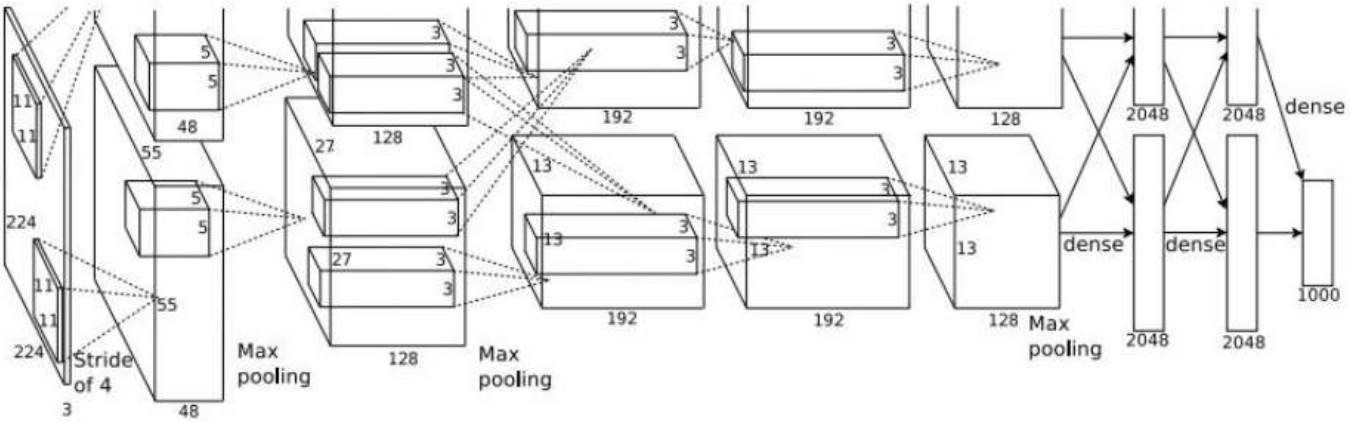
Lecture 9 CNN Architecture

一、AlexNet

1.基本网络结构图



2.结构解析

(1)卷积层和池化层参数体积计算

这是卷积层的：

Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**
Parameters: (11*11*3)*96 = **35K**

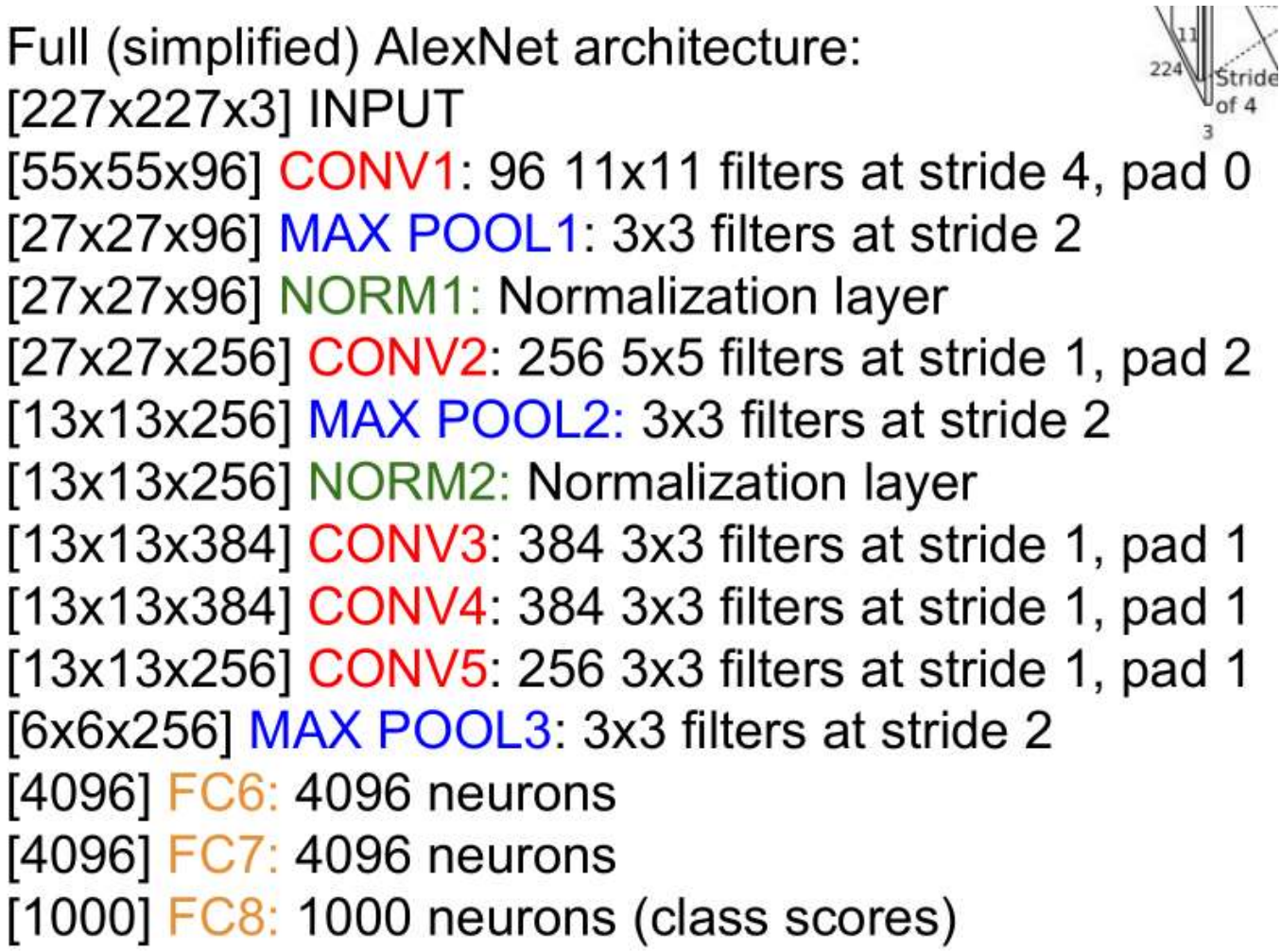
这是池化层的：

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96
Parameters: 0!

这里需要注意的是，因为池化处理只需要去取不同的区域然后做最大池化或者最小池化就好了，所以不存在在训练阶段需要学习的参数，因为参数数为0。

(2)整体结构的确定

下面的图就是解析了AlexNet整体结构



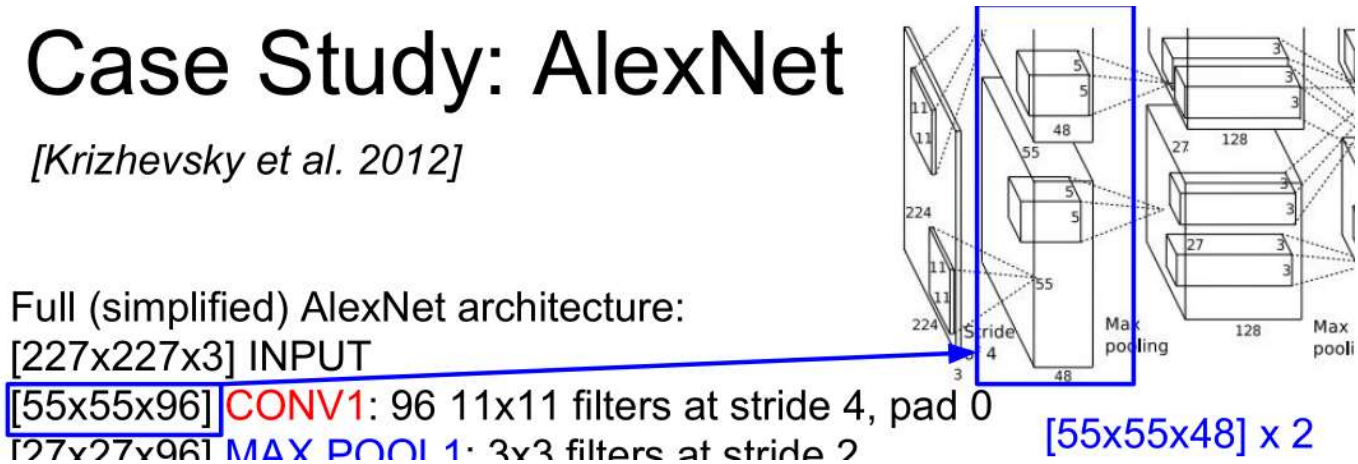
(3)一些注意事项

AlexNet使用到的一些优化和构建的技巧：

Details/Retrospectives:

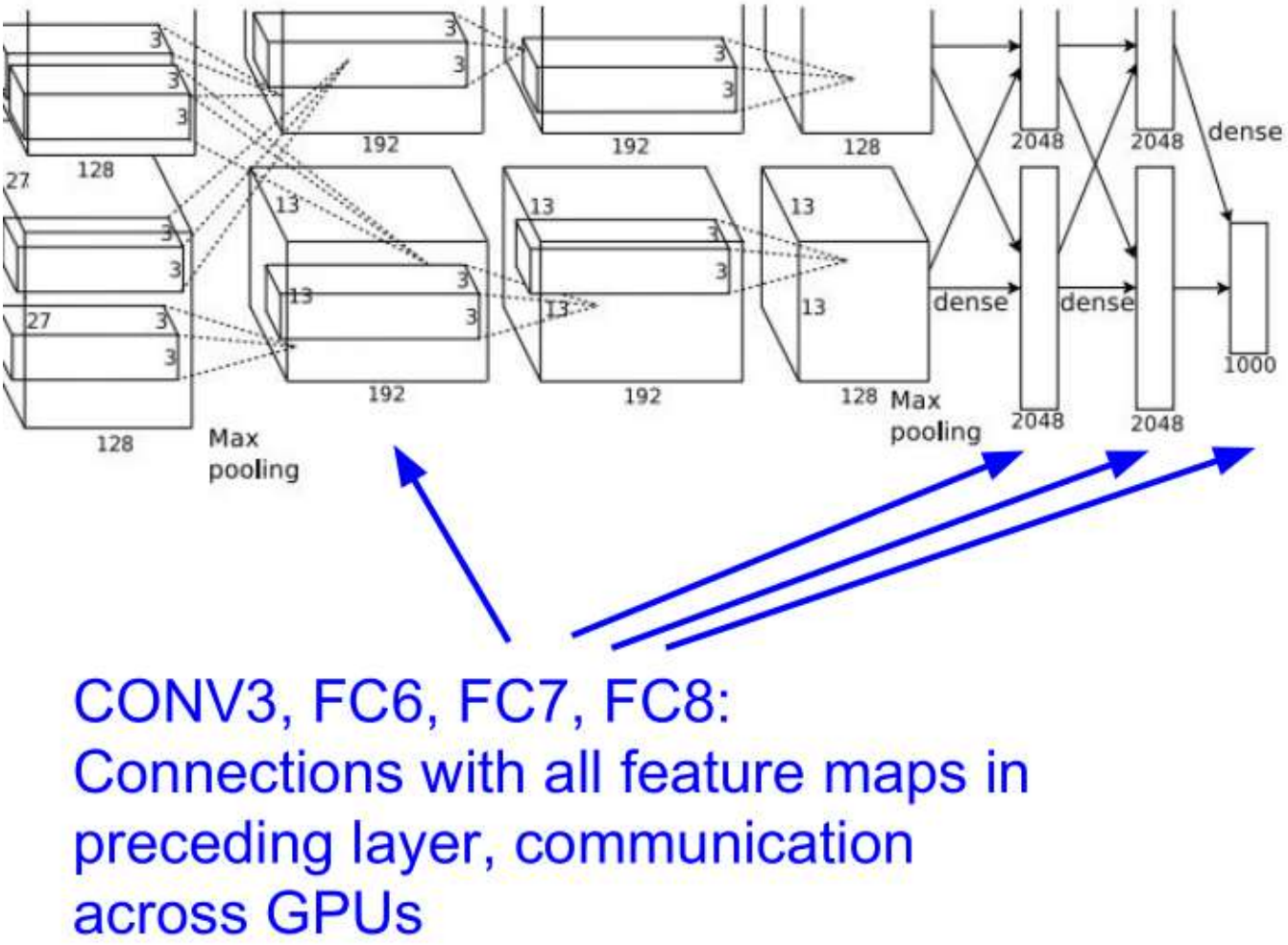
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

AlexNet因为一些历史问题而选择将整体结构做一个拆分：



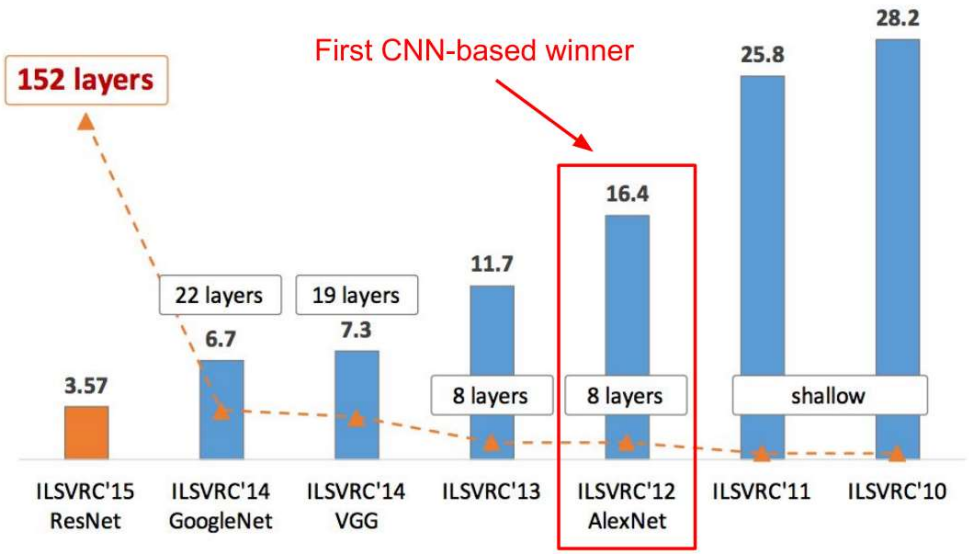
原因是当时2012年的时候，他是在GTX580的GPU上训练的，因为GPU的内存只有3G，没有办法容纳全部的网络，所以当时的设计者就将网络分为两部分在GPU上运行，使得网络可行。

还有一个就是，在第三层卷积核最后的全连接层中存在着不同GPU之间的通信，以达到fmap的统一性：



(4)ILSVRC比赛结果展示

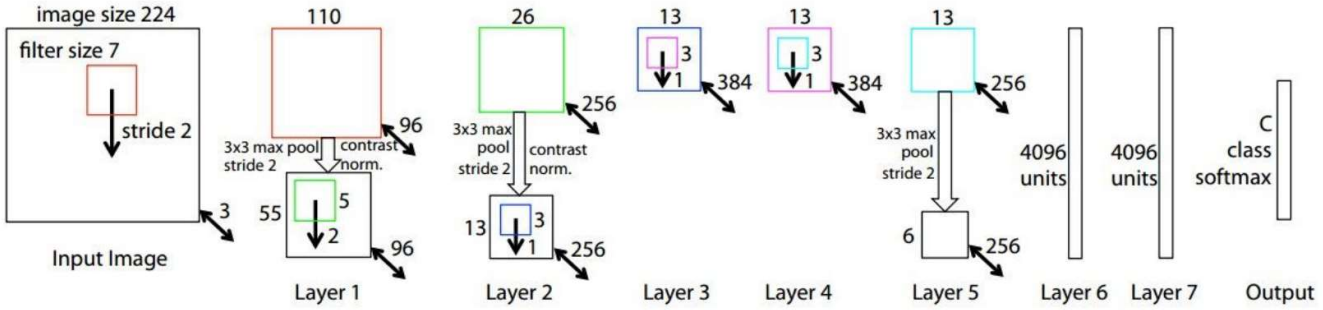
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



2013年获胜的网络是ZFNet，和AlexNet很像，只是改变了一些超参和步长之类的东西而已。

ZFNet

[Zeiler and Fergus, 2013]



TODO: remake figure

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: 16.4% -> 11.7%

二、VGGNet

1.基本结构

Case Study: VGGNet

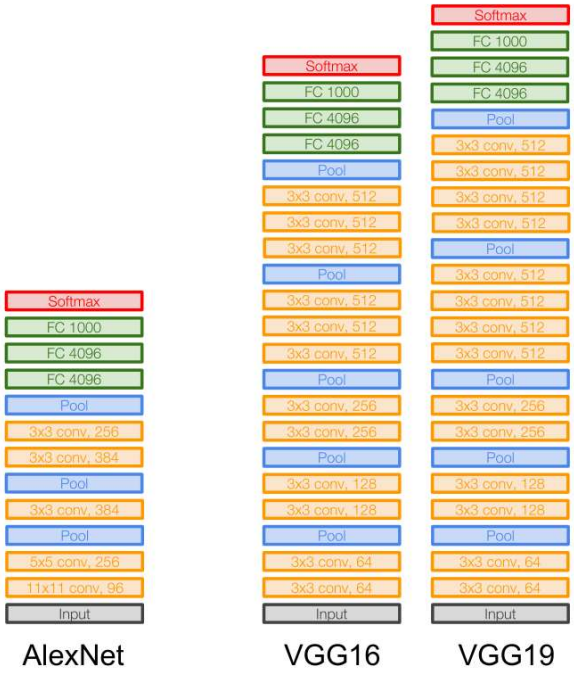
[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

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

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14



在这里需要去提出来的疑问就是：为什么VGG选择了比AlexNet更小的卷积核，优点在哪里？

Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

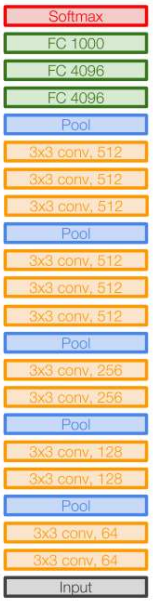
But deeper, more non-linearities

And fewer parameters: $3 * (3^2C^2)$ vs. 7^2C^2 for C channels per layer

下面展示整个网络结构的参数计算：

(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



VGG16

通过上面这张图我们需要掌握两个知识点：

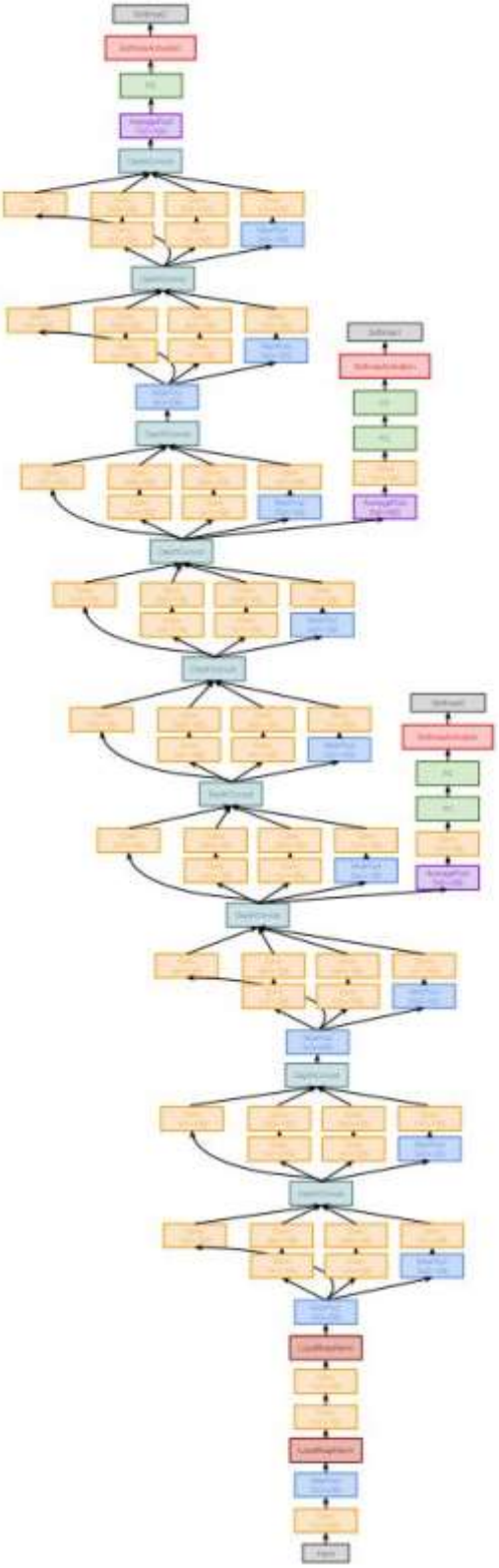
- 类似于VGG16这样的神经网络，他的参数量主要集中在最后的全连接层和一开始的卷积层
- 其实是注意各层的命名方式比如第一层的卷积为conv1-1,第二层卷积是conv1-2

2.一些小的Details

Details:

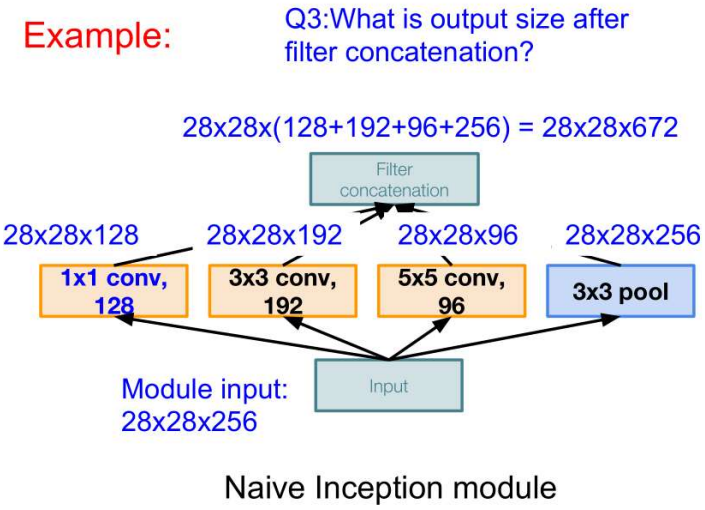
- ILSVRC'14 2nd in classification, 1st in localization
- Similar training procedure as Krizhevsky 2012
- No Local Response Normalisation (LRN)
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks

三、GoogleNet



1.Inception module

这个层是GoogleNet的核心知识点，他打破了一种观念就是：**一层之内并不是一定要只有单一的卷积核，可以有多种卷积核来提取信息。**

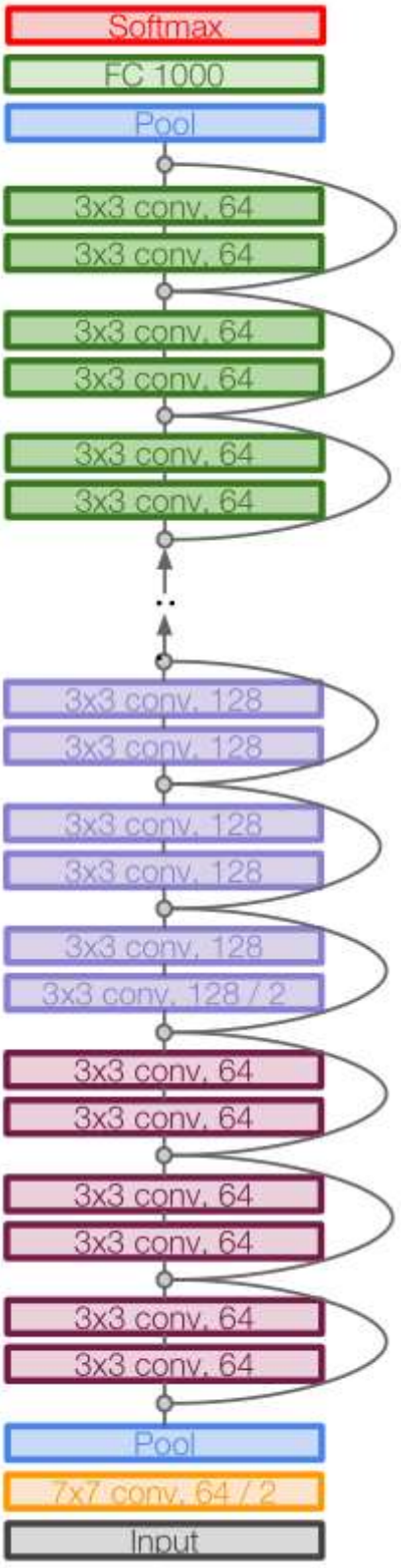


这个层唯一的问题就是存在着计算代价太大的问题。

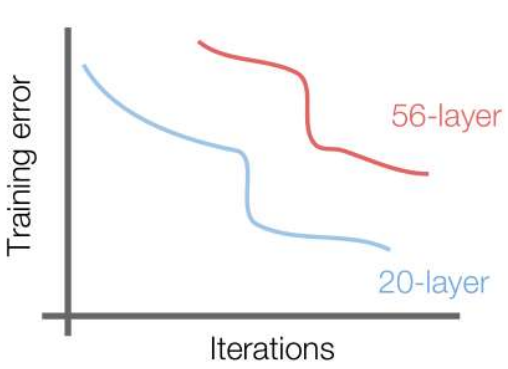
上面的1x1的卷积是为了降维，这个比较好理解。这样的降维是高维到低维，类似于一个瓶颈一样，所以称这一层是"**bottleneck layer**"。

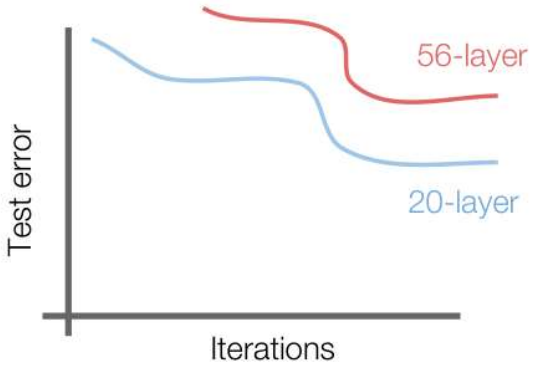
四、ResNet

基本结构如下图


Softmax
FC 1000
Pool
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128 / 2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
Pool
7x7 conv, 64 / 2
Input

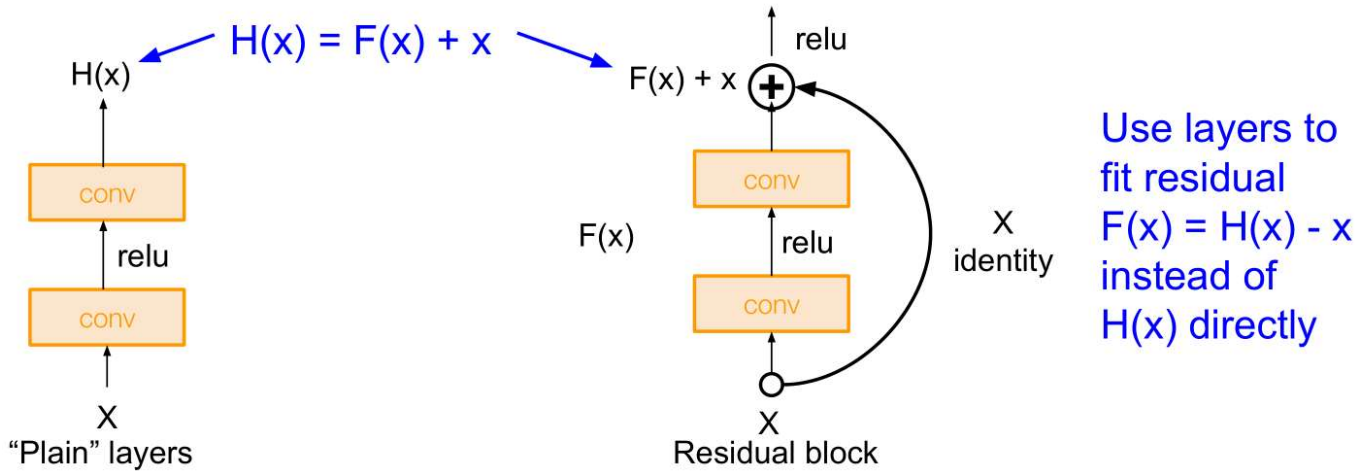
1.提出背景





提出的主要原因是上图。上图中可以发现，56层的网络无论是在训练还是在测试的过程中都不如20层的网络表现的好，这其实是反常规的。因为最起码两者的表现应当是差不多的，如果出现过拟合，也应该是56层的网络在训练阶段比20层的要好，但是现在出现了相反的情况。这就导致了ResNet的产生。

2.原理



上图就是一个残差块的计算方式，其中存在着两个理论：

- 第一，这个残差块有两条通路，一个是F(x)，一个是x，那么就存在着如果在足够深的网络过程中网络结果达到了最优，那么F(x)那个部分就会被push到0，这样就会使得最优的状态得到保存 —— x。这样就不会存在最优解得不到的问题了。
- 第二，就是如果F(x)和x的维度不一样，那么x也就不是x了，就会变成Wx，做一个降维处理。


res2b
res2b_relu
res2c_branch2a
bn2c_branch2a
scale2c_branch2a
res2c_branch2a_relu
res2c_branch2b
bn2c_branch2b
scale2c_branch2b
res2c_branch2b_relu
res2c_branch2c
bn2c_branch2c
scale2c_branch2c
res2c
res2c_relu
res2c
res2c_relu
res3a_branch2a
bn3a_branch2a
scale3a_branch2a
res3a_branch2a_relu
res3a_branch2b
bn3a_branch2b
scale3a_branch2b
res3a_branch2b_relu
res3a_branch1
bn3a_branch1
scale3a_branch1
res3a_branch2c
bn3a_branch2c
scale3a_branch2c
res3a
res3a_relu
http://blog.csdn.net/lanran2

3.Details

Training ResNet in practice:

- 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

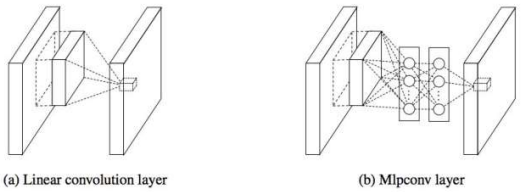
五、Other Architectures to know

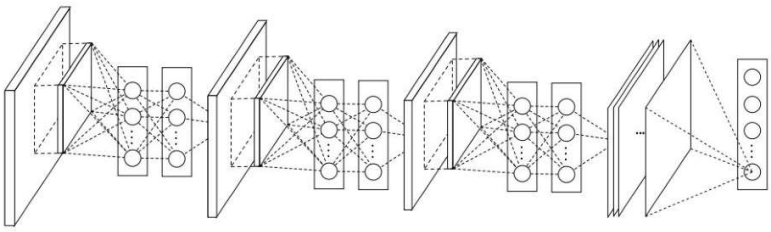
1.NiN

Network in Network (NiN)

[Lin et al. 2014]

- Mlpconv layer with “micronetwork” within each conv layer to compute more abstract features for local patches
- Micronetwork uses multilayer perceptron (FC, i.e. 1x1 conv layers)
- Precursor to GoogLeNet and ResNet “bottleneck” layers
- Philosophical inspiration for GoogLeNet



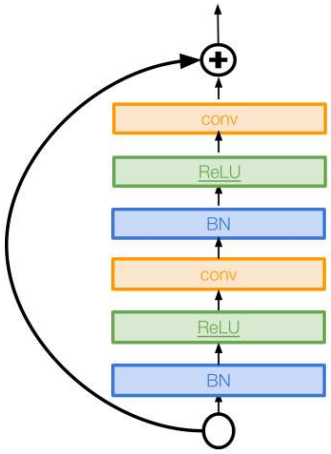


2.ResNet残差块的改进

Identity Mappings in Deep Residual Networks

[He et al. 2016]

- Improved ResNet block design from creators of ResNet
- Creates a more direct path for propagating information throughout network (moves activation to residual mapping pathway)
- Gives better performance

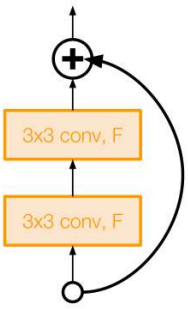


3.Wide Residual Networks

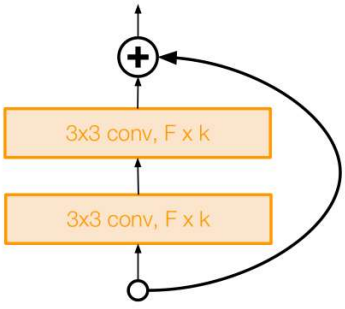
Wide Residual Networks

[Zagoruyko et al. 2016]

- Argues that residuals are the important factor, not depth
- User wider residual blocks (F x k filters instead of F filters in each layer)
- 50-layer wide ResNet outperforms 152-layer original ResNet
- Increasing width instead of depth more computationally efficient (parallelizable)



Basic residual block



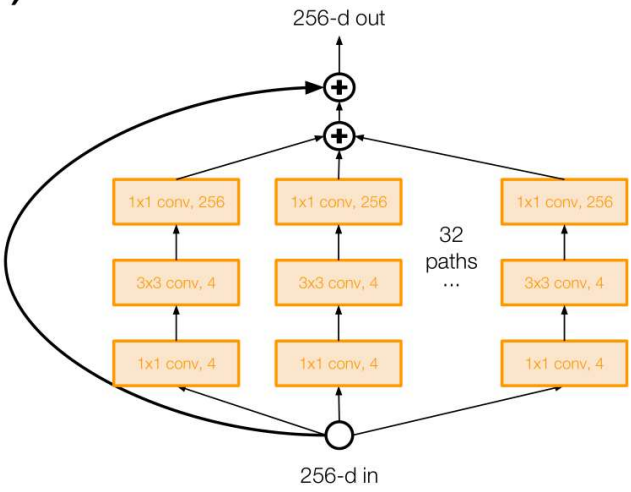
Wide residual block

4.ResNeXt

Aggregated Residual Transformations for Deep Neural Networks (ResNeXt)

[Xie et al. 2016]

- Also from creators of ResNet
- Increases width of residual block through multiple parallel pathways ("cardinality")
- Parallel pathways similar in spirit to Inception module

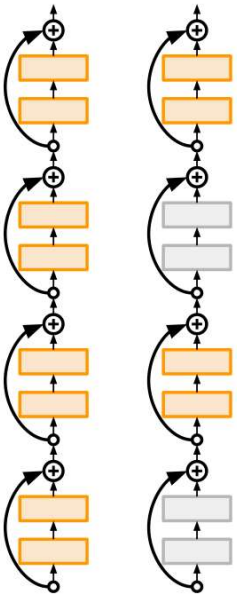


5.Deep Networks with Stochastic Depth(之前Justin Johnson的课上有提到过这个网络结构)

Deep Networks with Stochastic Depth

[Huang et al. 2016]

- Motivation: reduce vanishing gradients and training time through short networks during training
- Randomly drop a subset of layers during each training pass
- Bypass with identity function
- Use full deep network at test time



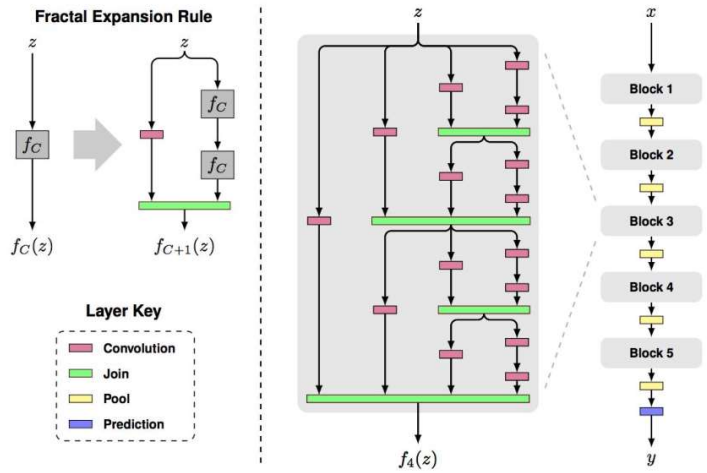
6.FractualNet

Beyond ResNets...

FractalNet: Ultra-Deep Neural Networks without Residuals

[Larsson et al. 2017]

- Argues that key is transitioning effectively from shallow to deep and residual representations are not necessary
- Fractal architecture with both shallow and deep paths to output
- Trained with dropping out sub-paths
- Full network at test time



Figures copyright Larsson et al., 2017. Reproduced with permission.

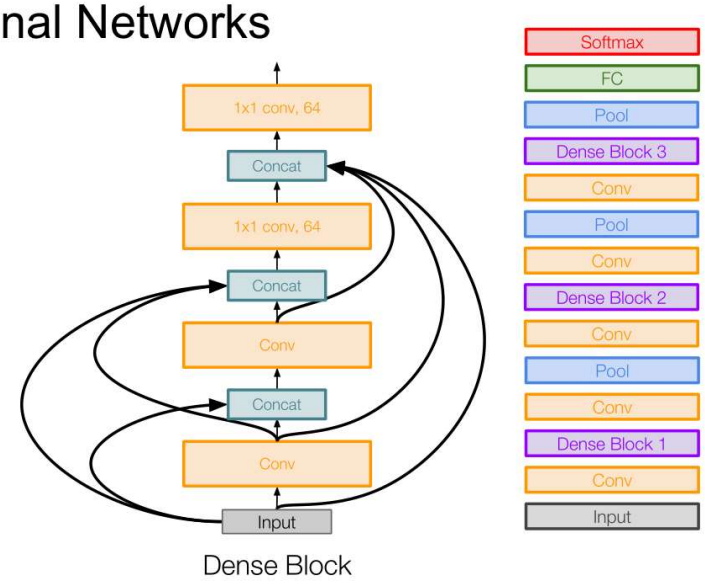
7.DenseNet

Beyond ResNets...

Densely Connected Convolutional Networks

[Huang et al. 2017]

- Dense blocks where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse



8.SqueezeNet

Efficient networks...

SqueezeNet: AlexNet-level Accuracy With 50x Fewer Parameters and <0.5Mb Model Size

[Iandola et al. 2017]

- Fire modules consisting of a ‘squeeze’ layer with 1x1 filters feeding an ‘expand’ layer with 1x1 and 3x3 filters
- AlexNet level accuracy on ImageNet with 50x fewer parameters
- Can compress to 510x smaller than AlexNet (0.5Mb)