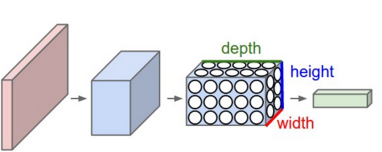
Convolutional Neural Networks

**结构总览**

卷积网络中，每层的神经元组织成3维：宽度(width)，高度(height)，深度(depth)（这里的深度值得不是网络层测的深度，而是一个激活容积(activation volume)的第三个维度）。



**构成卷积网络所用的层**

输入层(INPUT),卷积层(CONV)，池化层(POOL),激活层(RELU)和全连接层(FC)。将这些层次堆叠起来构成一个完整的卷积网络结构。

总的来说结构是：[输入-卷积-激活-池化-全连接]。

**卷基层：**

**整体感知**

卷积层的参数包括了一个可学习过滤器集合。每个过滤器空间上是很小的（宽度和高度上），但是它的扩展是输入容积的整个深度。

在前向过程中，滑动每个过滤器贯穿整个输入容量，并计算过滤器和当前位置上输入的点积。当滑动过整个输入的宽和高之后，每一个过滤器都会产生一个对应的二维激活映射图(activation map)。将所有这些二维映射图沿着深度唯独堆叠起来生成输出。

**局部连接(Local Connectivity)**

当处理高维度输入时（例如：图片），我们让每一个神经元仅与输入容积(input volume)的局部区域相连。这种连接的空间扩展是一种被称为接受域(receptive field)的神经元的超参数（hyperparapmeter）。链接在沿着深度轴的扩展总是与输入容积的深度相同。

**空间排列**(**Spatial arrangement**)

输出容积(output volume)中有多少个神经元？或者说这些神经元是如何被排列的？

有3个超参数控制着输出容积：深度，步长和补0。

1. **深度**：对应着我们使用的过滤器的个数，每个过滤器学着去观察输入的不同方面。
2. **步长**：我们需要指定过滤器滑动的长度。步长为1，每次移动一个像素，步长为2，每次移动2个像素，……（步长大于2的更大值在实践中很少用到）。这样随着步长的增大，产生的输出容积会越来越小。
3. **补0**：有时候，为输入的周边扩充0会很方便。补0的大小（size of this **zero-padding**）是一个超参数。补0的好处是它允许我们控制输出容积的空间大小(宽和高)。

我们可以计算输出容积的空间大小(宽和高)，用下面的等式：

。

W：输入容积的大小

F:卷积层神经元的接受域大小

S:接受域的滑动步长

P:补0的行数或列数

使用 (W−F+2P)/S+1计算输出容积的单维度。

通常情况下，当步长为1时，补0的个数为P=(F−1)/2可以保证输入和输出有相同的空间大小（宽和高）。

**步长的限制。**空间排列的这些超参数（W,F,S,P）是相互限制的。例如：W=10，P=0, F=3时，那么S=2就不可能，因为使用上面等式算出的值是4.5，不是整数。

**参数共享**

卷积层的共享参数主要是用来控制权重参数的个数。假设经过输入层后的第一个卷积层有96个卷积核且它的容量是（11\*11\*3），输入的尺寸大小（宽和高）是55\*55，那么第一层权重参数总数为：55\*55\*96\*11\*11\*3 = 105,705,600。**共享参数**指的是，沿着深度方向(depth)，让卷积层的每一个深度上的神经元共享一组权重参数和一个偏执项。

**池化层**

卷积网络中，在连续的卷积层中间周期性的插入池化层是很常见的。它的作用是逐渐的减少网络中表征的空间大小（宽和高）而到达减少参数和计算的总量(Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network)，因此也控制过拟合。

池化层是在深度方向的每一个特征图上使用最大池化操作(using the Max operation)独立的进行操作。最常用的形式是一个尺寸为2\*2，步长为2的核的池化层 在输入容量的每一个深度层次上沿着宽和高进行下采样，并丢弃75%的激活值。

更一般的形式是，池化层：

接受容量为W1×H1×D1的输入

需要2个超参数：

空间（宽和高）扩展核F，步长S

产生一个容量为W2×H2×D2的输出：

W2 = (W1 - F) / S + 1

H2 = (W1 - F) / S + 1

D2 = D1

池化层一般是不用补0的

没有额外的参数引入

除了最大池化外，还有平均池化和2范式池化。相较于最大池化，平均池化过去经常用，现在则很少有用，因为最大池化在实践中效果更好。

**全连接层**

全连接层的神经元与其前一层中的所有激活出入都有连接。这种方式与传统的神经网络连接方式一致。

**卷积网络结构**

**层次模式**

最常用的卷积网络结构是：

INPUT->[[CONV -> RELU]\*N -> POOLING?]\*M -> [FC -> RELU]\*K->FC

\*表示重复，？表示可选项。此外，通常情况下

0 <= N <= 3，M >= 0，0 <= K <= 3，

为什么堆叠小的卷积核要好过一个大的卷积核？

假设我们堆叠了3个3x3个卷积层（每次卷积层后都接了一个非线性花的激活函数），在这样的安排下，第一个卷积层的每一个神经元在输入数据中有一个3\*3的视野，同样第二个卷积层在第一个卷积层上的有一个3\*3的视野，因此第二层在输入数据中有一个5\*5的视野。同样的第三个卷积层在第二个卷积层上有一个3\*3的视野，因此第三个卷积层在输入数据中有一个7\*7的视野。假设我们不用3个堆叠的卷积层，而用一个7x7的卷积核来代替，那么同样在找一个卷积层上的每个神经元在输入数据上都有一个7\*7的视野。但是这么做有几个不好的地方：

1. 卷积层的每一个神经元的计算是通过一个线性函数得到的，而前一种方式包含的非线性使得得到的特征表达能力更强。
2. 假设每一层的深度是C，那么单独一个卷积层的7x7的方式包含的参数个数为：C\*(7\*7\*C) = 49C2。而堆叠三个3x3的方式包含的参数个数为：3\*(C\*(3\*3\*C)) = 27C2。

直观上来看，堆叠小的卷积核得到的特征要比一个大的卷积核得到的特征更具表达能力。

**层规模**

**输入层**：输入层的大小应该可以被2整出很多次。常用的规格是32，64，96，224，384，512.

**卷积层**：卷积层应该使用较小的卷积核（例如，3x3或者最多5x5），S=1，且特别总要的是，为输入容积补0以保持卷积层不会改变输入层的空间尺寸(即宽和高)。对于一般的F，P=(F-1)/2的补0法可以保留输入容积的尺寸。

**池化层**：池化层的作用是负责降低输入的空间维度。最常用的设定是使用2x2的接受域的最大池化。