第3章
卷积神经网络

让机器更好地理解和服务人类

图像信息

任务：理解图像内容

方法：卷积神经网络

序列信息

任务：理解语音/文字/视频

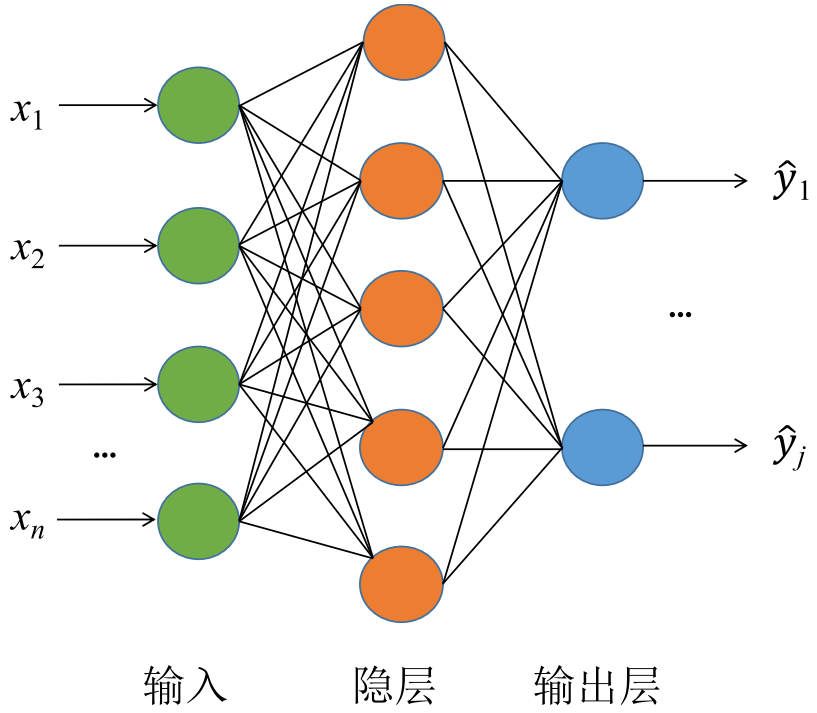
方法：循环神经网络

为什么需要卷积神经网络

为什么需要卷积神经网络

- **计算机视觉**

输入图像

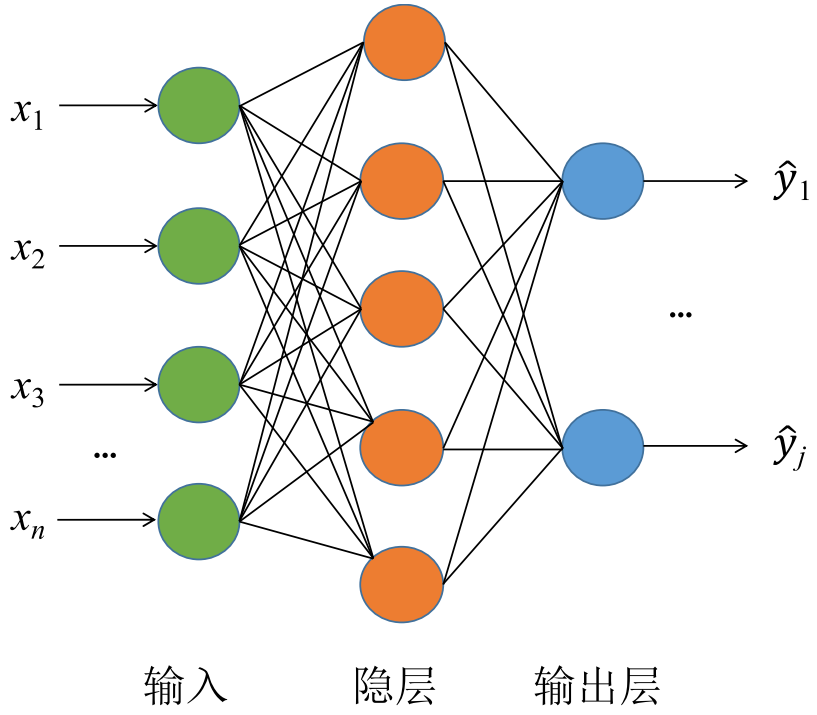


- 输入图像大小为 32x32，输入数据量为 32x32x3 = 3072
- 隐层神经元个数为 100，第一层权值数量为 3072 x100 = 307200

为什么需要卷积神经网络

- **实际场景中，往往需要更大的输入图像以及更深的网络结构。**

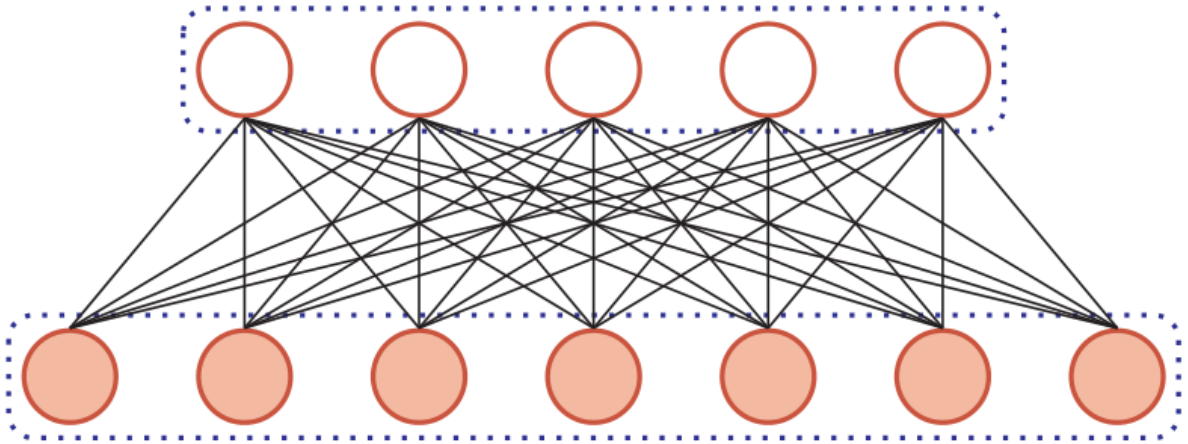
输入图像



- 输入图像大小为 1024x1024，第一层隐层神经元个数为 1000
- 第一层权重数量级为 10^9，过多的参数会导致过拟合
- 卷积神经网络可以有效减少权重数量

为什么需要卷积神经网络

- **权重矩阵的参数非常多**



思考：参数过多导致什么问题？

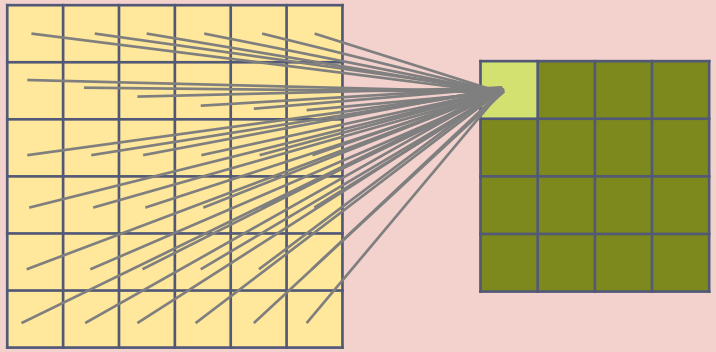
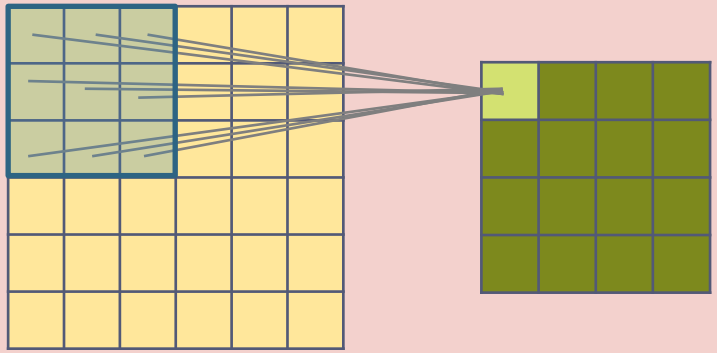
- 神经网络的训练困难；
- 过拟合。

思考：全连接的形式导致什么问题？

- 对输入的细微改变敏感，难以提取局部不变性特征。

- **局部不变性特征**
  - 自然图像中的物体都具有**局部不变性**特征，比如尺度缩放、平移、旋转等操作不影响其语义信息。
  - 而全连接前馈网络很难提取这些局部不变特征。

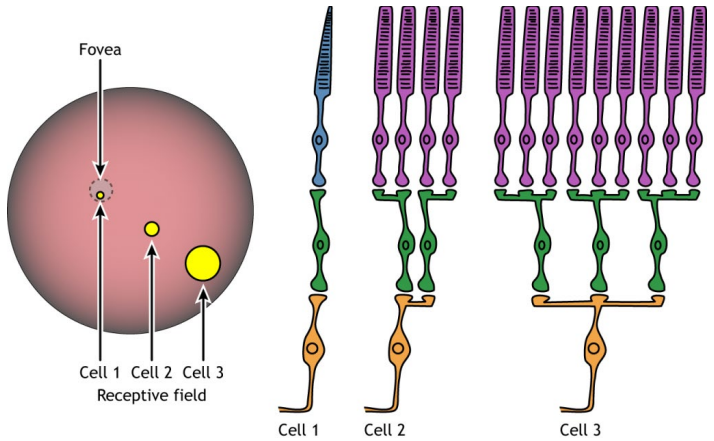
卷积神经网络

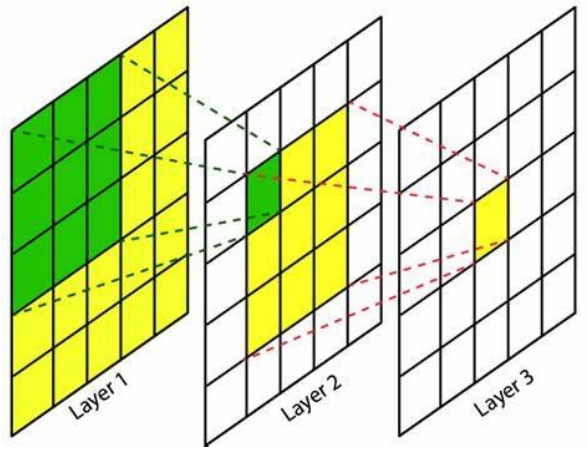
	全连接	卷积
局部连接		
权重共享	所有神经元之间的连接都使用不同权重。	输出层神经元共用同一组权重，进一步减少权重数量。
权重数量	$w_i \times h_i \times w_o \times h_o$	$f \times f$

卷积神经网络

- **卷积神经网络（Convolutional Neural Networks，CNN）**
  - 是一种特殊的前馈神经网络
  - 受生物学上**感受野**（Receptive Field）的机制而提出的

在视觉神经系统中，一个神经元的**感受野**是指视网膜上的特定区域，只有这个区域内的刺激才能够激活该神经元。





- **卷积神经网络有三个结构上的特性：**
  - 局部连接
  - 权重共享
  - 空间或时间上的次采样

卷积

卷积

- **卷积最早被用在信号处理中，用于计算信号的延迟累积。**
  - 假设一个信号发生器每个时刻t产生一个信号x_t，其信息的衰减率为w_k，即在k−1个时间步长后，信息为原来的 w_k 倍
    - 假设$w_1 = 1, w_2 = 1/2, w_3 = 1/4$
- **时刻t收到的信号y_t 为当前时刻产生的信息和以前时刻延迟信息的叠加**

$$
\begin{aligned}
y_t &= 1 \times x_t + 1/2 \times x_{t-1} + 1/4 \times x_{t-2} \\
&= w_1 \times x_t + w_2 \times x_{t-1} + w_3 \times x_{t-2} \\
&= \sum_{k=1}^{3} w_k \cdot x_{t-k+1}.
\end{aligned}
$$

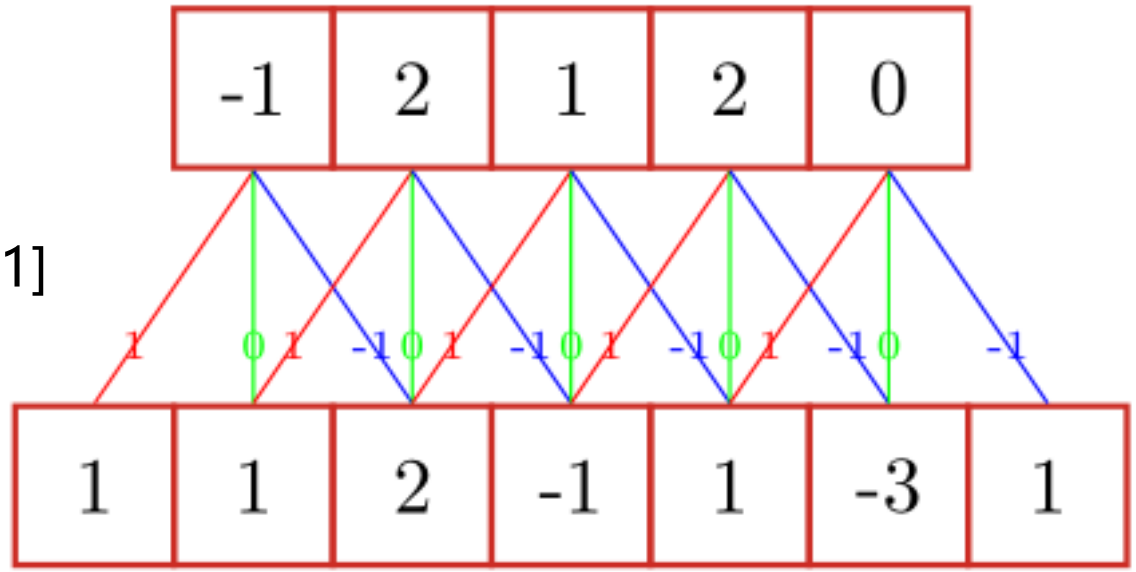
滤波器（filter）或卷积核（convolution kernel）

卷积

- **卷积最早被用在信号处理中，用于计算信号的延迟累积。**
  - 给定一个收入信号序列 x 和滤波器 w 卷积的输出为：

$$y_t = \sum_{k=1}^{m} w_k x_{t-k+1}$$

Filter: [-1,0,1]



二维卷积

- 在图像处理中，图像是以二维矩阵的形式输入到神经网络中，因此我们需要二维卷积。

$$\mathbf{y} = \mathbf{w} \otimes \mathbf{x},$$

$$y_{ij} = \sum_{u=1}^{m} \sum_{v=1}^{n} w_{uv} \cdot x_{i-u+1,j-v+1}.$$

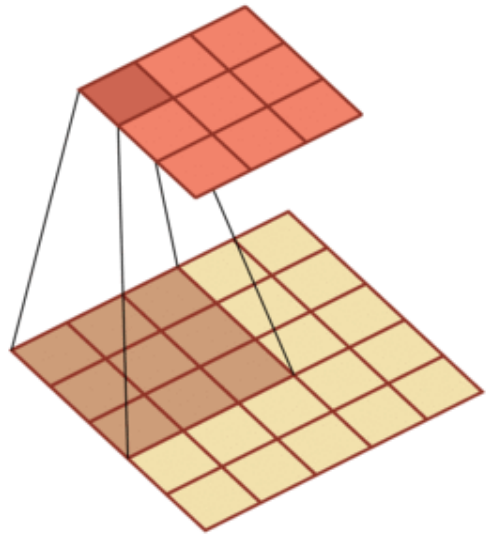
1	1	1 ×−1	1 ×0	1 ×0
-1	0	-3 ×0	0 ×0	1 ×0
2	1	1 ×0	-1 ×0	0 ×1
0	-1	1	2	1
1	2	1	1	1

$\otimes$

1	0	0
0	0	0
0	0	-1

$=$

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

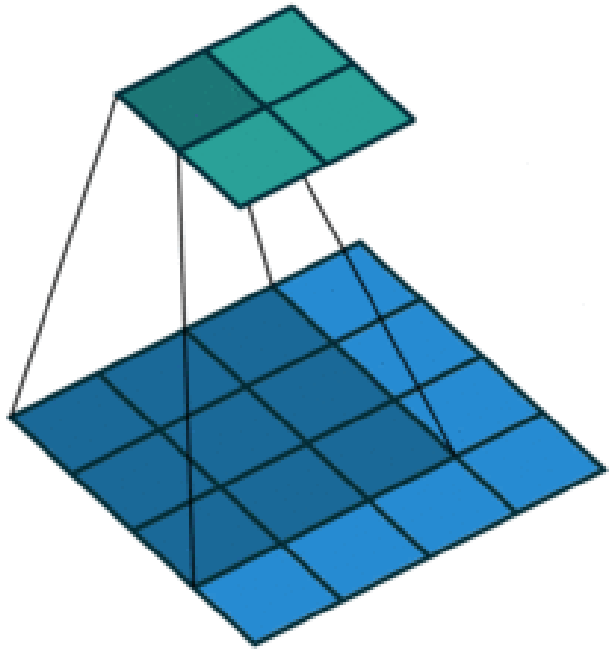


二维卷积

- 在图像处理中，图像是以二维矩阵的形式输入到神经网络中，因此我们需要二维卷积。

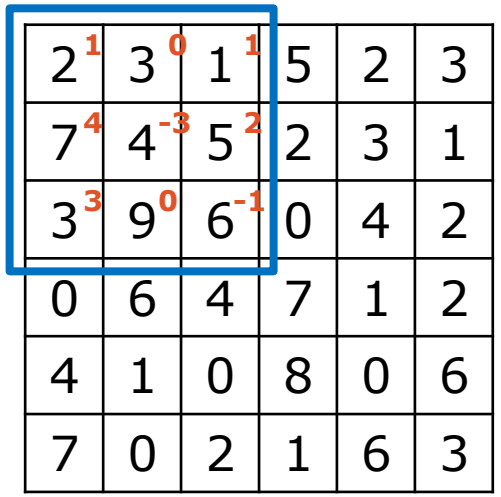
Input				Kernel			Output	
0	1	2	*	0	1	=	19	25
3	4	5		2	3		37	43
6	7	8						

$$0\times0+1\times1+3\times2+4\times3=19,$$
$$1\times0+2\times1+4\times2+5\times3=25,$$
$$3\times0+4\times1+6\times2+7\times3=37,$$
$$4\times0+5\times1+7\times2+8\times3=43.$$



卷积

- 神经网络中的卷积实际为计算矩阵内积：



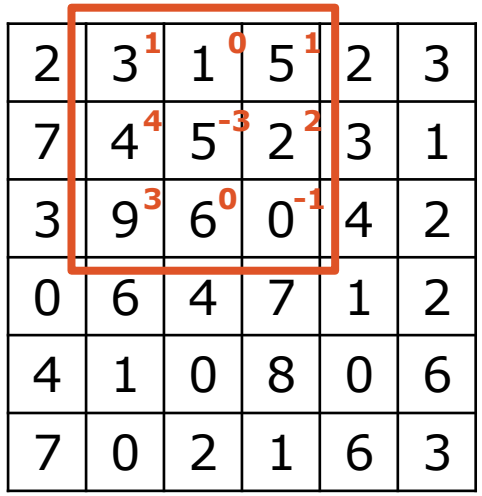
2^{1}	3^{0}	1^{1}	5	2	3
7^{4}	4^{-3}	5^{2}	2	3	1
3^{3}	9^{0}	6^{-1}	0	4	2
0	6	4	7	1	2
4	1	0	8	0	6
7	0	2	1	6	3

*

1	0	1
4	-3	2
3	0	-1

=

32			



2	3^{1}	1^{0}	5^{1}	2	3
7	4^{4}	5^{-3}	2^{2}	3	1
3	9^{3}	6^{0}	0^{-1}	4	2
0	6	4	7	1	2
4	1	0	8	0	6
7	0	2	1	6	3

*

1	0	1
4	-3	2
3	0	-1

=

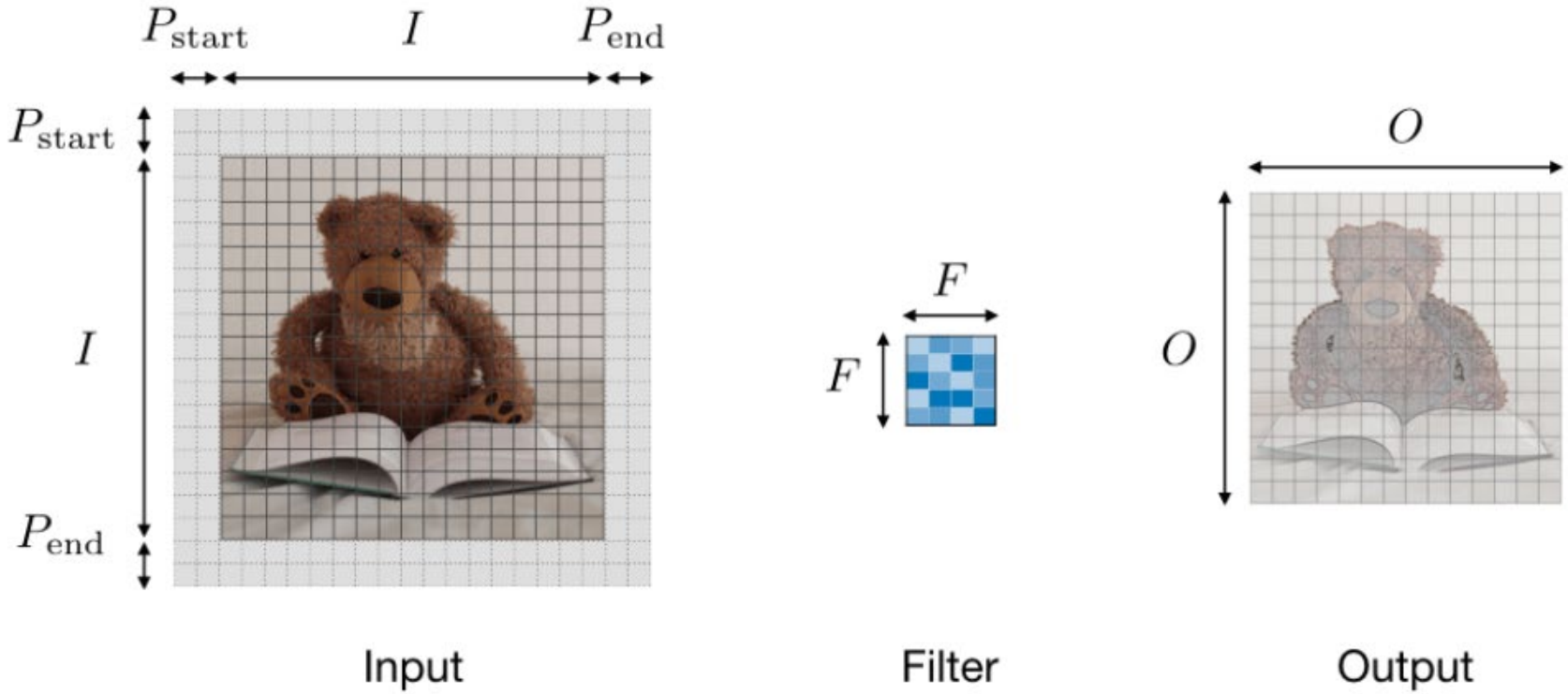
32	**40**		

步长（Stride）、填充（Padding）

- **卷积有两个重要的超参数：**
  - 步长（Stride），填充（Padding）
  - 步长：卷积核在输入数据上每次移动的像素距离。既包括水平方向的移动，也包括垂直方向的移动。
  - 填充：填充是指在将卷积核应用到输入数据之前，在输入数据的边界周围人为添加额外像素的操作。通常添加的像素值为0（称为零填充）。

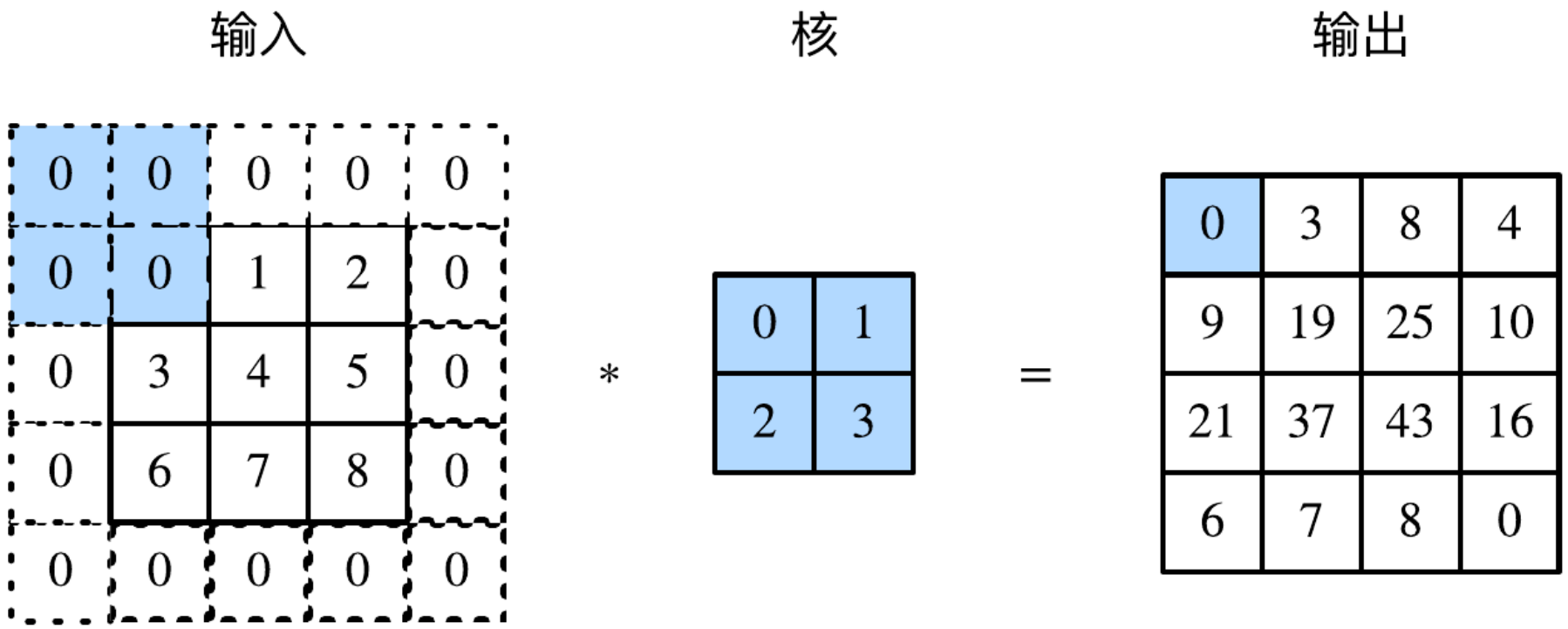
填充（Padding）

- 离散卷积的边缘效应



填充（Padding）

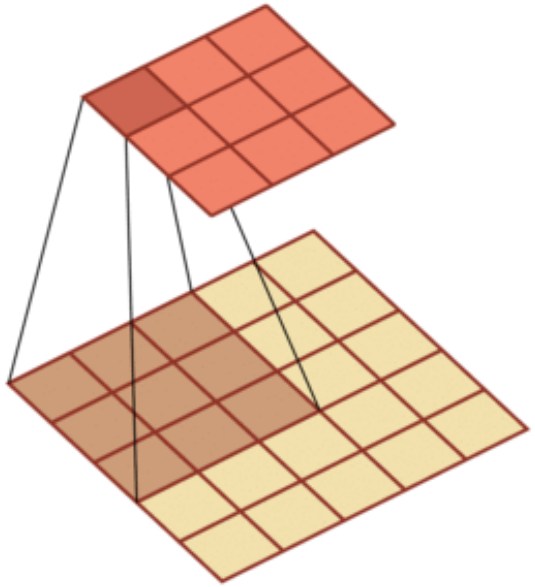
- **离散卷积的边缘效应**
  - Zero-Padding，edge-padding，reflect-padding



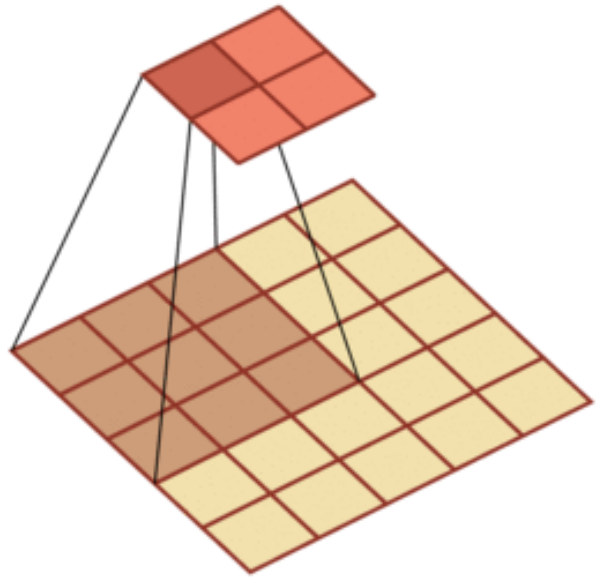
步长 (Stride)

填充 (Padding)

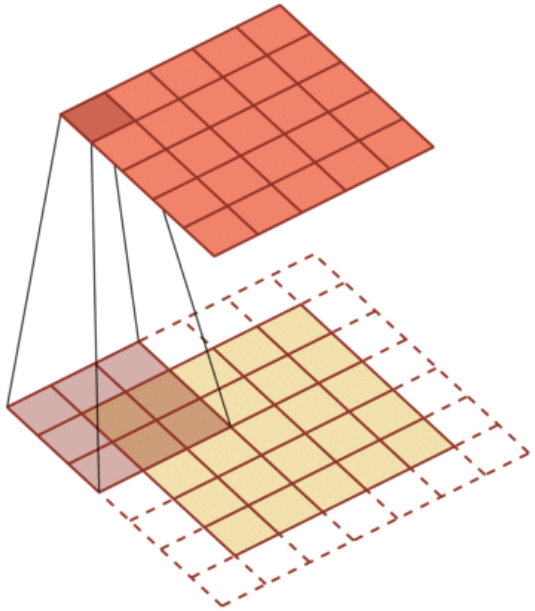
步长1，
零填充0



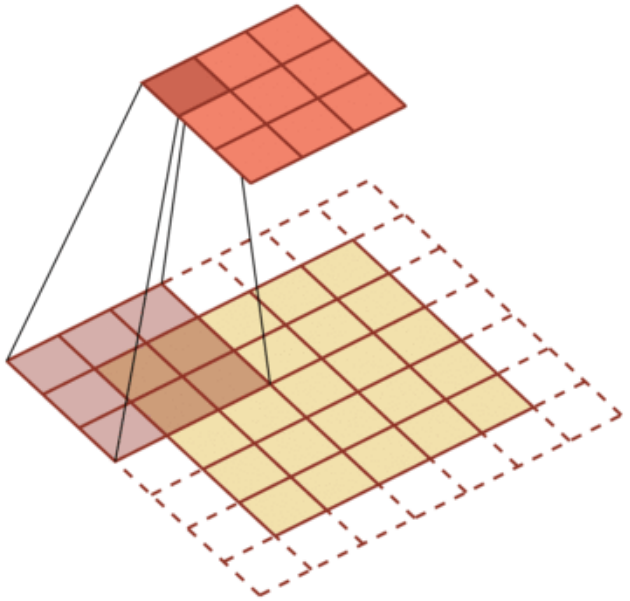
步长2
零填充0



步长1，
零填充1



步长2，
零填充1



步长（Stride）、填充（Padding）

- 思考：

假设输入长度为M，步长为S，卷积核大小为K，零填充为P，那么输出长度是多少?

$$\left\lfloor \frac{M + 2P - K}{S} \right\rfloor + 1$$

卷积的计算

输入

3	0	8	4
6	5	1	0
0	7	0	3
9	1	2	1

*

卷积核W

0	2	3
3	1	0
2	0	1

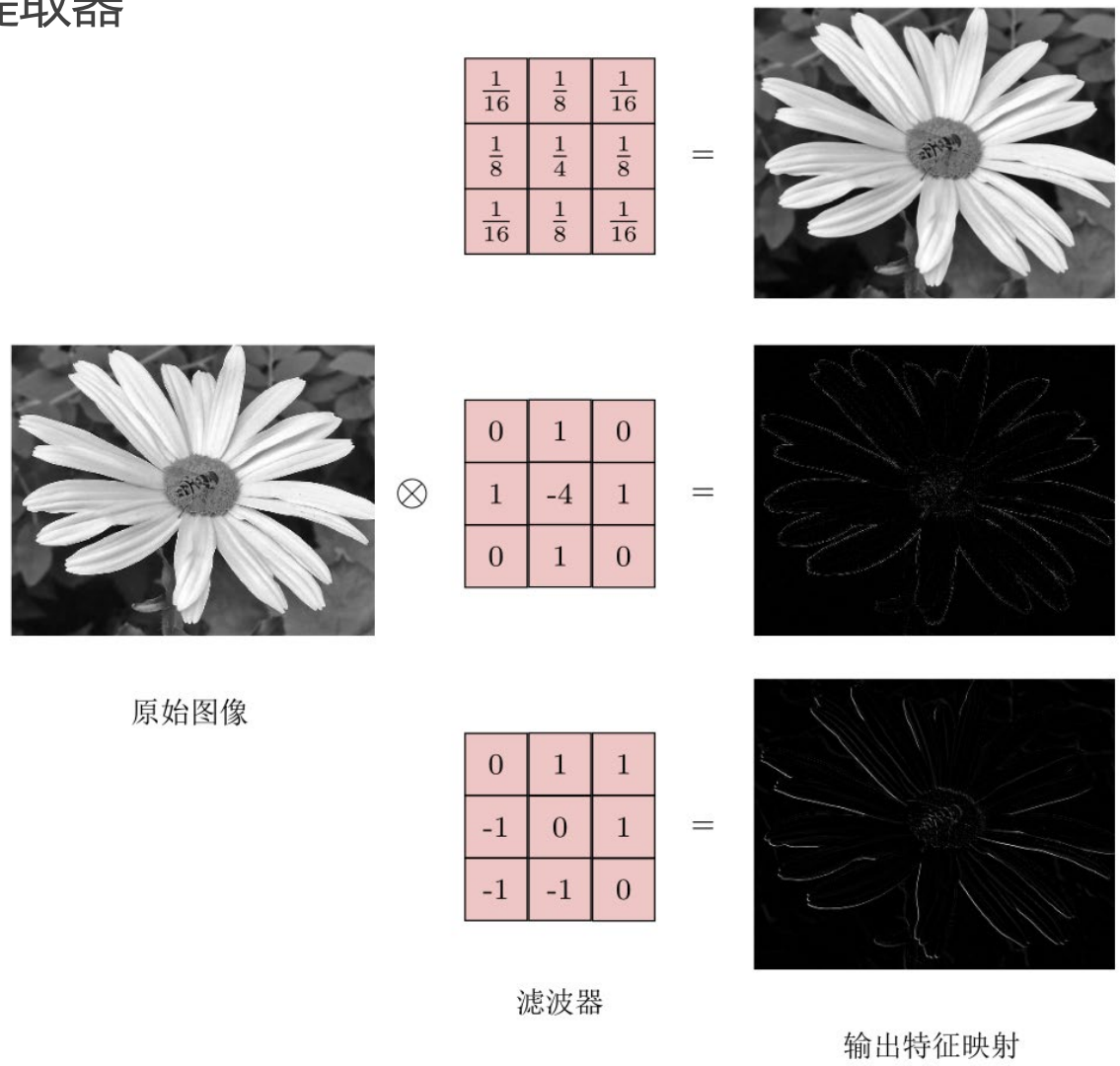
偏置b=1

= ?

取P=0, S=1（不进行填充补0，步长为1）

卷积作为特征提取器

- **特征映射（Feature Map）：图像经过卷积后得到的特征。**
  - 卷积核看成一个特征提取器

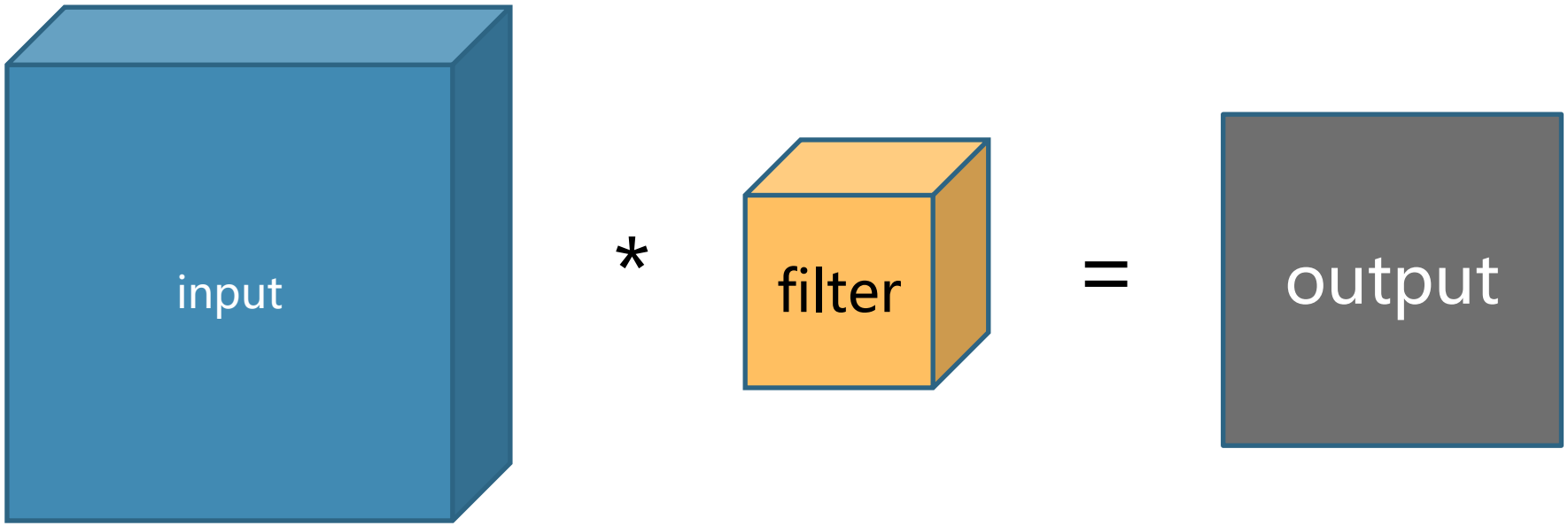


卷积作为特征提取器

- 卷积层如何检测特征

多输入通道

- 多输入特征图单输出特征图卷积运算

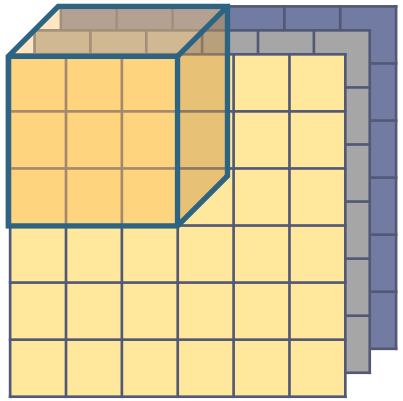


多输入通道

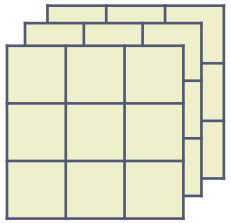
- 多输入特征图

  单输出特征图

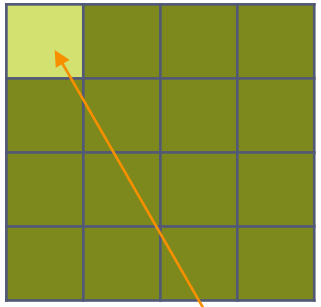
  卷积运算



*



=



6x6x3　　3x3x3　　4x4

C = 0

0	0	0
0	2	2
0	1	2

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

C = 1

0	0	0
0	0	2
0	1	2

*

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

=

2-2-1+2
+
0-2+0+2
+
(-1)+0+0+2

=

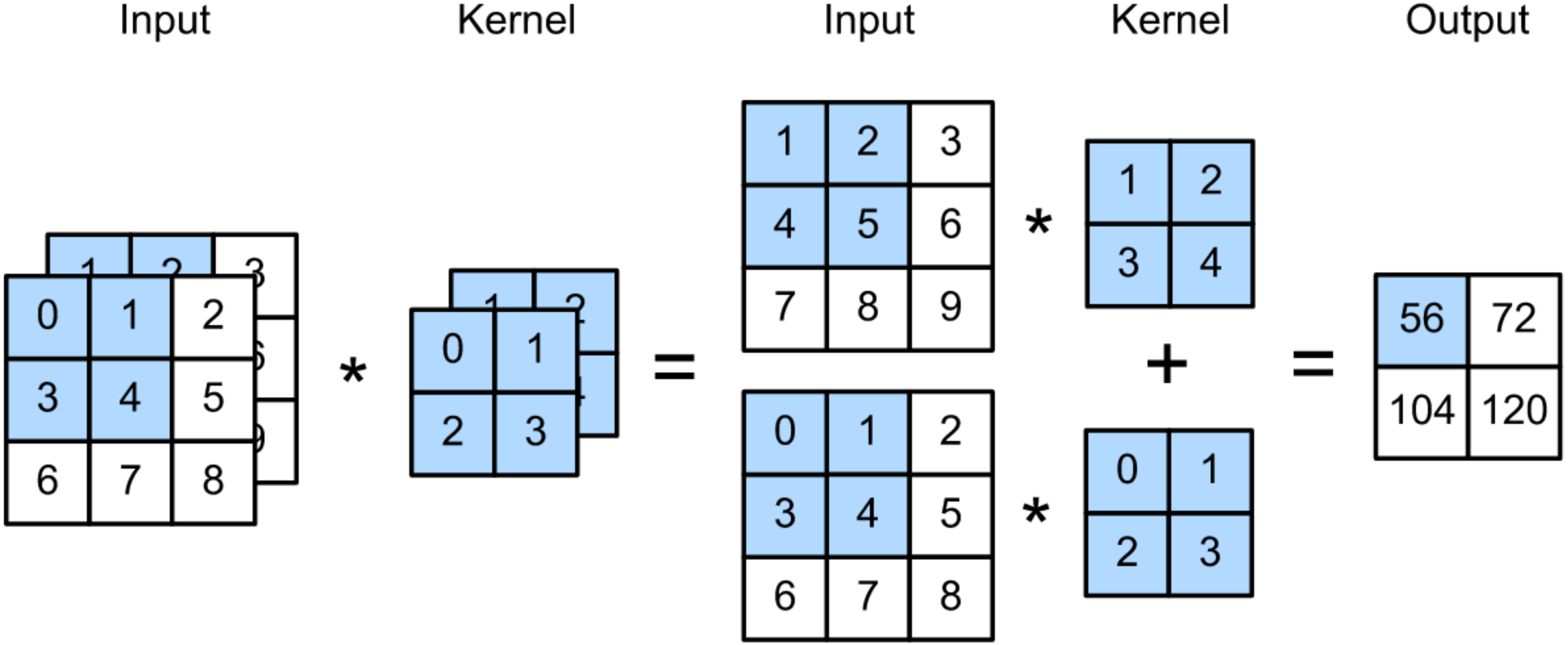
2

C = 2

0	0	0
0	1	1
0	0	2

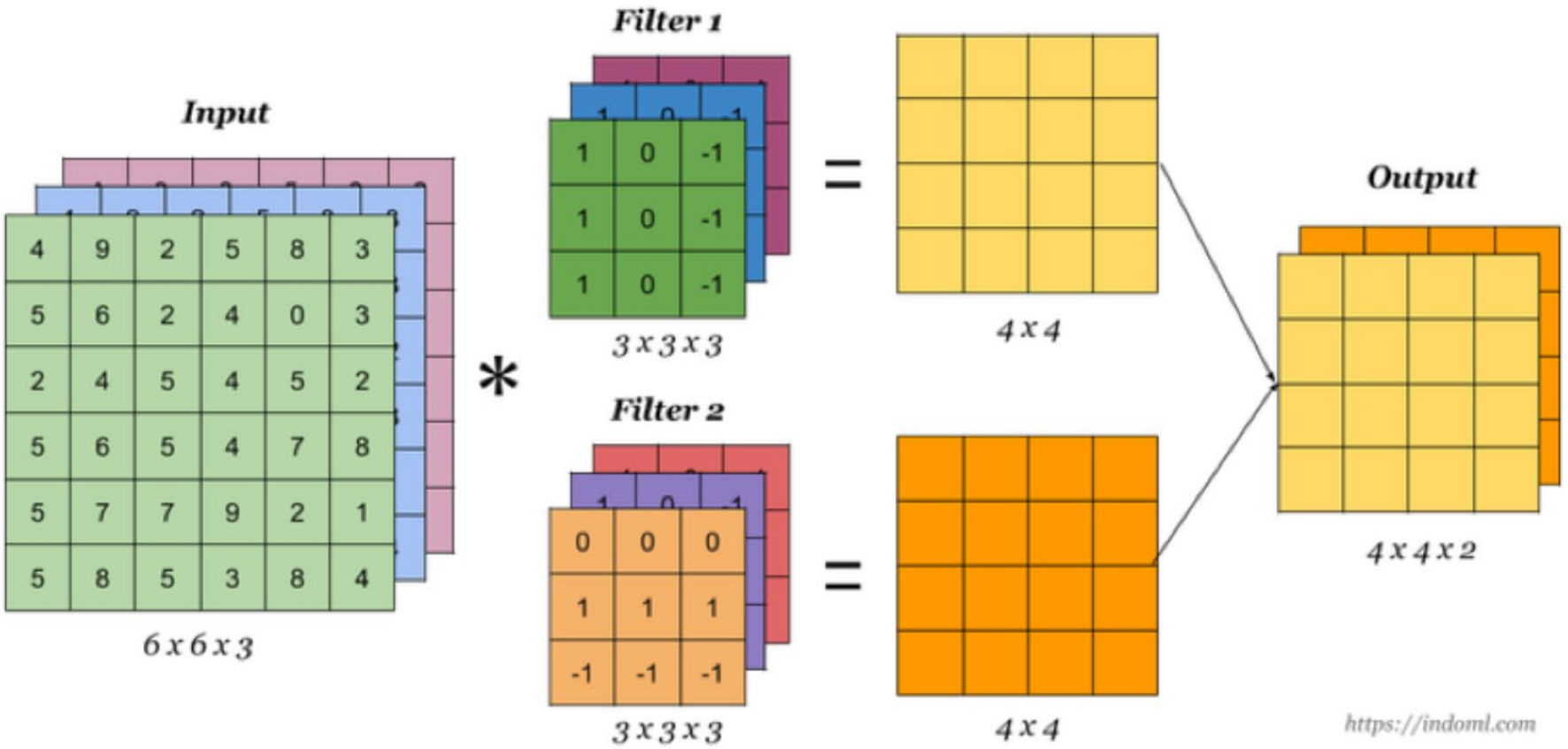
1	-1	-1
-1	-1	0
-1	1	1

多输入通道


Input
Kernel
Input
Kernel
Output
0 1 2
3 4 5
6 7 8
*
0 1
2 3
=
1 2 3
4 5 6
7 8 9
*
1 2
3 4
0 1 2
3 4 5
6 7 8
*
0 1
2 3
+
=
56 72
104 120

多输入多输出通道

- 多输出通道的卷积：使用多个卷积核

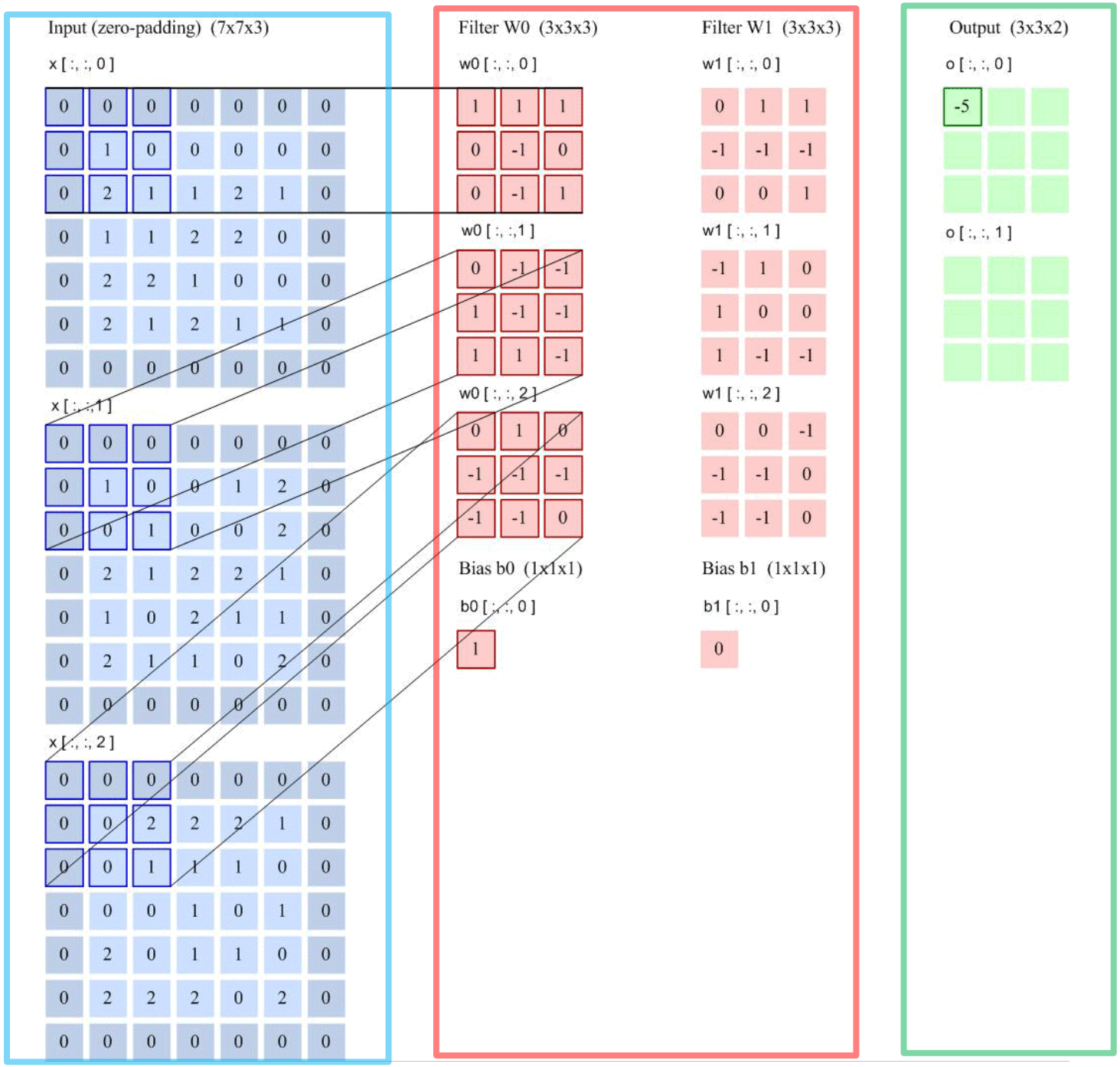


- $size = 3$
- $c\_in = 3$
- $c\_out = 2$
- $stride = 1$
- $padding = 0$

步长2

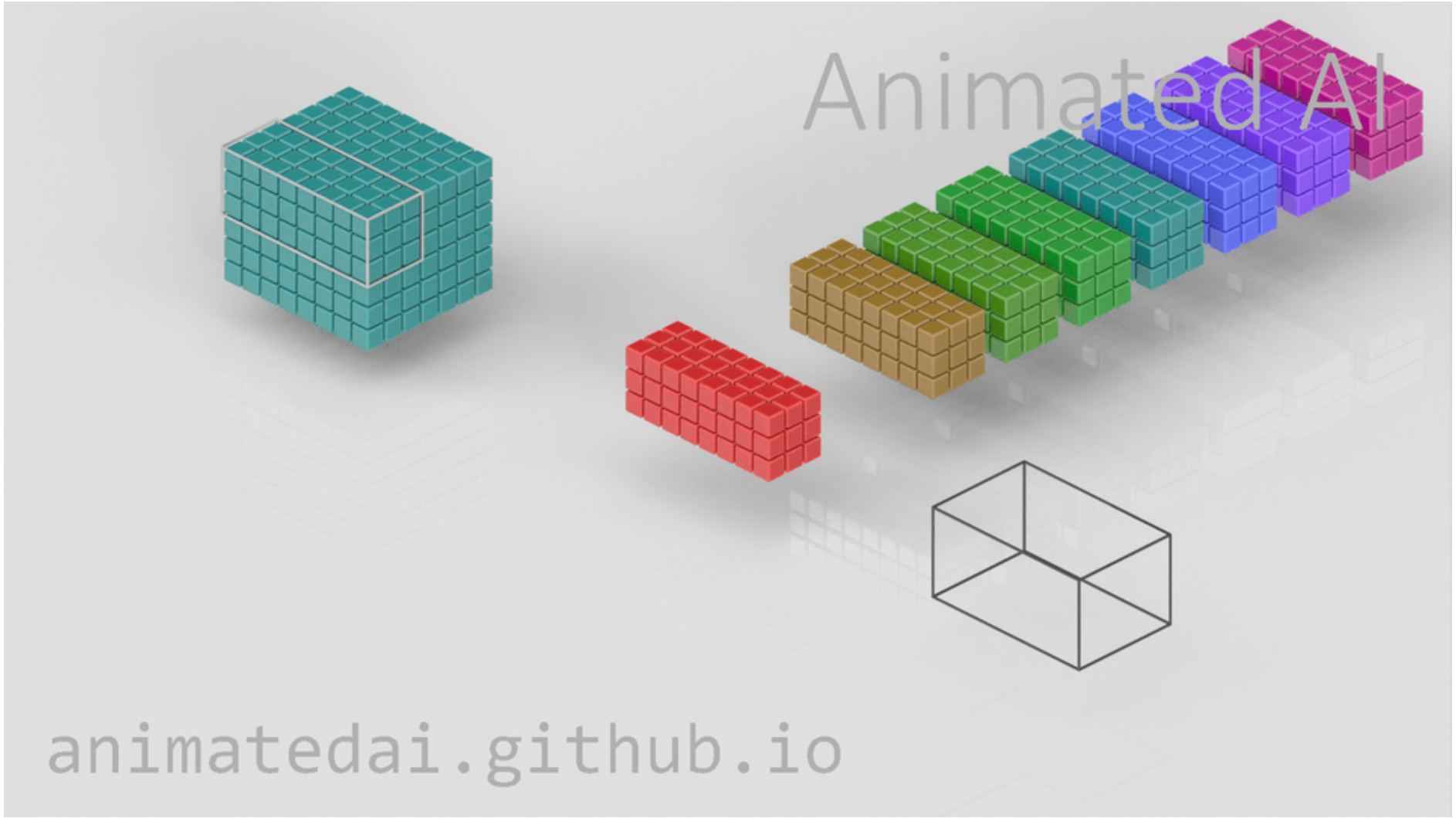
filter个数3

3*3 填充



多输入多输出通道

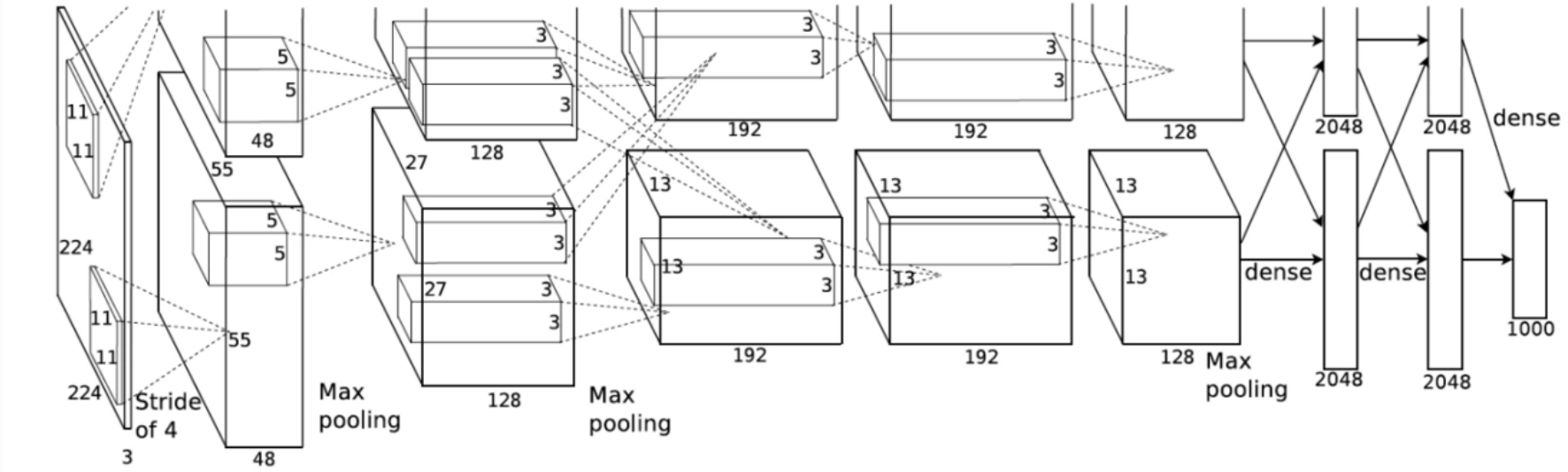
- 典型的卷积层为3维结构



多输入多输出通道

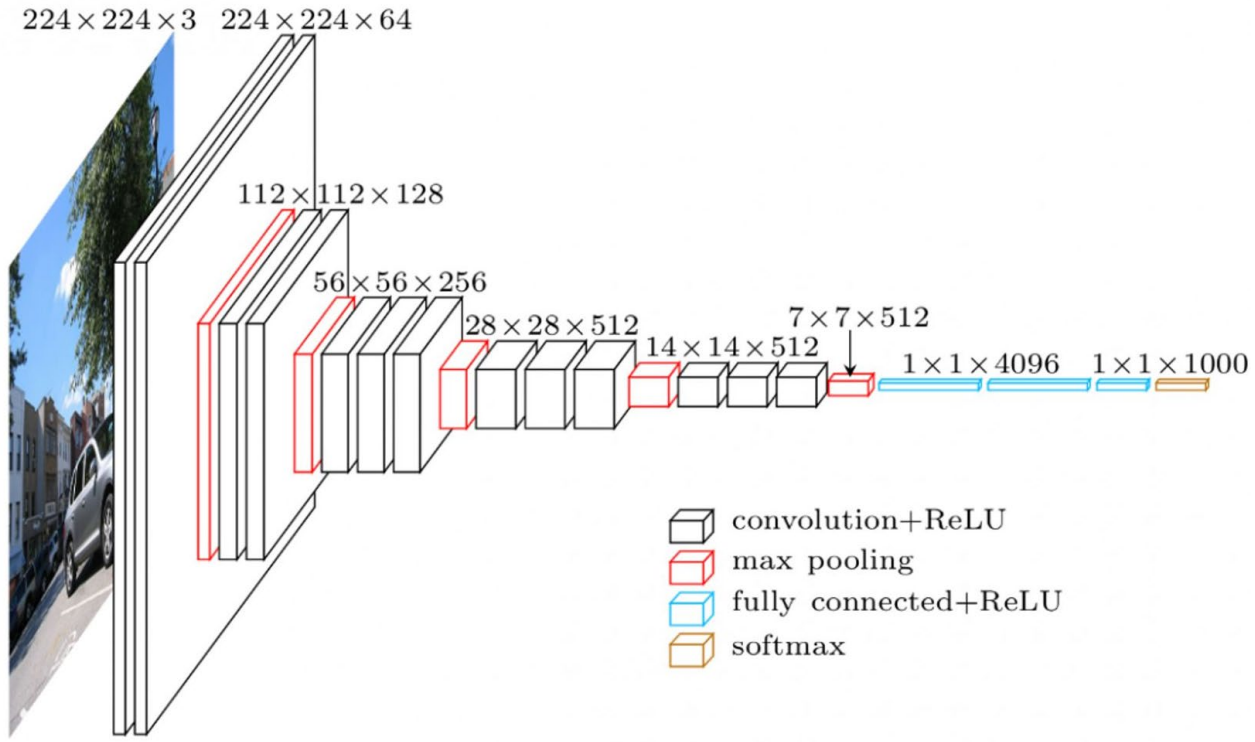
- 每个输出通道可以识别特定的模式

- 因此，卷积层一般都需要设计多个卷积核，以从不同的角度进行特征提取



多输入多输出通道

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 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

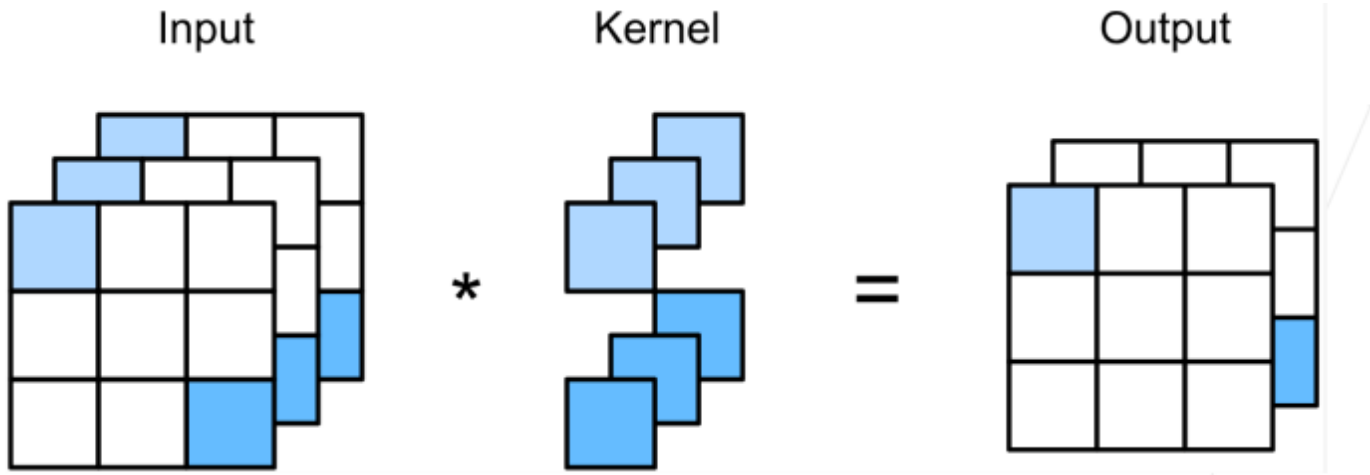


卷积参数量及浮点计算量

- 假设输入特征图包含C_i个通道，输出特征图包含C_o个通道，卷积核的长、宽均为K，输出特征图的长、宽分别为L_h、L_w，那么该卷积层的参数量是多少？浮点计算量是多少？

1×1卷积层

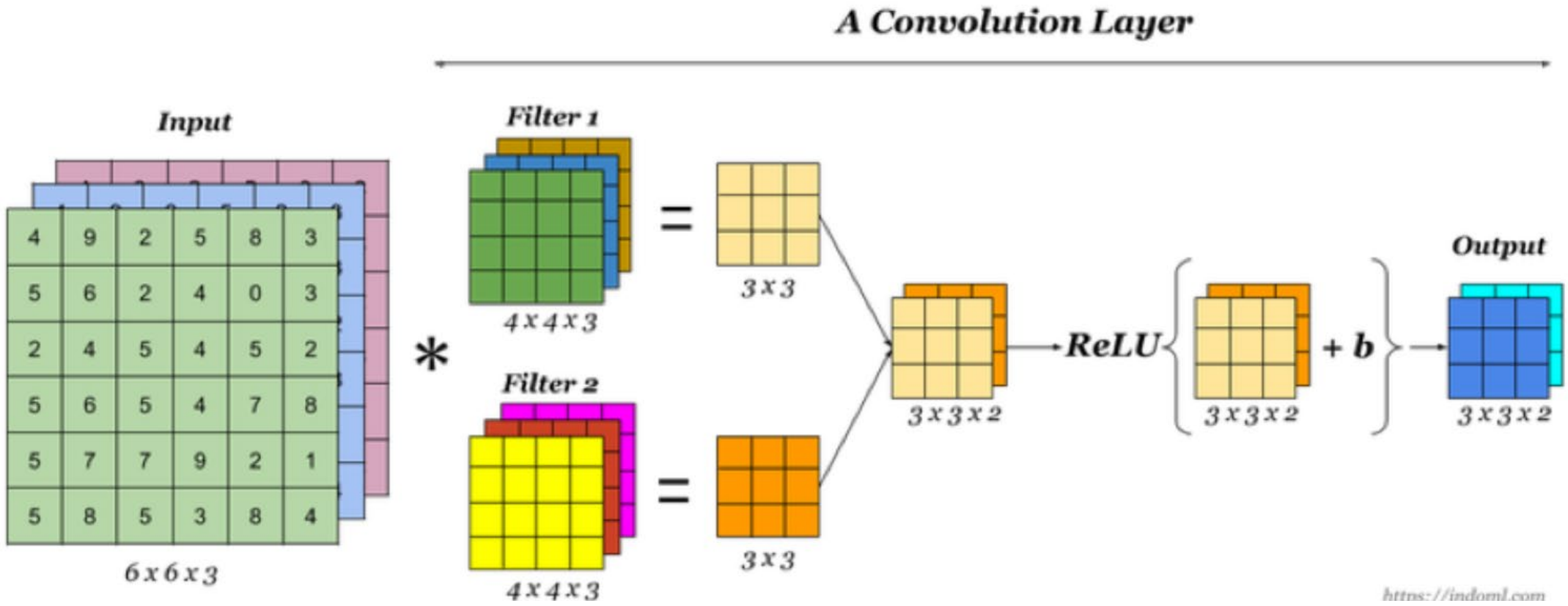
- 1×1卷积：卷积核长宽均为1——有什么作用?



- 1×1卷积不识别空间模式，只融合通道。

激活

- 卷积层后，同样需要经过激活函数



激活

- 卷积层后，同样需要经过激活函数

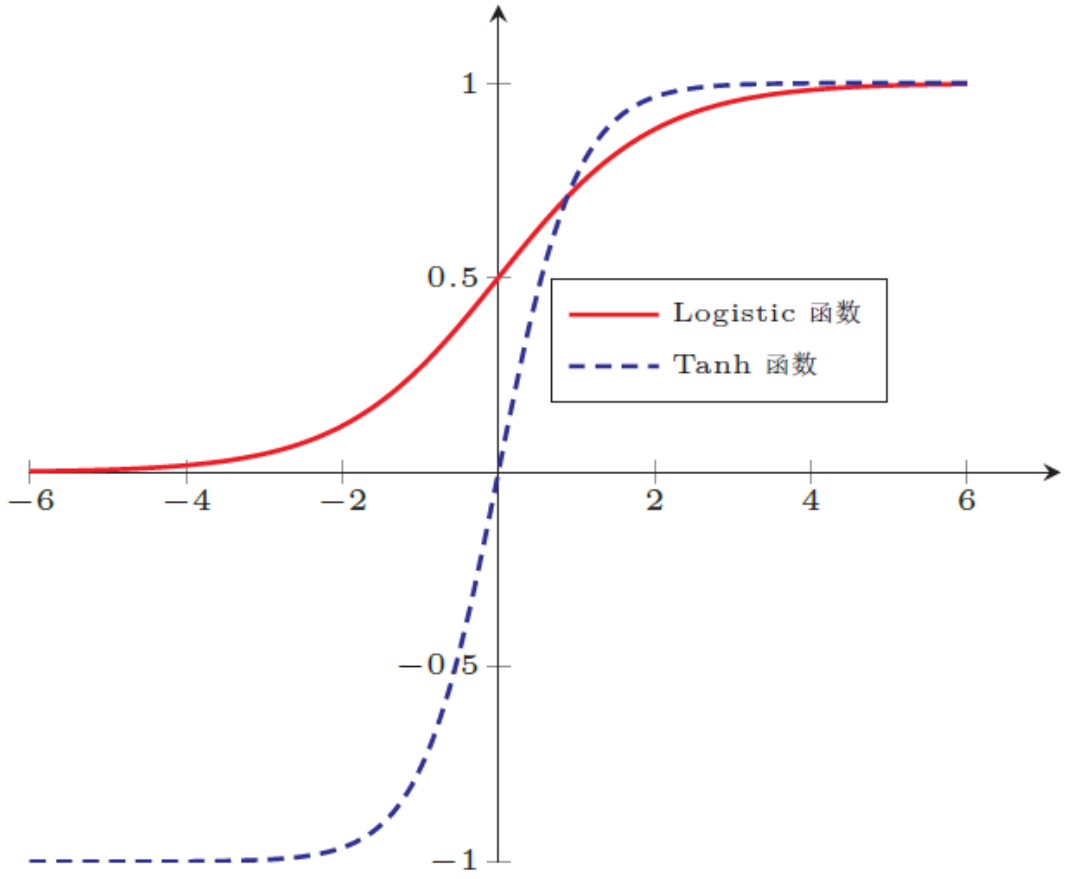


图 4.2 Logistic 函数和 Tanh 函数

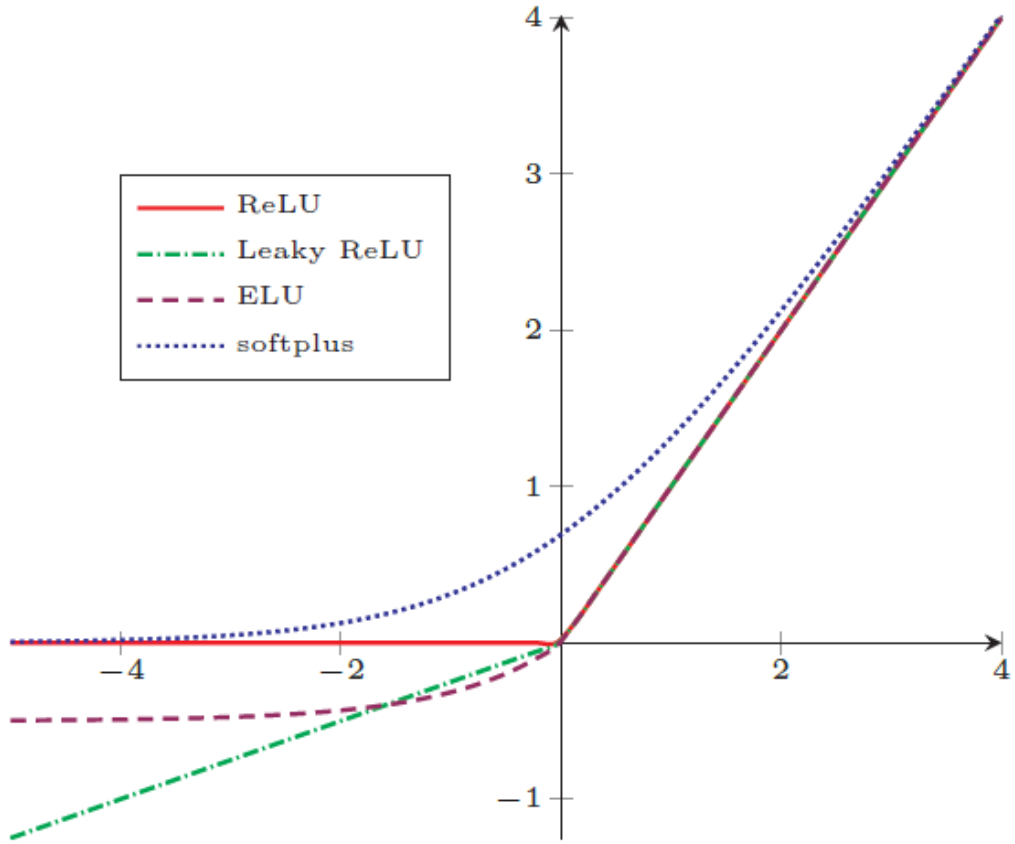
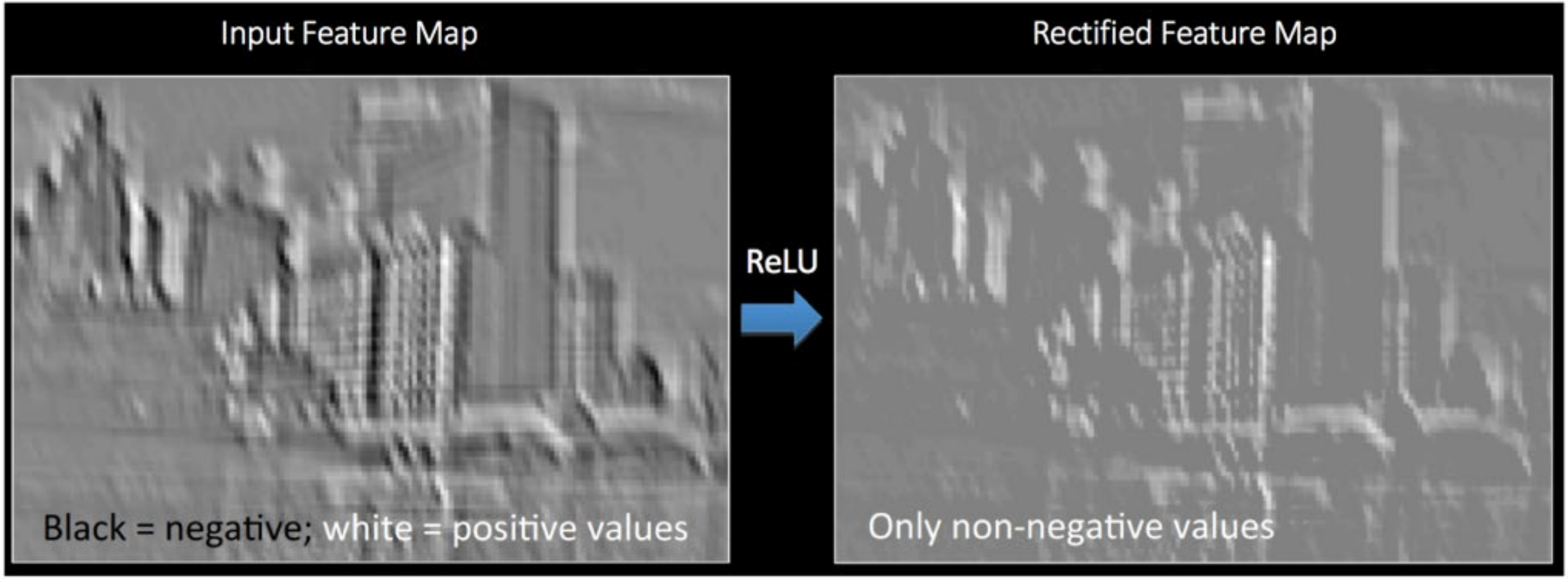


图 4.4 ReLU、Leaky ReLU、ELU 以及 Softplus 函数

激活


Input Feature Map
Rectified Feature Map
ReLU
Black = negative; white = positive values
Only non-negative values

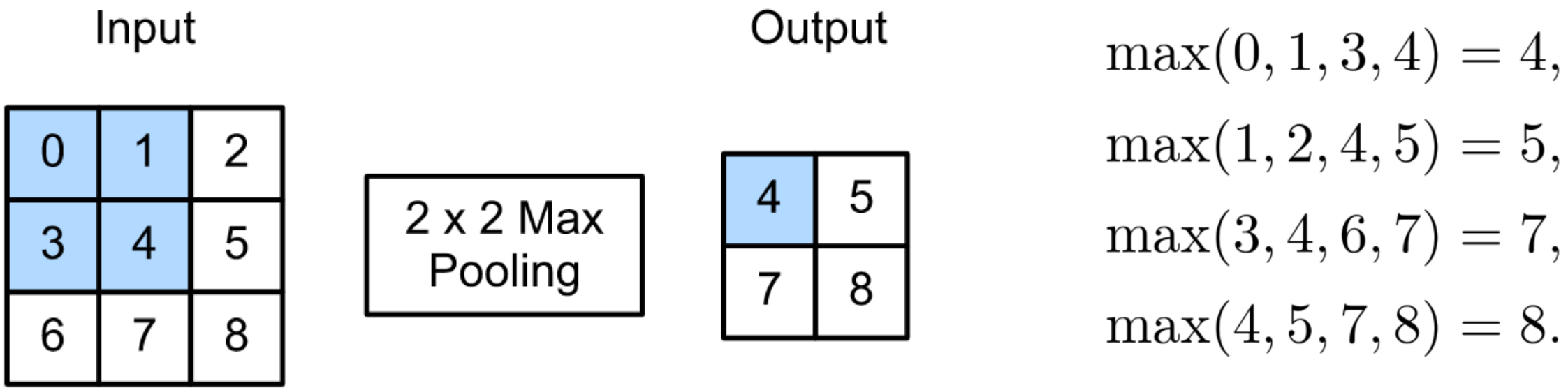
卷积总结

- 卷积层将输入与卷积核进行交叉相关运算，并加上偏置，得到输出；
- 卷积核、偏置都是可学习参数；
- 卷积核大小、步长、填充、输出通道数都是卷积层超参数；
- 卷积核个数决定输出通道数；
- 1×1卷积能够实现通道融合。

汇聚（池化）

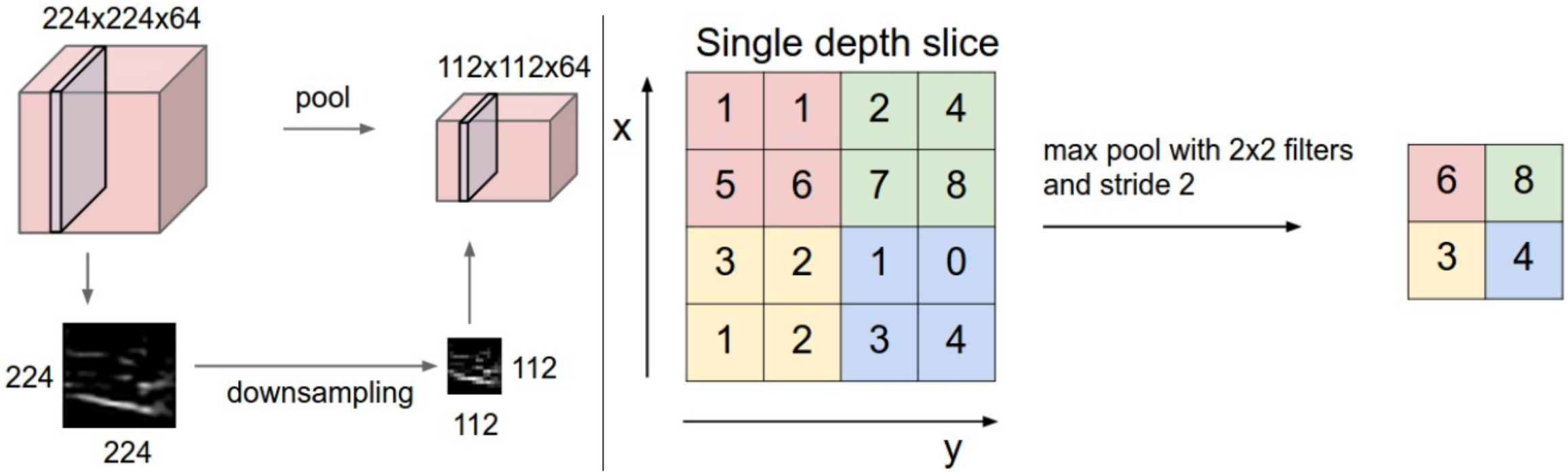
汇聚 / 池化 (Pooling)

- 汇聚（池化）是一种降采样操作，能够提供一定程度的平移不变性
- 最大池化（Max-Pooling）：返回滑动窗口中的最大值
- 平均池化（Avg-Pooling）：返回滑动窗口中的平均值



$$\max(0,1,3,4)=4,$$
$$\max(1,2,4,5)=5,$$
$$\max(3,4,6,7)=7,$$
$$\max(4,5,7,8)=8.$$

汇聚 / 池化 (Pooling)



Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

汇聚 / 池化 (Pooling)

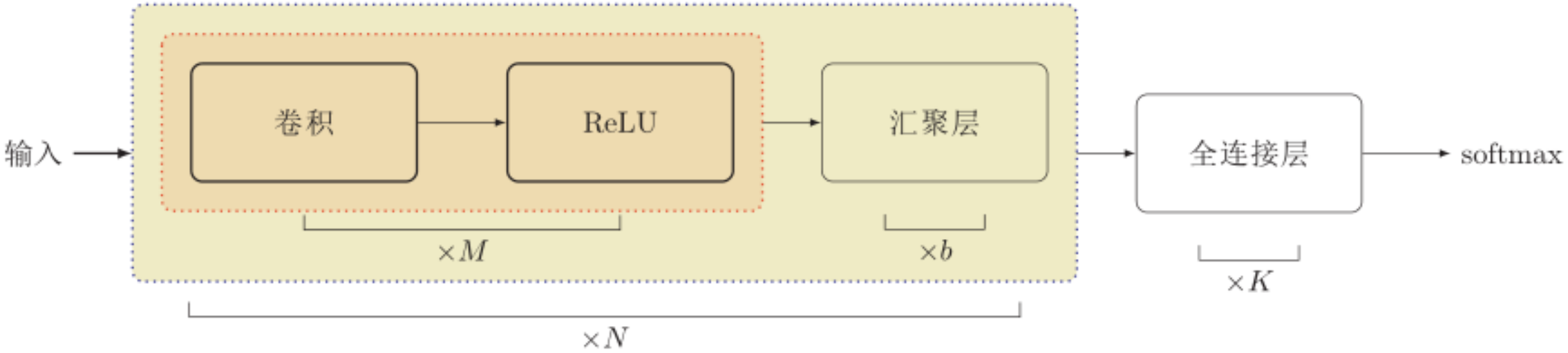
- **填充、步长与多通道：**
  - 填充、步长的概念与卷积层类似；
  - 没有可学习参数；
  - 每个输入通道分别应用池化层，得到相应的输出通道，也就是说不改变通道数量，输出通道数=输入通道数。

思考：比较卷积与汇聚（池化）运算的异同。

卷积神经网络

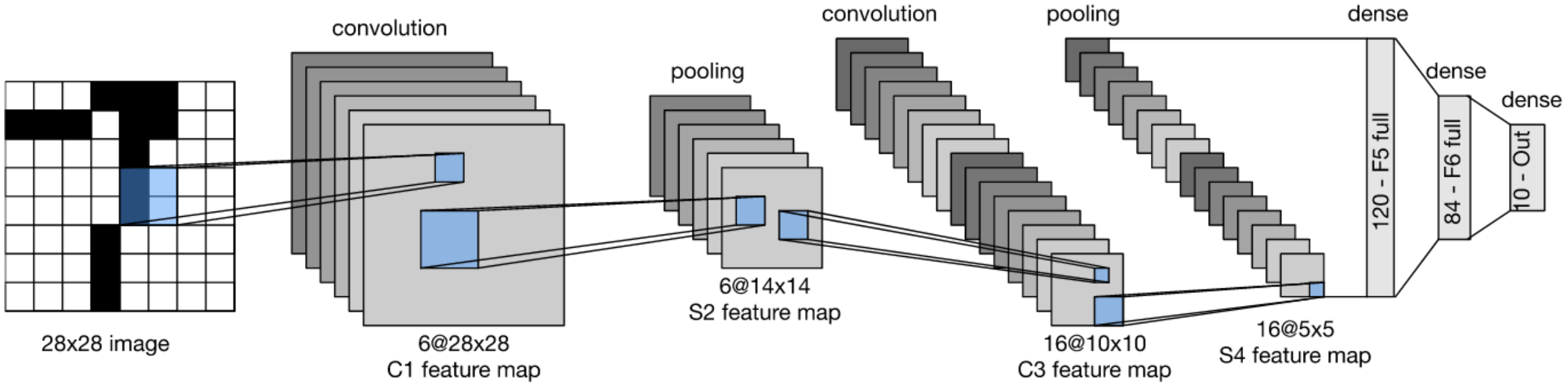
卷积神经网络的结构

- **卷积网络是由卷积层、汇聚层、全连接层交叉堆叠而成。**
  - 趋向于小卷积、大深度
  - 趋向于全卷积
- **典型结构**



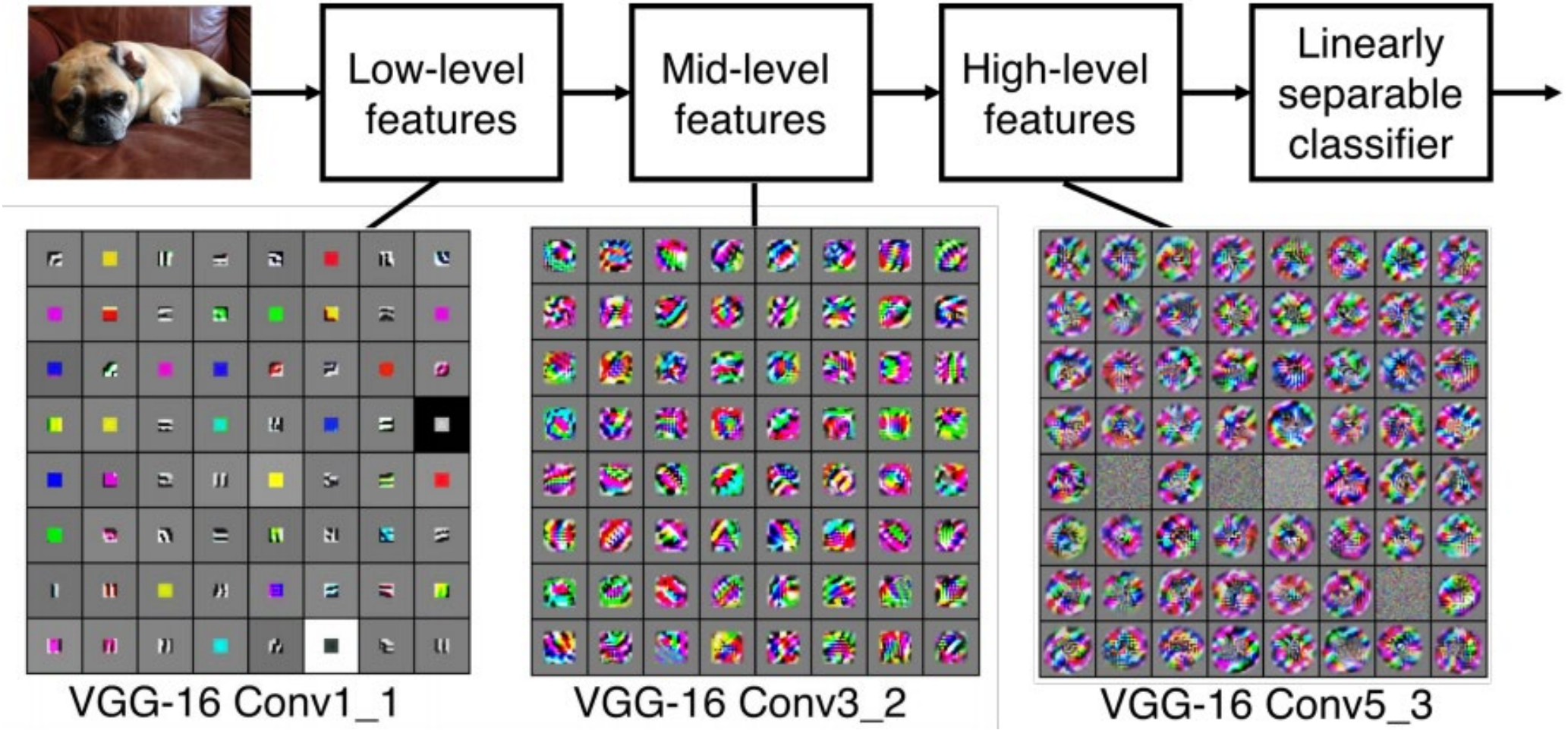
  - 一个卷积块为连续M 个卷积层和b个汇聚层（M通常设置为2 ~ 5，b为0或1）。一个卷积网络中可以堆叠N 个连续的卷积块，然后在接着K 个全连接层（N 的取值区间比较大，比如1 ~100或者更大；K一般为0 ~2）。

示例：LeNet-5

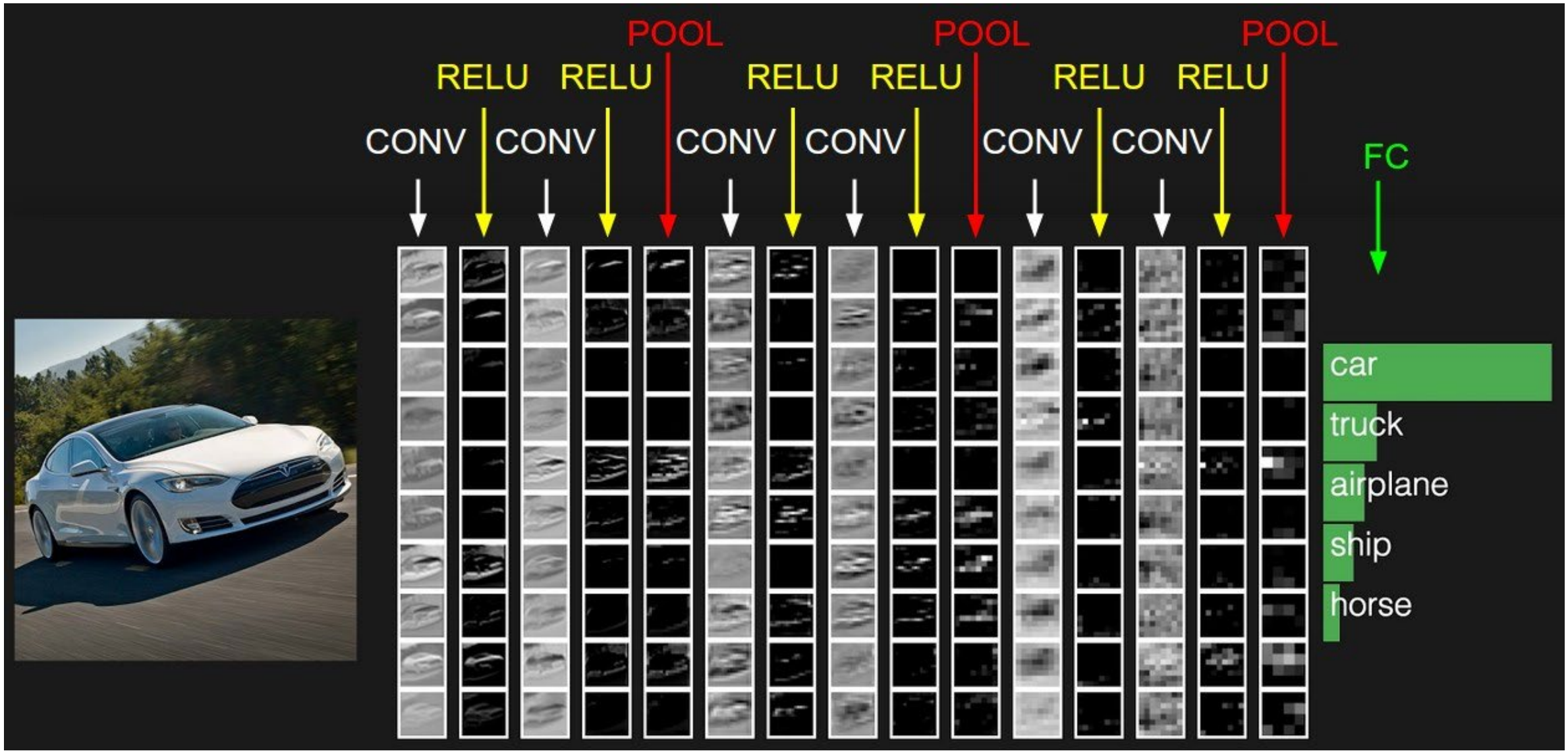

convolution
pooling
convolution
pooling
dense
dense
dense
120 - F5 full
84 - F6 full
10 - Out
28x28 image
6@28x28
C1 feature map
6@14x14
S2 feature map
16@10x10
C3 feature map
16@5x5
S4 feature map

表示学习

- CNN以图像的原始像素作为输入，基于输出层定义的损失函数使用反向传播算法端到端(End-to-end)学习，从而自动学习得到图像底层到高层的层次化语义表达



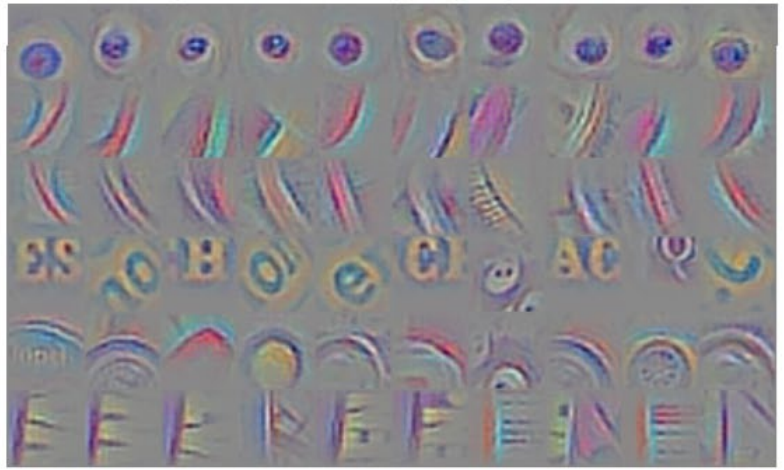
表示学习


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

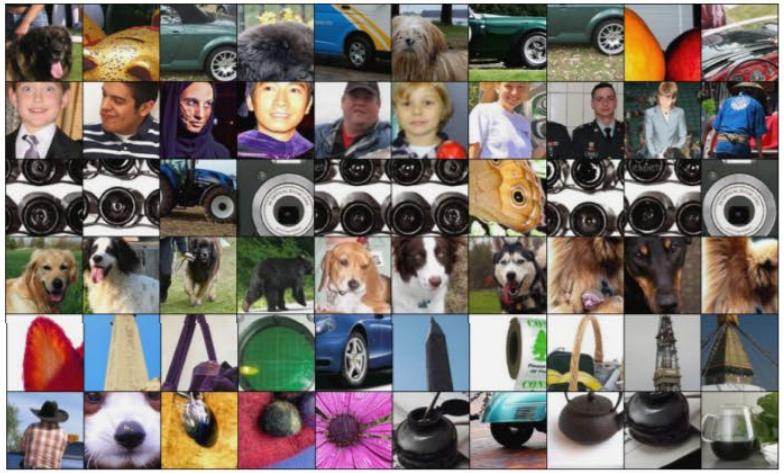
卷积神经网络学到了什么?

- 浅层学习局部特征，深层学习整体特征

conv6



conv9



Springenberg, J. T.; Dosovitskiy, A.; Brox, T. & Riedmiller, M. Striving for simplicity: the all convolutinal net ICML, 2015, 1-12

如何设计卷积神经网络

- **选择“深”而非“广”的网络结构**
  - 深度网络可从局部到整体“理解图像”：学习复杂特征时（例如人脸识别），浅层的卷积层感受野小，学习到局部特征，深层的卷积层感受野大，学习到整体特征。
  - 深度网络可减少权重数量：以宽度换深度，用多个小卷积替代一个大卷积，在获得更多样特征的同时所需权重数量也更少。

卷积神经网络可视化

Convolution Visualizer (ezyang.github.io)

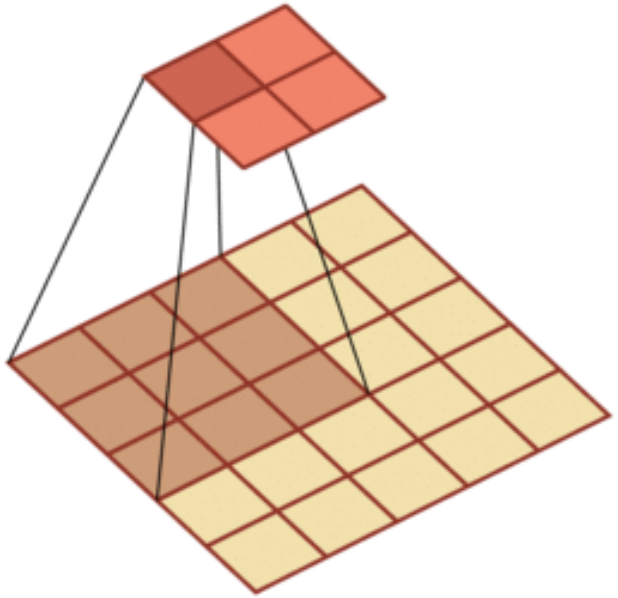
2DConv on RGB image (thomelane.github.io)

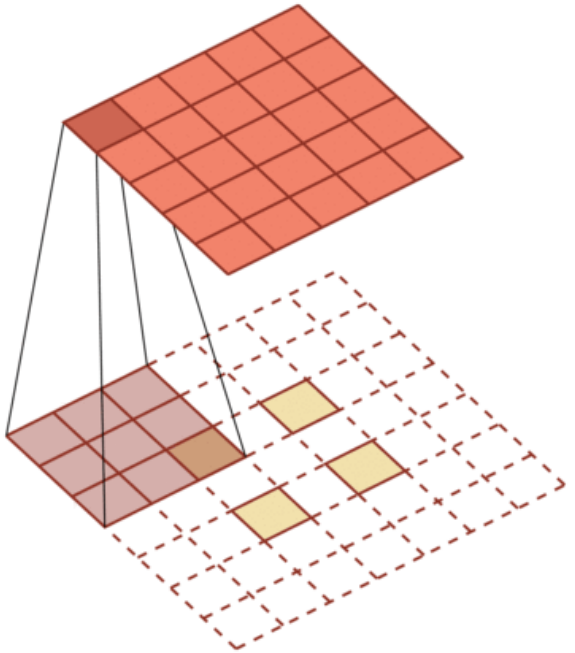
CNN Explainer (poloclub.github.io)

其他类型的卷积

转置卷积 / 微步卷积（Transpose / Fractional-Stride Convolution）

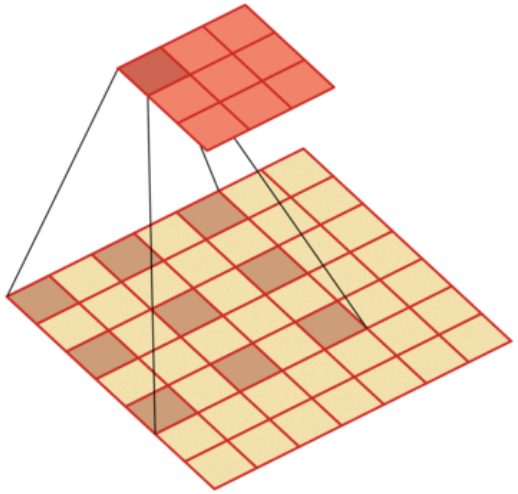
- 下采样（Down-sampling）和上采样（Up-sampling）
- 低维特征映射到高维特征



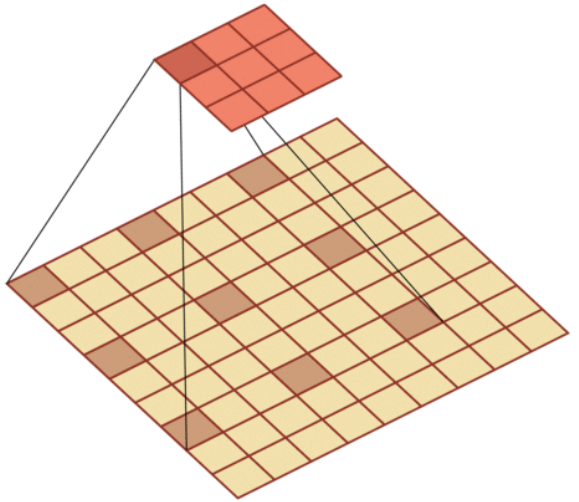


空洞卷积（Dilated Convolution）

- **如何增加输出单元的感受野**
  - 增加卷积核的大小
  - 增加层数来实现
  - 在卷积之前进行汇聚操作

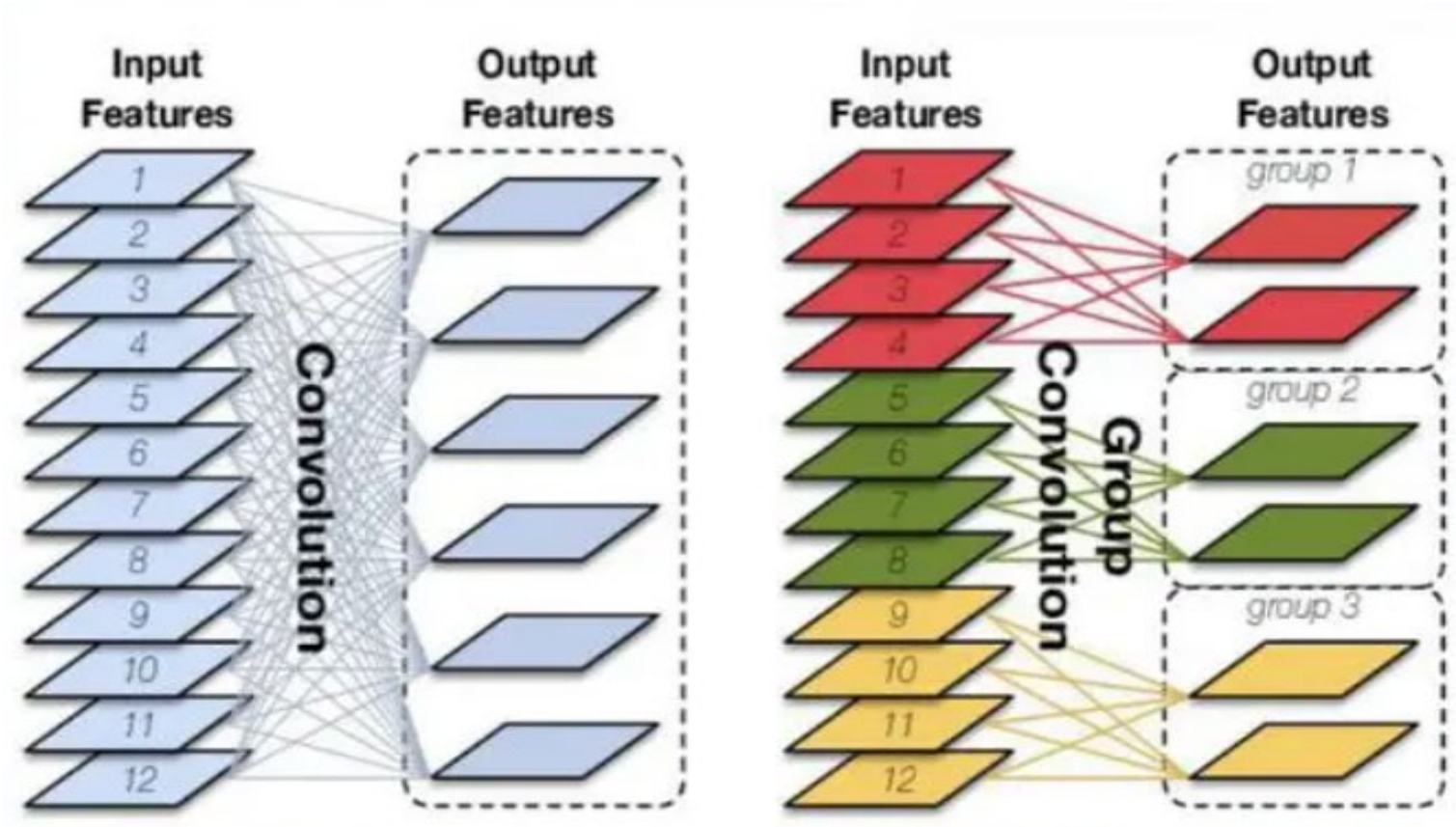


- **空洞卷积**
  - 通过给卷积核插入“空洞”来变相地增加其大小
  - 通过扩张率（Dilation Rate）调节“空洞”大小



分组卷积（Group Convolution）

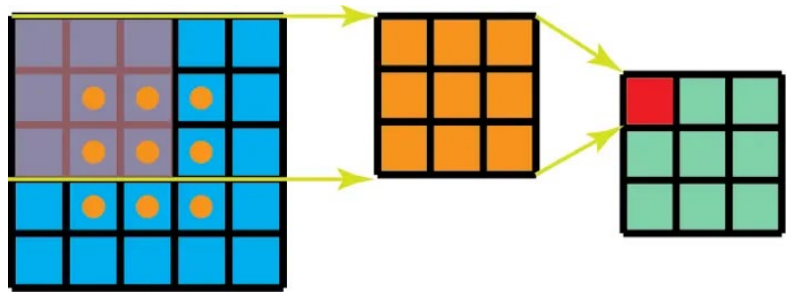
- **将输入特征图按通道分为若干组，对每组分别进行常规卷积；**
  - 分组卷积最早被用于AlexNet，现在被广泛用于各种轻量化模型中，用于减少运算量和参数量。



可分离卷积（Separable Convolution）

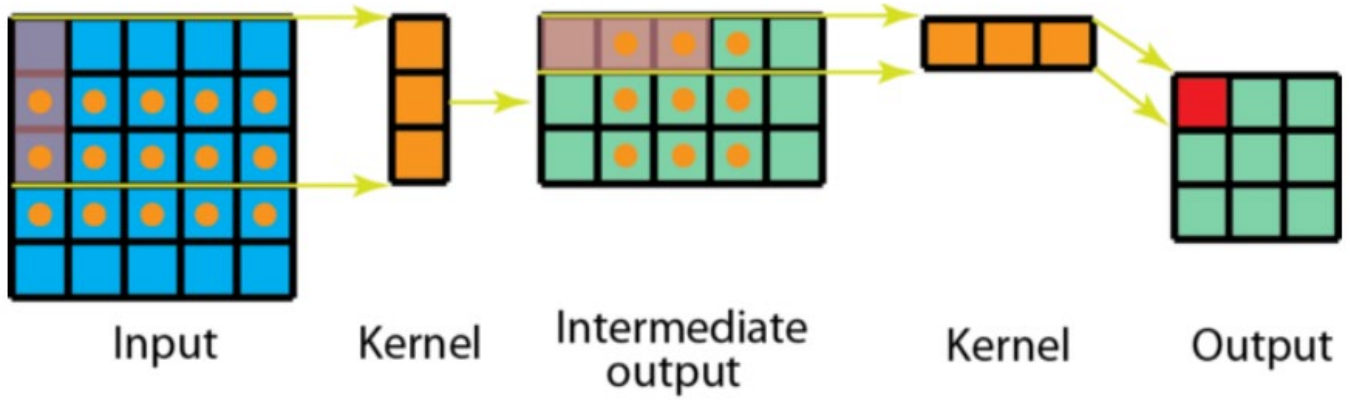
- 分为空间可分离卷积、深度可分离卷积；

标准卷积



81次乘法运算

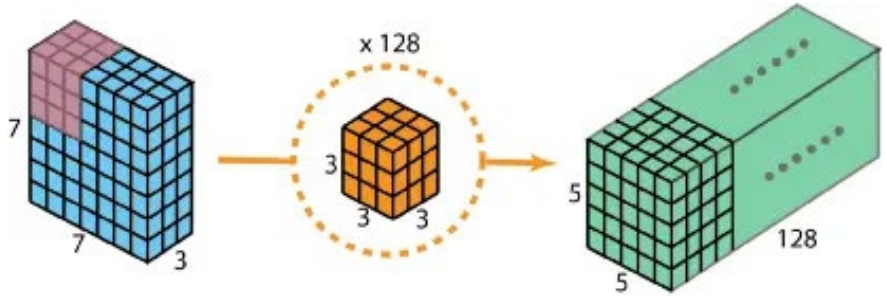
空间可分离卷积



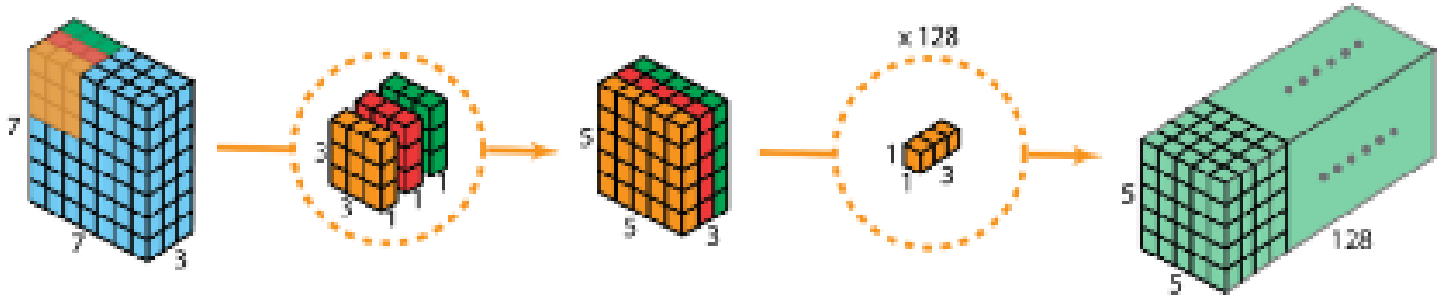
72次乘法运算

可分离卷积（Separable Convolution）

- 分为空间可分离卷积、深度可分离卷积；



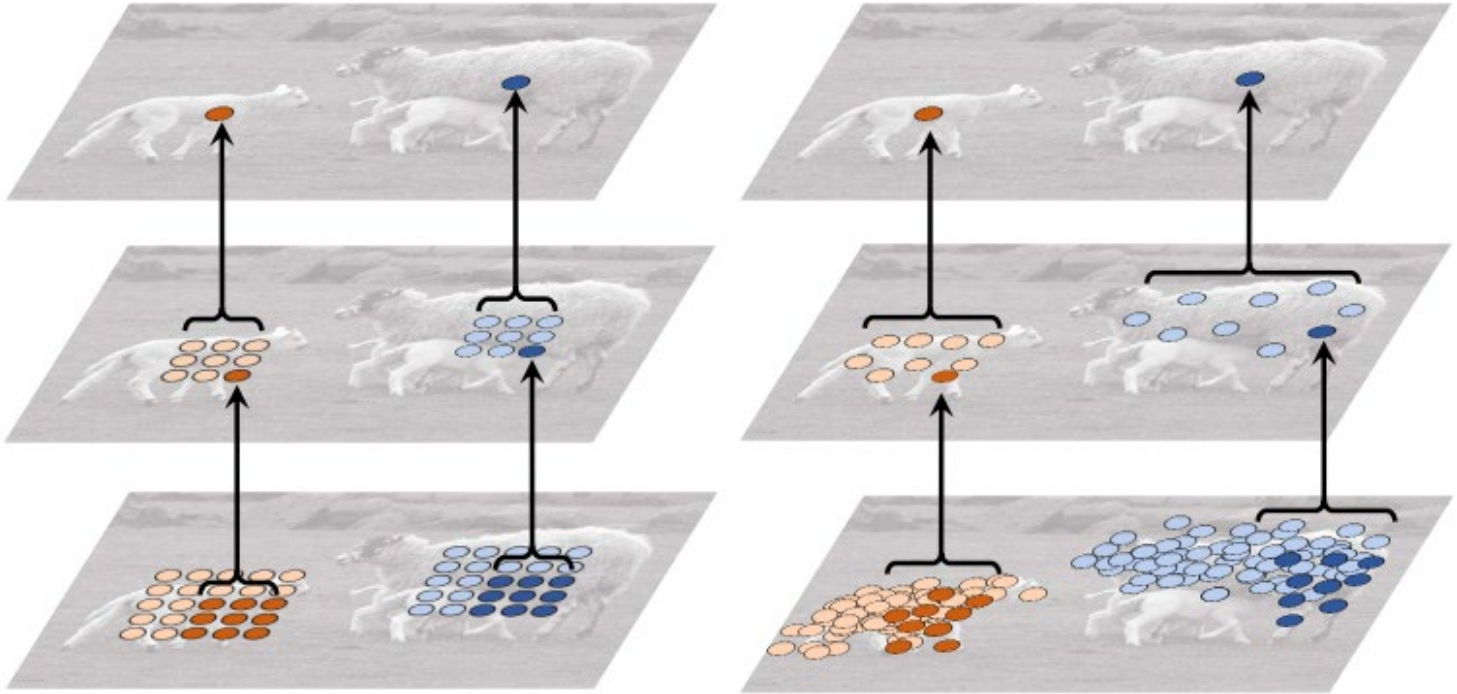
标准卷积，每个卷积核得到1个Channel的Feature Map



深度可分离卷积，在输入的每个Channel分别进行卷积，然后利用若干个1×1卷积进行通道融合，得到最终的Feature Map

可变形卷积（Separable Convolution）

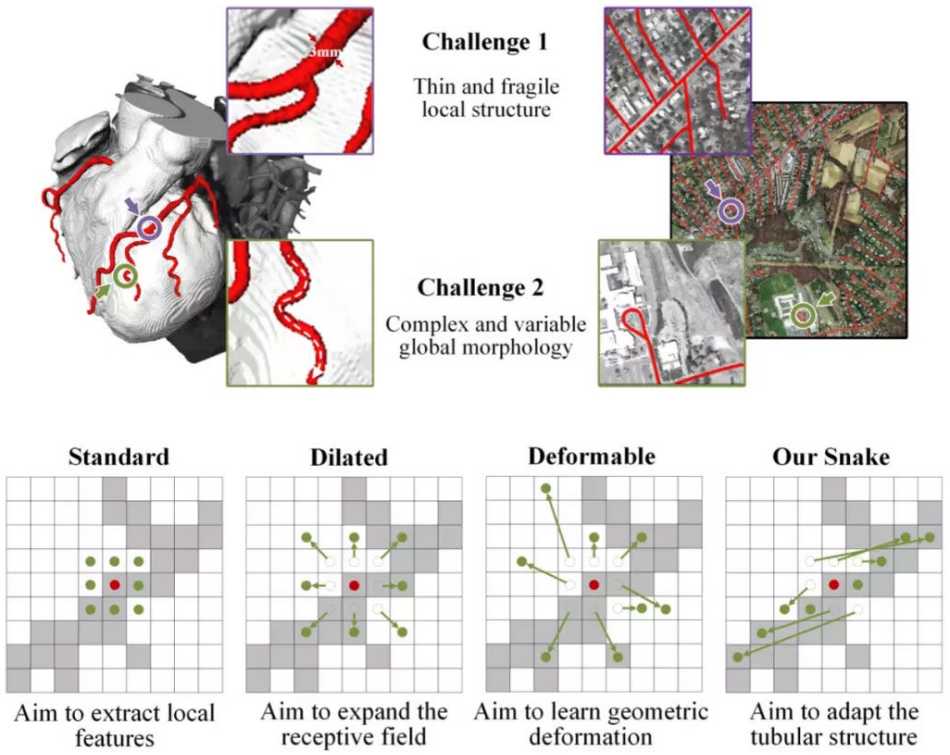
- **可变形卷积的采样位置是可变、可学习的**
  - 常规卷积的卷积核为固定的大小与形状，对于形状规则的物体效果较好，但不一定适用于形状复杂的物体。

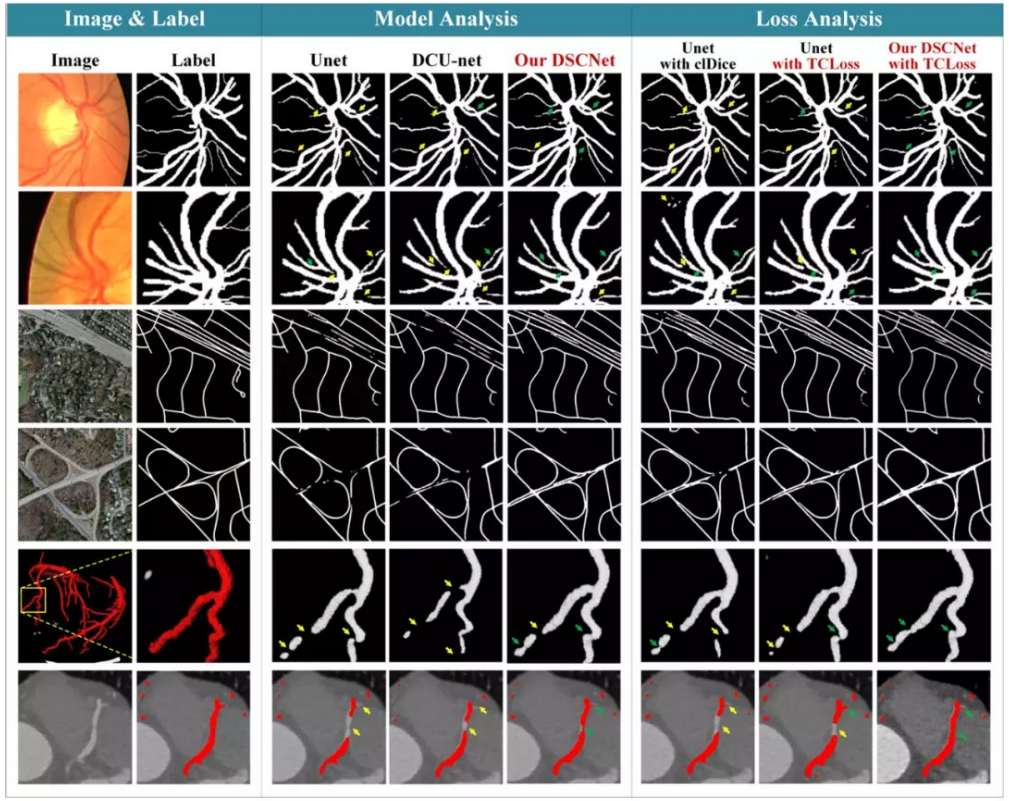


(a) standard convolution　　(b) deformable convolution

动态蛇形卷积（Dynamic Snake Convolution）

- **将连续性约束加入卷积核的设计中**
  - 主要用于在图像分割任务中提取细长微弱的局部结构

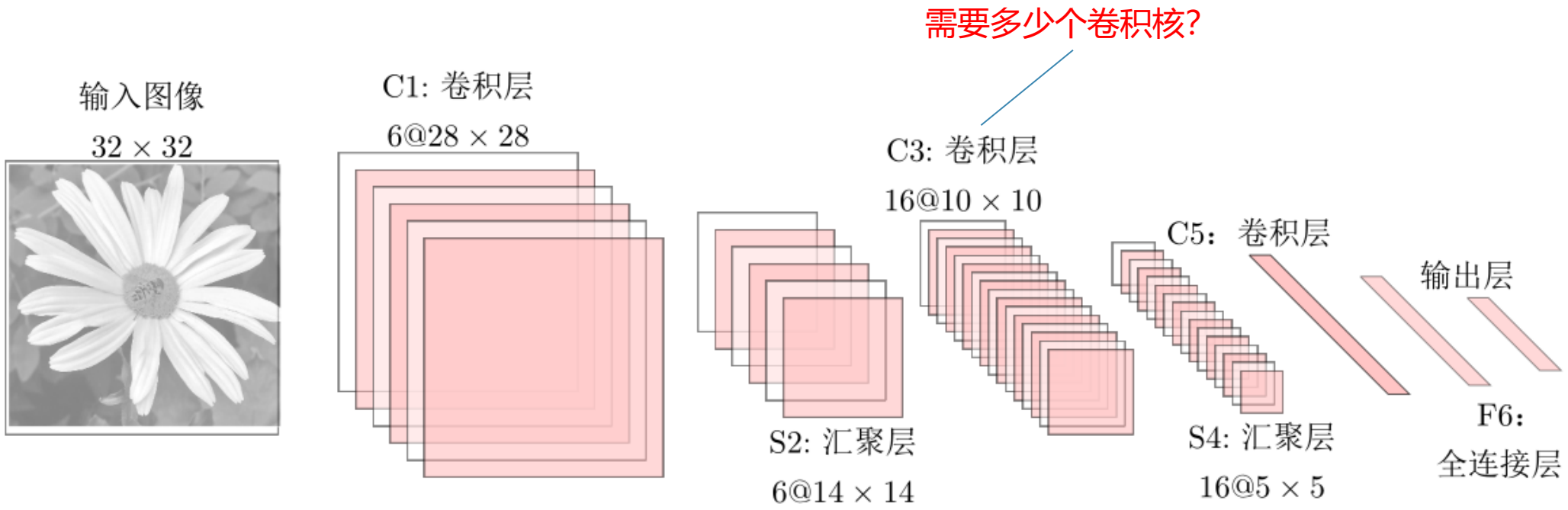




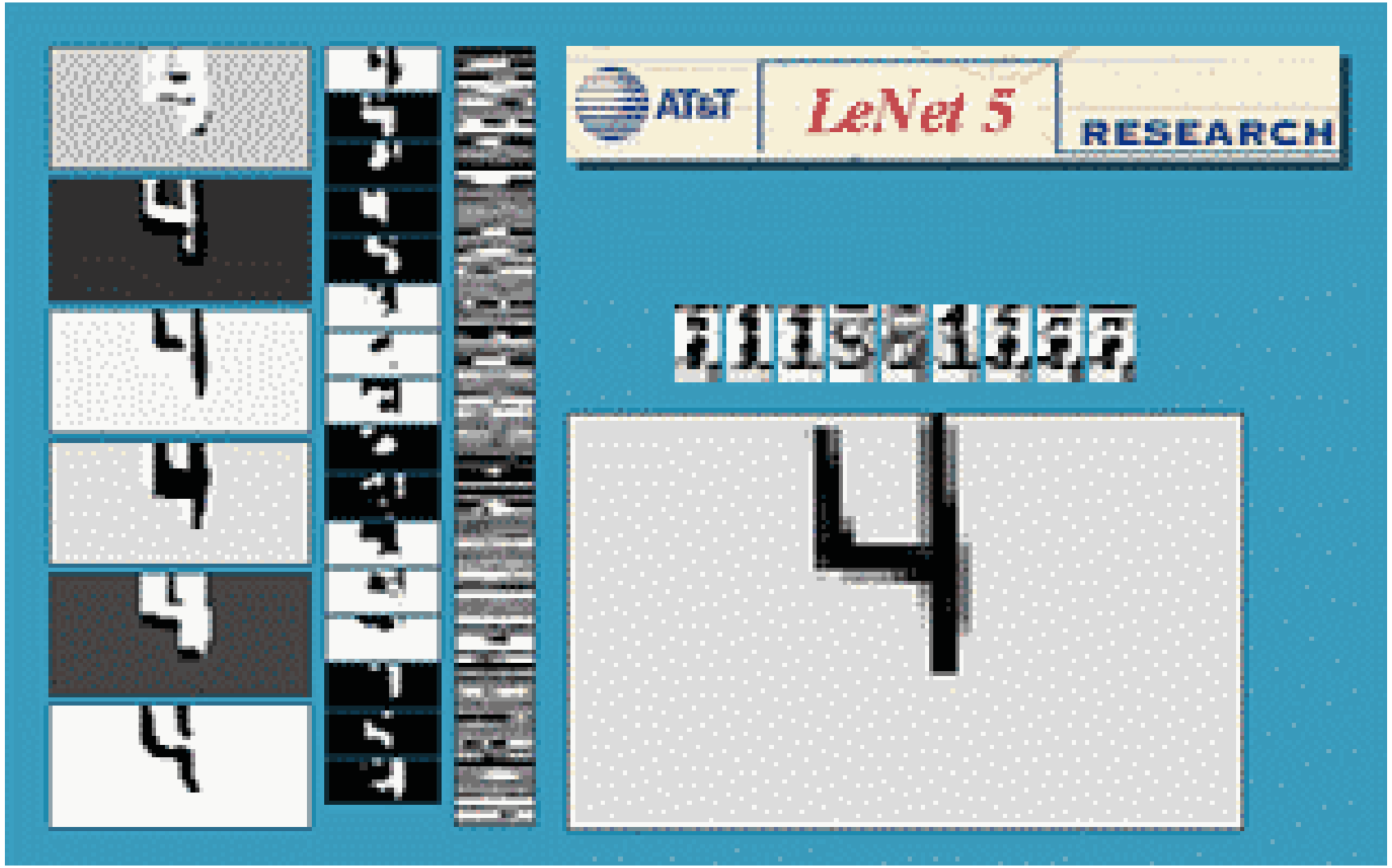
经典卷积神经网络结构

LeNet-5

- **LeNet-5 是一个早期非常成功的神经网络模型。**
  - 由Yann LeCun（AT&T Bell Lab）提出；
  - 基于 LeNet-5 的手写数字识别系统在 90 年代被美国很多银行使用，用来识别支票上面的手写数字；
  - LeNet-5 共有 7 层，其中5层具有可学习参数。

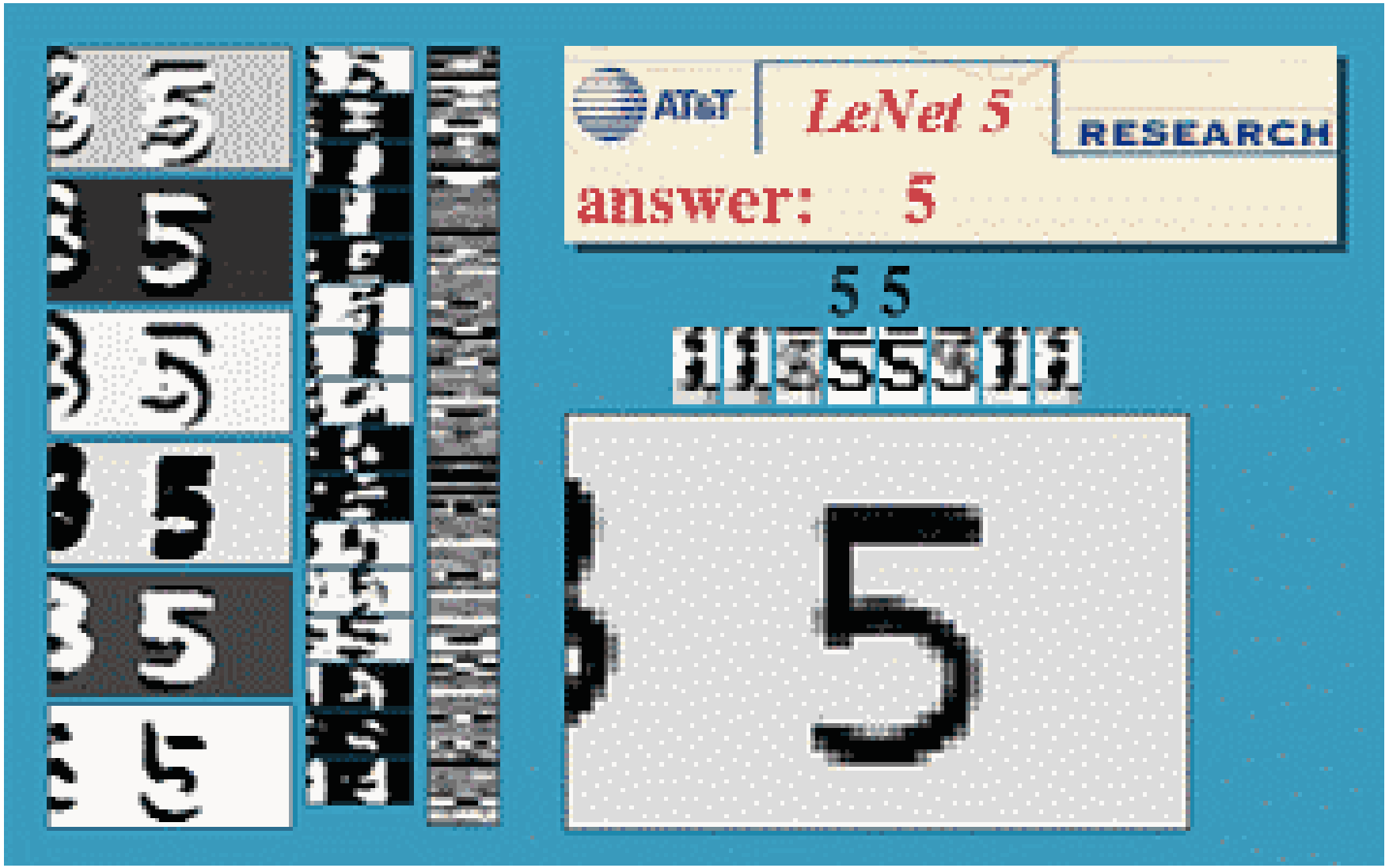


LeNet-5



https://atcold.github.io/pytorch-Deep-Learning/zh/week03/03-2/

LeNet-5



https://atcold.github.io/pytorch-Deep-Learning/zh/week03/03-2/

LeNet-5

- **MNIST**
  - 50,000训练数据；
  - 10,000测试数据；
  - 图像大小为28×28；
  - 10个类别。

AlexNet

Large Scale Visual Recognition Challenge

AlexNet

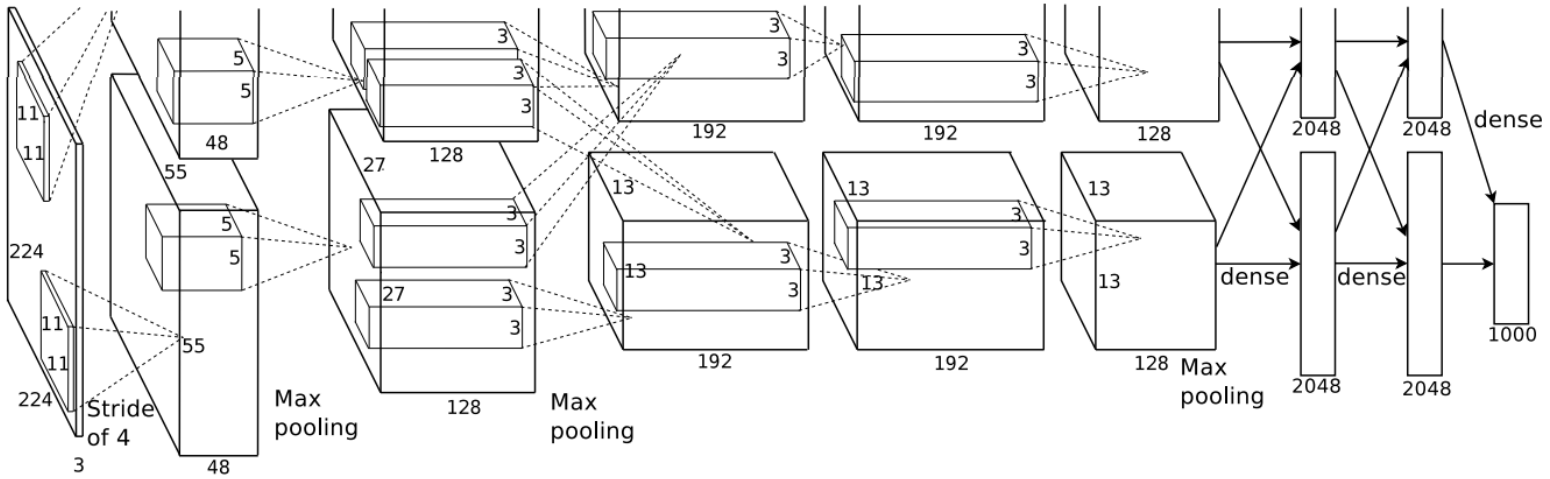
2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

AlexNet

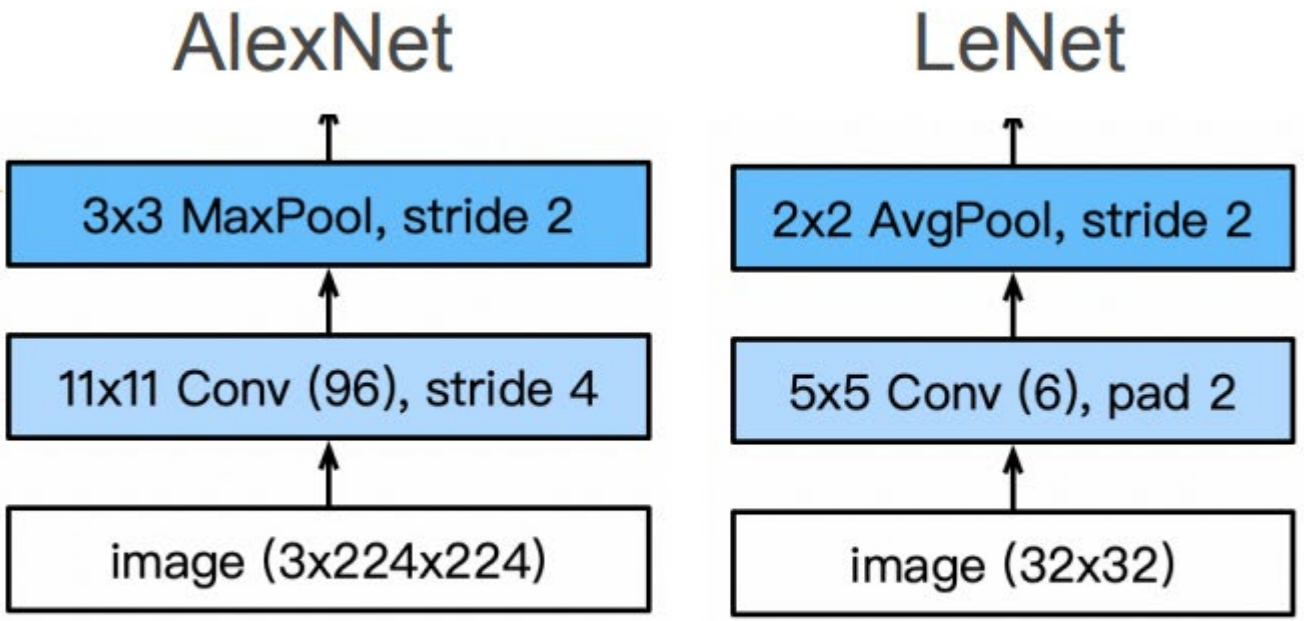
- **2012 ILSVRC winner**
  - top 5 error of 16% compared to runner-up with 26% error
  - 第一个现代深度卷积网络模型
    - 首次使用了很多现代深度卷积网络的一些技术方法：
    - 使用GPU进行并行训练，采用了ReLU作为非线性激活函数，使用Dropout防止过拟合，使用MaxPooling方法而不是AvgPooling，使用数据增强
  - 5个卷积层、3个汇聚层和3个全连接层



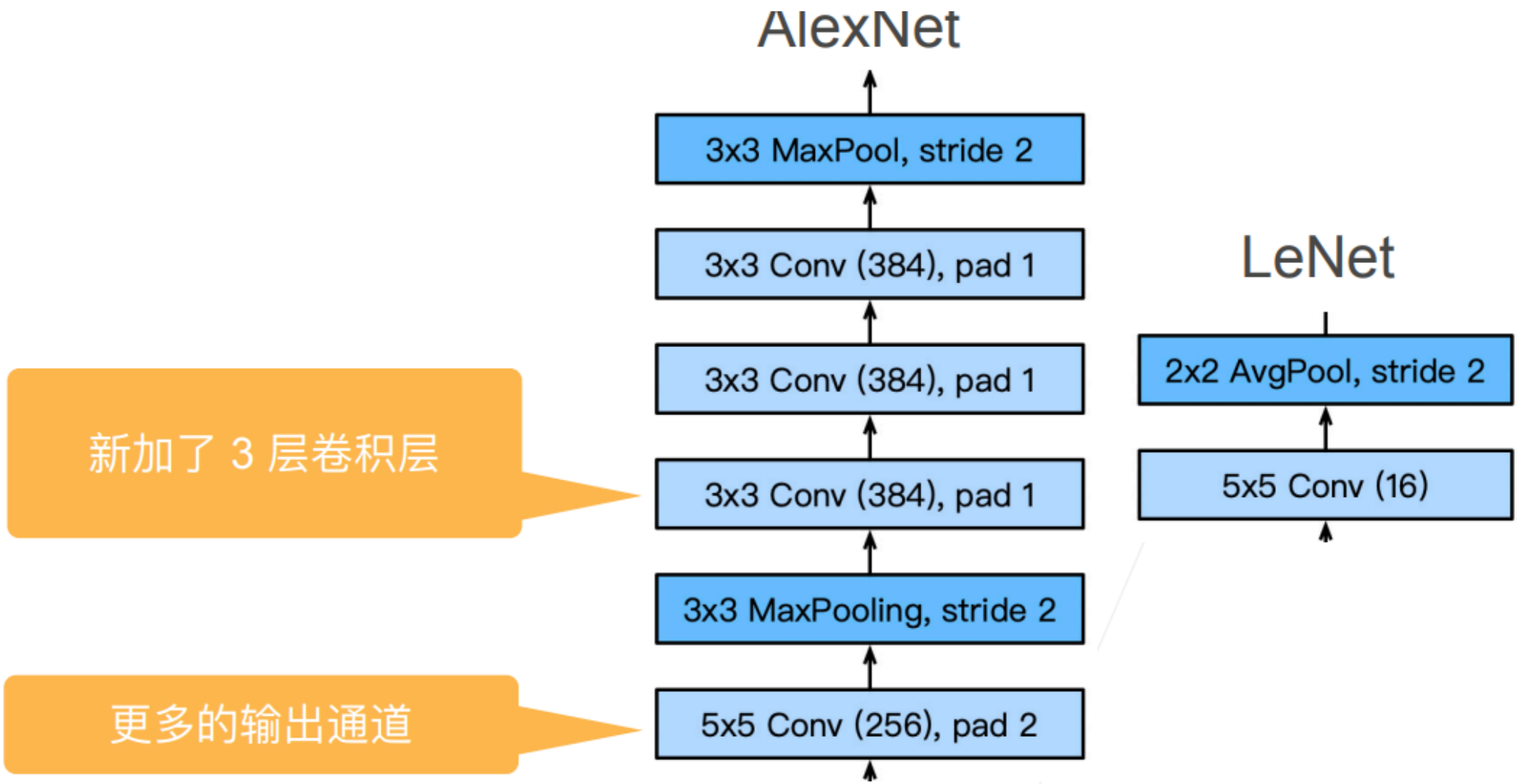
AlexNet

更大的池化窗口，使用最大池化层

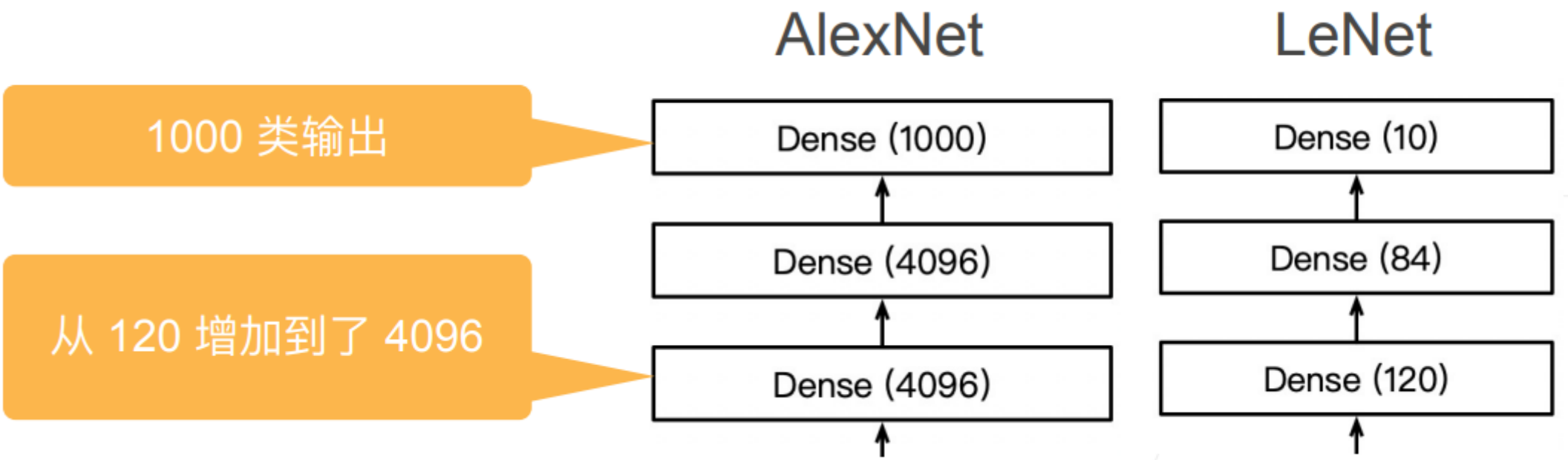
更大的核窗口和步长，因为图片更大了



AlexNet


AlexNet
3x3 MaxPool, stride 2
3x3 Conv (384), pad 1
3x3 Conv (384), pad 1
3x3 Conv (384), pad 1
3x3 MaxPooling, stride 2
5x5 Conv (256), pad 2
新加了 3 层卷积层
更多的输出通道
LeNet
2x2 AvgPool, stride 2
5x5 Conv (16)

AlexNet


AlexNet
LeNet
1000 类输出
从 120 增加到了 4096
Dense (1000)
Dense (4096)
Dense (4096)
Dense (10)
Dense (84)
Dense (120)

AlexNet

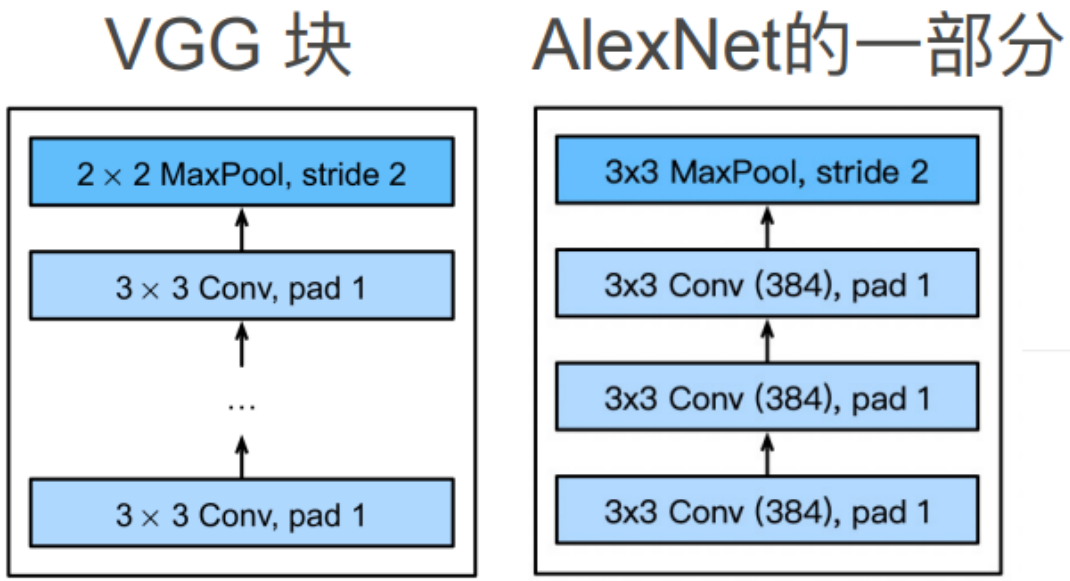
- 激活函数从Sigmoid改为ReLU（缓解梯度消失问题）
- 添加Dropout处理
- 数据增强

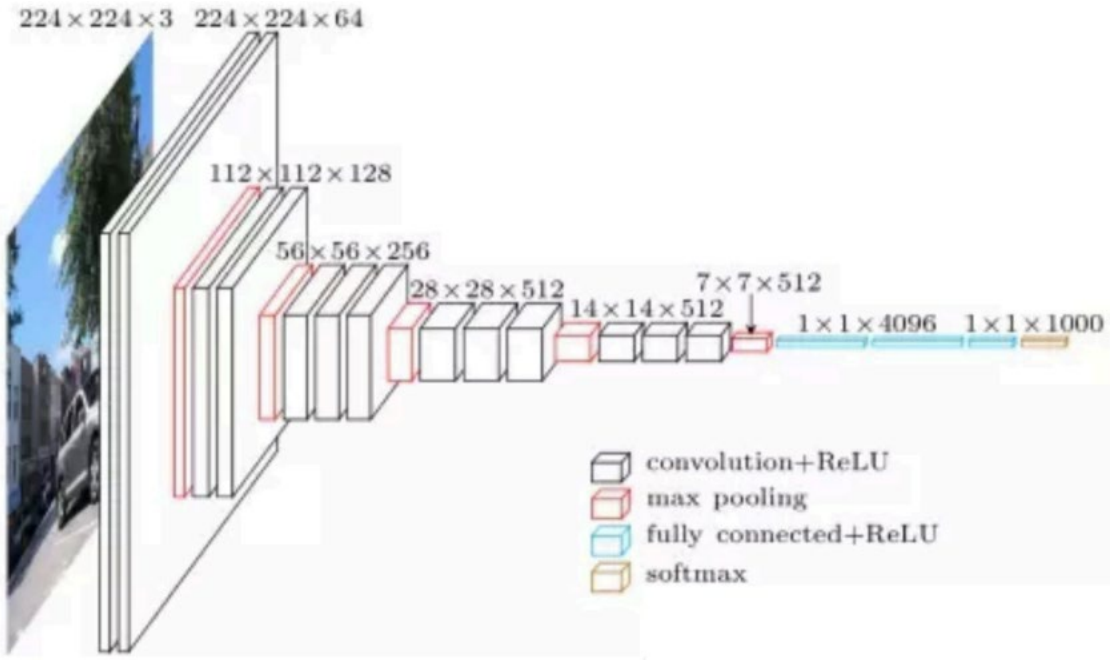
AlexNet

	参数个数		FLOP	
	AlexNet	LeNet	AlexNet	LeNet
Conv1	35K	150	101M	1.2M
Conv2	614K	2.4K	415M	2.4M
Conv3-5	3M		445M	
Dense1	26M	0.48M	26M	0.48M
Dense2	16M	0.1M	16M	0.1M
Total	46M	0.6M	1G	4M
Increase	11x	1x	250x	1x

VGG

- VGG：便于构建更深、更大、结构化的卷积神经网络
- 核心思想：将卷积层组合成块（VGG Block）
- 使用堆叠的小卷积核，而不是一个大卷积核





卷积神经网络的主要问题

- **卷积层需要较少的参数，但卷积层后的第一个全连接层参数规模庞大：**

LeNet 16x5x5x120 = 48k

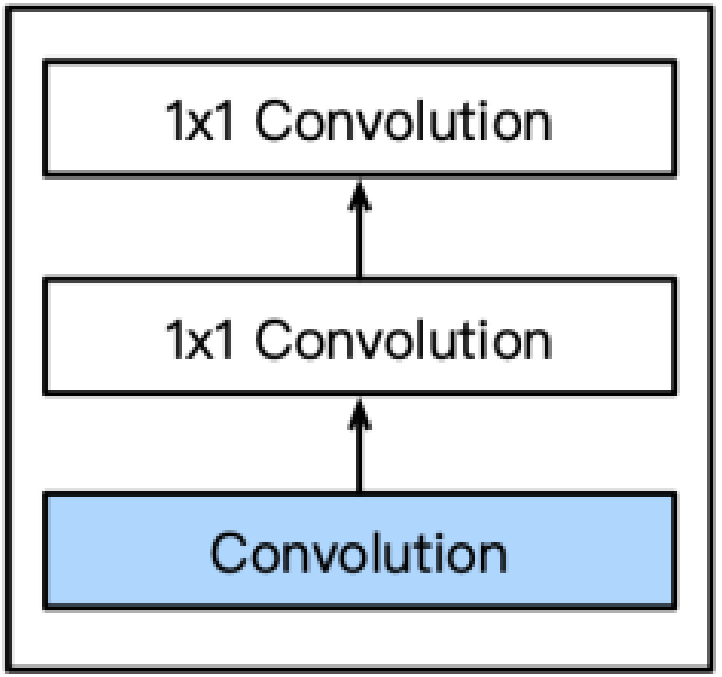
AlexNet 256x5x5x4096 = 26M

VGG 512x7x7x4096 = 102M

- **内存、计算带宽消耗巨大；**
- **容易过拟合。**

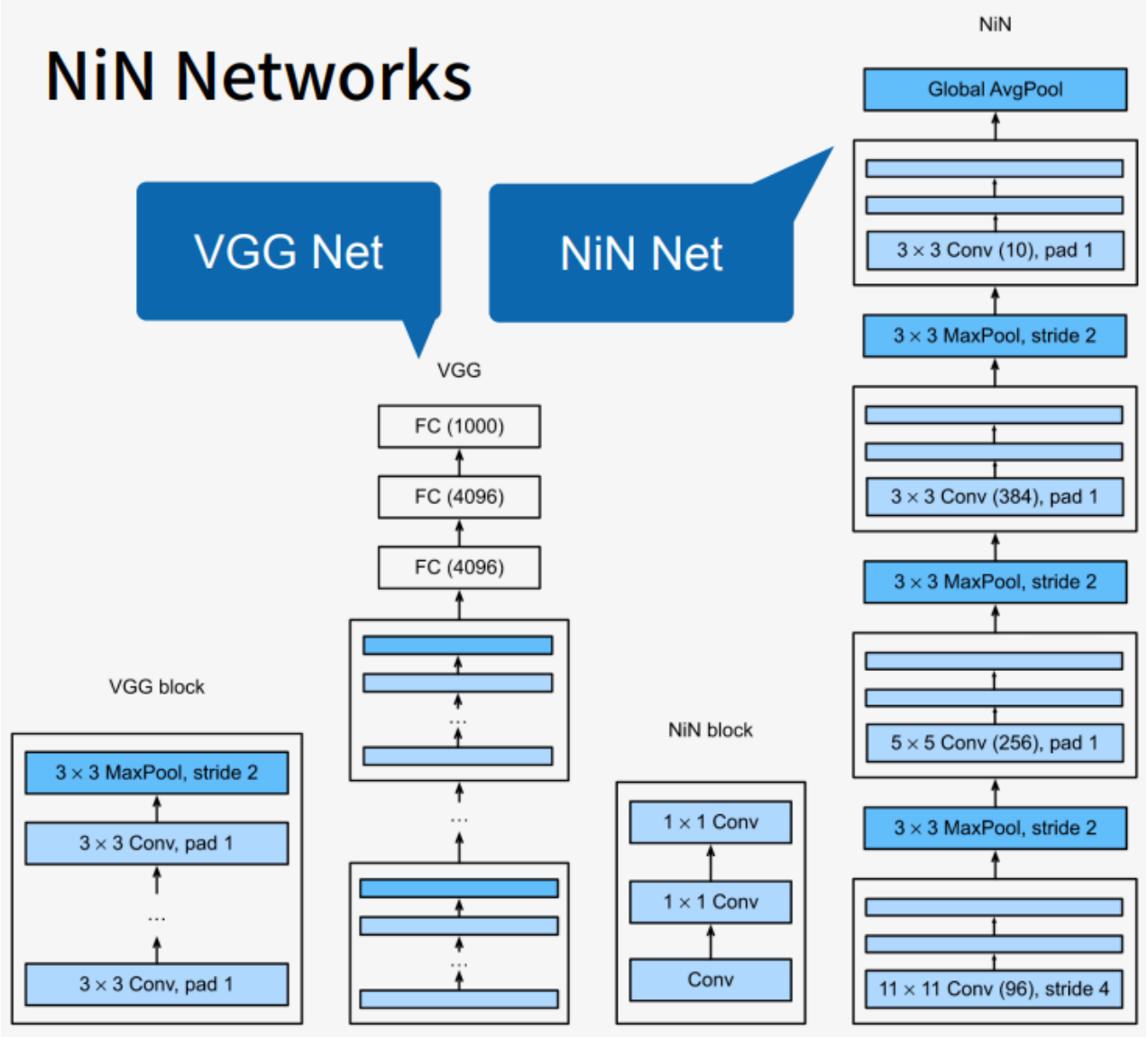
NiN

- **Network in Network：“网络中的网络”；**
- **NiN块：**
  - 一个卷积层后面跟两个1×1卷积层（相当于全连接）；
  - 步长（Stride）为1，无填充（Padding）；
  - 输出形状与卷积层输出一致；
  - 代替了全连接层的作用。
- **NiN的结构特点：**
  - 不使用全连接层；
  - 交替使用NiN块和步长为2的MaxPooling层；
  - 最后使用全局平均池化层（GAP）得到输出（输入通道数是类别数）。



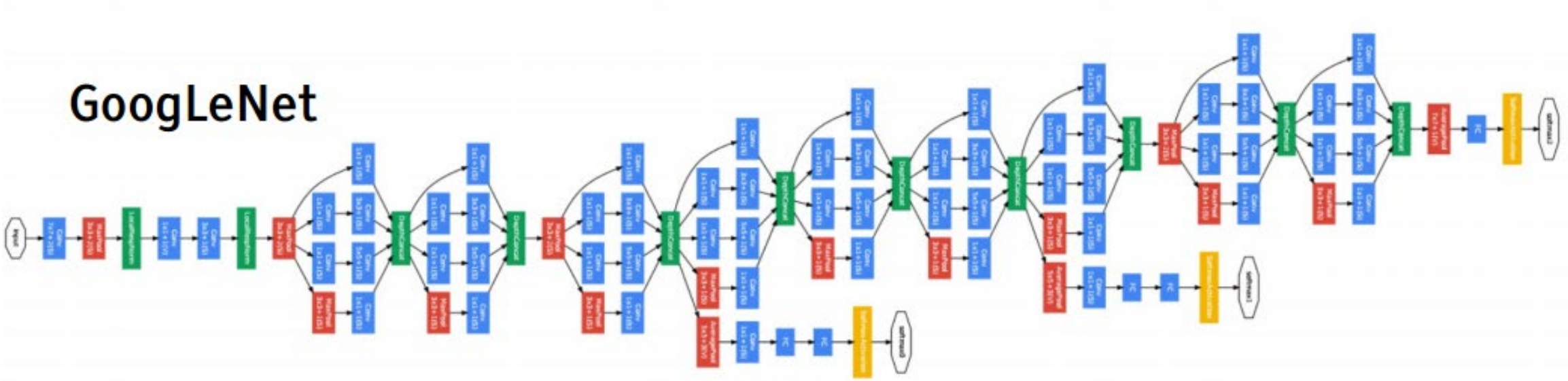
NiN

- **NiN块使用卷积层加两个1×1卷积层**
  - 后者对每个像素增加了非线性性
- **NiN使用全局平均池化代替VGG和AlexNet中的全连接层**
  - 不容易过拟合，更少的参数个数



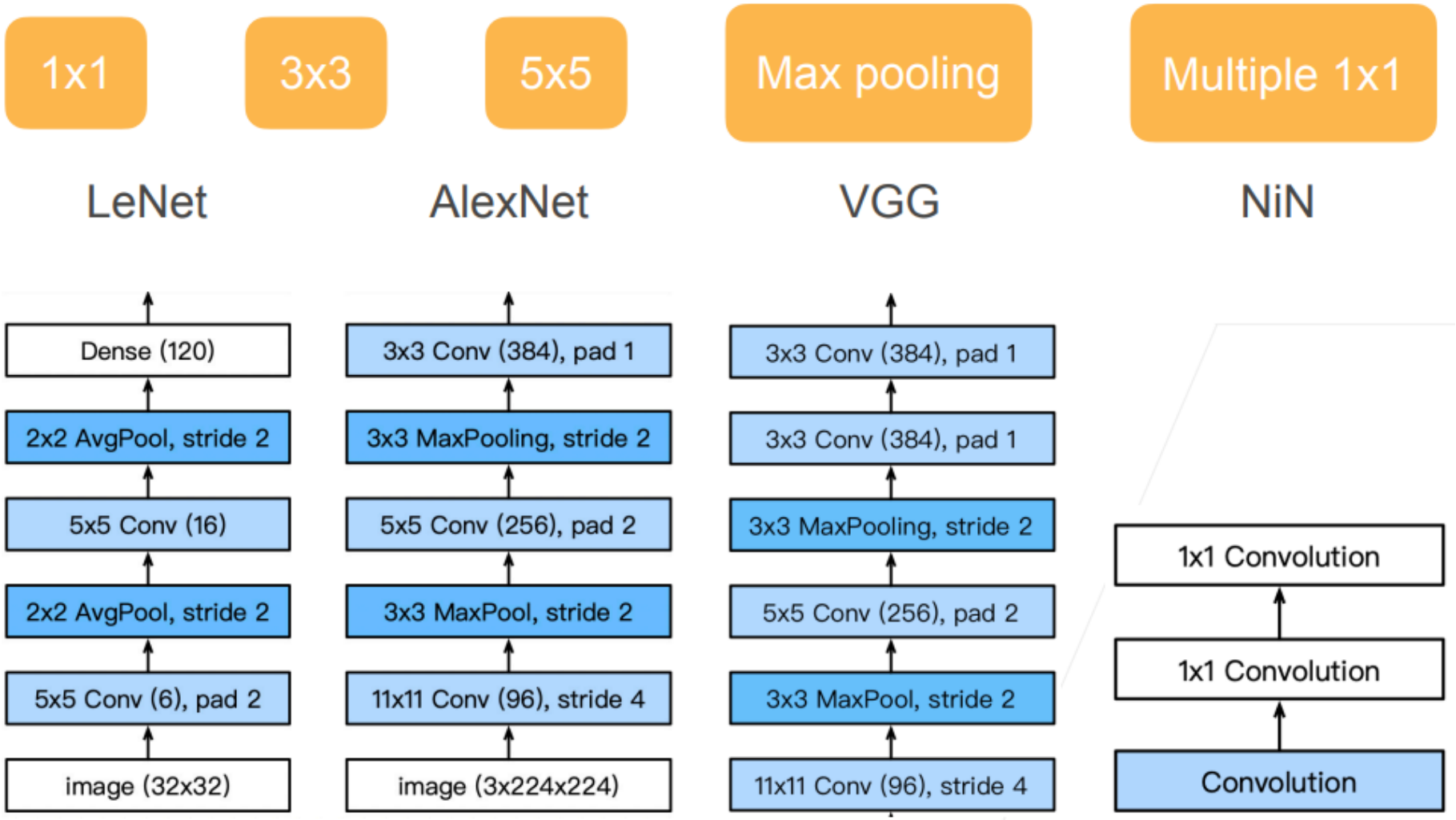
GoogLeNet

- **2014 ILSVRC winner （22层）**
  - 参数：GoogLeNet：4M VS AlexNet：60M
  - 错误率：6.7%
  - Inception网络是由有多个inception模块和少量的汇聚层堆叠而成。



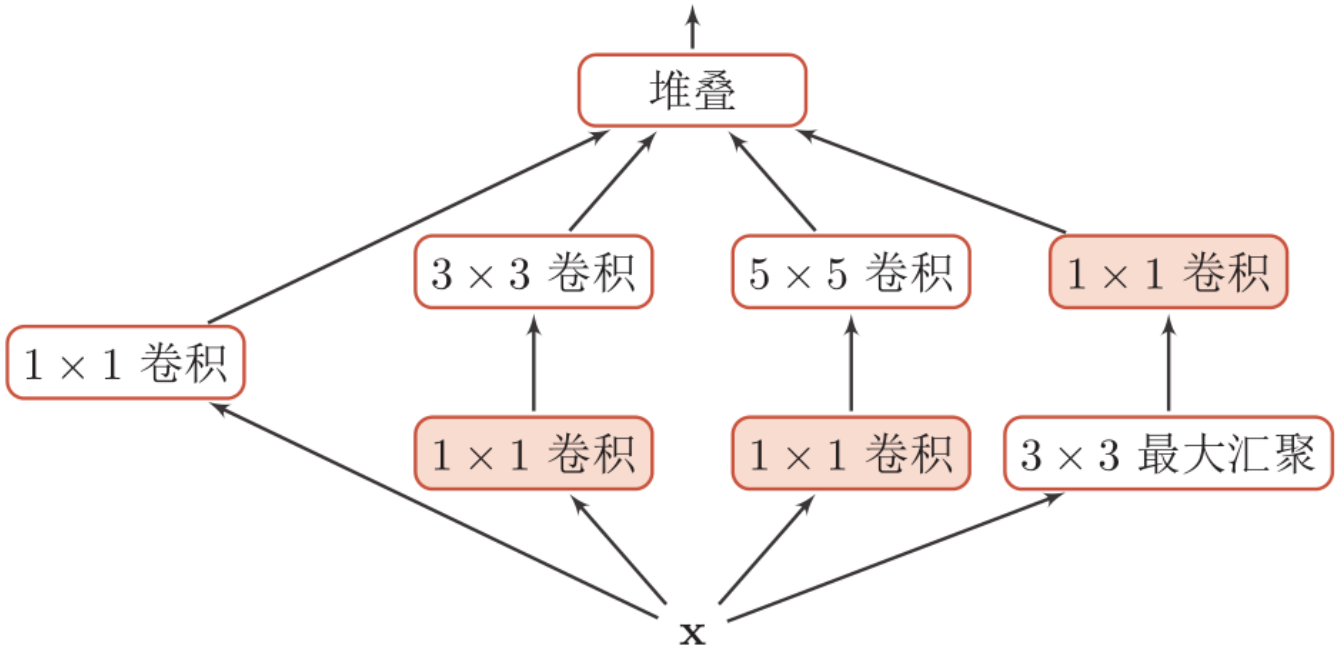
GoogLeNet

- 如何确定卷积层超参数?



Inception模块 v1

- **在卷积网络中，如何设置卷积层的卷积核大小是一个十分关键的问题。**
  - 在Inception网络中，一个卷积层包含多个不同大小的卷积操作，称为Inception模块。
  - Inception模块同时使用1 × 1、3 × 3、5 × 5等不同大小的卷积核，并将得到的特征映射在深度上拼接（堆叠）起来作为输出特征映射。



Inception模块 v3

- **用多层小卷积核替换大卷积核，以减少计算量和参数量。**
  - 使用两层3x3的卷积来替换v1中的5x5的卷积
  - 使用连续的nx1和1xn来替换nxn的卷积。

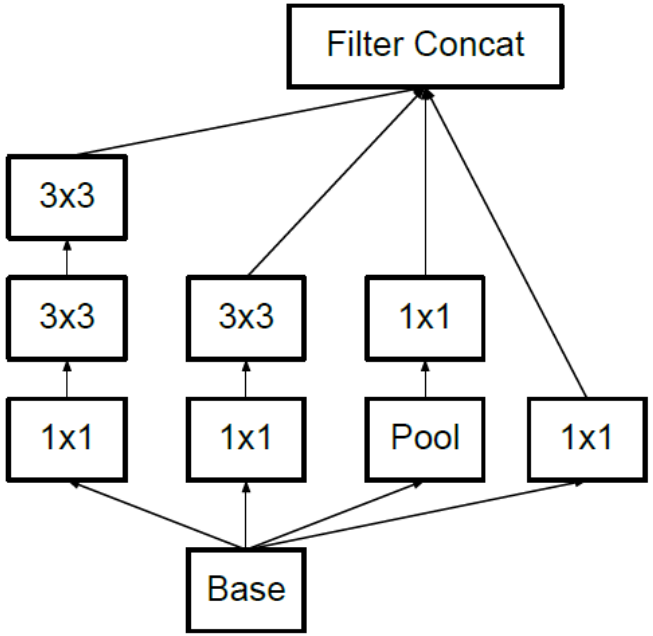


Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle 3 of Section 2.

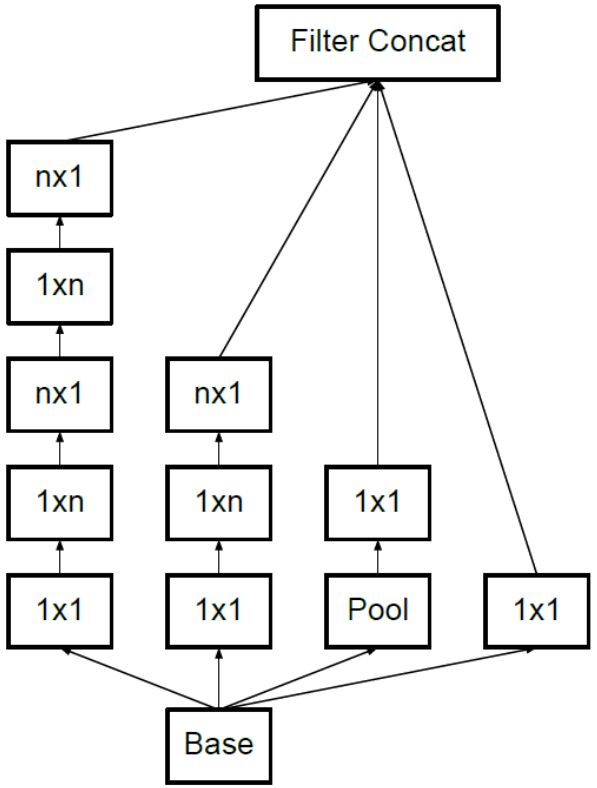


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle 3)

Inception总结

- 以Inception块为基本结构，该结构包含4条有不同超参数的卷积、池化的路径抽取不同层面的信息；
- 模型参数少，计算复杂度低；
- 是第一个可以达到上百层的网络。

ResNet

- 堆叠更多的层总是能提高精度吗?
- 网络退化现象

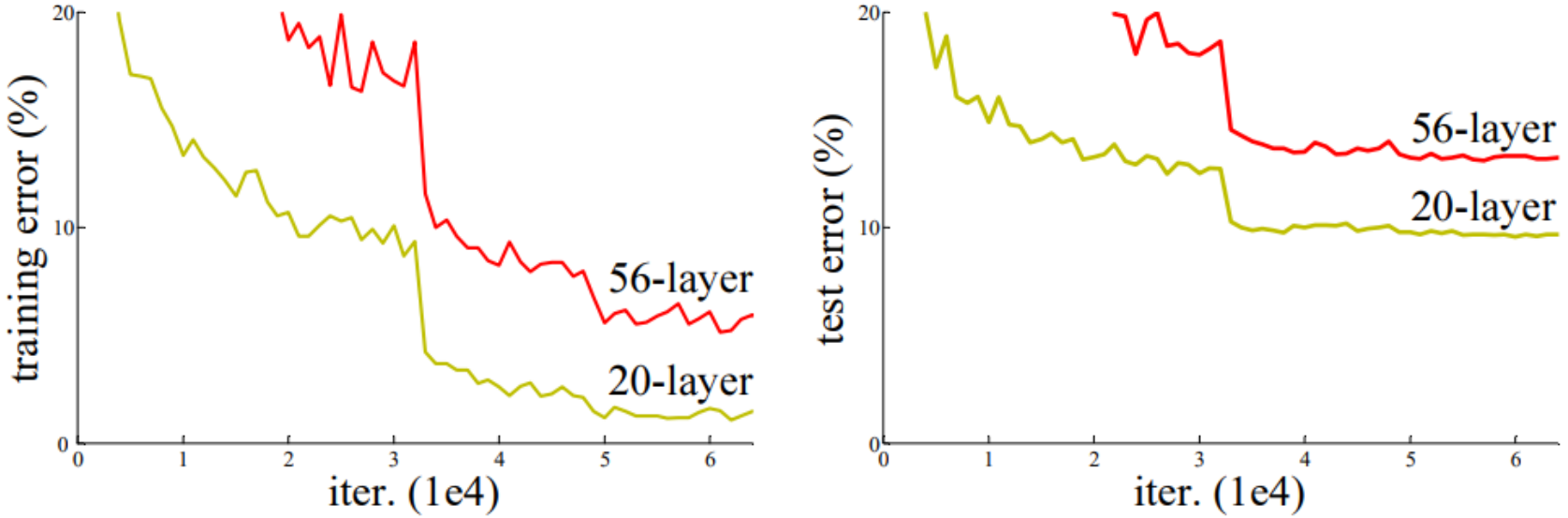
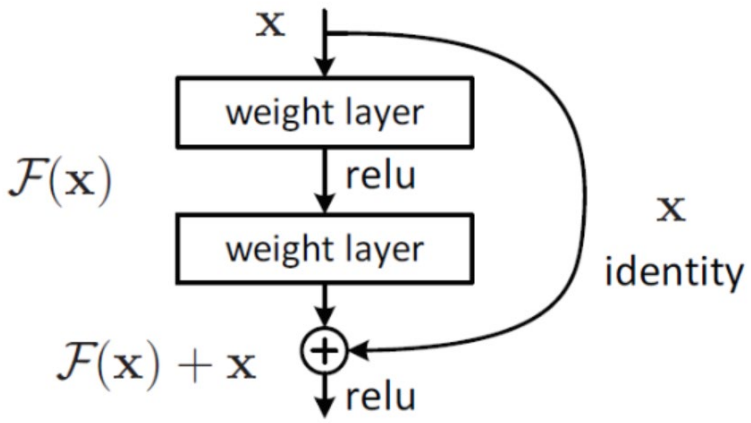


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

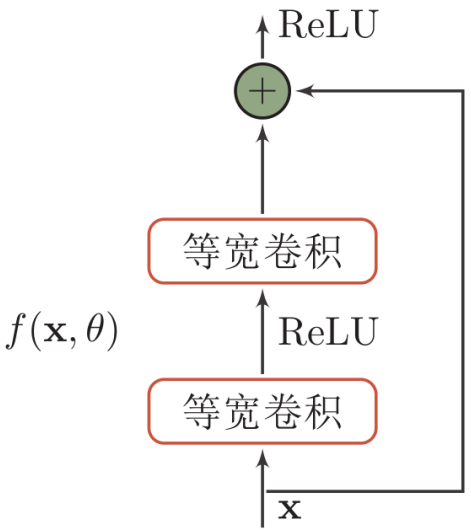
ResNet

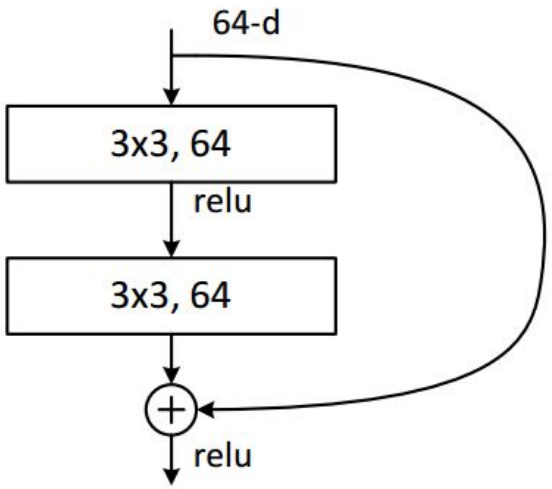
- **残差网络（Residual Network，ResNet）通过给非线性的卷积层增加直连边（Shortcut）的方式来提高信息的传播效率。**
  - 假设在一个深度网络中，我们期望一个非线性单元（可以为一层或多层的卷积层）f(x,θ)去逼近一个目标函数为h(x)。
  - 将目标函数拆分成两部分：恒等函数和残差函数

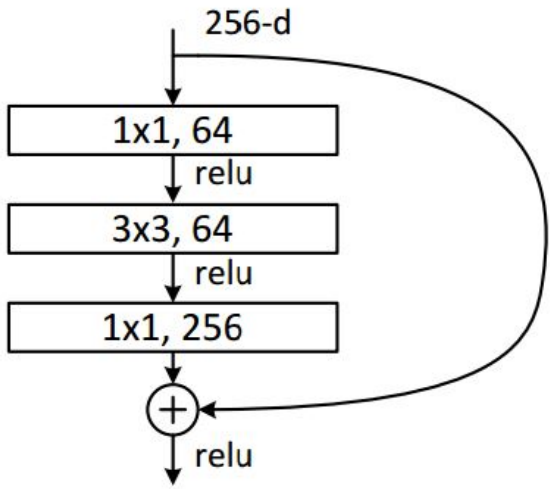
$$h(\mathbf{x}) = \underbrace{\mathbf{x}}_{\text{恒等函数}} + \underbrace{\big(h(\mathbf{x}) - \mathbf{x}\big)}_{\text{残差函数}} \rightarrow f(\mathbf{x}, \theta)$$



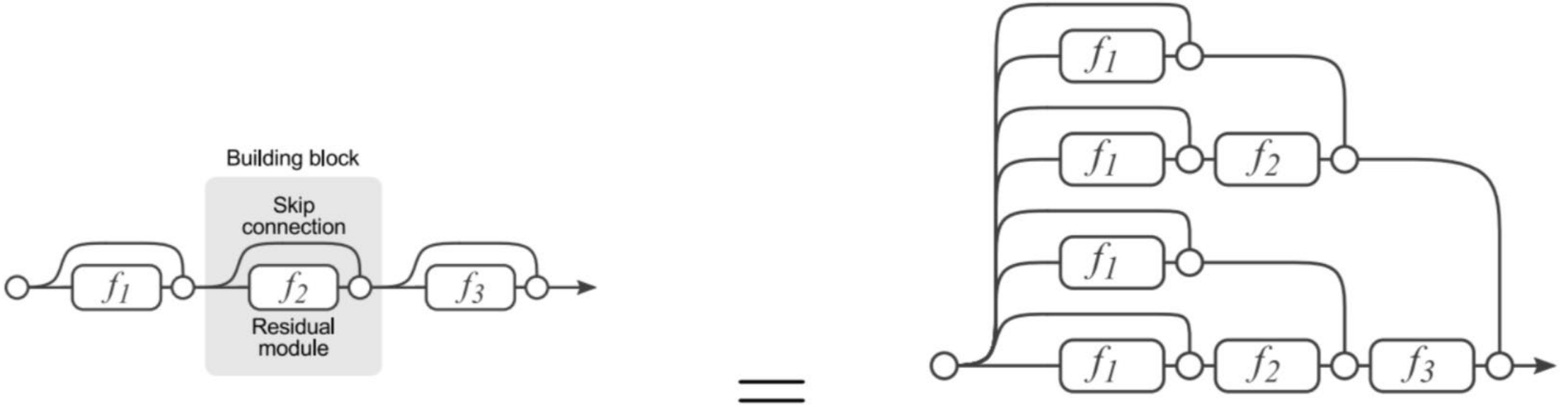
ResNet


ReLU
等宽卷积
$f(\mathbf{x}, \theta)$
ReLU
等宽卷积
x


64-d
3x3, 64
relu
3x3, 64
relu


256-d
1x1, 64
relu
3x3, 64
relu
1x1, 256
relu

ResNet

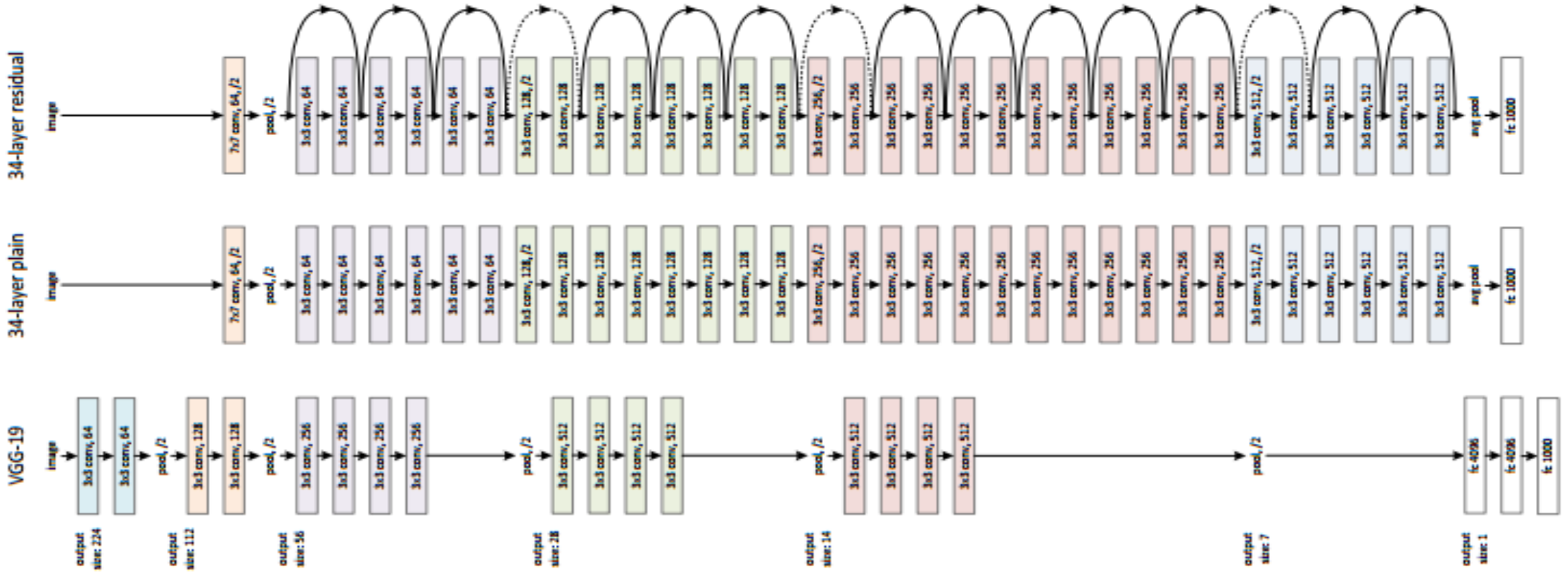


(a) Conventional 3-block residual network

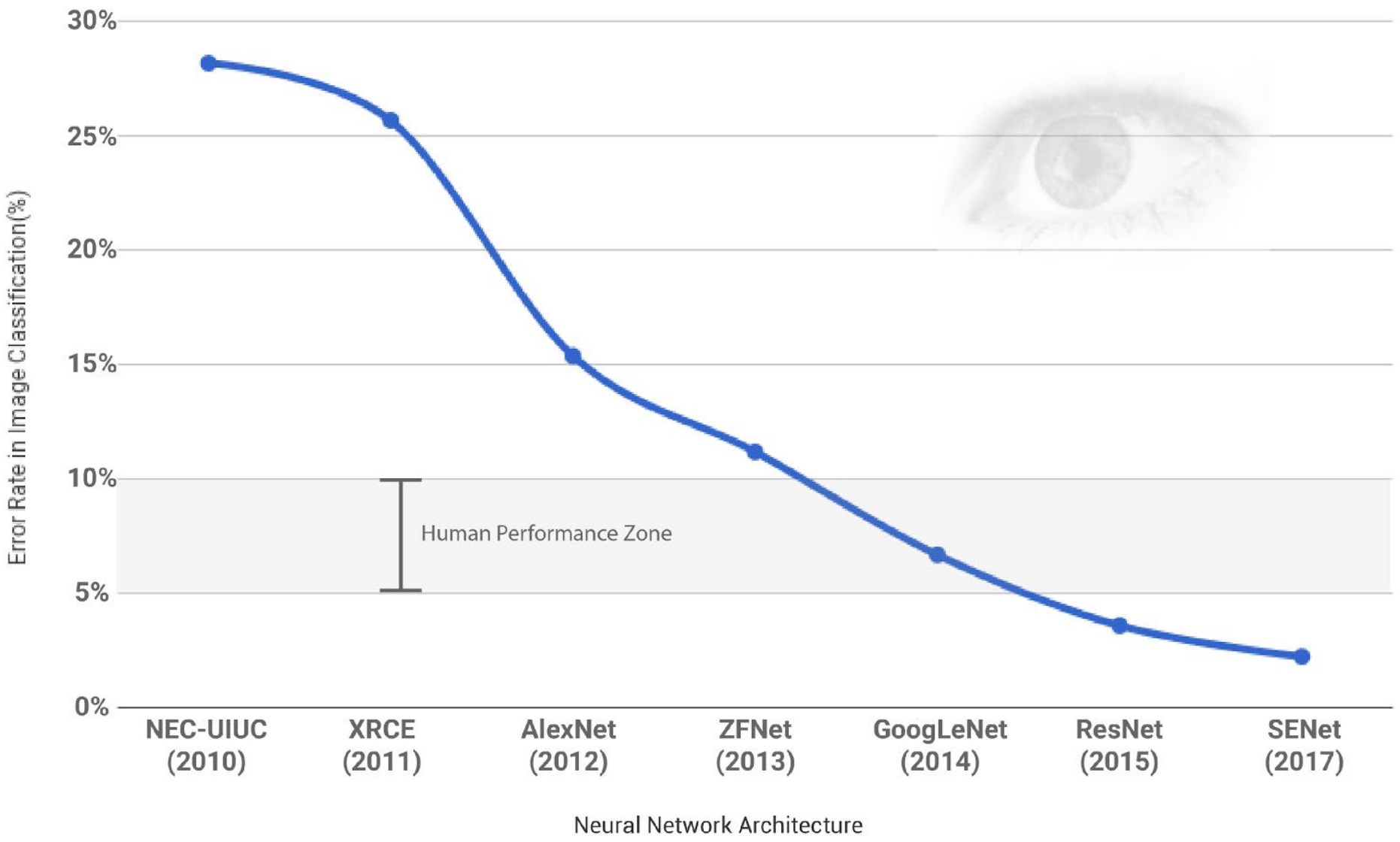
(b) Unraveled view of (a)

ResNet

- **2015 ILSVRC winner （152层）**
  - 错误率：3.57%



ResNet


30%
25%
20%
15%
10%
5%
0%
Error Rate in Image Classification(%)
Human Performance Zone
NEC-UIUC (2010)
XRCE (2011)
AlexNet (2012)
ZFNet (2013)
GoogLeNet (2014)
ResNet (2015)
SENet (2017)
Neural Network Architecture

ResNet

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [40] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test *set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis*

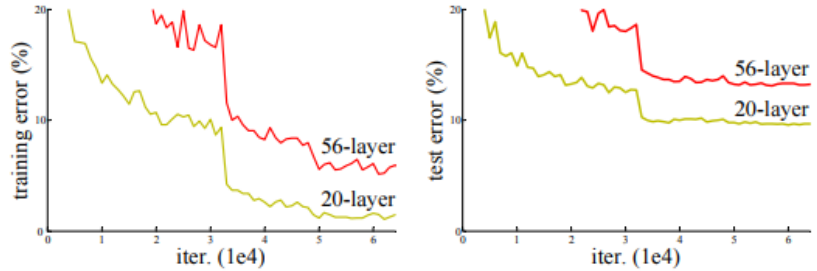


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

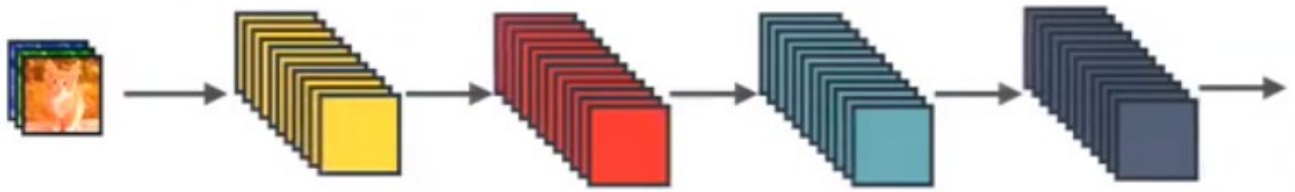
greatly benefited from very deep models.

Driven by the significance of depth, a question arises: *Is learning better networks as easy as stacking more layers?* An obstacle to answering this question was the notorious

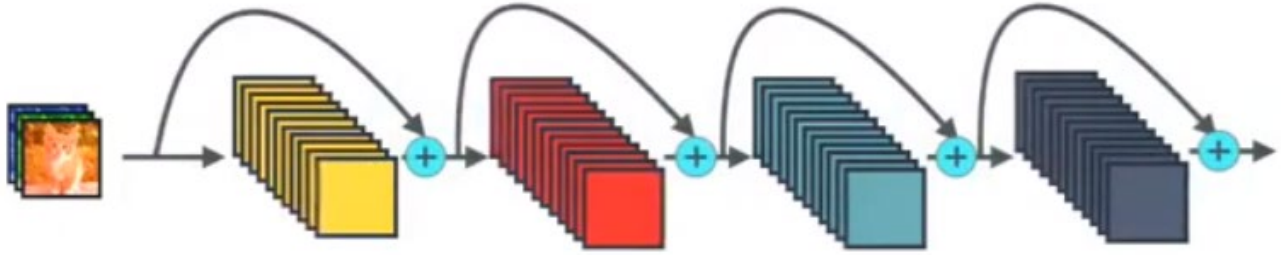
ResNet

DenseNet

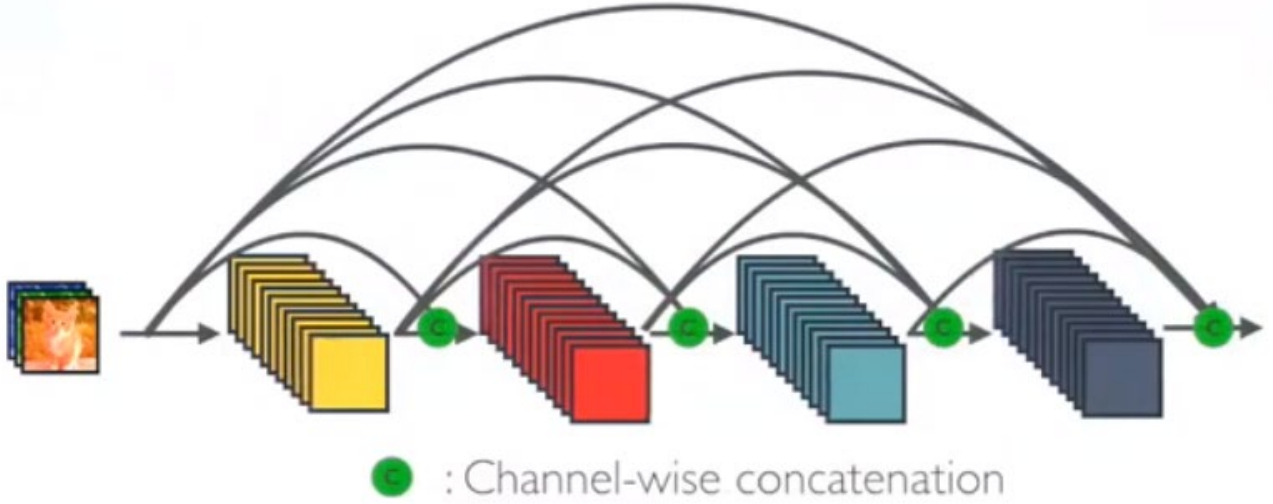
Standard CNN



ResNet

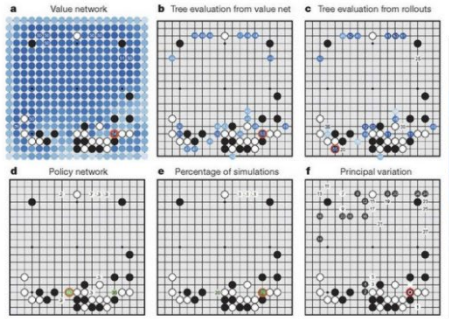


DenseNet



卷积神经网络的应用

AlphaGo

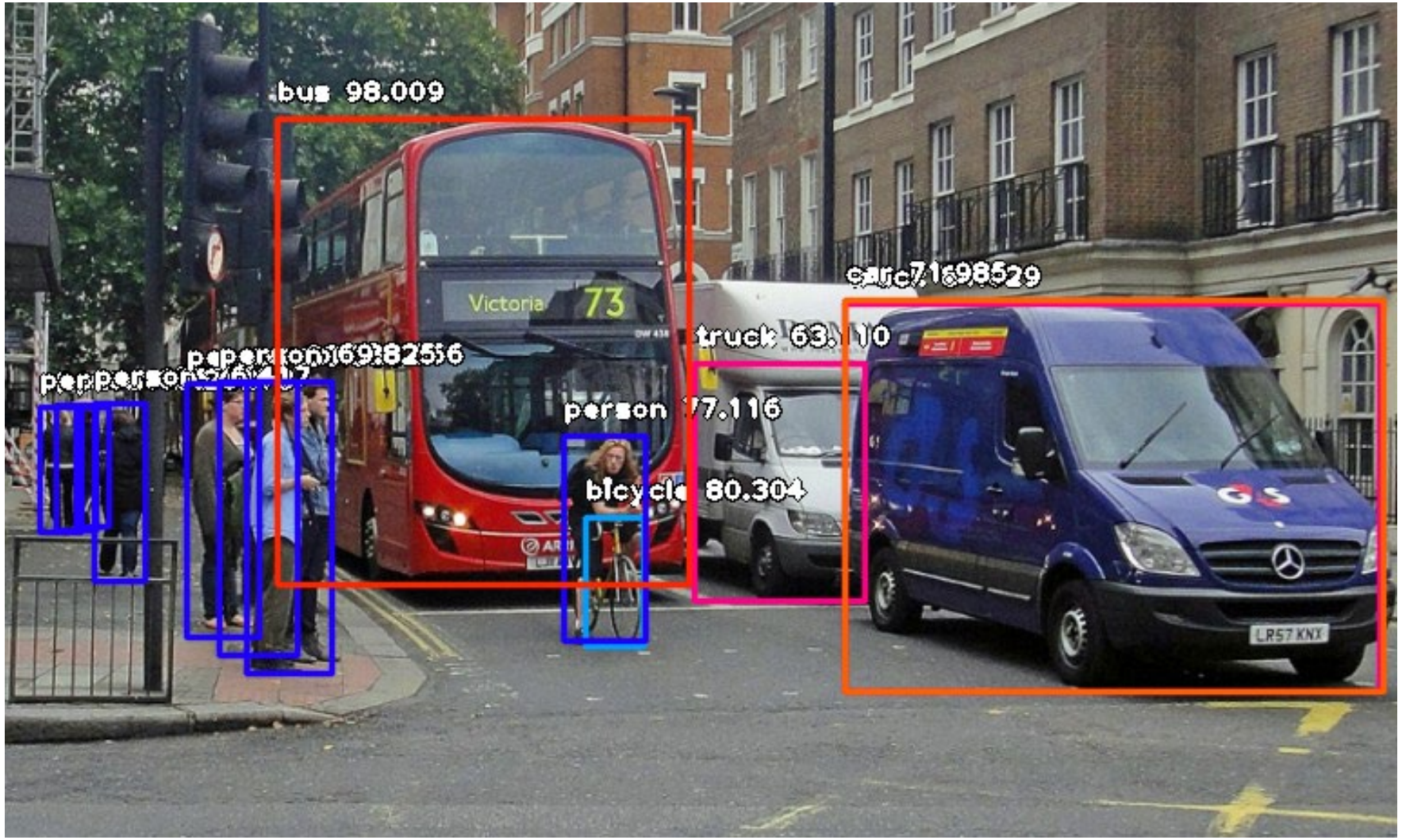


The input to the policy network is a 19 × 19 × 48 image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23 × 23 image, then convolves *k* filters of kernel size 5 × 5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21 × 21 image, then convolves *k* filters of kernel size 3 × 3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1 × 1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used $k = 192$ filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

policy network:
[19x19x48] Input
CONV1: 192 5x5 filters , stride 1, pad 2 => [19x19x192]
CONV2..12: 192 3x3 filters, stride 1, pad 1 => [19x19x192]
CONV: 1 1x1 filter, stride 1, pad 0 => [19x19] *(probability map of promising moves)*

- 分布式系统：1202个CPU和176块GPU
- 单机版：48个CPU和8块GPU
- 走子速度：3 毫秒-2 微秒

目标检测（Object Detection）



Mask RCNN

Figure 4. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).

风格迁移和图像生成

Q&A