**课时5-卷积神经网络**

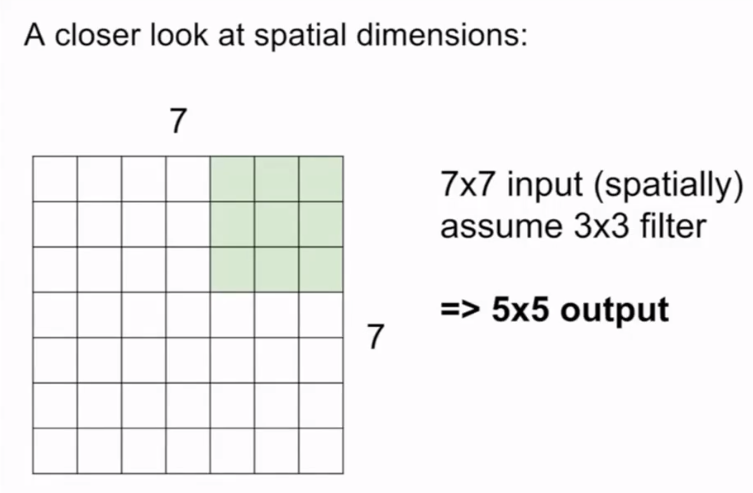


图5.1

一个图像到底是如何产生输出的激活映射（activation map）的呢？假定我们使用的是7\*7的输入图像，（为了方便假设成这样），我们有一个3\*3的卷积核，接下来将这个卷积核套在左上角，然后做乘法和点积，将所有的值乘起来得到我们的第一个值。接下来将卷积核从左往右滑动，从中得到另一个激活映射值，连续不断的得到新值，最后将得到一个5\*5的输出。如图5.1所示，基本上滑动卷积核是在水平方向上的5个空间位置以及垂直方向上的5个空间位置，

前面我们说滑动的时候有多个选项可选，每次滑动经过的区间叫做步长。最初我们用1作为步长，现在我们用2作为步长看看会发生什么？当采用值为2的步长时，我们会得到一个3\*3的输出。如图5.2所示。

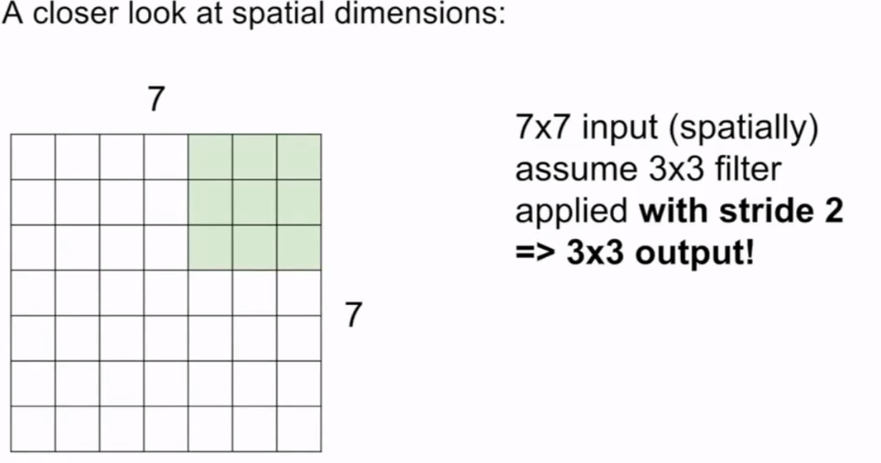


图5.2

步长为3呢？步长为3时候，和图像好像不能很好的拟合，说明行不通，我们不会做这样的卷积，因为他会导致不对称的输出。如图5.3

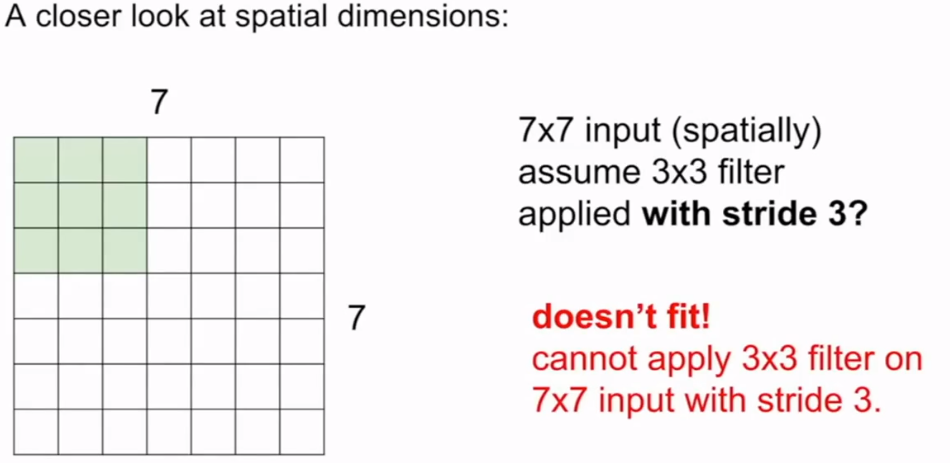


图5.3

来看一下如何计算输出尺寸的？如图5.4。假设输入的维度为*N*，卷积核大小为*F*，滑动步长为stride，最后输出的结果为：(N-F)/stride +1。

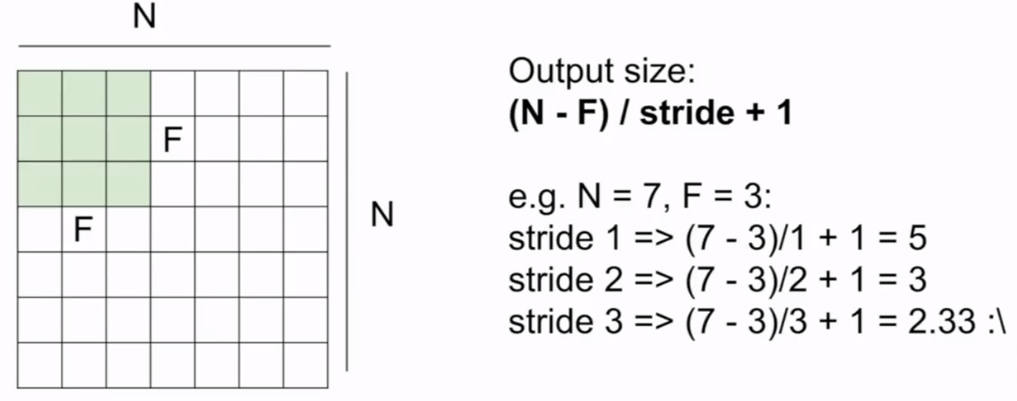


图5.4

如果输入是一样的，就是7\*7，采用零填充边界，卷积核的大小为3\*3，步长为1，输出大小会变成怎样？如果用上面的公式，N=7, F=3，那么输出为5\*5。但是现在实际的输出为7\*7，如图5.5。这里需要说明的是，在这里，这个公式的使用前提是没有0填充像素，但如果我们的确要填充它，然后你还需要一个新的输出，再用到前面的公式的时候，N=9，不是等于7，最后输出一个7\*7的激活映射。如图5.5

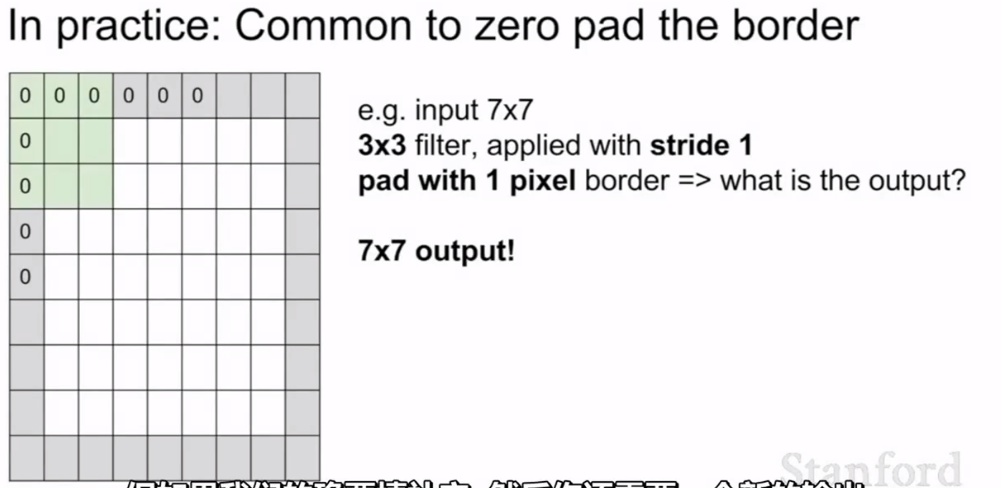


图5.5

**Q**:实际的输出是多少？是7\*7还是7\*7\*3？

Ans:输出是7\*7然后再乘以你使用的卷积核的数目。还应记得每个输出要在输入的整个深度上做点积。在这个例子里将产生一个1\*7\*7的激活映射图的输出，那么深度就是使用的卷积核的数量。

**Q:**当我们有一个32\*32\*3的输入时，这是怎么和前面关联起来的？

Ans：你的卷积核是一个3\*3再乘以你的输入深度的空间卷积核。就是3\*3\*3.

Q:零填充是否在图像的角落添加了一些额外的特征？

Ans: 在边缘处添加一些人为成分能够帮助探测到这个图像模板的某些特征，并且处理实际问题是合理的。

Q：如果有非方形的图片，我们会需要使用横纵不同的步长吗？是否需要使用横纵不同的步长？

Ans：我们通常是用同样的步长，因为我们通常是在方形区域内操作，通常在每个地方使用相同的步长。从某种意义上来说，他有点像你在看图片时的分辨率，通常你可能会想匹配你的垂直和水平分辨率。可以这么做，但一般不这么使用。

Q：我们为什么做零补充？

Ans:我们做零补充的方式是保持和我们之前的输入大小相同。我们最开始用的是7\*7，如果让卷积核从左上方角落开始，将所有的东西填入，那么我们之后将会得到一个更小的输出。但是我们想尽量保持全尺寸输出，所以用0填充一下。

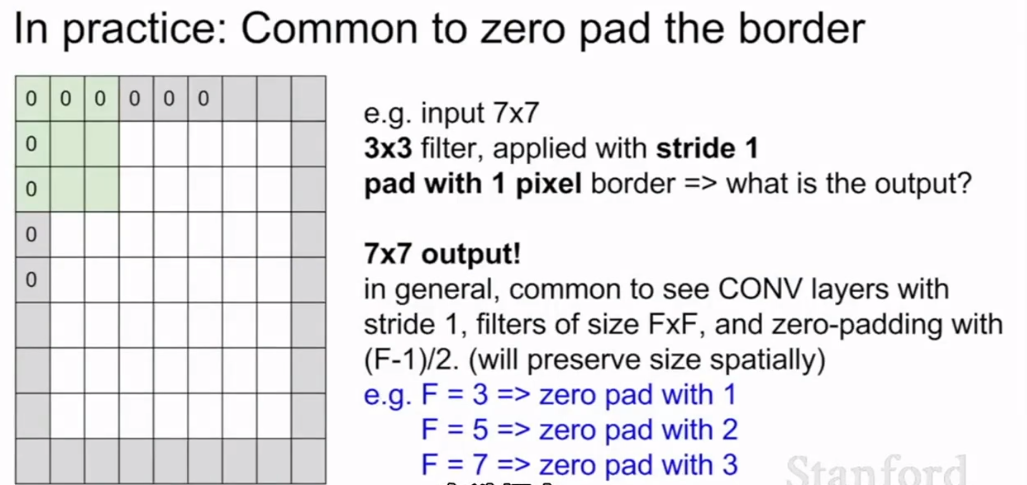


图5.7

关于如何填充能够基本上保证想要的输出尺寸，以及在边角区域应用卷积核，在选择上，步长及大小，卷积核以及卷积核的大小，零填充的个数。通常卷积核的大小为3\*3, 5\*5，7\*7等等。比如3\*3，为了保持图像空间的尺寸，在图像上进行宽度为1的零填充。具体的过程及填入零的大小见图5.7。不进行零填充的图像在卷积核输出的激活映射图中会迅速变小，这会损失一些信息，这样对于图像处理是不好的，你只能用很少的值来表示你的原始图像，同时每次图像变小我们都会损失一些边角信息、

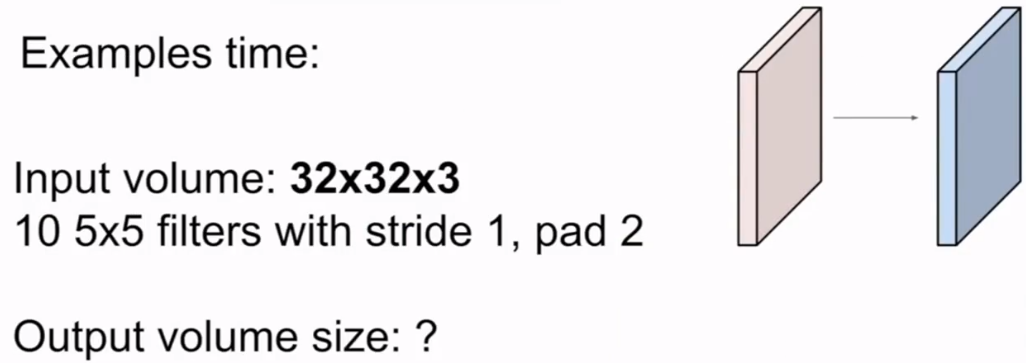


图5.8

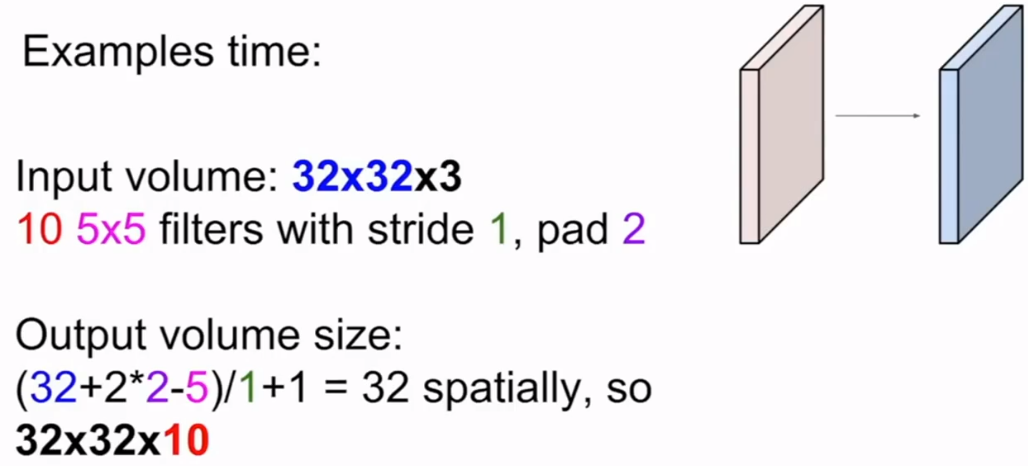


图5.9

看一个简单的例子。如图所示，假如输入是3层的32\*32的图形，用10个5\*5的卷积核（这里注意的是5\*5的卷积核指的是5\*5\*3的大小，深度没有指出来），步长为1，0填充宽度为2（即每个边上增加宽度为2的零填充），那么运用前面的公式，得到10个激活映射，输出应该为32\*32\*10。如图5.8、5.9所示。

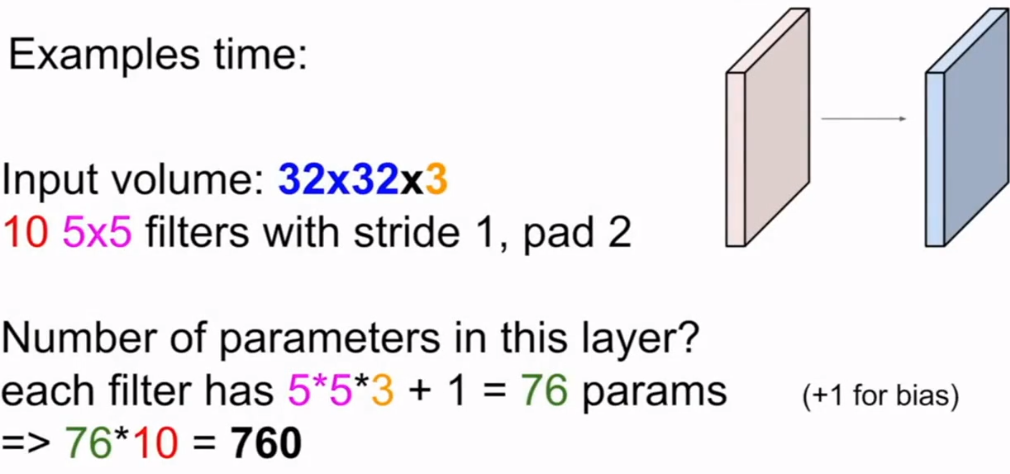


图5.10

那么参数的个数有多少个呢？每个卷积核的大小为5\*5\*3，加上每个卷积核有一个偏置项b，那么每个卷积核就有76个参数，10个卷积核共760个参数。如图5.10这里需要注意的是一个5\*5\*3的卷积核用一个数字来当bias，不是一个矩阵来当bias。

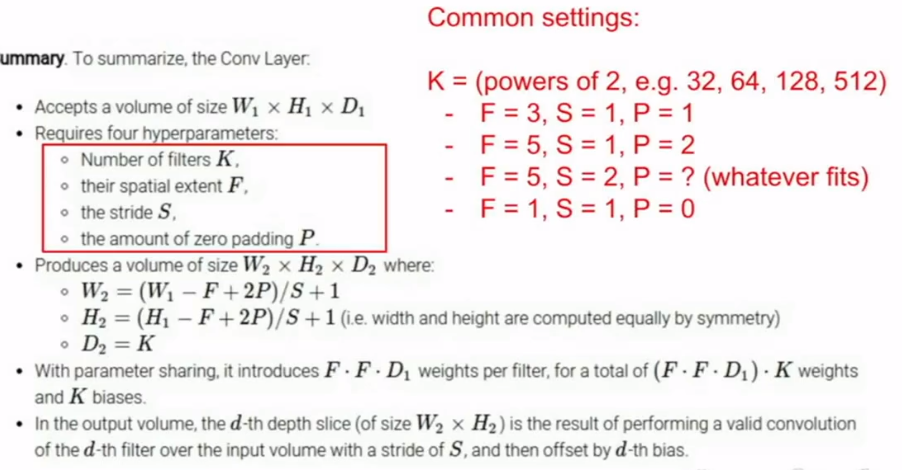


图5.11

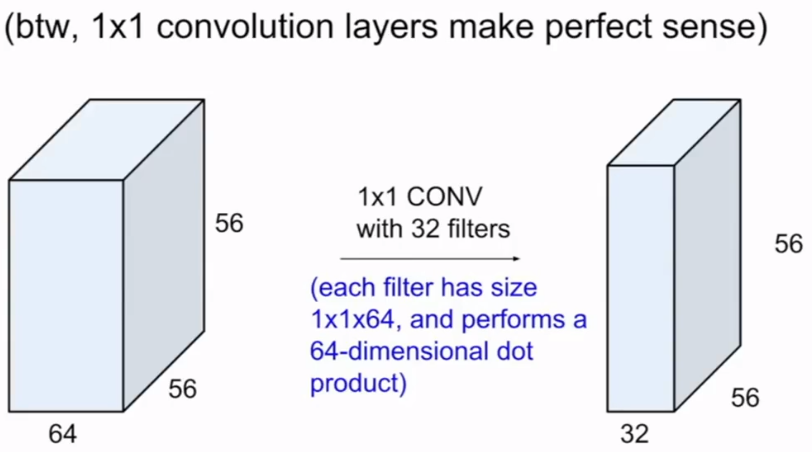


图5.12

以上图5.11是总结以及一些常用的设置。另一方面，我们也可以做1\*1的卷积，用这个卷积核在输入的整个深度上做一个点积。如果我们的输入大小是56\*56\*64，然后用32个1\*1的卷积核进行计算，输出就是56\*56再乘以卷积核的个数32。如图5.12。

Q:如何直观的来确定所使用的步长呢？

Ans:这个和图形的分辨率有关。当我们使用更大的步长时，我们所获得输出是一个降采样之后的图片，这个降采样之后的图片可以说是一种池化的处理，不过又比池化在某些时候更好一些，因为你能获得和降采样处理图片的相同效果。并且这个时候会缩小在每一层所处理的激活映射的尺寸，且之后还会影响到你所用到的参数的数目。

因为神经元具有局部连接性，所以我们并不关注所输入图片的全部，而是关注图像空间的一个局部区域。保留这个空间结构，就可以基于各层的激活映射进行推导。

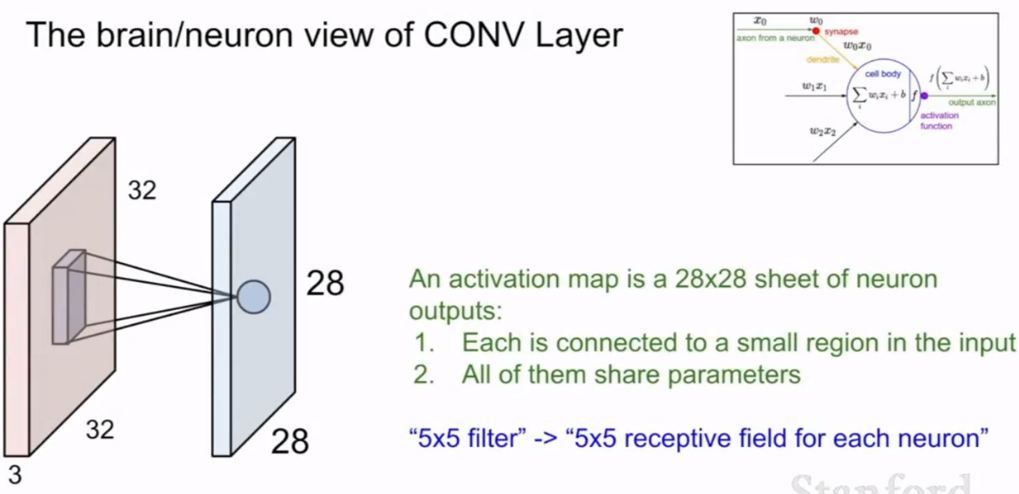


图5.13

接下来介绍一些术语。有一个5\*5的卷积核，我们可以称他为这个神经元的一个5\*5的感受野（receptive field）,因为这个感受野基本上讲的就是神经元在这个输入区域所能接受到的视野。对于每一个5\*5的卷积核，我们将它划过整个图像空间，但是这些卷积核都具有相同的权重，具有相同的参数。如图5.13。

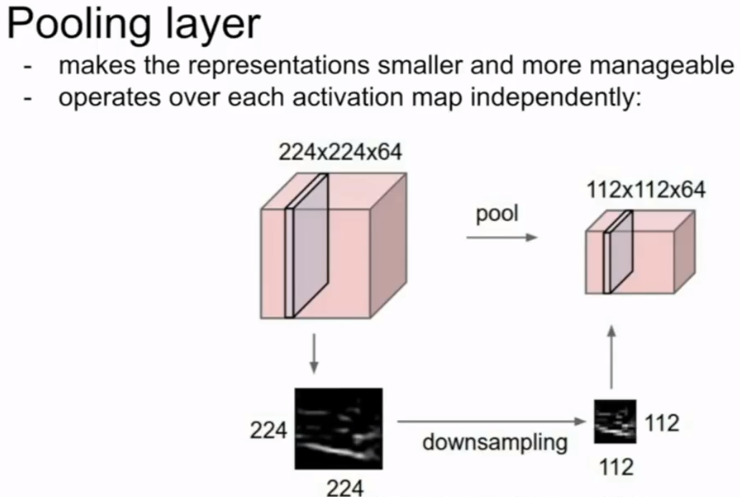


图5.14

关于池化层，他只是在做降采样处理吗？对输入的全体也有效吗？比如说对224\*224\*64的输入，在空间上进行降采样处理，最后的得到的是一个112\*112的结果，最重要的一点是，我们不做深度方向上的池化处理，而是只做平面上的，所以输入的深度和输出的深度是一样的。如图5.14.

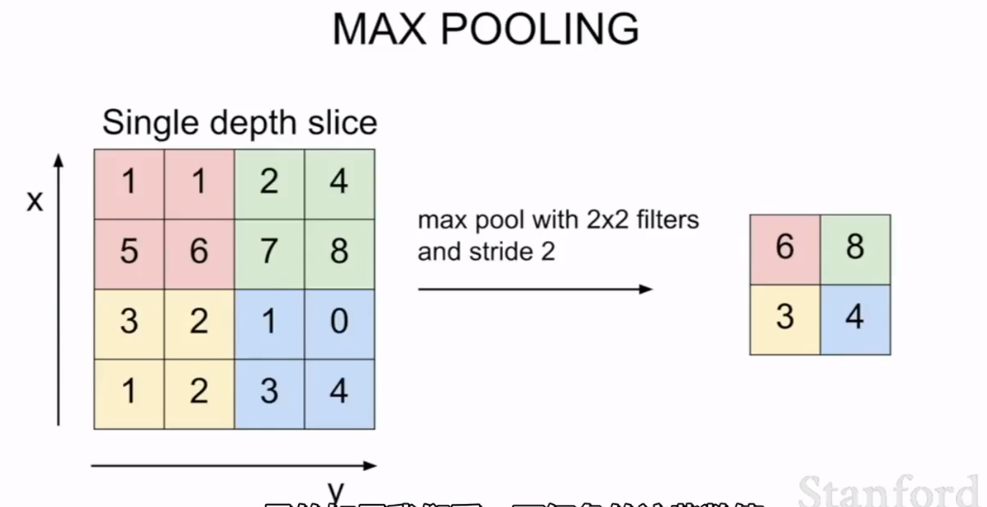


图5.15

最常见的方法是最大池化，在这个例子当中，池化层也有一个卷积核的大小，而且卷积核的大小和我们所要池化处理区域的大小是相同的。在这个例子中，如果我们使用2\*2的卷积核，并设定步长为2，我们让这个卷积核滑过整个输入部分，就像前面卷积时做的那样，不过我们不进行取点积的运算，而是只是提取其中最大的值，所在图像区域的输入最大值。如图5.15.

Q：这是否是通常的做法，就是设定步长，使他们不会相互重叠?

Ans:是的，对于池化层来说是这样的，通常的做法就是让他们没有任何重叠。你可以这么认为，我们希望进行降采样处理，对于给定的区域这样避免重叠的处理和只用一个数值来表示整个区域是有道理的，然后我们接着看下一个区域，并且继续下去。（为什么呢？没懂）

Q:为什么最大值池化要好于像均值池化之类的方式？

Ans:均值池化不是不能用，但最大值池化通常用得更多一些，因为他有类似这样的意义：我有一堆神经元的激活值在这儿，每个值都在一定程度上表示了，在这个位置某个神经元的激发程度，或者是某组卷积核的激发程度，你可以把最大池化看成这组卷积核在图像任意区域受激程度的表示，所以如果需要做检测，识别之类的任务，最大池化是最直观的，不管你要从图片里面找光线还是其他的类似信息，不管在区域内什么地方，用一个最显著的最大值来激活它。（这里的意思是不是，比如我在做图像识别的时候，要对整个卷积核的参数进行训练，我们采用最大池化的方式，然后训练的时候为了尽可能的识别准确，由于是对整个图像区域做卷积，卷积核会尽可能的找出这个物体识别的共有特征，然后放到输出激活映射里面，我们在训练的过程中会不断的增大那个最大池化的值，然后选取的时候就选取这个值来识别物体，而且不断缩小其他没被选中的最大池化值，然后在最后会增加识别的准确率。）

Q:既然池化和步长滑动都在降采样，能不能只池化不滑动步长，或者只滑动步长不池化？

Ans；实际上，在实践中，在一些神经网络的结构中，人们已经开始在用滑动代替池化来做降采样。我们可以把问题看成比例性的滑动，这么做可能会有稍微更好一些的结果。所以结果是可以的。

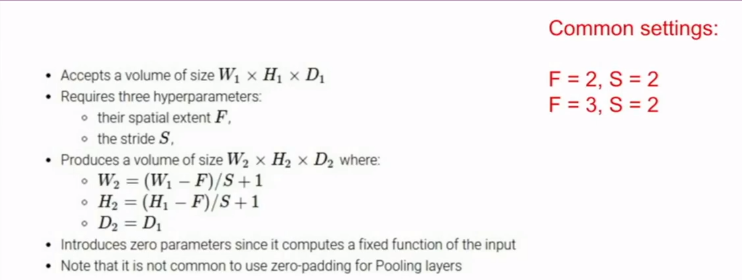


图5.16

以上图5.16是关于池化的一些信息以及常用设置，需要注意的是我们一般不在池化层做零填充，因为池化层只做降采样，这样就不会导致卷积核扫过边缘时有一部分超出了输入的范围。池化层的典型设置是2\*2的卷积核加步长2，或者3\*3的卷积核加步长3。

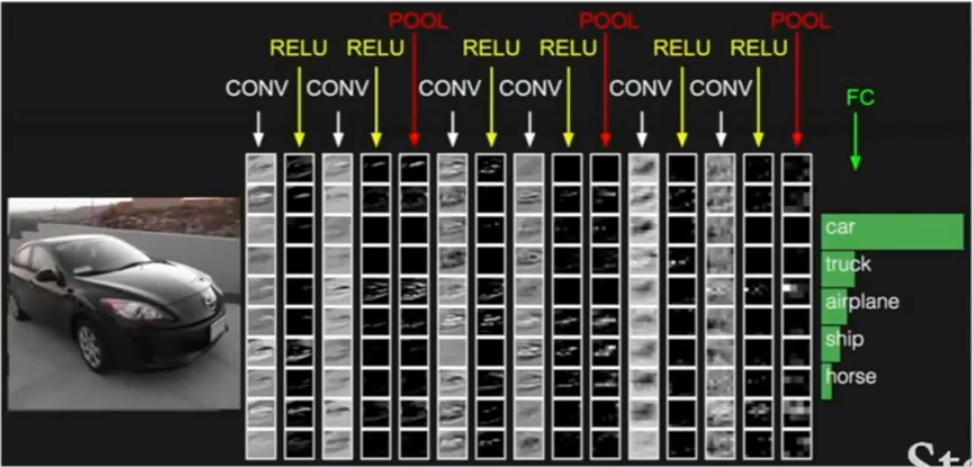


图5.17

经过卷积和池化之后，最后是一个全连接层，在这里我们把卷积网络最后一层的输出（即一个固定尺寸的矩阵，我们知道他的长宽高），把这样的矩阵直接拉平，这样就获得了一组一维输入，与全连接神经网络连接得到最后的全连接层，即每一个卷积图输出相连接的权重。现在不需要之前的空间结构了，而是在最后把所有的内容汇聚到一起，我们根据这些信息来得到一些结论，最后得到像分值一样的输出，如图5.17所示。

Q:5.17图上的最右边这一列怎么理解？比如pool

Ans:这里我们展示的每一列都是激活映射的输出，某一层的输出，一开始我们有个车，经过卷积层，得到的是每一个滑过输入图像的卷积核的激活映射，把激活映射送进ReLU得到一些值，然后一直往下走，在池化层得到的数据就是前面ReLU层的输出，池化层做了降采样，取出所有卷积核对应位置上的最大值，如果看下这个层化层的输出，他和ReLU层是一样的，只是做了降采样。

Q:仅仅取得到少量信息，怎么根据这个来区分？

Ans:每一个池化层输出的值，实际上是数据经过了整个网络处理后累积的结果。在最前面的层，每一个值都表示了上一阶段的某个概念，可以在最底层寻找边缘或者一些简单的结构，经过卷积层你看到的输出的第一列小图，一般是某些特定的激活特征，比如边缘在各个位置的体现，越往后得到的越复杂更高层的表示，下一个卷积层则会表现出，比如图片中的一些边角区域。所有这些都有含义，输入不再是原始图片，而是之前的输出，比如边缘的表示图，他在边缘图上进一步计算来得到更加复杂的内容，检测更复杂的事物。这样当你到达最后一个池化层的时候，每个值都代表了一组复合模板的激活情况。这样得到的全连接层把所有的信息聚合到一起，得到一组分类的分值。