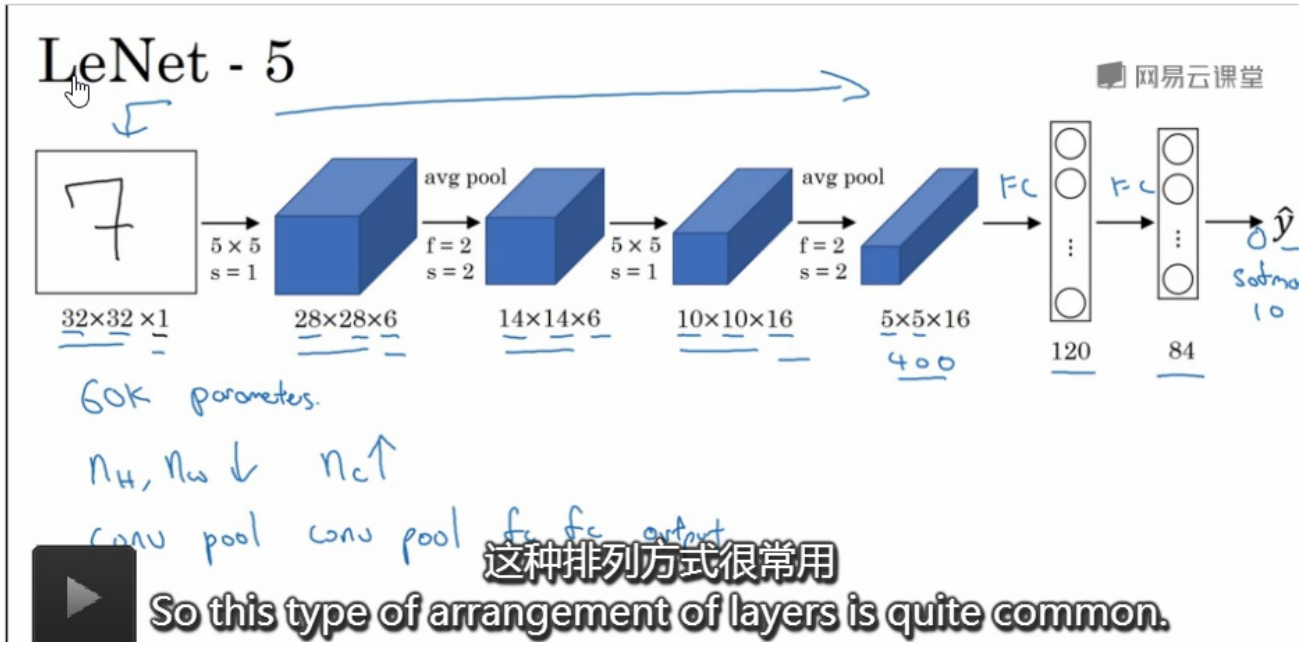
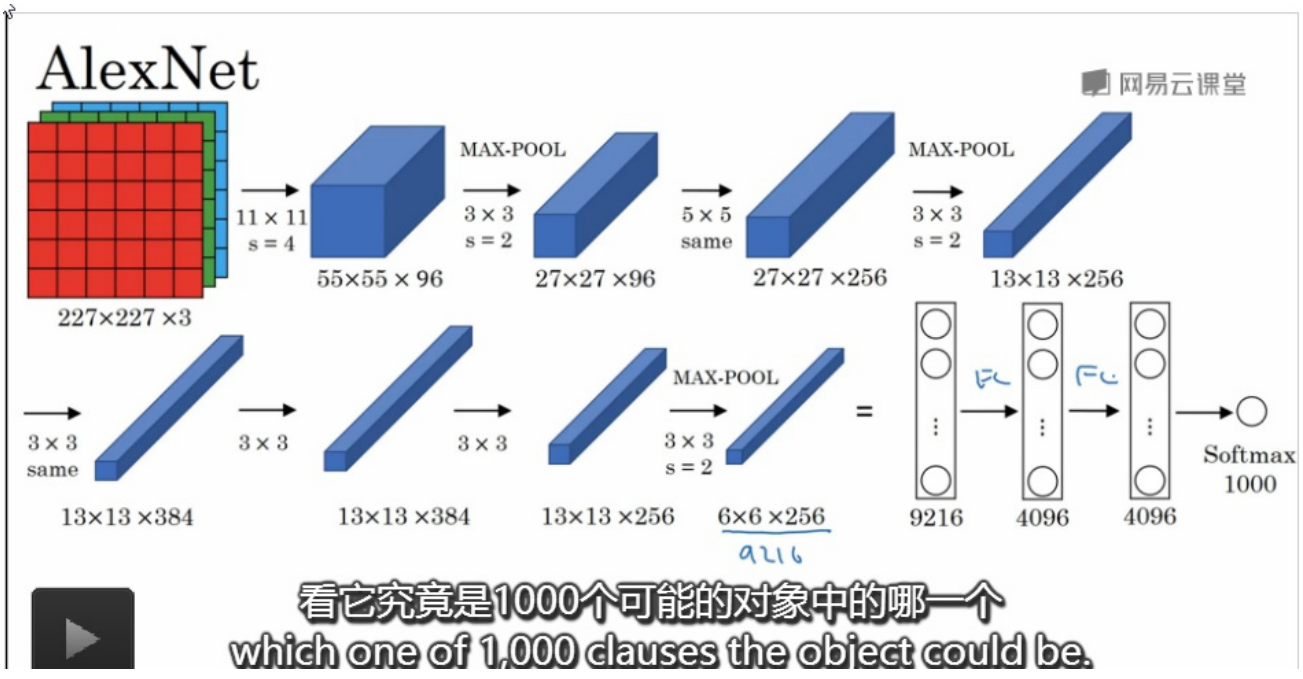
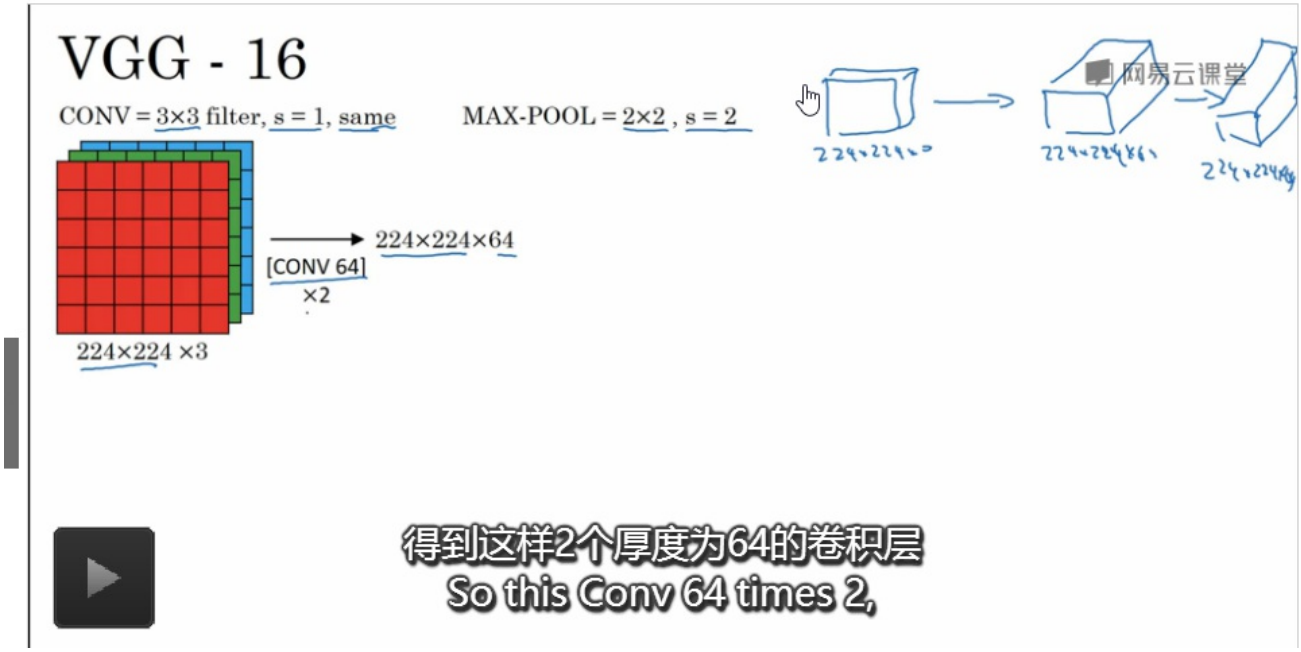
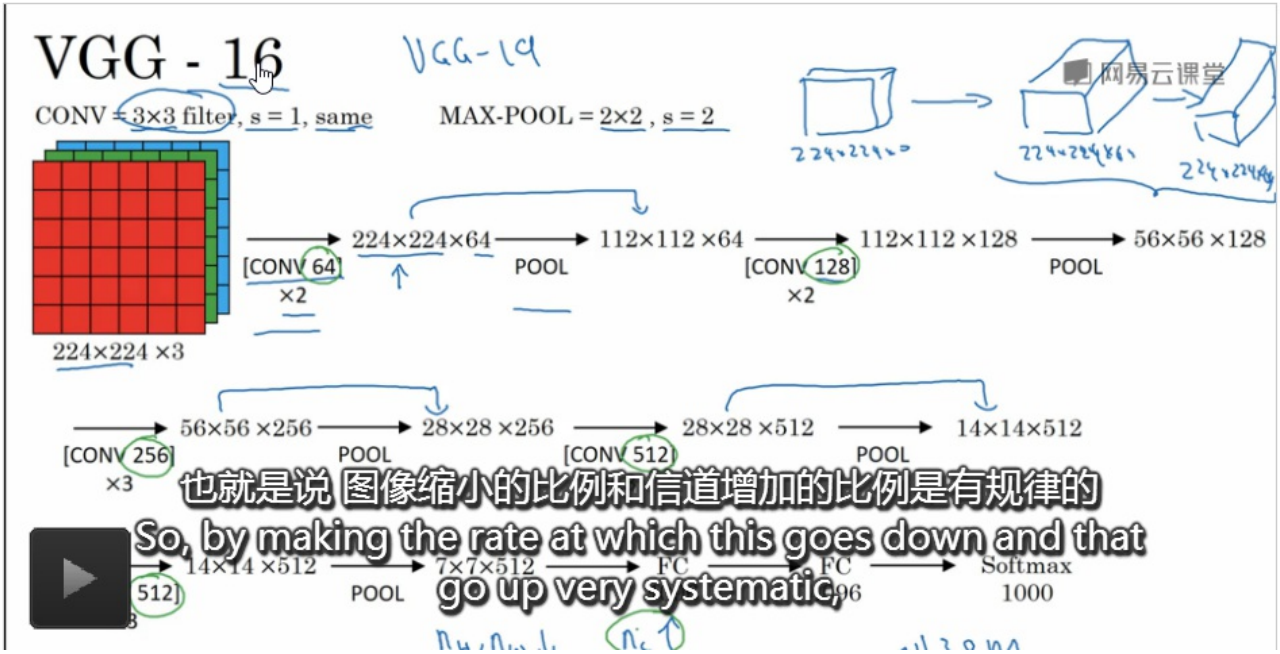
第四篇第二周


LeNet - 5
网易云课堂
5 × 5
s = 1
avg pool
f = 2
s = 2
5 × 5
s = 1
avg pool
f = 2
s = 2
FC
FC
32×32 ×1
28×28×6
14×14×6
10×10×16
5×5×16
120
84
60K parameters.
conv pool conv pool fc fc output
这种排列方式很常用
So this type of arrangement of layers is quite common.


AlexNet
网易云课堂
11 × 11
s = 4
MAX-POOL
3 × 3
s = 2
5 × 5
same
MAX-POOL
3 × 3
s = 2
227×227 ×3
55×55 × 96
27×27 ×96
27×27 ×256
13×13 ×256
3 × 3
same
3 × 3
3 × 3
MAX-POOL
3 × 3
s = 2
=
FC
FC
Softmax
1000
13×13 ×384
13×13 ×384
13×13 ×256
6×6 ×256
9216
9216
4096
4096
看它究竟是1000个可能的对象中的哪一个
which one of 1,000 clauses the object could be.

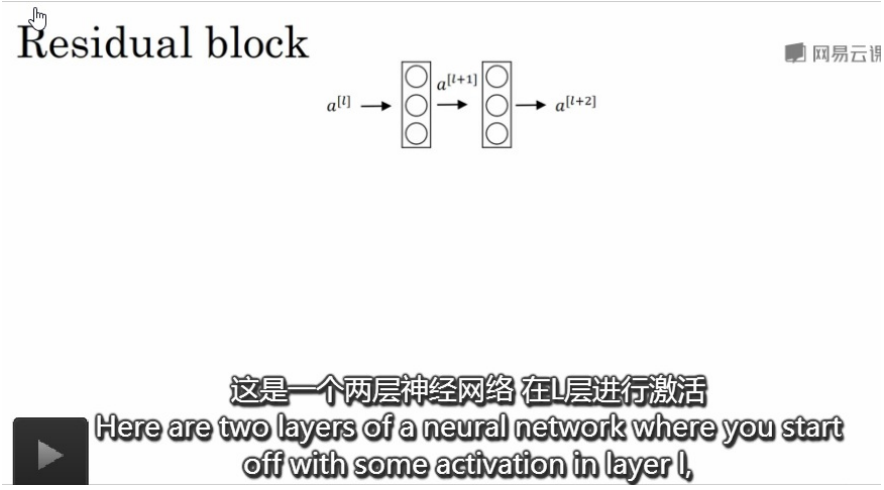




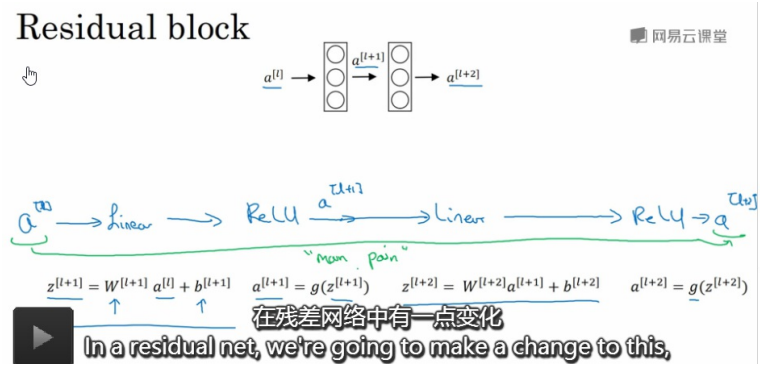
3、残差网络

非常非常深的神经网络是非常难训练的，因为存在梯度爆炸或者梯度消失问题，接下来我们学习跳远连接，它可以从某一网络层获得激活，然后迅速反馈给另外一层甚至神经网络的更深层，我们可以利用跳远连接构建能够训练深度网络的resNet。

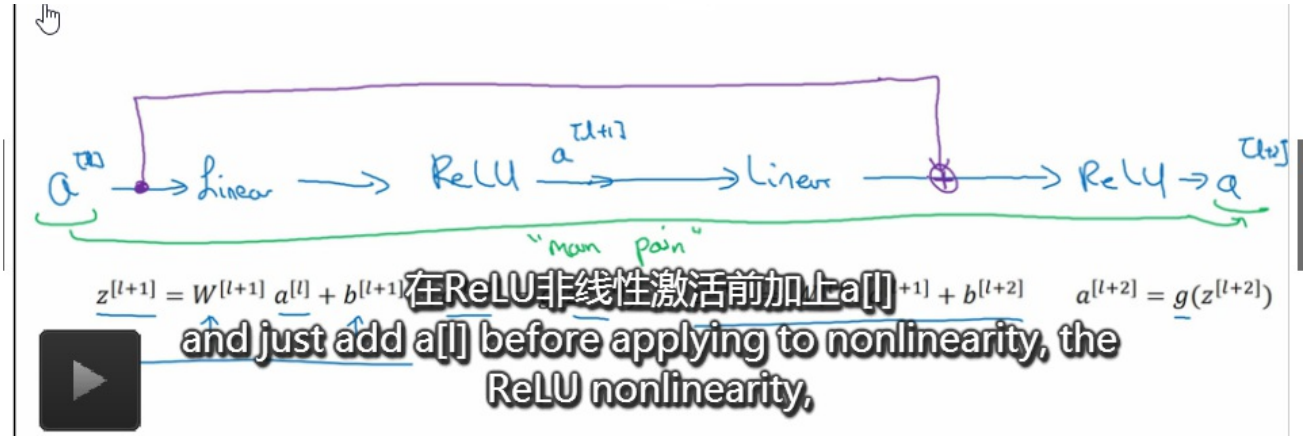
resNet是由残差块构建的，什么是残差块呢？



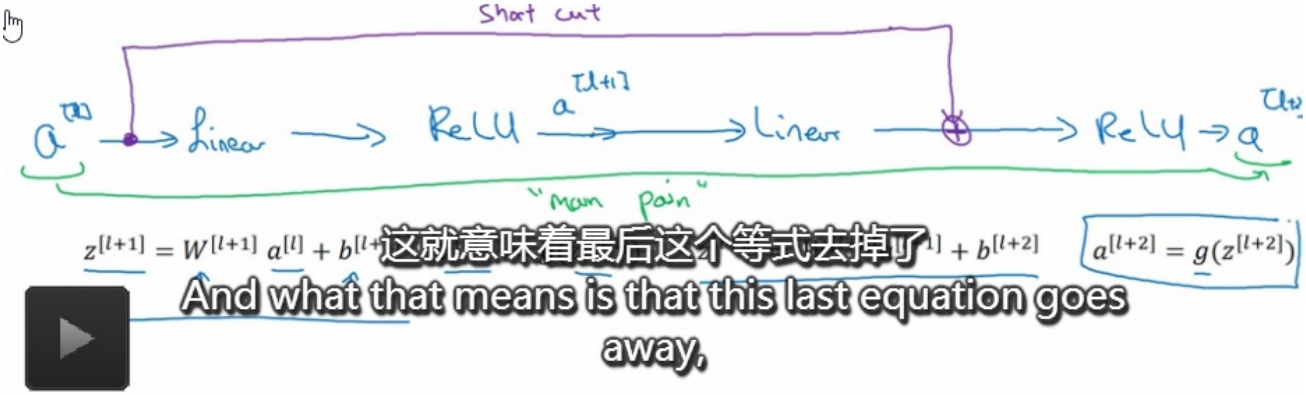
计算过程是从a[l]开始，首先进行线性激活，然后通过relu激活函数得到a[l+1],接下来再次进行线性激活，然后再次使用relu激活函数得到a[l+2]，换句话说，信息流从a[l]流到a[l+2],需要经过上面的所有步骤，即这个网络的主路径：



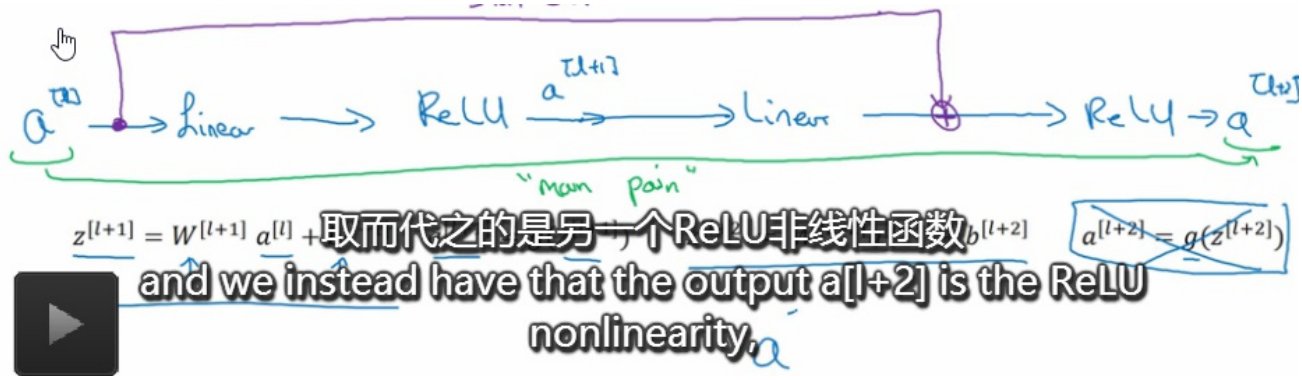
在残差网络中有一点变化，我们将a[l]直接向后拷贝到神经网络的深层，在relu激活函数前加一个a[l],



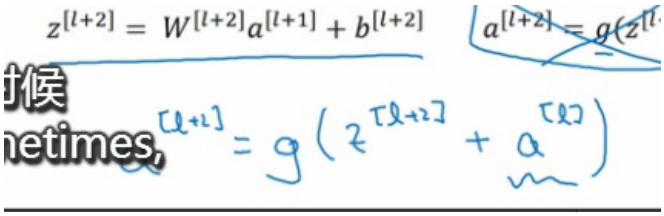
这是一条捷径，a[l]的信息直接到达神经网络的更深层，不再沿着主路径传递，



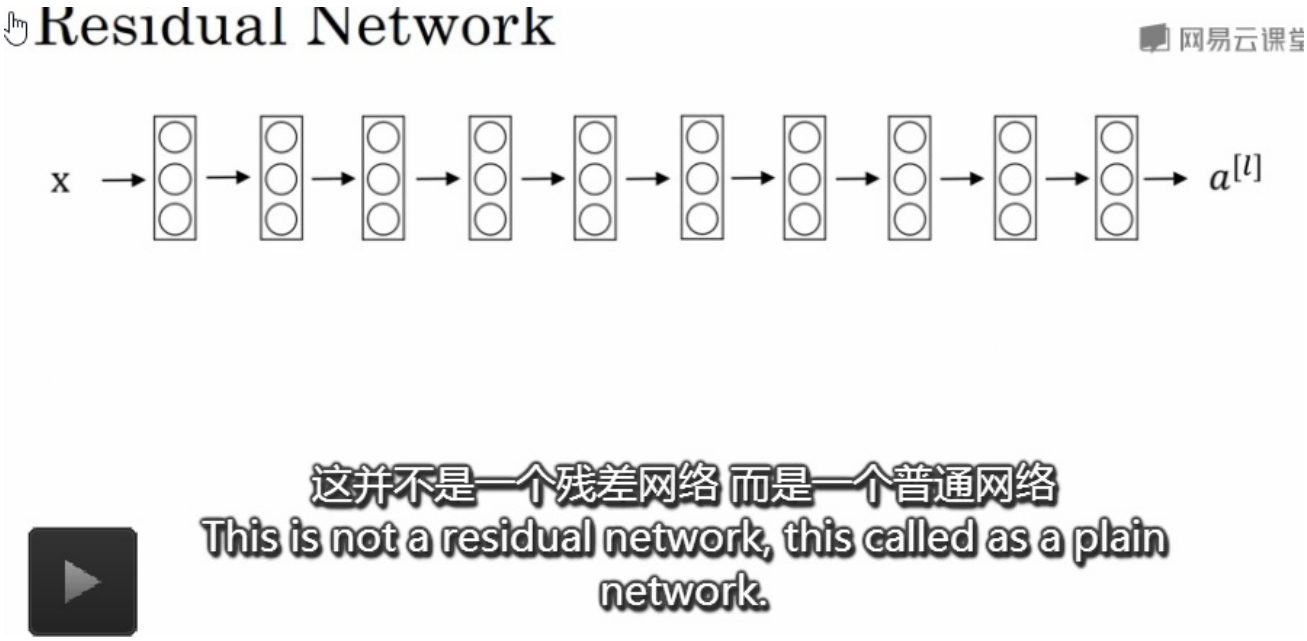
取而代之的是另外一个relu的非线性函数，



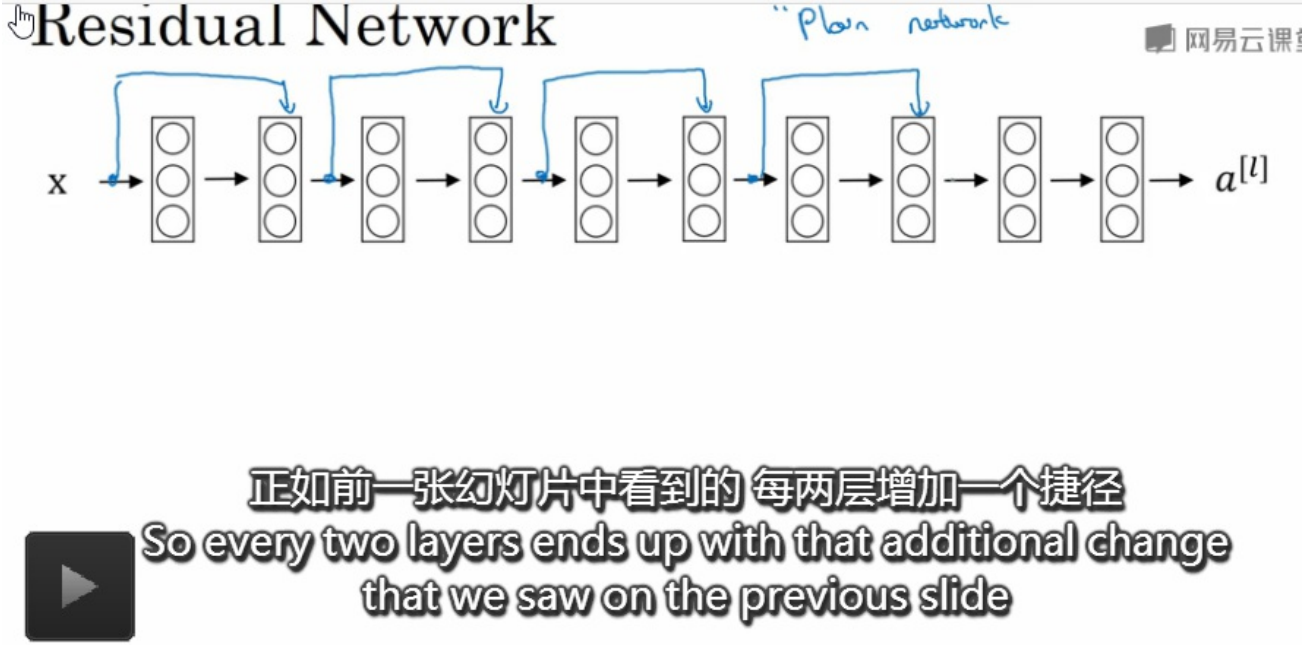
仍然会z[l+2]进行z函数处理，但这次要加上a[l]。也就是加上的这个a[l]产生的一个残差块，实际上这条捷径是在进行Relu非线性激活之前加上的，而这里的每一个节点都执行了非线性函数和relu激活函数,所以a[l]插入的时机是在非线性激活之后relu激活之前。



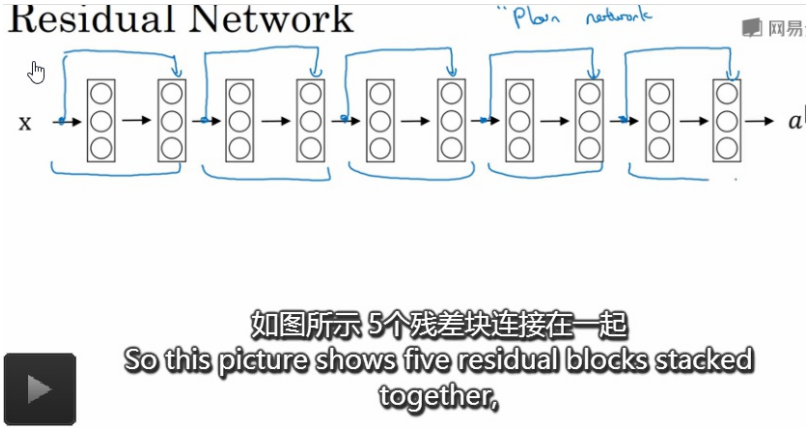
也可以叫做跳远连接，是指a[l]跳过一层或者好几层，从而将信息传递到神经网络的更深层，使用残差块能够训练更深的神经网络，所以构建resNet网络就是通过将很多这样的残差块，堆积在一起，形成一个深度神经网络。



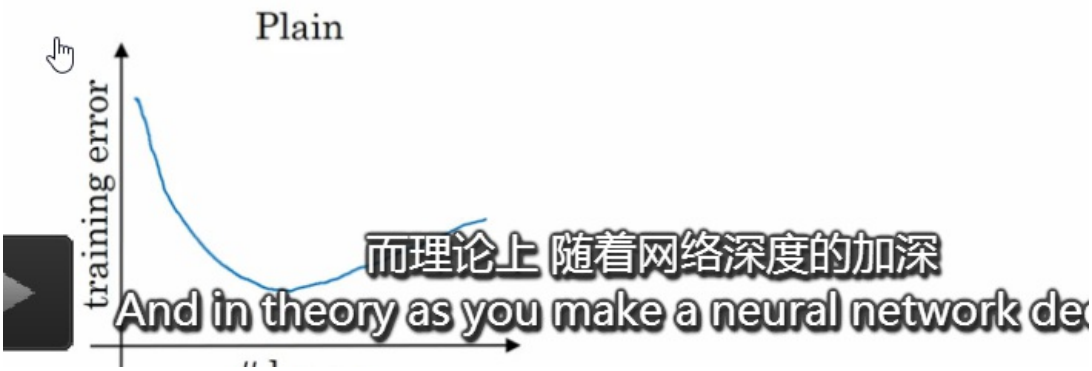
把它变成resNet的方法就是加上所有的跳远连接。



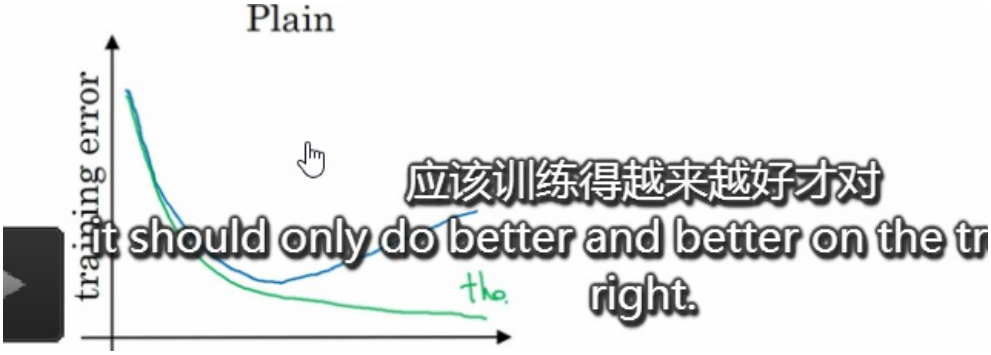
构成一个残差块



构成一个残差网络，如果我们使用标准的优化算法去训练一个普通的网络，比如说梯度下降或者其他热门的优化算法，如果没有多余的残差，没有这些捷径，或者说跳远连接，凭经验，你会发现随着网络深度的加深，训练误差会先减小然后增加。

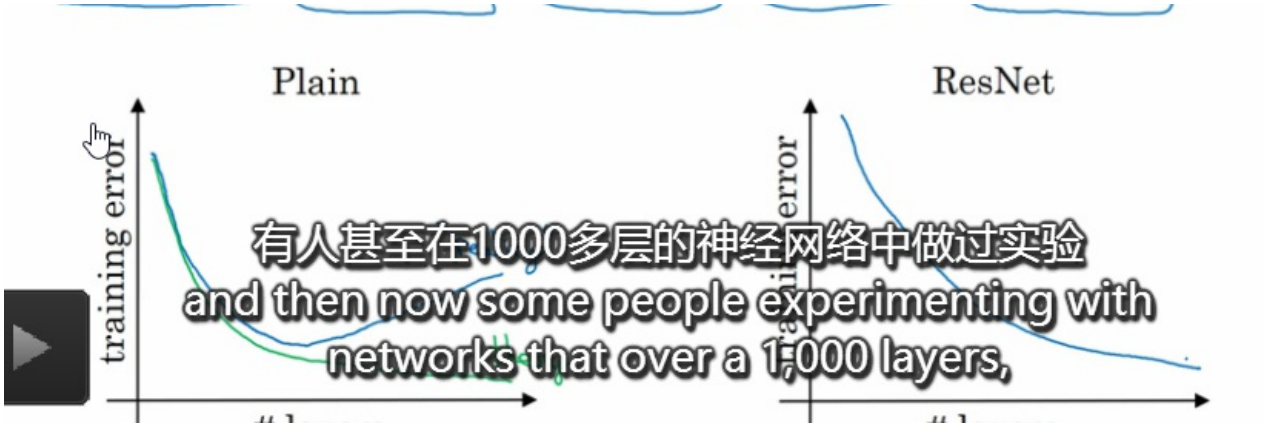


理论上随着网络深度的加深，应该训练的越来越好才对



也就是说，理论上网络深度越深越好，但是实际上，如果没有残差网络，对于一个普通的网络来说，深度越深意味着，用优化算法越难训练，随着网络深度的加深，训练误差会越来越多，但是有了resNet就不一定了，即使网络再深，训练的

表现却不错，比如说误差会减小，就算是训练深度达到100层的网络也不例外看。



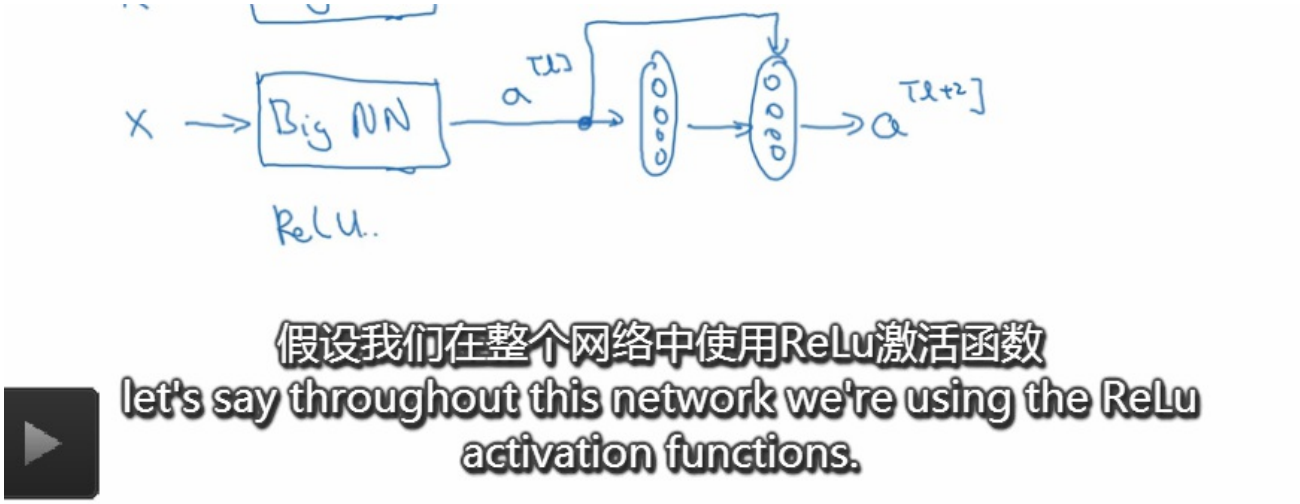
但是对于x的激活或者这些中间的激活，能够达到网络的更深层，这种方式确实有助于解决梯度消失或者梯度爆炸问题。

4、残差网络为什么有用。

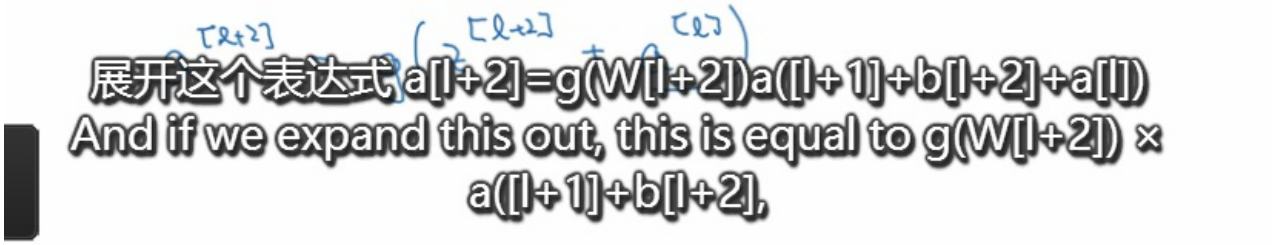
残差网络为什么构建更深的网络时，还不降低他们在训练集上的效率

神经网络越深，他在训练集上的表现效率就会有所减弱。

a[l+2]=g(w[l+2]a[l+1]+b[l+2]+a[l])如果w[l+2]为0，为了方便起见b也为0，前面几项就没了，因为为0.最后等于g(a[l])也就是a[l],因为我们假设使用relu激活函数，并且所有激活值都是负的，g(a[l])是对应于非负数的relu函数。所以a[l+2]=a[l].。

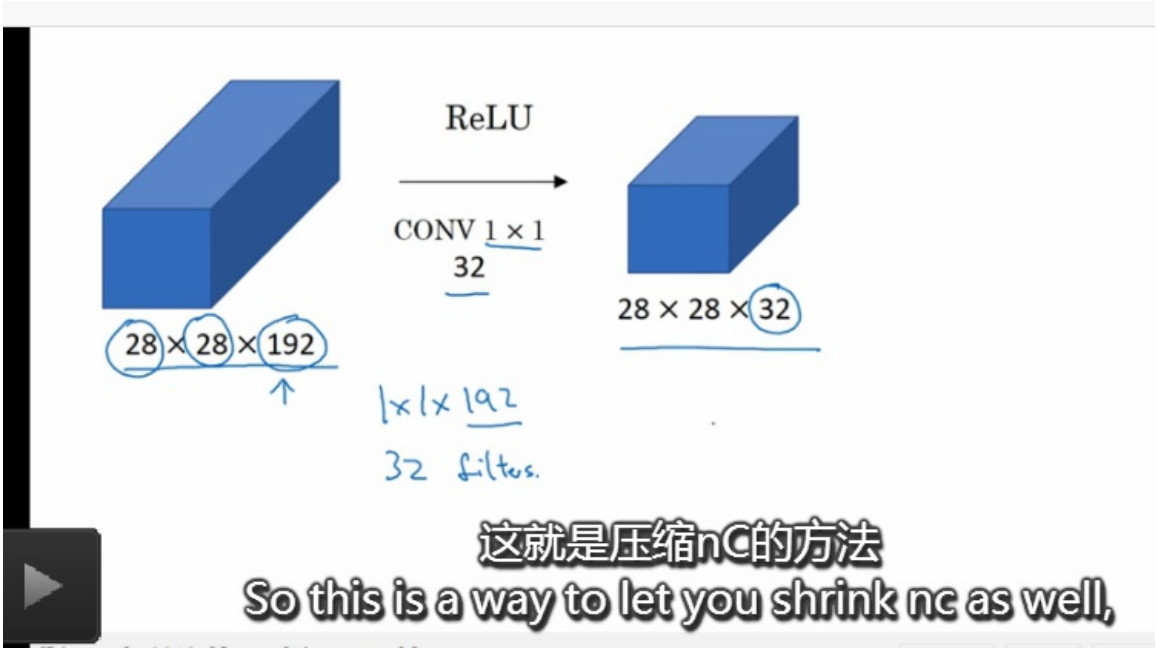


假设我们整个神经网络使用relu激活函数，所有激活值都大于等于0，包括输入X的非零异常值，因为relu激活函数输出的值要么为0要么为正数



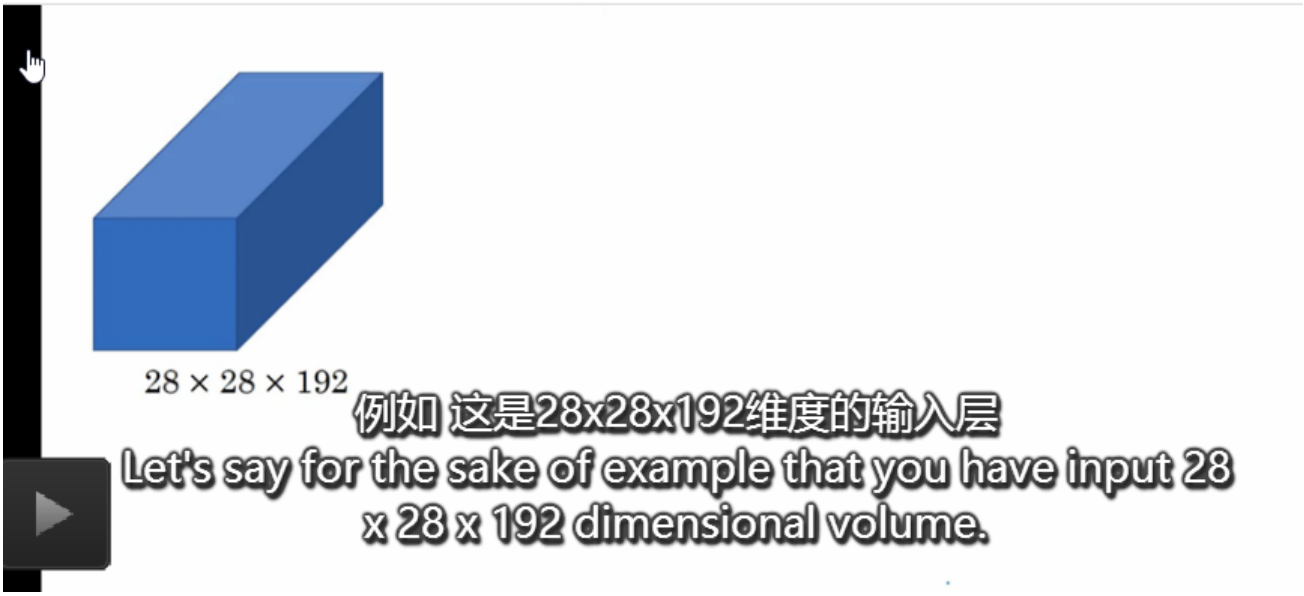
a[l+2]=g(w[l+2]a[l+1]+b[l+2]+a[l])如果使用正则化或者系数权值矩阵，他们会压缩W[l+2]的值，如果对b进行权重衰减，亦可达到同样的效果

5、1*1的卷积神经网络，可以压缩通道的数目，

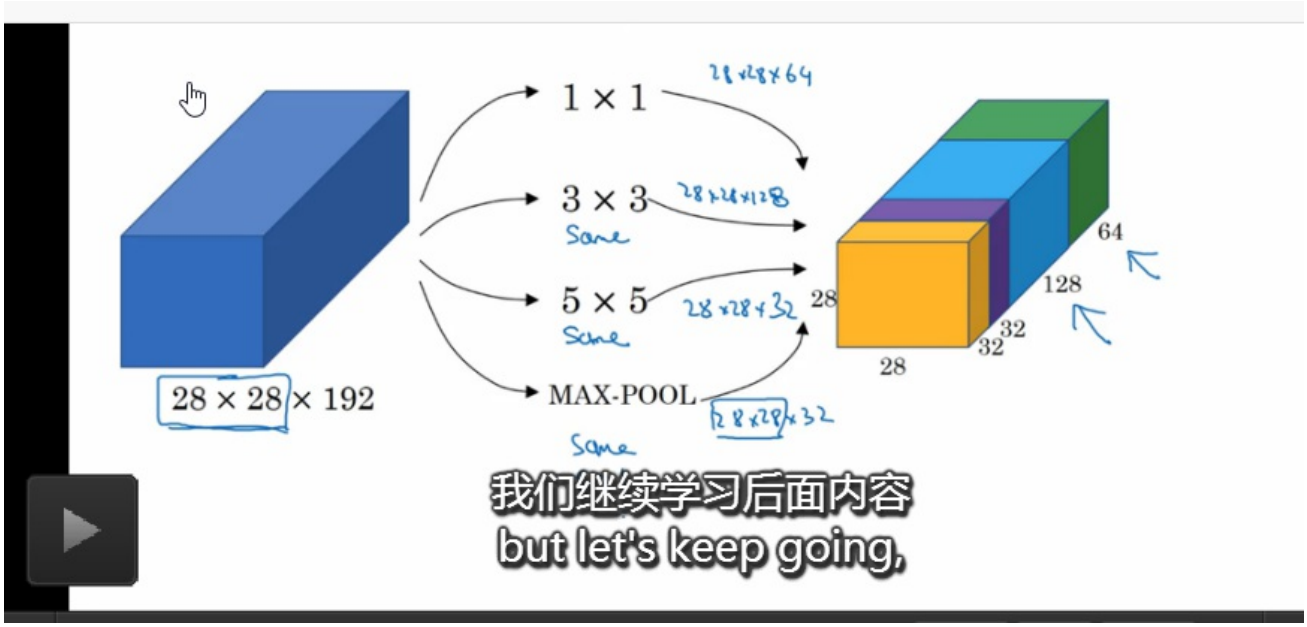


1*1的卷积层实现的重要功能就是，它给神经网络添加了一层非线性的激活函数，从而减少或者保持通道数不变，当然如果你愿意，你也可以增加通道数量。

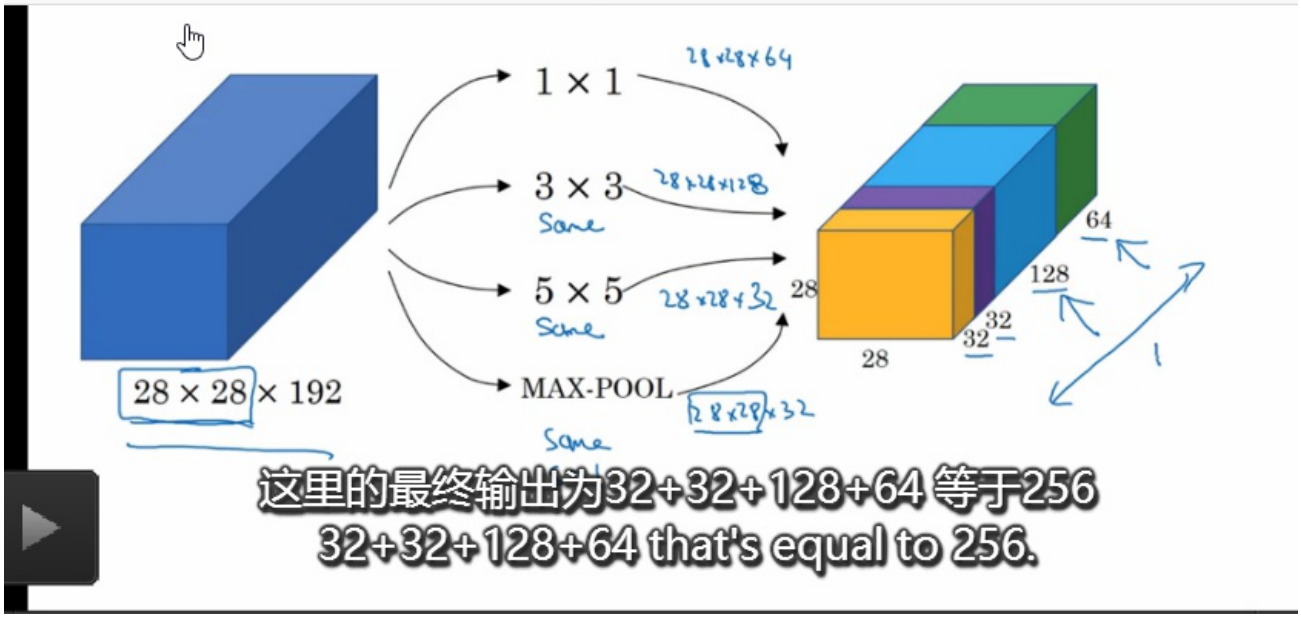
6、Inception网络的作用代替你来决定采用1*3,3*3还是5*5之类的过滤器，虽然网络结构变得更复杂，但是网络表现确实非常好，例子：



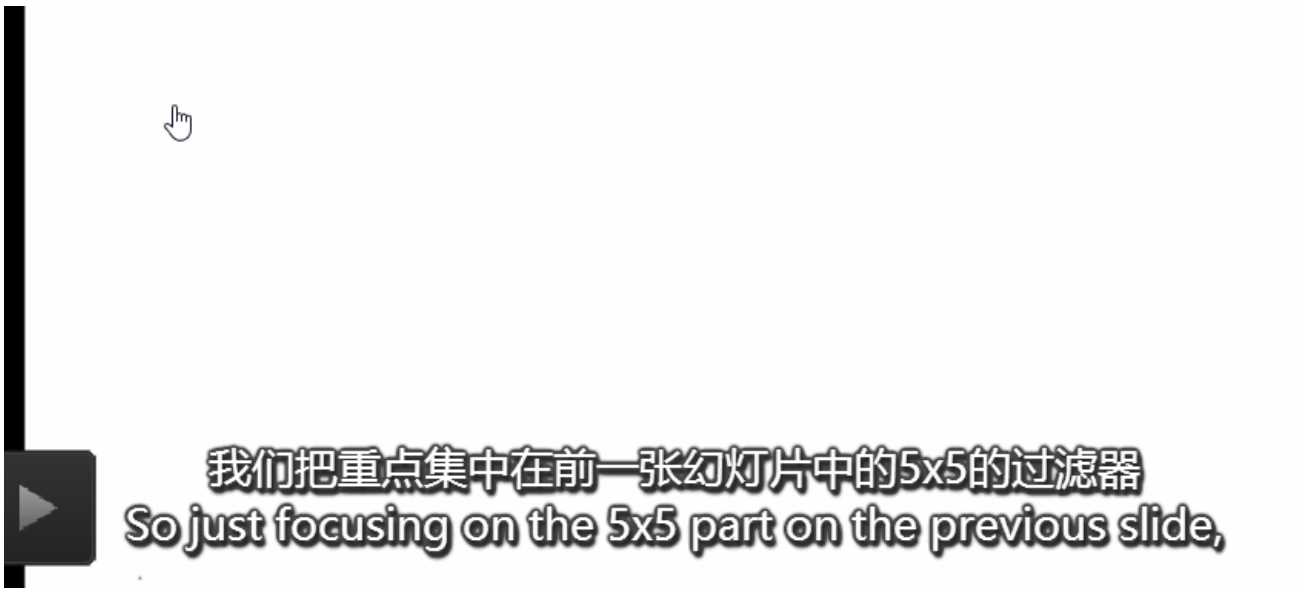
28*28*192维度的输入层，Inception网络的作用是代替人工来确定卷积层中的过滤器类型，或者确定是否要构建卷积层或者池化层，演示开始，如果使用1*1的卷积，输出结果会是28*28*某个值，假设输出是28*28*64，如果是3*3的过滤器，那么输出是28*28*128，然后我们把第二个值堆在第一个值上，为了匹配维度，我们应用相同的卷积，输出维度依然是28*28，和输入维度相同，或许你会说，我希望提高网络的表现，用5*5的过滤器会更好，输出变成28*28*32，注意维度还是28*28，然后你就不想再卷积了，那么讲究用池化操作，得到一些不同的输出结果，我们也把它堆积起来，这里的池化输出是28*28*32，为了匹配所有维度，我们需要对最大池化使用padding，它是一种特殊的池化形式，因为如果输入维度是28*28，则输出相应维度也是28*28，然后再进行池化padding不变步长为1,。



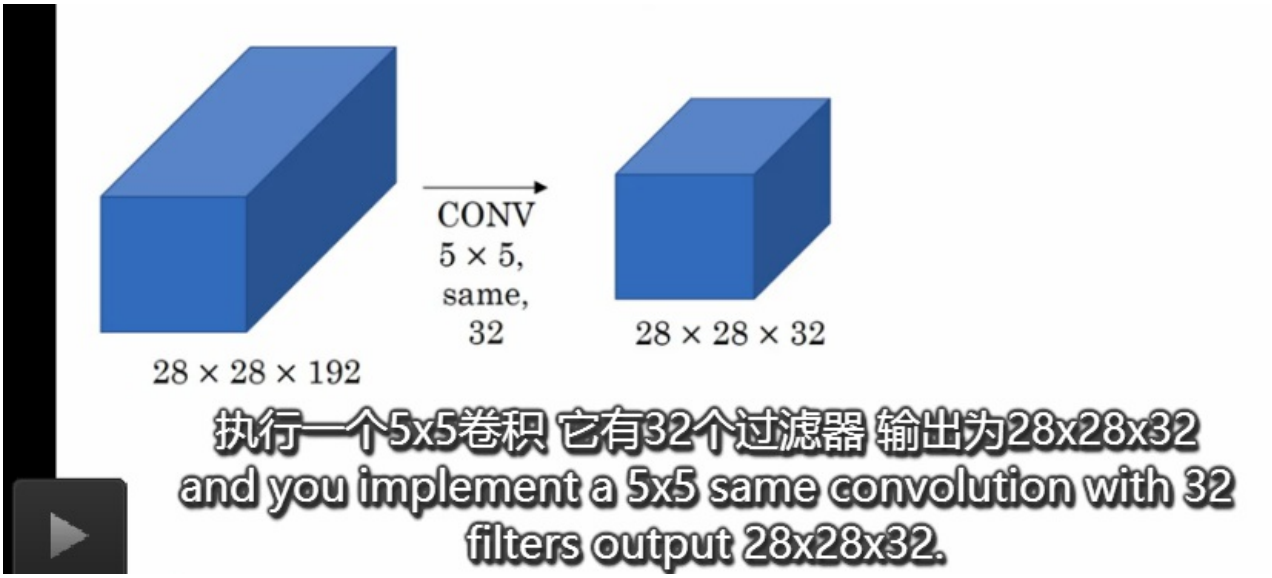
有了这样的Inception模块，你就可以输出某个量，因为它累加了所有数据



它的输入模块是28*28*129，输出为28*28*256，这就是Inception模块的核心内容，基本思想是Inception网络不需要人为决定使用哪个过滤器、是否需要池化，而是由网络自行确定这些参数，你可以给网络添加这些参数的所有可能的值，然后把这些输出连接起来，让网络自己学习它需要怎么样的参数，采用哪些过滤器组合，不难发现，Inception层有一个问题，就是计算代价成本，接下来就来计算5*5过滤器在该模块中的计算成本。

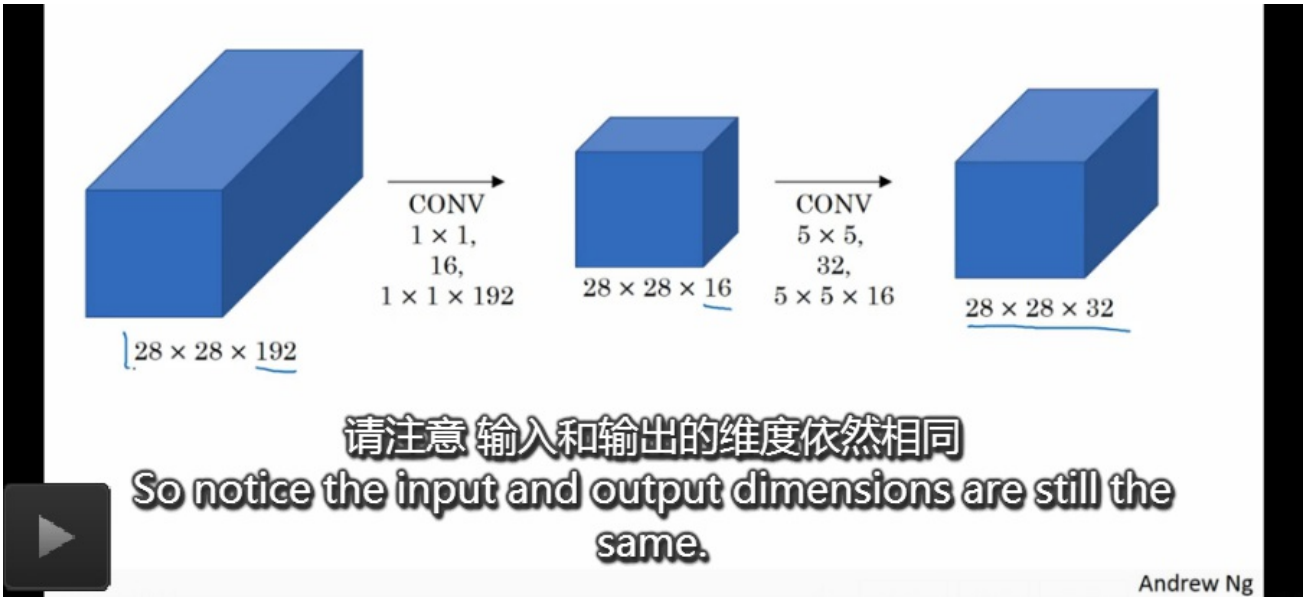


输入是28*28*192



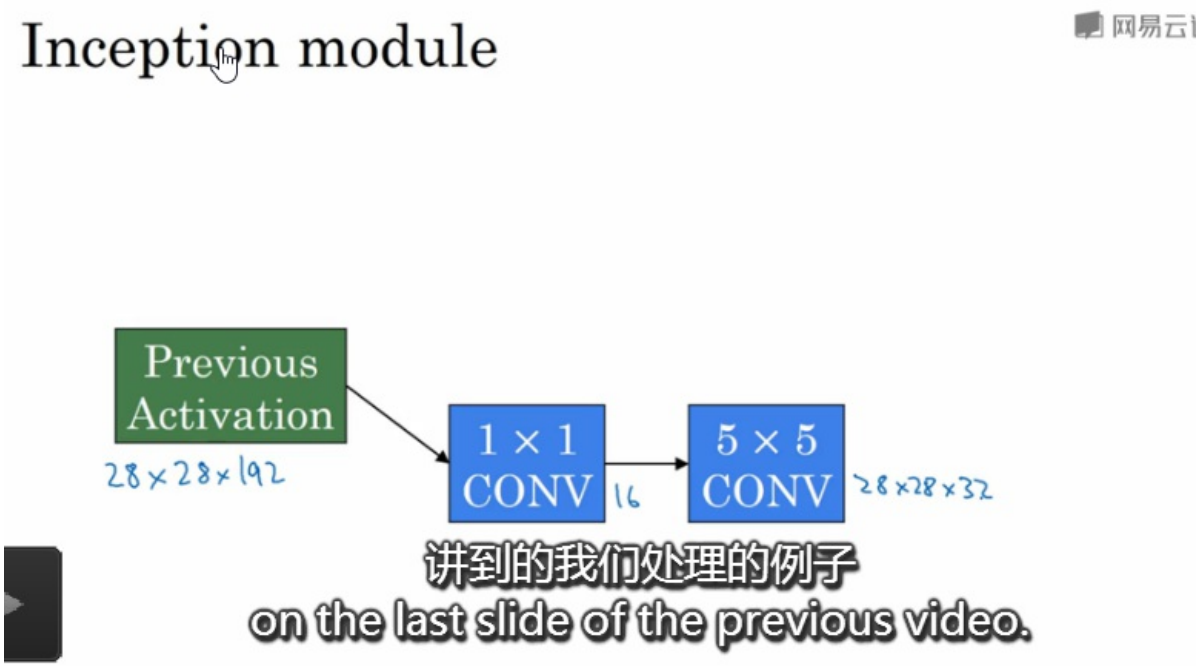
接下来计算28*28*32的计算成本，它有32个过滤器，因为输出有32个信道，每个过滤器大小为5*5*192，输出大小为28*28*32，所以要计算28*28*192个数，对于输出的每个值，都要执行5*5*192次乘法运算，所以乘法运算的总数为，每个输出值所需要的乘法次数乘以输出值个数，把这些数相乘，结果为28*28*32*28*28*1921=1.2亿，为了降低计算成本，我们用计算成本除以因子10，结果它减少了到了原来的10分之1，

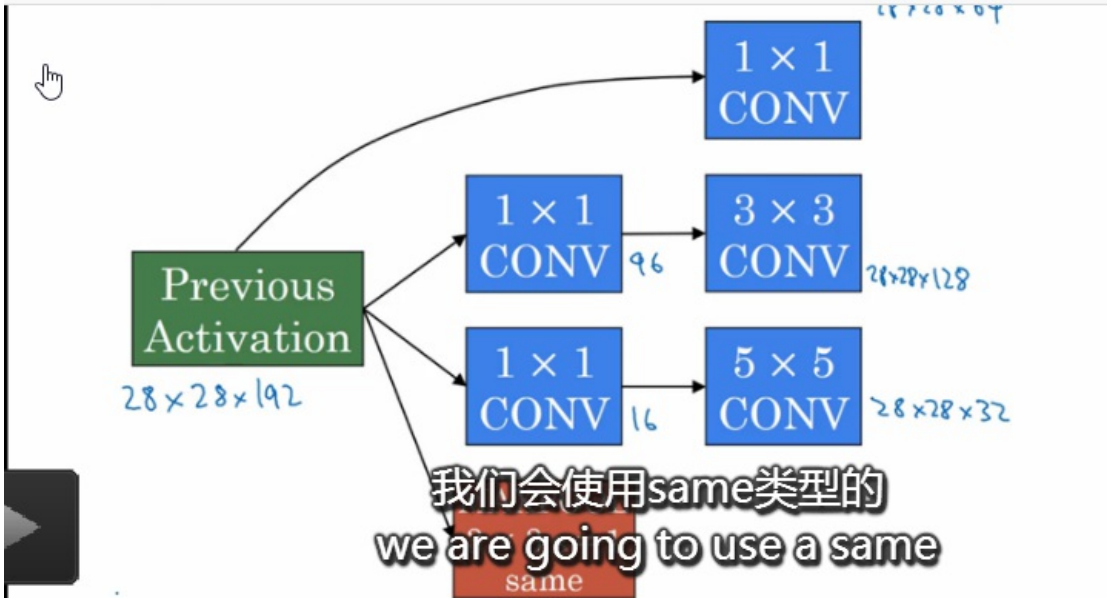
还有一种方法是，先进行1*1的卷积，使得原来的信道减小到16，然后再进行5*5的卷积得到结果为28*28*32。：



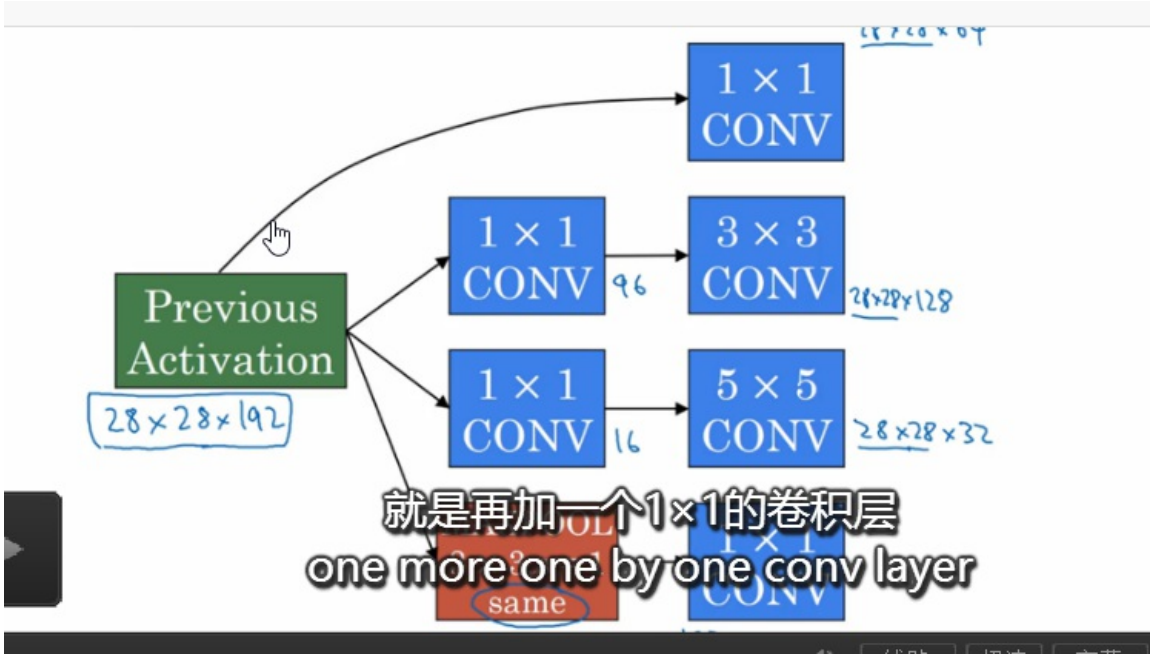
1*1的计算成本为28*28*16*192,240万，5*5过滤器的计算成本为28*28*32*5*5*15等于1000万，所以总成本为1240万个参数的计算量，总结一下，你在构建神经网络的时候，不想要决定使用1*1还是3*3还是5*5的过滤器的时候，Inception模块是最好的选择，1*1的卷积核，可以构建瓶颈层这样就可以大大减小计算的成本。你可能会问，仅仅大幅缩小表示层的规模，会不会影响神经网络的性能，事实证明，主要构建合理的瓶颈层，你既可以缩小表示层的规模，又不会降低网络性能，从而大大节省计算，这就是Inception模块的主要思想，

7、Inception网络

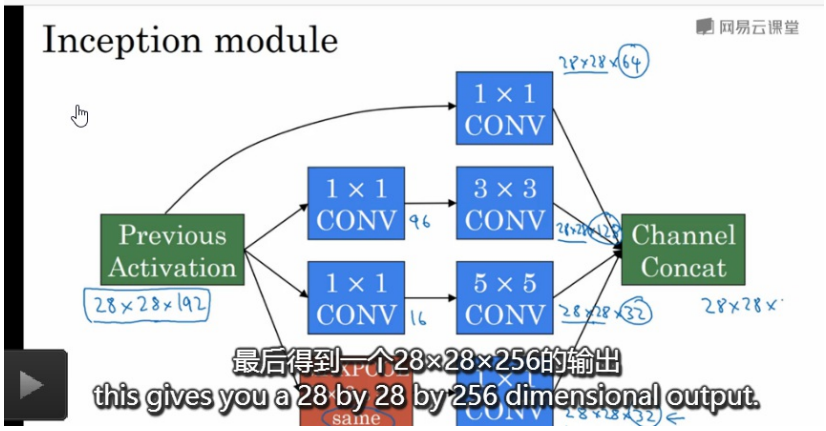


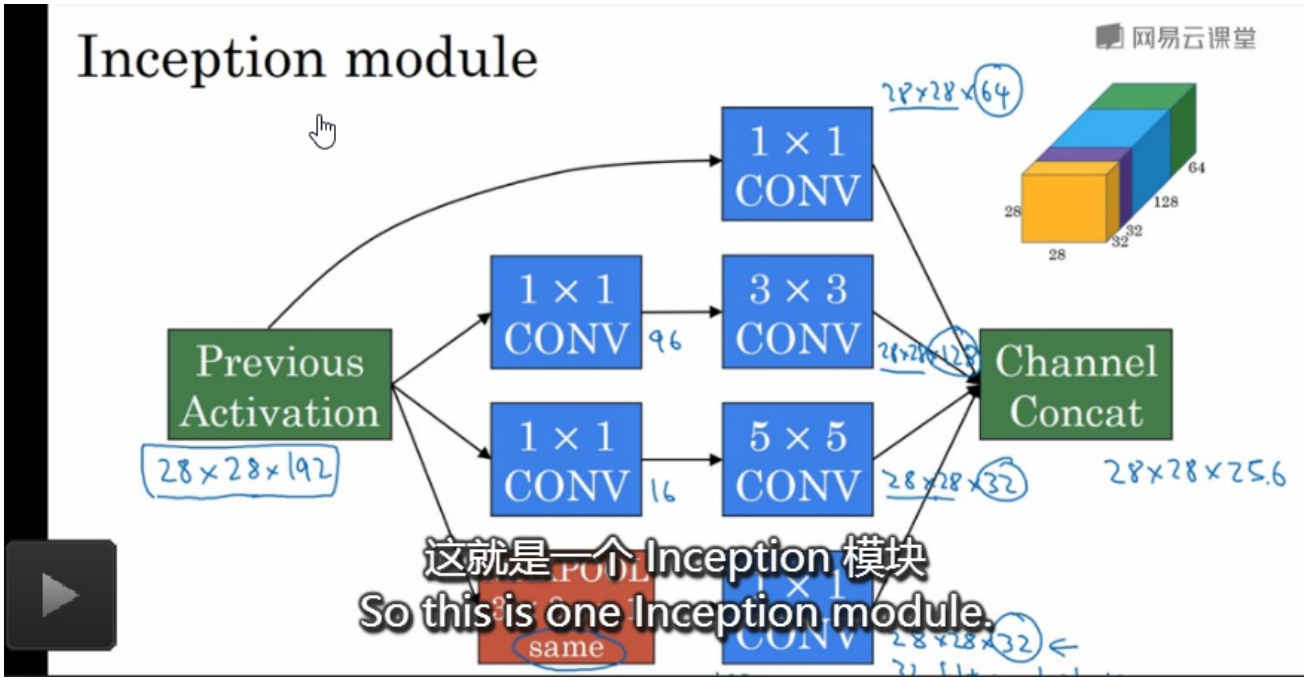


为了将最后每个模块都组合再一起，我们就要使得输出的长和宽的维度都保持一致，所以max pooling要进行填充same类型，但是如果你进行了最大池化，通道数还是不变的，还是192个通道，所以看起来它会有很多通道，我们实际要做的，就是再加上一个卷积层：

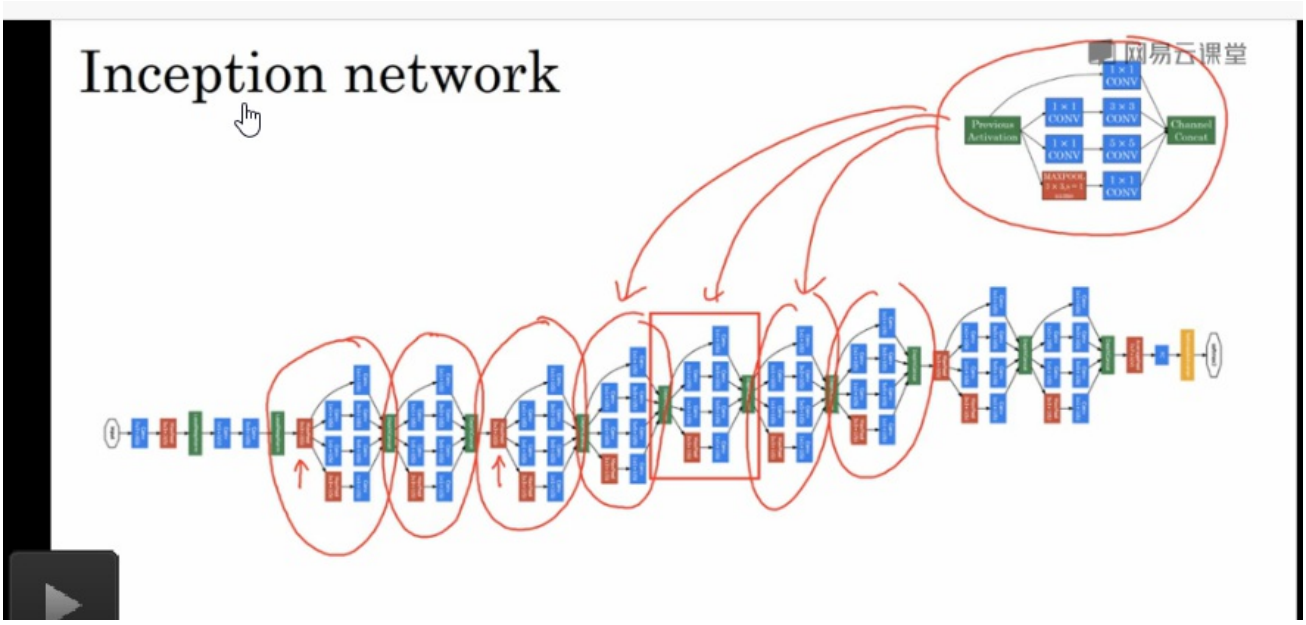


目的是为了将通道数缩小，也就是使用32个1*1*192的过滤器，最终的输出结果为28*28*32维度，所以最后通道数缩小为32个，所以这样就避免池化层占据大多数的通道，最后将这些方块全部拼接在一起，其实就是把各层的通道都加起来，最后得到一个28*28*256维度的特征图





而Inception所做的，就是将这些模块都组合再一起。



其实论文中每个Inception层还有很多分支，其实就是softmax层这样就可以确保我们即使是在隐藏层或者中间层，我们仍然可以参与特征的计算，他们也能预测图片的分类，它在Inception网络中起到调节的作用，并且能防止网络发生过拟合。

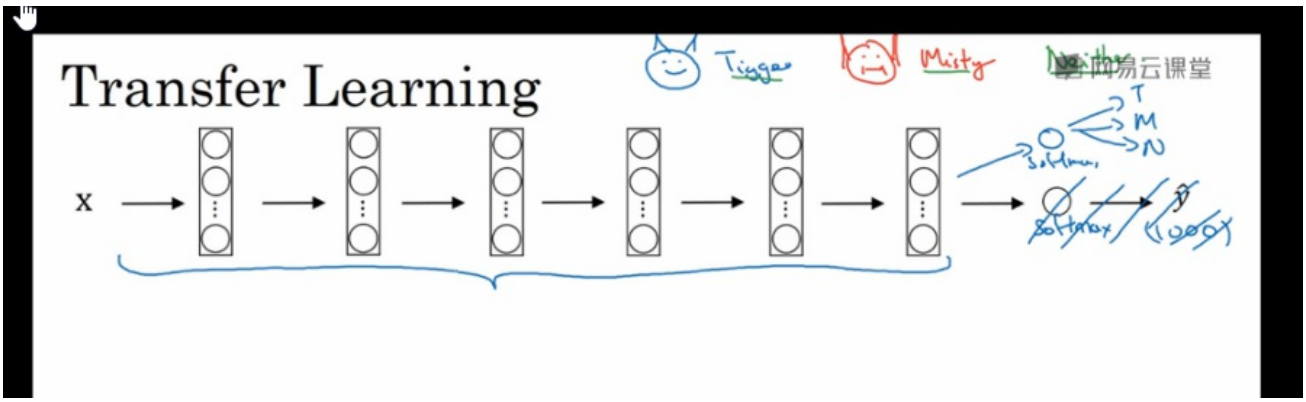
8、如何使用开源的实现方案

实用性建议：事实证明，很多神经网络复杂细致，难以复制，因为一些参数调整的细节问题，例如学习率的衰减等等，会影响性能，所以吴恩达建议使用论文上思想时，最好在网上寻找 一个开源的实现，因为如果你能得到作者的实现，比你从头开始实践要快得多，虽然从零开始是一个很好的锻炼，假设你对残差网络感兴趣，并想使用它，那么就直接找就可以了。你要开发视觉识别项目，选择一个你喜欢的框架，下载下来，这样做的优点在于，这些网络通常都需要训练很长的时间，这些可能都是别人预先训练好的网络，你就可以利用这些网络进行迁移学习，

9、迁移学习

如果你要做一个计算机视觉应用，相比从头训练权重，或者说随机初始化权重开始，如果你下载别人已经训练好的网络权重，你通常能够进展的相当快，用这个作为预训练，然后转到你感兴趣的任务上，计算机视觉的研究社区，非常喜欢，把许多数据集上传到网上，比如ImageNet等等数据集，你就可以下载别人训练好的网络权重参数，把它当做一个很好的初始化，用到你自己的神经网络上去，用迁移学习，把公共的数据集的知识迁移到你的问题上去，举一个例子：

比如你要训练一个猫的检测系统，用来检测你自己的宠物猫，但是现在你的训练数据集很小，你该怎么办呢，你可以从网上下载神经网络的开源实现，不仅把代码下载下来，也要把权重下载下来，有很多训练好的神经网络，你都可以下载下来，比如下ImageNet上训练好的1000个分类任务的网络，因此这个网络最后一定有一个softmax层，它可以输出1000个可能类别之一，你可以去掉这个softmax层，创建你自己的softmax单元



对于网络的其他层，我们使用它训练好的参数就可以了，也就是freeze其他所有层的参数，你只要训练softmax相关层的参数就可以了，最后你的softmax层有三个输出，通过使用其他人训练好的权重，你可能得到一个很好的性能，即使只有一个小的数据集，所以这个能加速训练的技巧就是，如果我们先计算第一层，计算特征或者激活值，然后把它保存到硬盘里，你所要做的就是把前面冻结的当做一固定的函数，在这个神经网络的前半部分，取任意输入图像X,然后计算它的某个特征向量，这样你就训练一个很浅的softmax层，这样做的优点在于你就不用遍历整个训练集，再重新训练这个激活值了，如果你的数据集越来越大，那么你就可以考虑冻结更少的层，当你的训练集非常大的时候，你可以把他们训练好的权重值作为初始值，重新进行训练，