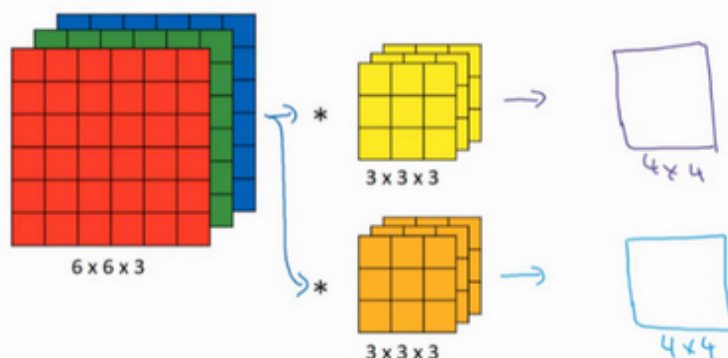


## 1.7 单层卷积网络（One layer of a convolutional network）

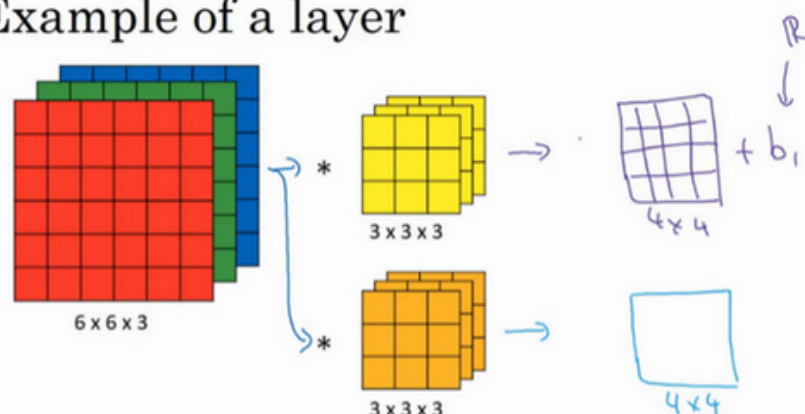
今天我们要讲的是如何构建卷积神经网络的卷积层，下面来看个例子。

### Example of a layer



上节课，我们已经讲了如何通过两个过滤器卷积处理一个三维图像，并输出两个不同的  $4 \times 4$  矩阵。假设使用第一个过滤器进行卷积，得到第一个  $4 \times 4$  矩阵。使用第二个过滤器进行卷积得到另外一个  $4 \times 4$  矩阵。

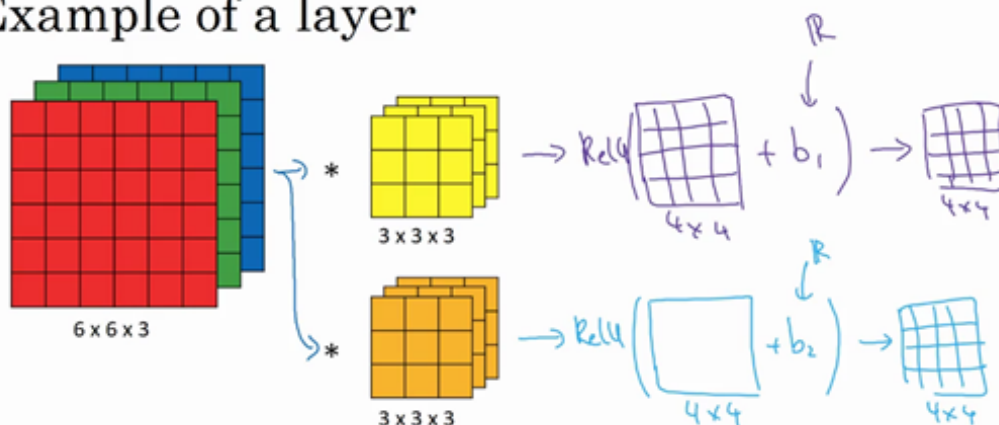
### Example of a layer



最终各自形成一个卷积神经网络层，然后增加偏差，它是一个实数，通过 **Python** 的广播机制给这 16 个元素都加上同一偏差。然后应用非线性函数，为了说明，它是一个非线性激活函数 **ReLU**，输出结果是一个  $4 \times 4$  矩阵。

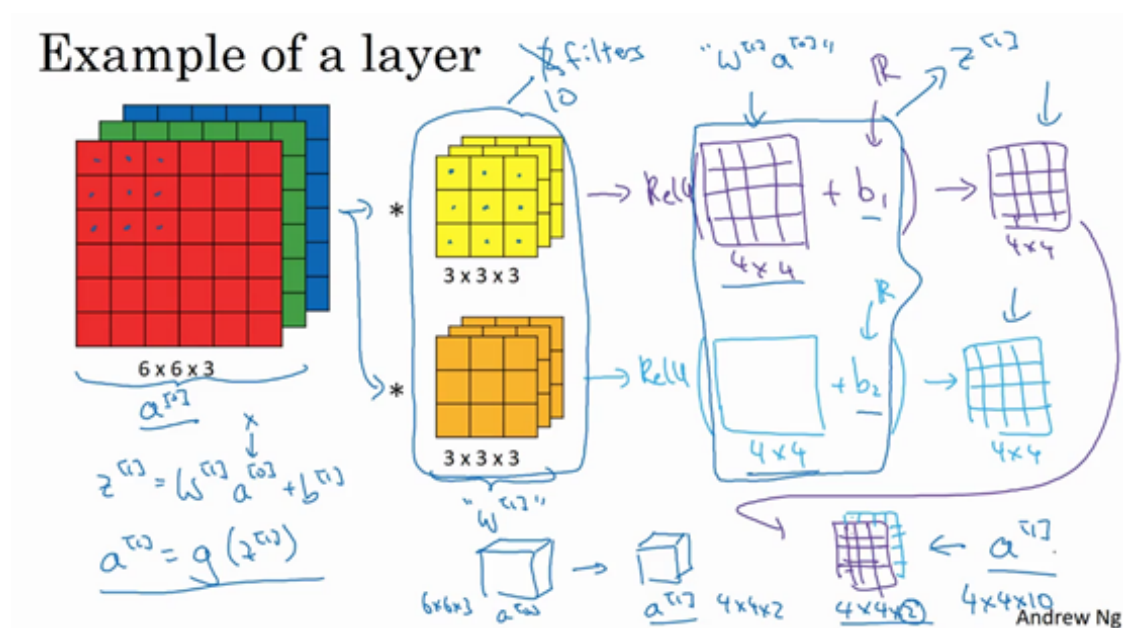
对于第二个  $4 \times 4$  矩阵，我们加上不同的偏差，它也是一个实数，16 个数字都加上同一个实数，然后应用非线性函数，也就是一个非线性激活函数 **ReLU**，最终得到另一个  $4 \times 4$  矩阵。然后重复我们之前的步骤，把这两个矩阵堆叠起来，最终得到一个  $4 \times 4 \times 2$  的矩阵。我们通过计算，从  $6 \times 6 \times 3$  的输入推导出一个  $4 \times 4 \times 2$  矩阵，它是卷积神经网络的一层，把它映射到标准神经网络中四个卷积层中的某一层或者一个非卷积神经网络中。

## Example of a layer



注意前向传播中一个操作就是  $z^{[1]} = W^{[1]}a^{[0]} + b^{[1]}$ ，其中  $a^{[0]} = x$ ，执行非线性函数得到  $a^{[1]}$ ，即  $a^{[1]} = g(z^{[1]})$ 。这里的输入是  $a^{[0]}$ ，也就是  $x$ ，[这些过滤器用变量  \$W^{\[1\]}\$  表示](#)。在卷积过程中，我们对这 27 个数进行操作，其实是  $27 \times 2$ ，因为我们用了两个过滤器，我们取这些数做乘法。实际执行了一个线性函数，得到一个  $4 \times 4$  的矩阵。卷积操作的输出结果是一个  $4 \times 4$  的矩阵，它的作用类似于  $W^{[1]}a^{[0]}$ ，也就是这两个  $4 \times 4$  矩阵的输出结果，然后加上偏差。

## Example of a layer



这一部分（图中蓝色边框标记的部分）就是应用激活函数 **ReLU** 之前的值，它的作用类似于  $z^{[1]}$ ，最后应用非线性函数，得到的这个  $4 \times 4 \times 2$  矩阵，成为神经网络的下一层，也就是激活层。

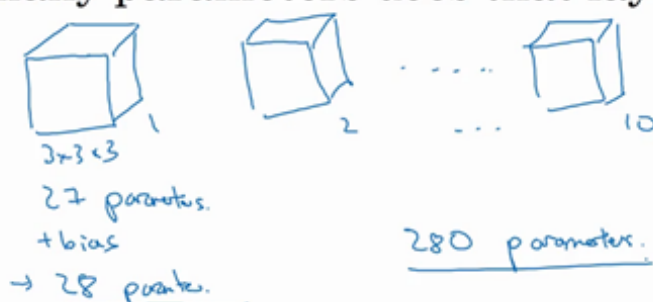
这就是  $a^{[0]}$  到  $a^{[1]}$  的演变过程，首先执行线性函数，然后所有元素相乘做卷积，具体做法是运用线性函数再加上偏差，然后应用激活函数 **ReLU**。这样就通过神经网络的一层把一个  $6 \times 6 \times 3$  的维度  $a^{[0]}$  演化为一个  $4 \times 4 \times 2$  维度的  $a^{[1]}$ ，这就是卷积神经网络的一层。

示例中我们有两个过滤器，也就是有两个特征，因此我们才最终得到一个  $4 \times 4 \times 2$  的输

出。但如果我们用了 10 个过滤器，而不是 2 个，我们最后会得到一个  $4 \times 4 \times 10$  维度的输出图像，因为我们选取了其中 10 个特征映射，而不仅仅是 2 个，将它们堆叠在一起，形成一个  $4 \times 4 \times 10$  的输出图像，也就是  $a^{[1]}$ 。

## Number of parameters in one layer

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?



为了加深理解，我们来做一个练习。假设你有 10 个过滤器，而不是 2 个，神经网络的一层是  $3 \times 3 \times 3$ ，那么，这一层有多少个参数呢？我们来计算一下，每一层都是一个  $3 \times 3 \times 3$  的矩阵，因此每个过滤器有 27 个参数，也就是 27 个数。然后加上一个偏差，用参数  $b$  表示，现在参数增加到 28 个。上一页幻灯片里我画了 2 个过滤器，而现在我们有 10 个，加在一起是  $28 \times 10$ ，也就是 280 个参数。

请注意一点，不论输入图片有多大， $1000 \times 1000$  也好， $5000 \times 5000$  也好，参数始终都是 280 个。用这 10 个过滤器来提取特征，如垂直边缘，水平边缘和其它特征。即使这些图片很大，参数却很少，这就是卷积神经网络的一个特征，叫作“避免过拟合”。你已经知道如何提取 10 个特征，可以应用到大图片中，而参数数量固定不变，此例中只有 28 个，相对较少。

最后我们总结一下用于描述卷积神经网络中的一层（以  $l$  层为例），也就是卷积层的各种标记。

## Summary of notation

If layer  $l$  is a convolution layer:

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

$p^{[l]}$  = padding

$s^{[l]}$  = stride

Input:  $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]} \leftarrow$

Output:  $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

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

这一层是卷积层，用  $f^{[l]}$  表示过滤器大小，我们说过过滤器大小为  $f \times f$ ，上标  $[l]$  表示  $l$  层中过滤器大小为  $f \times f$ 。通常情况下，上标  $[l]$  用来标记  $l$  层。用  $p^{[l]}$  来标记 **padding** 的数量，**padding** 数量也可指定为一个 **valid** 卷积，即无 **padding**。或是 **same** 卷积，即选定 **padding**，如此一来，输出和输入图片的高度和宽度就相同了。用  $s^{[l]}$  标记步幅。

这一层的输入会是某个维度的数据，表示为  $n \times n \times n_c$ ， $n_c$  某层上的颜色通道数。

我们要稍作修改，增加上标  $[l-1]$ ，即  $n^{[l-1]} \times n^{[l-1]} \times n_c^{[l-1]}$ ，因为它是上一层的激活值。

此例中，所用图片的高度和宽度都一样，但它们也有可能不同，所以分别用上下标  $H$  和  $W$  来标记，即  $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$ 。那么在第  $l$  层，图片大小为  $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$ ， $l$  层的输入就是上一层的输出，因此上标要用  $[l-1]$ 。神经网络这一层中会有输出，它本身会输出图像。其大小为  $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$ ，这就是输出图像的大小。

前面我们提到过，这个公式给出了输出图片的大小，至少给出了高度和宽度， $\lfloor \frac{n+2p-f}{s} + 1 \rfloor$ （注意： $\lfloor \frac{n+2p-f}{s} + 1 \rfloor$  直接用这个运算结果，也可以向下取整）。在这个新表达式中， $l$  层输出图像的高度，即  $n_H^{[l]} = \lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \rfloor$ ，同样我们可以计算出图像的宽度，用  $W$  替换参数  $H$ ，即  $n_W^{[l]} = \lfloor \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \rfloor$ ，公式一样，只要变化高度和宽度的参数我们便能计算输出图像的高度或宽度。这就是由  $n_H^{[l-1]}$  推导  $n_H^{[l]}$  以及  $n_W^{[l-1]}$  推导  $n_W^{[l]}$  的过程。

那么通道数量又是什么？这些数字从哪儿来的？我们来看一下。输出图像也具有深度，通过上一个示例，我们知道它等于该层中过滤器的数量，如果有 2 个过滤器，输出图像就是  $4 \times 4 \times 2$ ，它是二维的，如果有 10 个过滤器，输出图像就是  $4 \times 4 \times 10$ 。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的数量。如何确定过滤器的大小呢？我们知道卷积一个  $6 \times 6 \times 3$  的图片需要一个  $3 \times 3 \times 3$  的过滤器，因此过滤器中通道的数量必须与输入中通道

的数量一致。因此，输出通道数量就是输入通道数量，所以过滤器维度等于  $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$ 。

If layer l is a convolution layer:

$f^{[l]}$  = filter size  
 $p^{[l]}$  = padding  
 $s^{[l]}$  = stride  
 $n_c^{[l]}$  = number of filters  
 → Each filter is:  $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$   
 Activations:  $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$   
 Weights:  $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$   
 bias:  $n_c^{[l]} - (1, 1, 1, n_c^{[l]})$  ← #f: (1, 1, 1,  $n_c^{[l]}$ ) in layer l.

Input:  $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$   
 Output:  $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$   

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

$$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$$

应用偏差和非线性函数之后，这一层的输出等于它的激活值  $a^{[l]}$ ，也就是这个维度（输出维度）。 $a^{[l]}$  是一个三维体，即  $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$ 。当你执行批量梯度下降或小批量梯度下降时，如果有  $m$  个例子，就是有  $m$  个激活值的集合，那么输出  $A^{[l]} = m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$ 。如果采用批量梯度下降，变量的排列顺序如下，首先是索引和训练示例，然后是其它三个变量。

该如何确定权重参数，即参数  $w$  呢？过滤器的维度已知，为  $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$ ，这只是个过滤器的维度，有多少个过滤器，这 ( $n_c^{[l]}$ ) 是过滤器的数量，权重也就是所有过滤器的集合再乘以过滤器的总数量，即  $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$ ，损失数量  $L$  就是  $l$  层中过滤器的个数。

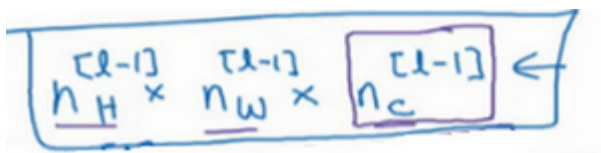
最后我们看看偏差参数，每个过滤器都有一个偏差参数，它是一个实数。偏差包含了这些变量，它是该维度上的一个向量。后续课程中我们会看到，为了方便，偏差在代码中表示为一个  $1 \times 1 \times 1 \times n_c^{[l]}$  的四维向量或四维张量。

$$w \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$$

$$b \rightarrow 1 \times 1 \times 1 \times n_c^{[l]}$$

卷积有很多种标记方法，这是我们最常用的卷积符号。大家在线搜索或查看开源代码时，关于高度，宽度和通道的顺序并没有完全统一的标准卷积，所以在查看 **GitHub** 上的源代码或阅读一些开源实现的时候，你会发现有些作者会采用把通道放在首位的编码标准，有时所有变量都采用这种标准。实际上在某些架构中，当检索这些图片时，会有一个变量或参数来

标识计算通道数量和通道损失数量的先后顺序。只要保持一致，这两种卷积标准都可用。很遗憾，这只是一部分标记法，因为深度学习文献并未对标记达成一致，但课上我会采用这种卷积标识法，按高度，宽度和通道损失数量的顺序依次计算。



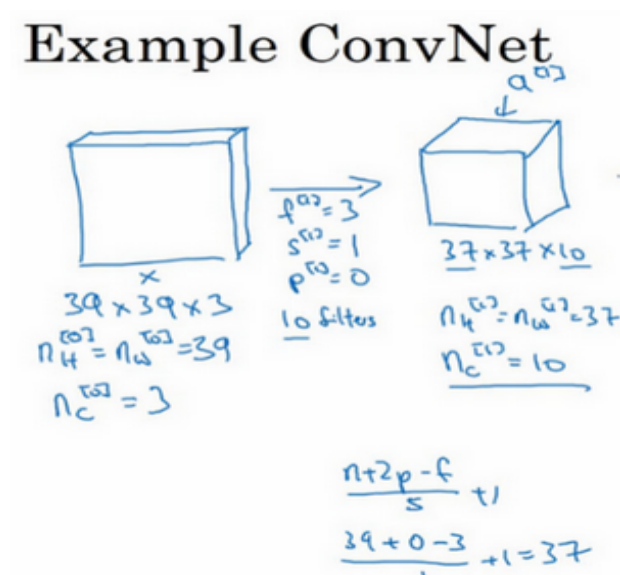
我知道，忽然间接触到这么多新的标记方法，你可能会说，这么多怎么记呢？别担心，不用全都记住，你可以通过本周的练习来熟悉它们。而这节课我想讲的重点是，卷积神经网络的某一卷积层的工作原理，以及如何计算某一卷积层的激活函数，并映射到下一层的激活值。了解了卷积神经网络中某一卷积层的工作原理，我们就可以把它们堆叠起来形成一个深度卷积神经网络，我们下节课再讲。



## 1.8 简单卷积网络示例 (A simple convolution network example)

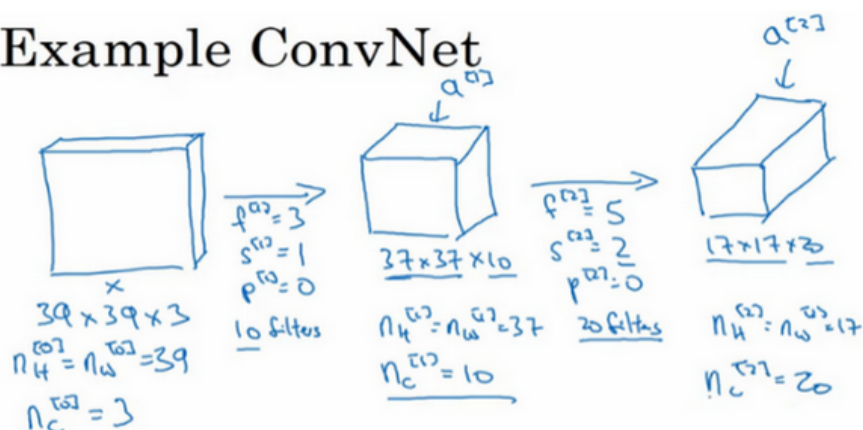
上节课，我们讲了如何为卷积网络构建一个卷积层。今天我们看一个深度卷积神经网络的具体示例，顺便练习一下我们上节课所学的标记法。

假设你有一张图片，你想做图片分类或图片识别，把这张图片输入定义为 $x$ ，然后辨别图片中有没有猫，用 0 或 1 表示，这是一个分类问题，我们来构建适用于这项任务的卷积神经网络。针对这个示例，我用了一张比较小的图片，大小是  $39 \times 39 \times 3$ ，这样设定可以使其中一些数字效果更好。所以  $n_H^{[0]} = n_W^{[0]}$ ，即高度和宽度都等于 39， $n_C^{[0]} = 3$ ，即 0 层的通道数为 3。



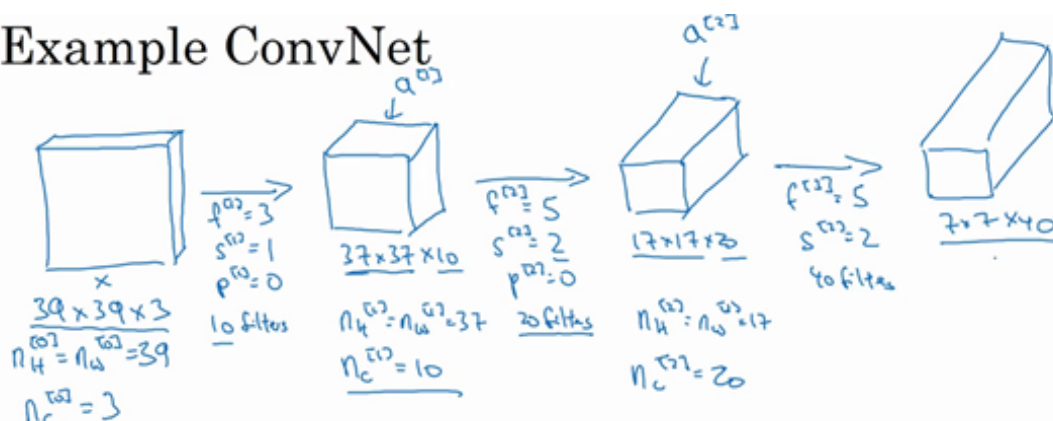
假设第一层我们用一个  $3 \times 3$  的过滤器来提取特征，那么  $f^{[1]} = 3$ ，因为过滤器是  $3 \times 3$  的矩阵。 $s^{[1]} = 1$ ， $p^{[1]} = 0$ ，所以高度和宽度使用 **valid** 卷积。如果有 10 个过滤器，神经网络下一层的激活值为  $37 \times 37 \times 10$ ，写 10 是因为我们用了 10 个过滤器，37 是公式  $\frac{n+2p-f}{s} + 1$  的计算结果，也就是  $\frac{39+0-3}{1} + 1 = 37$ ，所以输出是  $37 \times 37$ ，它是一个 **valid** 卷积，这是输出结果的大小。第一层标记为  $n_H^{[1]} = n_W^{[1]} = 37$ ， $n_C^{[1]} = 10$ ， $n_C^{[1]}$  等于第一层中过滤器的个数，这 ( $37 \times 37 \times 10$ ) 是第一层激活值的维度。

## Example ConvNet

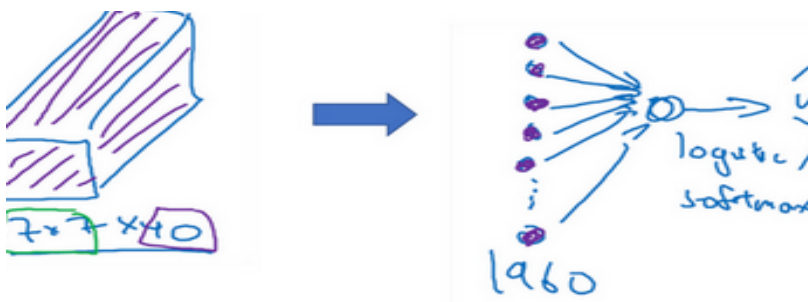


假设还有另外一个卷积层，这次我们采用的过滤器是  $5 \times 5$  的矩阵。在标记法中，神经网络下一层的  $f = 5$ ，即  $f^{[2]} = 5$  步幅为 2，即  $s^{[2]} = 2$ 。padding 为 0，即  $p^{[2]} = 0$ ，且有 20 个过滤器。所以其输出结果会是一张新图像，这次的输出结果为  $17 \times 17 \times 20$ ，因为步幅是 2，维度缩小得很快，大小从  $37 \times 37$  减小到  $17 \times 17$ ，减小了一半还多，过滤器是 20 个，所以通道数也是 20， $17 \times 17 \times 20$  即激活值  $a^{[2]}$  的维度。因此  $n_H^{[2]} = n_W^{[2]} = 17$ ， $n_C^{[2]} = 20$ 。

## Example ConvNet



我们来构建最后一个卷积层，假设过滤器还是  $5 \times 5$ ，步幅为 2，即  $f^{[2]} = 5$ ， $s^{[3]} = 2$ ，计算过程我跳过了，最后输出为  $7 \times 7 \times 40$ ，假设使用了 40 个过滤器。padding 为 0，40 个过滤器，最后结果为  $7 \times 7 \times 40$ 。



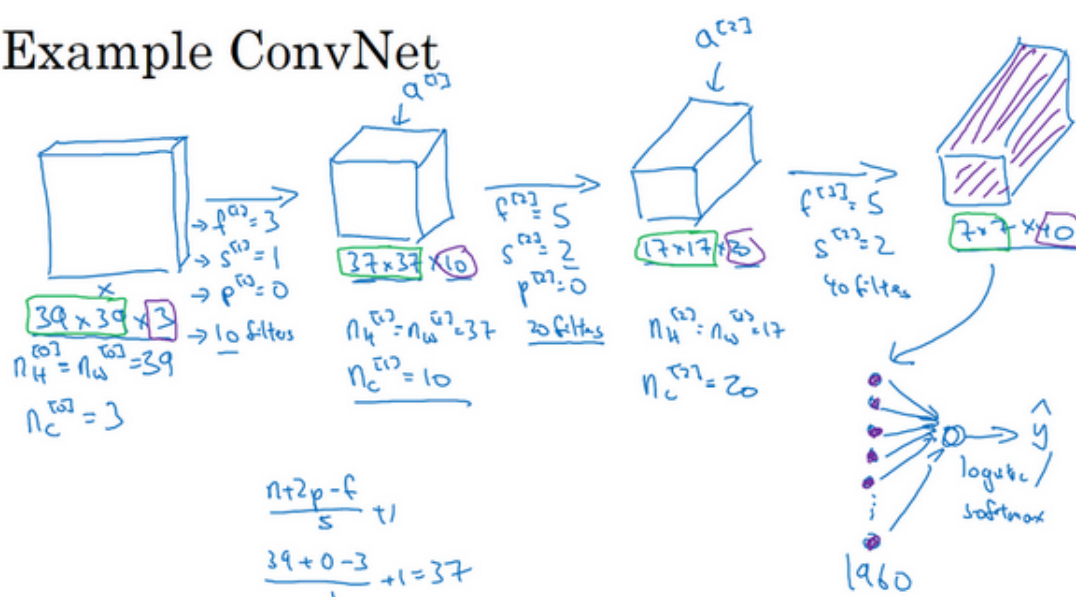
到此，这张  $39 \times 39 \times 3$  的输入图像就处理完毕了，为图片提取了  $7 \times 7 \times 40$  个特征，计算出来就是 1960 个特征。然后对该卷积进行处理，可以将其平滑或展开成 1960 个单元。平滑处



理后可以输出一个向量，其填充内容是 **logistic** 回归单元还是 **softmax** 回归单元，完全取决于我们是想识图片上有没有猫，还是想识别  $K$  种不同对象中的一种，用  $\hat{y}$  表示最终神经网络的预测输出。明确一点，最后这一步是处理所有数字，即全部的 1960 个数字，把它们展开成一个很长的向量。为了预测最终的输出结果，我们把这个长向量填充到 **softmax** 回归函数中。

这是卷积神经网络的一个典型范例，设计卷积神经网络时，确定这些超参数比较费工夫。要决定过滤器的大小、步幅、padding 以及使用多少个过滤器。这周和下周，我会针对选择参数的问题提供一些建议和指导。


## Example ConvNet



而这节课你要掌握的一点是，随着神经网络计算深度不断加深，通常开始时的图像也要更大一些，初始值为  $39 \times 39$ ，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小，从 39 到 37，再到 17，最后到 7。而通道数量在增加，从 3 到 10，再到 20，最后到 40。在许多其它卷积神经网络中，你也可以看到这种趋势。关于如何确定这些参数，后面课上我会更详细讲解，这是我们讲的第一个卷积神经网络示例。

一个典型的卷积神经网络通常有三层，一个是卷积层，我们常常用 **Conv** 来标注。上一个例子，我用的就是 **CONV**。还有两种常见类型的层，我们留在后两节课讲。一个是池化层，我们称之为 **POOL**。最后一个是全连接层，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。幸运的是，池化层和全连接层比卷积层更容易设计。后两节课我们会快速讲解这两个概念以便你更好的了解神经网络中最常用的这几种层，你就可以利用它们构建更强大的网络了。

## Types of layer in a convolutional network:

- Convolution (conv) ←
  - Pooling (pool) ←
  - Fully connected (FC) ←
- 

再次恭喜你已经掌握了第一个卷积神经网络，本周后几节课，我们会学习如何训练这些卷积神经网络。不过在这之前，我还要简单介绍一下池化层和全连接层。然后再训练这些网络，到时我会用到大家熟悉的反向传播训练方法。那么下节课，我们就先来了解如何构建神经网络的池化层。