计算机视觉与卷积神经网络

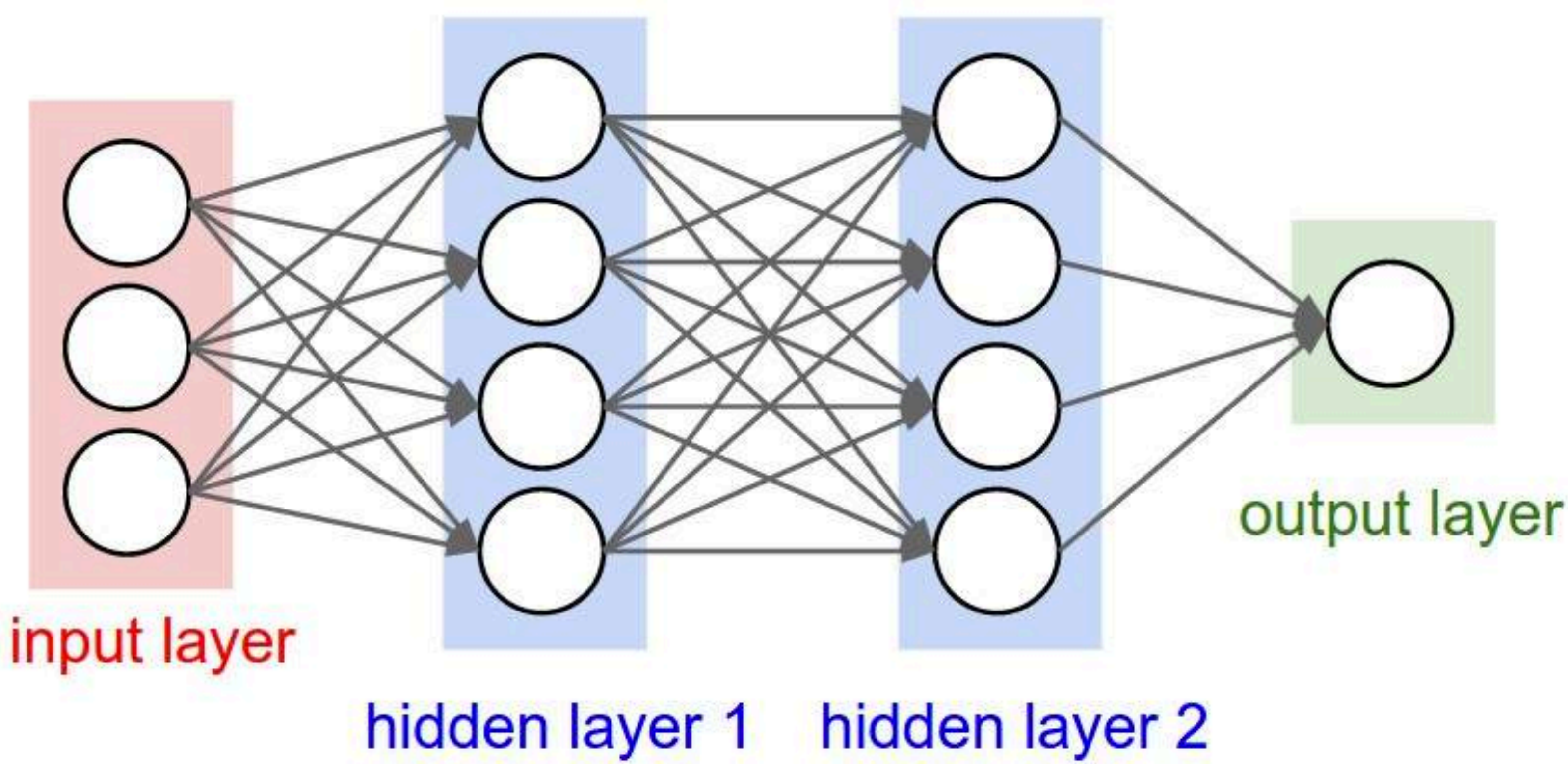
七月在线 寒小阳
2016年12月10日

主要内容

- 神经网络与卷积神经网络
  1. 层级结构
  2. 数据处理
  3. 训练算法
  4. 优缺点
- 正则化与Dropout
  1. 正则化与Dropout处理
  2. Dropout理解
- 典型结构与训练
  1. 典型CNN
  2. 训练与调优

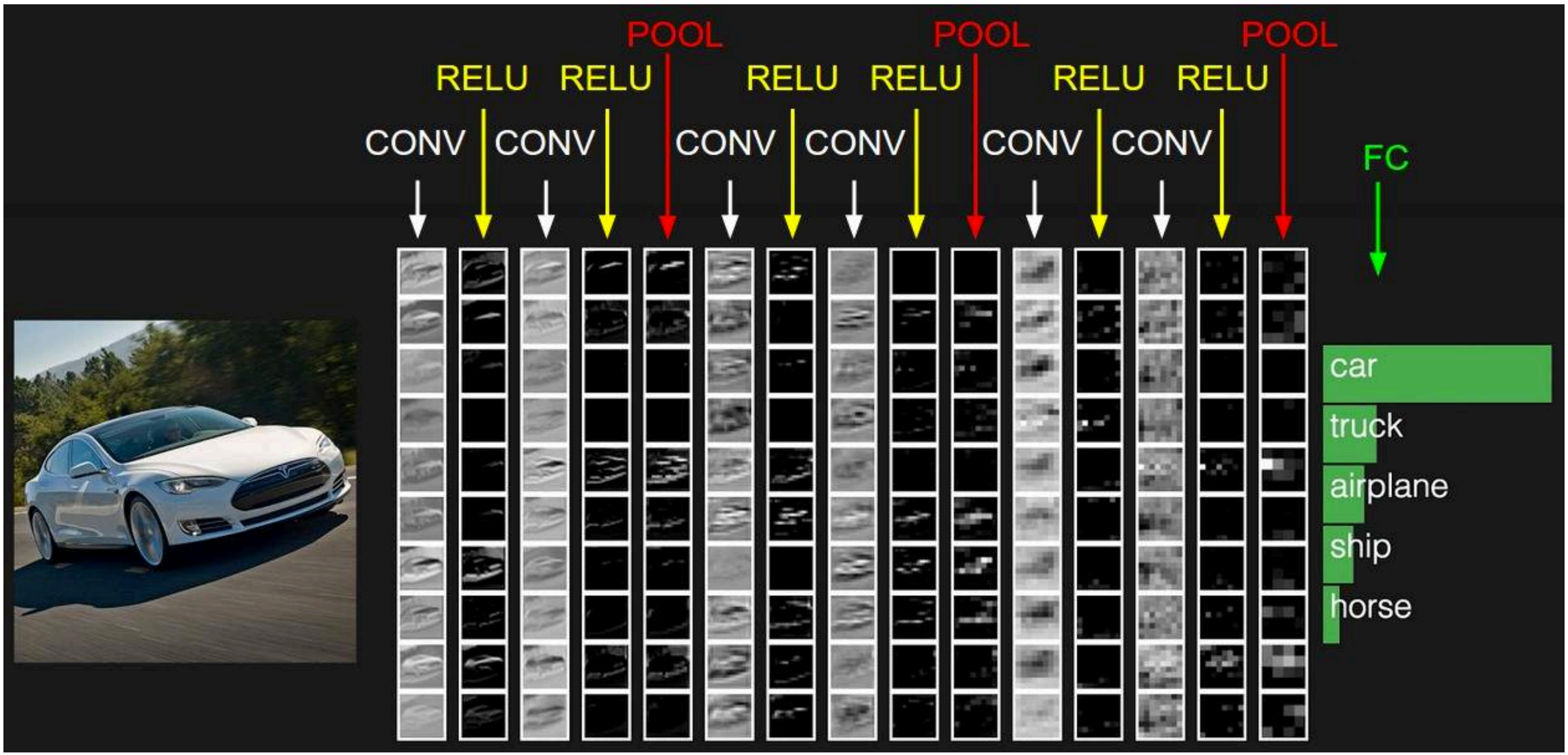
神经网络到卷积神经网络

- 人工神经网络能用到计算机视觉上吗？ 能
- 为什么还需要卷积神经网络？
- 卷积神经网络和人工神经网络的差异在哪？



卷积神经网络层级结构

- 保持了层级网络结构
- 不同层次有不同形式(运算)与功能



主要是以下层次

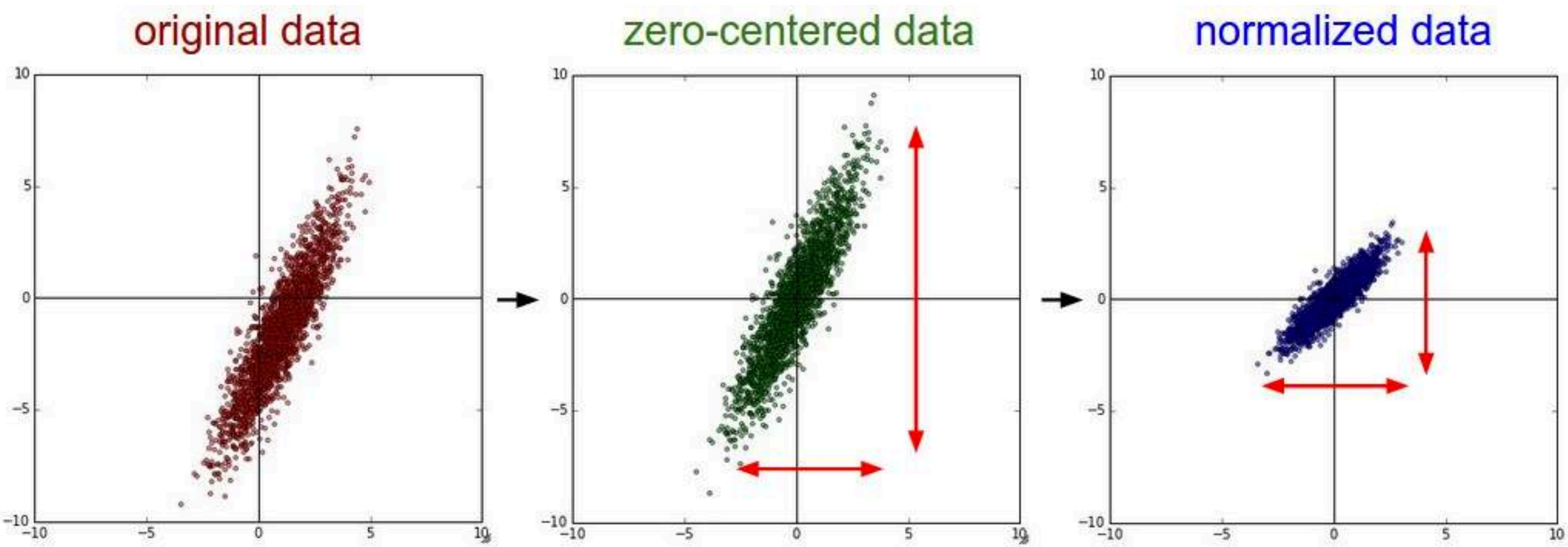
- 数据输入层/ Input layer
- 卷积计算层/ CONV layer
- ReLU激励层 / ReLU layer
- 池化层 / Pooling layer
- 全连接层 / FC layer
- Batch Normalization层(可能有)

数据输入层/ Input layer

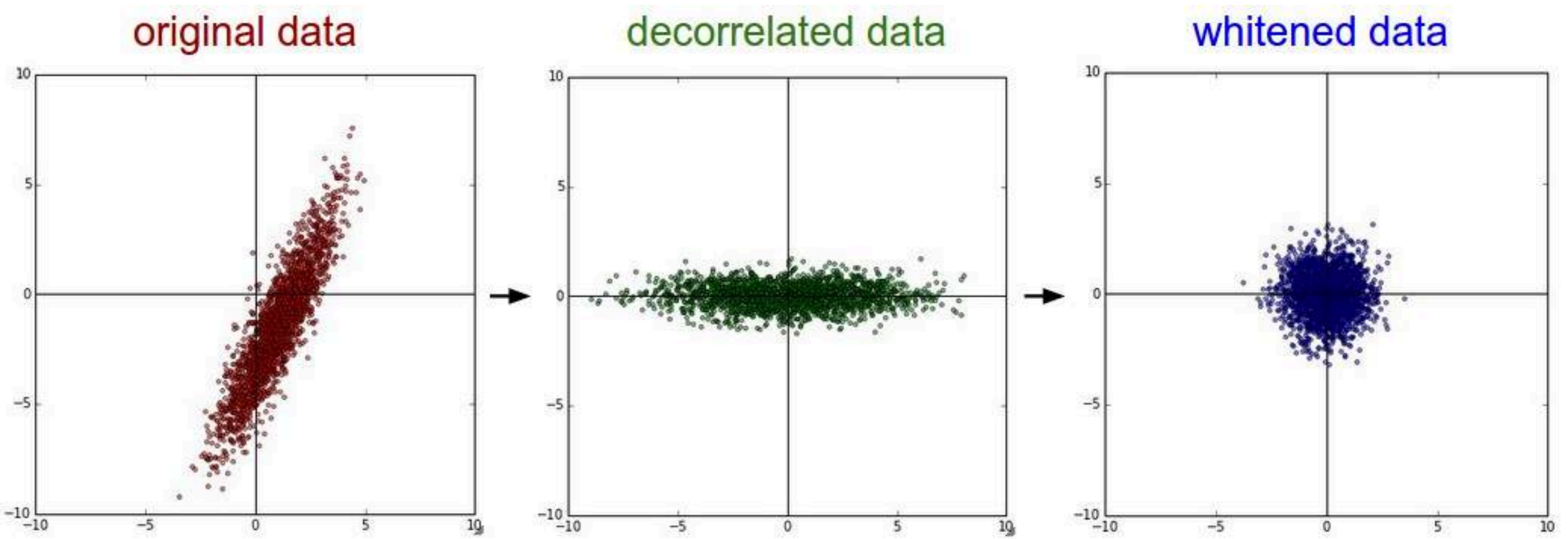
有3种常见的数据处理方式

- 去均值
  - 把输入数据各个维度都中心化到0
- 归一化
  - 幅度归一化到同样的范围
- PCA/白化
  - 用PCA降维
  - 白化是对数据每个特征轴上的幅度归一化

去均值与归一化


original data
zero-centered data
normalized data
10
5
0
−5
−10
−10
−5
0
5
10

□ 去相关与白化

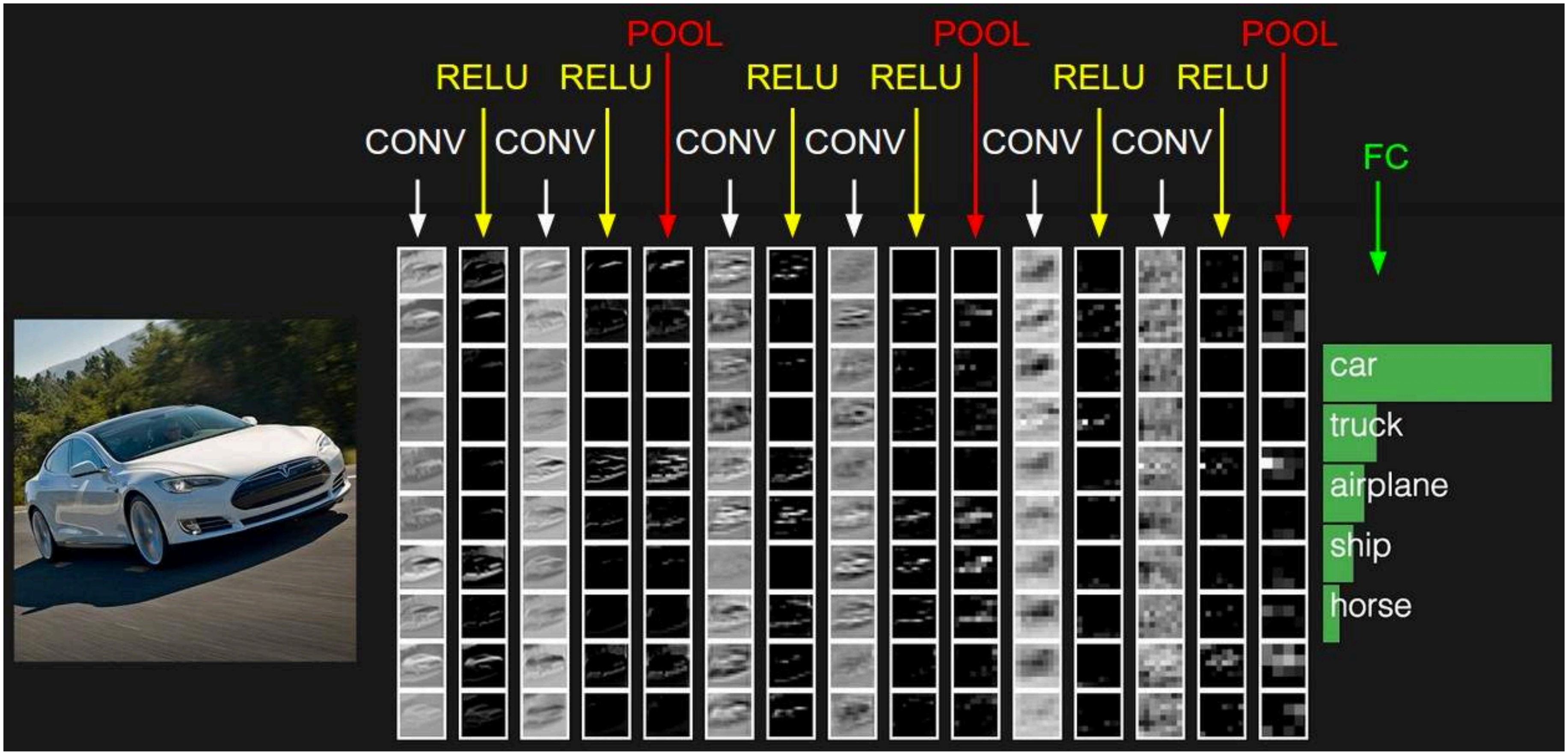



```
X -= np.mean(X, axis = 0)
cov = np.dot(X.T, X) / X.shape[0]
U,S,V = np.linalg.svd(cov)
Xrot = np.dot(X,U)
```

```
Xwhite = Xrot / np.sqrt(S + 1e-5)
```

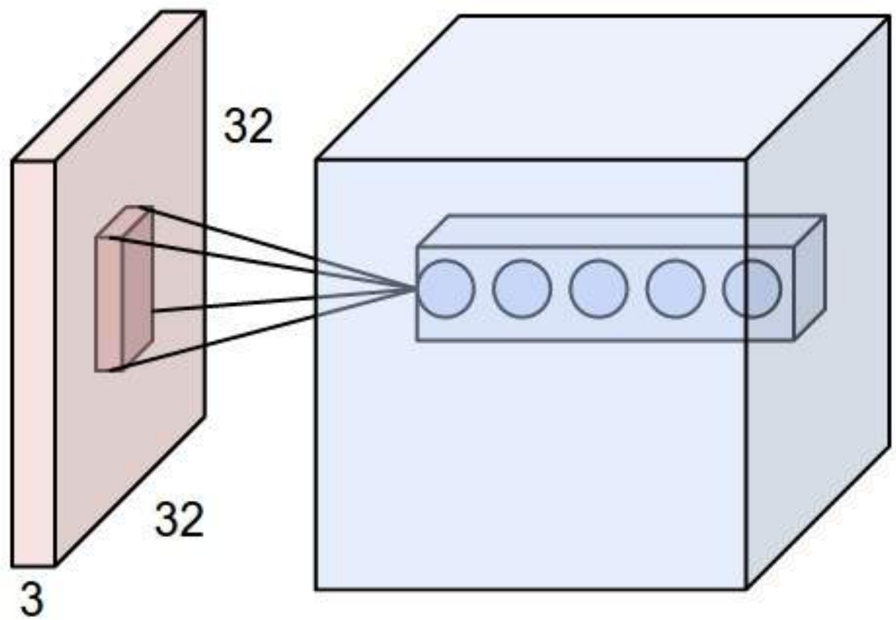
卷积神经网络层级结构

- 保持了层级网络结构
- 不同层次有不同形式(运算)与功能



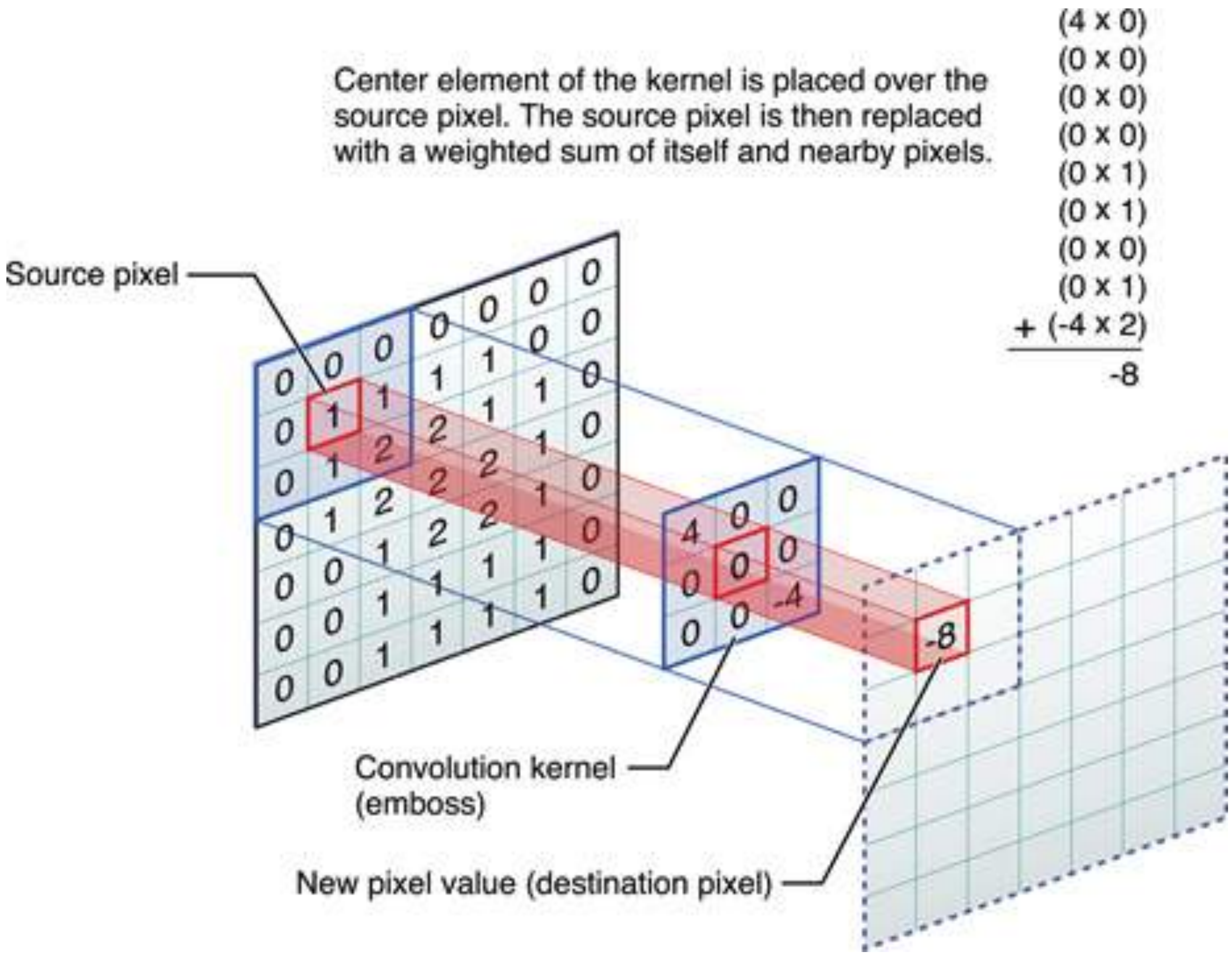
卷积计算层/ CONV layer

- 局部关联。每个神经元看做一个filter。
- 窗口(receptive field)滑动，filter对局部数据计算
- 涉及概念：
  - 深度/depth
  - 步长/stride
  - 填充值/zero-padding



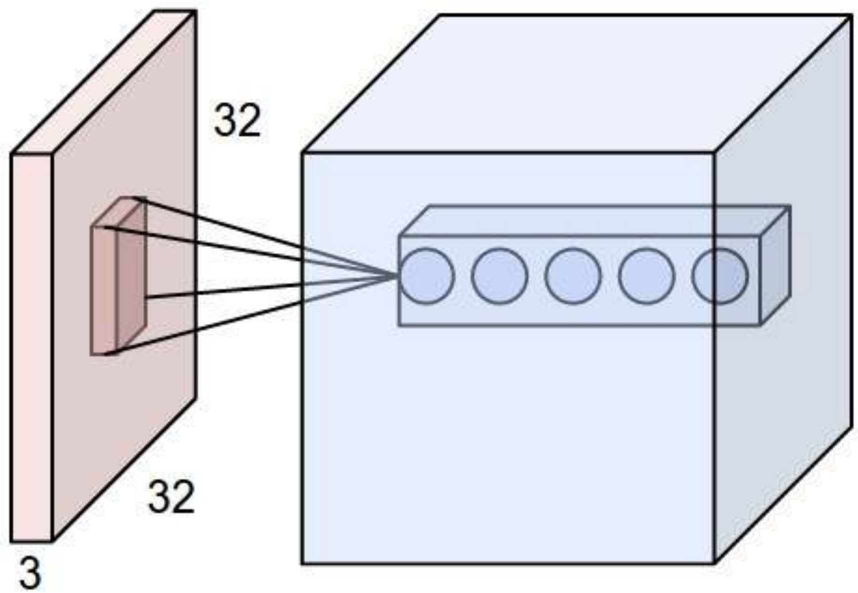
cs231n.github.io/assets/conv-demo/index.html

卷积计算层/ CONV layer



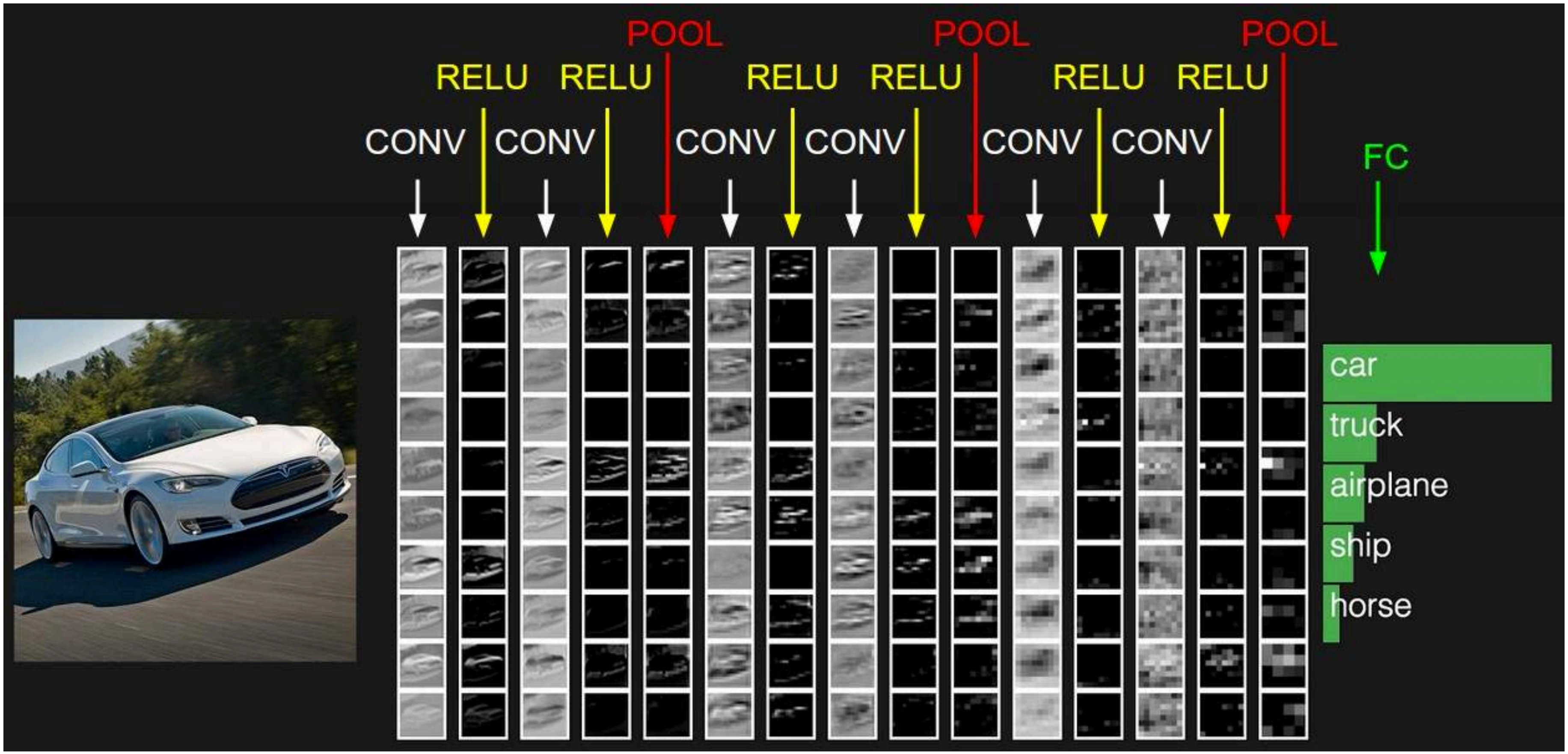
卷积计算层/ CONV layer

- **参数共享机制**
  假设每个神经元连接数据窗的权重是固定的
- 固定每个神经元连接权重，可以看做模板
  每个神经元只关注一个特性
- 需要估算的权重个数减少：AlexNet 1亿 => 3.5w
- 一组固定的权重和不同窗口内数据做内积：卷积



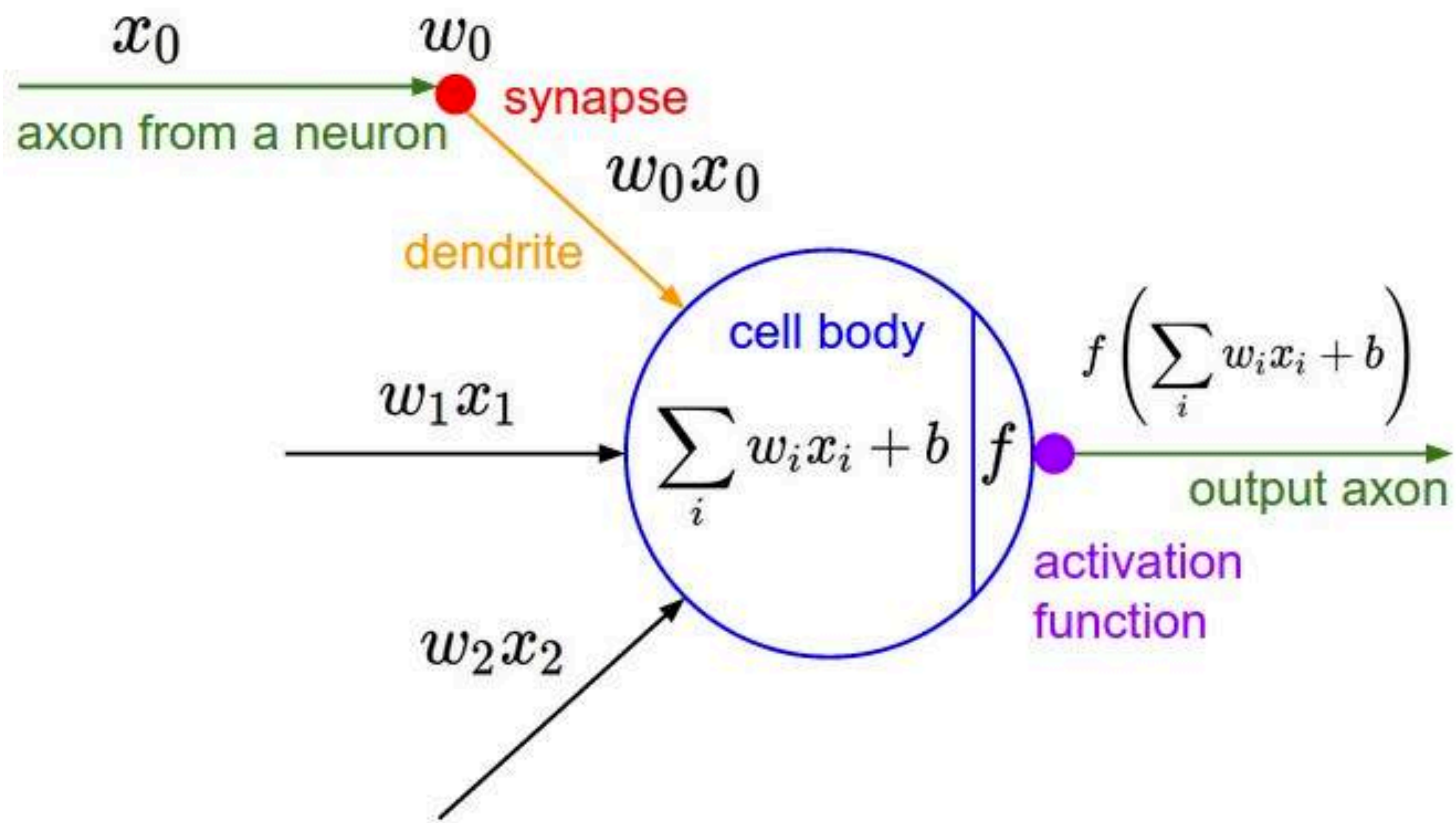
卷积神经网络层级结构

- 保持了层级网络结构
- 不同层次有不同形式(运算)与功能



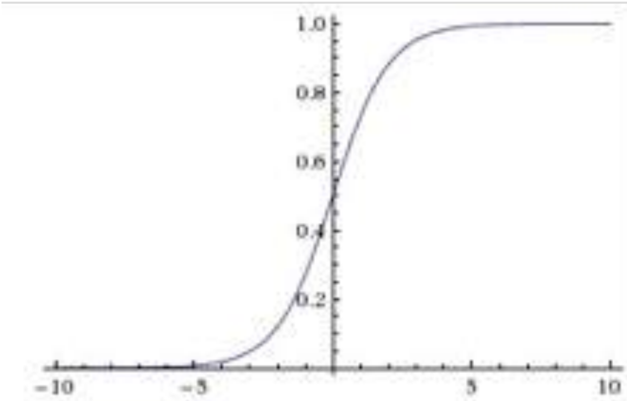
激励层(ReLU)

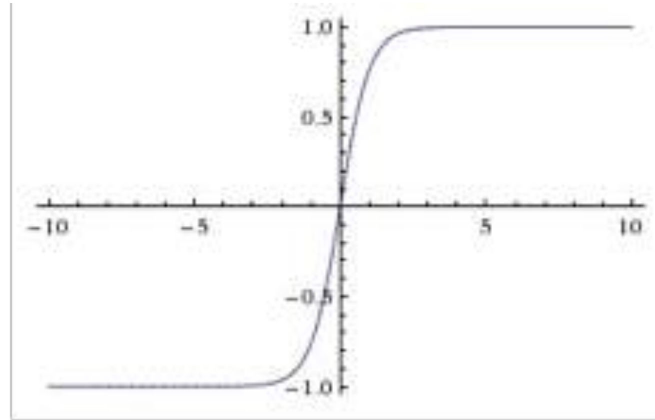
把卷积层输出结果做非线性映射

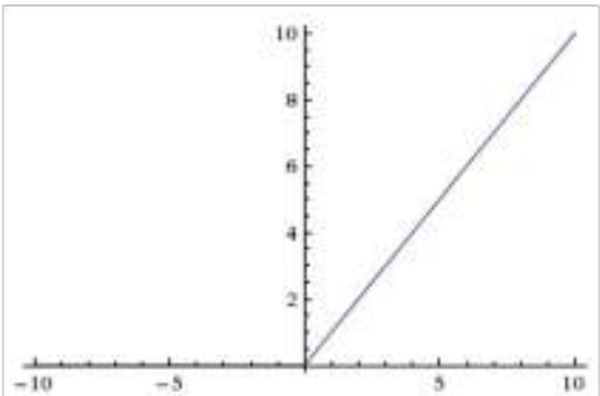


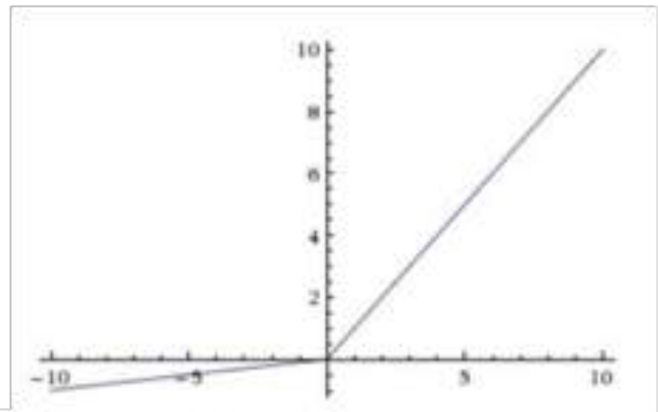
□ 激励层(ReLU)

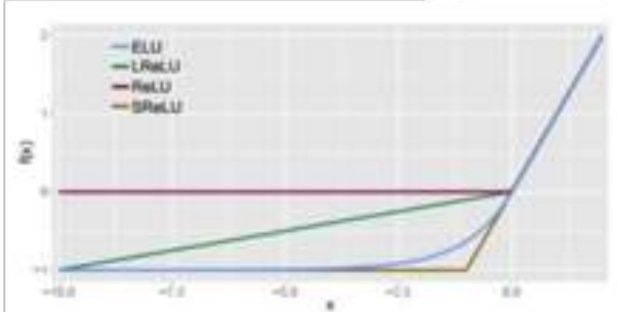
- 把卷积层输出结果做非线性映射
  - Sigmoid
  - Tanh(双曲正切)
  - ReLU
  - Leaky ReLU
  - ELU
  - Maxout











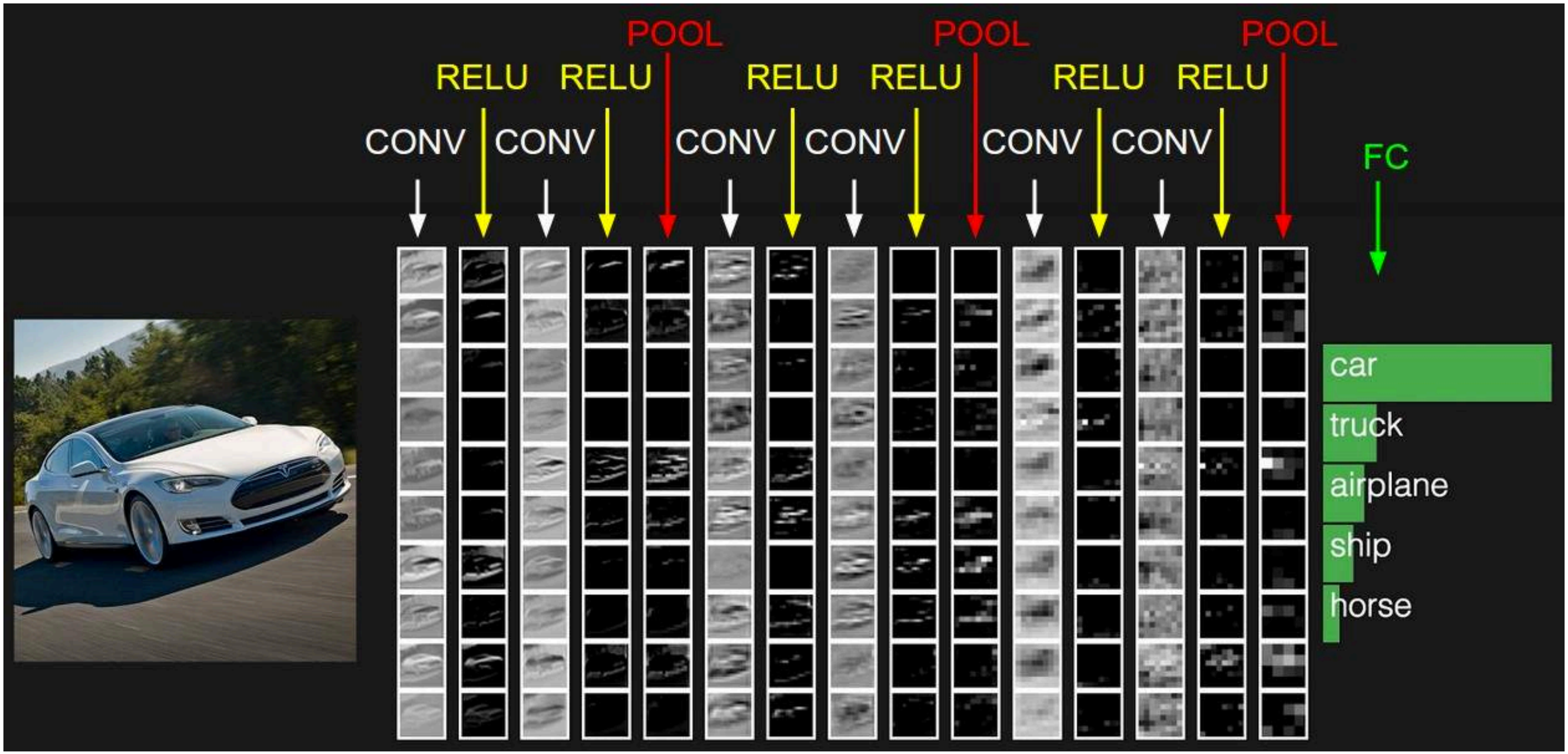
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \le 0 \end{cases}$$

卷积神经网络层级结构

- 激励层(实际经验)
  ① CNN尽量不要用sigmoid！不要用sigmoid！不要用sigmoid！
  ② 首先试RELU，因为快，但要小心点
  ③ 如果2失效，请用Leaky ReLU或者Maxout
  ④ 某些情况下tanh倒是有不错的结果，但是很少

卷积神经网络层级结构

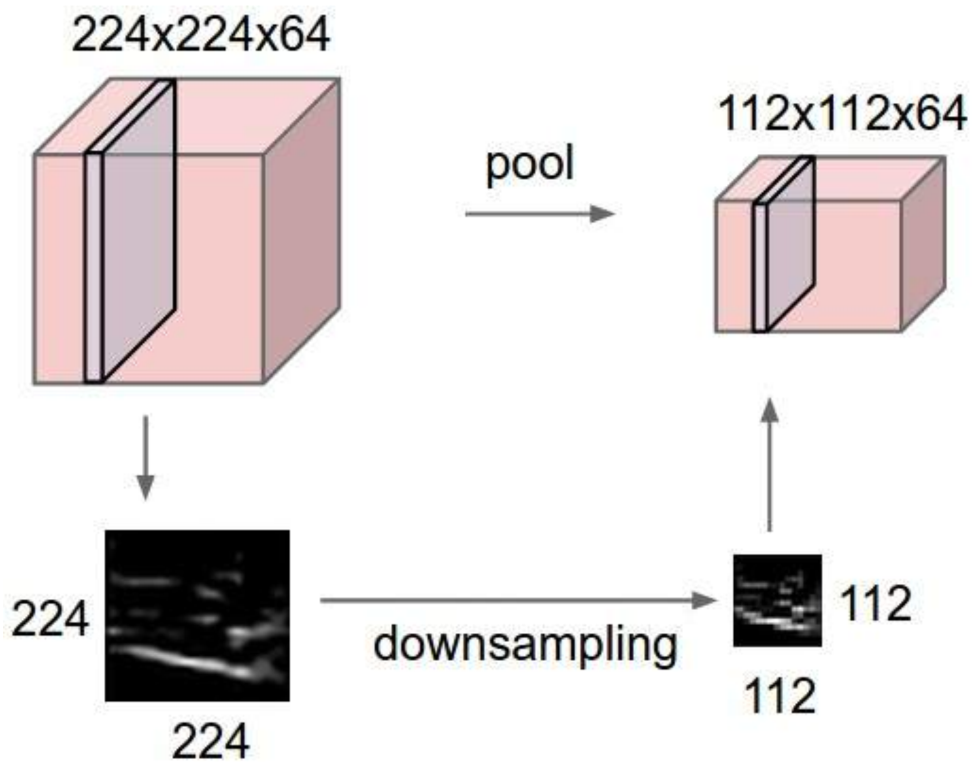
- 保持了层级网络结构
- 不同层次有不同形式(运算)与功能



卷积神经网络层级结构

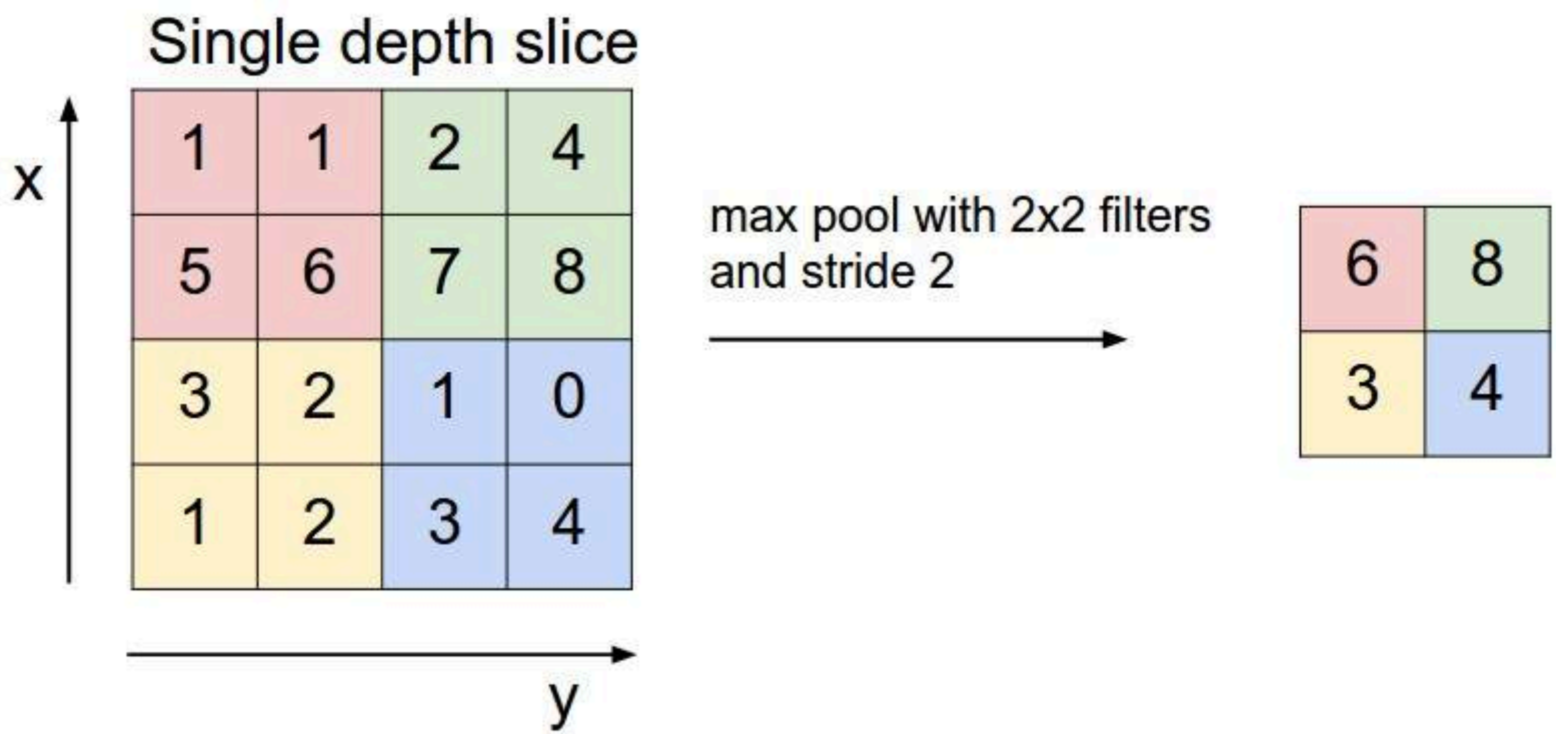
池化层 / Pooling layer

- 夹在连续的卷积层中间
- 压缩数据和参数的量，减小过拟合



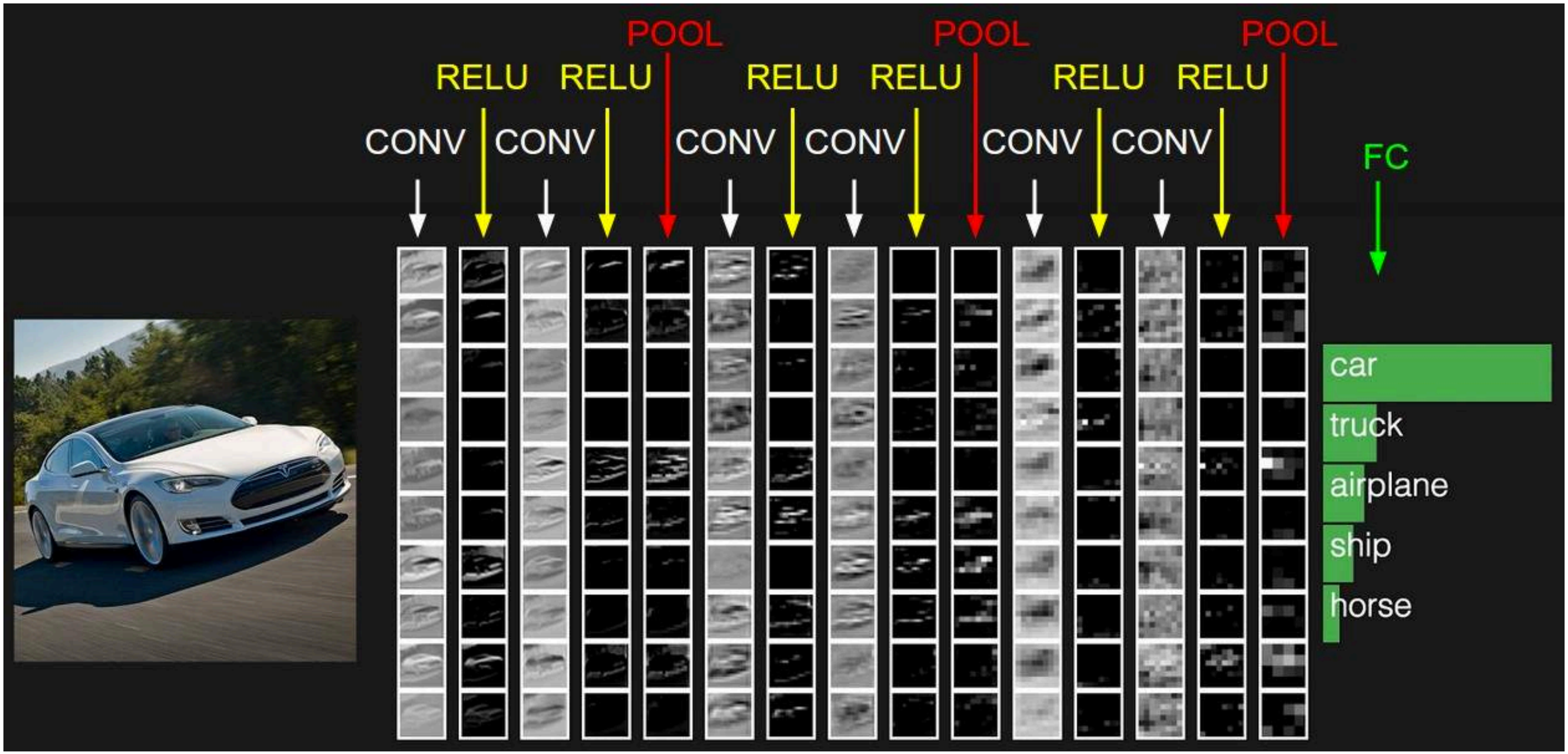
卷积神经网络层级结构

- 池化层 / Pooling layer
  - Max pooling
  - average pooling



卷积神经网络层级结构

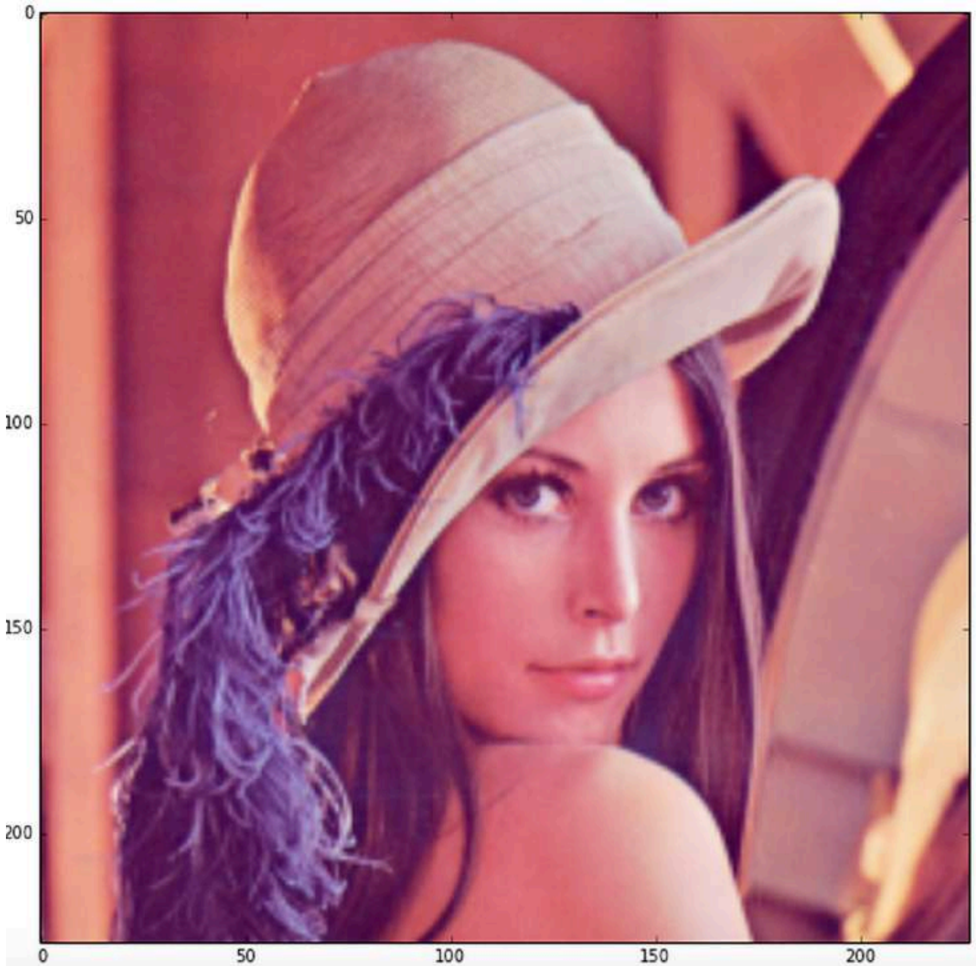
- 保持了层级网络结构
- 不同层次有不同形式(运算)与功能



卷积神经网络层级结构

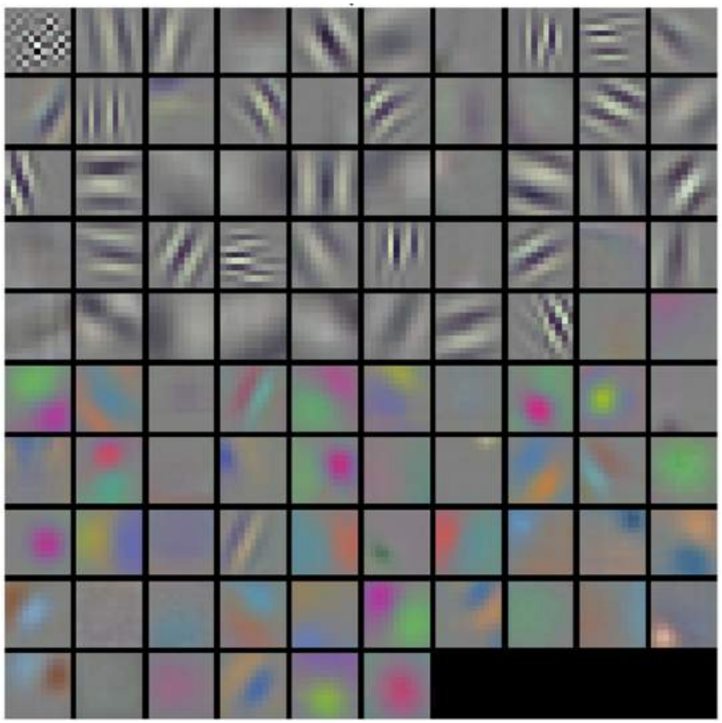
- 全连接层 / FC layer
  - 两层之间所有神经元都有权重连接
  - 通常全连接层在卷积神经网络尾部
- 一般CNN结构依次为
  - INPUT
  - [[CONV -> RELU]*N -> POOL?]*M
  - [FC -> RELU]*K
  - FC

卷积神经网络卷积层可视化理解

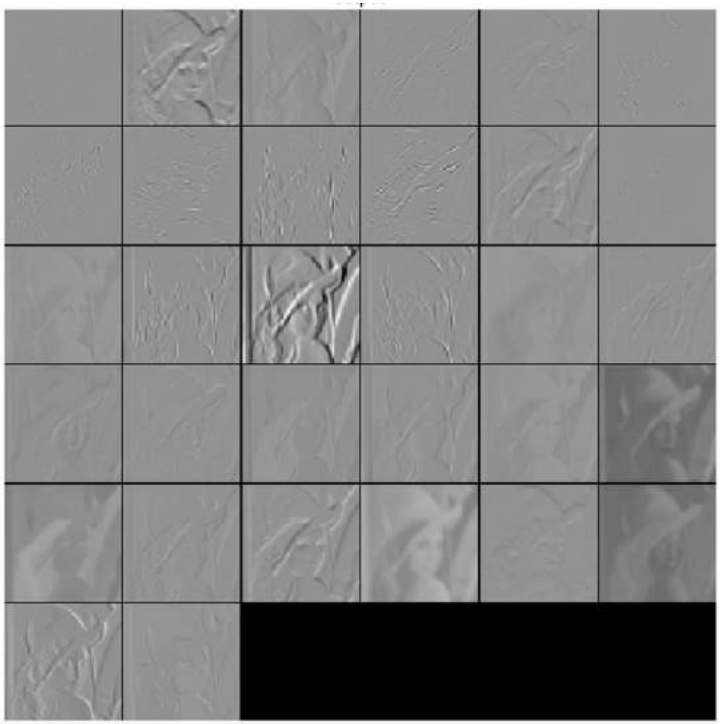


卷积神经网络卷积层可视化理解

- CONV Layer 1



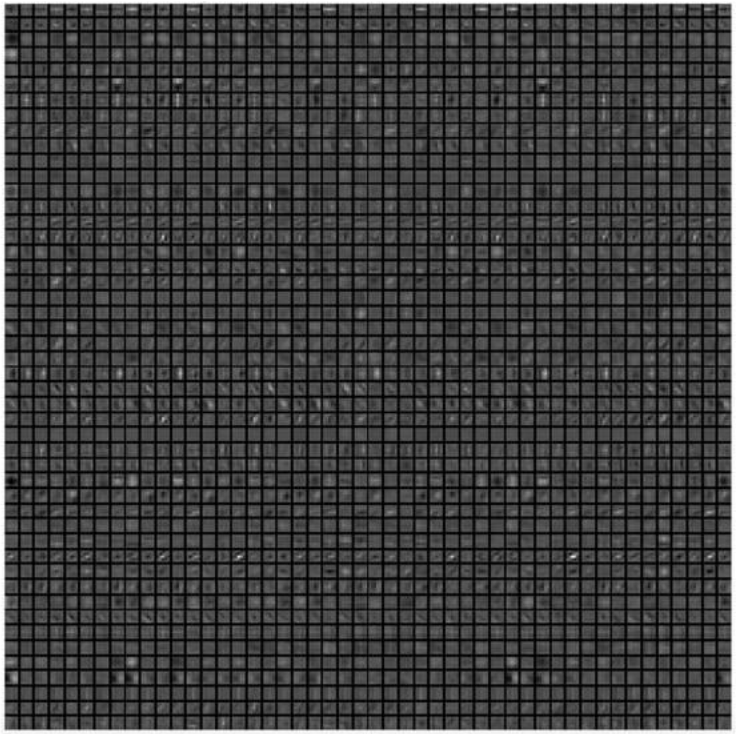
filters



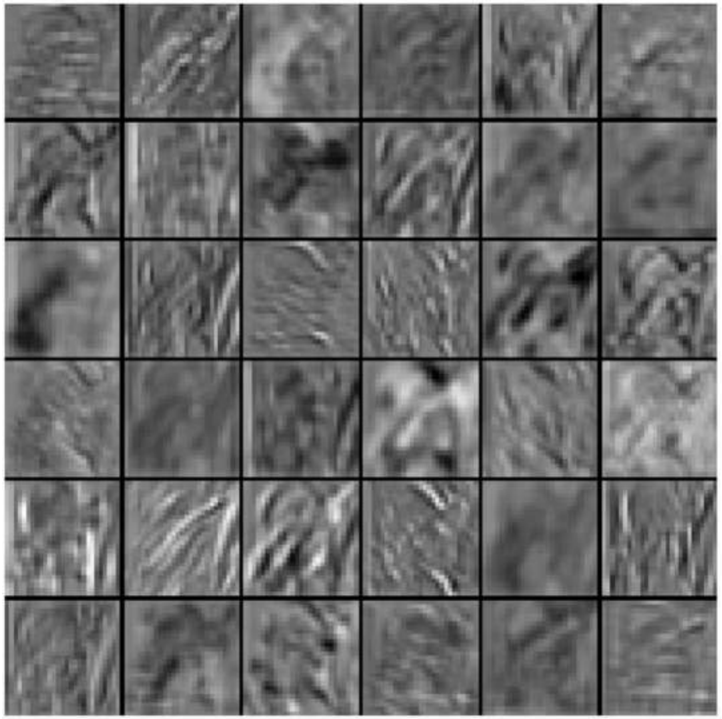
data

卷积神经网络卷积层可视化理解

- CONV Layer 2

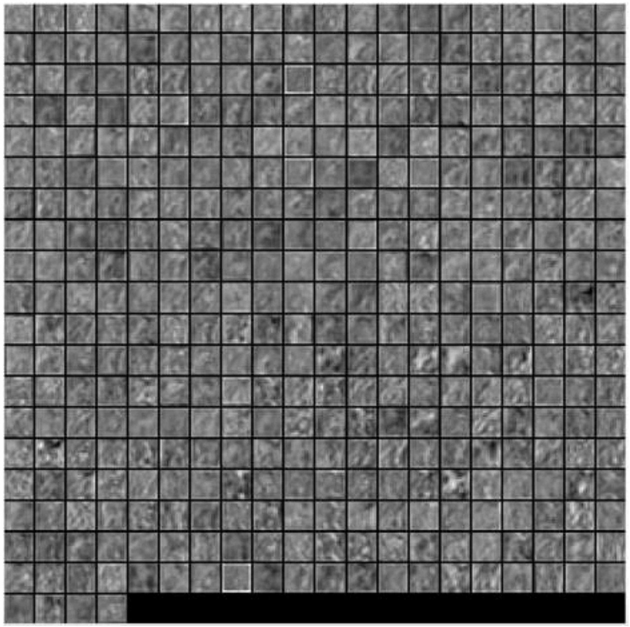


filters

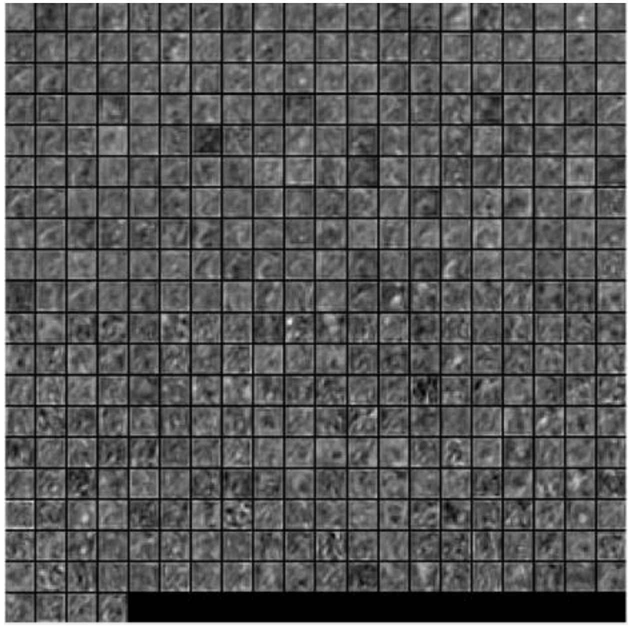


data

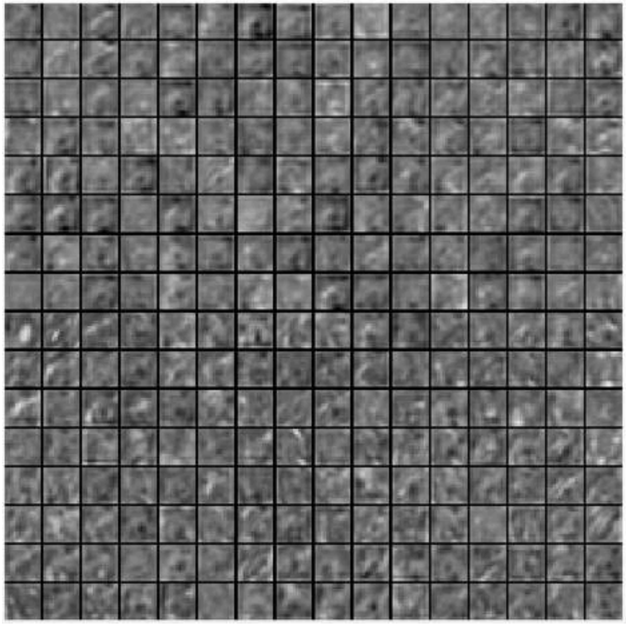
卷积神经网络卷积层可视化理解



Conv3 layer data



Conv4 layer data



Conv5 layer data

卷积神经网络训练算法

- 同一般机器学习算法，先定义Loss function，衡量和实际结果之间差距。
- 找到最小化损失函数的W和b，CNN中用的算法是SGD。
- SGD需要计算W和b的偏导
- BP算法就是计算偏导用的。
- BP算法的核心是求导链式法则。

$$\frac{dy}{dt} = \frac{dy}{dx}\frac{dx}{dt}$$

$$\frac{\partial y}{\partial x_i} = \sum_{\ell=1}^{m} \frac{\partial y}{\partial u_\ell}\frac{\partial u_\ell}{\partial x_i}$$

卷积神经网络训练算法

- BP算法利用链式求导法则，逐级相乘直到求解出dW和db。
- 利用SGD/随机梯度下降，迭代和更新W和b

卷积神经网络优缺点

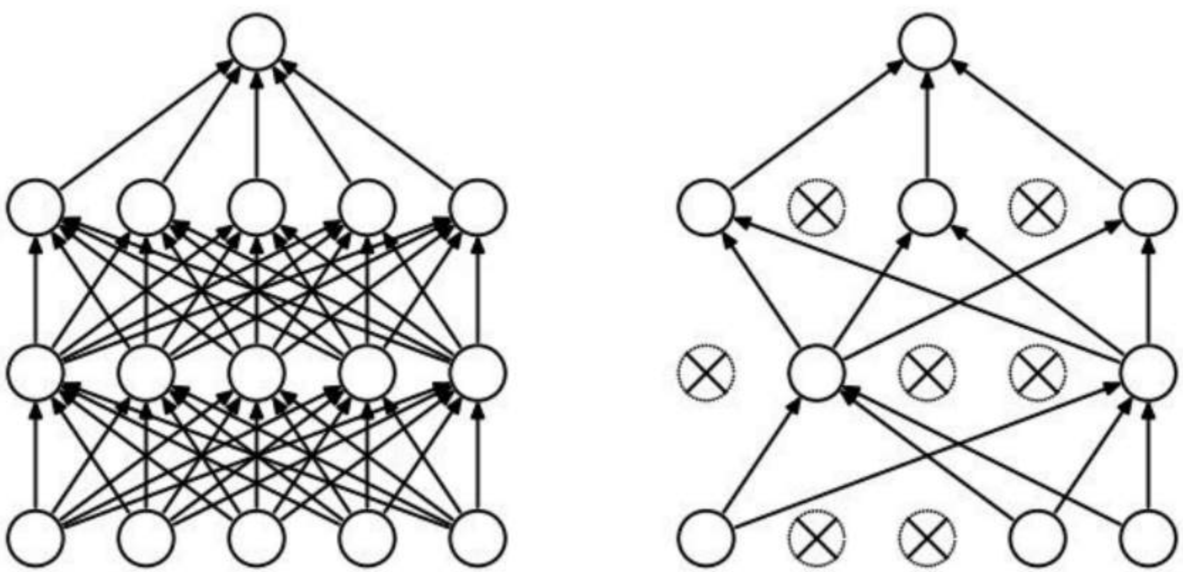
- 优点
  - 共享卷积核，对高维数据处理无压力
  - 无需手动选取特征，训练好权重，即得特征
  - 深层次的网络抽取图像信息丰富，表达效果好
- 缺点
  - 需要调参，需要大样本量，训练最好要GPU
  - 物理含义不明确

正则化与Dropout

- 神经网络学习能力强可能会过拟合。
- Dropout(随机失活)正则化：别一次开启所有学习单元

Regularization: **Dropout**

"randomly set some neurons to zero in the forward pass"

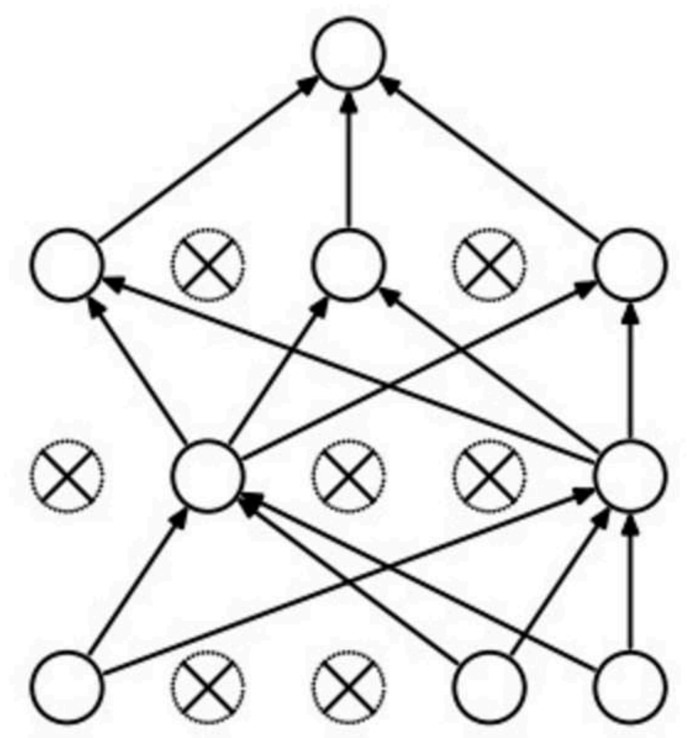


(a) Standard Neural Net (b) After applying dropout.

[Srivastava et al., 2014]

Dropout

```python
p = 0.5 # 设定dropout的概率，也就是保持一个神经元激活状态的概率

def train_step(X):
  """ X contains the data """

  # 3层神经网络前向计算
  H1 = np.maximum(0, np.dot(W1, X) + b1)
  U1 = np.random.rand(*H1.shape) < p # 第一次Dropout
  H1 *= U1 # drop!
  H2 = np.maximum(0, np.dot(W2, H1) + b2)
  U2 = np.random.rand(*H2.shape) < p # 第二次Dropout
  H2 *= U2 # drop!
  out = np.dot(W3, H2) + b3

  # 反向传播：计算梯度... (这里省略)
  # 参数更新... (这里省略)

def predict(X):
  # 加上Dropout之后的前向计算
  H1 = np.maximum(0, np.dot(W1, X) + b1) * p
  H2 = np.maximum(0, np.dot(W2, H1) + b2) * p
  out = np.dot(W3, H2) + b3
```



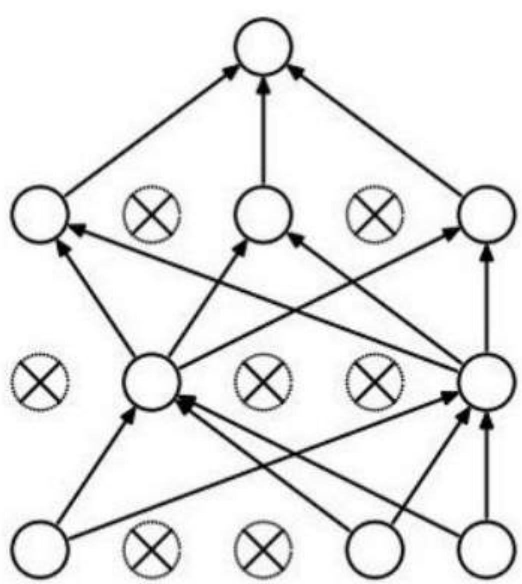
Dropout

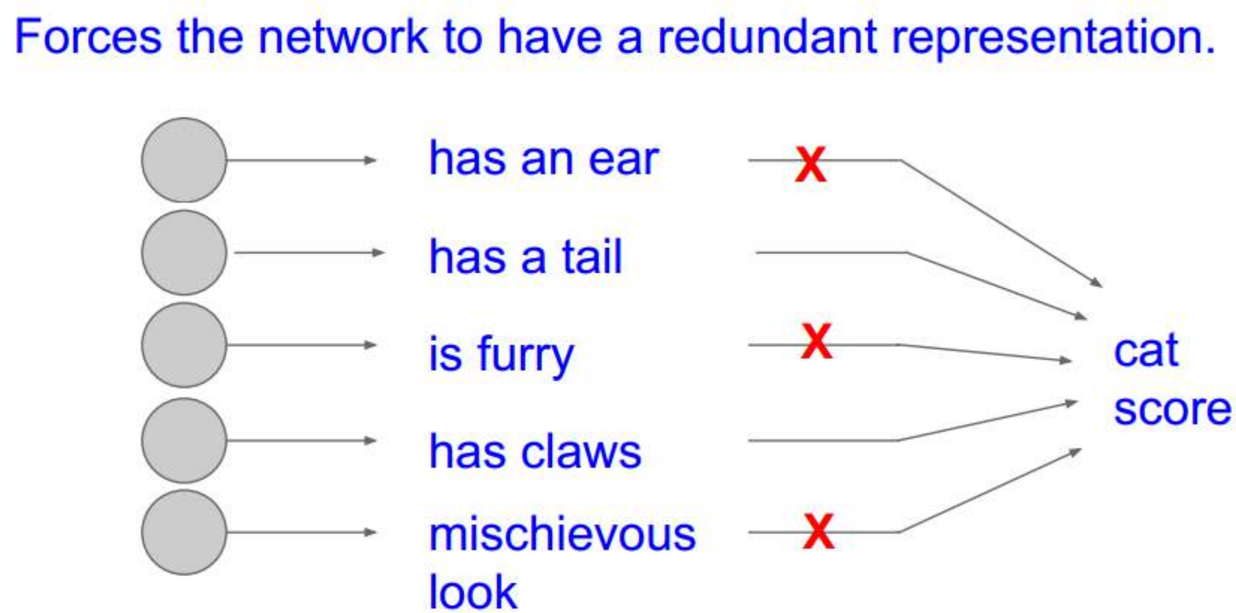
■ 实际实现：把预测阶段的时间转移到训练上

```
p = 0.5 # dropout的概率，也就是保持一个神经元激活状态的概率

def train_step(X):
  # f3层神经网络前向计算
  H1 = np.maximum(0, np.dot(W1, X) + b1)
  U1 = (np.random.rand(*H1.shape) < p) / p # 注意到这个dropout中我们除以p，做了一个inverted dropout
  H1 *= U1 # drop!
  H2 = np.maximum(0, np.dot(W2, H1) + b2)
  U2 = (np.random.rand(*H2.shape) < p) / p # 这个dropout中我们除以p，做了一个inverted dropout
  H2 *= U2 # drop!
  out = np.dot(W3, H2) + b3

  # 反向传播：计算梯度... (这里省略)
  # 参数更新... (这里省略)

def predict(X):
  # 直接前向计算，无需再乘以p
  H1 = np.maximum(0, np.dot(W1, X) + b1)
  H2 = np.maximum(0, np.dot(W2, H1) + b2)
  out = np.dot(W3, H2) + b3
```

Dropout理解

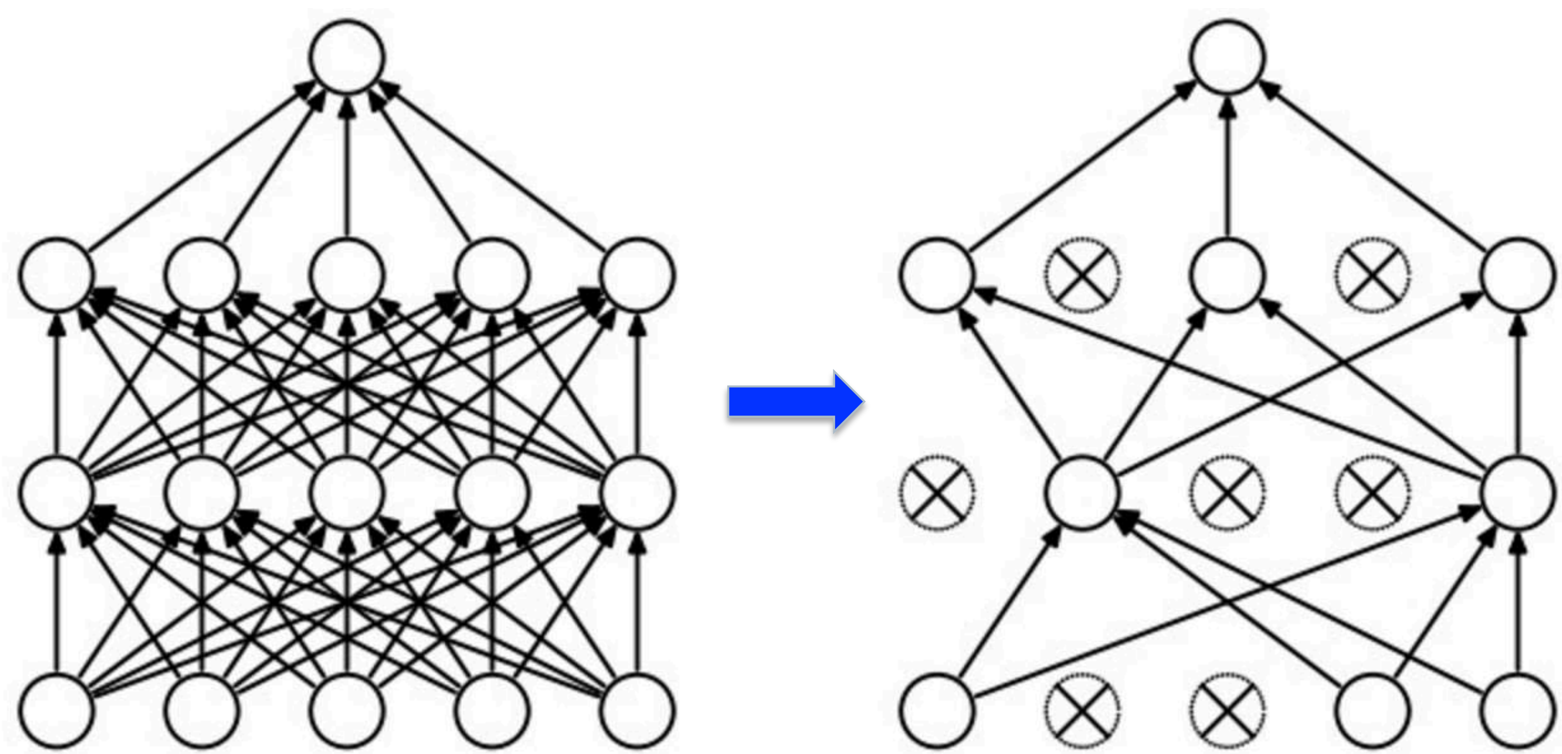
- 防止过拟合的第1种理解方式
  - 别让你的神经网络记住那么多东西(虽然CNN记忆力好)
  - 就是一只猫而已，要有一些泛化能力





Dropout理解

- 防止过拟合的第2种理解方式：
  - 每次都关掉一部分感知器，得到一个新模型，最后做融合。不至于听一家所言。



正则化与Dropout

- 对Dropout想要有更细致的了解，参见

- 2014, Hinton, etc

《Dropout: A Simple Way to Prevent Neural Networks from Overfitting》

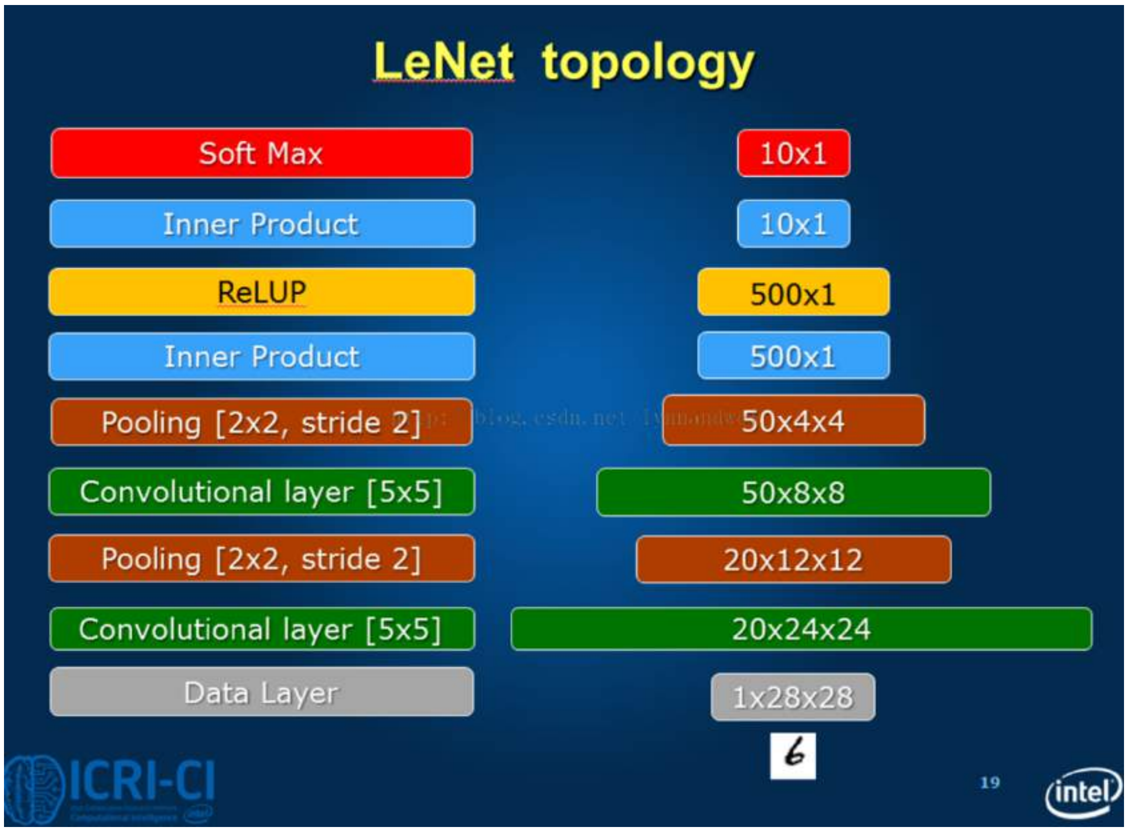
- 2013, Stefan Wager, etc 《Dropout Training as Adaptive Regularization》

卷积神经网络典型CNN

- LeNet，这是最早用于数字识别的CNN
- AlexNet，2012 ILSVRC比赛远超第2名的CNN，比LeNet更深，用多层小卷积层叠加替换单大卷积层。
- ZF Net，2013 ILSVRC比赛冠军
- GoogLeNet，2014 ILSVRC比赛冠军
- VGGNet，2014 ILSVRC比赛中的模型，图像识别略差于GoogLeNet，但是在很多图像转化学习问题(比如object detection)上效果很好
- ResNet，2015ILSVRC比赛冠军，结构修正(残差学习)以适应深层次CNN训练。

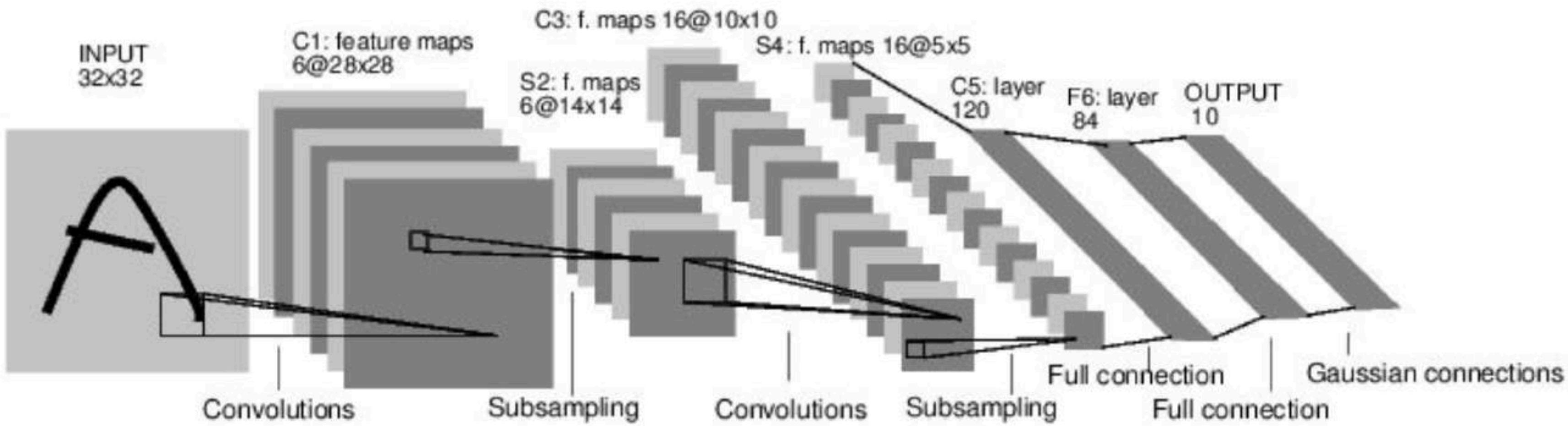
卷积神经网络典型CNN

- Lenet



卷积神经网络典型CNN

- Lenet



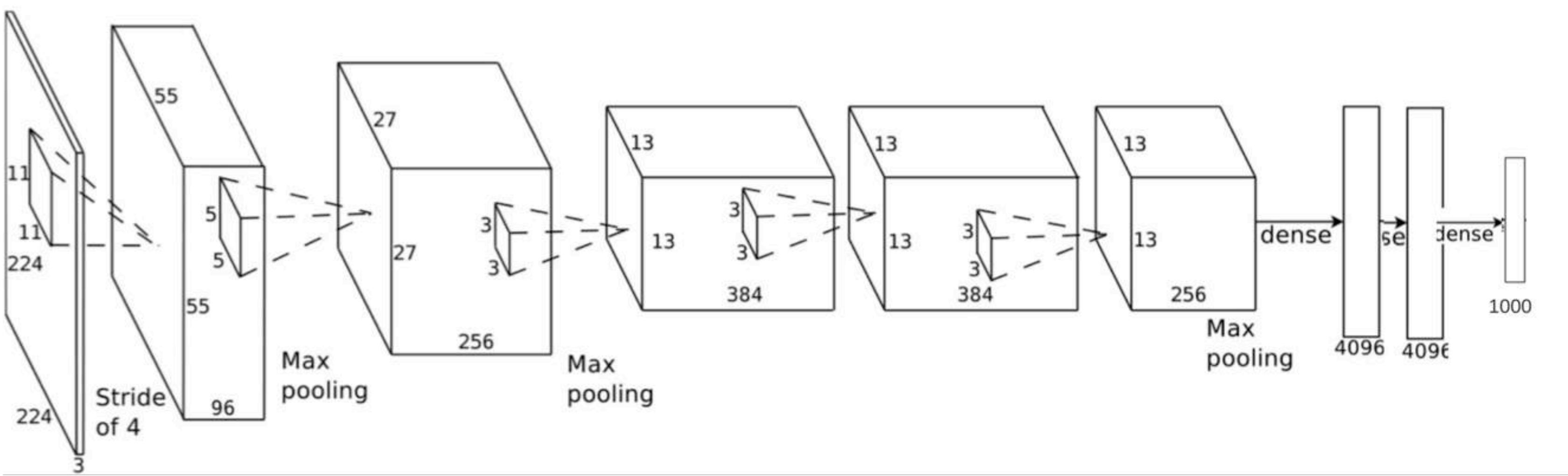
Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

卷积神经网络典型CNN

AlexNet

- 2012 Imagenet 比赛第一，Top5准确度超出第二10%

- The number of neurons in each layer is given by 253440, 186624, 64896, 64896, 43264, 4096, 4096, 1000



卷积神经网络典型CNN

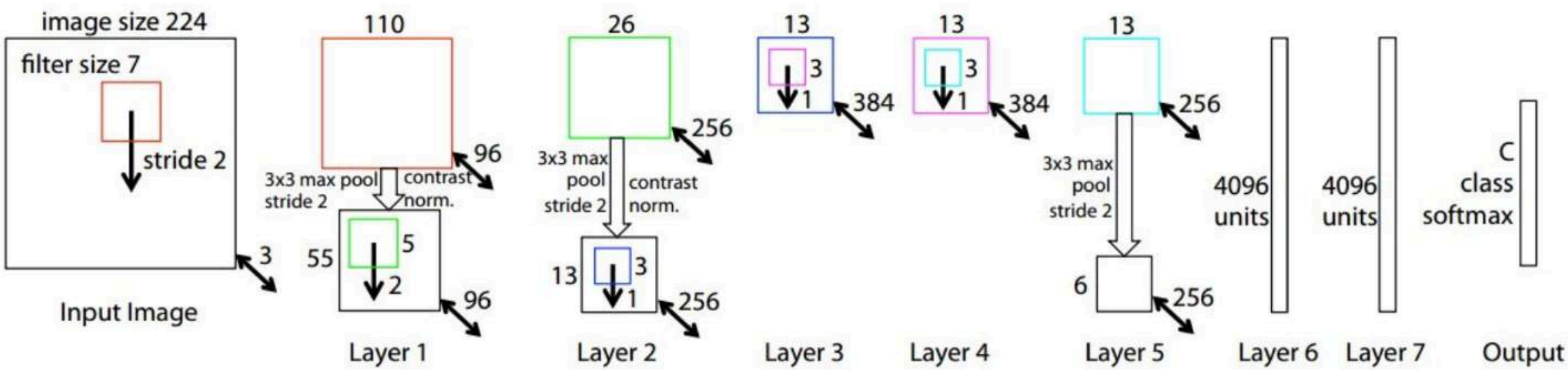
AlexNet

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%

卷积神经网络典型CNN

- ZFNet



AlexNet but:
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 15.4% -> 14.8%

卷积神经网络典型CNN

- VGG

```
INPUT: [224x224x3]
CONV3-64: [224x224x64]
CONV3-64: [224x224x64]
POOL2: [112x112x64]
CONV3-128: [112x112x128]
CONV3-128: [112x112x128]
POOL2: [56x56x128]
CONV3-256: [56x56x256]
CONV3-256: [56x56x256]
CONV3-256: [56x56x256]
POOL2: [28x28x256]
CONV3-512: [28x28x512]
CONV3-512: [28x28x512]
CONV3-512: [28x28x512]
POOL2: [14x14x512]
CONV3-512: [14x14x512]
CONV3-512: [14x14x512]
CONV3-512: [14x14x512]
POOL2: [7x7x512]
FC: [1x1x4096]
FC: [1x1x4096]
FC: [1x1x1000]
```

卷积神经网络典型CNN

VGG

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

best model

11.2% top 5 error in ILSVRC 2013
->
7.3% top 5 error

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

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

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

卷积神经网络典型CNN

VGG

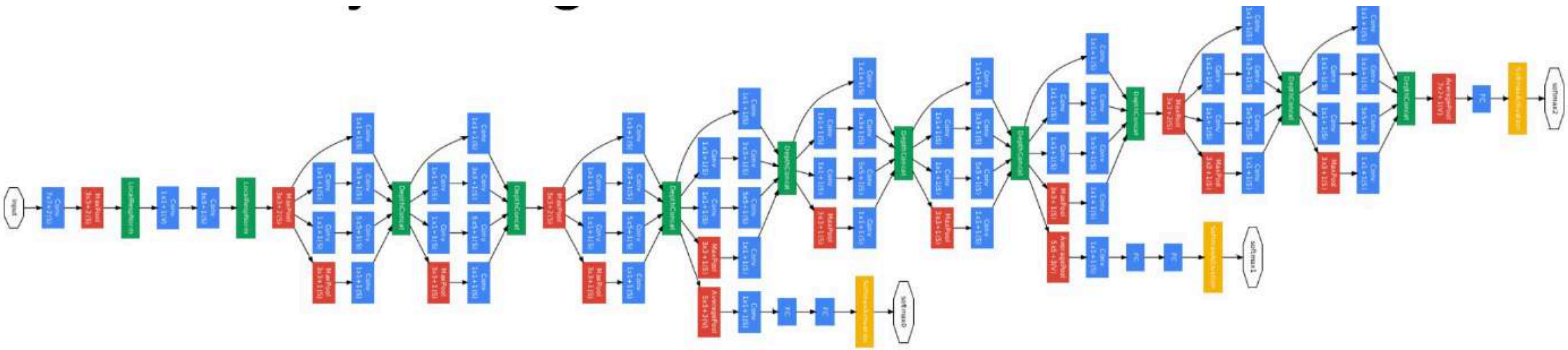
INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

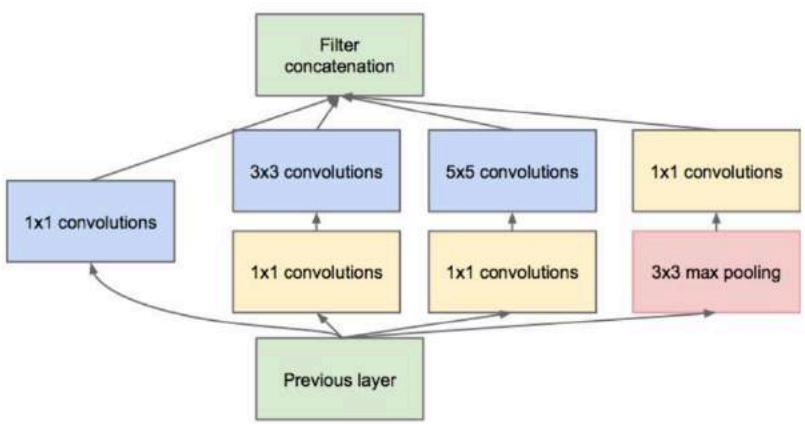
TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

ConvNet Configuration			
B	C	D	
13 weight layers	16 weight layers	16 weight layers	19
put (224 × 224 RGB image)			
conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	cc cc
maxpool			
conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	co co
maxpool			
conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
FC-4096			
FC-4096			
FC-1000			
soft-max			

卷积神经网络典型CNN

- GoogLeNet





Inception module

ILSVRC 2014 winner (6.7% top 5 error)

卷积神经网络典型CNN

- GoogLeNet

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

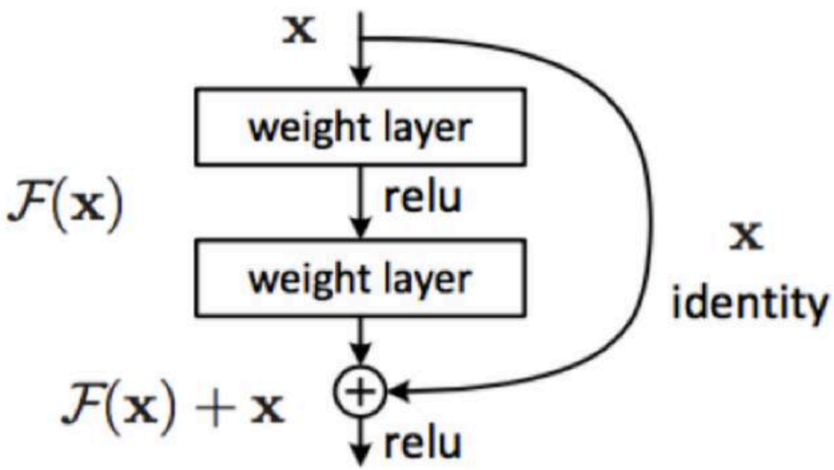
Fun features:

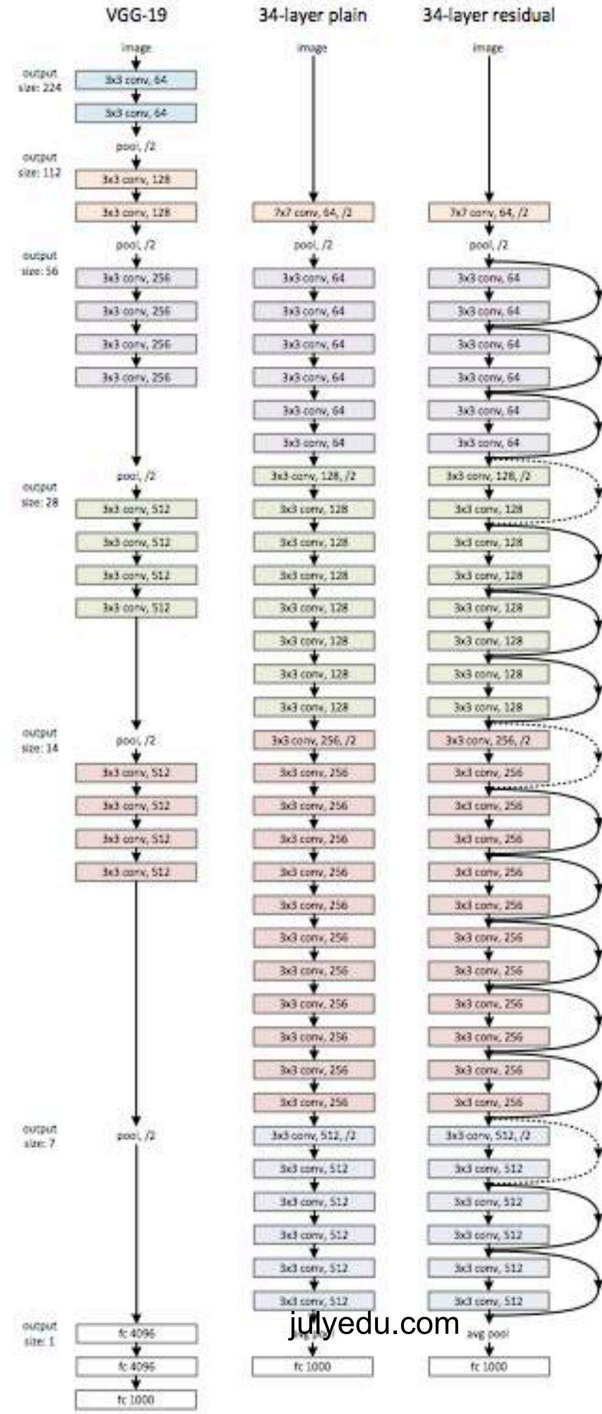
- Only 5 million params!
(Removes FC layers completely)

Compared to AlexNet:
- 12X less params
- 2x more compute
- 6.67% (vs. 16.4%)

卷积神经网络典型CNN

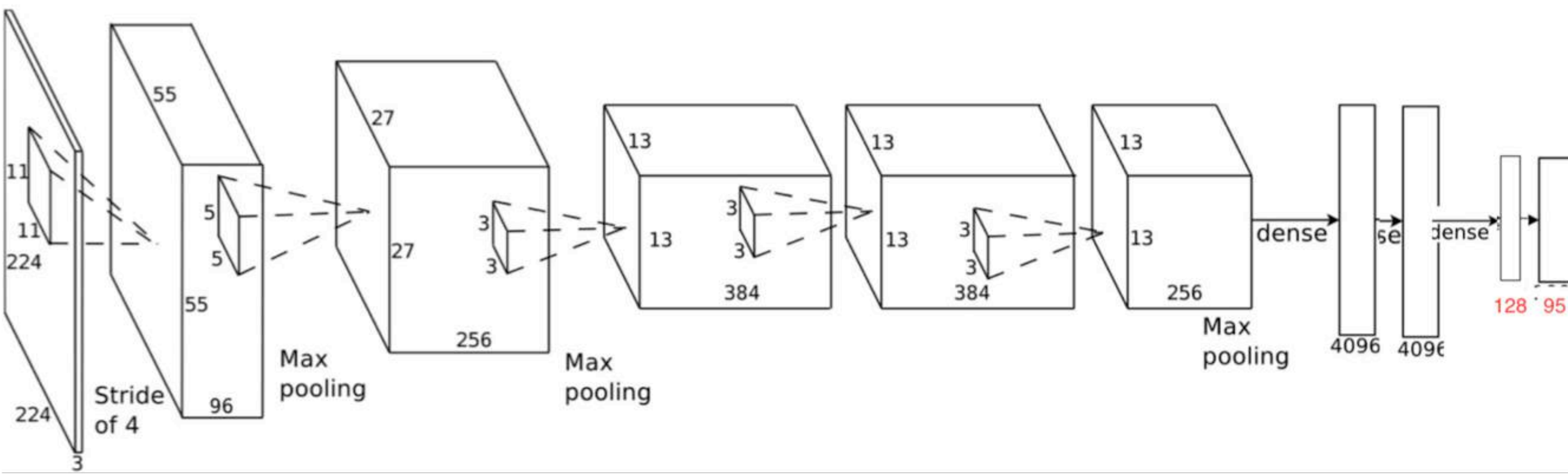
- ResNet，即Deep Residual Learning network
- 微软亚洲研究院提出， ILSVRC 2015冠军 比VGG还要深8倍。





卷积神经网络调优训练

- The number of neurons in each layer is given by 253440, 186624, 64896, 64896, 43264, 4096, 4096, 128, 95



CNN图像识别示例

详见课程ipython notebook

感谢大家！

恳请大家批评指正！