中国科学院大学

University of Chinese Academy of Sciences

Deep Learning

Convolutional Neural Network

Xinfeng Zhang（张新峰）

School of Computer Science and Technology

University of Chinese Academy of Sciences

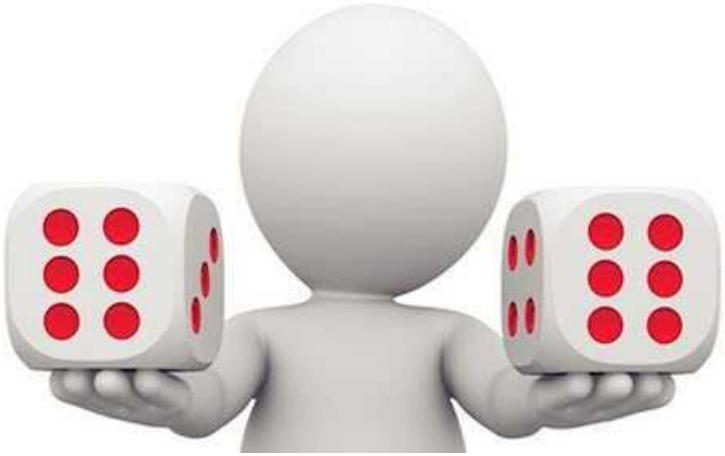
Email: xfzhang@ucas.ac.cn

提纲

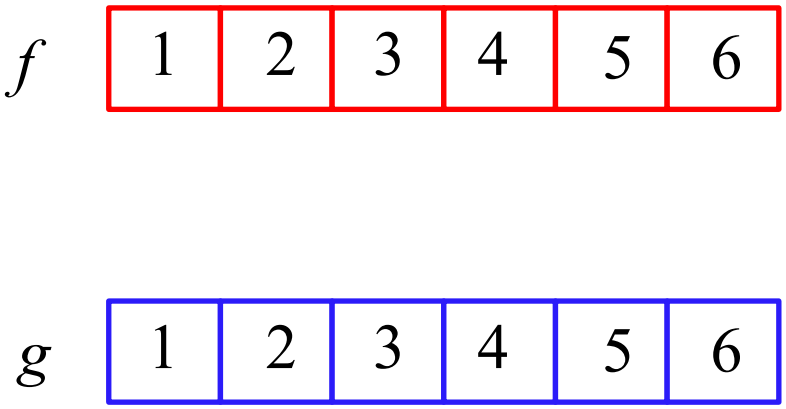
- 卷积的基本概念
- Hubel-Weisel实验/神经认知机
- 卷积神经网络基本原理
- 经典卷积神经网络
- 卷积神经网络的主要应用
- 中英文术语对照

1 卷积的基本概念

示例:求扔两个色子结果之和为4的概率



示例:求扔两个色子结果之和为4的概率



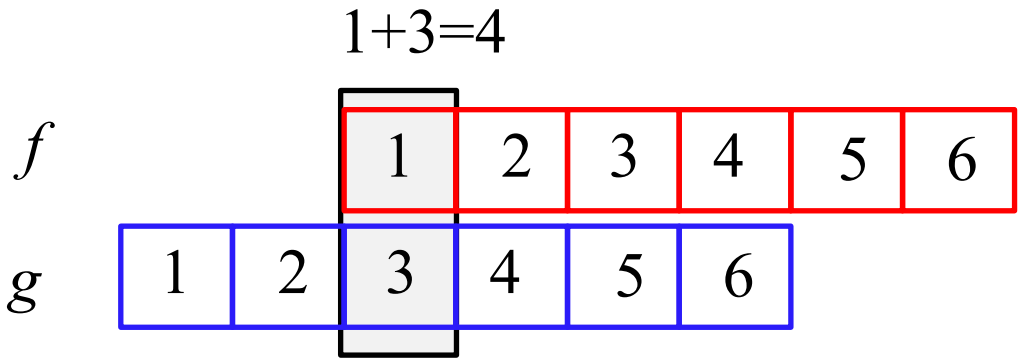
*f*表示第一个色子

f(1)表示第一个色子投出为1的概率

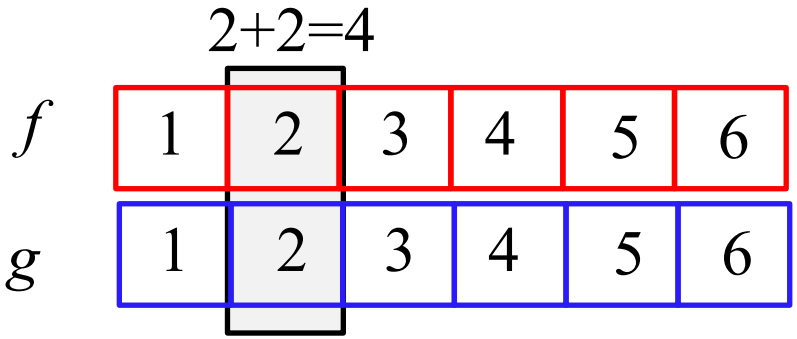
f(2)、*f*(3)、…以此类推

*g*表示第二个色子

示例:求扔两个色子结果之和为4的概率

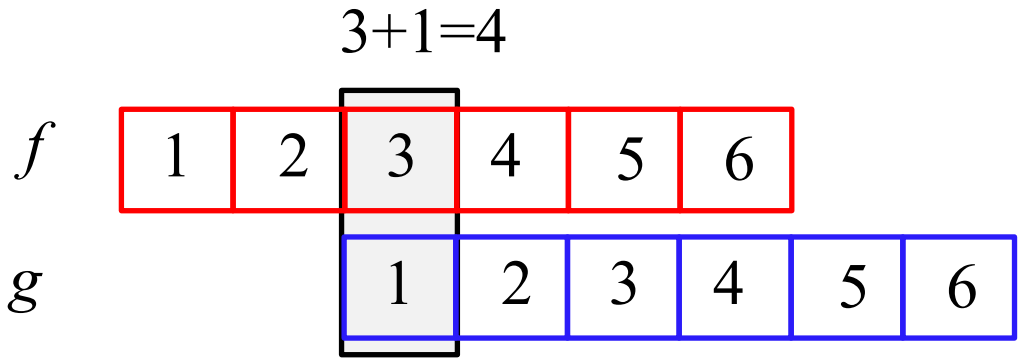


出现概率为：$f(1)g(3)$



出现概率为：$f(2)g(2)$

示例:求扔两个色子结果之和为4的概率



出现概率为：$f(3)g(1)$

- 两枚色子点数加起来为4的概率为：

  $f(1)g(3)+f(2)g(2)+f(3)g(1)$

- 写成标准的形式：

$$(f*g)(4)=f(4)*g(4)=\sum_{m=1}^{3}f(m)g(4-m)$$

示例:求存款复利

本金	第一年	第二年	第三年	第四年	第五年
100	$100\times(1.05)^1$	$100\times(1.05)^2$	$100\times(1.05)^3$	$100\times(1.05)^4$	$100\times(1.05)^5$

本金	第一年	第二年	第三年	第四年	第五年
+100	$100\times(1.05)^1$	$100\times(1.05)^2$	$100\times(1.05)^3$	$100\times(1.05)^4$	$100\times(1.05)^5$
	+100	$100\times(1.05)^1$	$100\times(1.05)^2$	$100\times(1.05)^3$	$100\times(1.05)^4$

本金	第一年	第二年	第三年	第四年	第五年
+100	$100\times(1.05)^1$	$100\times(1.05)^2$	$100\times(1.05)^3$	$100\times(1.05)^4$	$100\times(1.05)^5$
	+100	$100\times(1.05)^1$	$100\times(1.05)^2$	$100\times(1.05)^3$	$100\times(1.05)^4$
		+100	$100\times(1.05)^1$	$100\times(1.05)^2$	$100\times(1.05)^3$
			+100	$100\times(1.05)^1$	$100\times(1.05)^2$
				+100	$100\times(1.05)^1$
					+100

示例:求存款复利

复利(5年) 复利(4年) 复利(3年) 复利(2年) 复利(1年) 复利(0年)

$$100 \times (1.05)^5 + 100 \times (1.05)^4 + 100 \times (1.05)^3 + 100 \times (1.05)^2 + 100 \times (1.05)^1 + 100 \times (1.05)^0$$

第0年存入的钱 第1年存入的钱 第2年存入的钱 第3年存入的钱 第4年存入的钱 第5年存入的钱

$$\sum_{i=0}^{5} f(i)\, g(5-i), \qquad where\ f(i) = 100, g(5-i) = (1.05)^{5-i}$$

存钱函数 复利计算函数

$$f(\tau)\ (0 \le \tau \le t)$$

$$g(t-\tau) = (1+5\%)^{t-\tau}$$

$$\int_0^t f(\tau) g(t-\tau) d\tau = \int_0^t f(\tau)(1+5\%)^{t-\tau} d\tau$$

卷积公式

❑ 卷积(Convolution)

– $(f*g)(n)$成为f和g的卷积，连续卷积和离散卷积可以表达为如下形式：

$$(f*g)(n)=\int_{-\infty}^{\infty} f(\tau)g(n-\tau)d\tau$$ 连续卷积

$$n=\tau+(n-\tau)$$

$$(f*g)(n)=\sum_{\tau=-\infty}^{\infty} f(\tau)g(n-\tau)$$ 离散卷积

卷积的应用

- 输出=输入*系统
- 主要应用
  - 统计学中加权平均法
  - 概率论中两个独立变量之和概率密度的计算
  - 信号处理中的线性系统
  - 物理学的线性系统
  - 图像处理中的应用（卷积神经网络）

卷积的应用:加权移动平均法

- 移动平均法(如月销量预测)

$$F_t = (F_{t-n} + F_{t-n+1} + \cdots + F_{t-1})/n$$

- 加权移动平均法(如月销量预测)

$$F_t = w_n \times F_{t-n} + w_{n-1} \times F_{t-n+1} + \cdots + w_1 \times F_{t-1}$$

$$= \sum_{i=1}^{n} w_i \times F_{t-i}$$

$$w_1 + w_2 + \cdots + w_n = 1$$

卷积的应用:独立变量之和的概率密度计算

- 设X、Y随机变量，已知联合概率密度函数$f(x,y)$，求Z=X+Y的概率密度

$$f_Z(z)=\int_{-\infty}^{\infty} f(z-y,y)dy$$

$$f_Z(z)=\int_{-\infty}^{\infty} f(x,z-x)dx$$

- X、Y相互独立时

$$f_Z(z)=\int_{-\infty}^{\infty} f_X(z-y)f_Y(y)dy=\int_{-\infty}^{\infty} f_X(x)f_Y(z-x)dx$$

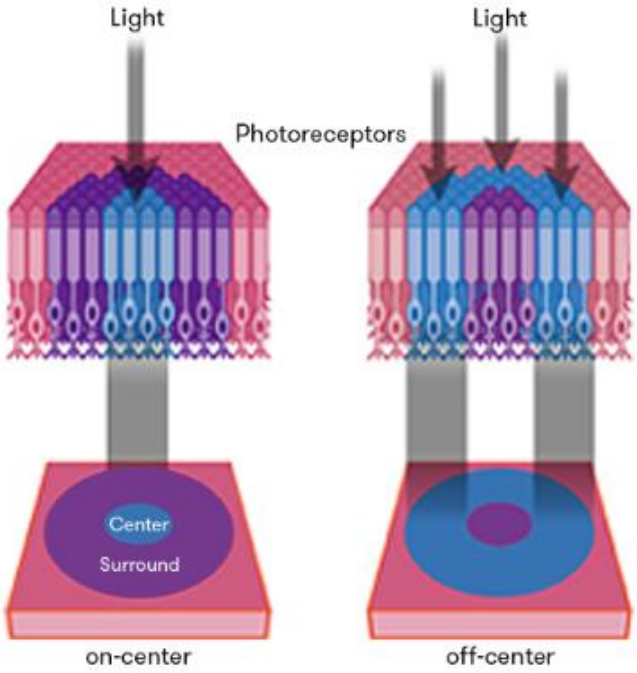
2

Hubel-Weisel实验/神经认知机

感受野

❑ Receptive field:

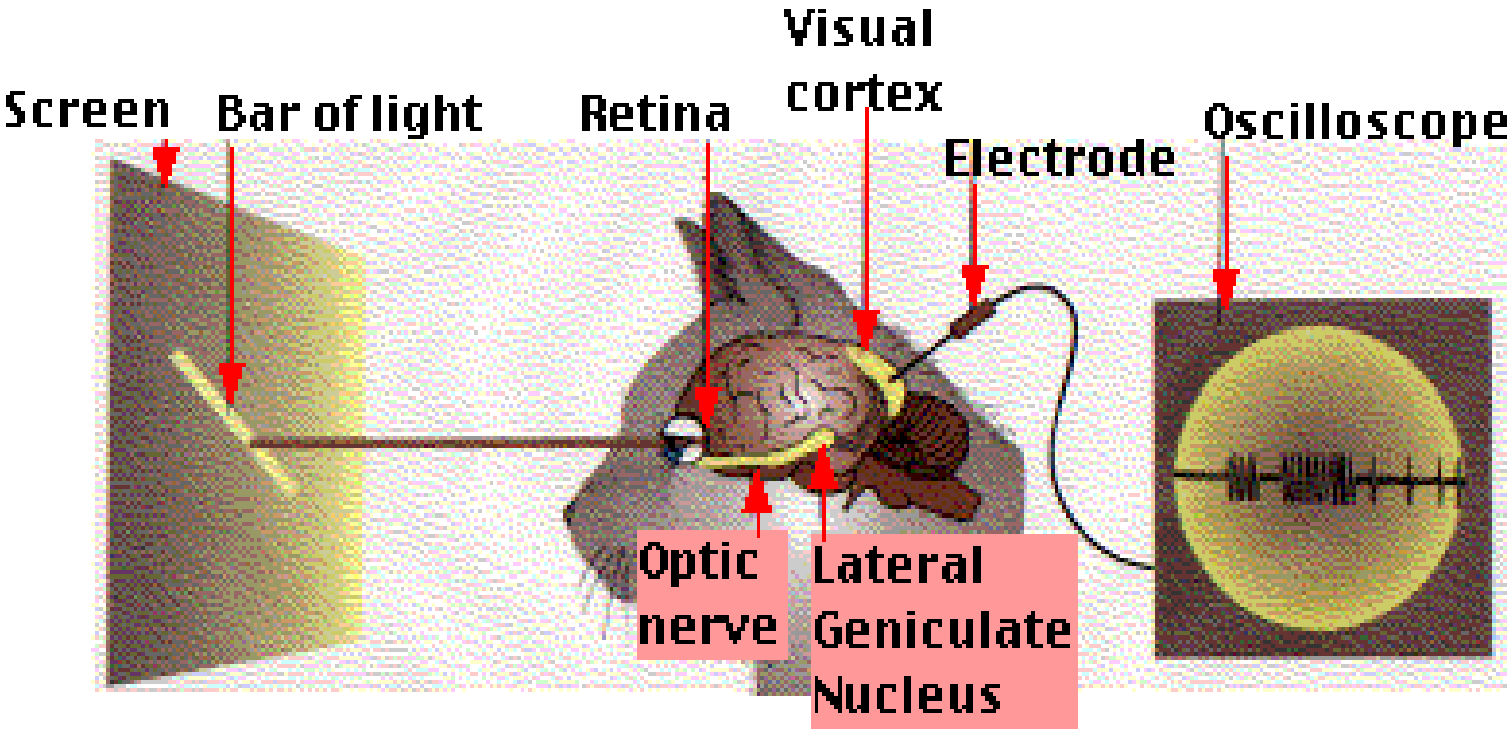
- The receptive field is a portion of sensory space that can elicit neuronal responses when stimulated
- 感受野是感觉空间的一部分，当它受到刺激时可以引起神经元的反应



Receptive field = center + surround

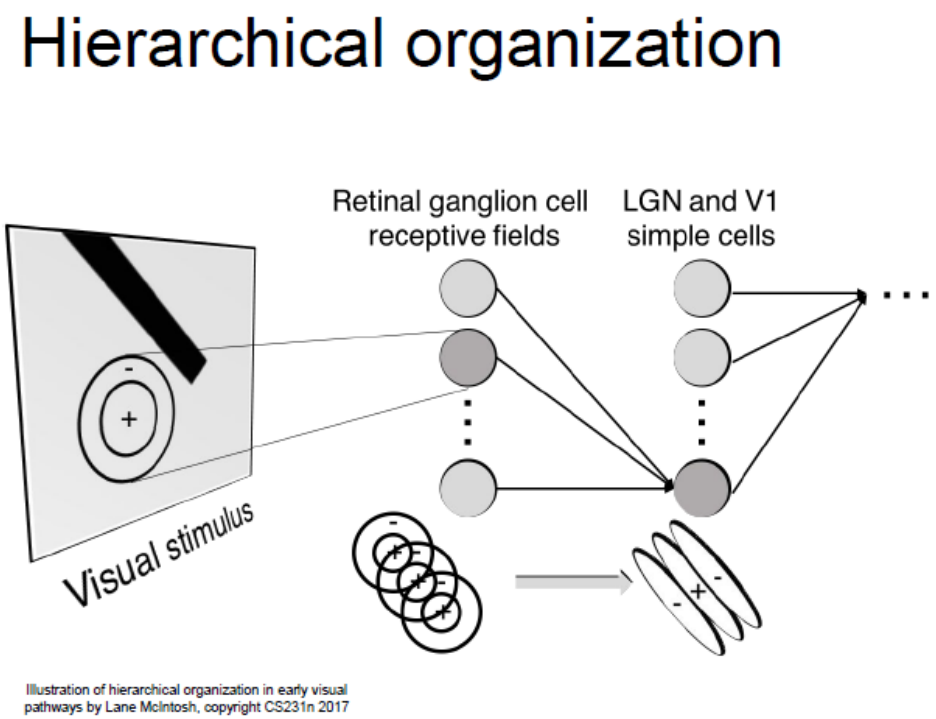
Hubel-Weisel实验 (1959年)

- 猫的纹状皮层中单个神经元的感受野



Hubel-Weisel实验 (1959年)

❑ 视觉神经元细胞的不同响应

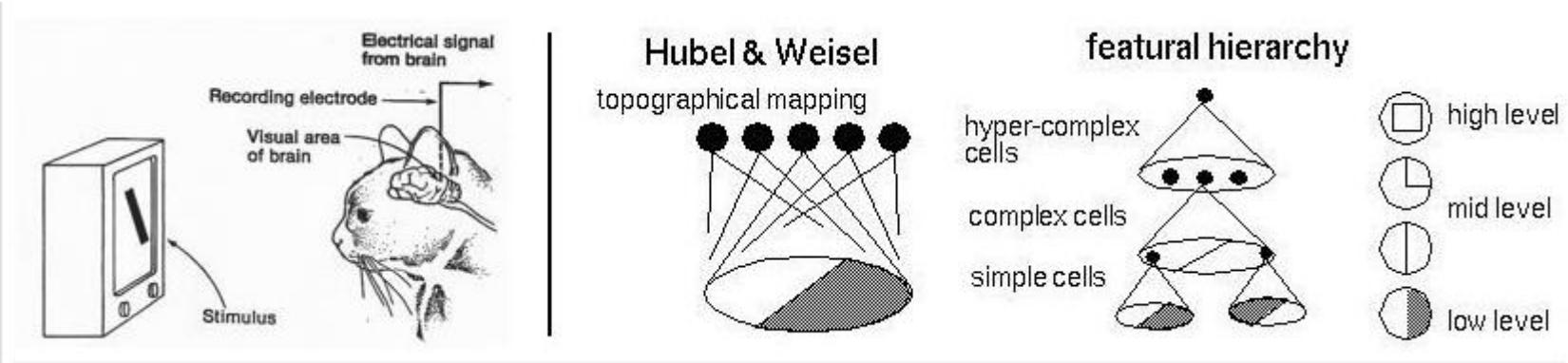


Simple cells:
Response to light orientation

Complex cells:
Response to light orientation and movement

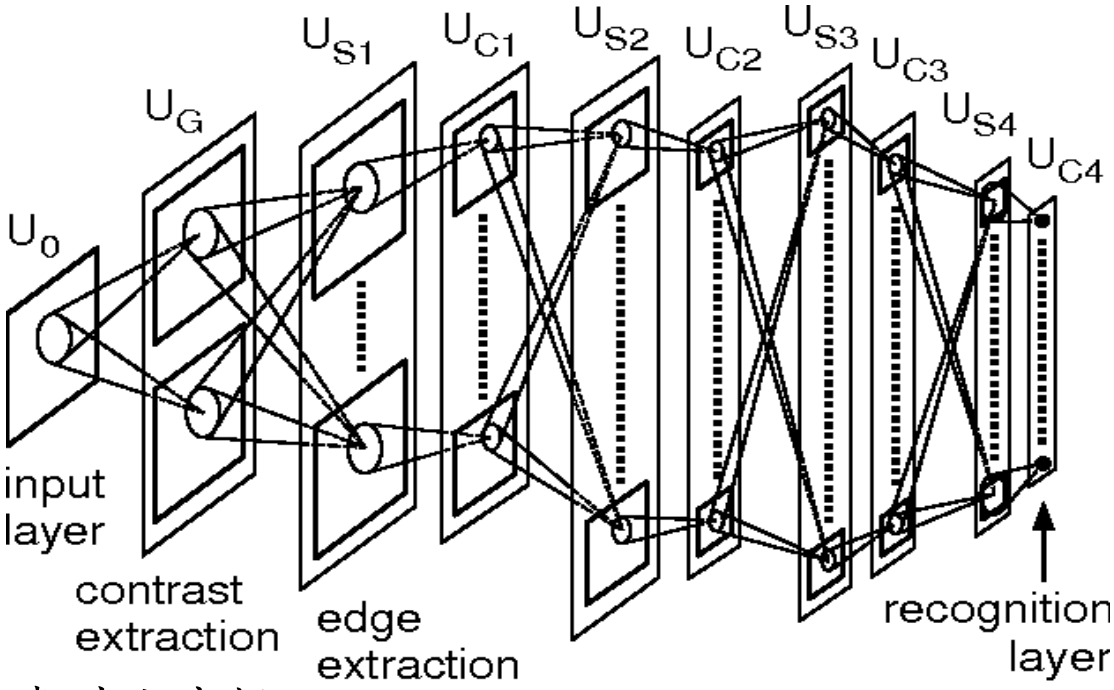
Hypercomplex cells:
response to movement with an end point

Hubel-Weisel实验 (1959年)



神经认知机 (1980年)

❑ 日本福岛邦彦提出的一种假设的生物视觉系统的数学模型



- G层：负责对比度提取
- 中间S层（简单细胞层）：负责图像特征提取，如边缘
- 中间C层（复杂细胞层）：抗变形，畸变容错
- 最后C层：输出识别结果

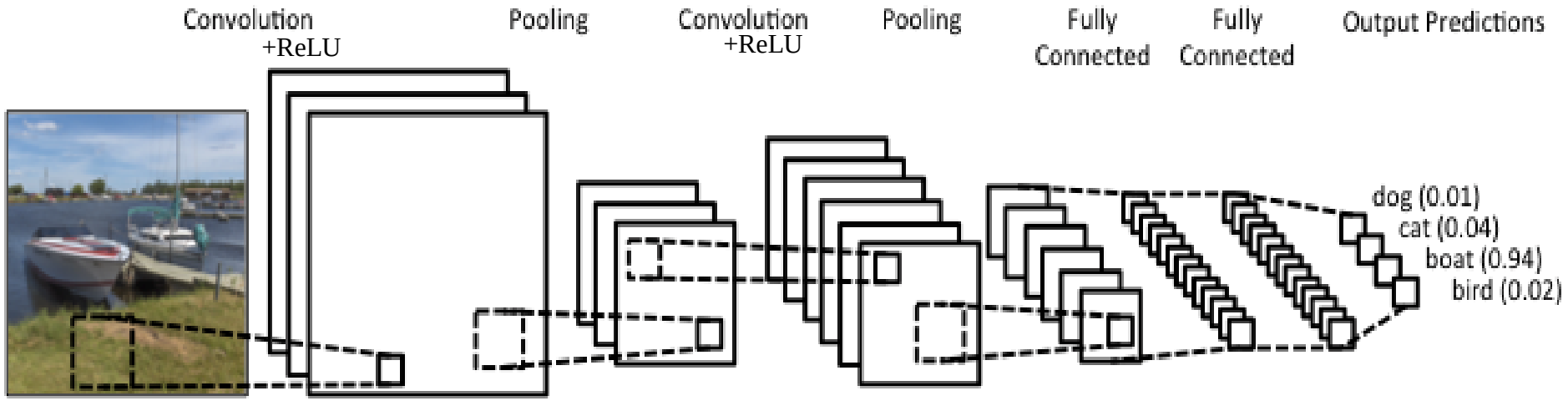
3

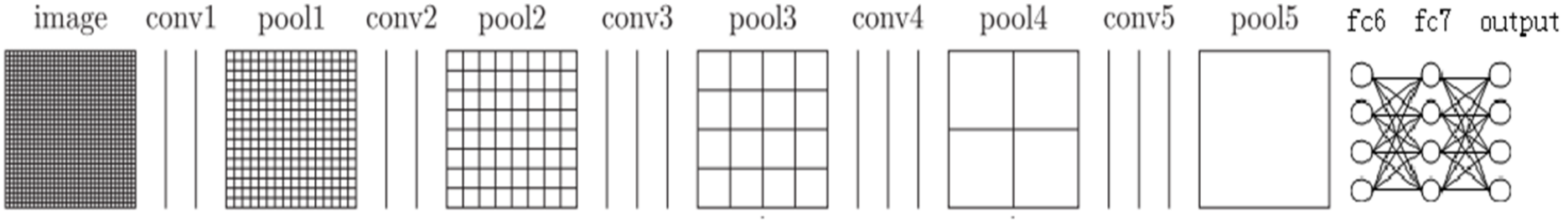
卷积神经网络基本原理

卷积神经网络的基本结构

- 卷积层 (Convolution)
- 激活函数(Active Function)
- 池化层 (Pooling)
- 全连接层 (Full-connected)
- 输出层（Output）

卷积神经网络的基本结构





卷积层

❑ 二维卷积运算

– 给定二维的图像I作为输入，二维卷积核K，卷积运算可表示为

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

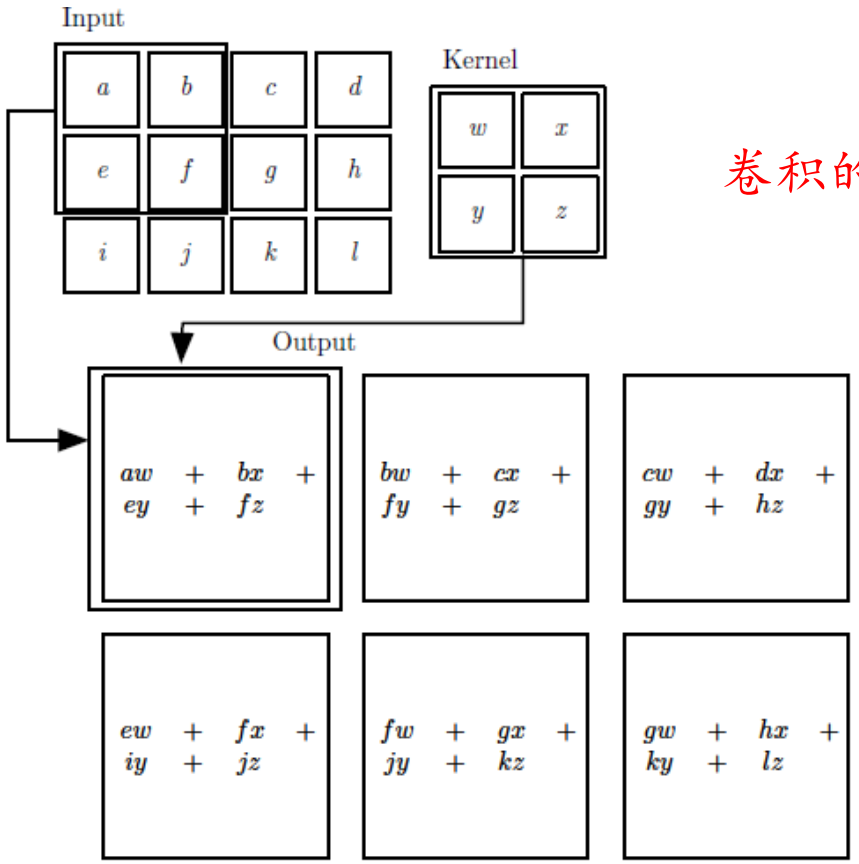
卷积核需要进行上下翻转和左右反转

$$S(i,j) = sum(\begin{bmatrix} I(i-2,j-2) & I(i-2,j-1) & I(i-2,j) \\ I(i-1,j-2) & I(i-1,j-1) & I(i-1,j) \\ I(i,j-2) & I(i,j-1) & I(i,j) \end{bmatrix} .* \begin{bmatrix} K(2,2) & K(2,1) & K(2,0) \\ K(1,2) & K(1,1) & K(1,0) \\ K(0,2) & K(0,1) & K(0,0) \end{bmatrix})$$

卷积层

❑ 卷积神经网络中的“卷积”

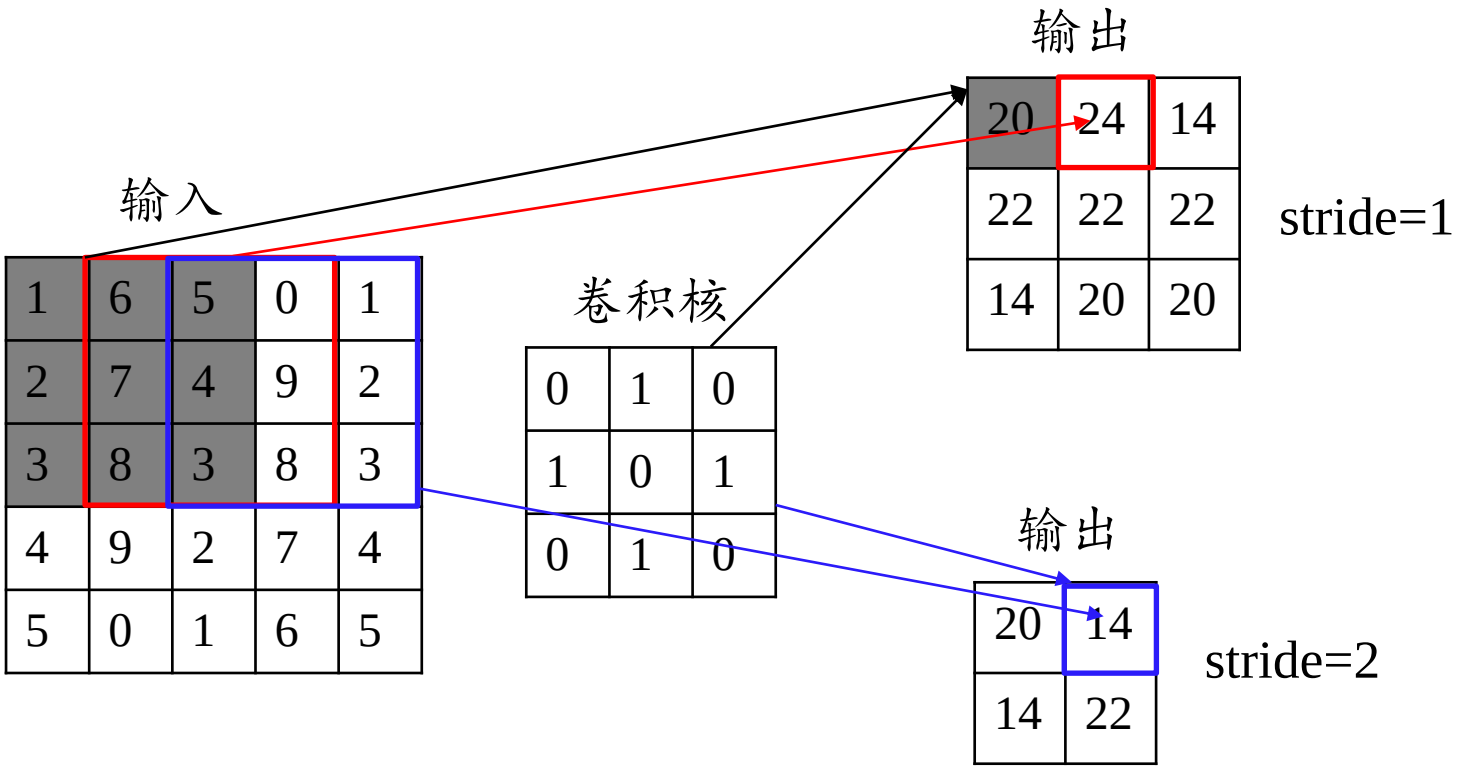
– 实际是互相关 $S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$



卷积的步长(stride): 卷积核移动的步长

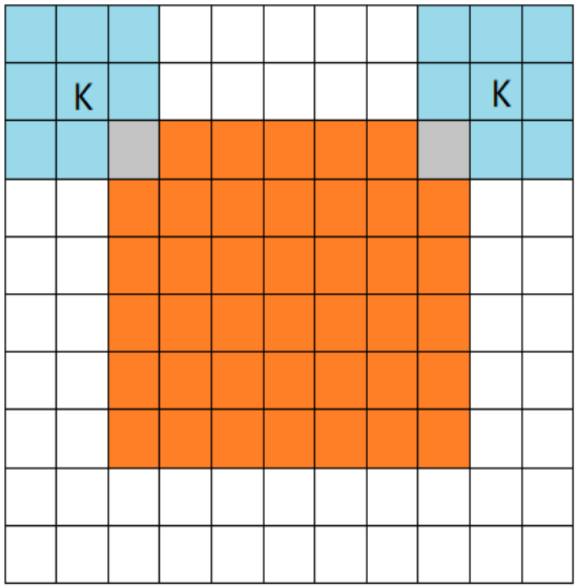
卷积层

❑ 不同步长示例

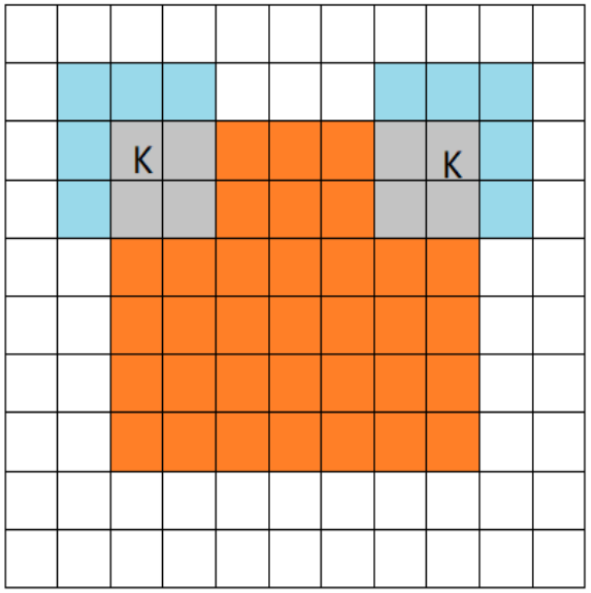

输出
20 24 14
22 22 22
14 20 20
stride=1
输入
1 6 5 0 1
2 7 4 9 2
3 8 3 8 3
4 9 2 7 4
5 0 1 6 5
卷积核
0 1 0
1 0 1
0 1 0
输出
20 14
14 22
stride=2

卷积层

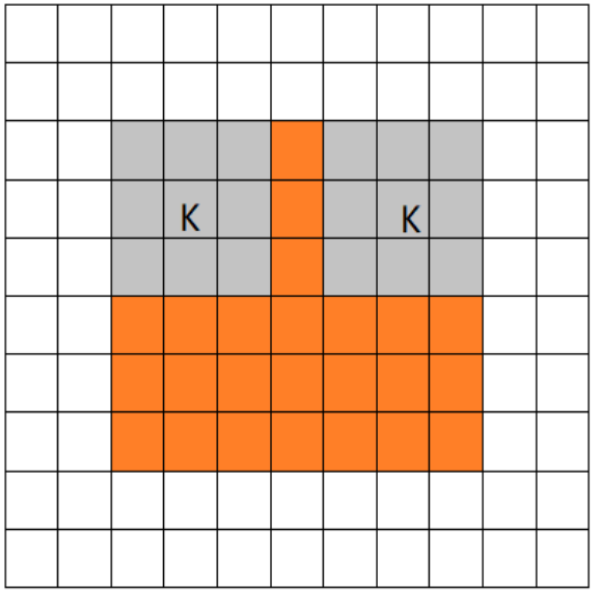
❑ 卷积的模式： **Full, Same和Valid**



Full



Same



Valid

卷积层

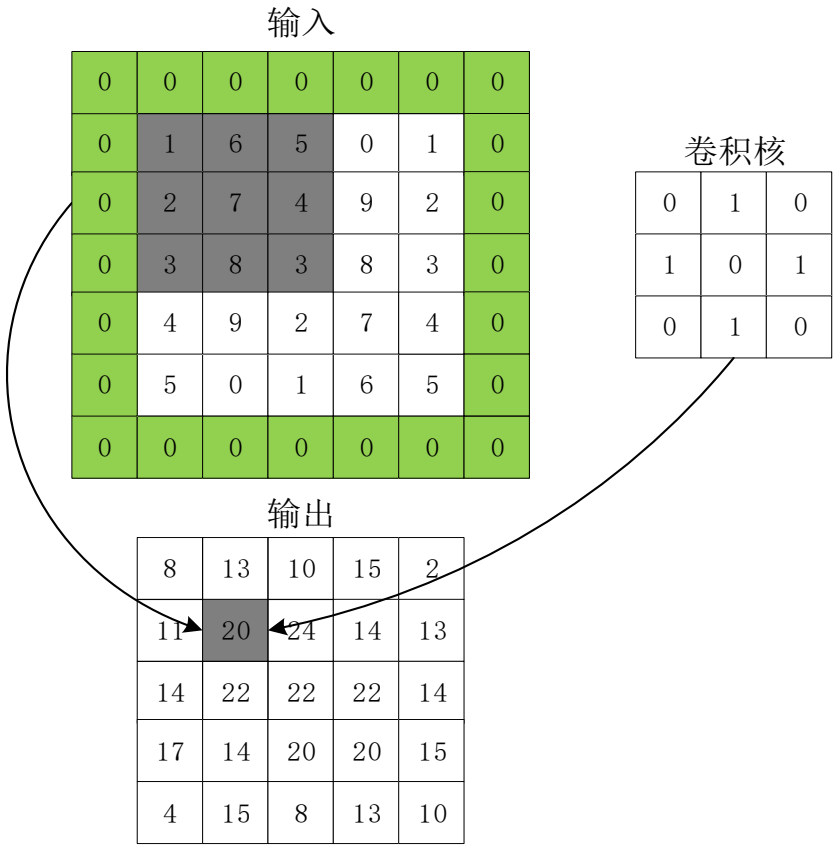
❑ 数据填充

– 如果我们有一个$n \times n$的图像，使用$f \times f$的卷积核进行卷积操作，在进行卷积操作之前我们在图像周围填充p层数据，输出的维度：

$$(n-f+1) \times (n-f+1)$$

$$\downarrow$$

$$(n+2p-f+1) \times (n+2p-f+1)$$

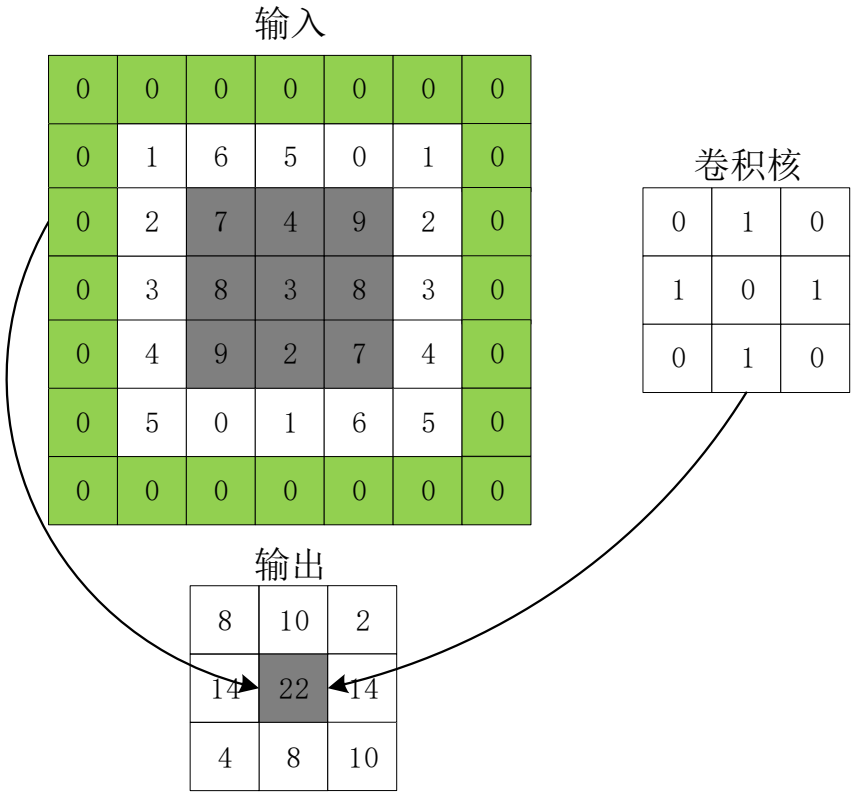


卷积层

❑ 卷积输出维度

– 如果我们有一个$n \times n$的图像，使用$f \times f$的卷积核进行卷积操作，在进行卷积操作之前我们在图像周围填充p层数据，步幅为s。输出的维度为

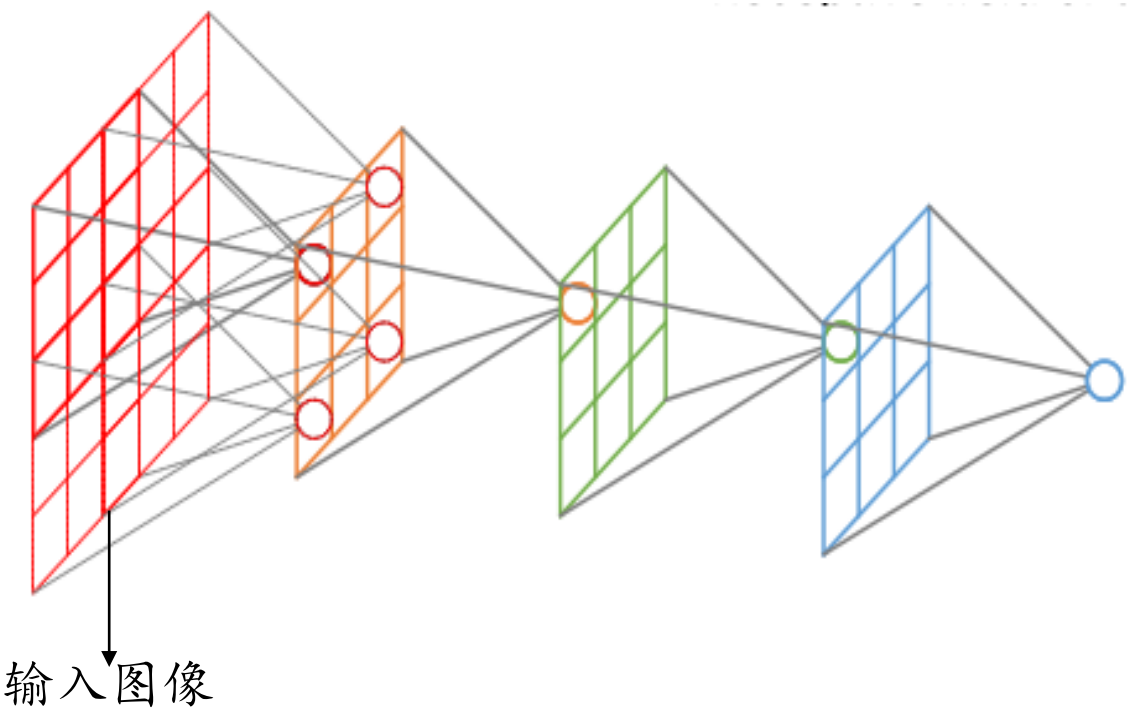
$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$



卷积神经网络的感受野

❑ 感受野

– 卷积神经网络每一层输出的特征图（feature map）上的像素点在输入图片上映射的区域大小，即特征图上的一个点对应输入图上的区域



卷积神经网络的感受野

❑ 感受野的计算

– 可以采用从后往前逐层计算

- 第i层的感受野大小和第i-1层的卷积核大小和步长有关系，同时也与第（i-1）层感受野大小有关
- 假设最后一层(卷积层或池化层)输出特征图感受野的大小(相对于其直接输入而言)等于卷积核的大小

$$RF_i = (RF_{i+1} - 1) \times s_i + K_i$$

第i层步长stride

第i层卷积核大小
Kernel Size

卷积神经网络的感受野

❑ 感受野的计算

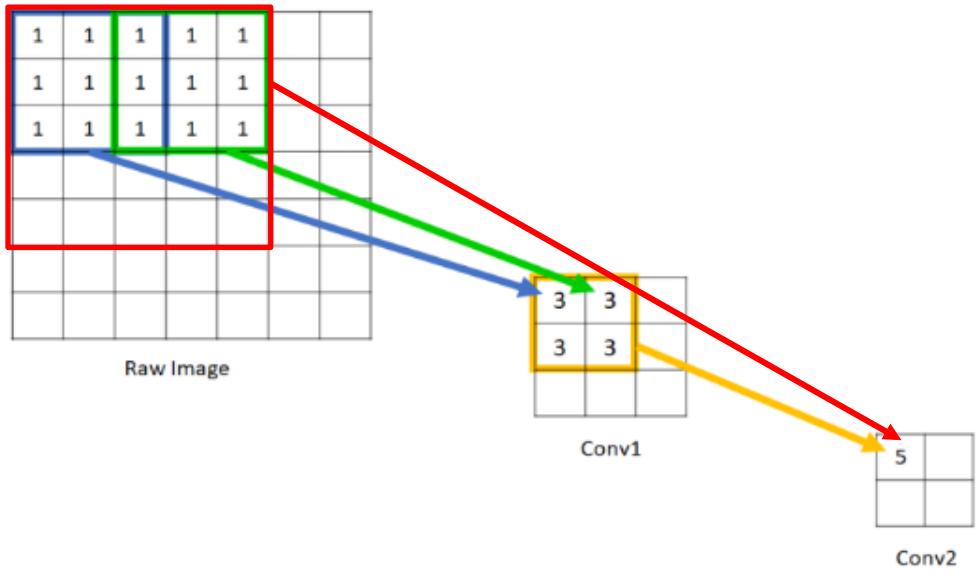
– 例如2个卷积层，第1层3x3卷积核，stride是2，第2层2x2卷积核

- RF_2=2
- RF_1=(2-1)*2+3=5;即输出在输入图片上的感受野为5x5

$$RF_i = (RF_{i+1} - 1) \times s_i + K_i$$

第i层步长stride

第i层卷积核大小
Kernel Size



卷积神经网络的感受野

❑ 感受野的计算

– 例如稍微复杂点的网络

Layer Type	Kernel Size	Stride
Input		
Conv1	3x3	1
Pool1	2x2	2
Conv2	3x3	1
Pool2	2x2	2
Conv3	3x3	1
Conv4	3x3	1
Pool3	2x2	2
Output		

Pool3: RF=2

Conv4: RF=(2-1)*1+3=4

Conv3: RF=(4-1)*1+3=6

Pool2: RF=(6-1)*2+2=12

Conv2: RF=(12-1)*1+3=14

Pool1: RF=(14-1)*2+2=28

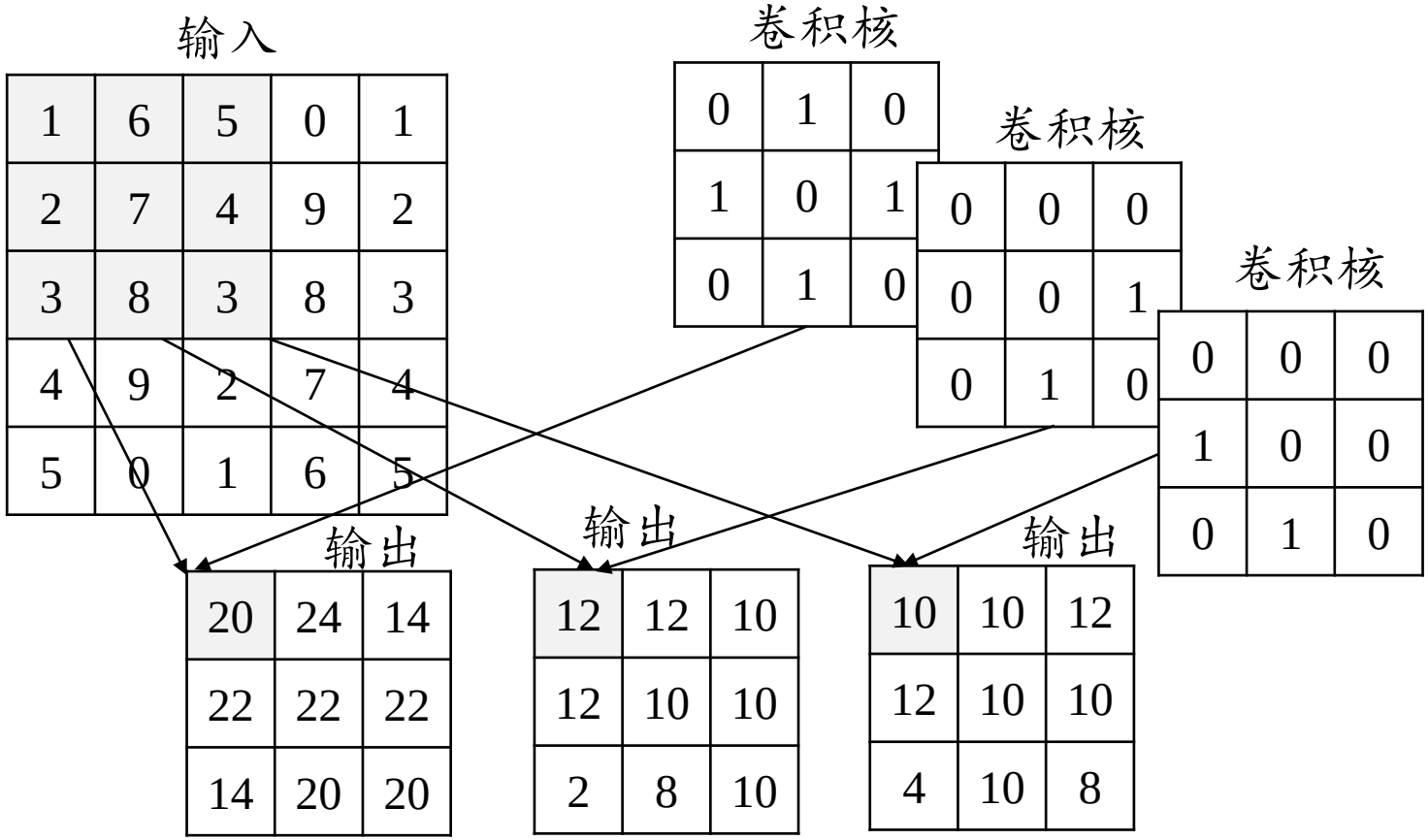
Conv1: RF=(28-1)*1+3=30

Pool3输出的特征图在输入图片上的感受野为30x30

卷积层

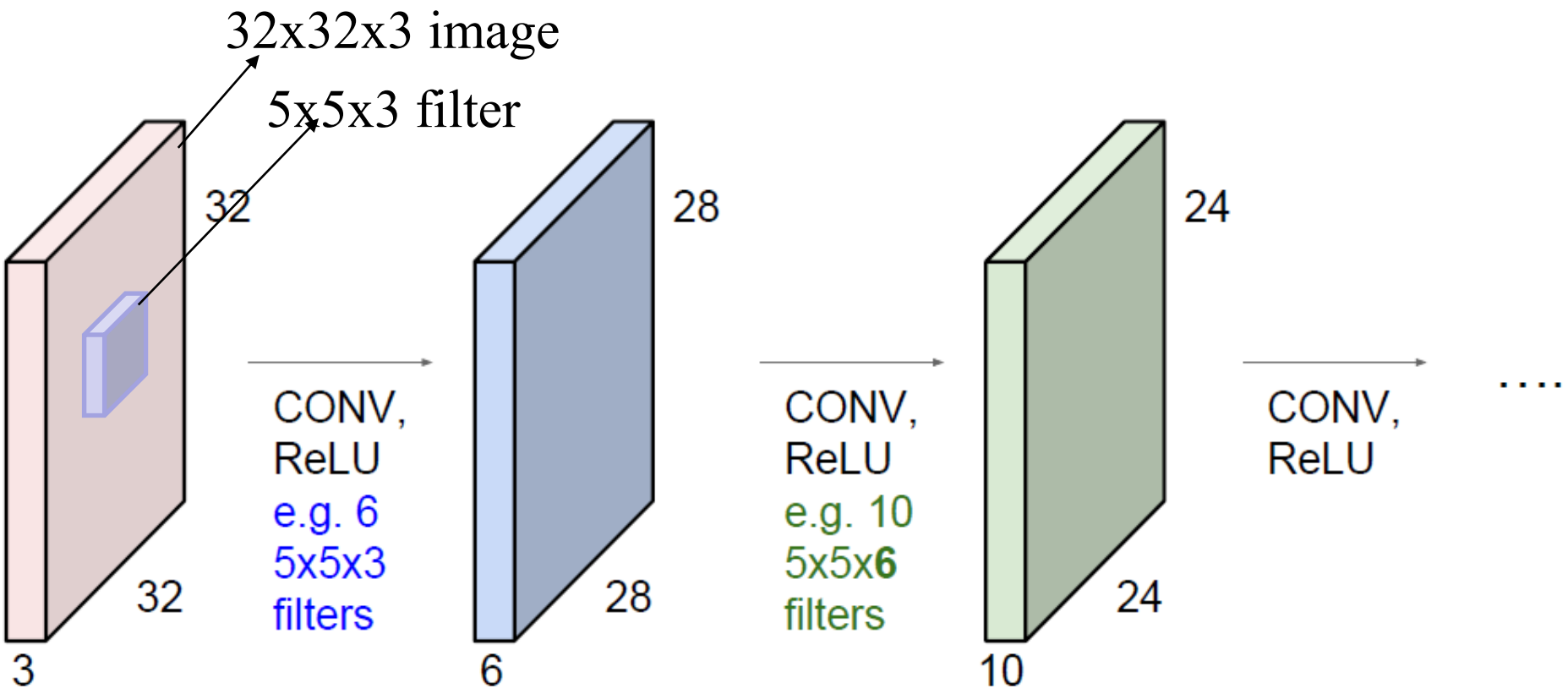
❑ 卷积层的深度(卷积核个数)

– 一个卷积层通常包含多个尺寸一致的卷积核



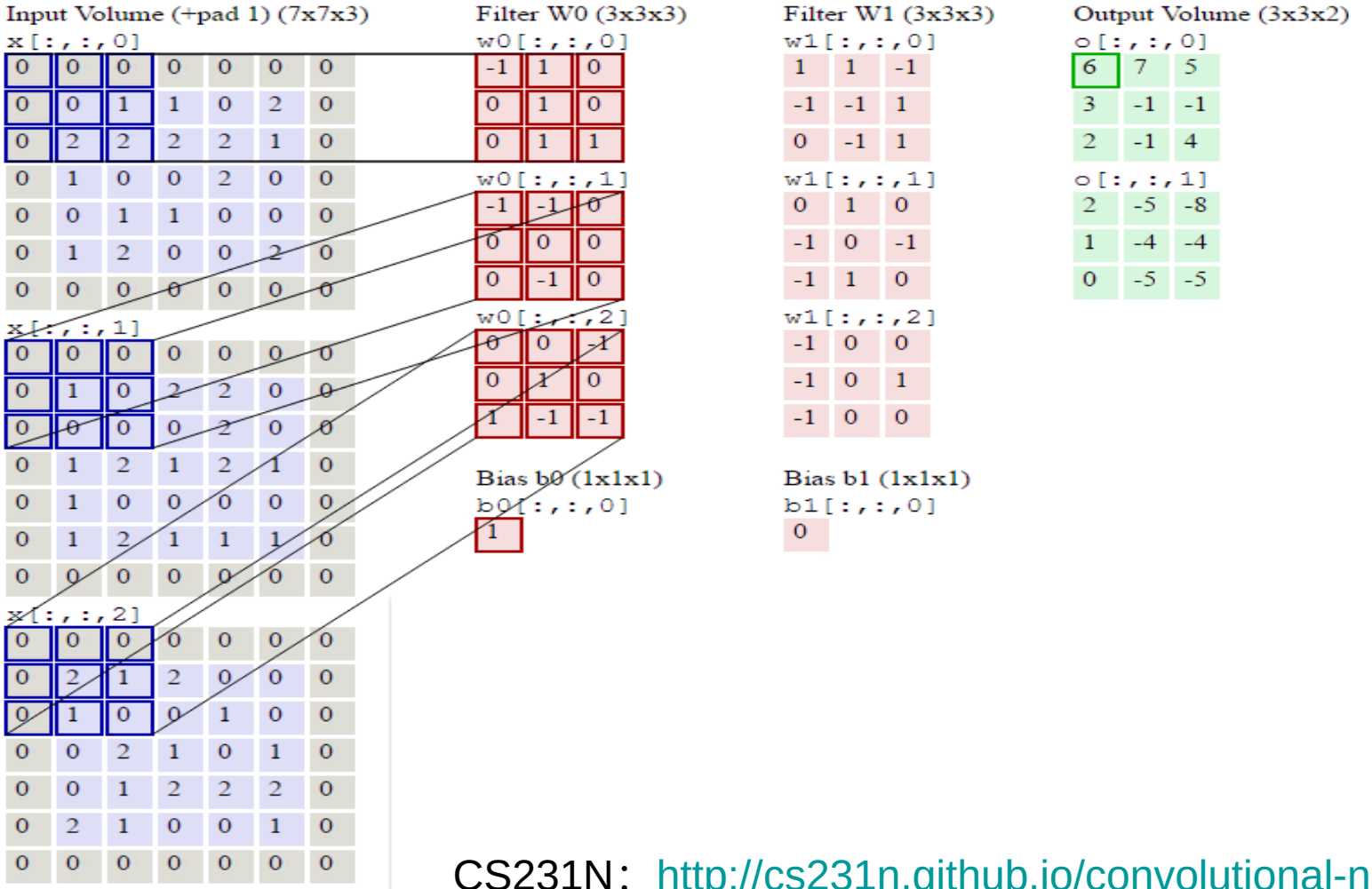
卷积层

- 卷积层的深度(卷积核个数)



卷积层

❑ 卷积过程



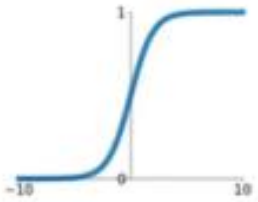
CS231N：http://cs231n.github.io/convolutional-networks/

激活函数

- 激活函数是用来加入非线性因素，提高网络表达能力
  - 卷积神经网络中最常用的是ReLU，Sigmoid使用较少

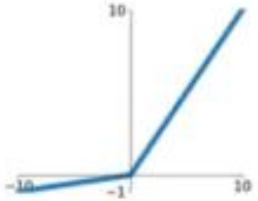
Sigmoid

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



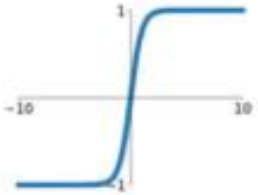
Leaky ReLU

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



tanh

$$\tanh(x)$$

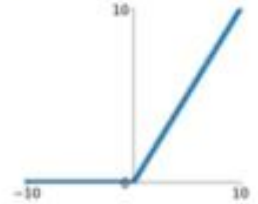


Maxout

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

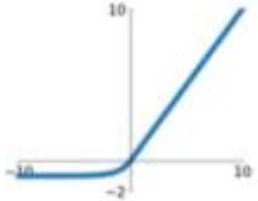
ReLU

$$\max(0, x)$$



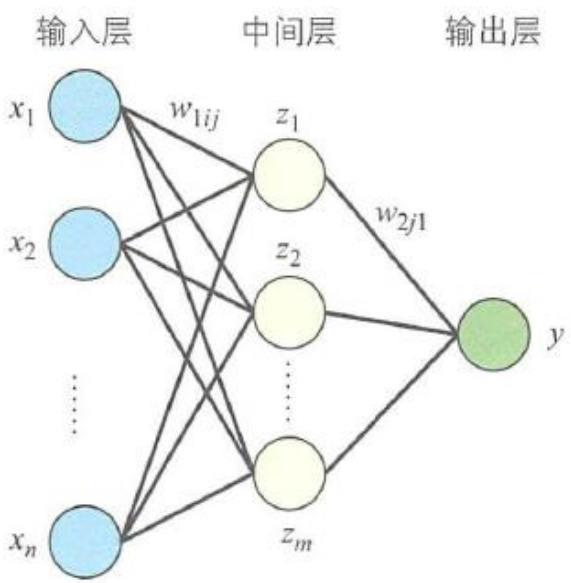
ELU

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



激活函数

- 激活函数是用来加入非线性因素，提高网络表达能力
  - 卷积神经网络中最常用的是ReLU，Sigmoid使用较少
  - 梯度消失



$$\frac{\partial E}{\partial w_{1ij}} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial u_{21}}\frac{\partial u_{21}}{\partial w_{1ij}}$$

$$\frac{\partial u_{21}}{\partial w_{1ij}} = \frac{\partial u_{21}}{\partial z_j}\frac{\partial z_j}{\partial w_{1ij}}$$

$$\frac{\partial z_j}{\partial w_{1ij}} = \frac{\partial z_j}{\partial u_{1j}}\frac{\partial u_{1j}}{\partial w_{1ij}}$$

$$\frac{\partial E}{\partial w_{1ij}} = -(r-y)\,y(1-y)\,w_{2j1}\,z_j(1-z_j)\,x_i$$

函数的导数　激活函数的导数　连接权重的导数　激活函数的导数　输入值

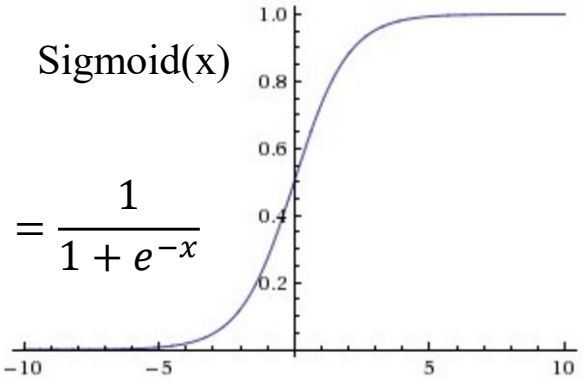
中间层与输出层之间　输入层与中间层之间

激活函数

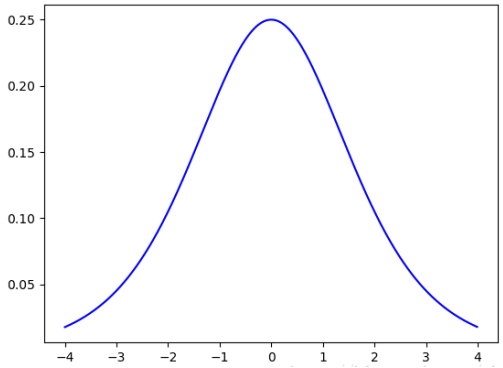
- 激活函数是用来加入非线性因素，提高网络表达能力
  - 卷积神经网络中最常用的是ReLU，Sigmoid使用较少
  - 梯度消失

Sigmoid(x)

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



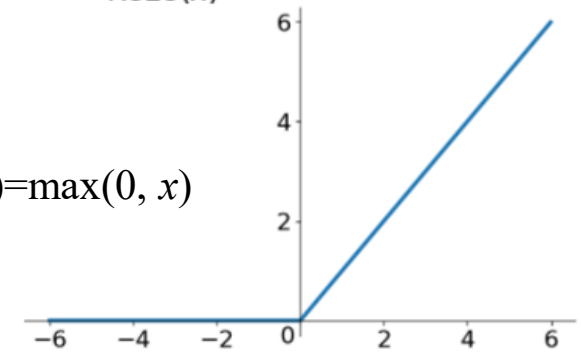
dSigmoid(x)/dx



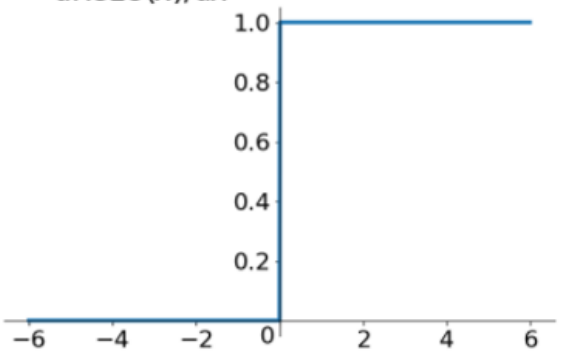
$$\frac{e^{-x}}{(1+e^{-x})^2}$$

ReLU(x)

ReLU(x)=max(0, x)



dReLU(x)/dx



激活函数

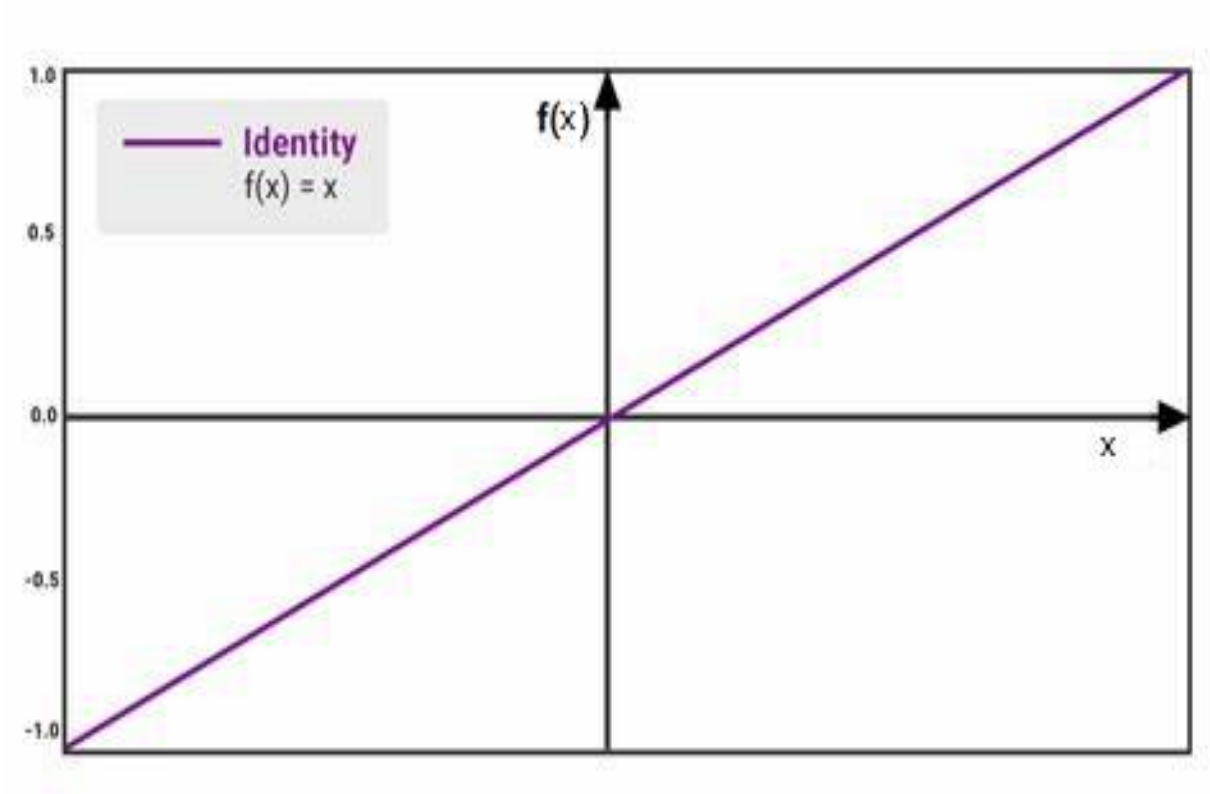
❑ 常见的激活函数总结

- Identity： $f(x) = x,\ x \in (-\infty, +\infty)$
- Rectified Linear Unit (ReLU) ： $f(x) = \begin{cases} 0, x < 0 \\ x, x \geq 0 \end{cases}, x \in (-\infty, +\infty)$
- Parametric ReLU (PReLU) ： $f(x) = \begin{cases} \alpha x, x < 0 \\ x, x \geq 0 \end{cases}, x \in (-\infty, +\infty)$
- Exponential Linear Unit (ELU)： $f(x) = \begin{cases} \alpha(e^x - 1), x < 0 \\ x, x \geq 0 \end{cases}, x \in (-\infty, +\infty)$
- Maxout：$\max(w_1^T x + b_1, w_2^T x + b_2, \ldots, w_n^T x + b_n)$

激活函数

❑ Identity

– 适合线性任务，相对稳定，但是对卷积输出没有影响



激活函数

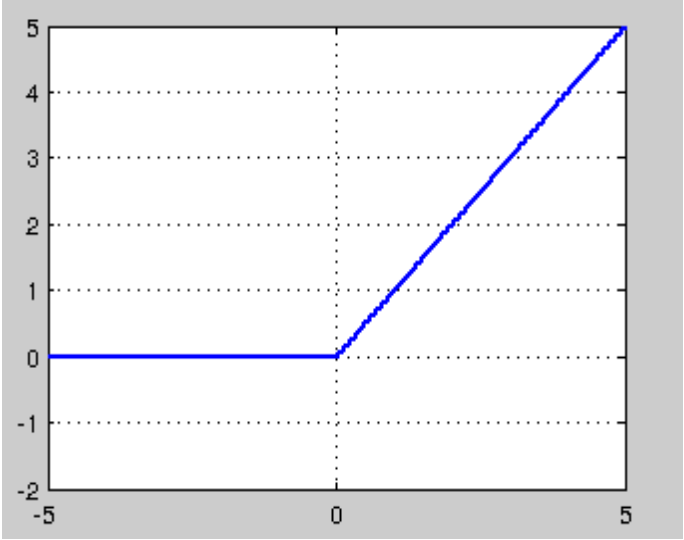
❑ ReLU函数的优点：

- 计算速度快，ReLU函数只有线性关系，比Sigmoid和Tanh要快很多
- 输入为正数的时候，不存在梯度消失问题

❑ ReLU函数的缺点：

- 强制性把负值置为0，可能丢掉一些特征
- 当输入为负数时，权重无法更新，导致“神经元死亡”（学习率不要太大）

$$f(x)=\begin{cases}0, x<0\\ x, x\geq 0\end{cases},$$

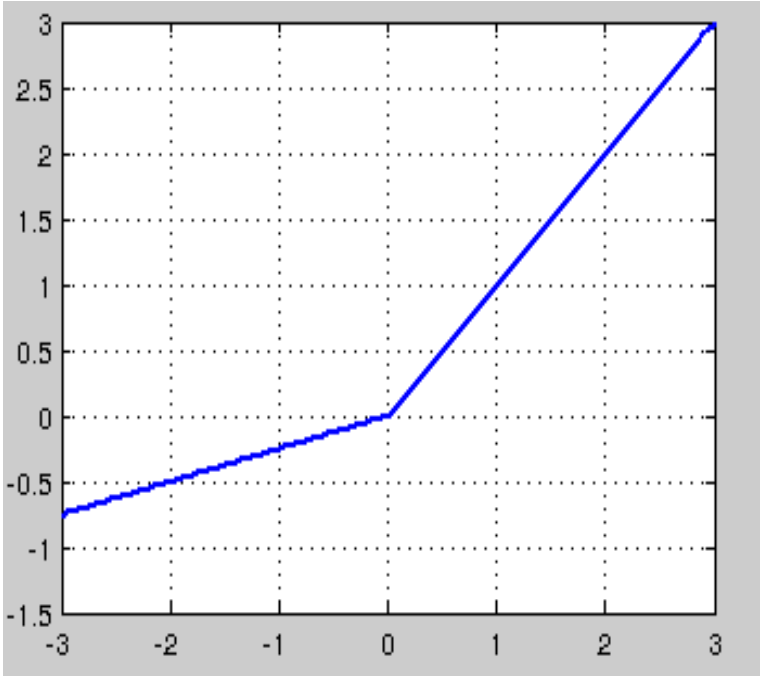


激活函数

- **Parametric ReLU (PReLU)**
  - 当α=0.01时，称作Leaky ReLU
  - 当α从高斯分布中随机产生时，称为Randomized ReLU（RReLU）

$$f(x)=\begin{cases}\alpha x, x<0\\ x, x\geq 0\end{cases}$$

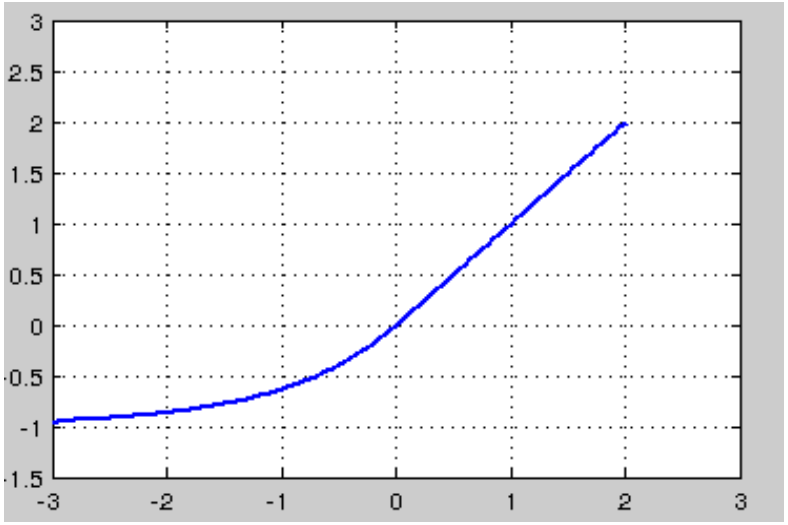
- **PReLU函数的优点：**
  - 比sigmoid/tanh收敛快
  - 解决了ReLU的“神经元死亡”问题
- **PReLU函数的缺点：**
  - 需要再学习一个参数，工作量变大



激活函数

- **ELU函数的优点：**
  - 处理含有噪声的数据有优势
  - 更容易收敛
- **ELU函数的缺点：**
  - 计算量较大，收敛速度较慢

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

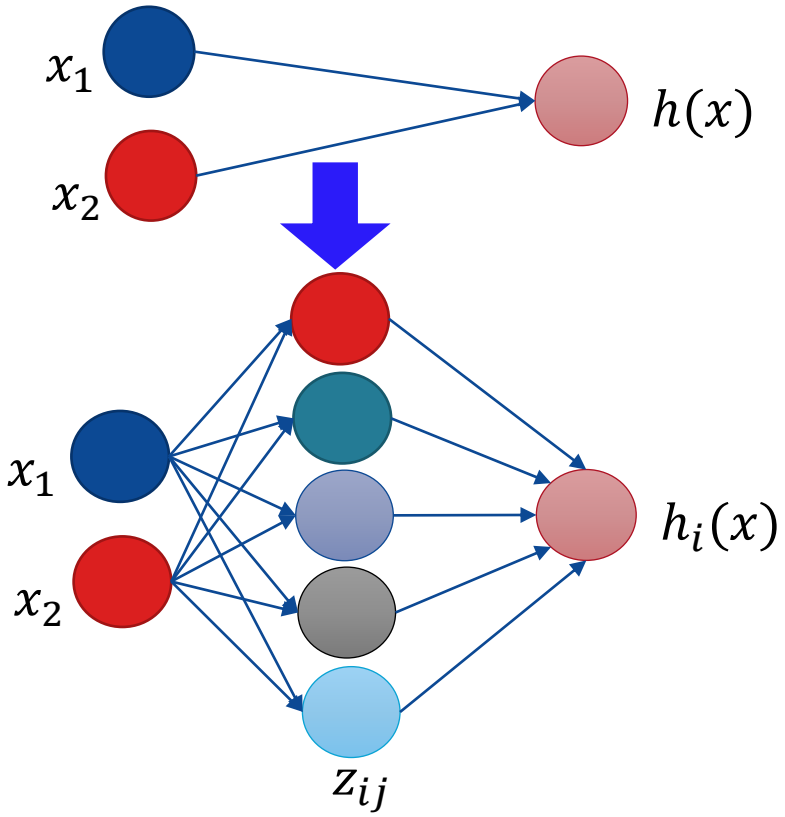


激活函数

- **Maxout函数增加一层神经网络，神经元个数n自定**

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

- **Maxout函数的优点：**
  - 能够缓解梯度消失
  - 规避了ReLU神经元死亡的缺点
- **Maxout函数的缺点：**
  - 增加了参数和计算量



激活函数

- CNN在卷积层尽量不要使用Sigmoid和Tanh，将导致梯度消失。
- 首先选用ReLU，使用较小的学习率，以免造成神经元死亡的情况。
- 如果ReLU失效，考虑使用Leaky ReLU、PReLU、ELU或者Maxout，此时一般情况都可以解决

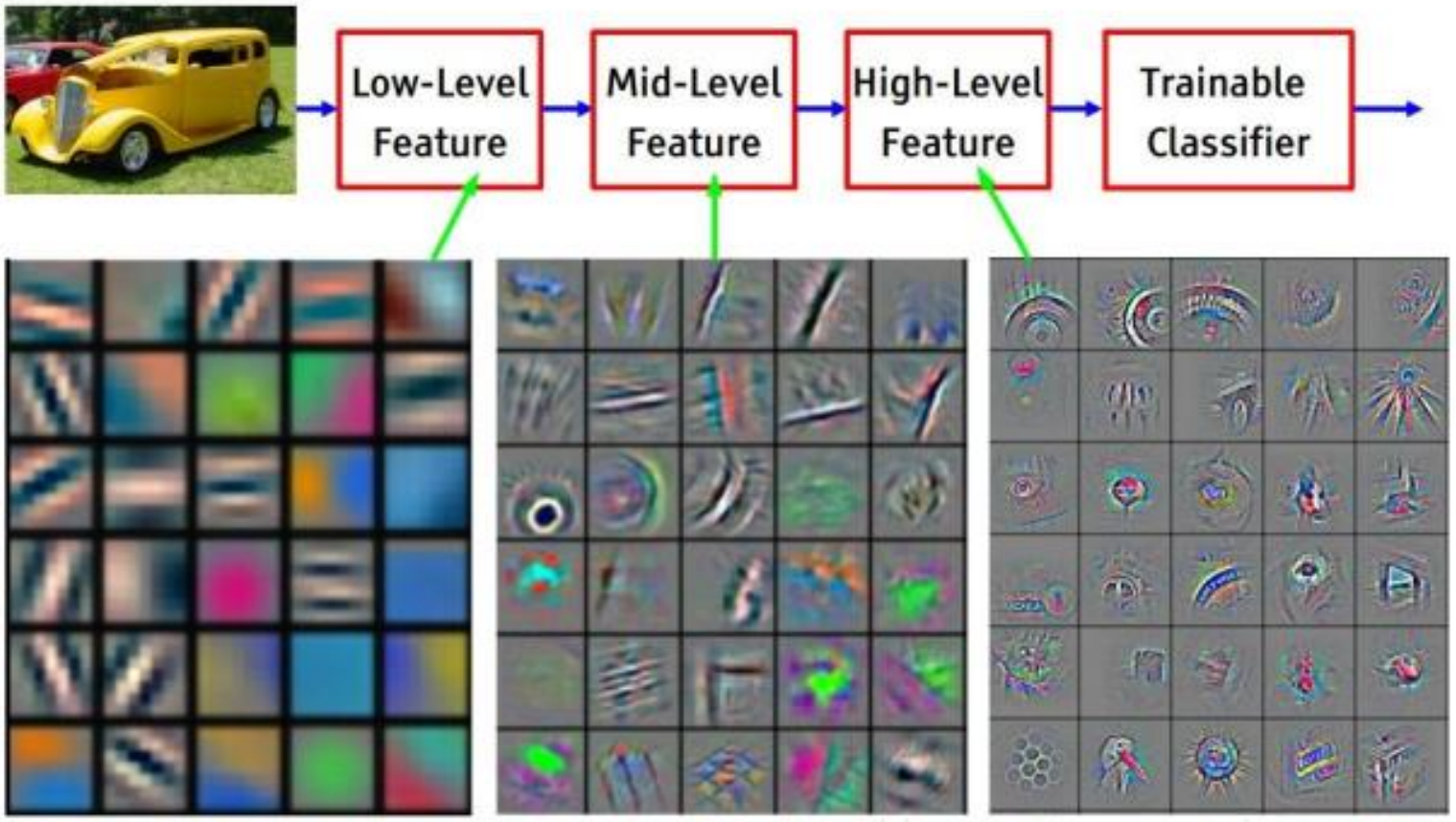
卷积层(激活函数)

❑ 特征图

- 浅层卷积层：提取的是图像基本特征，如边缘、方向和纹理等特征
- 深层卷积层：提取的是图像高阶特征，出现了高层语义模式，如“车轮”、“人脸”等特征

卷积层(激活函数)

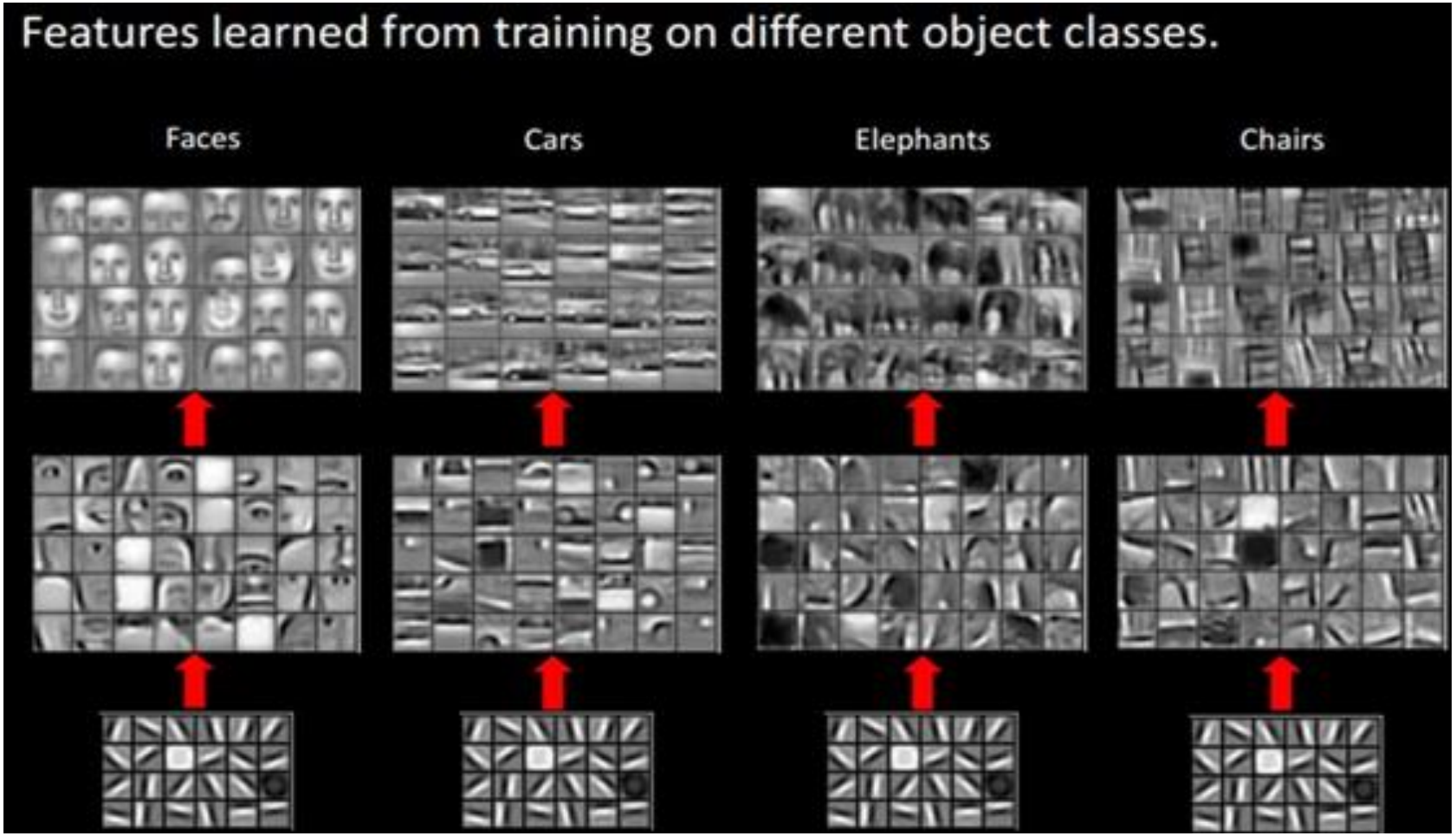
❑ 特征图



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

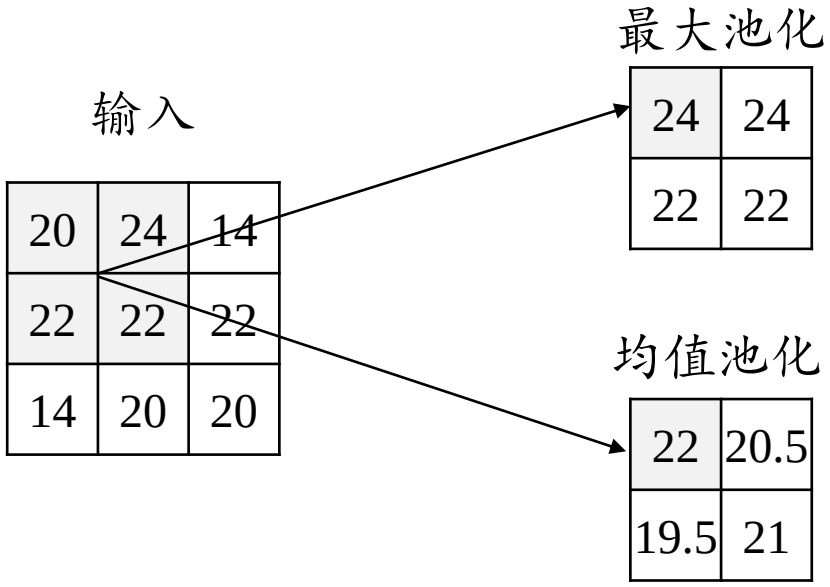
卷积层(激活函数)

- 特征图


Features learned from training on different object classes.
Faces
Cars
Elephants
Chairs

池化层

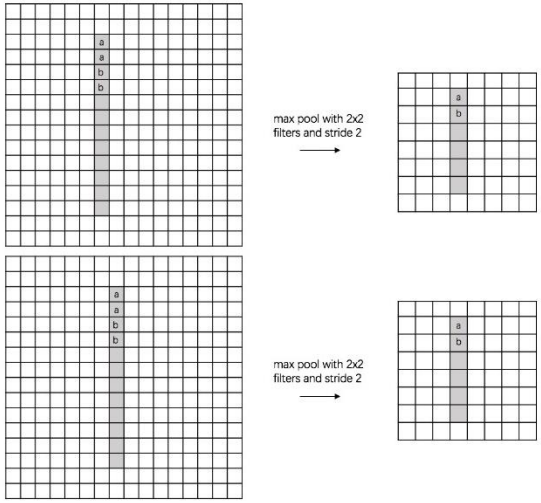
- 池化操作使用某位置相邻输出的总体统计特征作为该位置的输出，常用**最大池化(max-pooling)和均值池化(average-pooling)**
- 池化层不包含需要训练学习的参数，仅需指定池化操作的核大小、操作步幅以及池化类型



池化层

❑ 池化的作用

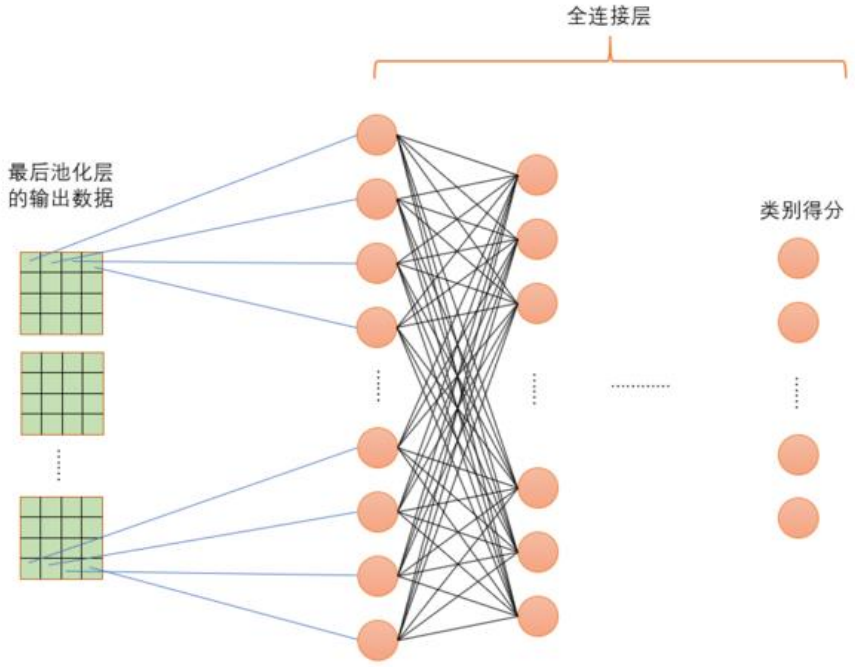
- 减少网络中的参数计算量，从而遏制过拟合
- 增强网络对输入图像中的小变形、扭曲、平移的鲁棒性(输入里的微小扭曲不会改变池化输出——因为我们在局部邻域已经取了最大值/平均值)
- 帮助我们获得不因尺寸而改变的等效图片表征。这非常有用，因为这样我们就可以探测到图片里的物体，不管它在哪个位置

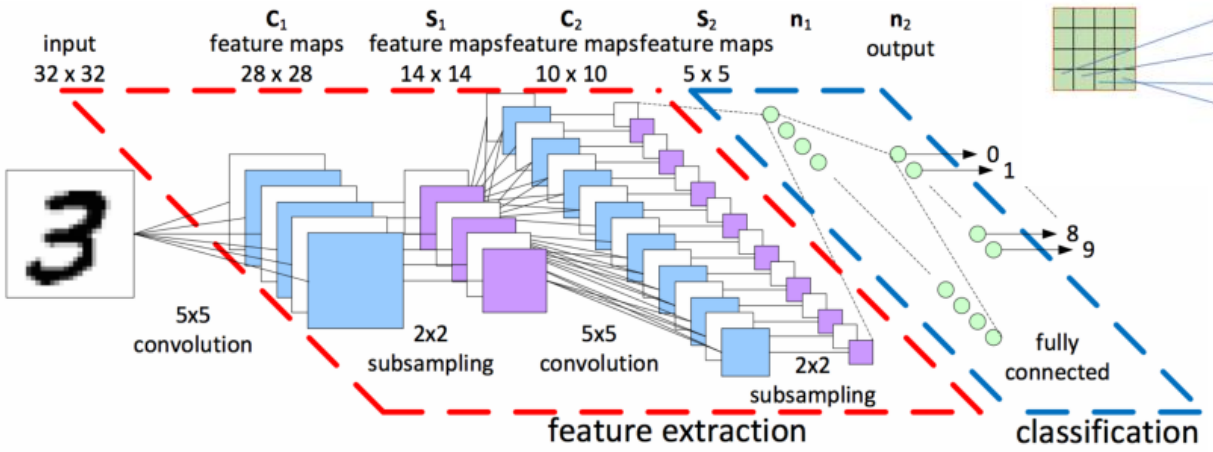


全连接层

❑ 全连接层

- 对卷积层和池化层输出的特征图（二维）进行降维
- 将学到的特征表示映射到样本标记空间的作用

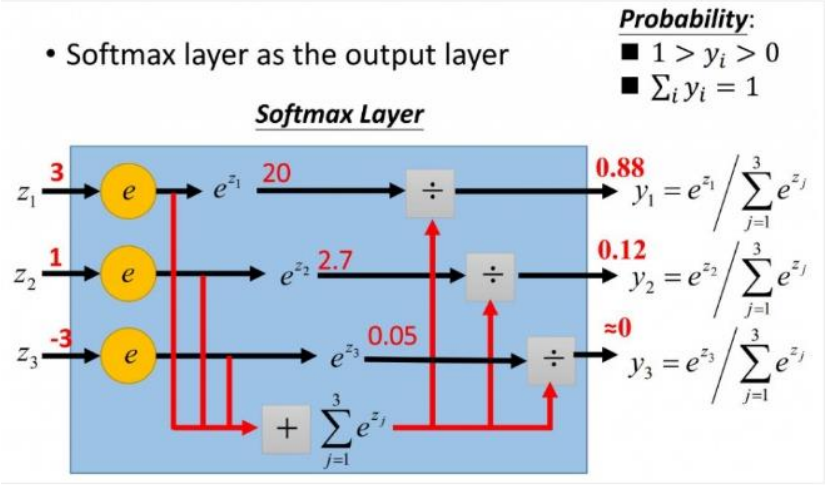




输出层

❑ 对于分类问题：使用Softmax函数

$$y_i = \frac{e^{z_i}}{\sum_{i=1}^{n} e^{z_i}}$$



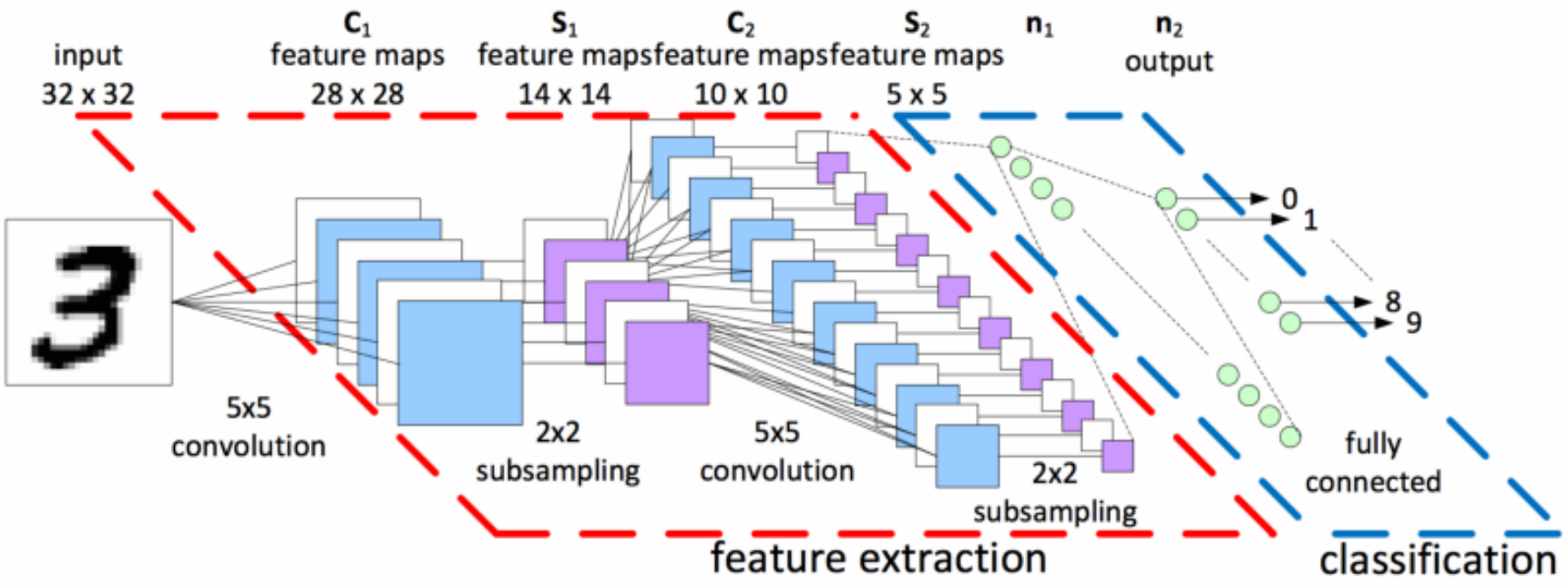
❑ 对于回归问题：使用线性函数

$$y_i = \sum_{m=1}^{M} w_{im} x_m$$

卷积神经网络的训练

- **Step 1**: 用随机数初始化所有的卷积核和参数/权重
- **Step 2**: 将训练图片作为输入，执行前向步骤（卷积，ReLU，池化以及全连接层的前向传播）并计算每个类别的对应输出概率。
- **Step 3**: 计算输出层的总误差：总误差=1/2 ∑ (目标概率–输出概率)^2
- **Step 4**: 反向传播算法计算误差相对于所有权重的梯度，并用梯度下降法更新所有的卷积核和参数/权重的值，以使输出误差最小化
- 卷积核个数、卷积核尺寸、网络架构这些参数，是在Step 1之前就已经固定的，且不会在训练过程中改变——只有卷积核矩阵和神经元权重会更新

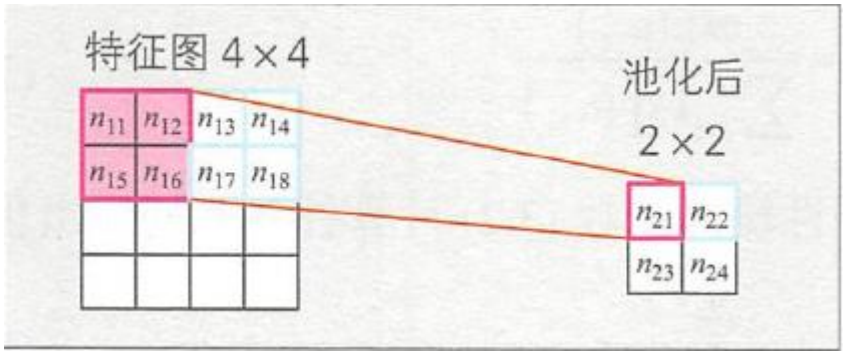
卷积神经网络的训练

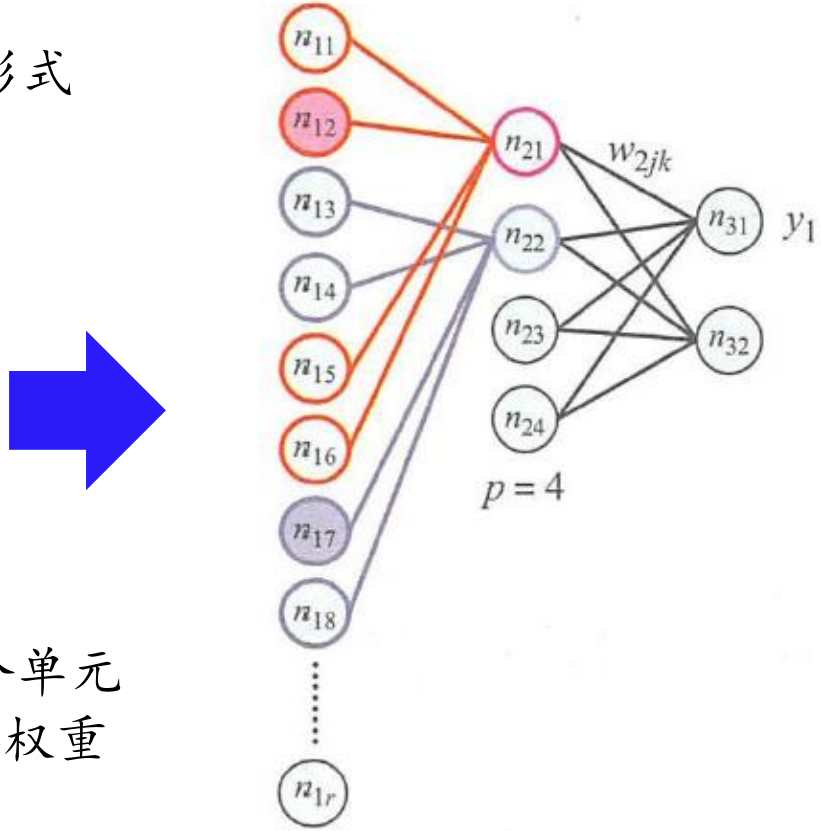

input
32 x 32
C1
feature maps
28 x 28
S1
feature maps
14 x 14
C2
feature maps
10 x 10
S2
feature maps
5 x 5
n1
n2
output
0
1
8
9
5x5
convolution
2x2
subsampling
5x5
convolution
2x2
subsampling
fully
connected
feature extraction
classification

卷积神经网络的训练

❑ 池化层的训练

- 和多层神经网络一样，卷积神经网络中的参数训练也是使用误差反向传播算法
- 将池化层改为多层神经网络的形式





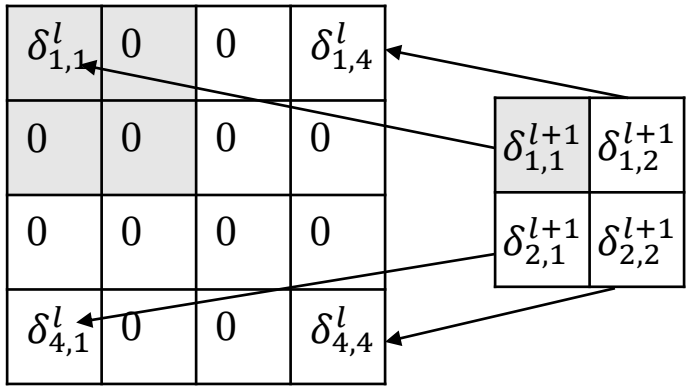
对2 × 2 区域应用最大池化时，与4个单元建立连接，激活值最大的单元的连接权重为1，其他单元的连接权重为0

卷积神经网络的训练

❑ 池化层的训练

- 和多层神经网络一样，卷积神经网络中的参数训练也是使用误差反向传播算法
- 将池化层改为多层神经网络的形式

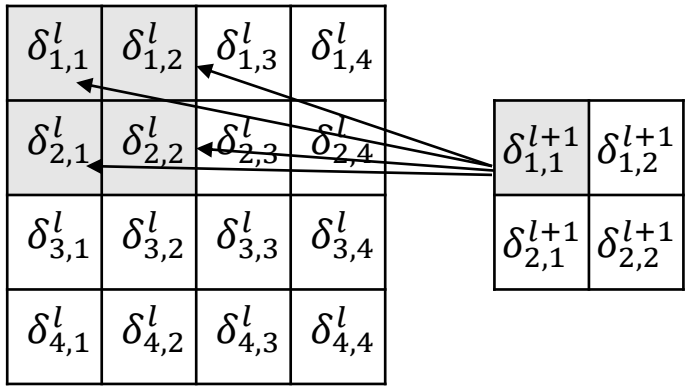
最大池化



输入层梯度　　输出层梯度

$\delta^l_{1,1}=\delta^{l+1}_{1,1}; \delta^l_{1,4}=\delta^{l+1}_{1,2}; \delta^l_{4,1}=\delta^{l+1}_{2,1}; \delta^l_{4,4}=\delta^{l+1}_{2,2};$

平均池化



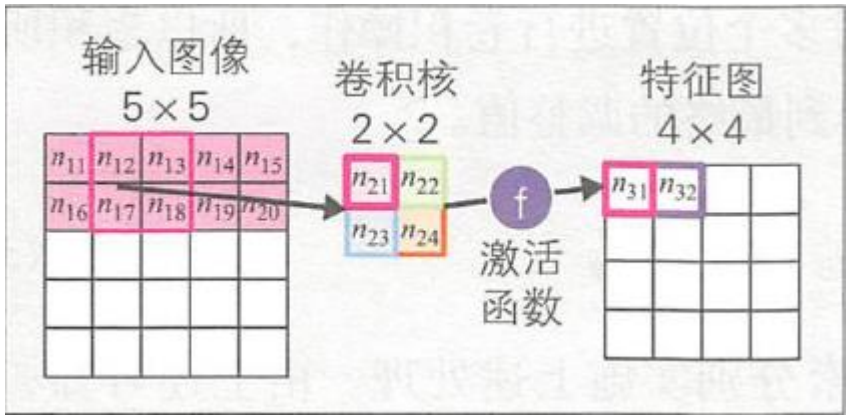
输入层梯度　　输出层梯度

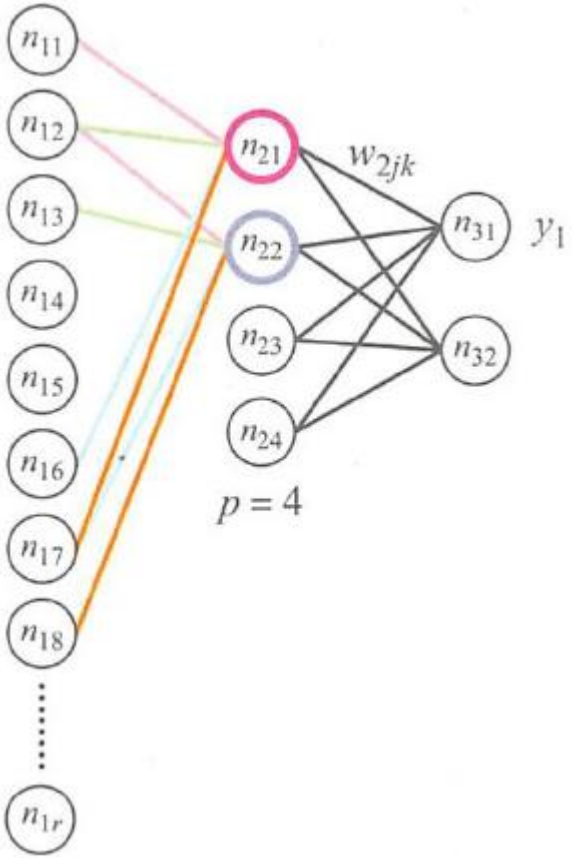
$\delta^l_{1,1}=\delta^{l+1}_{1,1}/4; \delta^l_{1,2}=\delta^{l+1}_{1,1}/4; \delta^l_{2,1}=\delta^{l+1}_{1,1}/4; \delta^l_{2,2}=\delta^{l+1}_{1,1}/4;$

卷积神经网络的训练

❑ 卷积层的训练

– 将卷积层也改为多层神经网络的形式

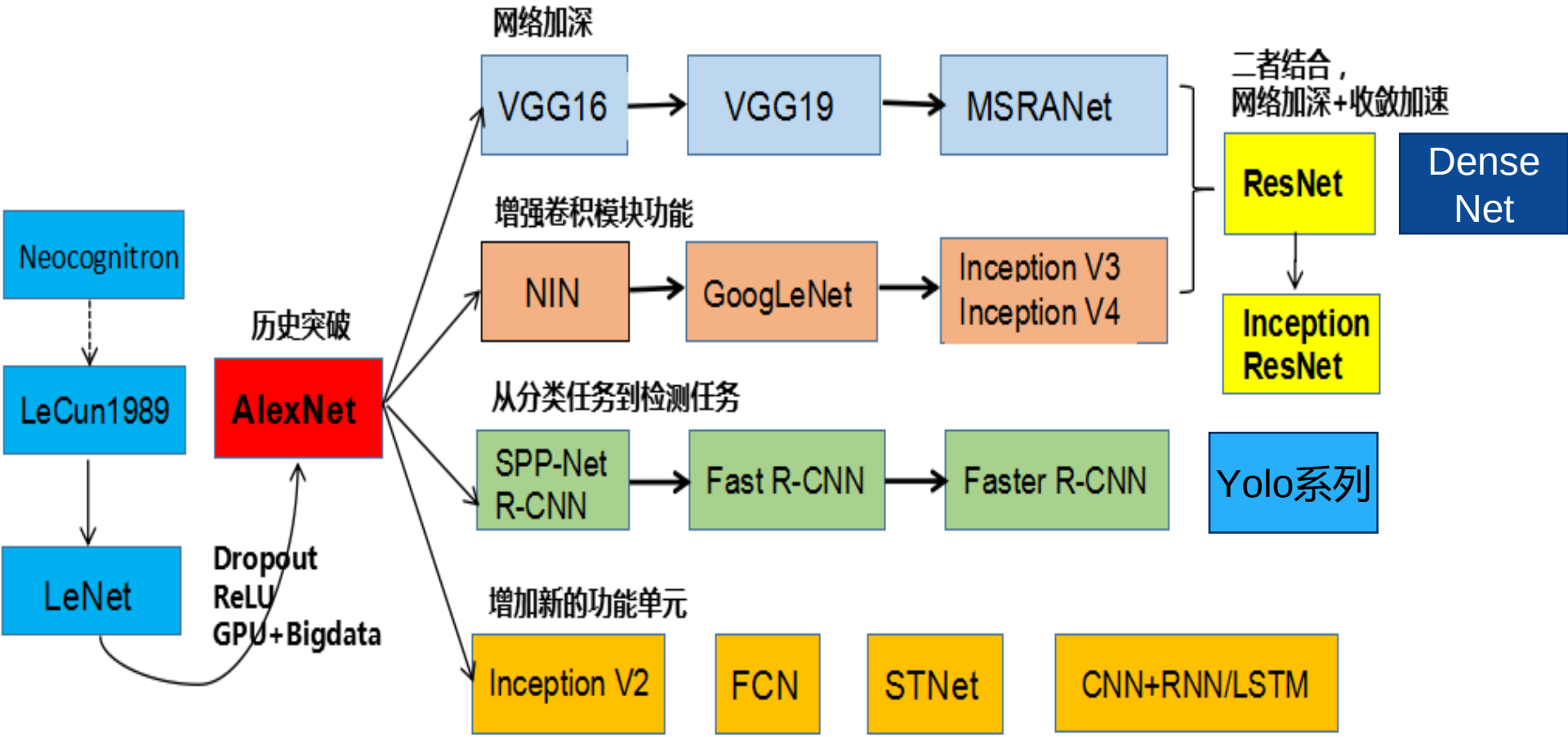




只与特定单元相连接，卷积核相当于权重w_{1ij}

4 经典卷积神经网络

卷积神经网络结构的演变

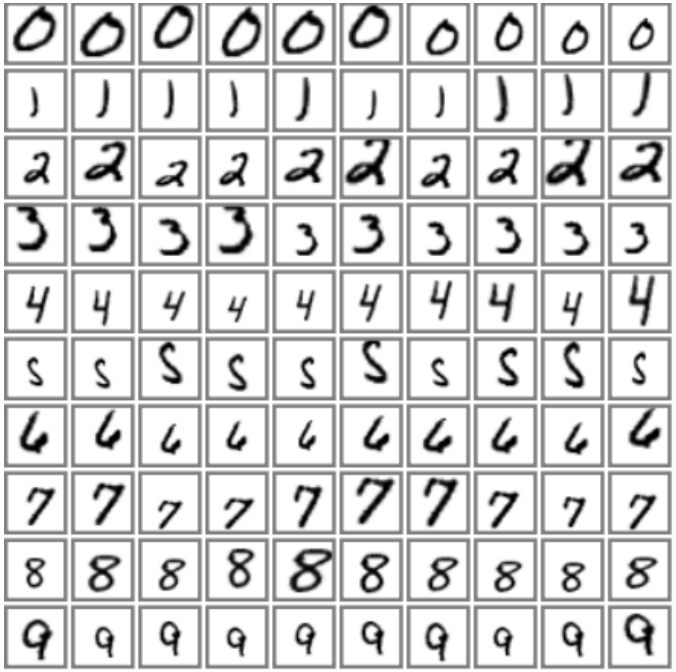


经典卷积神经网络

- LeNet-5
- AlexNet
- VGGNet
- Inception Net
- ResNet
- Densenet
- R-CNN系列
- YOLO系列

LeNet-5

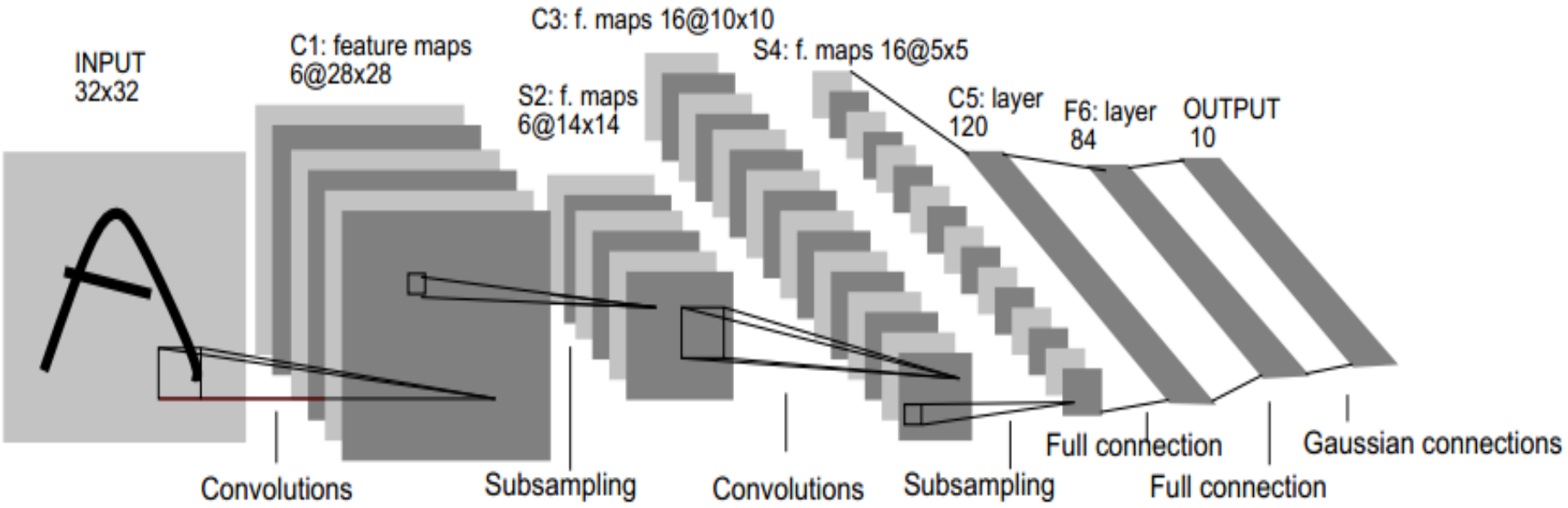
- LeNet-5由LeCun等人提出于1998年提出
- 主要进行手写数字识别和英文字母识别
- 经典的卷积神经网络，LeNet虽小，各模块齐全，是学习CNN的基础。
- http://yann.lecun.com/exdb/lenet/



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, November 1998.

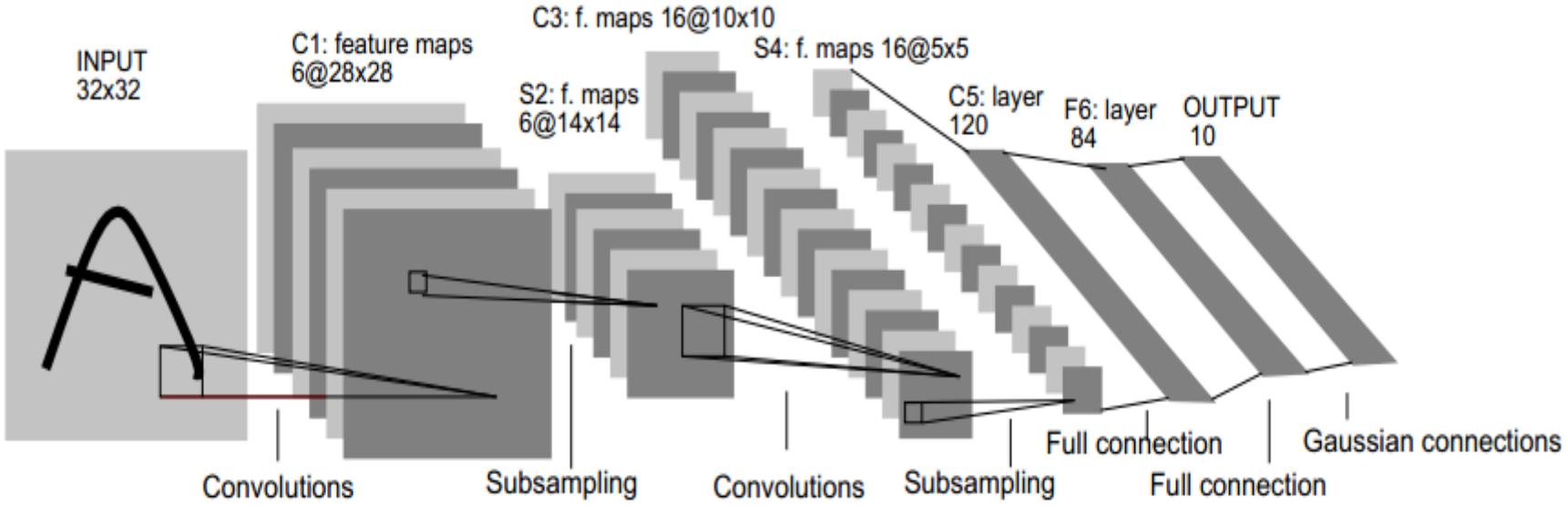
LeNet-5

❑ 网络结构


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

LeNet-5

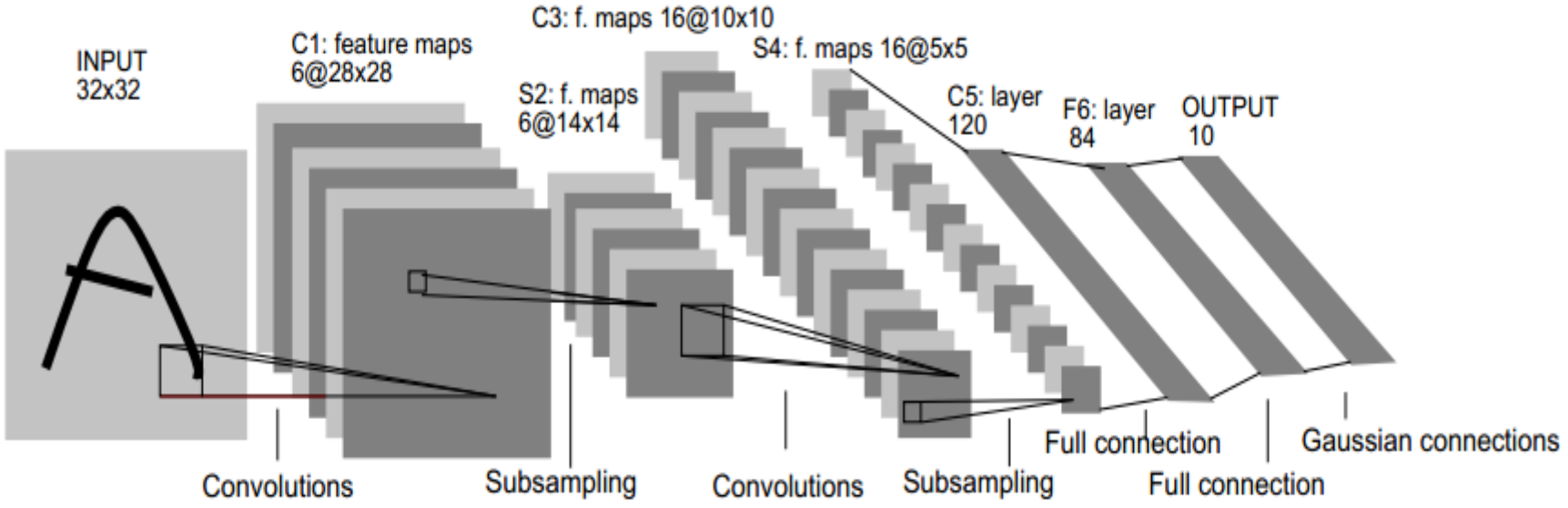
- **输入层**
  - 32*32的图片，也就是相当于1024个神经元
- **C1层（卷积层）**
  - 选择6个5*5的卷积核，得到6个大小为32-5+1=28的特征图，也就是神经元的个数为6*28*28=4704



LeNet-5

❑ S2层（下采样层）

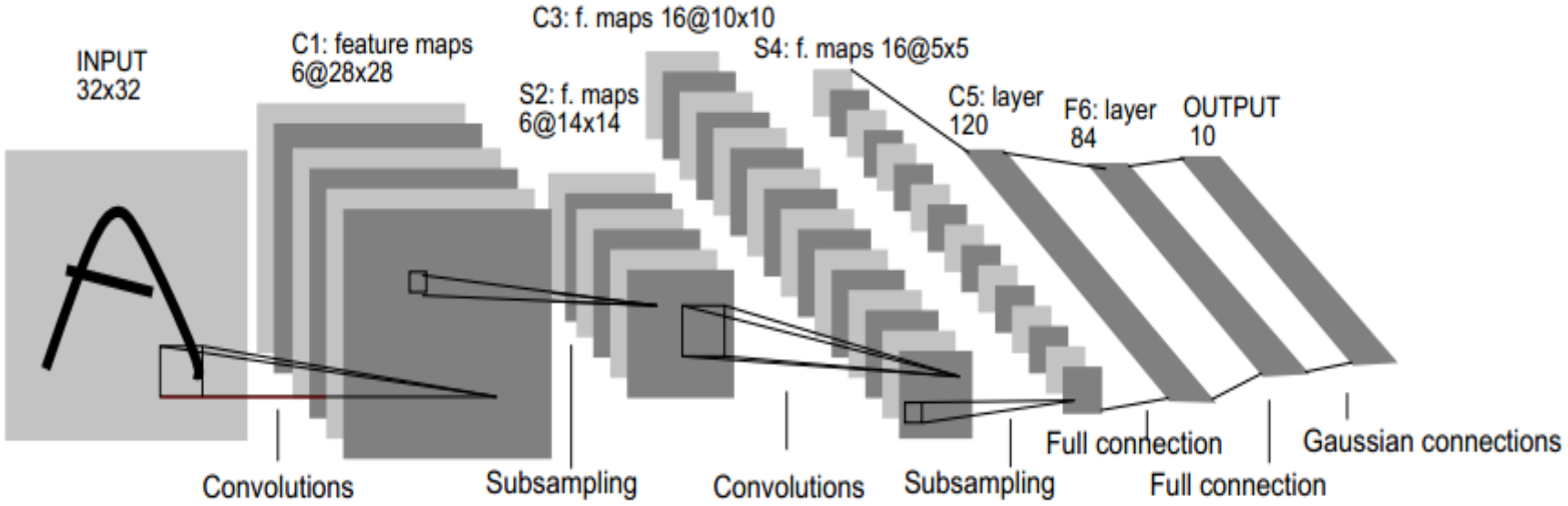
– 每个下抽样节点的4个输入节点求和后取平均（平均池化），均值乘上一个权重参数加上一个偏置参数作为激活函数的输入，激活函数的输出即是下一层节点的值。池化核大小选择2*2，得到6个14*14大小特征图



LeNet-5

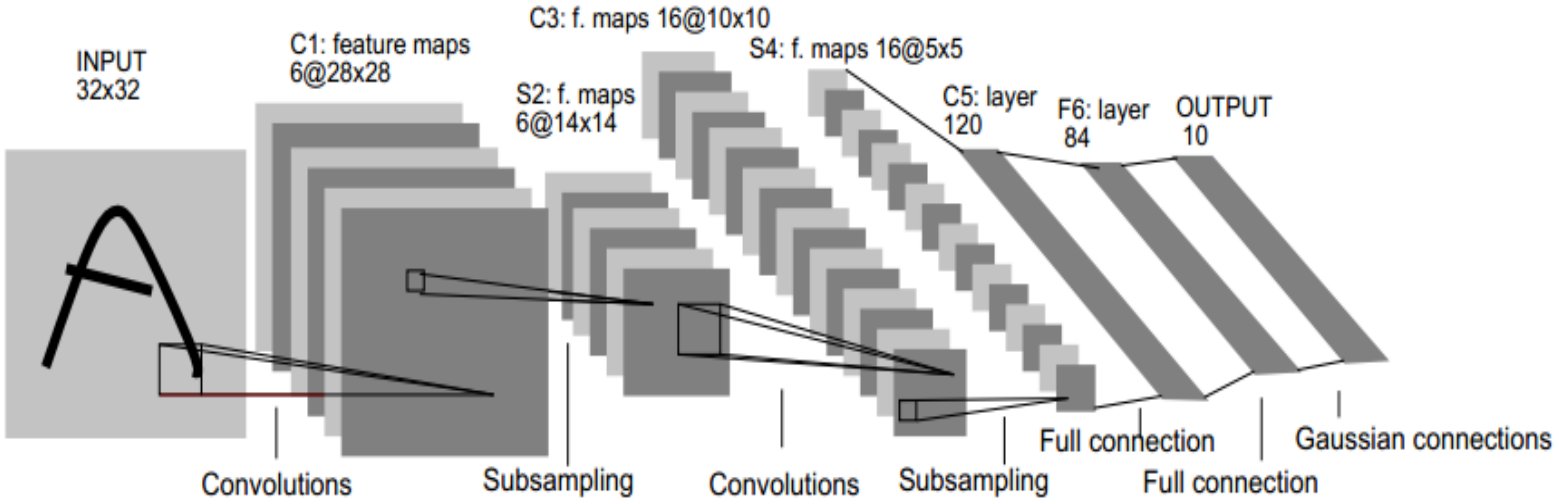
❑ C3层（卷积层）

– 用5*5的卷积核对S2层输出的特征图进行卷积后，得到6张10*10新图片，然后将这6张图片相加在一起，然后加一个偏置项b，然后用激活函数进行映射，就可以得到1张10*10的特征图。我们希望得到16张10*10的特征图，因此我们就需要参数个数为16*(6*(5*5))=16*6*(5*5)个参数



LeNet-5

- **S4层（下采样层）**
  - 对C3的16张10*10特征图进行最大池化，池化核大小为2*2，得到16张大小为5*5的特征图。神经元个数已经减少为：16*5*5=400
- **C5层（卷积层）**
  - 用5*5的卷积核进行卷积，然后我们希望得到120个特征图，特征图大小为5-5+1=1。神经元个数为120



LeNet-5

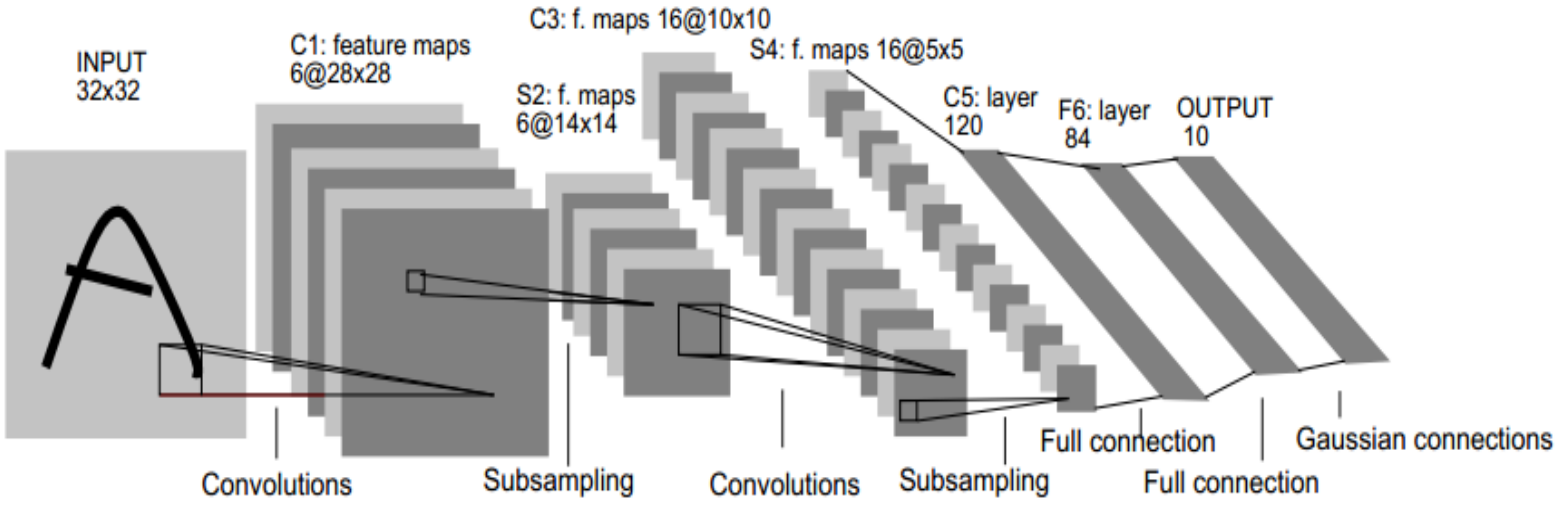
❑ F6层（全连接层）

– 有84个节点，该层的训练参数和连接数都是(120 + 1)x84=10164

❑ Output层

– 共有10个节点，分别代表数字0到9，如果节点i的输出值为0，则网络识别的结果是数字i。采用的是径向基函数(RBF)的网络连接方式

$$y_i = \sum_j (x_j - w_{ij})^2$$



LeNet-5

❑ 总结

– 卷积核大小、卷积核个数（特征图需要多少个）、池化核大小（采样率多少）这些参数都是变化的，这就是所谓的CNN调参，需要学会根据需要进行不同的选择

```
# Defining the network (LeNet-5)
class LeNet5(torch.nn.Module):

    def __init__(self):
        super(LeNet5, self).__init__()
        # Convolution (In LeNet-5, 32x32 images are given as input. Hence padding of 2 is done below)
        self.conv1 = torch.nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5, stride=1, padding=2, bias=True)
        # Max-pooling
        self.max_pool_1 = torch.nn.MaxPool2d(kernel_size=2)
        # Convolution
        self.conv2 = torch.nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5, stride=1, padding=0, bias=True)
        # Max-pooling
        self.max_pool_2 = torch.nn.MaxPool2d(kernel_size=2)
        # Fully connected layer
        self.fc1 = torch.nn.Linear(16*5*5, 120)   # convert matrix with 16*5*5 (= 400) features to a matrix of 120 features (columns)
        self.fc2 = torch.nn.Linear(120, 84)       # convert matrix with 120 features to a matrix of 84 features (columns)
        self.fc3 = torch.nn.Linear(84, 10)        # convert matrix with 84 features to a matrix of 10 features (columns)
```

AlexNet

- AlexNet由Hinton的学生Alex Krizhevsky于2012年提出
- 获得ImageNet LSVRC-2012(物体识别挑战赛)的冠军，1000个类别120万幅高清图像
  - Error: 26.2%(2011) →15.3%(2012)
- 通过AlexNet确定了CNN在计算机视觉领域的王者地位

A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

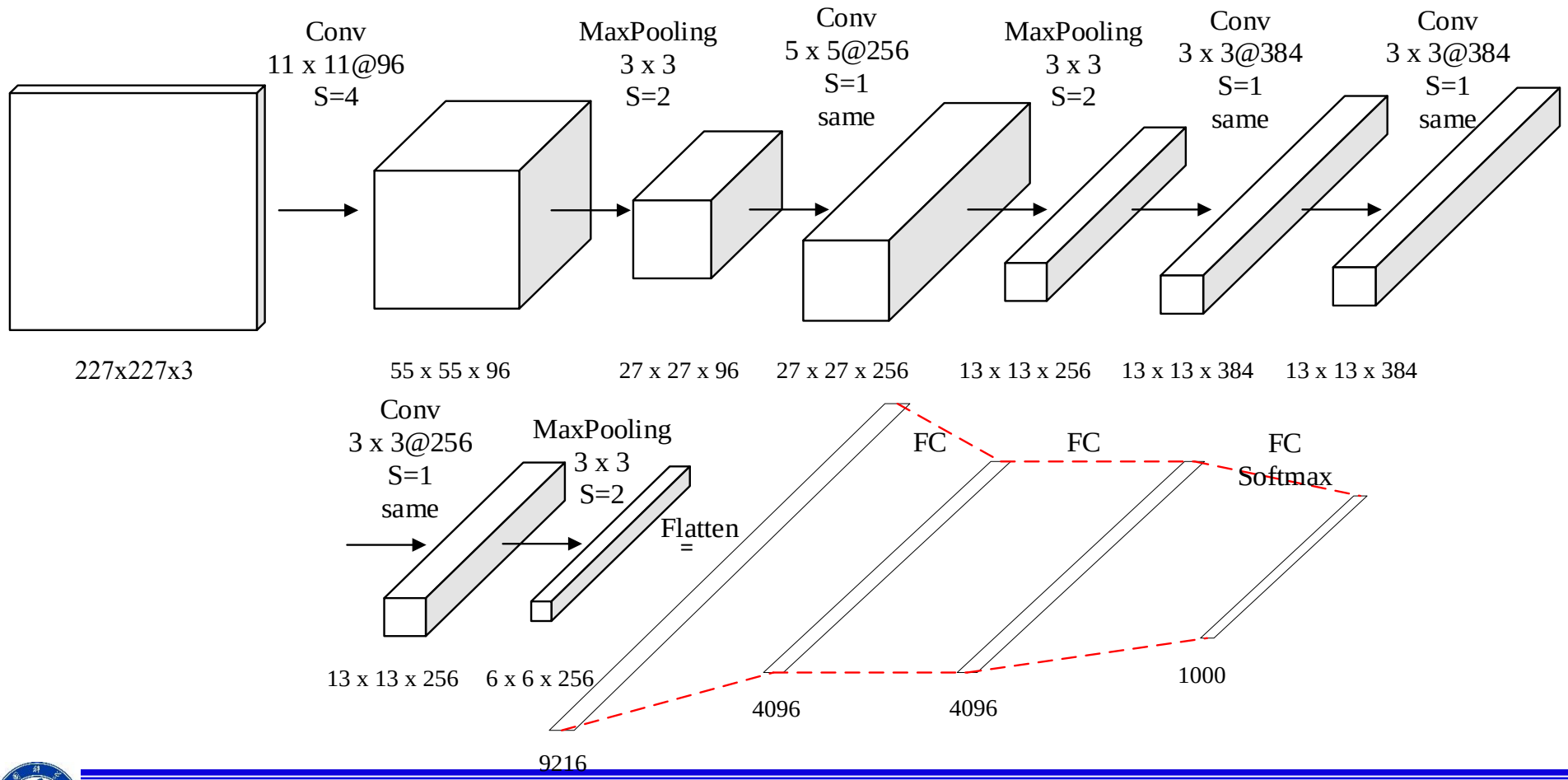
AlexNet

- 首次成功应用ReLU作为CNN的激活函数
- 使用Dropout丢弃部分神元，避免了过拟合
- 使用重叠MaxPooling(让池化层的步长小于池化核的大小)，一定程度上提升了特征的丰富性
- 使用CUDA加速训练过程
- 进行数据增强，原始图像大小为256×256的原始图像中重复截取224×224大小的区域，大幅增加了数据量，大大减轻了过拟合，提升了模型的泛化能力

A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

AlexNet

- AlexNet可分为8层（池化层未单独算作一层），包括5个卷积层以及3个全连接层



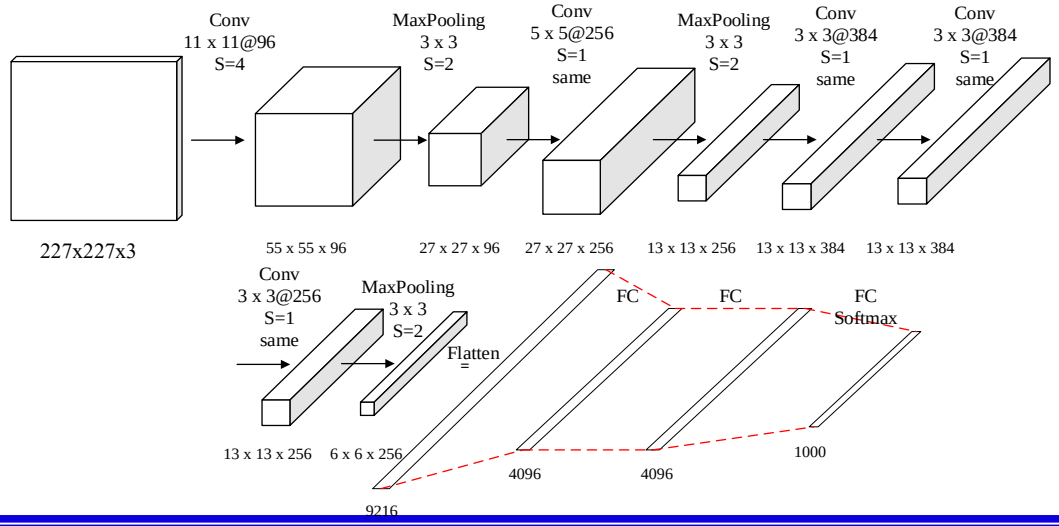
AlexNet

❑ 输入层

– AlexNet首先使用大小为224×224×3图像作为输入(实际输入227×227×3)

❑ 第一层（卷积层）

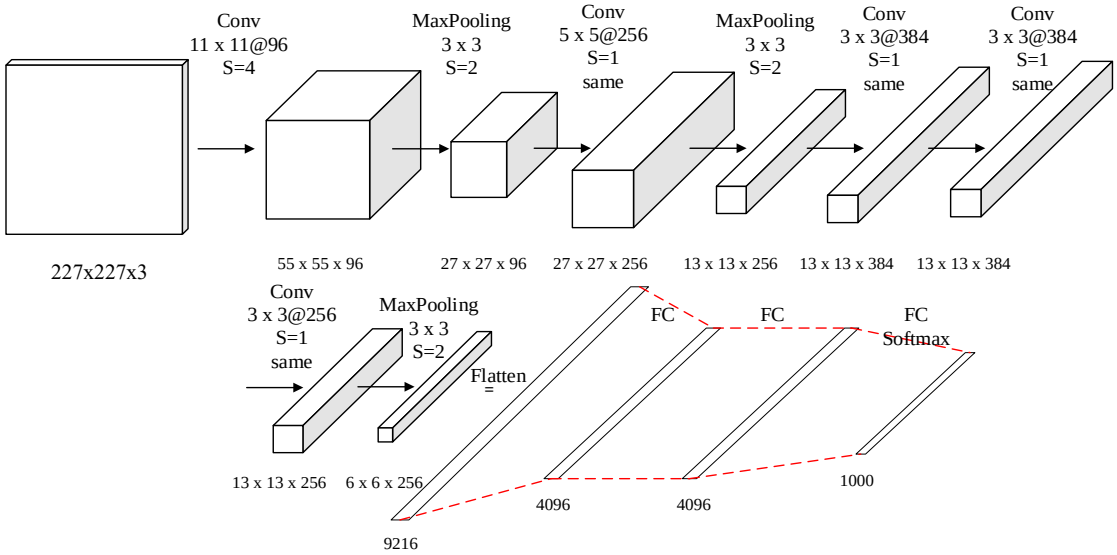
– 包含96个大小为11×11的卷积核，卷积步长为4，因此第一层输出大小为55×55×96；然后构建一个核大小为3×3、步长为2的最大池化层进行数据降采样，进而输出大小为27×27×96



AlexNet

❑ 第二层（卷积层）

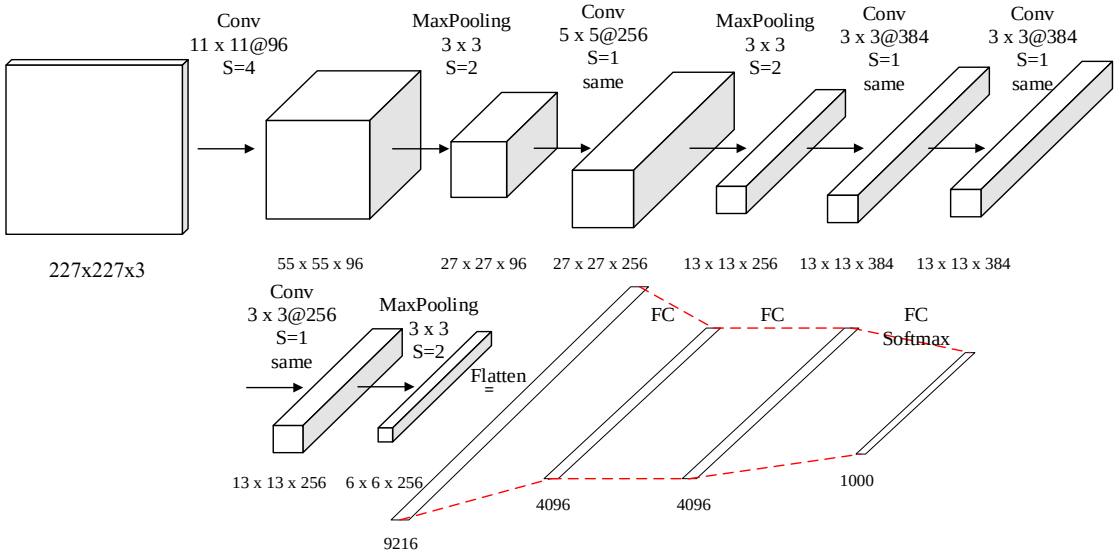
– 包含256个大小为5×5卷积核，卷积步长为1，同时利用padding保证输出尺寸不变，因此该层输出大小为27×27×256；然后再次通过核大小为3×3、步长为2的最大池化层进行数据降采样，进而输出大小为13×13×256



AlexNet

❑ 第三层与第四层（卷积层）

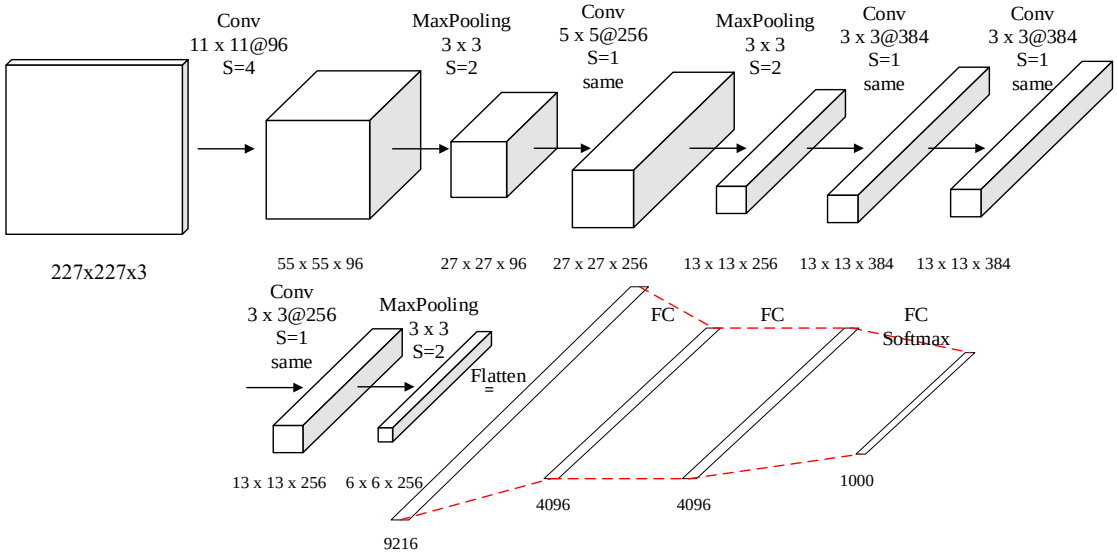
– 均为卷积核大小为3×3、步长为1的same卷积，共包含384个卷积核，因此两层的输出大小为13×13×384



AlexNet

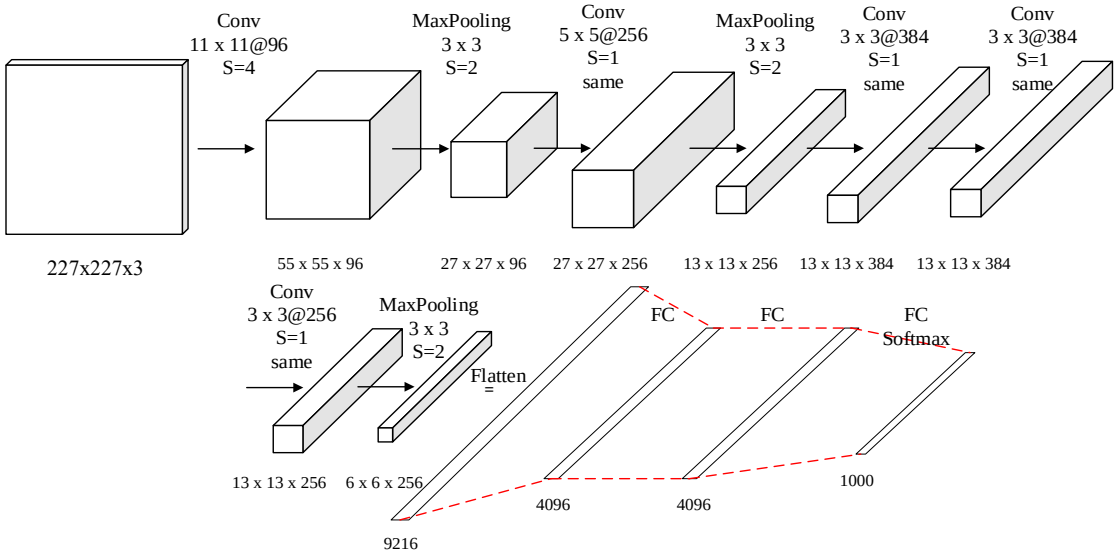
❑ 第五层（卷积层）

– 同样为卷积核大小为3×3、步长为1的same卷积，但包含256个卷积核，进而输出大小为13×13×256；在数据进入全连接层之前再次通过一个核大小为3×3、步长为2的最大池化层进行数据降采样，数据大小降为6×6×256，并将数据扁平化处理展开为9216个单元



AlexNet

- 第六层、第七层和第八层（全连接层）
  - 全连接加上Softmax分类器输出1000类的分类结果



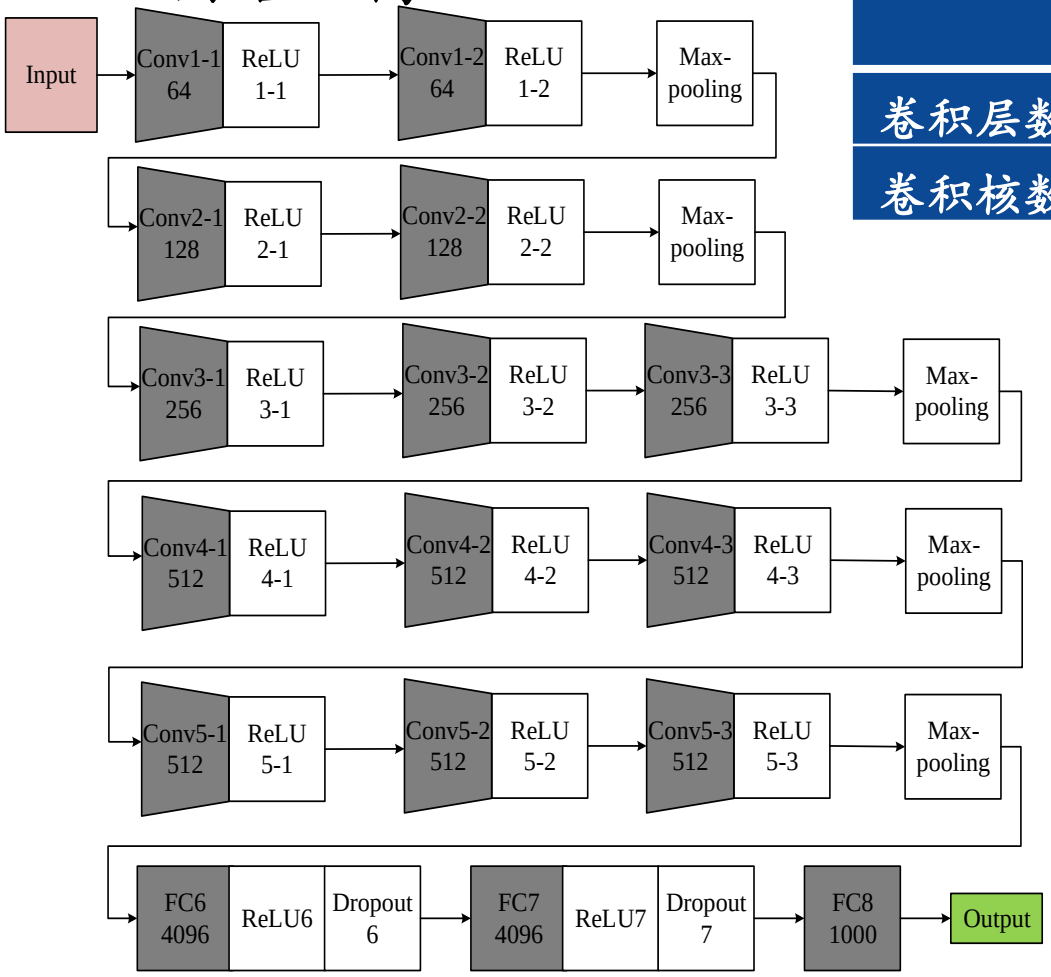
VGG-16

- VGGNet由剑桥大学和DeepMind公司提出
- 获得ImageNet LSVRC-2014亚军
- 比较常用的是VGG-16，结构规整，具有很强的拓展性
- 相较于AlexNet，VGG-16网络模型中的卷积层均使用3*3的卷积核，且均为步长为1的same卷积，池化层均使用2*2的池化核，步长为2

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

VGG-16

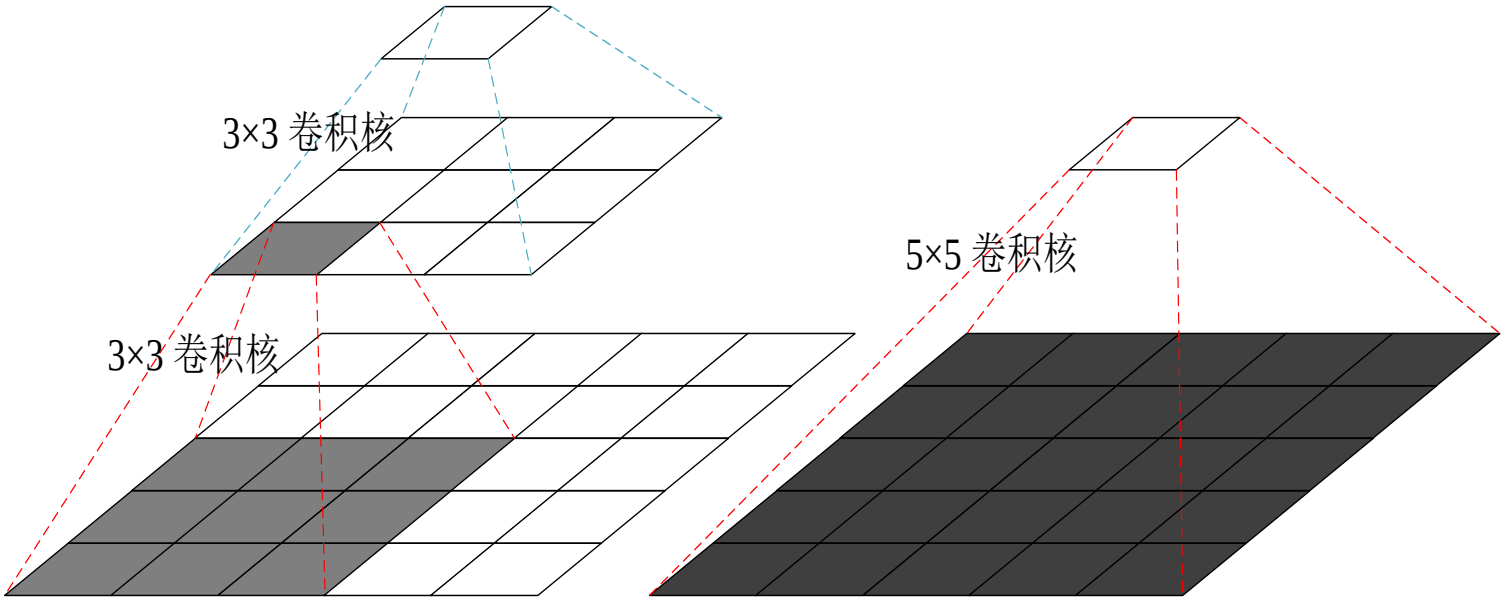
❑ 网络结构



	Conv1	Conv2	Conv3	Conv4	Conv5
卷积层数	2	2	3	3	3
卷积核数	64	128	256	512	512

VGG-16

- 两个卷积核大小为3*3的卷积层串联后的感受野尺寸为5*5，相当于单个卷积核大小为5*5的卷积层
- 两者参数数量比值为(2*3*3)/(5*5)=72%，前者参数量更少
- 此外，两个的卷积层串联可使用两次ReLU激活函数，而一个卷积层只使用一次

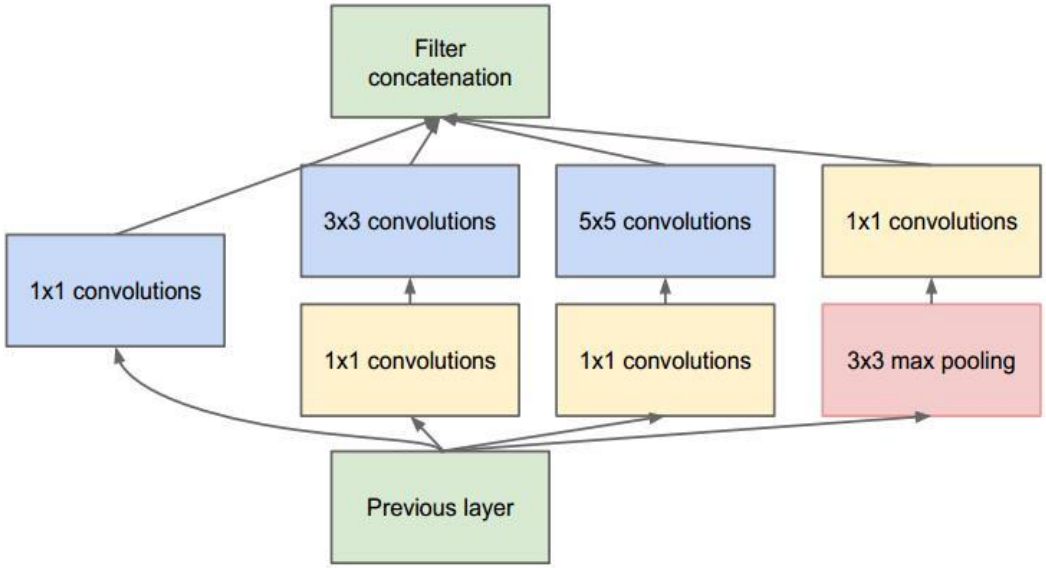


Inception Net

- Google公司2014年提出
- 获得ImageNet LSVRC-2014冠军
- 文章提出获得高质量模型最保险的做法就是增加模型的深度(层数)或者是其宽度(层核或者神经元数)，采用了22层网络
- Inception四个版本所对应的论文及ILSVRC中的Top-5错误率：
  - [v1] Going Deeper with Convolutions: 6.67%
  - [v2] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift: 4.8%
  - [v3] Rethinking the Inception Architecture for Computer Vision: 3.5%
  - [v4] Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning: 3.08%

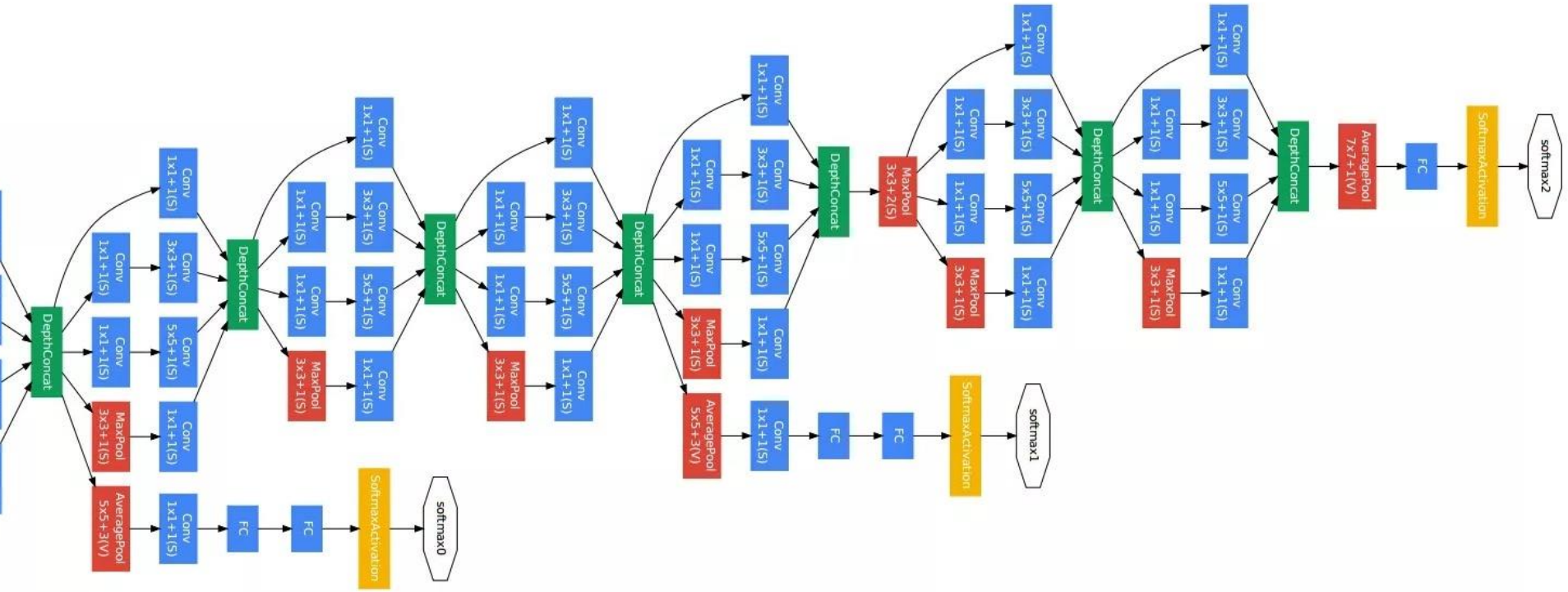
Inception Net

- 深度：层数更深，采用了22层，在不同深度处增加了两个loss来避免上述提到的梯度消失问题
- 宽度：Inception Module包含4个分支，在卷积核3x3、5x5之前、max pooling之后分别加上了1x1的卷积核，起到了降低特征图厚度的作用



Inception Module

Inception Net



ResNet

- ResNet（Residual Neural Network），又叫做残差神经网络，是由微软研究院的何凯明等人2015年提出
- 获得ImageNet ILSVRC 2015比赛冠军
- 获得CVPR2016最佳论文奖

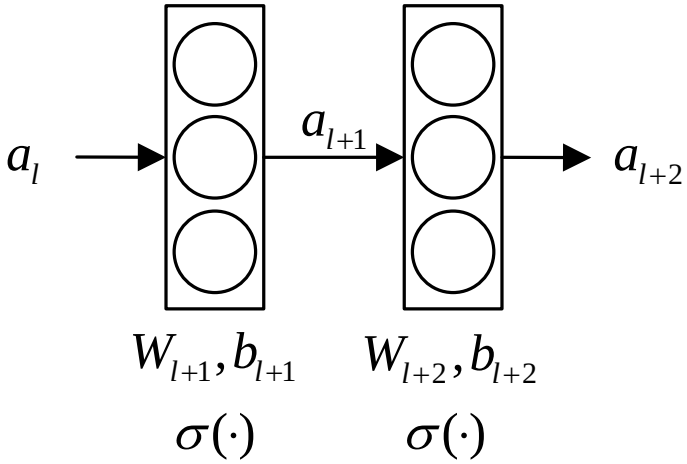
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR 2016: 770-778

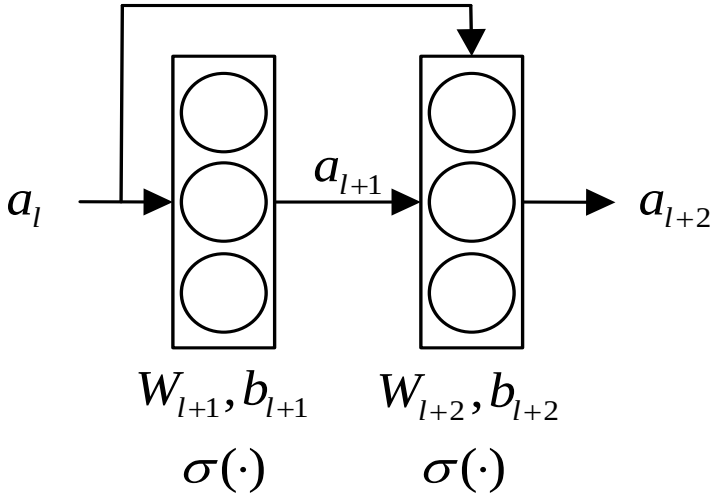
ResNet

- 随着卷积网络层数的增加，误差的逆传播过程中存在的梯度消失和梯度爆炸问题同样也会导致模型的训练难以进行
- 甚至会出现随着网络深度的加深，模型在训练集上的训练误差会出现先降低再升高的现象
- 残差网络的引入则有助于解决梯度消失和梯度爆炸问题

ResNet

- ResNet的核心是叫做残差块（Residual block）的小单元，残差块可以视作在标准神经网络基础上加入了跳跃连接（Skip connection）



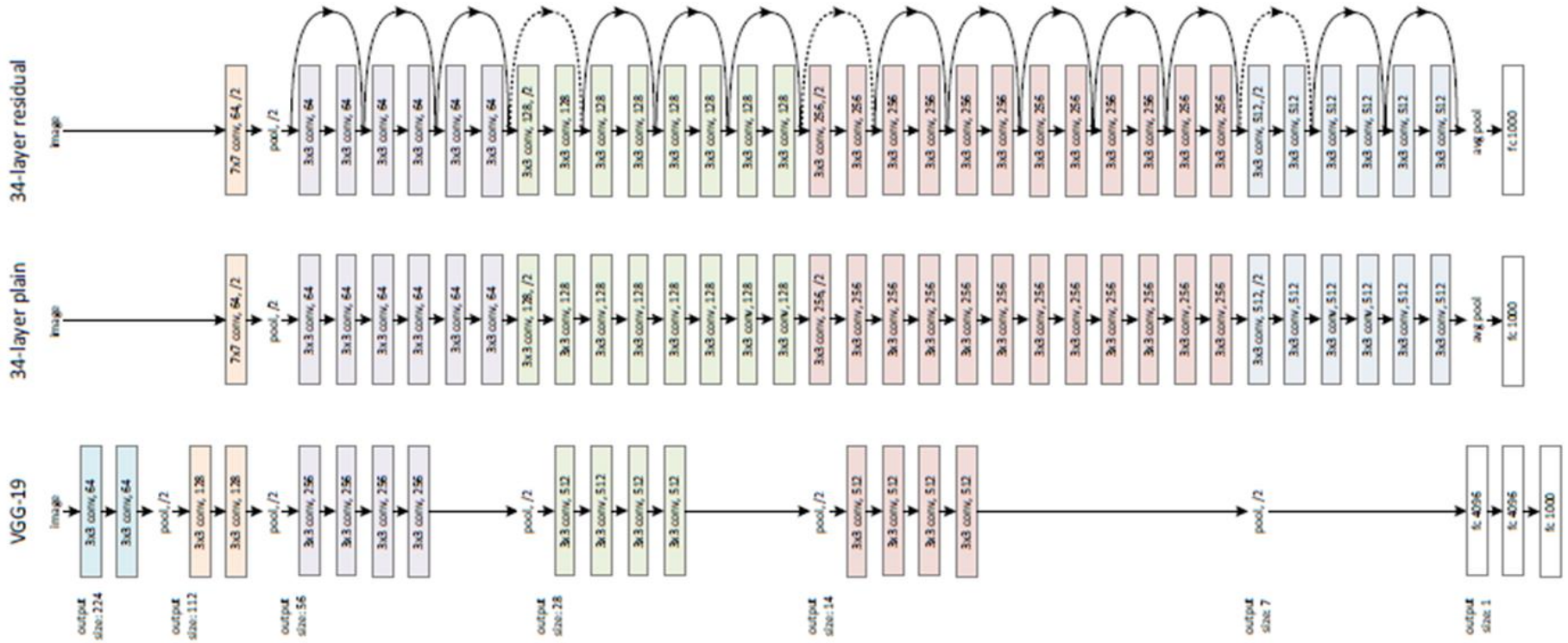


$$a_{l+1} = \sigma(W_{l+1}a_l + b_{l+1})$$
$$a_{l+2} = \sigma(W_{l+2}a_{l+1} + b_{l+2})$$

$$a_{l+1} = \sigma(W_{l+1}a_l + b_{l+1})$$
$$a_{l+2} = \sigma(W_{l+2}a_{l+1} + b_{l+2} + a_l)$$

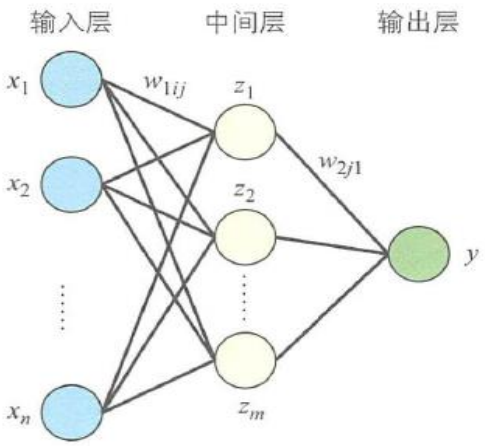
ResNet

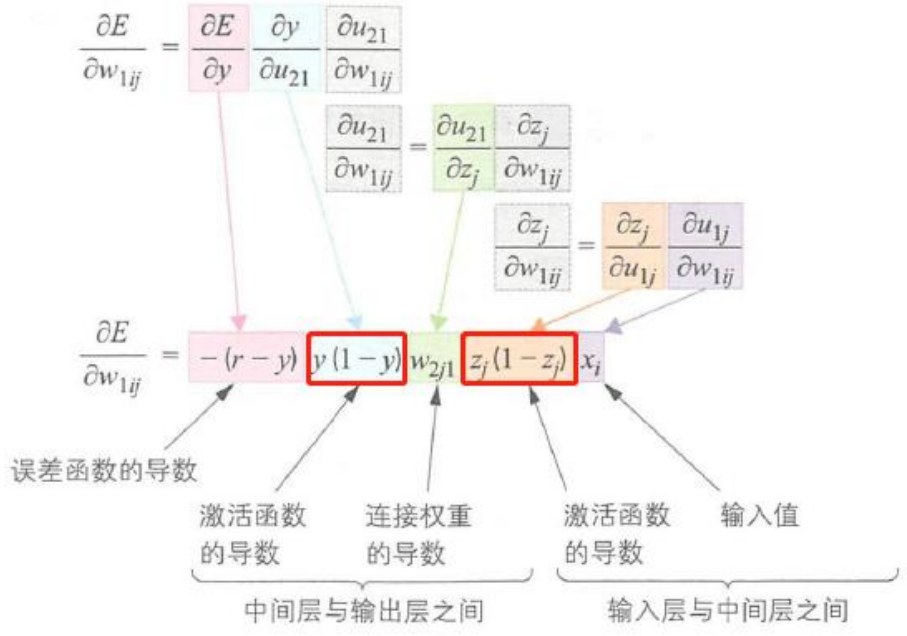
❑ 网络结构比较

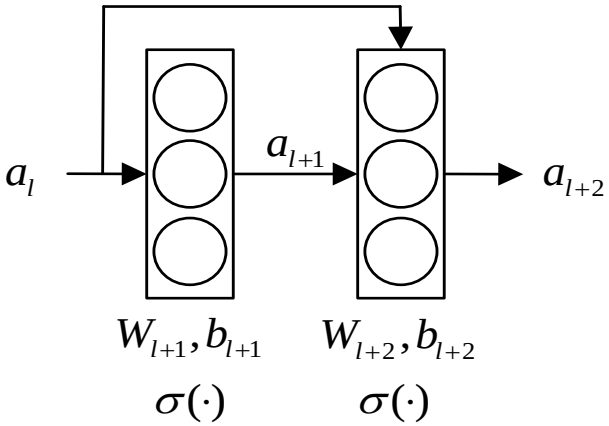


ResNet

❑ Skip connections的作用





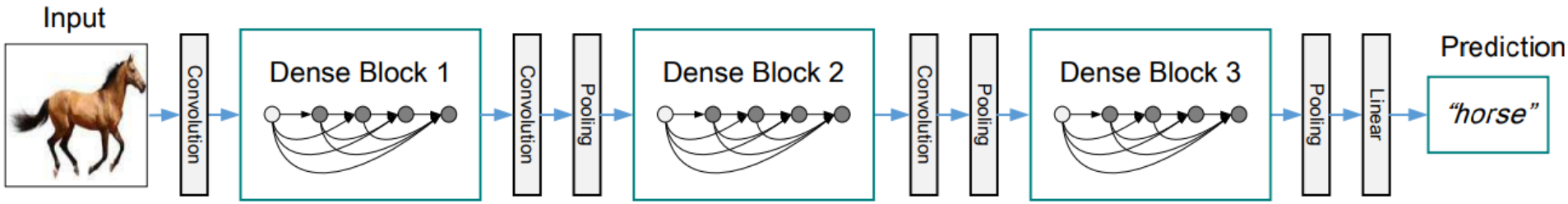


$$a_{l+1} = \sigma(W_{l+1}a_l + b_{l+1})$$

$$a_{l+2} = \sigma(W_{l+2}a_{l+1} + b_{l+2} + a_l)$$

DenseNet

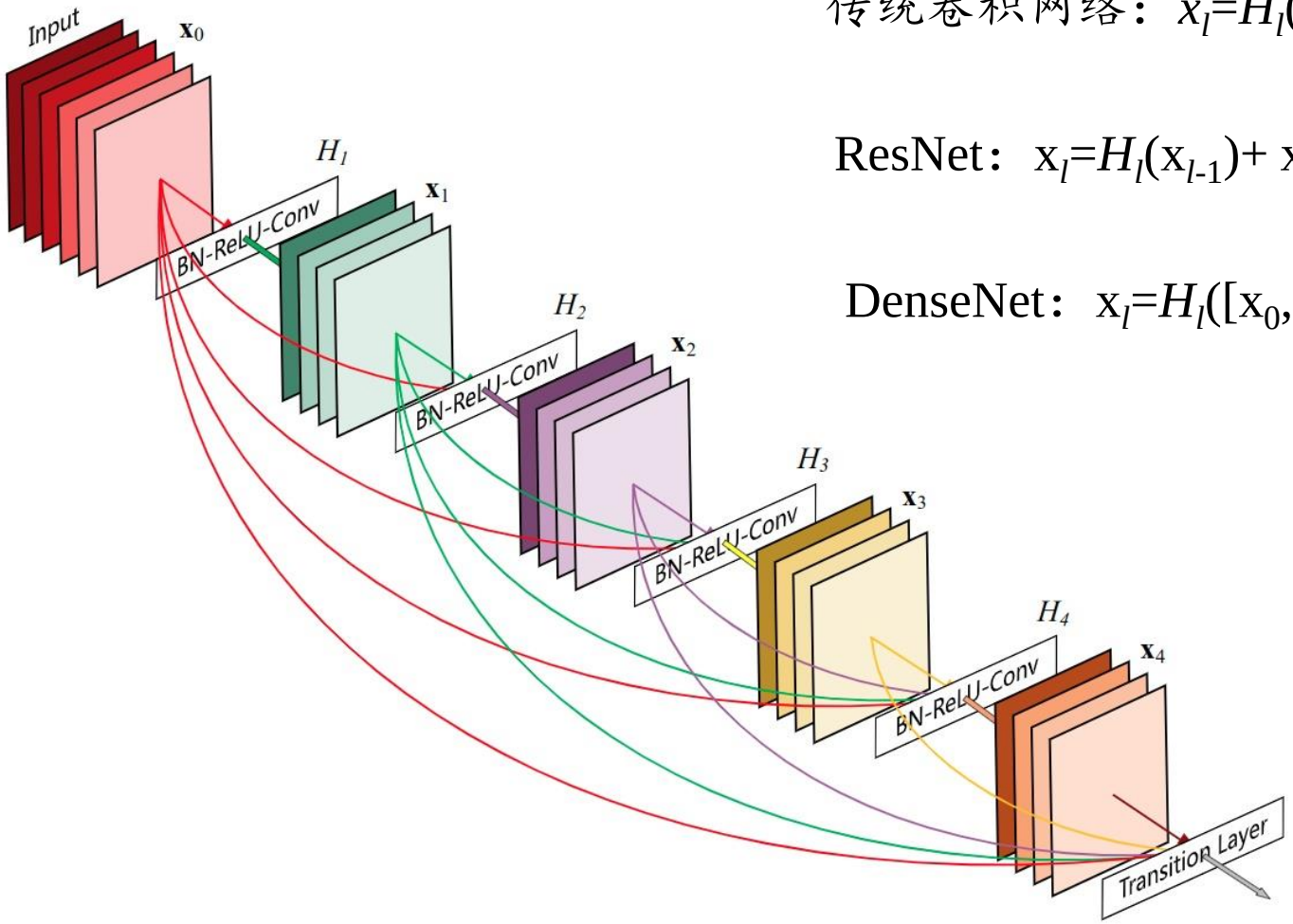
- DenseNet中，两个层之间都有直接的连接，因此该网络的直接连接个数为L(L+1)/2。
- 对于每一层，使用前面所有层的特征映射作为输入，并且使用其自身的特征映射作为所有后续层的输入



Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).

DenseNet

❑ 5层的稠密块示意图



传统卷积网络：$x_l=H_l(x_{l-1})$

ResNet：$x_l=H_l(x_{l-1})+ x_{l-1}$

DenseNet：$x_l=H_l([x_0,x_1,\ldots,x_{l-1}])$

DenseNet

- DenseNets可以自然地扩展到数百个层，而没有表现出优化困难
- 在实验中，DenseNets随着参数数量的增加，在精度上产生一致的提高，而没有任何性能下降或过拟合的迹象
- DenseNet的优点：
  - 缓解了消失梯度问题
  - 加强了特征传播，鼓励特征重用
  - 一定程度上减少了参数的数量

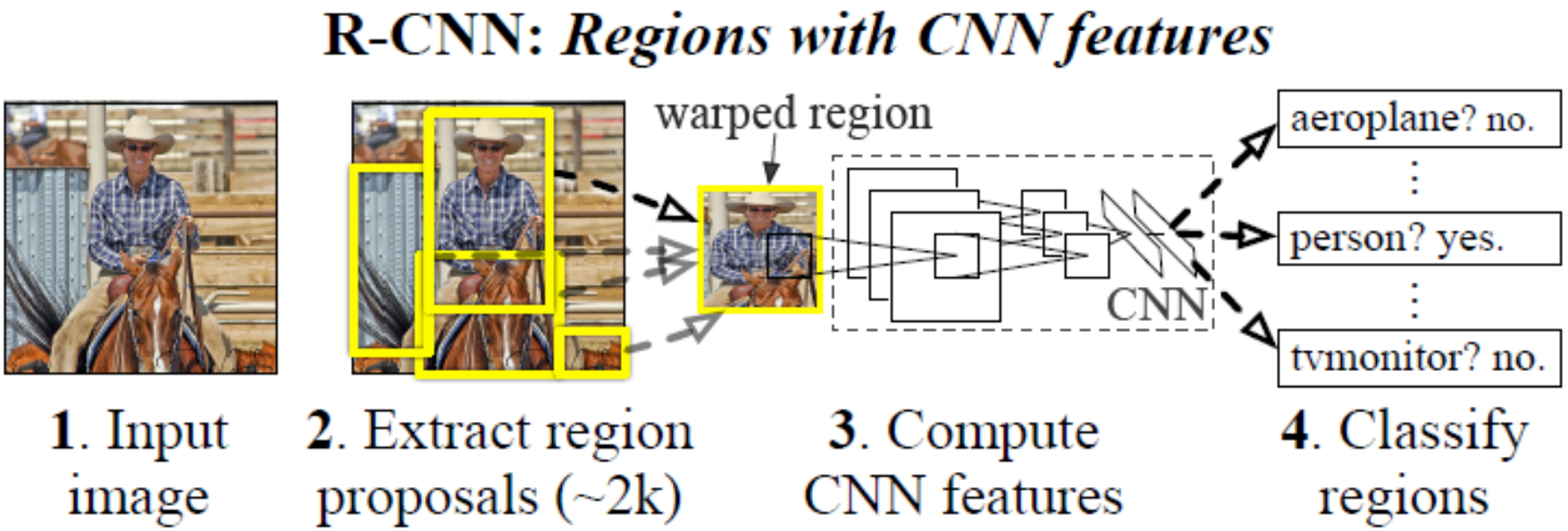
R-CNN系列

❑ R-CNN

- Region-CNN的缩写，主要用于目标检测。
- 来自2014年CVPR论文“Rich feature hierarchies for accurate object detection and semantic segmentation”
- 在 Pascal VOC 2012 的数据集上，能够将目标检测的验证指标 mAP 提升到 53.3%，这相对于之前最好的结果提升了整整 30%
- 采用在 ImageNet 上已经训练好的模型，然后在 PASCAL VOC 数据集上进行 fine-tune

R-CNN系列

❑ Object detection system overview



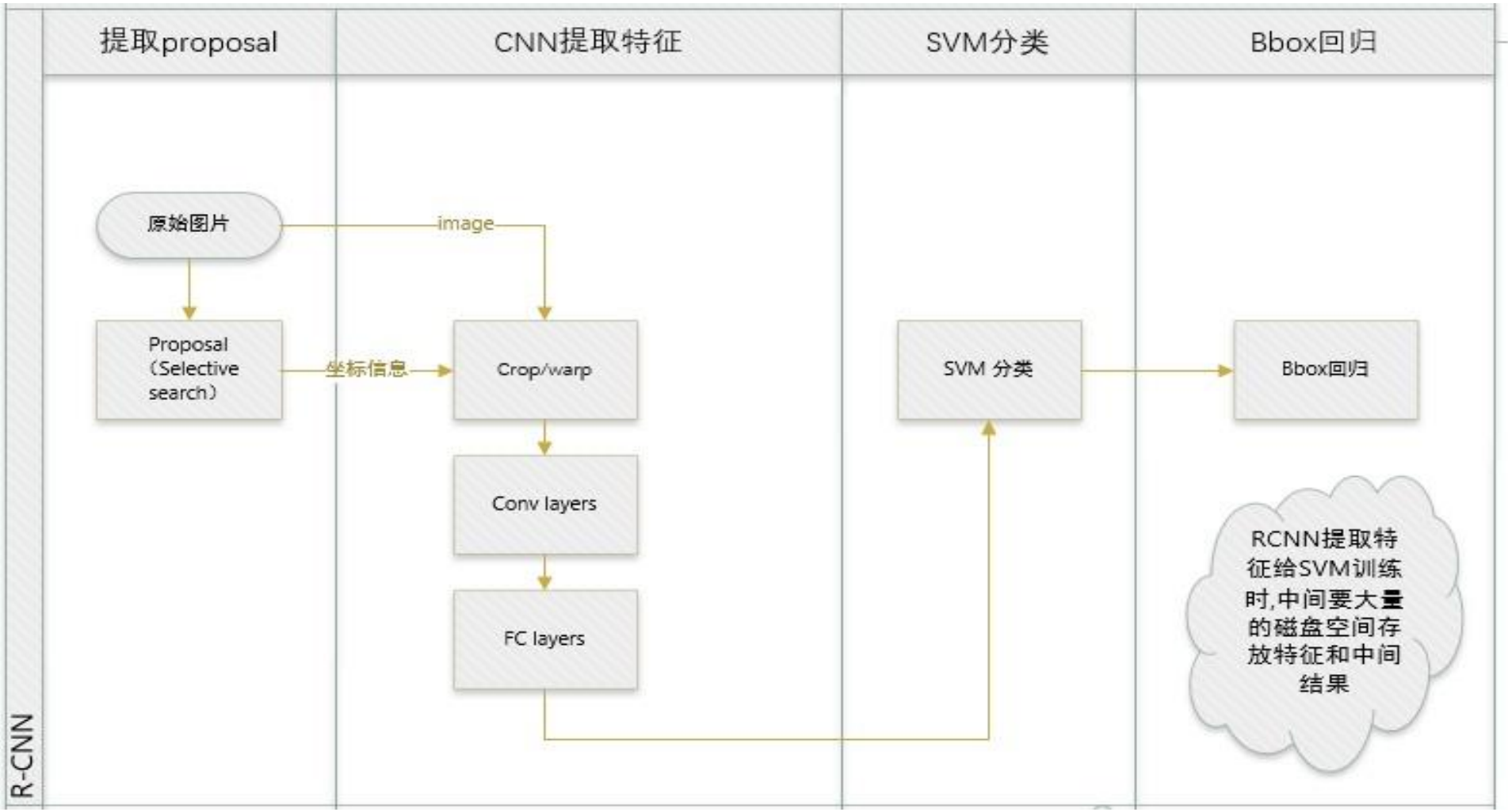
Ross B. Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. CVPR 2014: 580-587

R-CNN系列

❑ 实现过程

- 区域划分：给定一张输入图片，从图片中提取 2000 个类别独立的候选区域，R-CNN 采用的是 Selective Search 算法
- 特征提取：对于每个区域利用 CNN 抽取一个固定长度的特征向量，R-CNN 使用的是 Alexnet
- 目标分类：再对每个区域利用 SVM 进行目标分类
- 边框回归：Bounding box Regression（Bbox回归)进行边框坐标偏移优化和调整

R-CNN系列



- Crop就是从一个大图扣出网络输入大小的patch，比如227×227
- Warp把一个边界框bounding box的内容resize成227×227

R-CNN系列

❑ Selective Search 算法

- 核心思想：图像中物体可能存在的区域应该有某些相似性或者连续性的，选择搜索基于上面这一想法采用子区域合并的方法提取bounding boxes候选边界框
- 首先，通过图像分割算法将输入图像分割成许多小的子区域
- 其次，根据这些子区域之间的相似性(主要考虑颜色、纹理、尺寸和空间交叠4个相似) 进行区域迭代合并。每次迭代过程中对这些合并的子区域做bounding boxes(外切矩形)，这些子区域的外切矩形就是通常所说的候选框

R-CNN系列

❑ Selective Search算法步骤

- Step 0：生成区域集R，参见论文《Efficient Graph-Based Image Segmentation》
- Step 1：计算区域集R里每个相邻区域的相似度S={s1,s2,…}
- Step 2：找出相似度最高的两个区域，将其合并为新集，添加进R
- Step 3：从S中移除所有与step2中有关的子集
- Step 4：计算新集与所有子集的相似度
- Step 5：跳至step2，直至S为空

R-CNN系列

❑ Selective Search算法步骤

Algorithm 1: Hierarchical Grouping Algorithm

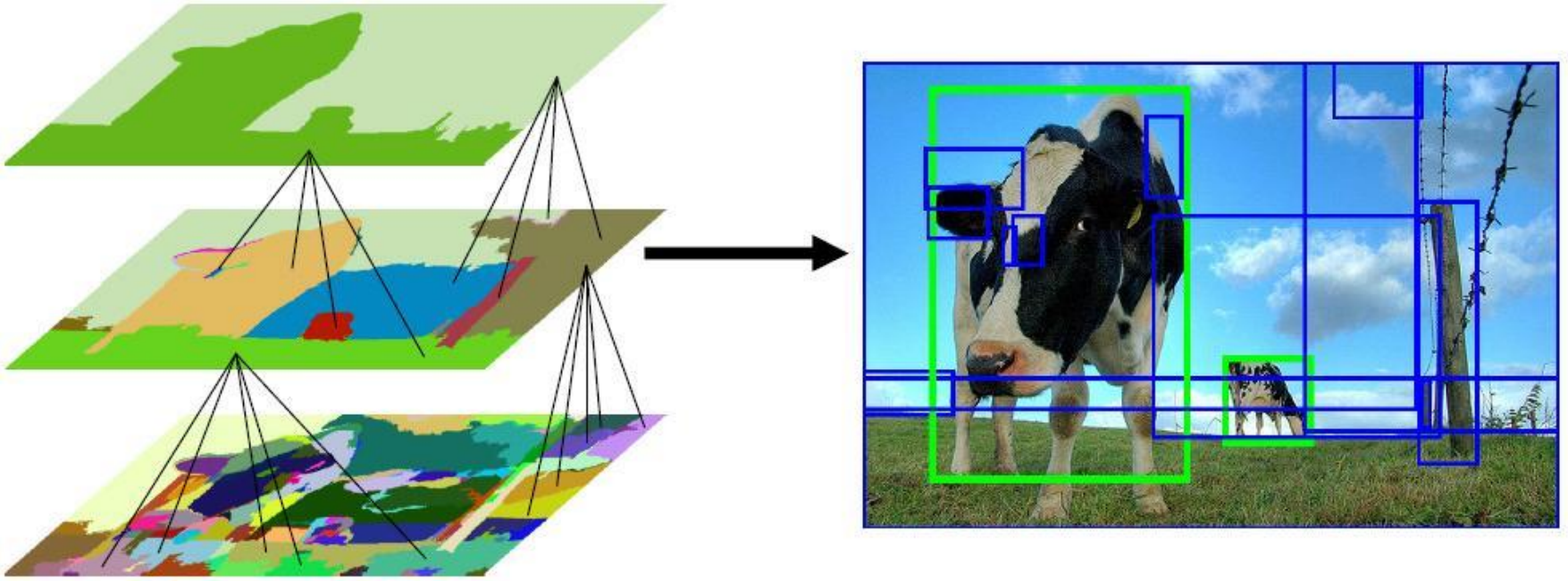
Input: (colour) image
Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
Initialise similarity set $S = \emptyset$
foreach *Neighbouring region pair* (r_i, r_j) **do**
    Calculate similarity $s(r_i, r_j)$
    $S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ **do**
    Get highest similarity $s(r_i, r_j) = \max(S)$
    Merge corresponding regions $r_t = r_i \cup r_j$
    Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$
    Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$
    Calculate similarity set S_t between r_t and its neighbours
    $S = S \cup S_t$
    $R = R \cup r_t$

Extract object location boxes L from all regions in R

R-CNN系列



R-CNN系列

R-CNN系列

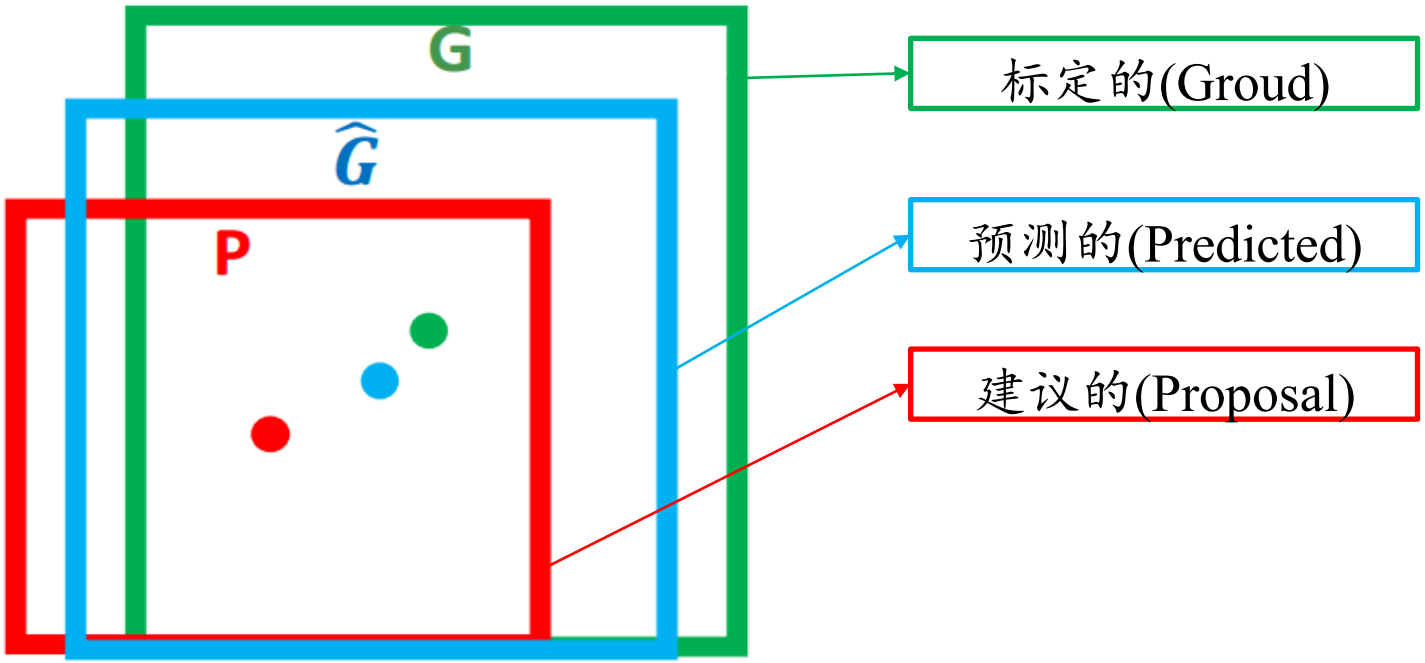
❑ Bbox回归

– 核心思想：通过平移和缩放方法对物体边框进行调整和修正

R-CNN系列

❑ Bbox回归

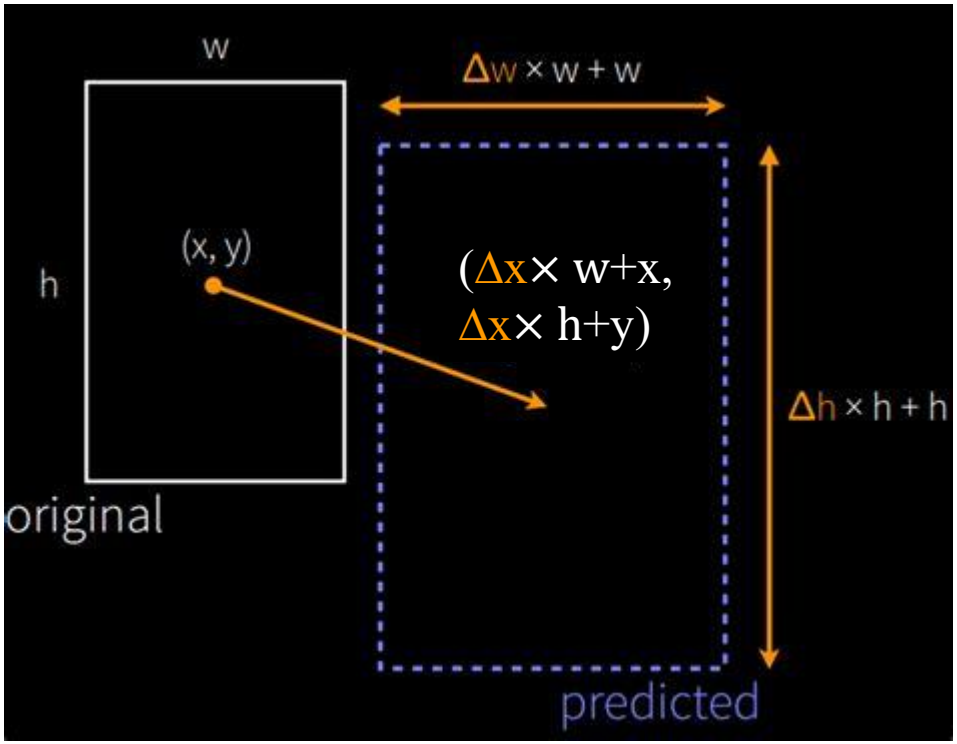
- bounding box的表示为(x,y,w,h)，即窗口的中心点坐标和宽高
- Bbox回归就是找到函数f,将(P_x,P_y,P_w,P_h)映射为更接近(G_x,G_y,G_w,G_h)的$(\hat{G}_x,\hat{G}_y,\hat{G}_w,\hat{G}_h)$



R-CNN系列

❑ Bbox回归

– 核心思想：通过平移和缩放方法对物体边框进行调整和修正



$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp(d_w(P))$$
$$\hat{G}_h = P_h \exp(d_h(P)).$$

P是proposal内的CNN特征

R-CNN系列

❑ 性能对比

VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 [17]†	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA [32]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets [35]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM [15]†	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN BB	**71.8**	**65.8**	**53.0**	**36.8**	**35.9**	**59.7**	**60.0**	**69.9**	**27.9**	**50.6**	**41.4**	**70.0**	**62.0**	**69.0**	**58.1**	**29.5**	**59.4**	**39.3**	**61.2**	**52.4**	**53.7**

Table 1: Detection average precision (%) on VOC 2010 test. R-CNN is most directly comparable to UVA and Regionlets since all methods use selective search region proposals. Bounding box regression (BB) is described in Section 3.4. At publication time, SegDPM was the top-performer on the PASCAL VOC leaderboard. †DPM and SegDPM use context rescoring not used by the other methods.

mAP: mean Average Precision, 是多标签图像分类任务中的评价指标。AP衡量的是学出来的模型在给定类别上的好坏，而mAP衡量的是学出的模型在所有类别上的好坏

R-CNN系列

❑ SPPnet (Spatial Pyramid Pooling): 空间金字塔网络

– R-CNN主要问题：每个Proposal独立提取CNN features，分步训练

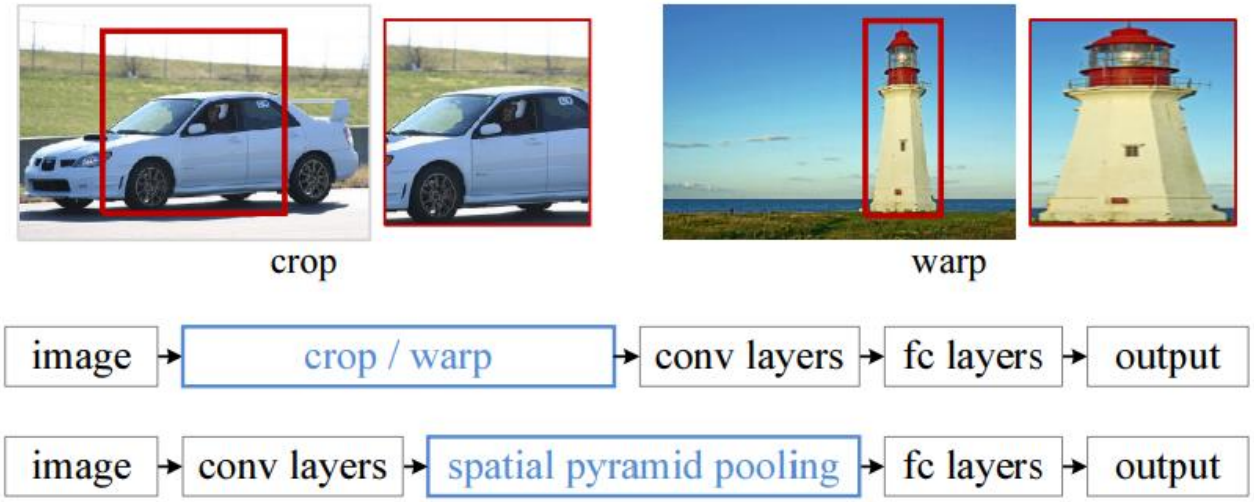


Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Trans. Pattern Anal. Mach. Intell. 37(9): 1904-1916 (2015)

R-CNN系列

❑ SPPnet: 空间金字塔网络

– R-CNN主要问题：每个Proposal独立提取CNN features，分步训练

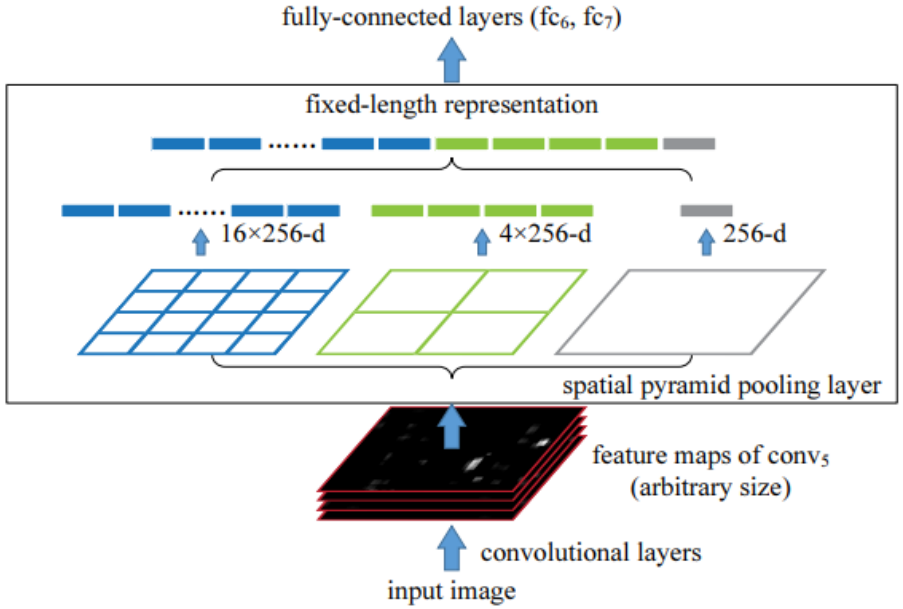


Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Trans. Pattern Anal. Mach. Intell. 37(9): 1904-1916 (2015)

R-CNN系列

❑ Fast R-CNN

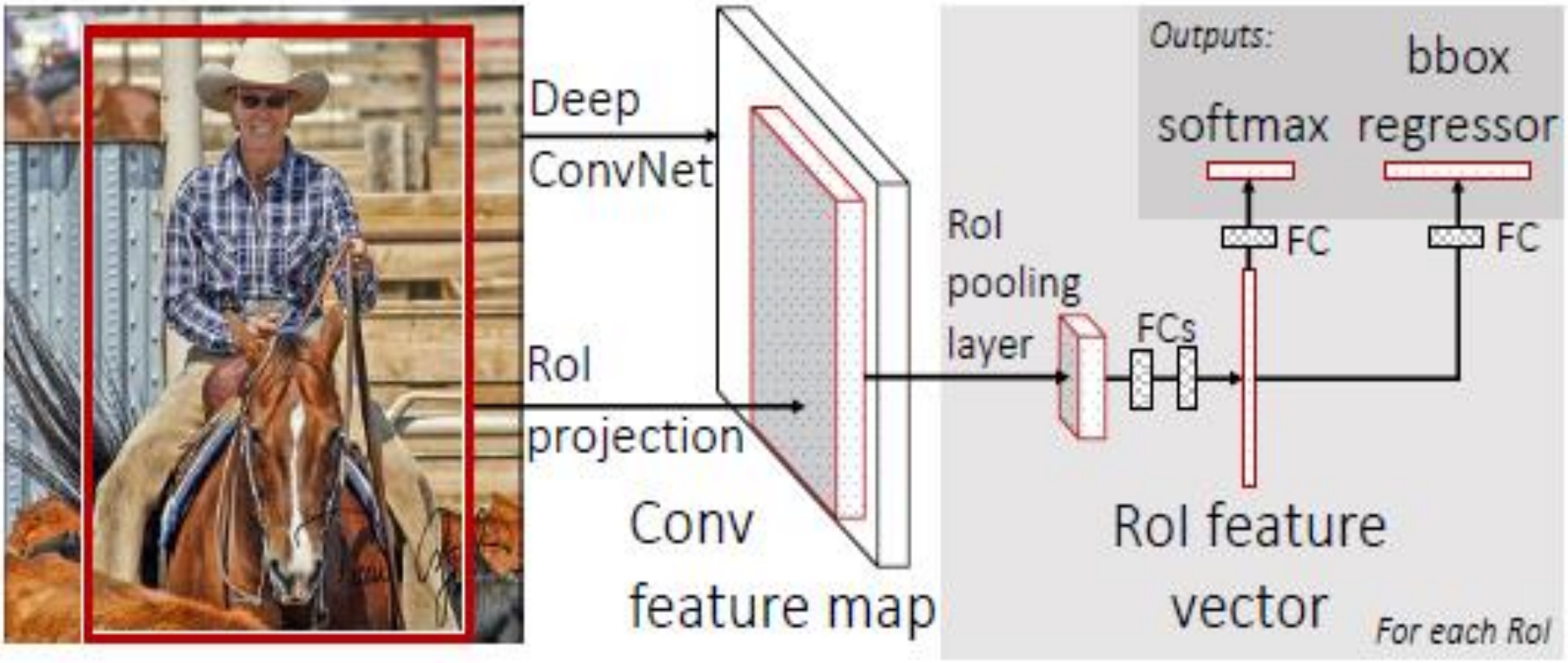
- R-CNN和SPPnet问题：训练步骤过多，需要训练SVM分类器，需要额外的回归器，特征也是保存在磁盘上

- 联合学习（joint training）：把SVM、Bbox回归和CNN阶段一起训练，最后一层的Softmax换成两个：一个是对区域的分类Softmax，另一个是对Bounding box的微调。训练时所有的特征不再存到硬盘上，提升了速度。
- ROI Pooling层：实现了单尺度的区域特征图的Pooling

Ross B. Girshick. Fast R-CNN. ICCV 2015: 1440-1448

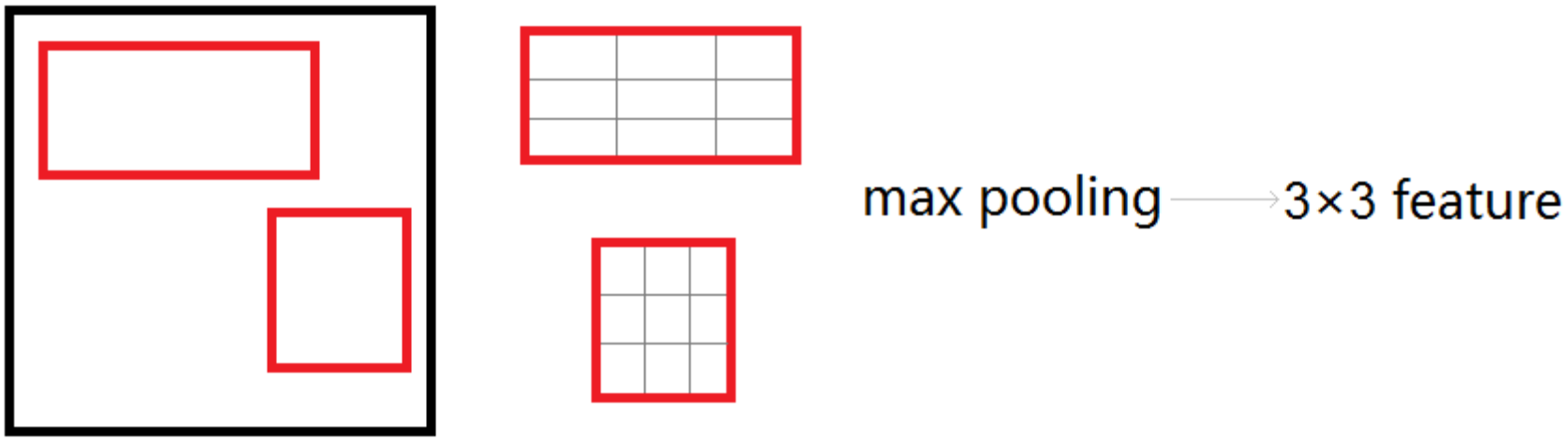
R-CNN系列

❑ 概述


Deep
ConvNet
RoI
projection
Conv
feature map
RoI
pooling
layer
FCs
Outputs:
softmax
bbox
regressor
FC
FC
RoI feature
vector
For each RoI

R-CNN系列

- ROI Pooling层：将每个候选区域均匀分成M×N块，对每块进行max pooling，将特征图上大小不一的候选区域转变为大小统一的数据，送入下一层



R-CNN系列

❑ 性能对比

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
SPPnet BB [11]†	07 \ diff	73.9	72.3	62.5	51.5	44.4	74.4	73.0	74.4	42.3	73.6	57.7	70.3	74.6	74.3	54.2	34.0	56.4	56.4	67.9	73.5	63.1
R-CNN BB [10]	07	73.4	77.0	63.4	45.4	**44.6**	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	**35.6**	66.8	67.2	70.4	**71.1**	66.0
FRCN [ours]	07	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
FRCN [ours]	07 \ diff	74.6	**79.0**	68.6	57.0	39.3	79.5	**78.6**	81.9	**48.0**	74.0	67.4	80.5	80.7	74.1	69.6	31.8	67.1	68.4	75.3	65.5	68.1
FRCN [ours]	07+12	**77.0**	78.1	**69.3**	**59.4**	38.3	**81.6**	**78.6**	**86.7**	42.8	**78.8**	**68.9**	**84.7**	**82.0**	**76.6**	**69.9**	31.8	**70.1**	**74.8**	**80.4**	70.4	**70.0**

Table 1. **VOC 2007 test** detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07 \ diff**: **07** without "difficult" examples, **07+12**: union of **07** and VOC12 trainval. †SPPnet results were prepared by the authors of [11].

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
SegDeepM	12+seg	**82.3**	75.2	67.1	50.7	**49.8**	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	**41.5**	**71.9**	62.2	73.2	**64.6**	67.2
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1
FRCN [ours]	07++12	82.0	**77.8**	**71.6**	**55.3**	42.4	**77.3**	**71.7**	**89.3**	**44.5**	**72.1**	**53.7**	**87.7**	**80.0**	**82.5**	**72.7**	36.6	68.7	**65.4**	**81.1**	62.7	**68.8**

Table 2. **VOC 2010 test** detection average precision (%). BabyLearning uses a network based on [17]. All other methods use VGG16. Training set key: **12**: VOC12 trainval, **Prop.**: proprietary dataset, **12+seg**: **12** with segmentation annotations, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval.

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	63.2
NUS_NIN_c2000	Unk.	80.2	73.8	61.9	43.7	**43.0**	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	**38.6**	**68.3**	58.0	68.7	63.3	63.8
R-CNN BB [10]	12	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3	62.4
FRCN [ours]	12	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
FRCN [ours]	07++12	**82.3**	**78.4**	**70.8**	**52.3**	38.7	**77.8**	**71.6**	**89.3**	**44.2**	**73.0**	**55.0**	**87.5**	**80.5**	**80.8**	**72.0**	35.1	**68.3**	**65.7**	**80.4**	**64.2**	**68.4**

Table 3. **VOC 2012 test** detection average precision (%). BabyLearning and NUS_NIN_c2000 use networks based on [17]. All other methods use VGG16. Training set key: see Table 2, **Unk.**: unknown.

R-CNN系列

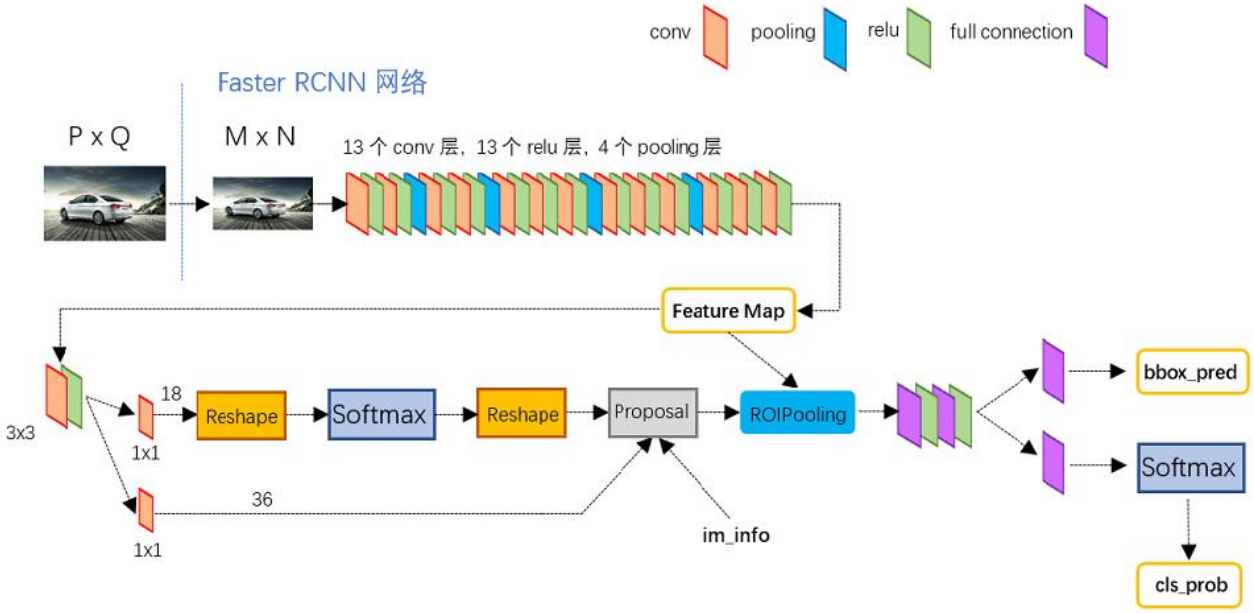
❑ 效率对比

- Fast R-CNN trains the very deep VGG16 network 9× faster than R-CNN, is 213× faster at test-time, and achieves a higher mAP on PASCAL VOC 2012.
- Compared to SPPnet, Fast R-CNN trains VGG16 3×faster, tests 10× faster, and is more accurate.

R-CNN系列

❑ Faster R-CNN

– RPN(Region Proposal Network)：使用全卷积神经网络来生成区域建议（Region proposal），替代之前的Selective search

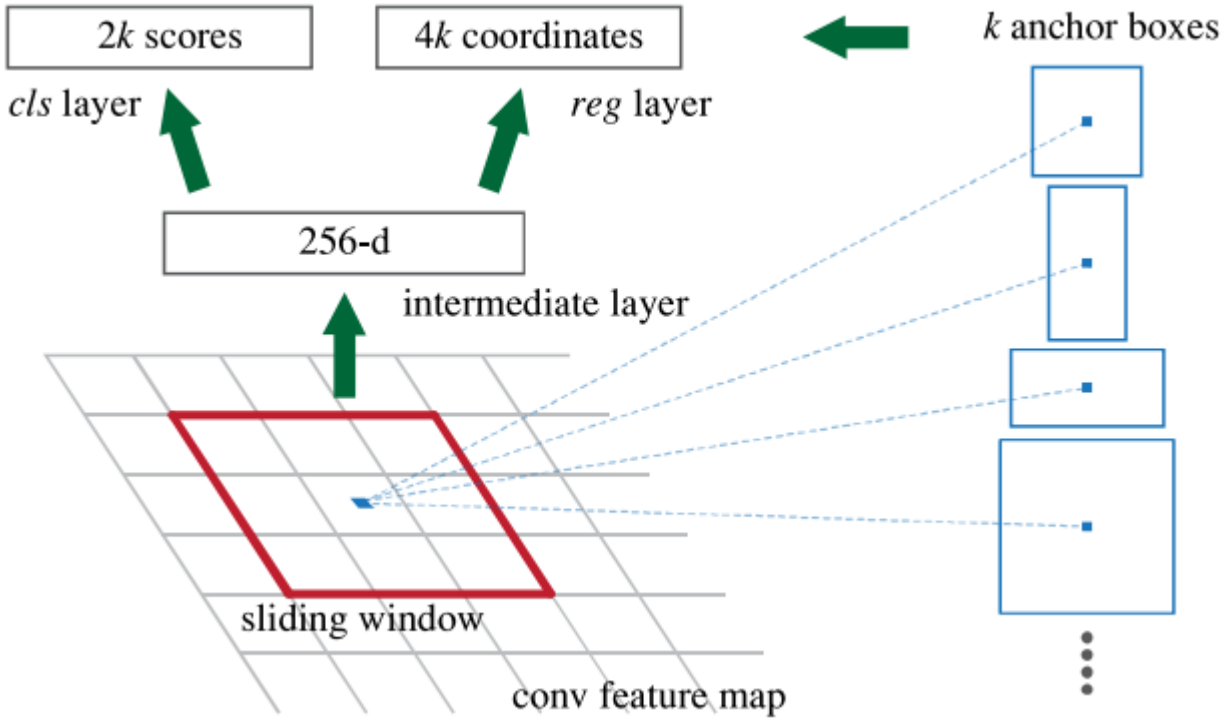


Shaoqing Ren, Kaiming He, Ross B. Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Trans. Pattern Anal. Mach. Intell. 39(6): 1137-1149 (2017)

R-CNN系列

❑ RPN

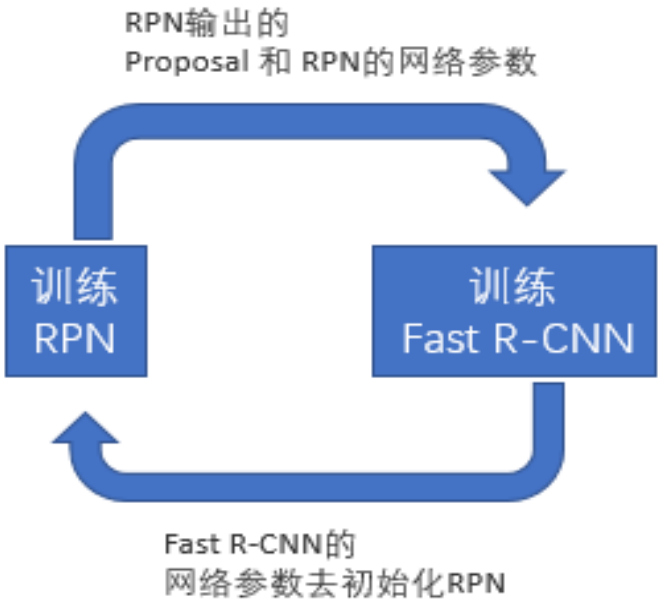
- Anchor内是某个object 的概率
- Anchor边界框回归输出(Δx, Δy, Δw, Δh)

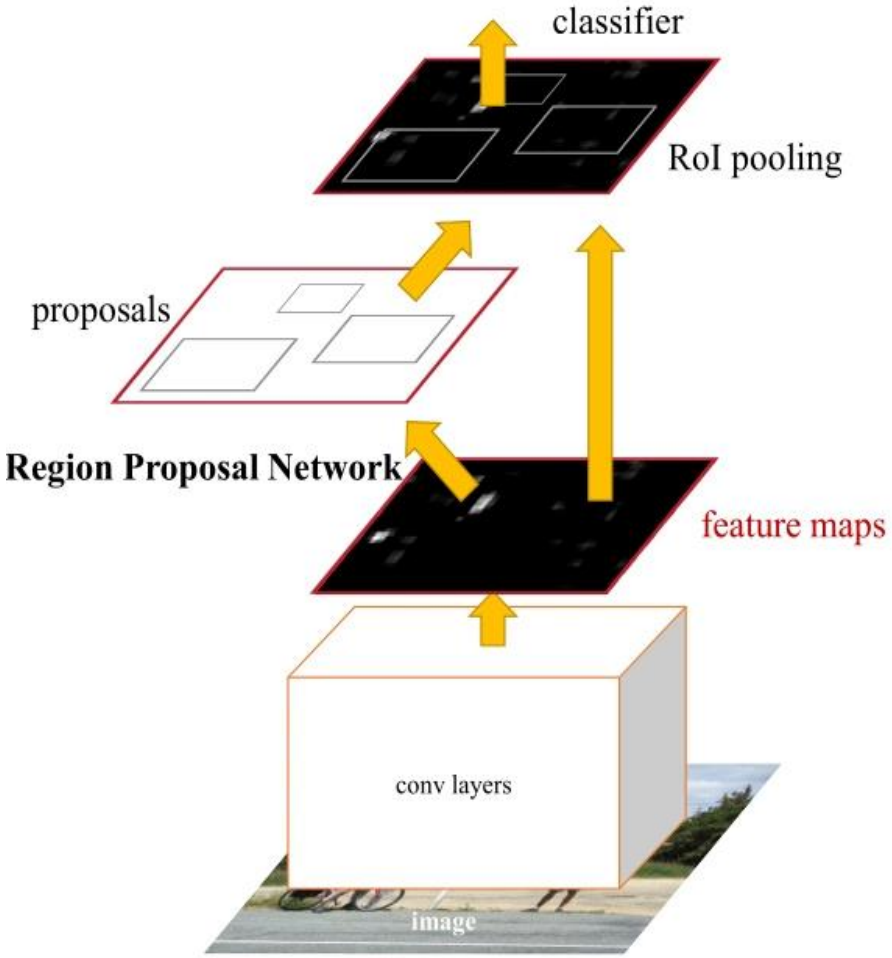


R-CNN系列

❑ Faster R-CNN训练方式

- Alternating training
- Approximate joint training





R-CNN系列

❑ 性能对比

– 200ms per image on the COCO dataset

TABLE 12
Object Detection Results (Percent) on the **MS COCO** Dataset

method	proposals	training data	COCO val		COCO test-dev	
			mAP@.5	mAP@	mAP@.5	mAP@
Fast R-CNN [2]	SS, 2000	COCO train	-	-	35.9	19.7
Fast R-CNN [impl. in this paper]	SS, 2000	COCO train	38.6	18.9	39.3	19.3
Faster R-CNN	RPN, 300	COCO train	41.5	21.2	42.1	21.5
Faster R-CNN	RPN, 300	COCO trainval	-	-	**42.7**	**21.9**

YOLO系列

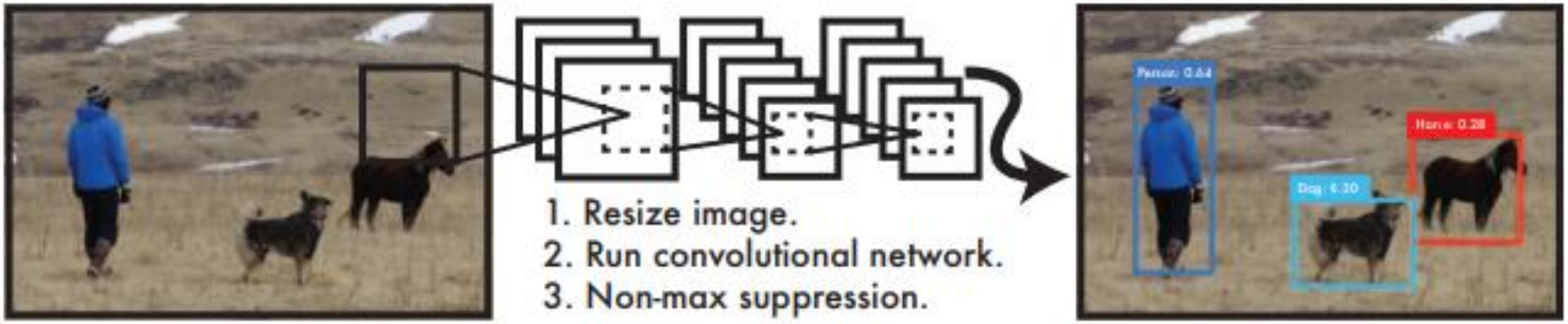
❑ YOLO

- 与R-CNN系列最大的区别是用一个卷积神经网络结构就可以从输入图像直接预测bounding box和类别概率，实现了End2End训练
- 速度非常快，实时性好
- 可以学到物体的全局信息，背景误检率比R-CNN降低一半，泛化能力强
- 准确率还不如R-CNN高，小物体检测效果较差

Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. CVPR 2016: 779-788

YOLO系列

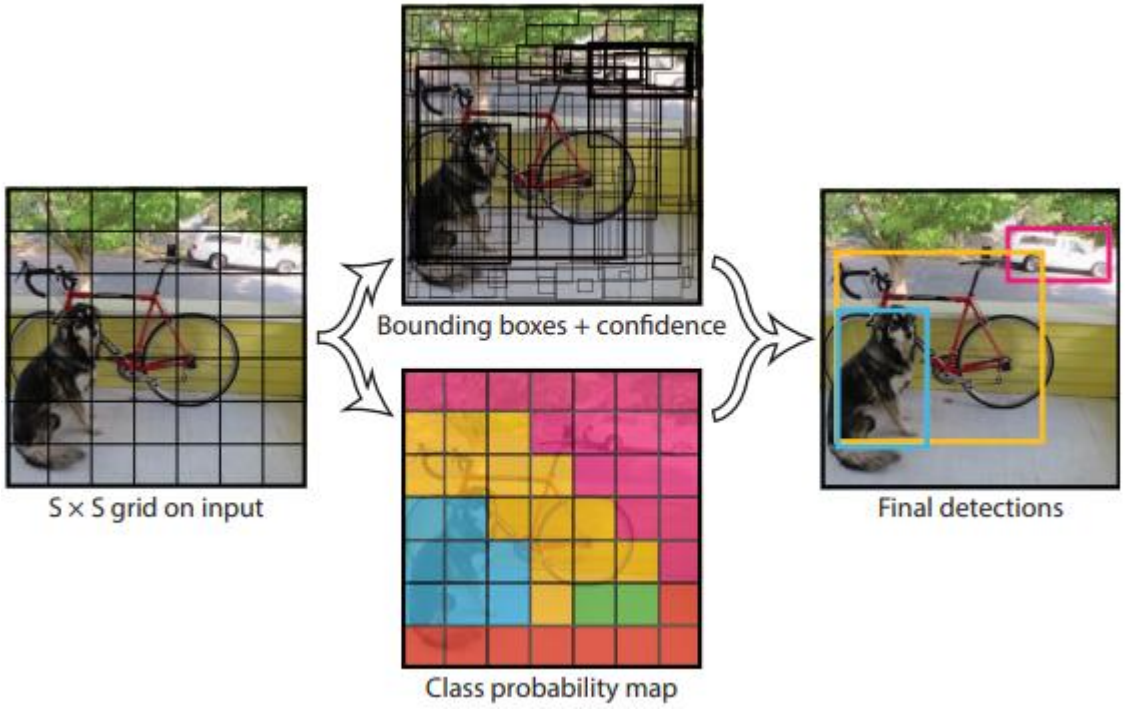
❑ 目标检测和识别



The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence

YOLO系列

❑ 目标检测和识别

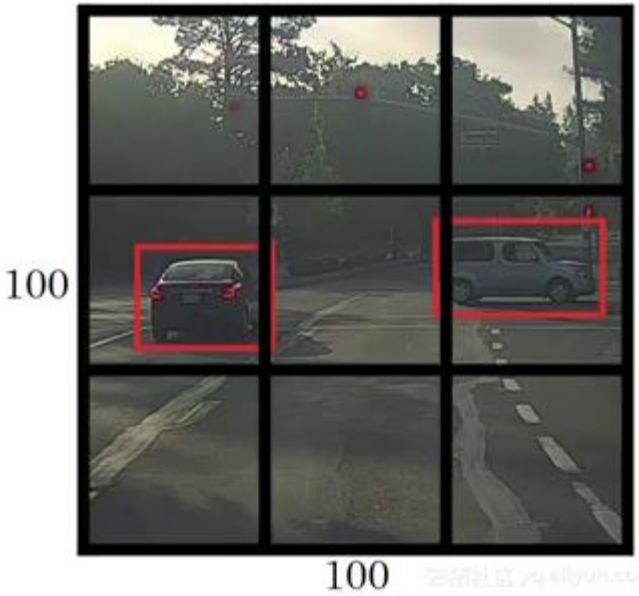


The Model. Our system models detection as a regression problem. It divides the image into an S×S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an S×S× (B∗5+ C) tensor.

YOLO系列

❑ 目标检测和识别

These predictions are encoded as an S×S× (B∗5+ C) tensor

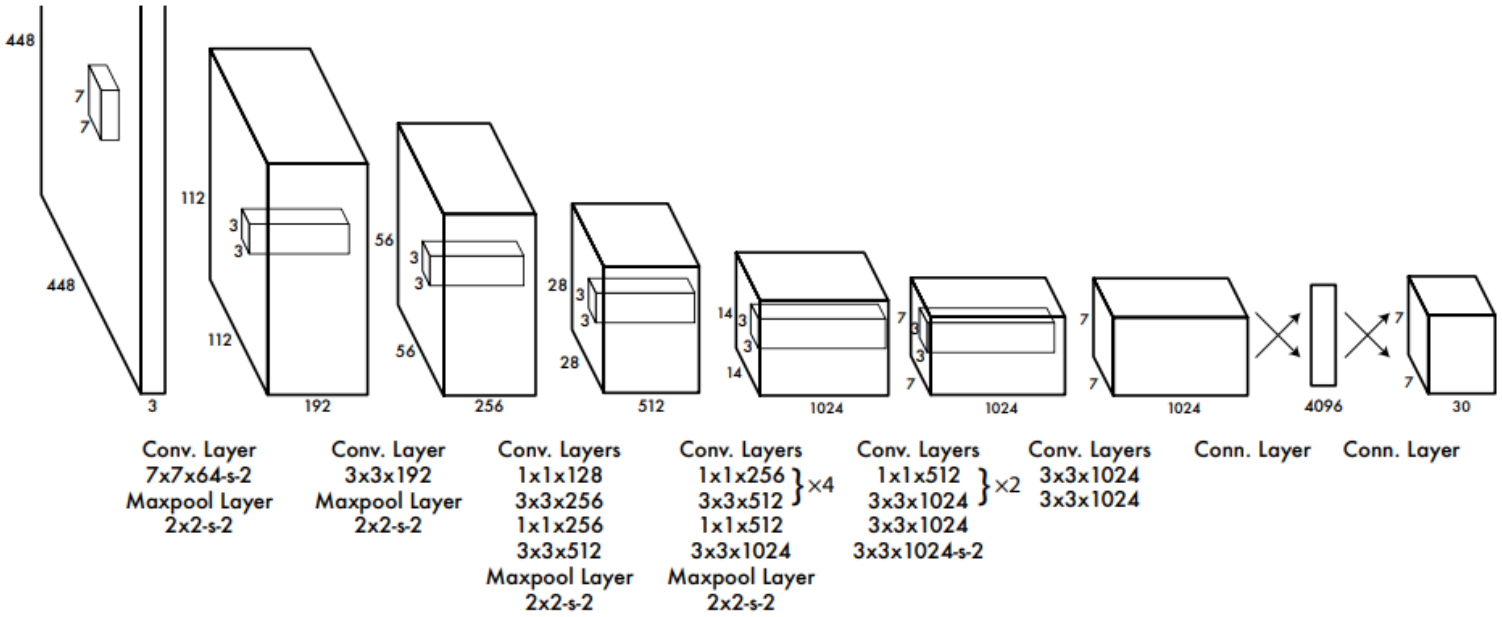


$$y = \begin{bmatrix} pc \\ bx \\ by \\ bh \\ bw \\ c1 \\ c2 \\ c3 \end{bmatrix}$$

YOLO系列

❑ 网络结构

– 24个卷积层和2个全连接层



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection

YOLO系列

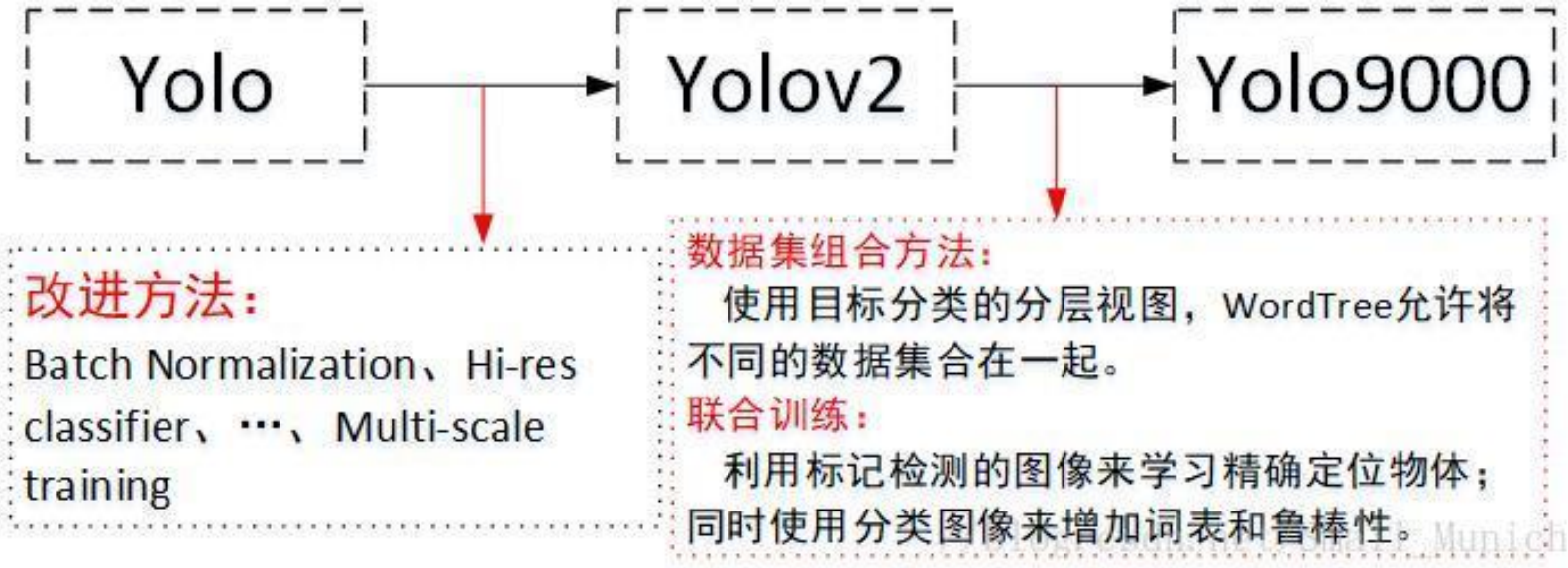
❑ 性能对比

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

YOLO系列

❑ YOLO2和YOLO9000



Joseph Redmon, Ali Farhadi. YOLO9000: Better, Faster, Stronger. CVPR 2017: 6517-6525

YOLO系列

❑ 性能分析

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

PASCAL VOC2012 test detection results

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN [5]	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster R-CNN [15]	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
YOLO [14]	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300 [11]	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD512 [11]	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
ResNet [6]	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLOv2 544	07++12	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7

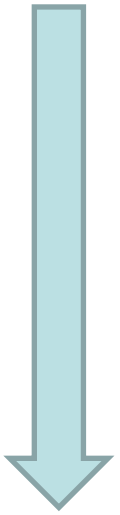
5

卷积神经网络的主要应用

卷积神经网络的主要应用

❑ 图像处理领域主要应用

- 图像分类（物体识别）：整幅图像的分类或识别
- 物体检测：检测图像中物体的位置进而识别物体
- 图像分割：对图像中的特定物体按边缘进行分割
- 图像回归：预测图像中物体组成部分的坐标

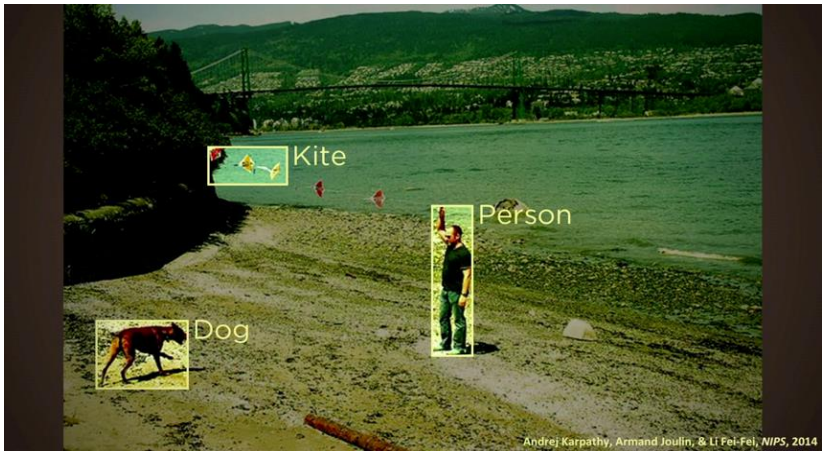


细化

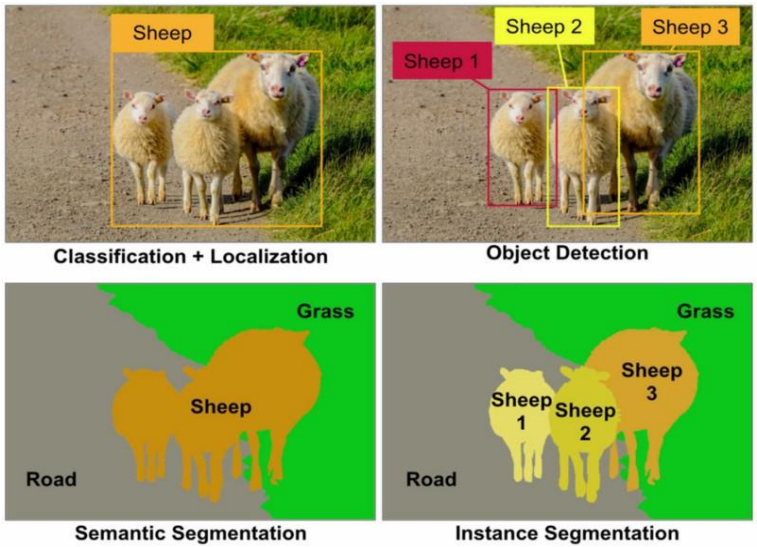
卷积神经网络的主要应用

物体识别

物体检测



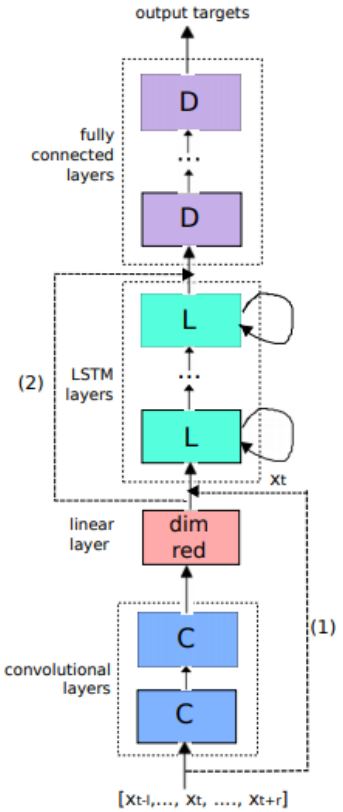
图像分割



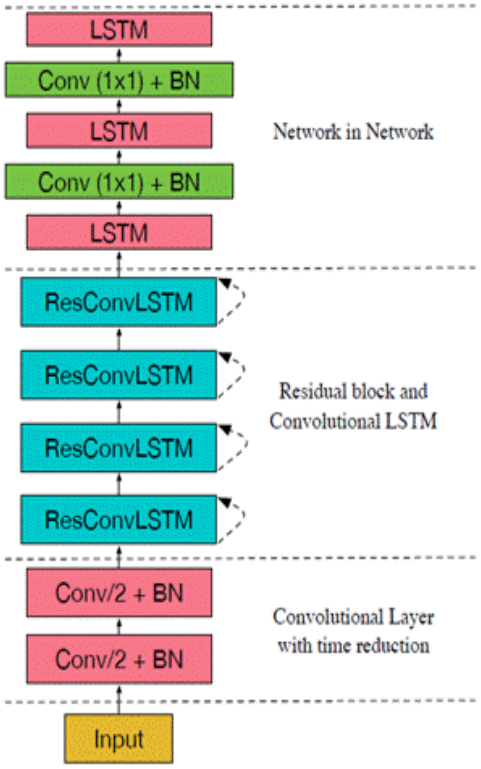
图像回归

卷积神经网络的主要应用

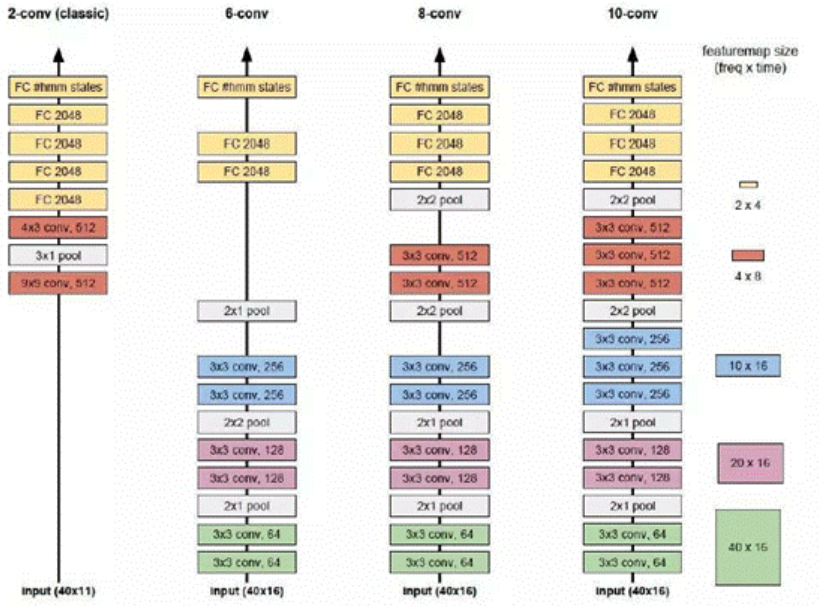
❑ 语音识别领域主要应用



CLDNN(Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks)



Google Deep CNN



IBM Deep CNN

卷积神经网络的主要应用

❑ 自然语言处理领域主要应用

- 情感分析：分析文本体现的情感（正负向、正负中或多态度类型）
- Yoon Kim: Convolutional Neural Networks for Sentence

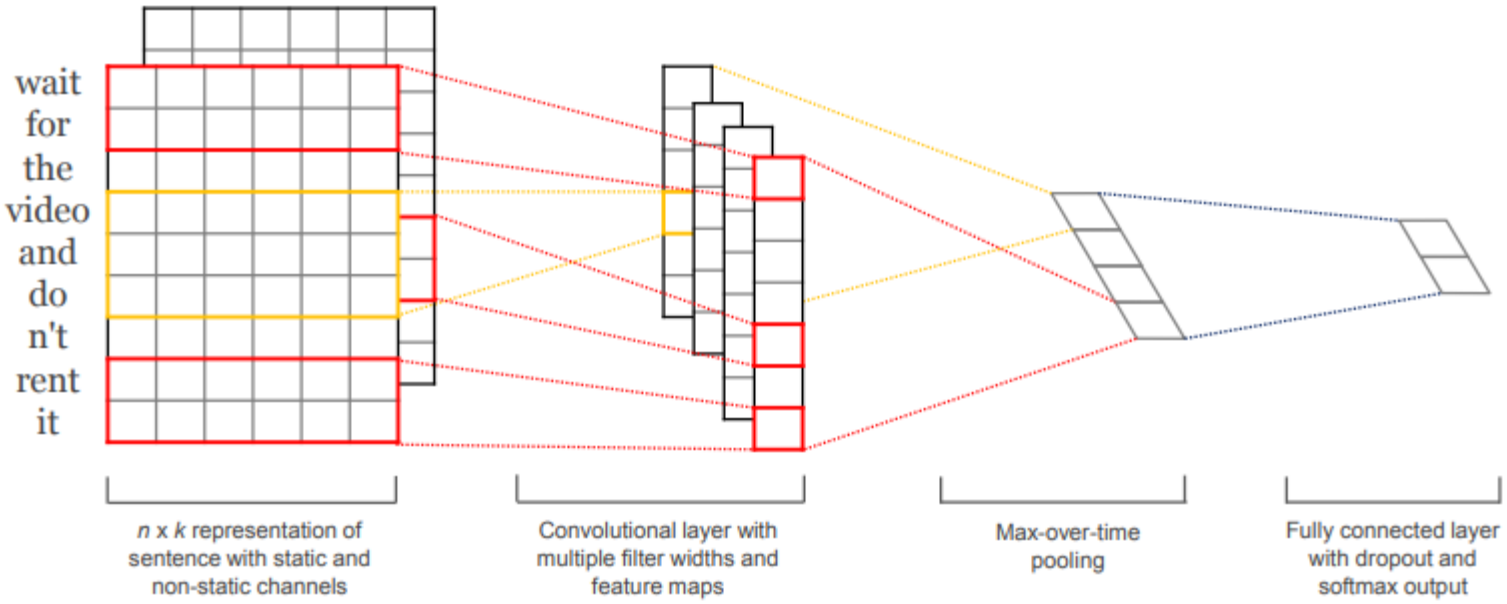


Figure 1: Model architecture with two channels for an example sentence.

6 中英文术语对照

中英文术语对照

- 卷积：Convolution
- 神经认知机：Neocognitron
- 感受野：Receptive field
- 感光细胞：Photoreceptor cell
- 水平细胞：Horizontal cell
- 双极细胞：Bipolar cell
- 神经节细胞：Ganglion cell
- 示波器：Oscilloscope
- 电极：Electrode
- 视网膜：Retina
- 视觉皮层：Visual cortex
- 视神经：Optic nerve

中英文术语对照

- 外侧膝状体：Lateral Geniculate Nucleus
- 卷积神经网络：Convolutional Neural Network
- 卷积核：Convolutional kernel
- 池化：Pooling
- 池化核：Pooling kernel
- 零填充：Zero-padding
- 特征图：Feature map
- 步幅：Stride
- 降采样：Down sampling
- 最大池化：Max-pooling
- 均值池化：Average-pooling
- 残差神经网络：Residual Neural Network

中英文术语对照

- 残差块：Residual block
- 跳跃连接：Skip connection
- 径向基函数：Radial Basis Function，RBF
- 区域CNN: Region-CNN
- 选择性搜索：Selective Search
- 区域建议网络：Region Proposal Network，RPN
- 边框回归：Bounding Box Regression

谢谢！