卷积神经网络

主要内容

- 卷积网络的引出
- 几种经典的卷积网络
  - LeCun的卷积网络（1989）
  - Krizhevsky等AlexNet（2012）
  - 何凯明等ResNet（2016）

卷积神经网络

- 语音和图像都是矩阵描述的数据
  - 比较规整：line或者grid
  - 这类数据（信号）在信号处理领域非常常见
  - 卷积、相关这样计算与概念也被引入到神经网络
  - 受到人类视觉系统影响、并借鉴
  - 把神经网络理解成电路系统

卷积和相关运算

- 信号处理角度
  - 信号输入系统（线性、时不变），输出信号为输入信号与系统冲击响应的卷积
- 从数学角度
  - 两个函数上的运算、表示一个函数对另一个函数的形状的调整
  - 数学上的定义也是借鉴了信号处理的定义，不过去除了一些物理意义
- 无论哪个角度，卷积定义如下

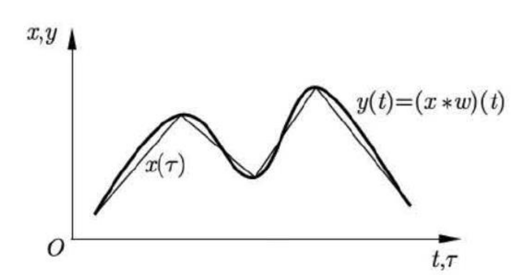
$$h(t) = (f \circledast g)(t) = \int_{-\infty}^{+\infty} f(\tau)\, g(t-\tau)\mathrm{d}\tau$$

卷积和相关

- 一个函数输入信号（函数），那么另一个为卷积核函数
  - 例子：高斯核函数w对信号x进行平滑（为什么？）

$$y(t) = (x \circledast w)(t) = \int_{-\infty}^{+\infty} x(\tau)\, w(t-\tau)\, \mathrm{d}\tau$$

$$w(t) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{t^2}{2\sigma^2}\right)$$



- 相关计算

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)\, g(t+\tau) \mathrm{d}\tau$$

  - 可以从相似度角度理解
  - 本课程采用卷积术语来称谓相关计算

离散二维卷积

- X为输入信号矩阵，w为卷积核矩阵，则二者卷积结果为

$$\boldsymbol{Y} = \boldsymbol{W} * \boldsymbol{X}$$

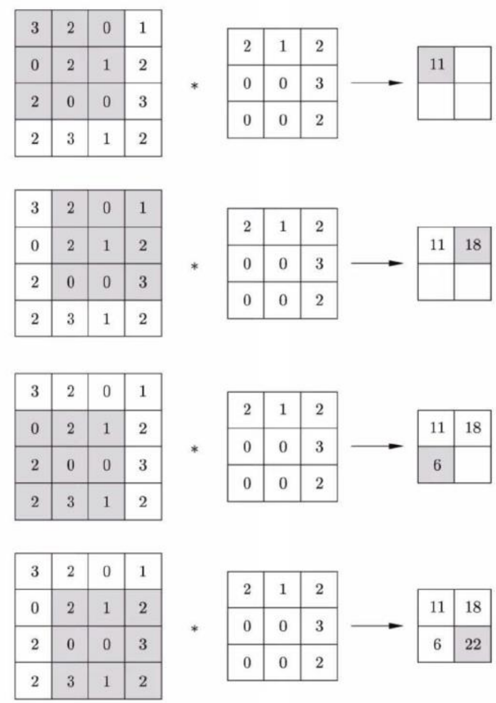
其中，

$$\boldsymbol{Y} = [y_{kl}]_{K \times L}$$

$$y_{kl} = \sum_{m=1}^{M} \sum_{n=1}^{N} w_{m,n} x_{k+m-1,l+n-1}$$

- 例子

$$\boldsymbol{Y} = \begin{bmatrix} 2 & 1 & 2 \\ 0 & 0 & 3 \\ 0 & 0 & 2 \end{bmatrix} * \begin{bmatrix} 3 & 2 & 0 & 1 \\ 0 & 2 & 1 & 2 \\ 2 & 0 & 0 & 3 \\ 2 & 3 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 11 & 18 \\ 6 & 22 \end{bmatrix}$$

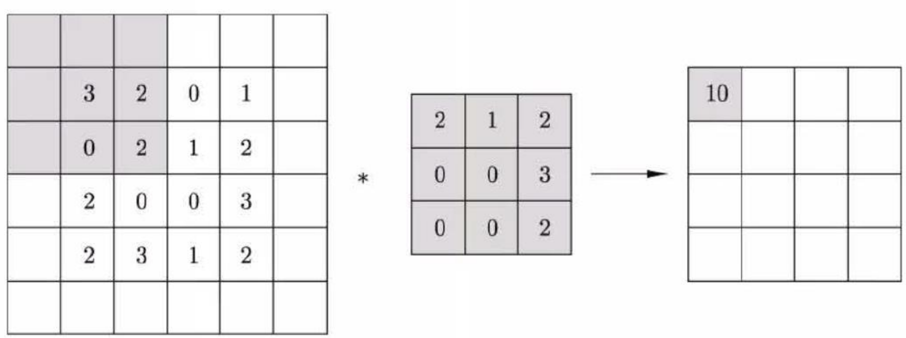


填充和步幅

- 上述卷积操作会使原始信号范围变小
  - 解决方案是在输入矩阵四周进行填充

$$\tilde{X} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 2 & 0 & 1 & 0 \\ 0 & 0 & 2 & 1 & 2 & 0 \\ 0 & 2 & 0 & 0 & 3 & 0 \\ 0 & 2 & 3 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

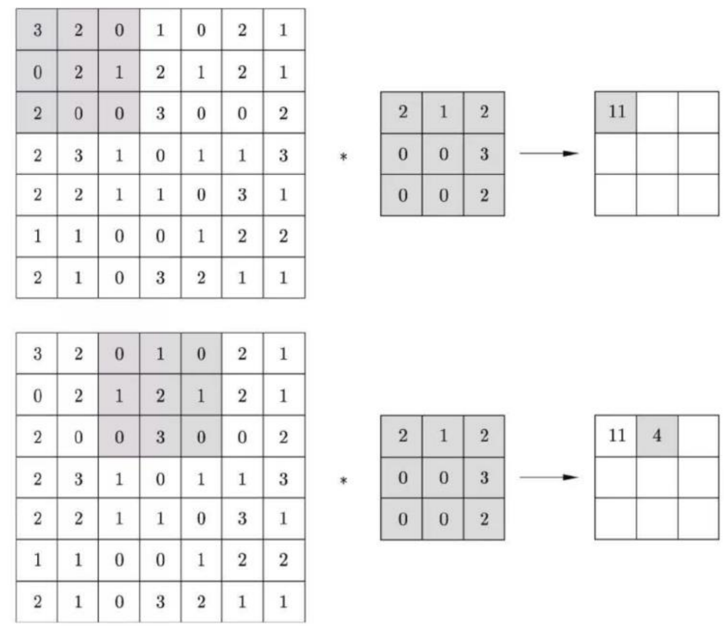
$$Y = \begin{bmatrix} 2 & 1 & 2 \\ 0 & 0 & 3 \\ 0 & 0 & 2 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 2 & 0 & 1 & 0 \\ 0 & 0 & 2 & 1 & 2 & 0 \\ 0 & 2 & 0 & 0 & 3 & 0 \\ 0 & 2 & 3 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 10 & 2 & 7 & 0 \\ 13 & 11 & 18 & 1 \\ 10 & 6 & 22 & 4 \\ 11 & 7 & 12 & 3 \end{bmatrix}$$



填充和步幅

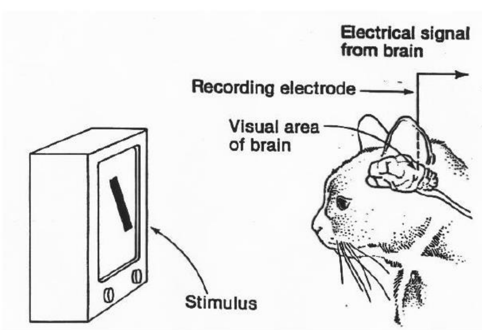
- 步幅：卷积核在输入信号矩阵上滑动时，每步间的水平和竖直方向上的步长

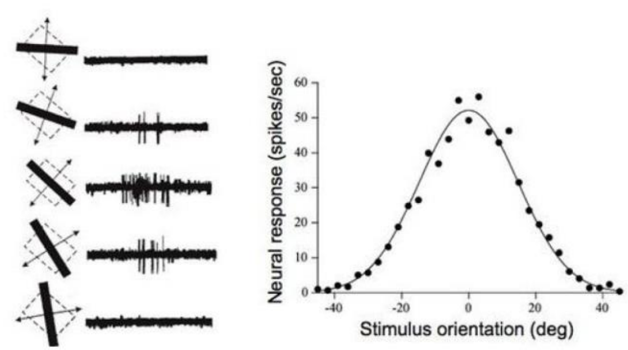
$$
\boldsymbol{Y}=\begin{bmatrix}2&1&2\\0&0&3\\0&0&2\end{bmatrix}*\begin{bmatrix}3&2&0&1&0&2&1\\0&2&1&2&1&2&1\\2&0&0&3&0&0&2\\2&3&1&0&1&1&3\\2&2&1&1&0&3&1\\1&1&0&0&1&2&2\\2&1&0&3&2&1&1\end{bmatrix}=\begin{bmatrix}11&4&11\\9&6&15\\8&10&13\end{bmatrix}
$$



引入卷积的目的

- 图像数据中，相同的一些底层结构反复出现，同时复杂的对象有时是基本结构的组合结果
  - 因此，设计小尺度的卷积核，并让其在输入矩阵上滑动，并计算相关结果
  - 而网络的层次结构也为结构的组合带来便利
- 减少参数量
  - 全连接网络中的参数量很大
- 受到哺乳动物视觉系统启发



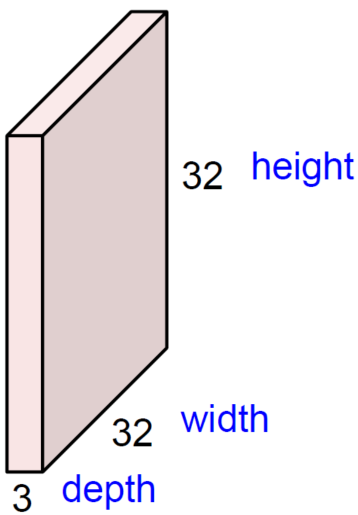


卷积网络

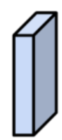
- 卷积层

Filters的范围与输入信号的depth一致

32x32x3 image



5x5x3 filter



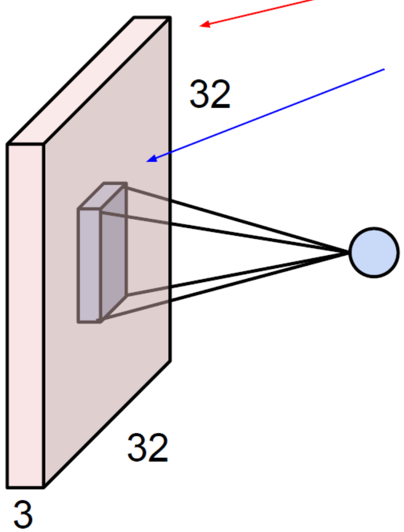
Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

卷积网络

- 卷积层

32x32x3 image

5x5x3 filter w

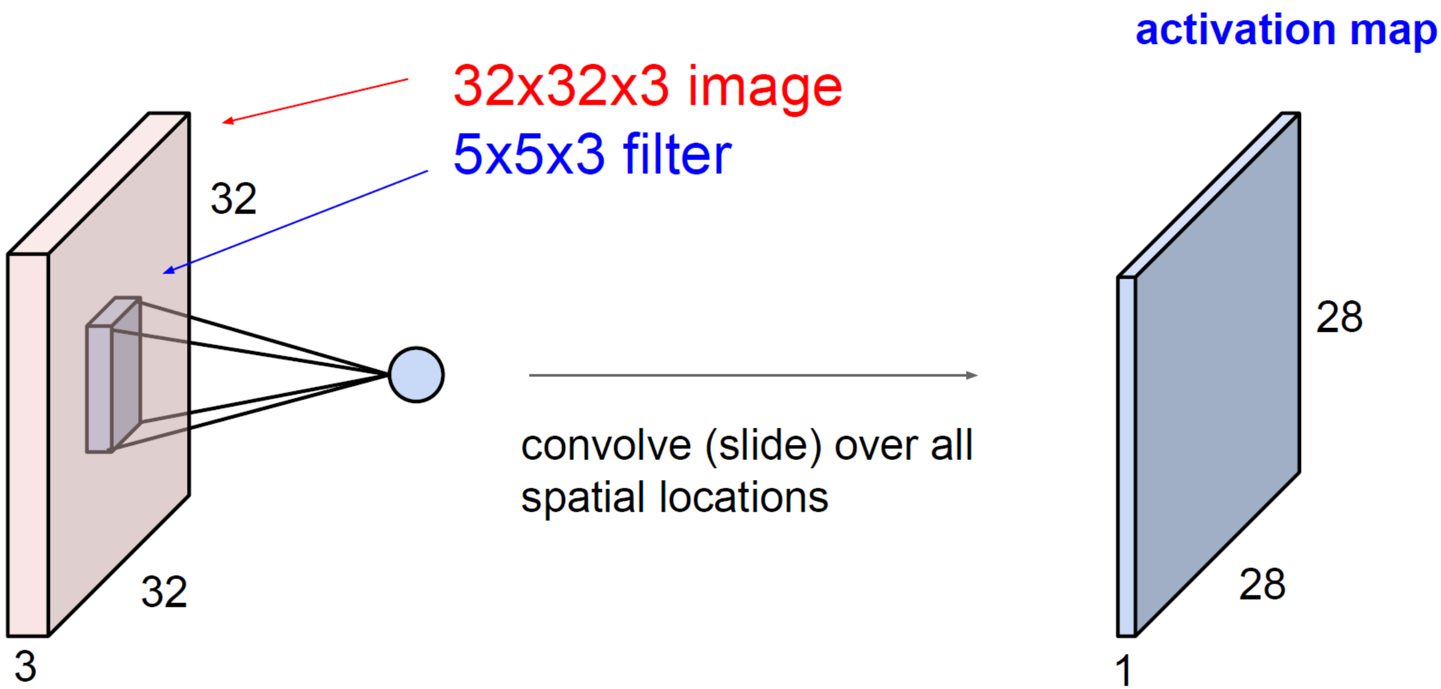


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

卷积网络

- 卷积层（三维卷积）

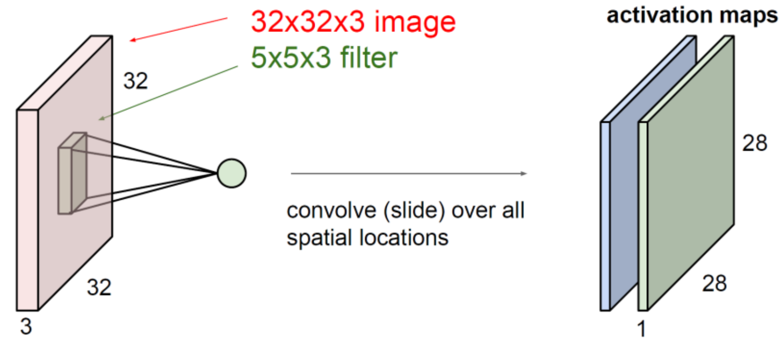


$$Y = X * W = X_R * W_R + X_G * W_G + X_B * W_B$$

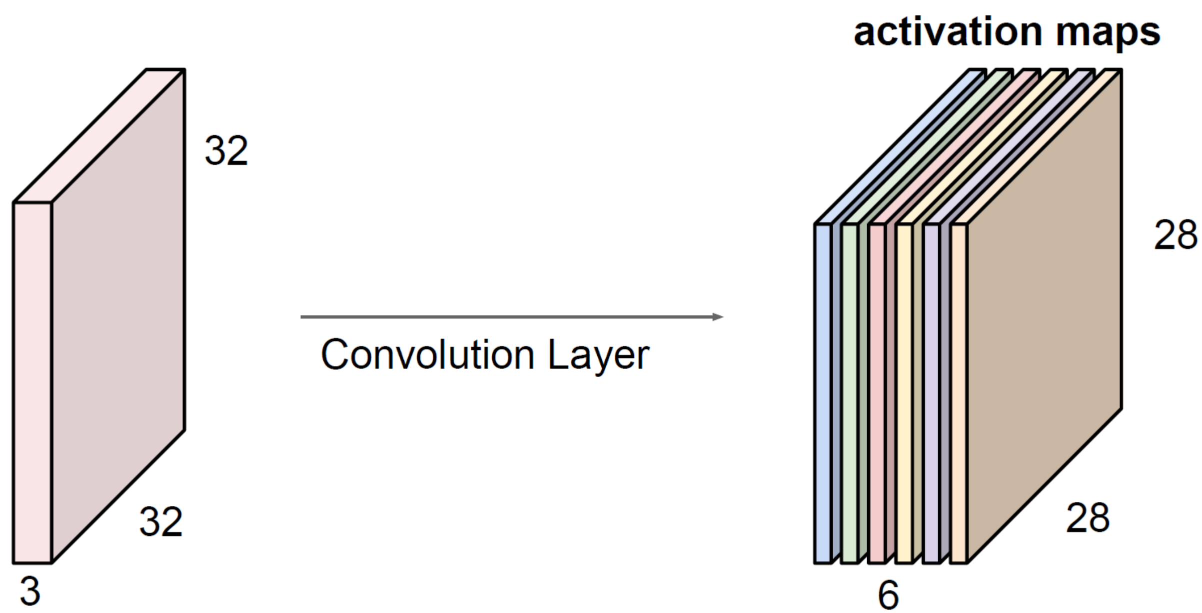
$$W = (W_R, W_G, W_B)$$

3个二维卷积核（构成一个三维卷积核）

卷积网络



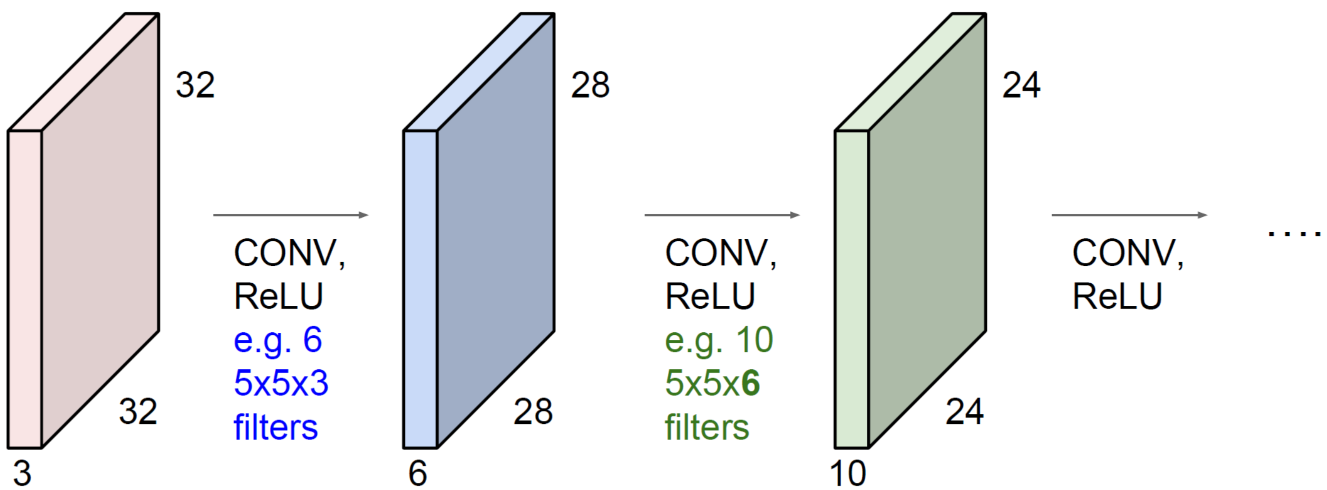
- 卷积层
  - 更多的卷积核，例如可以构造一个6个channel（depth）的新图像



  - Channel类比于彩色图像中的RGB

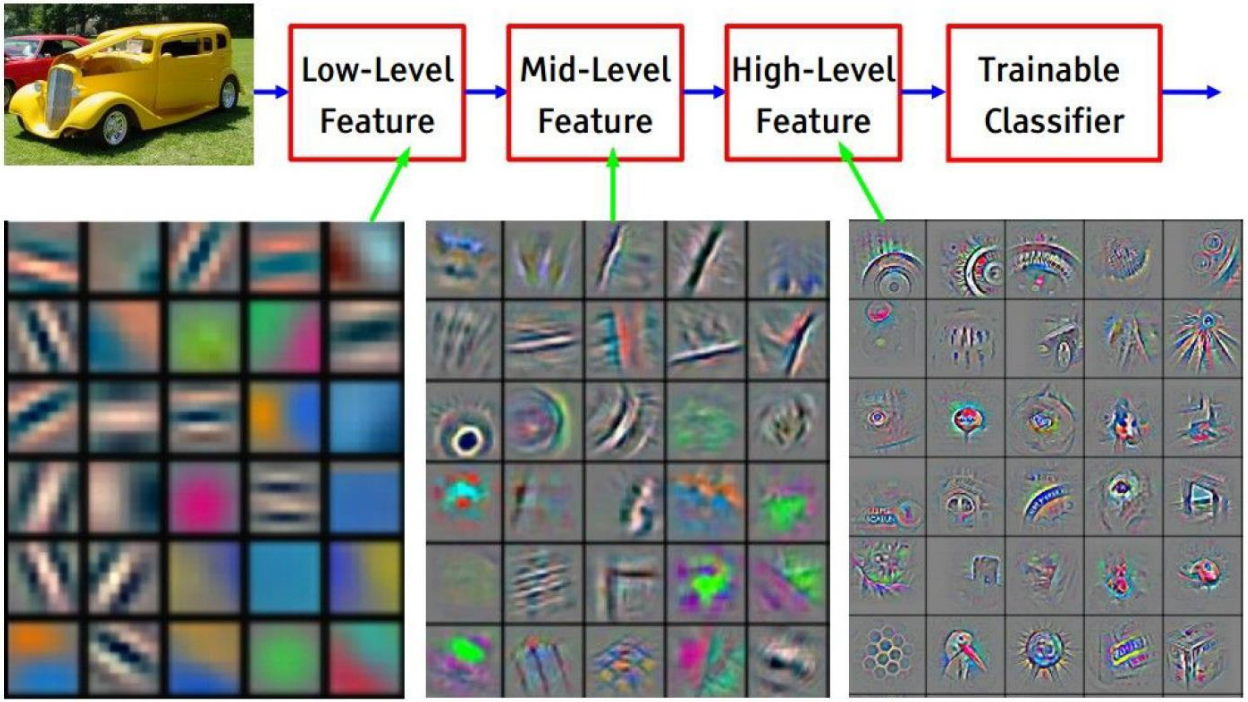
卷积网络

- 卷积网络是一系列卷积+激活函数构造而成
  - 级联多层的意义？
  - 为何卷积后要有激活函数？



卷积网络

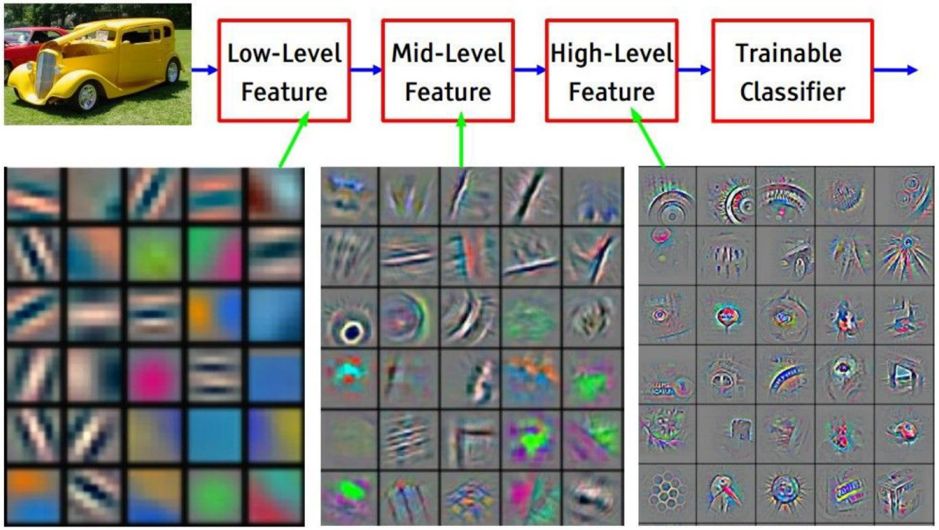
- 卷积核的语义解释



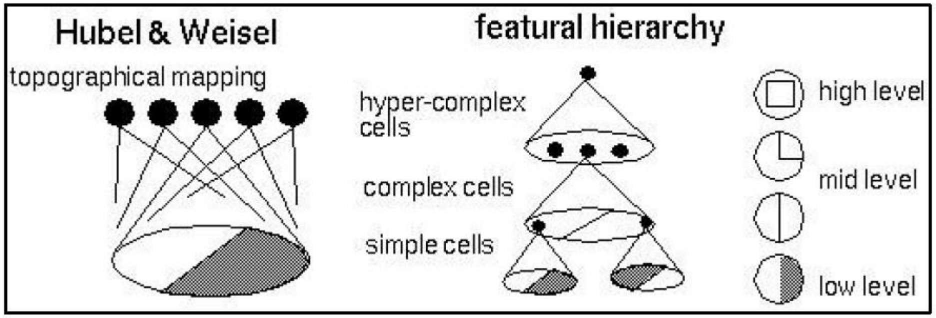
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

卷积网络

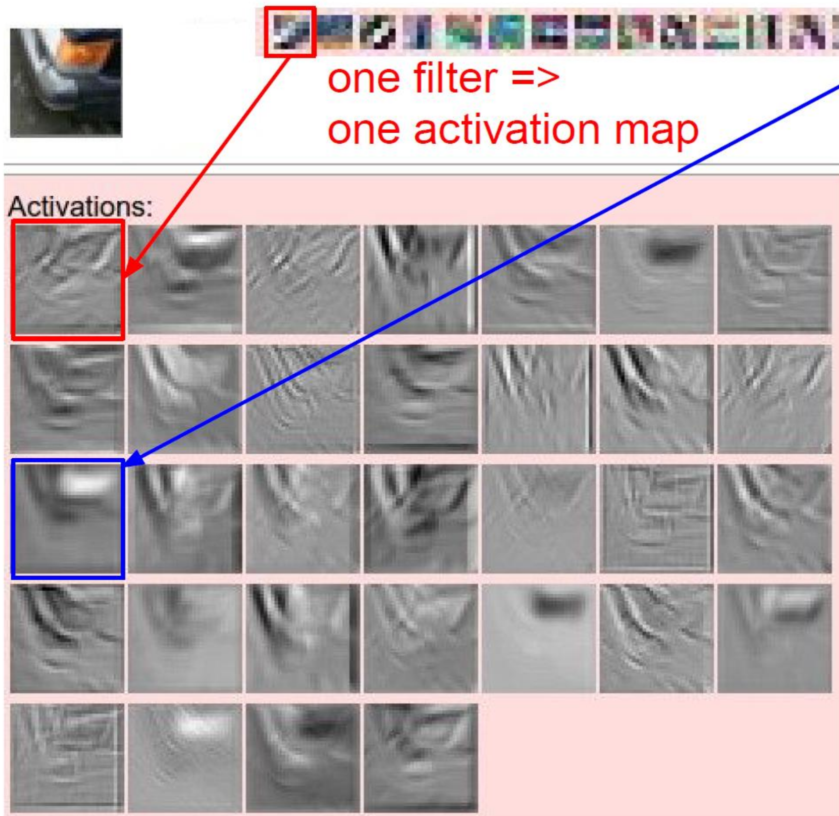
- 卷积核的语义解释
- 与视觉系统的对比（但是更深的网络意味着什么？）



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



卷积网络



example 5x5 filters
(32 total)

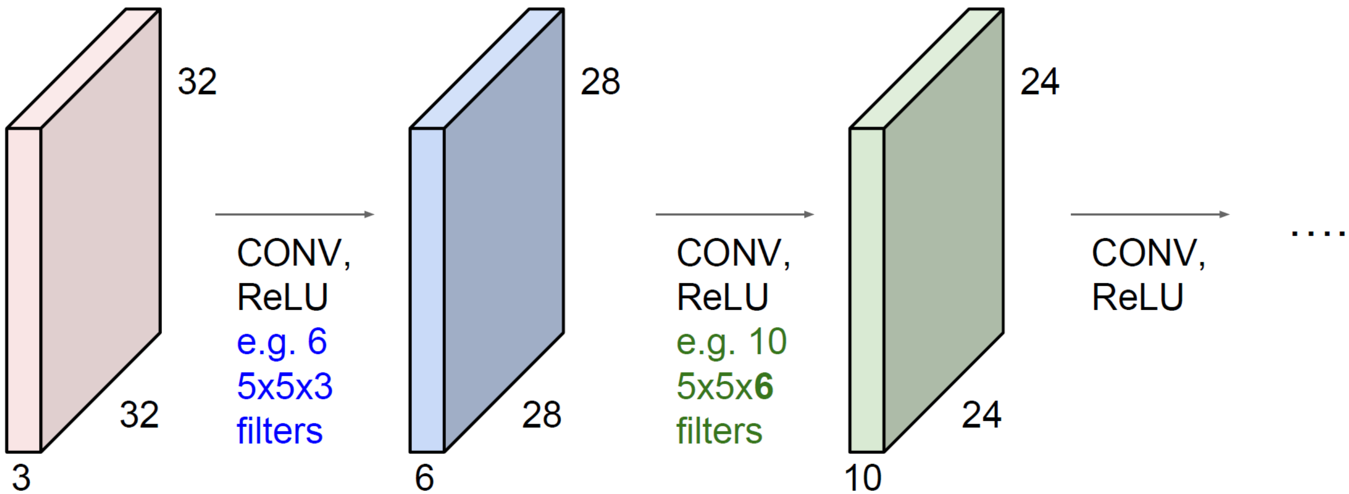
We call the layer convolutional because it is related to convolution of two signals:

$$y_{kl} = \sum_{m=1}^{M} \sum_{n=1}^{N} w_{m,n} x_{k+m-1, l+n-1}$$

elementwise multiplication and sum of a filter and the signal (image)

卷积网络

- 填充操作的意义：如果没有padding，激活图会越来越小，性能变差



输出卷积层尺寸和参数量

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size:

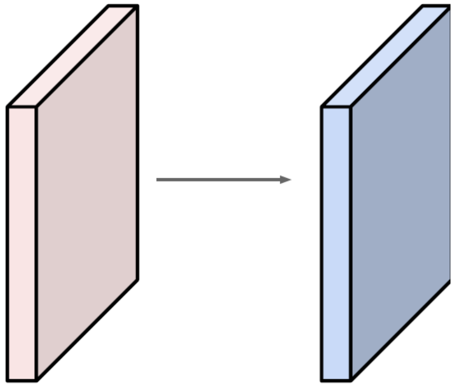
(32+2*2-5)/1+1 = 32 spatially, so

32x32x10

Number of parameters in this layer?

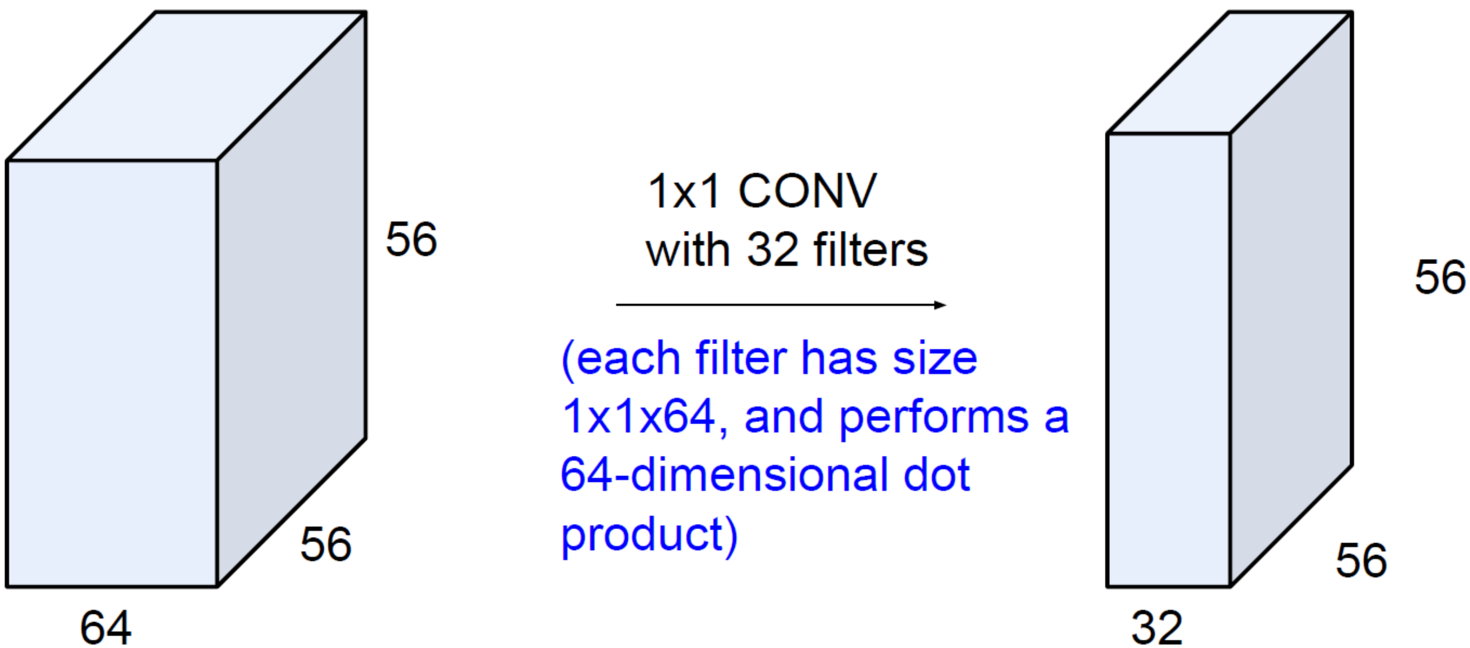
each filter has 5*5*3 + 1 = 76 params (+1 for bias)

=> 76*10 = **760**



卷积网络

- 1x1卷积



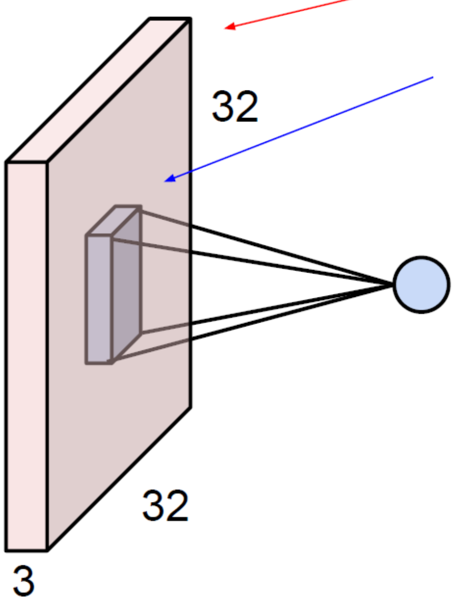
56
56
64
1x1 CONV
with 32 filters
(each filter has size
1x1x64, and performs a
64-dimensional dot
product)
56
56
32

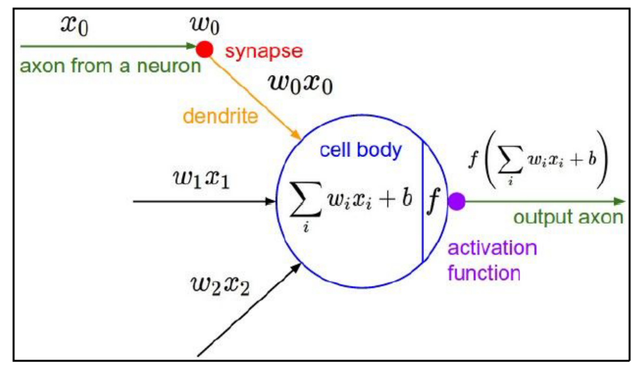
卷积网络

- 卷积层的神经观点

32x32x3 image

5x5x3 filter



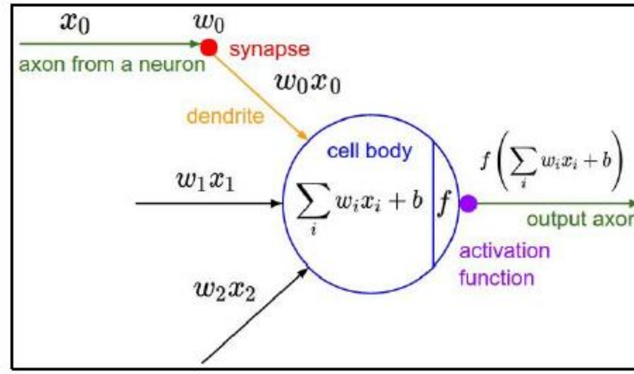


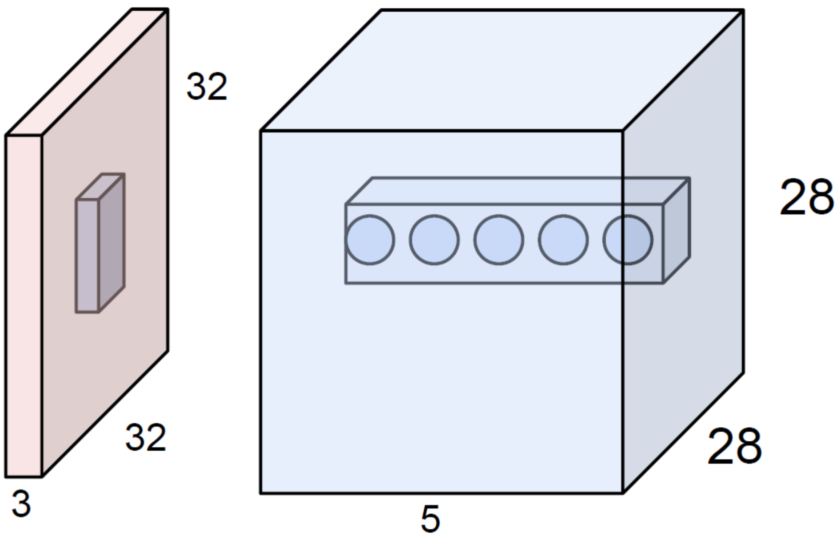
It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

卷积网络

- 卷积层的神经观点

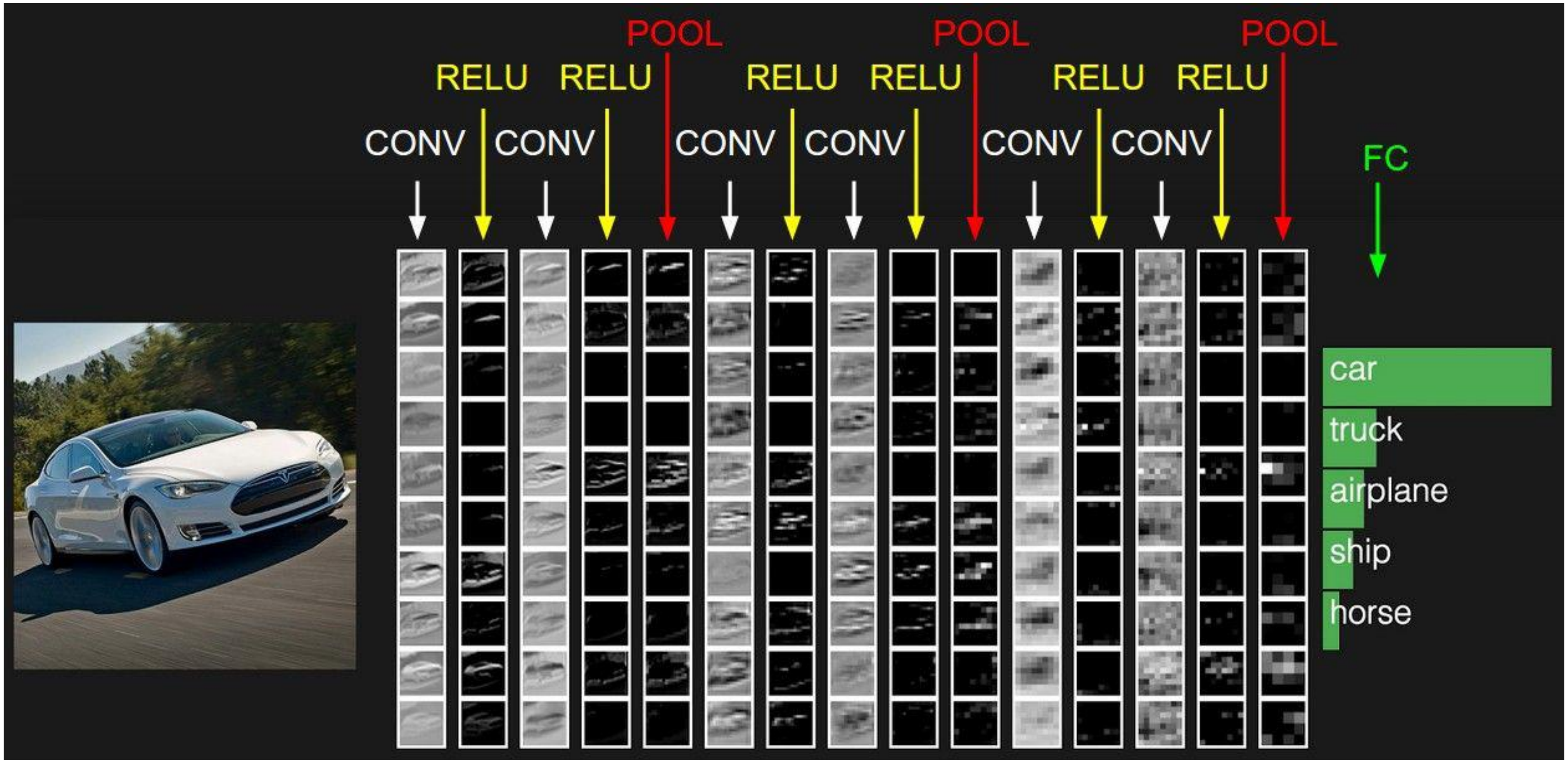




E.g. with 5 filters, CONV layer consists of neurons arranged in a 3D grid (28x28x5)

There will be 5 different neurons all looking at the same region in the input volume

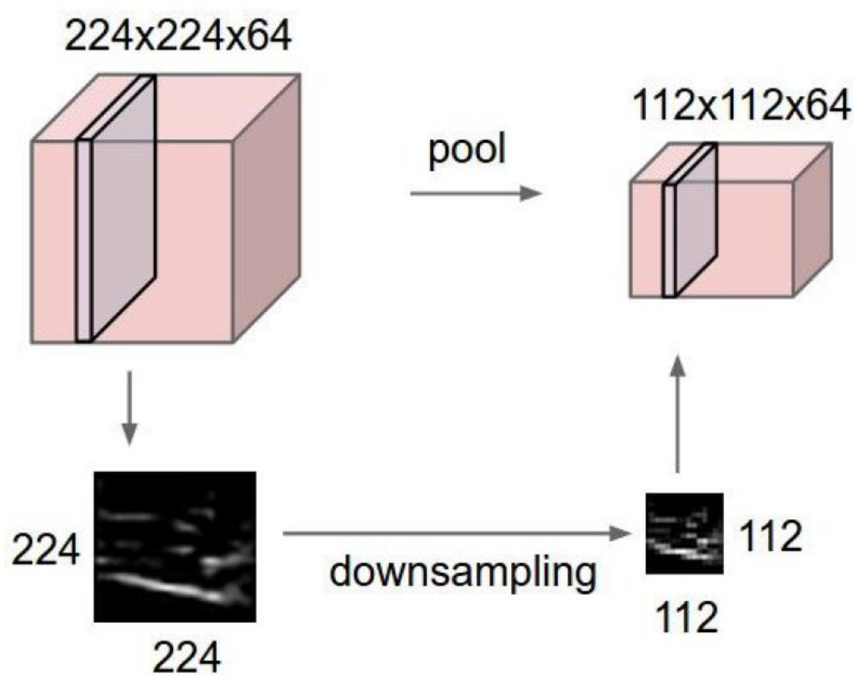
一个多类分类神经网络



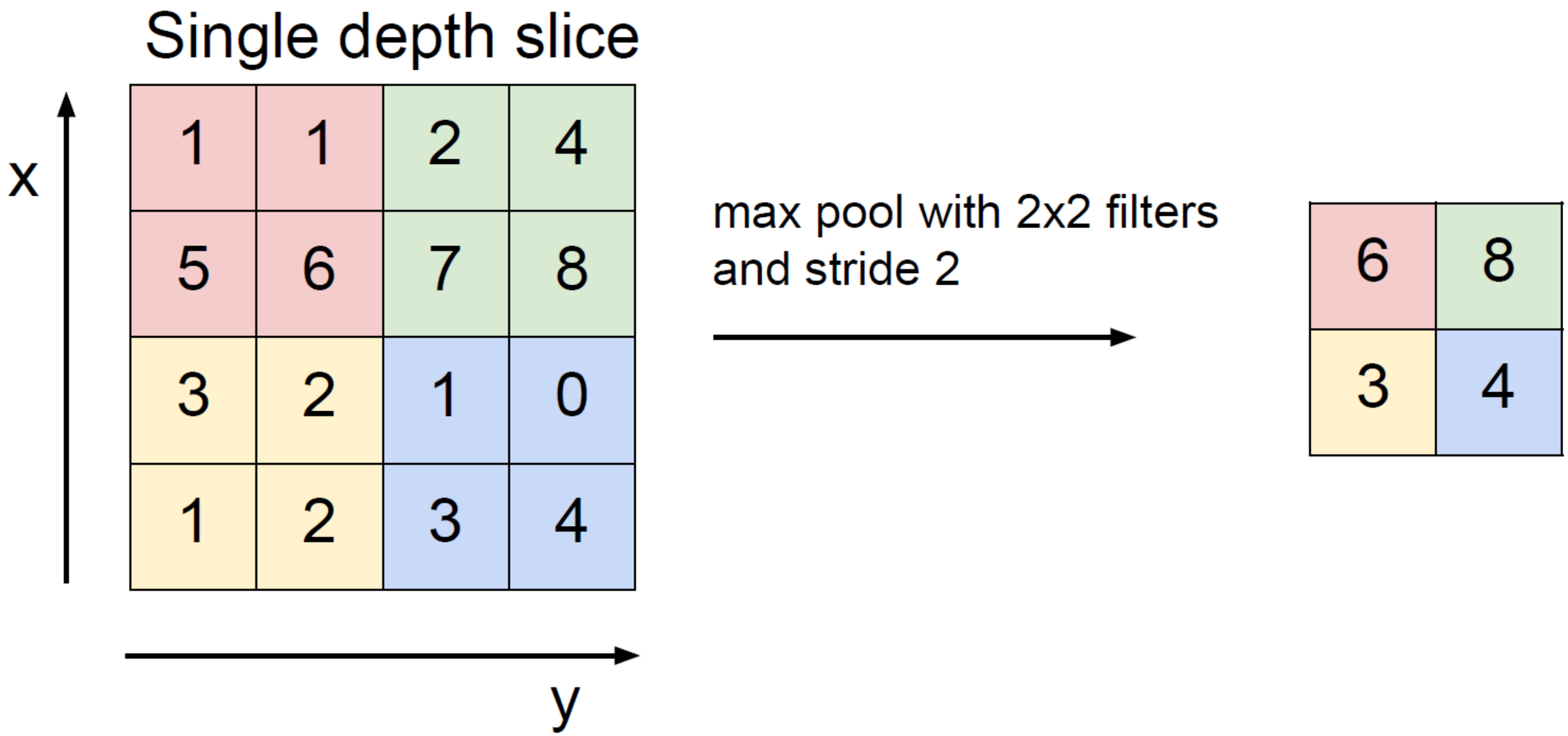
POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Pooling

- 使表示更小并且更好操作

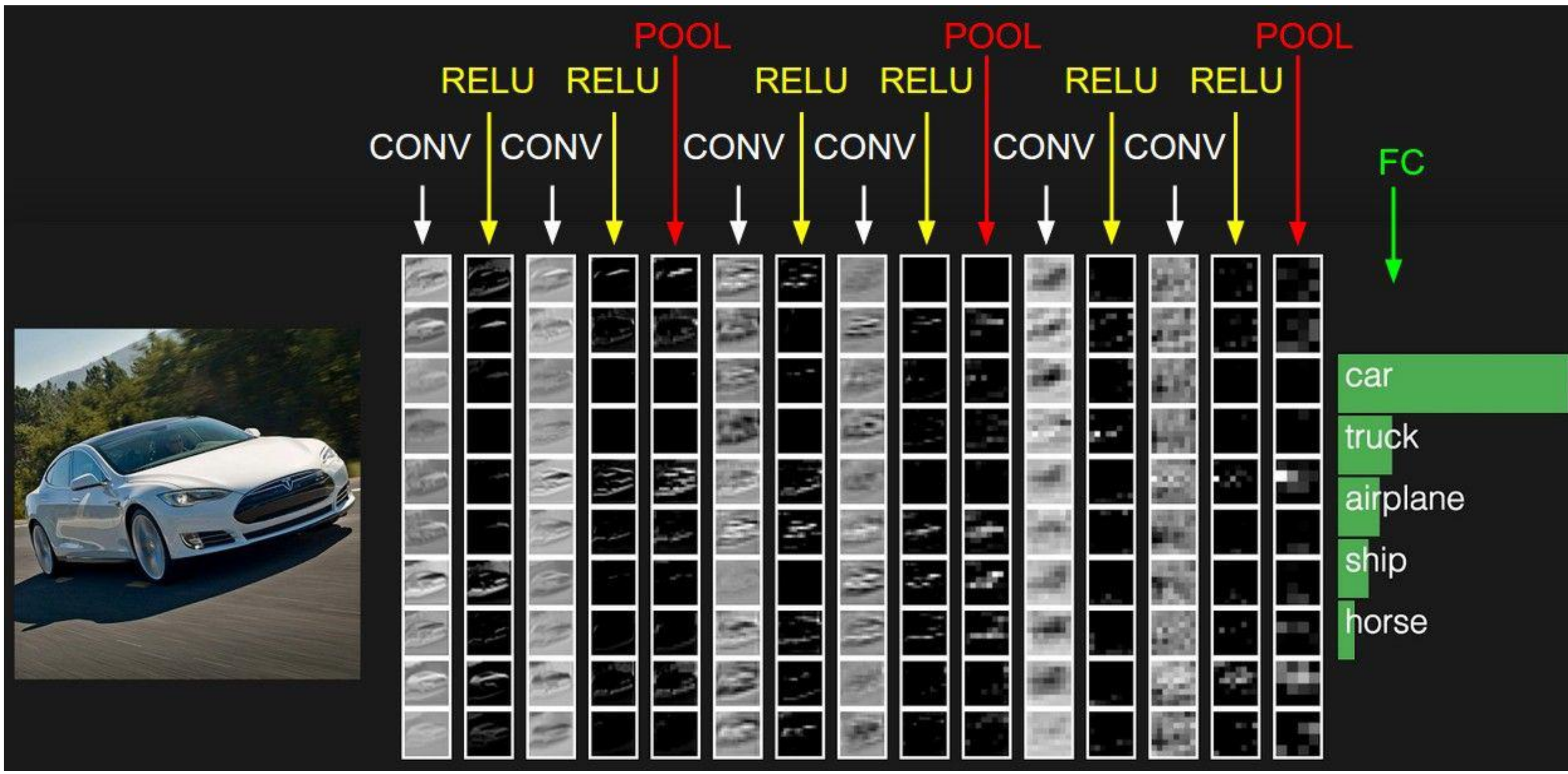


Max pooling


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

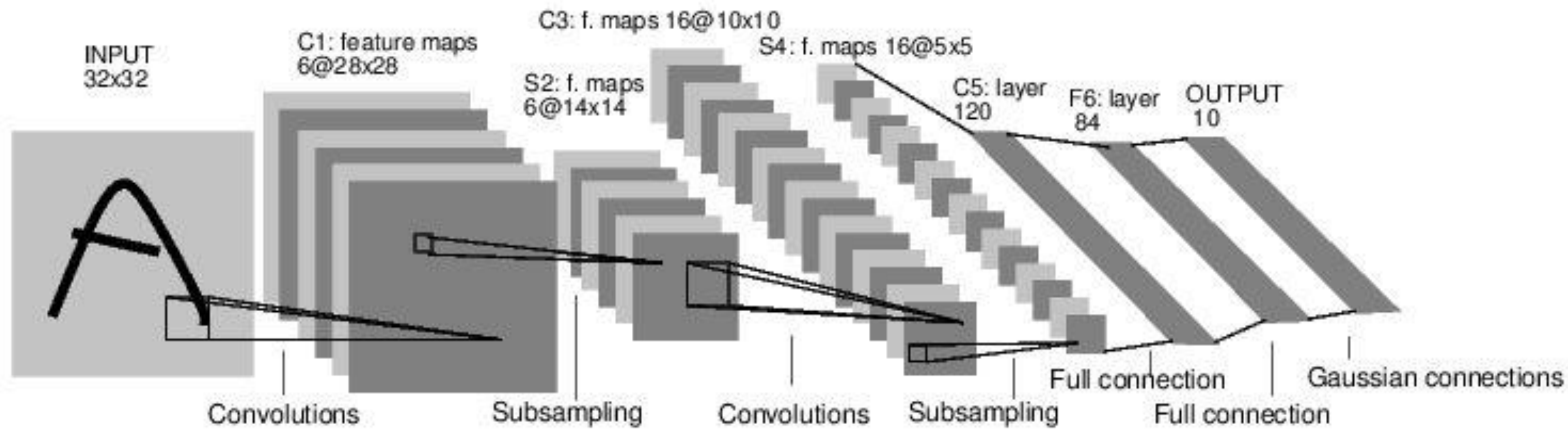
全连接层

- 每个神经元链接上一层的所有神经元



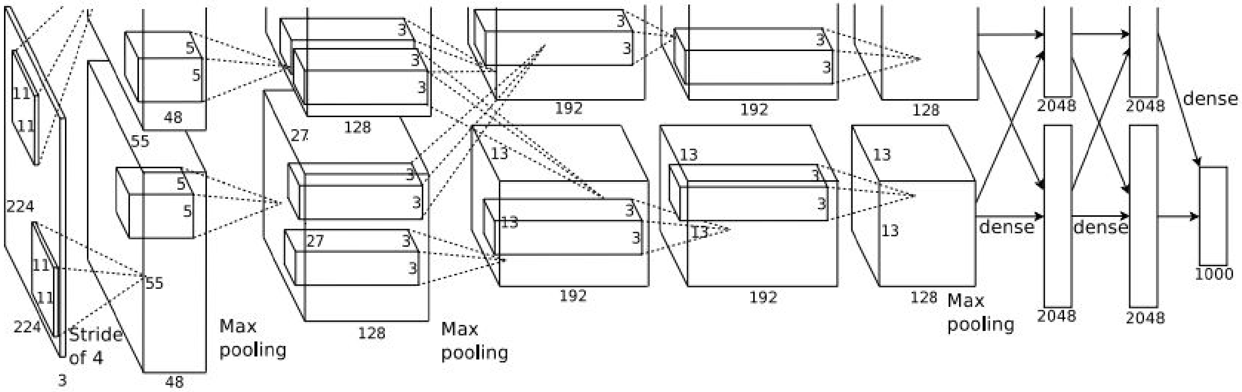
LeNet-5

- LeCun 1998

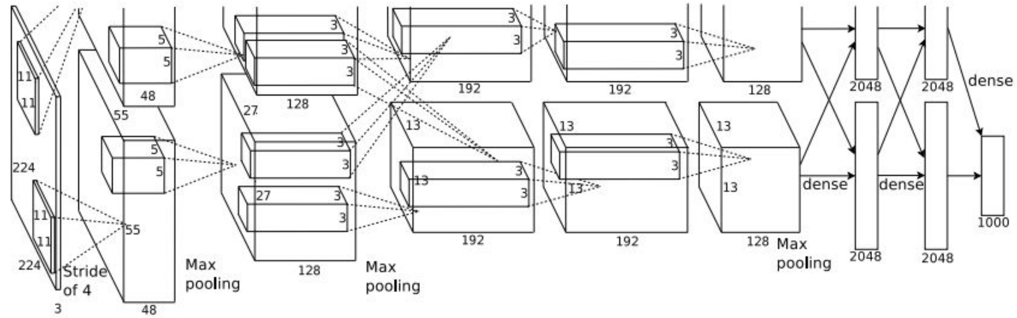


AlexNet

- Input: 227x227x3 images
- After CONV1: 55x55x96
- **Second layer** (POOL1): 3x3 filters applied at stride 2
- Output volume: 27x27x96
- Parameters: 0!



AlexNet



Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

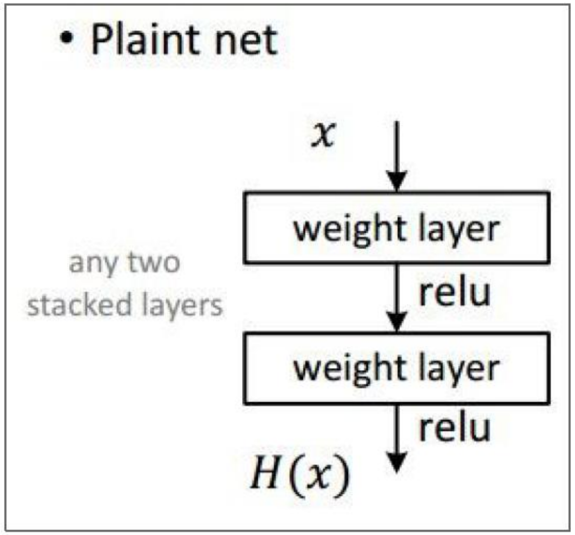
[4096] FC7: 4096 neurons

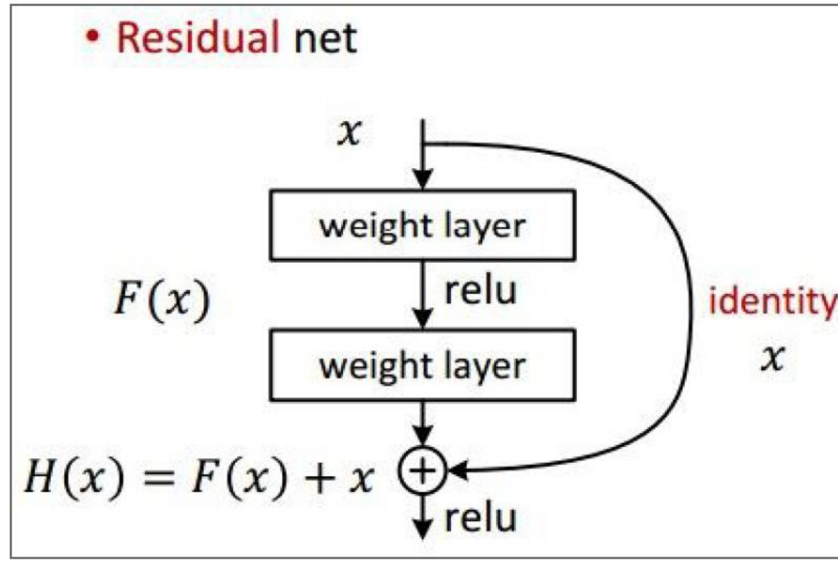
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:

- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

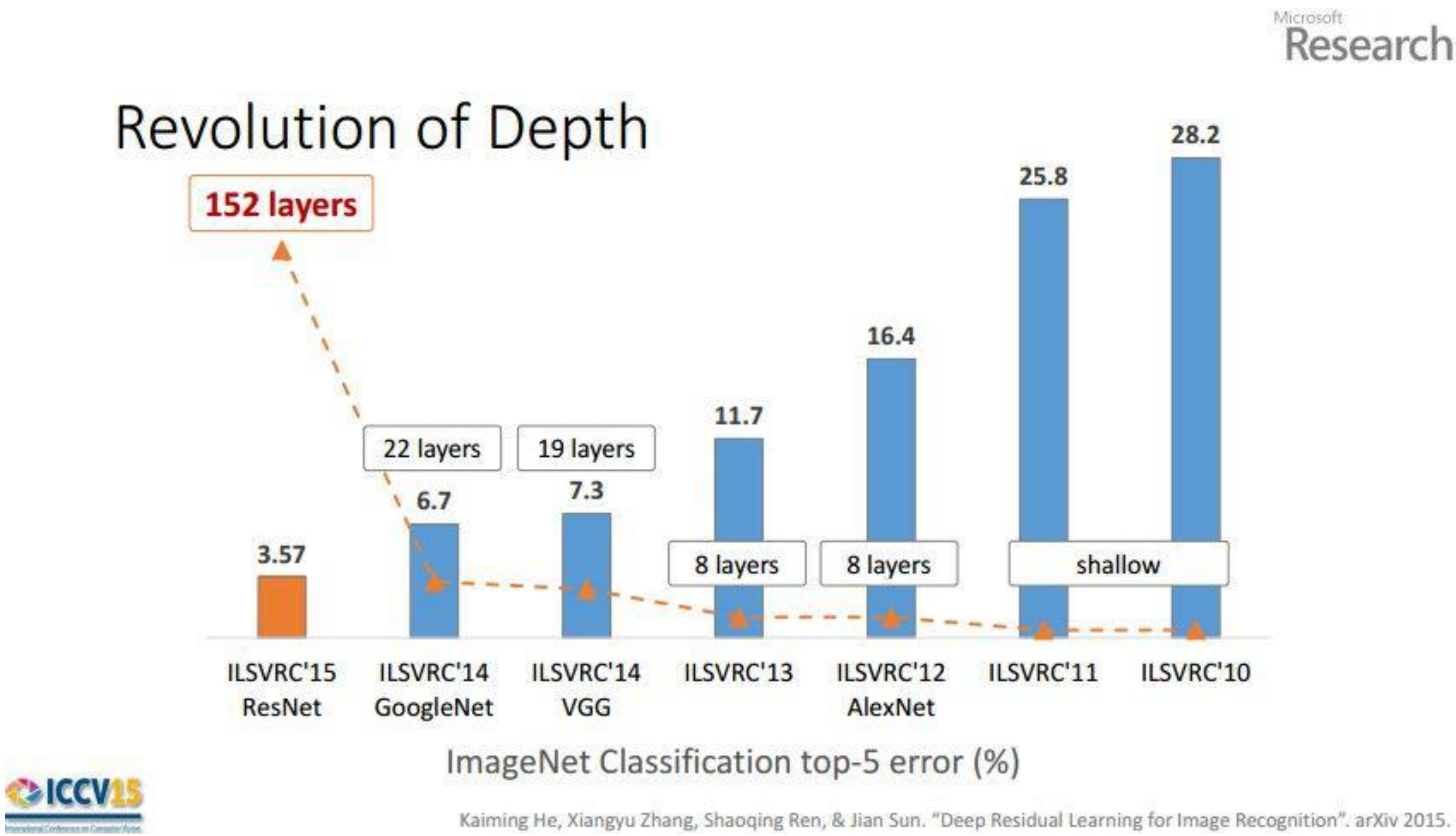
残差网络（He 2015）





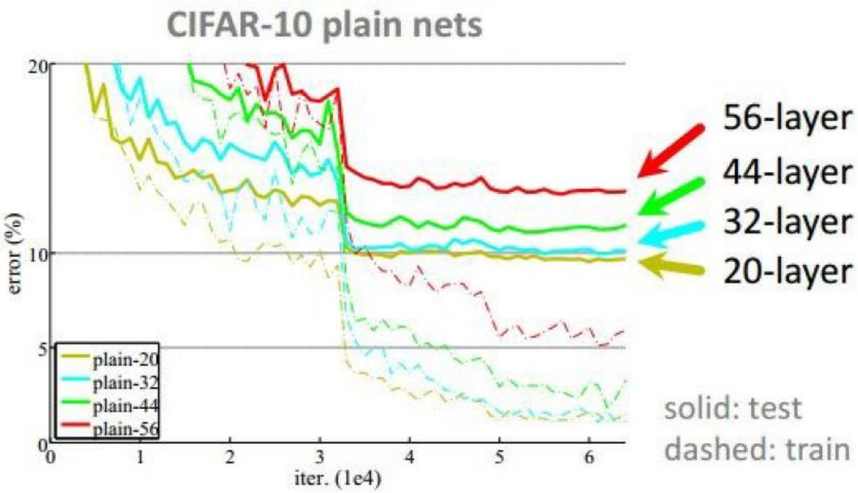
- 为什么这么连接?

加大神经网络的层数有意义

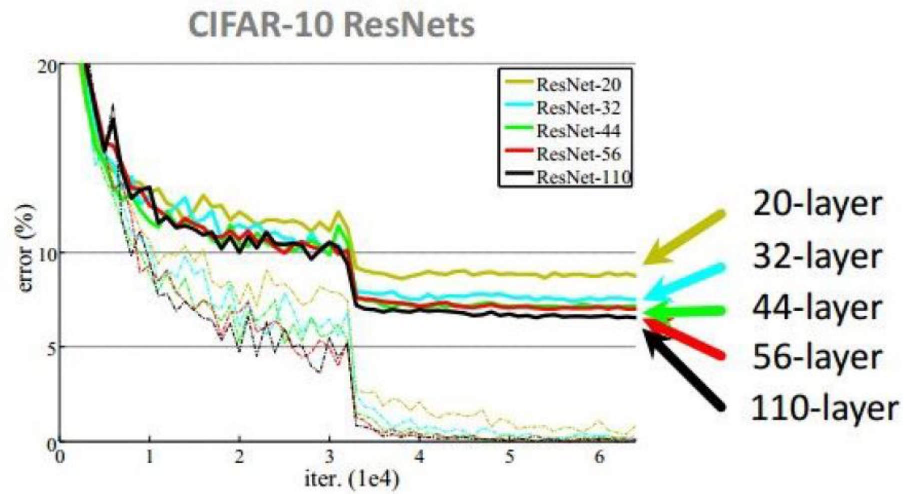


slide from Kaiming He's

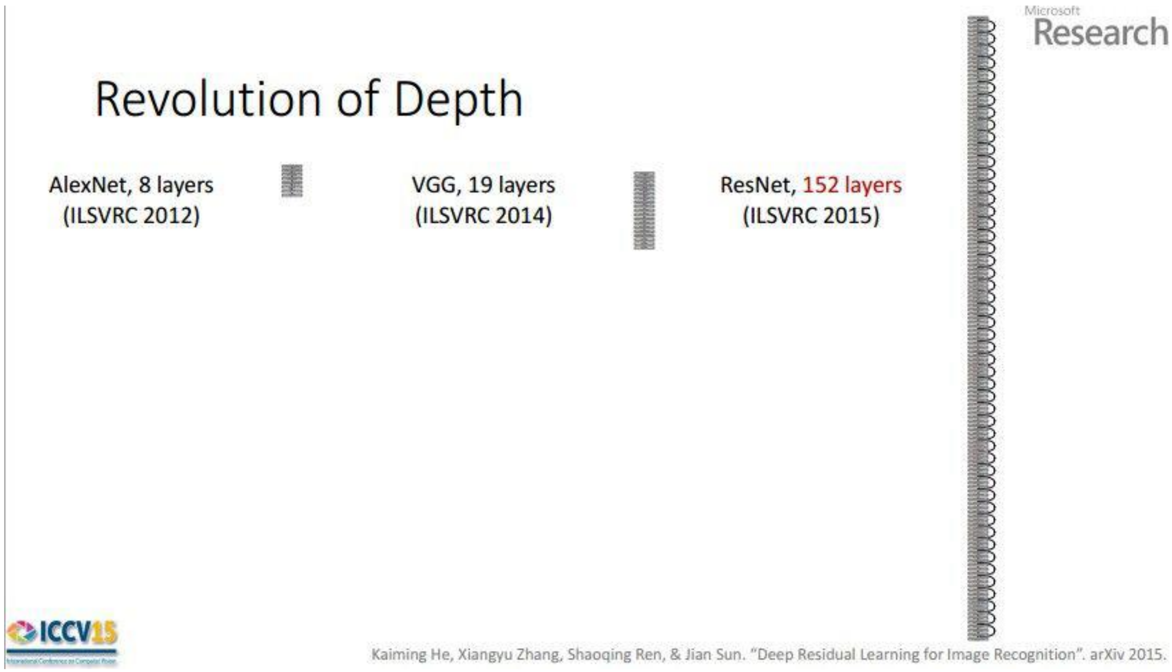
CIFAR-10实验



CIFAR-10 plain nets
error (%)
iter. (1e4)
plain-20
plain-32
plain-44
plain-56
56-layer
44-layer
32-layer
20-layer
solid: test
dashed: train



CIFAR-10 ResNets
error (%)
iter. (1e4)
ResNet-20
ResNet-32
ResNet-44
ResNet-56
ResNet-110
20-layer
32-layer
44-layer
56-layer
110-layer



2-3 weeks of training on 8 GPU machine

at runtime: faster than a VGGNet! (even though it has 8x more layers)

- Batch Normalization after every CONV layer
- Xavier/2 initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used