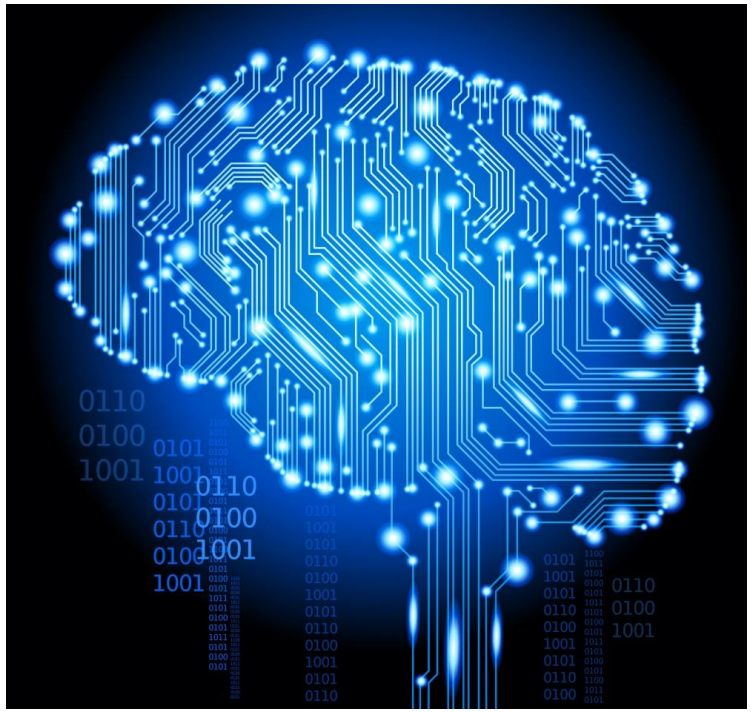
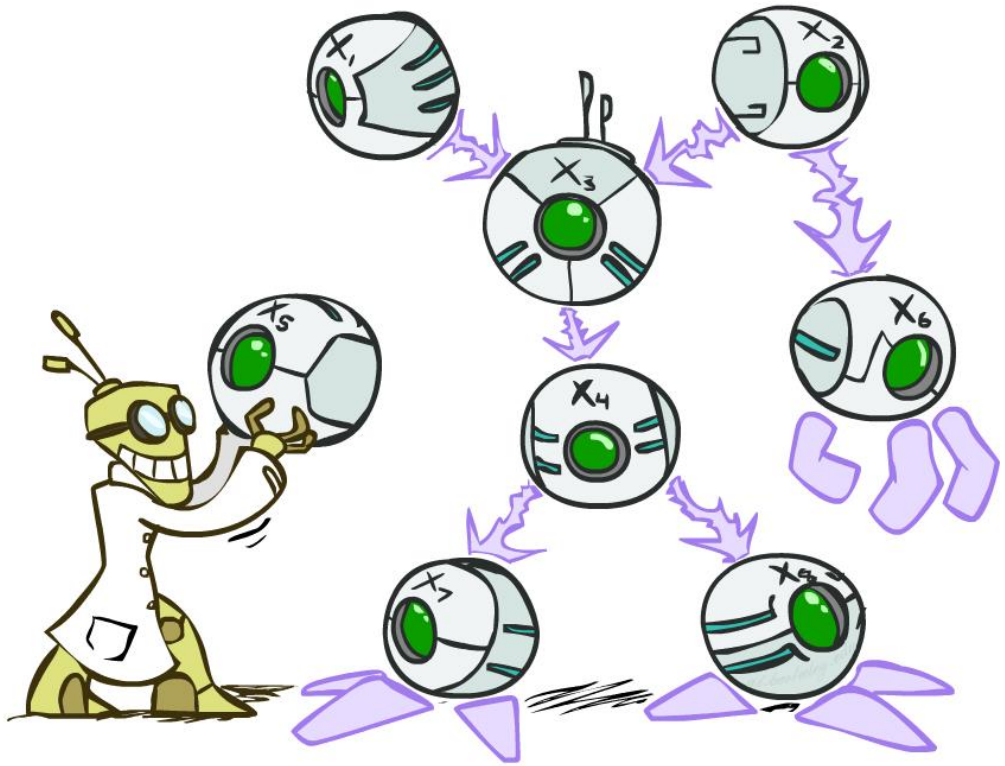
图像数据的深度学习模型

中国科学院自动化研究所
吴高巍
gaowei.wu@ia.ac.cn
2022-9-22



内容

- 卷积神经网络（Convolutional Neural Network, CNN）
- CNN实例
- 图像数据应用



卷积神经网络

——Convolutional Neural Networks, CNN

图像数据

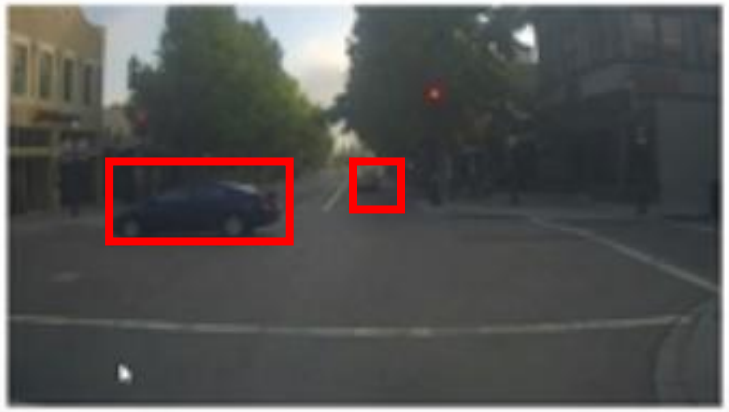
计算机视觉的应用

Image Classification

64x64

Cat? (0/1)

Object detection



Neural Style Transfer

Deep Learning on large images

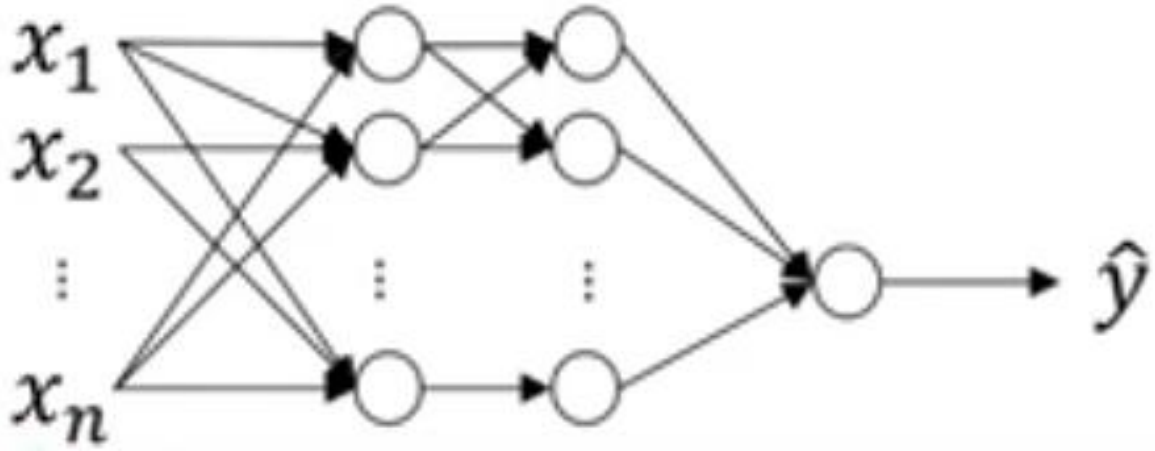
$64 \times 64 \times 3$

Cat? (0/1)

12288

$1000 \times 1000 \times 3$

3Million



Convolutional Neural Networks卷积神经网络

- 卷积神经网络是一种特殊的深层神经网络模型
  - 它的神经元间的连接是非全连接的
  - 同一层中某些神经元之间的连接的权重是共享的（即相同的）。

- 20世纪60年代，Hubel和Wiesel研究猫脑皮层
  - 用于局部敏感和方向选择的神经元，其独特的网络结构可以有效地降低反馈神经网络的复杂性

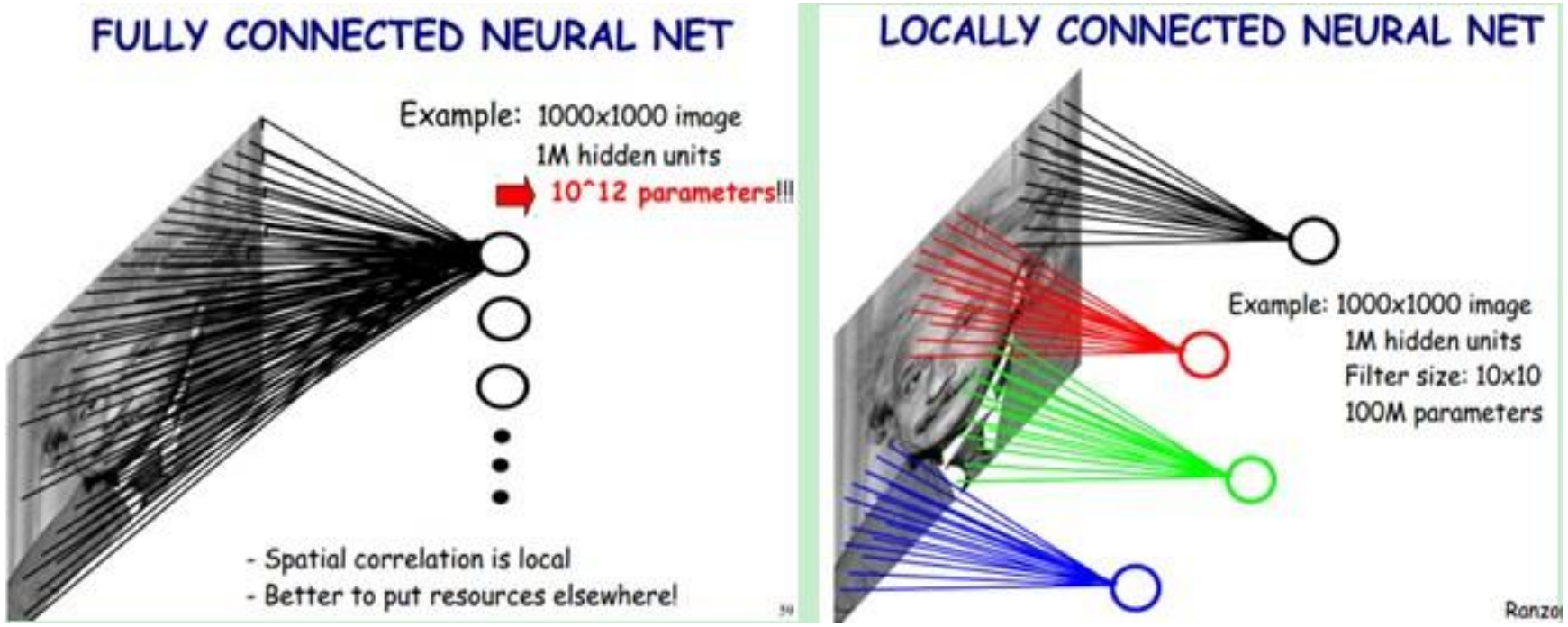
Hubel-Wiesel结构

- 基于猫的初级视皮层(VI区)的研究。
  - 简单细胞
  - 复杂细胞

- 两层神经网络模拟初级视皮层中的简单细胞和复杂细胞
  - 每层的神经元被组织成二维平面
  - “简单细胞”层提取其输入中的局部特征
  - “复杂细胞”层组合“简单细胞”层中相应的子区域，使得整个网络对局部变换具有一定的不变性。

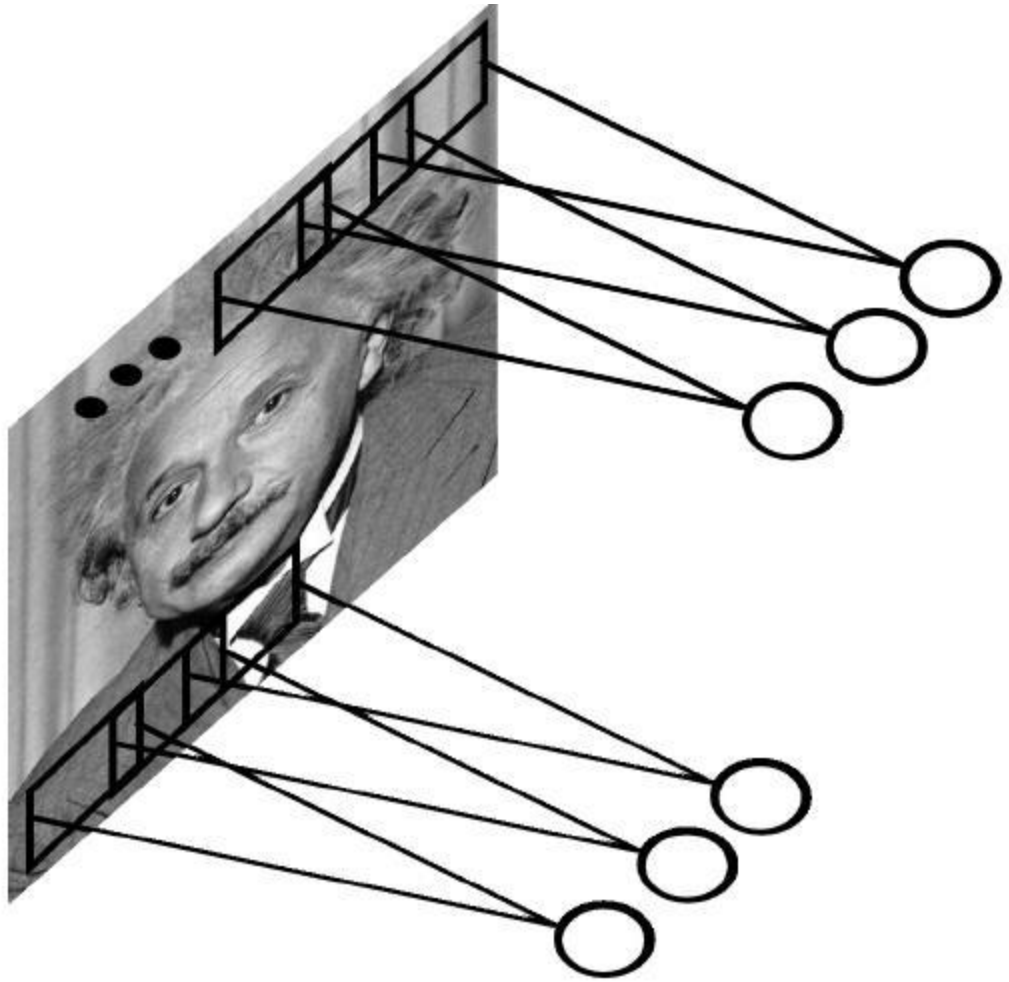
局部连接

- 局部感知野
  - 图像的空间联系也是局部的像素联系较为紧密，而距离较远的像素相关性则较弱。
  - 减少了需要训练的权值数目



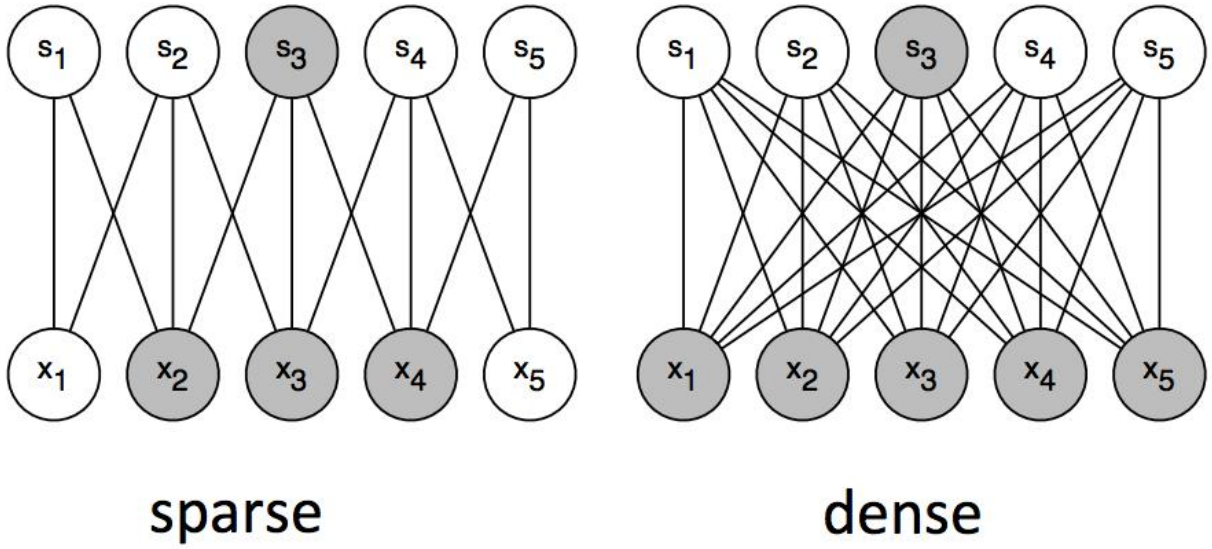
局部连接

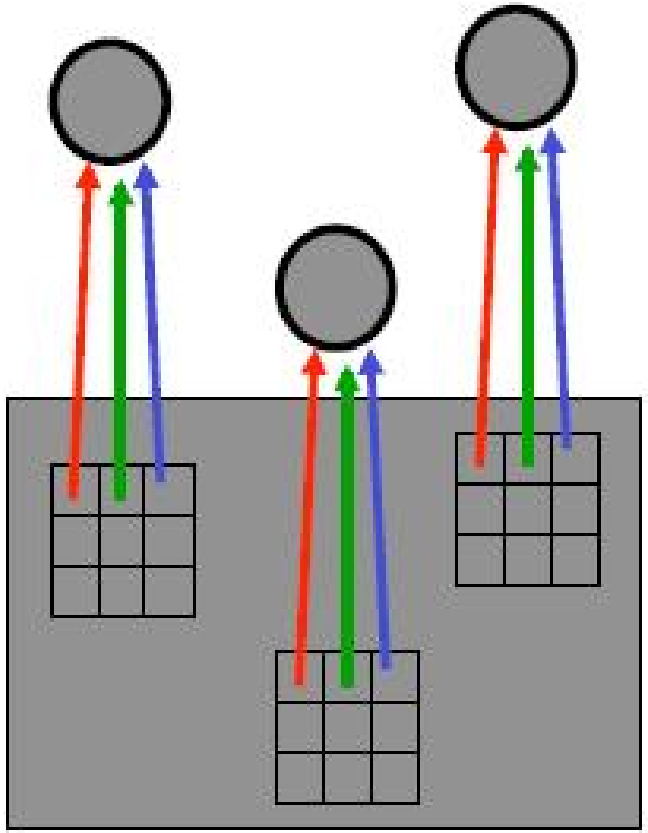
- 参数共享
  - 图像的一部分的统计特性与其他部分是一样的。
  - 在输入的不同位置检测同一种特征
  - 平移不变性



卷积

- 稀疏连接
- 参数共享

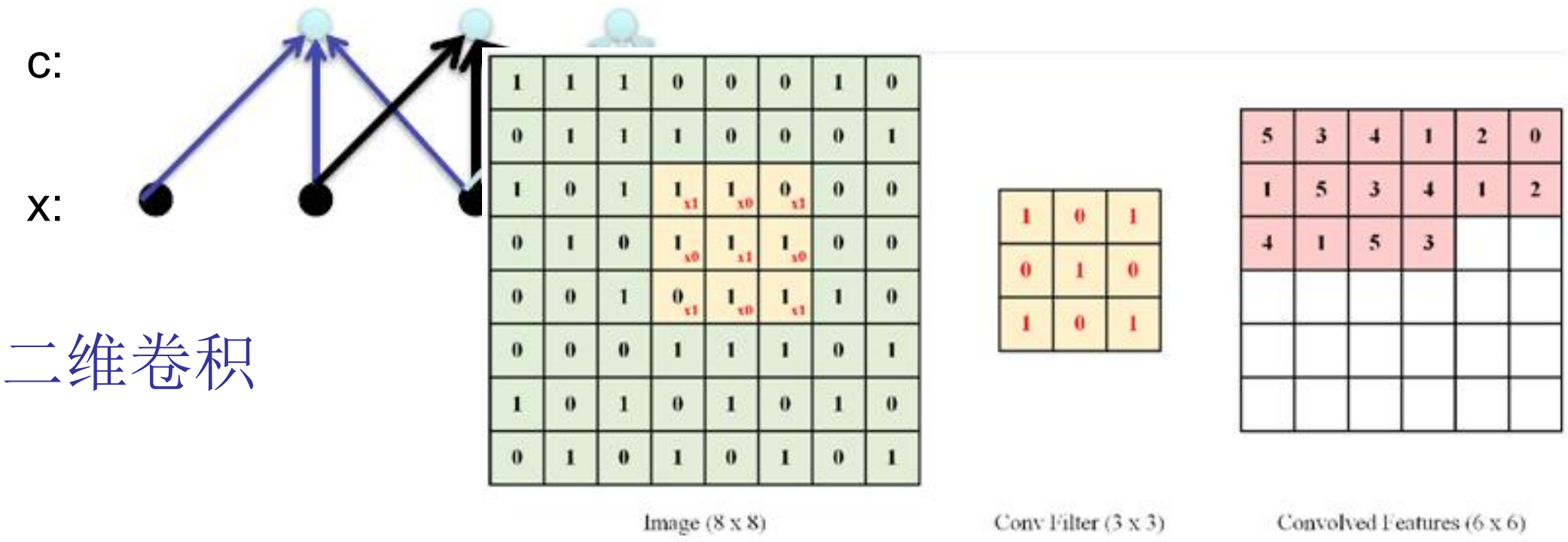




Convolution卷积

- 一维卷积

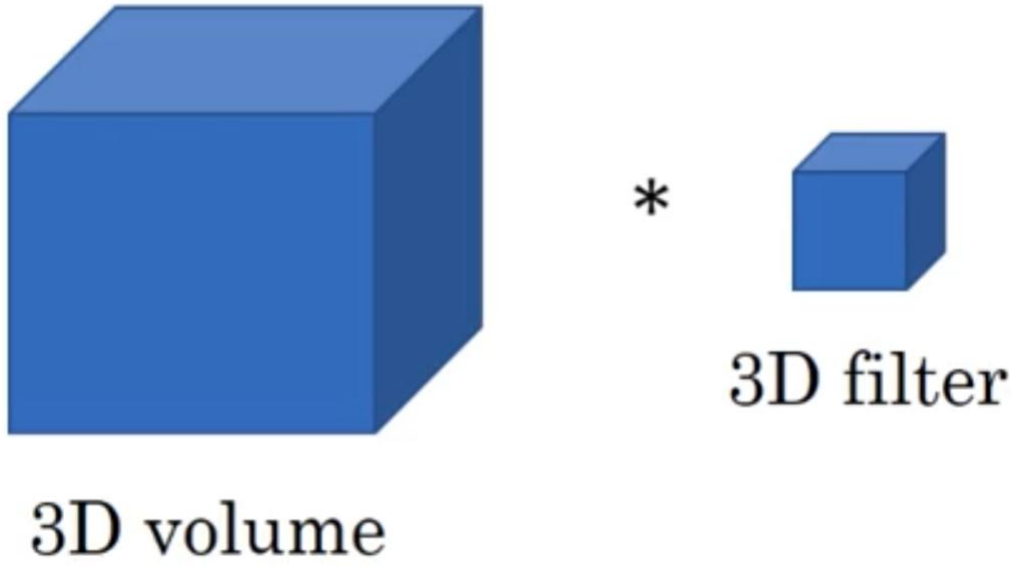
$$c_j = \mathbf{m}^T \mathbf{x}_{j-|\mathbf{m}|+1:j}$$



- 二维卷积

Convolution卷积

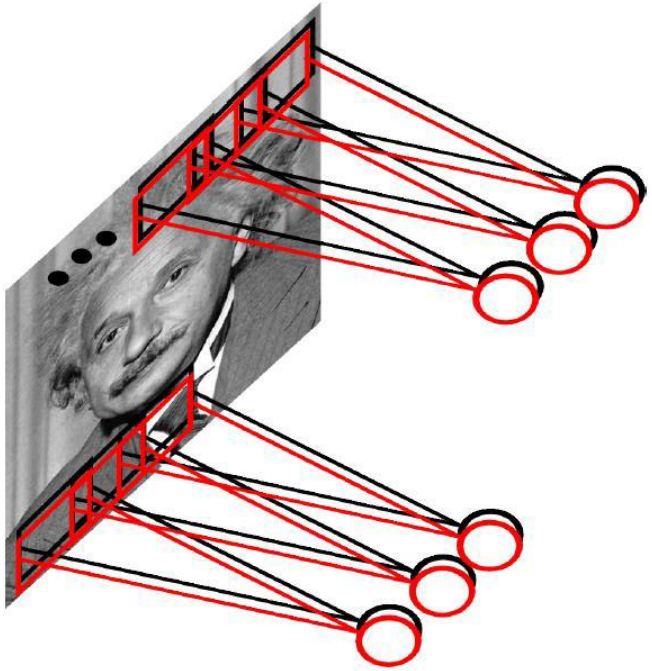
- 三维卷积

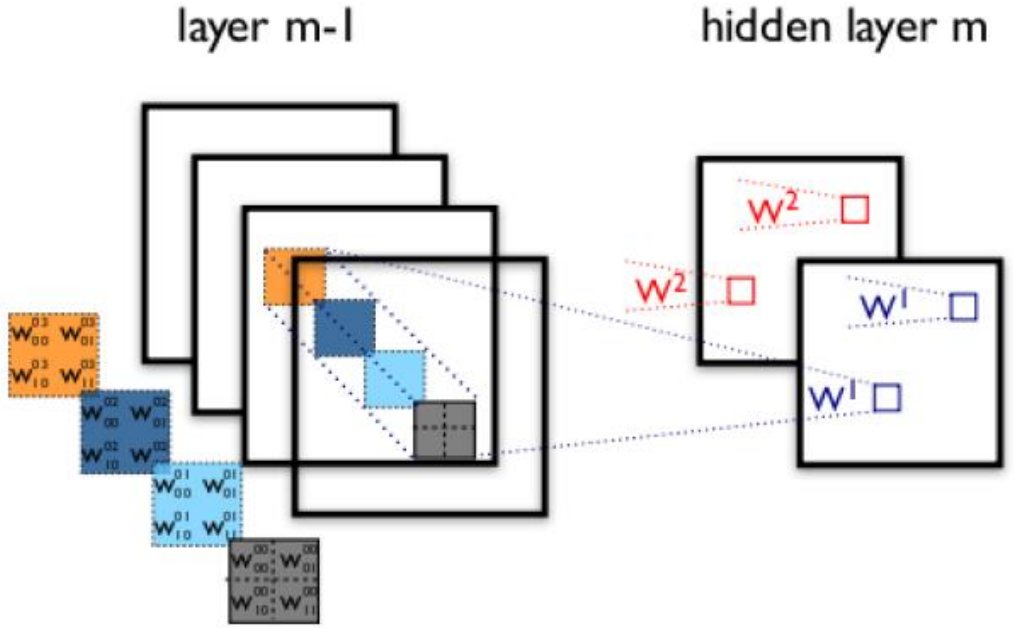


  - 假设输入数据的大小为a1×a2×a3，过滤器大小为f，即过滤器维度为f×f×f。
  - 三维卷积最终的输出为(a1−f+1)×(a2−f+1)×(a3−f+1)

多卷积核

- 每个卷积核都会将图像生成为另一幅图像。
  - 两个卷积核就可以生成两幅图像，这两幅图像可以看做是一张图像的不同的通道。

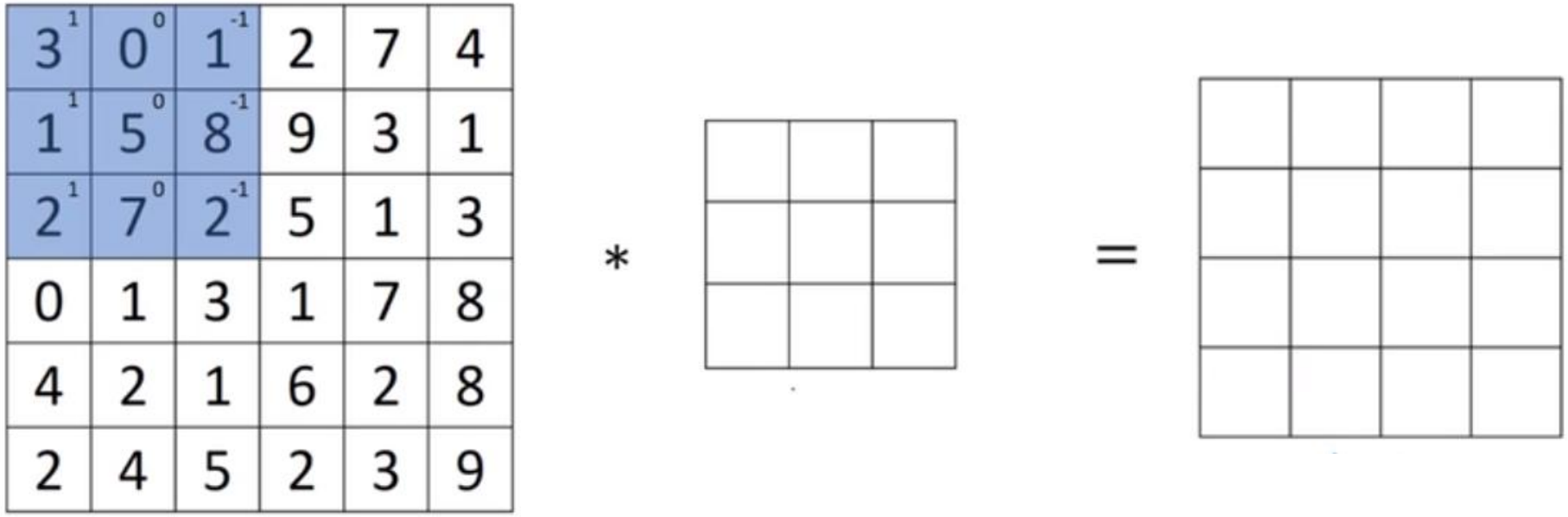




由4个通道卷积得到2个通道的过程

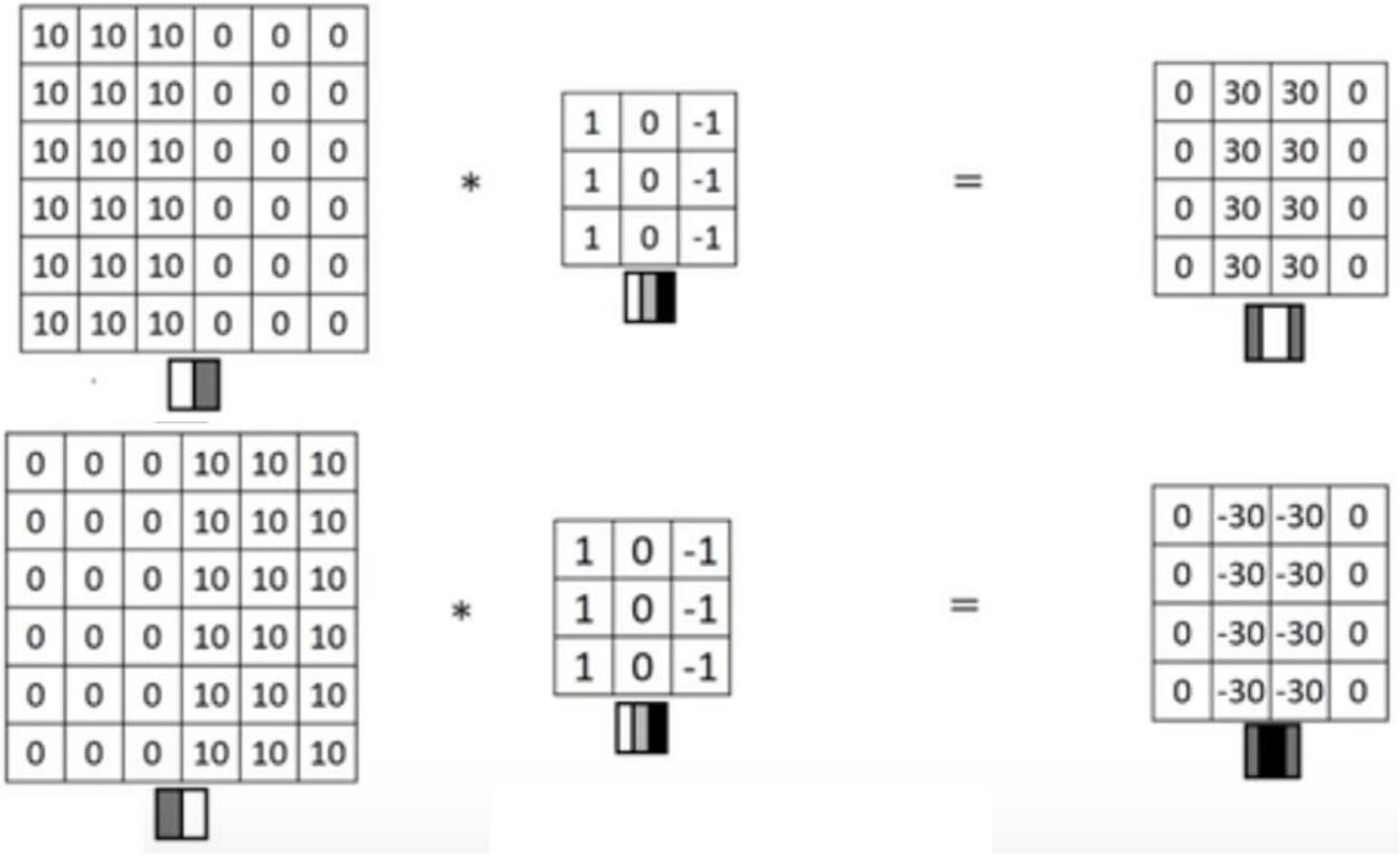
边缘检测示例

- 卷积运算是输入图像与过滤器（也叫核）进行的运算，得到输出图像。
  - 卷积核与图像对应的位置相乘求和得到一个新值



边缘检测示例

- Vertical edge detection

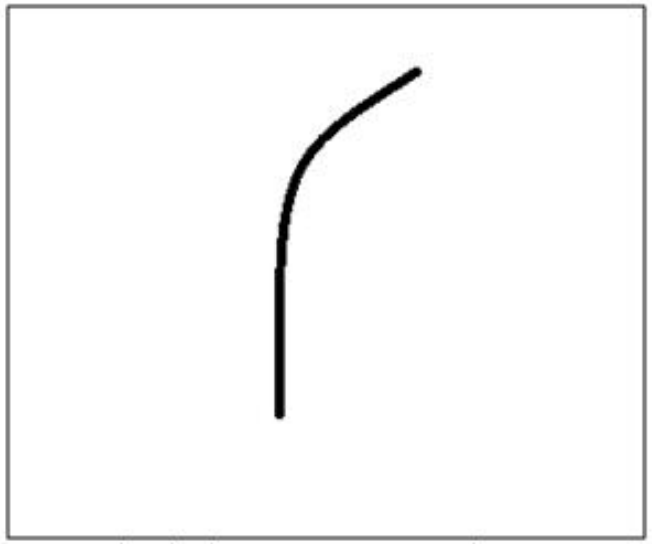


卷积Convolution

- 假定要识别图像中的特定曲线，也就是说，对这种曲线有很高的输出，对其他形状则输出很低，
- 这也就像是神经元的激活

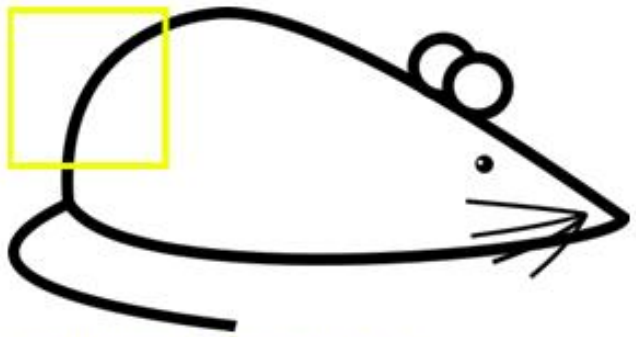
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

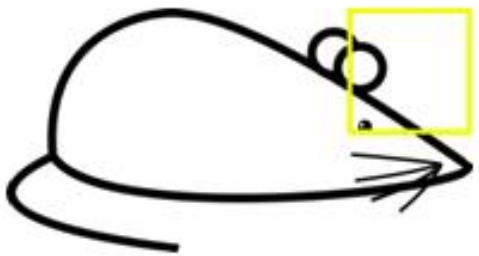
Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

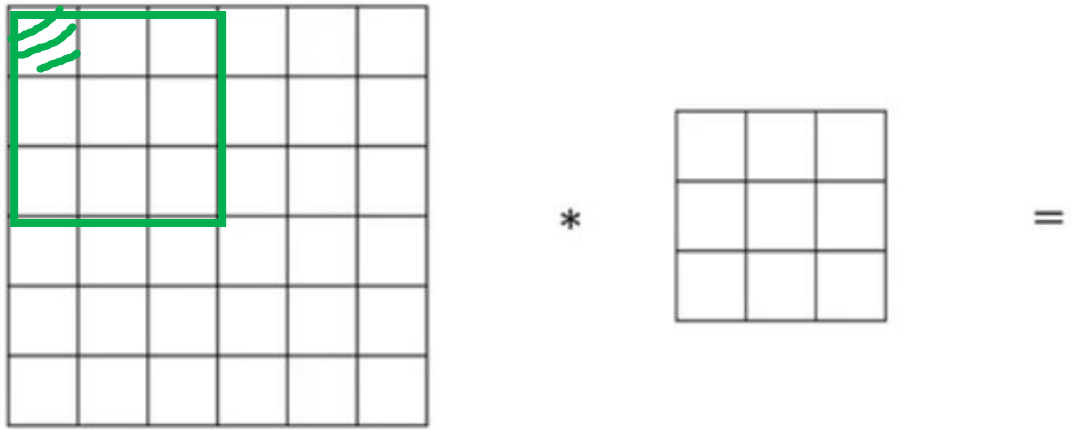
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

Padding

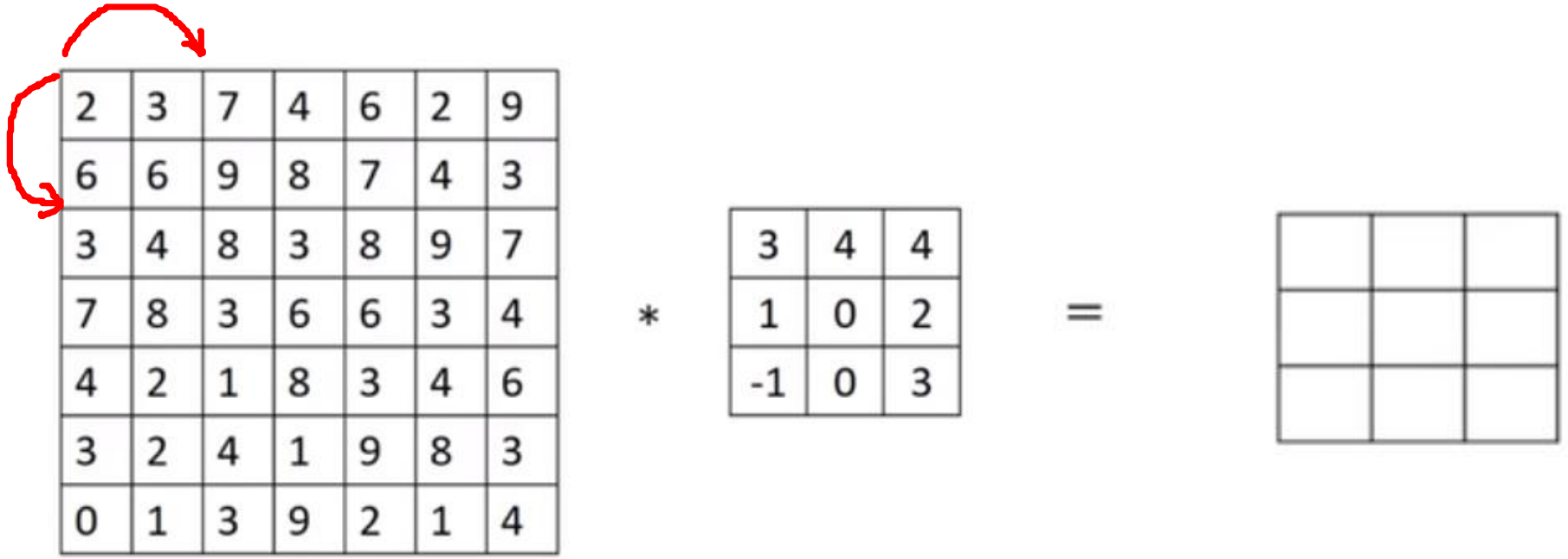
- 边缘不填充
  - 随着不断卷积，图像会变得越来越小，有时你可不想让它变小
  - 最角落的点只被使用了一次，这意味着在下传的过程中丢掉了图像边缘位置的信息。



  - 在步长为1时有公式：n+2p-f+1为输出的尺寸

卷积步长

- 卷积中的步幅是另一个构建卷积神经网络的基本操作



步长为2

- 输入与输出的尺寸关系

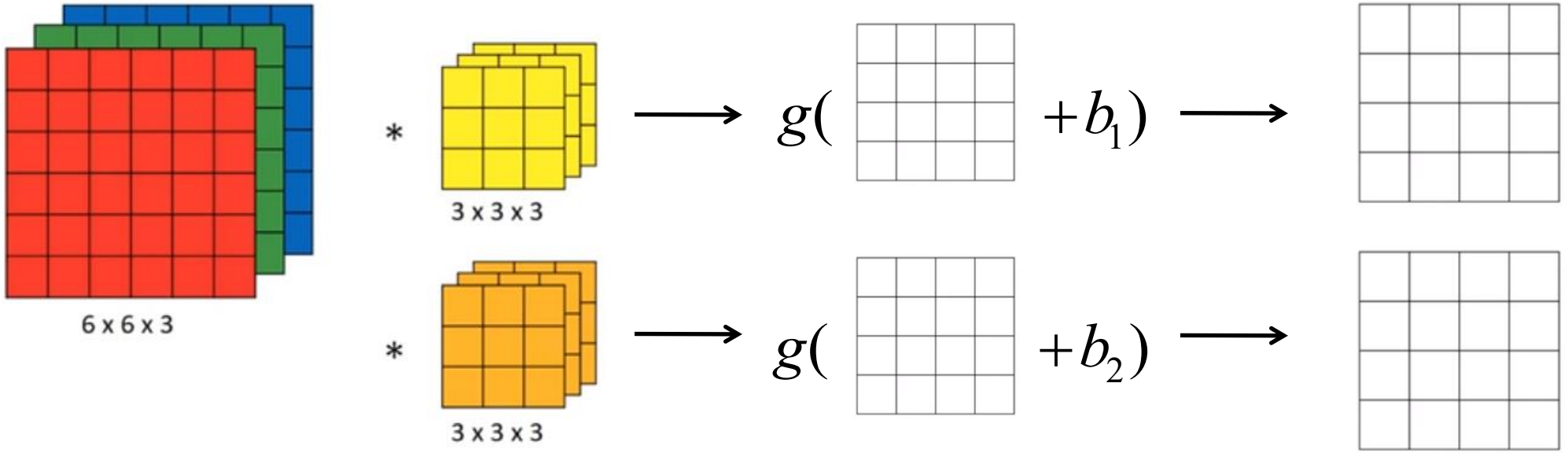
$n \times n$ image　　$f \times f$ filter

padding p　　stride s

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

单层卷积网络

- 每一个卷积核的输出对应一个实数b（偏差），然后在进行激活函数的非线性转换得到输出



$$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

- 参数数量？

卷积层

- If layer l is a convolution layer:

$$f^{[l]} = \text{filter size}$$
$$p^{[l]} = \text{padding}$$
$$s^{[l]} = \text{stride}$$

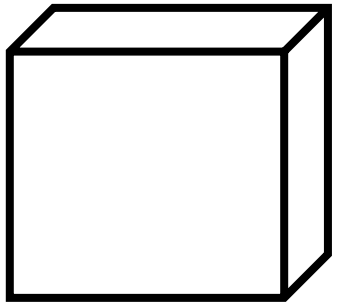
$$\text{Input}: \quad n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l-1]}$$

$$\text{Output}: n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$$

$$n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

- 卷积核 $f^{[l]} \times f^{[l]} \times n_C^{[l-1]}$
- 权值 $f^{[l]} \times f^{[l]} \times n_C^{[l-1]} \times n_C^{[l]}$
- 偏置 $n_c^{[l]}$

简单示例



$39\times39\times3$

$n_H^{[0]}=n_W^{[0]}=39$

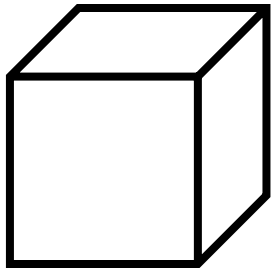
$n_c^{[0]}=3$

$\overrightarrow{f^{[1]}}=3$

$s^{[1]}=1$

$p^{[1]}=0$

10filter



$37\times37\times10$

$n_H^{[1]}=n_W^{[1]}=37$

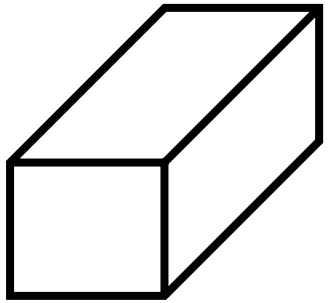
$n_c^{[1]}=10$

$\overrightarrow{f^{[2]}}=5$

$s^{[2]}=2$

$p^{[2]}=0$

20filter



$17\times17\times20$

$n_H^{[2]}=n_W^{[2]}=17$

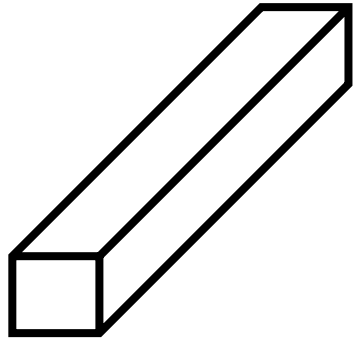
$n_c^{[2]}=20$

$\overrightarrow{f^{[3]}}=5$

$s^{[3]}=2$

$p^{[3]}=0$

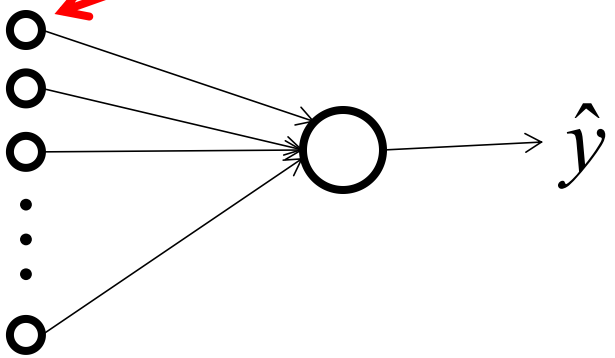
40filter



$7\times7\times40$

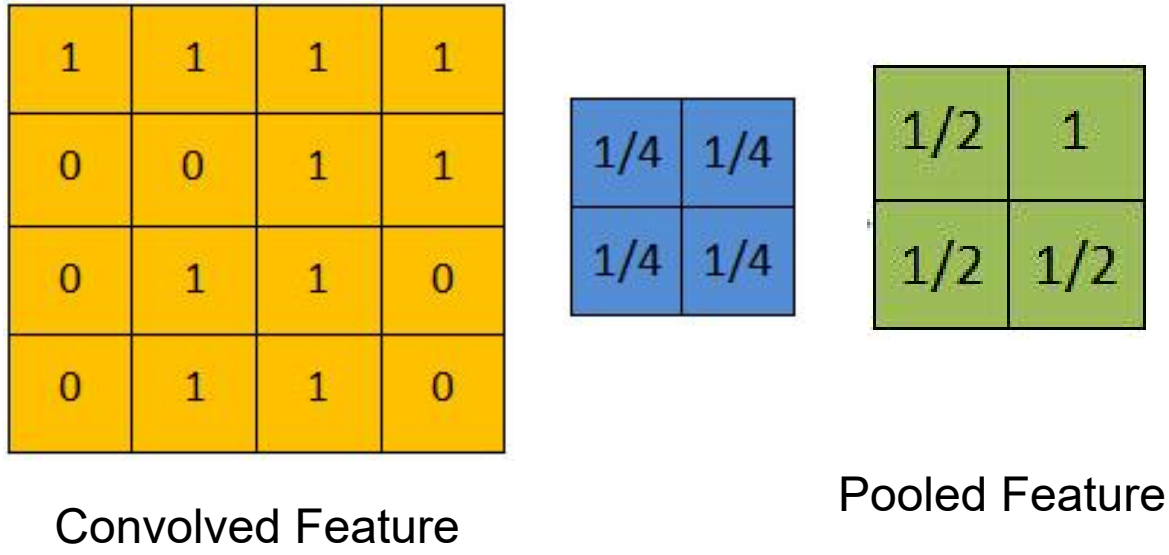
$n_H^{[3]}=n_W^{[3]}=7$

$n_c^{[3]}=40$



Pooling池化

- 通过卷积获得了特征之后，下一步利用这些特征去做分类。
  - 使用卷积时是利用了图像的“静态”特征
  - Pooling, 对不同位置的特征进行聚合统计
- 子采样
  - Average pool $\frac{1}{s^2}\sum x_i$
  - Max pool $\max\{x_i\}$
  - L2 pool $\sqrt{\frac{1}{s^2}\sum x_i^2}$



Pooling layer

- Max pooling

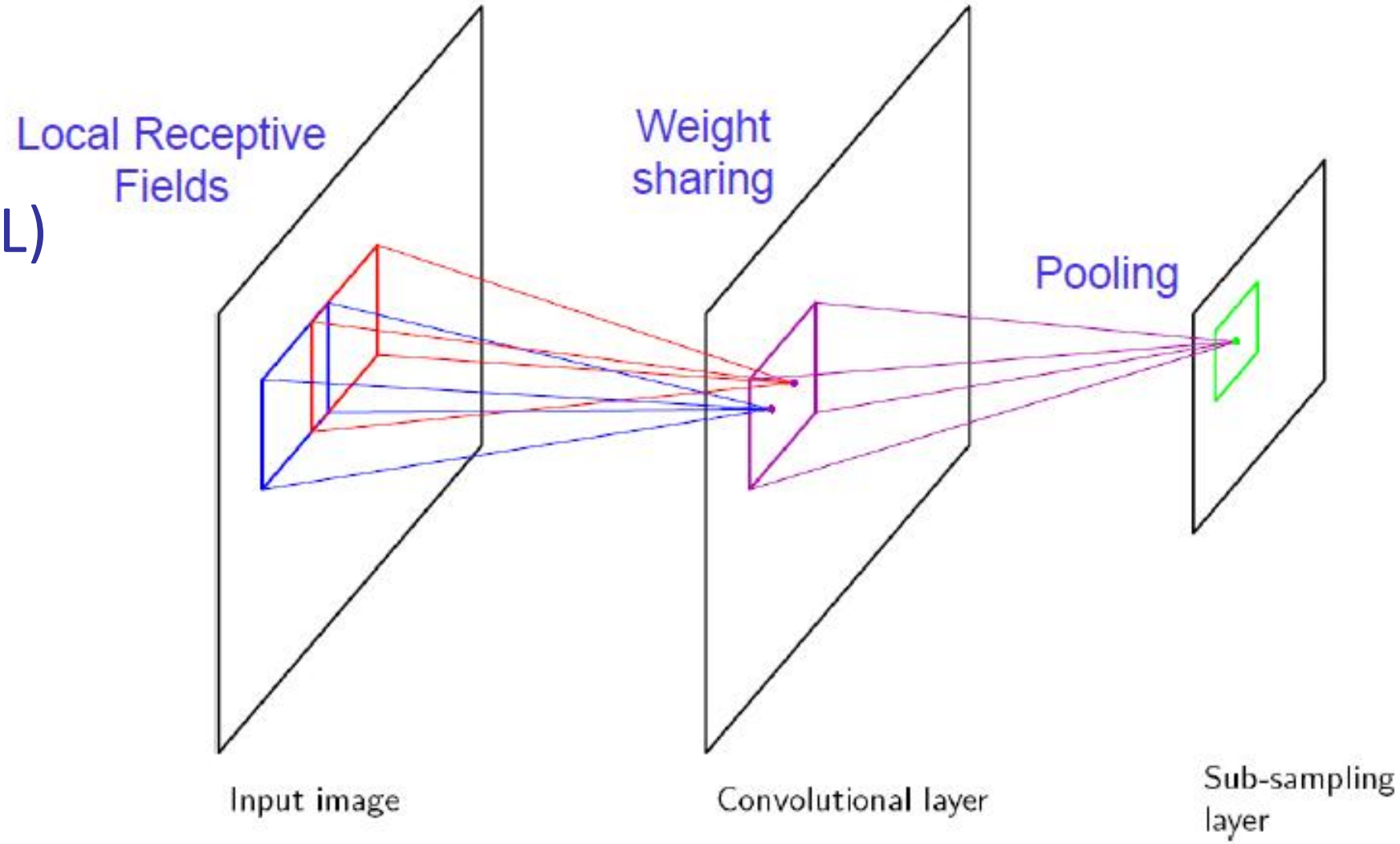
- Average pooling

Pooling

- 池化层中没有需要学习的参数，所以通常不把池化层当做独立的一层来看。
- 池化层是一般不会设置padding，即一般padding为0。
- filter size为2，stride为2是最常见的参数设置，尺寸图像缩小为原来的一半。
- 卷积时用的尺寸计算公式同样适用于池化层。

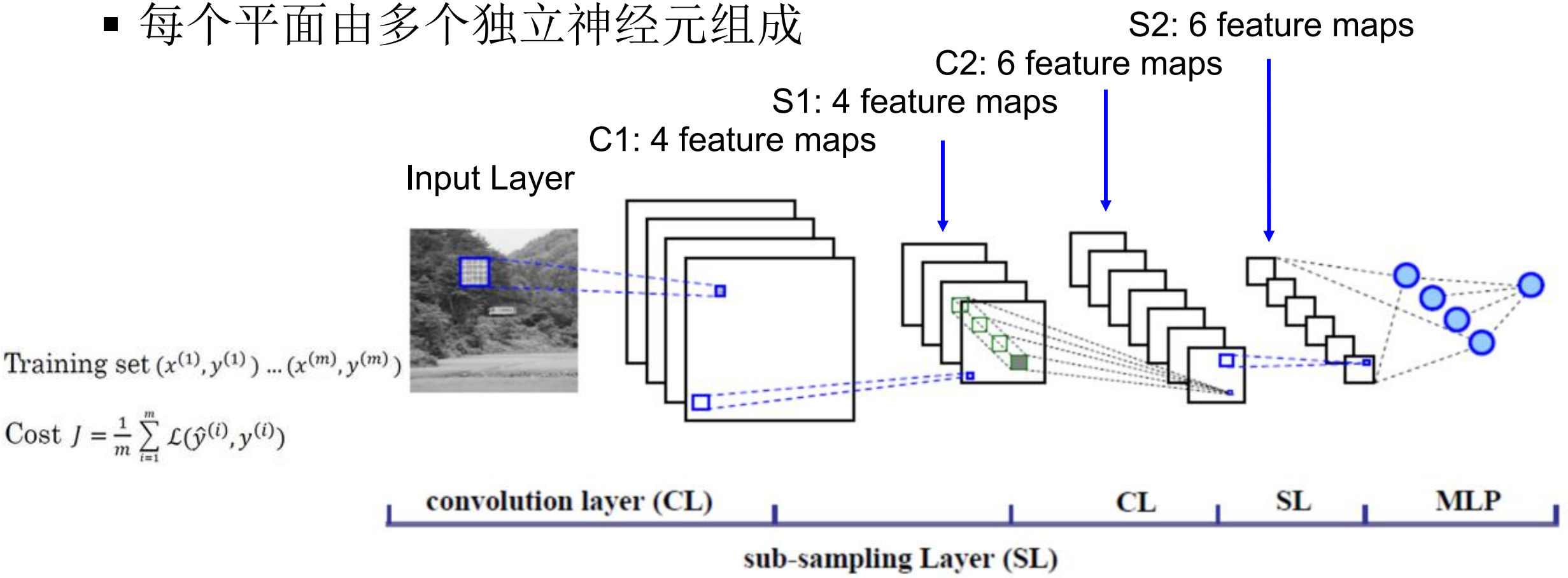
CNN基本结构

- 卷积层(conv)
- 子采样层(POOL)



CNN结构

- 卷积神经网络是一个多层的神经网络
  - 每层由多个二维平面组成
  - 每个平面由多个独立神经元组成



CNN训练过程

- 监督训练
- Bp算法
  - 向前传播

从样本集中取一个样本(Xp,Yp)，将X输入网络

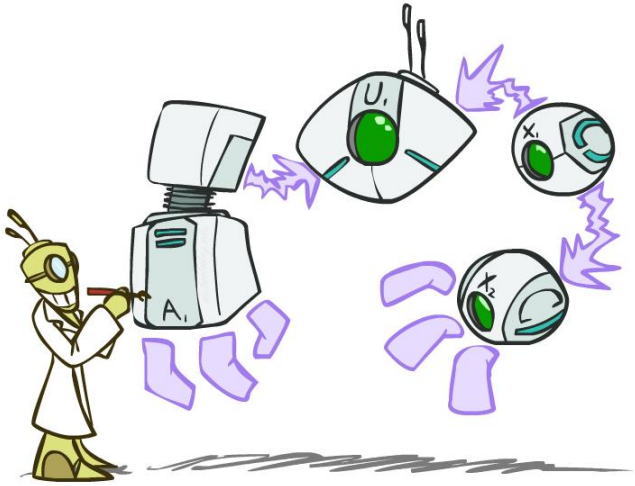
计算相应的实际输出Op

$Op=Fn(...(F_2(F_1(XpW^{(1)})W^{(2)})...)W^{(n)})$

  - 向后传播

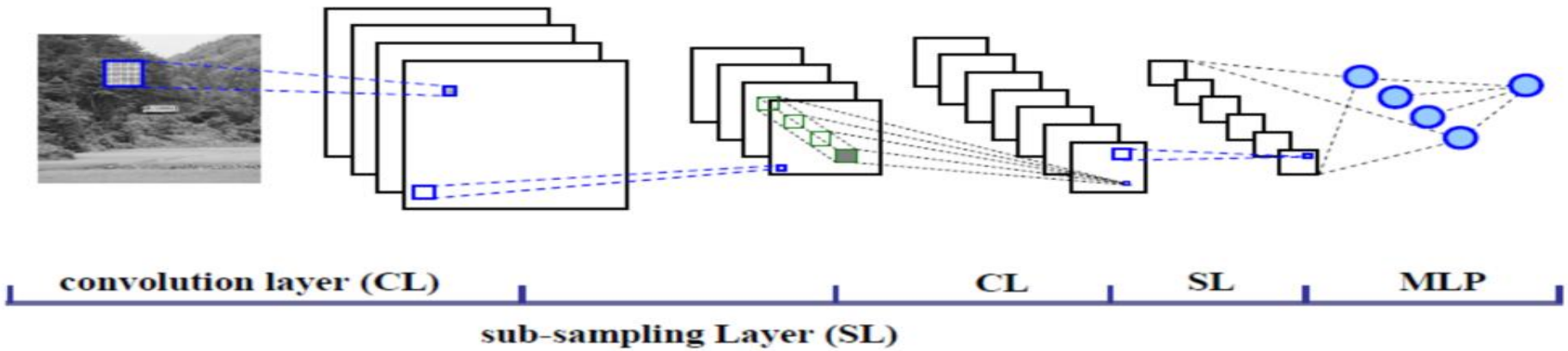
计算实际输出Op与相应的理想输出Yp的差

按极小化误差的方法反向传播调整权矩阵



CNN反向传播

- 代价函数
  - 最小化平方误差(MSE)，最小化相对熵(Relative Entropy)
- 反向传播主要考虑三个方面：
  - 输出层，代价函数的确定及求导
  - Pooling，数据的下采样及残差的上采样
  - 卷积层，数据的卷积运算及残差的反卷积运算



BP算法

- 误差反传

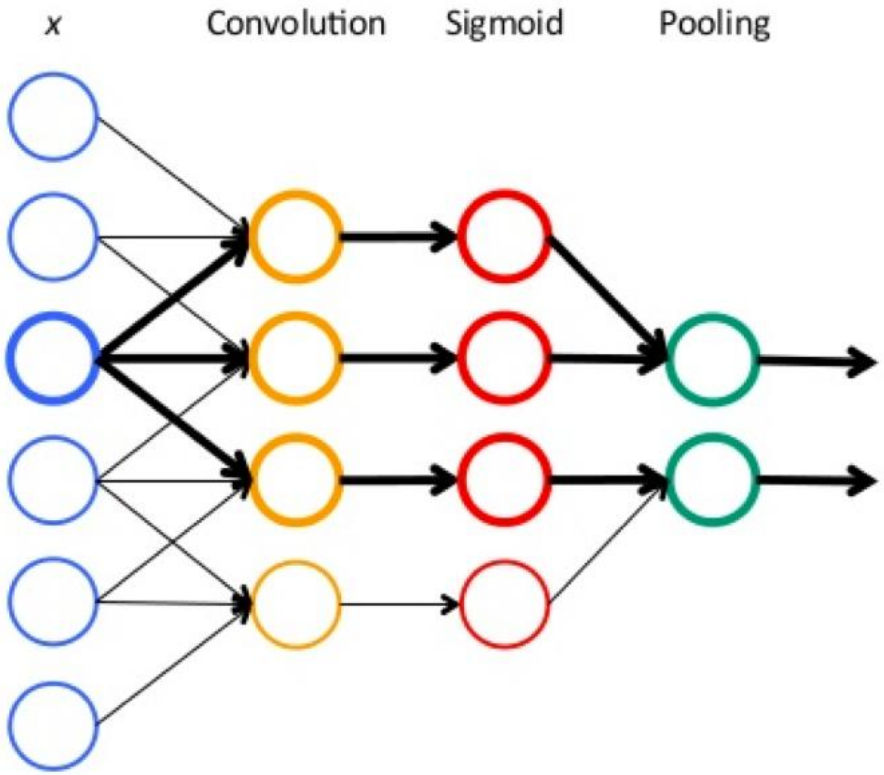
$$\Delta \omega_{ji} = -\eta x_j y_i (1 - y_i)(y_i - d_i)$$

$$\Delta \omega_{kj} = -\eta x_k y_j (1 - y_j) \sum_{i=1}^{n} \omega_{ji} y_i (1 - y_i)(y_i - d_i)$$

$$\delta = \frac{\partial L}{\partial z} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z}$$

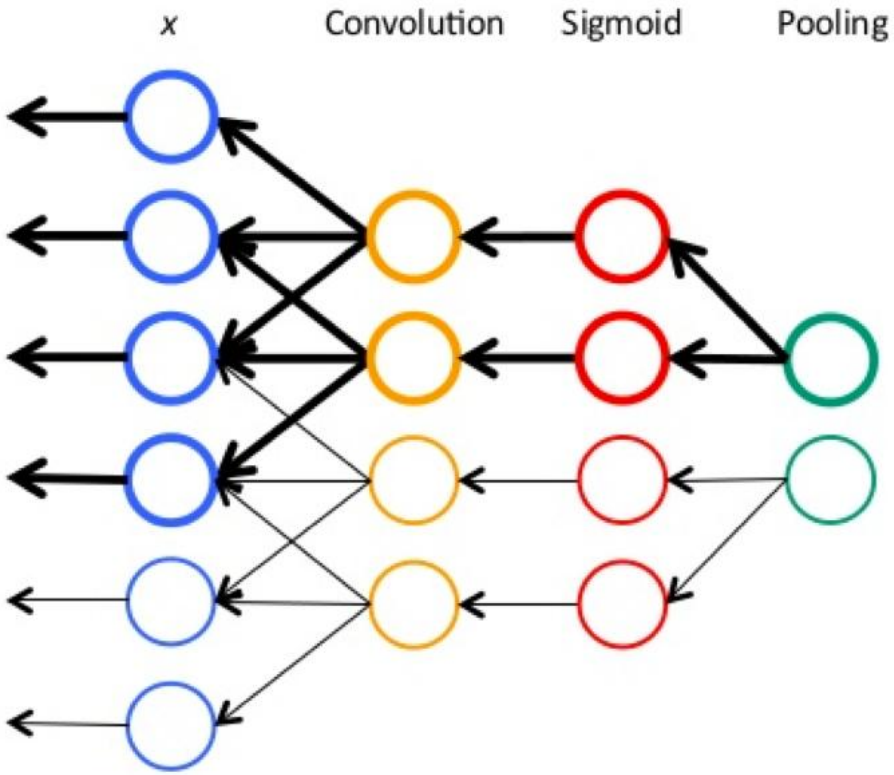
$$\Delta \omega_{ji} = -\eta x_j \delta_i, \quad \delta_i = g_i'(y_i - d_i)$$

$$\Delta \omega_{kj} = -\eta x_k \delta_j, \quad \delta_j = g_j' \sum_{i=1}^{n} \omega_{ji} \delta_i$$



$$\left(f^{(pool)} \circ f^{(sigm)} \circ f_w^{(conv)}\right)(x)$$

Forward propagation



Backward propagation

残差的反向传播

- Pooling层
  - 假设卷积层大小为4×4, pooling区域大小为2×2, pooling后得到的矩形大小也为2*2

1	3
2	4

0.25	0.25	0.75	0.75
0.25	0.25	0.75	0.75
0.5	0.5	1	1
0.5	0.5	1	1

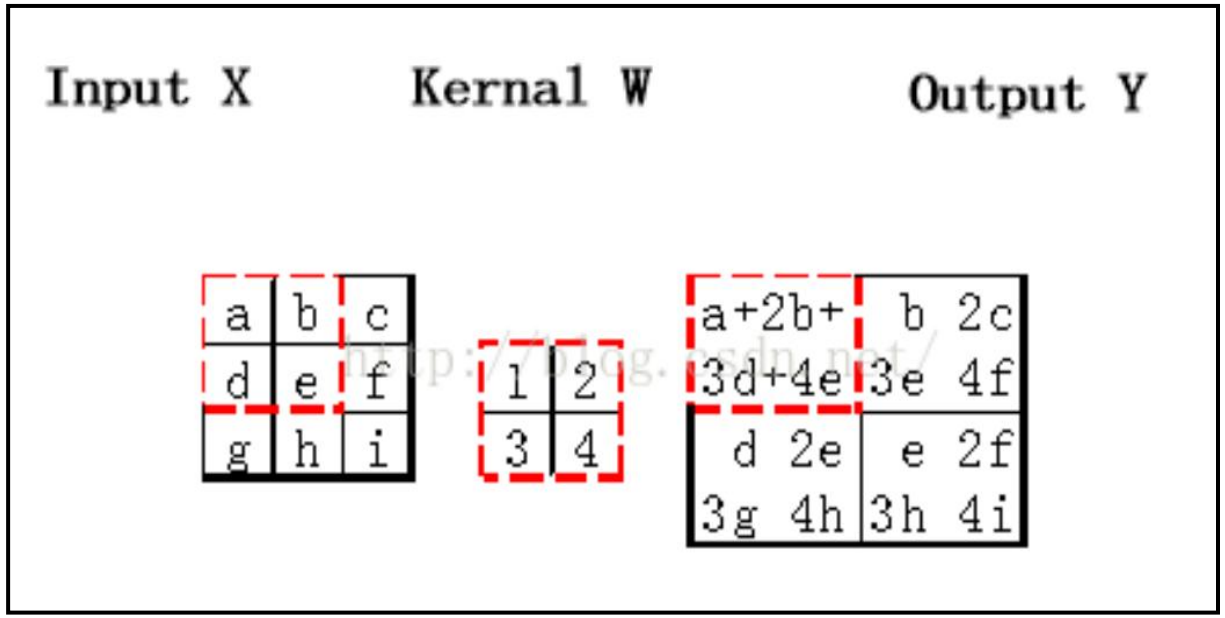
Average pool

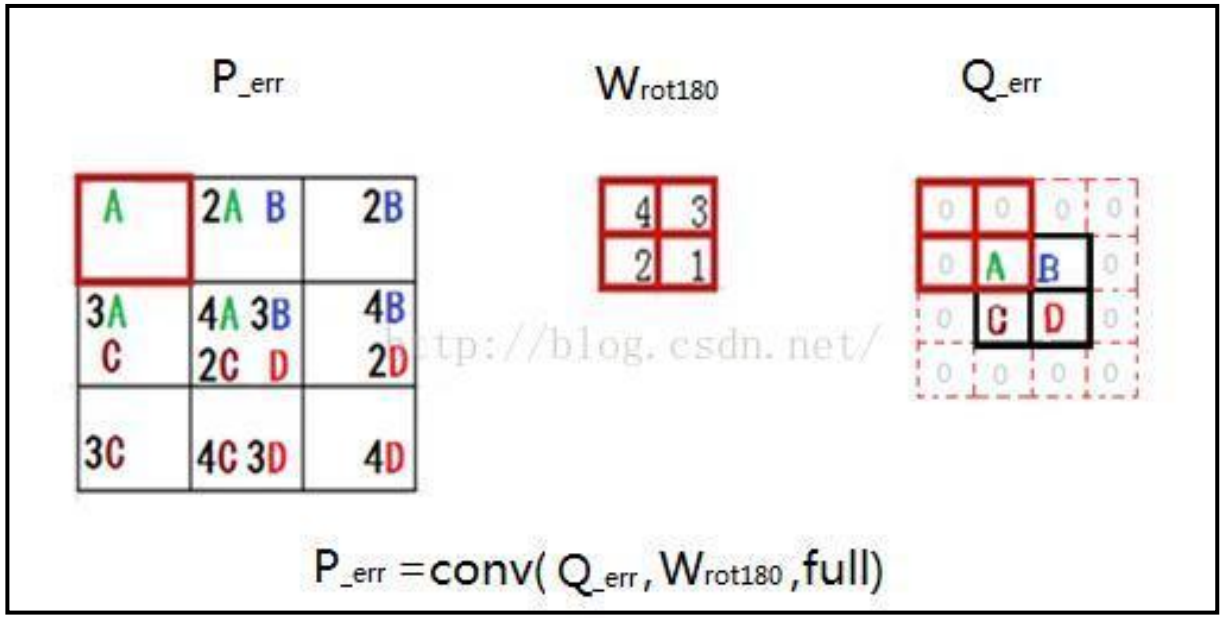
0	0	0	3
0	1	0	0
2	0	0	0
0	0	4	0

Max pool

残差的反向传播

- 卷积层
  - 卷积 vs. 反卷积





残差的反向传播

- 卷积层
  - 通道图大小为3×3，2个特征核，核大小为2×2

1	3
2	2

0	0	0	0
0	1	3	0
0	2	2	0
0	0	0	0

★

0.1	0.2
0.2	0.4

=

0.1	0.5	0.6
0.4	1.6	1.6
0.4	1.2	0.8

2	1
1	1

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

★

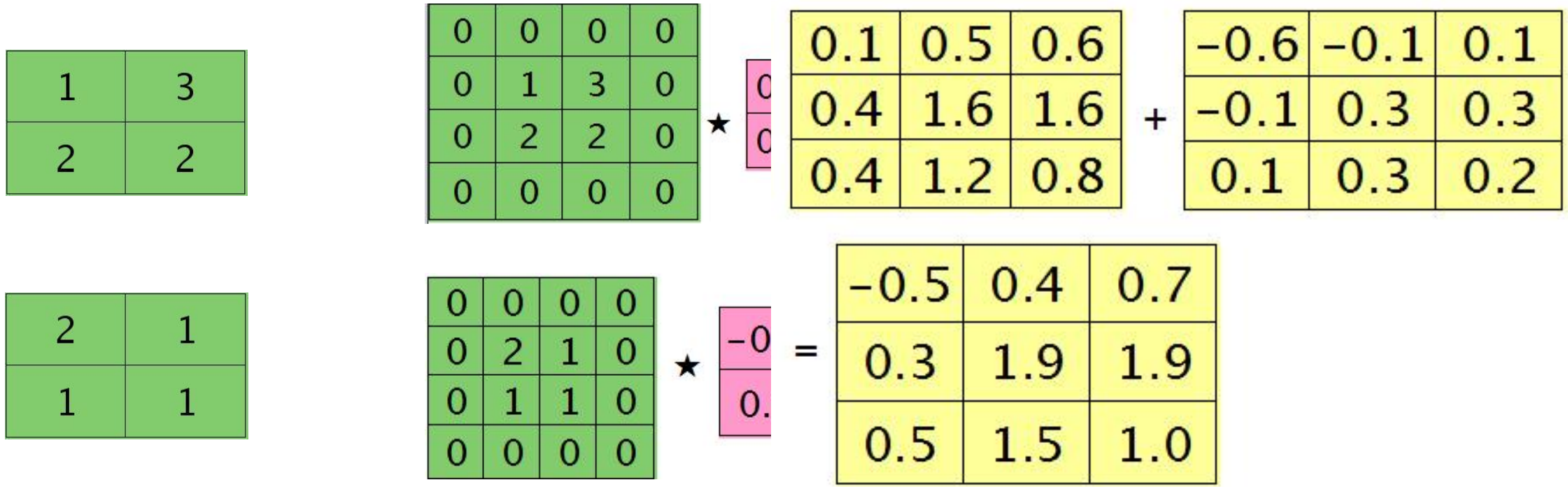
-0.3	0.1
0.1	0.2

=

-0.6	-0.1	0.1
-0.1	0.3	0.3
0.1	0.3	0.2

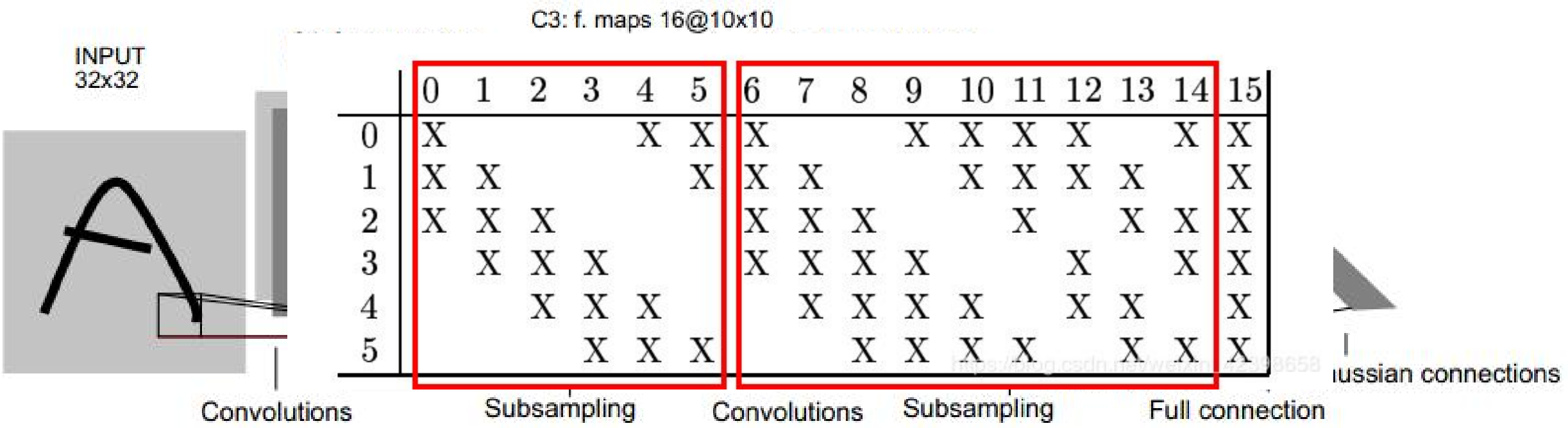
残差的反向传播

- 卷积层
  - 通道图大小为3×3，2个特征核，核大小为2×2

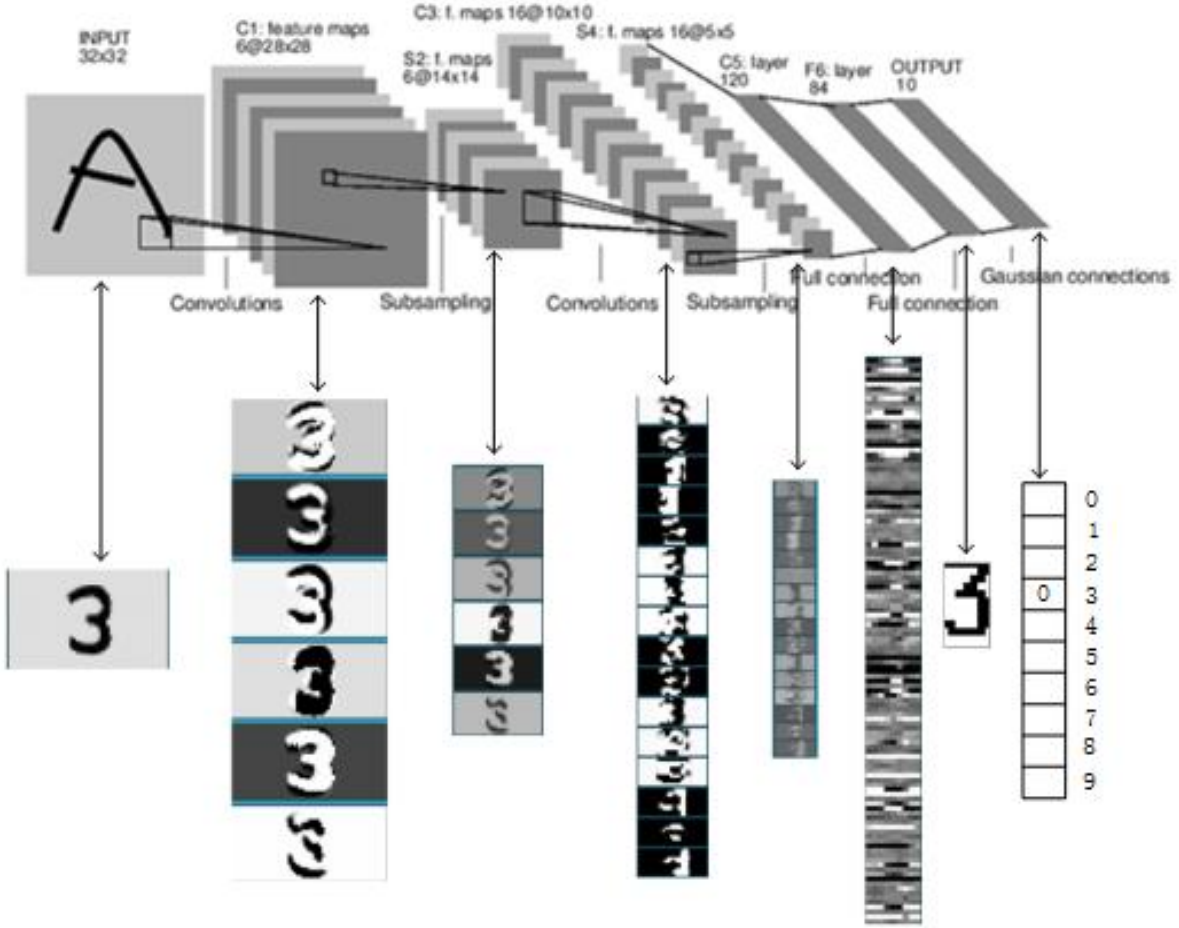


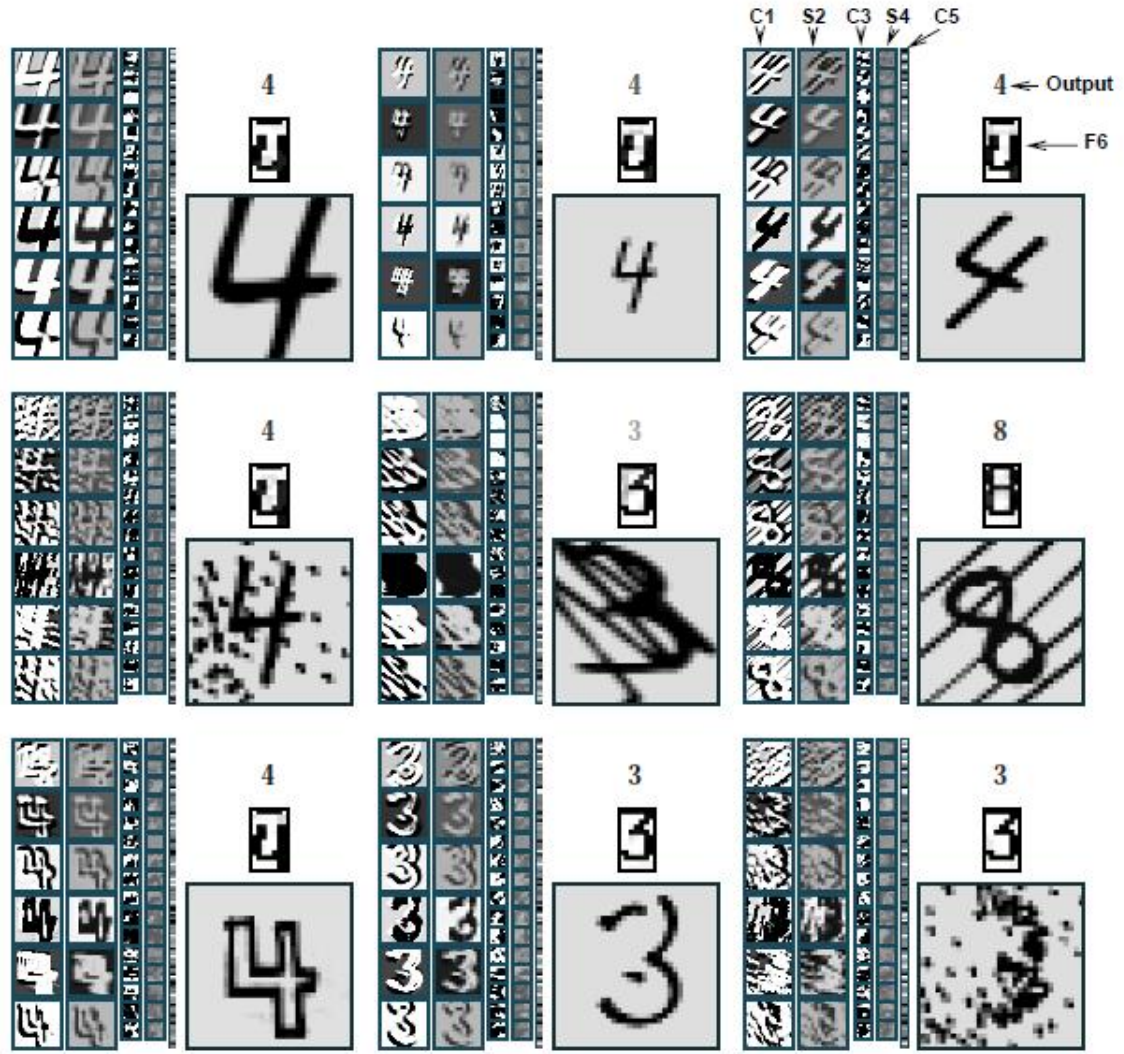
例：文字识别系统LeNet-5

- 当年美国大多数银行就是用它来识别支票上面的手写数字的。



文字识别系统LeNet-5



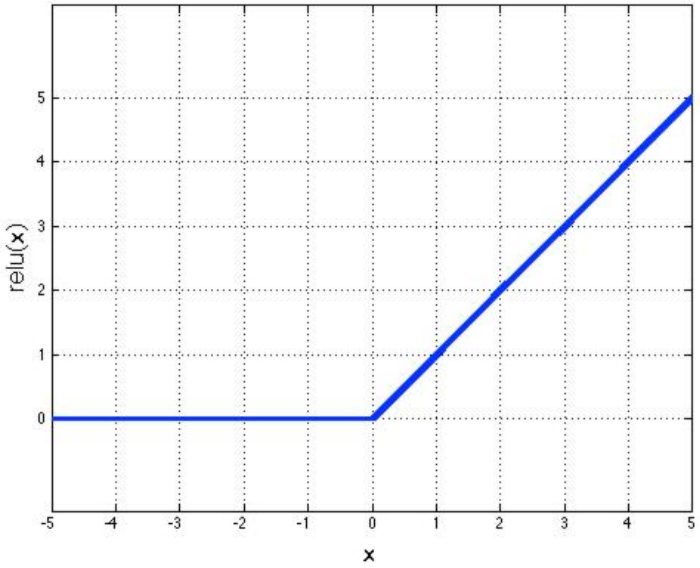


卷积神经网络

- 卷积网络的核心思想：
  - 将局部感受野、权值共享以及时间或空间亚采样这三种结构思想结合起来获得了某种程度的位移、尺度、形变不变性。
- 层间联系和空域信息的紧密关系，使其适于图像处理和理解。
  - 图像和网络的拓扑结构能很好的吻合
- 避免了显式的特征抽取，而隐式地从训练数据中进行学习
  - 特征提取和模式分类同时进行，并同时在训练中产生；
  - 权重共享可以减少网络的训练参数，使神经网络结构变得更简单，适应性更强。

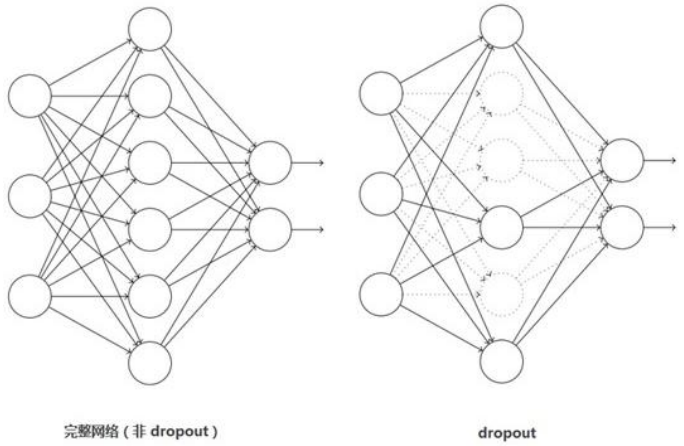
CNN的改进

- Rectified linear function
  - 加速收敛
  - 稀疏化



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

- dropout
  - 将隐层节点以一定概率清0
  - 可以将dropout看作是模型平均的一种



CNN的改进

- local Contrast Normalization
  - Subtracting a low-pass smoothed version of the layer
  - Just another convolution in fact (with fixed coefficients)
  - Lots of variants (per feature map, across feature maps, …)

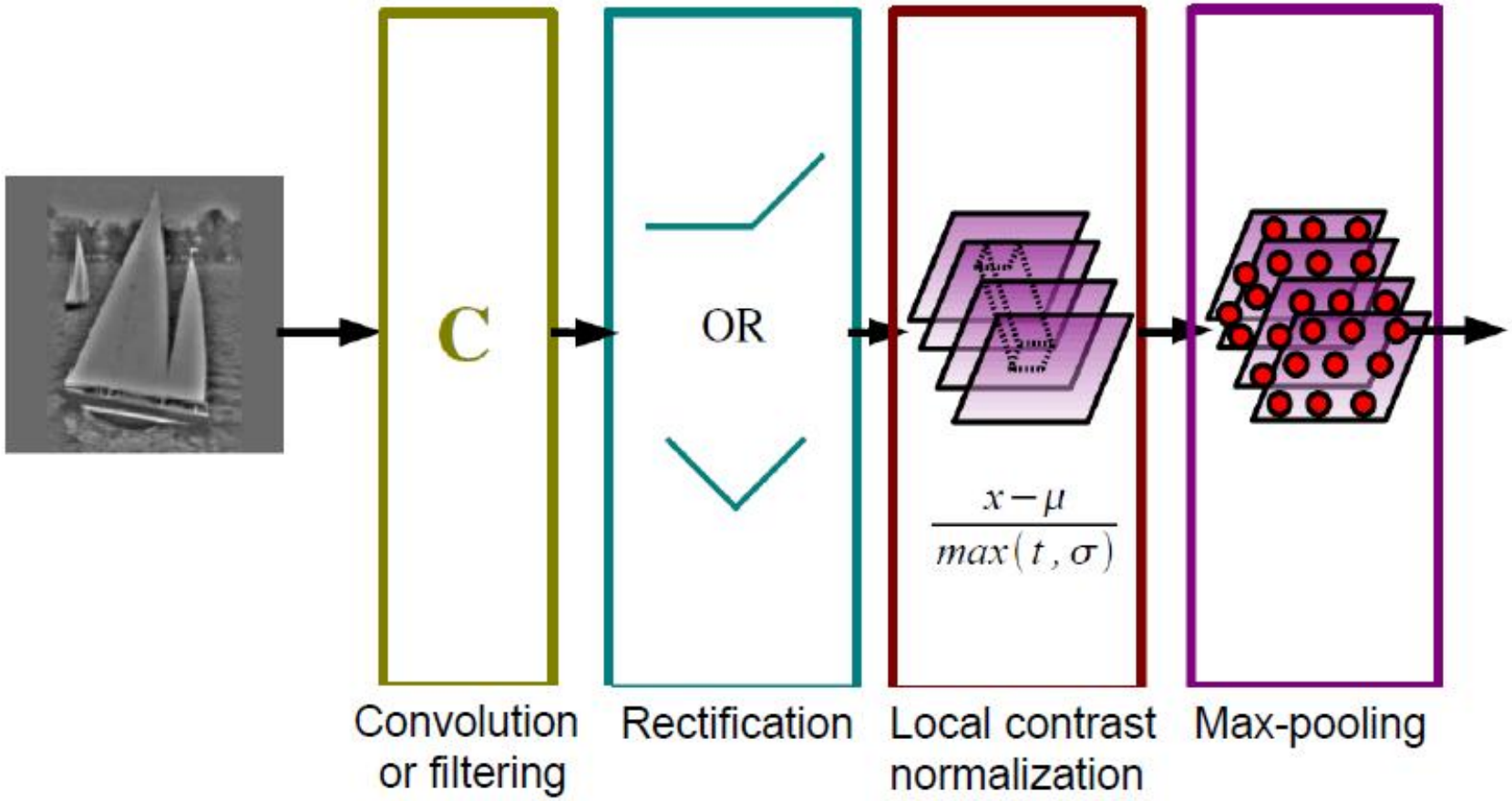
  - Empirically, seems to help a bit (1-2%) on ImageNet

$$v_{ijk} = x_{ijk} - \sum_{ipq} w_{pq} . x_{i,j+p,k+q} \qquad \sum_{ipq} w_{pq} = 1$$

$$y_{ijk} = v_{ijk} / \max(c, \sigma_{jk}) \qquad \sigma_{jk} = (\sum_{ipq} w_{pq} . v^2_{i,j+p,k+q})^{1/2}$$

CNN改进

- 非线性变换、池化

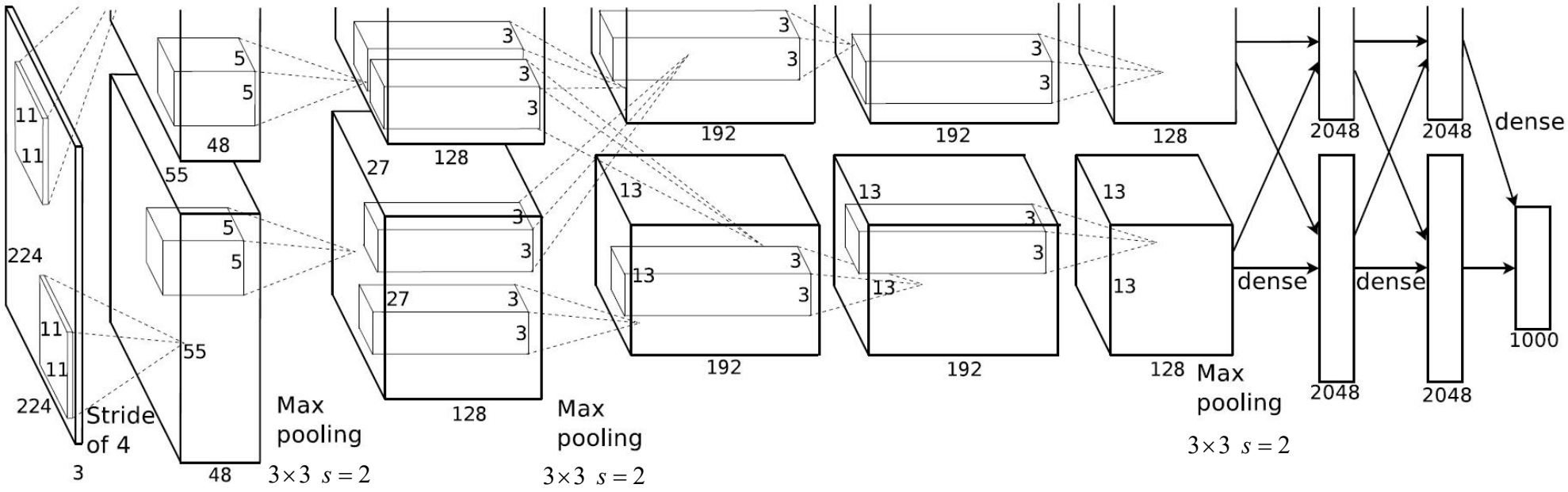


(Jarret et al., 2009)

卷积神经网络实例

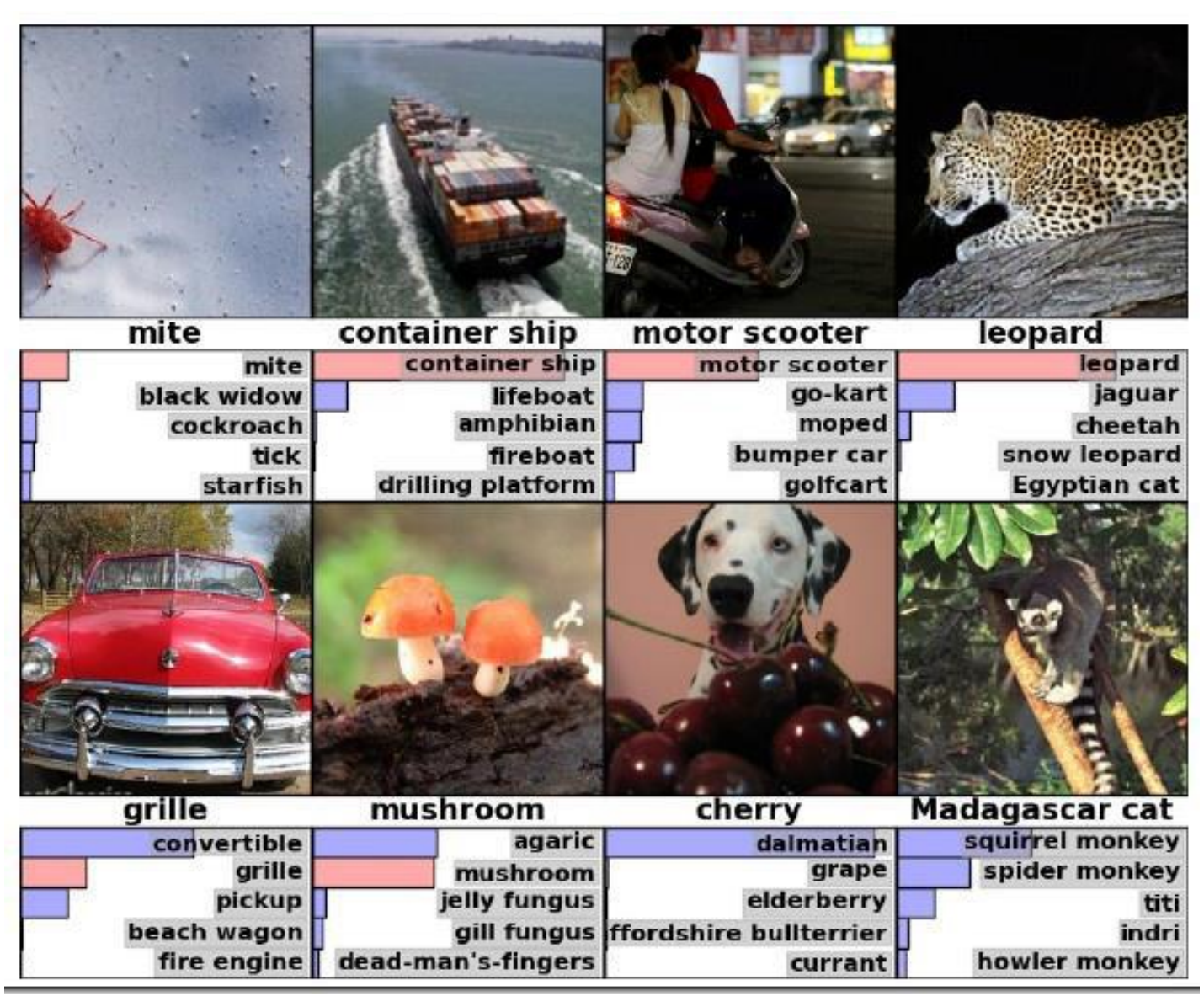
ImageNet CNN (AlexNet)

- Structure (conv-relu-maxpool-norm)
- Very good implementation, running on two GPUs
- ReLU transfer function. Dropout trick.
- Also trains on full ImageNet (15M images, 15000 classes)



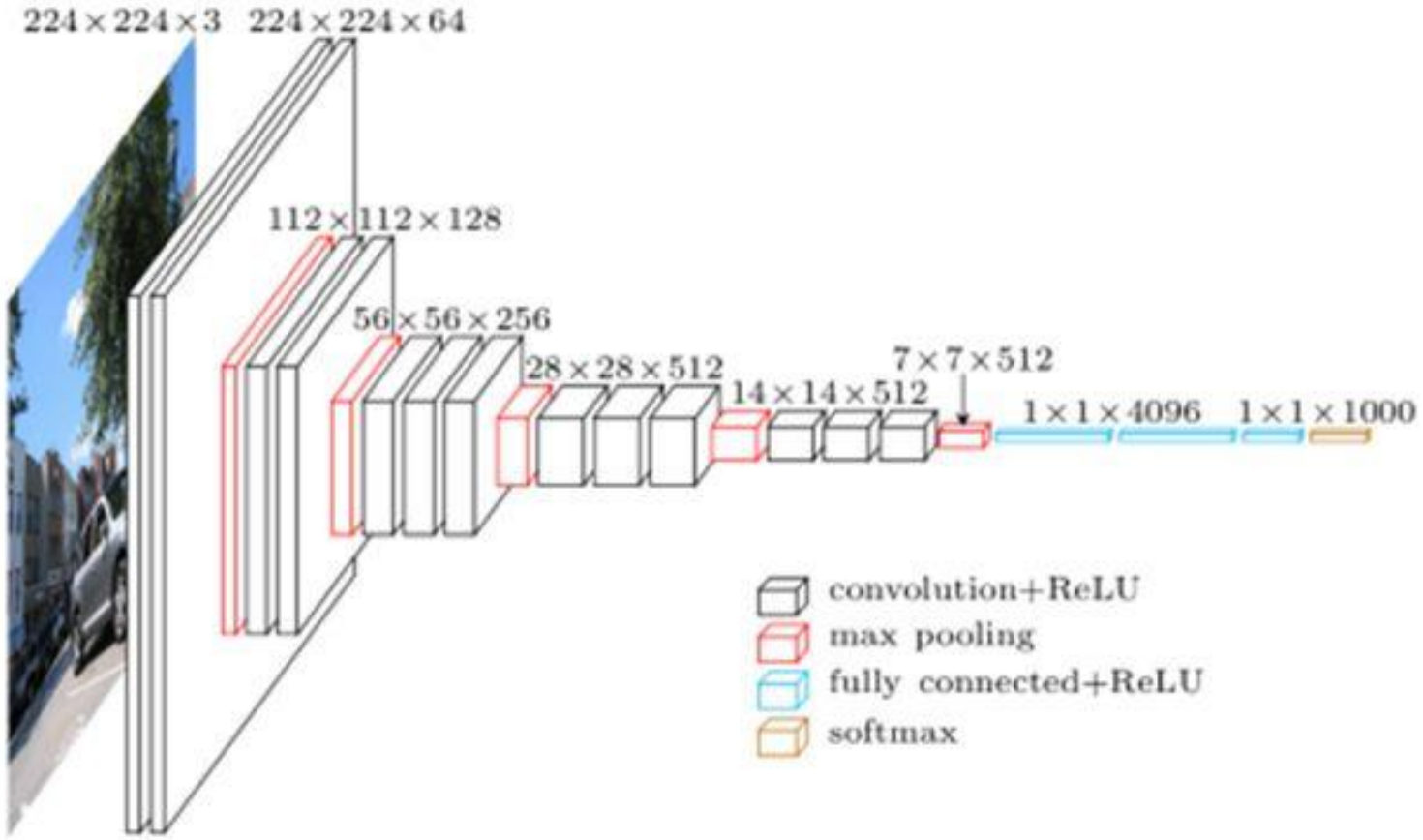
(Kirzhevsky, Sutskever, Hinton, 2012)

ImageNet CNN



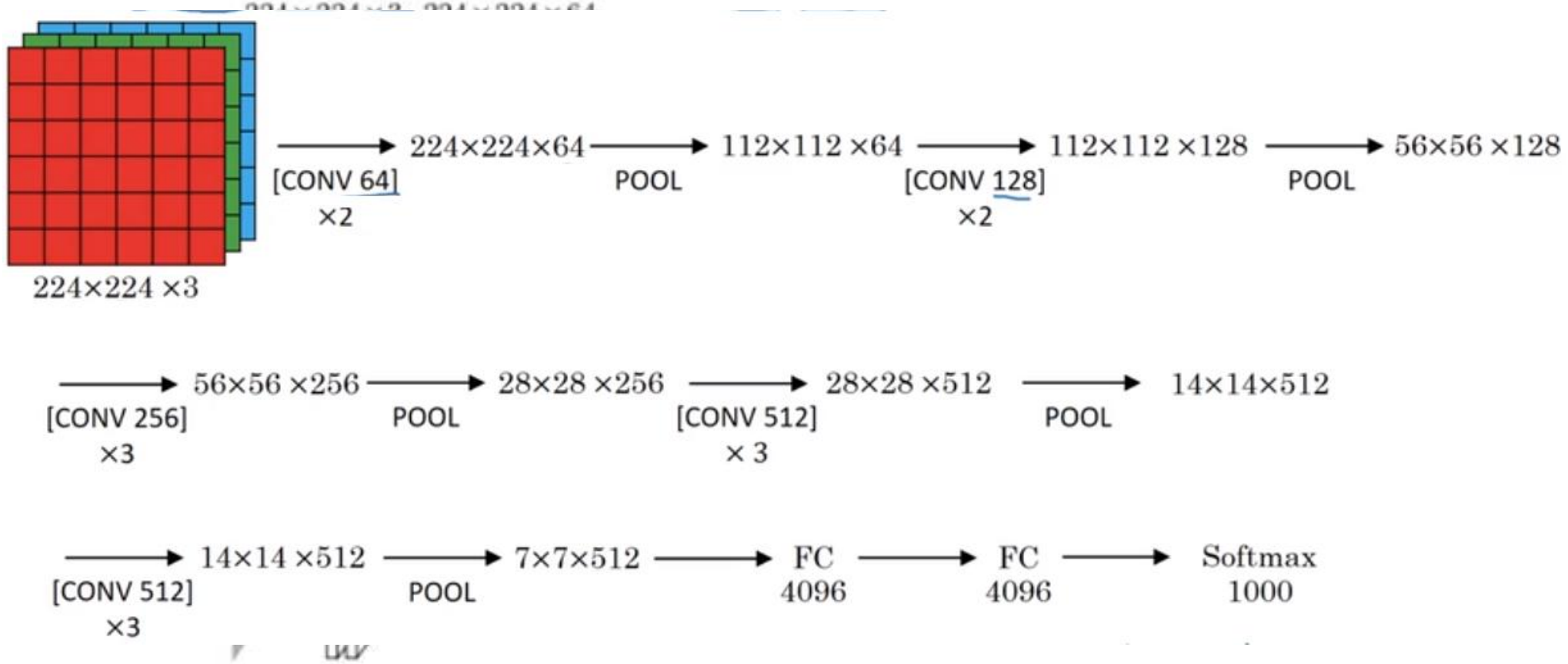
VGG Net

- VGG Net的结构



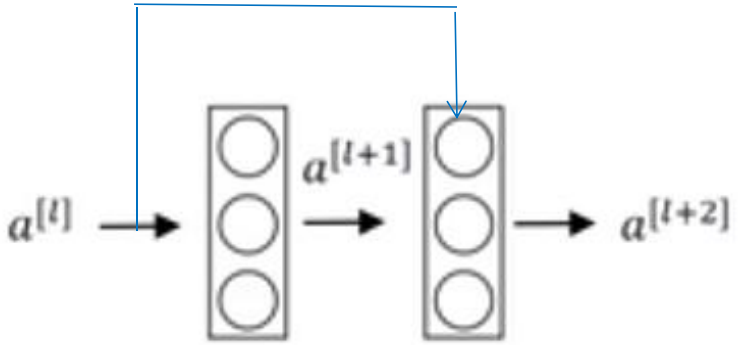
VGG Net

- VGG Net的结构



残差网络（Residual Networks（ResNets））

- Residual Blook

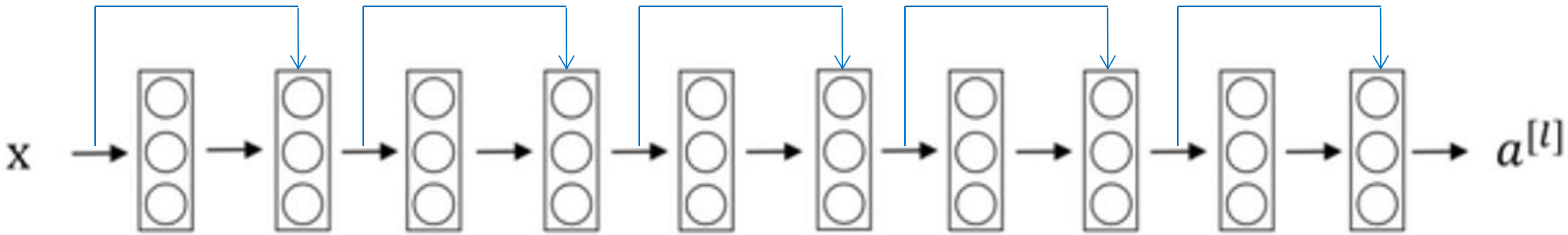


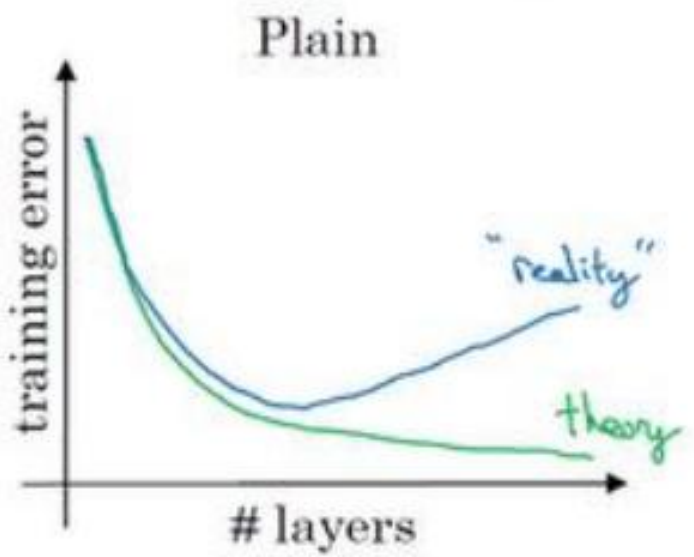
$$a^{[l]} \rightarrow \text{Linear} \rightarrow \text{ReLU} \rightarrow a^{[l+1]} \rightarrow \text{Linear} \rightarrow \text{ReLU} \rightarrow a^{[l+2]}$$

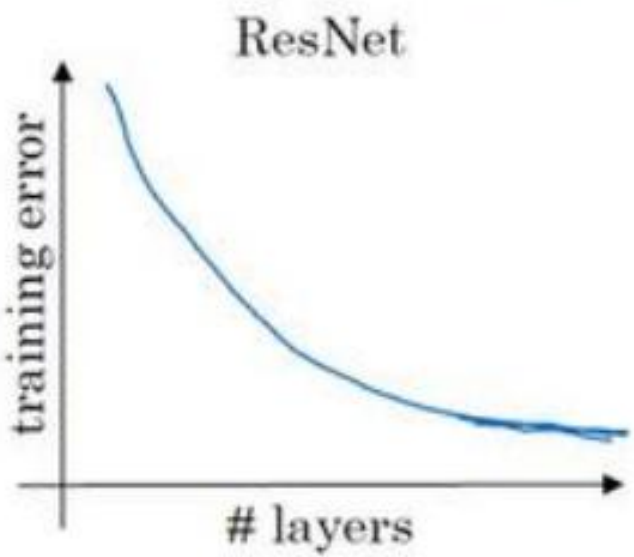
$$z^{[l+1]} = W^{[l+1]}\, a^{[l]} + b^{[l+1]} \quad a^{[l+1]} = g(z^{[l+1]}) \quad z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]} \quad \cancel{a^{[l+2]} = g(z^{[l+2]})}$$

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

残差网络

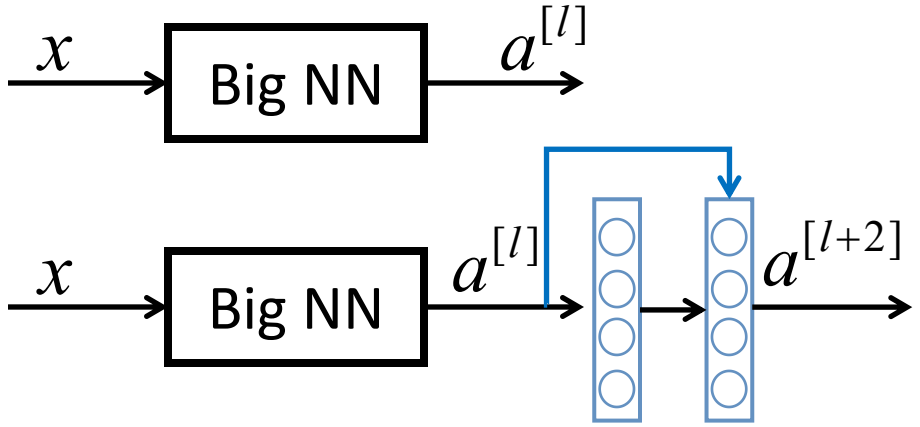

x
$a^{[l]}$


Plain
training error
"reality"
theory
layers


ResNet
training error
layers

残差网络为什么有用？

- 因为残差网络很容易学习恒等式函数，所以随着网络加深，至少不会让网络变差。



ReLU $a \geq 0$

$$a^{[l+2]} = g(z^{[l+2]} + W_s a^{[l]})$$
$$= g(W^{[l+2]} a^{[l+1]} + b^{[l+2]} + a^{[l]})$$

$$W^{[l+2]} = 0 \quad b^{[l+2]} = 0$$
$$W^{[l+2]} a^{[l+1]} + b^{[l+2]} = 0$$

$$\mathrm{a}^{[l+2]} = g(\mathrm{a}^{[l]}) = \mathrm{a}^{[l]}$$

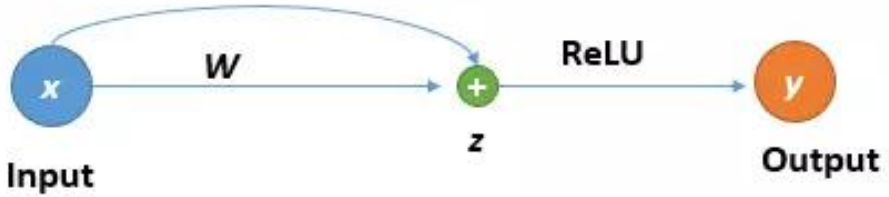
残差网络为什么有用？

- 学习结果对**网络权重的波动变化**更敏感

Plain network:

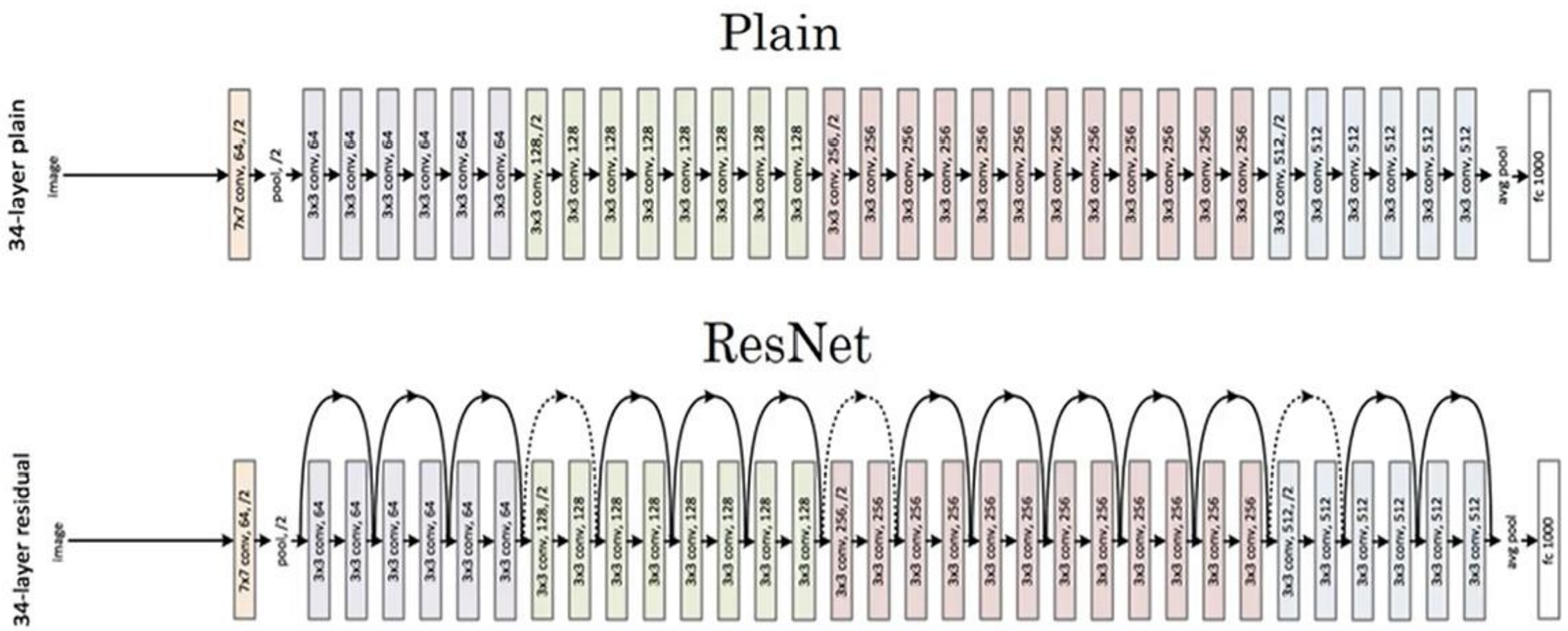
x = 1
w1 = 1.1, y1 = 1.1
w2 = 1.2, y2 = 1.2
Δw = 0.1, Δy = 0.1, Δ = 0.1/1.1 = **9** %

Res network:



x = 1
w1 = 0.1, z1 = 0.1 , y1 = 1.1
w2 = 0.2, z2 = 0.2, y2 = 1.2
Δw = 0.1, Δz = 0.1, Δ = 0.1/0.1 = **100** %

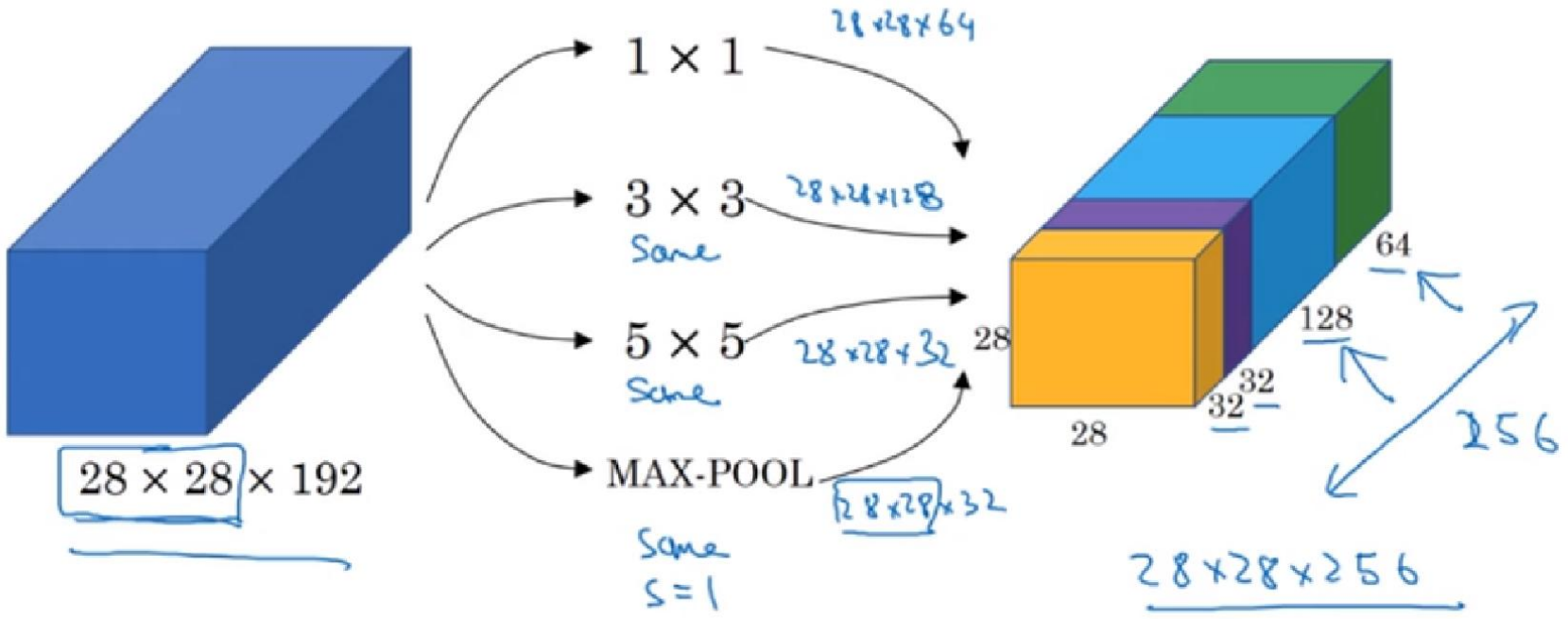
残差网络ResNet



[He et al. Deep residual learning for image recognition]

Inception 网络

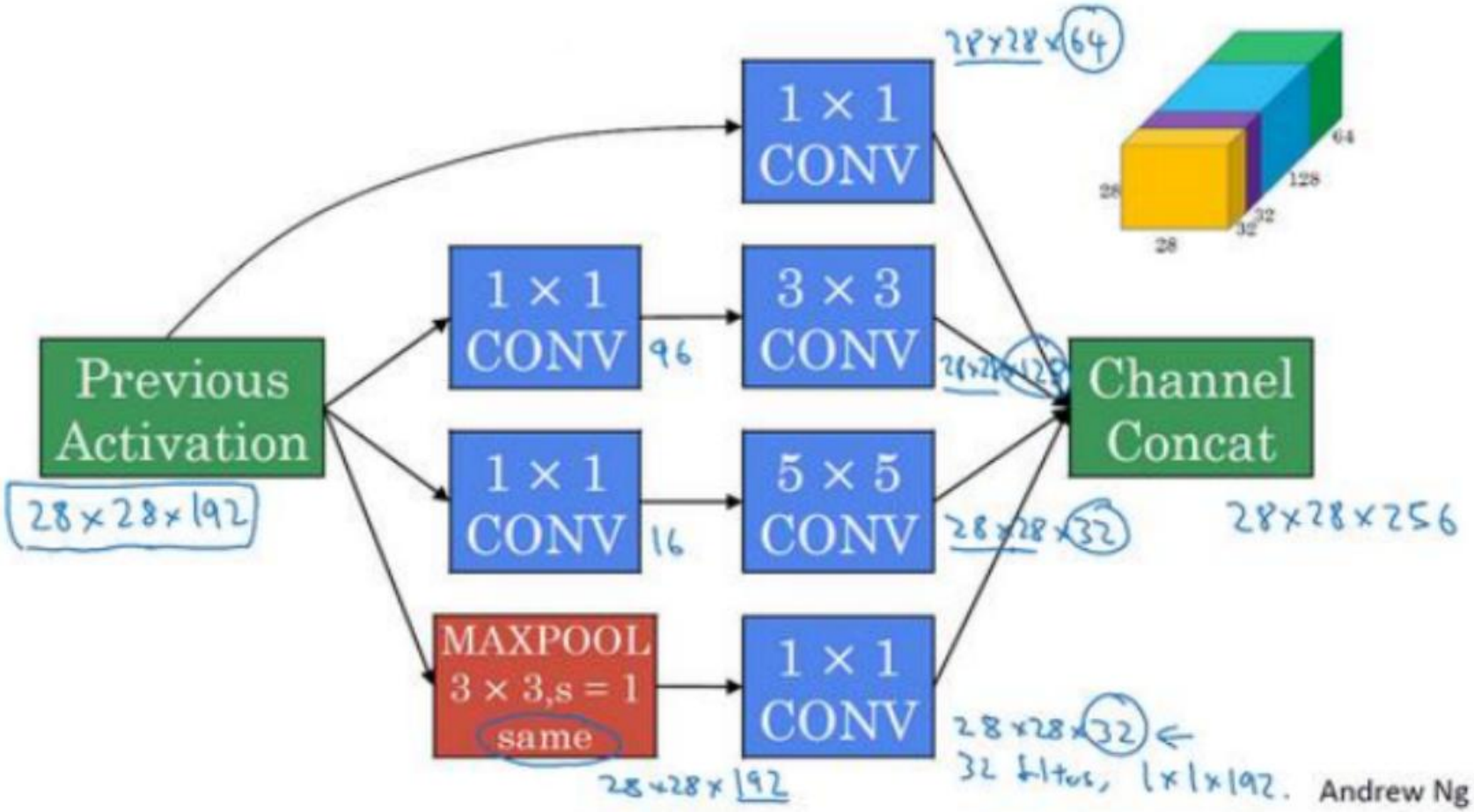
- Inception模块



[Szegedy et al. 2014. Going deeper with convolutions]

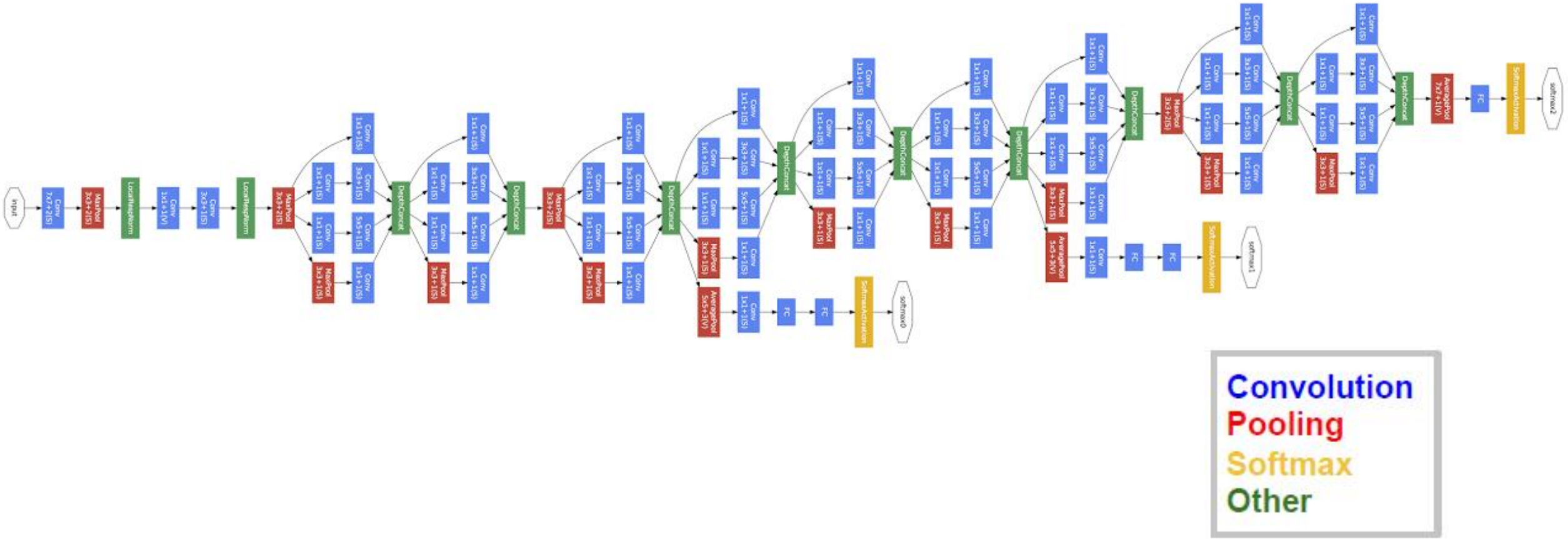
GoogleLeNet---Very Deep

- Inception Module



GoogleLeNet---Very Deep

- Inception网络



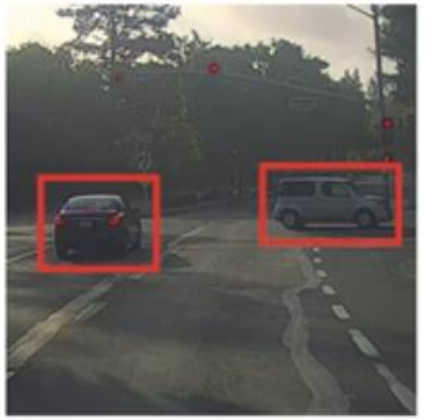
图像数据应用

目标定位

Image classification

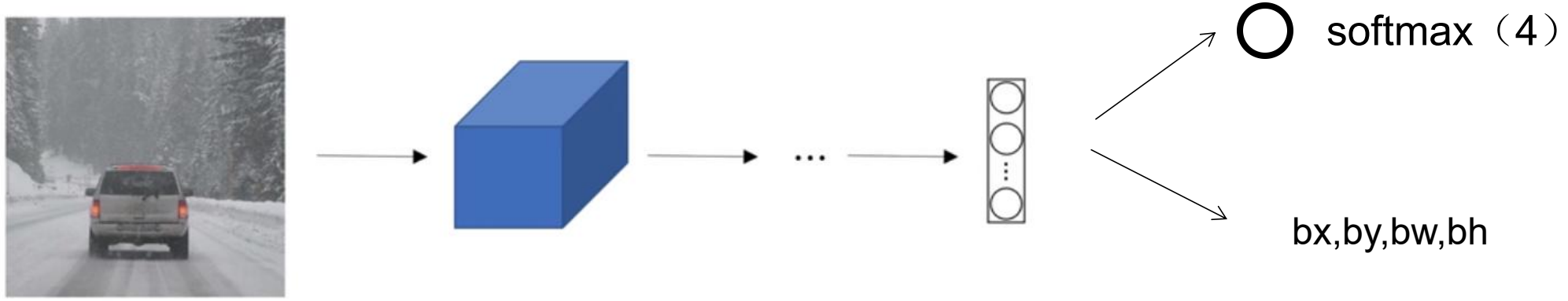
Classification with localization

Detection



目标定位

- Classification with localization



1 - pedestrian
2 - car
3 - motorcycle
4 - background

目标定位

Defining the target label y

1 - pedestrian
2 - car
3 - motorcycle
4 - background

Need to output b_x, b_y, b_h, b_w, class label (1-4)

$x =$

$y =$ $\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ is there an object?

$$\mathcal{L}(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \dots + (\hat{y}_8 - y_8)^2 & \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_1 = 0 \end{cases}$$

(x, y) $\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$ 1

$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$ p_c ← "don't care"

2

特征点检测

b_x, b_y, b_h, b_w

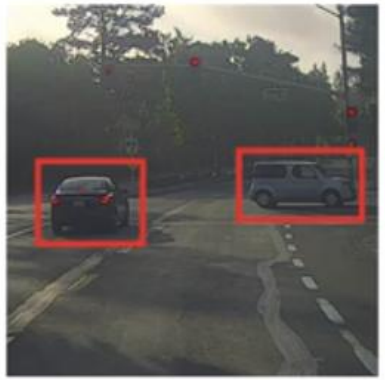
l_{1x}, l_{1y},
l_{2x}, l_{2y},
l_{3x}, l_{3y},
l_{4x}, l_{4y}
⋮
l_{64}, l_{64y}

} x, y

l_{1x}, l_{1y},
⋮
l_{31x} l_{21y}

目标检测

- Car detection example



Training set:

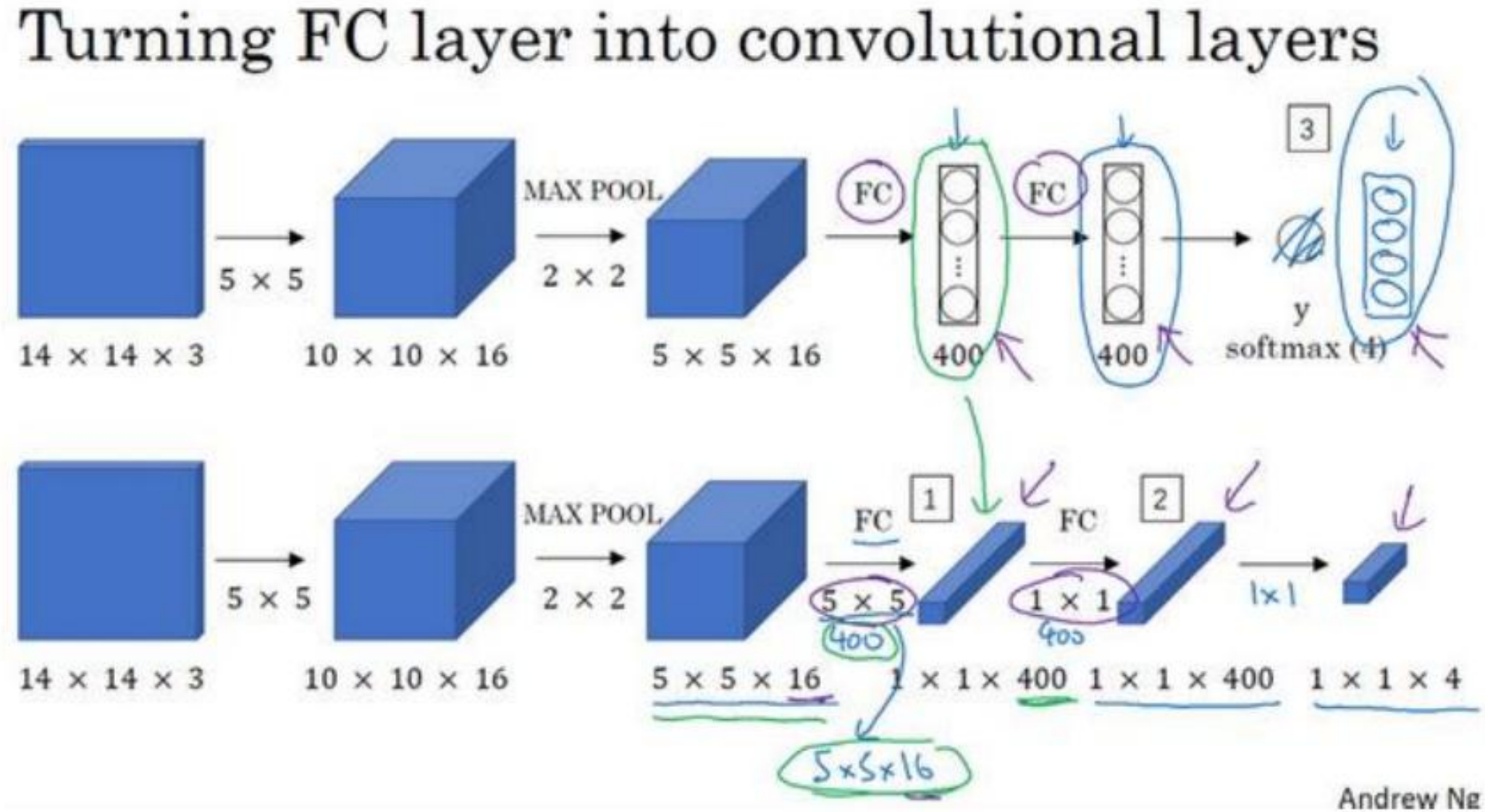
x	y
	1
	1
	1
	0
	0

→ ConvNet → y

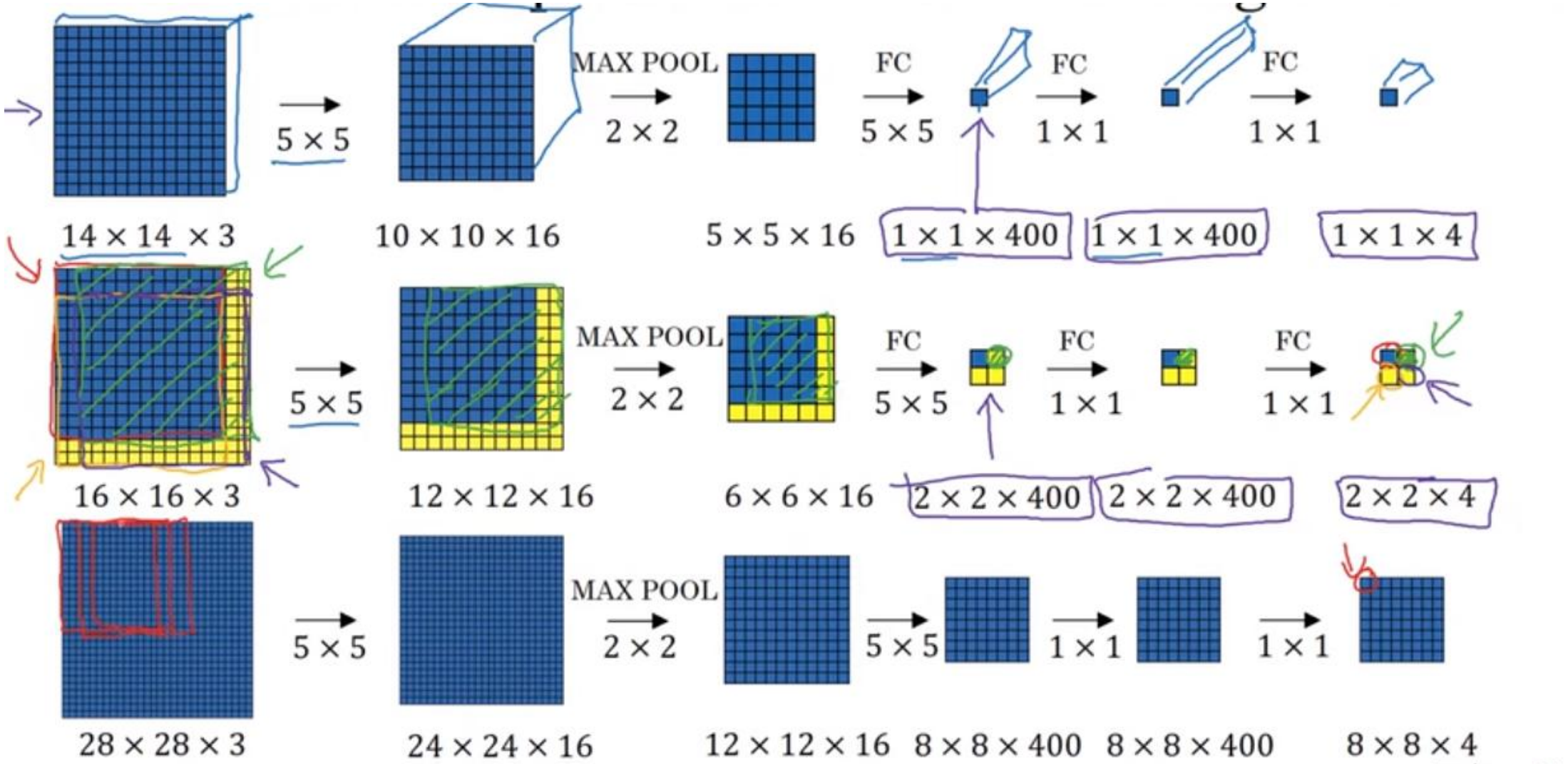
目标检测

- Sliding windows detection

卷积的滑动窗口实现


Turning FC layer into convolutional layers
5 × 5
MAX POOL
2 × 2
FC
FC
3
y
softmax (4)
14 × 14 × 3
10 × 10 × 16
5 × 5 × 16
400
400
5 × 5
MAX POOL
2 × 2
FC
1
FC
2
1 × 1
1×1
5 × 5 × 16
1 × 1 × 400
1 × 1 × 400
1 × 1 × 4
5×5×16
Andrew Ng

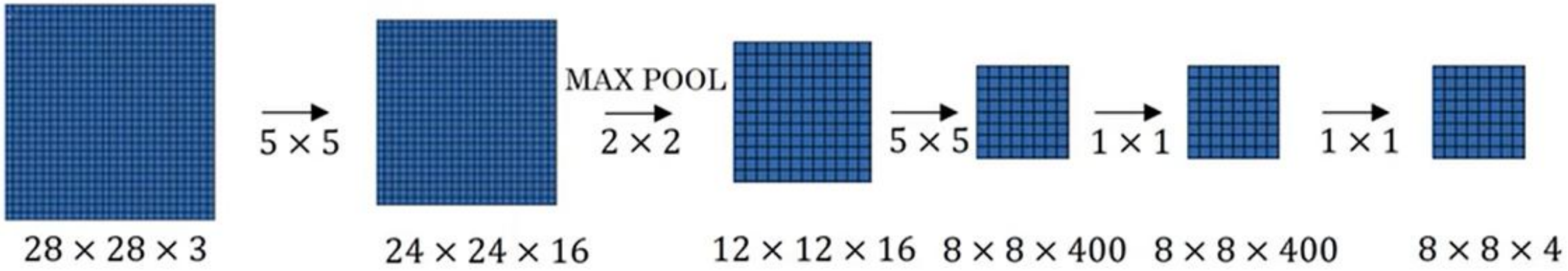
卷积的滑动窗口实现

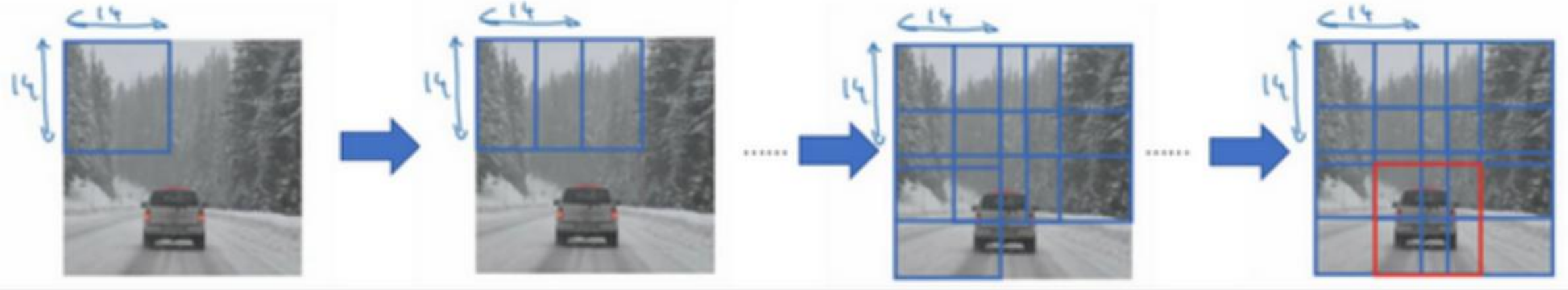


[Sermanet et al., 2014, OverFeat: Integrated recognition, localization and detection using convolutional networks]

Andrew Ng

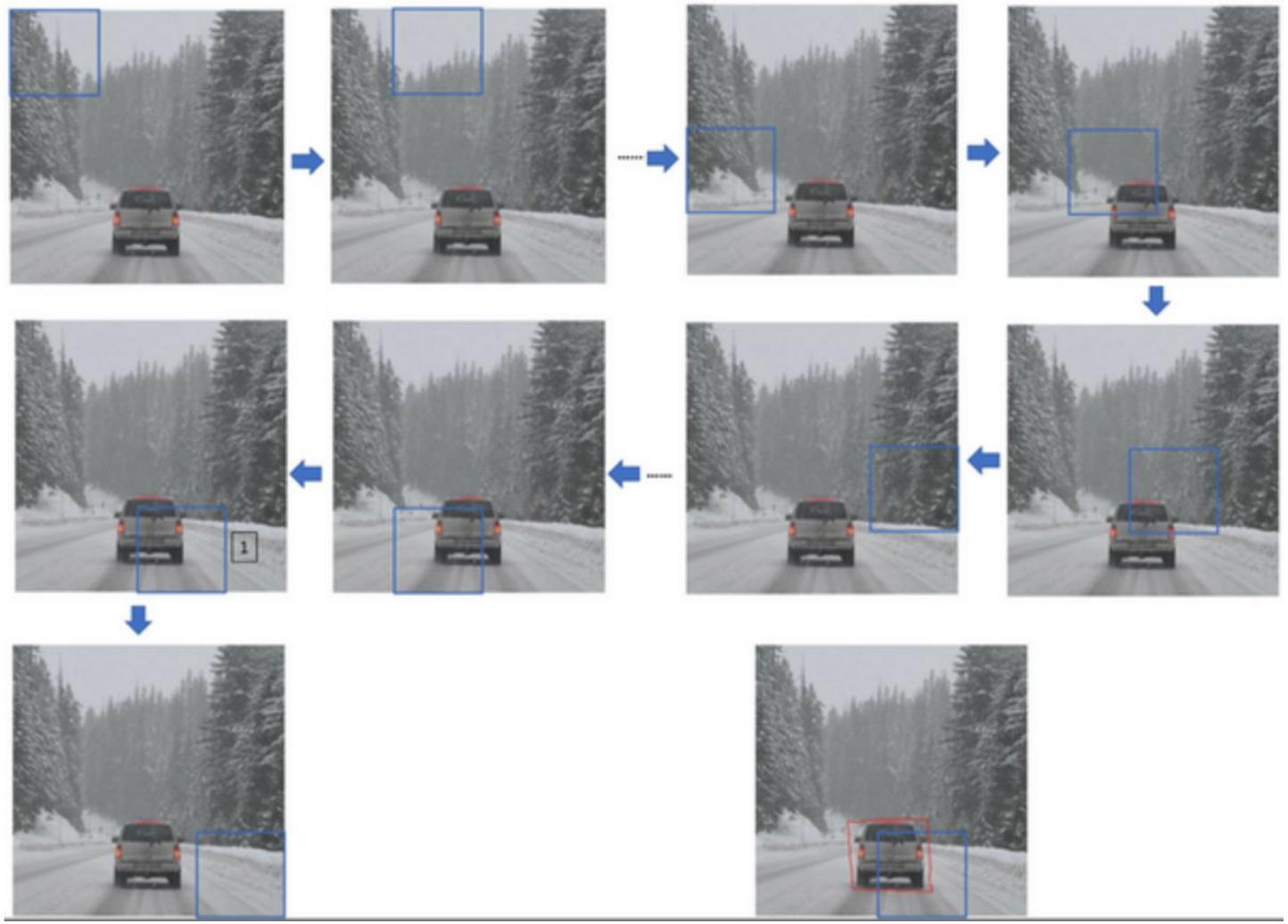
卷积的滑动窗口实现



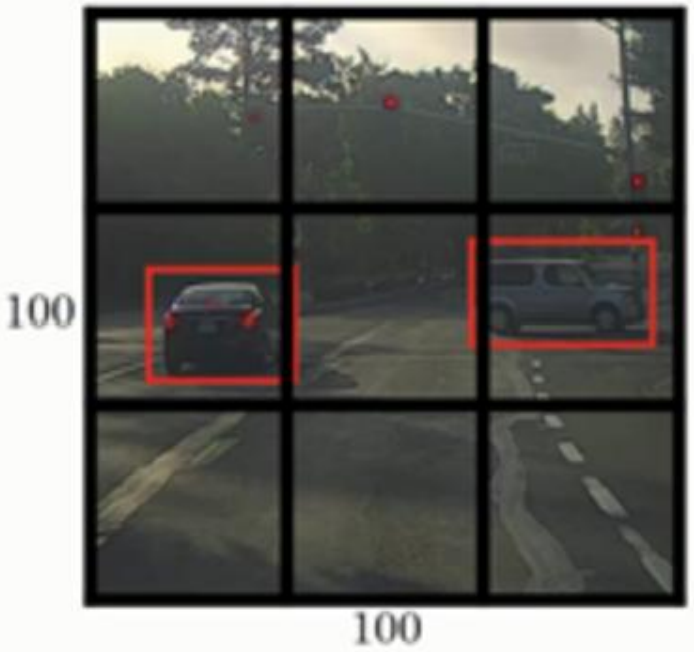


Bounding Box预测

- 滑动窗口法



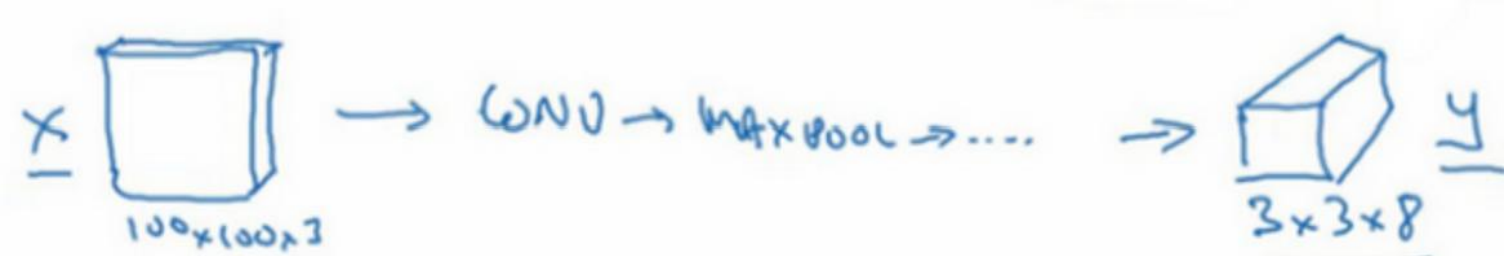
YOLO算法（you only look once）



Labels for training

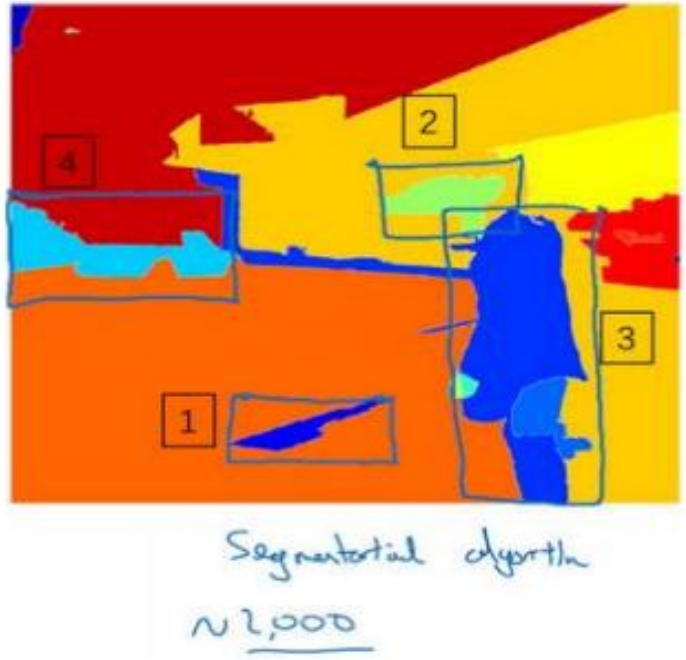
For each grid cell:

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$



候选区域（Region proposals）

- R-CNN：
  - 首先使用传统图像分割的方法，对图像进行分割，然后提取出2000个可能有对象的候选框，这样相对于最开始提到的滑动框的方式减少了非常多的框，然后送入网络



Propose regions. Classify proposed rgions one at a time.
Output label+bounding box.

候选区域

- Fast RCNN：
  - 即先将整个图像进行卷积，然后再将候选框映射到卷积之后的输出做分类，其实就是上面提到的卷积整幅图减少重复计算。

- Faster RCNN:
  - 由于使用传统的图像分割选出候选框需要非常多的时间，所以使用卷积神经网络来选出候选框。

人脸识别

- 人脸验证（face verification）：
  - 1对1，输入一个照片，名字或者ID，然后判断这个人是否是本人。
- 人脸识别（face recognition）：
  - 1对多，判断这个人是否是系统中的某一个人。

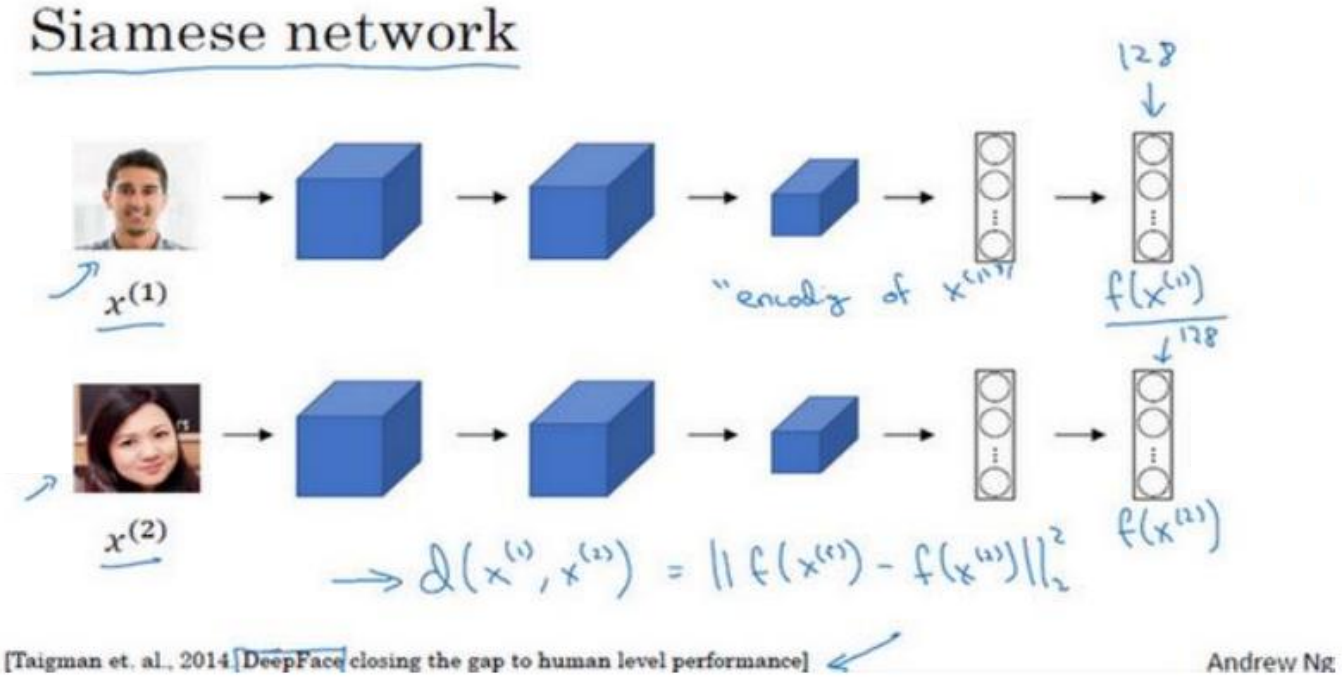
One-shot学习

- 例
  - 一个公司的员工，一般每个人只给一张工作照（如4个人），这时网络输出五个单元，分别代表他们以及都不是。
  - 这样设计会有两个明显的问题：第一是每个人只给了一张照片，训练样本太少；第二是如果这时候又来了一个员工，那么网络的输出得重新调整，显然不合理。

- 实际使用的方法
  - 是训练一个差异度函数（degree of difference between images）的网络，给网络输入两张照片，如果是同一个人则输出很小的值（差异很小），如果不是同一个人则输出很大的值，这时只要将来的人与系统中的照片逐一比较即可。同时如果来了新员工，也只需要将他的照片放入到系统中就可以了。

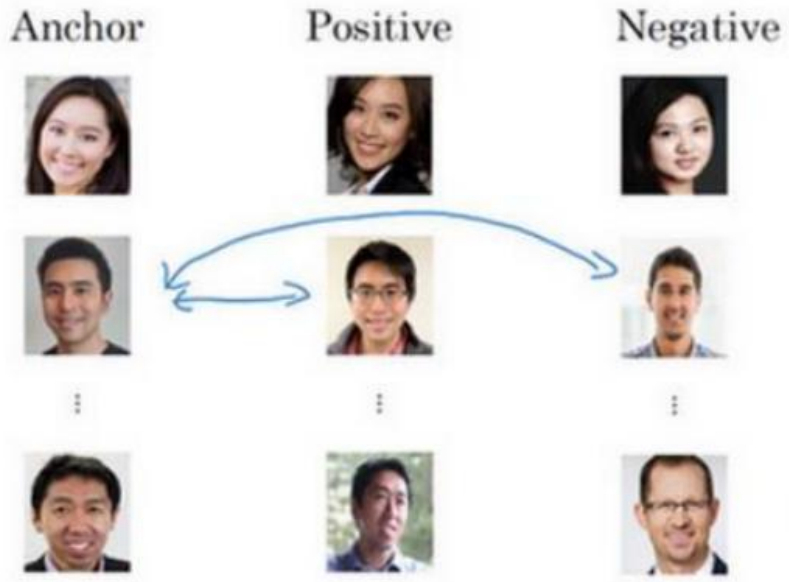
Siamese网络

- Siamese的输出不经过softmax激活函数做分类，直接就是输出一个向量，相当于把每个人映射成一个向量，然后判断两张图片是否是同一个人时只需要，分别输入这同一个网络，然后求输出向量差的范数



Triplet损失

- 首先训练集中必须有相同人的照片，三张照片为一组



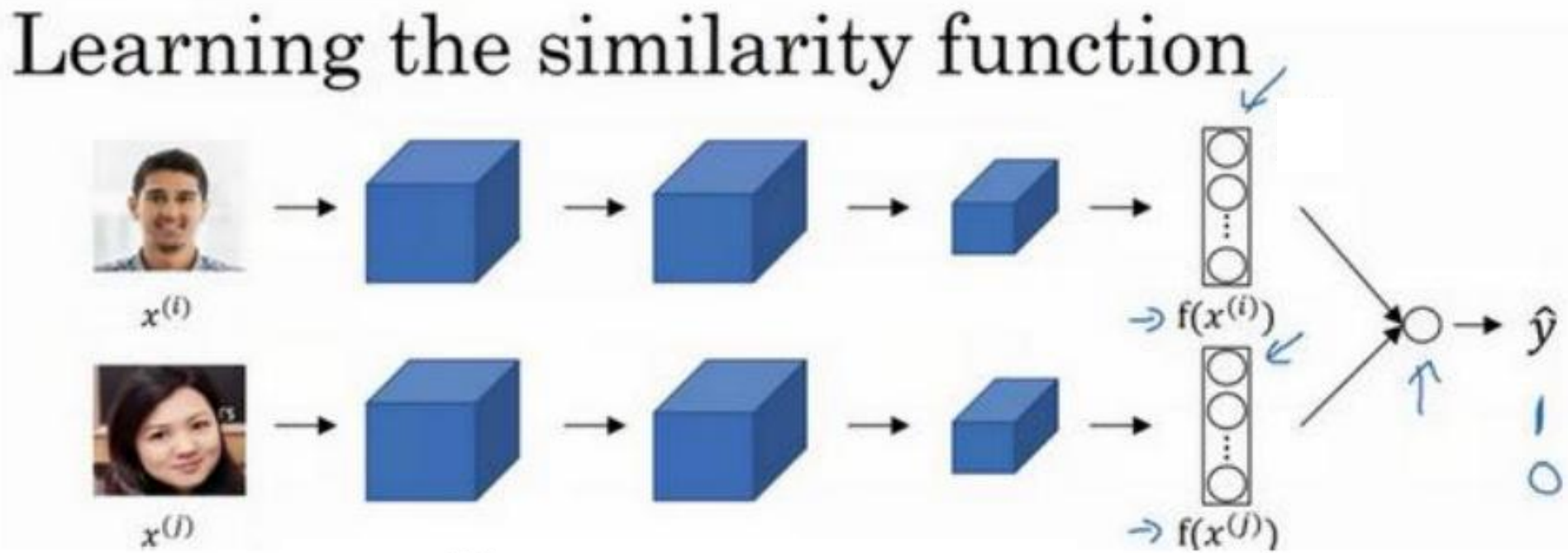
- 损失函数
  - 将两张相同的图片的距离比两张不同图片的距离的差值（注意此处差值已经是一个负值了）小于某个阈值时，其误差视为0

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + a \leq 0,$$

$$L(A, P, N) = max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + a, 0)$$

面部验证与二分类

- 可以用二分类的思想来训练Siamese网络，同样是用同一个网络，输出两张照片的向量，这时候在后面将向量对应元素做差的绝对值作为一个特征，再创建一层逻辑回归即可，输出0、1



$$\hat{y} = \sigma(\sum_{k=1}^{128} w_i |f(x^{(i)})_k - f(x^{(j)})_k| + b)$$

小结

- 卷积神经网络（Convolutional Neural Network, CNN）
- CNN实例
  - 经典网络、常用网络
- 图像数据应用
  - 目标定位、特征点检测、目标检测、人脸识别

Thanks !