卷积神经网络
Convolution Neural Network

主讲：邓伟洪
http://www.pris.net.cn/introduction/teacher/dengweihong
模式识别与智能系统实验室

人工智能学院
北京邮电大学

一些历史

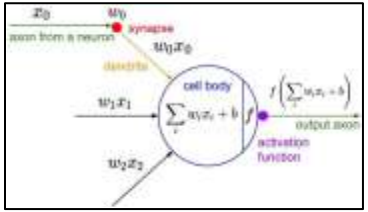
Mark I感知器是第一个实现的感知器算法。
这台机器与一台使用20×20硫化镉光电池的照相机相连，产生400像素的图像。

这台机器与一台使用20×20硫化镉光电池的照相机相连，产生400像素的图像。

字母表中可识别的字母 $$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

更新规则：

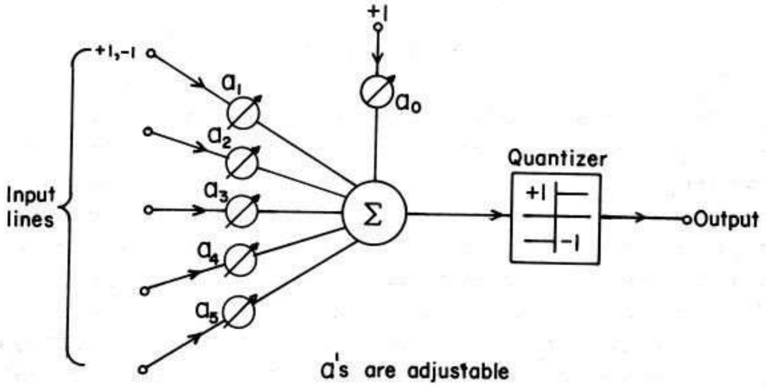
$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i},$$

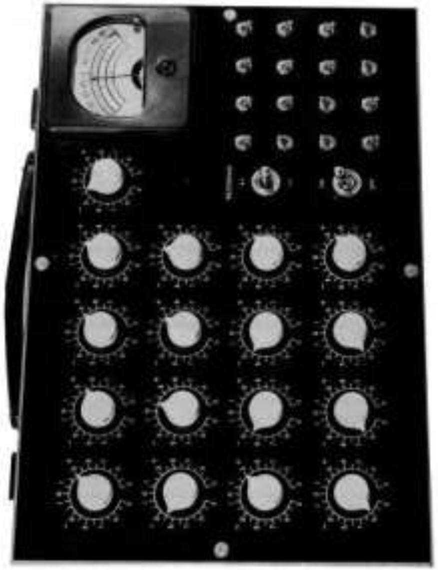


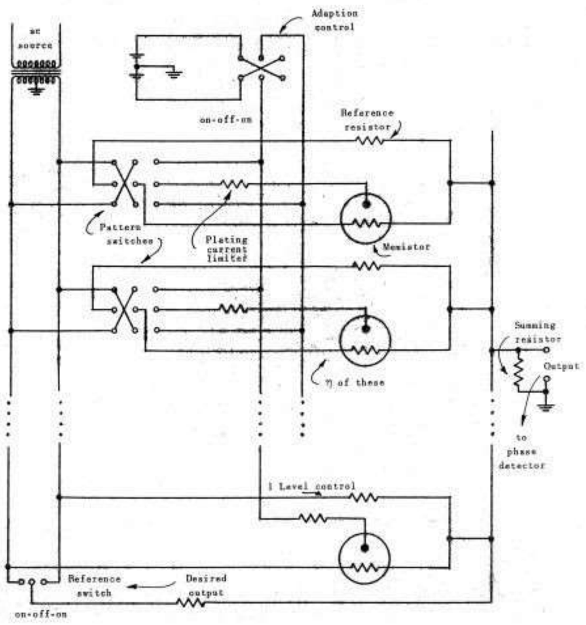
Frank Rosenblatt, ~1957: Perceptron

这些图片由Rocky Acosta根据CC-BY3.0授权。

一些历史







这些图片复制自Widrow 1960，斯坦福电子实验室技术报告，经斯坦福大学特别收藏部许可。

Widrow and Hoff, ~1960: Adaline/Madaline

一些历史：

Hubel & Wiesel,

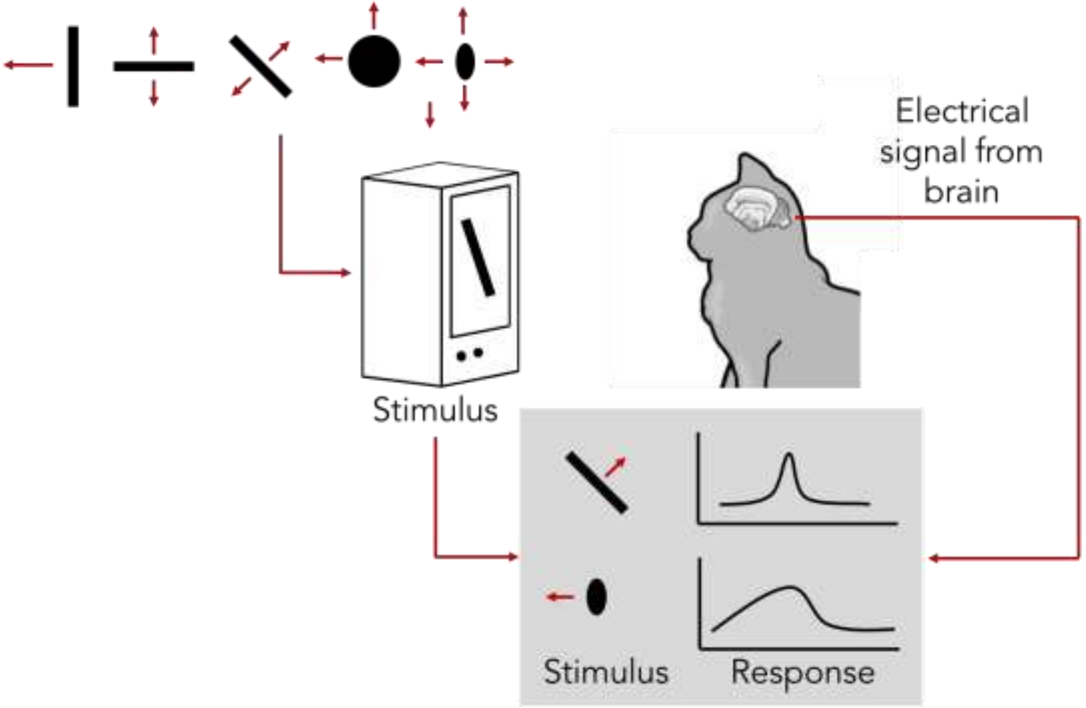
1959

猫纹状体皮层单个神经元的感受野

1962

猫视觉皮层的感受野、双眼交互作用和功能结构

1968...

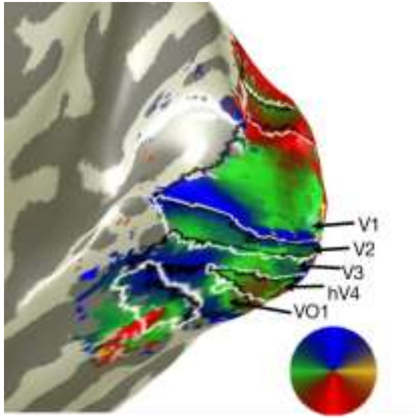


CNX OpenStax的猫的图片是在CC-by-4.0下授权的；进行了更改

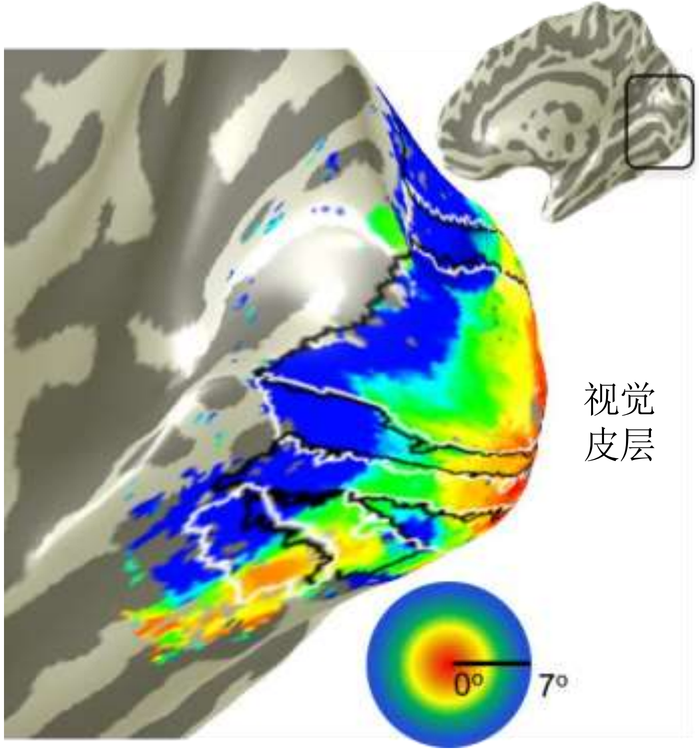
一些历史

皮层地形图：

皮层附近的细胞代表视野中的邻近区域

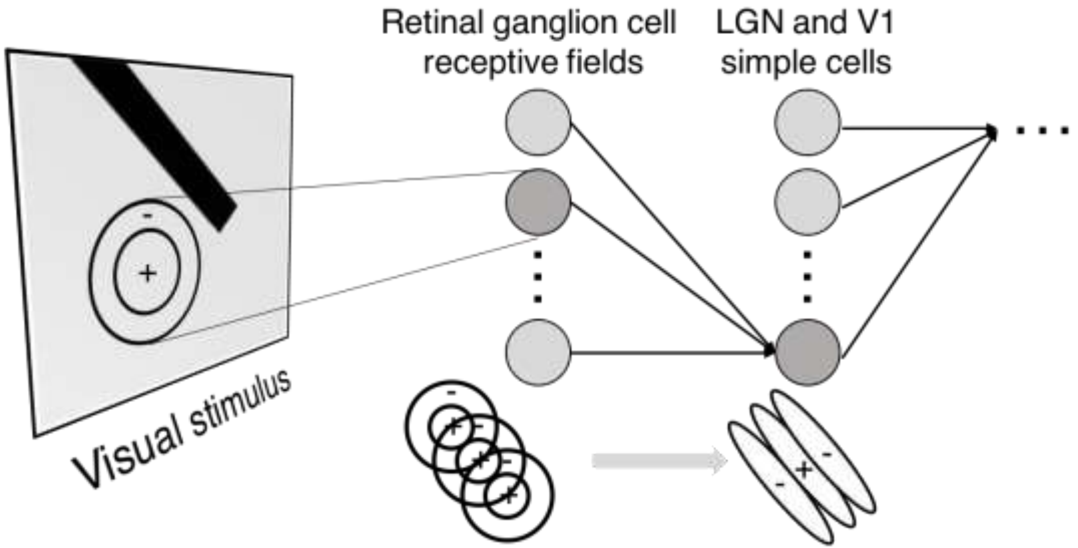


人类大脑



视网膜整形图片由杰西·戈麦斯在斯坦福视觉与感知神经科学实验室提供。

分层组织

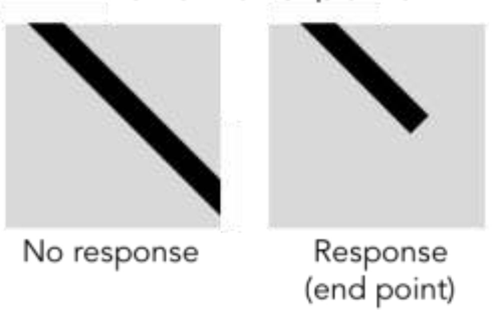


Lane McIntosh的早期视觉路径中的分层组织插图，版权所有CS231n 2017

Simple cells:
Response to light orientation

Complex cells:
Response to light orientation and movement

Hypercomplex cells:
response to movement with an end point



一些历史：

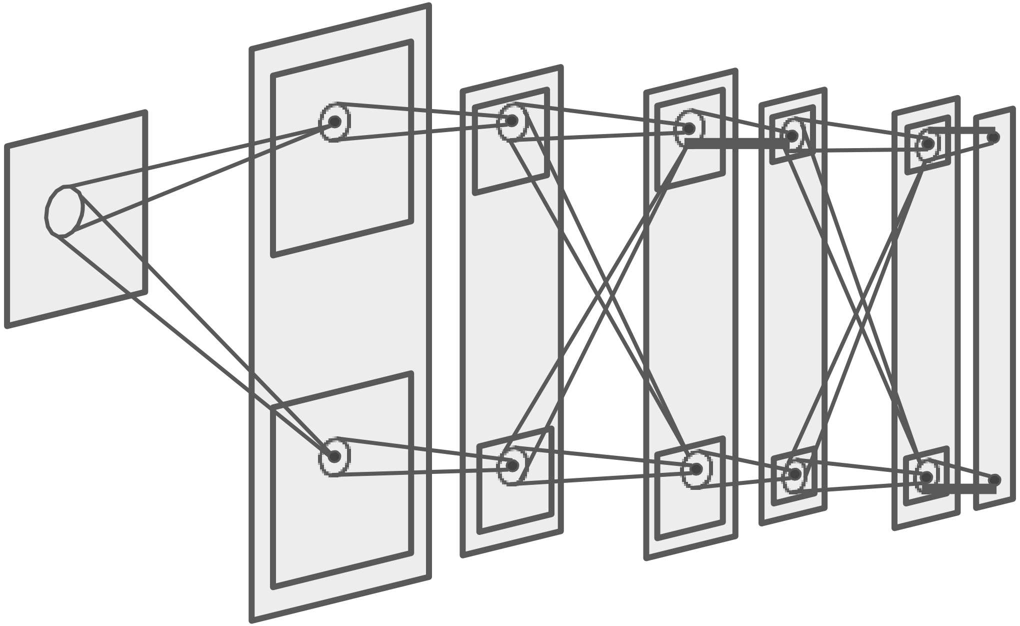
Neocognitron

[Fukushima 1980]

“三明治”架构(SCSCSC...)

简单的单元：可修改的参数

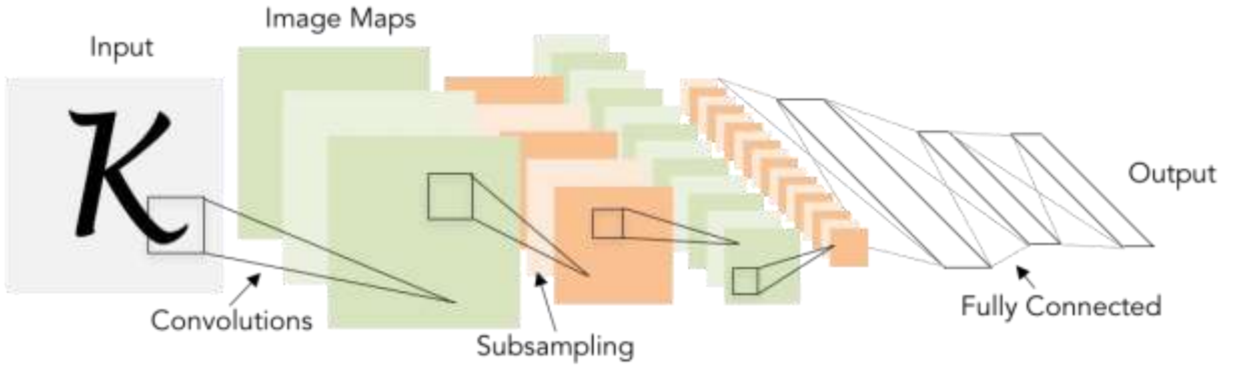
复杂的单元：执行池化操作



一些历史：

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]



LeNet-5

一些历史：

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]

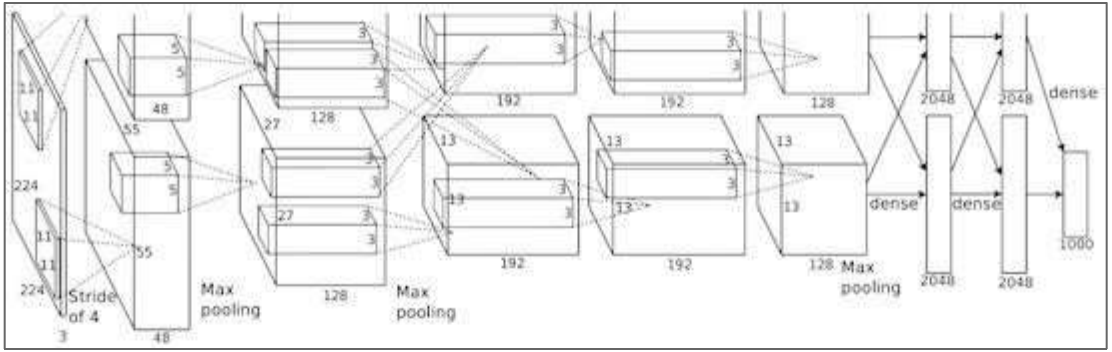


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

“AlexNet”

飞速发展直至今天：卷积网络无处不在

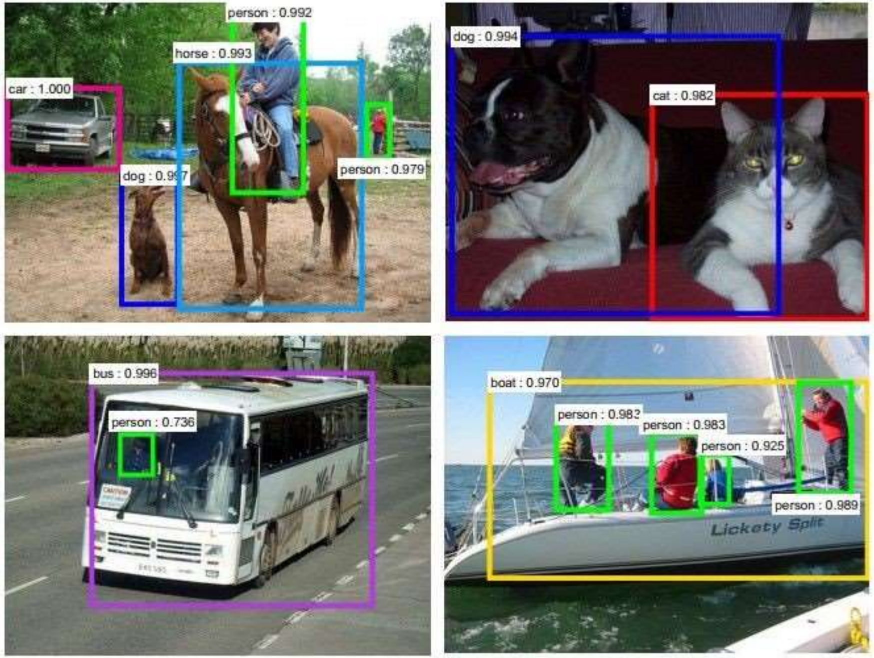
分类

恢复

数字版权Alex Krizhevsky，Ilya Sutskever和Geoffrey Hinton，2012年。经许可复制。

飞速发展直至今天：卷积网络无处不在

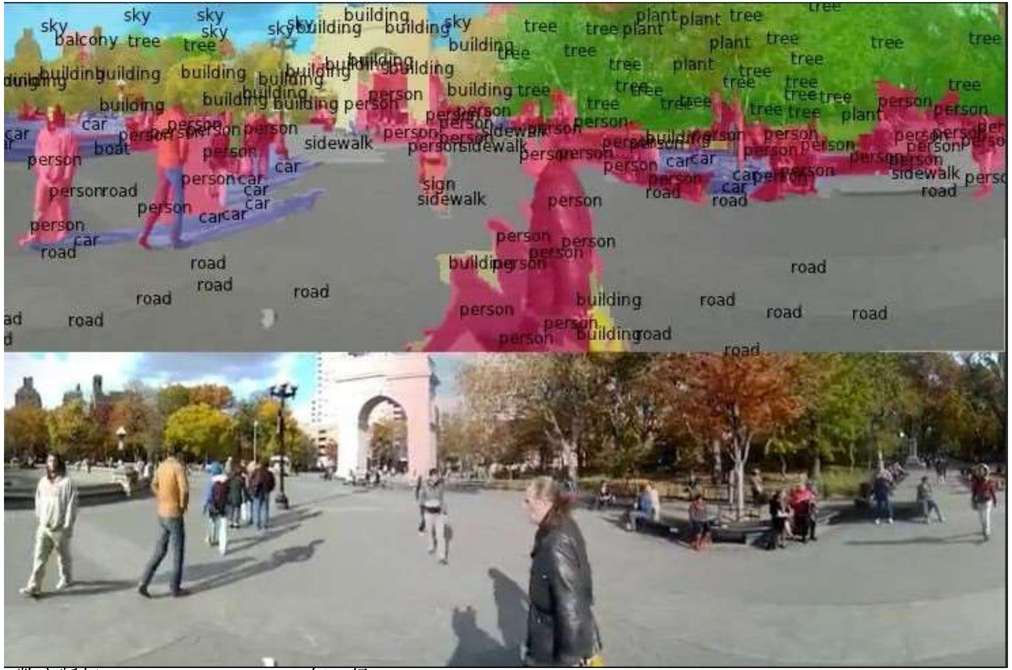
检测



数字版权Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun， 2015年。经许可复制.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

分割



数字版权Clement Farabet，2012年。经许可复制。

[Farabet et al., 2012]

飞速发展直至今天：卷积网络无处不在

Lane McIntosh摄。版权所有CS231n 2017。

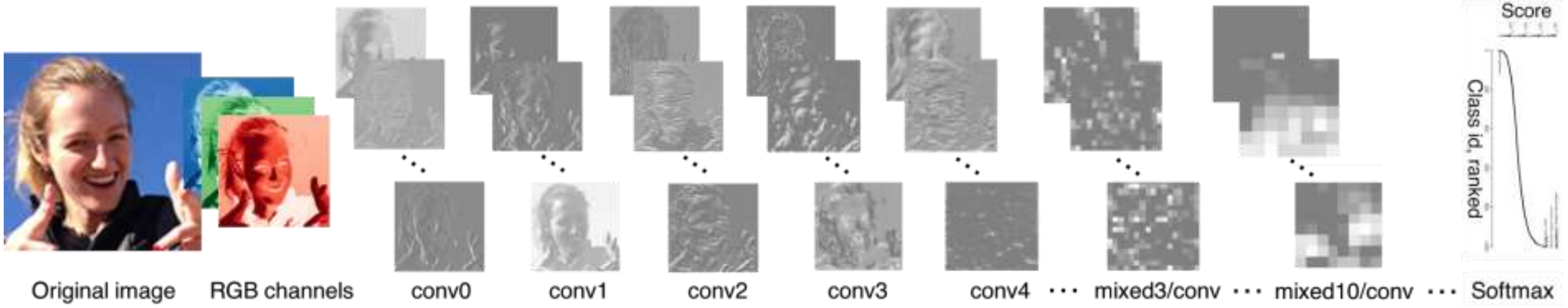
自动驾驶汽车

这个图片来自GBPublic_PR
根据CC-BY 2.0许可。

NVIDIA Tesla line
(这些是rye01.stanford.edu上的GPU)

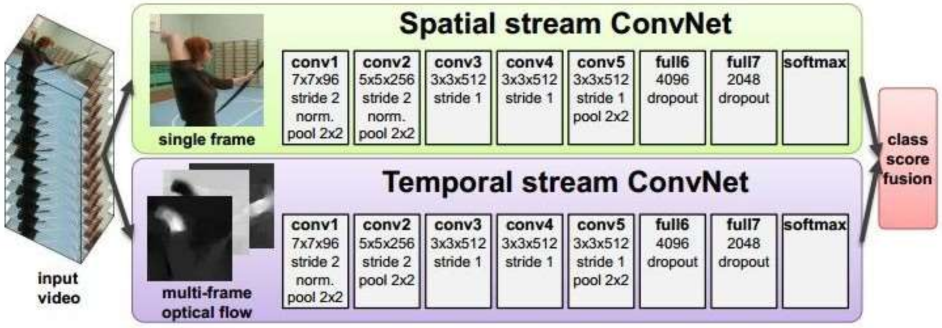
请注意，对于嵌入式系统，典型的设置将涉及NVIDIA Tegras，并集成GPU和基于ARM的CPU内核。

飞速发展直至今天：卷积网络无处不在



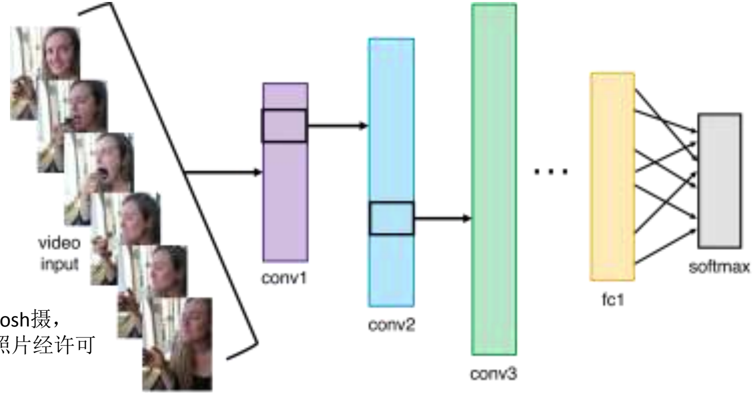
[Taigman et al. 2014]

Inception-v3体系结构的激活[Szegedy等。 [2015年]图片为Emma McIntosh的图片，经允许使用。图和架构并非来自Taigman等。 2014。



[Simonyan et al. 2014]

数字版权Simonyan等，2014。经许可复制。

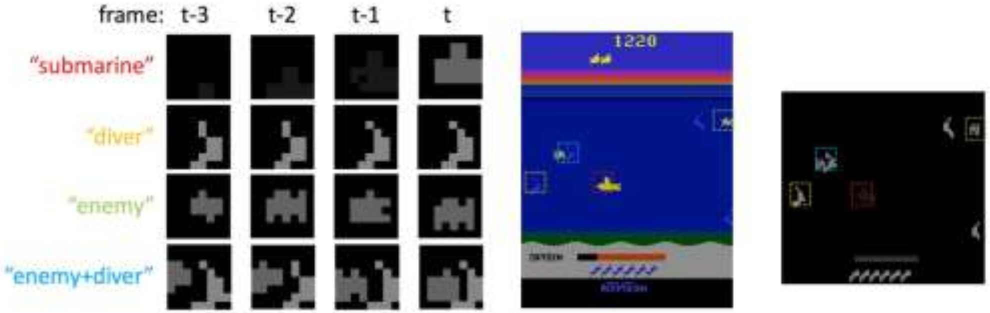


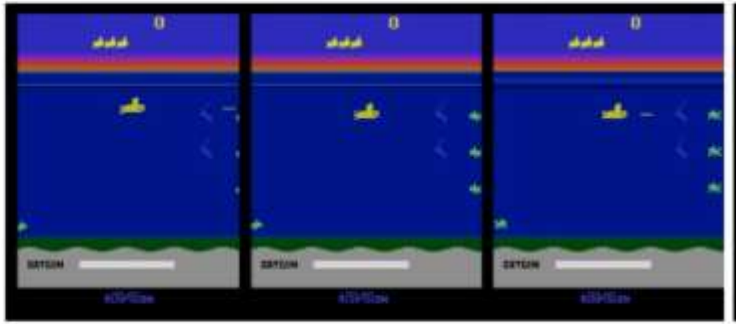
插图由Lane McIntosh摄，Katie Cumnock的照片经许可使用。

飞速发展直至今天：卷积网络无处不在

图像是姿势估计的示例，实际上并非来自Toshev＆Szegedy2014。版权所有Lane McIntosh。

[Toshev, Szegedy 2014]

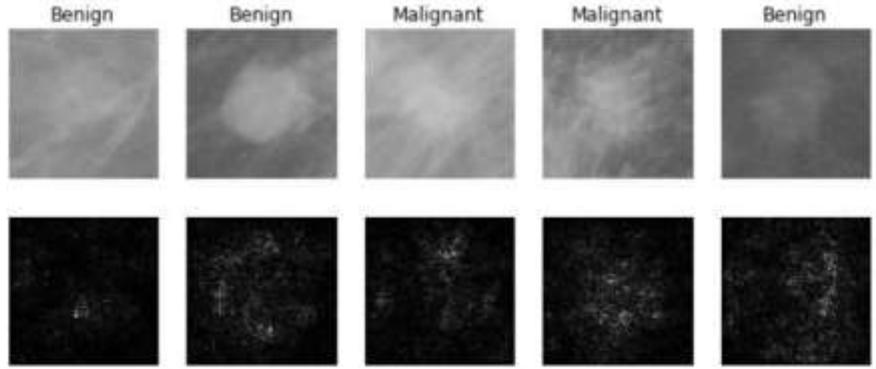




[Guo et al. 2014]

数字版权Xiaoxiao Guo, Satinder Singh， Honglak Lee, Richard Lewis, and Xiaoshi Wang， 2014年。经许可复制

飞速发展直至今天：卷积网络无处不在



[Levy et al. 2016]

图由Levy等授予版权。 2016。经许可复制。

[Dieleman et al. 2014]

从左到右：NASA的公共领域，允许使用的ESA / Hubble，NASA的公共领域和公共领域。

[Sermanet et al. 2011]
[Ciresan et al.]

Lane McIntosh摄。版权所有CS231n 2017.

Christin Khan拍摄的这个图片属于公共领域，最初来源于美国NOAA。

鲸鱼识别, Kaggle挑战赛

Lane McIntosh的照片和图；并非来自Mnih和Hinton，2010年论文的实际示例。

Mnih and Hinton, 2010

图片字幕

没有问题

小错误

有点联系

坐在草地上的白色泰迪熊

一个穿着棒球服的人扔一个球

一个女人手里拿着一只猫

一个人在冲浪板上冲浪

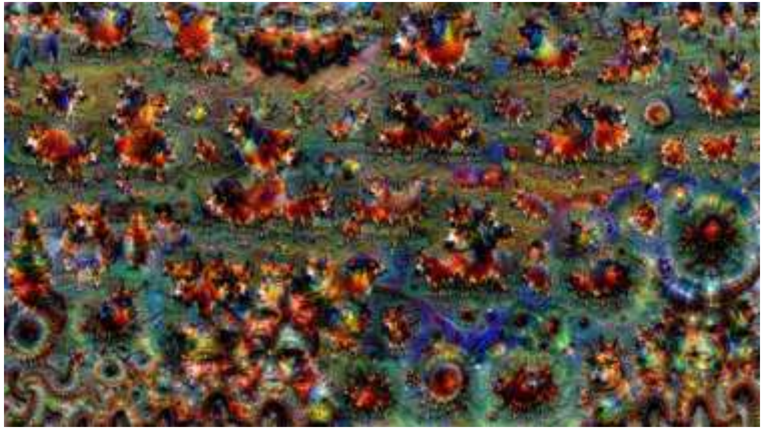
一只猫坐在地板上的手提箱

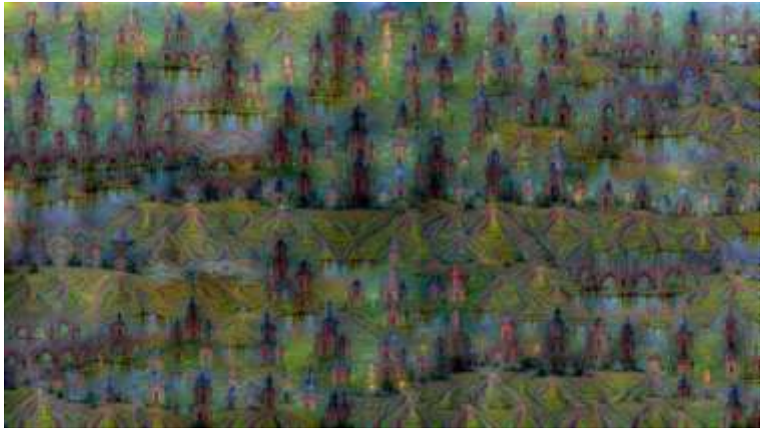
一个女人站在海滩上抱着冲浪板

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

所有图像均来自CC0公共领域：
https://pixabay.com/en/luggage-antique-cat-1643010/
https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/
https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/
https://pixabay.com/en/woman-female-model-portrait-adult-983967/
https://pixabay.com/en/handstand-lake-meditation-496008/
https://pixabay.com/en/baseball-player-shortstop-infield-1045263/

Justin Johnson使用Neuraltalk2生成的字幕。





数字版权贾斯汀·约翰逊（Justin Johnson），2015年。经许可复制。使用Inceptionism方法从Google Research的博客文章中生成。

原始图片是CC0公共领域。

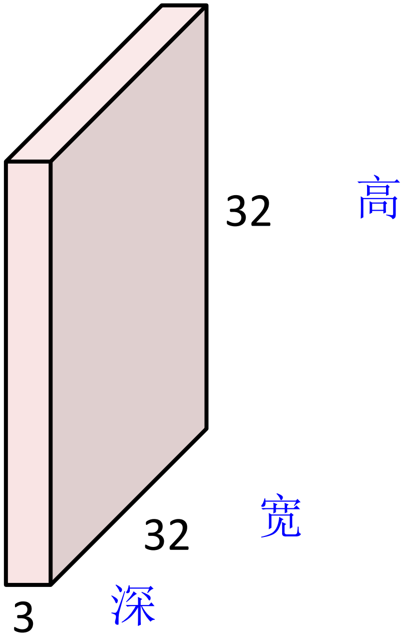
梵高的《繁星之夜》和《树的根》属于公共领域散景的图像属于公共领域风格化的图像，版权属于贾斯汀·约翰逊，2017年；经许可转载

Gatys et al, “Image Style Transfer using Convolutional Neural Networks”, CVPR 2016
Gatys et al, “Controlling Perceptual Factors in Neural Style Transfer”, CVPR 2017

卷积神经网络

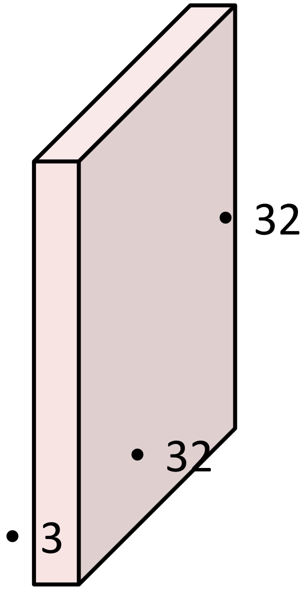
卷积层

32x32x3 图片 -> 保留空间结构

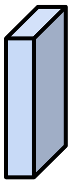


卷积层

- 32x32x3 图片



- 5x5x3 过滤器

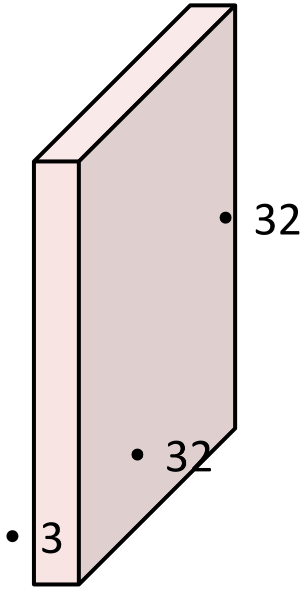


- 将滤镜与图像进行**卷积**，即“在空间上滑动图像，计算点积”

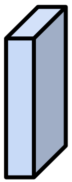
卷积层

过滤器始终扩展输入图片的整个深度

- 32x32x3 图片



- 5x5x3 过滤器

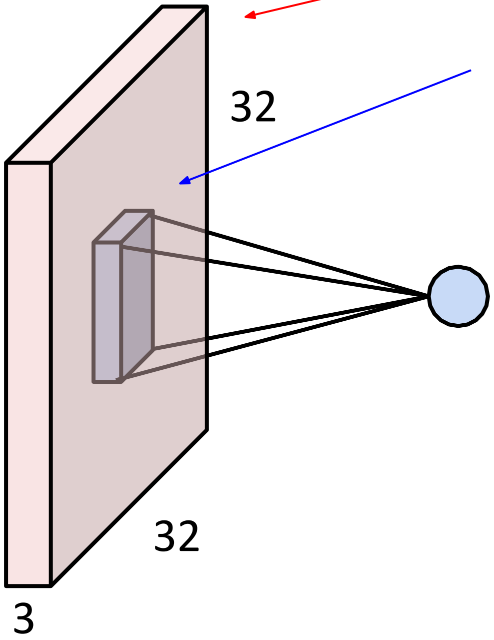


- 将滤镜与图像进行**卷积**，即“在空间上滑动图像，计算点积”

卷积层

32x32x3 图片

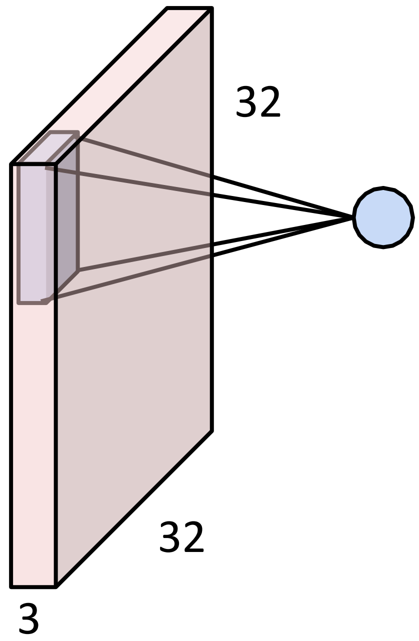
5x5x3 过滤器 w



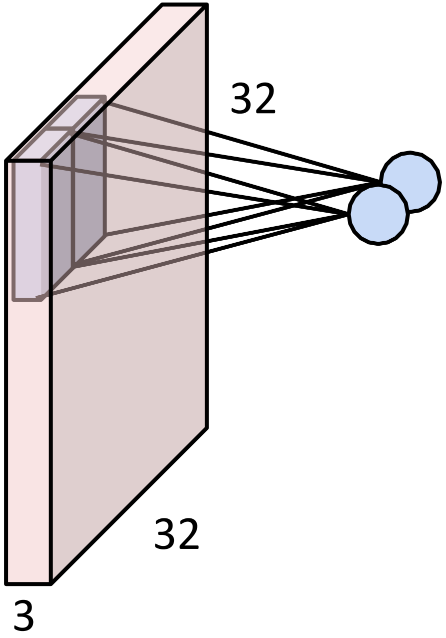
数字1：
在滤镜和图像的5x5x3小块之间取点积的结果
(即5 * 5 * 3 = 75维点积+偏差)

$$w^T x + b$$

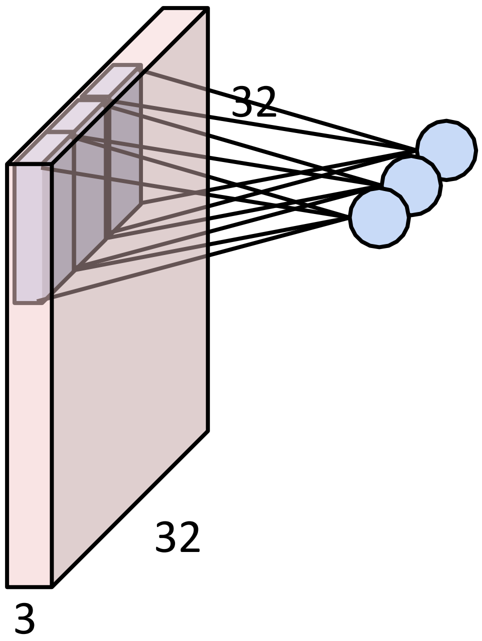
卷积层



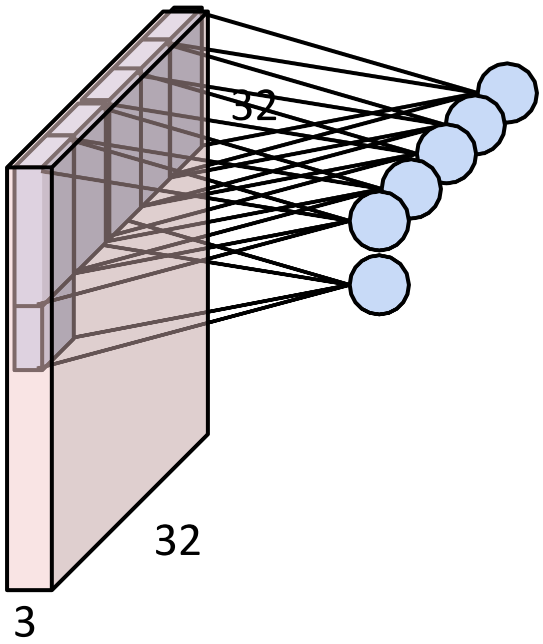
卷积层



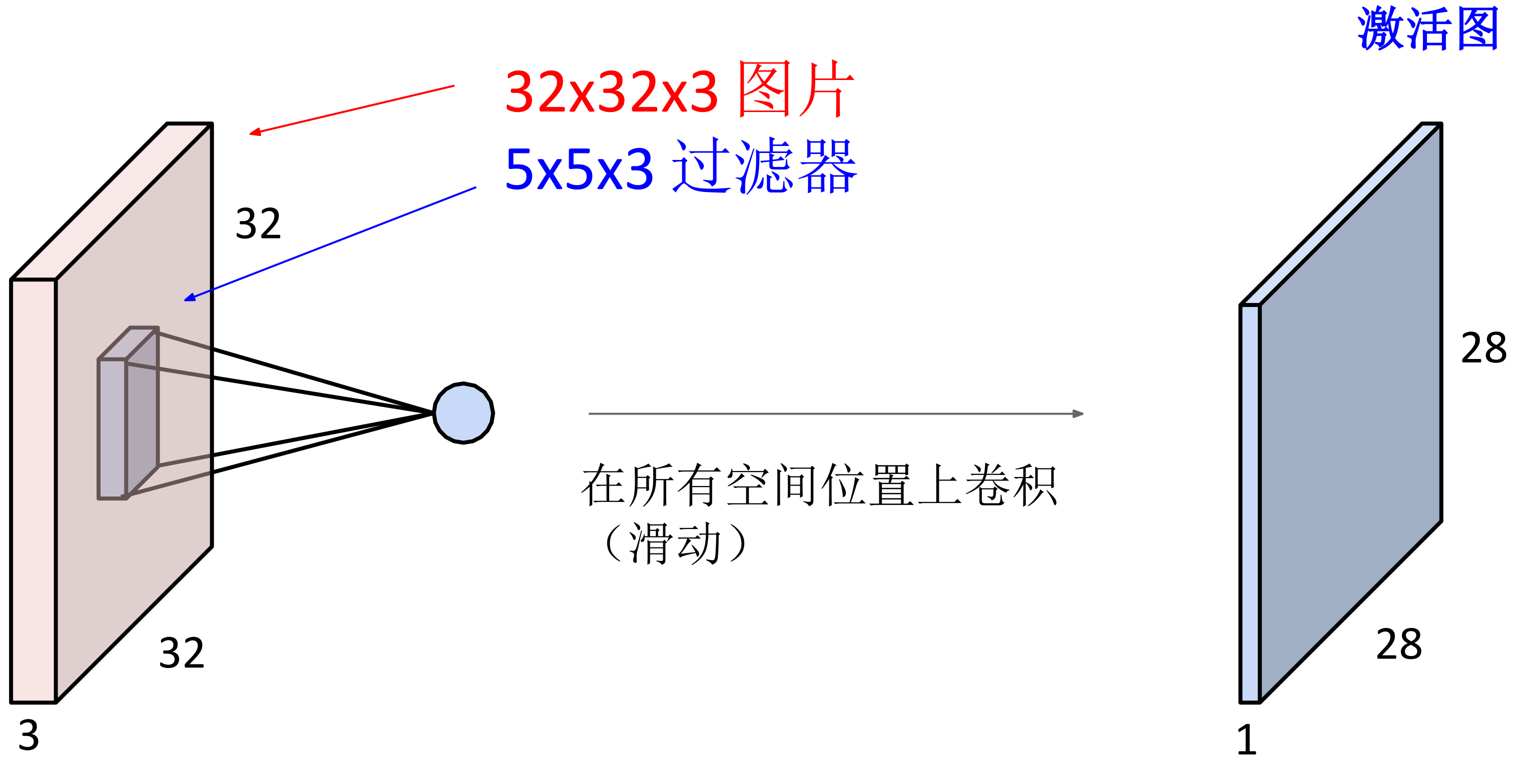
卷积层



卷积层

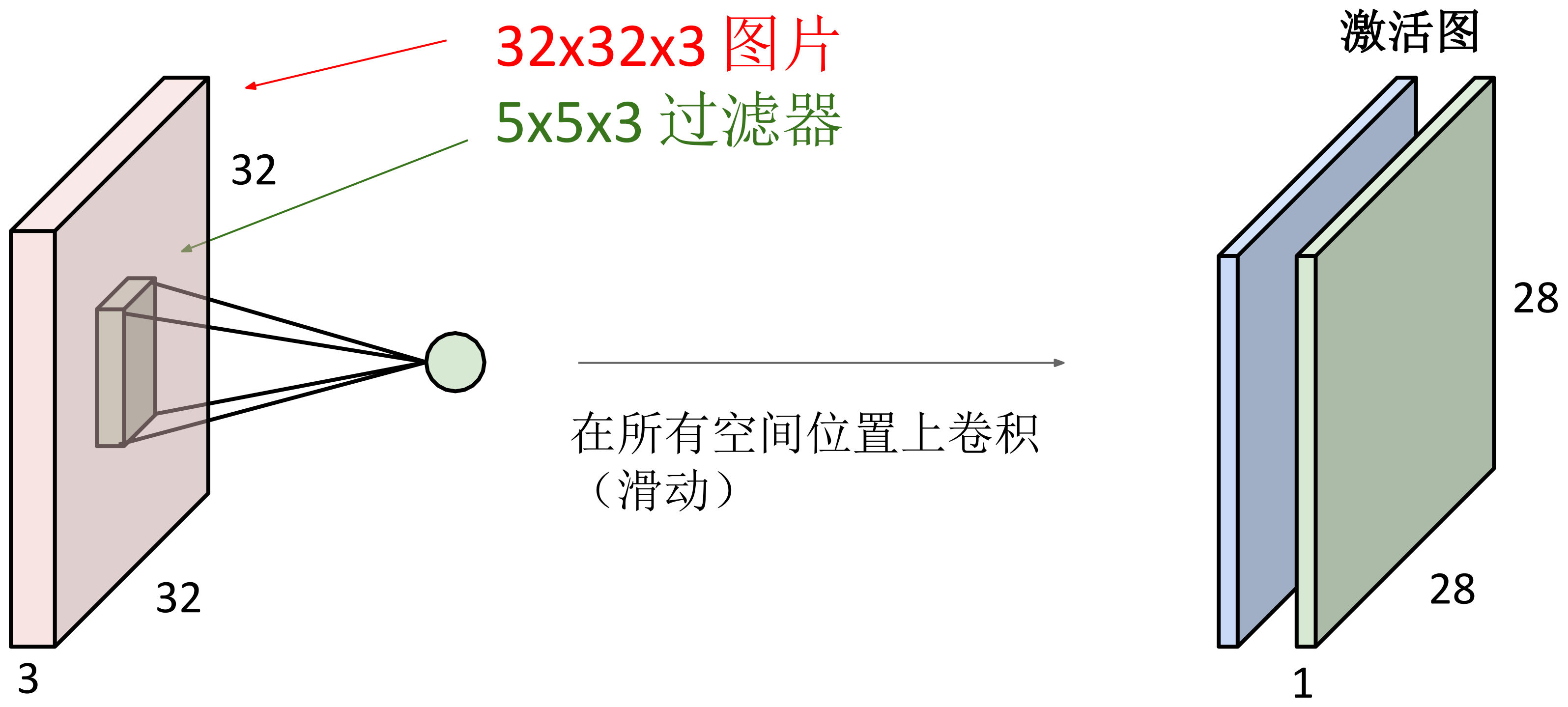


卷积层

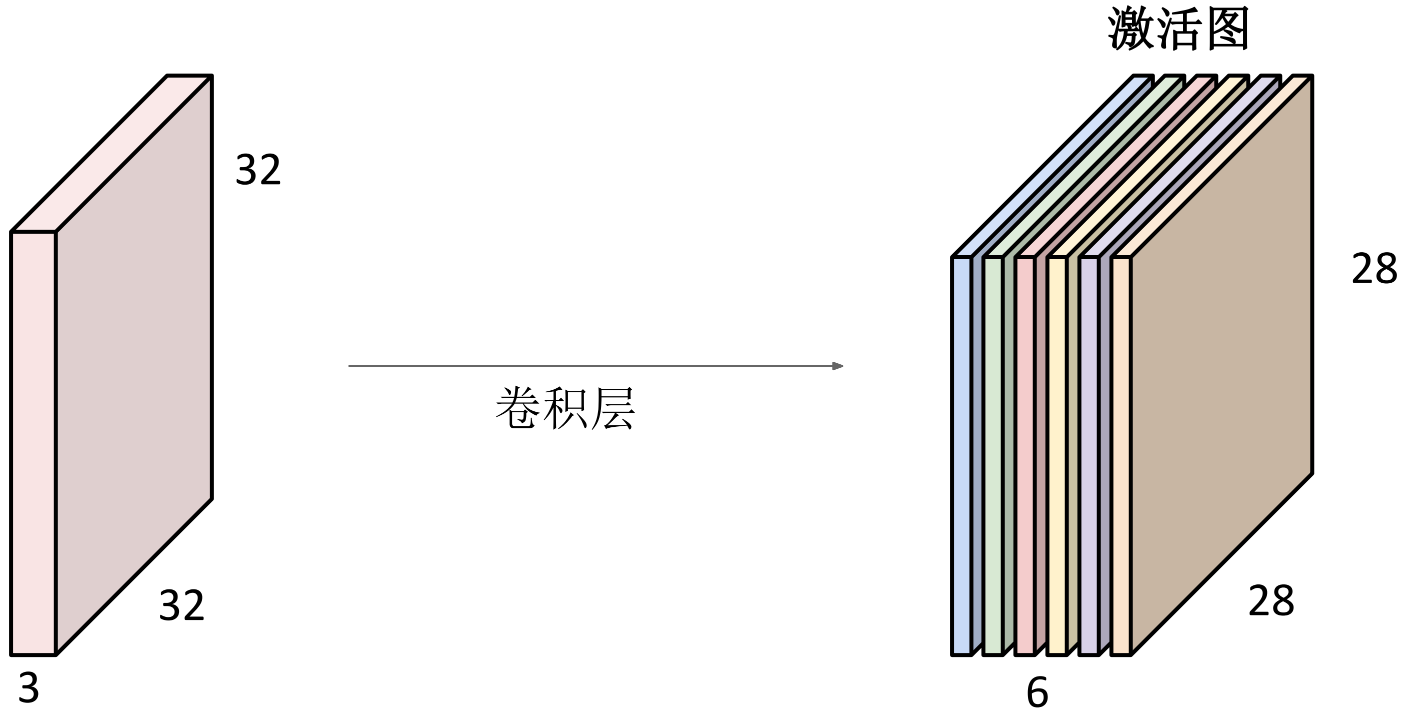


卷积层

考虑第二个绿色滤波器

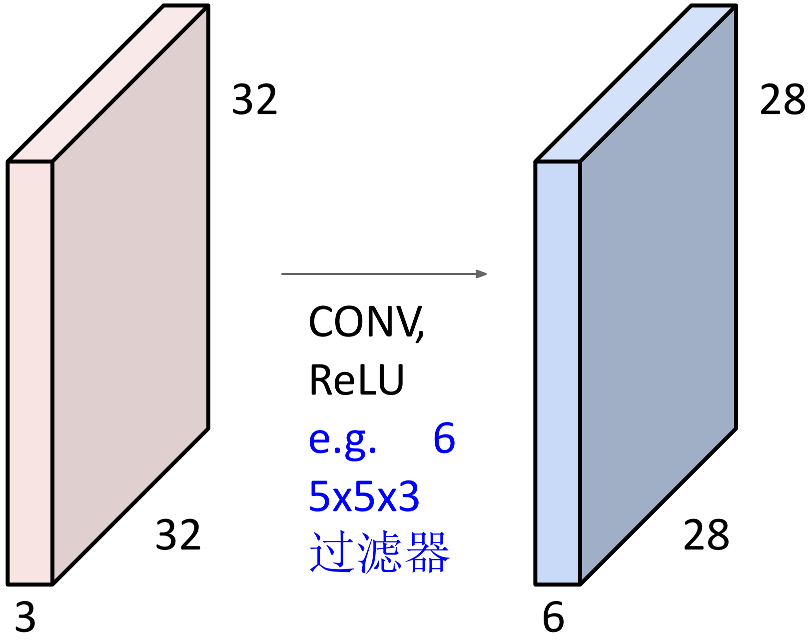


例如，如果我们有6个5x5过滤器，我们将获得6个单独的激活图：

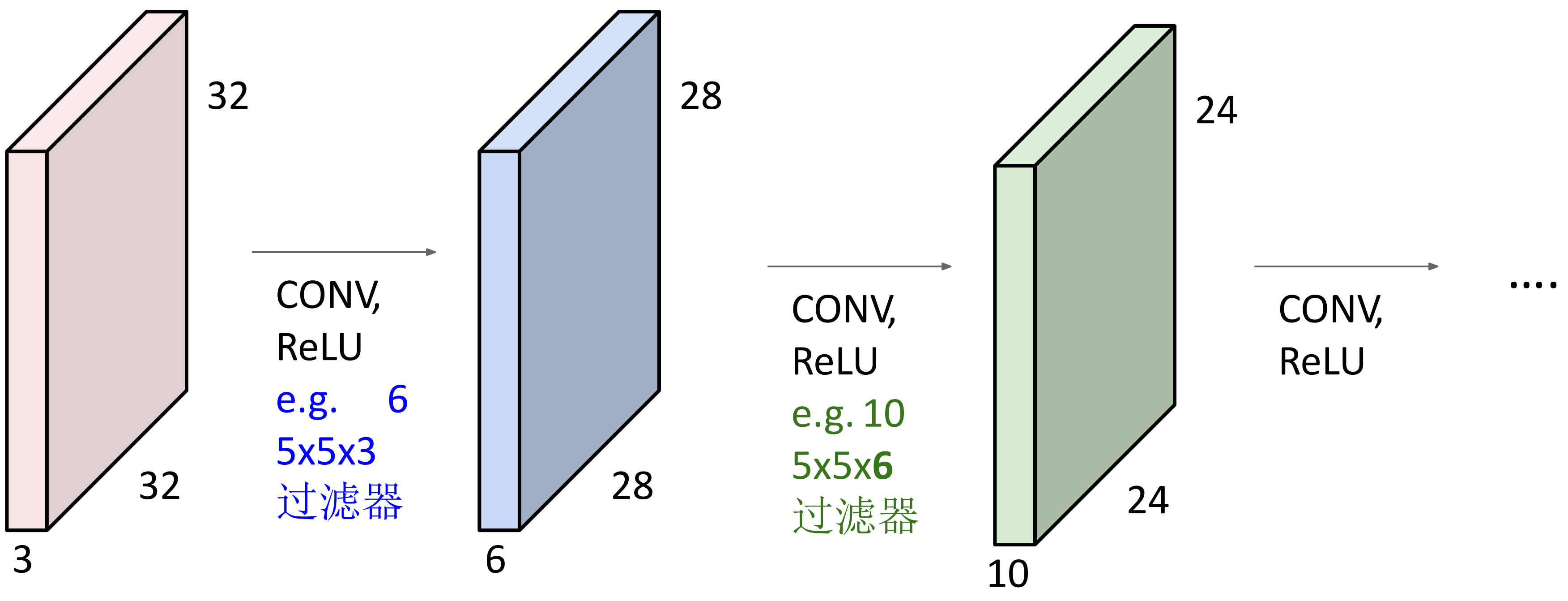


我们将它们堆叠起来，以获得尺寸为28x28x6的“新图片”！

预览：卷积网络是一系列卷积层，散布着激活功能



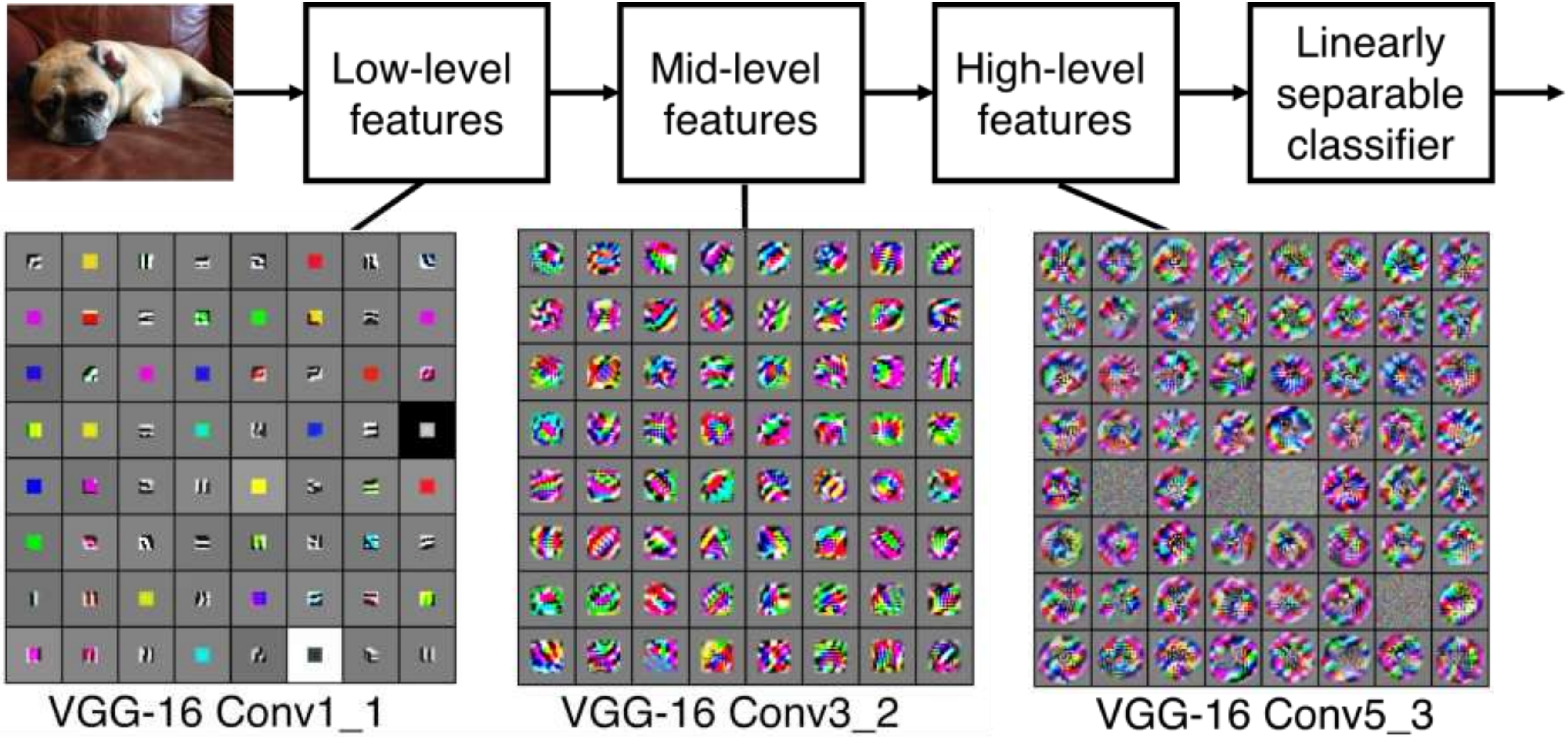
预览：卷积网络是一系列卷积层，散布着激活功能



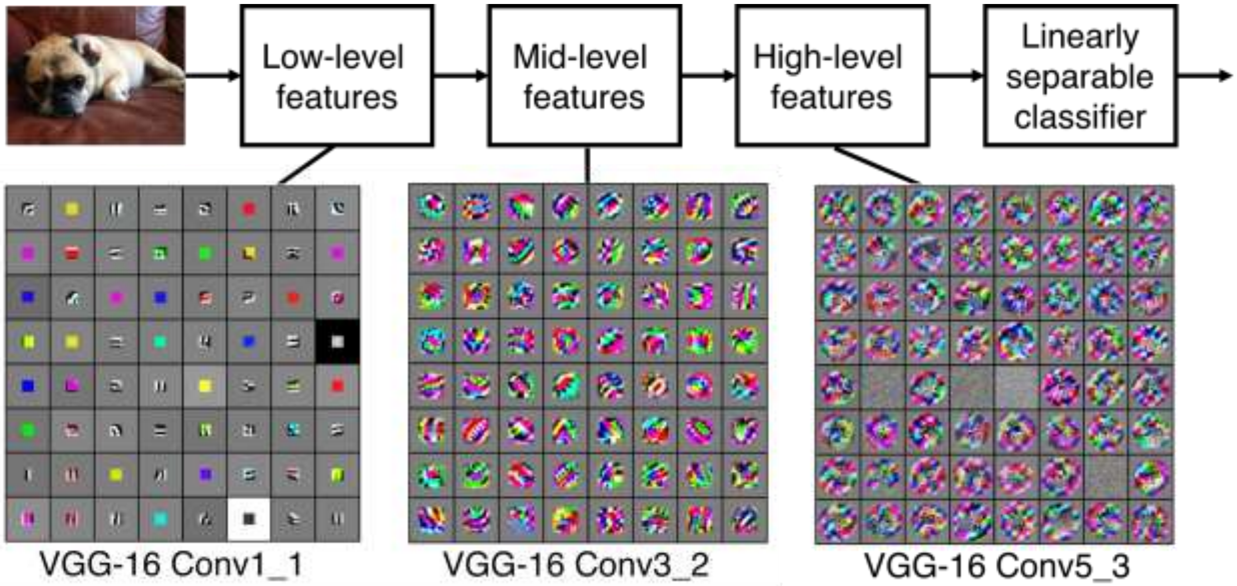
预览

[Zeiler and Fergus 2013]

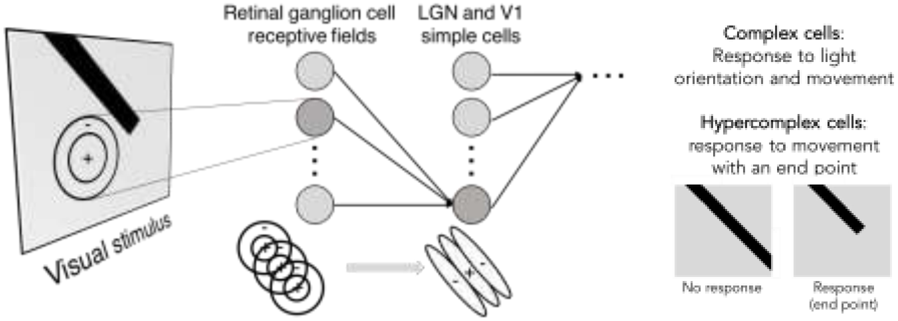
Lane McIntosh展示的VGG-16。 [Simonyan和Zisserman，2014年]的VGG-16架构。



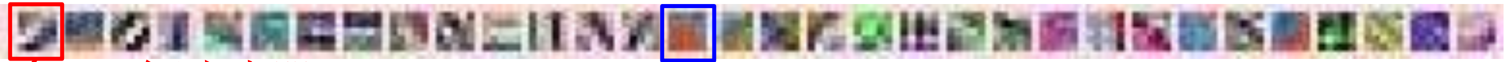
预览



Low-level features
Mid-level features
High-level features
Linearly separable classifier
VGG-16 Conv1_1
VGG-16 Conv3_2
VGG-16 Conv5_3

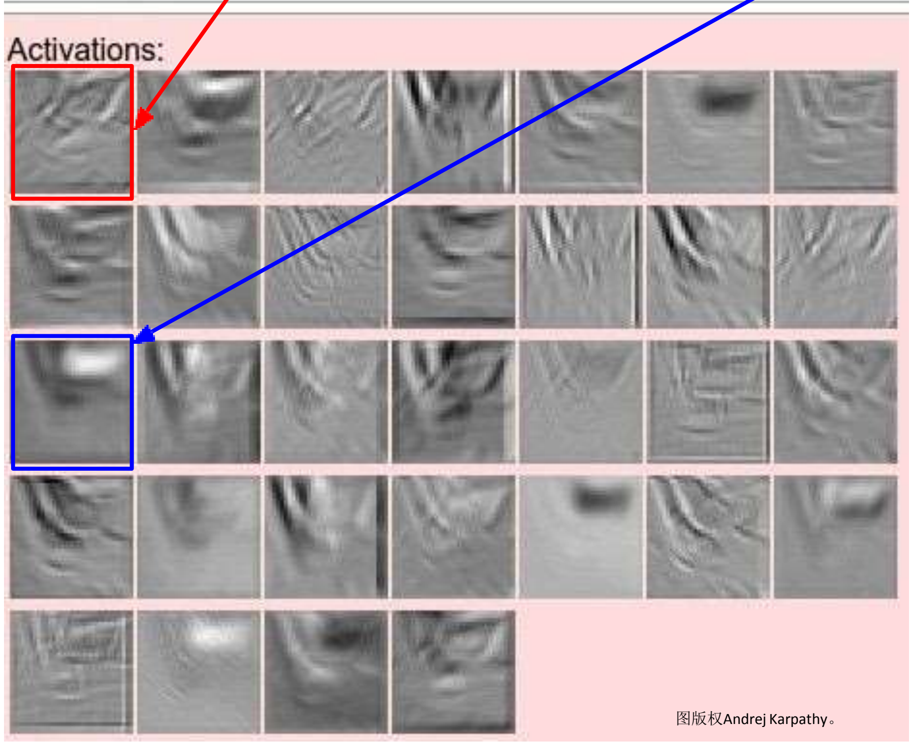


Retinal ganglion cell receptive fields
LGN and V1 simple cells
Visual stimulus
Complex cells:
Response to light orientation and movement
Hypercomplex cells:
response to movement with an end point
No response
Response (end point)



一个过滤器 =>
一个激活图

示例5x5过滤器
（共32个）



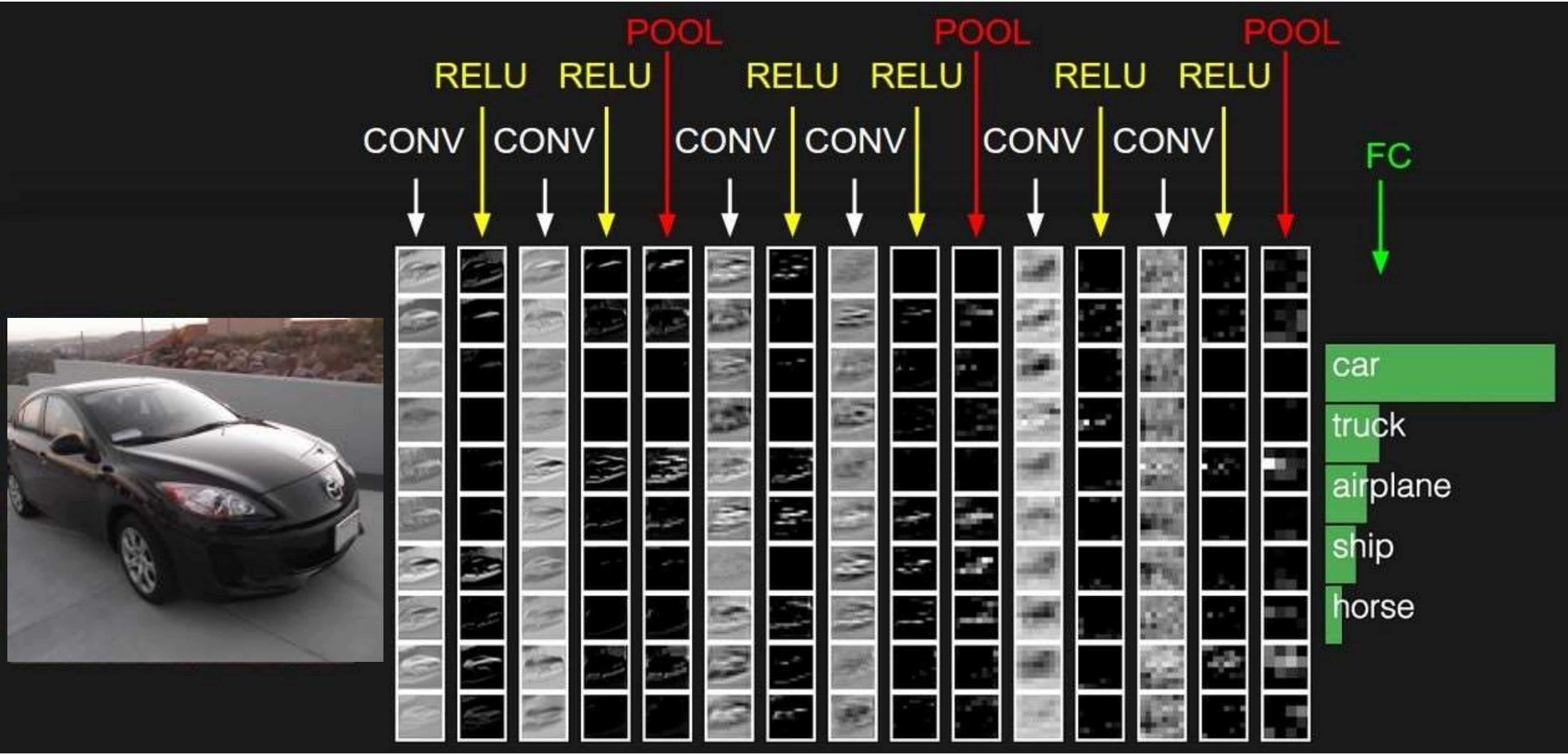
图版权Andrej Karpathy。

我们称其为卷积层，因为它与两个信号的卷积有关：

$$f[x,y]*g[x,y] = \sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty} f[n_1,n_2]\cdot g[x-n_1,y-n_2]$$

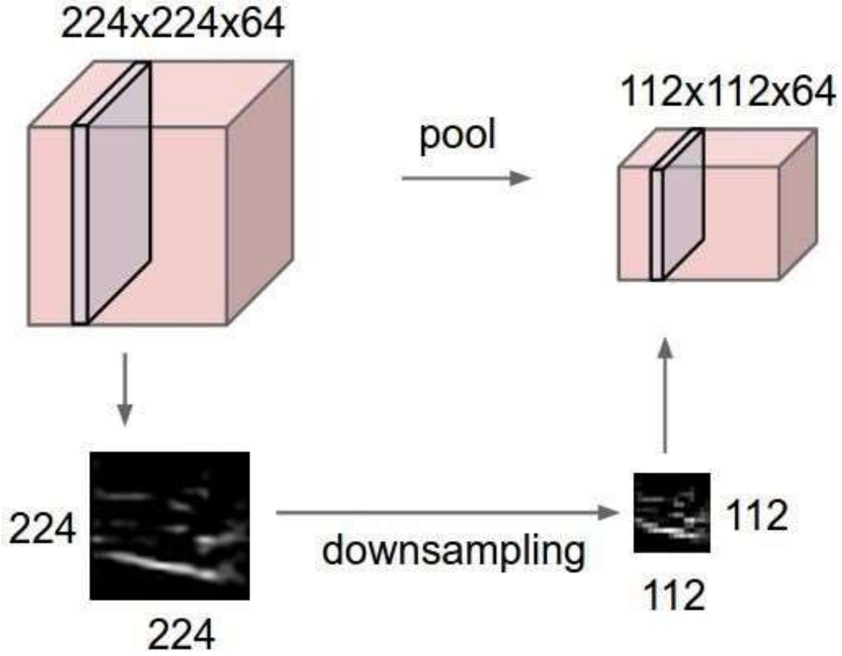
滤波器与信号（图像）的逐元素相乘与和

预览：

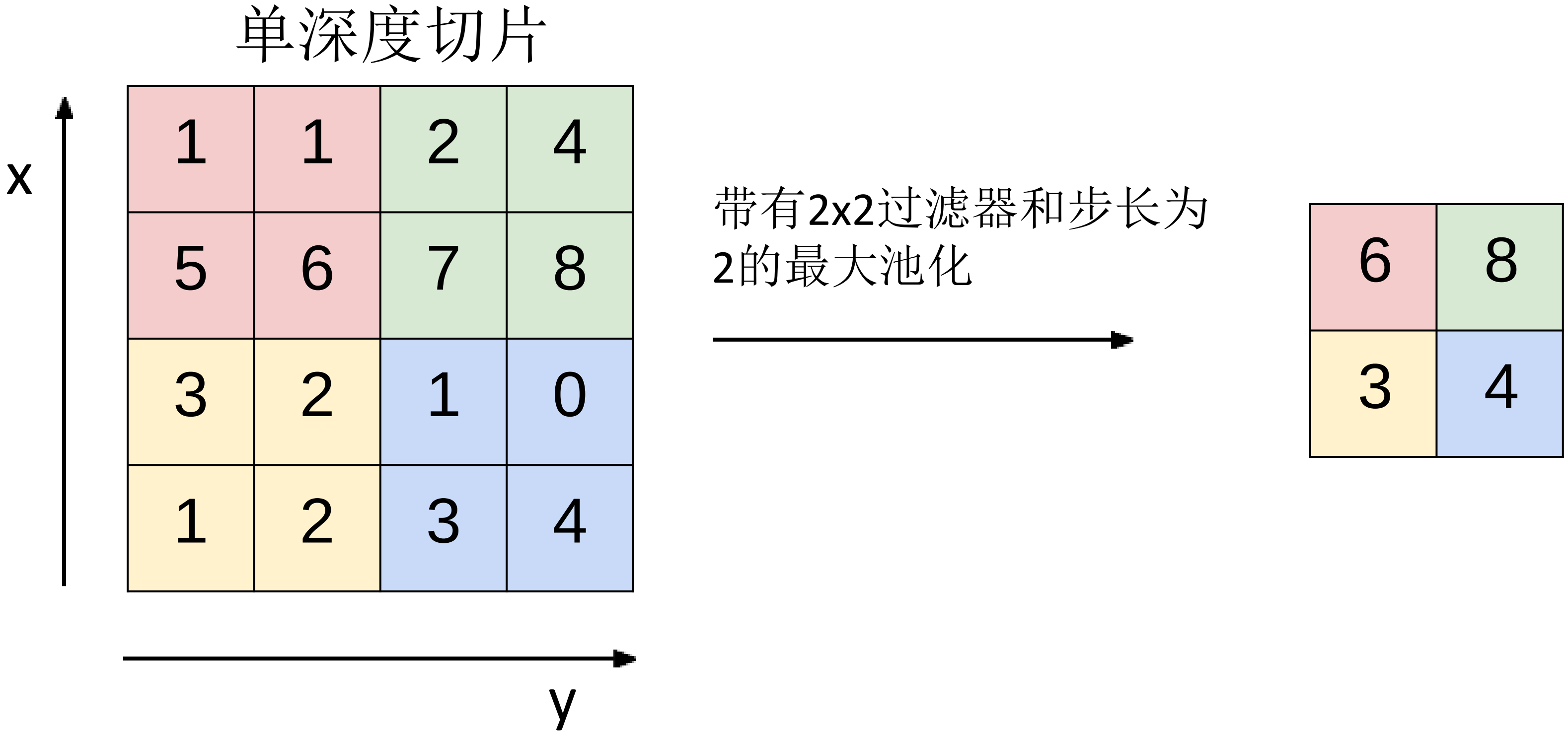

POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

池化层

- 使表示更小，更易于管理
- 对每个激活图进行独立操作：

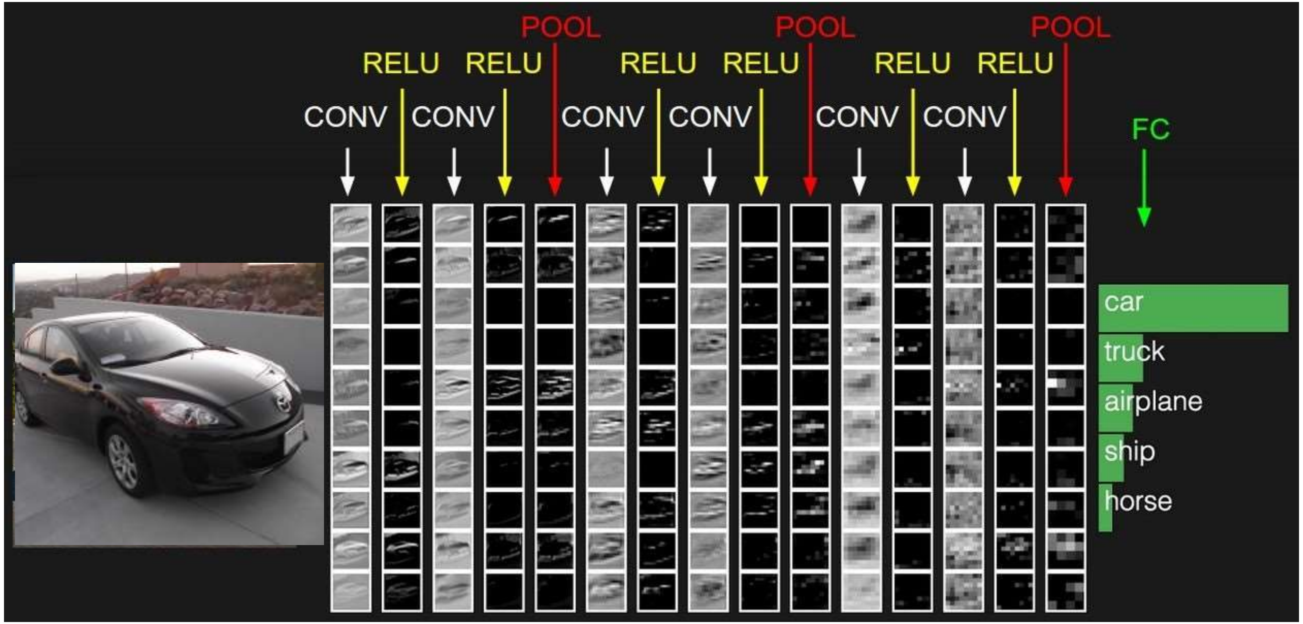


最大池化



全连接层(FC layer)

- 包含连接到整个输入量的神经元，就像普通的神经网络一样



总结

- 卷积层堆叠CONV，POOL，FC层
- 朝着更小的过滤器和更深的架构发展的趋势
- 摆脱POOL / FC层的趋势（仅CONV）
- 从历史上看，结构看起来像

  [(CONV-RELU)*N-POOL?]*M-(FC-RELU)*K,SOFTMAX

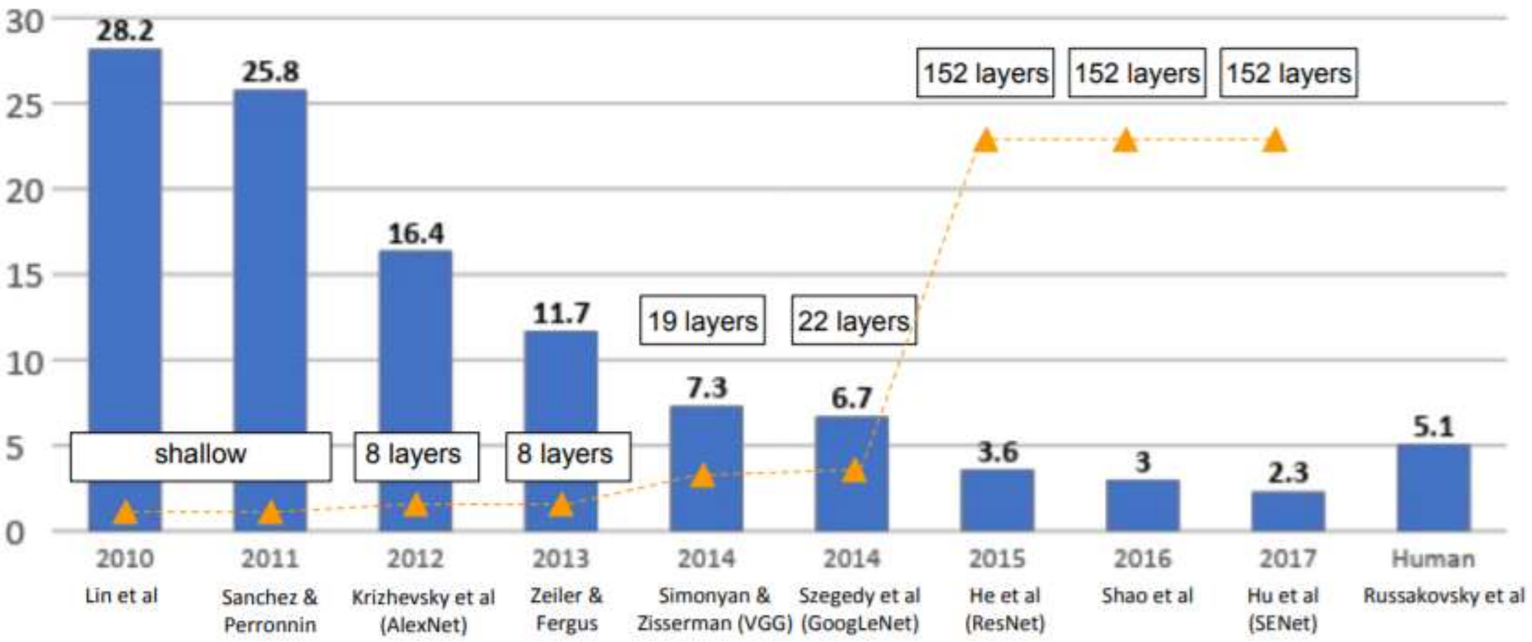
  其中N通常是~5, M是较大的值, 0 <= K <= 2.

  - 但是诸如ResNet / GoogLeNet之类的最新进展已经挑战了这种范例

主流CNN结构演进

- AlexNet
- VGGNet
- GoogLeNet
- ResNet

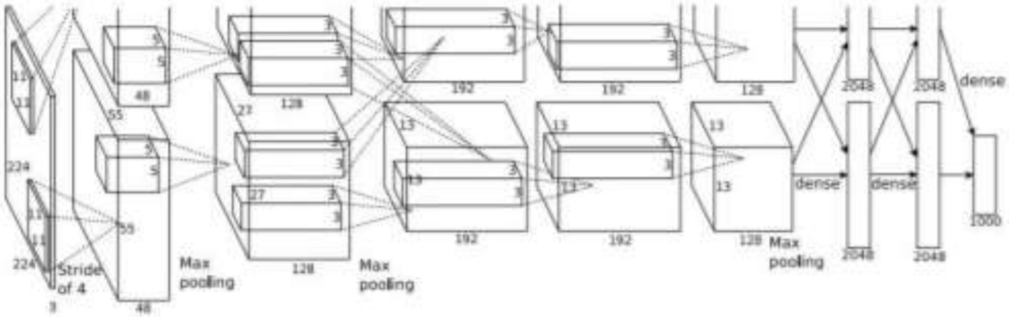
ImageNet 分类挑战(ILSVRC)



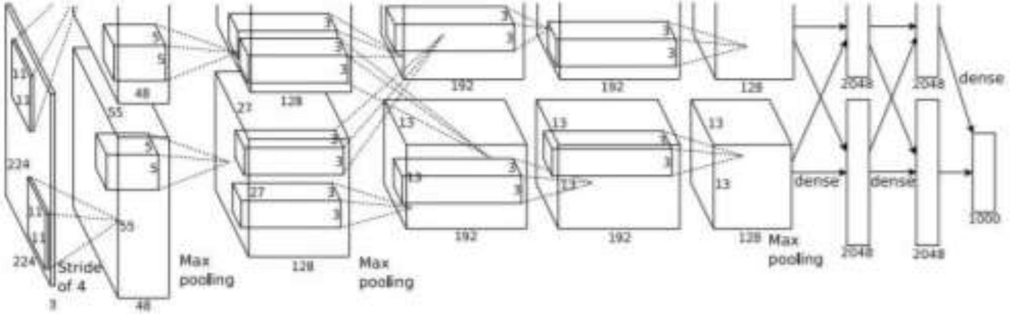
AlexNet

结构

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

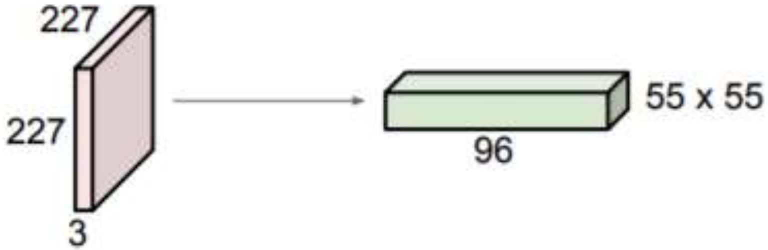


AlexNet



输入：227x227x3 图像

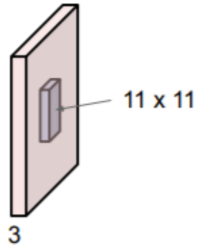
第一层(CONV1):96个 11x11 滤波器，步长为4



输出？

(227-11)/4+1 = 55

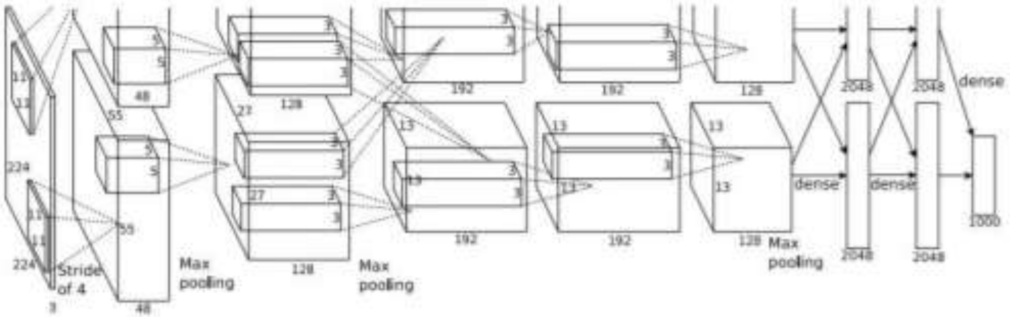
故为55x55x96



这一层又有多少参数？

(11x11x3)*96 = 35K

AlexNet



输入：227x227x3 图像

经过第一层后：55x55x96

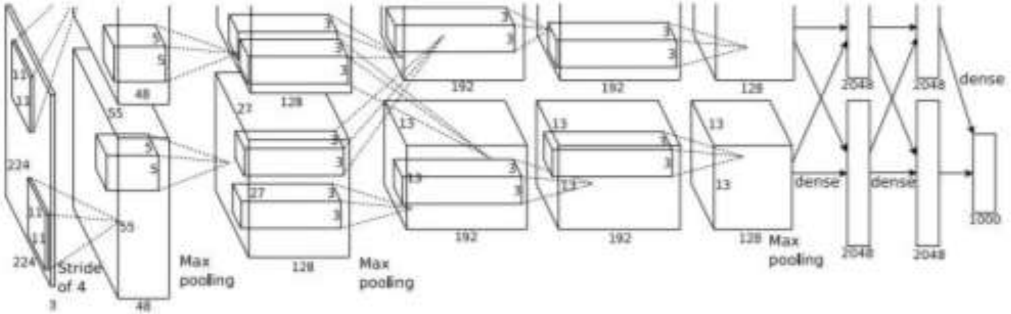
第二层(POOL1):3x3滤波器，步长为2　输出？

(55-3)/2+1 = 27　　故为27x27x96

这一层又有多少参数？

0!

AlexNet



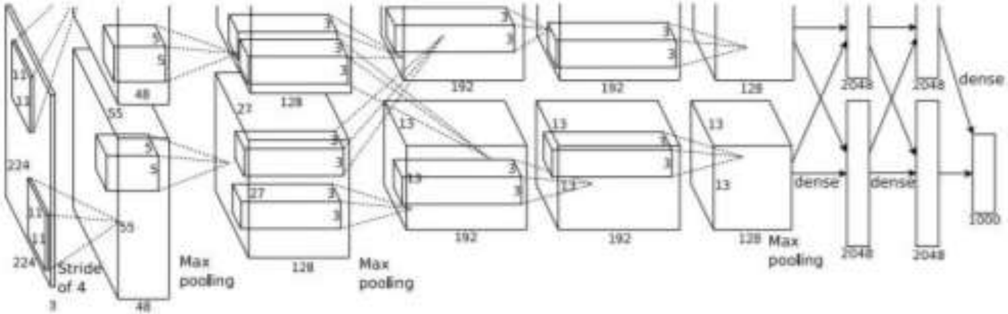
AlexNet完整的结构和输出

[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)

一些细节:

- 第一次使用ReLU
- 引入很多增广方法
- Dropout 0.5
- Batch size 128
- 学习率1e-2 当测试率瓶颈时调整为1/10

AlexNet



AlexNet完整的结构和输出

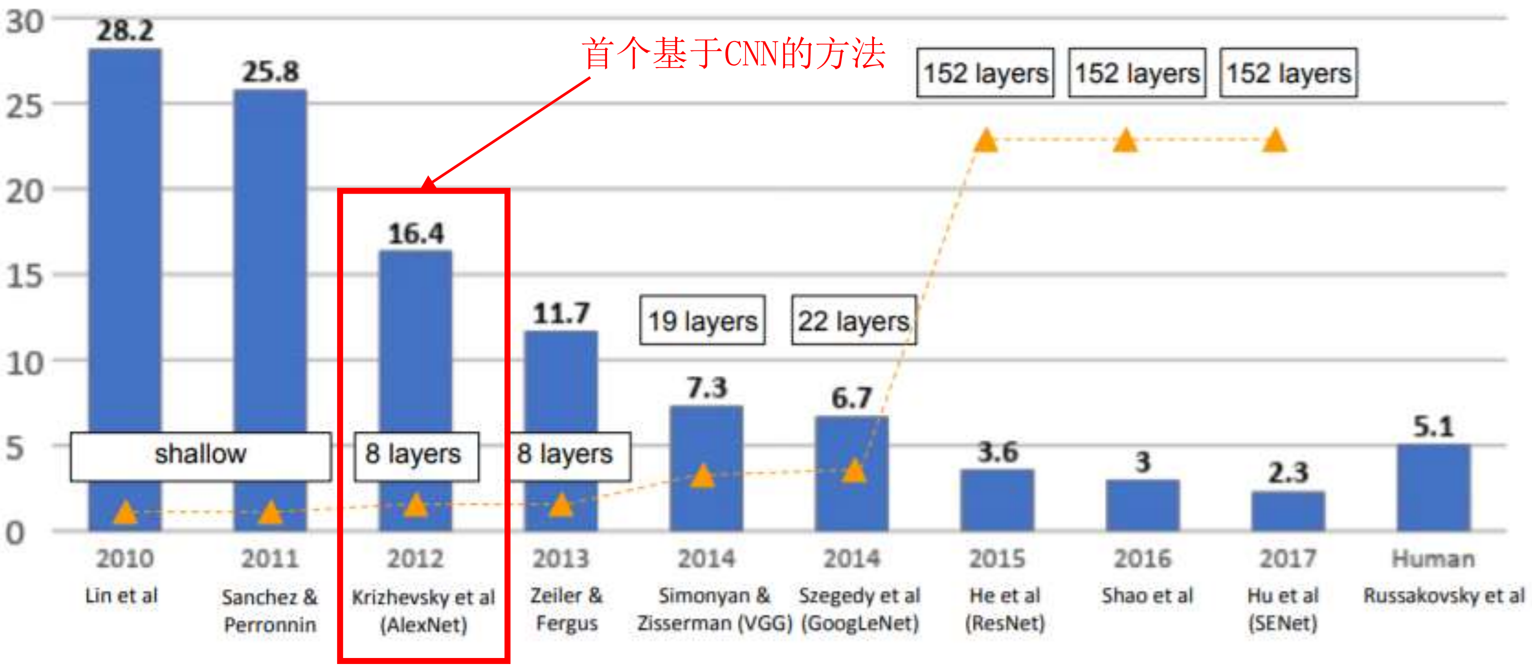
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] 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)

由于当时算力的不足，整个网络分布在了两块GPU上，导致CONV1， CONV2， CONV3， CONV4， CONV5，只和同在一个卡上的特征图相连

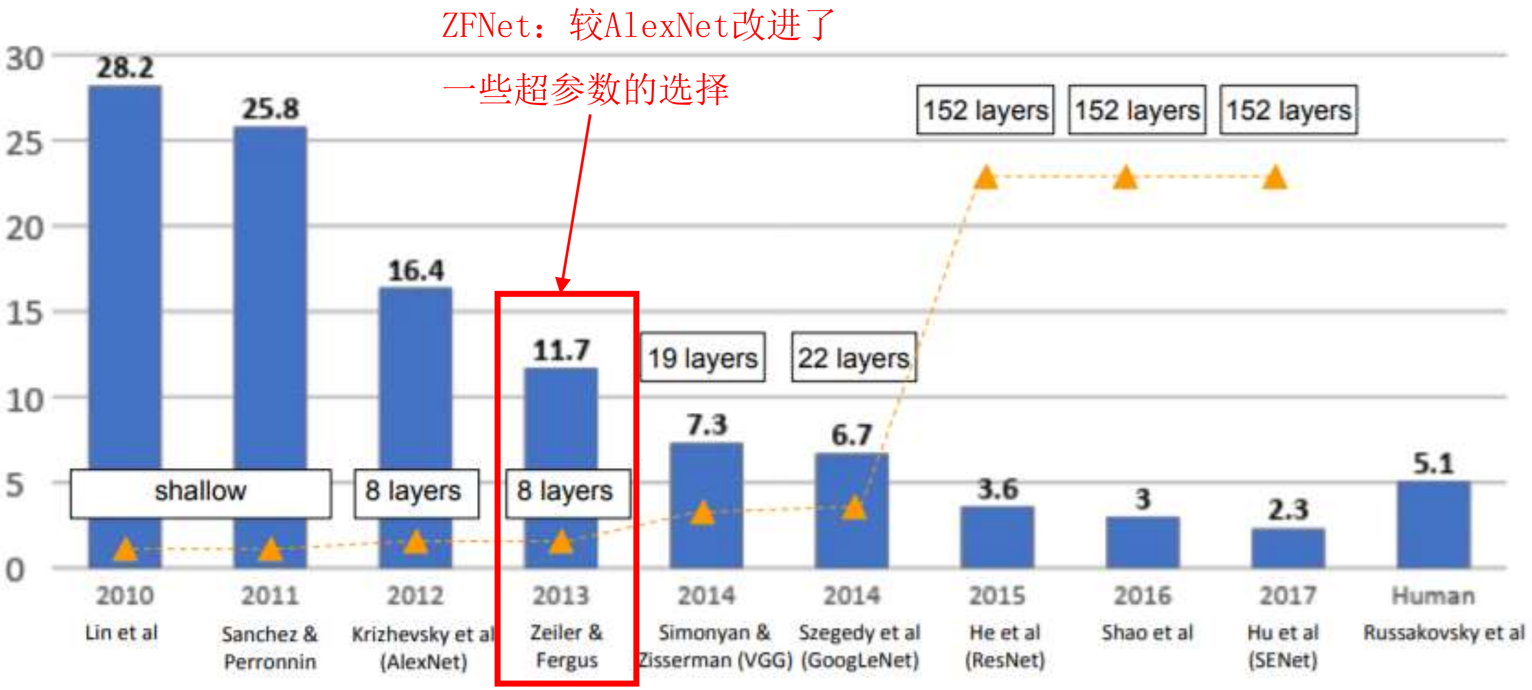
CONV3，FC6，FC7，FC8与所有之前的特征相连

AlexNet也是第一个采用CNN在ILSVRC取胜的方法

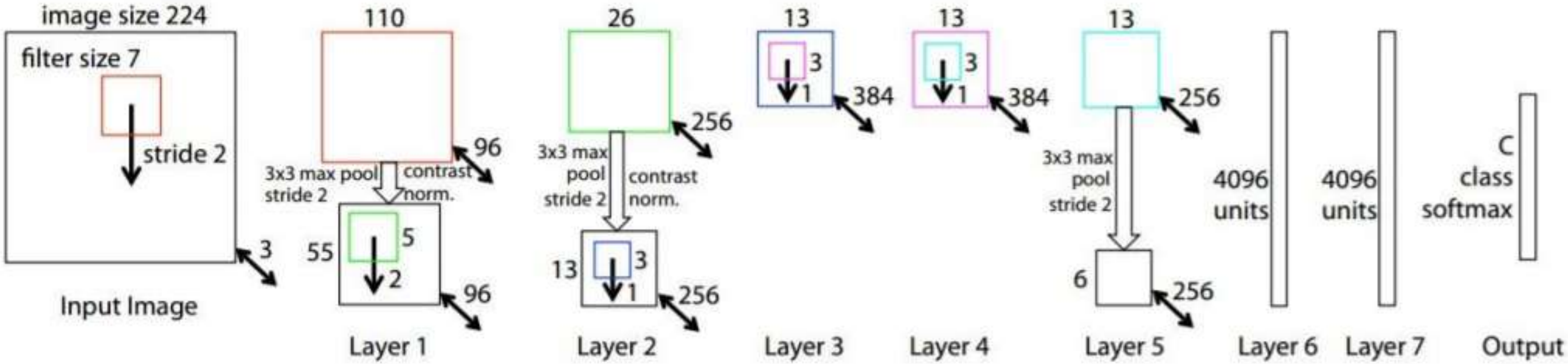
ImageNet 分类挑战(ILSVRC)



ImageNet 分类挑战(ILSVRC)



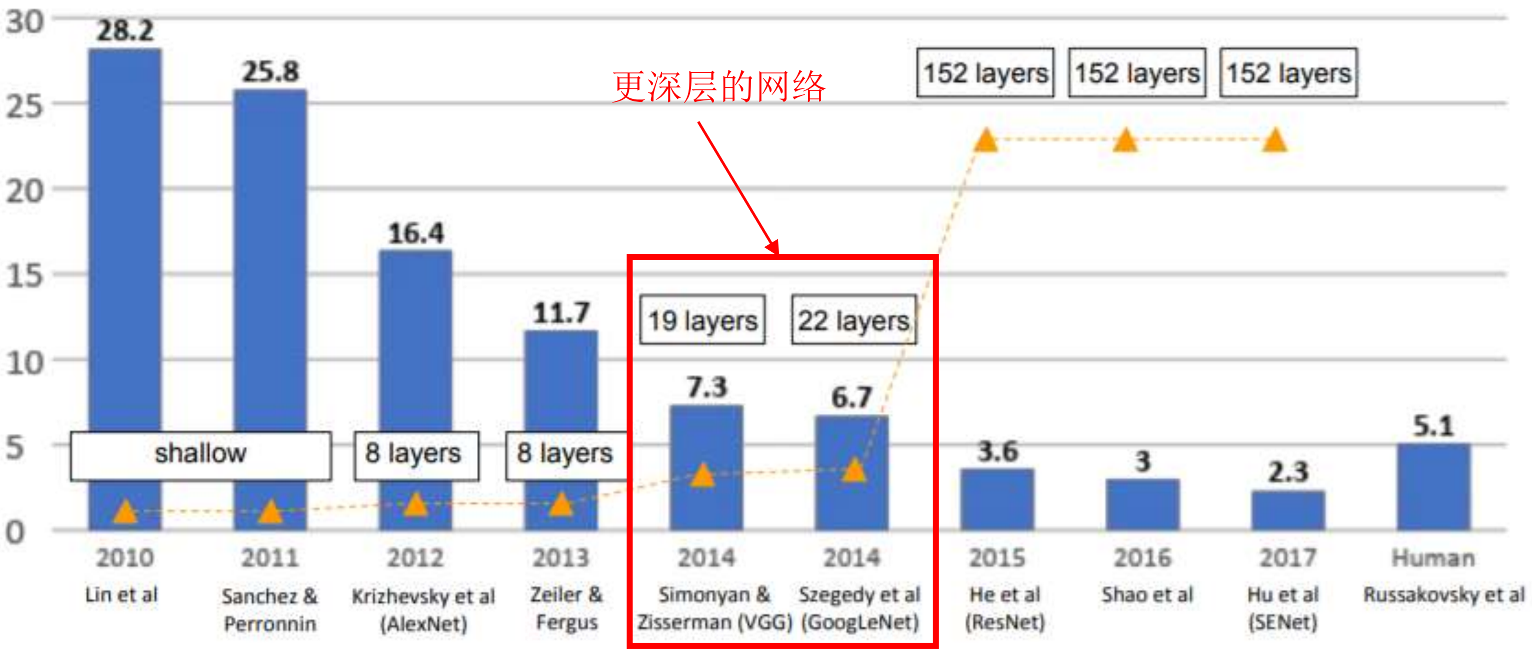
ZFNet



CONV1:从(11x11 步长4)改为了(7x7 步长2)

CONV3,4,5:滤波器数量从384，384，256 变为512，1024，512

ImageNet 分类挑战(ILSVRC)



VGGNet

用更小的卷积核

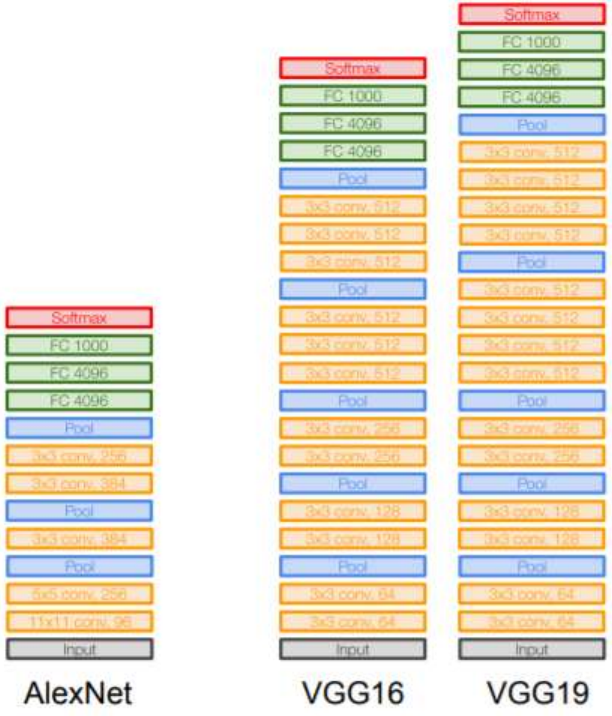
更深的网络

从AlexNet的8层

变为16/19层

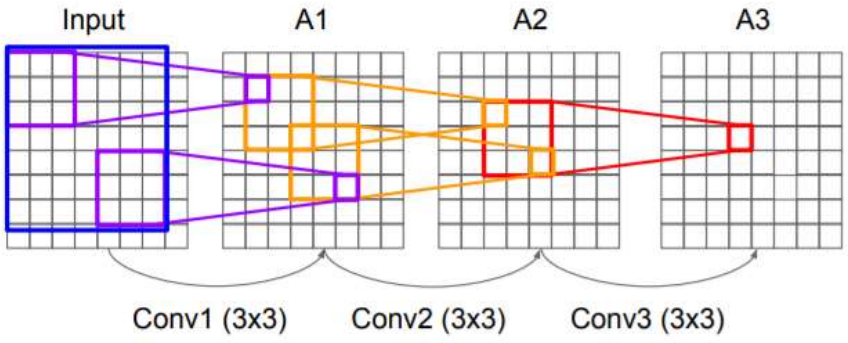
只采用3x3的卷积核，stride 1 pad 1

以及 2x2的最大池化，stride 2



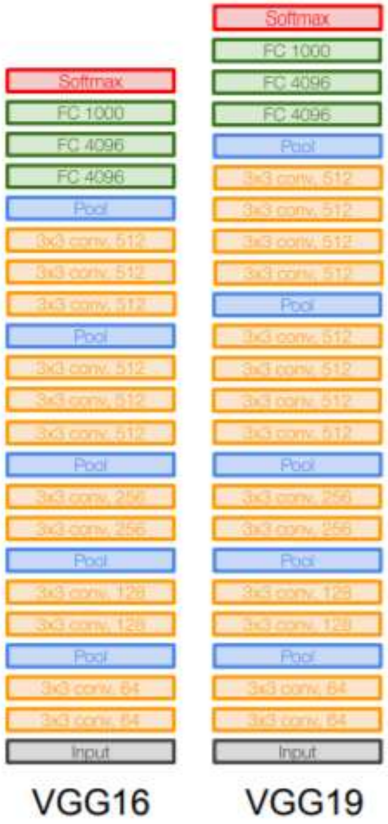
VGGNet

为什么用3x3的卷积核？



在有同样感受野的情况下，拥有更多的非线性层

同时具有更少的参数量



VGGNet

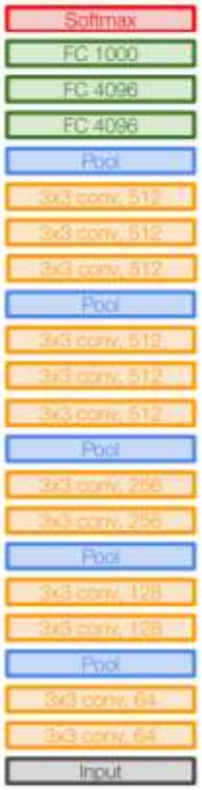
(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
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

大部分的内存占用

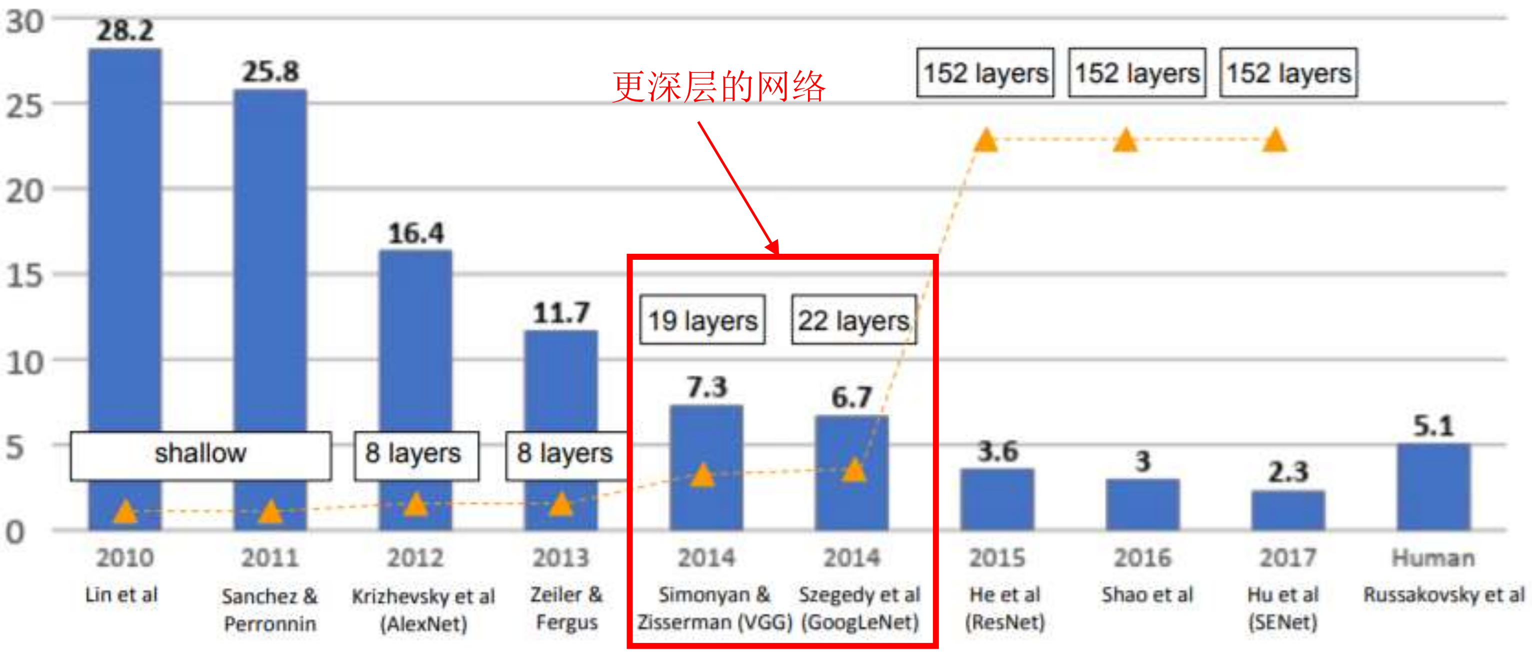
大部分的参数

TOTAL memory: 24M * 4 bytes ~= 96MB / image (for a forward pass)
TOTAL params: 138M parameters



VGG16

ImageNet 分类挑战(ILSVRC)

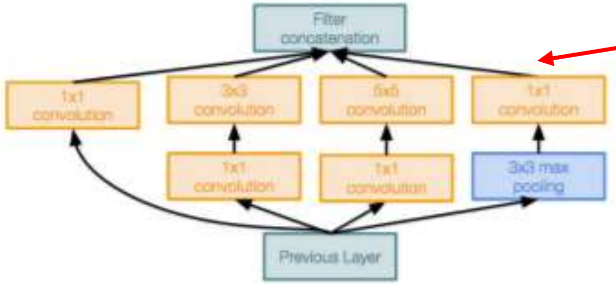


GoogLeNet

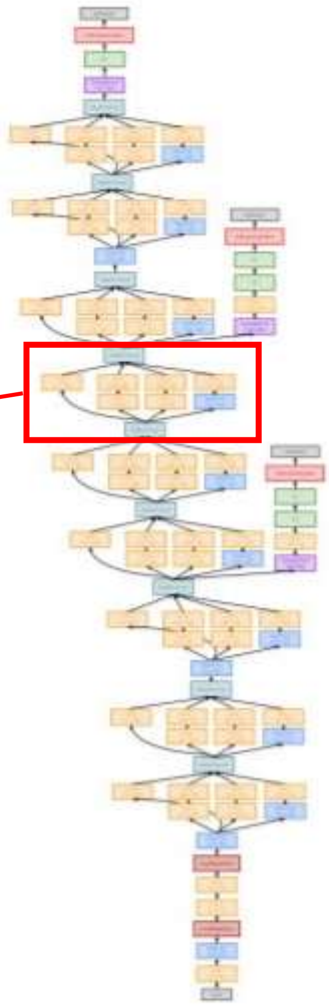
- 22层
- 只有5million的参数

  比AlexNet少12倍

  比VGG-16少27倍
- 采用了有效的Inception模块
- 没有FC层



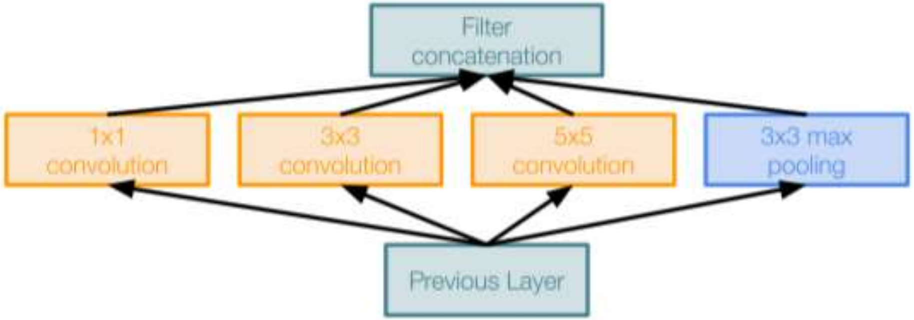
Inception module



简单的Inception 模块

- 并行的进行三种不同大小卷积核的计算
- 将不同卷积核的计算结果连接起来

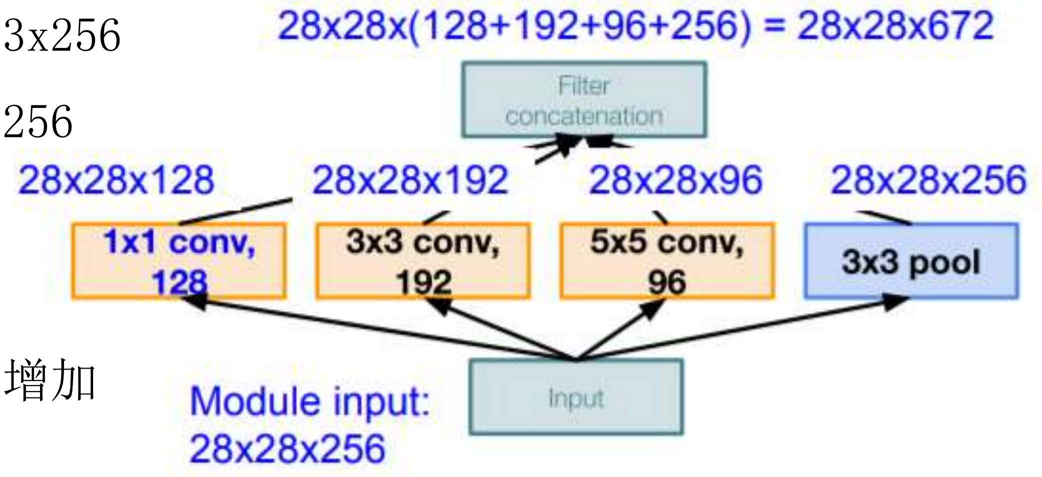
- 这样会有什么问题？



Naive Inception module

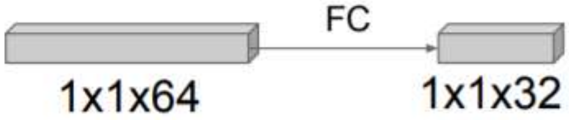
简单的Inception 模块

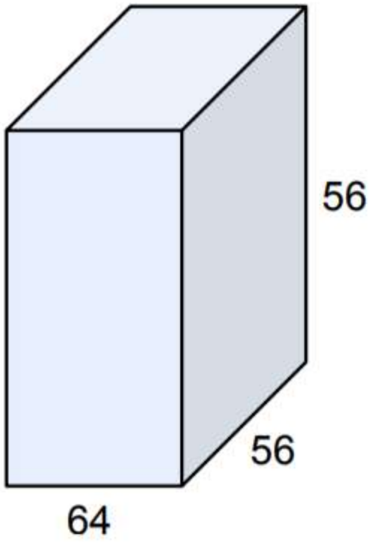
- [1x1 conv, 128] 28x28x128x1x1x256
- [3x3 conv, 192] 28x28x192x3x3x256
- [5x5 conv, 96] 28x28x96x5x5x256
- 总共: 854M ops



- 计算代价太大，同时只能不断地增加特征图的深度
- 利用1x1卷积核

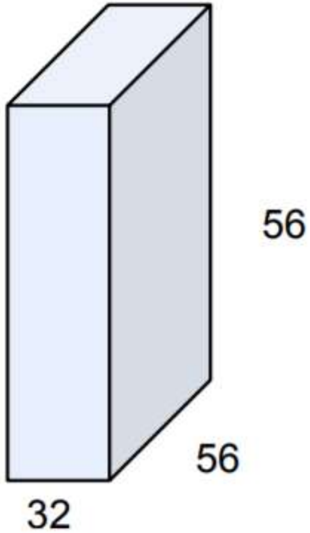
1x1 卷积核



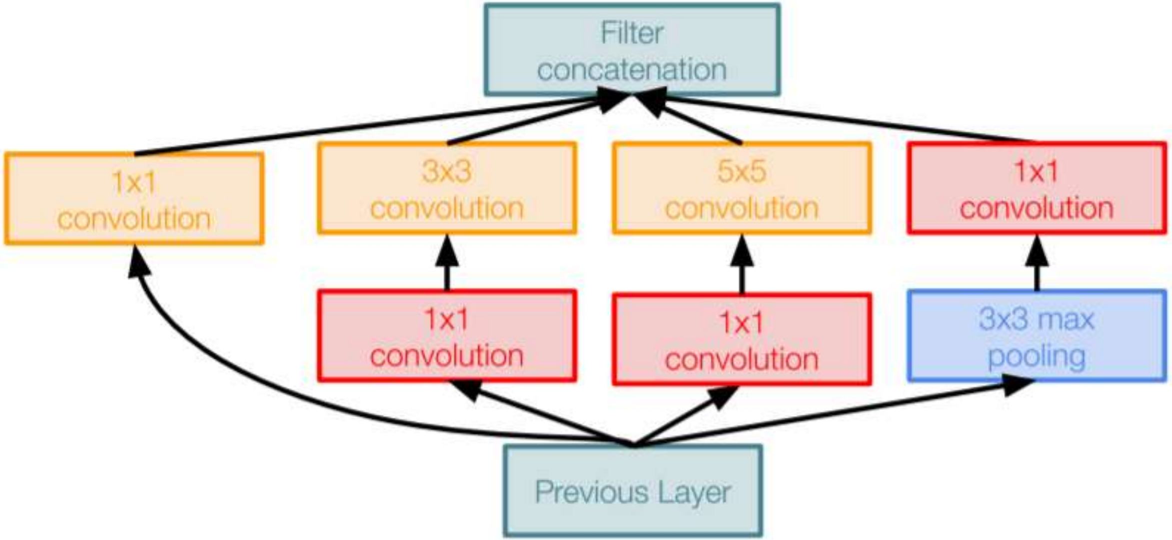


32个 1x1 conv

- 保持空间维度的同时减少深度

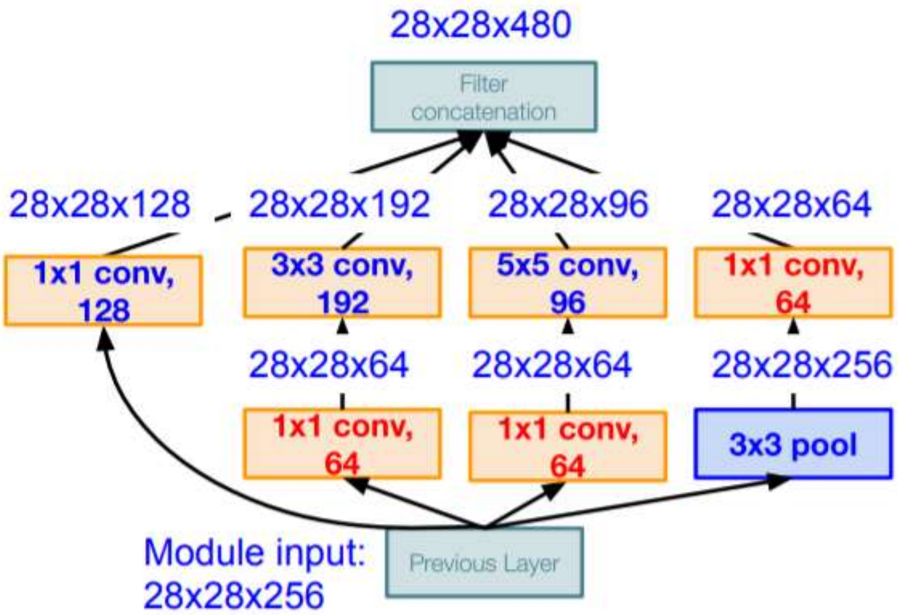


实际的Inception 模块



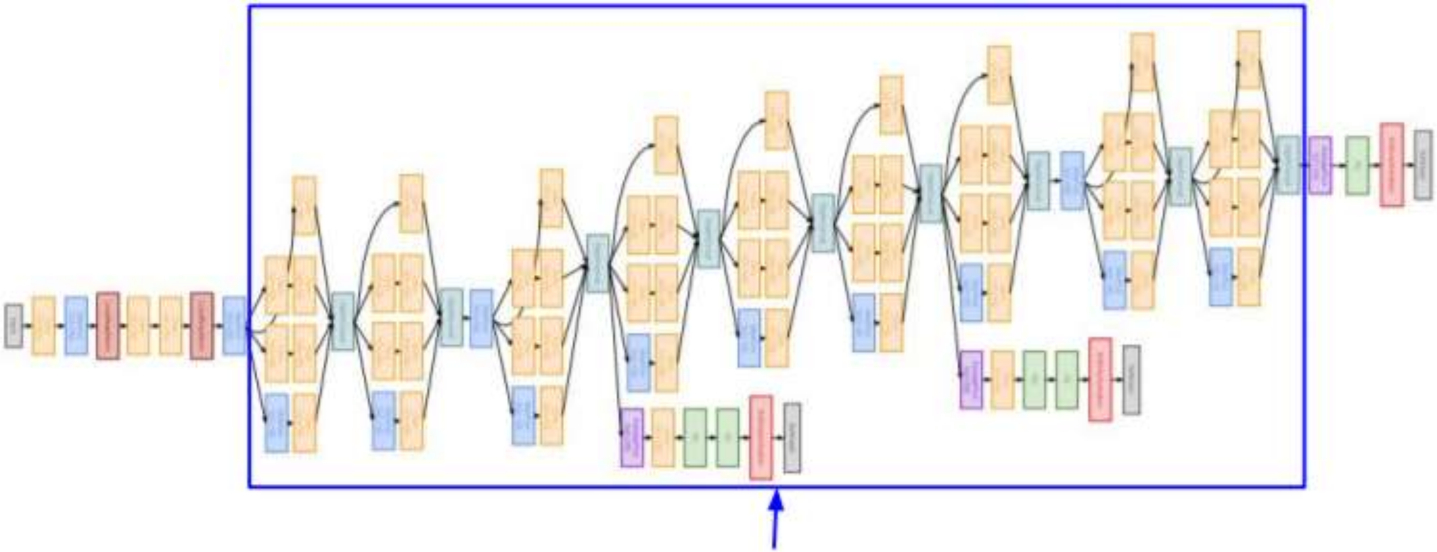
实际的Inception 模块

- [1x1 conv, 64] 28x28x64x1x1x256
- [1x1 conv, 64] 28x28x64x1x1x256
- [1x1 conv, 128] 28x28x128x1x1x256
- [3x3 conv, 192] 28x28x192x3x3x64
- [5x5 conv, 96] 28x28x96x5x5x64
- [1x1 conv, 64] 28x28x64x1x1x256
- 总共: 358M ops



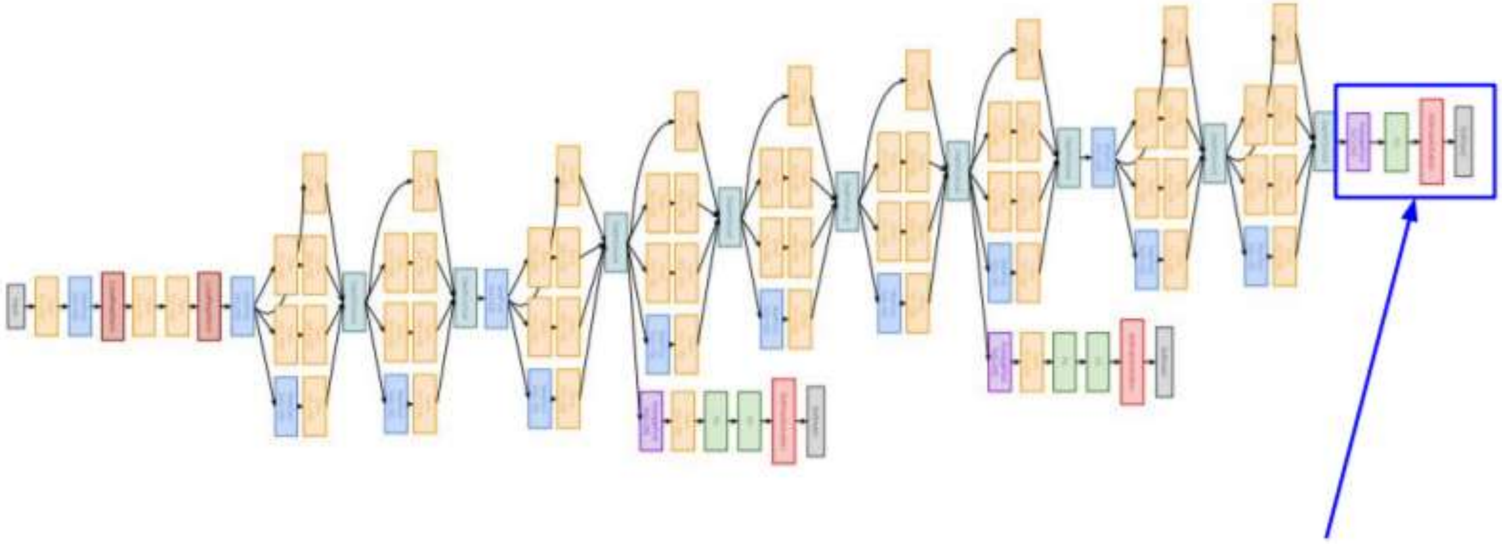
- 减少计算代价的同时能调整深度

GoogLeNet



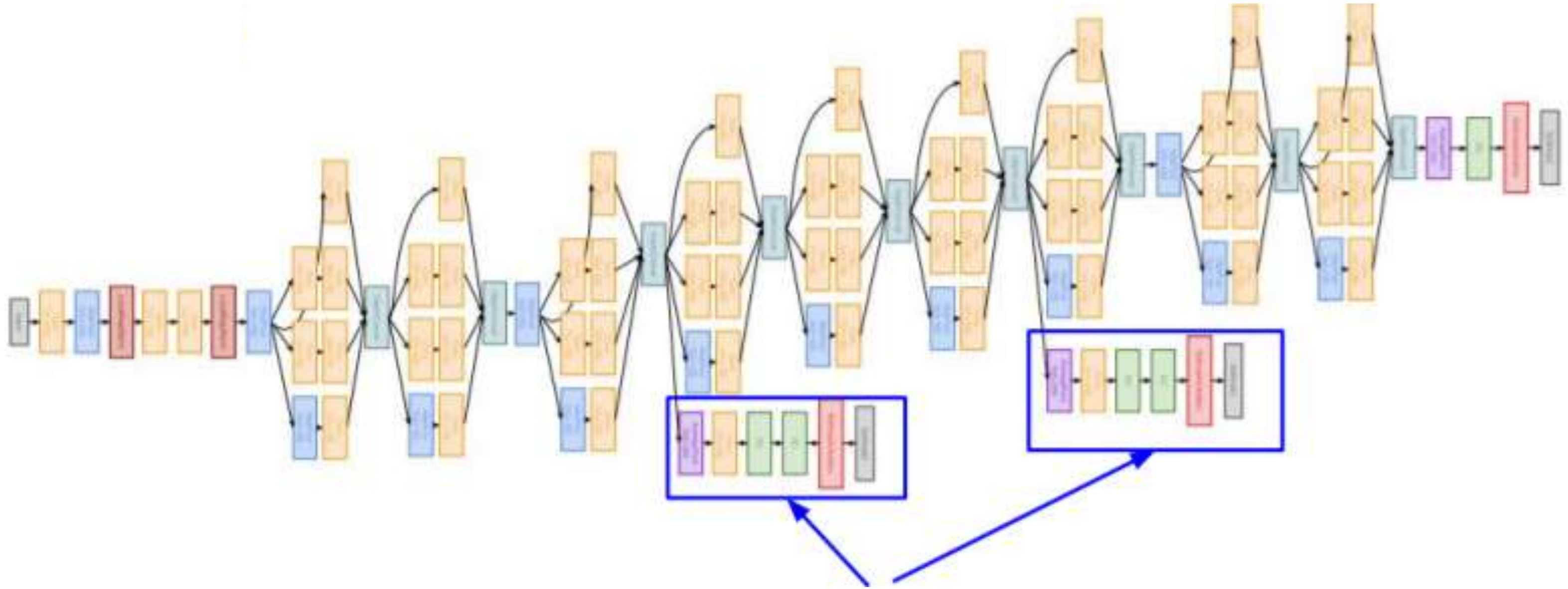
堆叠的Inception模块

GoogLeNet



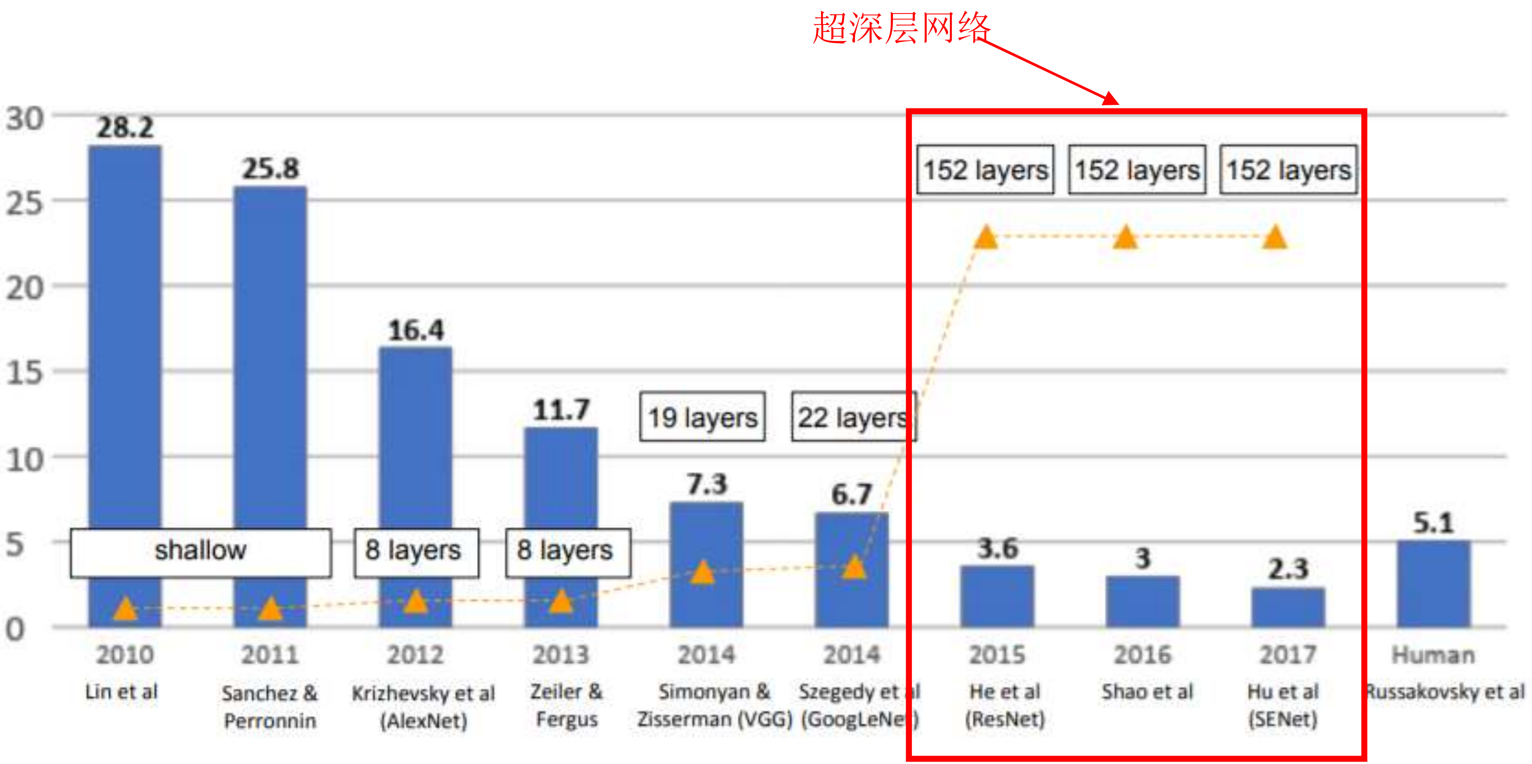
分类输出

GoogLeNet



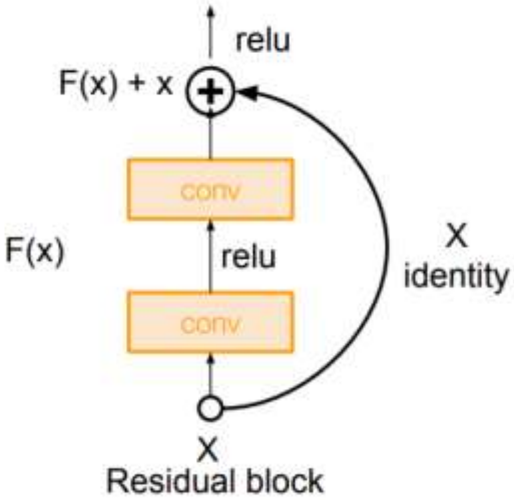
辅助分类输出，为底层注入梯度

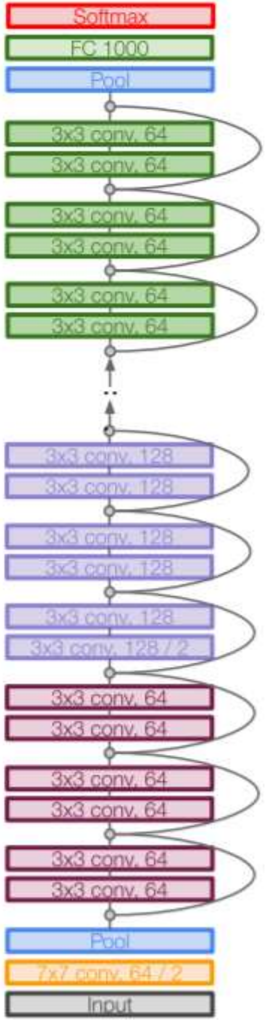
ImageNet 分类挑战(ILSVRC)



ResNet

- 152层
- 引入残差模块
- 横扫了当年的分类以及检测比赛



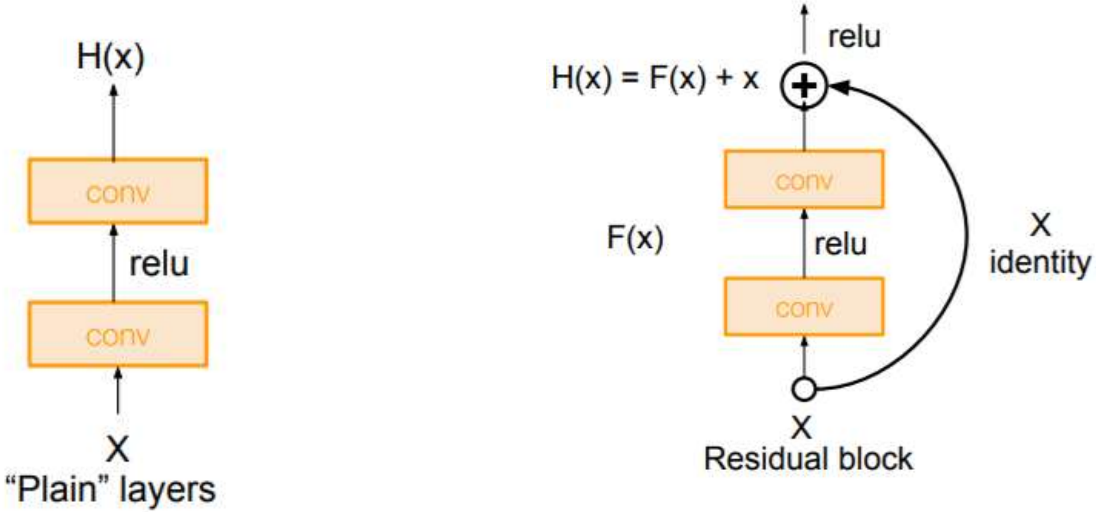


模型退化

- 56层的网络在训练和测试都表现得较差
- 并非是由于过拟合导致的
- 深度网络更难训练

残差模块

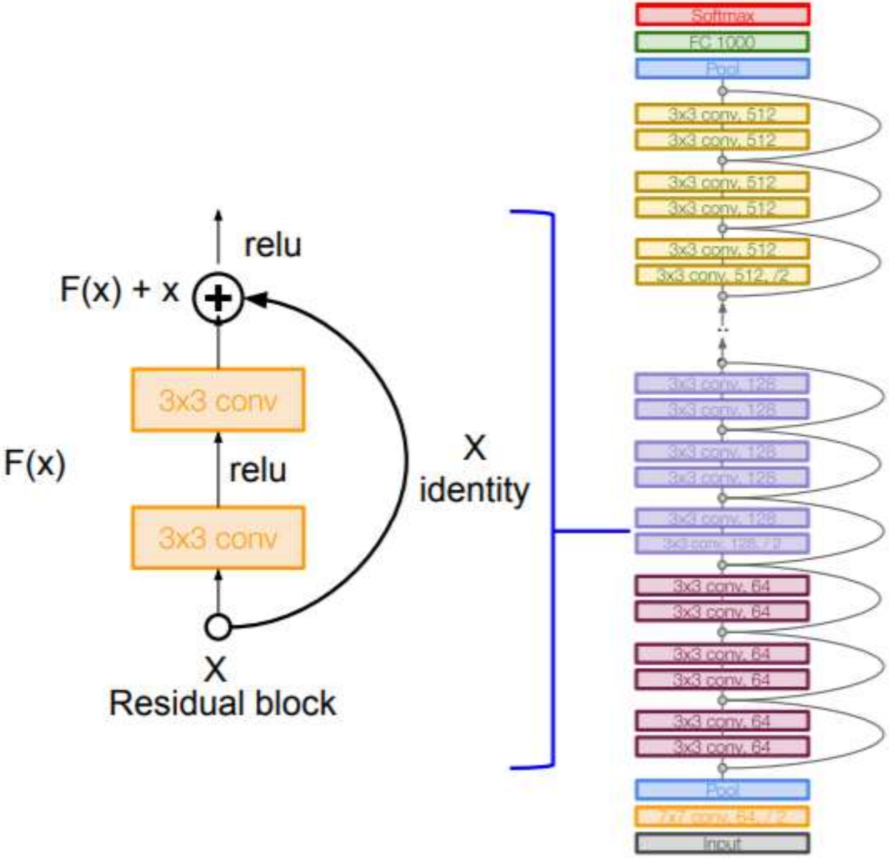
尝试学习输入和输出之间的差



H(x) = x if F(x) = 0

ResNet

- 残差模块的不断堆叠
- 初始位置加入卷积层
- 输出处仅采用单层的FC完成对输出的适配



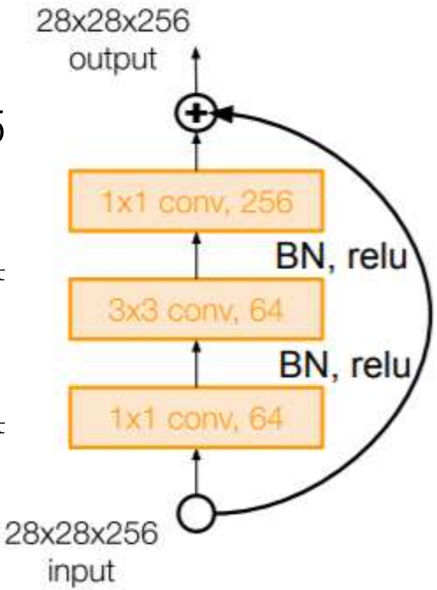
ResNet

- 针对于深层的网络(ResNet50+)引入1x1卷积核帮助提升效率

28x28x256

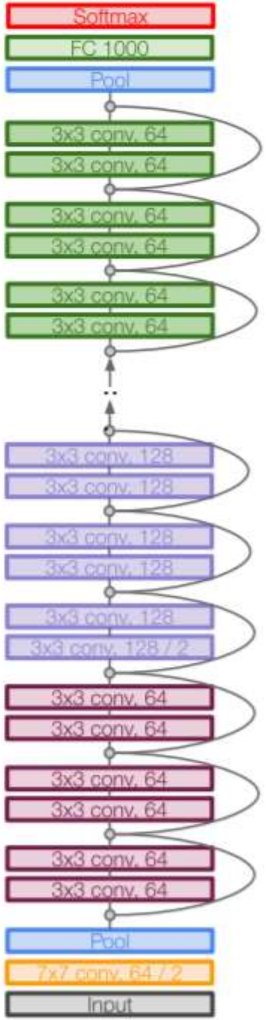
28x28x64

28x28x64

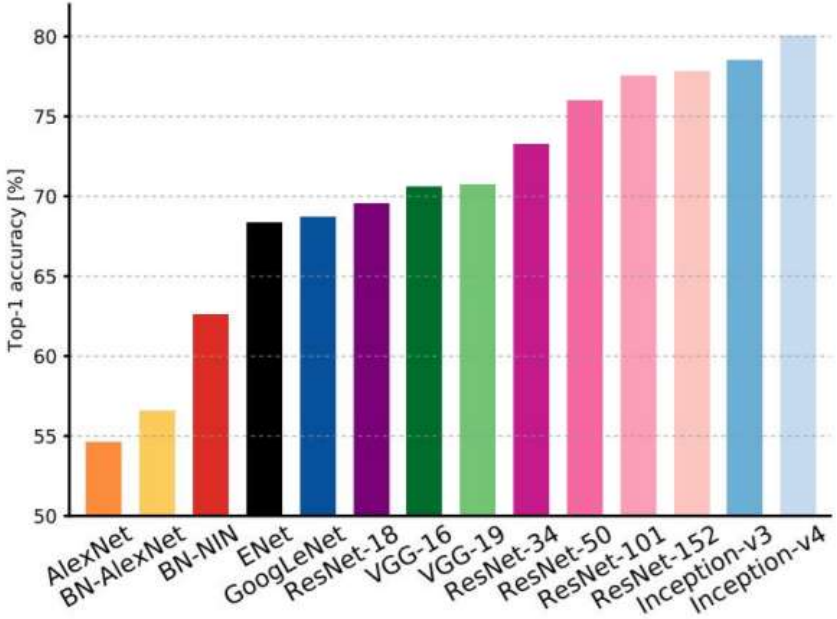


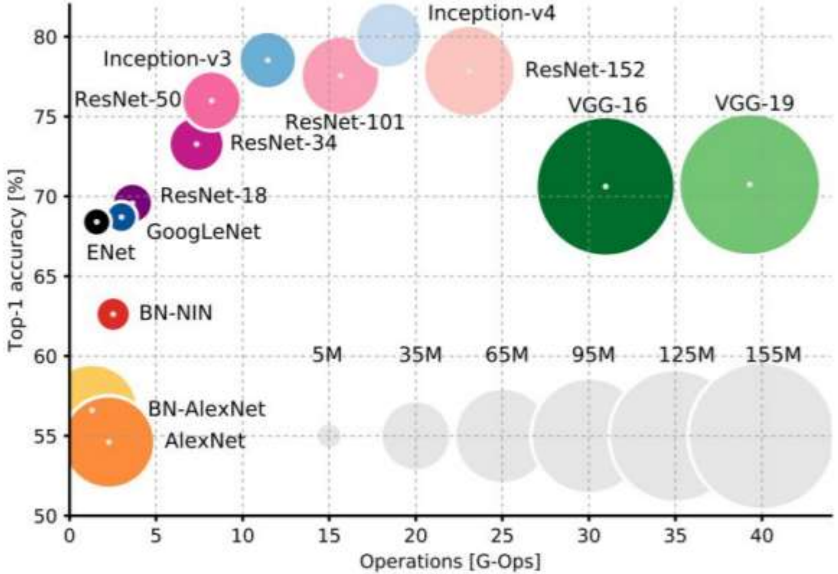
ResNet训练

- 每层卷积后引入BN
- 运用Xavier初始化
- SGD + Momentum (0.9)
- 学习率为0.1每次测试集正确率不升后降为1/10
- Batch size 为256
- Weight decay为1e-5
- 不引入drop out



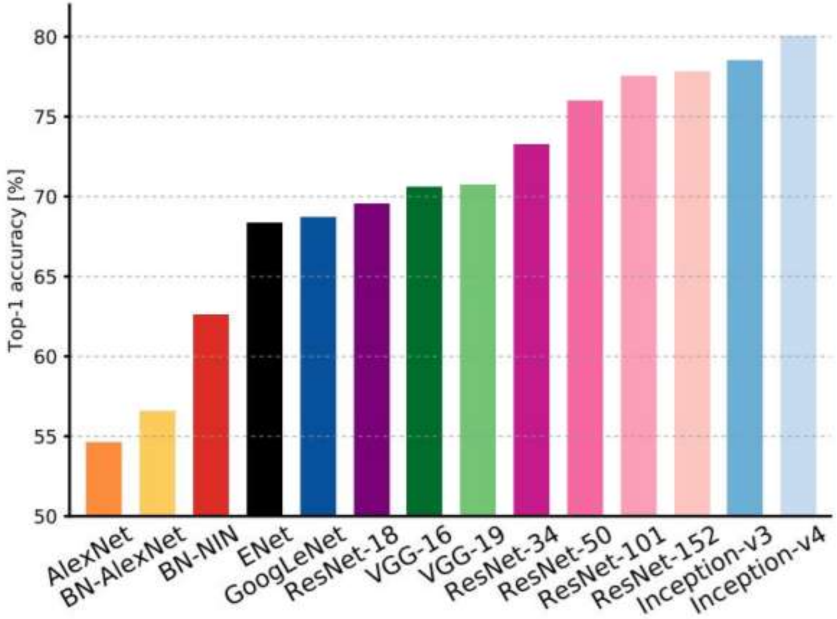
复杂性比较

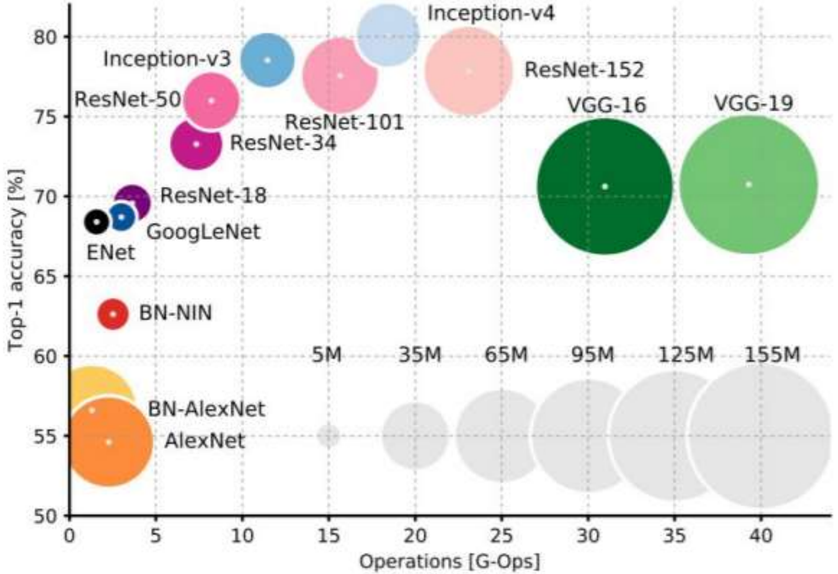




- Inception-v4:ResNet+Inception
- VGG:最多的参量，最多的操作

复杂性比较



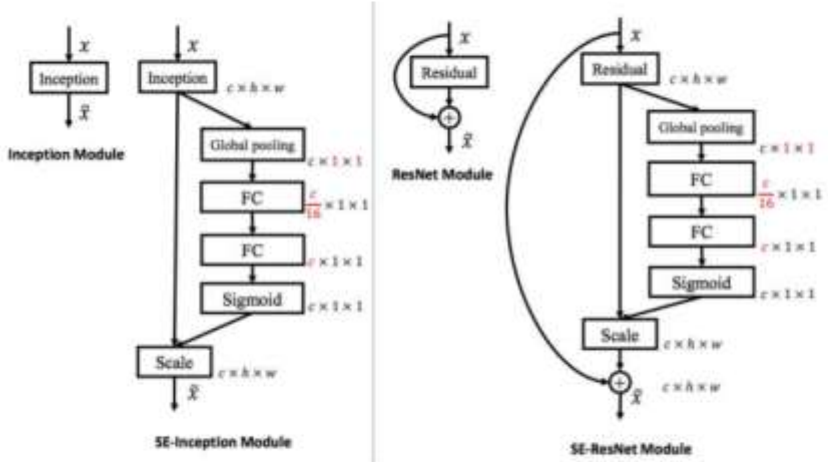


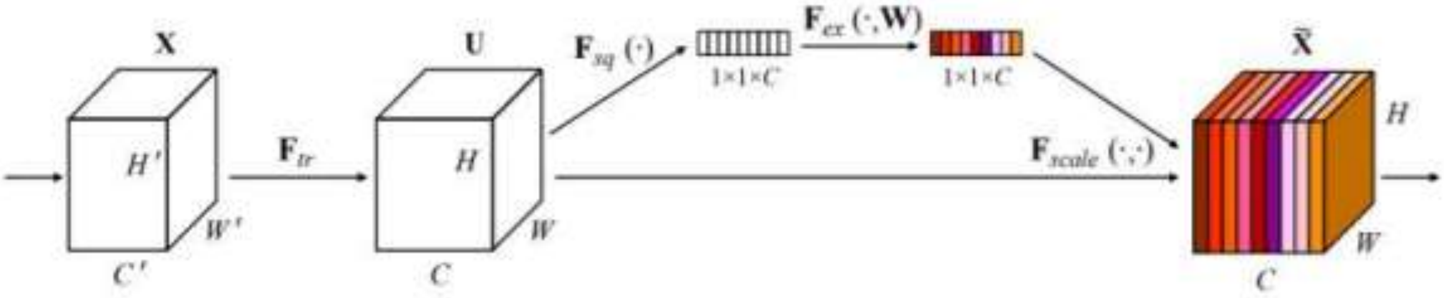
- GoogLeNet：最有效率的模型
- AlexNet：计算量小，但内存占用多，效果低

针对于ResNet的改进

Squeeze-and-Excitation Networks (SENet)

- 增加了模型再校准的模块，对特征图权重进行自适应性调整
- 全局信息以及2个全连接层被用来确定特征图的权重

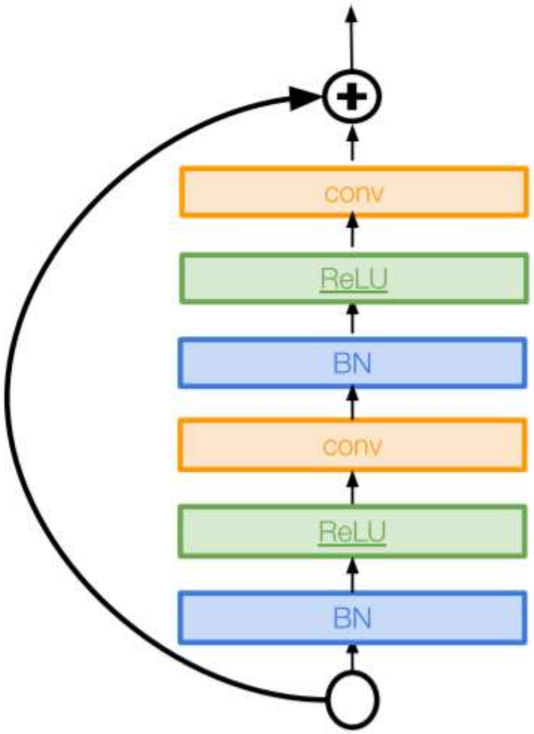




针对于ResNet的改进

Identity Mappings in Deep Residual Networks

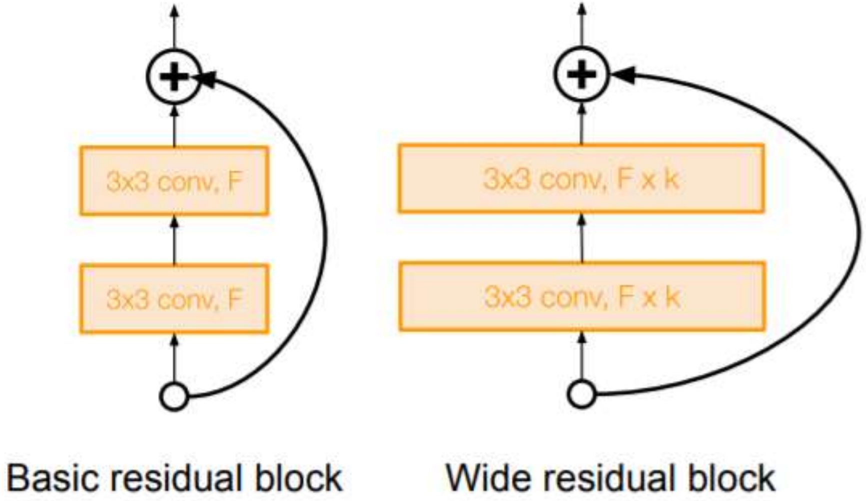
- 提升了ResNet模块的设计
- 创建一个更直接的路径在整个网络中传播信息
- 获得了更好的效果



针对于ResNet的改进

Wide Residual Networks

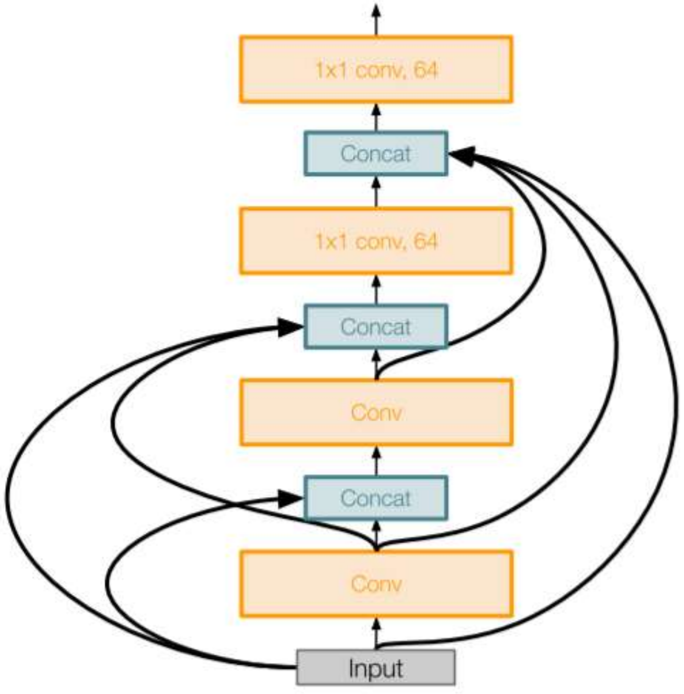
- 认为重要的因素是残差而非深度
- 用更宽的残差块(Fxk 替代 F)
- 50层的更宽ResNet较ResNet-152效果更好
- 提升宽度而非深度让计算更有效率



其他的想法

Densely Connected Convolutional Networks (DenseNet)

- Dense块中的每一层都和之后的层有连接
- 减轻梯度消失，加强特征传播，鼓励特征重用
- 50层效果好于ResNet152



翻译：梁嘉豪