吴恩达《卷积神经网络》精炼笔记（2）-- 深度卷积模型：案例研究

原创：红色石头 AI有道 2018-04-26

1 Why Look at Case Studies

本文将主要介绍几个典型的CNN案例。通过对具体CNN模型及案例的研究，来帮助我们理解知识并训练实际的模型。

典型的CNN模型包括：

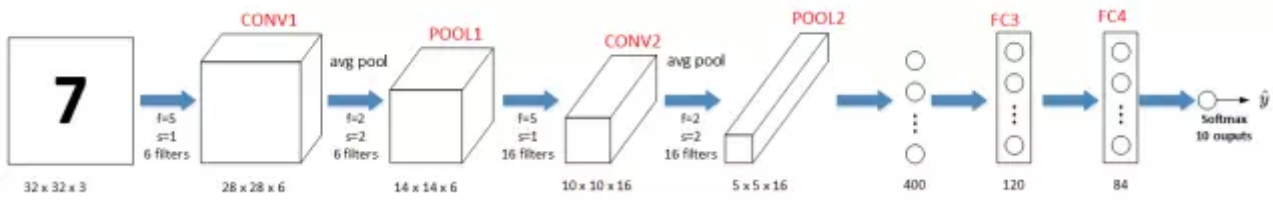
- **LeNet-5**
- **AlexNet**
- **VGG**

除了这些性能良好的CNN模型之外，我们还会介绍Residual Network（ResNet）。其特点是可以构建很深很深的神经网络（目前最深的好像有152层）。

另外，还会介绍Inception Neural Network。接下来，我们将一一讲解。

2 Classic Networks

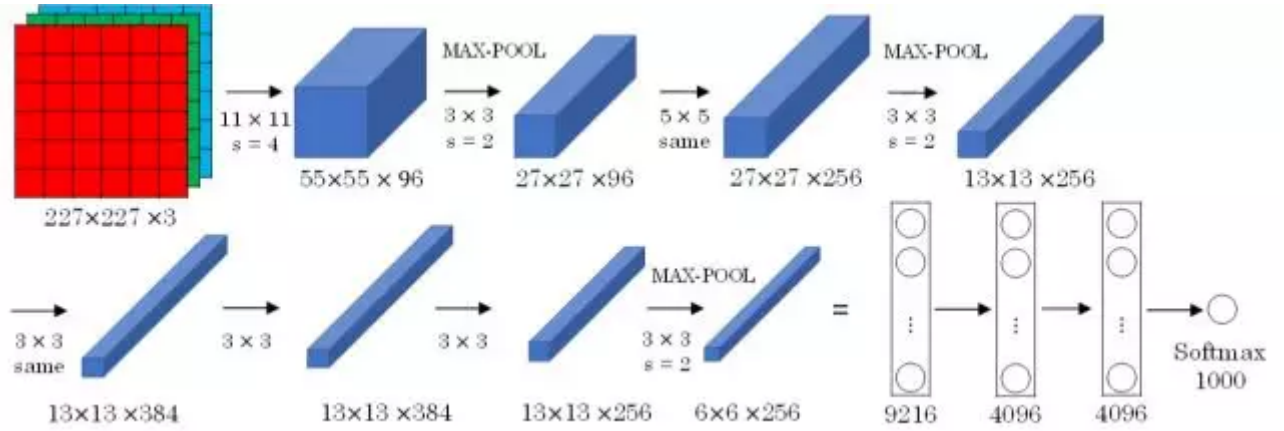
LeNet-5模型是Yann LeCun教授于1998年提出来的，它是第一个成功应用于数字识别问题的卷积神经网络。在MNIST数据中，它的准确率达到大约99.2%。典型的LeNet-5结构包含CONV layer，POOL layer和FC layer，顺序一般是CONV layer->POOL layer->CONV layer->POOL layer->FC layer->FC layer->OUTPUT layer，即y^y^。下图所示的是一个数字识别的LeNet-5的模型结构：



该LeNet模型总共包含了大约6万个参数。值得一提的是，当时Yann LeCun提出的LeNet-5模型池化层使用的是average pool，而且各层激活函数一般是Sigmoid和tanh。现在，我们可以根据需

要，做出改进，使用max pool和激活函数ReLU。

AlexNet模型是由Alex Krizhevsky、Ilya Sutskever和Geoffrey Hinton共同提出的，其结构如下所示：

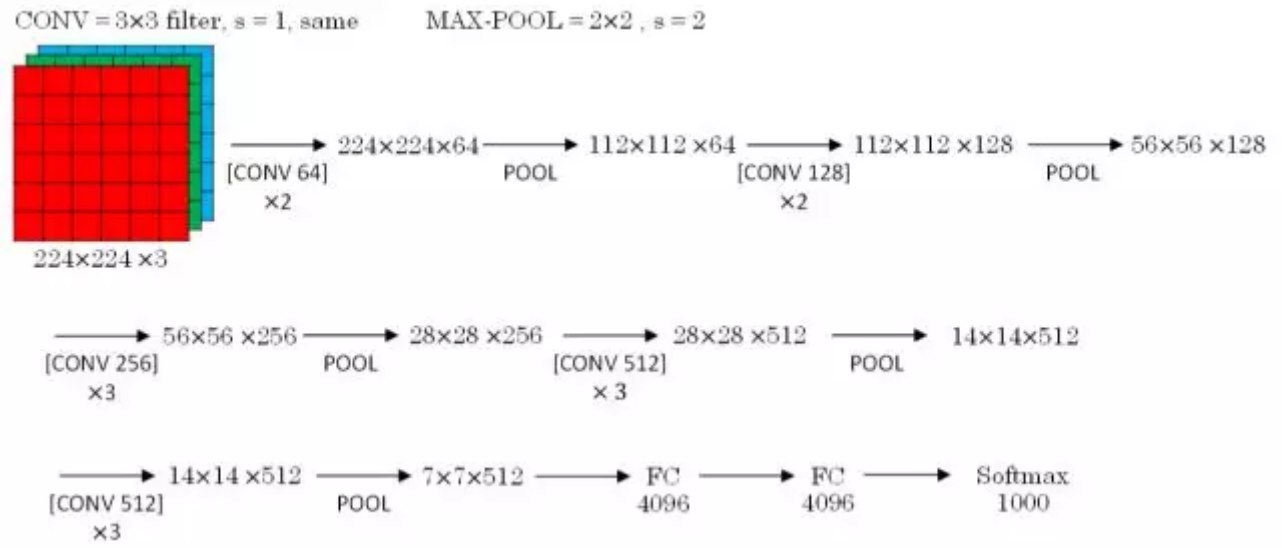


AlexNet模型与LeNet-5模型类似，只是要复杂一些，总共包含了大约6千万个参数。同样可以根据实际情况使用激活函数ReLU。原作者还提到了一种优化技巧，叫做Local Response Normalization(LRN)。而在实际应用中，LRN的效果并不突出。

VGG-16模型更加复杂一些，一般情况下，其CONV layer和POOL layer设置如下：

- **CONV = 3x3 filters, s = 1, same**
- **MAX-POOL = 2x2, s = 2**

VGG-16结构如下所示：

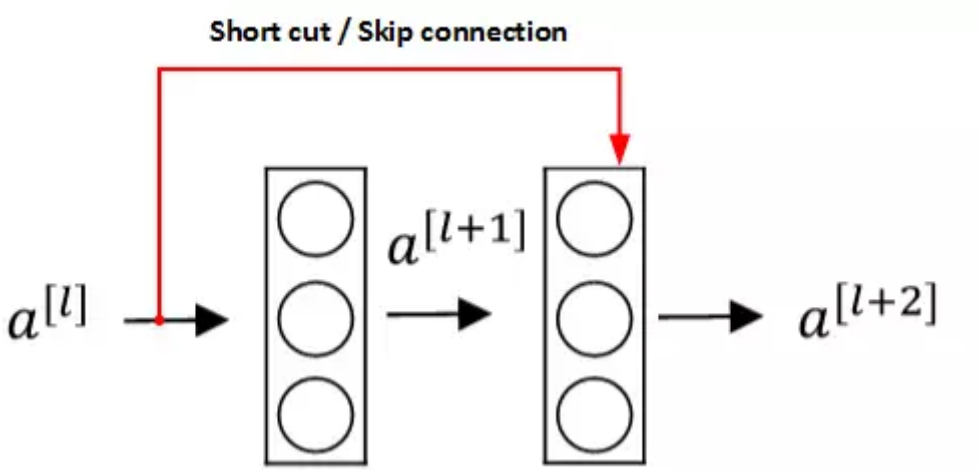


VGG-16的参数多达1亿3千万。

3 ResNets

我们知道，如果神经网络层数越多，网络越深，源于梯度消失和梯度爆炸的影响，整个模型难以训练成功。解决的方法之一是人为地让神经网络某些层跳过下一层神经元的连接，隔层相连，弱化每层之间的强联系。这种神经网络被称为Residual Networks(ResNets)。

Residual Networks由许多隔层相连的神经元子模块组成，我们称之为Residual block。单个Residual block的结构如下图所示：



上图中红色部分就是skip connection，直接建立$a^{[l]}$与$a^{[l+2]}$之间的隔层联系。相应的表达式如下：

$$z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]}$$

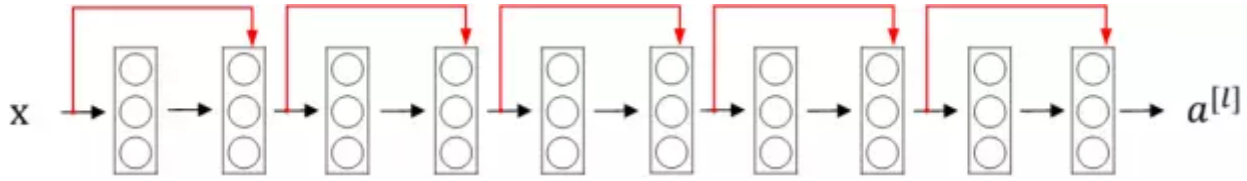
$$a^{[l+1]} = g(z^{[l+1]})$$

$$z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]}$$

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

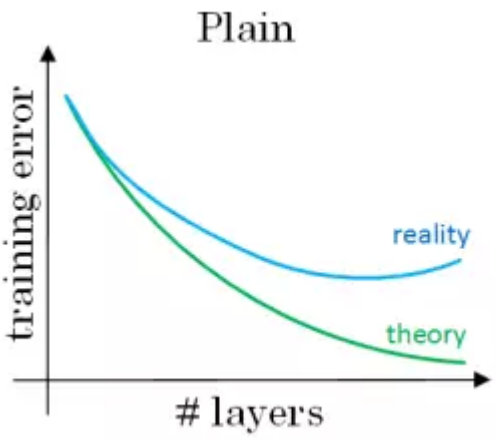
$a^{[l]}$直接隔层与下一层的线性输出相连，与$z^{[l+2]}$共同通过激活函数（ReLU）输出$a^{[l+2]}$。

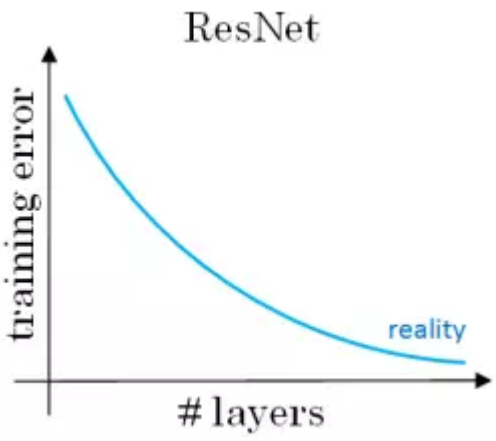
该模型由Kaiming He, Xiangyu Zhang, Shaoqing Ren和Jian Sun共同提出。由多个Residual block组成的神经网络就是Residual Network。实验表明，这种模型结构对于训练非常深的神经网络，效果很好。另外，为了便于区分，我们把非Residual Networks称为Plain Network。



Residual Network的结构如上图所示。

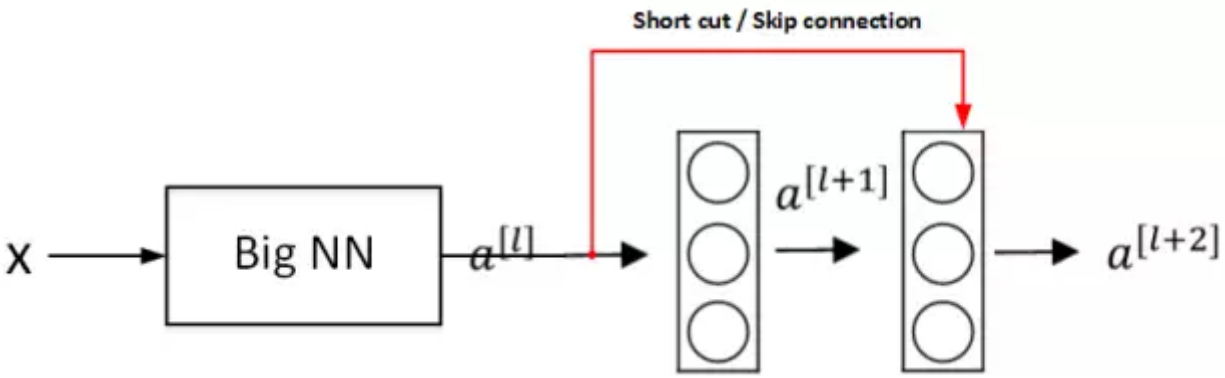
与Plain Network相比，Residual Network能够训练更深层的神经网络，有效避免发生发生梯度消失和梯度爆炸。从下面两张图的对比中可以看出，随着神经网络层数增加，Plain Network实际性能会变差，training error甚至会变大。然而，Residual Network的训练效果却很好，training error一直呈下降趋势。





4 Why ResNets Work

下面用个例子来解释为什么ResNets能够训练更深层的神经网络。



如上图所示，输入x经过很多层神经网络后输出$a^{[l]}$，$a^{[l]}$经过一个Residual block输出$a^{[l+2]}$。$a^{[l+2]}$的表达式为：

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]}) = g(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]})$$

输入x经过Big NN后，若$W^{[l+2]} \approx 0$，$b^{[l+2]} \approx 0$，则有：

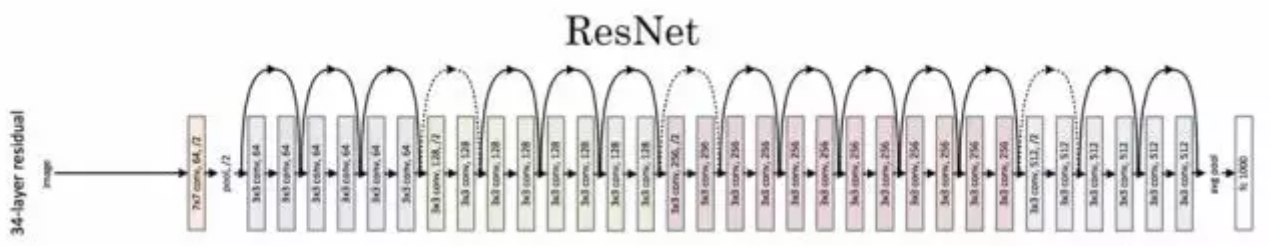
$$a^{[l+2]} = g(a^{[l]}) = ReLU(a^{[l]}) = a^{[l]} \quad when\ a^{[l]} \geq 0$$

可以看出，即使发生了梯度消失，$W^{[l+2]} \approx 0$，$b^{[l+2]} \approx 0$，也能直接建立$a^{[l+2]}$与$a^{[l]}$的线性关系，且$a^{[l+2]} = a^{[l]}$，这其实就是identity function。$a^{[l]}$直接连到$a^{[l+2]}$，从效果来说，相当于直接忽略了$a^{[l]}$之后的这两层神经层。这样，看似很深的神经网络，其实由于许多Residual blocks的存在，弱化削减了某些神经层之间的联系，实现隔层线性传递，而不是一味追求非线性关系，模型本身也就能"容忍"更深层的神经网络了。而且从性能上来说，这两层额外的Residual blocks也不会降低Big NN的性能。

当然，如果Residual blocks确实能训练得到非线性关系，那么也会忽略short cut，跟Plain Network起到同样的效果。

有一点需要注意的是，如果Residual blocks中$a^{[l]}$和$a^{[l+2]}$的维度不同，通常可以引入矩阵W_s，与$a^{[l]}$相乘，使得$W_s * a^{[l]}$的维度与$a^{[l+2]}$一致。参数矩阵W_s有来两种方法得到：一种是将W_s作为学习参数，通过模型训练得到；另一种是固定W_s值（类似单位矩阵），不需要训练，W_s与$a^{[l]}$的乘积仅仅使得$a^{[l]}$截断或者补零。这两种方法都可行。

下图所示的是CNN中ResNets的结构：



ResNets同类型层之间，例如CONV layers，大多使用same类型，保持维度相同。如果是不同类型层之间的连接，例如CONV layer与POOL layer之间，如果维度不同，则引入矩阵Ws。

5 Networks in Networks and 1x1 Convolutions

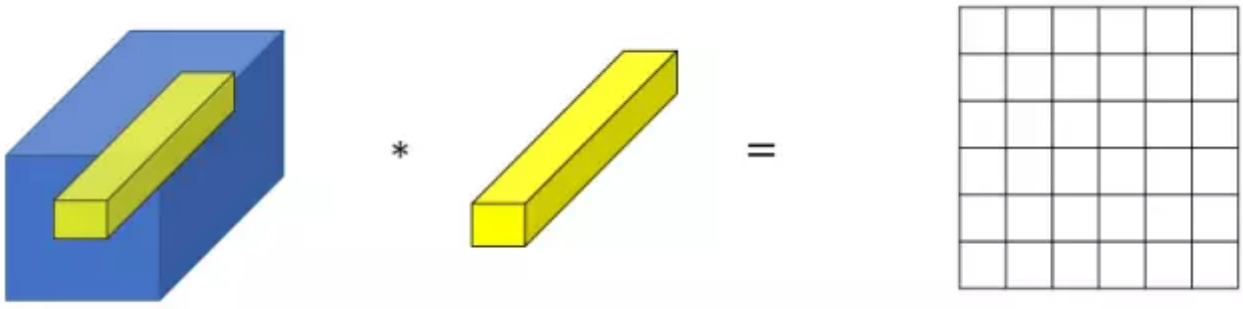
Min Lin, Qiang Chen等人提出了一种新的CNN结构，即1x1 Convolutions，也称Networks in Networks。这种结构的特点是滤波器算子filter的维度为1x1。对于单个filter，1x1的维度，意味着卷积操作等同于乘积操作。

1	2	3	6	5	8
3	5	5	1	3	4
2	1	3	4	9	3
4	7	8	5	7	9
1	5	3	7	4	8
5	4	9	8	3	5

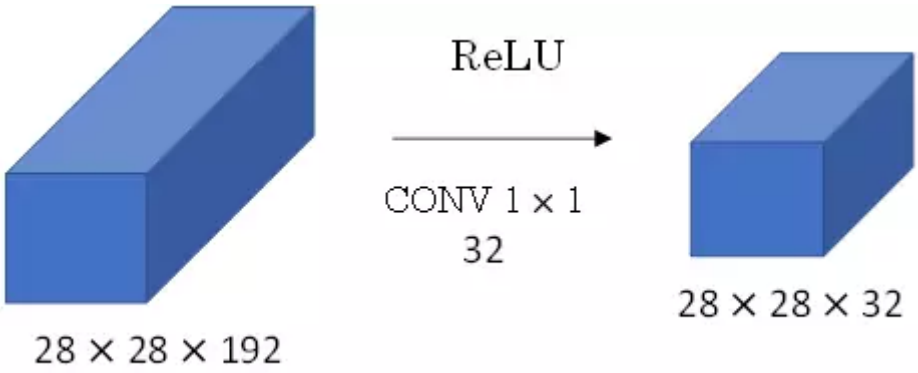
* [1] =

1	2	3	6	5	8
3	5	5	1	3	4
2	1	3	4	9	3
4	7	8	5	7	9
1	5	3	7	4	8
5	4	9	8	3	5

那么，对于多个filters，1x1 Convolutions的作用实际上类似全连接层的神经网络结构。效果等同于Plain Network中$a^{[l]}$到$a^{[l+1]}$的过程。这点还是比较好理解的。

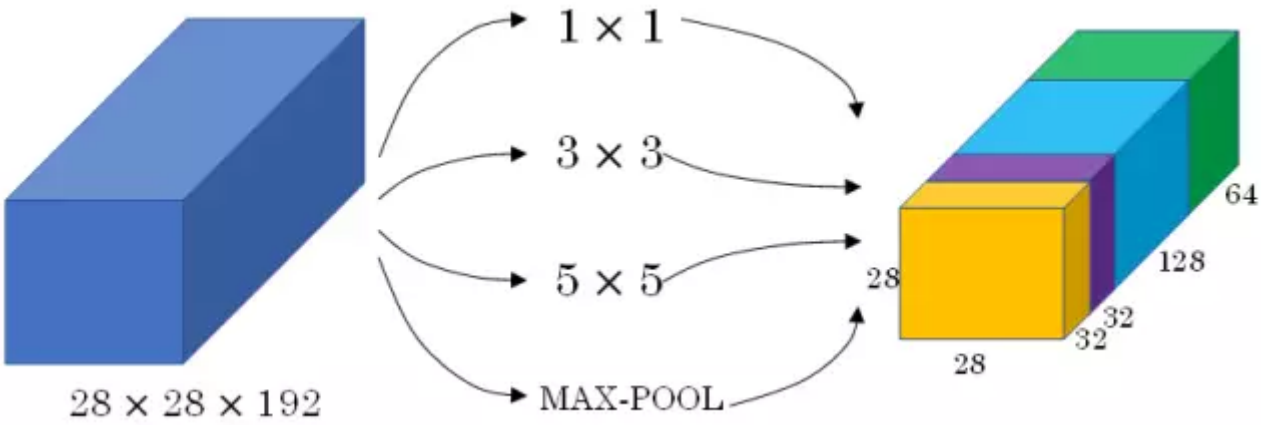


1x1 Convolutions可以用来缩减输入图片的通道数目。方法如下图所示：



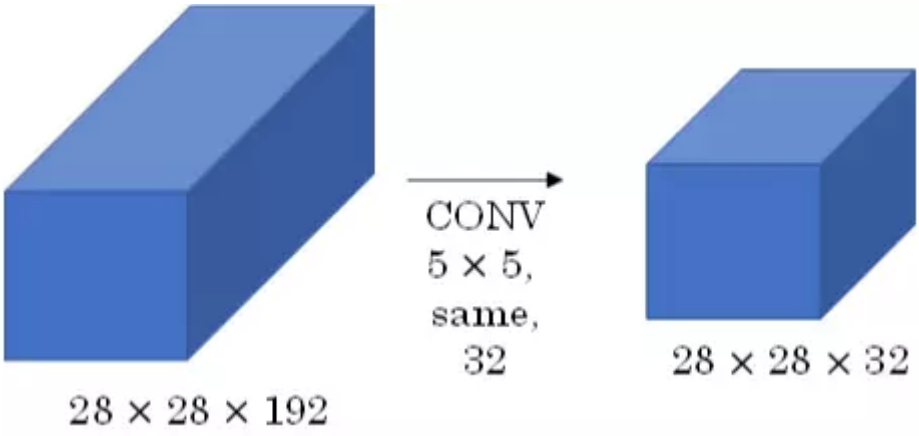
6 Inception Network Motivation

之前我们介绍的CNN单层的滤波算子filter尺寸是固定的，1x1或者3x3等。而Inception Network在单层网络上可以使用多个不同尺寸的filters，进行same convolutions，把各filter下得到的输出拼接起来。除此之外，还可以将CONV layer与POOL layer混合，同时实现各种效果。但是要注意使用same pool。

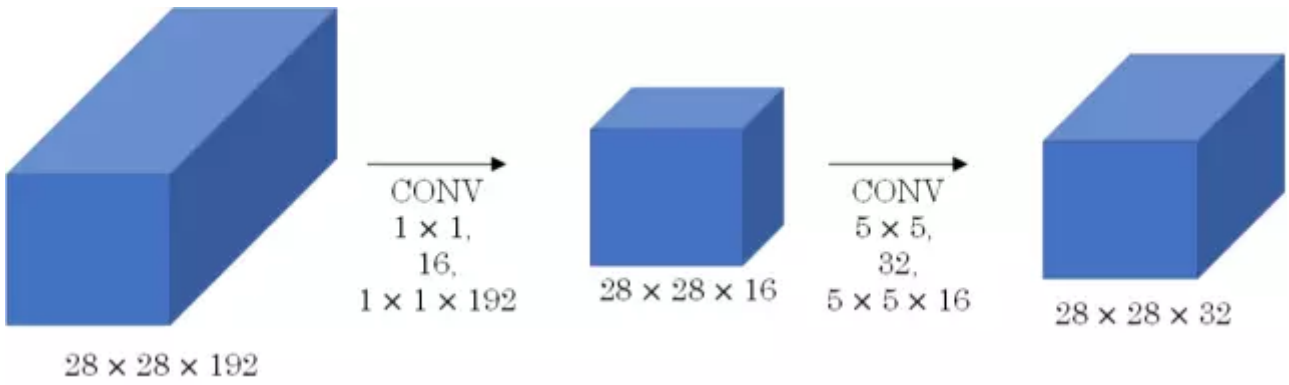


Inception Network由Christian Szegedy, Wei Liu等人提出。与其它只选择单一尺寸和功能的filter不同，Inception Network使用不同尺寸的filters并将CONV和POOL混合起来，将所有功能输出组合拼接，再由神经网络本身去学习参数并选择最好的模块。

Inception Network在提升性能的同时，会带来计算量大的问题。例如下面这个例子：



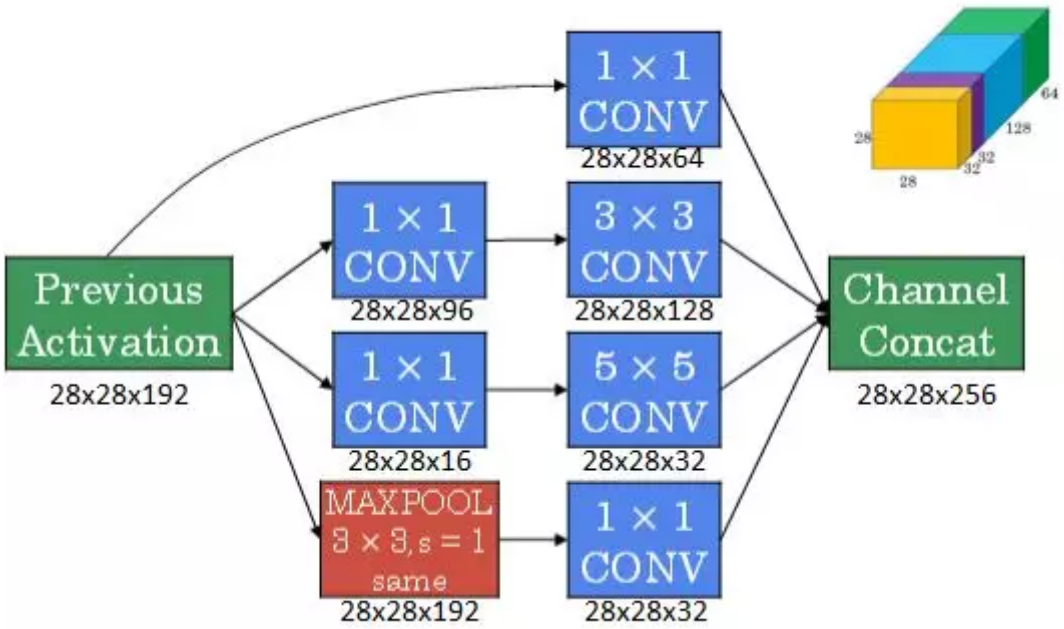
此CONV layer需要的计算量为：28x28x32x5x5x192=120m，其中m表示百万单位。可以看出但这一层的计算量都是很大的。为此，我们可以引入1x1 Convolutions来减少其计算量，结构如下图所示：



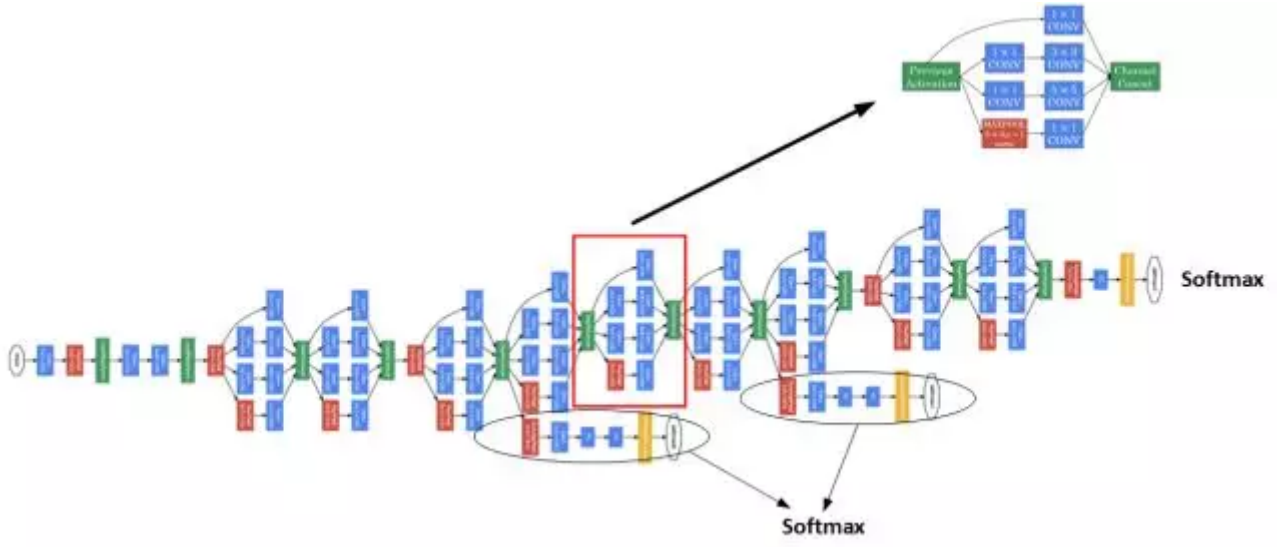
通常我们把该1x1 Convolution称为“瓶颈层”（bottleneck layer）。引入bottleneck layer之后，总共需要的计算量为：28x28x16x192+28x28x32x5x5x16=12.4m。明显地，虽然多引入了1x1 Convolution层，但是总共的计算量减少了近90%，效果还是非常明显的。由此可见，1x1 Convolutions还可以有效减少CONV layer的计算量。

7 Inception Network

上一节我们使用1x1 Convolution来减少Inception Network计算量大的问题。引入1x1 Convolution后的Inception module如下图所示：



多个Inception modules组成Inception Network，效果如下图所示：



上述Inception Network除了由许多Inception modules组成之外，值得一提的是网络中间隐藏层也可以作为输出层Softmax，有利于防止发生过拟合。

8 Using Open-Source Implementation

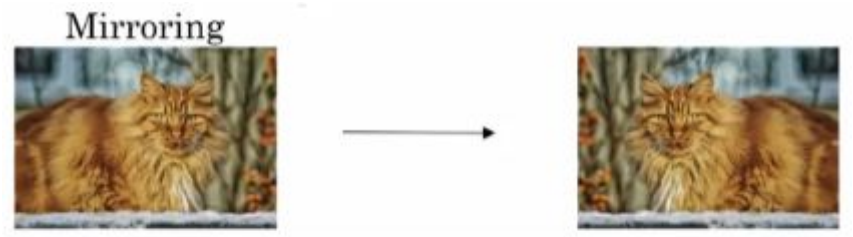
本节主要介绍GitHub的使用，GitHub是一个面向开源及私有软件项目的托管平台，上面包含有许多优秀的CNN开源项目。关于GitHub具体内容不再介绍，有兴趣的小伙伴自行查阅。

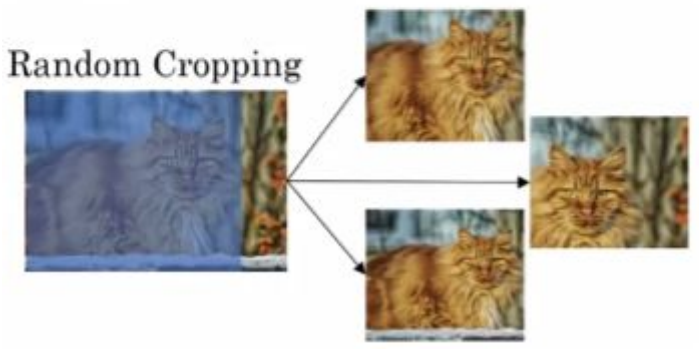
9 Transfer Learning

有关Transfer Learning的相关内容，我们在**吴恩达《构建机器学习项目》精炼笔记（2）– 机器学习策略（下）**中已经详细介绍过，这里就不再赘述了。

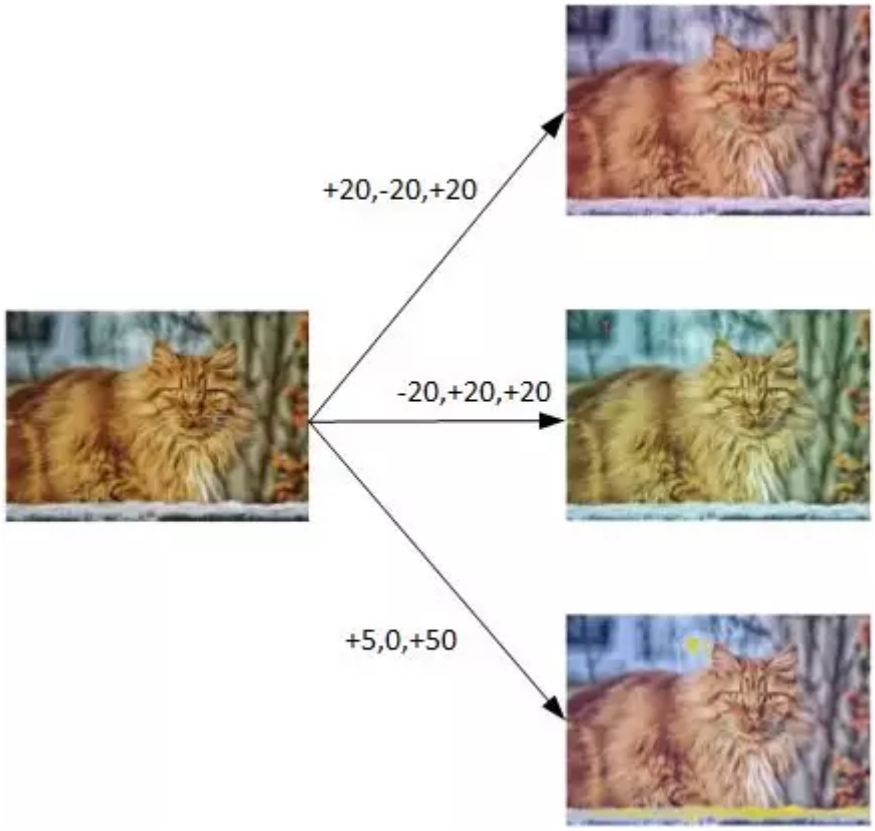
10 Data Augmentation

常用的Data Augmentation方法是对已有的样本集进行Mirroring和Random Cropping。





另一种Data Augmentation的方法是color shifting。color shifting就是对图片的RGB通道数值进行随意增加或者减少，改变图片色调。

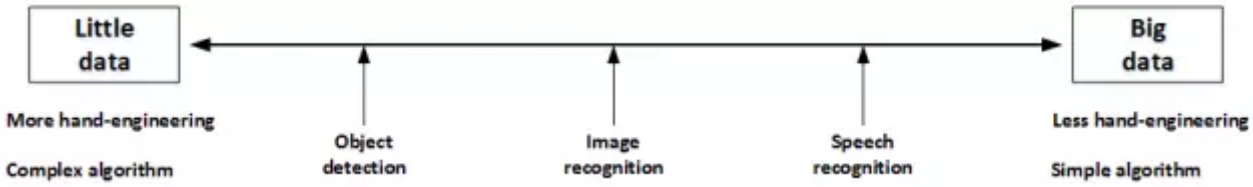


除了随意改变RGB通道数值外，还可以更有针对性地对图片的RGB通道进行PCA color augmentation，也就是对图片颜色进行主成分分析，对主要的通道颜色进行增加或减少，可以采用高斯扰动做法。这样也能增加有效的样本数量。具体的PCA color augmentation做法可以查阅AlexNet的相关论文。

最后提一下，在构建大型神经网络的时候，data augmentation和training可以由两个不同的线程来进行。

11 State of Computer Vision

神经网络需要数据，不同的网络模型所需的数据量是不同的。Object dection，Image recognition，Speech recognition所需的数据量依次增加。一般来说，如果data较少，那么就需要更多的hand-engineering，对已有data进行处理，比如上一节介绍的data augmentation。模型算法也会相对要复杂一些。如果data很多，可以构建深层神经网络，不需要太多的hand-engineering，模型算法也就相对简单一些。



值得一提的是hand-engineering是一项非常重要也比较困难的工作。很多时候，hand-engineering对模型训练效果影响很大，特别是在数据量不多的情况下。

在模型研究或者竞赛方面，有一些方法能够有助于提升神经网络模型的性能：

- **Ensembling: Train several networks independently and average their outputs.**
- **Multi-crop at test time: Run classifier on multiple versions of test images and average results.**

但是由于这两种方法计算成本较大，一般不适用于实际项目开发。

最后，我们还要灵活使用开源代码：

- **Use architectures of networks published in the literature**
- **Use open source implementations if possible**
- **Use pretrained models and fine-tune on your dataset**