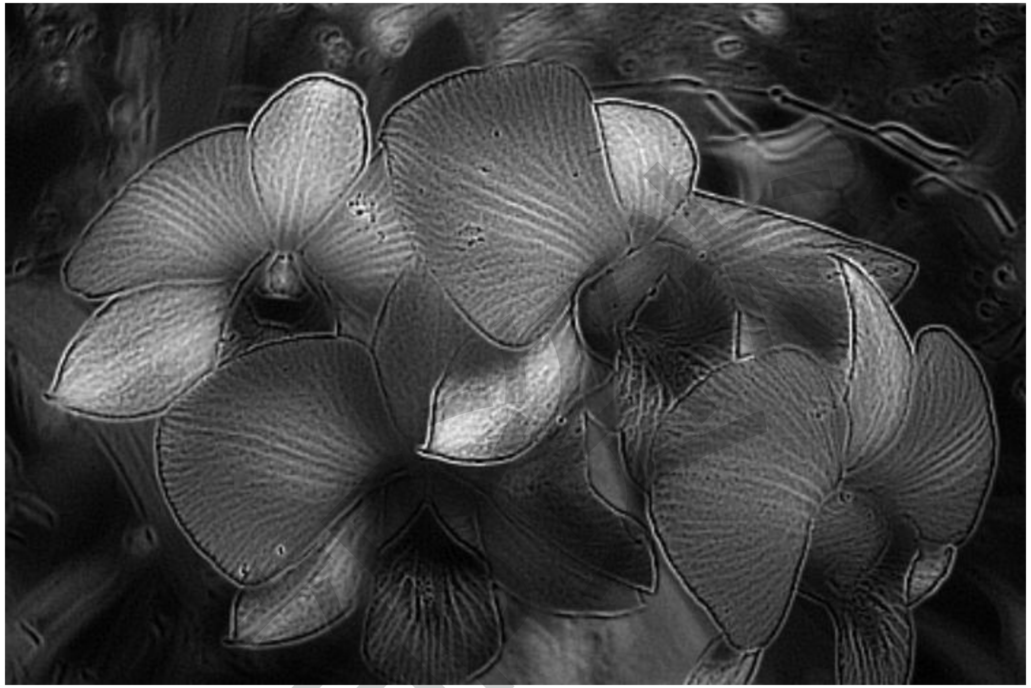
灰度图是一个二维矩阵，每一个像素值都是 0-255 的整数

0 代表黑色，255 代表白色

-1	-1	0
-1	0	1
0	1	1

浮雕

图片经卷积后效果



1	1	1
1	-7	1
1	1	1

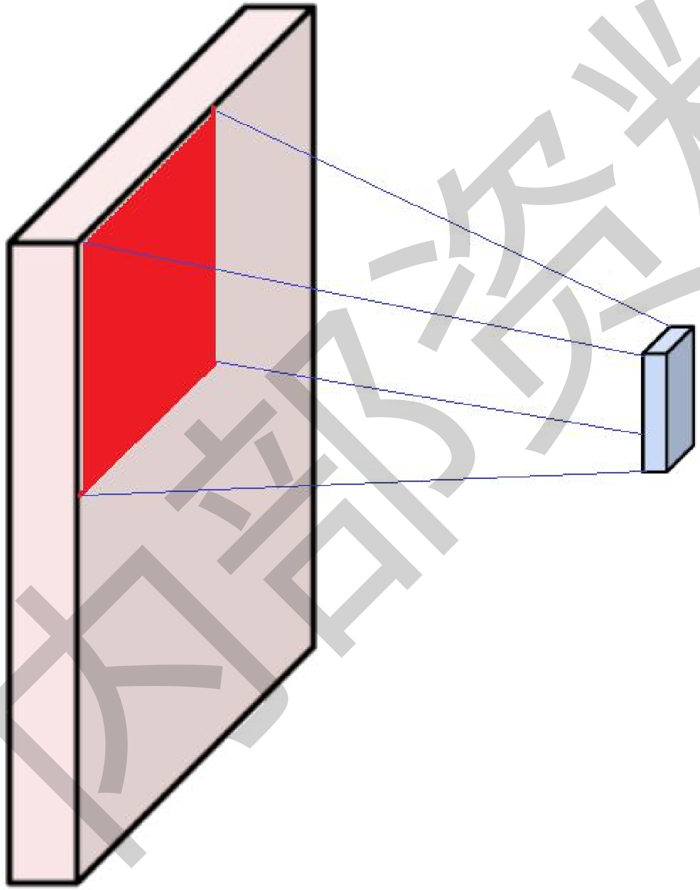
边缘

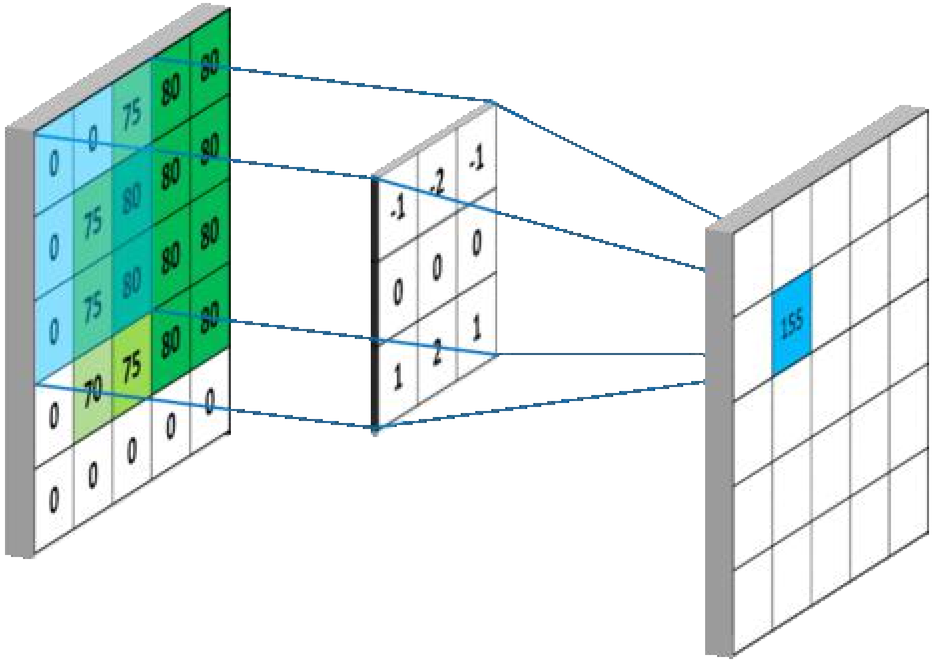
-1	-1	-1
-1	8	-1
-1	-1	-1

边缘

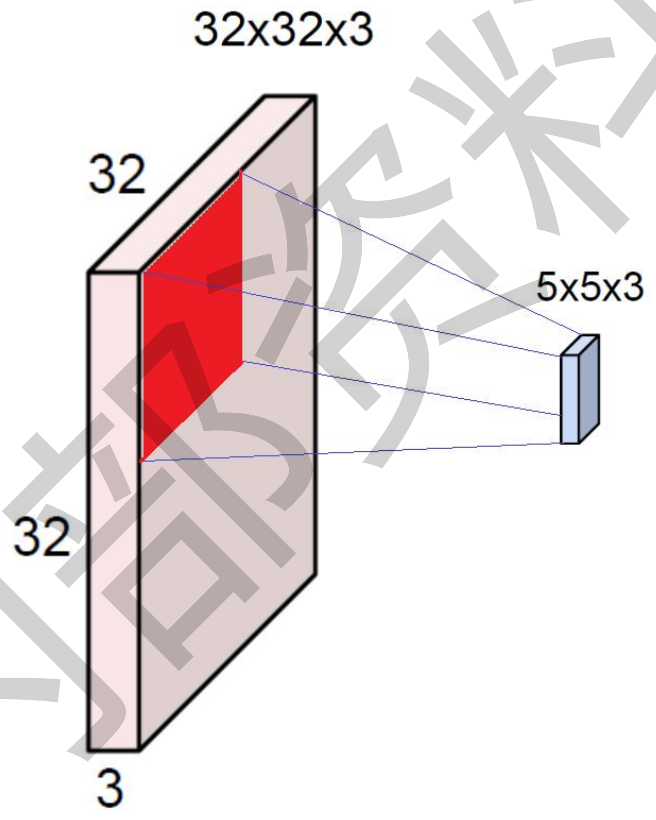
在线 Flash 示例：
https://graphics.stanford.edu/courses/cs178/applets/convolution.html

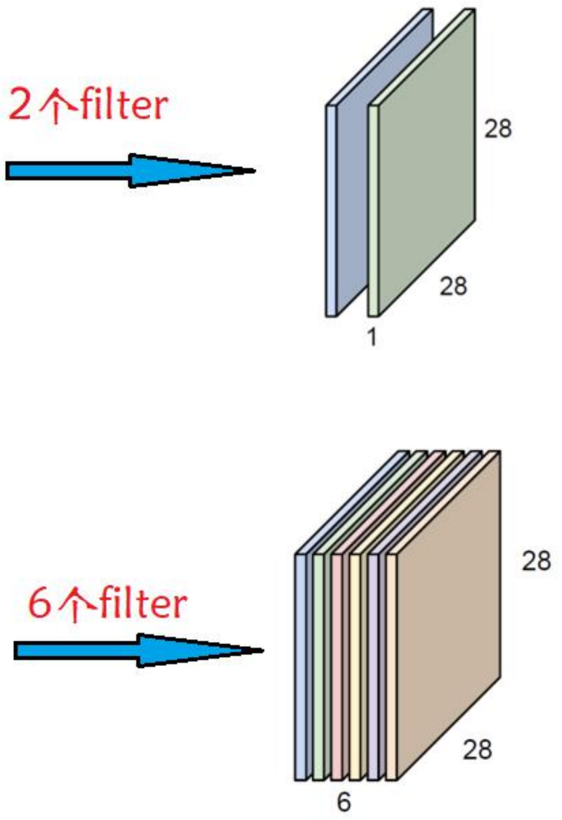
卷积层的具体工作过程





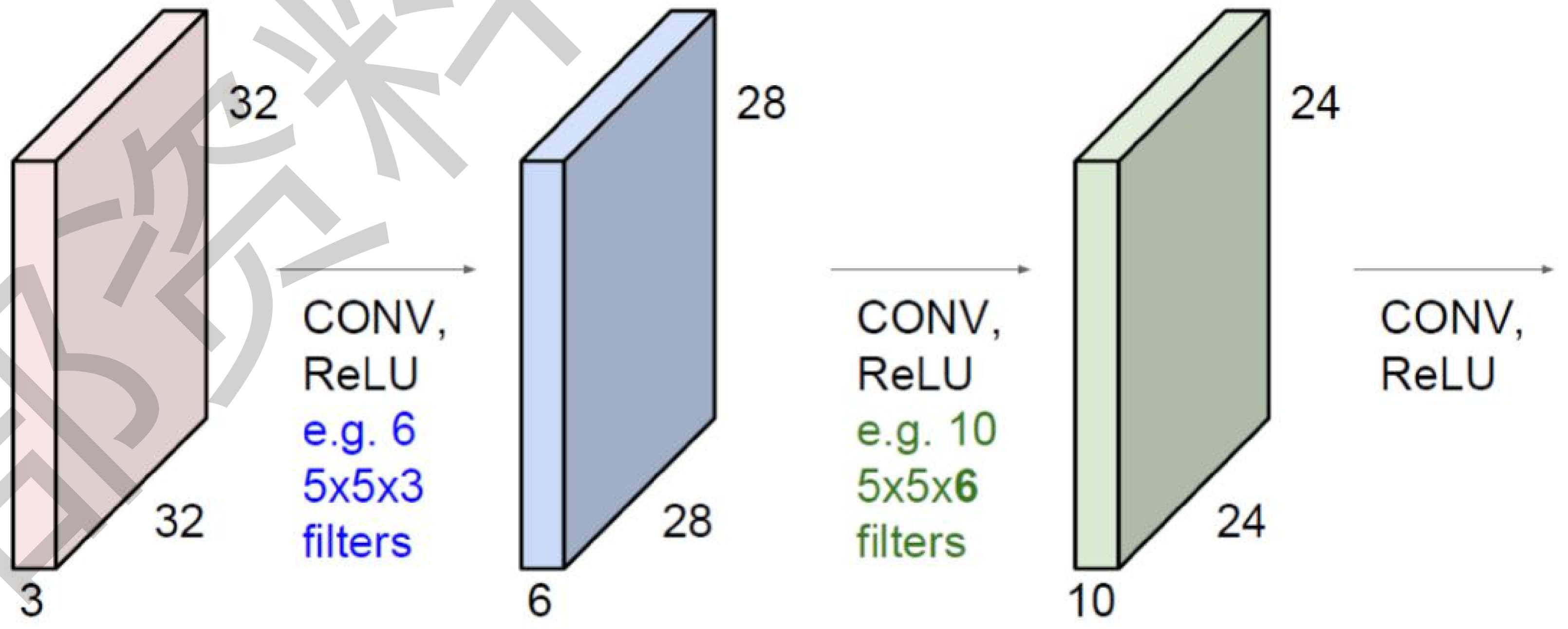
卷积层的具体工作过程





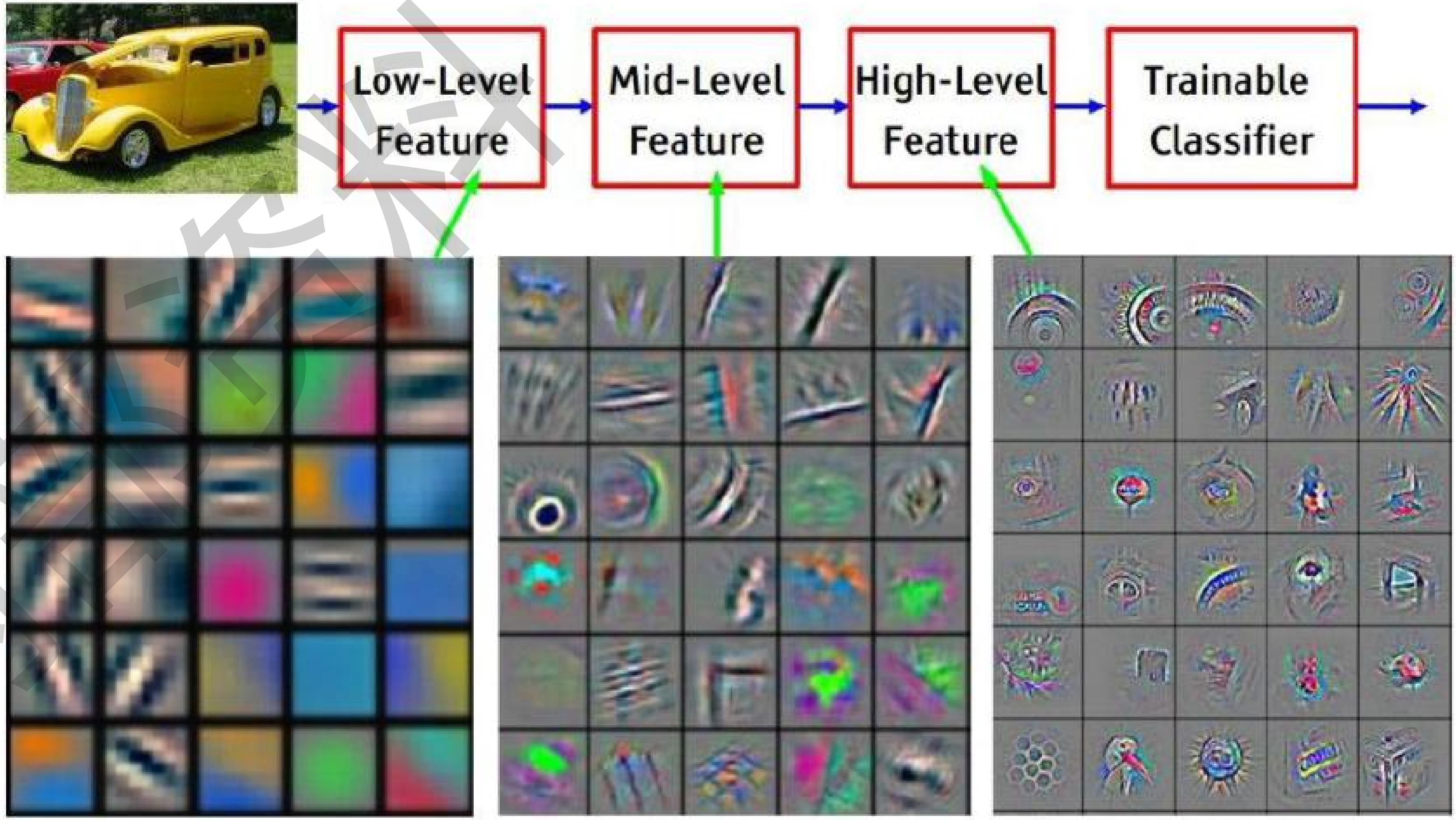
特征图

多个卷积层



32
32
3
CONV,
ReLU
e.g. 6
5x5x3
filters
28
28
6
CONV,
ReLU
e.g. 10
5x5x6
filters
24
24
10
CONV,
ReLU

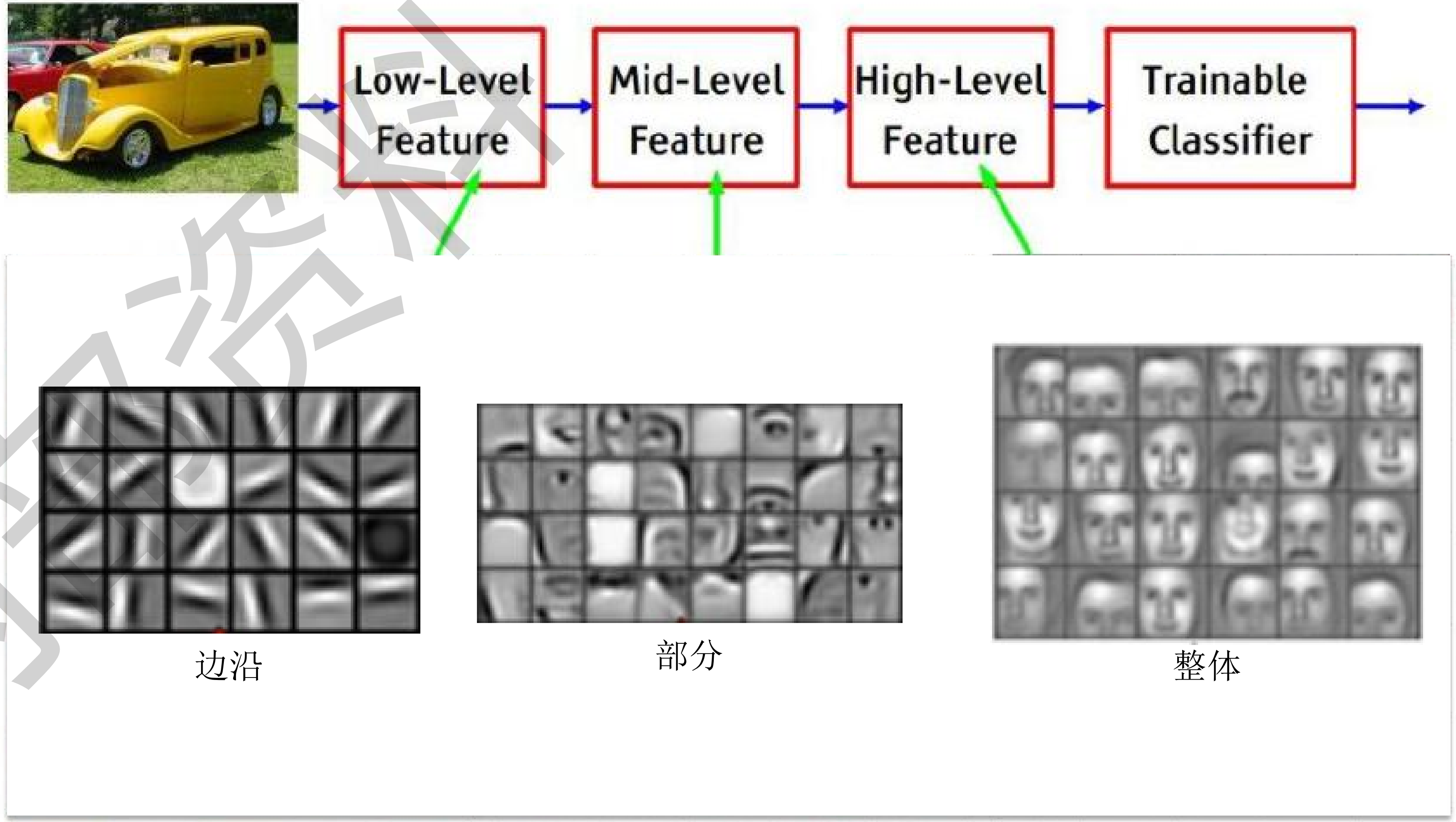
通过卷积操作提取特征



Low-Level Feature
Mid-Level Feature
High-Level Feature
Trainable Classifier

通过卷积操作提取特征

百度智能云

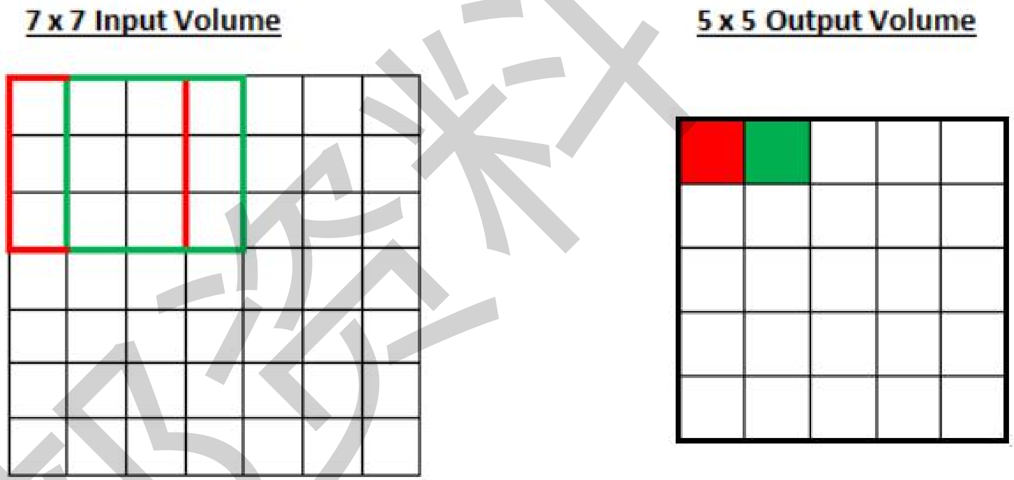


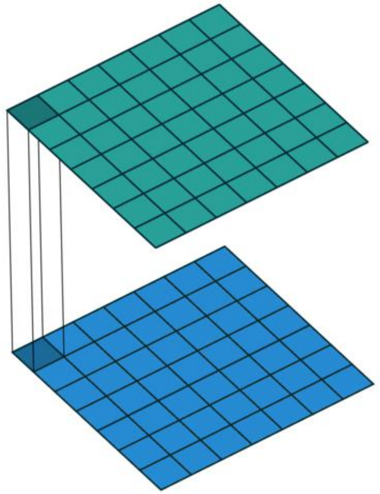
卷积核的分析与计算

卷积操作的参数——步长

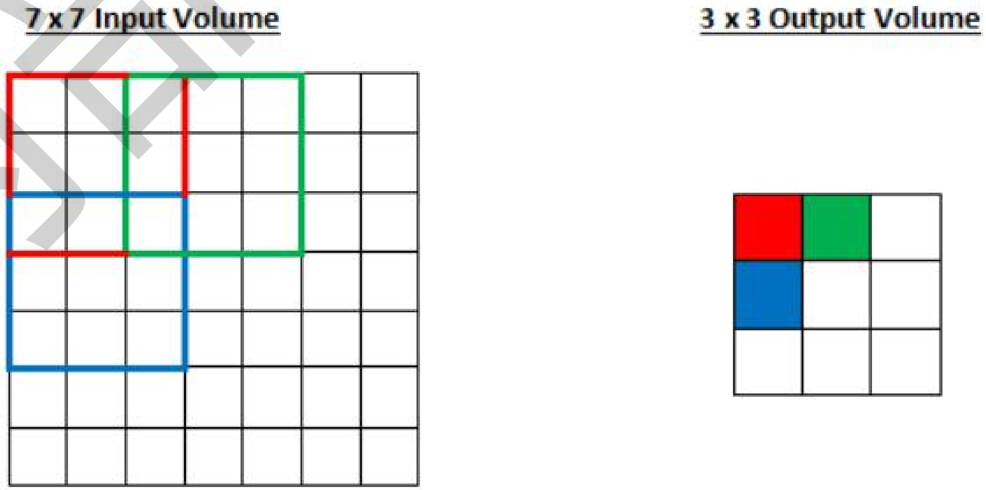
步长stride：反映了filter滑动一次的距离

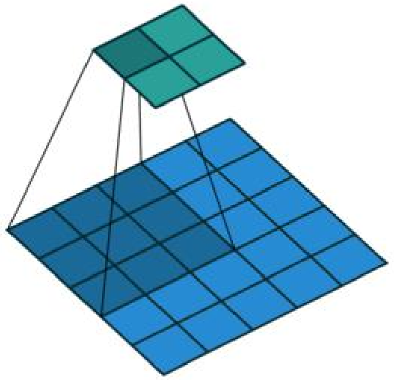
步长为1





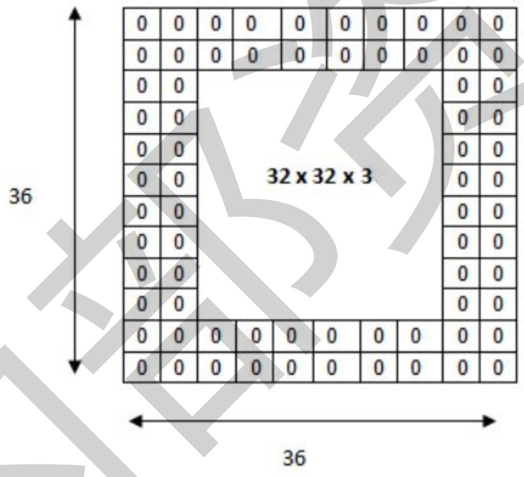
步长为2

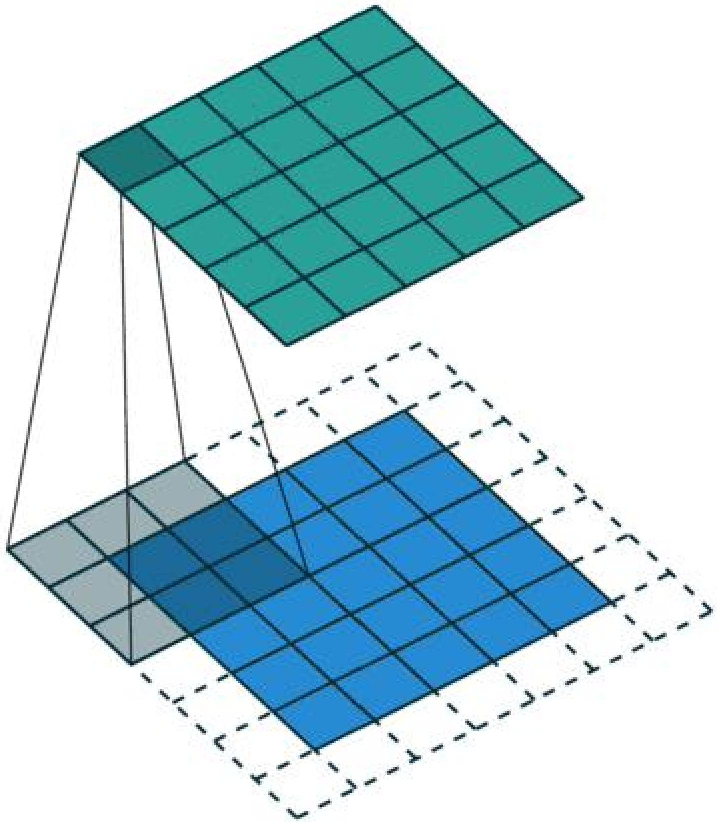




卷积操作的参数——填充

Padding：边界填充





特征图体积的计算

输入大小为：W1 x H1 x D1

需要指定的超参数：filter个数（K），filter大小（F），步长（S），边界填充（P）

输出：

$$W_2 = (W_1 - F + 2P)/S + 1$$
$$H_2 = (H_1 - F + 2P)/S + 1$$
$$D_2 = K$$

一个例子：

输入：**32x32x3**

filters：5x5 **10**个

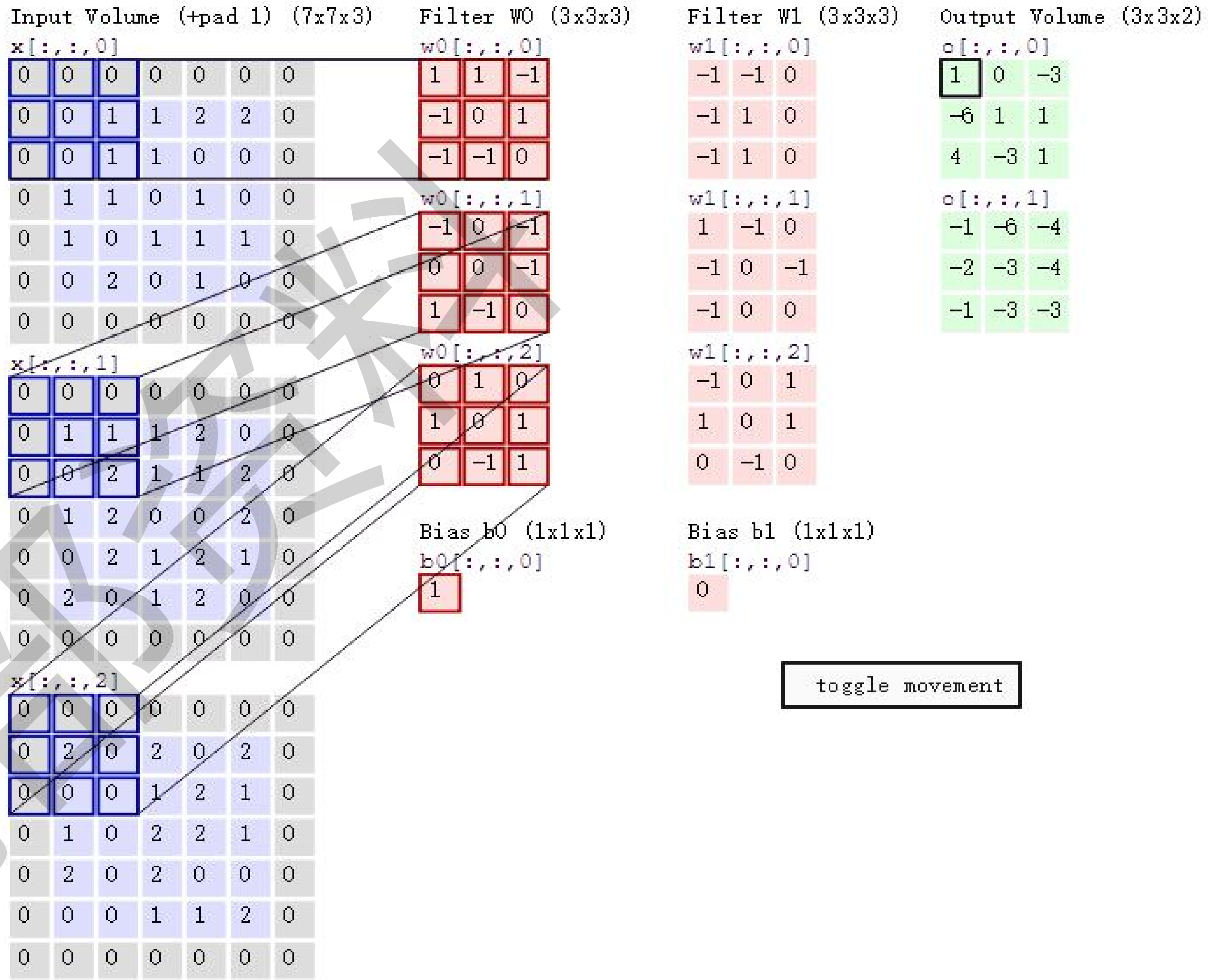
stride 1, pad 2

输出：

(32+2*2-5)/1+1 = 32

32x32x10

百度智能云


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

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	1	1	2	2	0
0	0	1	1	0	0	0
0	1	1	0	1	0	0
0	1	0	1	1	1	0
0	0	2	0	1	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	1	1	1	2	0	0
0	0	2	1	1	2	0
0	1	2	0	0	2	0
0	0	2	1	2	1	0
0	2	0	1	2	0	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	0	2	0	2	0
0	0	0	1	2	1	0
0	1	0	2	2	1	0
0	2	0	2	0	0	0
0	0	0	1	1	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

1	1	-1
-1	0	1
-1	-1	0

w0[:,:,1]

-1	0	-1
0	0	-1
1	-1	0

w0[:,:,2]

0	1	0
1	0	1
0	-1	1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

-1	-1	0
-1	1	0
-1	1	0

w1[:,:,1]

1	-1	0
-1	0	-1
-1	0	0

w1[:,:,2]

-1	0	1
1	0	1
0	-1	0

Bias b1 (1x1x1)

b1[:,:,0]

0

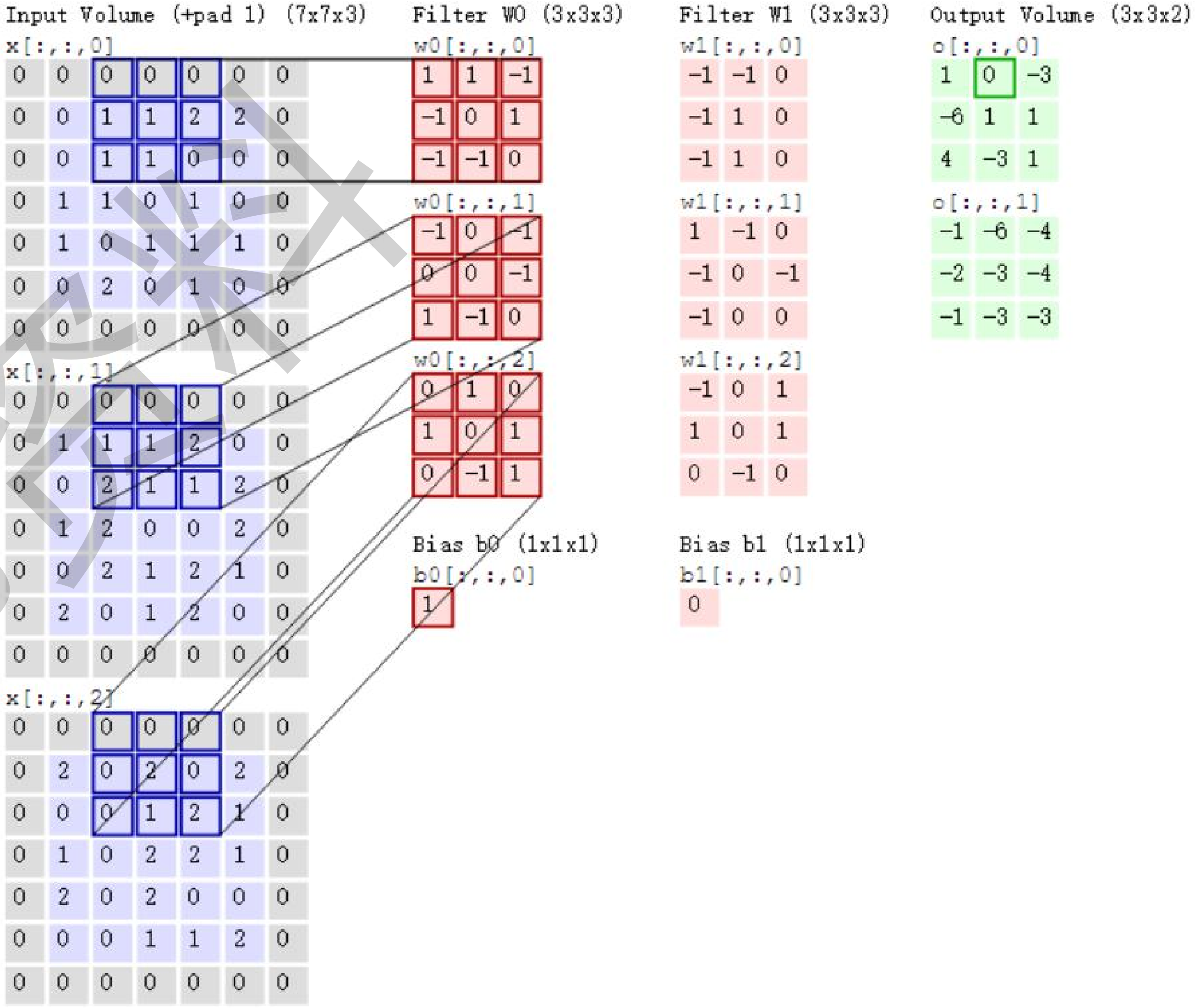
Output Volume (3x3x2)

o[:,:,0]

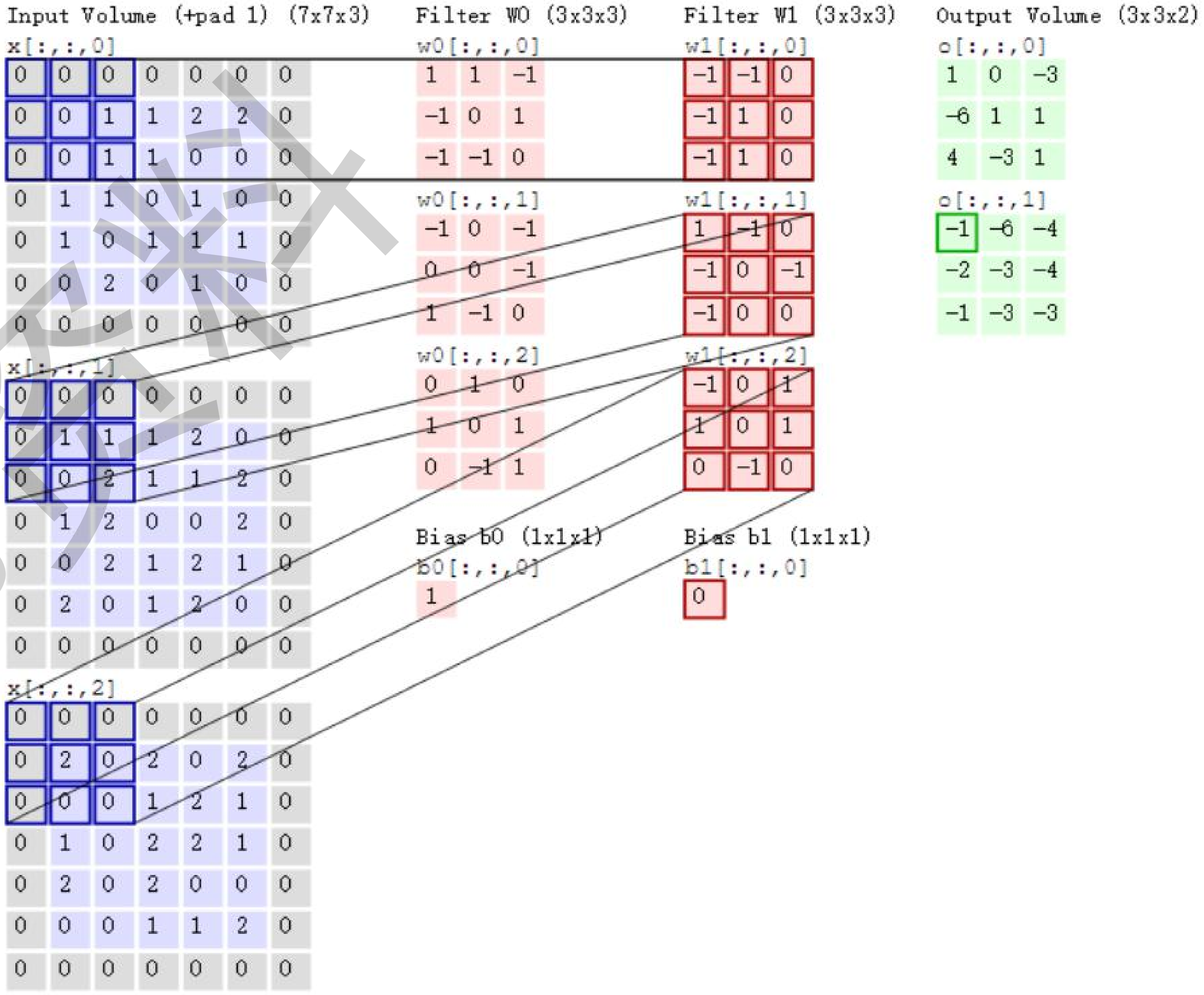
1	0	-3
-6	1	1
4	-3	1

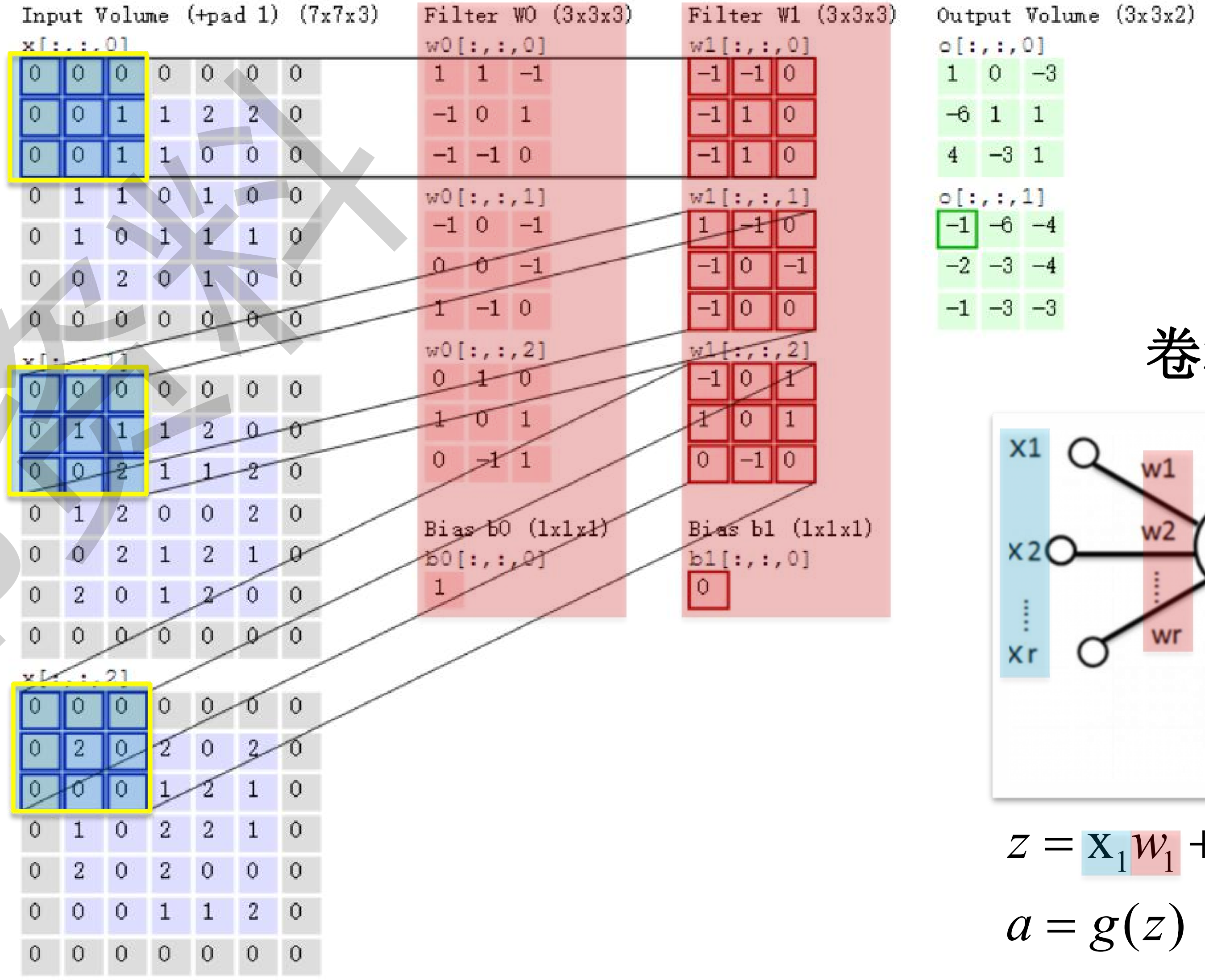
o[:,:,1]

-1	-6	-4
-2	-3	-4
-1	-3	-3

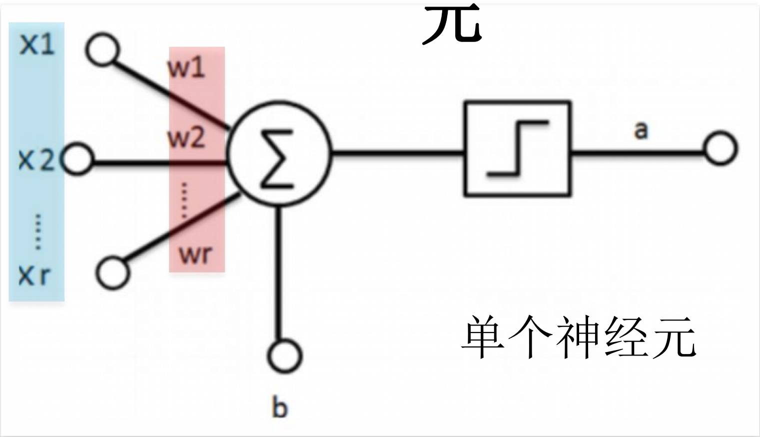

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 1 1 2 2 0
0 0 1 1 0 0 0
0 1 1 0 1 0 0
0 1 0 1 1 1 0
0 0 2 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 1 1 2 0 0
0 0 2 1 1 2 0
0 1 2 0 0 2 0
0 0 2 1 2 1 0
0 2 0 1 2 0 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 2 0 2 0 2 0
0 0 0 1 2 1 0
0 1 0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

百度智能云


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



卷积核 = 神经元

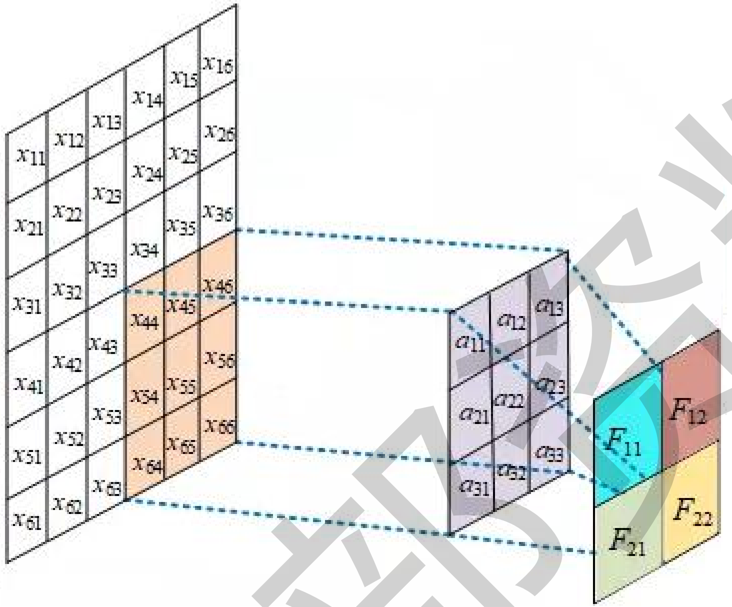


$$z = x_1 w_1 + x_2 w_2 + \ldots + x_r w_r + b$$

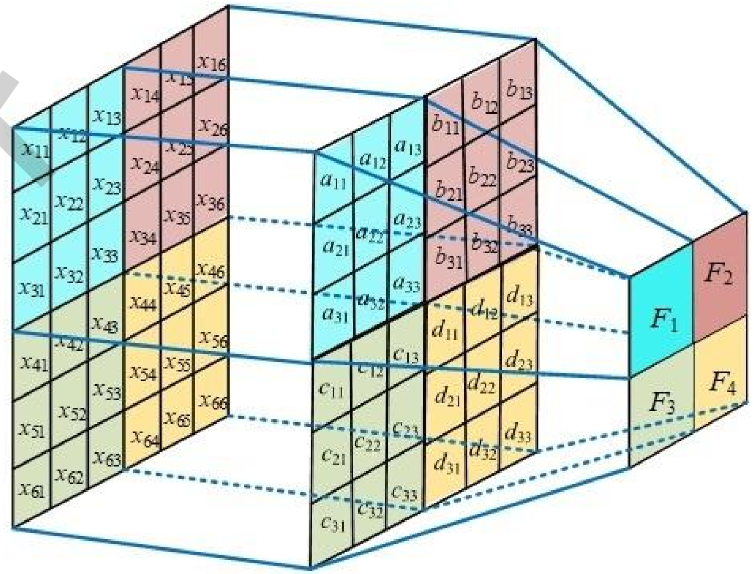
$$a = g(z)$$

局部连接　　　　　　权值共享

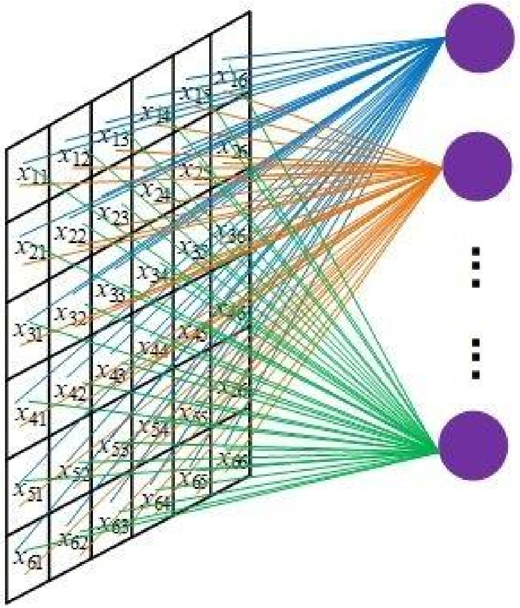
权值共享



权值共享



局部连接



全连接

权值共享极大的减少了参数的数量
大大提高了计算速度，减少内存消耗

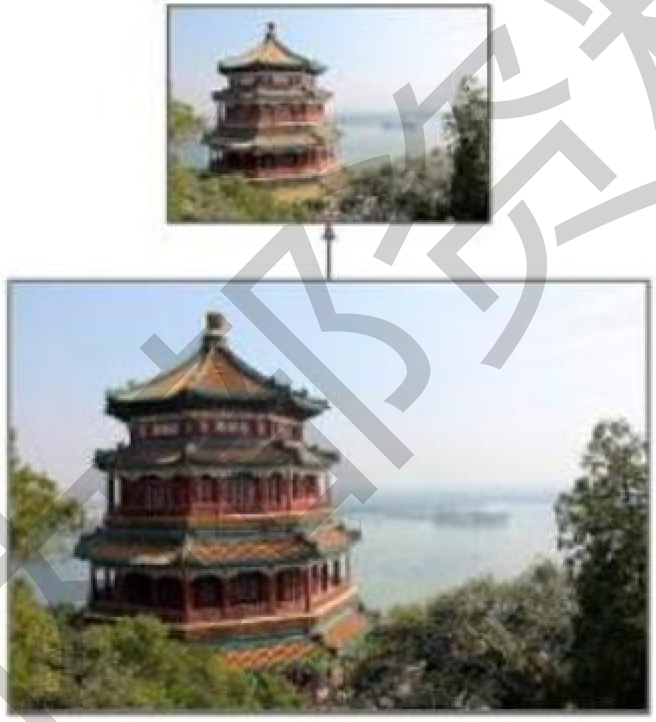
输入：32x32x3

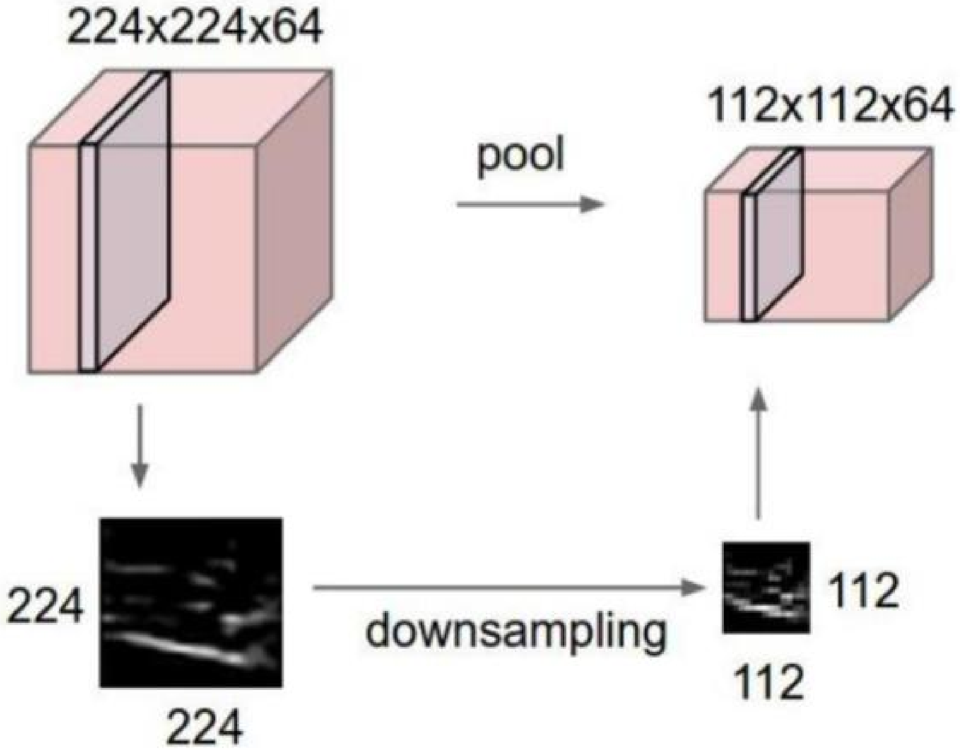
filters：5x5 10个

池化操作

池化

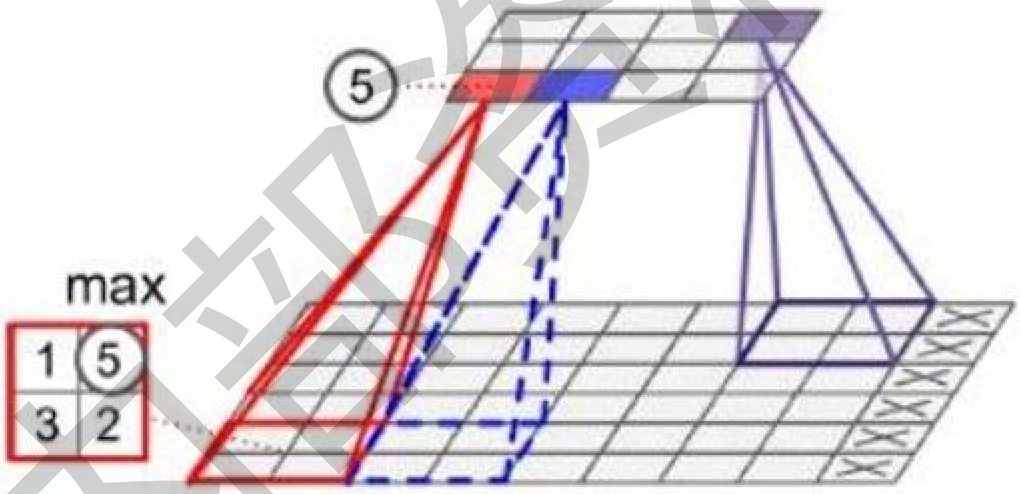
- 使得原始图片的尺寸变小了

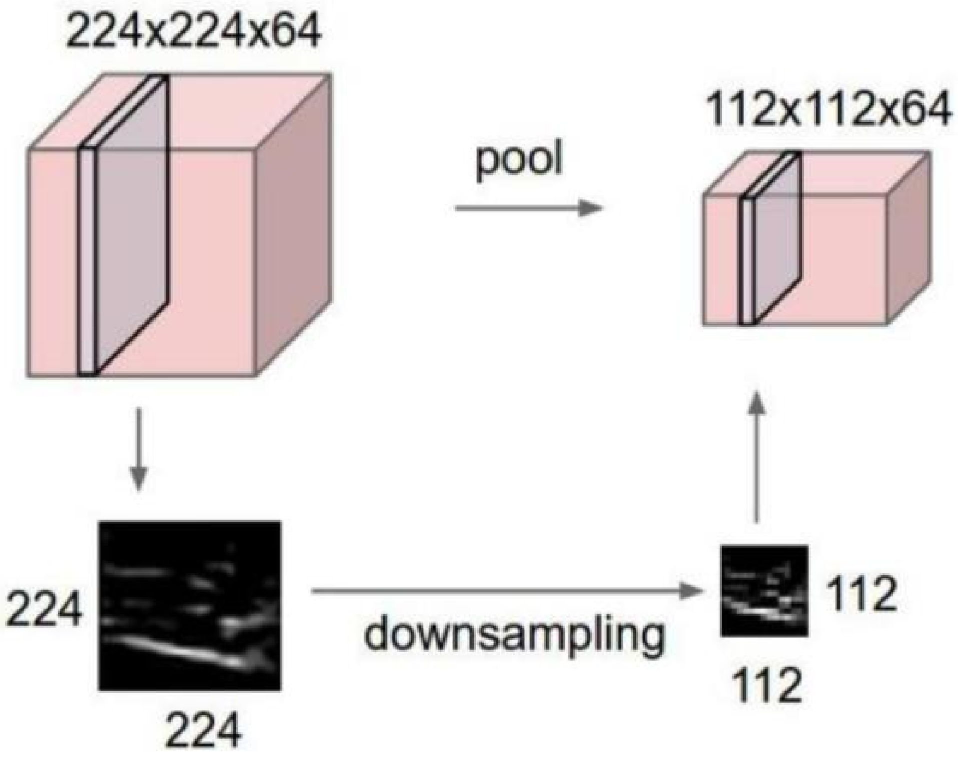




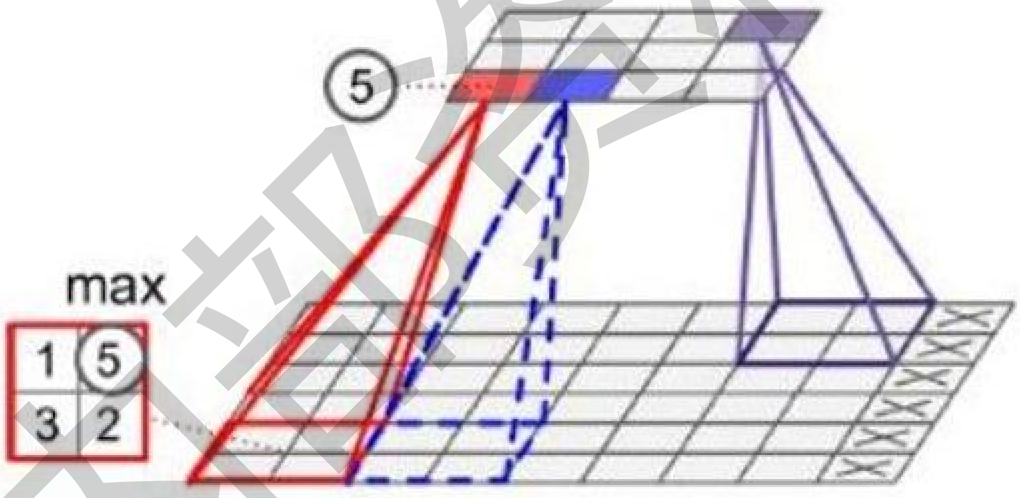
池化

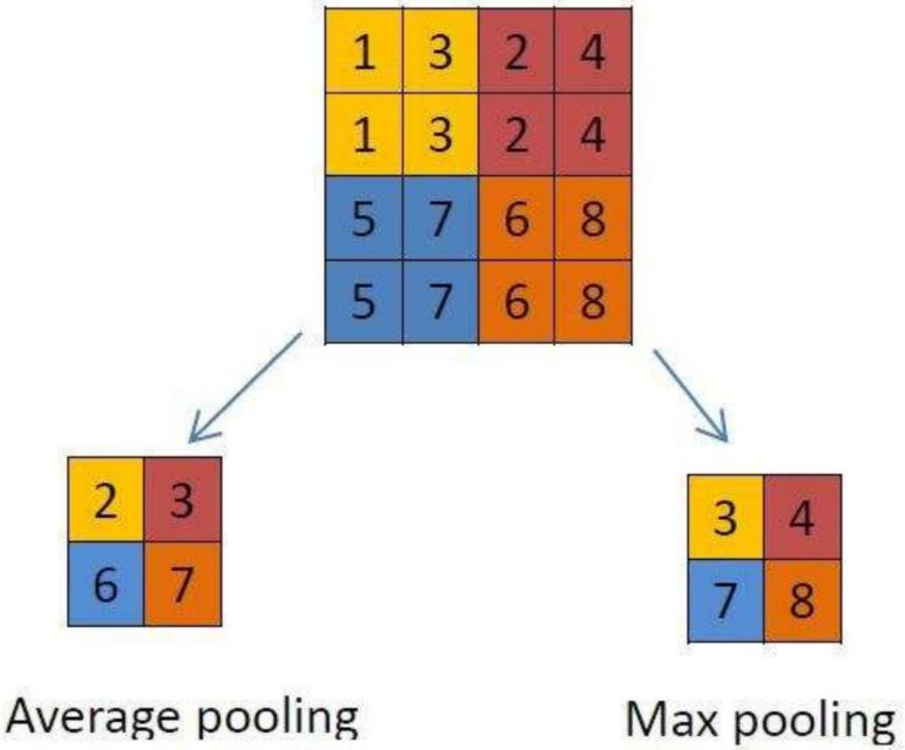
- 使得原始图片的尺寸变小了





池化层的运算


5
max
1 5
3 2


1 3 2 4
1 3 2 4
5 7 6 8
5 7 6 8
2 3
6 7
3 4
7 8
Average pooling
Max pooling

最大池化

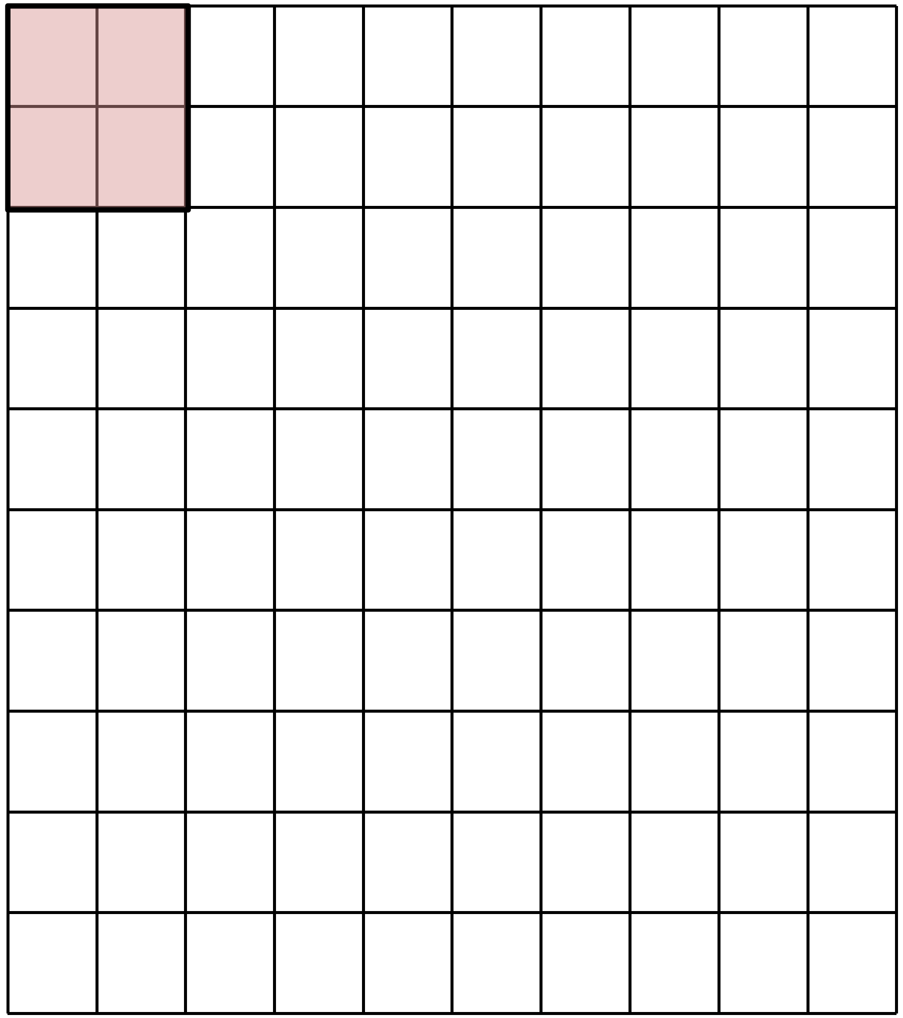
1	3	2	9
7	4	1	5
8	5	2	3
4	2	1	4

7	9
8	

池化层pooling

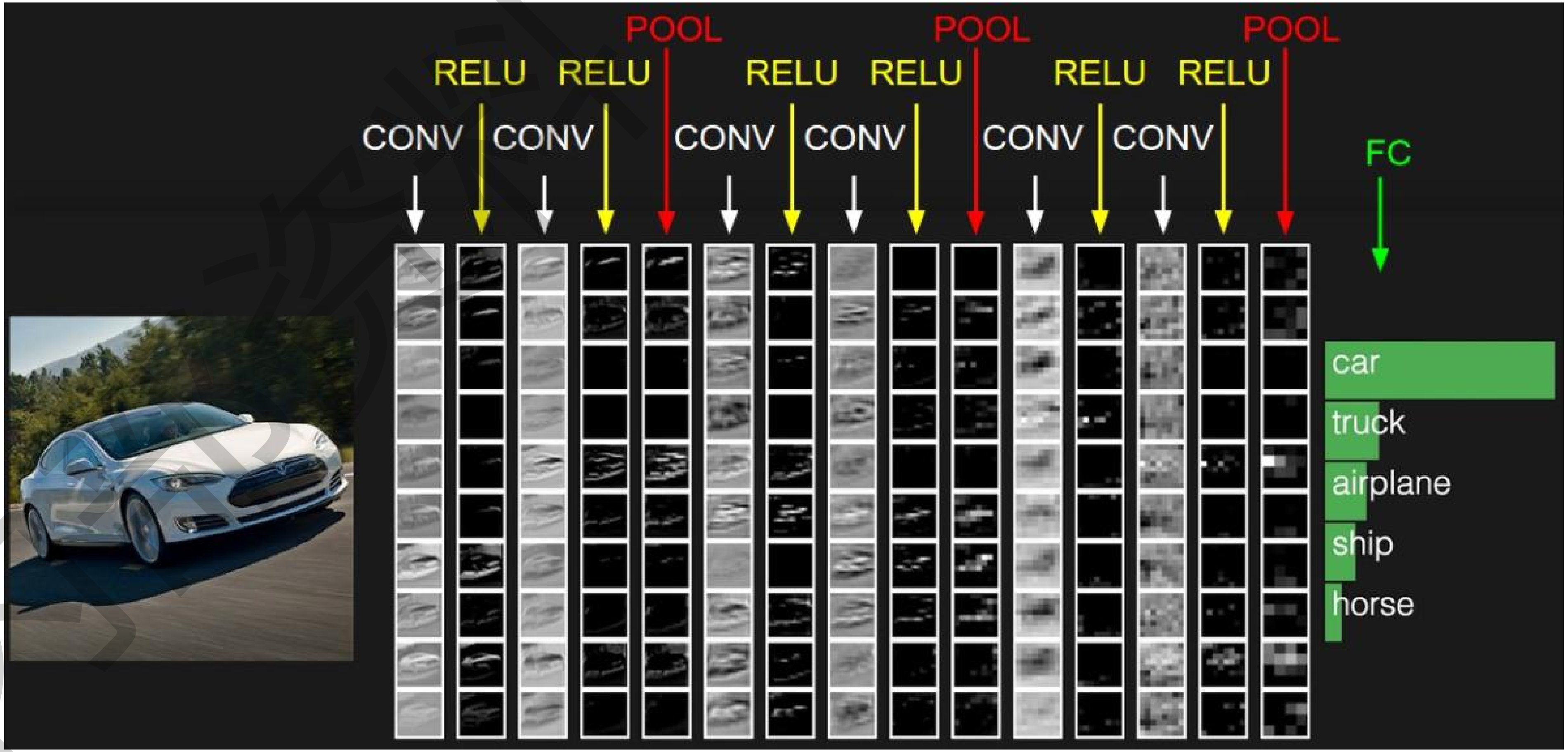
- 通常池化操作是不重叠的
  - 即 size = stride
- 通常使用 max-pooling

最大池化：
相当于计算窗口
在窗口内取最大值



Feature map = 10 * 10 * 1

CNN网络整体结构


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

谢谢大家