**CNN的发展历史**

Lenet，1986年

Alexnet，2012年

GoogleNet，2014年

VGG，2014年

Deep Residual Learning，2015年

ResNeXt, 2016年

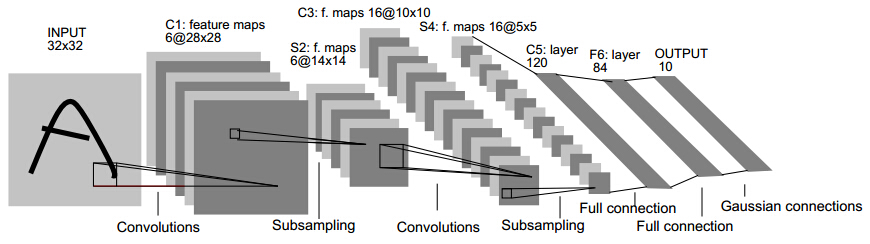
DenseNet，2017年

ImageNet历年冠军和相关CNN模型

<https://www.cnblogs.com/liaohuiqiang/p/9609162.html>

# Lenet

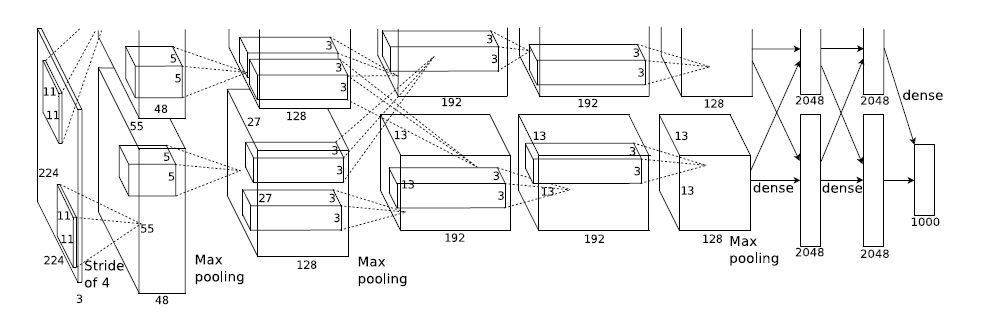
由两个卷积层，两个池化层，以及两个全连接层组成。 卷积都是5\*5的模板，stride=1，池化都是MAX。



# Alexnet

2012年，Imagenet比赛冠军的model——Alexnet [2]（以第一作者alex命名）。这个model的意义比后面那些model都大很多，首先它证明了CNN在复杂模型下的有效性，然后GPU实现使得训练在可接受的时间范围内得到结果，确实让CNN和GPU都大火了一把，顺便推动了有监督deep learning的发展。

模型结构见下图，别看只有寥寥八层（不算input层），但是它有60M以上的参数总量，事实上在参数量上比后面的网络都大。



从上图可以明显看到网络结构分为上下2层，这是表示网络分布在2个GPU上，因为一个NVIDIA GTX 580 GPU只有3GB内存，装不下这么大的网络。为了减少GPU之间的通信，第2，4，5个卷积层只连接同一个GPU上的上一层的kernel maps(见名词解释)。第3个卷积层连接第二层的所有kernel maps。全连接层的神经元连接到上一层所有的神经元。

这个图有点点特殊的地方是卷积部分都是画成上下两块，意思是说吧这一层计算出来的feature map分开，但是前一层用到的数据要看连接的虚线，如图中input层之后的第一层第二层之间的虚线是分开的，是说二层上面的128map是由一层上面的48map计算的，下面同理；而第三层前面的虚线是完全交叉的，就是说每一个192map都是由前面的128+128=256map同时计算得到的。

AlexNet将LeNet的思想发扬光大，把CNN的基本原理应用到了很深很宽的网络中。**AlexNet主要使用到的新技术点**如下：

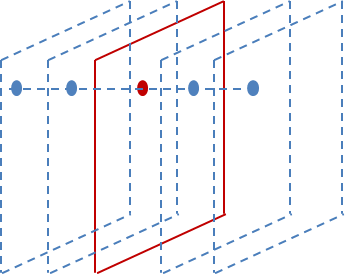
（1）成功使用ReLU作为CNN的激活函数，并验证其效果在较深的网络超过了Sigmoid，成功解决了Sigmoid在网络较深时的梯度弥散问题。虽然ReLU激活函数在很久之前就被提出了，但是直到AlexNet的出现才将其发扬光大。

（2）训练时使用Dropout随机忽略一部分神经元，以避免模型过拟合。Dropout虽有单独的论文论述，但是AlexNet将其实用化，通过实践证实了它的效果。在AlexNet中主要是最后几个全连接层使用了Dropout。

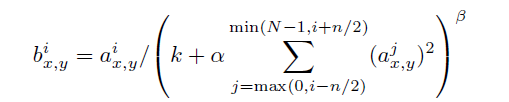
（3）在CNN中使用重叠的最大池化。此前CNN中普遍使用平均池化，AlexNet全部使用最大池化，避免平均池化的模糊化效果。并且AlexNet中提出让步长比池化核的尺寸小，这样池化层的输出之间会有重叠和覆盖，提升了特征的丰富性。

（4）提出了LRN ( Local Response Normalization) 局部响应归一化层，对局部神经元的活动创建竞争机制，使得其中响应比较大的值变得相对更大，并抑制其他反馈较小的神经元，增强了模型的泛化能力。

**LRN** ( Local Response Normalization) 局部响应归一化层，做的事是对当前层的输出结果做平滑处理。下面是示意图：



前后几层（对应位置的点）对中间这一层做一下平滑约束，计算方法是：



对以当前层为中心的n层数据做平滑。N表示通道数(channel)，a,n/2,k,α,β分别表示函数中的input, depth\_radius, bias, alpha, beta，其中n/2, k, α, β都是自定义的，详细介绍：

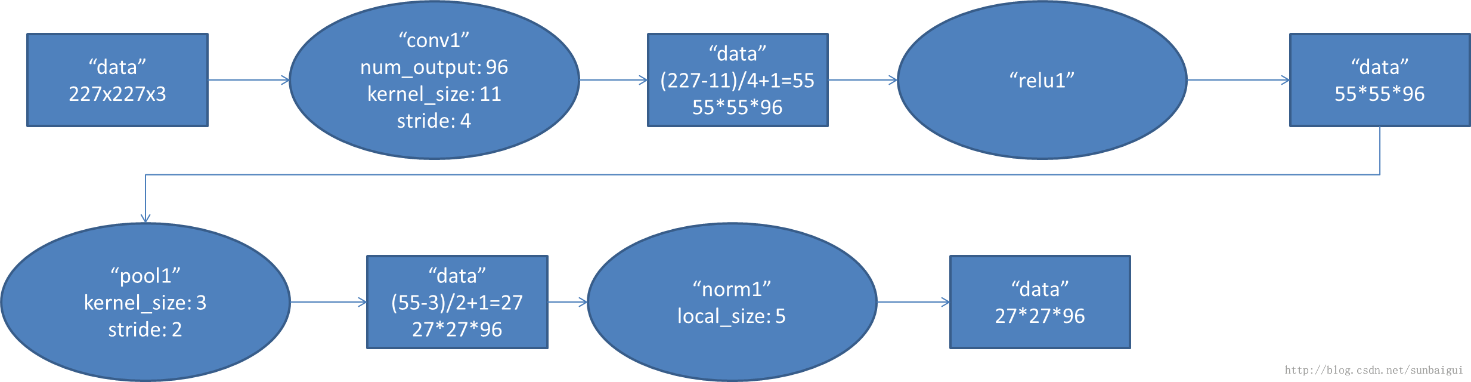
<https://blog.csdn.net/program_developer/article/details/79430119>

<https://blog.csdn.net/u014296502/article/details/78839881>

**Alexnet模型结构**

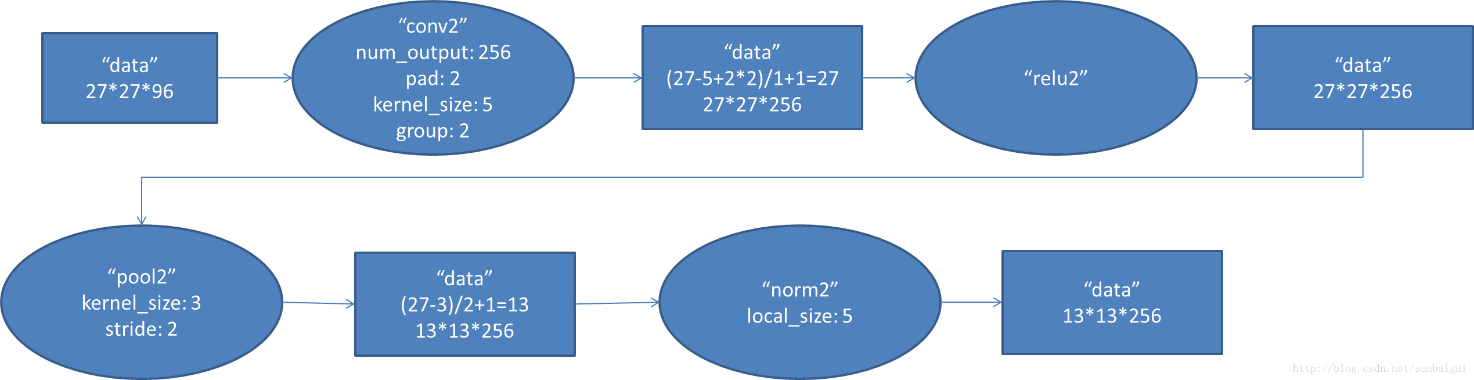
具体打开Alexnet的每一阶段（含一次卷积主要计算）来看[2][3]：

（1）con - relu - pooling - LRN



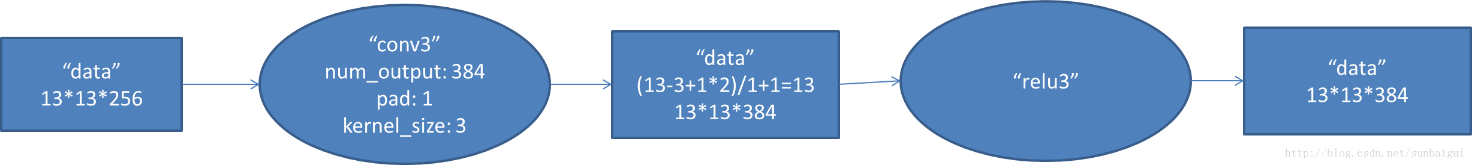
具体计算都在图里面写了，要注意的是input层是227\*227，而不是paper里面的224\*224，这里可以算一下，主要是227可以整除后面的conv1计算，224不整除。如果一定要用224可以通过自动补边实现，不过在input就补边感觉没有意义，补得也是0。

（2）conv - relu - pool - LRN

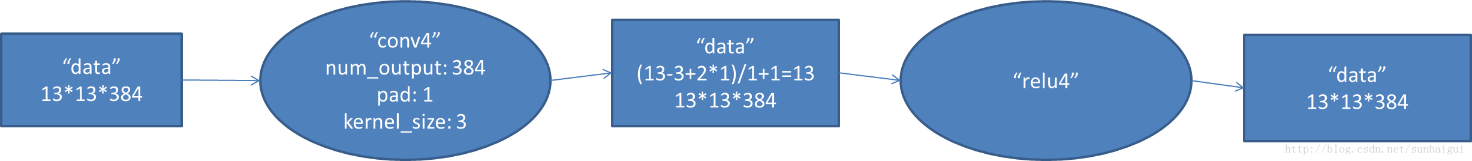


和上面基本一样，唯独需要注意的是group=2，这个属性强行把前面结果的feature map分开，卷积部分分成两部分做。

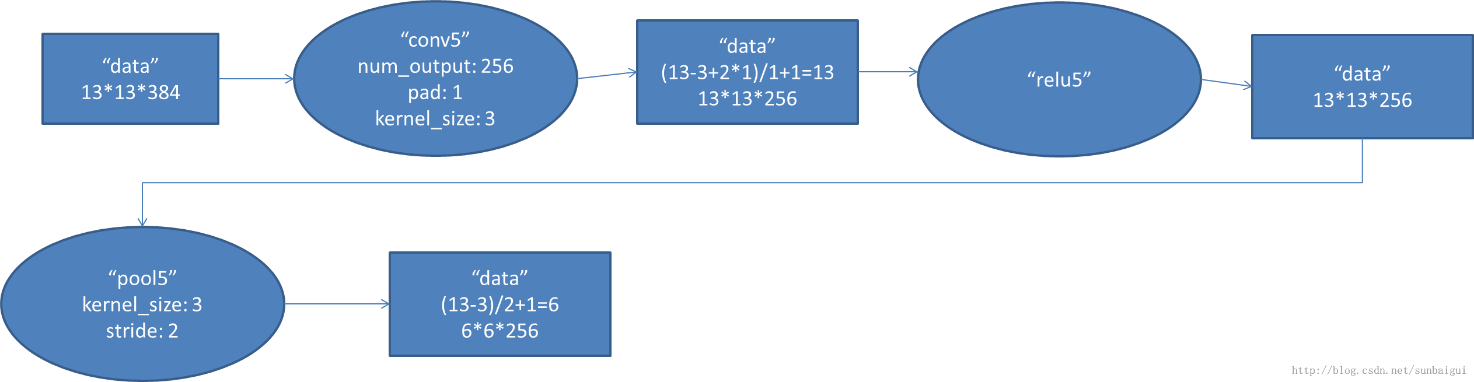
（3）conv - relu



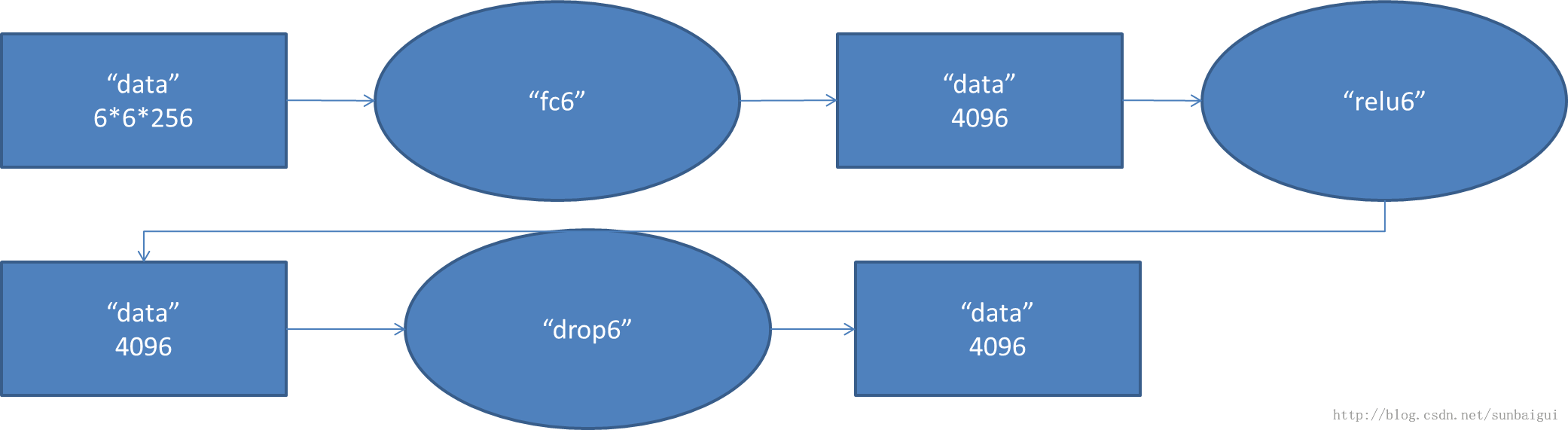
（4）conv-relu

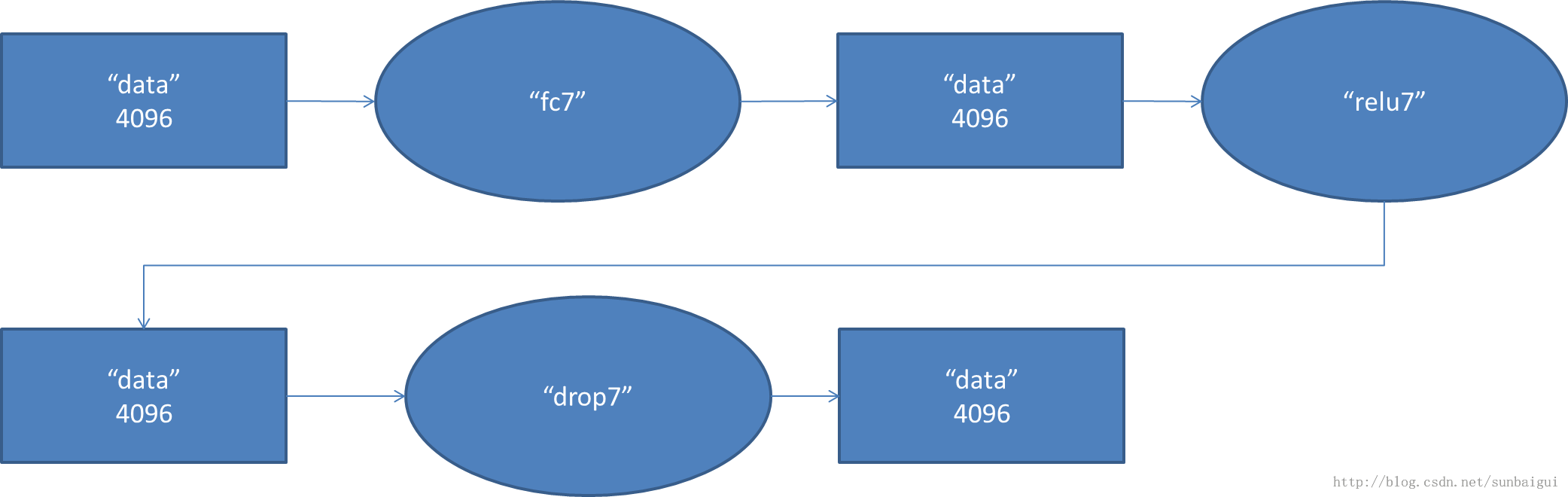
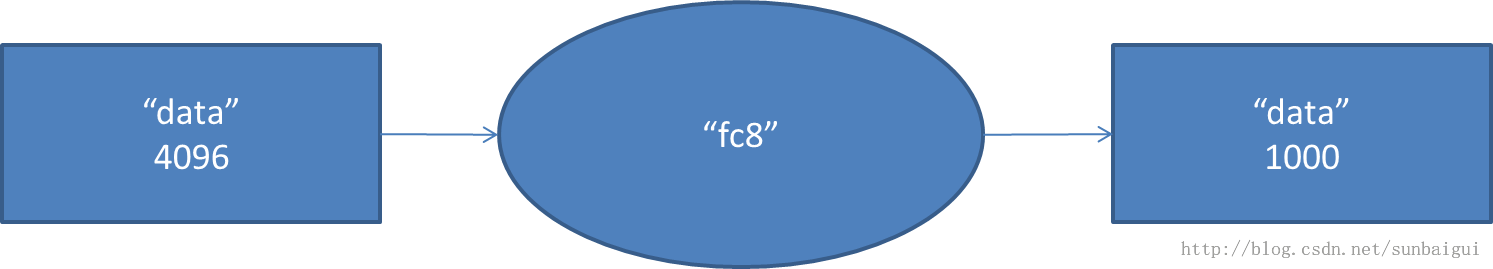


（5）conv - relu - pool

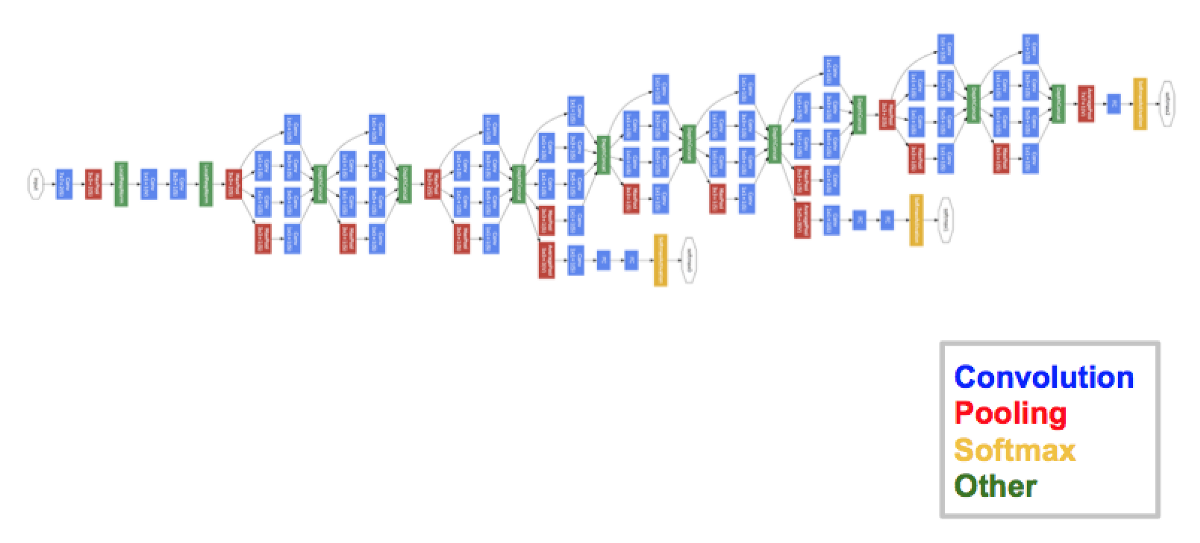


（6）fc - relu - dropout



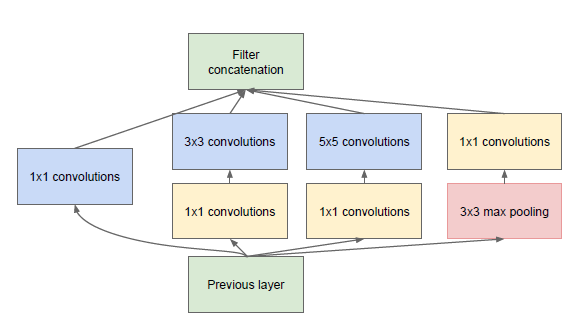
这里有一层特殊的dropout层，在alexnet中是说在训练的以1/2概率使得隐藏层的某些neuron的输出为0，这样就丢到了一半节点的输出，BP的时候也不更新这些节点。   
（7）   
fc - relu - dropout   
  
（8）fc - softmax   


# GoogLeNet

14年比赛冠军的model，这个model证明了一件事：用更多的卷积，更深的层次可以得到更好的结构。

22层，参数数量从60 million (AlexNet) 降到 4 million.

该模型的创新在于Inception，这是一种网中网的结构，即原来的结点也是一个网络。Inception一直在不断发展，目前已经V2、V3、V4了，感兴趣的同学可以查阅相关资料。Inception的结构如图所示，其中1\*1卷积主要用来降维，用了Inception之后整个网络结构的宽度和深度都可扩大，能够带来2-3倍的性能提升。

Inception结构图

      对**Inception的结构**做以下说明：

       1. 采用不同大小的卷积核意味着不同大小的感受野，最后拼接意味着不同尺度特征的融合；

       2. 之所以卷积核大小采用1、3和5，主要是为了方便对齐。设定卷积步长stride=1后，只要分别设定pad=0、1、2，那么卷积后便可以得到相同维度的特征，然后这些特征就可以直接拼接在一起了；

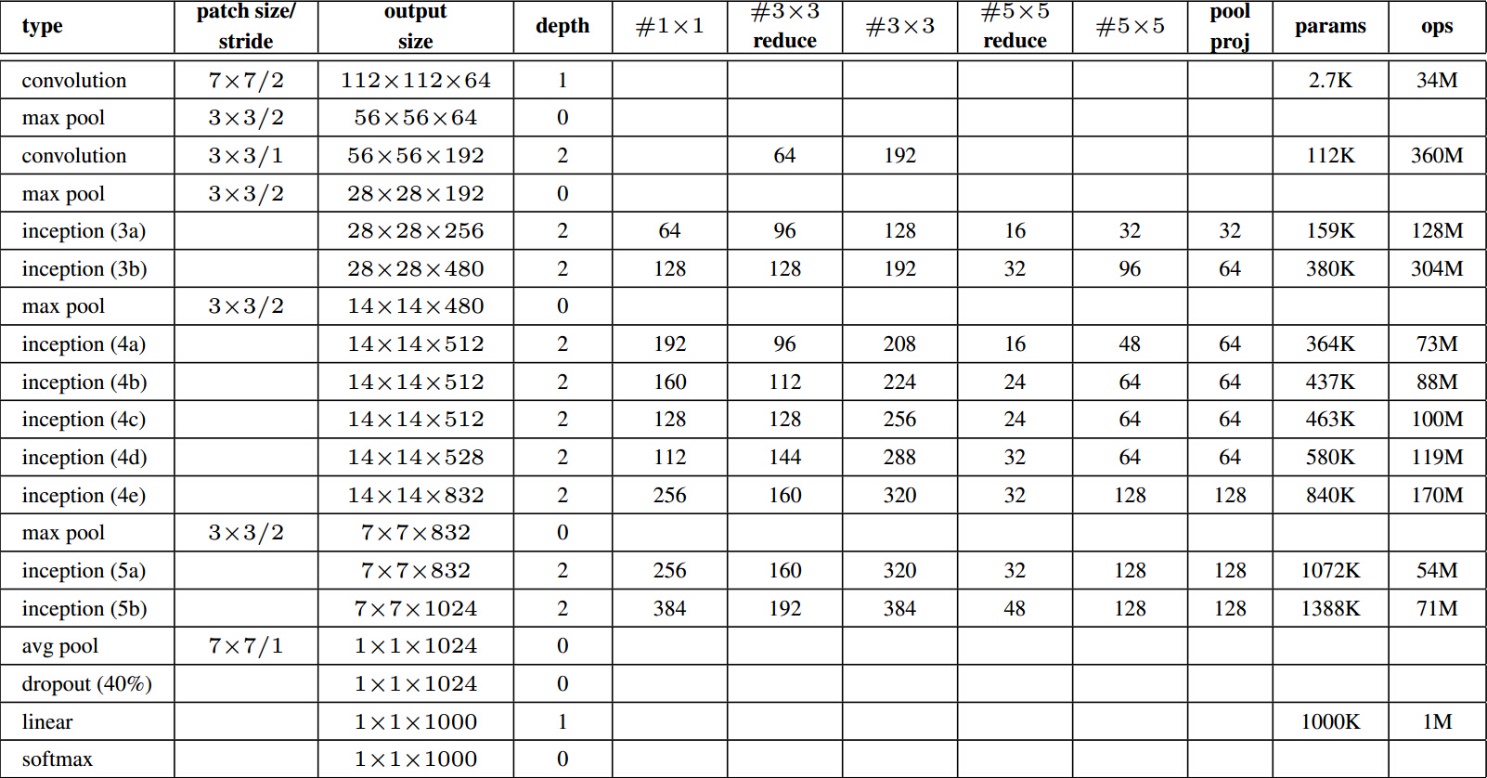
       3 . 文章说很多地方都表明pooling挺有效，所以Inception里面也嵌入了;

       4 . 网络越到后面，特征越抽象，而且每个特征所涉及的感受野也更大了，因此随着层数的增加，3x3和5x5卷积的比例也要增加。

       5. 使用5x5的卷积核仍然会带来巨大的计算量。 为此，文章借鉴NIN2，采用1x1卷积核来进行降维。 例如：假定上一层的输出为100x100x128，经过具有256个5x5卷积核的卷积层处理之后(stride=1，pad=2)，输出数据为100x100x256，其中，卷积层的参数为128x5x5x256。假如上一层输出先经过具有32个1x1卷积核的卷积层，再经过具有256个5x5卷积核的卷积层，那么最终的输出数据仍为为100x100x256，但卷积层参数量已经减少为128x1x1x32 + 32x5x5x256，大约减少了4倍。

       此外，该模型最后采用了average pooling来代替全连接层。但是，实际在最后还是加了一个全连接层，主要是为了方便以后大家finetune。

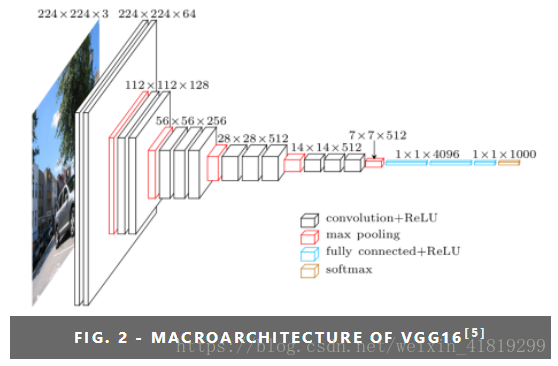
计算量如下图，可以看到参数总量并不大，但是计算次数是非常大的。



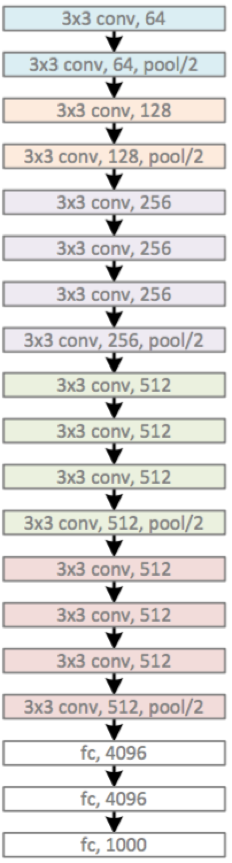
# VGG

**VGG-16层和VGG-19层**

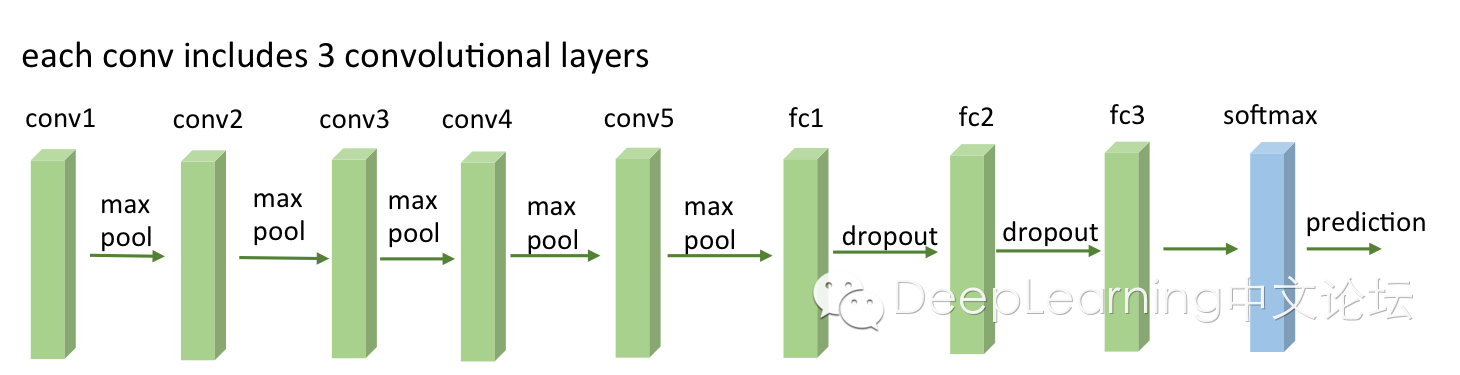
**VGG16模型（2+2+3+3+3个卷积层，3个全连接层）：**



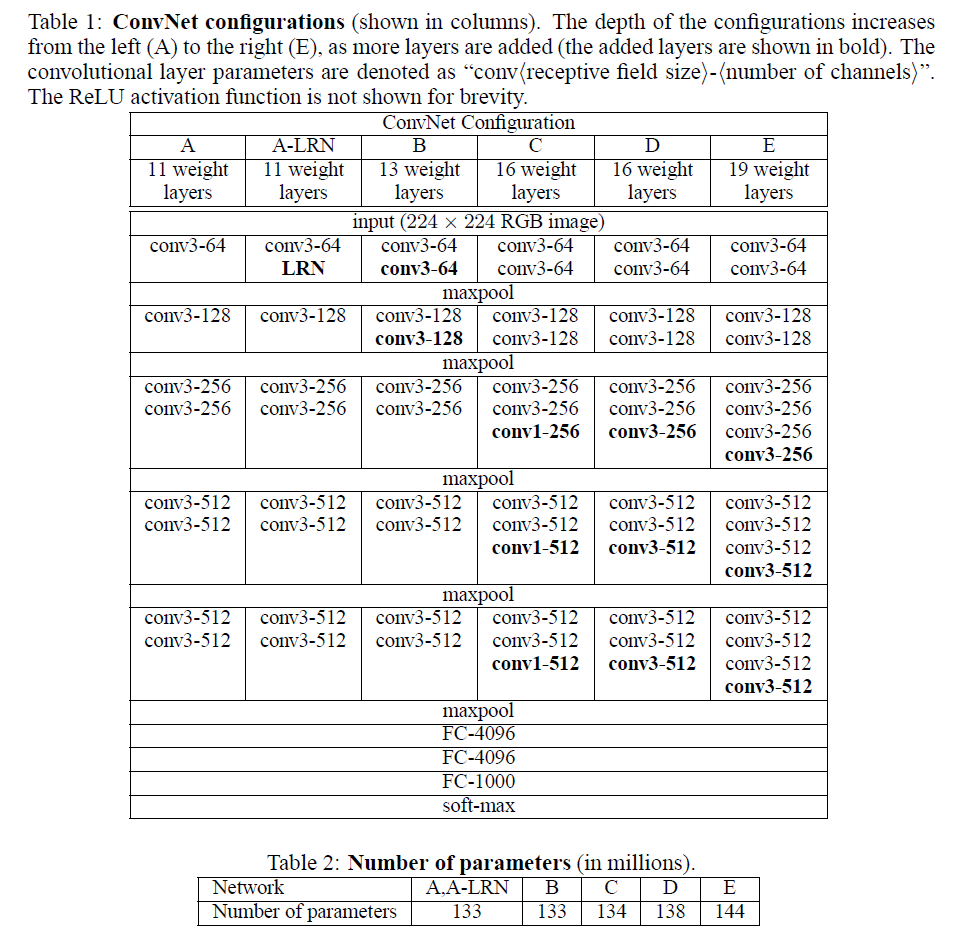
VGG19模型：



**VGG16层模型与AlexNet模型相比较：**







1：相较与AleNet,VGGNet最大的改进或者说区别就是用小size的Filter代替大size的Filter。两个3\*3的卷积层的有限感受野是5\*5（）如下图用两个3\*3的卷积核代替一个5\*5的卷积核）三个3\*3卷积层可以替代7\*7的卷积核

2：多个3\*3的卷基层比一个大尺寸filter卷基层有更多的非线性，增强泛化能力。

3 : 3\*3是最小的能够捕获上下左右和中心概念的尺寸，对于1\*1的卷积核的作用是在不影响输入输出维数的情况下，对输入线进行线性形变，然后通过Relu进行非线性处理，增强网络泛化能力

VGGNet是google和牛津大学一起研发的卷积神经网络

该网络结构有如下的特点：

(1)VGG全部使用3\*3卷积核、2\*2池化核，不断加深网络结构来提升性能。

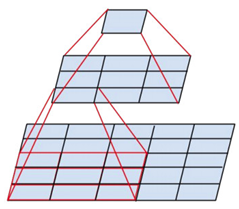
(2)A到E网络变深，参数量没有增长很多，参数量主要在3个全连接层。

(3)训练比较耗时的依然是卷积层，因计算量比较大。

(4)VGG有5段卷积，每段有2~3个卷积层，每段尾部用池化来缩小图片尺寸。

(5)每段内卷积核数一样，越靠后的段卷积核数越多：64–128–256–512–512。

之所以使用3\*3的卷积核堆叠起来是因为，这样可以扩大局部的感受野的同时又降低了参数的数量，三个3\*3的卷积层串联相当于1个7\*7的卷积层，但3个串联的3\*3的卷积层有更少的参数量，有更多的非线性变换（3次ReLU激活函数），使得CNN对特征的学习能力更强。



同时在训练的时候有一个trick就是先训练层数比较小的的简单网络，然后再复用这些简单网络的权重初始化后面几个复杂的网络，使得训练的收敛更加快。

而在预测的时候，使用multi-scale的方法，将要预测的数据scale到一个比原来数据更小的尺寸同时预测，然后再最后使用平均的分类结果，提高了预测的准确率和图片的利用率。btw，训练的时候也是用了multi-scale的方法做数据增强。

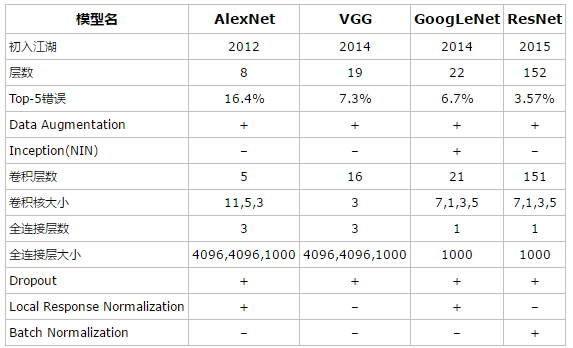
论文作者总结了几个观点包括：

1.LRN层的作用不大

2.越深的网络效果越好

3.大一些的卷积核可以学习到更大的空间特征

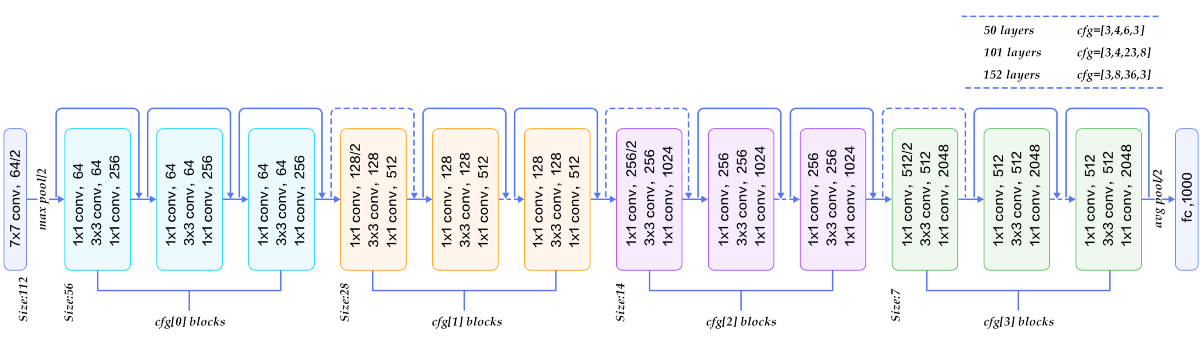
在我看来就是一个成本和效率的问题，为了降低训练的成本而使用了更小一点的卷积核（因为大的卷积核要训练的参数比较多），至于越深的网络越好，是因为有更多的非线性变换，在高维空间里可以学到低维空间无法区分的特征。

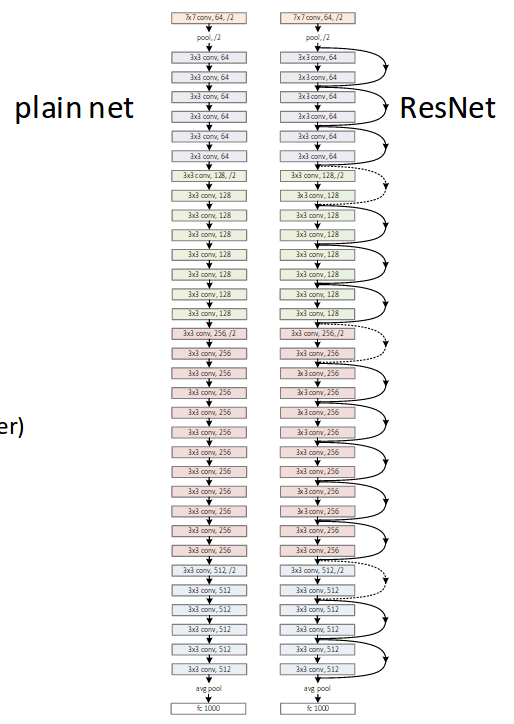


虽然VGGNet比AlexNet的参数要多，但反而需要更少的迭代次数就可以收敛，主要原因是更深的网络和更小的卷积核带来的隐式的正则化效果。

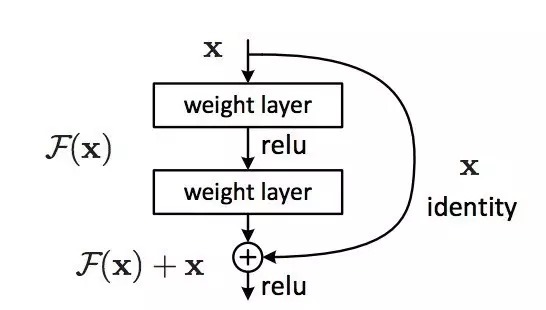
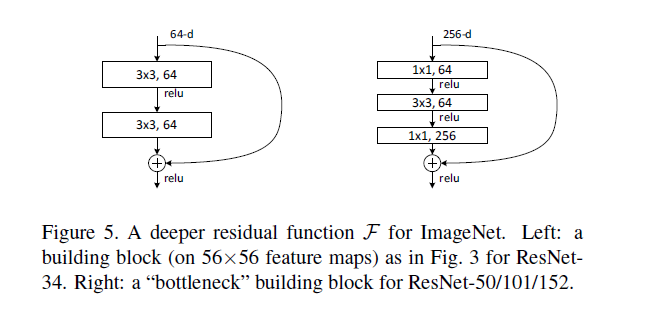
# ResNet

这个model是2015年底最新给出的，也是15年的imagenet比赛冠军。可以说是进一步将conv进行到底，其特殊之处在于设计了“bottleneck”形式的block（有跨越几层的直连）。最深的model采用的152层！！下面是一个34层的例子，更深的model见表格。





ResNet：没有最深，只有更深（152层），听说目前层数已突破一千。该模型要学习的是残差函数：F(x)=H(x)-x，残差块的结构如图所示：

实际中，考虑计算的成本，对残差块做了计算优化：将两个3x3的卷积层替换为1x1 + 3x3 + 1x1, 如上图所示。

**1×1卷积核作用**

（1） - 降维

　　通过控制卷积核的数量达到通道数大小的放缩。而池化层只能改变高度和宽度，无法改变通道数。

（2） - 增加非线性

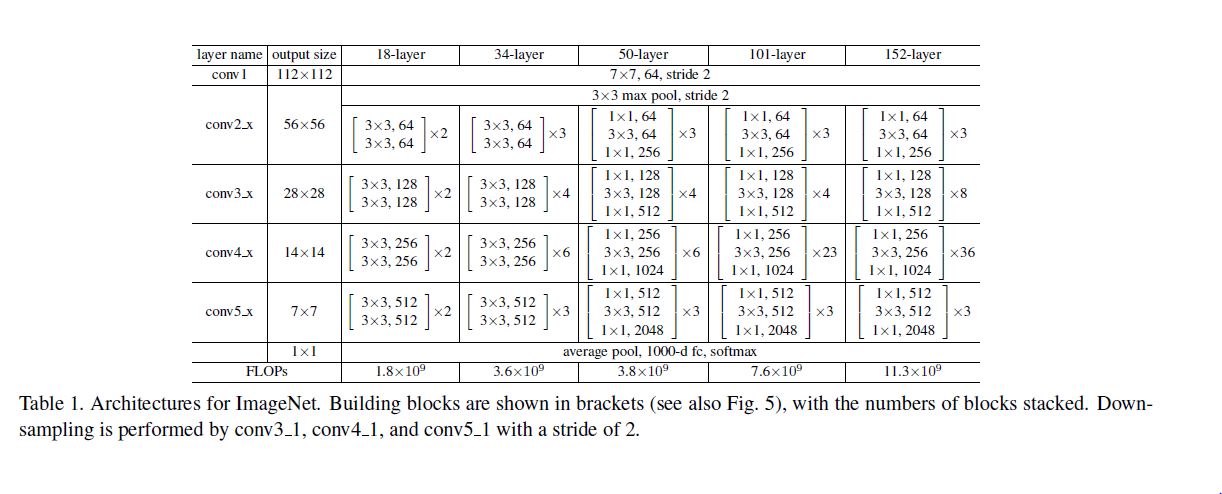
　　1×1卷积核的卷积过程相当于全连接层的计算过程，并且还加入了非线性激活函数relu，从而可以增加网络的非线性，使得网络可以表达更加复杂的特征。

（3） - 减少参数

　　在Inception Network中，由于需要进行较多的卷积运算，计算量很大，可以通过引入1×1确保效果的同时减少计算量。

<https://www.cnblogs.com/tianqizhi/p/9665436.html>

设计了“bottleneck”形式的block（有跨越几层的直连），block的构成见下图：



用全局平均池化GAP代替全连层FC，解决全连接层参数冗余的问题，但FC的优势在于在迁移学习中可改善微调的效果。

也就是说，即使我并不知道多少层是最佳，我通过残差模块，即使已经错过最佳深度我至少模型的精度不会有影响。起初看到这种网络模型很是奇怪，如果是一个浅层网络就能达到饱和，那么后面的残差结构目标是学习一个恒等映射，那么学习目标为F(x)接近为0。既然这样，为什么要去学习这个映射？直接写个恒等函数，或者直接设置F(x)=0 输出为x不就行了？残差网络的目的是学到y=x恒等映射函数，那么不就相当于加上的残差网络在最后没起到作用吗？那么为什么会有效呢？

首先这个饱和的浅层网络本身就不好寻找，有可能在达到饱和浅层网络深度之前，由于误差的原因模型精度已经下降。那么为什么持续增加层，让模型学习一个恒等映射就会使得模型表达变好呢？

进一步理解：

F是求和前网络映射，H是从输入到求和后的网络映射。比如把5映射到5.1，那么引入残差前是F'(5)=5.1，引入残差后是H(5)=5.1, H(5)=F(5)+5, F(5)=0.1。这里的F'和F都表示网络参数映射，引入残差后的映射对输出的变化更敏感。比如原来是从5.1到5.2，映射F'的输出增加了1/51=2%，而对于残差结构从5.1到5.2，映射F是从0.1到0.2，增加了100%。明显后者输出变化对权重的调整作用更大，所以效果更好。残差的思想都是去掉相同的主体部分，从而突出微小的变化。

理论解释：

<http://www.cnblogs.com/gczr/p/10127723.html>

<https://www.cnblogs.com/lulixin/p/10165478.html>

这也就是当网络模型我们已经设计到一定的深度，出现了精准度下降，如果使用残差结构就会很容易的调节到一个更好的效果，即使你不知道此刻的深度是不是最佳，但是起码准确度不会下降。代码实现也比较简单，原本的输出结果由F(x)替换为输出F(x)+X，如果维度相同则直接相加，如果维度不同则利用1\*1的卷积核变换。

# 作业

ResNeXt，2016

DenseNet，2017

**作业：学习DenseNet和ResNeXt模型，写学习报告（这两个模型的设计原理及创新点）**