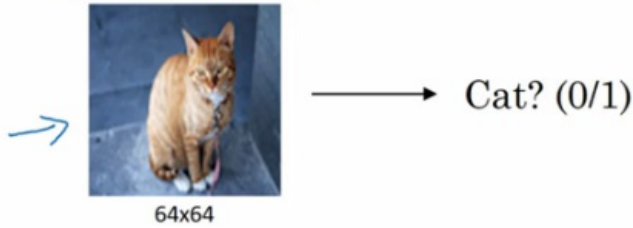


第四篇第一周

第四篇第一周.note1、计算机视觉

例子：图片分类，图片检测

Image Classification



Object detection



叫做目标检测
problem is object detection.

风格迁移:

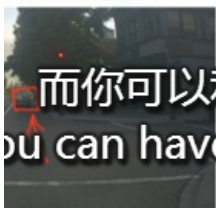
Classification



Neural Style Transfer



Object detection



而你可以利用神经网络将它们融合到一起
you can have a neural network put them together

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 $64 \times 64 \times 3$ 的图片的话，特征向量的维度是12288，但是 $1000 \times 1000 \times 3$ 的图片的话，它足足有1M那么大，特征向量的维度是 $1000 \times 1000 \times 3$ ，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

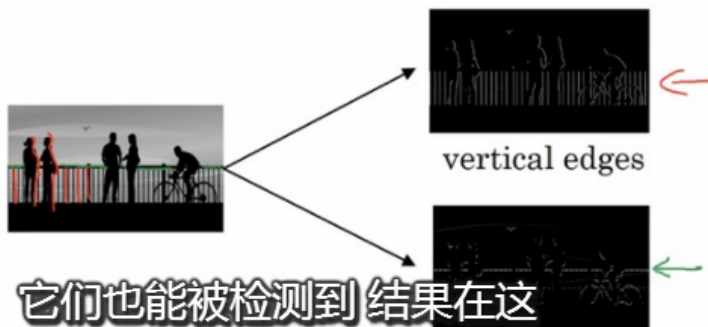
[300万, 1000]，这就意味着隐藏层1就有30亿个参数，这么多参数，难以获得足够的图片来防止神经网络发生过拟合，同时处理30亿个参数的神经网络，对内存的需求也是让人不能接受的。

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。



它们也能被检测到 结果在这
And then also gets detected, and it's roughly here. Andrew Ng

如何在图片中检测边缘呢，再看一个例子：

👆

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6x6

这是一个6x6的灰度图像
Here is a 6x6 grayscale image.

6x6的灰度图，通道为1，检测垂直边缘，我们可以构造出3x3的矩阵，也就是3x3的过滤器，然后使用filter对图片的像素矩阵进行卷积运算，结果为4x4的矩阵：

Vertical edge detection

网易云课

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6x6

"convolution"

*

1	0	-1
1	0	-1
1	0	-1

3x3
filter

=

4x4

一个4x4的图像

which you can think of as a 4x4 image.

Andr

卷积运算过程，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，

Vertical edge detection

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6x6

"convolution"

*

1	0	-1
1	0	-1
1	0	-1

3x3
filter

=

-5			

4x4

在这写上-5

And so I'm going to fill in -5 over here.

Vertical edge detection

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

3	0 ¹	1 ⁰	2 ⁻¹	7	4
1	5 ¹	8 ⁰	9 ⁻¹	3	1
2	7 ¹	2 ⁰	5 ⁻¹	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6x6

"convolution"

*

1	0	-1
1	0	-1
1	0	-1

3x3
filter

=

-5	-4		

4x4

得到 -4 接下来也一样

you end up with -4, and so on.

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

Vertical edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

6x6

*

1	0	-1
1	0	-1
1	0	-1

尽管它也可以被画成黑的
although maybe it could also be drawn as black.

得到的结果:

Vertical edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

6x6

*

1	0	-1
1	0	-1
1	0	-1

3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

就是由这个3x3块
is obtained by taking the element-wise products and

得到最右边的图，中间比较亮，相当于检测到了边缘，



这个6x6图像
ng detected this vertical edge



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

Handwritten:

1	0	-1
2	0	-2
1	0	-1

Sobel

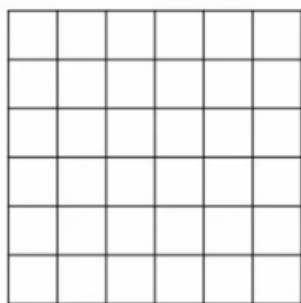
Sobel

还有一种常用的过滤器Scharr过滤器

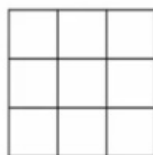
3	0	-3
10	0	-10
3	0	-3

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。

Padding



*



=



3×3
 $f \times f$



6×6
 $n \times n$

你的图像就会变得很小了

before your image starts getting really small

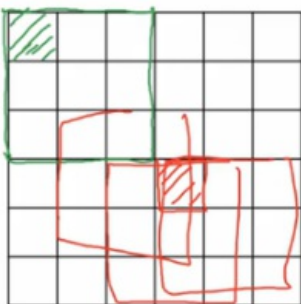
你可不想让你的图片每次做完一个边缘检测时都变小。

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，

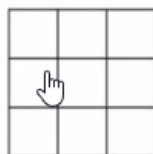
Padding

网易云音乐

31



*



=

3×3
 $f \times f$



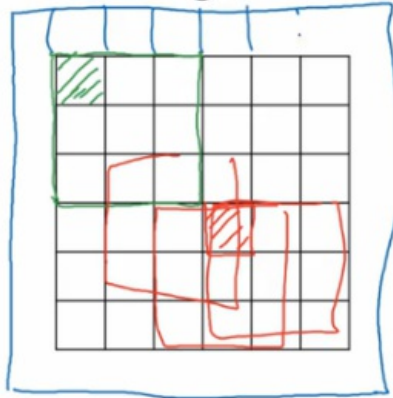
6×6
 $n \times n$

所以那些在角落或者边缘区域的像素点

and so it's as if pixels on the corners or on the edges

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

Padding



- throw away into from edge

$$\begin{array}{ccc}
 & * & = \\
 \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} & & \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array} \\
 3 \times 3 & & f \times f
 \end{array}$$

再填充一层像素

with an additional border of one pixel all around edges.

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。

Valid and Same convolutions

网易云课堂

31%

→ no padding

$$\begin{array}{lclcl}
 \text{"Valid":} & n \times n & * & f \times f & \rightarrow n - f + 1 \times n - f + 1 \\
 & 6 \times 6 & * & 3 \times 3 & \rightarrow 4 \times 4
 \end{array}$$



"Same": Pad so that output size is the same as the input size.

你的输出大小和输入大小是一样的
so the output size is the same as the input size.

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 $n \times n$ ，卷积核为 $f \times f$ ，padding为 p ，步长 $s=2$ ，输出为 $(n+2p-f)/2$

Summary of convolutions

$n \times n$ image $f \times f$ filter

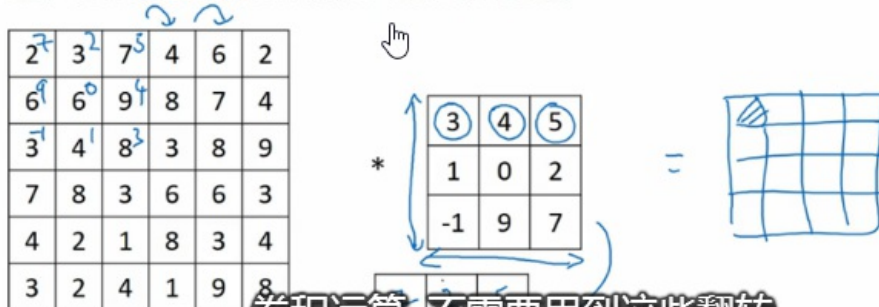
padding p stride s

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

可以选择所有的数使结果是整数
And it is nice we can choose all of these numbers

Technical note on cross-correlation vs. convolution

Convolution in math textbook:



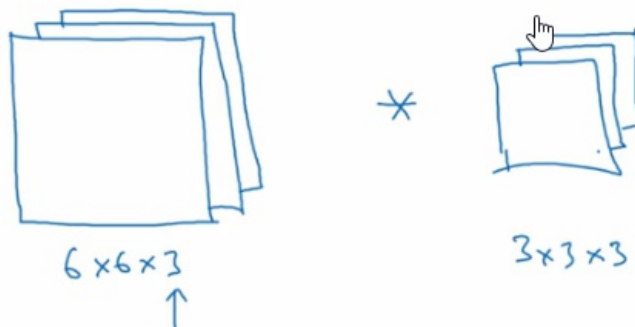
卷积运算 不需要用到这些翻转

the convolution operator without bothering to use these flips.

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 $6 \times 6 \times 3$

Convolutions on RGB images



这样这个过滤器也有三层

So, the filter itself will also have three layers

通道数必须相等，

Convolutions on RGB image

The diagram illustrates the convolution process on an RGB image. On the left, a 6x6x3 image (red, green, and blue channels) is shown. In the center, a 3x3x3 kernel is highlighted, with handwritten notes indicating it contains "27 numbers". An arrow points from the kernel to a single yellow cube, representing one of the 27 numbers. To the right, a 4x4 grid represents the resulting feature map. The text "把它们都加起来" (Add them all up) and "add up the 27 numbers," is written below the kernel.

6x6x3

3x3x3

27 numbers

4x4

把它们都加起来
add up the 27 numbers,

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。

达 Andrew Ng

The diagram shows two parallel convolution operations. On the left, a 6x6x3 image is shown. Two different 3x3x3 kernels are applied to it, each resulting in a 4x4 feature map. The top kernel is labeled "vertical edge" and the bottom kernel is labeled "horizontal edge". The resulting feature maps are shown as 4x4 grids. The text "所以通过把这两个输出堆叠在一起" (so that by stacking these two together,) and "so that by stacking these two together," is written below the feature maps.

6x6x3

3x3x3

vertical edge

horizontal edge

4x4

4x4

所以通过把这两个输出堆叠在一起
so that by stacking these two together,

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个卷积核去提取特征，输出的而是特征图的个数

If you have 10 filters that are $3 \times 3 \times 3$ in one layer of a neural network, how many parameters does that layer have?



这就是卷积神经网络的一个特征叫作“避免过拟合”

So this is really one property of convolutional neural nets that makes them less prone to over fitting,

→ 28 params.

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，

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$n_c^{[l]}$ = number of filters

Each filter is:



Input:

$$n_h^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$$

Output:

$$n_h^{[l]} \times n_w^{[l]} \times n_c^{[l]}$$

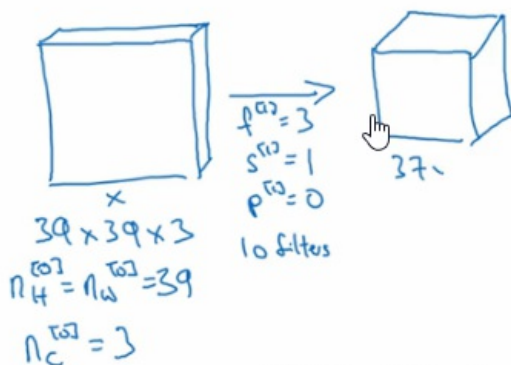
$$n_{hw}^{[l]} = \left\lfloor \frac{n_h^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

因此过滤器中信道的数量必须与输入中信道的数量一致
and so the number of channels in your filter must match
the number of channels in your input.

8、简单的示例：

输入是一张 $39 \times 39 \times 3$ 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 $37 \times 37 \times 10$ ：

Example ConvNet

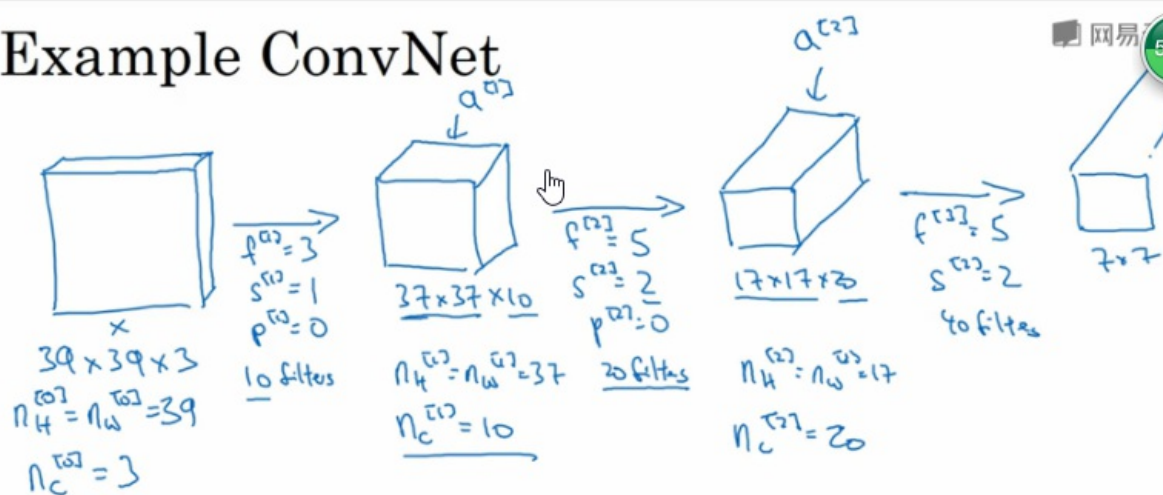


神经网络下一层的激活值为 $37 \times 37 \times 10$

Then, the activations in this next layer of the neural network will be 37 by 37 by 10.

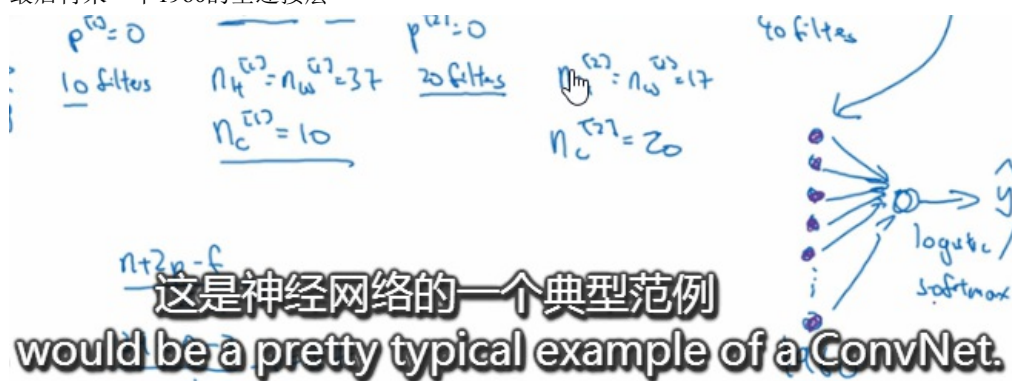
接着采用 5×5 的过滤器，假设此时步长为2，padding为9，20个filter，那么输出图片的大小为 $17 \times 17 \times 20$ ，最后一层再来一个 5×5 的filter，个数为40个，步长为2，结果为 $7 \times 7 \times 40$ 大小。

Example ConvNet



padding为0 40个过滤器 最后结果为7x7x40
No padding, 40 filters, you end up with 7 by 7 by 40.

最后再来一个1960的全连接层



这是神经网络的一个典型范例
would be a pretty typical example of a ConvNet.

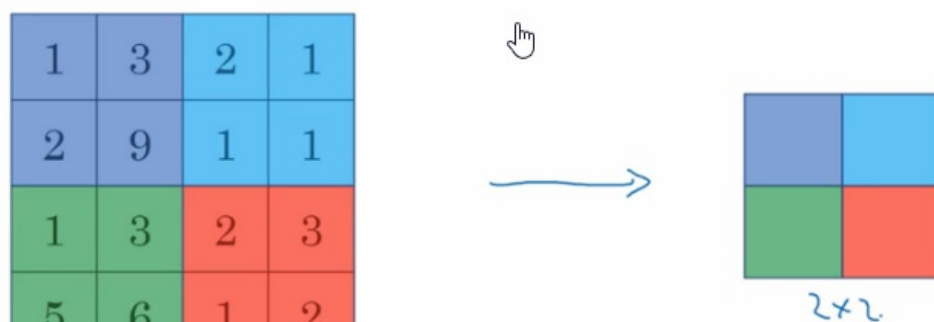
上面就是卷积神经网络的过程，卷积神经网络通常由三层构成，分别是卷积层，池化层，全连接层、

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。

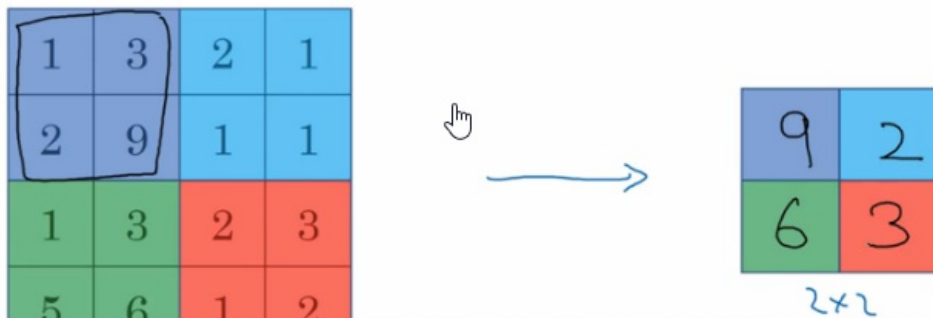
使用2*2的过滤器步长为2，最大池化如下图所示：

Pooling layer: Max pooling



输出的每个元素都是其对应颜色区域中的最大元素值
each of the outputs will just be the max from the correspondingly shaded region.

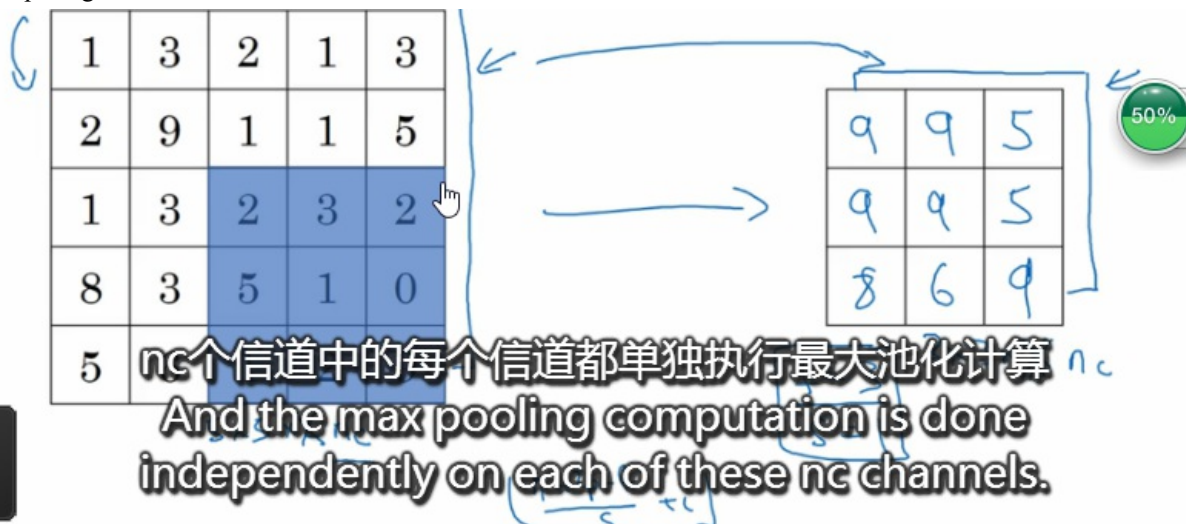
Pooling layer: Max pooling



左下区域的最大值是6 右下区域的最大值是3

Lower left, the biggest number is 6, and lower right, the biggest number is 3.

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



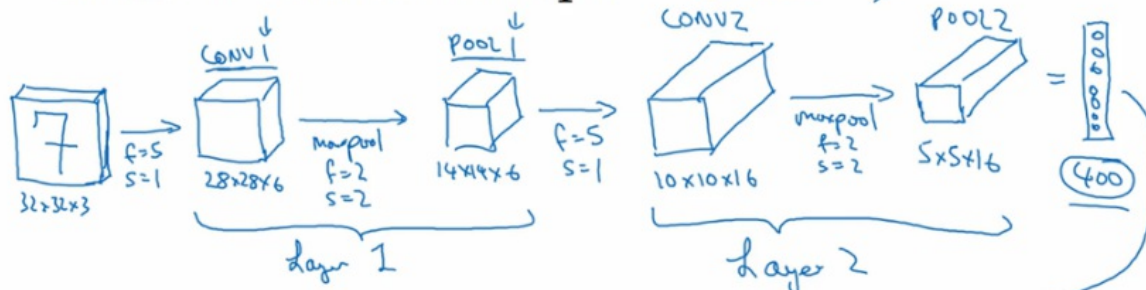
最常用的pooling用的是2*2步长为2的，

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

Neural network example (LeNet-5)



最后用这84个单元填充一个softmax单元

And finally, you now have 84 row numbers that you can feed to a softmax unit.

Neural network example

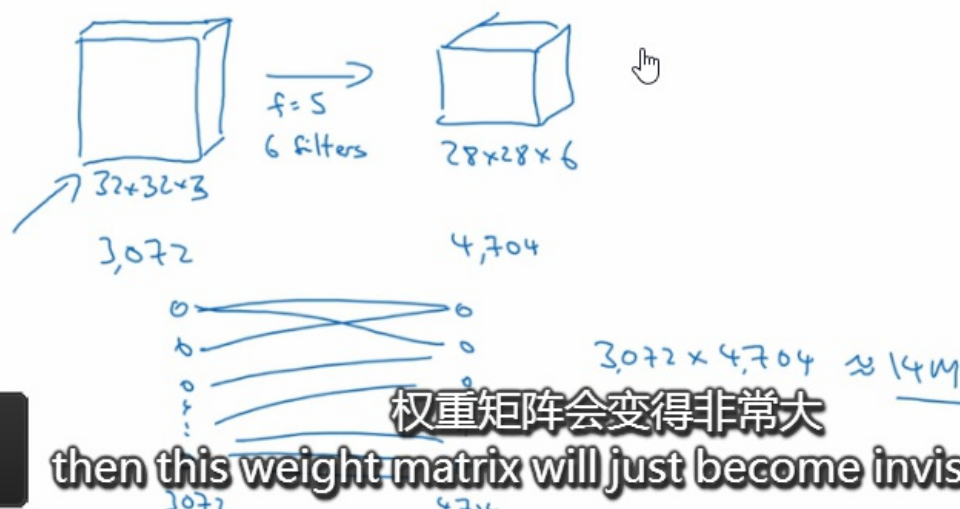
	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{(0)}$	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208 ←
POOL1	(14,14,8)	1,568	0 ←
CONV2 (f=5, s=1)	(10,10,16)	1,600	416 ←
POOL2	(5,5,16)	400	0 ←
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

随着神经网络的加深 激活值会逐渐变小
go down gradually as you go deeper in the neural network.

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。

Why convolutions



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，

