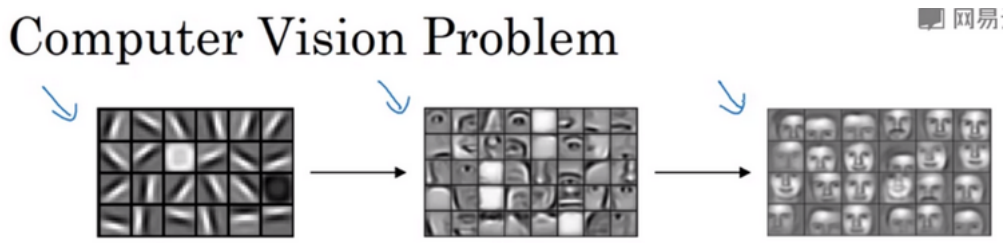
吴恩达老师深度学习视频课笔记：卷积神经网络

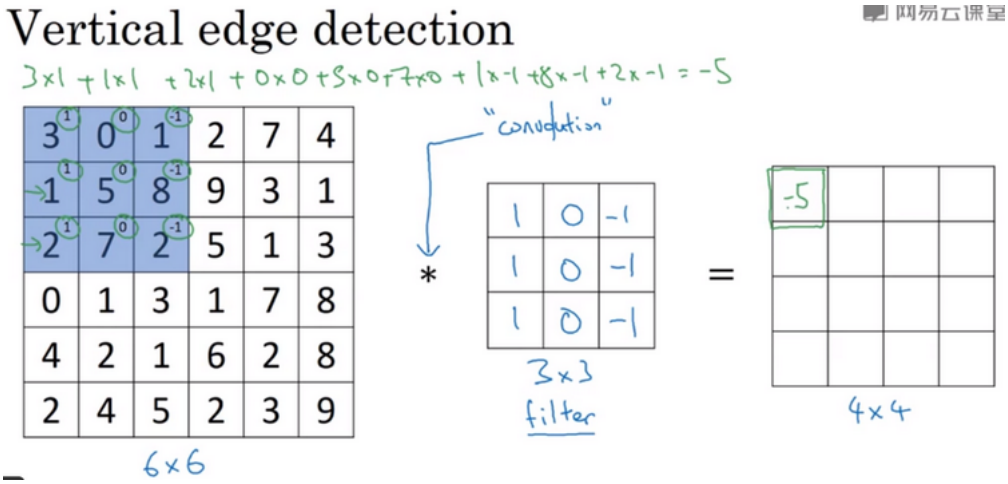
2018年05月10日 09:28:00 fengbingchun 阅读数：2236 更多

计算机视觉：包括图像分类(image classification)、目标检测(object detection)、风格迁移(neural style transfer)等等。

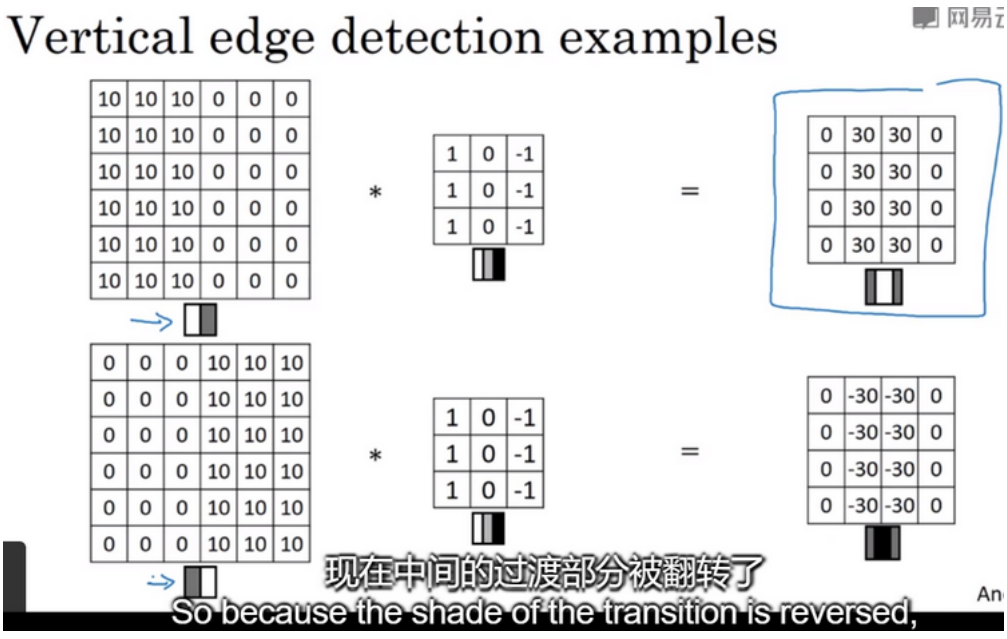
边缘检测示例：神经网络的前几层可以检测边缘，然后后面几层可能检测到物体的部分，接下来靠后的一些层可能检测到完整的物体，如下图示例：



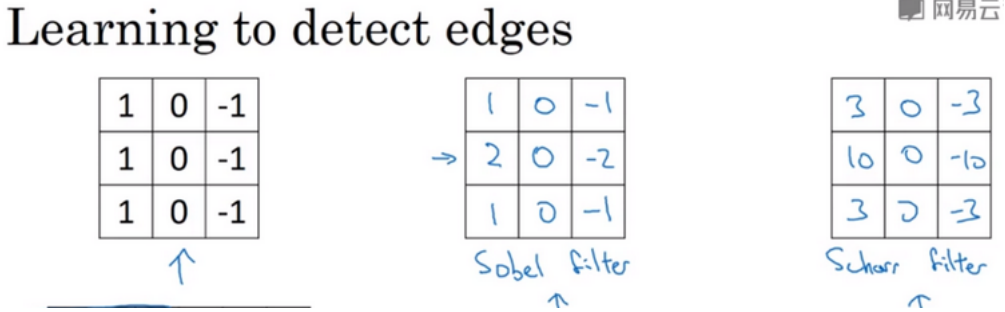
在卷积神经网络术语中，它被称为过滤器(filter)，在论文中，有时它被称为核(kernel)而不是过滤器(filter)。卷积运算过程，如下图，用卷积运算实现垂直边缘检测：



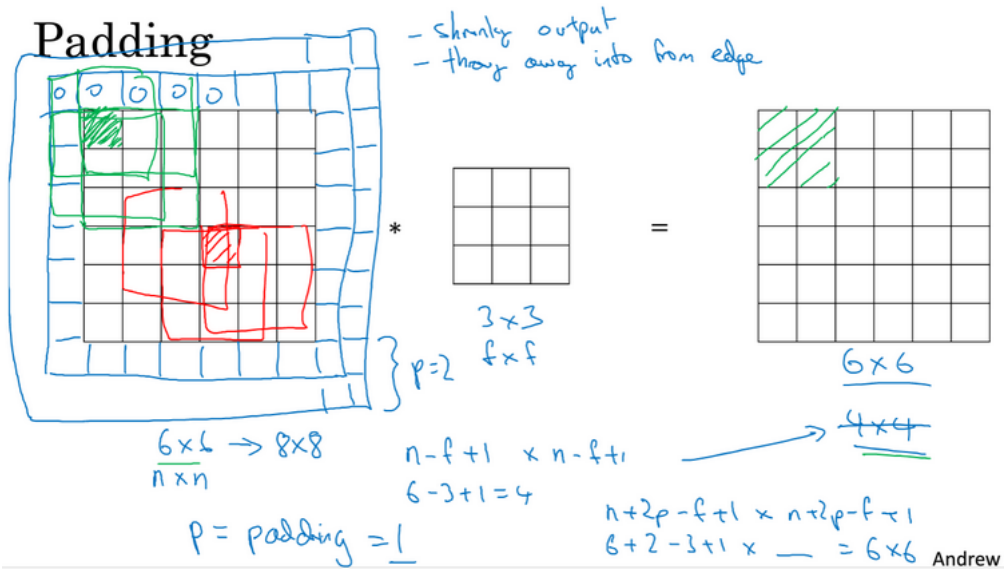
更多边缘检测示例：正边、负边其实就是由亮到暗与由暗到亮的区别，即边缘的过渡(edge transitions)，如下图：



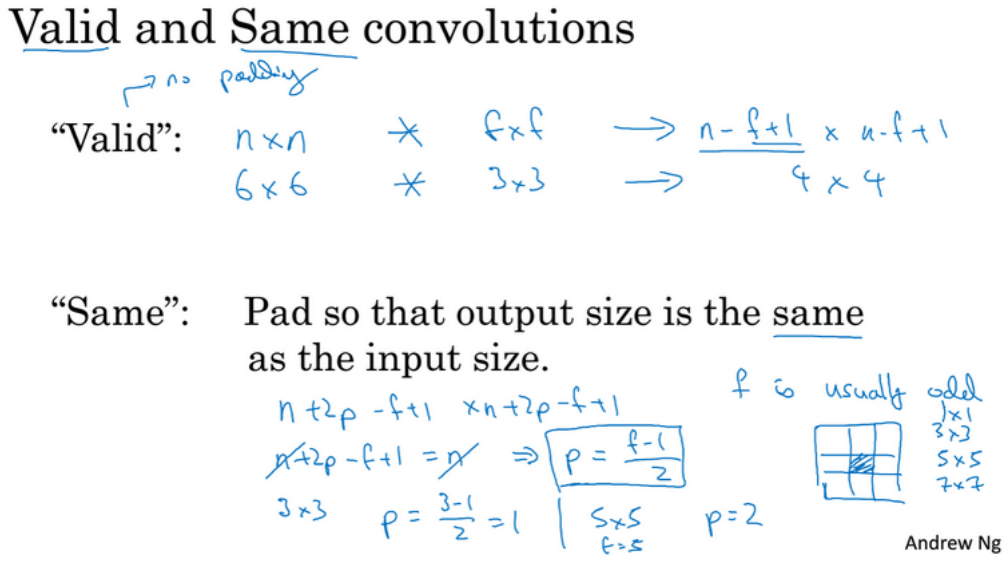
更多的过滤器，如下图：通过使用不同的过滤器，可以找出垂直或水平的边缘。还有其它过滤器，如Sobel过滤器、Scharr过滤器。一般将垂直过滤器，顺时针翻转90度，就会得到水平过滤器。一般垂直过滤器，左边是正值，中间是0，右边是负值；而一般水平过滤器，上边是正值，中间是0，下边是负值。在深度学习中，你不一定要去使用研究者们推荐的这些过滤器，可以把矩阵中的这9个数字，当成9个参数，并且在之后可以学习使用反向传播算法，其目标就是理解这9个参数，通过反向传播，你可以学习另一种滤波器，这种滤波器对于数据的捕捉能力，甚至可以胜过之前任何的滤波器(单纯的水平边缘和垂直边缘)，它可以检测出45度、75度或73度，甚至是任何角度的边缘。



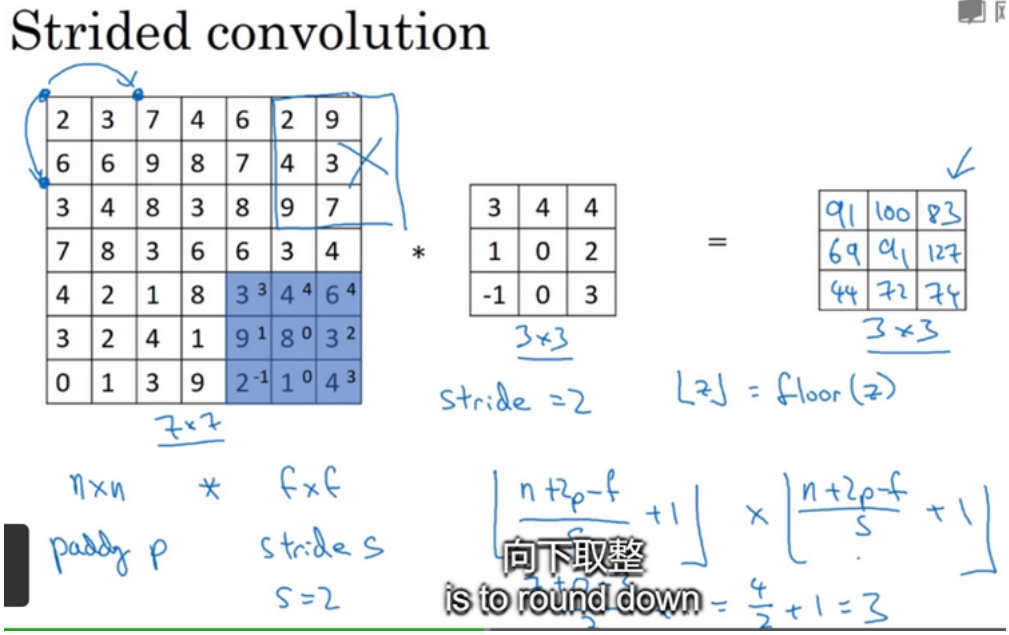
Padding：有一个n*n的图像，用一个f*f的过滤器做卷积，那么输出的结果维即大小就是(n-f+1)*(n-f+1)。按照这种运算会有2个缺点：第一个缺点是每次做卷积操作，你的图像就会缩小，作了几次卷次操作，可能会缩小到1*1的大小；第二个缺点是，边角的像素，这个像素点只会被一个输出所触碰或者使用，但是如果在中间的像素点，就会有许多的区域与之重叠，所以那些在角落或者边缘区域的像素点在输出中采用较少，意味着你丢掉了图像边缘位置的许多信息。为了解决这两个问题，你可以在卷积操作之前填充所处理的图像，可以沿着图像边缘，在填充一层像素，如下图，卷积后会得到和原始图像一样大小的图像。习惯上，你可以用0填充，如果p是填充的数量，那么输出就变成了(n+2p-f+1)*(n+2p-f+1)。

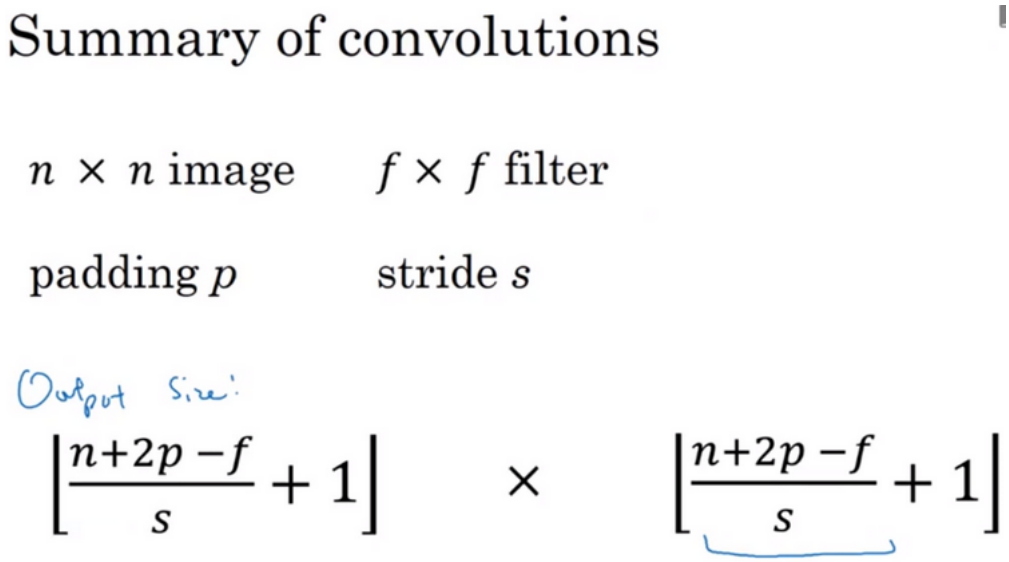


至于选择填充多少个像素，通常有两个选择，分别叫做Valid卷积和Same卷积。Valid卷积意味着不填充(no dapping)，即p=0。Same卷积意味着填充后你的输入大小和输出大小是一样的，即p=(f-1)/2,如下图，习惯上，计算机视觉中，f通常是奇数。

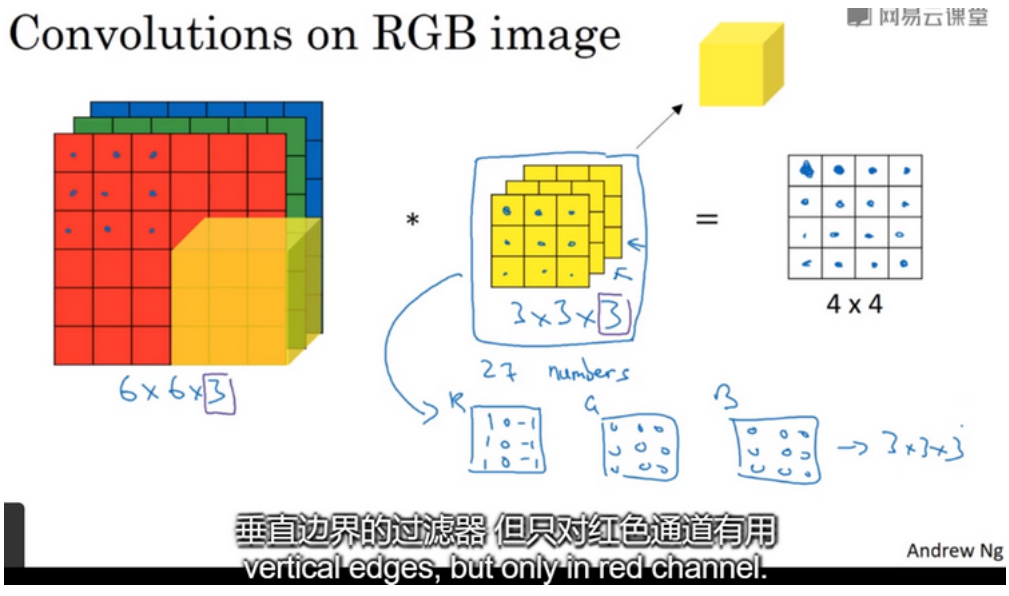


卷积步长：如下图，假如输入图像为n*n，过滤器为f*f，padding为p，步长(stride)为s，则输出大小为((n+2p-f)/s+1)*((n+2p-f)/s+1)。如果商不是整数，我们向下取整，即floor函数，这个原则实现的方式是，你只在篮框完全包括在图像或填充完的图像内部时才对它进行运算。如果有任意一个蓝框移动到了外面，那么你就不要进行相乘操作，这是一个惯例。

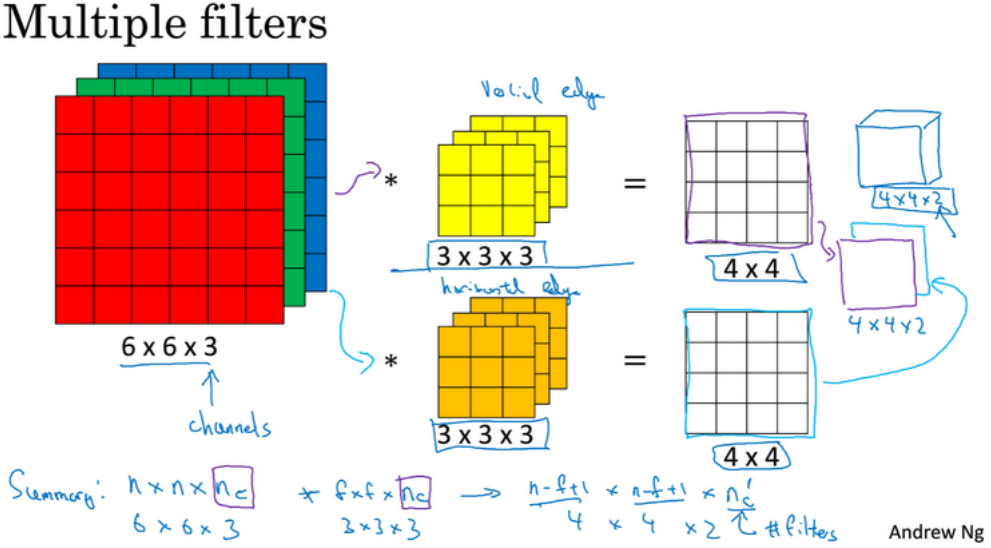




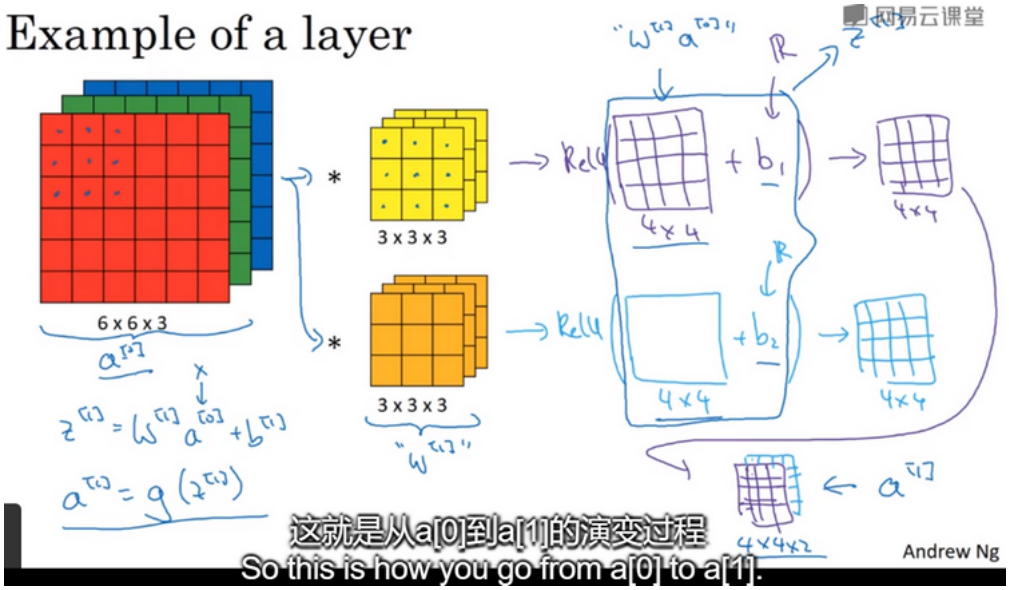
Convolutions over volumes：在BGR图像上进行卷积操作，如下图，依次取过滤器这27个数，然后乘以相应的红、绿、蓝通道中的数字，然后把这些数加起来，就得到了输出的数。图像的通道数必须和过滤器的通道数一致。过滤器的参数选择不同，你就可以得到不同的特征检测器。按照计算机视觉的惯例，当你的输入有特定的高、宽和通道数时，你的过滤器可以有不同的高、不同的宽，但是必须有一样的通道数。理论上，我们的过滤器，只关注红色通道、或者只关注绿色通道是可行的。



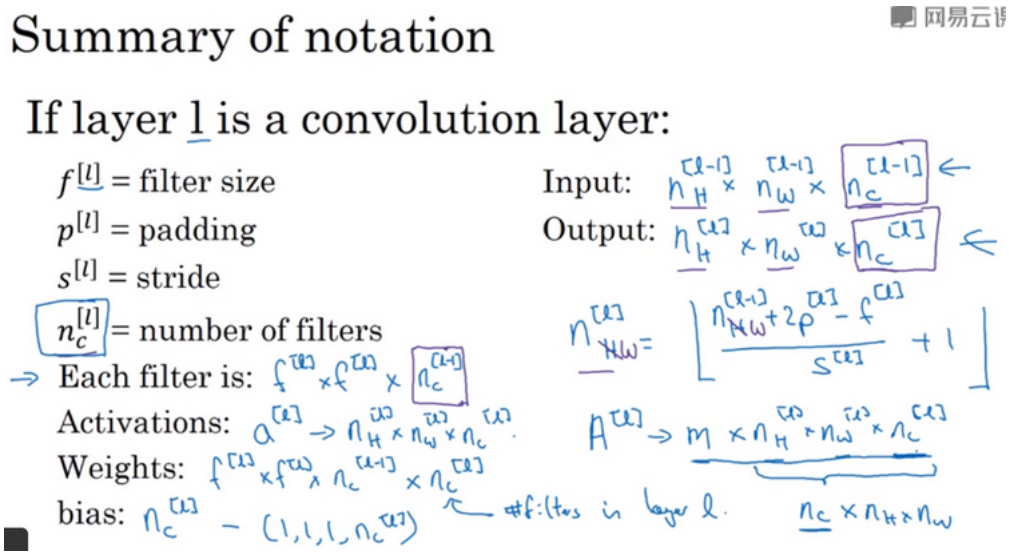
Multiplefilters: 如下图，可以同时使用两个过滤器，其中一个过滤器可能用来检测垂直边缘，另一个过滤器用来检测水平边缘，输出结果为4*4*2。如果你有一个$n*n*n_c$的输入图，然后卷积上一个$f*f*n_c$的过滤器，然后得到一个$(n-f+1)*(n-f+1)*n'_c$的输出，其中n'_c为过滤器的个数。在上面这个式子中是假设步长为1并且没有padding。



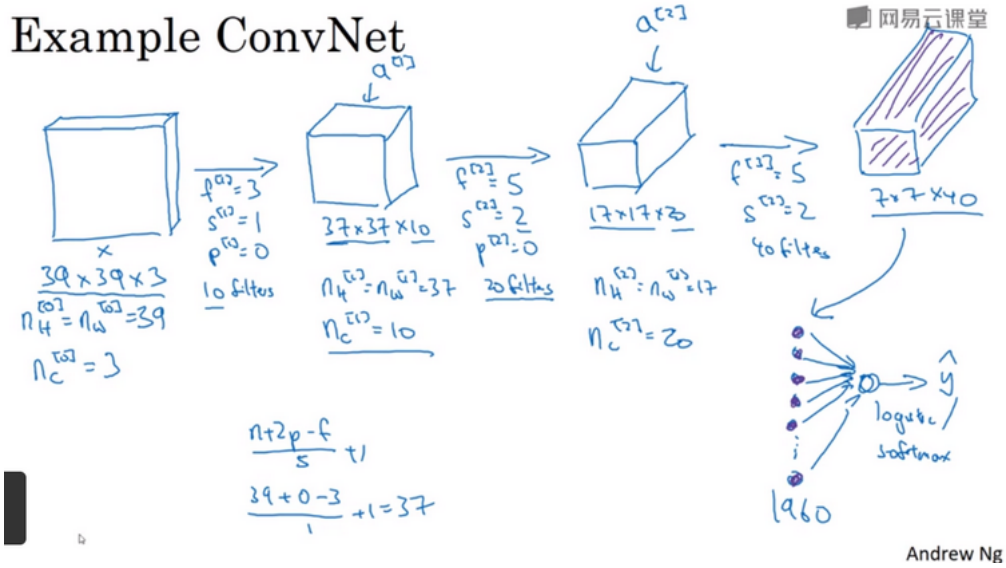
单层卷积网络：如下图，在6*6的BGR图像上进行卷积操作，有2个3*3*3的过滤器，通过卷积后产生2个4*4的结果，在此结果上加上偏差(bias)，再应用非线性激活函数ReLU，再把两个结果矩阵堆叠起来，最终得到一个4*4*2的矩阵。这就是卷积神经网络的一层。



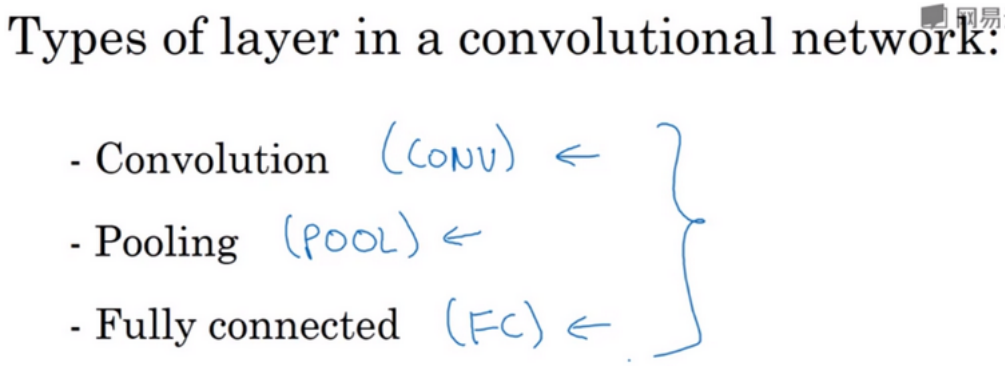
输出图像中的通道数量就是神经网络中这一层所使用的过滤器数量。过滤器中通道的数量必须与输入中通道的数量一致。每个过滤器都有一个偏差参数，它是一个实数。如下图：



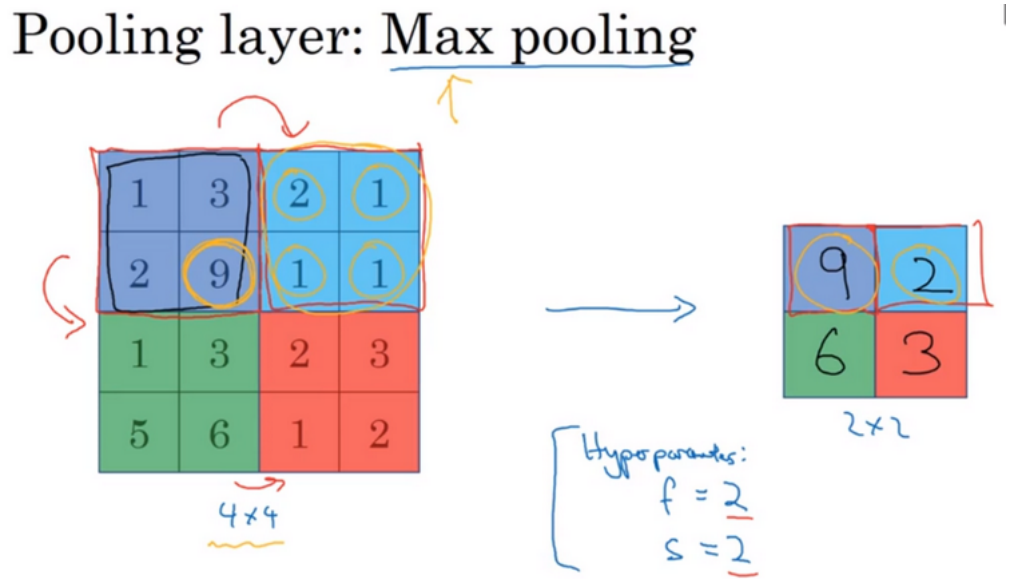
简单卷积网络示例：假设有一个卷积网络，用来识别输入的图像中是否含有猫。输入图像的大小为39*39*3，第一层卷积层用3*3的filter来检测特征，stride为1，padding为0，这层共有10个filters，这层输出将是37*37*10。第二层也是卷积层用5*5的filter，stride为2，padding为0，这层共有20个filters，输出将是17*17*20。最后一层卷积层，用5*5的filter, stride为2，这层共有40个filters，输出将是7*7*40，即1960特征，可以将其平滑(flatten)或展开(unroll)成1960个单元，即输出一个长向量，那时和logistic回归或softmax进行计算最终得出神经网络的预测输出，如下图所示。随着神经网络计算深度不断加深，通常开始时图像会较大，高度和宽度会在一段时间内保持一致，然后随着网络深度的加深而逐渐减少，而通道数(numberof channels)在增加。在其它许多卷积神经网络中也有相似操作。



一个典型的卷积网络通常有三种类型的层：一个是卷积层(Convolution)，通常用Conv来标注；一个是池化层(Pooling)，经常叫做POOL；还有一个是全连接层(Fully connected)，用FC表示。虽然仅用卷积层也有可能构建出很好的神经网络，但大部分神经网络架构师依然会添加池化层和全连接层。一般池化层和全连接层比卷积层更容易设计。如下图所示：

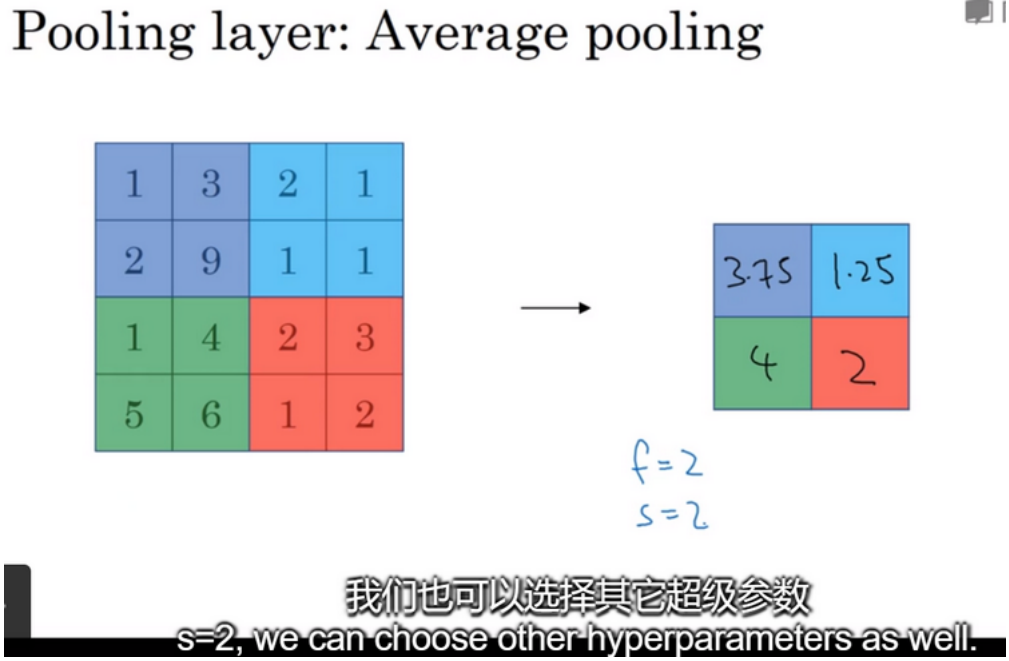


池化层：除了卷积层，卷积网络也经常使用池化层来缩减模型的大小，提高计算速度，同时提高所提取特征的鲁棒性。池化类型有最大池化(maxpooling)，如下图所示，在此例中，filter的大小为2，stride位2 ，这两个是最大池化的超参。最大池化运算的实际作用就是：如果在过滤器中提取到某个特征，那么保留其最大值；如果没有提取到某个特征，可能不存在这个特征，那么其中的最大值也还是很小。

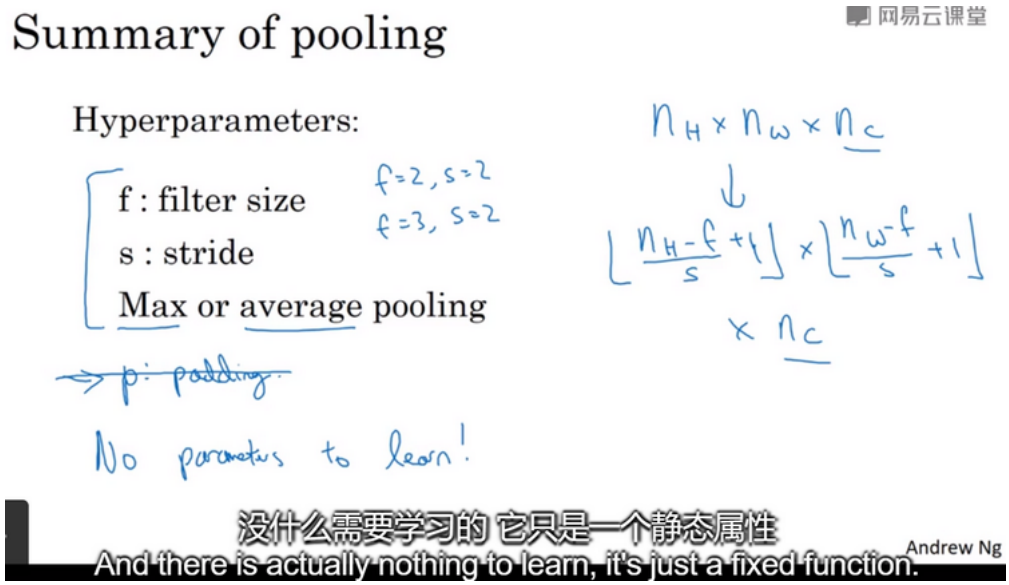


计算卷积层输出大小的公式同样适用于最大池化：((n+2p-f)/s)+1，这个公式也可以计算最大池化的输出大小。计算最大池化的方法就是分别对每个通道执行相同的计算过程，n_c个通道中每个通道都单独执行最大池化运算。

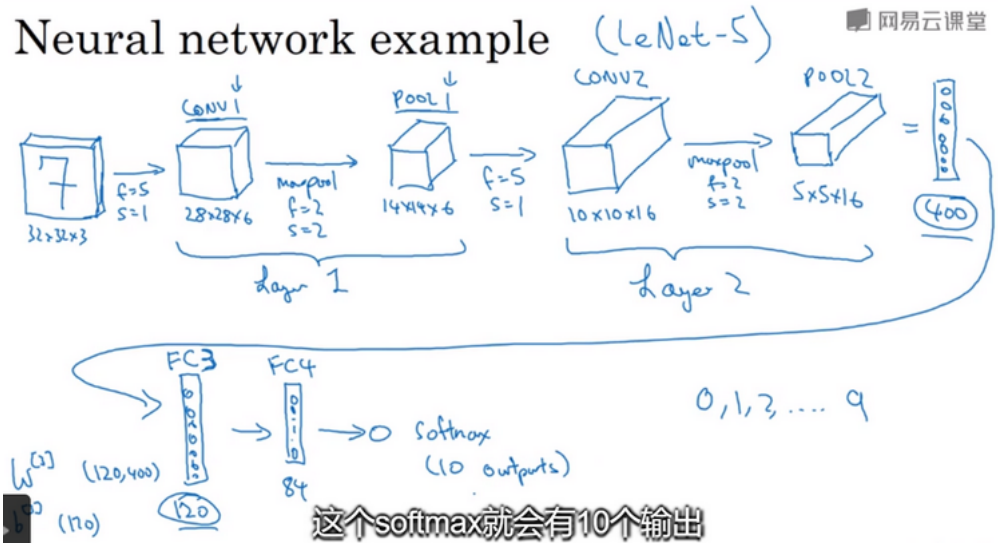
平均池化：另外一种类型的池化，它不太常用，选取的不是每个过滤器的最大值，而是平均值，如下图所示：



池化的超级参数包括过滤器大小(filter size)和步长(stride)。其中f=2,s=2应该频率比较高，其效果相当于高度和宽度缩减一半。你也可以根据自己的意愿增加表示padding的其它超级参数，但是很少这么用。最大池化时，往往很少用到超级参数padding，当然也有例外情况。最大池化的输入是$n_h*n_w*n_c$，假设没有padding，输出为$((n_h-f/s)+1)*((n_w-f/s)+1)*n_c$，如下图所示。需要注意的一点是，池化过程中没有需要学习的参数。反向传播没有参数适用于最大池化。最大池化只是计算神经网络某一层的静态属性。



卷积神经网络示例：假设有一张32*32*3的输入图像，如下图所示,用于手写体数字识别，想识别它是从0到9这10个数字中的哪一个。让我们来构建一个神经网络来实现这个功能。此网络结构和LeNet-5非常相似。假设第一层使用filter大小为5*5，stride为1，padding为0，filter的个数是6，那么输出为28*28*6，将这层标记为CONV1，它有6个filter，增加了bias，应用了非线性函数(激活函数)，可能是ReLU，最后输出CONV1的结果。然后构建一个池化层，选用最大池化，超参f=2,s=2,padding=0，最终输出为14*14*6，将这层标记为POOL1。在卷积网络文献中，卷积有两种分类，一类卷积是一个卷积层和一个池化层一起作为一层；另一类卷积是把卷积层作为一层，而池化层单独作为一层。人们在计算神经网络有多少层时，通常只是统计具有权重和参数的层，因此池化层没有权重和参数，只有一些超级参数。这里采用的是把CONV1和POOL1共同作为一个卷积，并标记为Layer1。接着在构建一个卷积层，filter为5*5，stride为1，padding为0，使用16个filter，输出一个10*10*16的矩阵，标记为CONV2，然后最大池化，f=2,s=2,输出为5*5*16的矩阵，标记为POOL2，这是Layer2。现在将POOL2平整化为一个大小为400(即5*5*16)的一维向量。然后利用这400个单元构建下一层。下一层有120个单元，这是第一个全连接层，标记为FC3，此连接层的权重$W^{[3]}$为(120,400),$b^{[3]}$为(120,1)。接着再添加一个全连接层，有84个单元，标记为FC4。最后用这84个单元填充一个softmax单元，这个softmax会有10个输出。

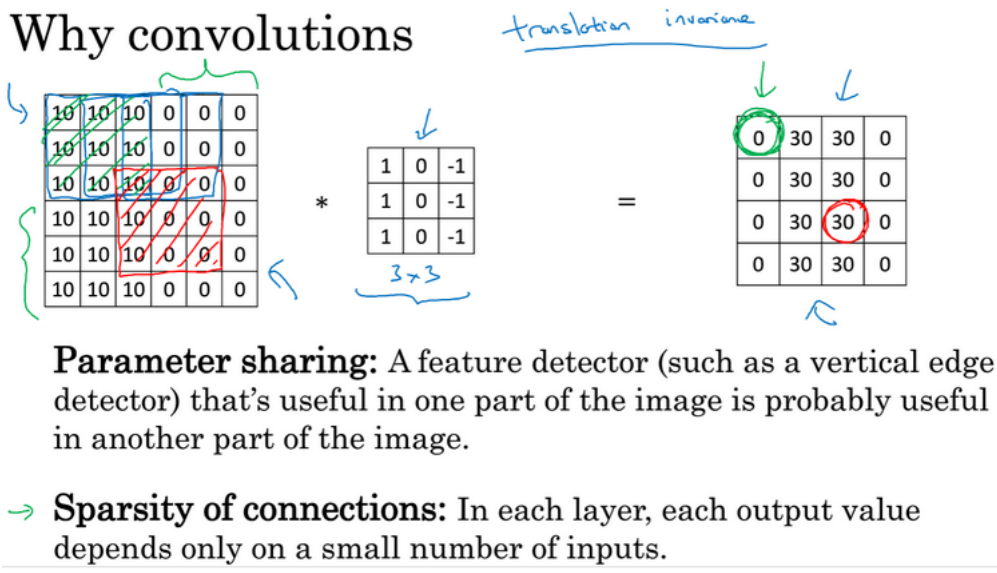


在神经网络中，另一种常见模式就是一个或多个卷积层后跟随一个池化层，然后一个或多个卷积层后再跟一个池化层，然后是几个全连接层，最后是一个softmax。

神经网络的激活值形状(activation shape)、激活值大小(activation size)和参数数量：如上例以32*32*3作为输入的神经网络架构，如下图所示：注意事项：(1)、池化层和最大池化层没有任何参数；(2)、卷积层的参数相对较少；(3)、许多参数都存在于神经网络的全连接层；(4)、观察可发现，随着神经网络的加深，激活值size会逐渐变小，如果激活值size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。

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

Why convolutions：和只用全连接层相比，卷积层的两个主要优势在于：参数共享(parameter sharing)和稀疏连接(sparsity of connections)。如下图所示：



GitHub：https://github.com/fengbingchun/NN_Test