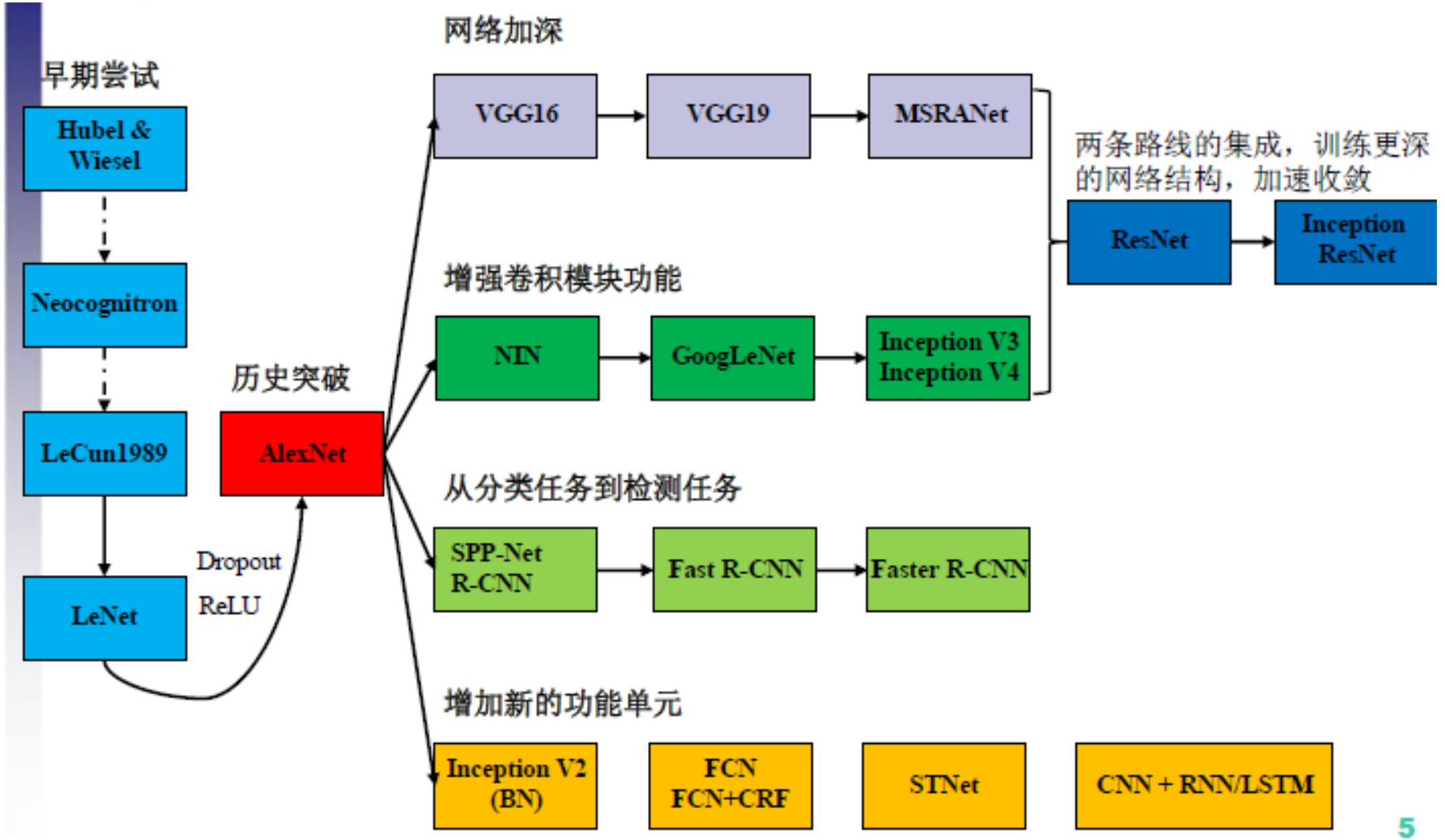
大型卷积神经网络

CNN演化


早期尝试
Hubel & Wiesel
Neocognitron
LeCun1989
LeNet
Dropout
ReLU
历史突破
AlexNet
网络加深
VGG16
VGG19
MSRANet
增强卷积模块功能
NIN
GoogLeNet
Inception V3
Inception V4
两条路线的集成，训练更深的网络结构，加速收敛
ResNet
Inception ResNet
从分类任务到检测任务
SPP-Net
R-CNN
Fast R-CNN
Faster R-CNN
增加新的功能单元
Inception V2 (BN)
FCN
FCN+CRF
STNet
CNN + RNN/LSTM

ImageNet

ImageNet 挑战ISLVRC （*ImageNet Large Scale Visual Recognition Challenge*）

- 自2010年以来，每年度ImageNet大规模视觉识别挑战赛（ILSVRC），研究团队在给定的数据集上评估其算法，并在几项视觉识别任务中争夺更高的准确性。

- ISLVRC2012

- 训练集：1,281,167张图片+标签
  验证集：50,000张图片+标签
  测试集：100,000张图片
- http://www.image-net.org/

English foxhound

An English breed slightly larger than the American foxhounds originally used to hunt in packs

454 pictures | 37.57% Popularity Percentile | Wordnet IDs

- survivor (0)
- range animal (0)
- creepy-crawly (0)
- domestic animal, domesticated animal (213)
  - domestic cat, house cat, Felis domesticus, Felis catus (18)
  - dog, domestic dog, Canis familiaris (189)
    - pooch, doggie, doggy, barker, bow-wow (0)
    - hunting dog (101)
      - sporting dog, gun dog (28)
      - dachshund, dachsie, badger dog (1)
      - terrier (37)
      - courser (0)
      - hound, hound dog (29)
        - Plott hound (0)
        - wolfhound (2)
        - Scottish deerhound, deerhound (0)
        - coonhound (2)
        - foxhound (3)
          - Walker hound, Walker foxhound (0)
          - American foxhound (0)
          - English foxhound (0)
        - Weimaraner (0)
        - otterhound, otter hound (0)
        - bloodhound, sleuthhound (0)
        - Norwegian elkhound, elkhound (0)
        - Saluki, gazelle hound (0)
        - Afghan hound, Afghan (0)
        - staghound (0)
        - greyhound (2)
        - beagle (0)
        - harrier (0)
        - basset, basset hound (0)
        - bluetick (0)
        - redbone (0)

Treemap Visualization | Images of the Synset | Downloads

AlexNet

AlexNet由Alex Krizhevsky于2012年提出，夺得2012年ILSVRC比赛的冠军，top5预测的正确率为83.6%，远超第一名。AlexNet采用8层的神经网络，5个卷积层和3个全连接层(3个卷积层后面加了最大池化层)，包含6亿3000万个链接，6000万个 参数和65万个神经元。

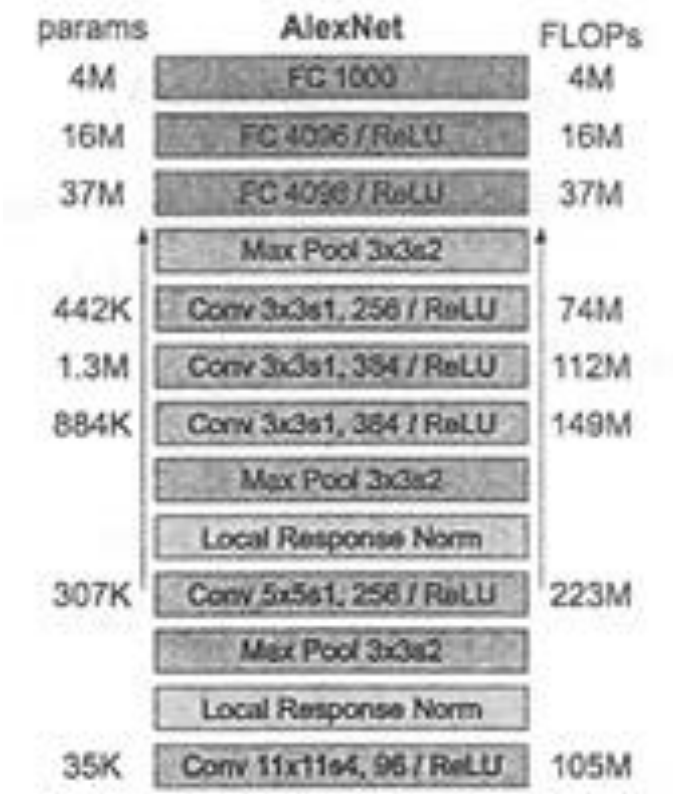
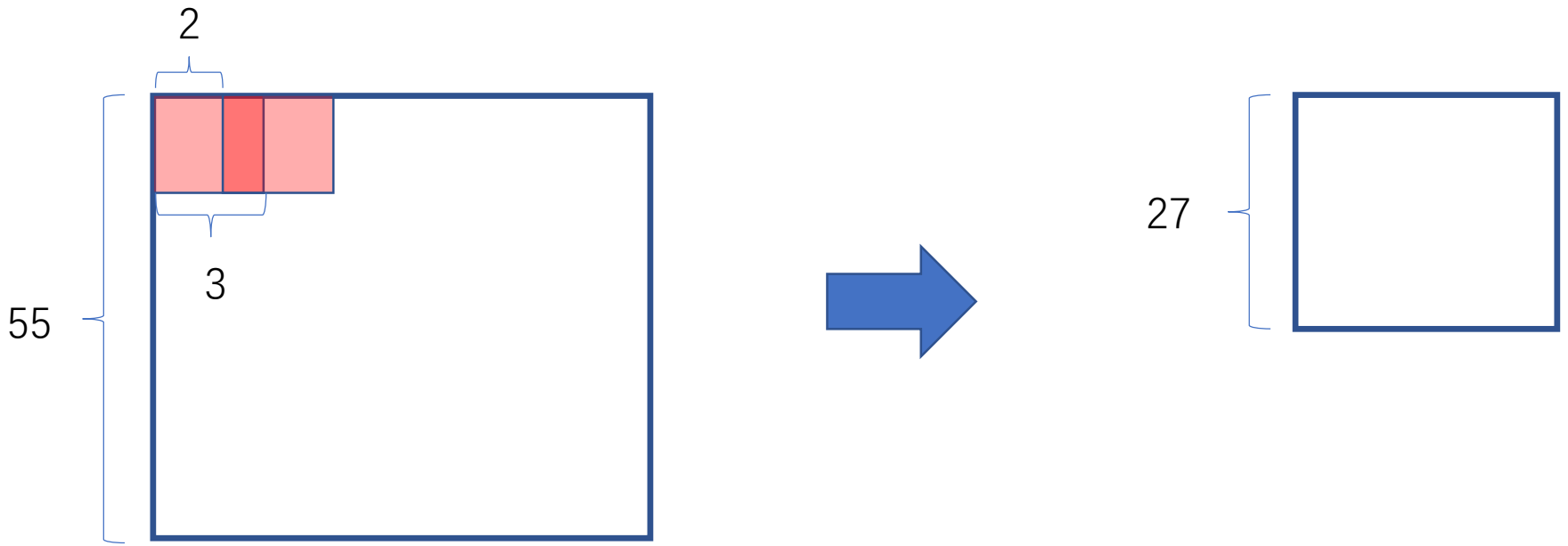


图 6-5　AlexNet 每层的超参数及参数数量

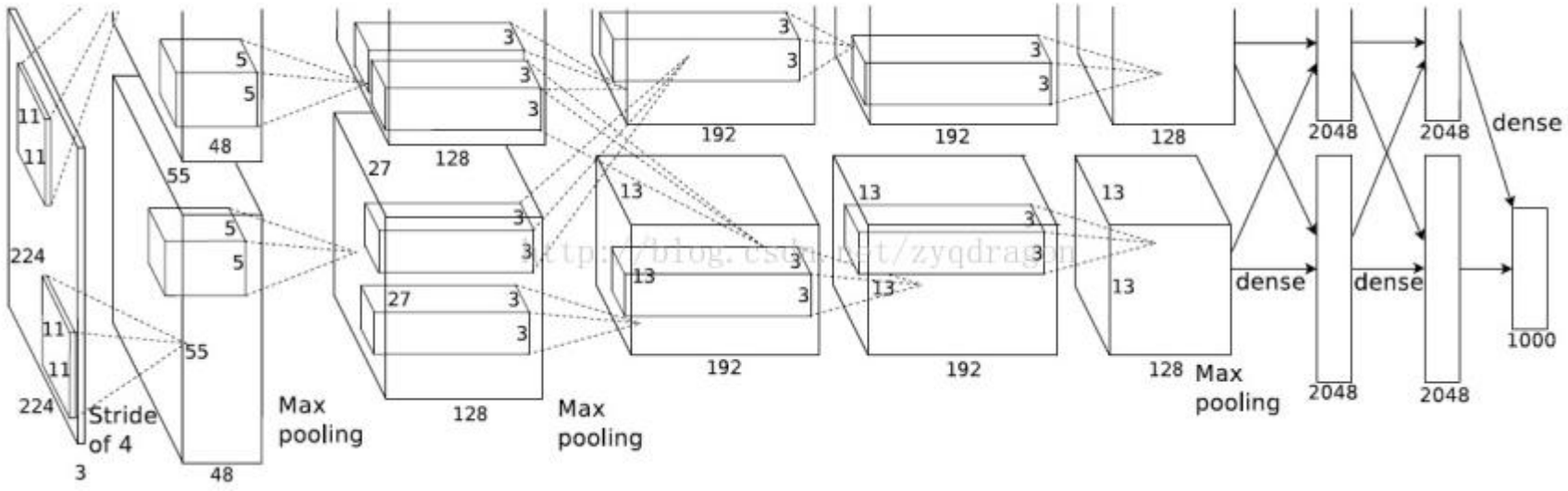
重叠pool池化层

- 这些像素层还需要经过pool运算（池化运算）的处理，池化运算的尺度由预先设定为3*3，运算的步长为2，则池化后的图像的尺寸为：（55-3）/2+1=27。即经过池化处理过的规模为27*27*96.



双GPU并行运行

- 为提高运行速度和提高网络运行规模，作者采用双GPU的设计模式。并且规定GPU只能在特定的层进行通信交流。其实就是每一个GPU负责一半的运算处理。



LRN局部响应归一化

- ReLU本来是不需要对输入进行标准化，但本文发现进行局部标准化能提高性能。

$$b_{x,y}^{i} = a_{x,y}^{i} / (k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a_{x,y}^{j})^{2})$$

Xavier初始化

$f'(0) = 1$

$$y = w_1 x_1 + \cdots + w_{n_i} x_{n_i} + \mathrm{b}$$

$$Var(w_i x_i) = E[w_i]^2 Var(x_i) + E[x_i]^2 Var(w_i) + Var(w_i)\ Var(x_i)$$

$$Var(w_i x_i) = Var(w_i)\ Var(x_i)$$

$$Var(y) = n_i Var(w_i)\ Var(x_i)$$

$$Var(w_i) = \frac{1}{n_i}$$

$$\forall i, \quad n_i Var[W^i] = 1$$

$$\forall i, \quad n_{i+1} Var[W^i] = 1$$

Xavier初始化

$$\forall i, \quad Var[W^i] = \frac{2}{n_i + n_{i+1}}$$

$$Var = \frac{(b-a)^2}{12}$$

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}} \right]$$

数据增益

- 增强图片数据集最简单和最常用的方法是在不改变图片核心元素（即不改变图片的分类）的前提下对图片进行一定的变换，比如在垂直和水平方向进行一定的唯一，翻转等。
- AlexNet用到的第一种数据增益的方法：是原图片大小为256*256中随机的提取224*224的图片，以及他们水平方向的映像。

Dropout

- 结合多个模型的预测值是减少错误的有效方法，但是对于训练时间用好几天的大型神经网络太耗费时间。Dropout是有效的模型集成学习方法，具有0.5的概率讲隐藏神经元设置输出为0。运用了这种机制的神经元不会干扰前向传递也不影响后续操作。因此当有输入的时候，神经网络采样不同的结构，但是这些结构都共享一个权重。这就减少了神经元适应的复杂性。测试时，用0.5的概率随机激活神经元。dropout减少了过拟合，也使收敛迭代次数增加一倍。

实验结果

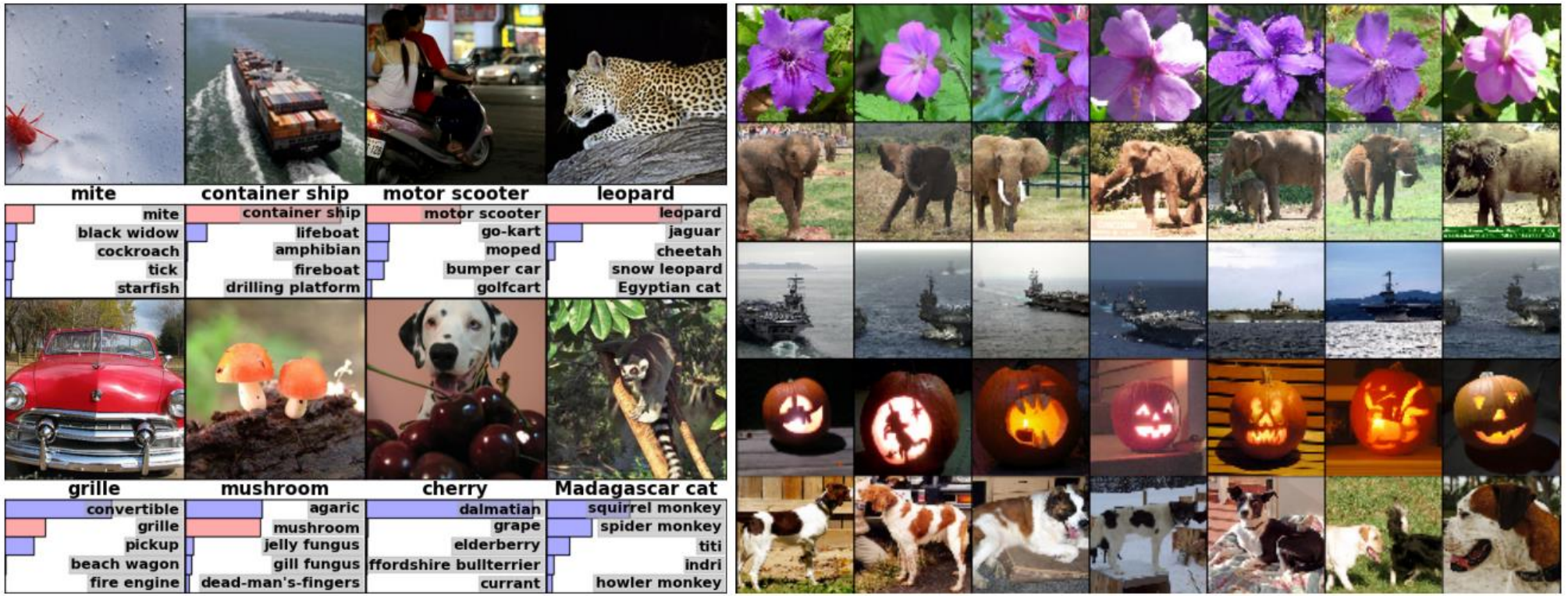
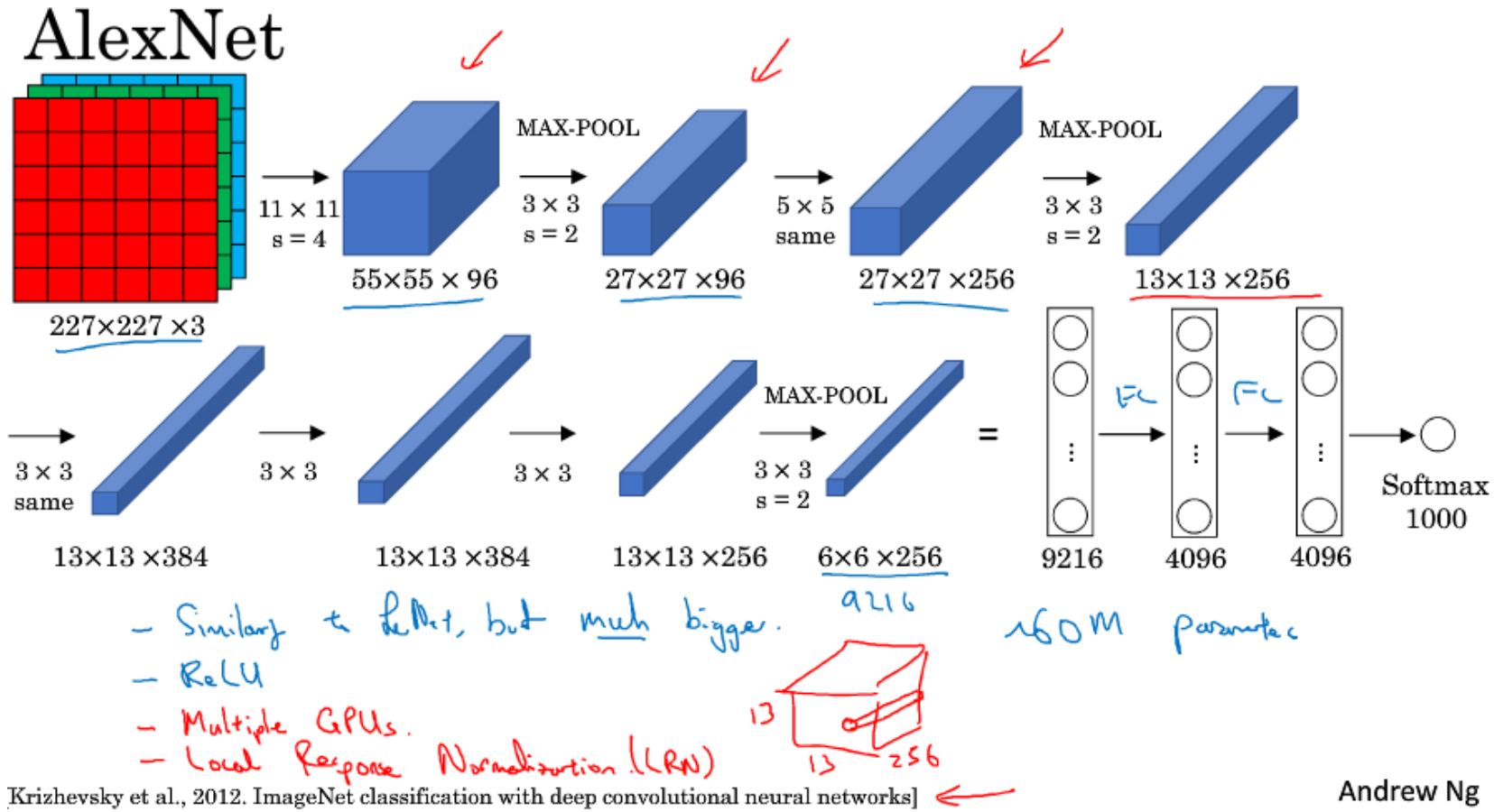


Figure 4: **(Left)** Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). **(Right)** Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

总体架构

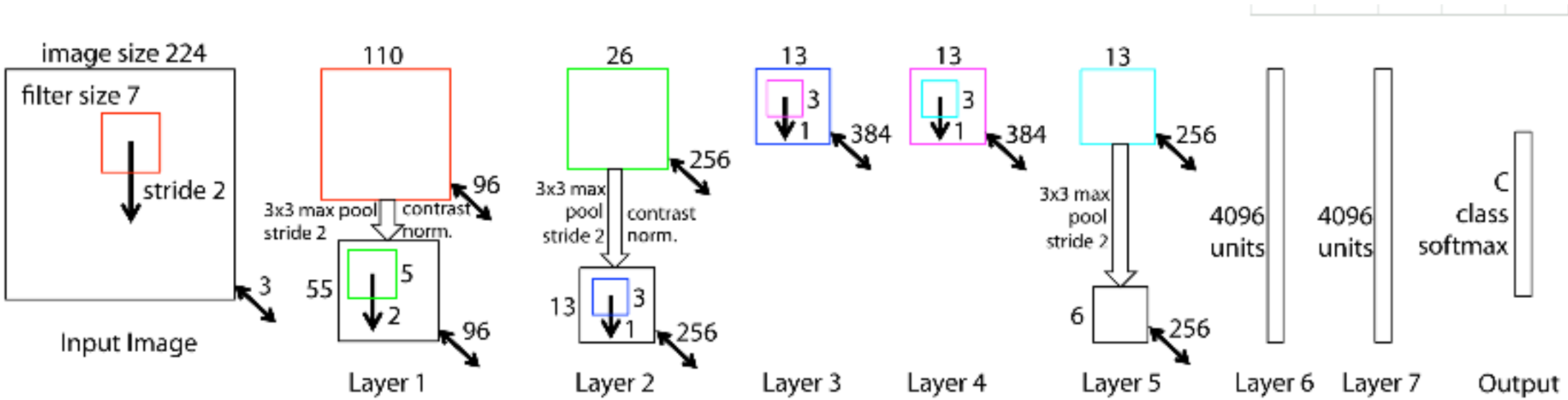


Andrew Ng

Keras实现

```
def AlexNet():

    model = Sequential()
    model.add(Conv2D(96,(11,11),strides=(4,4),input_shape=
(227,227,3),padding='valid',activation='relu',kernel_initializer='uniform'))
    model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))
    model.add(Conv2D(256,(5,5),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))
    model.add(Conv2D(384,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(Conv2D(384,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(Conv2D(256,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))
    model.add(Flatten())
    model.add(Dense(4096,activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(4096,activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1000,activation='softmax'))
    return model
```

2013年冠军ZFnet



Keras实现

```
def ZF_Net():
    model = Sequential()
    model.add(Conv2D(96,(7,7),strides=(2,2),input_shape=
(224,224,3),padding='valid',activation='relu',kernel_initializer='uniform'))
    model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))
    model.add(Conv2D(256,(5,5),strides=(2,2),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))
    model.add(Conv2D(384,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(Conv2D(384,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(Conv2D(256,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(MaxPooling2D(pool_size=(3,3),strides=(2,2)))
    model.add(Flatten())
    model.add(Dense(4096,activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(4096,activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1000,activation='softmax'))
    return model
```

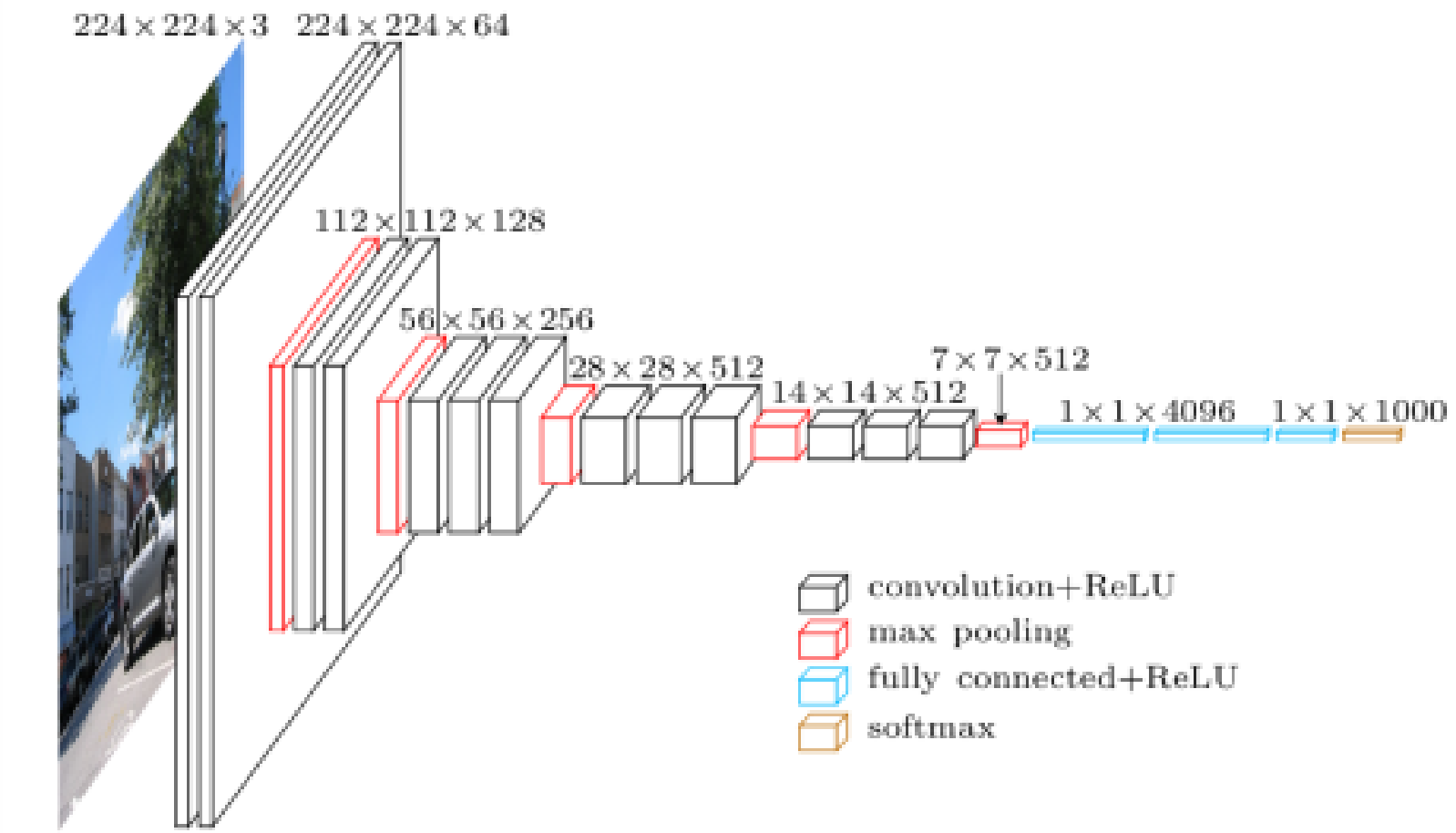
VGG网络

- VGG卷积神经网络是牛津大学在2014年提出来的模型。当这个模型被提出时，由于它的简洁性和实用性，马上成为了当时最流行的卷积神经网络模型。它在图像分类和目标检测任务中都表现出非常好的结果。在2014年的ILSVRC比赛中，VGG 在Top-5中取得了92.3%的正确率。

VGG网络体系

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					

VGG16网络结构


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

为什么3个3x3的卷积可以代替7x7的卷积?

- 个3x3的卷积，使用了3个非线性激活函数，增加了非线性表达能力，使得分割平面更具有可分性
- 减少参数个数。对于C个通道的卷积核，7x7含有参数7^2C^2， 3个3x3的参数个数为$3*3^2C^2$，参数大大减少

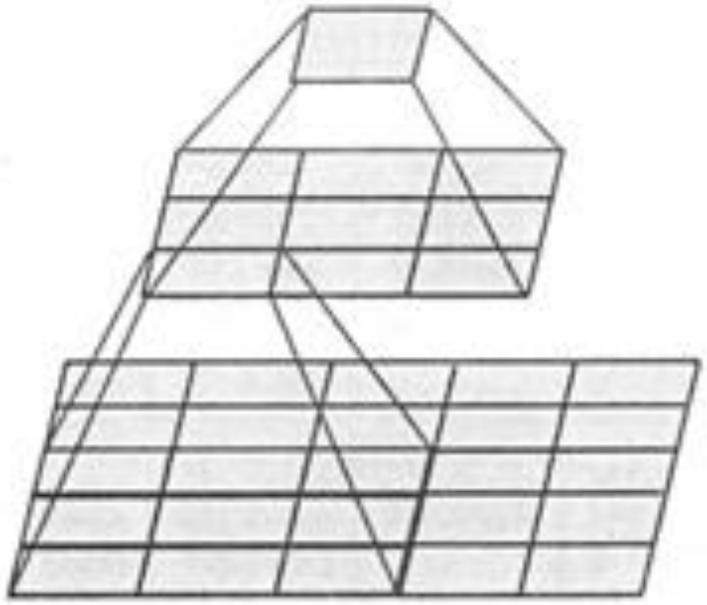
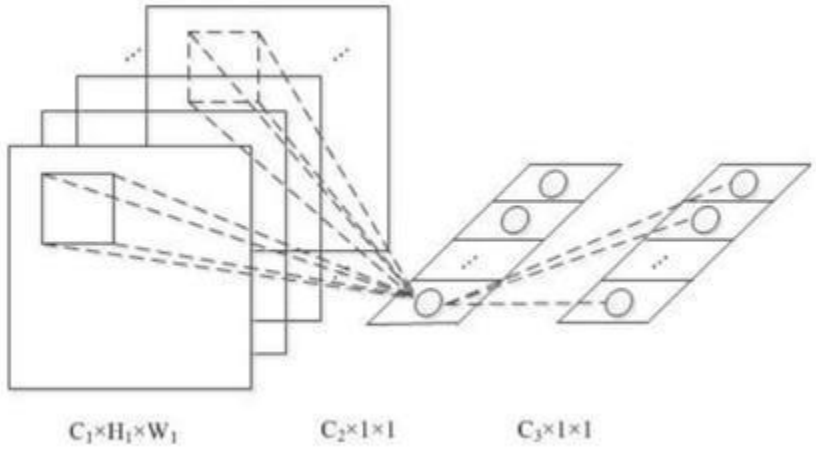


图 6-8　两个串联 3×3 的卷积层功能类似于一个 5×5 的卷积层

1x1卷积核的作用

- 使用1*1的卷积层来增加线性变换，输出的通道数量上并没有发生改变。这里提一下1*1卷积层的其他用法，1*1的卷积层常被用来提炼特征，即多通道的特征组合在一起，凝练成较大通道或者较小通道的输出，而每张图片的大小不变。有时1*1的卷积神经网络还可以用来替代全连接层。



正则化方法

- 增加了对权重的正则化，$5*10^{-4}||W||^2$
- 对FC全连接层进行dropout正则化，dropout ratio = 0.5

虽然模型的参数和深度相比AlexNet 有了很大的增加，但是模型的训练迭代次数却要求更少：a)正则化+小卷积核，b)特定层的预初始化

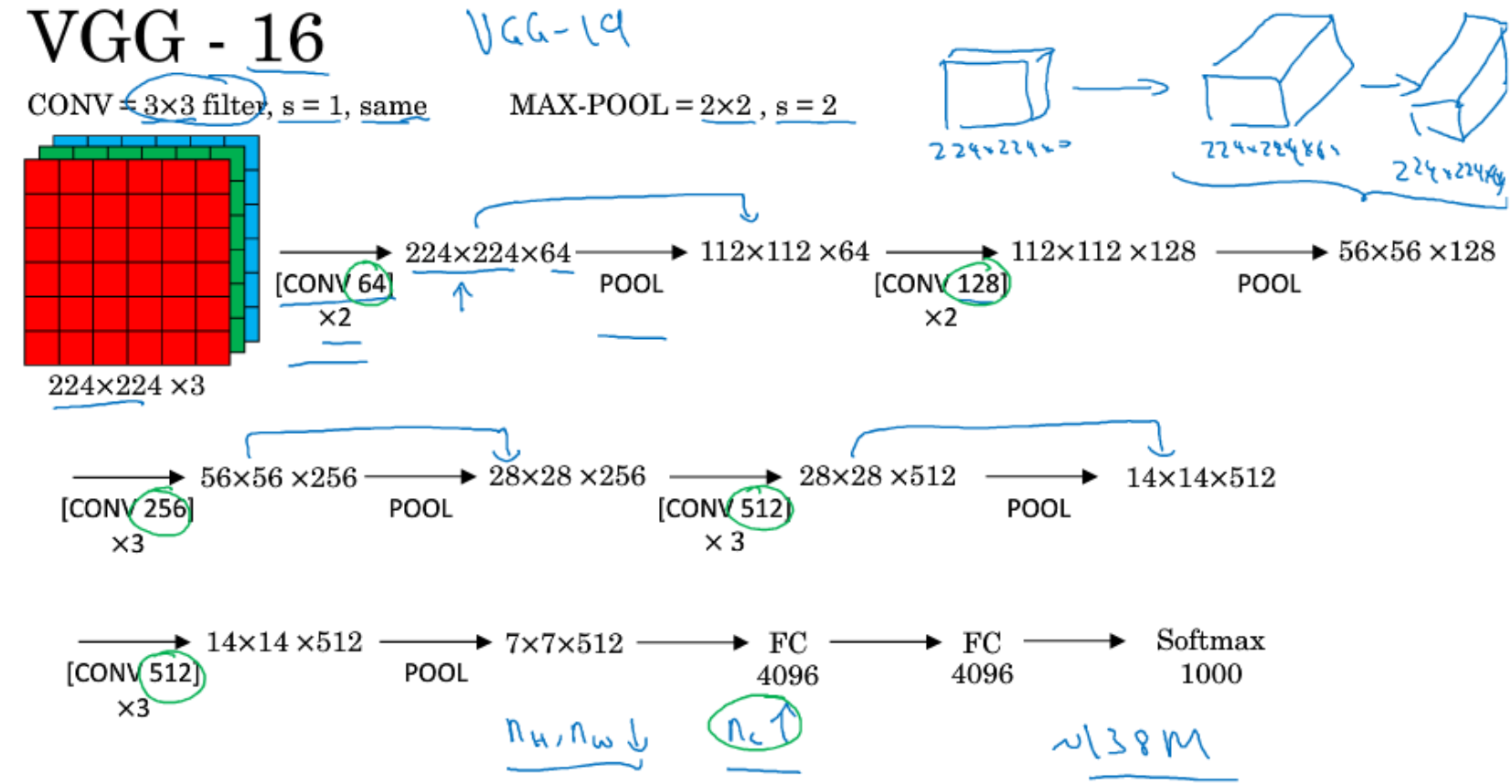
数据生成方式

- 方法1: 在S=224和S=384的尺度下，对图像进行224x224区域随机裁剪
- 方法2: 令S随机的在[S_{min},S_{max}]区间内值，放缩完图像后，再进行随机裁剪（其中S_{min}=256,S_{max}=512)

预测方式

- 作者考虑了两种预测方式:
- 方法1: multi-crop，即对图像进行多样本的随机裁剪，然后通过网络预测每一个样本的结构，最终对所有结果平均。
- 方法2: densely，利用FCN的思想，将原图直接送到网络进行预测，将最后的全连接层改为1x1的卷积，这样最后可以得出一个预测的score map，再对结果求平均。

总体架构



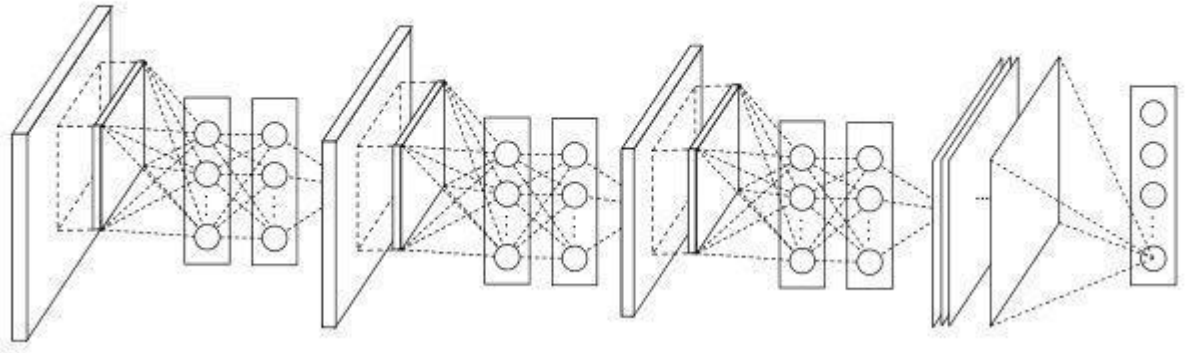
[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

Andrew Ng

Keras实现

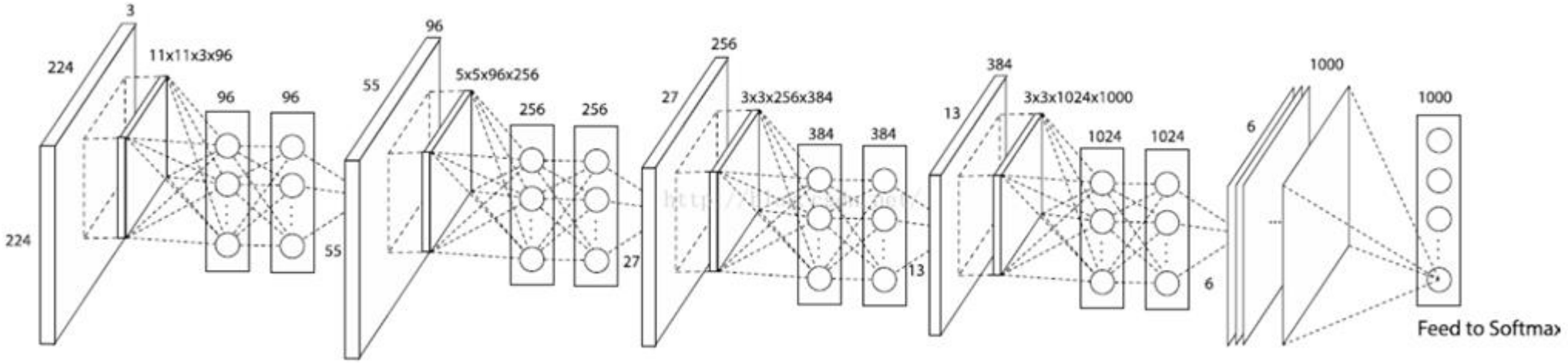
```
def VGG_16():
    model = Sequential()

    model.add(Conv2D(64,(3,3),strides=(1,1),input_shape=
(224,224,3),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(Conv2D(64,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(128,(3,2),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(Conv2D(128,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(256,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(Conv2D(256,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(Conv2D(256,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(512,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(Conv2D(512,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(Conv2D(512,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(512,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(Conv2D(512,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(Conv2D(512,(3,3),strides=(1,1),padding='same',activation='relu',kernel_initializer='uniform'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Flatten())
    model.add(Dense(4096,activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(4096,activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1000,activation='softmax'))

    return model
```

Network in Network

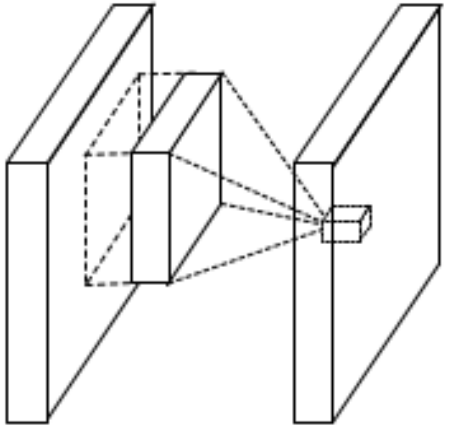


Network In Network 是发表于2014年ICLR的一篇paper。这篇文章采用较少参数就取得了Alexnet的效果，Alexnet参数大小为230M，而Network In Network仅为29M

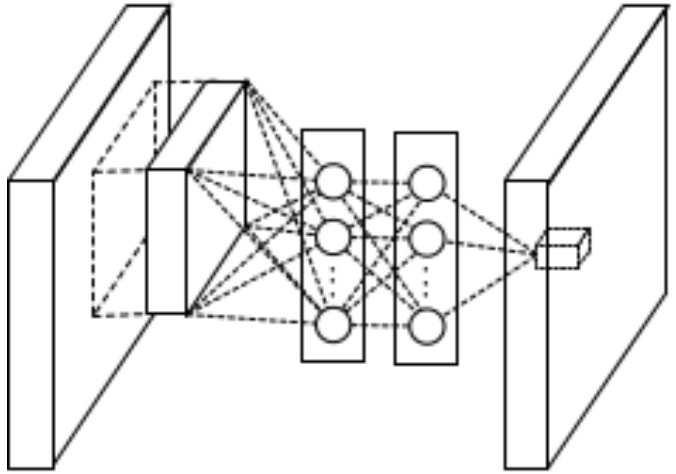
Network in network参数



Network in Network

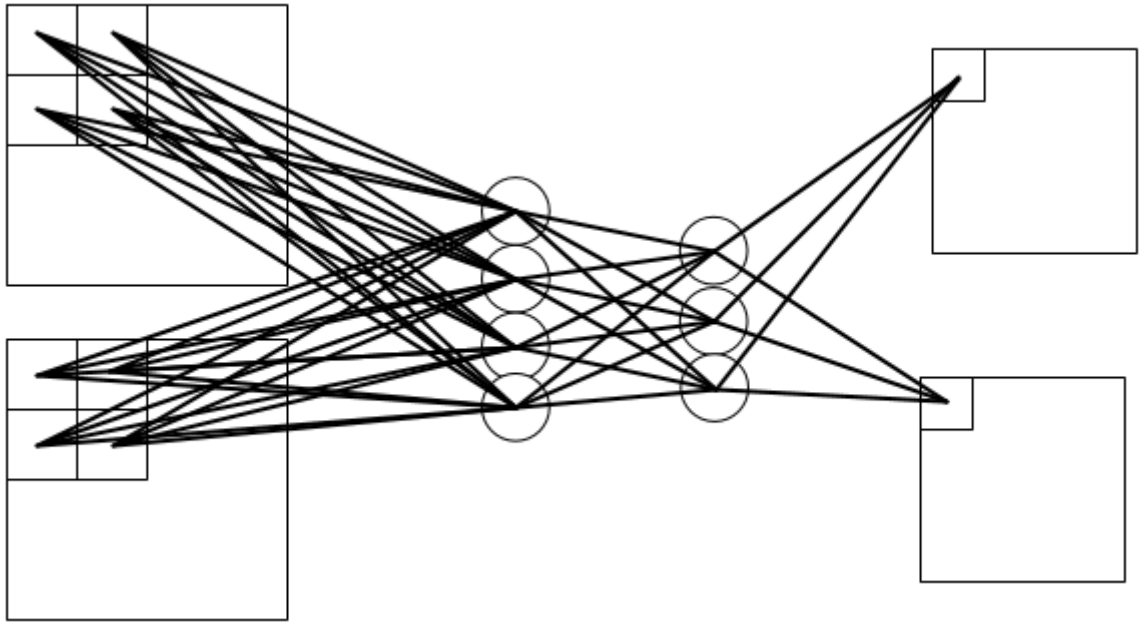


(a) Linear convolution layer



(b) Mlpconv layer

MLPCONV 细节

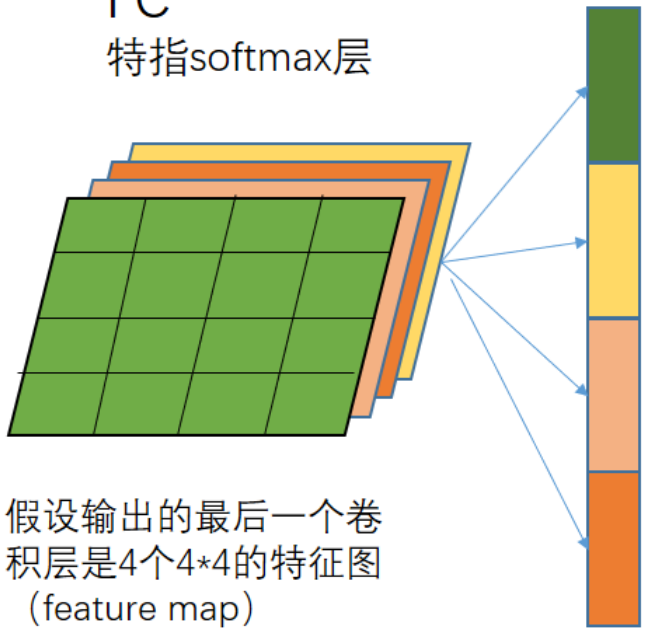


在 卷积神经网络中，无论是输入还是输出，不同的 feature map 之间的卷积核是不相同的;
在mlpconv中，不同的 feature map 之间的开头与能结尾之间的权值不一样，而在 隐含层之间的权值是共享的;

全局平均池化

FC

特指softmax层



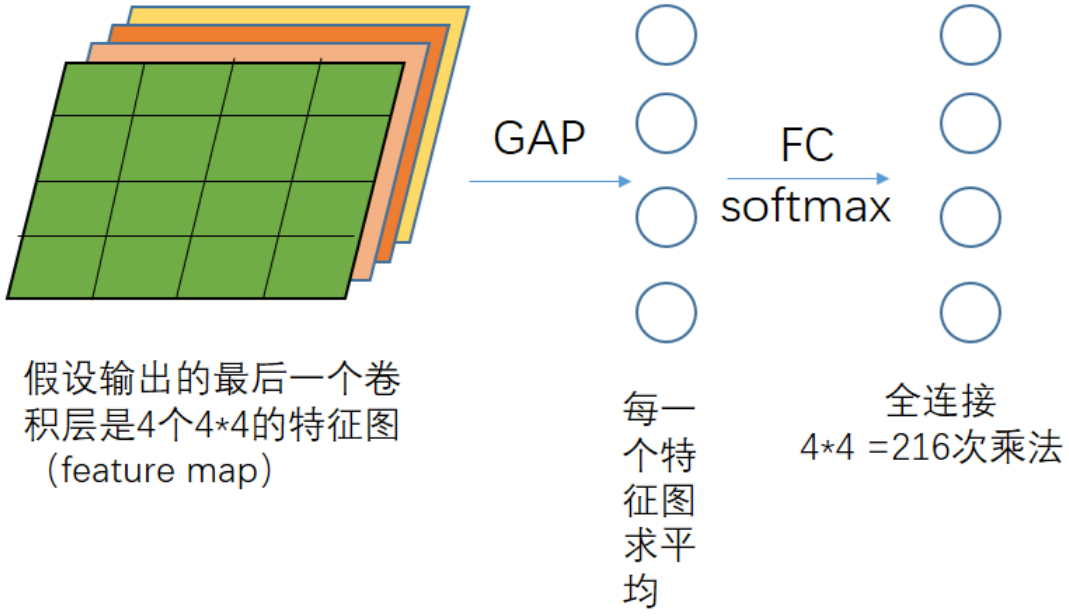
假设输出的最后一个卷积层是4个4*4的特征图（feature map）

全连接

计算量： 4*4*4*4 =216次乘法

GAP

在softmax层加一个GAP



假设输出的最后一个卷积层是4个4*4的特征图（feature map）

每一个特征图求平均

全连接

4*4 =216次乘法

GoogleNet

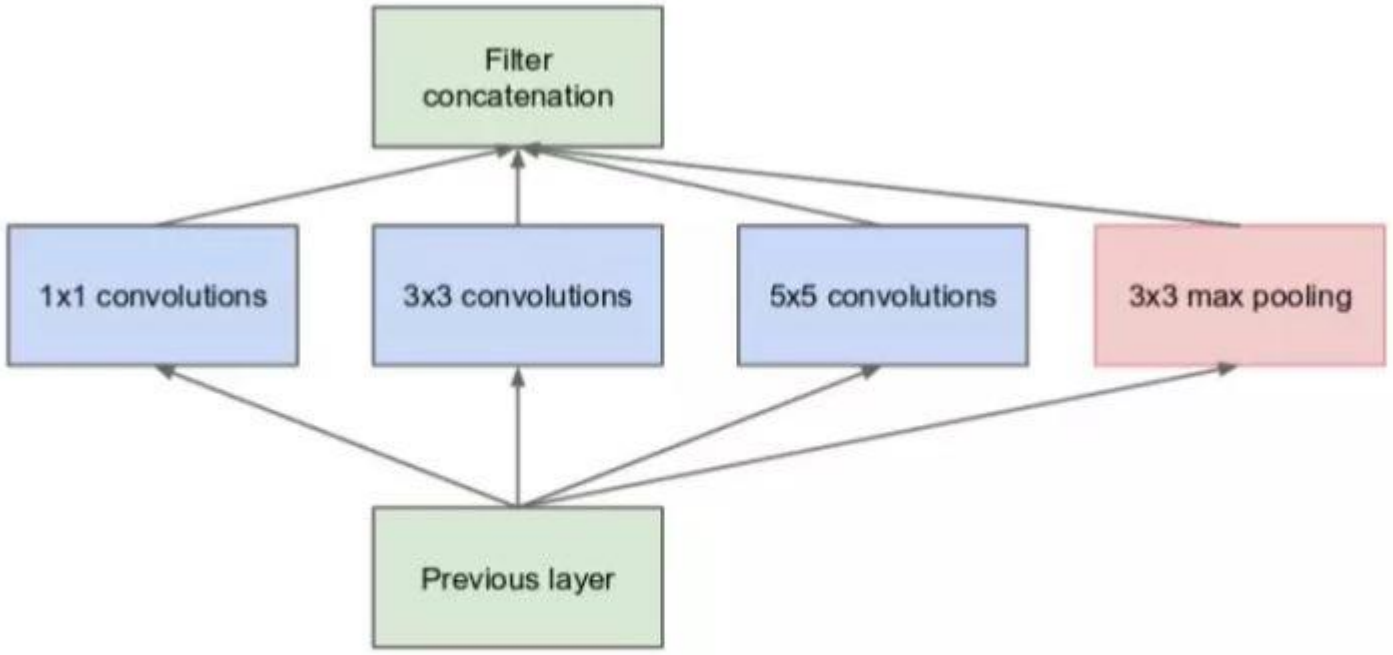
- 在《Going deeper with convolutions》论文中，作者提出一种深度卷积神经网络 Inception，它在 ILSVRC14 中达到了当时最好的分类和检测性能。该架构的主要特点是更好地利用网络内部的计算资源，这通过一个精心制作的设计来实现，该设计允许增加网络的深度和宽度，同时保持计算预算不变。为了优化质量，架构决策基于赫布原则和多尺度处理。作者向 ILSVRC14 提交使用该架构的模型即 GoogLeNet，这是一个 22 层的深度网络，它的质量是在分类和检测领域进行了评估。

图像中突出部分的大小差别很大



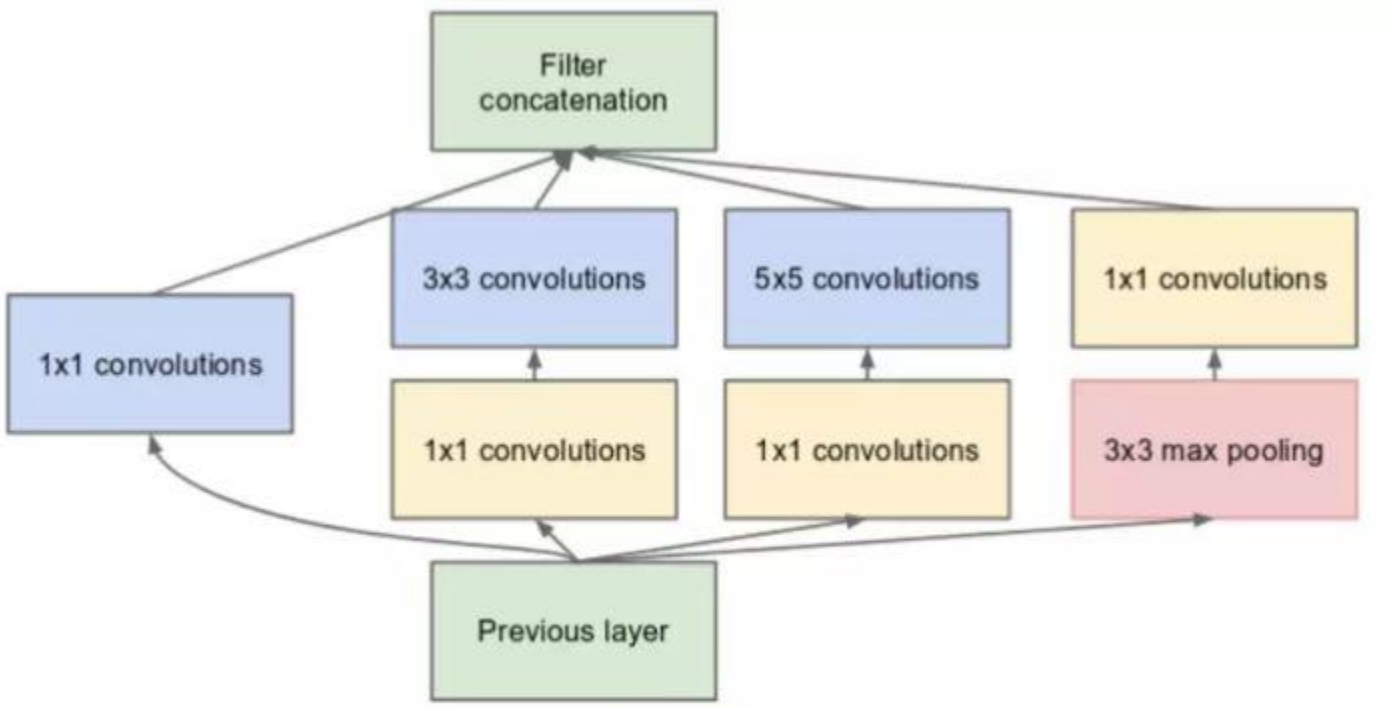
•由于信息位置的巨大差异，为卷积操作选择合适的卷积核大小就比较困难。信息分布更全局性的图像偏好较大的卷积核，信息分布比较局部的图像偏好较小的卷积核。
•非常深的网络更容易过拟合。将梯度更新传输到整个网络是很困难的。
•简单地堆叠较大的卷积层非常消耗计算资源。

多样卷积核设计



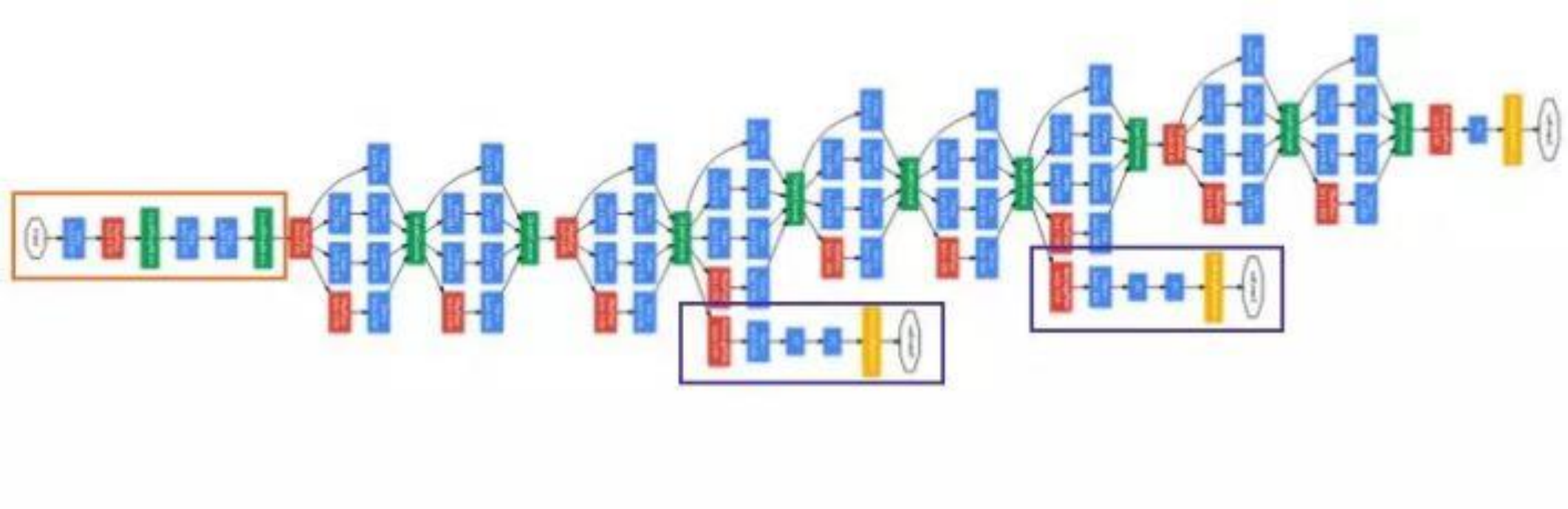
(a) Inception module, naïve version

添加1*1卷积核节约计算资源



(b) Inception module with dimension reductions

网络结构

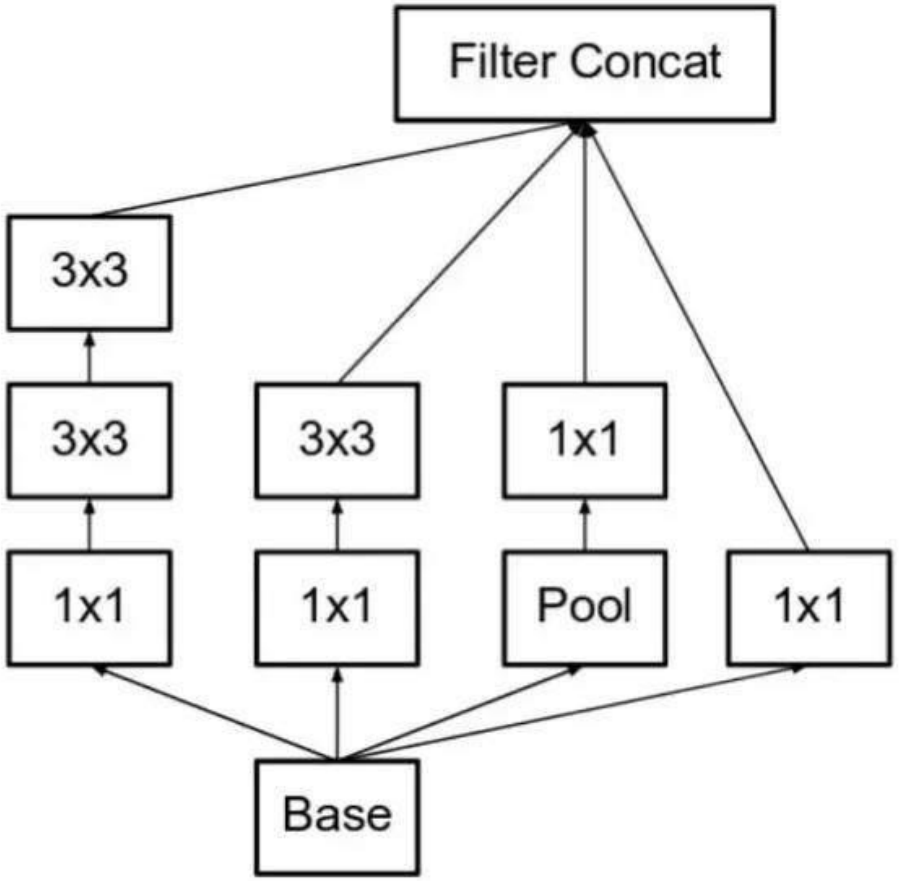


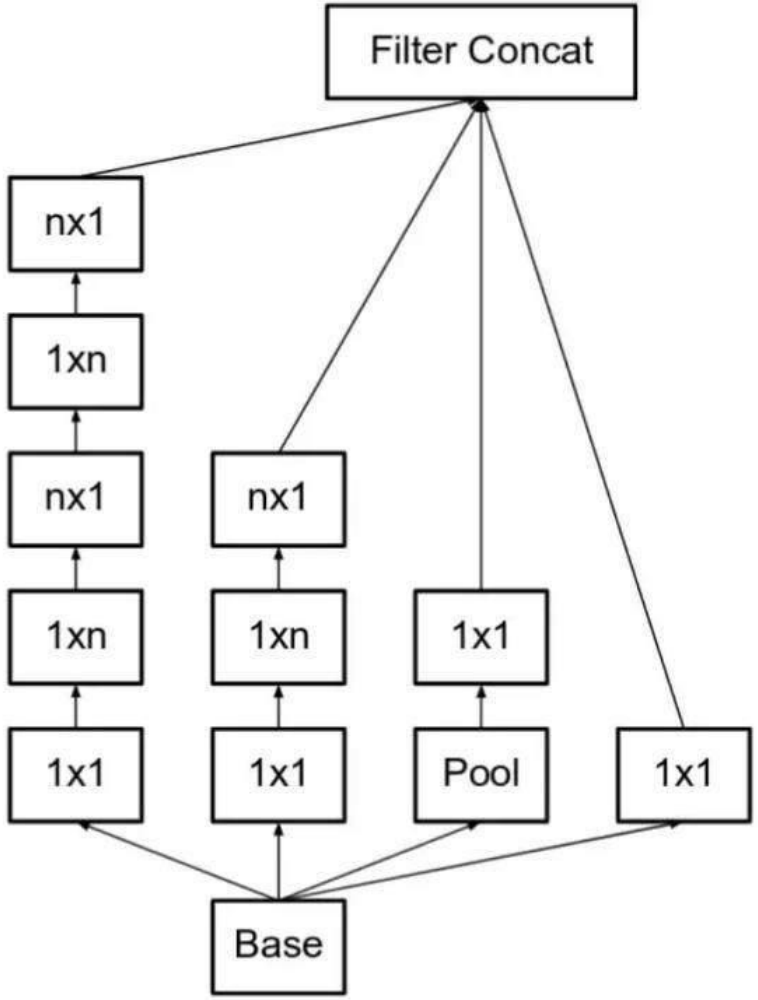
GoogLeNet 有 9 个线性堆叠的 Inception 模块。它有 22 层（包括池化层的话是 27 层）。该模型在最后一个 inception 模块处使用全局平均池化。

Inception v2和v3

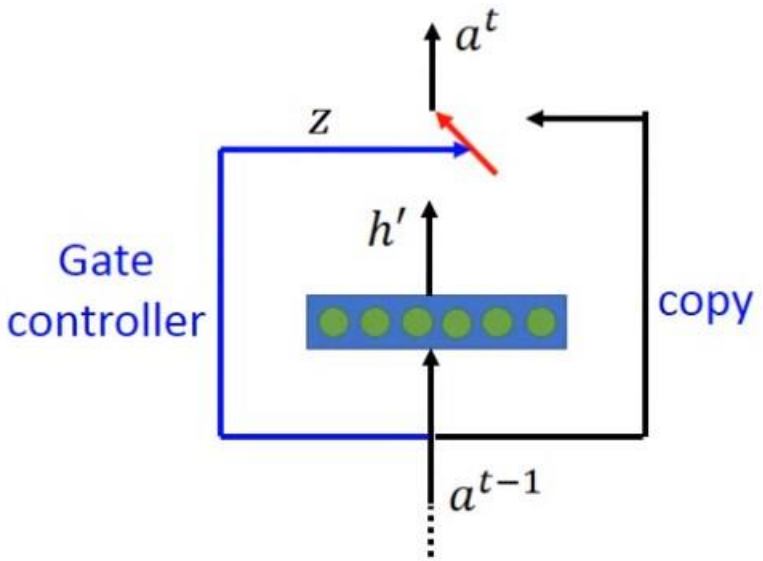
- Inception v2 和 Inception v3 来自同一篇论文《Rethinking the Inception Architecture for Computer Vision》，作者提出了一系列能增加准确度和减少计算复杂度的修正方法。

Inception v2





Highway 网络

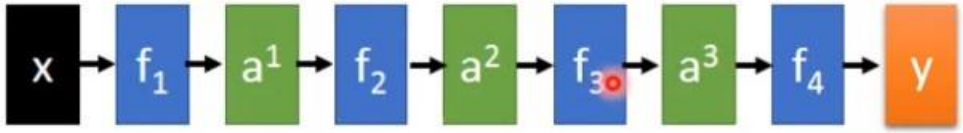


$$y = \sigma(W_y x + b_y) \quad (1)$$

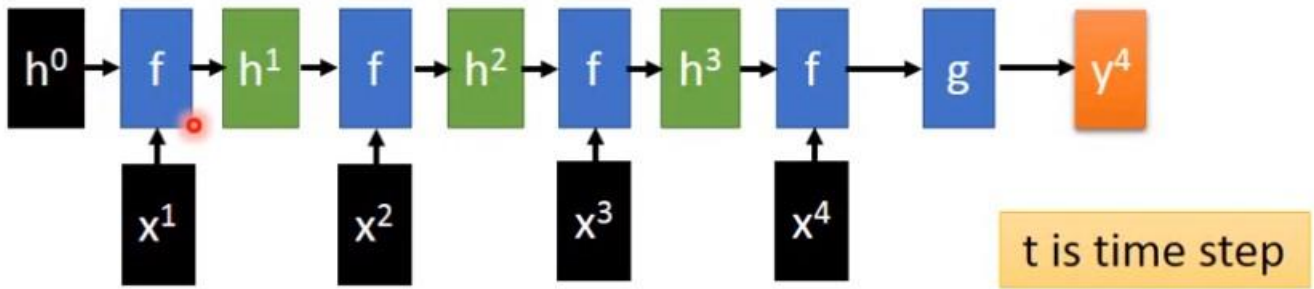
$$g = \sigma(W_g x + b_g) \quad (2)$$

$$h = g \odot y + (1 - g) \odot x \quad (3)$$

前馈网络和循环神经网络(RNN)

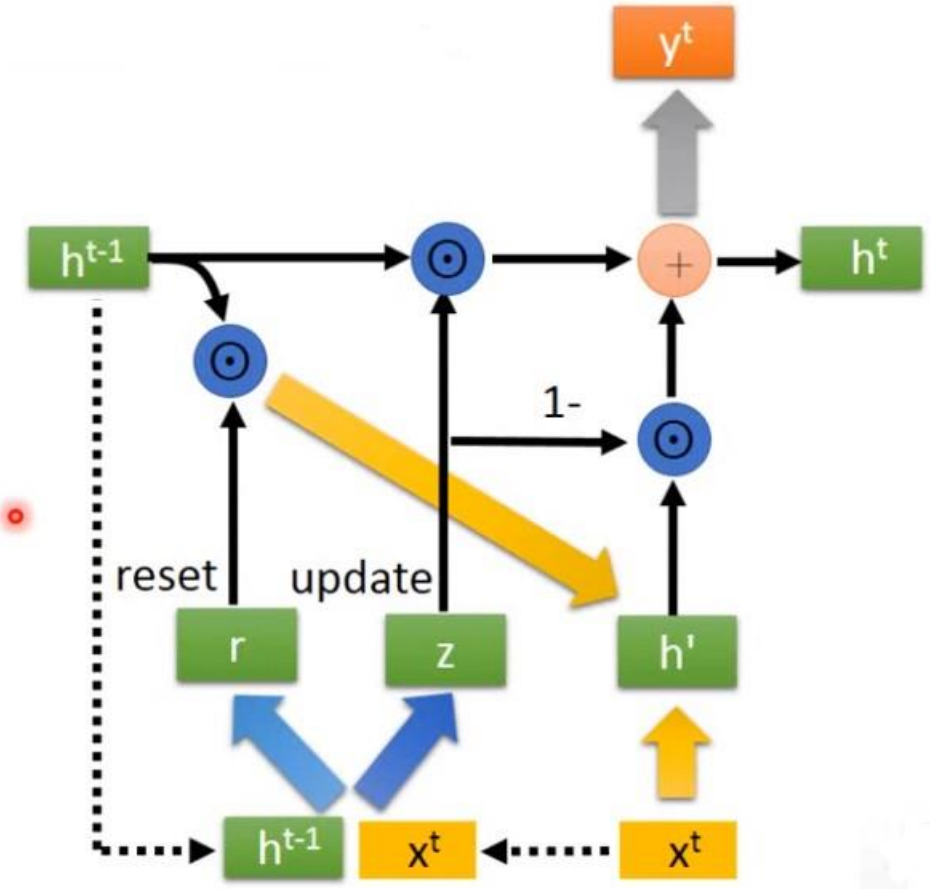


$$a^t = f_l(a^{t-1}) = \sigma(W^t a^{t-1} + b^t)$$ t is layer

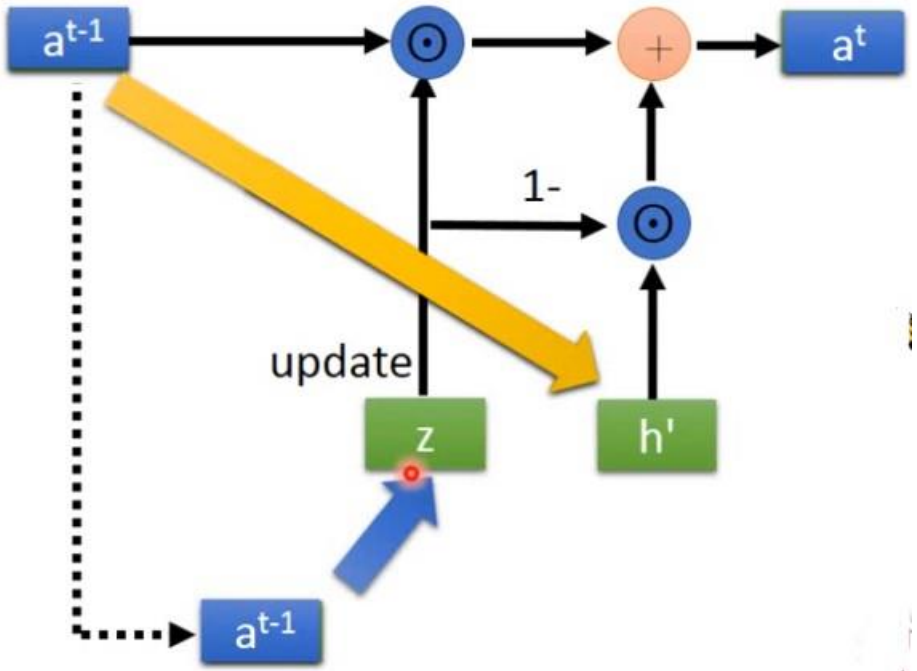


$$a^t = f(a^{t-1}, x^t) = \sigma\left(W^h a^{t-1} + W^i x^t + b^i\right)$$

Highway vs. GRU

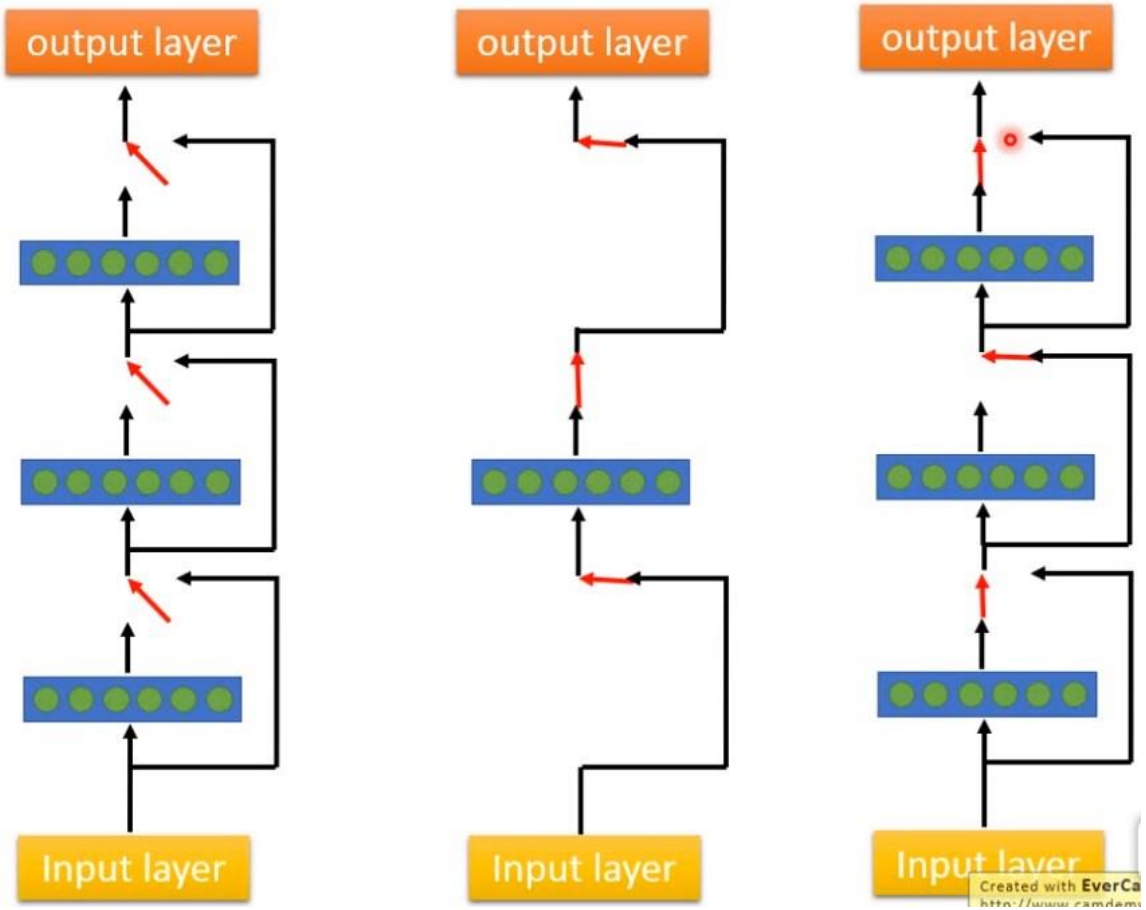


$$z_t = \sigma(W_z x_t + R_z h_{t-1} + b_z) \quad (11)$$
$$r_t = \sigma(W_r x_t + R_r h_{t-1} + b_r) \quad (12)$$
$$y_t = \tanh(W_h x_t + R_h(r_t \odot h_{t-1}) + b_h) \quad (13)$$
$$h_t = z_t \odot y_t + (1 - z_t) \odot h_{t-1} \quad (14)$$



$$y = \sigma(W_y x + b_y) \quad (1)$$
$$g = \sigma(W_g x + b_g) \quad (2)$$
$$h = g \odot y + (1 - g) \odot x \quad (3)$$

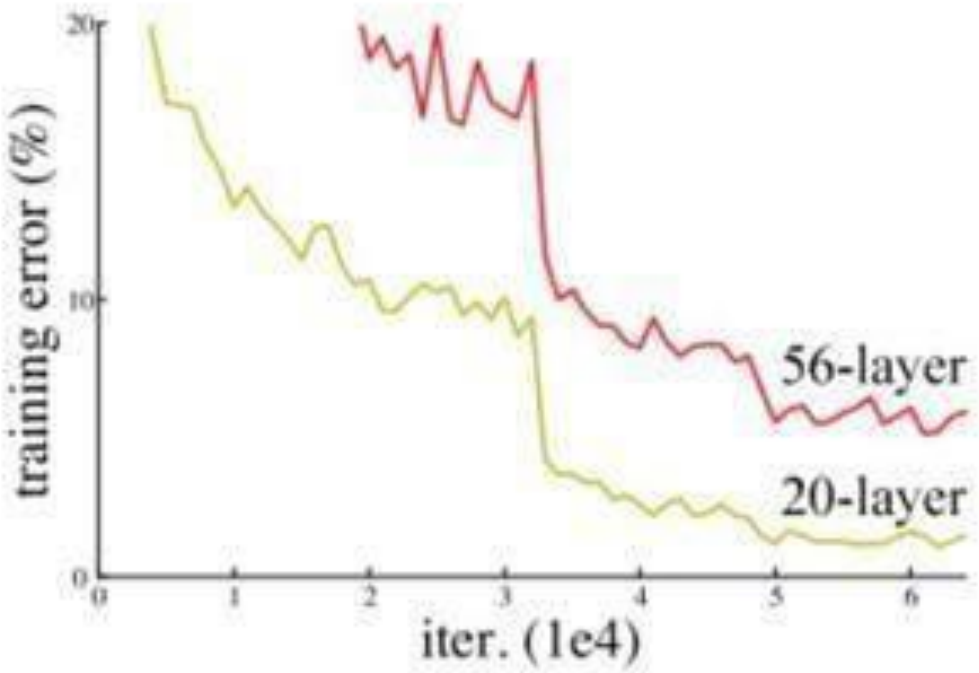
深层highway

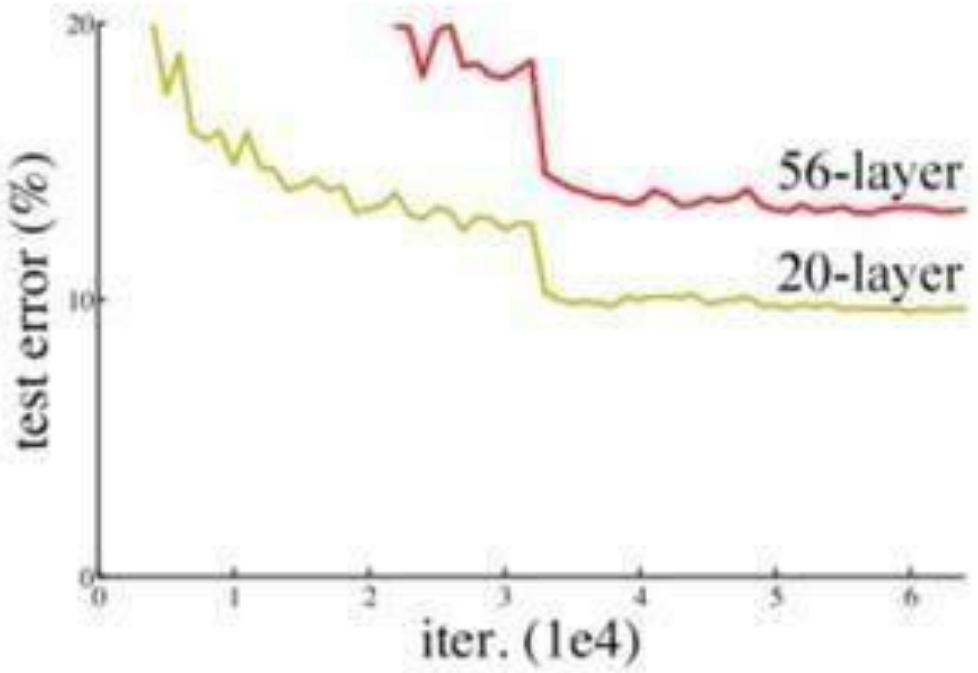

output layer
output layer
output layer
Input layer
Input layer
Input layer

ResNet

- ResNet在2015年被提出，在ImageNet比赛classification任务上获得第一名，因为它“简单与实用”并存，之后很多方法都建立在ResNet50或者ResNet101的基础上完成的，检测，分割，识别等领域都纷纷使用ResNet，Alpha zero也使用了ResNet，所以可见ResNet确实很好用。

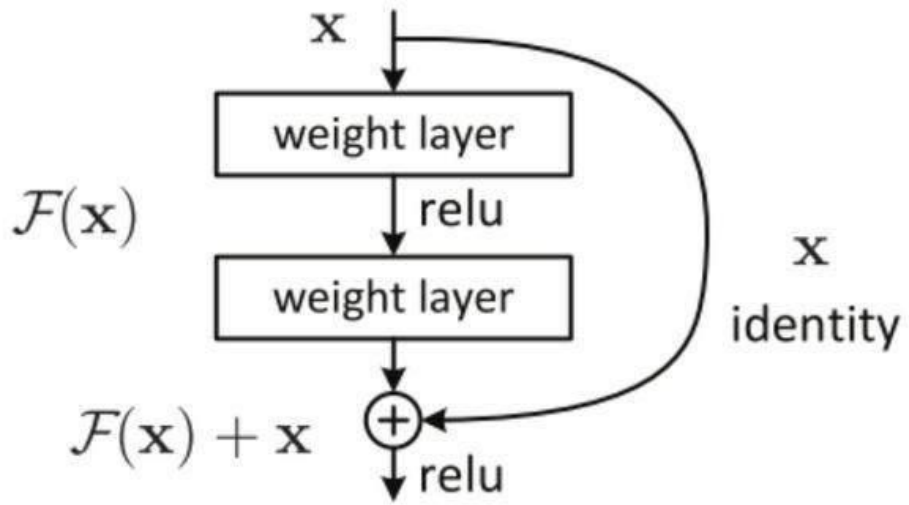
(网络退化）增加网络深度导致性能下降



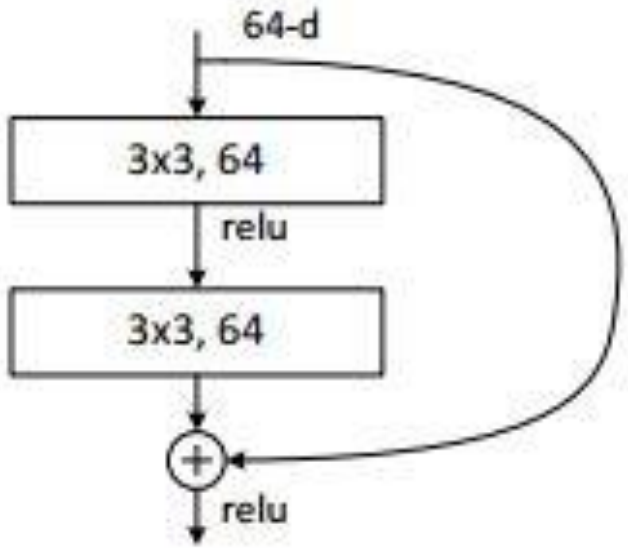


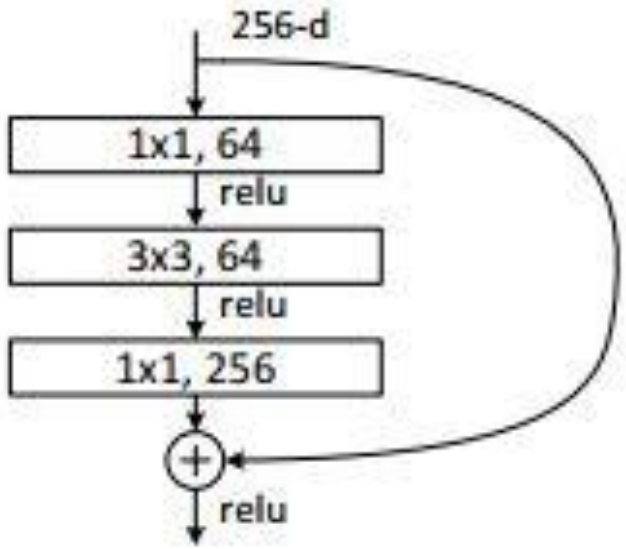
残差指的是什么？

- 其中ResNet提出了两种mapping：一种是identity mapping，指的就是图1中"弯弯的曲线"，另一种residual mapping，指的就是除了"弯弯的曲线"那部分，所以最后的输出是 y=F(x)+x



不同的残差结构


64-d
3x3, 64
relu
3x3, 64
relu


256-d
1x1, 64
relu
3x3, 64
relu
1x1, 256
relu

为什么要作两层旁路?

数学推导

ResNet

$$x_{l+2} = x_{l+1} + F(x_{l+1}, W_{l+1})$$

$$= x_l + F(x_l, W_l) + F(x_{l+1}, W_{l+1})$$

$$= x_{l-1} + F(x_{l-1}, W_{l-1}) + F(x_l, W_l) + F(x_{l+1}, W_{l+1})$$

$$\Rightarrow x_l = x_0 + \sum_{i=0}^{l-1} F(x_i, W_i)$$

① 无 shortcut 时

$$\frac{\partial Loss}{\partial x_l} = \frac{\partial Loss}{\partial x_{l+2}} \cdot \frac{\partial x_{l+2}}{\partial x_{l+1}} \cdot \frac{\partial x_{l+1}}{\partial x_l}$$

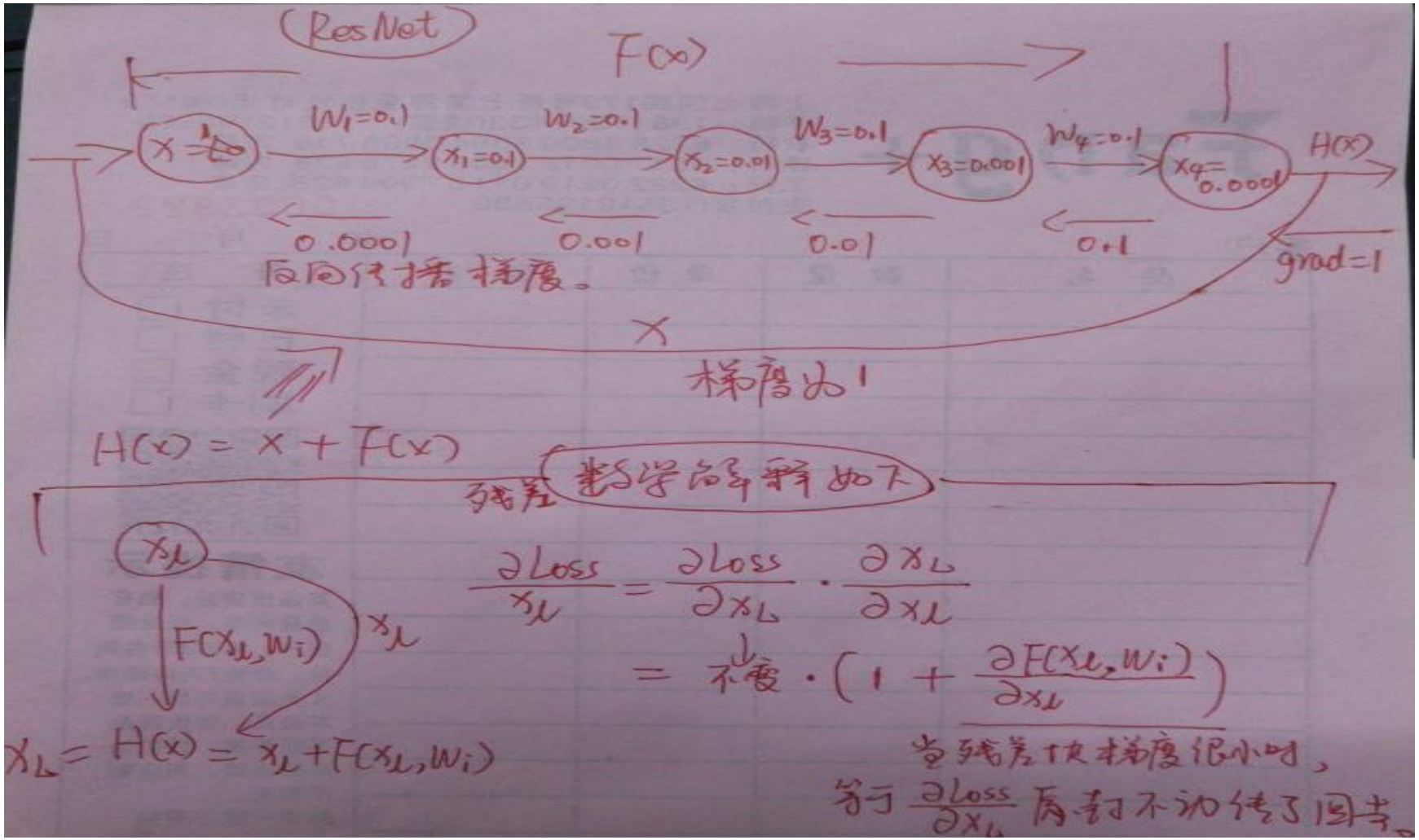
② 有 shortcut 时

$$\frac{\partial Loss}{\partial x_l} = \frac{\partial Loss}{\partial x_{l+2}} \cdot \frac{\partial (x_l + F(x_l, W_l) + F(x_{l+1}, W_{l+1}))}{\partial x_l}$$

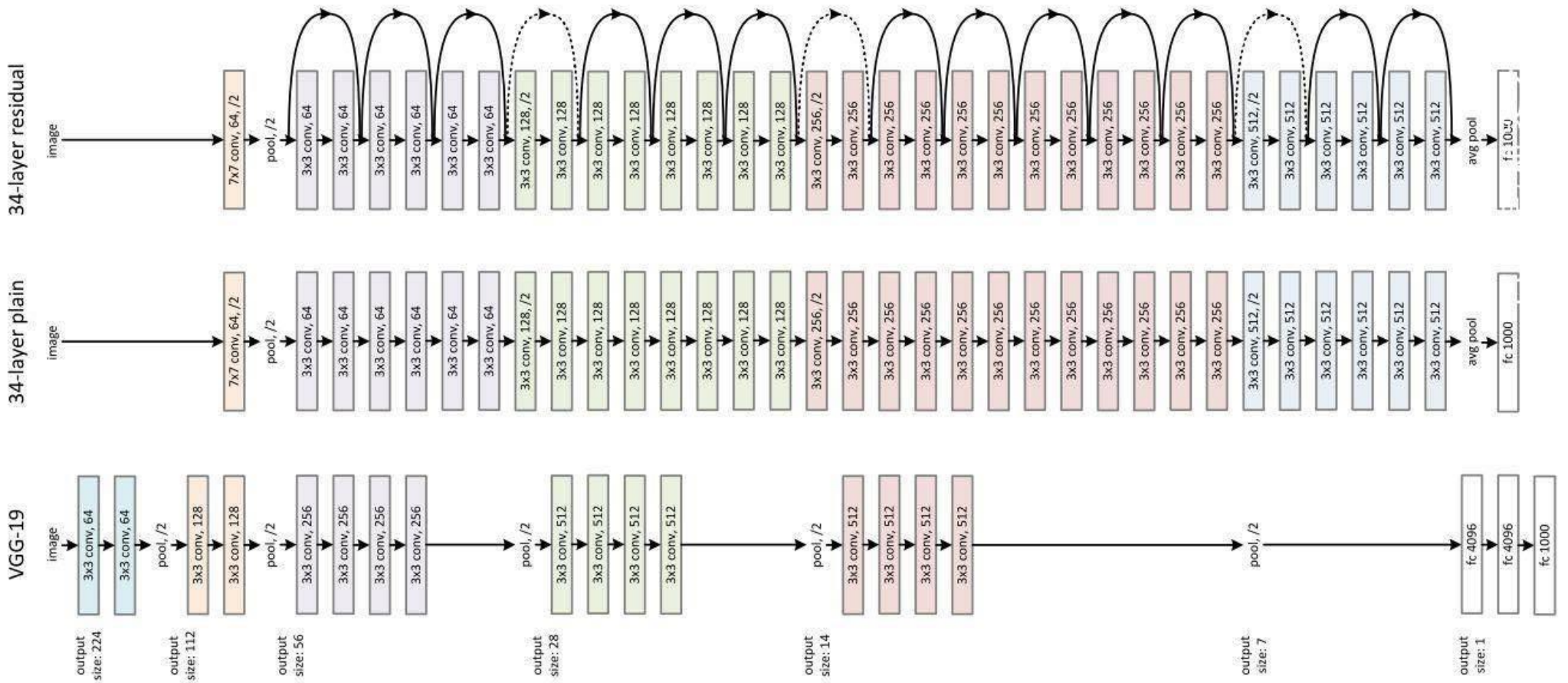
$$= \frac{\partial Loss}{\partial x_{l+2}} \cdot \left[1 + \frac{\partial}{\partial x_l} \sum_{i=l}^{L-1} F(x_i, W_i)\right]$$

$$= \frac{\partial Loss}{\partial x_{l+2}} + \frac{\partial Loss}{\partial x_{l+2}} \cdot \frac{\partial}{\partial x_l} \sum_{i=l}^{L-1} F(x_i, W_i)$$

数学推导


ResNet
F(x)
W1=0.1
W2=0.1
W3=0.1
W4=0.1
x=1
x1=0.1
x2=0.01
x3=0.001
x4=0.0001
H(x)
0.0001
0.001
0.01
0.1
反向传播梯度。
grad=1
梯度为1
H(x) = x + F(x)
残差
数学解释如下
xL
F(xL, Wi)
xL
xL = H(x) = xL + F(xL, Wi)
∂Loss/∂xl = ∂Loss/∂xL · ∂xL/∂xl
= 不要 · (1 + ∂F(xl, Wi)/∂xl)
当残差的梯度很小时，
等于 ∂Loss/∂xL 反射不加传了回来。

残差深度网络


VGG-19
image
3x3 conv, 64
3x3 conv, 64
pool, /2
3x3 conv, 128
3x3 conv, 128
pool, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
fc 4096
fc 4096
fc 1000
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1
34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000
34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000

实现和虚线

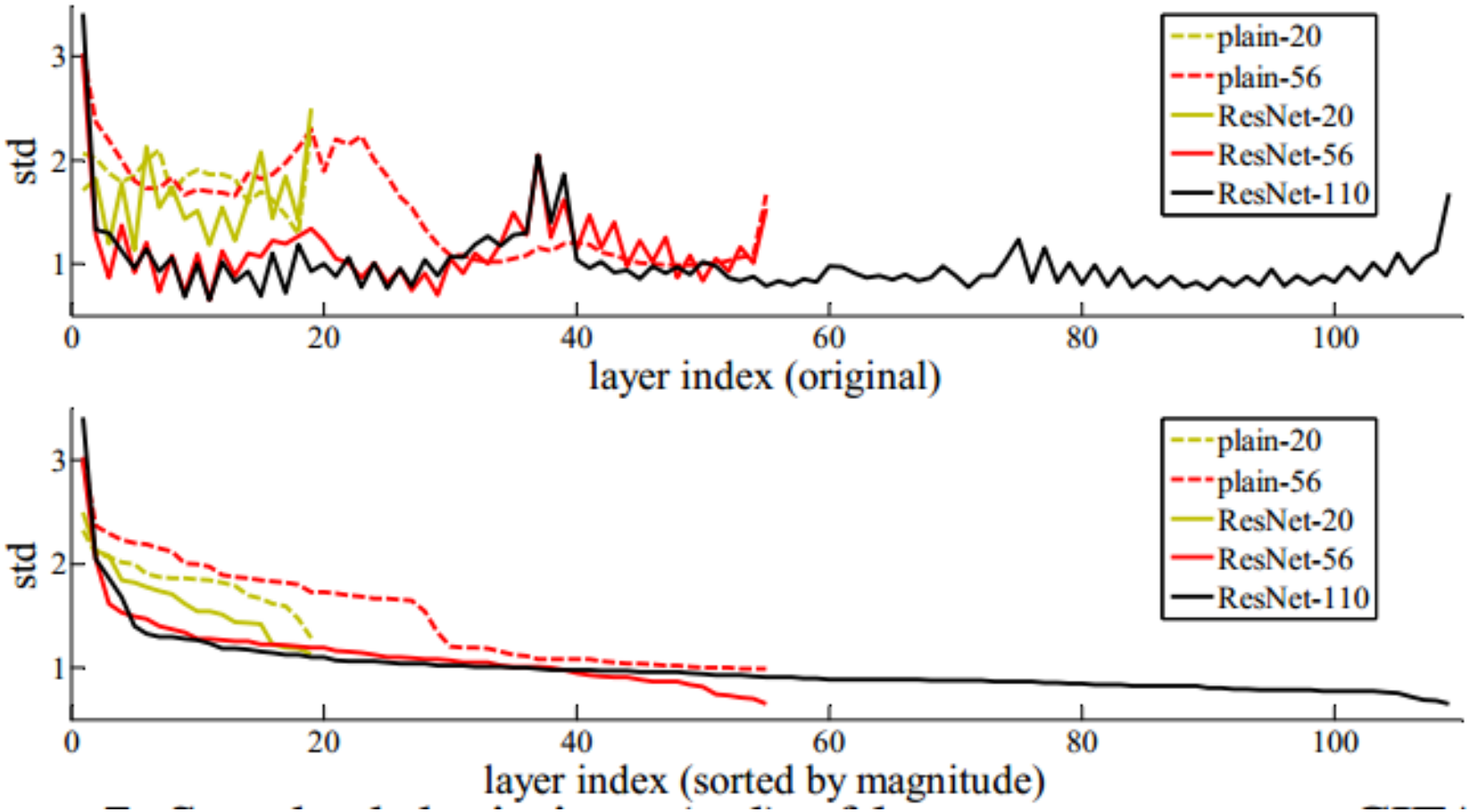
- 因为经过“shortcut connections（捷径连接）”后，H(x)=F(x)+x，如果F(x)和x的通道相同，则可直接相加，那么通道不同怎么相加呢。上图中的实线、虚线就是为了区分这两种情况的：
- 实线的Connection部分，表示通道相同，如上图的第一个粉色矩形和第三个粉色矩形，都是3x3x64的特征图，由于通道相同，所以采用计算方式为H(x)=F(x)+x
- 虚线的的Connection部分，表示通道不同，如上图的第一个绿色矩形和第三个绿色矩形，分别是3x3x64和3x3x128的特征图，通道不同，采用的计算方式为H(x)=F(x)+Wx，其中W是卷积操作，用来调整x维度的。

对网络深度的理解

更多的ResNet 结构

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ImageNet 上分类准确度对比



深层分析原因1:特征冗余

- 在正向卷积时，对每一层做卷积其实只提取了图像的一部分信息，这样一来，越到深层，原始图像信息的丢失越严重，而仅仅是对原始图像中的一小部分特征做提取。这显然会发生类似欠拟合的现象。加入shortcut结构，相当于在每个block中又加入了上一层图像的全部信息，一定程度上保留了更多的原始信息。

深层分析原因2:**特征集成**

- 加入shortcut后**相当于一个ensemble模型**，输出的结果是前面各个block及其组合一起做的一个投票选出的结果。即可以把ResNet网络看成是**多个子网络并行**，从实验中看，真实起作用的路径长度并不深，主要走是中等深度的网络。简单来说，就是做了**不同层次上的特征组合**。

深层分析原因3:特征具有层次性

- 对于同一个任务，对不同样本可能适合不同层次（复杂程度）的特征就可以完成。例如：对一瓶纯净水和一瓶饮料做二分类，如果一个样本是黄色的，更具颜色特征判断它显然会是饮料，如果样本是无色的，就需要通过它的味道或成分等更复杂的特征判断了。回到网络结构上面，浅层网络提取的是简单的特征，而简单和复杂的特征适用于不同的样本，没有shortcut时，对所有样本的分类都是利用最复杂的特征判断，费时费力；加入shortcut后，相当于保留了一些简单的特征用于判断，变得省时。这一观点主要解释了为什么ResNet网络能够更快收敛。

最终效果

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

Inception resnetV2

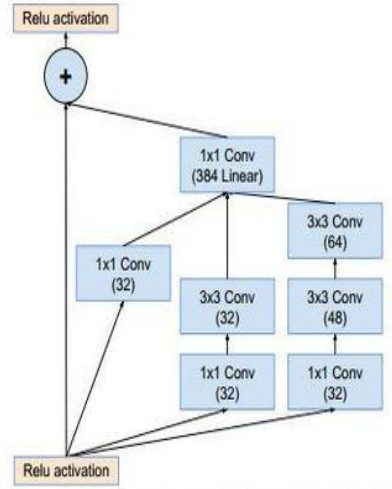


Figure 16. The schema for 35 × 35 grid (Inception-ResNet-A) module of the Inception-ResNet-v2 network.

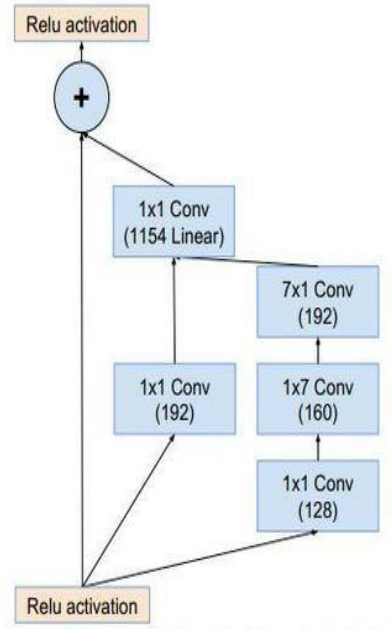


Figure 17. The schema for 17 × 17 grid (Inception-ResNet-B) module of the Inception-ResNet-v2 network.

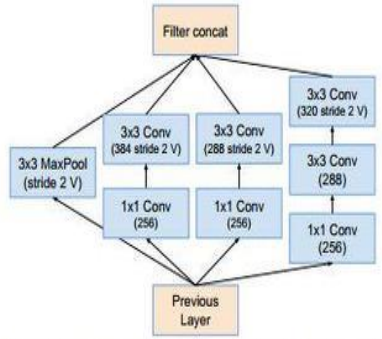


Figure 18. The schema for 17 × 17 to 8 × 8 grid-reduction module. Reduction-B module used by the wider Inception-ResNet-v1 network in Figure 15.

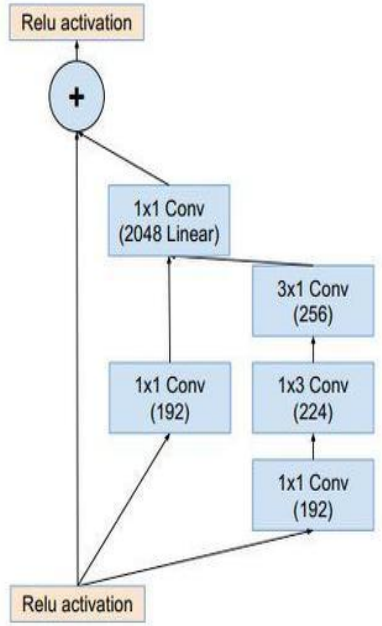
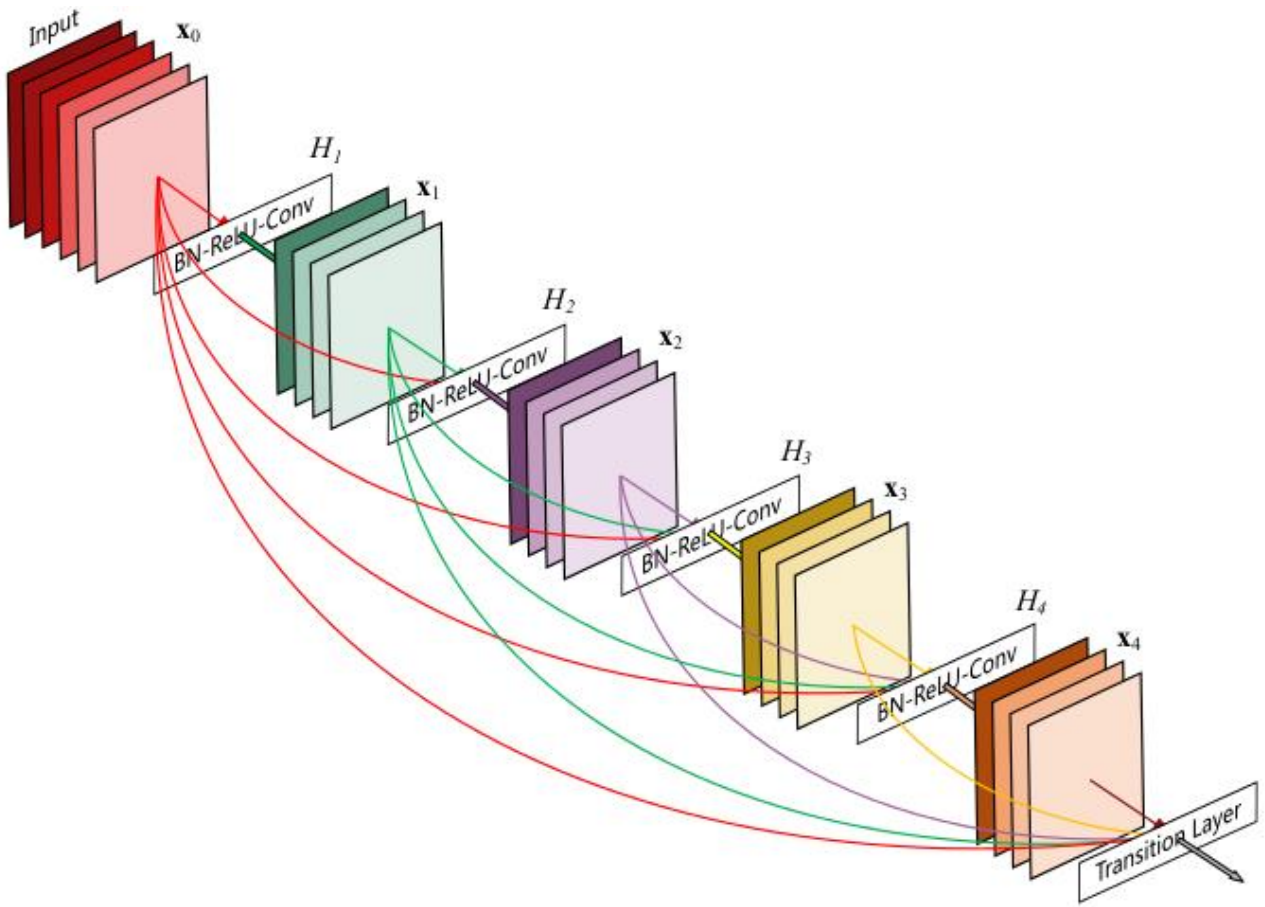


Figure 19. The schema for 8×8 grid (Inception-ResNet-C) module of the Inception-ResNet-v2 network.

Network	k	l	m	n
Inception-v4	192	224	256	384
Inception-ResNet-v1	192	192	256	384
Inception-ResNet-v2	256	256	384	384

Table 1. The number of filters of the Reduction-A module for the three Inception variants presented in this paper. The four numbers in the colums of the paper parametrize the four convolutions of Figure 7.

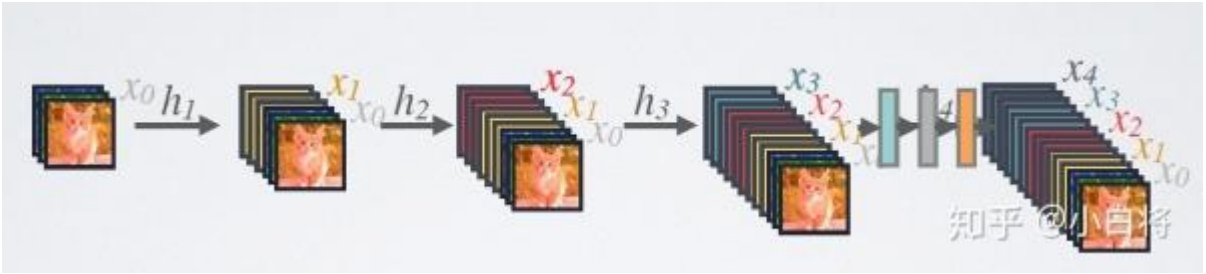
DenseNet


Input
$\mathbf{x}_0$
H_1
BN-ReLU-Conv
$\mathbf{x}_1$
H_2
BN-ReLU-Conv
$\mathbf{x}_2$
H_3
BN-ReLU-Conv
$\mathbf{x}_3$
H_4
BN-ReLU-Conv
$\mathbf{x}_4$
Transition Layer

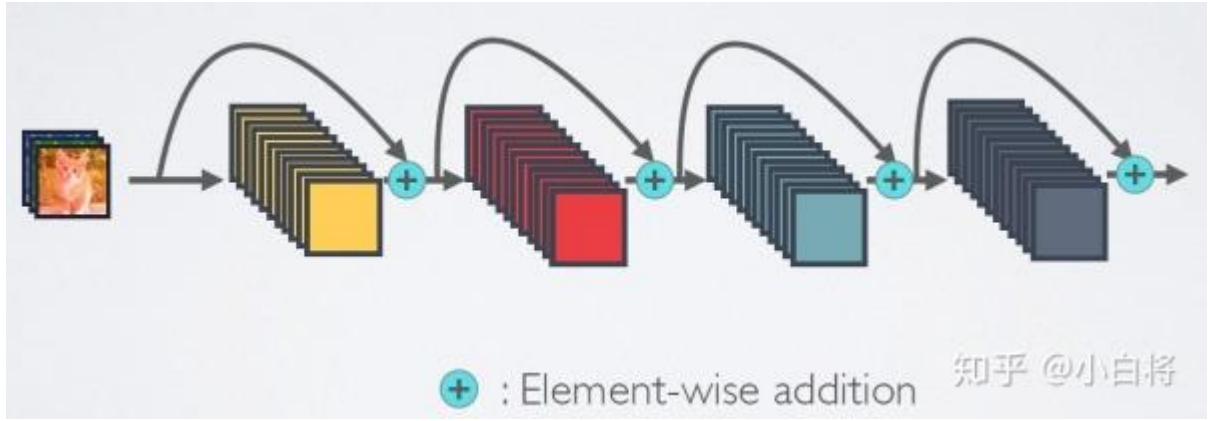
DenseNet

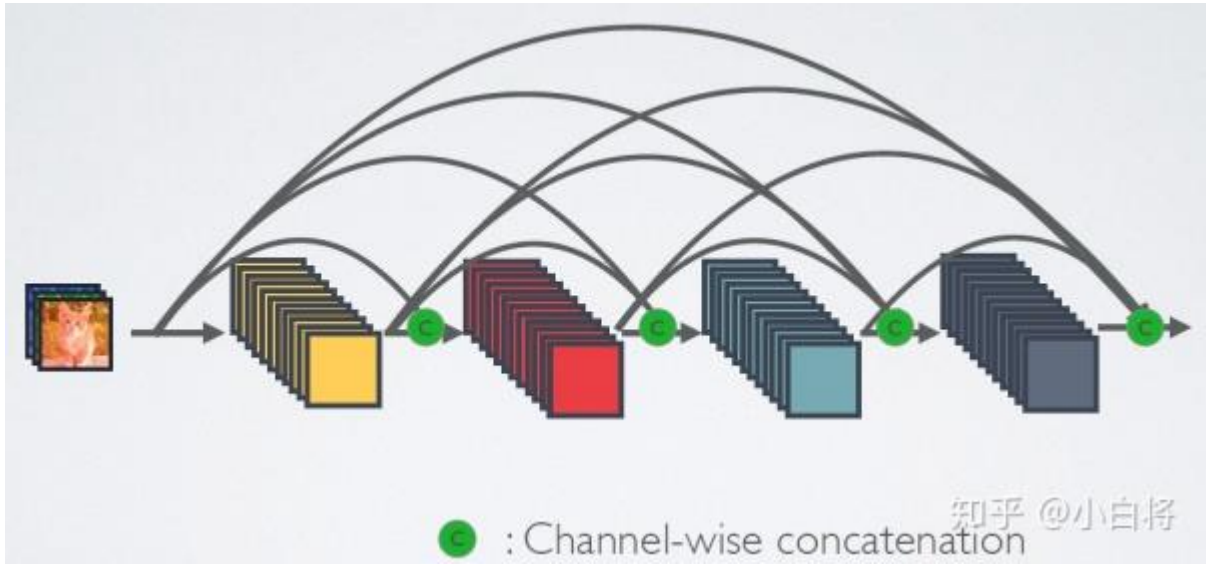
- Dense connectivity

在DenseNet结构中，讲每一层的输出都导入后面的所有层，与ResNet的相加不同的是，DenseNet结构使用的是连结结构（concatenate）。这样的结构可以减少网络参数，避免ResNet中可能出现的缺点（例如某些层被选择性丢弃，信息阻塞等）。



Densenet vs. Resnet





Densenet vs ResNet

普通DN

$$x_l = H_l(x_{l-1})$$

Resnet

$$x_l = H_l(x_{l-1}) + x_{l-1}$$

Densenet

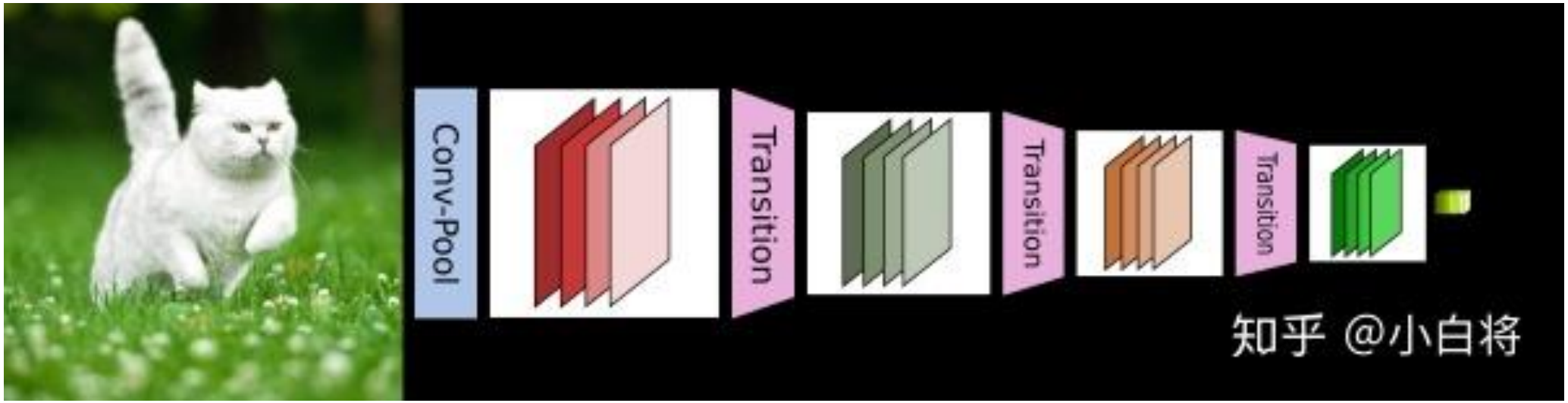
$$x_l = H_l([x_0, x_1, \ldots, x_{l-1}])$$

Densenet

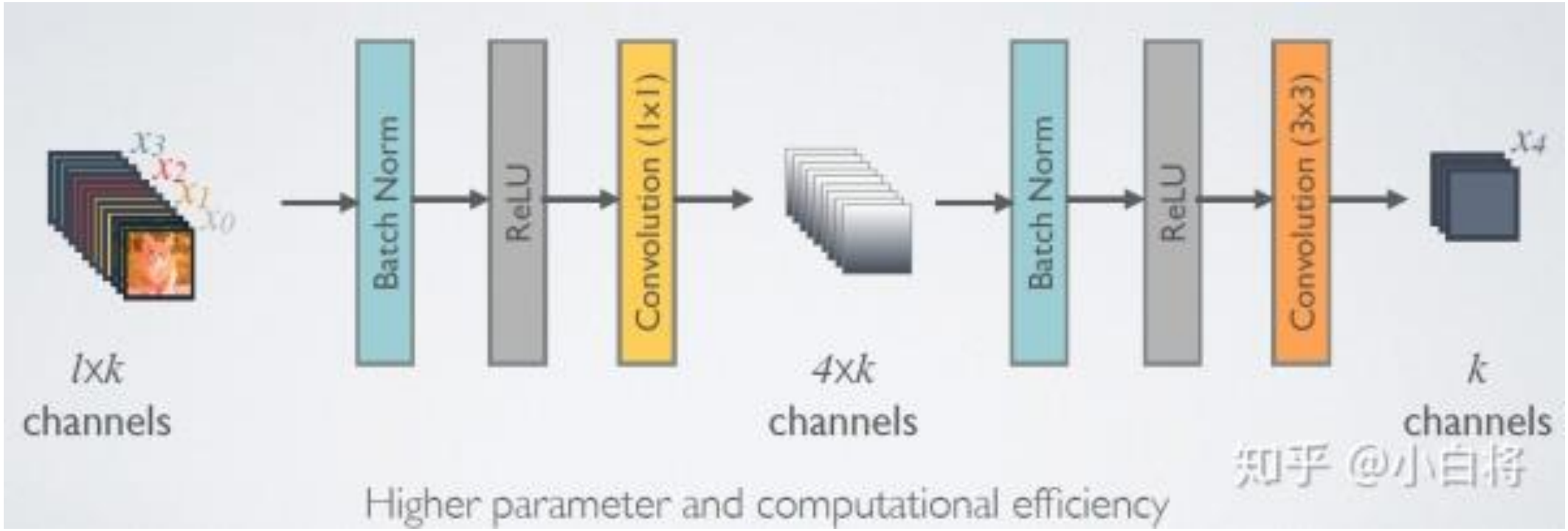
Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 36$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Table 1. DenseNet architectures for ImageNet. The growth rate for the first 3 networks is $k = 32$, and $k = 48$ for DenseNet-161. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

Bottleneck 和Transition layer



Bottleneck


x3
x2
x1
x0
Batch Norm
ReLU
Convolution (1x1)
Batch Norm
ReLU
Convolution (3x3)
x4
1xk
channels
4xk
channels
k
channels
知乎 @小白将
Higher parameter and computational efficiency

Densenet 优点

- **这个网络的优点如下：**
- **1.减轻了梯度弥散的问题,使模型不容易过拟合**
- **2.增强了特征在各个层之间的流动，**因为每一层都与初始输入层还有最后的由loss function得到的梯度直接相连
- **3.大大减少了参数个数，提高训练效率**

Mobilenet

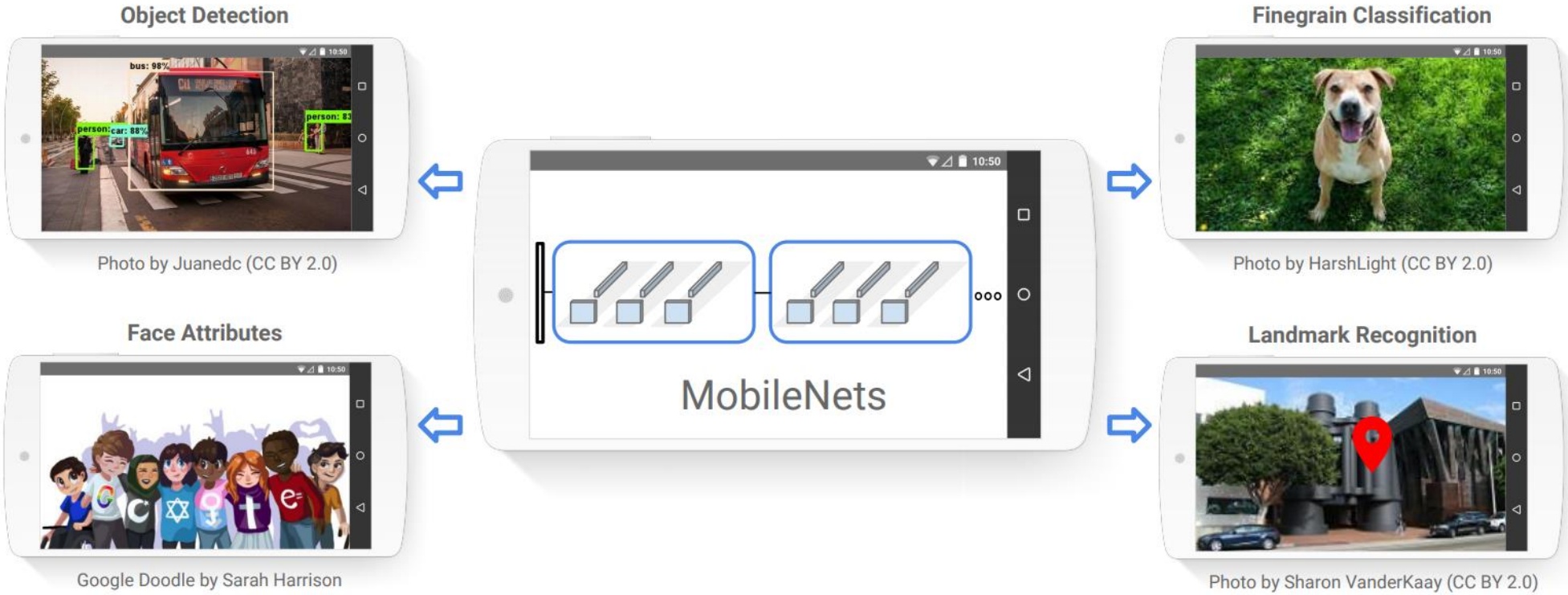
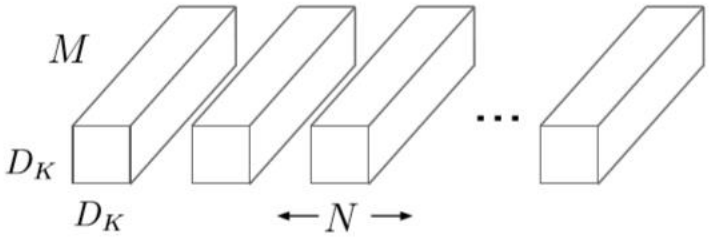
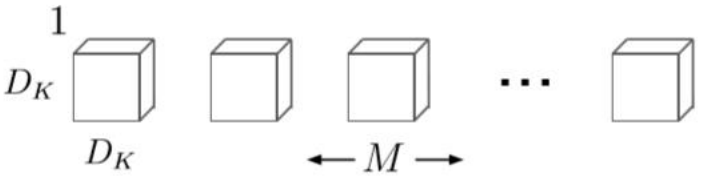


Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

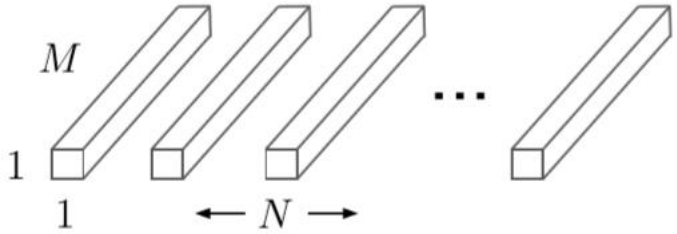
深度可分离卷积



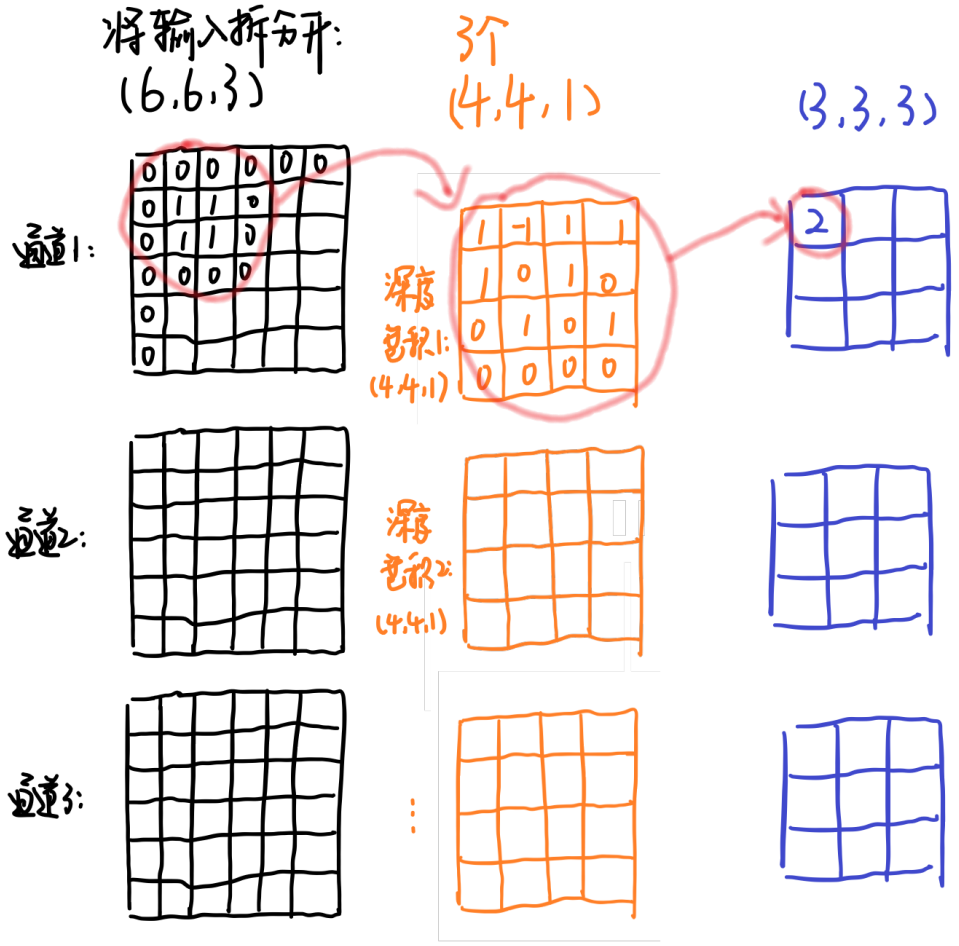
(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution



总结: 输入中第m个通道作用第m个深度卷积核,产生输出特征内的第m个通道.

深度可分离卷积

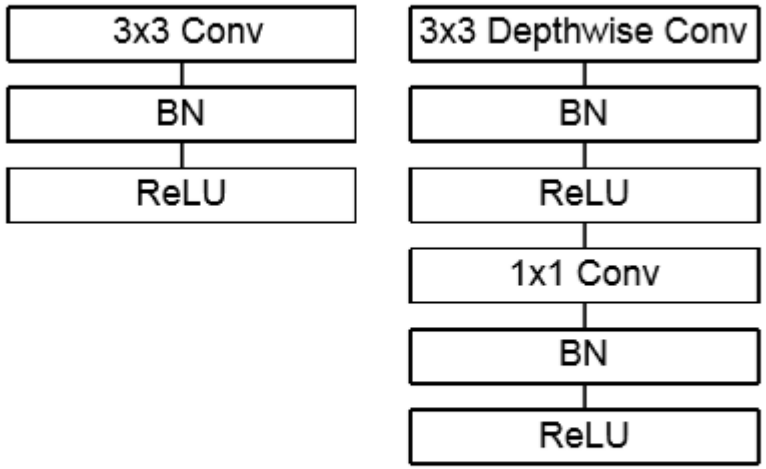


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

网络参数

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

历届冠军

模型	AlexNet	ZF Net	GoogLeNet	ResNet
时间（年）	2012	2013	2014	2015
层数（层）	8	8	22	152
Top 5 错误率	15.4%	11.2%	6.7%	3.57%
数据增强	√	√	√	√
Dropout	√	√		
批量归一化				√