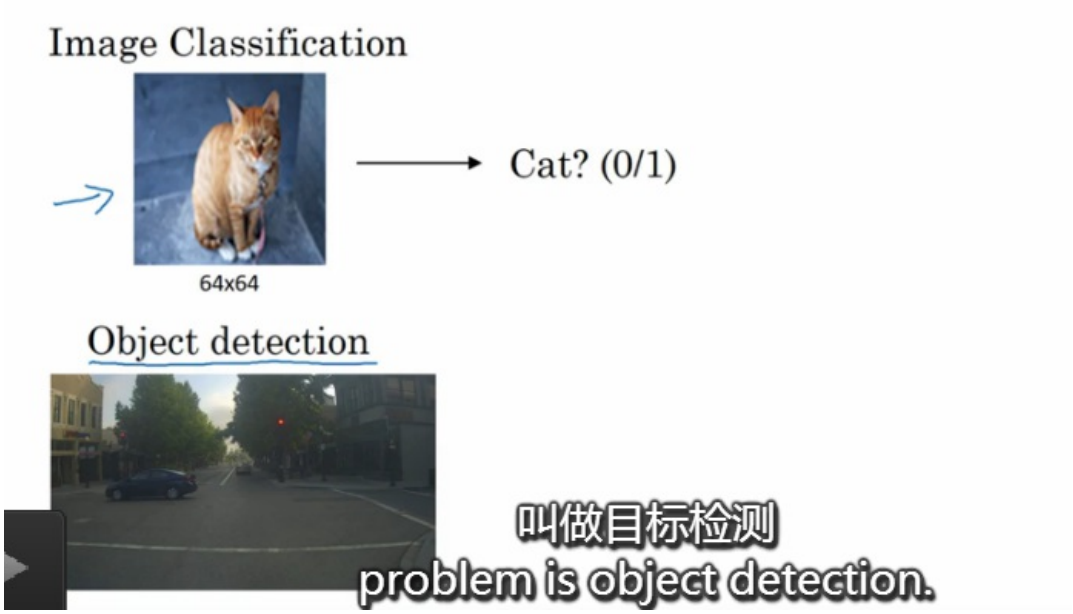
第四篇第一周

第四篇第一周.note1、计算机视觉

例子：图片分类，图片检测



风格迁移：

在应用计算机视觉时，要面临一个挑战，就是输入的数据可能非常的大，当64*64*3的图片的话，特征向量的维度是12288，但是1000*1000*3的图片的话，它足足有1M那么大，特征向量的维度是1000*1000*3，这个数字达到300万。如果你要输入300万的数据量，这就意味着特征向量X的维度达到300万，假设第一个隐藏层有1000个节点，如果使用全连接网络，W[1]矩阵的维度将是

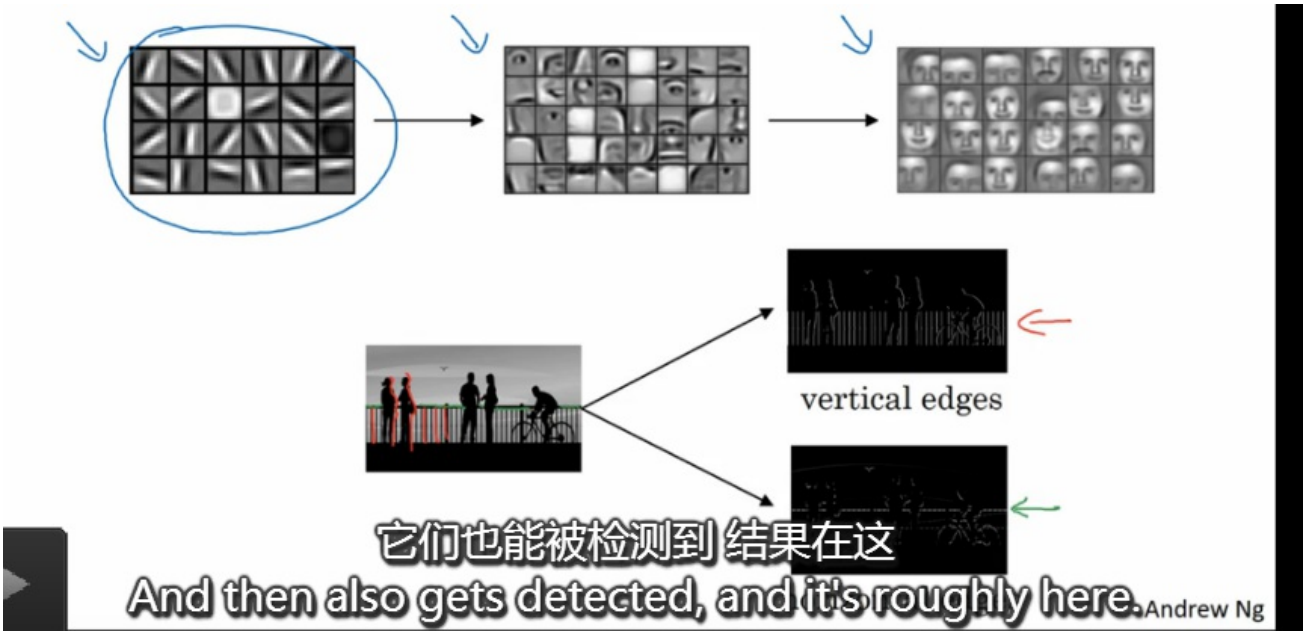
[300万，1000]，这就意味着隐藏层1就有30亿个参数，这么多参数，难以获得足够的图片来防止神经网络发生过拟合，同时处理30亿个参数的神经网络，对内存的需求也是让人不能接受的。

2、边缘检测示例

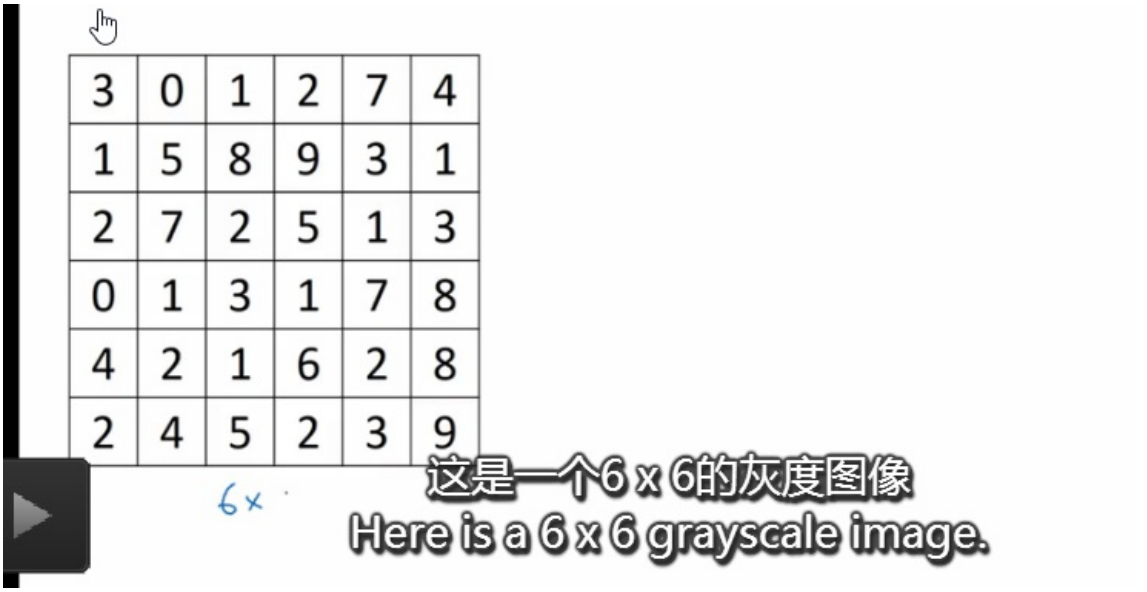
卷积运算，是卷积神经网络的最基本的组成部分，使用边缘检测作为入门样例，这节你将看到卷积神经网络是如何运算的。

例子：

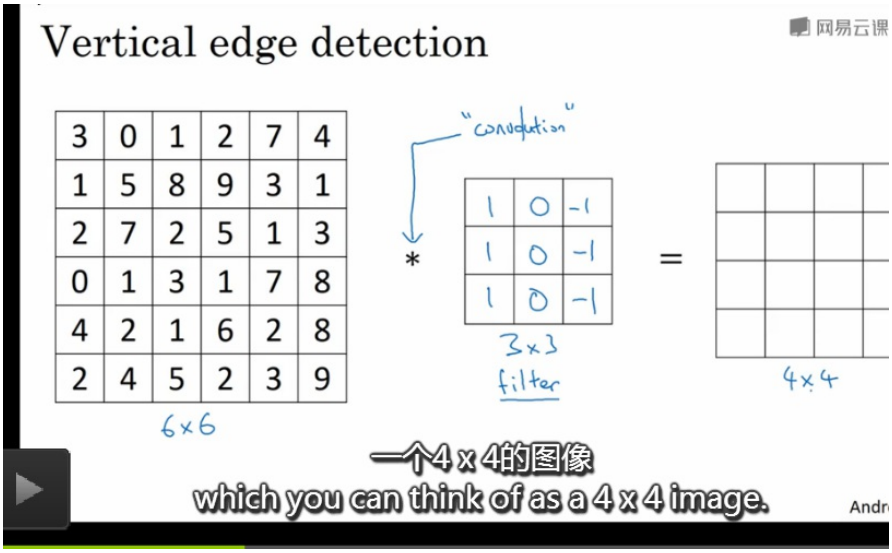
给你一张图片，检测这张图片里面有什么物体，你可能做的第一件事情就是检测图片中垂直的边缘，然后检测水平。



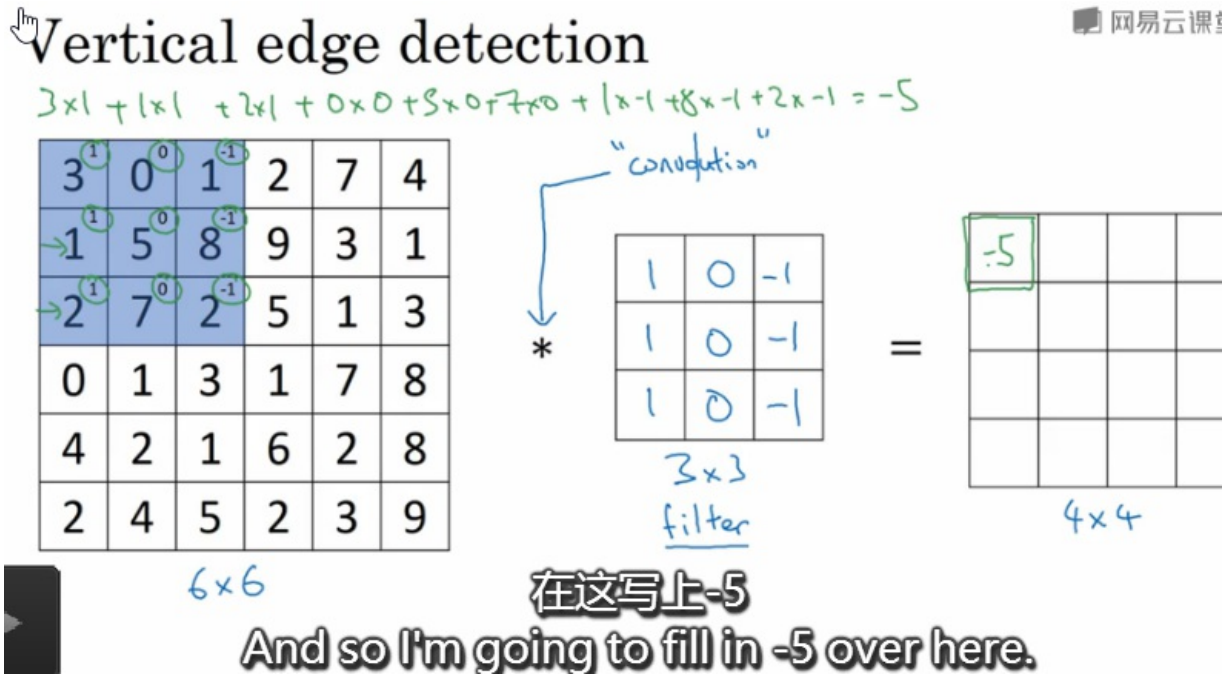
如何在图片中检测边缘呢，再看一个例子：

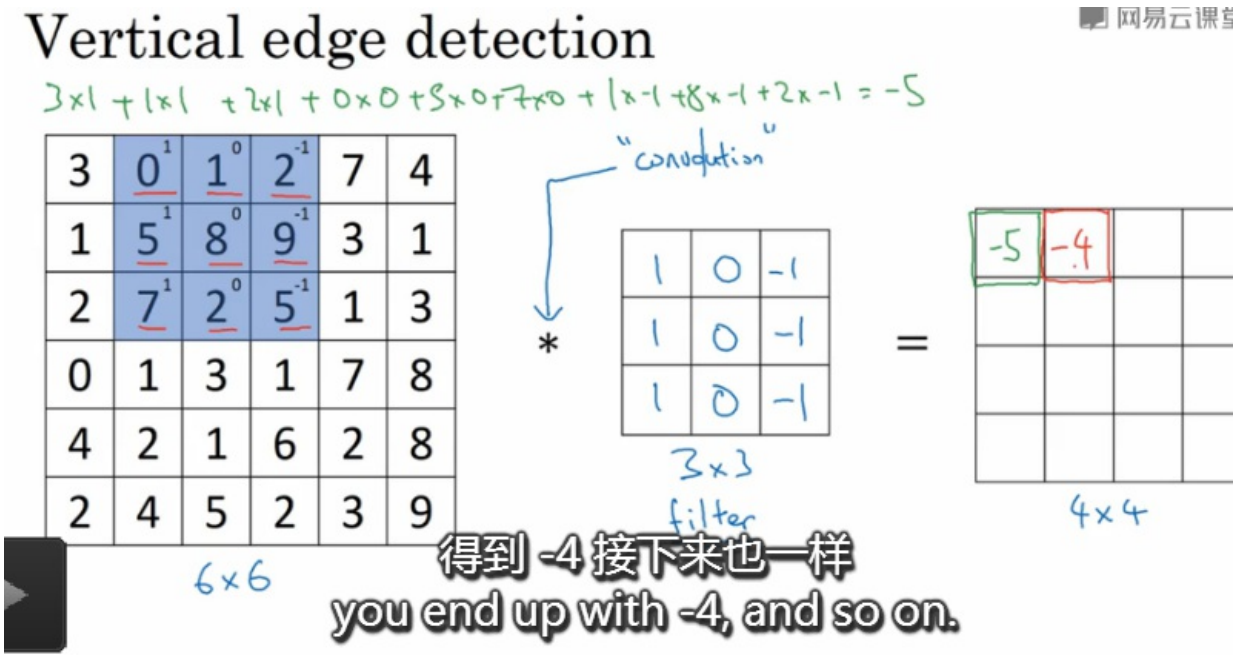


6*6的灰度图，通道为1，检测垂直边缘，我们可以构造出3*3的矩阵，也就是3*3的过滤器，然后使用filter对图片的像素矩阵进行卷集运算，结果为4*4的矩阵：



卷集运算过程，filter扣到上面去，从左上角开始扣，然后filter跟对应的像素矩阵每一个数相乘求和得到-5,假设步长为1，一次类推，

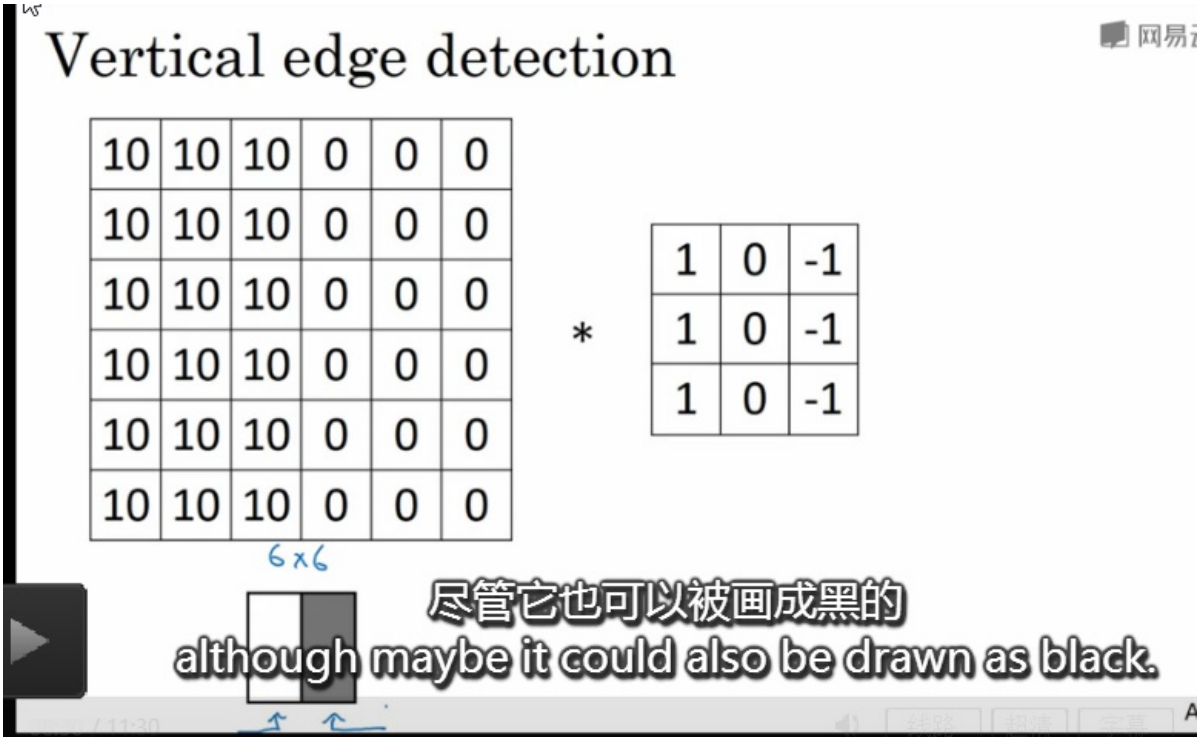




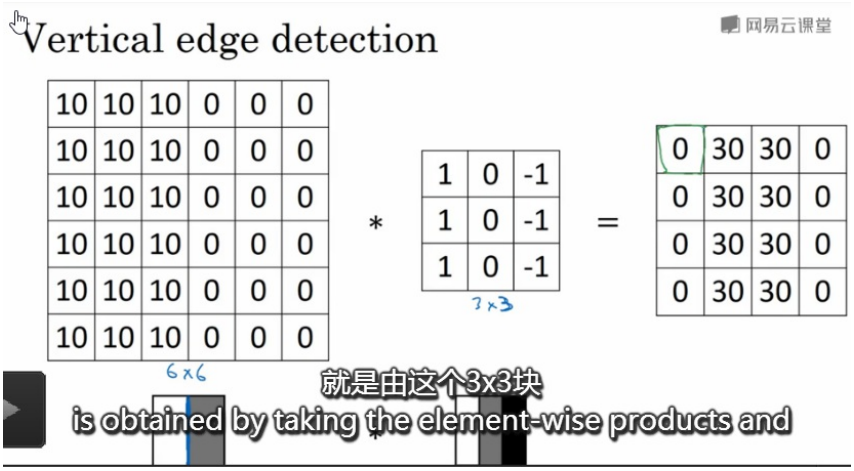
结果矩阵大小=（原始像素矩阵大小-filter大小）/步长+1

为什么相当于做了垂直检测呢？

再看一个例子：中间有一条很明显的过度线



得到的结果：

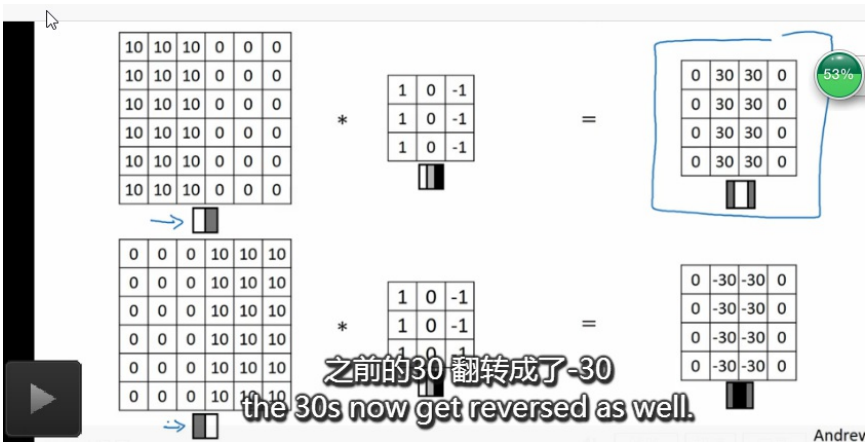


得到最右边的图，中间比较亮，相当于检测到了边缘，

结论：卷积运算，提供了一个方便的方法，来发现图像中的垂直边缘，

3、更多边缘检测例子

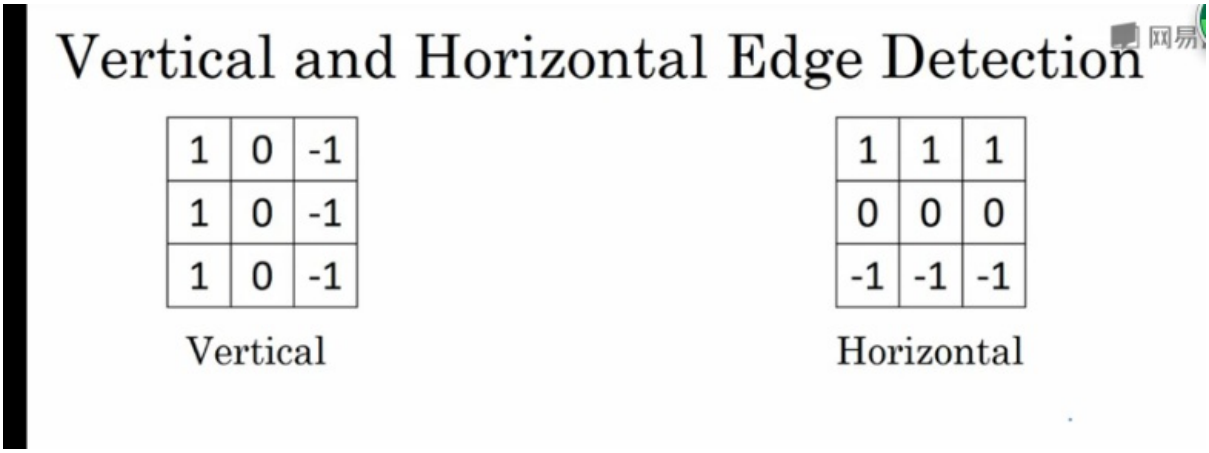
学会区分正边和负边，这就是由亮到暗和由暗到亮的区别，也就是边缘过渡



这种特殊的过滤器，确实可以为我们区分两种明暗变化的区别。

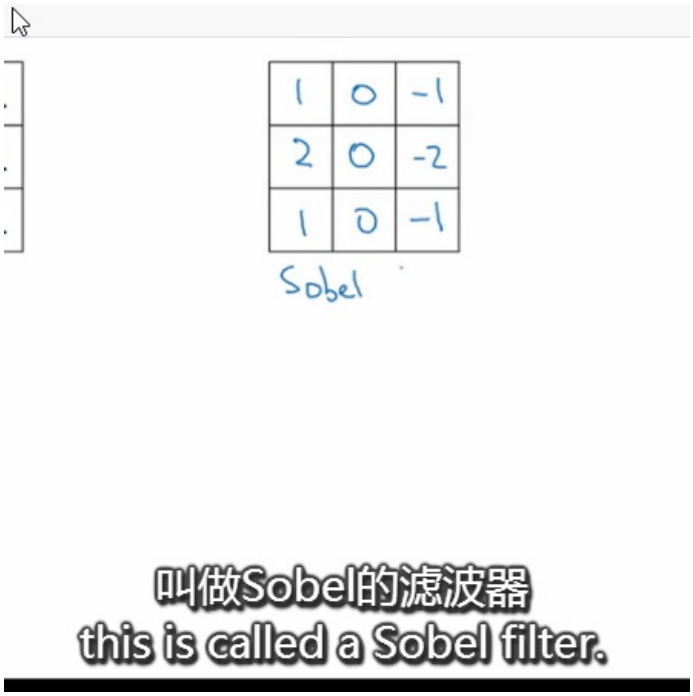
再来看看其他例子：

下图的左边这种过滤器可以检测出垂直的边缘，右边的过滤器可以检测出水平的边缘。过滤器的数值大小表示亮度，数值越大表示亮度越亮。

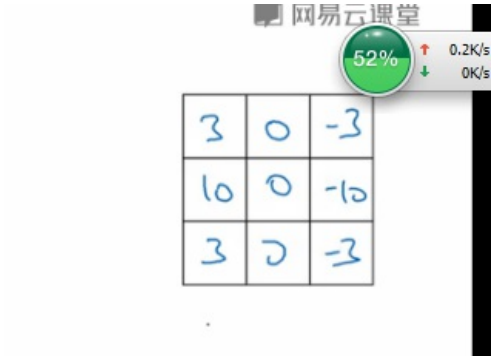


还有一个更复杂的例子：

Sobel过滤器：优点在于，增加了中间一行元素的权重，对于图片中间的像素点，鲁棒性会更好，



还有一种常用的过滤器Scharr过滤器

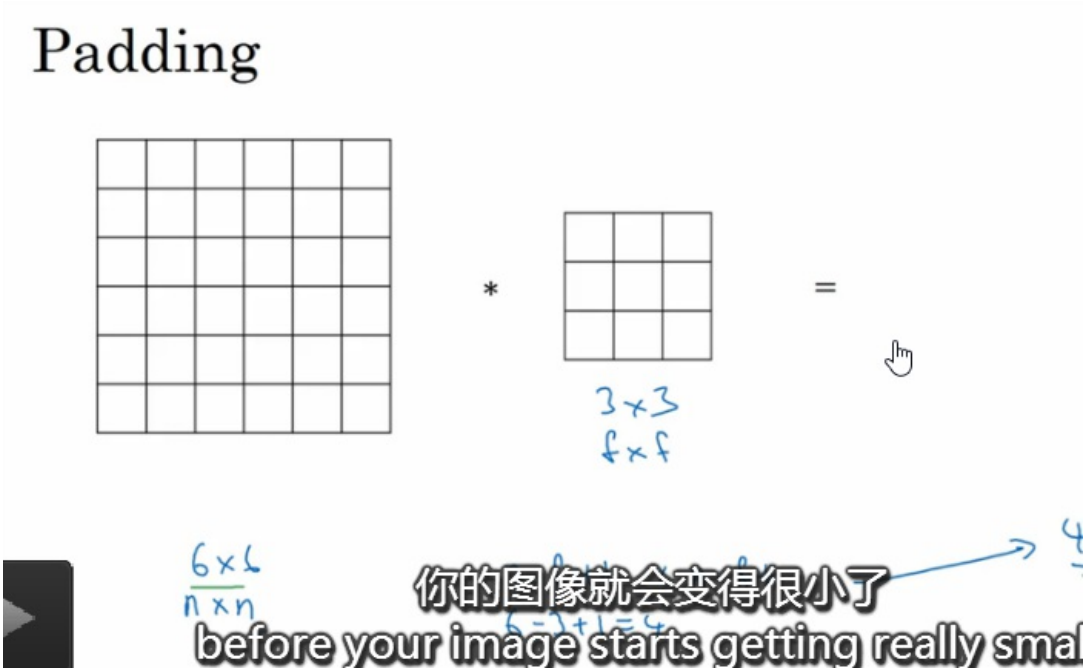


随着深度学习的发展。我们学习的其中一件事，就是当你真的想去检测出复杂图片的边缘，你并不一定要去使用，那些研究者所选择的这九个数，但是你可以从中受益匪浅，把这矩阵的九个数，当成9个参数，并且在之后，你可以学习使用反向传播算法，其目标就是去理解这9个参数，这样训练出来的过滤器可以胜过任何一种手写的像上面那样的过滤器，相比上面那种单纯的水平边和垂直边它可以检测出45度或者75度等任何角度的边缘，所以，将过滤器矩阵的所有数字，都设置成参数，通过反向传播算法，让神经网络自动去学习他们，你会发现神经网络，可以学习一些低级的特征，构成这些算法基础的，依然是卷积运算，使得反向传播算法能够让神经网络，学习任何它所需要的3*3的过滤器，并在整幅图像上应用它。

4、Padding

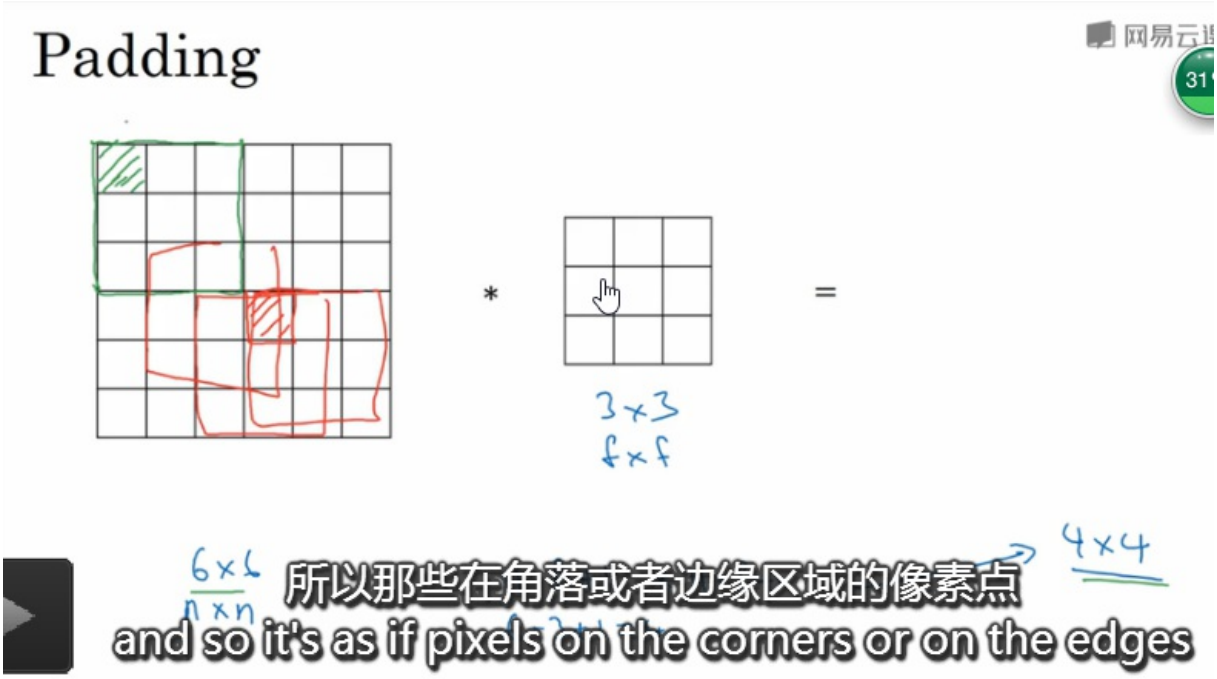
如果我们有一张n*n的图片，用一个f*f的过滤器做卷积，那么输出结果的维度就是(n-f+1)*(n-f+1).

这样有两个缺点，第一个缺点是每次做卷积操作，你的图像都会缩小，比如6*6的变成4*4的：

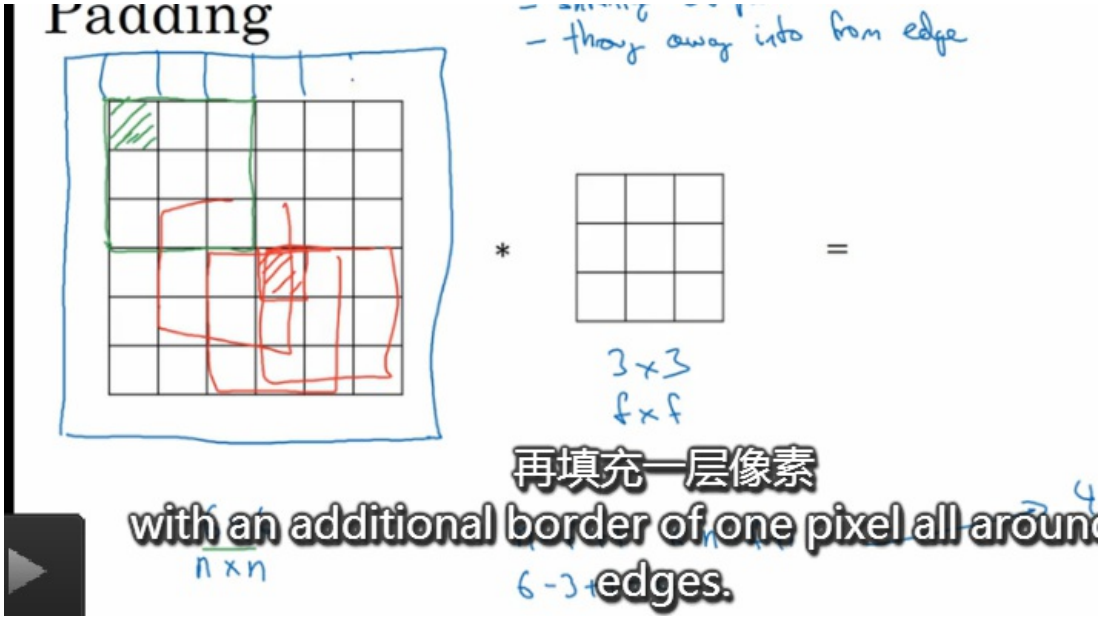


你可不想让你的图片每次做完一个边缘检测时都变小。

第二个缺点就是，你注意到角落边缘的像素，这个像素点只被一个输出所触碰或者说使用，



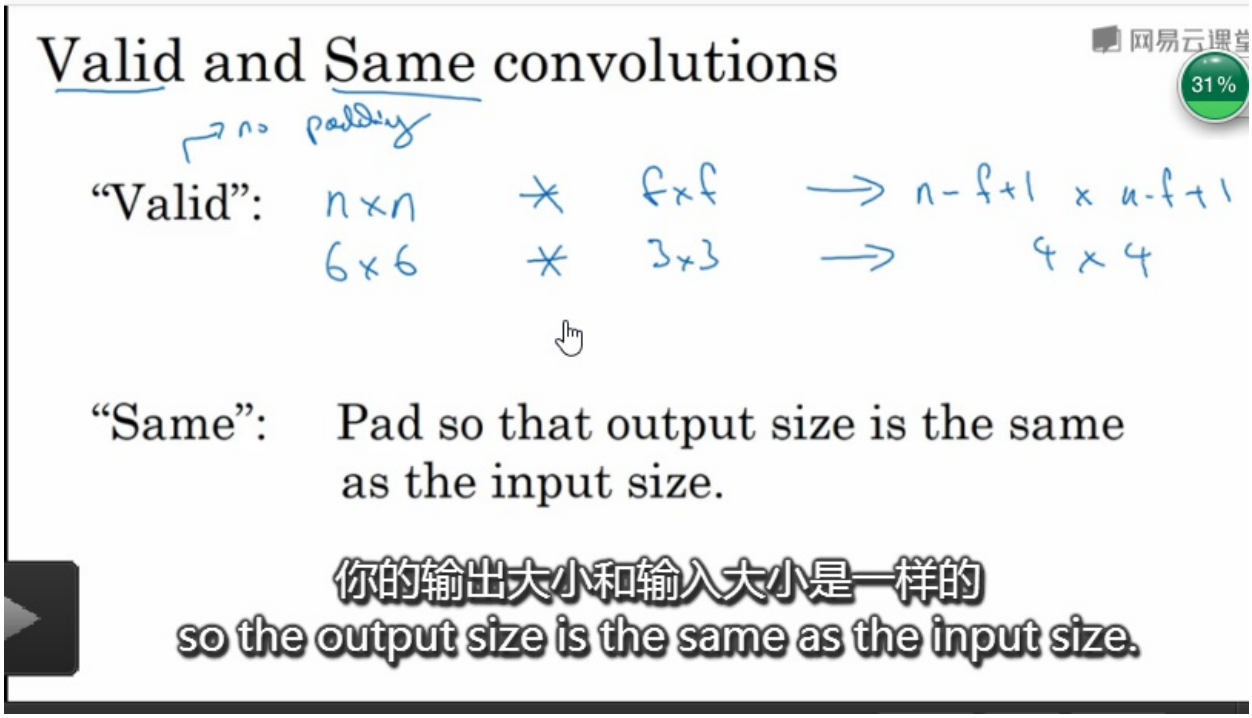
所以那些角落或者边缘区的像素点，在输出中采用较少，意味着你丢掉了图片边缘位置的许多信息，为了解决这两个问题，一是输出缩小，当我们建立深度神经网络时，你为什么不希望没进行一步操作，图片都变小，比如你有100层的神经网络，如果图片没进去一层都要缩小的话，经过100层之后，你将得到一个很小的图像。另外一个问题是图片边缘的大量信息，都丢失了，为了解决这些问题，你可以在卷积操作之前，填充这幅图片



6*6的就变成8*8的，用3*3的过滤器卷积之后还是6*6的

填充方式可以是0填充，其实如果你想的话，也可以用两个像素点去填充。

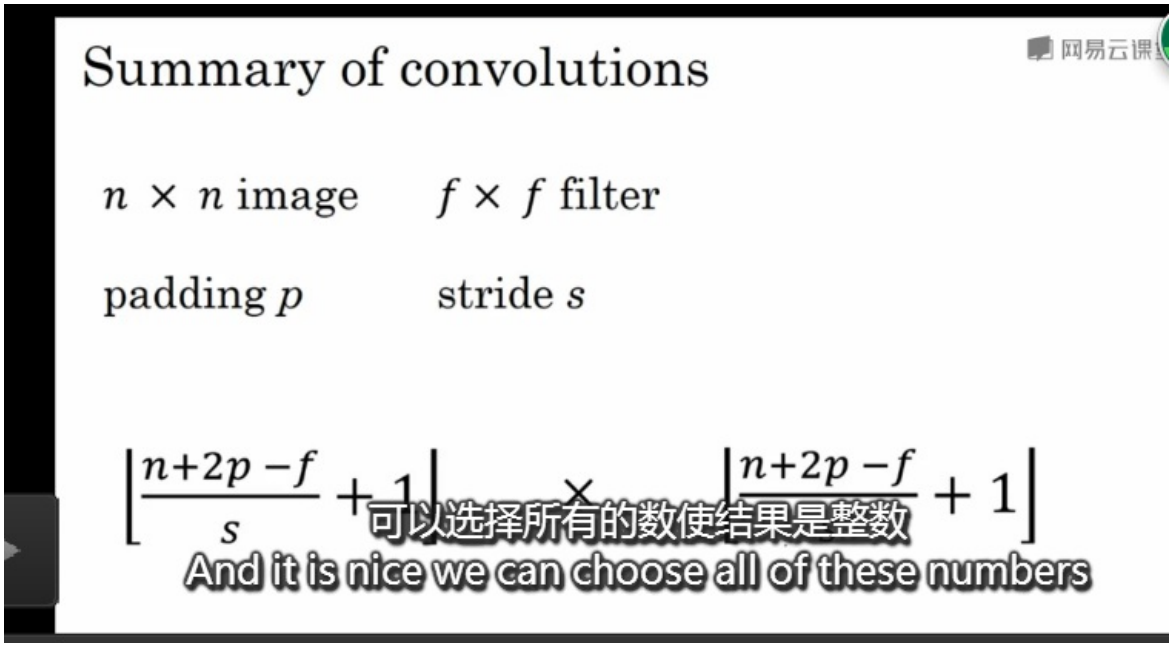
两种填充方式：Valid和Same填充，Valid意味着不填充，Same意味着你的输出大小和输入大小是一样的。

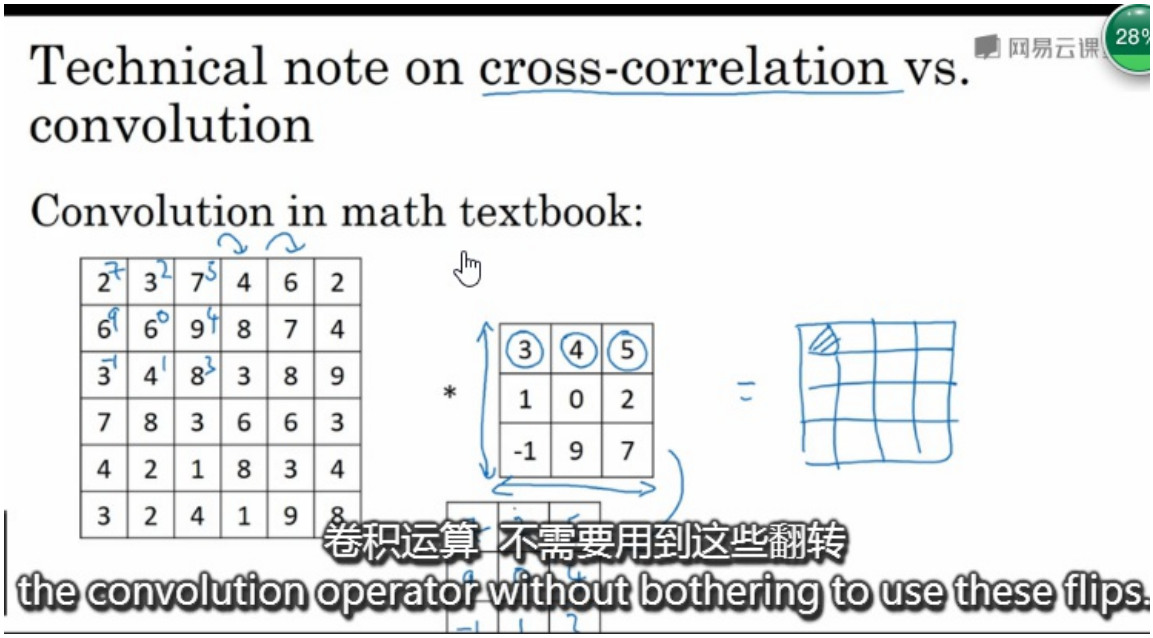


填充的p=(f-1)/2才能保证输入更输出是一致的，在计算机视觉中，f通常是奇数的，有两个原因，如果f是一个偶数，那么你只能使用一些不对称填充，只有在奇数条件下，才能自然的填充，另外一个原因是当你有个奇数维度的过滤器，他就有一个中心点，在计算机视觉中，如果有一个中心点会更方便，便于指出过滤的位置。

5、卷积步长

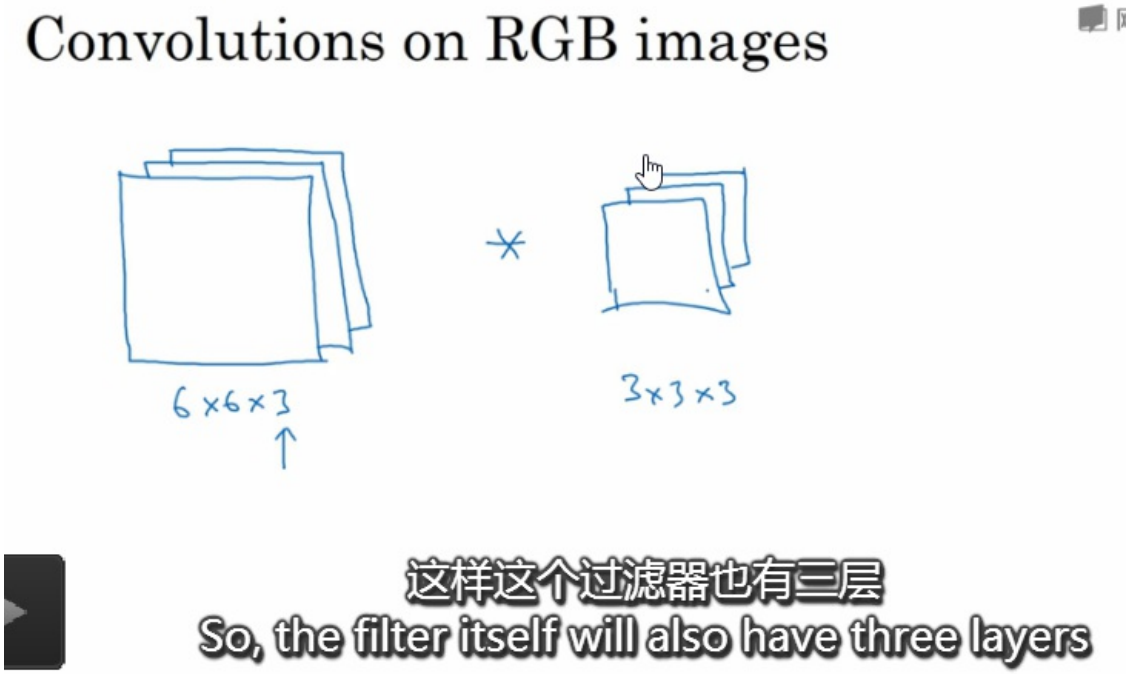
假设图片大小为n*n，卷积核为f*f，padding为p，步长s=2，输出为(n+2p-f)/2



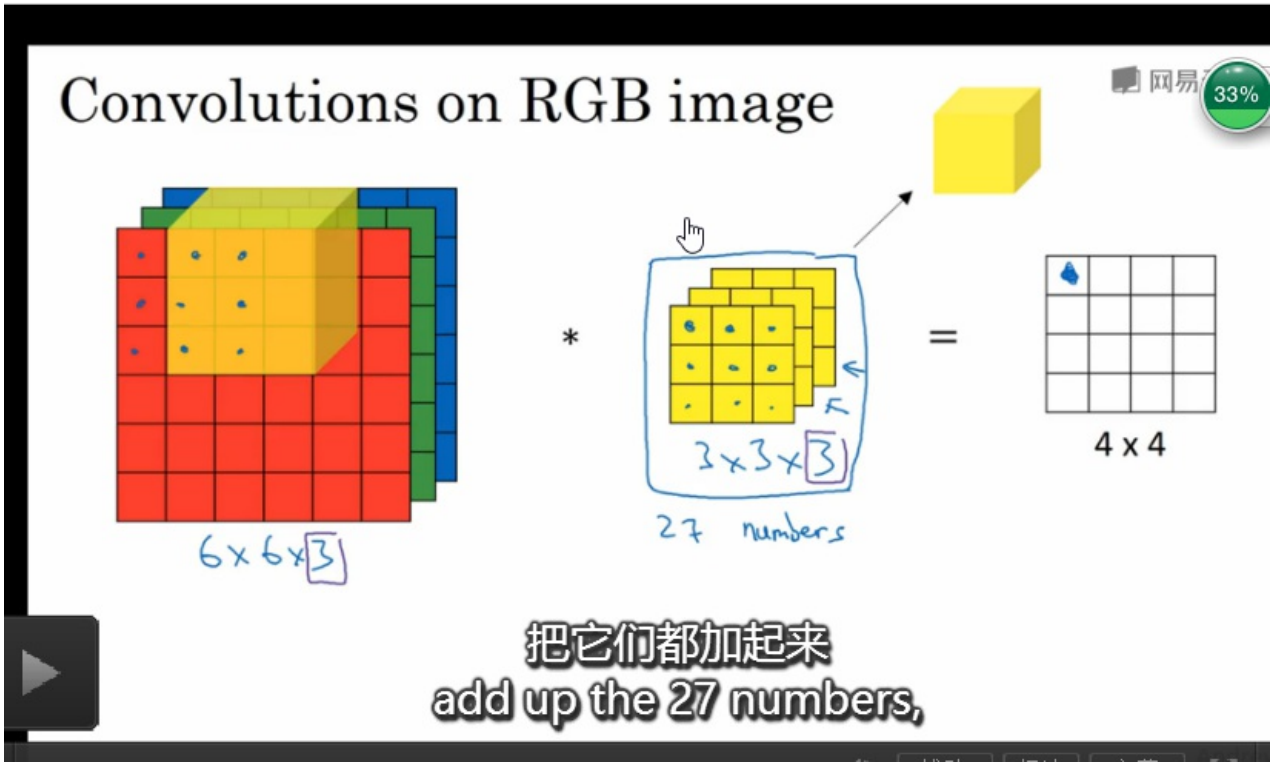


1.6卷积中的卷的体现

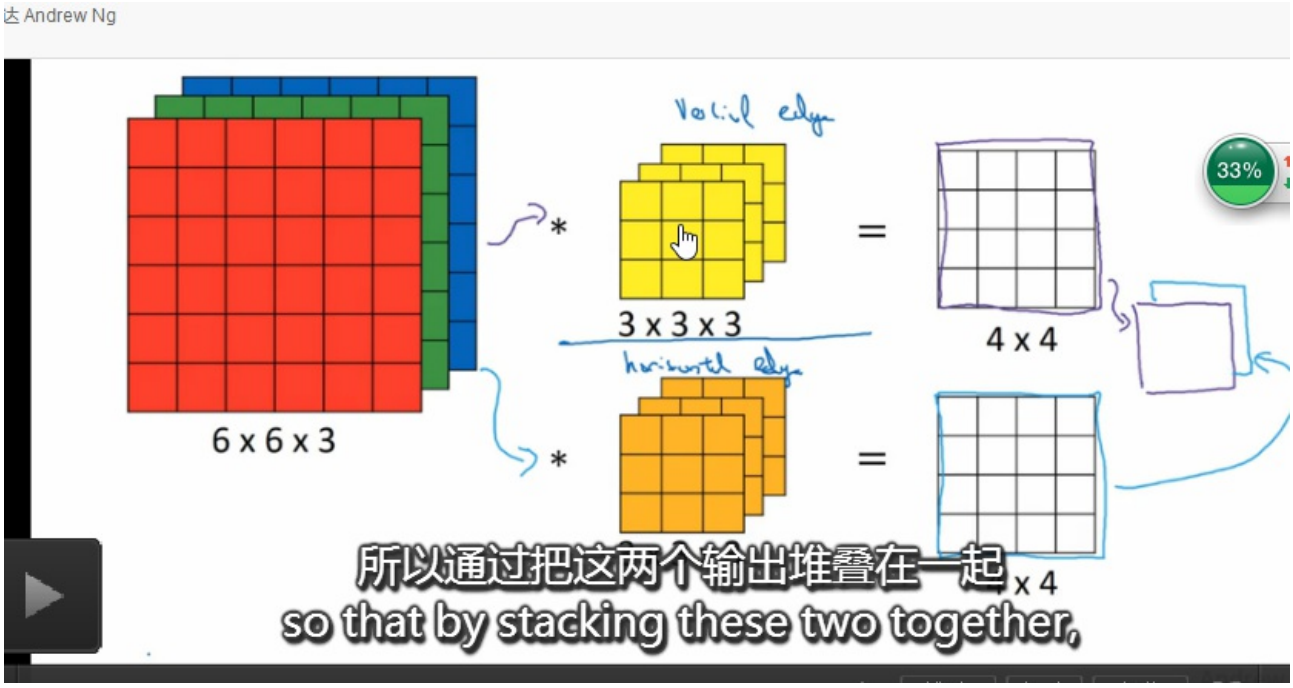
在三维立体上如何做卷积，例子，检测RGB图像的特征，假设为6*6*3



通道数必须相等，

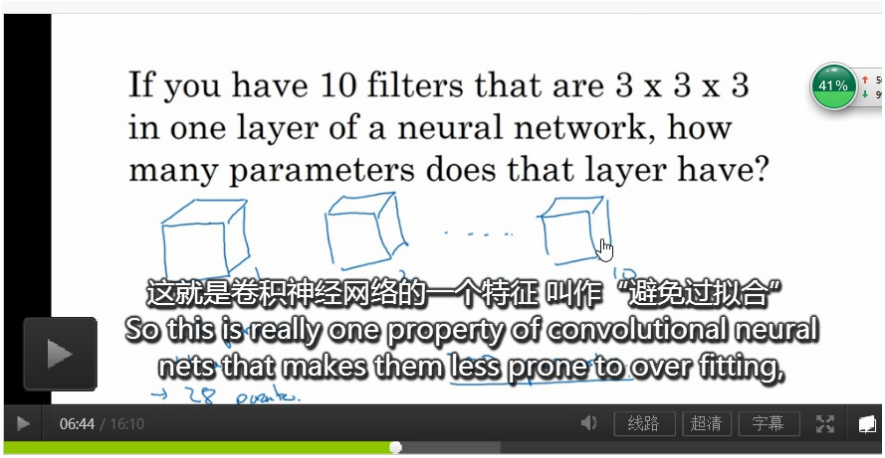


可以有多个过滤器对同一张图片进行特征提取，组成卷积之后的深度。

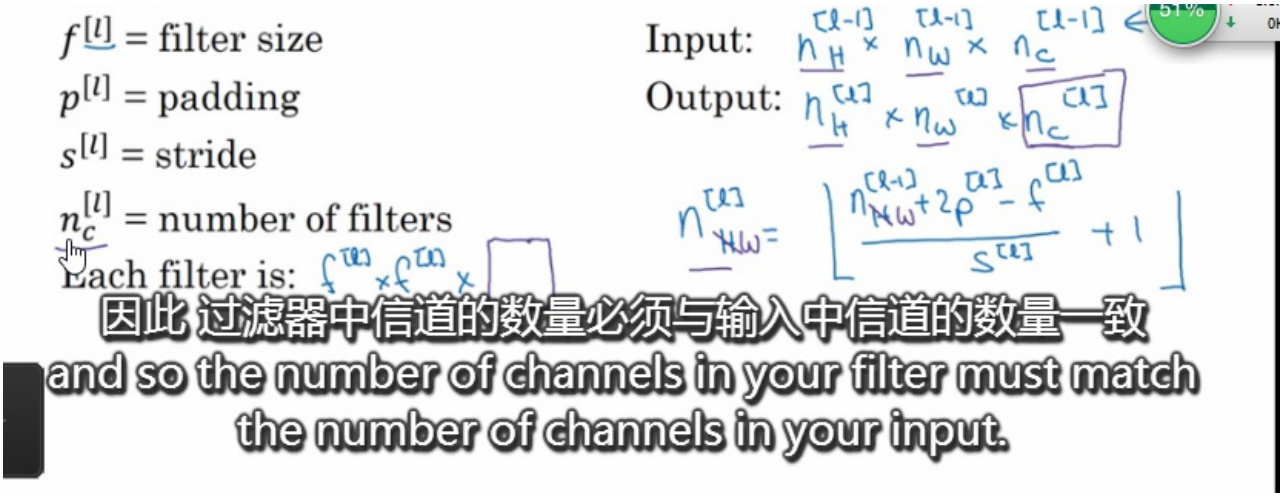


7、单层卷积神经网络

卷积神经网络的参数只跟卷积层的长度宽度和深度有关。跟图片大小无关。卷积输出的深度就是你有多少个卷积核去提取特征，输出的而是特征图的个数

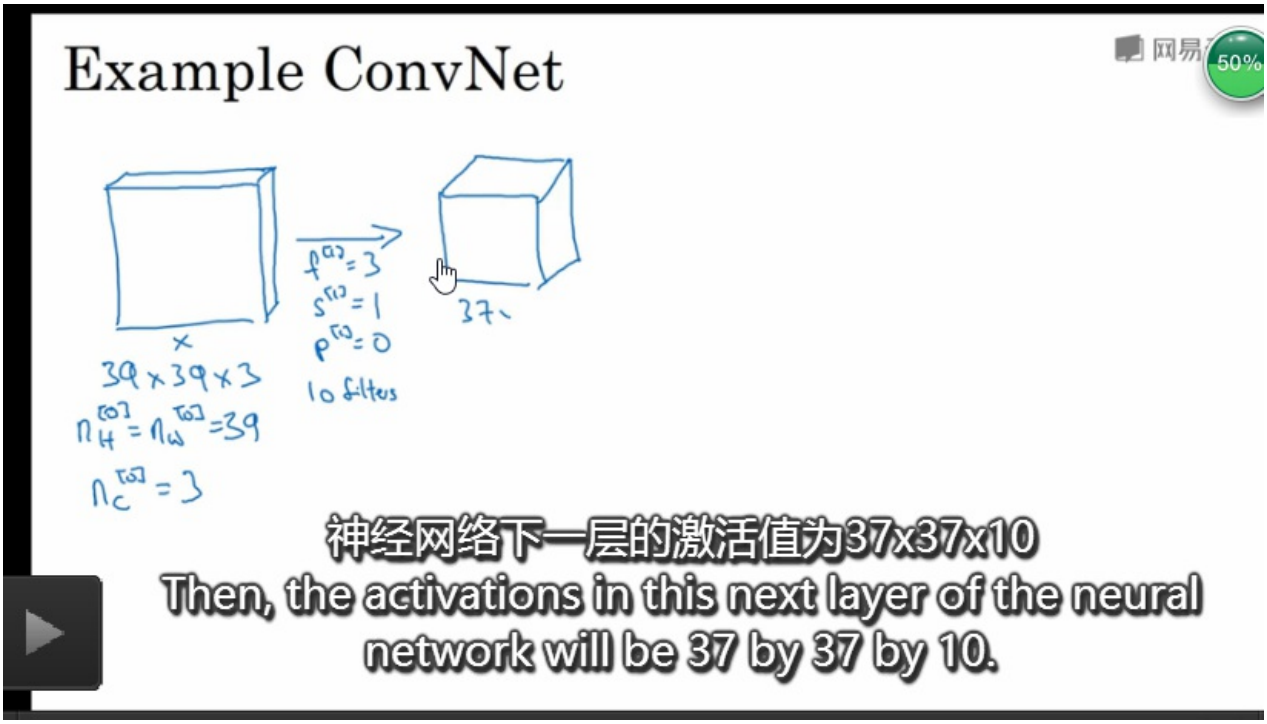


输出图片的深度，也就是上一层的过滤器的个数，如何确定输入卷积核的大小呢？卷积核的通道数要跟图片的通道数一致，



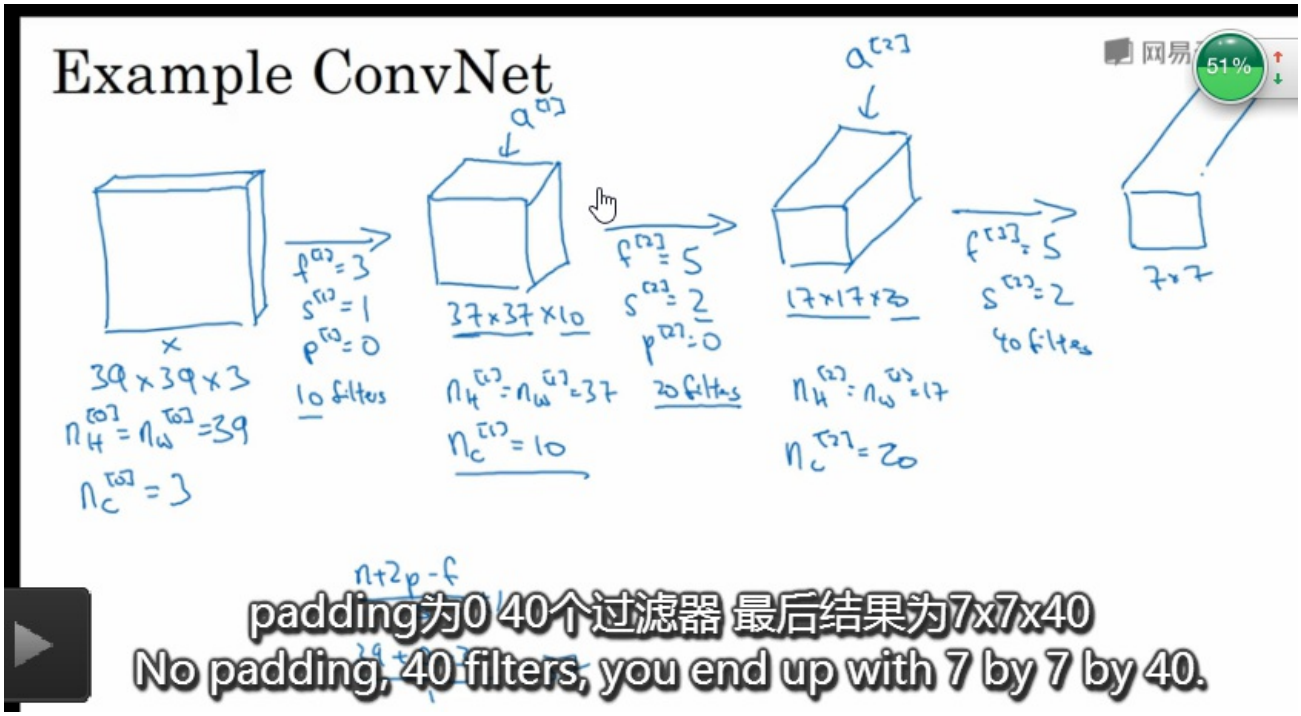
8、简单的示例：

输入是一张39*39*3的图片，用3*3的卷积核提取特征，此卷积核的通道数也是3，假设padding为0，同时有10个过滤器，那么神经网络下一层的激活值为37*37*10：

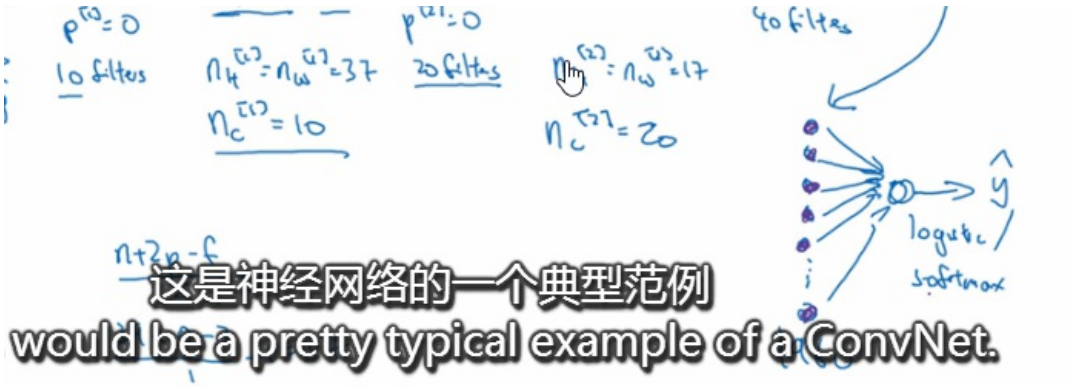


接着采用5*5的过滤器，假设此时步长为2，padding为9，20个filter,那么输出图片的大小为17*17*20,

最后一层再来一个5*5的filter,个数为40个，步长为2，结果为7*7*40大小。



最后再来一个1960的全连接层

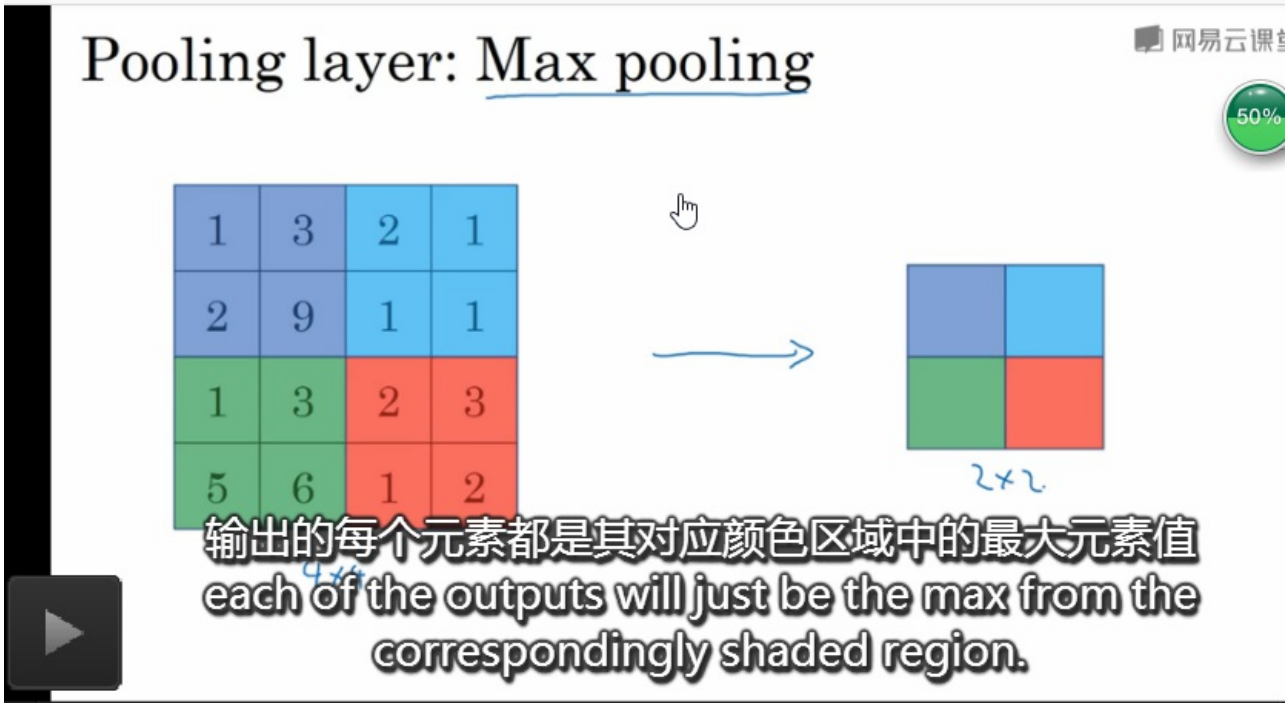


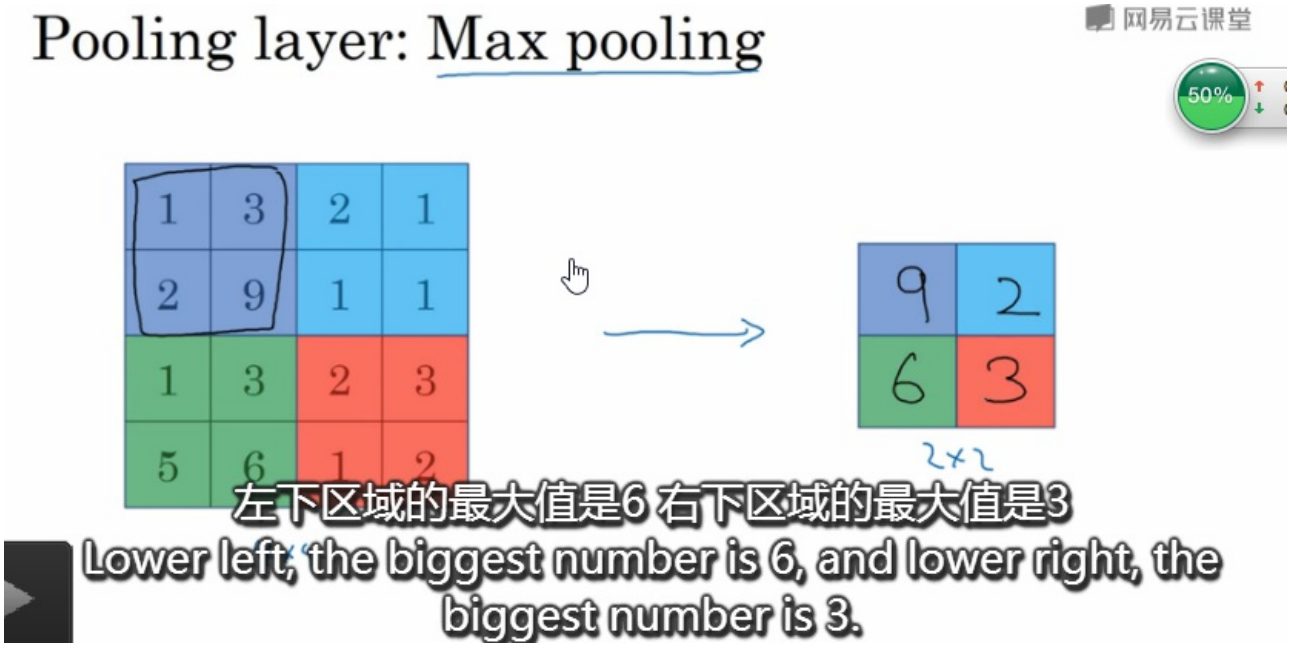
上面就是卷积神经网络的过程，卷积神经网络通常由三层构成，分别是卷积层，池化层，全连接层、

9、池化层

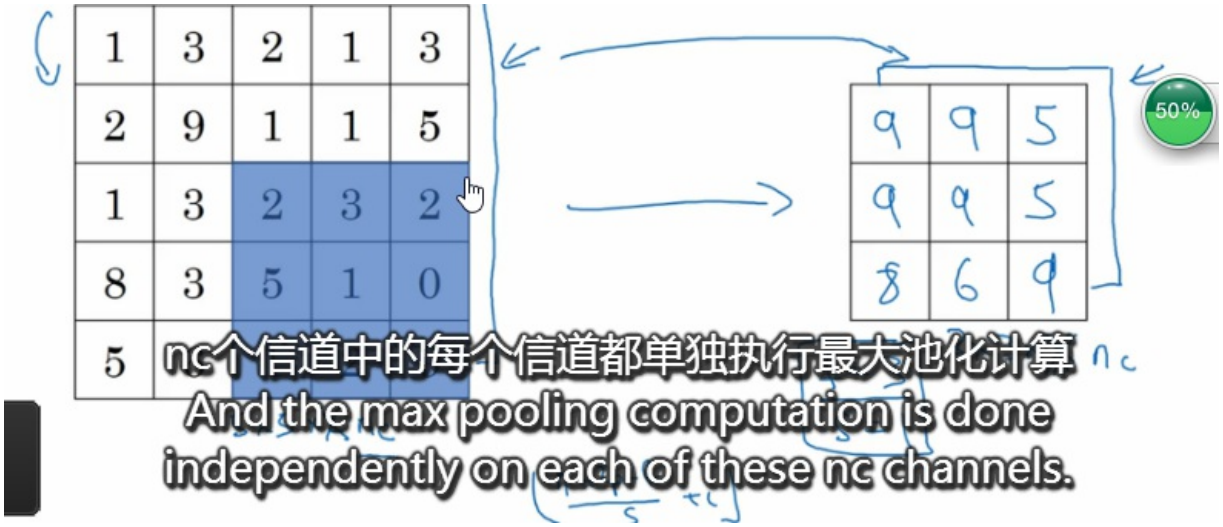
可以减小模型，提高速度，同时提高所提取特征的鲁棒性。

使用2*2的过滤器步长为2，最大池化如下图所示：





max pooling就是说如果提取到了某个特征，那么就保留最大的，每个通道单独计算最大池化

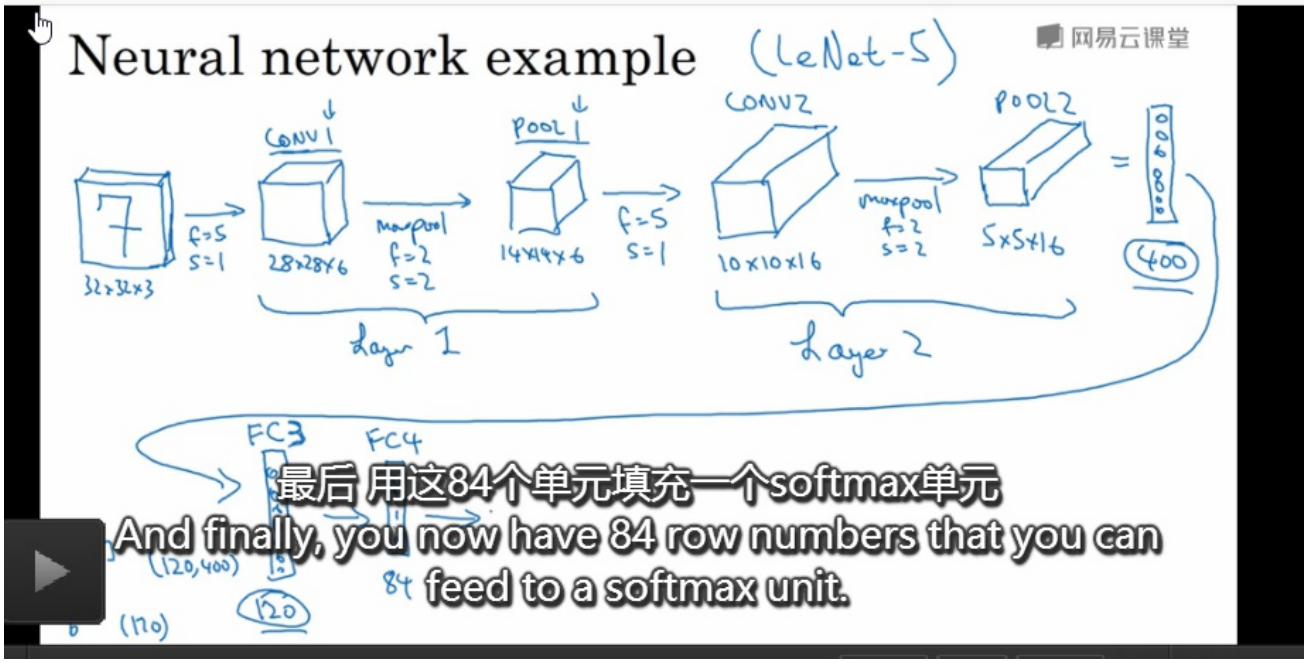


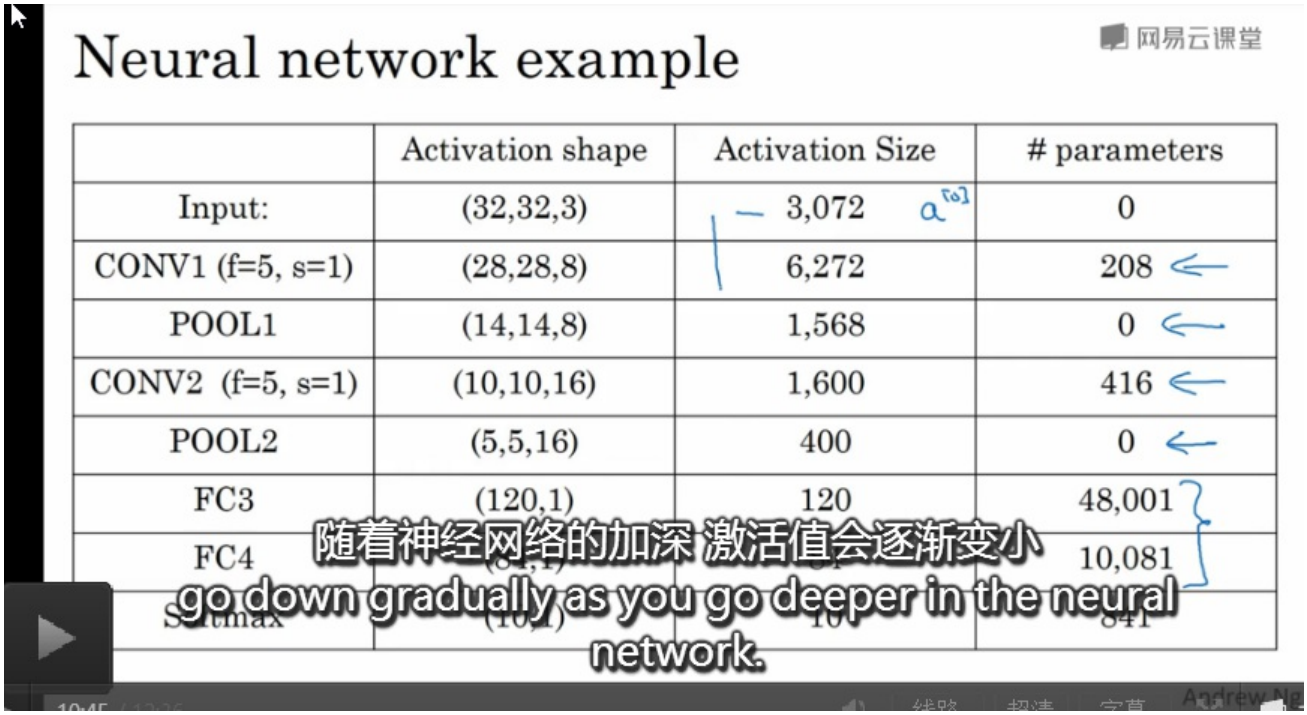
最常用的pooling用的是2*2步长为2的，

效果相当于长宽减半。

还有一种池化方法是平均池化，但是最大池化要比平均池化要常用，

10，简单例子

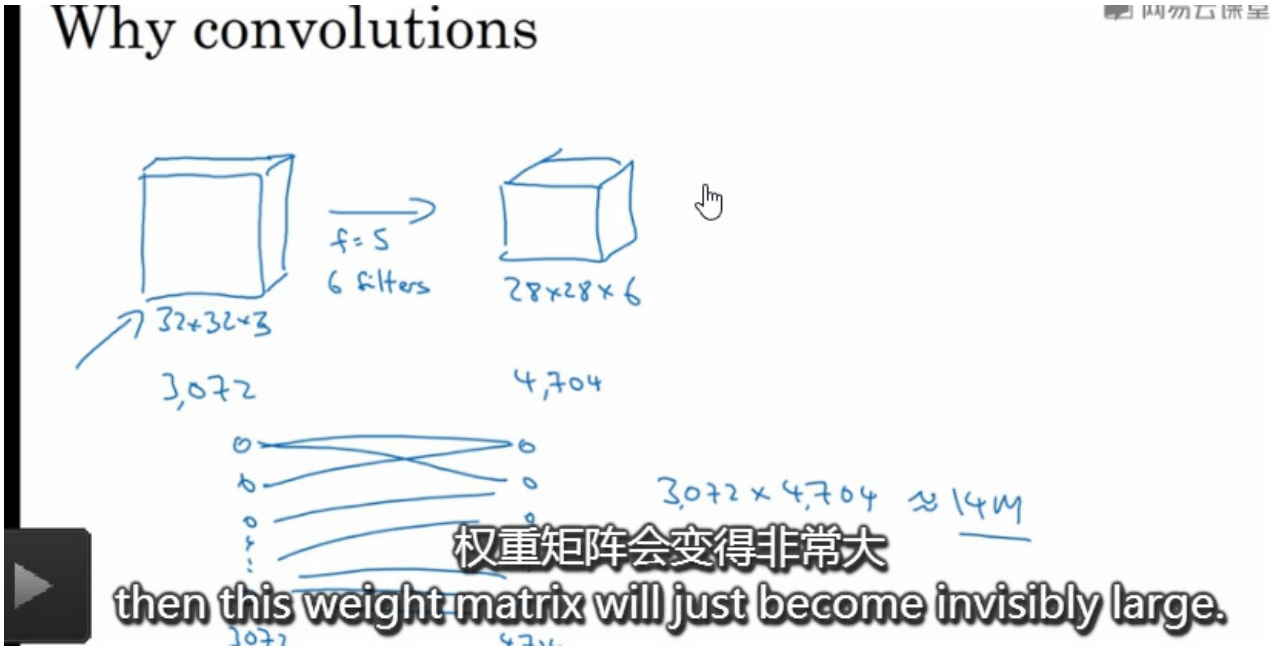




	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[0]}$	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4			10,081
Softmax			841

11、为什么要使用卷积神经网络

和全连接神经网络相比，卷积神经网络的优势在于共享权重和稀疏连接。假设一张32*32*3的图片矩阵。



卷积神经网络的参数，假设过滤器是5*5，那么一个过滤器有25个参数，在加上偏置参数，那么每个过滤器就有26个参数，假设有6个过滤器，参数共计156个，参数只与过滤器有关。卷积神经网络减少参数的另外一个原因是稀疏连接。输出节点至于输入图片矩阵的部分像素矩阵有关，也就是跟卷积核扣上去的那一小块矩阵有关。这就是稀疏连接的概念。卷积神经网络通过权重共享和稀疏连接来减少参数的。从而防止过度拟合，卷积神经网络善于捕捉平移不变，向右平移想个像素，图片中的猫依然清晰可见，因为卷积结构，即使移动几个像素，这张图片依然具有非常相似的特征，