



深度学习

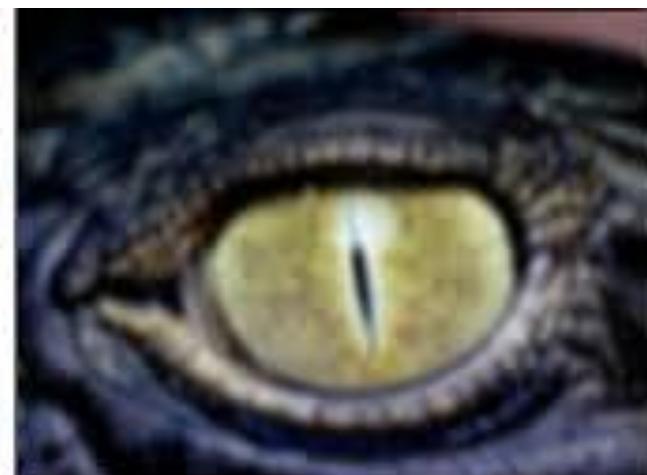
--卷积神经网络



美颜效果



Original image



Gaussian Blur filter applied



本节目标

- ◆ 能够理解并描述卷积神经网络的常见结构；
- ◆ 能够理解并描述卷积、池化、激活等操作的原理和作用；
- ◆ 能够说出常见的典型CNN模型；

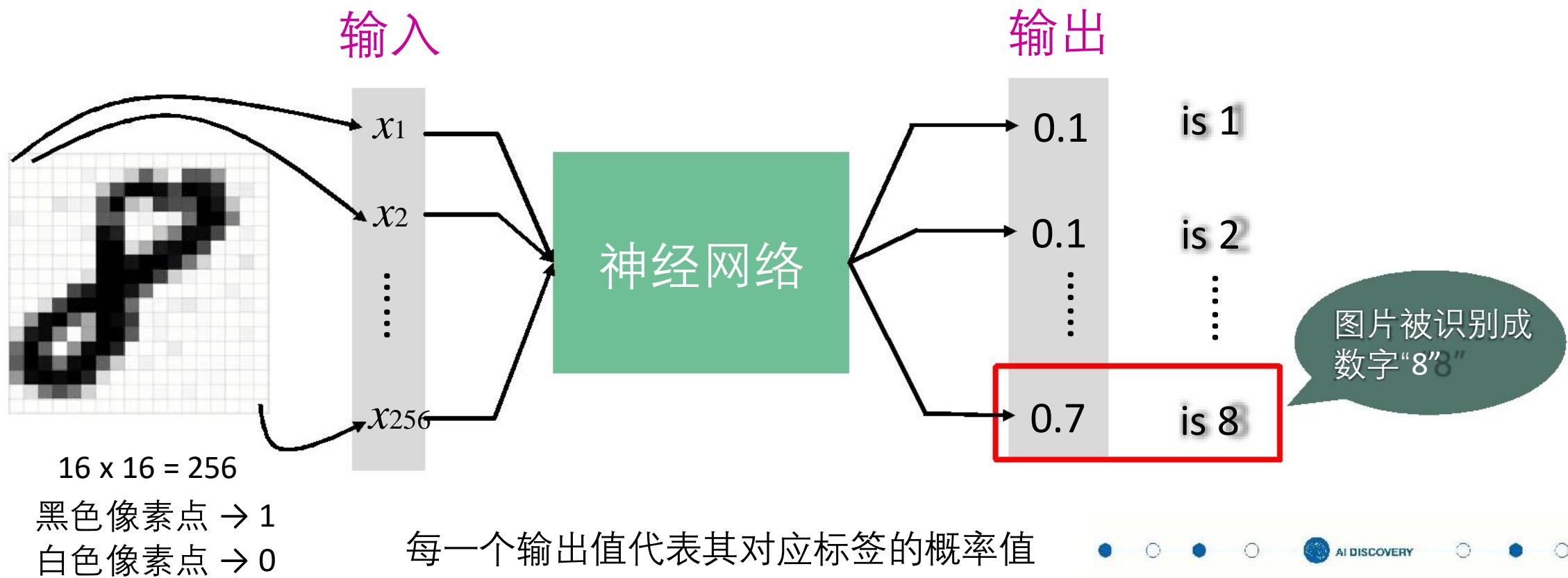
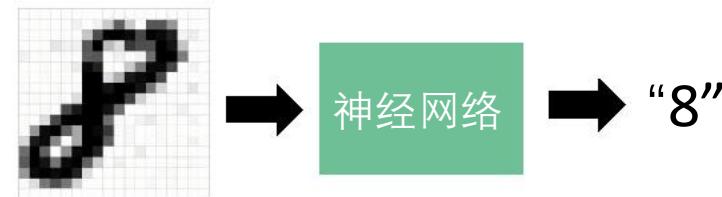
目录

CONTENTS

- 1. 人工神经网络用于分类**
- 2. 卷积网络结构**
 - 卷积层
 - 非线性激活层
 - 池化层
 - 全连接层
- 3. CNN网络的训练**
- 4. 经典的卷积神经网络**

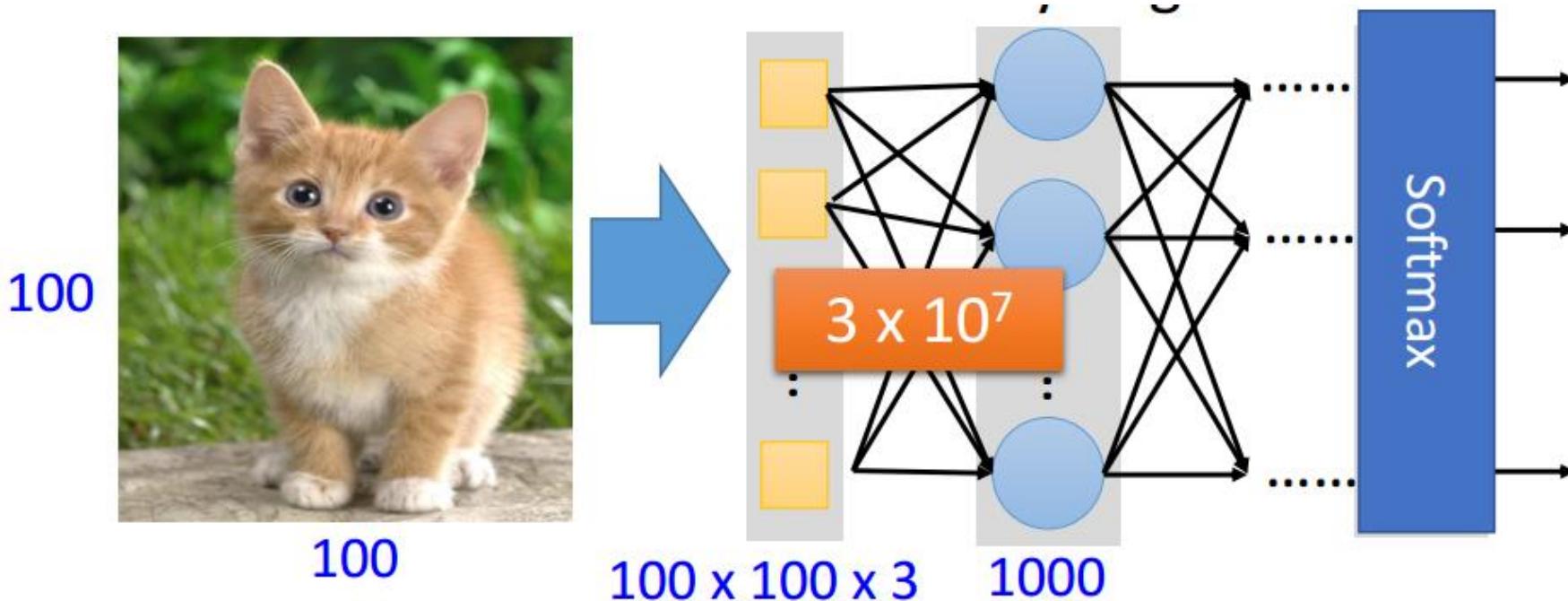


神经网络 进行图像分类



ANN处理图片

- ◆ 在传统神经网络中处理图像时，第一层完全连接层的网络将非常大



这种连接方式是必须的吗？可以简化全连接网络吗？

分类图像中的狗和猫

- ◆ 使用好的相机
- ◆ RGB图像具有 36M 个元素
- ◆ 使用 100 个神经元单隐含层的 MLP 模型：
 - 有 36 亿个参数
 - 超过地球上的狗和猫的数量 (900M 狗 + 600M 猫)



Dual

12MP

wide-angle and
telephoto cameras



单隐含层网络

100个 神经元

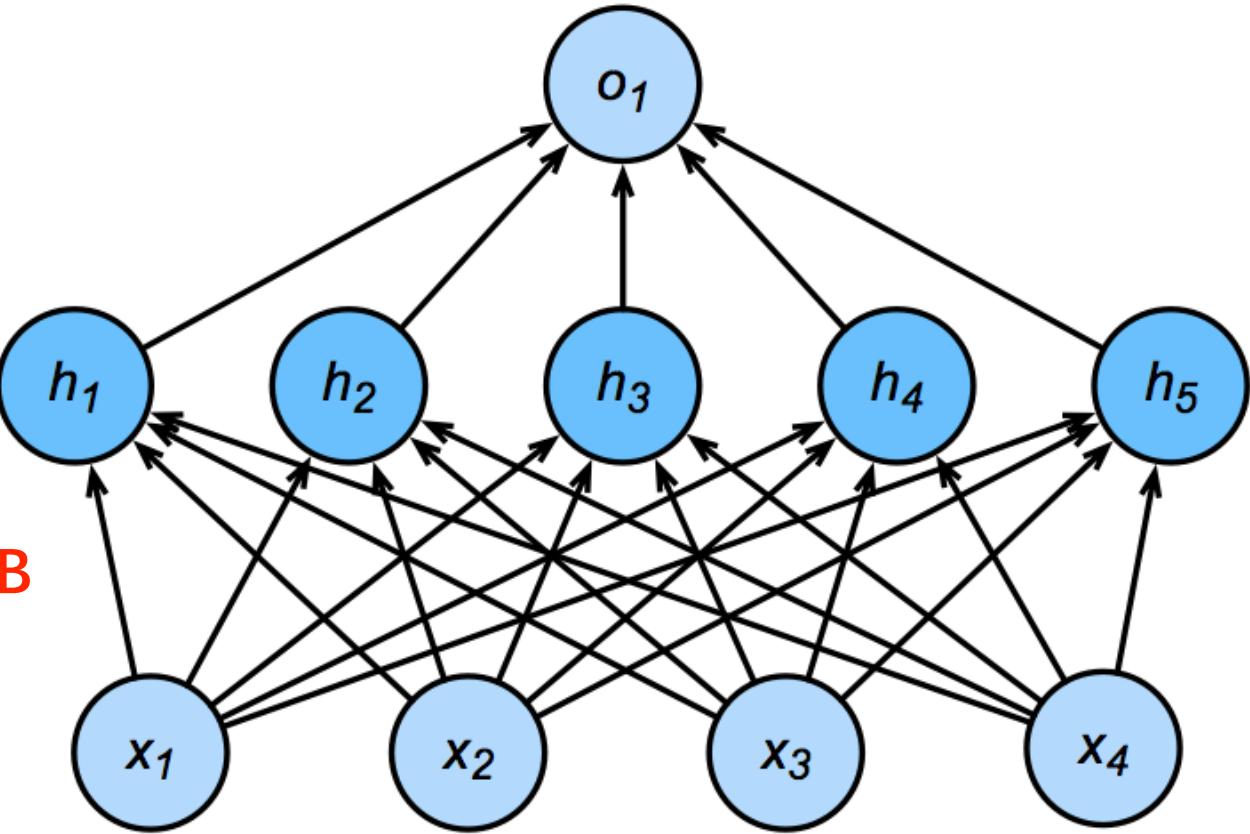
Output layer

36M 特征

Hidden layer

Input layer

3.6B 参数, 每个用4位表示 = 14GB



: 这才是第一个隐层，有什么机器能受得了这种计算？
: 从入门到放弃系列。。

$$\mathbf{h} = \sigma(\mathbf{Wx} + \mathbf{b})$$

图像模式的特征

◆ 图像模式的特性一

➤ 第一个发现：

鸢尾花仅出现在图像局部区域

并不是所有具有相似形态特征的

鸢尾花都位于图像的同一个位置

➤ 如何应用这个发现？

➤ 可能的做法：

1. 定义一种提取局部的特征的方法，可有效响应特定局部模式
 2. 用这种方法遍历整张图片

看一张猫的图片，可能看到猫的眼镜或者嘴巴就知道这是张猫片，



图像模式的特征

◆ 图像模式的特性一

➤ 第一个发现：

鸢尾花仅出现在图像局部区域

并不是所有具有相似形态特征的

鸢尾花都位于图像的同一个位置

➤ 如何应用这个发现？

➤ 可能的做法：

1. 定义一种提取局部的特征的方法，可有效响应特定局部模式
 2. 用这种方法遍历整张图片

: 应用一次该方法只能提取一个特征

: 所以对同一张图片输入，应该应用多次该方法





图像模式的特征

◆ 图像模式的特性二

➤ 第二个发现：

大小改变，鳶尾花仍然可以有效区分

➤ 如何利用这个特性？

➤ 可能的做法：

1. 在神经网络逐层累加的过程中，可以直接对图像进行缩放
 2. 缩放到适当大小后，可以在特征提取过程中得到有效响应



卷积神经网络的诞生

◆ 图像模式的特性——小结

➢ 第一个发现对应的可能的做法：

1. 定义一种提取局部的特征的方法，
可有效响应特定局部模式

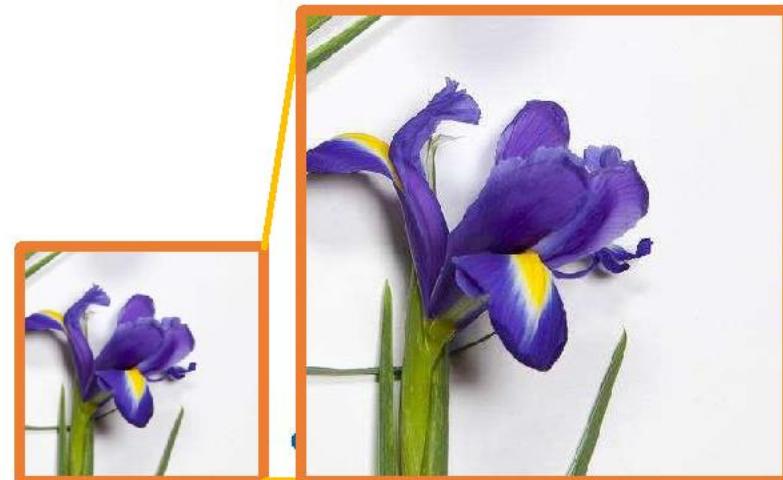
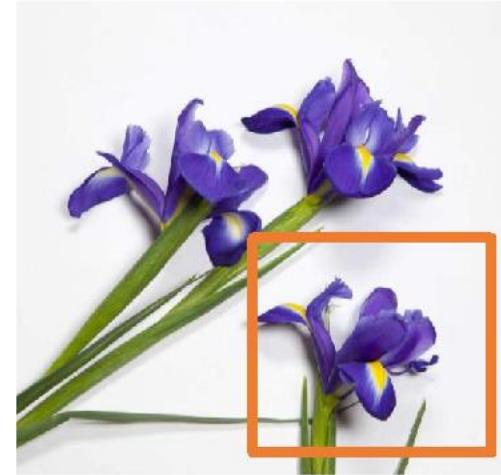
2. 用这种方法遍历整张图片

卷积：平移不变模式

➢ 第二个发现对应的可能的做法：

在神经网络逐层累加的过程中，可以直
接对图像进行缩放

池化：下采样被检测物体不变模式

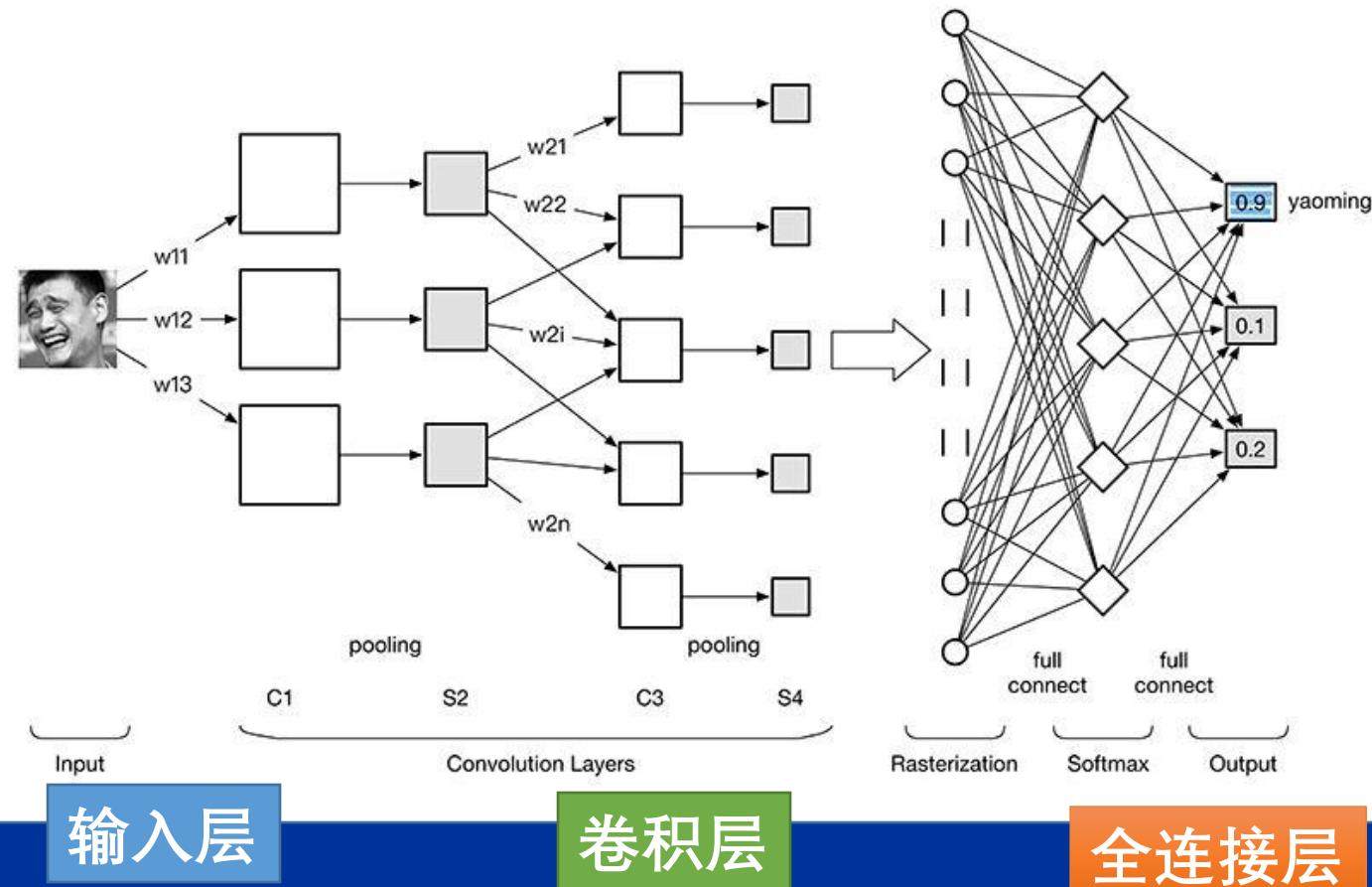


卷积神经网络CNN

◆ 卷积神经网络 (Convolutional Neural Networks, CNN) 是一类包含卷积计算且具有深度结构的前馈神经网络 (Feedforward Neural Networks)，是深度学习的代表算法之一。

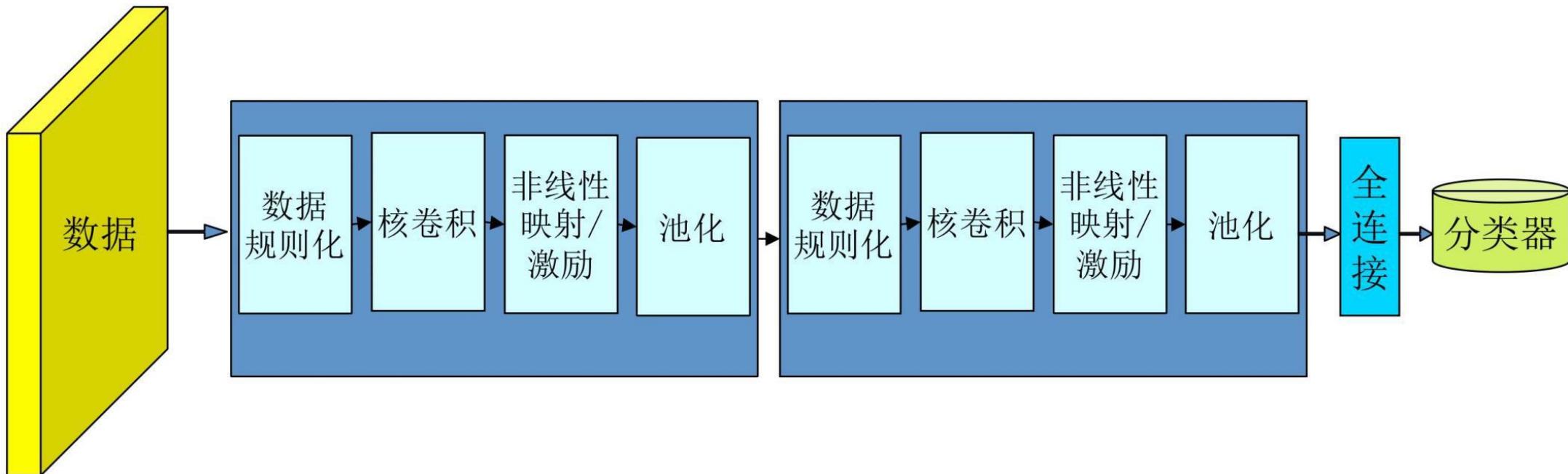
卷积神经网络CNN简介

- ◆ 在卷积神经网络的结构中，包含**卷积层**、**池化层**和**全连接层**三类常见结构。
- ◆ CNN的最后一层将其目标任务（分类、回归等）形式化为目标函数。



卷积神经网络的一般结构

1. 卷积层+ReLU和 池化层 的组合多次出现 提取特征
2. 多个全连接 或 特殊的CNN结构 作为输出层 作分类器/检测器/分割器



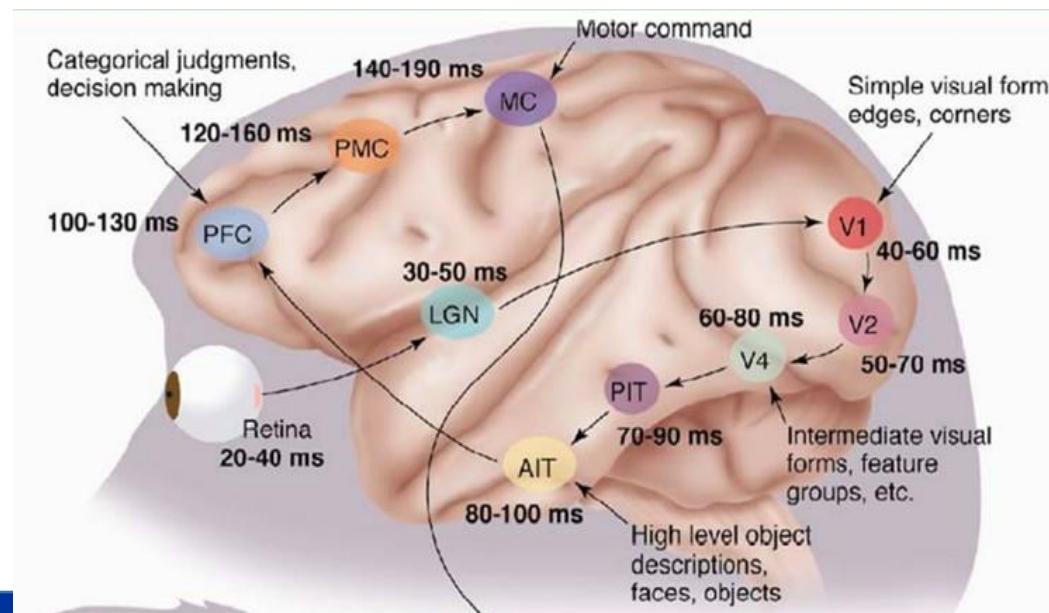


卷积网络的神经科学基础

- ◆ 卷积网络起源于哺乳动物视觉系统的神经科学实验。
- ◆ 神经生理学家 David Hubel 和 Torsten Wiesel 多年研究猫的脑内神经元处理图像的过程。

大脑视觉机理

- ◆ 1981 年的诺贝尔医学奖，颁发给了 David Hubel (出生于加拿大的美国神经生物学家) 和 Torsten Wiesel，以及 Roger Sperry。
- ◆ 前两位的主要贡献，是“发现了视觉系统的信息处理”：可视皮层是分级的，如图(a)所示。



图(a) 人脑视觉机理



- ◆ 1958 年， David Hubel 和 Torsten Wiesel 在 John Hopkins University, 研究瞳孔区域与大脑皮层神经元的对应关系。
- ◆ 他们在猫的后脑头骨上，开了一个3 毫米的小洞，向洞里插入电极，测量神经元的活跃程度。
- ◆ 然后，他们在小猫的眼前，展现各种形状、各种亮度的物体。并且，在展现每一件物体时，还改变物体放置的位置和角度。他们期望通过这个办法，让小猫瞳孔感受不同类型、不同强弱的刺激。



- ◆ 试验目的是去证明一个**猜测**。位于后脑皮层的不同视觉神经元，与瞳孔所受刺激之间，存在某种对应关系。一旦瞳孔受到某一种刺激，后脑皮层的某一部分神经元就会活跃。
- ◆ 最终发现了一种被称为“**方向选择性细胞** (Orientation Selective Cell) ”的神经元细胞。
- ◆ 当小猫瞳孔发现了眼前的物体的边缘，而且这个边缘指向某个方向时，这种神经元细胞就会活跃。



这个发现激发了人们对于神经系统的进一步思考。**神经-中枢-大脑的工作过程，或许是一个不断迭代、不断抽象的过程。**

这里的关键词有两个，一个是**抽象**，一个**迭代**。从原始信号，做低级抽象，逐渐向高级抽象迭代。人类的逻辑思维，经常使用高度抽象的概念。

例如，从原始信号摄入开始（瞳孔摄入像素），接着做初步处理（大脑皮层某些细胞发现边缘和方向），然后抽象（大脑判定，眼前的物体的形状，是圆形的），然后进一步抽象（大脑进一步判定该物体是只气球）。再比如人脸识别，如图(b)示。

分级视觉系统

进一步抽象：大脑进一步判定该物体是什么



抽象：大脑判定，眼前的物体的形状



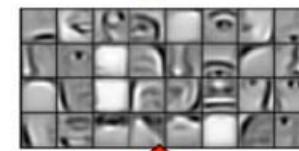
预处理：大脑皮层某些细胞发现边缘和方向



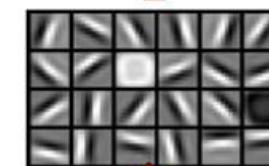
原始信号摄入：瞳孔摄入像素



object models



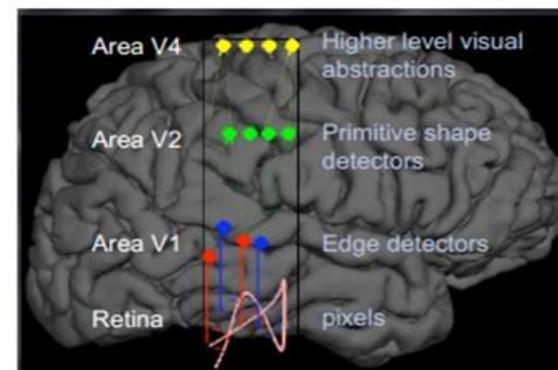
object parts
(combination
of edges)



edges



pixels



图(b) 人脸识别



- ◆这个生理学的发现，促成了计算机人工智能，在四十年后的突破性发展。
- ◆总的来说，人的视觉系统的信息处理是分级的。
 - 从低级的V1区提取边缘特征，再到V2区的形状或者目标的部分等，再到更高层，整个目标、目标的行为等。
 - 也就是说高层的特征是低层特征的组合，从低层到高层的特征表示越来越抽象，越来越能表现语义或者意图。
 - 抽象层面越高，存在的可能猜测就越少，就越利于分类。
 - 例如，单词集合和句子的对应是多对一的，句子和语义的对应又是多对一的，语义和意图的对应还是多对一的，这是个层级体系。



5.4 卷积神经网络

- ◆ 卷积神经网络 (Convolutional Neural Network, CNN) 是一种特殊的多层前馈神经网络，常用于监督学习。
- ◆ 早期的BP神经网络是全连接的，需要学习的参数量巨大，网络训练时间较长。
- ◆ 科学家发现：生物神经元有**感受野** (Receptive Field) 机制，研究人员受此启发，提出了卷积神经网络。
- ◆ 卷积神经网络模仿了感受野的机制，采用卷积运算，使得人工神经元只连接其周围一定范围内的神经元，连接数量减少了，权重（参数）量也减少了。
- ◆ 卷积神经网络最早用来完成图像处理任务，特别是在图像分类、目标检测和语义分割等任务上不断取得突破性进展。



5.4 卷积神经网络

- ◆ 对卷积神经网络的研究最早可以追溯到1979年，日本学者**福岛邦彦**(Kunihiko Fukushima)提出了命名为“neocognitron”的神经网络，它是一个具有深层结构的神经网络，也是最早被提出的深度学习算法之一。
- ◆ 1989年，杨乐昆 (Yann LeCun) 等人在《Backpropagation Applied to Handwritten Zip Code》一文中第一次使用“convolution”和“kernel”术语。提出采用随机梯度下降法训练神经网络模型，并首次使用了“卷积神经网络”一词，因此，杨乐昆被誉为“**卷积神经网络之父**”。

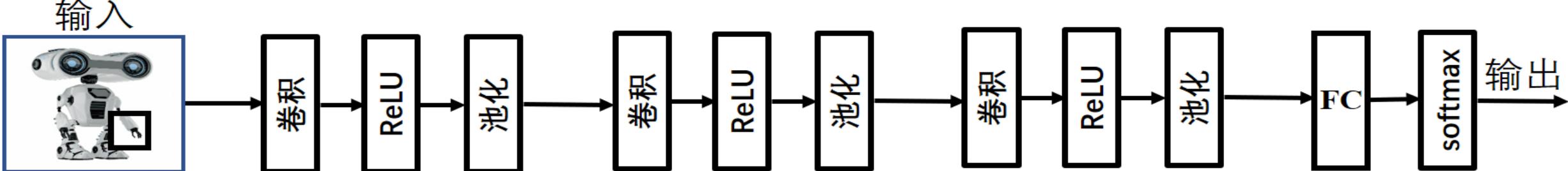
5.4.1 卷积神经网络的整体结构

一个CNN可以包含多个网络层，每层网络由多个独立的神经元组成。

◆ CNN中的网络层主要分为3种类型：**卷积层、池化层和全连接层**。

◆ 通常，每个卷积层与最后一个全连接层之后都会采用激活函数。

◆ 与多层前馈神经网络一样，CNN也可以像搭积木一样，通过叠加多个网络层来组



◆ 多层前馈神经网络可以看成是由“**全连接层+ReLU层**”组合拼装而成的，而CNN可以看成是由“**卷积层+ReLU层+池化层**”（有时也省略池化层）组合拼装而成的。

◆ CNN具有**3个重要特性**：权重共享、局部感知和亚采样。

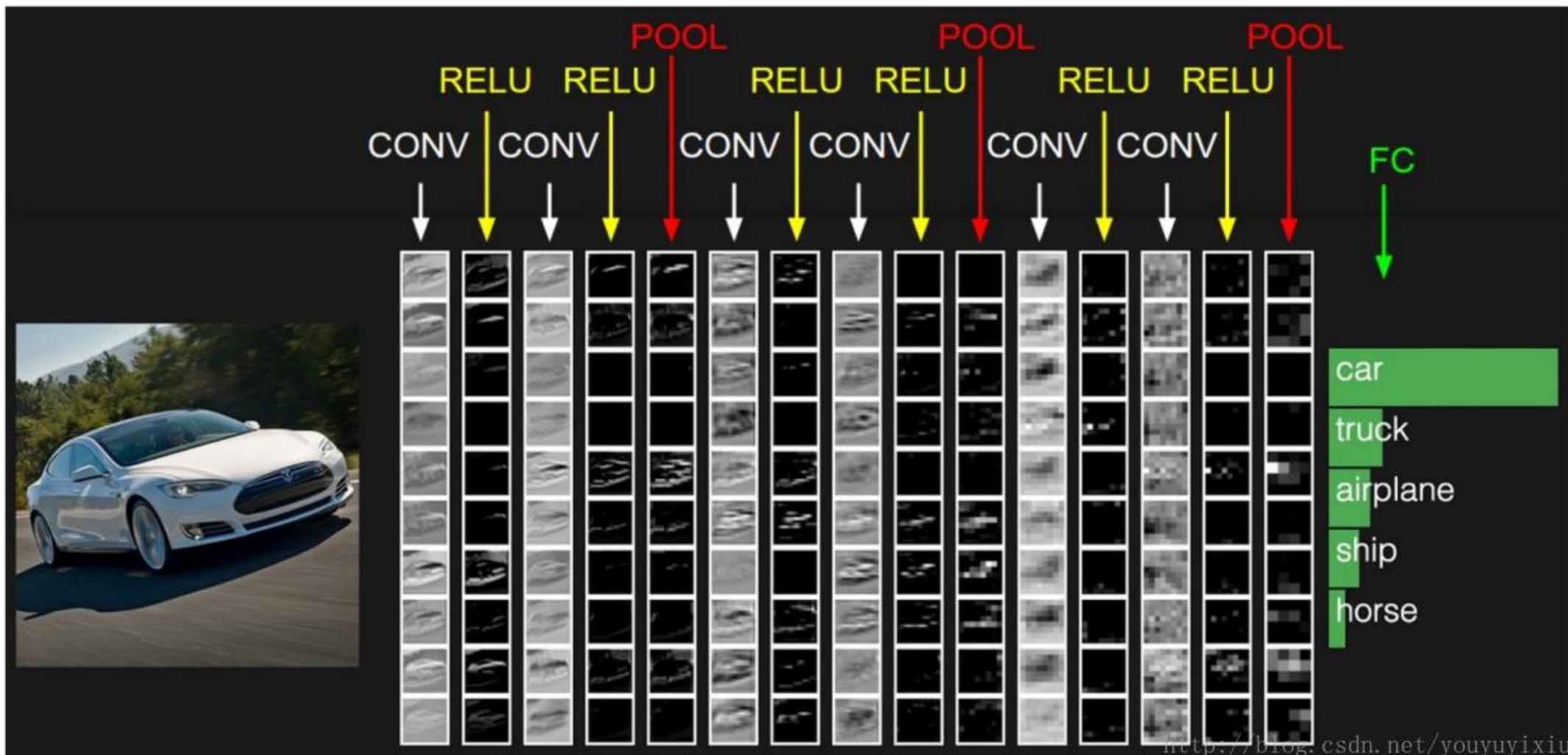
◆ 在卷积神经网络中，输入层输入的数据是训练样本或测试样本，其他网络层输入/输出的数据均称为**特征图**（Feature Map）。

5.4.1 卷积神经网络的整体结构

- ◆ 卷积神经网络不需要额外的特征工程，可以直接从图像中自动提取视觉特征，称为**学习式特征**；不用像传统图像处理技术那样提取**手工式特征**（如HOG、SIFT特征等）。
- ◆ CNN可识别具有极端可变性的模式，如手写体字符，且具有一定程度的**扭曲、平移、旋转、缩放不变性**，从而可以保留图像的空间特性，在图像处理方面具有显著优势。
- ◆ 与**CNN**通融有其他的神经网络一样，**CNN**也采用**反向传播算法训练模型**。
 - 1个输入层
 - 多个“卷积层+ReLU层+池化层”组合块，也可能是【n个“卷积层+ReLU层”+ 1个池化层】的组合
 - 多个连续的全连接层（中间不加激活函数）
 - 1个采用softmax函数的输出层。

卷积神经网络的整体结构

1. (卷积层+ReLU+ 池化层) 的组合多次出现：用于提取特征
2. 多个全连接 或 特殊的CNN结构作为输出层：用作分类器/检测器/分割器



5.4.1 卷积神经网络的整体结构

(1) 输入层。

- 灰度图： $W \times H$ 矩阵， 其中 W 为宽度， H 为高度， 图像深度为 1；
- 彩色图： $W \times H \times 3$ 像素矩阵，“3”表示 R、G、B 三个颜色通道，即图像深度。

(2) 卷积层。

- 卷积操作就是点积运算。
- **卷积层的功能**：用卷积操作对输入的原始数据/特征图**提取特征**，输出卷积运算后产生的特征图。
- **卷积层是CNN的核心部分**，一个卷积层可以包含若干个卷积核（Kernel），一个卷积核就是一个二维的、高度和宽度相同的权重矩阵，记为 $\text{Conv } F \times F$, $F << W$ 。
- $F \times F$ 的区域就是卷积核的**感受野**，卷积核与输入的图像或特征图只有 $F \times F$ 大小的局部连接；
- 在全连接神经网络中，隐藏层的神经元的感受视野是输入图像或特征图的全部大小，进行的是全局连接。

11.2.2 卷积操作

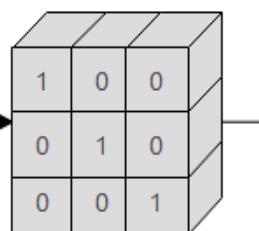
- ◆ **卷积(Convolution)**, 是信号处理与数字图像处理领域中的常用方法。通过对图像进行卷积处理，能够实现图像的基本模糊、锐化、降低噪声、提取边缘特征等功能。
- ◆ 基本概念：
- ◆ 卷积核
- ◆ 填充
- ◆ 步长

卷积的本质



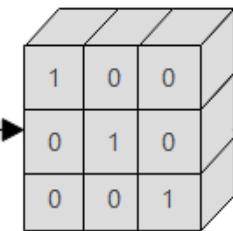
卷积操作的分步骤演示

0	0	0	0	0	0	0
0	4	3	2	1	0	
0	2	3	9	8	0	
0	1	2	3	4	0	
0	3	4	0	0	0	
0	0	0	0	0	0	



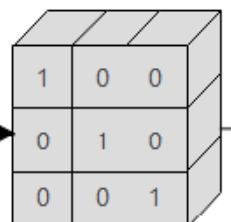
第1步

0	0	0	0	0	0	0
0	4	3	2	1	0	
0	2	3	9	8	0	
0	1	2	3	4	0	
0	3	4	0	0	0	
0	0	0	0	0	0	



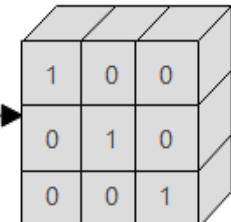
第2步

0	0	0	0	0	0	0
0	4	3	2	1	0	
0	2	3	9	8	0	
0	1	2	3	4	0	
0	3	4	0	0	0	
0	0	0	0	0	0	



第3步

0	0	0	0	0	0	0
0	4	3	2	1	0	
0	2	3	9	8	0	
0	1	2	3	4	0	
0	3	4	0	0	0	
0	0	0	0	0	0	



经过16步卷积运算后，最终完成状态

灰度图上单个卷积核的卷积运算示例

在下列灰度图像上，假设给定 4×4 的像素矩阵，卷积核大小为 2×2 ，滑动步长为2。

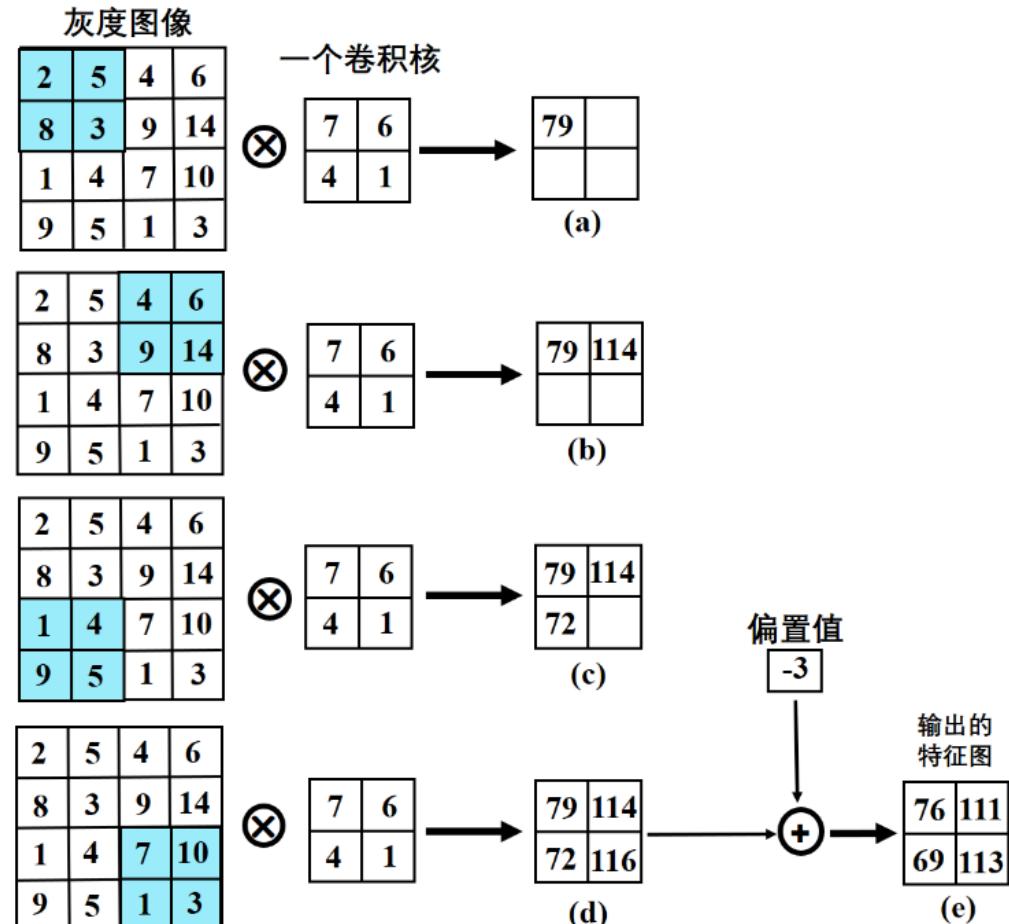
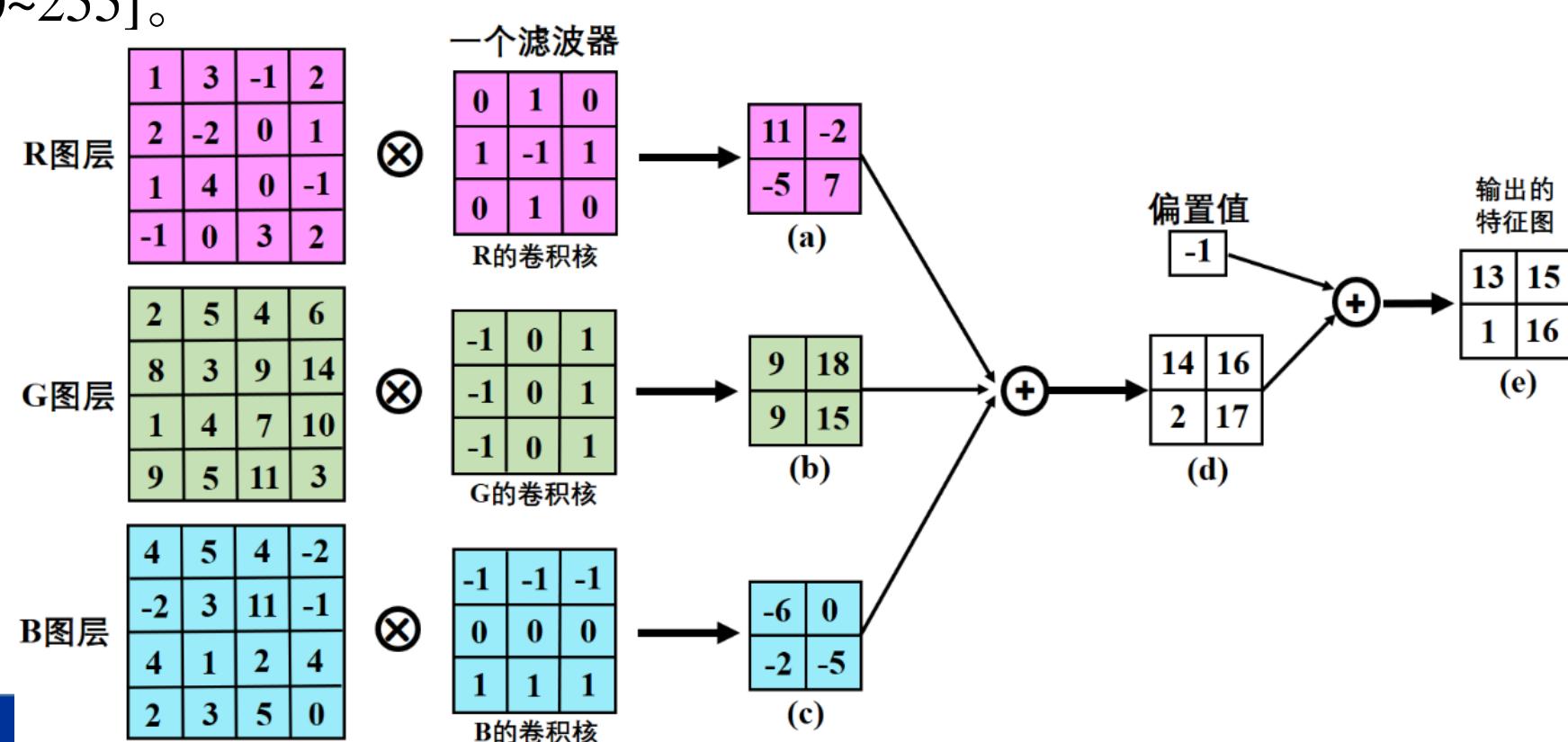


图5.12 灰度图像的卷积运算示例

5.4.2 卷积运算

2. RGB图像上的卷积运算

- 输入的彩色图像通常表示为 $W \times H \times 3$ 的像素矩阵，其中“深度”也称为通道。
- 彩色图像可以看作3个二维像素矩阵，每个像素矩阵存放所有像素的一种颜色值，像素取值范围为[0~255]。



RGB图上2个滤波器的卷积运算过程

R

Input Volume (+pad 1) (7x7x3)							Filter W0 (3x3x3)		
$x[:, :, 0]$							$w0[:, :, 0]$		
0	0	0	0	0	0	0	1	1	-1
0	0	1	1	2	2	0	-1	0	1
0	0	1	1	0	0	0	-1	-1	0
0	1	1	0	1	0	0	w0[:, :, 1]		
0	1	0	1	1	1	0	-1	0	-1
0	0	2	0	1	0	0	0	0	-1
0	0	0	0	0	0	0	1	-1	0

G

Input Volume (+pad 1) (7x7x3)							Filter W0 (3x3x3)		
$x[:, :, 1]$							$w0[:, :, 1]$		
0	0	0	0	0	0	0	0	1	0
0	1	1	1	2	0	0	1	0	1
0	0	2	1	1	2	0	0	-1	1
0	1	2	0	0	2	0	w0[:, :, 2]		
0	0	2	1	2	1	0	-1	0	1
0	2	0	1	2	0	0	1	0	1
0	0	0	0	0	0	0	0	-1	0

B

Input Volume (+pad 1) (7x7x3)							Filter W0 (3x3x3)			
$x[:, :, 2]$							$w0[:, :, 2]$			
0	0	0	0	0	0	0	0	0	0	
0	2	0	2	0	2	0	2	0	2	
0	0	0	1	2	1	0	1	2	1	
0	1	0	2	2	2	1	0	w0[:, :, 0]		
0	2	0	2	0	0	0	0	-1	1	
0	0	0	1	1	2	0	1	1	2	
0	0	0	0	0	0	0	0	-1	0	

第1个滤波器，即第1个神经元

Filter W0 (3x3x3)			Output Volume (3x3x2)		
$w0[:, :, 0]$			$o[:, :, 0]$		
-1	1	0	1	0	-3
-1	0	1	-6	1	1
-1	-1	0	4	-3	1
$w0[:, :, 1]$			$o[:, :, 1]$		
1	-1	0	-1	-6	-4
-1	0	-1	-2	-3	-4
-1	0	0	-1	-3	-3
$w0[:, :, 2]$			$o[:, :, 2]$		
-1	0	1	1	0	1
1	0	1	0	-1	0

第1个滤波器的输出

第2个滤波器的输出

滤波器的偏差

第2个滤波器，即第2个神经元

- ◆ 粉色矩阵是两个滤波器/神经元: ($w0, w1$)
- ◆ 滤波器的步长设为 2

卷积层的简单应用

◆卷积层的简单应用：检测图像中物体的边缘，
位置。

一个高和宽分别为1和2的卷积核K

输入：
6 x 8的图像,它中间4列为黑 (0) ,
其余为白 (1) 。

```
tensor([[1., 1., 0., 0., 0., 0., 0., 1.],  
       [1., 1., 0., 0., 0., 0., 1., 1.],  
       [1., 1., 0., 0., 0., 0., 1., 1.],  
       [1., 1., 0., 0., 0., 0., 1., 1.],  
       [1., 1., 0., 0., 0., 0., 1., 1.],  
       [1., 1., 0., 0., 0., 0., 1., 1.]])
```

```
tensor([[1, -1]])
```



即找到像素变化的

将输入X和卷积核K做互相关运算。
可以看出，将从白到黑的边缘和从
黑到白的边缘分别检测成了1和-1。
其余部分的输出全是0。

$Y = \text{corr2d}(X, K)$

```
tensor([[ 0.,  1.,  0.,  0.,  0., -1.,  0.],  
       [ 0.,  1.,  0.,  0.,  0., -1.,  0.],  
       [ 0.,  1.,  0.,  0.,  0., -1.,  0.],  
       [ 0.,  1.,  0.,  0.,  0., -1.,  0.],  
       [ 0.,  1.,  0.,  0.,  0., -1.,  0.],  
       [ 0.,  1.,  0.,  0.,  0., -1.,  0.]])
```

【例】对图像进行卷积处理



(a) 原图



(b) 边缘锐化



(c) 变暗

图像卷积运行结果

例子



(原始图片)

卷积核

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

卷积之后的特征图



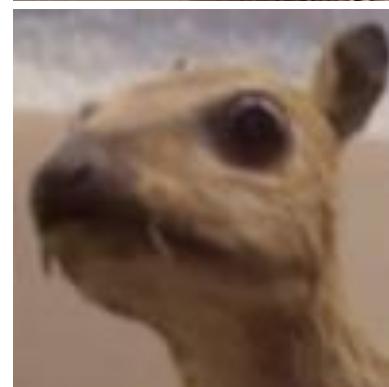
边缘检测

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



锐化



高斯模糊

卷积核可视化: <https://setosa.io/ev/image-kernels/>

5.4.2 卷积运算

3. 图像填充 (padding)

- 每做一次卷积运算，输出的特征图的尺寸都会减少若干个像素，经过若干个卷积层后，特征图尺寸会变得非常小。
- 另外，在卷积核移动的过程中，图像边缘的像素参与卷积运算的次数远少于图像内部的像素，这是因为边缘上的像素永远不会位于卷积核的中心，而卷积核也不能扩展到图像边缘区域以外，因此会导致图像边缘的大部分信息丢失。
- 若想尽可能多地保留原始输入图像的信息，可以在卷积操作之前，在原图像的周围填充p圈固定的常数，例如，填充常数0，这种操作称为**填充** (padding)。

在原特征图的四周填补一圈0值

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



卷积核

0	1	0
1	-1	1
0	1	0



输出保持原来尺寸的特征图

4	-5	6	-2
-2	11	-2	0
4	-5	6	4
2	6	-1	0



5.4.2 卷积运算

- 填充的主要目的是调整输入数据的大小，使得输出数据的形状保持与输入数据一致。需要根据具体情况来确定超参数 p 的值。
- 在实践中，当设置 $\text{padding} = \text{valid}$ 时，表示不填充0值，当 $\text{padding} = \text{same}$ 时，表示自动计算 p 值来填补0值，使得卷积运算前、后的特征图的尺寸相同。
- 采用填充技术有以下**两个作用**：
 - ①CNN的深度不再受卷积核大小的限制，CNN可以不断地堆叠卷积层。若不作填充，当前一层输出的特征图的尺寸比卷积核还小时，就无法再进行卷积操作，也就无法再增加卷积层了。
 - ②可以充分利用图像的边界信息，避免遗漏图像边界附近的重要信息。

5.4.2 卷积运算

4. 卷积运算后特征图尺寸的计算公式

- 填充的p值和步长s的值都会影响卷积输出特征图的大小。
- 设当前卷积层中滤波器的个数为 K , 输入特征图的尺寸为 $H \times W \times D$ (D 为通道数) , 卷积核的尺寸为 $F \times F$, 填充为 p , 步长为 s , 则执行卷积运算后, 输出特征图的尺寸 $H' \times W' \times D'$ 的计算公式为:

$$H' = \frac{H+2p-F}{s} + 1, \quad W' = \frac{W+2p-F}{s} + 1, \quad D'=K$$

其中, 主要超参数包括: 每个卷积层中滤波器的个数 K 、卷积核或滤波器的大小 F 、步长 s 、填充 p 。当前卷积层中学习参数的个数为 $(F \times F \times D + 1) \times K$ 。

卷积运算示例

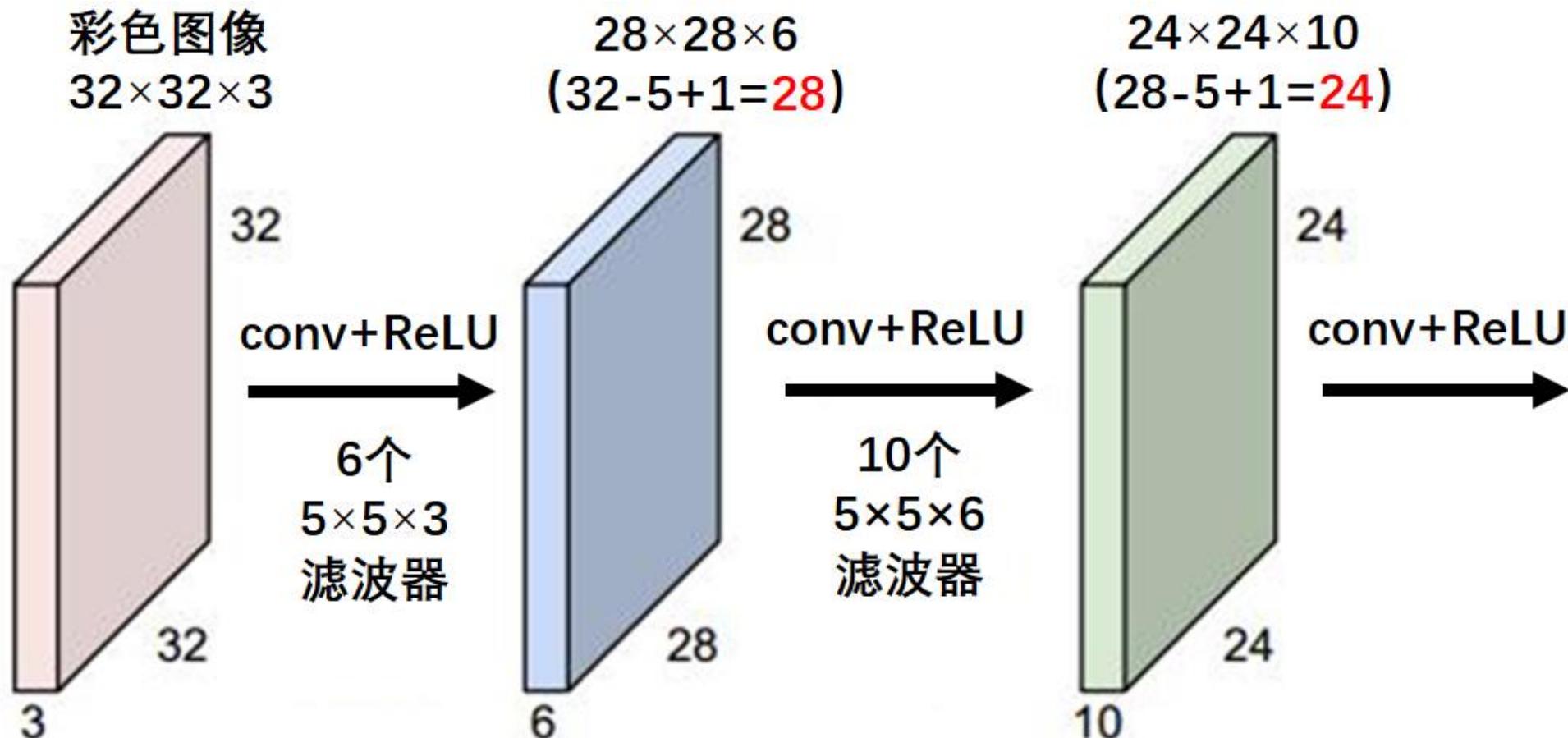


图5.15 两个卷积层的运算过程

Padding 填充

❖ 需要注意的参数：padding

➤ Padding = valid

- ✓ 不进行补零操作， $s=1$ 时，每卷积一次，宽和高方向的数据维度下降 $F-1$ ，其中 F 为卷积核大小

10	10	10	0	0
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0

padding: valid

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

0	0	0	0	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	0	0	0	0	0	0

padding: same

-20	0	20	20	0
-30	0	30	30	0
-30	0	30	30	0
-30	0	30	30	0
-20	0	20	20	0

- ✓ 在输入的周围进行0或复制填充
- ✓ 卷积前width=卷积后width，卷积前height=卷积后height
- ✓ $F=3$, stride=1, pad=1



步幅

- ◆ 卷积窗口从输入数组的最左上方开始，按从左往右、从上往下的顺序，依次在输入数组上滑动。我们将每次滑动的行数和列数称为步幅 (stride)。
- ◆ 一般来说，当高上步幅为 s_h ，宽上步幅为 s_w 时，输出形状为

$$\lfloor (n_h - k_h + p_h + s_h) / s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w) / s_w \rfloor.$$

- ◆ 在默认情况下，填充为0，步幅为1。

Stride 步长

❖ 需要注意的参数： stride

➤ Stride

- ✓ 一次滑动的步长
- ✓ 有height上的stride和width上的stride
- ✓ 图片中的stride = 2, 指在两个维度上的 stride都为2

10	10	10	0	0
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0

Stride =1: 一次滑动1格

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

0	30	30
0	30	30
0	30	30

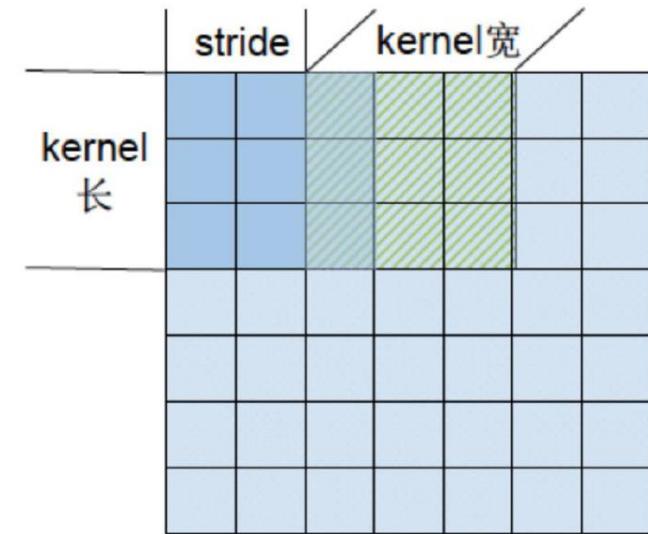
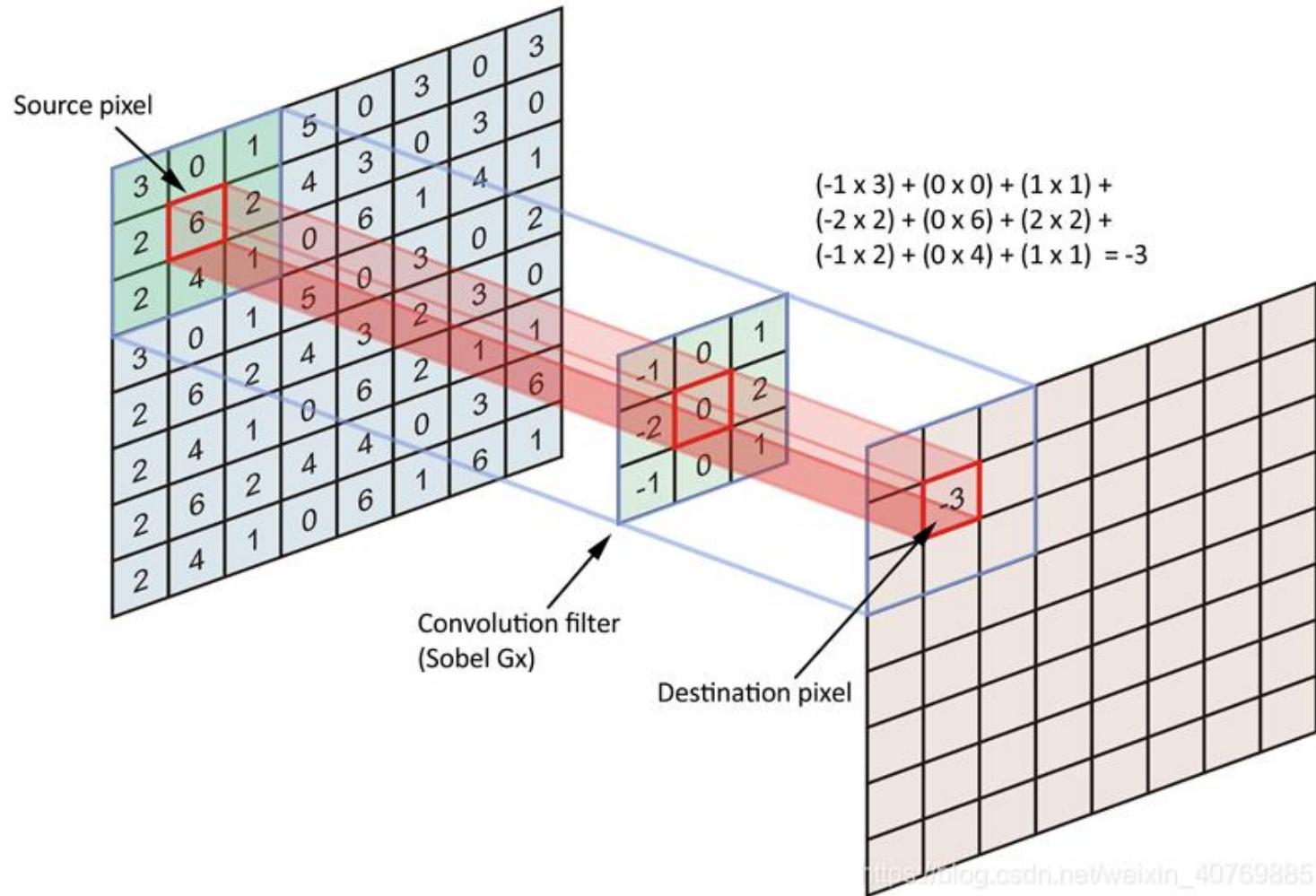
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0

Stride =2: 一次滑动2格

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

0	30
0	30

二维卷积



https://blog.osdn.net/weixin_40769885

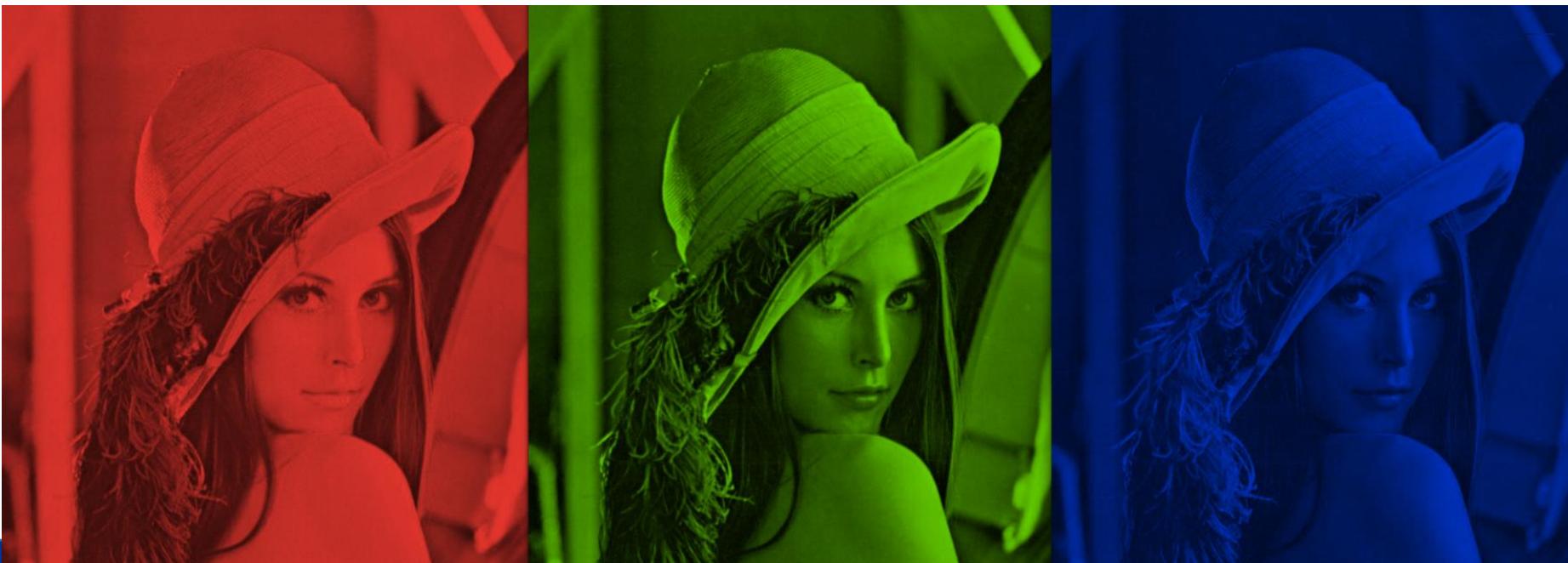


通过数据学习核数组

- ◆ 使用物体边缘检测中的输入数据 X 和输出数据 Y 来学习我们构造的核数组 K 。
- ◆ 我们首先构造一个卷积层，其卷积核将被初始化成随机数组。接下来在每一次迭代中，我们使用平方误差来比较 Y 和卷积层的输出，然后计算梯度来更新权重。

多个输入通道

- ◆ 彩色图像可能有 RGB 三个通道
- ◆ 转换为灰度会丢失信息



卷积层（多输入通道）

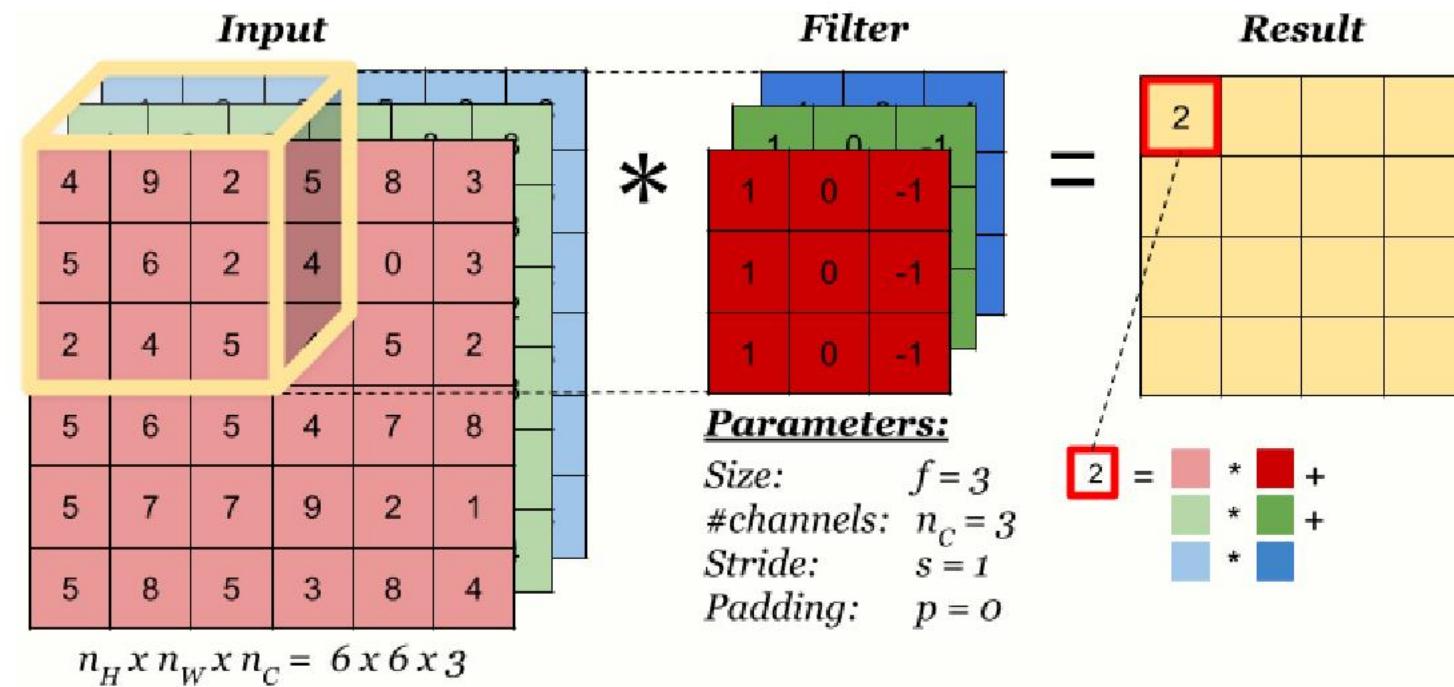
RGB图像上使用单卷积核：单个特征的抽取

➤ Notes 1

- ✓ RGB图像channel = 3
 - ✓ 对RGB图像进行操作的卷积核的深度为3
 - ✓ 右图以 $3 * 3$ 的卷积核为例

➤ Notes 1 update

- ✓ 卷积核有第三个维度： Depth
 - ✓ 卷积核的深度 = 上一层数据输入的深度 (channel数)



5.4.2 卷积运算

6.卷积神经网络结构的特点

(1) 局部连接。

- 人类对外界的认知一般是从局部到全局，先感知局部，再逐步认知全局。卷积即局部感受野。
- CNN模仿了人类的认识模式：一个神经元只与特征图局部区域中的元素相连。
- 每个卷积层的输入特征图或卷积核的大小是不同的，卷积核大小的不同意味着感受野范围的不同。
- 随着网络的加深，神经元的感受野范围逐层扩大，所提取图像特征的全局化程度越来越高，直到全连接层，全连接层中每个神经元的感受野覆盖了前一网络层的全部输出，得到的就是全局特征。



5.4.2 卷积运算

CNN是：

- 在卷积层，先用感受野小的卷积提取图像的局部特征；
- 在全连接层，再用全连接将所有特征重新组合在一起，提取全局特征。
- 局部连接实际上减少了神经元之间的连接数，也就减少了参数量，起到了降低计算量的作用。

5.4.2 卷积运算

(2) 权值共享。

- 权值共享是指卷积核在滑过整个图像时，其参数是固定不变的。
- 计算同一个通道的特征图的不同窗口时，卷积核中的权值是共享的，这样可以极大地减少参数量。
- 需要指出的是：同一卷积核只是针对同一通道的特征图共享权值，不同滤波器在同一通道上的卷积核不共享权值，不同通道上的卷积核也不共享权值。
- 一个滤波器的多个卷积核共享同一个偏置值。
- 在CNN的隐藏层中，**共享卷积核的参数可以减少学习参数的数量**，降低处理高维数据的计算压力。例如：AlexNet 的参数有1亿，采用共享权值后，减至 6000万。

可见，卷积神经网络的两个特点（局部连接和权值共享），都是减少学习参数量的方法。



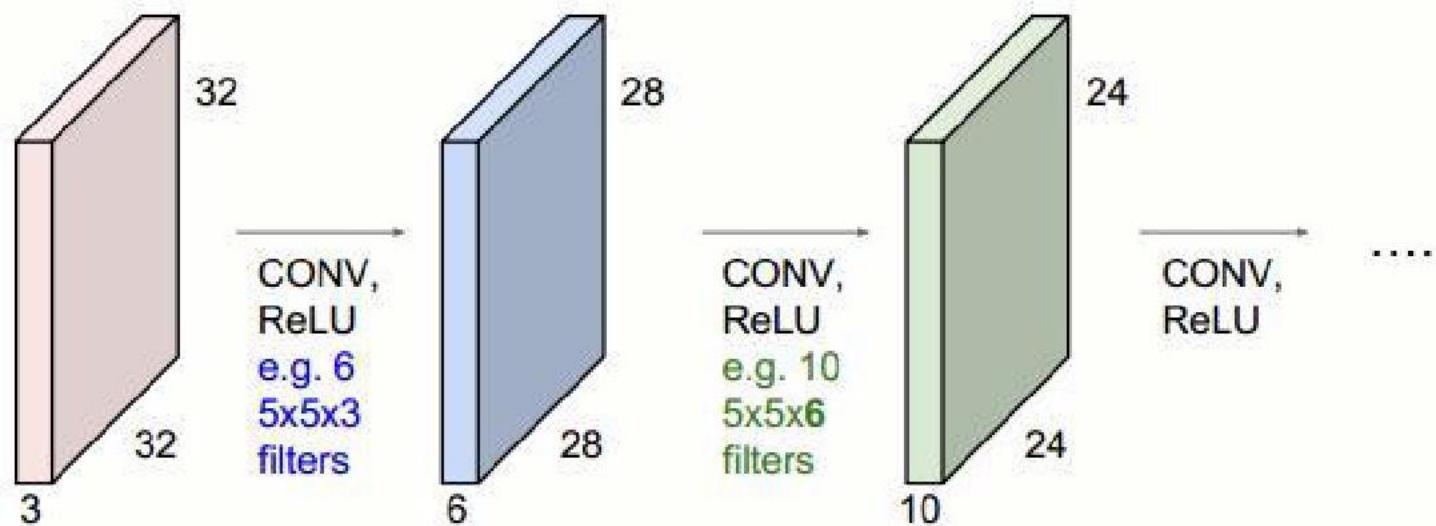
卷积层

卷积隐层的堆叠

➤ Notes 2

- ✓ 卷积核的个数 = 下一层数据的深度 = 下一卷积层 卷积核的深度
 - ✓ 卷积核的个数 = 提取特征的数量， 超参数， 可以自己调节

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



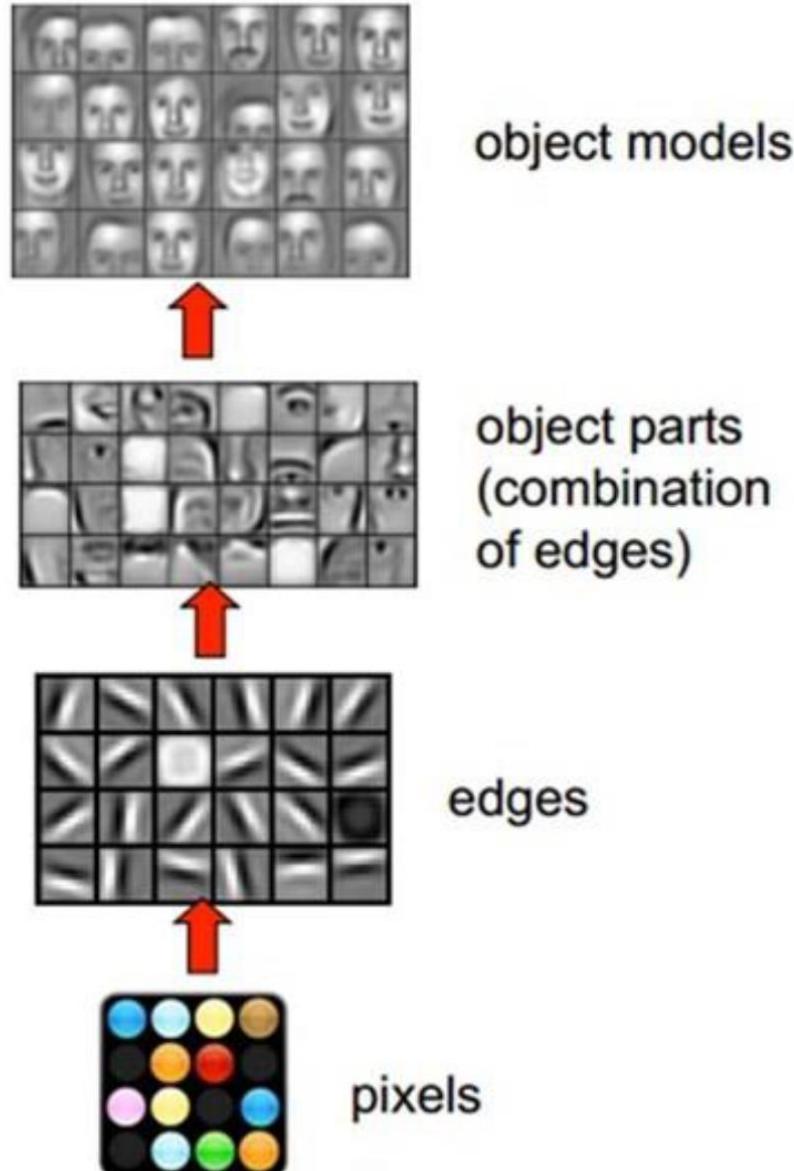


卷积层

◆ 隐层的卷积：特征组合

✓ 多层卷积：

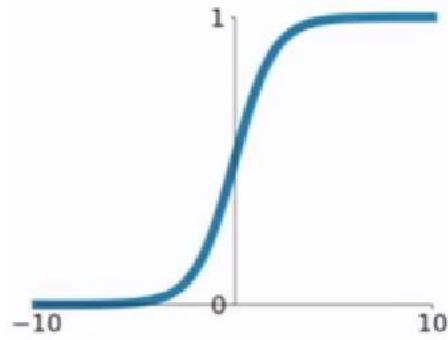
一层卷积得到的特征往往是局部的
层数越高，学到的特征就越全局化



激活函数

往模型中加入非线性元素，能表示更大范围的函数

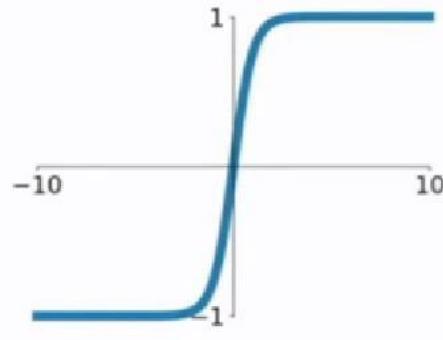
一般不在同一个网络中使用多种激活函数



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

指数运算，效率低

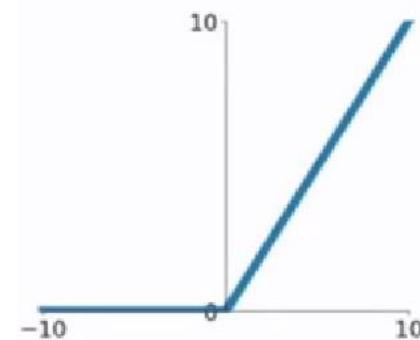
Sigmoid



$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

指数运算，效率低

Tanh

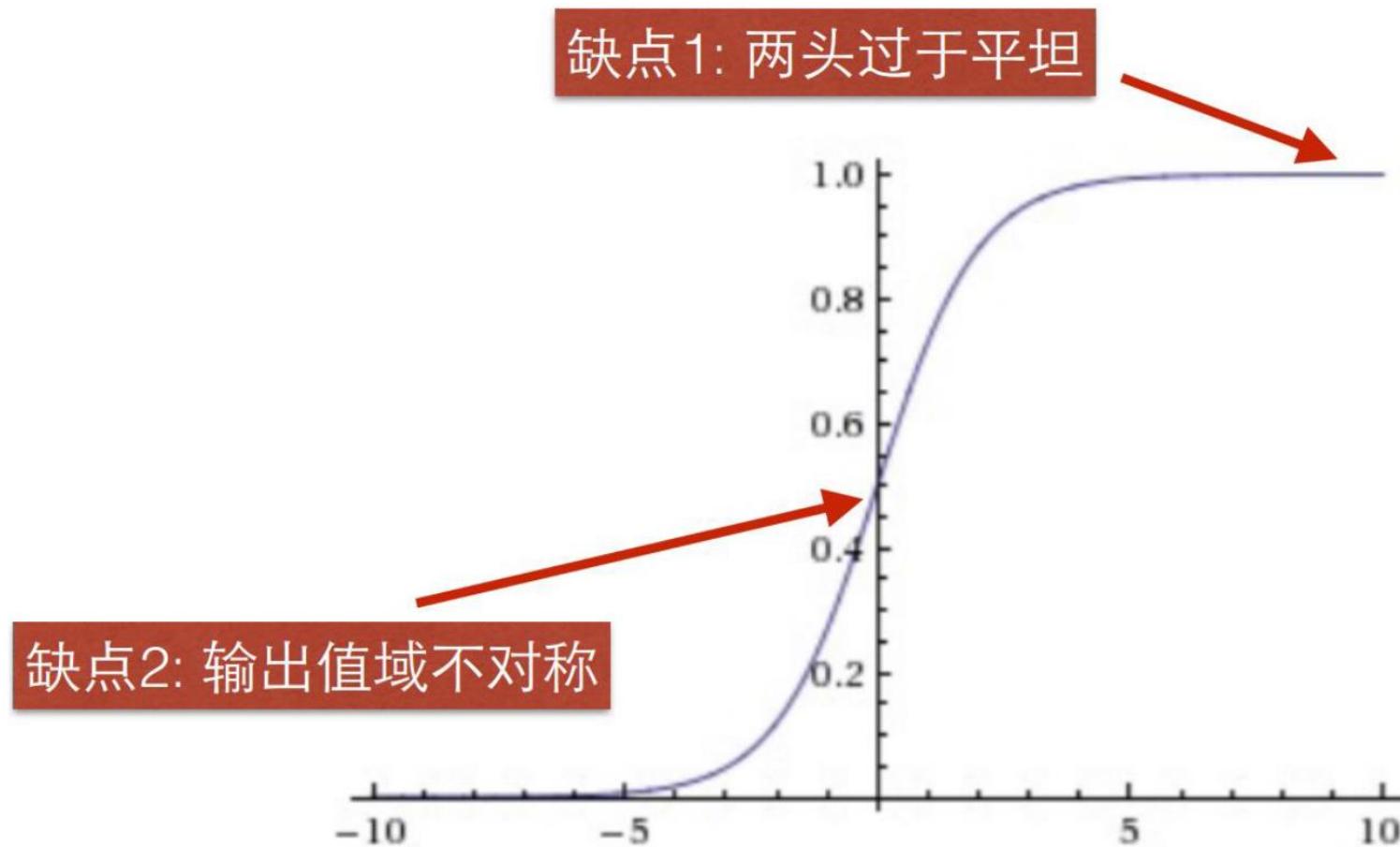


$$f(x) = \max(0, x)$$

线性运算，效率极高，最常用

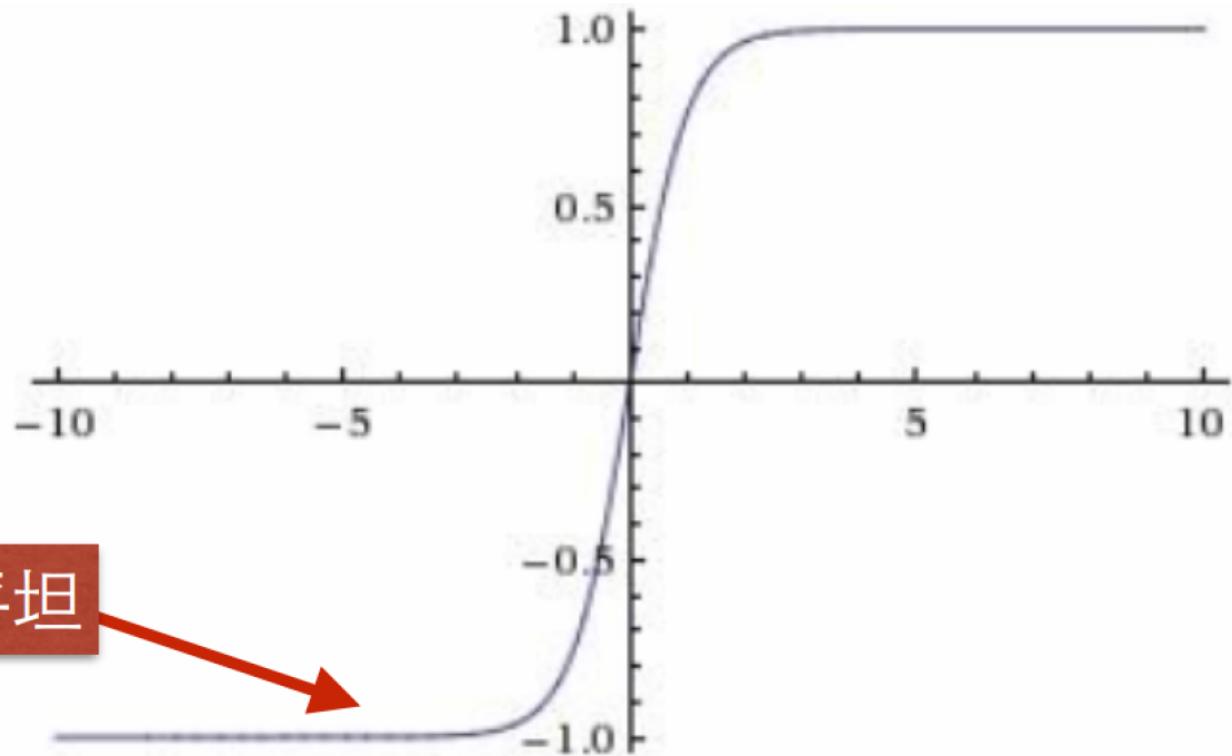
RELU

激活函数：Sigmoid函数



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

激活函数：tanh函数



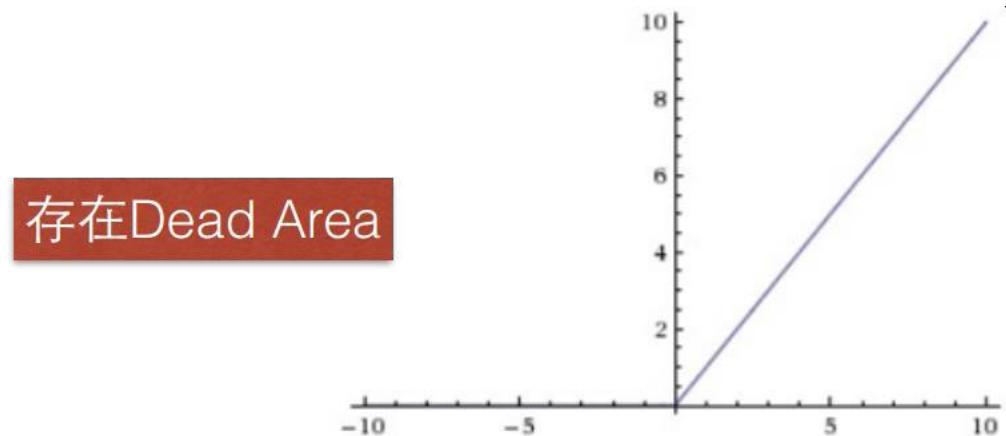
缺点：两头依旧过于平坦

tanh(x)

激活函数：ReLU函数

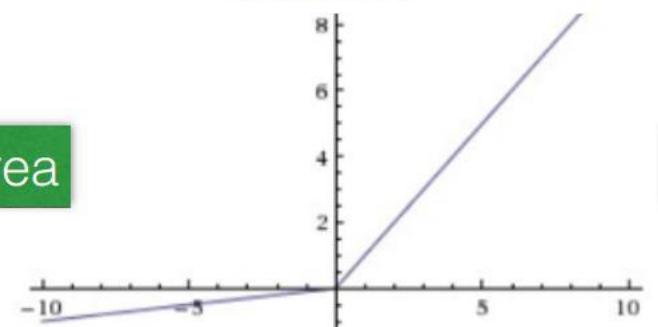
◆ Rectified Linear Unit

- 收敛速度比Sigmoid/tanh快
- 计算高效简单
- Dead Area中权重不更新



ReLU

不存在Dead Area

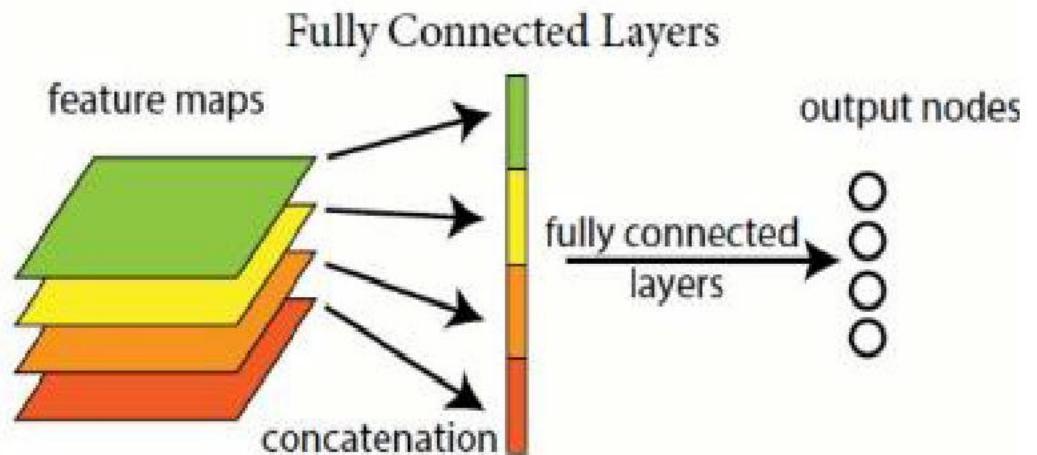


Leaky ReLU

全连接层分类

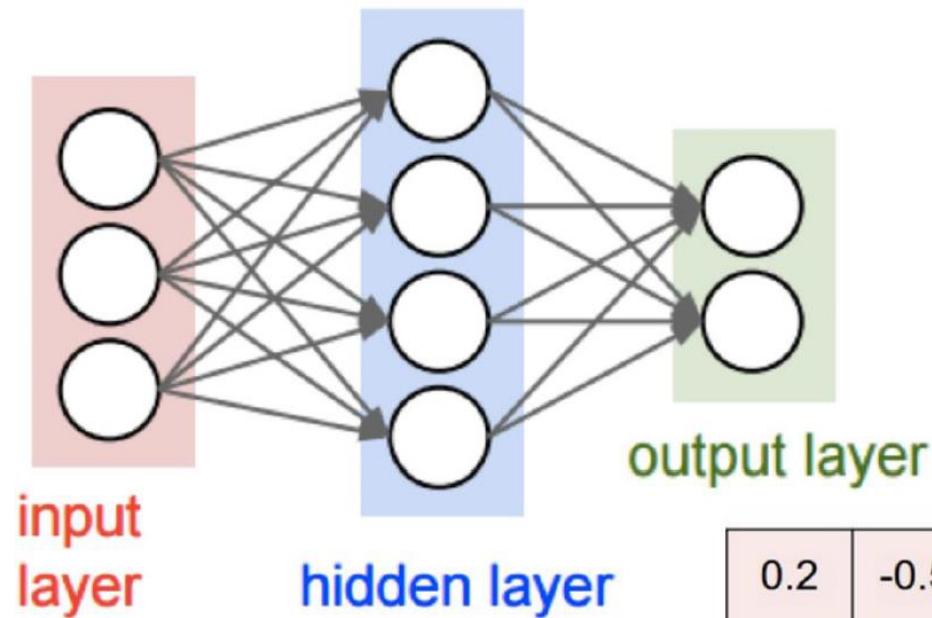
➤ Notes

- ✓ 将多层的特征映射抻直成一个一维的向量
- ✓ 采用全连接的方式将向量连接向输出层，
打破卷积特征的空间限制
对卷积层获得的不同的特征进行加权
最终目的是得到一个可以对不同类别进
行区分的得分
- ✓ 输出层就是对应每个类别的得分





权重



0.2	-0.5	0.1	2.0
1.5	1.3	2.1	0.0
0	0.25	0.2	-0.3

W

$$\begin{array}{c} \downarrow \\ \begin{matrix} 56 \\ 231 \\ 24 \\ 2 \end{matrix} \end{array} + \begin{array}{c} 1.1 \\ 3.2 \\ -1.2 \end{array} \rightarrow \begin{array}{c} -96.8 \\ 437.9 \\ 61.95 \end{array}$$

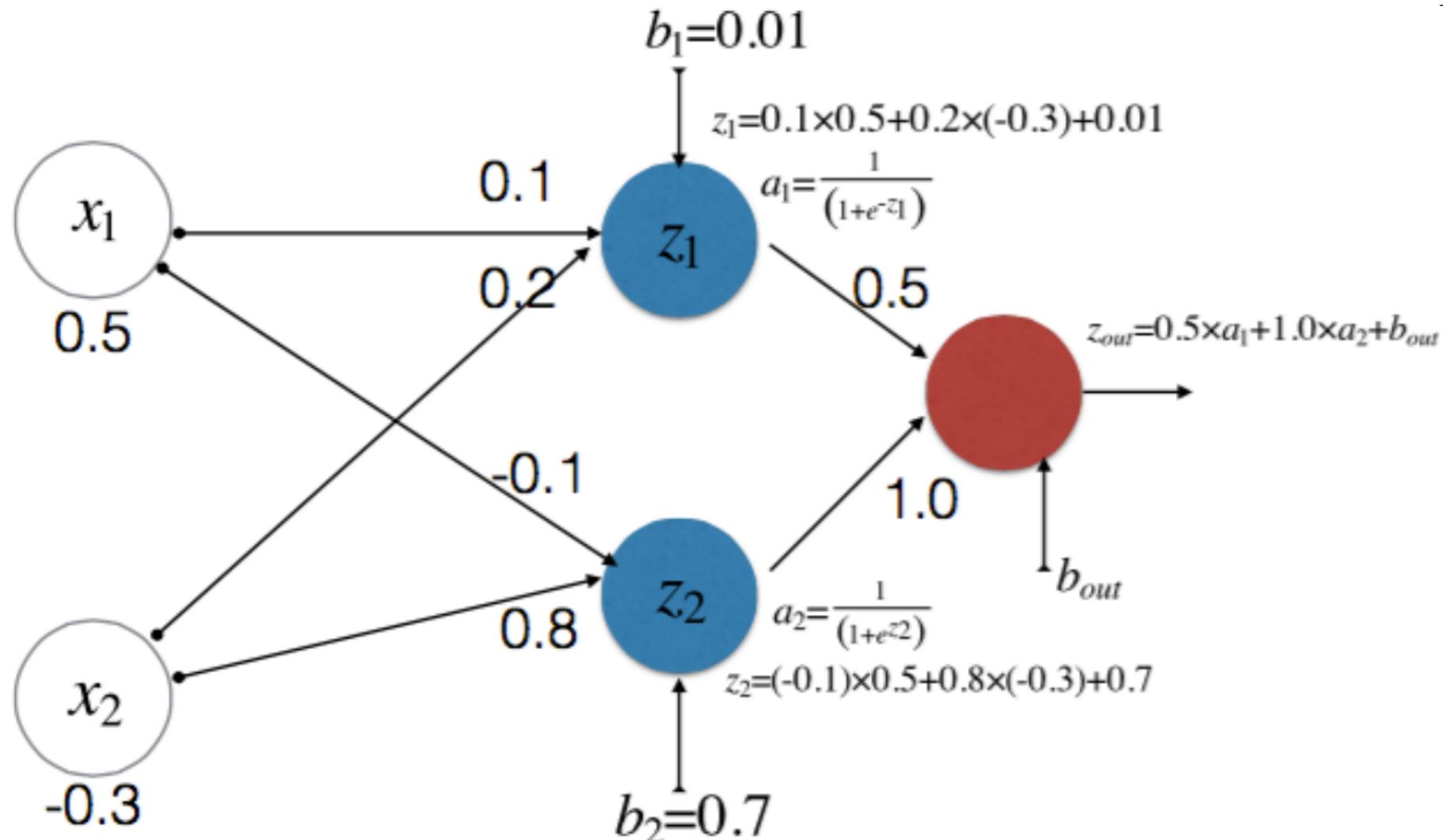
x_i

b

$f(x_i; W, b)$



前向传播





卷积神经网络的局限性

- ① 训练网络模型时，**不仅需要大量的训练样本，还需要高性能算力**，例如需要使用GPU，还需要花大量的时间调试超参数；
- ② **所提取特征的物理含义不明确**，即不知道每个卷积层提取到的特征表示什么含义，神经网络本身就是一种难以解释的“黑箱模型”；
- ③ 深度神经网络**缺乏完备的数学理论证明**，这也是深度学习一直面临的问题，目前仍无法解决。



5.4.3 激活函数

- ◆ 在多层CNN中，第 i 个卷积层的输出与第 $i+1$ 个卷积层的输入之间有一个函数映射，即激活函数。
- ◆ 激活函数是NN中的重要组成部分，它是对网络层输出的线性组合结果做**非线性映射**。
- ◆ 若激活函数是线性函数，则无论神经网络有多少层，整个网络的功能等价于感知机，网络的逼近能力十分有限。
- ◆ **激活函数的作用**就是给网络模型提供非线性的建模能力。
- ◆ 在现代CNN中，每个卷积层后面都有非线性激活函数，目的是向模型中加入非线性元素，以解决非线性问题。**一般在同一个网络中使用同一种激活函数**。
- ◆ 早期，CNN的隐藏层大多采用sigmoid函数作为激活函数，自从2012年起，几乎所有的CNN均采用ReLU系列函数做激活函数了。

5.4.3 激活函数

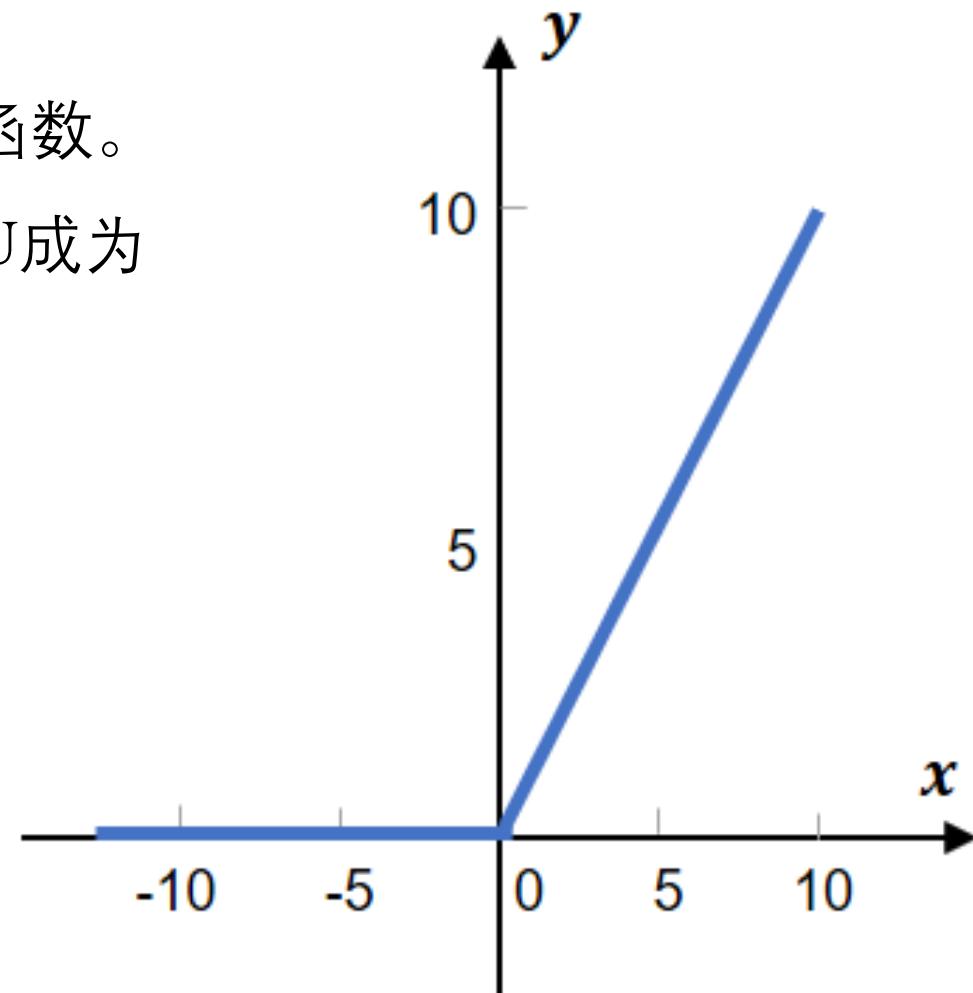
1. ReLU函数

- ReLU是修正线性单元的简称，是最常用的激活函数。
- AlexNet模型采用的激活函数是ReLU，从此ReLU成为深度神经网络模型中应用最广泛的激活函数。
- ReLU是一个简单的分段线性函数，但从整体看，ReLU是一个非线性函数：

$$y = f(x) = \max(x, 0) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

- ReLU函数是分段可导的，并人为规定在0处的梯度为0，其导数形式如下：

$$\frac{\partial y}{\partial x} = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$$



ReLU函数的优点

- (1) **计算简单且快，求梯度简单，收敛速度比 sigmoid与 tanh函数快得多**， ReLU仅需要做简单的阈值运算。
- (2) S型函数在x趋近于正负无穷时，函数的导数趋近于零，而ReLU的导数为0或常数，在一定程度上缓解了梯度消失的问题。
- (3) **ReLU具有生物上的可解释性**，有研究表明：人脑中同一时刻大概只有 1%~4% 的神经元处于激活状态，同时只响应小部分输入信号，屏蔽了大部分信号。sigmoid 函数和 tanh 函数会导致形成一个稠密的神经网络；而ReLU函数在 $x < 0$ 的负半区的导数为0，当神经元激活函数的值进入负半区时，该神经元不会被训练，使得网络具有稀疏性。可见，ReLU 只有大约50%的神经元保持处于激活状态，引入了稀疏激活性，使神经网络在训练时会有更好的表现。

ReLU函数的缺点

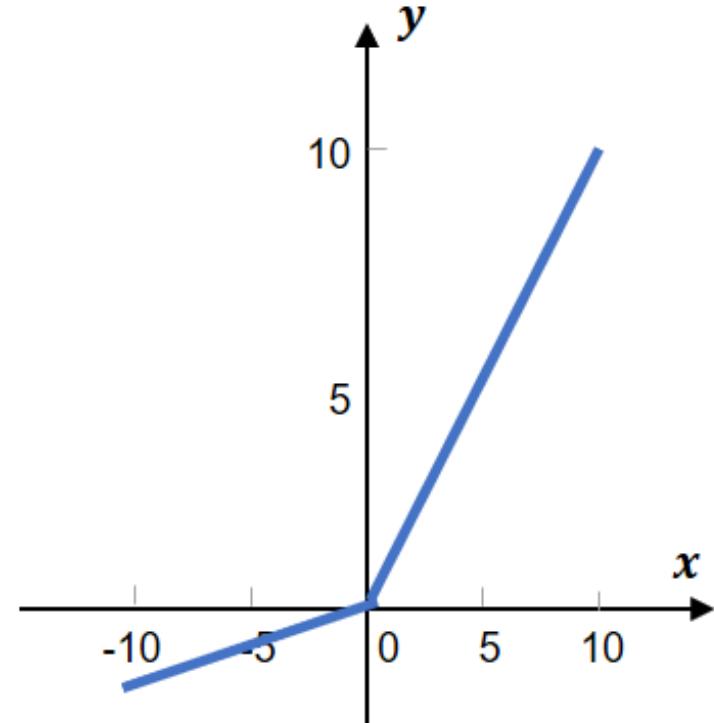
- (1) ReLU函数的**输出是非零中心化的**，使得后一层神经网络的偏置偏移，影响梯度下降的速度。
- (2) 采用ReLU函数，**神经元在训练时比较容易“死亡”**，即在某次不恰当地更新参数后，所有输入都无法激活某个神经元，则该神经元的梯度固定为0，导致无法更新参数，而且在之后的训练中，此神经元再也不会被激活，这种现象称为“**神经元死亡**”问题。在实际使用中，为了避免上述情况，提出了若干ReLU的变种，如 LeakyReLU 函数等。

5.4.3 激活函数

2. LeakyReLU函数

- LeakyReLU称为带泄露的ReLU，简记为LReLU，
- 它在ReLU梯度为0的区域保留了一个很小的梯度，以维持参数更新。
- LReLU函数的数学表达式：

$$y = f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases}$$



其中， $\alpha \in (0, 1)$ 是一个很小的常数，如0.01，当 $\alpha < 1$ 时，LReLU也可以写作：

$$f(x) = \max(x, \alpha x)$$

- LReLU函数的导数形式为： $\frac{\partial y}{\partial x} = \begin{cases} 1, & x > 0 \\ \alpha, & x \leq 0 \end{cases}$
- LReLU函数的导数总是不为零，**解决了一部分神经元死亡的问题。**
- 但在实际使用的过程中，LReLU函数并非总是优于ReLU函数。



(2) 卷积层

- 对灰色图像作卷积操作，用一个卷积核即可，因为一张灰色图像只有一个颜色通道；
- 对彩色图像作卷积操作，则同时需要有3个大小相同的卷积核，分别对彩色图像的R、G、B三个通道进行卷积运算，那么这个由若干个大小（Size，也称为尺寸）相同的卷积核堆叠而成的卷积核组就称为一个滤波器，记为 $\text{Conv F} \times \text{F} \times \text{D}$ ，其中D为该滤波器中包含的卷积核的个数，称为滤波器的深度（Depth），也称为滤波器的通道数，记为 #channel。
- 滤波器的深度取决于输入的图像或特征图的深度。
- 对于灰度图像，一个卷积核可以看作是深度为1的滤波器。

(2) 卷积层

- 一个卷积层中可以包含多个滤波器，而滤波器的个数是人为设定的，一个滤波器就是一个神经元。
- 通常，同层上的多个滤波器的大小、深度都相同，也是人为设定的，即超参数；
- 只是权重和偏置不同，需要训练得到，即学习参数。
- 每个滤波器，即神经元，只对输入图像或特征图的部分区域进行卷积运算，负责提取图像或特征图中的局部特征。
- 靠近输入层的卷积层提取的特征往往是局部的，越靠近输出层，卷积层提取的特征越是全局化的。
- 一个滤波器只关注一个特征，不同的滤波器用于提取不同的特征，如边缘、轮廓、纹理、颜色等。
- 每个滤波器只产生一层特征图，故有多少个滤波器，就会产生多少层特征图。一个神经网络中的所有神经元，就是构成了整张图像的特征提取器的集合。



5.4.1 卷积神经网络的整体结构

(3) 池化层。

- 池化层也称为下采样层、子采样、亚采样。
- 池化层的**作用**：选择具有代表性的特征，去除不重要的或冗余的特征，用以降低特征维数，从而减少参数量和计算开销。

(4) 全连接层。

- 全连接层**通常置于CNN尾部**，它与传统的全连接神经网络的连接方式相同，都是**相邻两层上的所有神经元两两相连**，每个连接都有权重。
- 两个相邻全连接层之间只进行线性转换，不用激活函数。
- 全连接层的**作用**主要是对卷积层和池化层提取的局部特征进行重新组合，得到全局信息，以减少特征信息的丢失。



5.4.4 池化运算

- ◆ 池化是一种非线性下采样，池化层也称为下采样层。
- ◆ 最常见的池化运算采用大小为 2×2 、步长为2的滑动窗口操作，有时窗口尺寸为 3，更大的窗口尺寸比较罕见。
- ◆ 实际上，池化也是一个特殊的卷积运算，它与普通卷积的区别有以下两点：
 - ① 池化卷积核的步长等于卷积核的大小，即 $s=F$ ，称为**不重叠的池化**；
 - ② 池化卷积核的权重不是通过学习获得的，而是允许人为选择进行下采样的方式，所以，**池化核的权重不是学习参数**。
- ◆ 两种常用的池化运算
 - **最大池化** (Max Pooling)，它取滑窗内所有元素中的最大值；
 - **平均池化** (Average Pooling)，它取滑窗内所有元素的平均值。

5.4.4 池化运算

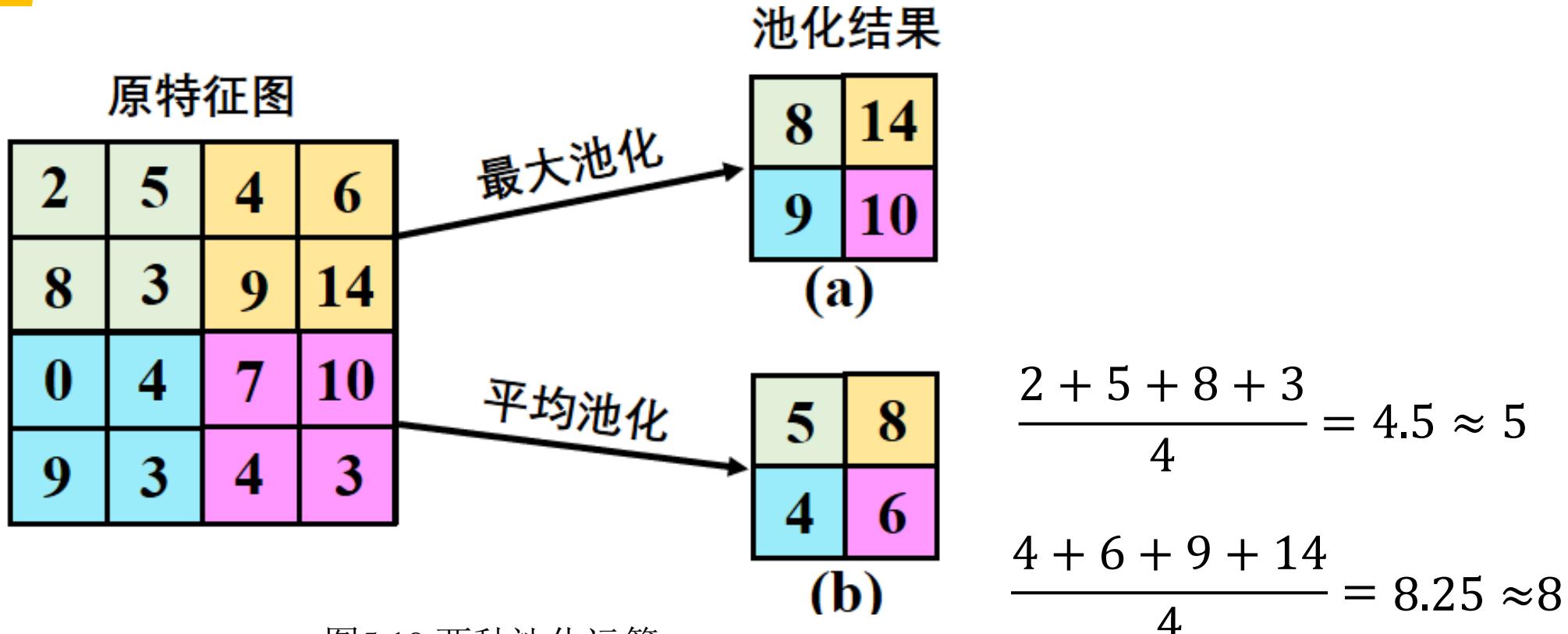


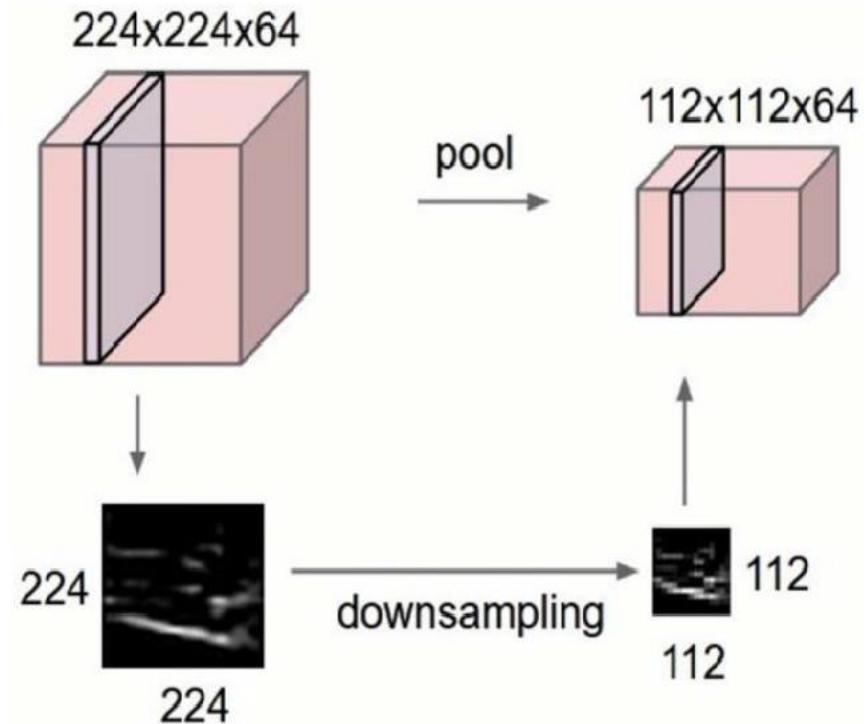
图5.19 两种池化运算

- ◆ **平均池化**操作可以较好地保留图像的背景信息，但是图像中的物体边缘会被钝化。
- ◆ **最大池化**操作可以更清晰地保留图像的纹理信息，在提取目标轮廓等特征时会更有效。

池化层

➤ 池化

- ✓ 在width和height维度上进行下采样，不改变depth的维度
- ✓ 右图相当于对输入数据使用了 2×2 , stride = 2的卷积核，但是该卷积核不是通过学习获得，而是人为定义的卷积核（不算做参数）
- ✓ 能够成倍减少计算量
- ✓ 相比stride，池化层可以选择进行下采样的方式





池化层

- ◆ 没有可学习的参数
- ◆ 在每个输入通道应用池化层以获得相应的输出通道
- ◆ $\# \text{输出通道} = \# \text{输入通道}$



5.4.4 池化运算

池化层夹在连续的“卷积层+ReLU”组合块中间，其作用如下。

(1) 保持尺度不变性。

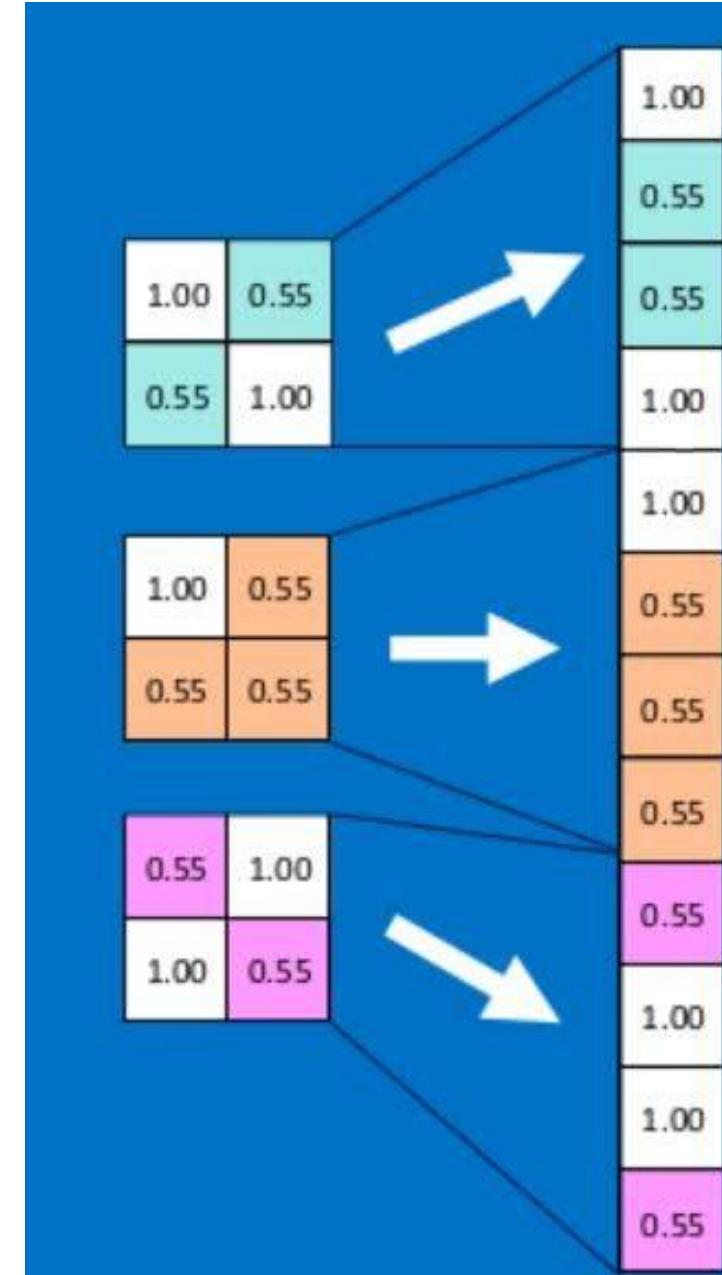
池化操作是对特征图中的元素进行选择，保留图像的重要特征，去掉一些冗余的信息，保留下来的信息具有尺度不变性的特征，可以很好地表达图像的特征。

(2) 降低特征维度。

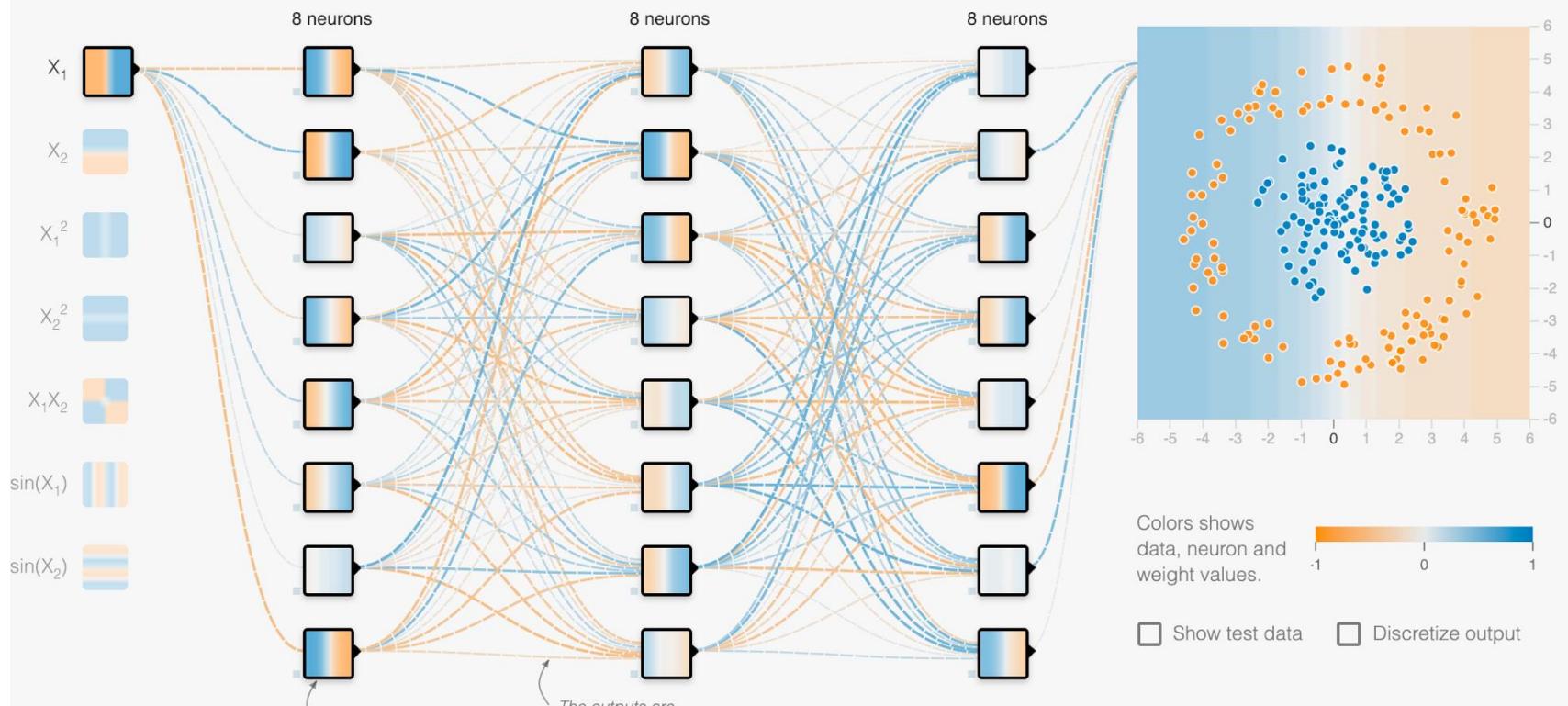
池化操作分别独立作用于特征图的每个通道上，降低所有特征图层的宽度和高度，但不改变图像的深度（通道数）。可以起到降低特征维度、防止过拟合的作用，同时可以减少参数量和计算开销。

全连接的操作方式

- ◆ 先将经过卷积、激活函数、池化的多层特征向量抻直成一个**一维的向量**；
- ◆ 再采用**全连接**的方式将上述**一维向量与输出层**连接。



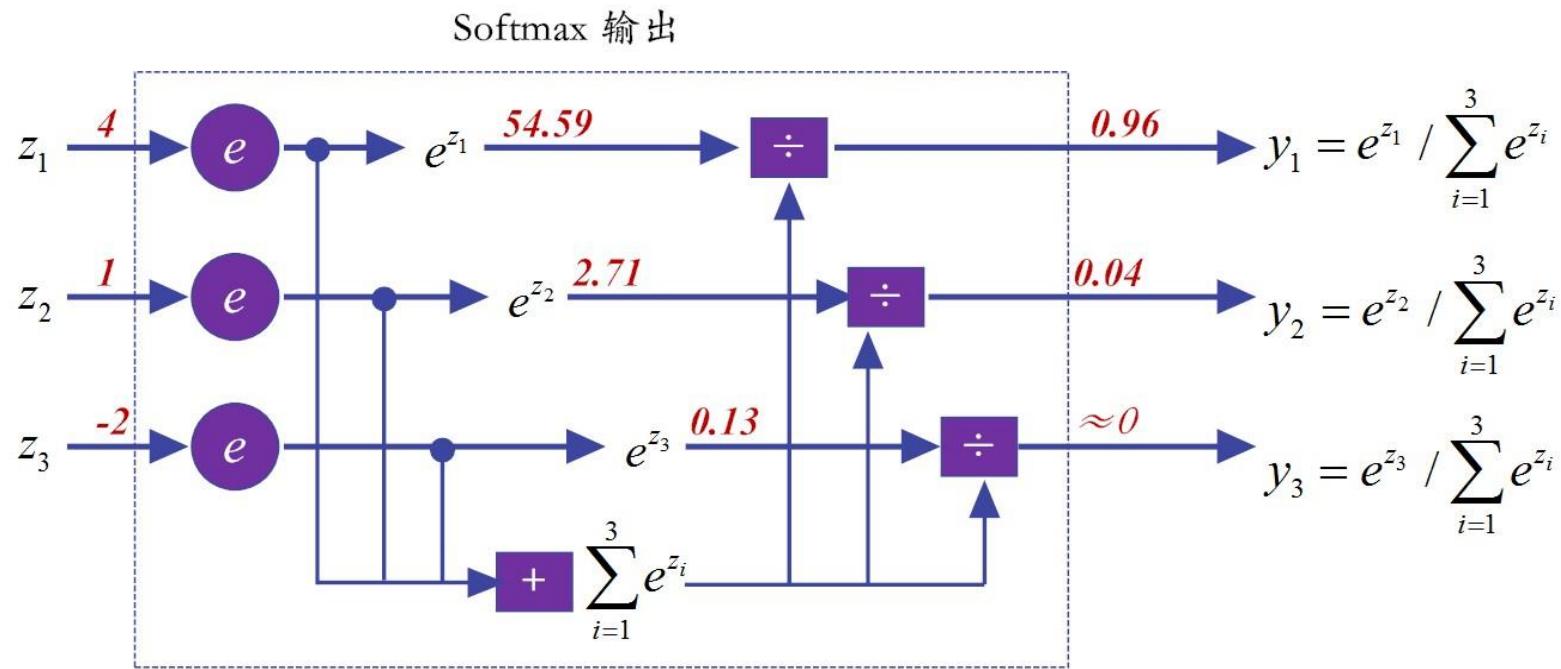
全连接层



全连接层 (Dense Layer) 目的：在于分类/回归等。

前面若干层（卷积、激活、池化等）目的：数据变换-->提取特征，作为全连接层的输入，为全连接层服务。

全连接层



$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (j = 1, 2, 3, \dots, k)$$



5.4.1 卷积神经网络的整体结构

(5) 输出层。

- 输出层用于输出结果，实际上就是最后一个全连接层之后的激活函数层。
- 根据回归或分类问题的需要选择不同激活函数，以获得最终想要的结果。
- 例如，二分类任务可以选择sigmoid作为激活函数，若是多分类任务，可以选择softmax作为激活函数。

◆ CNN的结构比传统多层全连接神经网络具有以下**两个方面优势**。

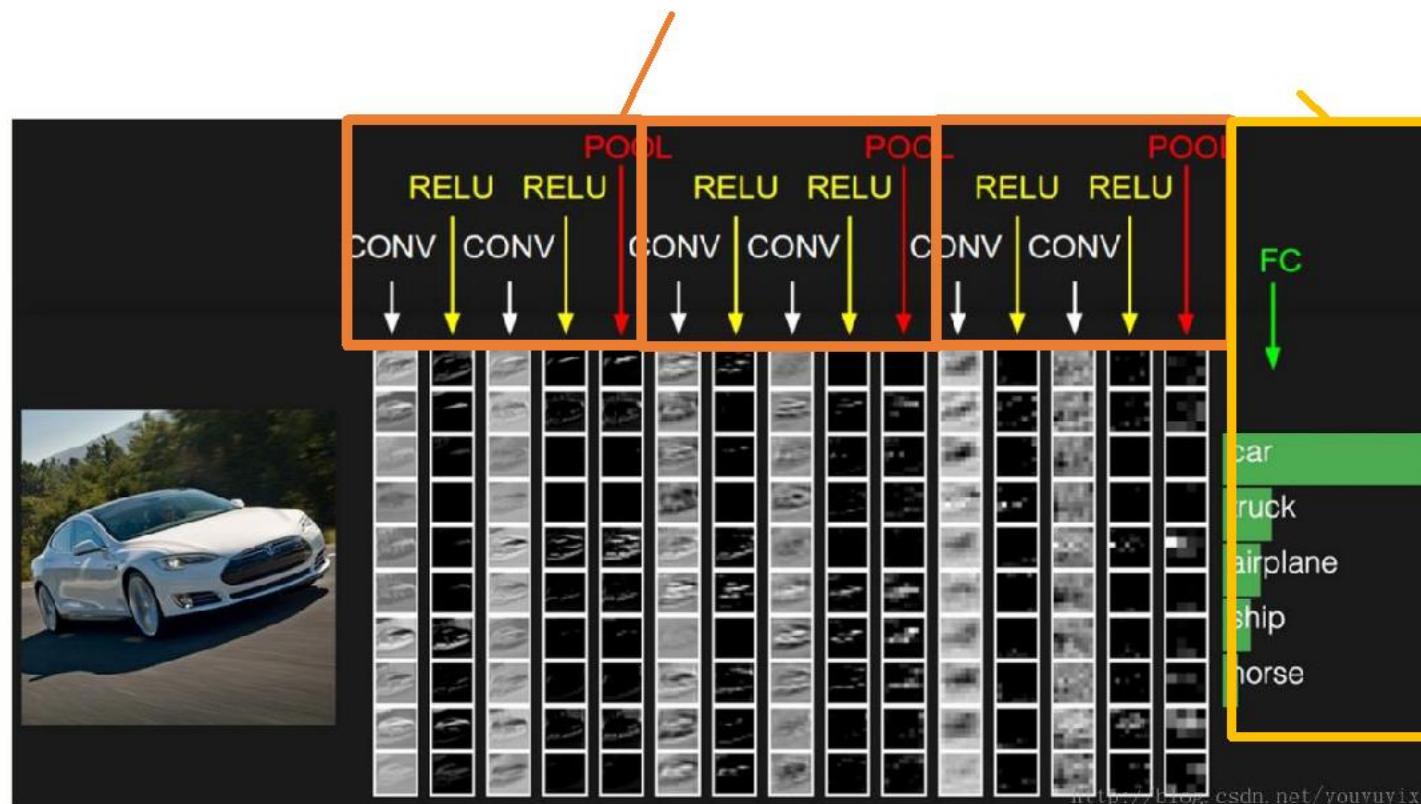
- (1) 连接方式不同，CNN的局部连接可大大减少参数量和计算量。
- (2) 对输入图像处理的方式不同，CNN保留了图像固有的空间特征。

CNN基础结构

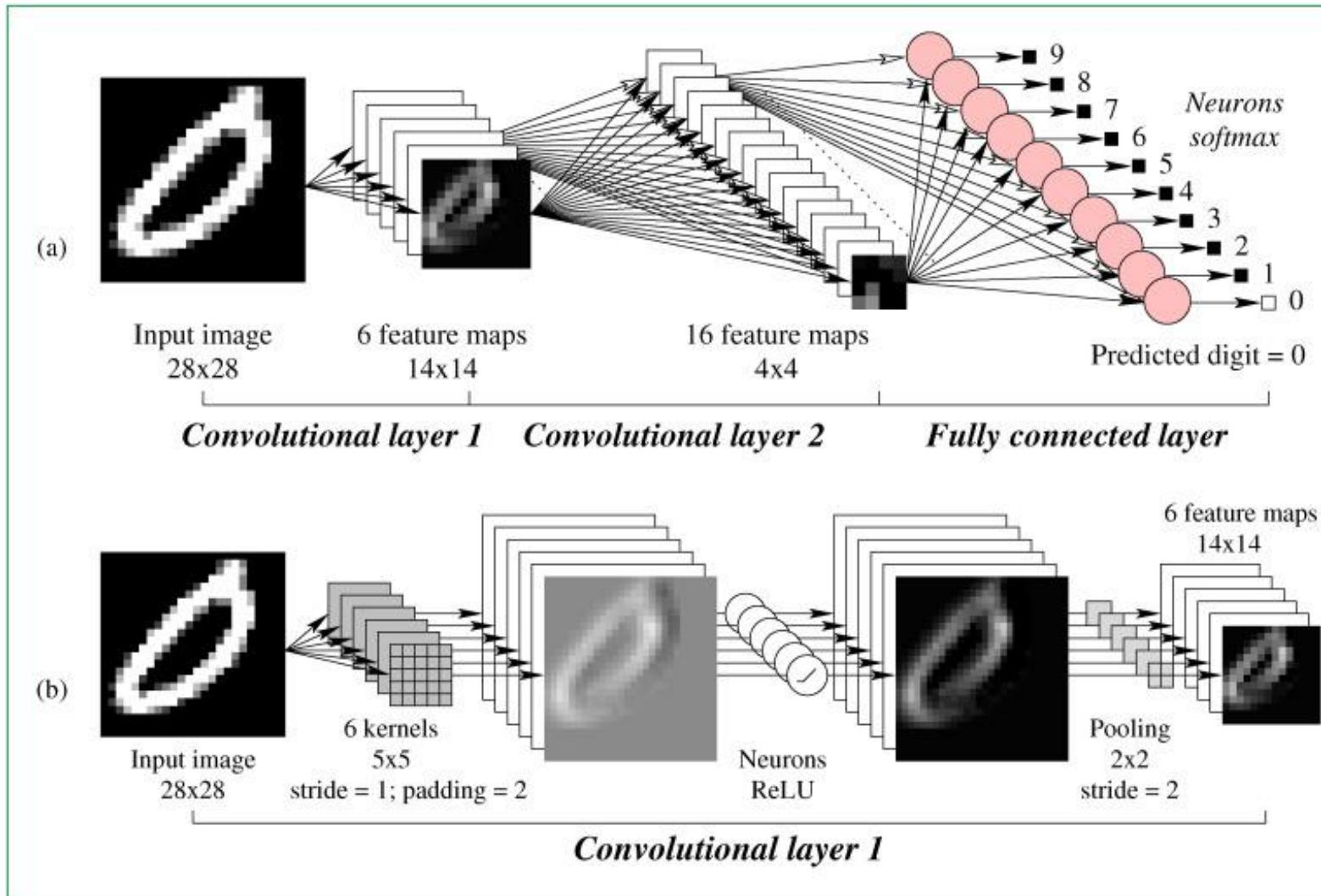
◆ CNN应用图像模式的一般框架（以分类为例）

卷积层+激活函数+池化层+全连接层

出现多次，用于提取特征 在最后出现一次或多次，用于做分类



卷积神经网络的结构



正则化手段

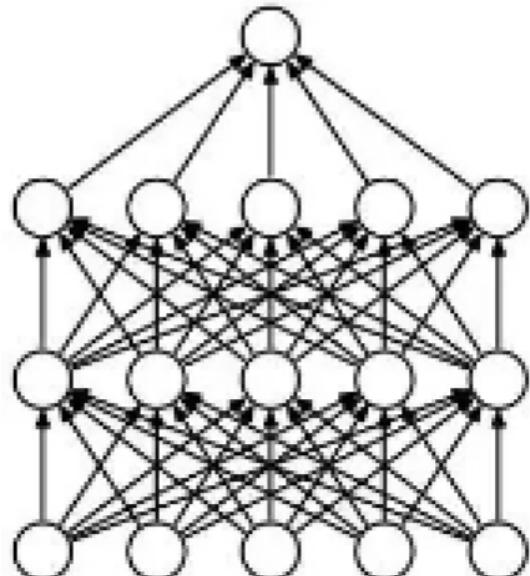
- 在原损失函数中增加L1/L2 norm项

$$L = \frac{1}{N} \sum_i L_i + \lambda \sum_j w_j^2$$

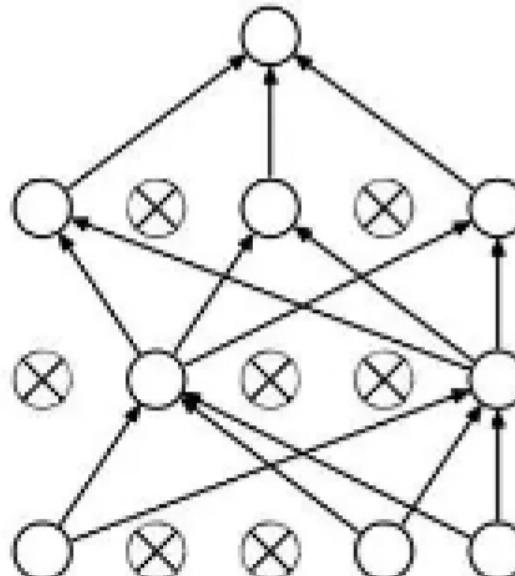
- 在训练的过程中对样本/权重增加噪声
 - 将256*256的图像随机选择224*224的子图作为网络的输入数据
- Dropout：训练过程中保持部分网络的连接

Dropout

- 随机扔掉(drop)一定比例的神经元
 - 在训练数据上防止了单元之间共同起作用
 - 隐层的单元不再依赖于其他单元
 - 迫使每一个单元学习到有效的特征



(a) Standard Neural Net

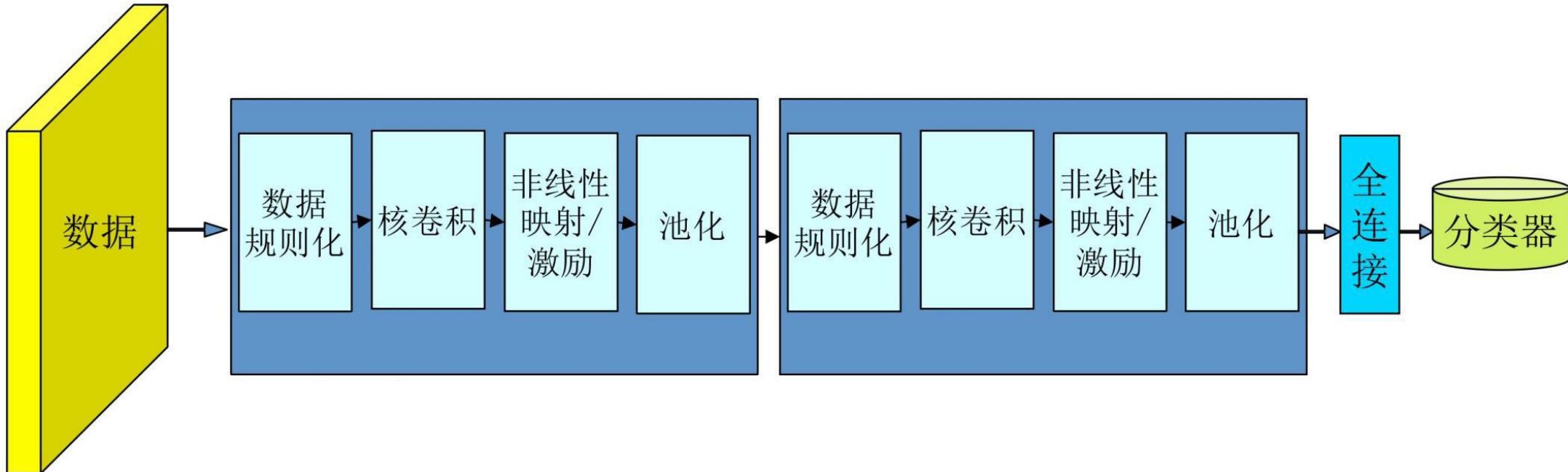


(b) After applying dropout.

网络搭建小结

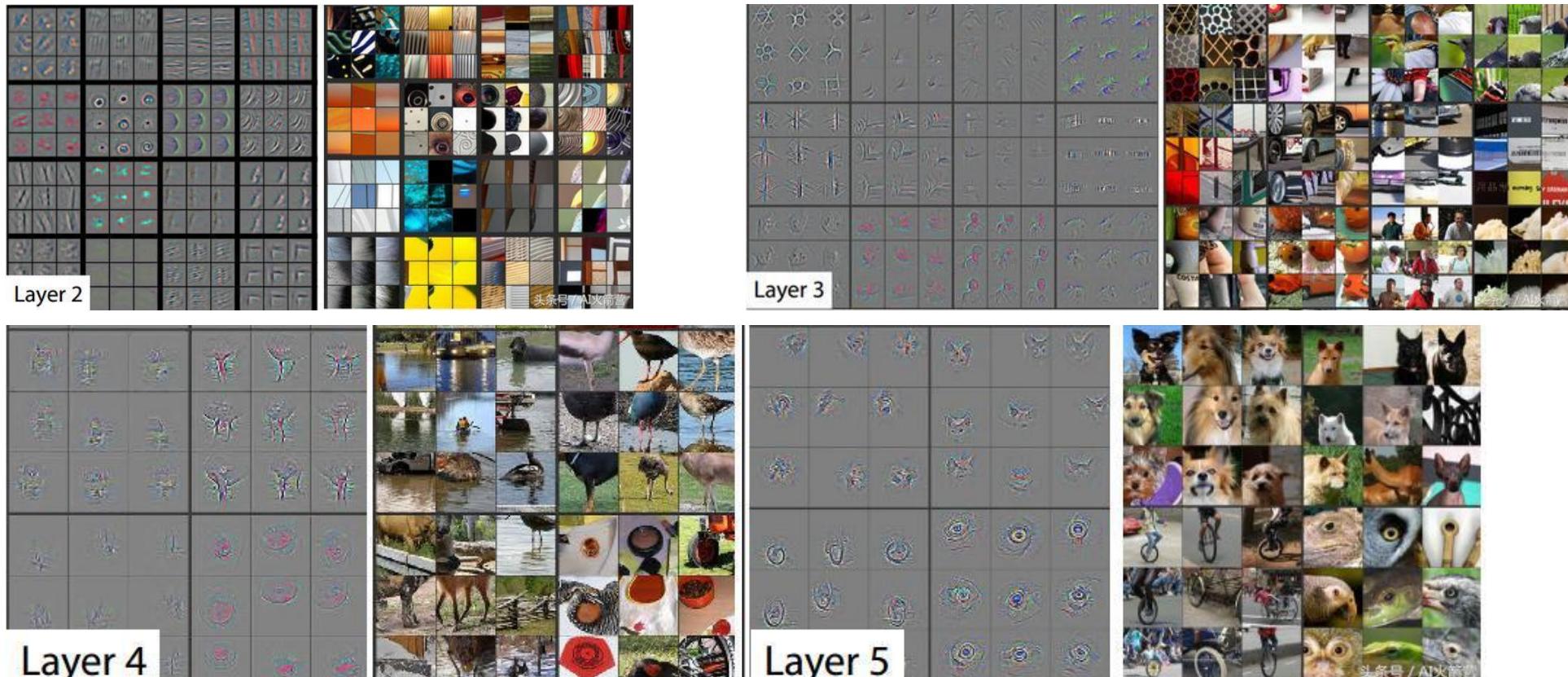
卷积神经网络的一般结构

1. 卷积层+ReLU和 池化层 的组合多次出现 提取特征
2. 多个全连接 或 特殊的CNN结构 作为输出层 作分类器/检测器/分割器



卷积神经网络到底学到了什么？

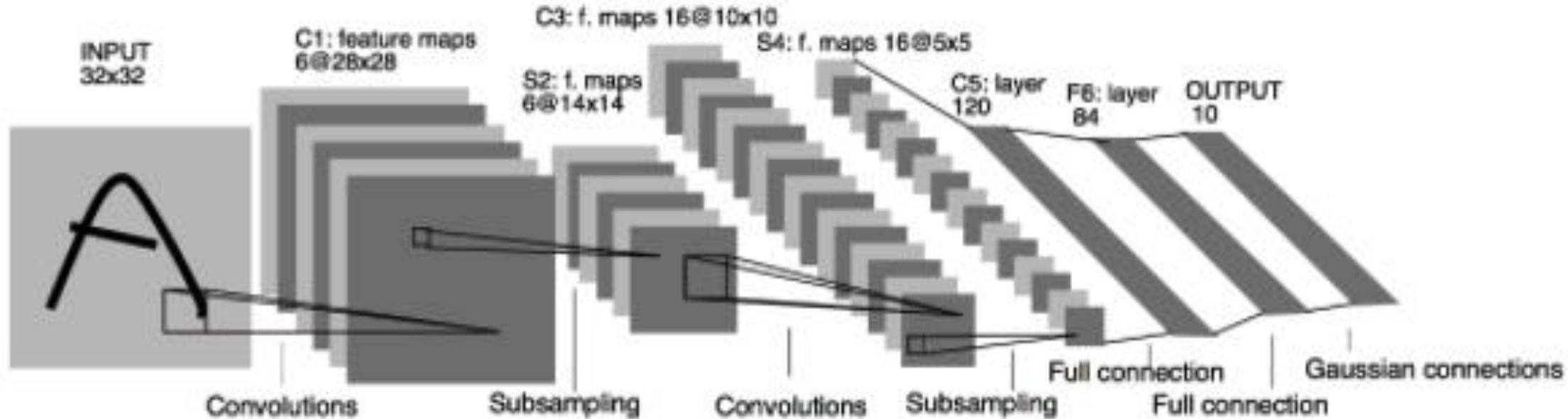
◆ https://www.sohu.com/a/274607482_100285715





经典的CNN

经典卷积神经网络LeNet模型



Yann LeCun在1998年提出用于文字识别的LeNet模型是非常经典的模型
第一个成功大规模应用的卷积神经网络，
在MNIST数据集中的正确率可以高达99.2%，其网络结构如图所示。

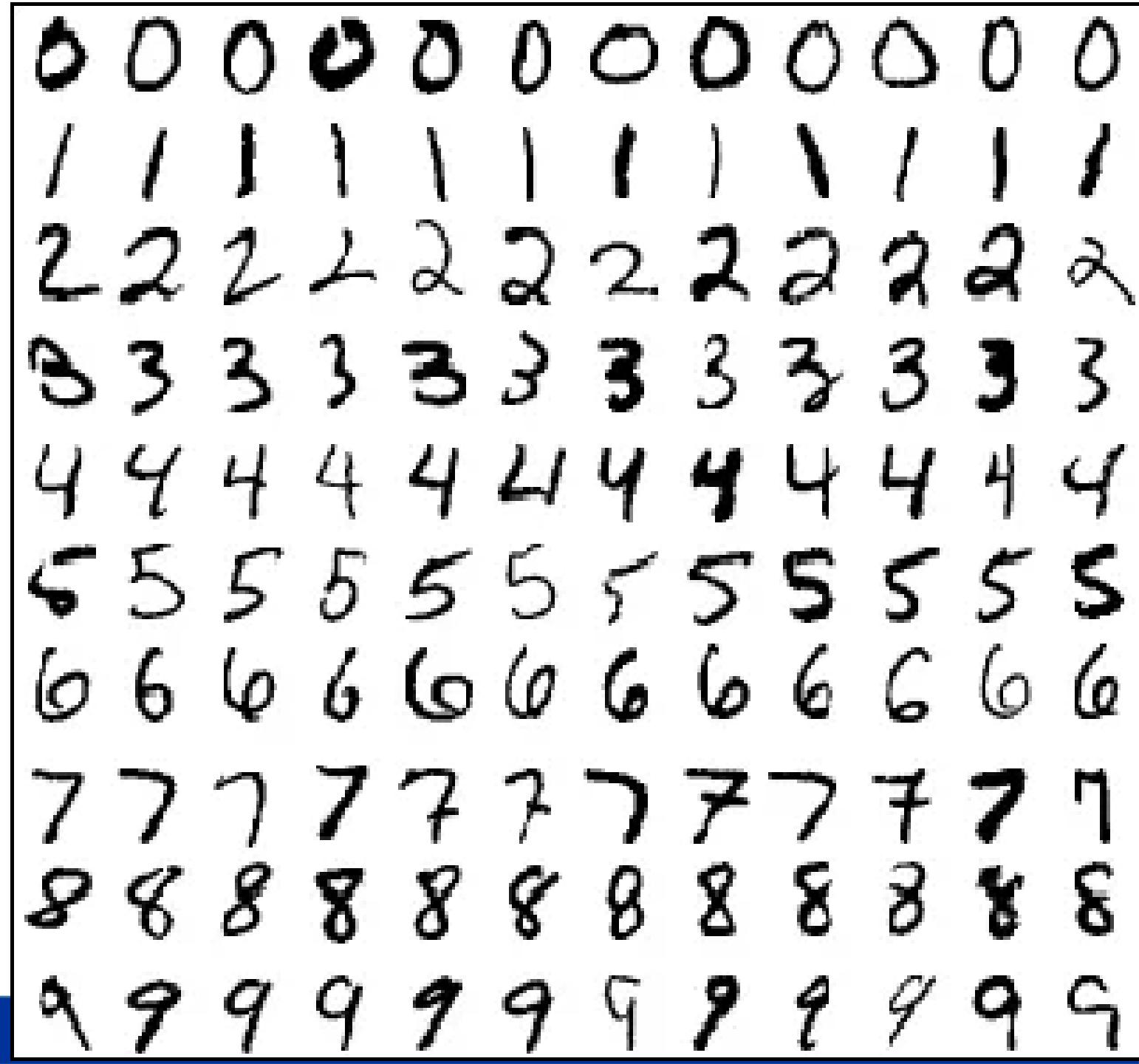


mnist数据集

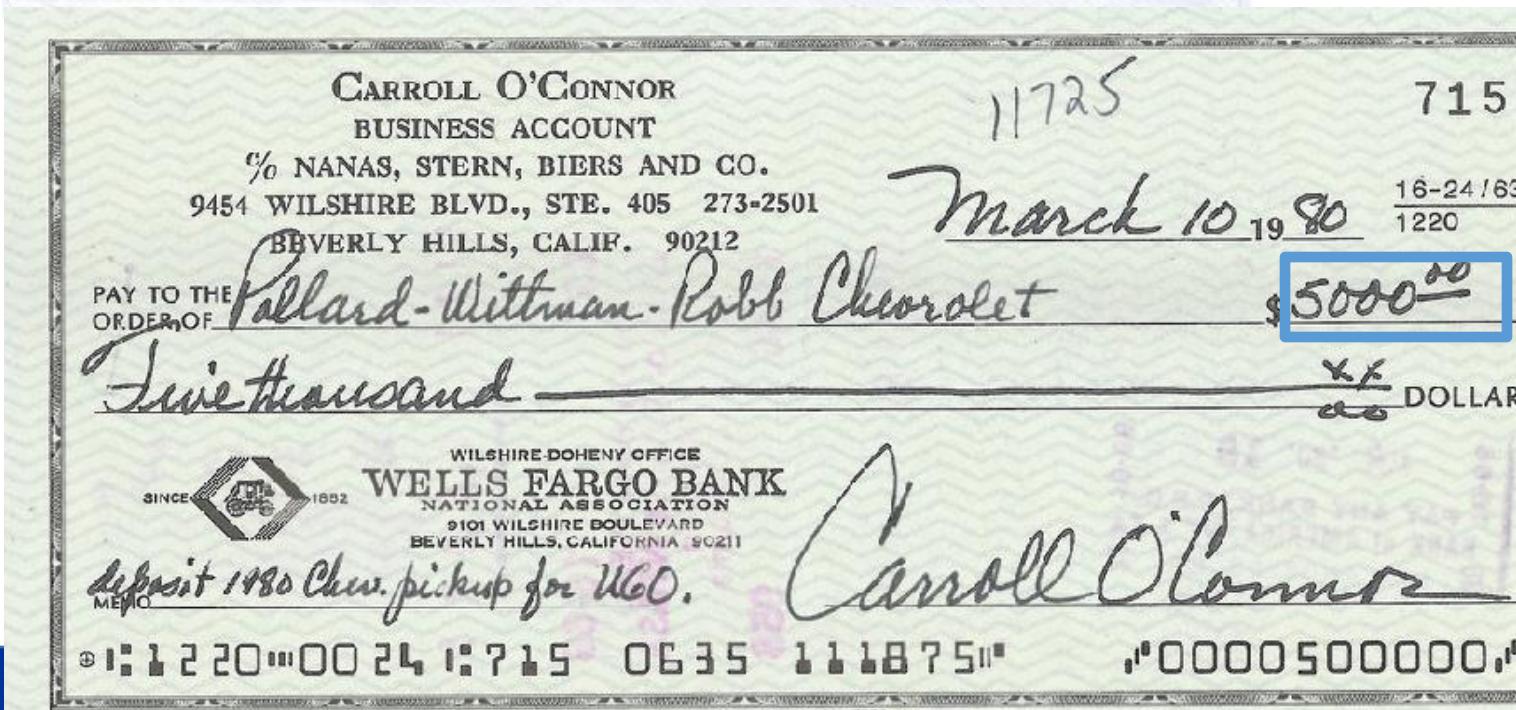
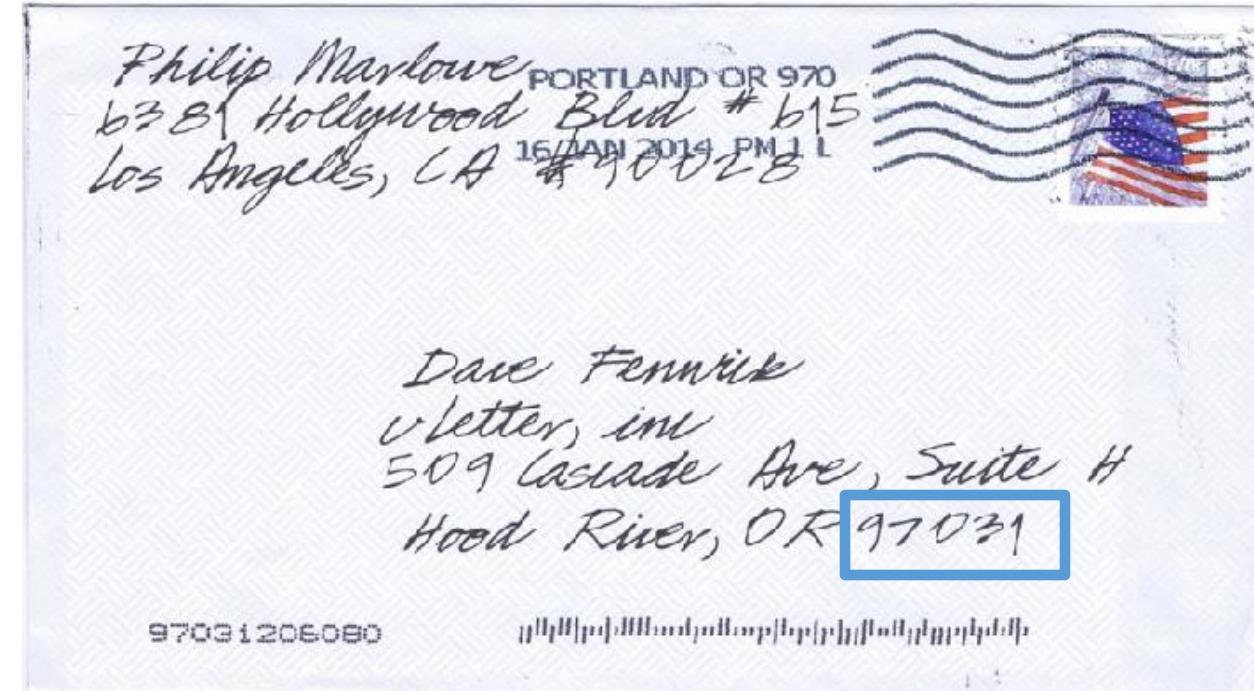
- ◆ MNIST 数据集来自美国国家标准与技术研究所, National Institute of Standards and Technology (NIST). 训练集 (training set) 由来自 250 个不同人手写的数字构成, 其中 50% 是高中学生, 50% 来自人口普查局 (the Census Bureau) 的工作人员. 测试集(test set) 也是同样比例的手写数字数据.
- ◆ MNIST 包括6万张28x28的训练样本, 1万张测试样本, 很多教程都会对它”下手”几乎成为一个“典范”, 可以说它就是计算机视觉里面的Hello World。
- ◆ 官网: <http://yann.lecun.com/exdb/mnist/>

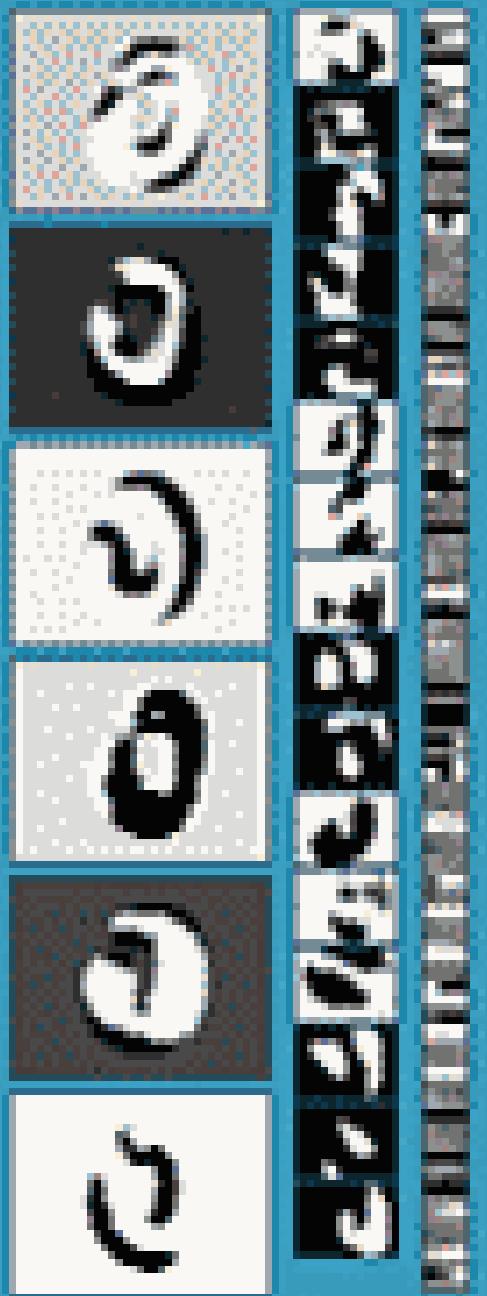


- ◆ 居中和缩放
- ◆ 50,000 个训练数据
- ◆ 10,000 个测试数据
- ◆ 图像大小28*28
- ◆ 10 类



手写的数字识别



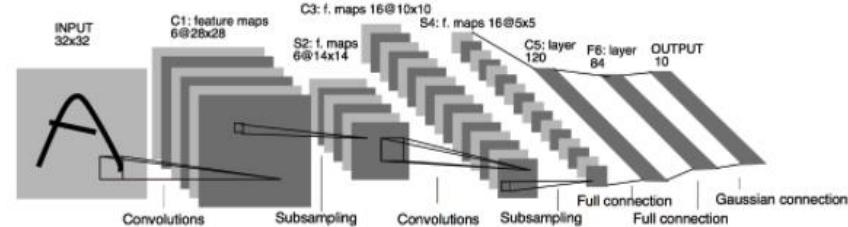


The image shows a screenshot of the LeNet-5 research software interface. At the top, there is a logo for AT&T and the text "LeNet 5 RESEARCH". Below this, the word "answer:" is followed by a large red digit "0". In the center, there is a 3x3 grid of smaller images showing the digit "0" at different stages of processing. Below this grid is a larger image of a handwritten digit, which appears to be a "4".

Y. LeCun, L. Bottou,
Y. Bengio, P.
Haffner, 1998
Gradient-based
learning applied to
document
recognition

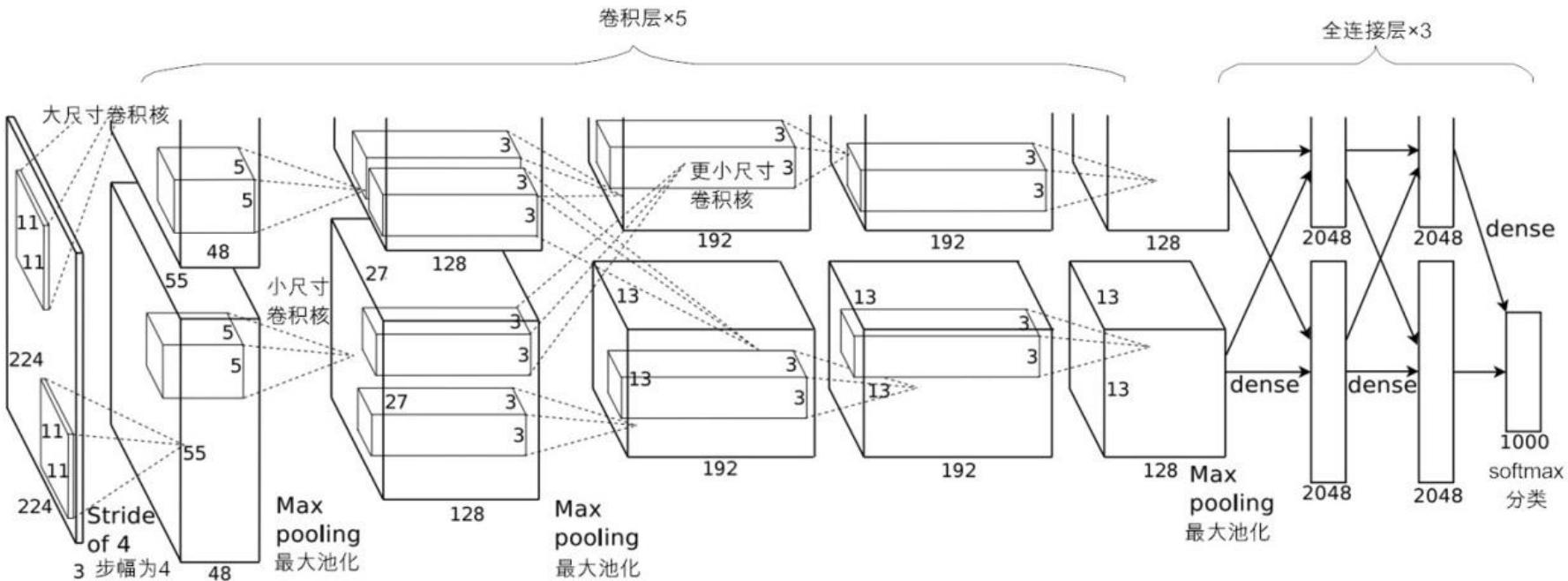


LeNet



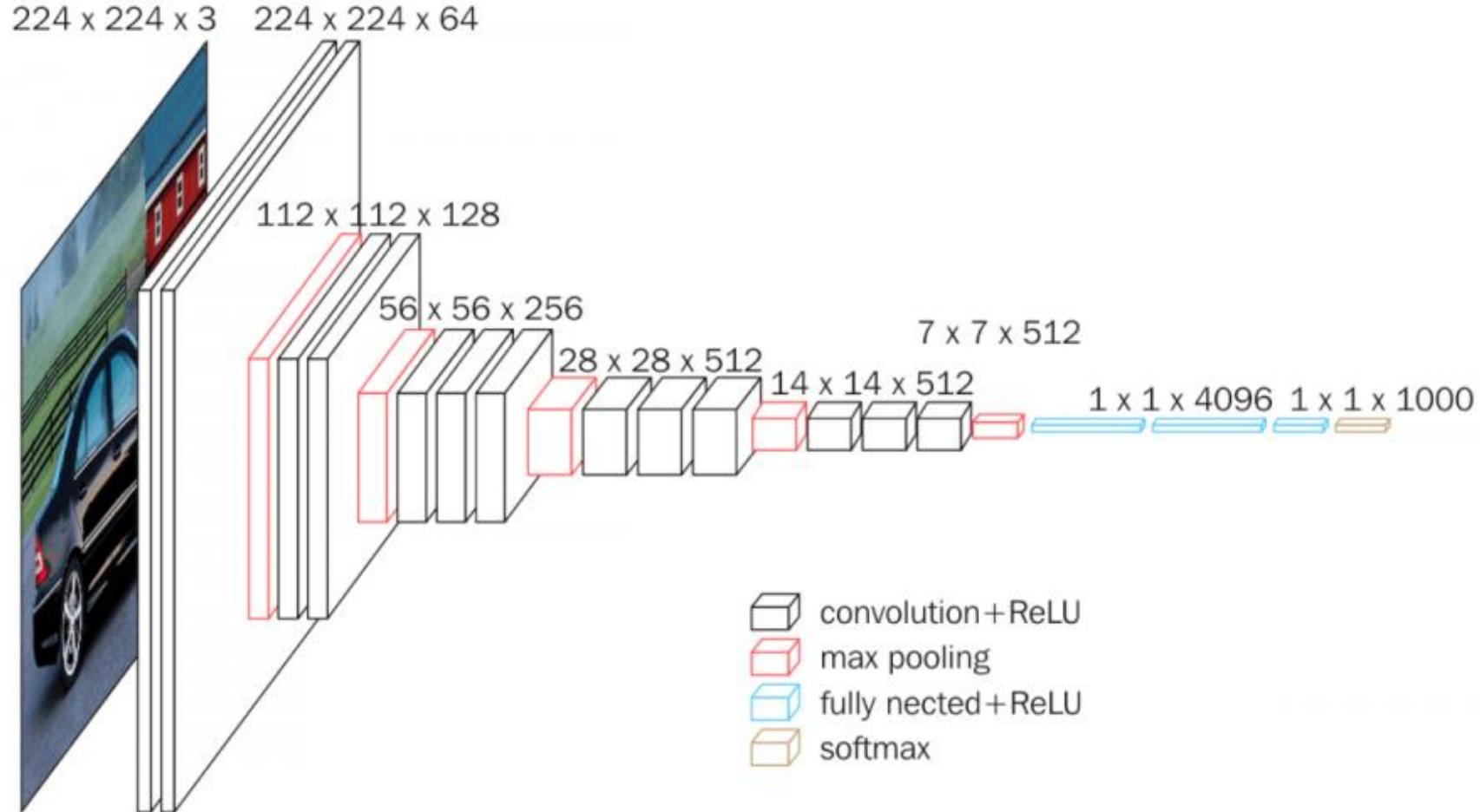
- ◆ LeNet-5卷积神经网络模型一共有7层，包含卷积层，池化层（下采样层），全连接层等。
- ◆ 以手写数字识别案例来看：
- ◆ 第一层（C1）是卷积层,分别采用了6个不同的卷积核，每个卷积核尺寸均为 5×5 。
- ◆ 第二层（S2）是一个 2×2 的池化层，对六个特征图分别池化得到6个 14×14 的特征图。
- ◆ 第三层（C3）又是一个卷积层，这次采用了16个 5×5 的卷积核得到16个 10×10 的特征图。
- ◆ 第四层（S4)是池化层，同样采用 2×2 的池化，对16个C3的特征图处理得到16个 5×5 的特征图。
- ◆ 第五层（C5）是一个包含120个神经元的卷积层，采用 5×5 的卷积核。
- ◆ 第6层（F6）则包含84个神经元，与C5进行全连接。
- ◆ 第7层，10类输出。

AlexNet



AlexNet的整体架构图

GoogLeNet



◆ VGGNet获得2014年ImageNet亚军， VGG是牛津大学 Visual Geometry Group (视觉几何组) 的缩写，以研究机构命名。

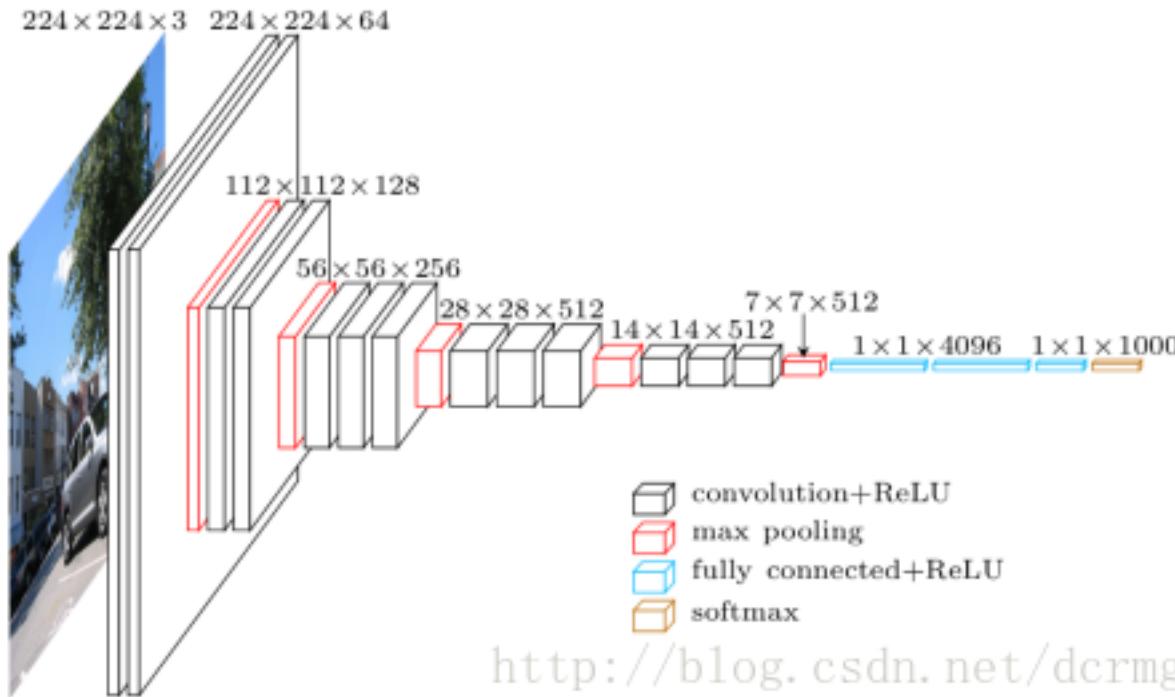


Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as “conv(receptive field size)-(number of channels)”. The ReLU activation function is not shown for brevity.

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 conv1-256	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
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
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A-A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

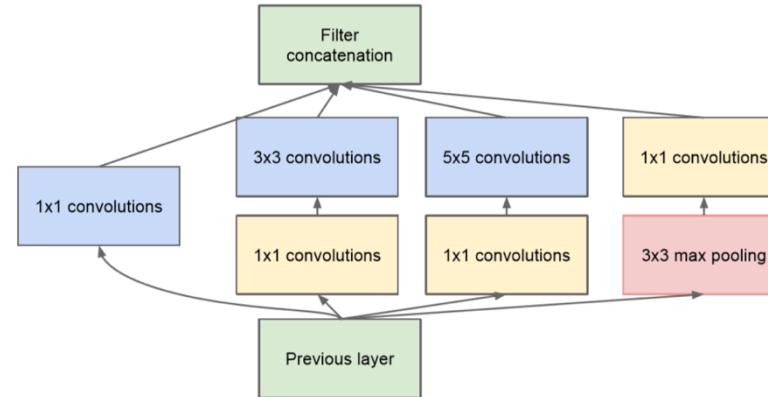
Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

GoogLeNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2	64	192					112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture

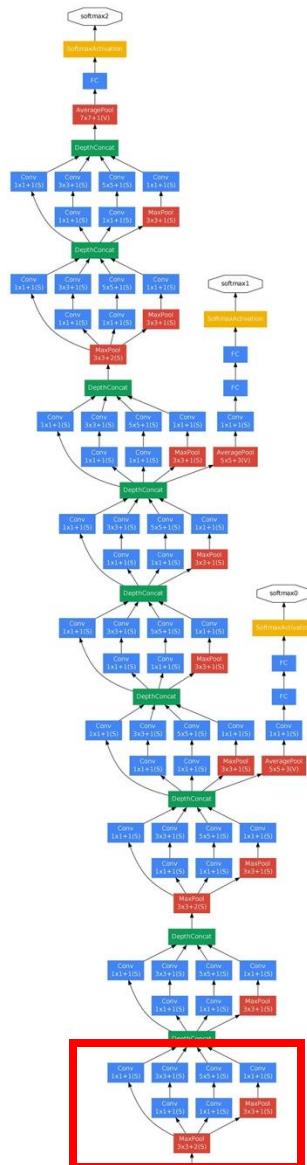
https://blog.csdn.net/qq_38807688



(b) Inception module with dimension reductions

https://blog.csdn.net/qq_38807688

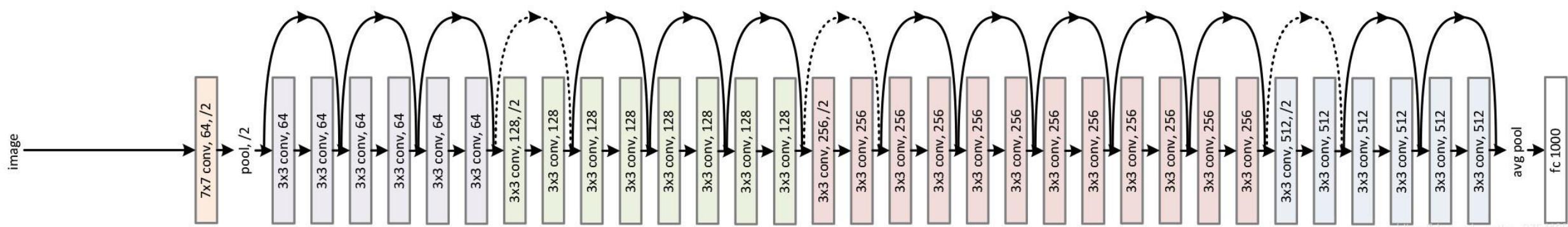
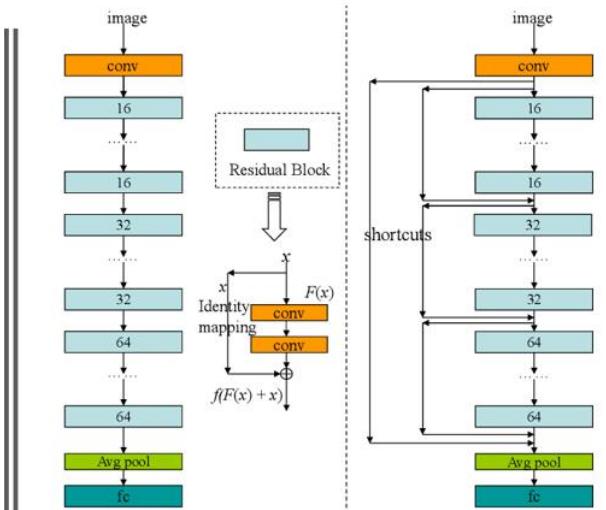
Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1-9. 2015.



ResNet残差网络



何恺明



He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112					
conv2_x	56×56	$[3\times 3, 64] \times 2$	$[3\times 3, 64] \times 3$	$1\times 1, 64$ $3\times 3, 64 \times 3$ $1\times 1, 256$	$1\times 1, 64$ $3\times 3, 64 \times 3$ $1\times 1, 256$	$1\times 1, 64$ $3\times 3, 64 \times 3$ $1\times 1, 256$
conv3_x	28×28	$[3\times 3, 128] \times 2$	$[3\times 3, 128] \times 4$	$1\times 1, 128$ $3\times 3, 128 \times 4$ $1\times 1, 512$	$1\times 1, 128$ $3\times 3, 128 \times 4$ $1\times 1, 512$	$1\times 1, 128$ $3\times 3, 128 \times 8$ $1\times 1, 512$
conv4_x	14×14	$[3\times 3, 256] \times 2$	$[3\times 3, 256] \times 6$	$1\times 1, 256$ $3\times 3, 256 \times 6$ $1\times 1, 1024$	$1\times 1, 256$ $3\times 3, 256 \times 23$ $1\times 1, 1024$	$1\times 1, 256$ $3\times 3, 256 \times 36$ $1\times 1, 1024$
conv5_x	7×7	$[3\times 3, 512] \times 2$	$[3\times 3, 512] \times 3$	$1\times 1, 512$ $3\times 3, 512 \times 3$ $1\times 1, 2048$	$1\times 1, 512$ $3\times 3, 512 \times 3$ $1\times 1, 2048$	$1\times 1, 512$ $3\times 3, 512 \times 3$ $1\times 1, 2048$
				average pool, 1000-d fc, softmax		
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

http://blog.csdn.net/qq_37541091



总结

- 卷积神经网络是一种特殊的多层前馈神经网络，常用于监督学习。
- **卷积神经网络结构**包括一个输入层、多个“卷积层+ReLU层+池化层（池化层有时可以省略）”组合块、多个连续的全连接层（中间不加激活函数）、一个采用softmax函数的输出层。
- 卷积运算前后特征图尺寸的变化可通过公式计算。
- **卷积神经网络具有局部连接和权值共享的特点**。每个卷积层可以有多个滤波器，一个滤波器就是一个神经元，一个滤波器可以包含多个卷积核，一个滤波器内的多个卷积核共用一个偏置值。
- **池化层的作用**是保持尺度不变性和降低特征维度。
- **激活函数**必须是非线性的，目的是大幅度提升深度神经网络的表达能力，用以逼近任何函数，提供非线性的建模能力。每个卷积层后面都有一个激活函数。