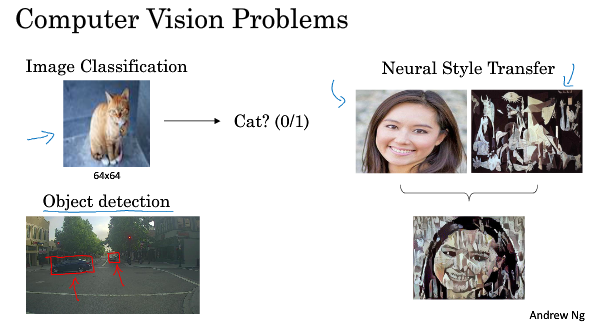
1. 卷积神经网络
   1. 计算机视觉

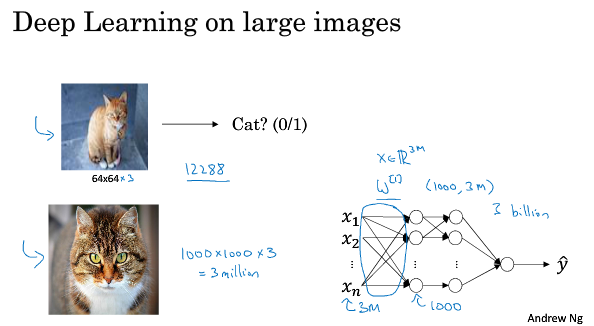
机器学习和计算机视觉可以帮助汽车查明周围的行人和汽车，帮助避开障碍物，还使得人脸识别技术变得更加精准。

第一计算机视觉的发展标志着新型应用产生的可能，通过学习使用这些工具，也许能创建新的产品和应用。第二，即使在计算机视觉上没有成就，由此衍生出来的神经网络结构和算法实际上启发了人们在计算机视觉和其他领域的交叉成果。

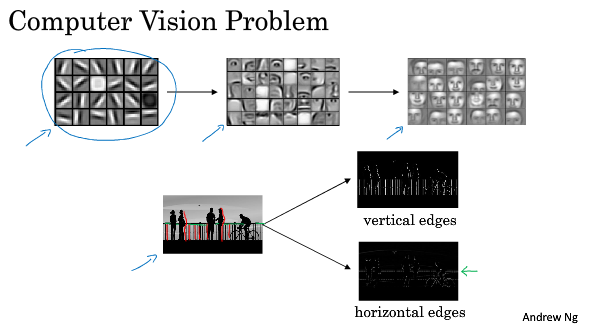
进入正式学习，图像识别，比如给这张64\*64的图片让计算机判断是不是猫图，或者目标检测，无人驾驶项目不一定识别出具体是什么，但要计算其他车辆的位置。所以在目标检测中，首先计算有哪些物体，再将它们模拟成一个个盒子，或用其他技术识别在图上的位置。又或者神经网络实现的图片风格迁移，有一张人照片，将这张图片转换成另外一种风格。



应用计算机视觉，很大的一个挑战就是数据的输入可能会非常大，一般操作的都是像素64\*64的图，实际数据量是64\*64\*3，这还好。如果操作更大像素的图片，特征向量维度高达三百万，那么在第一个隐藏层可能有1000个隐藏单元，所有的权重组成矩阵W[1],如果使用了标准的全连接网络，那么W[1]的维度是（1000,3百万），这种情况下难以获取足够的数据来防止神经网络发生过拟合和竞争需求，巨大的内存需求让人没法接受。但神经网络不可能只处理小图片，也要会大图片，所以学习卷积运算，卷积神经网络重要的一块。

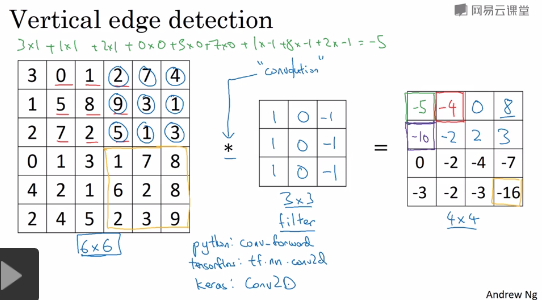


* 1. 边缘检测示例

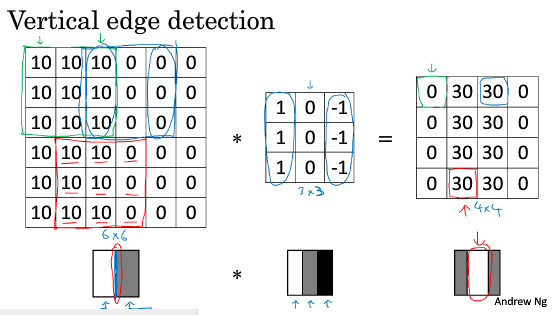


垂直边缘和水平边缘大概什么意思呢，就是神经网络的前几层可以检测边缘，然后后面的层可以检测到物体的部分，更靠后的可以检测到完整的物体。上图中可以大概的对垂直边缘进行直观的感受一下。

Convolutional operation卷积运算。使用边缘检测作为入门样例，看卷积是如何运算的。运算过程很简单，一个图片的像素矩阵\*（这里的乘号表示卷积运算）过滤器（一个维度更小的矩阵）=另一张图片。具体的运算就是使用过滤器附在原始图片上，对应元素进行相乘计算总和得到对着一个块的计算结果。不同的开发环境以及工具中，卷积运算的实现方法不一样：

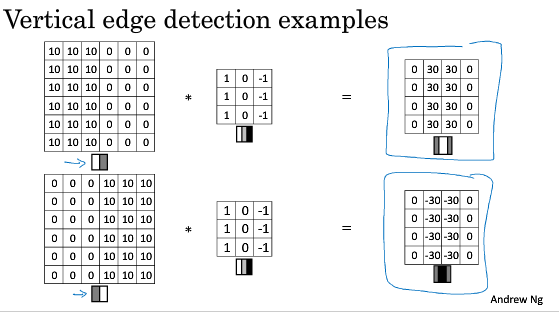


为什么这种操作能做垂直边缘过滤器呢，下图中原始图片上，像素值代表颜色更亮，像素值小则是颜色偏深一点，最后通过卷积运算得到的检测结果，中间两条30带比较亮的这块区域表示了垂直边缘，虽然这里的边缘看起来有点宽，那是因为我们原始图片太小，如果原始图片是1000\*1000的，那么垂直边缘就会显得更正常一点。

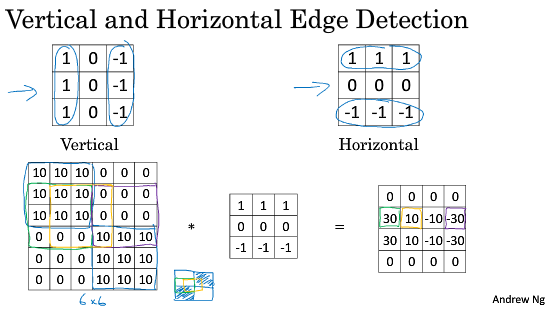


* 1. 更多边缘检测内容

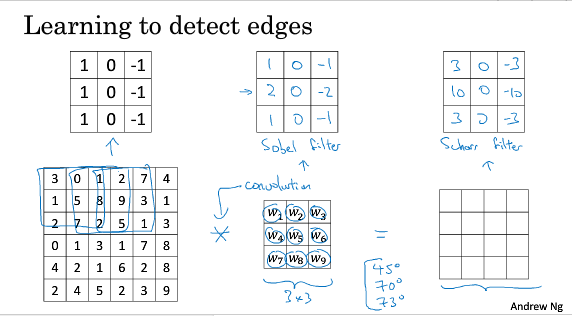
已经见识到用卷积运算实现垂直边缘检测，继续学习如何区分正边和负边，实际就是由亮到暗与由暗到亮的区别，也就是边缘的过渡。并且了解其他类型的过滤器，以及如何实现这些算法，或者自己编写一个边缘检测程序。



上图中的例子就是将最初的从亮到暗的图翻转了一下，得到从暗到亮，仍然使用3\*3的垂直边缘过滤器进行检测，得到新的检测结果。观察图中的数字，原先是计算结果30，说明是正边，从亮到暗，后来是-30，负边，从暗到亮过渡。如果不在乎这两者之间的区别，可以取输出矩阵的绝对值。但这个特殊的矩阵确实可以帮我们区分这两种明暗变化的区别。



上图中分别有垂直边缘检测过滤器和水平边缘检测过滤器，前者从左到右是从亮到暗，后者从上到下是从亮到暗，对于一个更复杂的例子，上图6\*6图片，与一个水平过滤器做卷积运算，可以从输出矩阵发现，30说明了是正边，从亮到暗，-30则是负边，从暗到亮，中间的10则是有两条正边带和一条负边带一起得到的中间值，再次说明，原始图片很小才这样，如果原始图片很大，则不会明显出现这种过渡带。



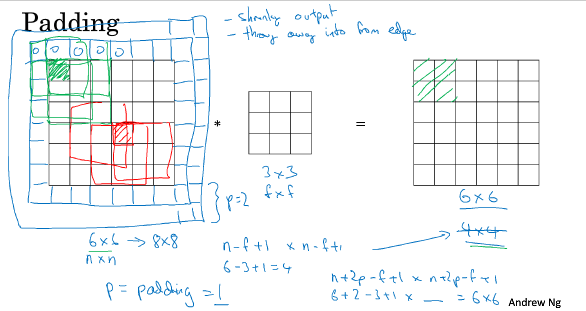
对于3\*3过滤器，我们只使用了其中一种数值组合，其他研究者还有其他数值组合，如Sobel filter（增加了中间一行元素的权重值，也就是处在图像中央的像素点，使得结果的鲁棒性会更好一点） 和Scharr filter。

随着深度学习发展，当你真正想去检测出复杂图像的边缘，不一定要使用已经有的这些数字组合，你可以把这9个数变成参数，使用反向传播算法去学习理解这9个参数，再与原始图像做卷积运算，就能得到一个效果更特别的过滤器。相比垂直检测器或水平检测器，这种方式可以得到检测任何角度边缘的过滤器。

* 1. Padding

构建深度神经网络，需要会的一个基本卷积操作就是padding（填充）。首先对于一个原始图像（n\*n）与一个过滤器（f\*f）进行卷积运算，最后得到的是(n-f+1)\*( n-f+1)的出入图像。

那么就会发现两个问题：第一，输出缩小，随着卷积操作次数越多，一个图片就会越来越小，最后可能就1\*1了，第二，边缘信息丢失，原始图像角落边的像素点只被一个输出多触及到，而在中间的像素点就会被过滤器很多次覆盖到，所以原始图像最边缘的像素点发挥的作用很小，意味着你丢掉了图像边缘位置。所以padding就是解决这两个问题的一个好工具。

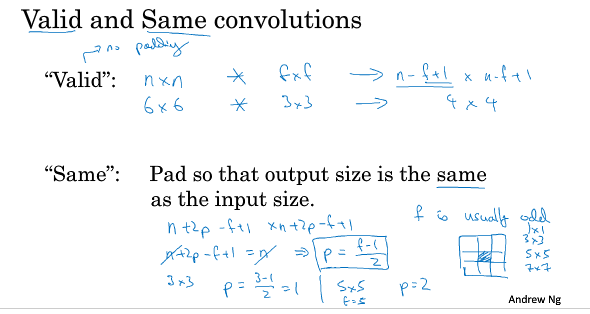


p表示的是在原始图片四周添加的几个像素点，1表示一圈都添加1个。而填充的这些像素点一般都设置为0。

有两种常见的填充方式：Valid padding 和 Same padding

Valid padding的意思就是no padding，不填充，最后得到【（n-f+1）\*(n-f+1)】，Same padding的意思是，填充之后经过一次卷积操作，输出大小和输入大小一样，那么,也就是说

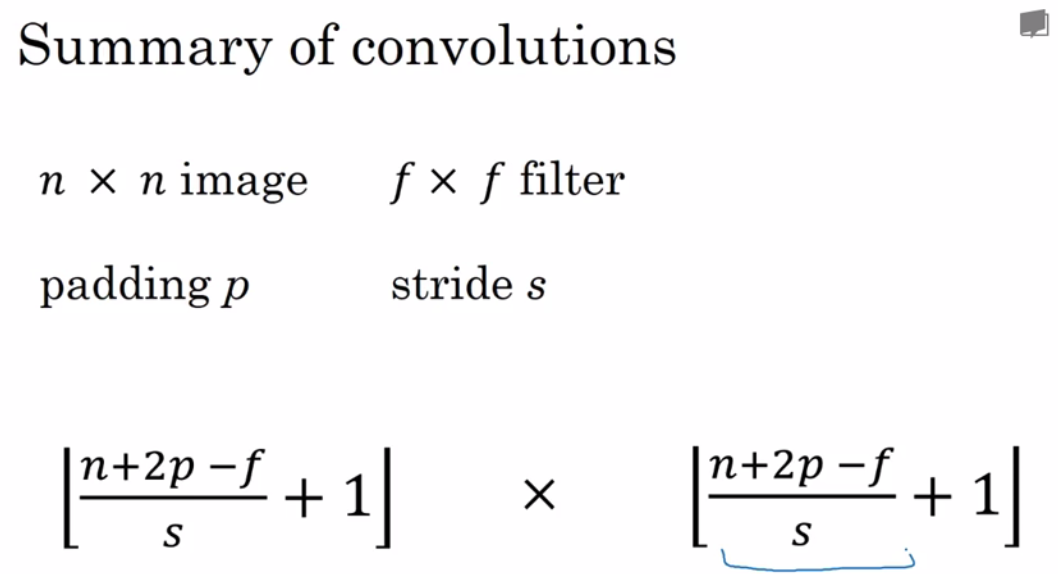
所以当f为奇数时，就能相应的填充后经过操作和原始图像大小一样。习惯上，计算机视觉f一般都是奇数，原因1：如果是偶数，只能使用一些不对称填充。原因2：一般奇数的话就会有一个中心点，有一个中心像素点可以方便的指出过滤器的位置。



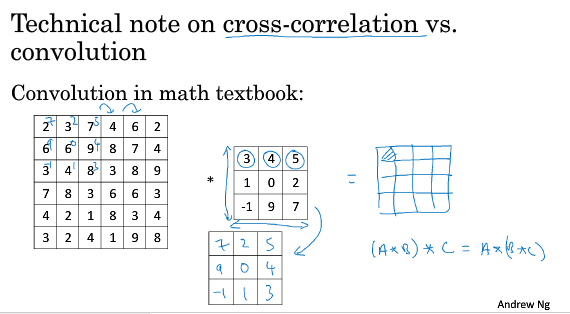
* 1. 卷积步长

设置卷积步长stride是构建卷积神经网络的基本操作，最后得到输入图片的维度是:s表示步长，p表示填充数。

如果商不是整数的话，向下取整，向下取最近的整数，，这个过滤器的覆盖必须完全在原始图像内才能算数。



再提一个cross-correlation VS convolutions互相关和卷积相关的建议，不会影响构建卷积神经网络的任何方式，取决于你读的教材关于信号处理，如果是教学教材，卷积的定义是做元素乘积求和之前，还有一个步骤就是3\*3滤波器沿水平和垂直轴翻转，用这个镜像做计算。有时候这个操作称为cross-correlation，但在有些文献中，又把这个称为卷积操作。按惯例，一般不进行翻转操作，所以我们说到的卷积都不进行这个操作，省略了双重镜像操作。



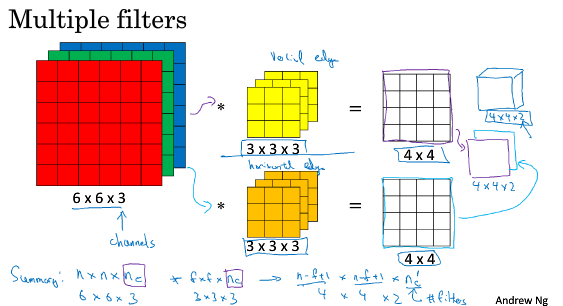
* 1. 卷积为何有效

不仅在二维上执行卷积，也看看三维立体上如何进行卷积操作。例如不是在灰度图片上，而是在彩色图片右RGB三个通道，那么更改的地方则是使用一个三维的过滤器。高\*宽\*通道数，原始图像和过滤器的通道数必须相同，最后得到的是二维矩阵，最后一个维度没有了。计算方式则是f\*f\*f个数对应相乘再求和。

对于三维过滤器，如果你只想检测红色通道图像的边缘，那么可以在三维过滤器第一层设置9个数，后面两层都为0。如果三个通道都检测垂直边界，那么久三个通道都使用111000-1-1-1组合。参数设置不同，可以得到不同的特征探测器。过滤器只关注某一个通道是可行的。

如果不仅要检测垂直边缘，想要同时检测垂直和水平边缘还有45度倾斜边缘怎么办，换句话说，同时有好多个过滤器？这种情况则是将原始图像分别和不同的过滤器进行卷积计算，再将得到的结果进行堆叠起来。比如，6\*6\*3与3\*3\*3的两个过滤器分别是垂直和水平进行卷积后得到两个4\*4输出矩阵，将两个矩阵堆叠在一起，得到4\*4\*2，这个2就表示有两个过滤器。

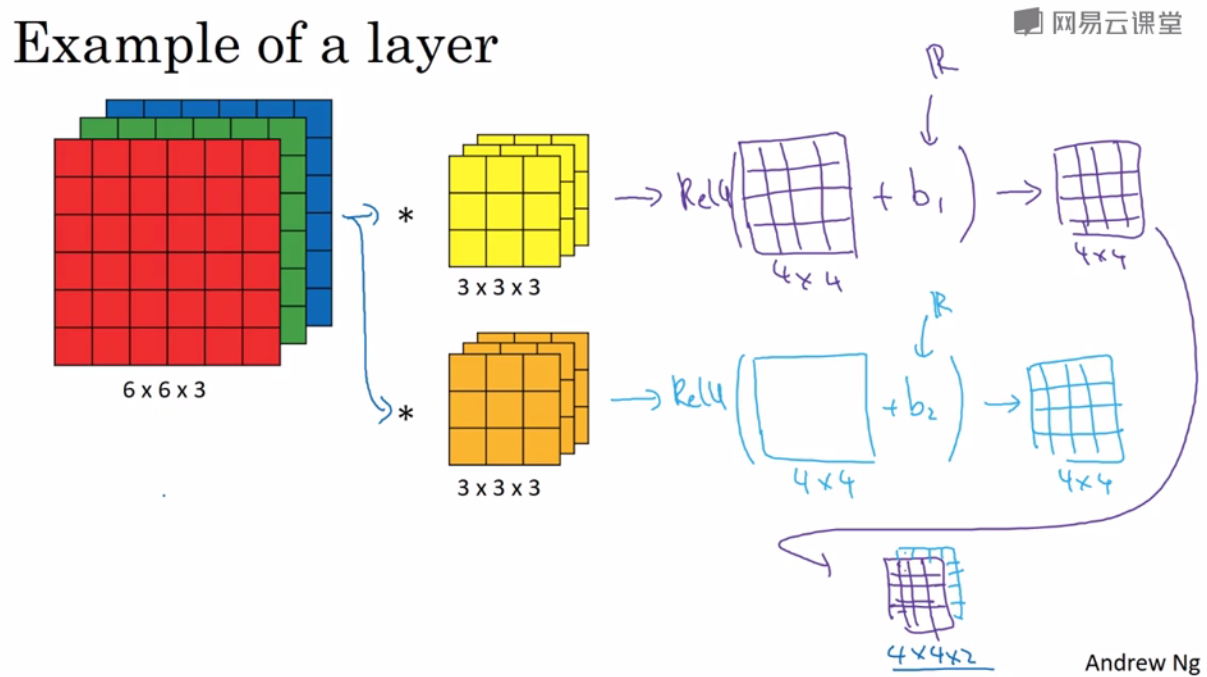
表示你用的过滤器的个数，也是下一层的通道数。



* 1. 单层卷积网络

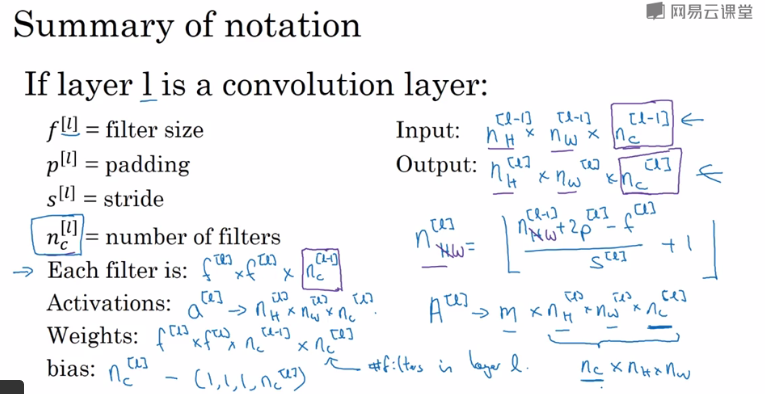
构建卷积神经网络的卷积层，对于原始图像，使用一个过滤器后，得到一个二维矩阵，加上一个实数b1，通过python的广播机制，其实是在16个数上都加上一个实数，再将这个矩阵放入非线性激活函数ReLU函数中，得到一个新的二维矩阵。同理，如果有两个过滤器，那么同样的操作，加上一个b2，得到新的二维矩阵，并且将两个矩阵堆叠起来，得到这一层的最后结果，也就是激活函数值，下一层的输入。把它映射到标准神经网络中四个卷积层的某一层，或者一个非卷积神经网络中。注意，前向传播一个操作就是，再执行非线性函数得到。

所以原始图像就是，过滤器就是变量。

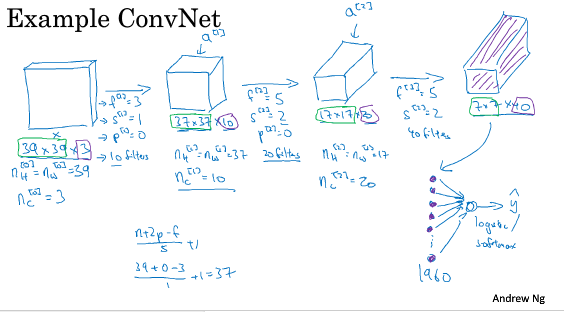


卷积神经网络的一个特征，“避免过拟合”，尽管原始图片很大很大，但是参数却很少。举个例子，比如使用了10个3\*3\*3的过滤器，那么这一层的参数就是个参数。已经知道如何提取10个特征可以应用到大图片上，而参数数量固定不变。

总结卷积神经网络某一层l层用到的所有符号标记，以及对应的大小和维度。



* 1. 简单卷积网络示例



对于经过很多卷积层之后，得到的应该是一个的一个三维立体结构，首先应该讲这个三维的数据平展成一个向量，再根据对应的题目要求选择激活函数，二分分类选择logistic，k类选择一类则是softmax。也就是最后一步，就是处理所有的特征数字。为了预测最终结果，把这个长向量填充到softmax回归函数中。

设计卷积神经网络时，确定这些超级参数比较费劲，要决定过滤器的大小，步长，padding填充数，以及使用多少个过滤器。随着神经网络计算的加深，通常开始时的图像更大一些，高度和宽度在一段时间内会保持一致，但是随着神经网络的加深而逐渐减小。而通道数量在增加。

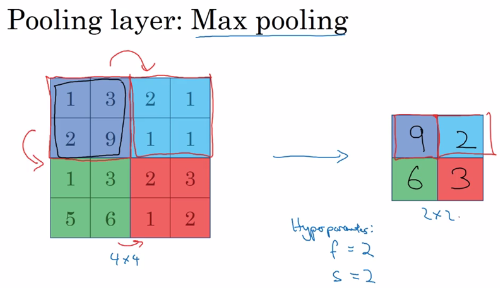
一个典型的卷积神经网络通常有三层，一个是卷积层，Conv，上面例子中就是用的卷积层，还有两种常见的层是，Pooling池化层，Fully connected全连接层FC。虽然仅用卷积层也能构建出很好的神经网络，但大部分神经网络架构师会添加池化层和全连接层。后两者比卷积层都更容易设计。

* 1. 池化层

除了卷积层，卷积网络也经常使用池化层来缩减模型的大小，提高计算速度，同时提高所提取特征的鲁棒性。池化操作有两种类型，最大池化和平均池化。

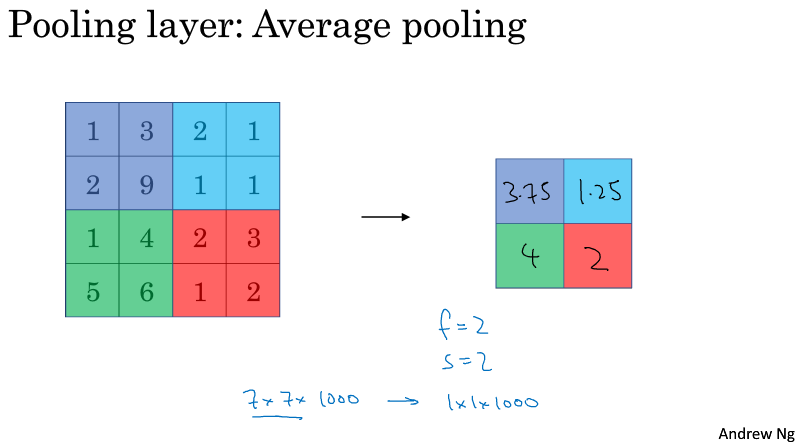
看一个对池化层作用理解的直观例子，把4\*4的输入看作是某些特征的集合，也就神经网络中某一层的反激活值的集合。数字大意味着可能提取了某些特定的特征，左上象限具有这个特征，可能是一个垂直边缘，显然左上角的特征右上角是没有的。最大化操作的功能就是只要在任何一个象限内提取到某个特征，它都会保留在最大池化的输出里。最大化的作用就是提取到某个最大特征，保留到输出，如果没有提取到，可能右上象限中不存在这个特征。

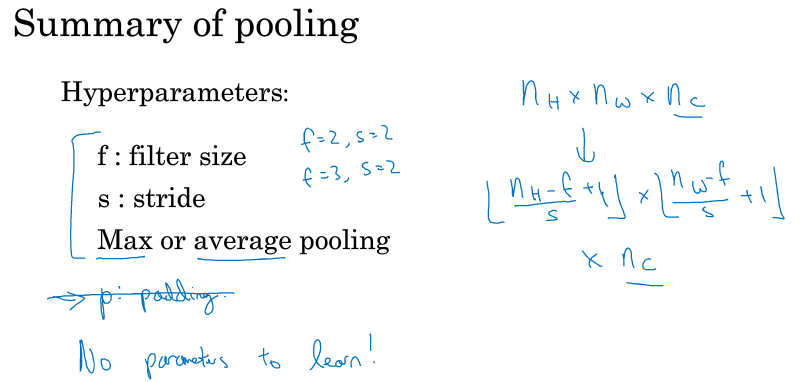
最大池化的操作就是同样是一个过滤器，附在原始图像上，不过不再是对应相乘求和，而是取最大值，对于一个4\*4的原始图像，经过f=2，s=2的操作后，就相当于划分为下图4个颜色，并且按照步长和过滤器大小，就是每个颜色块的最大值，最后得到一个2\*2的矩阵。如果有很多个通道，那么就是分别每个通道上都进行池化操作。所以原始图像有几个通道，最后的输出结果也就有几个通道。关于s和f都是池化层的超级参数，



人们使用最大池化，在很多实验中效果都很好，上面这个例子也能说明最大池化效率高的原因。其中有一点，池化层有一组超参数，但是并没有参数需要学习，实际上梯度下降没什么可以学的，一旦确定了f和s，就是一个固定运算。原始输入个通道中的每个通道都单独执行最大池化计算。

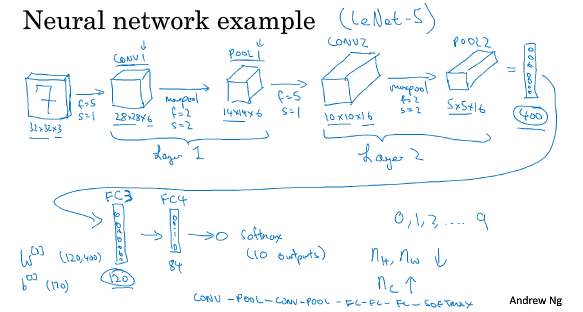
另外一种平均池化不怎么经常用，就是对每个过滤器都取平均值，而不是最大值了，最大池化更常用，但也有例外，对于很深的神经网络，可以用平均池化来分解规模为7\*7\*1000的网络的表示层，在整个空间内求平均值，得到1\*1\*1000的输出。



池化的超级参数包括过滤器的大小f和步长s，常用的是f=2，s=2，相当于宽度和高度都缩减一半，至于其它参数就看你用的最大池化还是平均池化了。还可以使用padding，但是很少这么用，最大池化时很少用到。最大池化的输入为,输出为,输入通道数跟输出通道数相同。没有参数需要反向传播，可以是自己手动设置的或者交叉验证得到的。

最大池化只是计算神经网络某一层的静态属性，没什么需要学习的。

* 1. 卷积神经网络示例

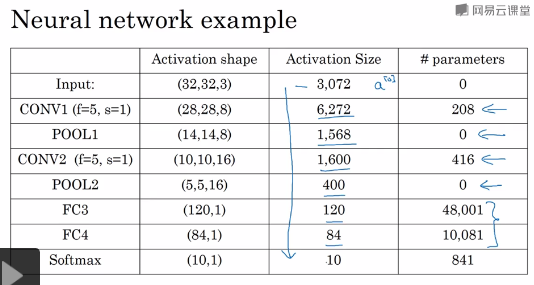


上图是一个识别手写数字7的卷积神经网络的结构，它与一个经典网络LeNet-5许多参数都很像，其中我们把卷积层和池化层作为一层，原因是我们习惯只把有权重的层认为是一层，而池化层是没有权重的，所以与前面的卷积层合为一层。全连接其实就是类似于前面两章学的单层神经网络，意思就是与后面一层的神经元一一连接，因此也会有对应W和b参数，维度大小也和之前定义的一样。在经过Pool2以后，将得到的400个特征平展为一个向量，即相当于一层有400个的神经网络层，然后经过FC3 FC4全连接层，最后使用softmax层，得到一个有10个参数的向量。

此例中的卷积神经网络很典型，看起来有很多参数，常规做法是尽量不要自己设置超级参数，而是查看文献别人采用了哪些超级参数，选一个别人任务中效果比较好的，有可能也适用于自己的应用程序。

随着神经网络深度的增加，高度和宽度通常都会减少，而通道数量会增加。在卷积神经网络中，一种比较常见的就是，一个或多喝卷几层后面跟随一个池化层，然后一个或多个卷积层后面再跟一个池化层，然后是几个全连接层，最后一个是softmax。

最后说说神经网络的激活值shape，激活值大小和参数数量，下面表可以自己完成。

要点，1、第一池化层和最大池化层没有参数，2、卷积层的参数相对较少，其实许多参数都存在于神经网络的全连接层，观察可发现，随着神经网络的加深，激活值会逐渐变小，如果激活值下降太快，也会影响网络性能。许多卷积网络都具有这些属性，模式上也相似。

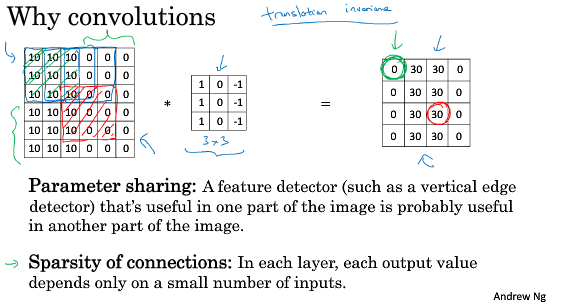
关于这一节，就是如何把这些基本模块整合起来，构建高效的神经网络，如何设置好的构建，办法就是大量阅读别人的案例。

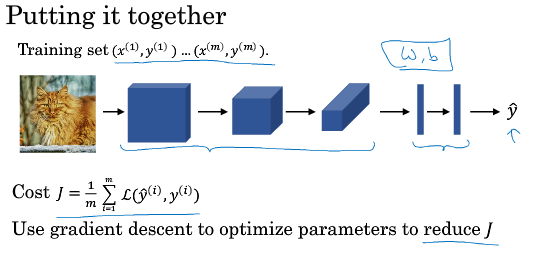
* 1. 为什么使用卷积

这一节讨论卷积在神经网络中应用好的原因，如何整合这些卷积，如何通过一个标注过的训练集训练卷积神经网络。

和只用全连接层相比，卷积层有两个主要优势在于参数共享和稀疏连接。例如上一节的例子，原始图像是32\*32\*3，卷积后28\*28\*6，如果是全连接，那么就有3072\*4704=140多万的参数，虽然以现在技术，可以用1400多万个参数来训练网络，但是原始图片算是小的，如果再大点呢，1000\*1000，权重矩阵就非常大。而如果使用卷积层，那么就有（5\*5+1）\*6=156个参数，卷积网络映射这么少参数有两个原因，一是参数共享，观察发现特征检测如垂直边缘检测如果适用于图片某个区域，那么也可能适用于图片的其他区域，每个特征检测器以及输出都可以在输入图片的不同区域使用同样的参数，以便提取垂直边缘或者其他特征。不仅适用于边缘检测这种低层次的特征，也适用于高阶特征，例如，提取脸上的眼睛。二就是使用稀疏连接，也就是说输出矩阵的某个值只跟原始图像的一部分像素点连接，其他像素点不会对输出产生任何影响。用这两种方式减少参数，以便于我们用更小的训练集来训练它。从而预防过拟合。

可能听过卷积网络善于不走translation invariance平移不变，通过观察可以发现，向右移动两个像素，图片中的猫依然清晰可见，因为卷积神经网络的机构，即使移动几个像素，这张图片仍然具有非常相似的特征，应该属于同样的输出标记。事实上我们用同一个过滤器生成各层中图片的所有像素值，希望网络通过自动学习变得更加健壮，以便更好的取得所期望的平移不变属性。这就是卷积或卷积网络在计算机视觉任务中表现良好的原因。





看看把所有层整合起来，如何训练，假设要构建一个猫检测器，有下列标记训练集，x表示一张图片，yhat是二进制标记，选定了一个神经网络，输入图片，增加卷积层和池化层，然后添加全连接层，最后输出一个softmax，即yhat，卷积层和连接层有不同的参数W和b，我们可以用任何参数集合来定义代价函数，一个类似于之前讲过的代价函数，并随机初始化参数W和b，Cost J等于神经网络对整个训练集的预测的损失总和/m，所以训练神经网络就是使用梯度下降法或者其他Momentum等来优化神经网络中的所有参数，以减少代价函数的值，通过这些操作就可以构建一个高效的猫咪检测器。

已经学习了卷积神经网络的基本模块，以及如何图片识别中高效整合这些模块。