# 卷积和池化

## 卷积

使用传统神经网络处理机器视觉的一个主要问题是输入层维度很大。例如一张64x64x3的图片，神经网络输入层的维度为12288。如果图片尺寸较大，例如一张1000x1000x3的图片，神经网络输入层的维度将达到3百万，使得网络权重W非常庞大。这样会造成两个后果，一是神经网络结构复杂，数据量相对不够，容易出现过拟合；二是所需内存、计算量较大。解决这一问题的方法就是使用卷积神经网络（CNN）。

按照我们上面讲的图片卷积，如果原始图片尺寸为n x n，filter尺寸为f x f，则卷积后的图片尺寸为(n-f+1) x (n-f+1)，注意f一般为奇数。这样会带来两个问题：

* **输卷积运算后，出图片尺寸缩小**
* **原始图片边缘信息对输出贡献得少，输出图片丢失边缘信息**

### Padding方式

#### 1.1.1 same padding（补0）

为了解决图片缩小的问题，可以使用padding方法，即把原始图片尺寸进行扩展，扩展区域补零，用p来表示每个方向扩展的宽度。如下图1.1所示：

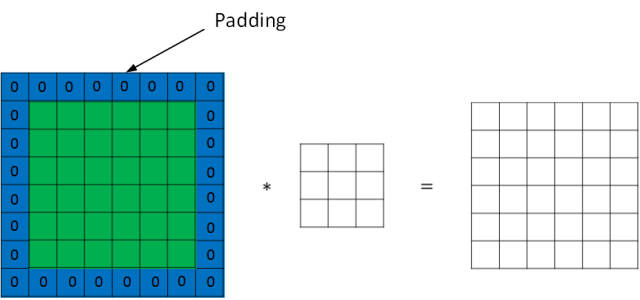


图1.1 Padding方式示意图

#### 1.1.2 Valid convolutions（padding=0）

没有padding操作，p=0，我们称之为“Valid convolutions”；有padding操作，p=(f−1)/2，我们称之为“Same convolutions”。

### 1.2步长（**Strided**）

Stride表示filter在原图片中水平方向和垂直方向每次的步进长度。之前我们默认stride=1。若stride=2，则表示filter每次步进长度为2，即隔一点移动一次。

值得一提的是，相关系数（cross-correlations）与卷积（convolutions）之间是有区别的。实际上，真正的卷积运算会先将filter绕其中心旋转180度，然后再将旋转后的filter在原始图片上进行滑动计算。filter旋转如下图1.2 所示：

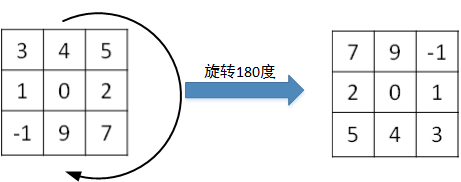


图1.2 卷积前操作

比较而言，相关系数的计算过程则不会对filter进行旋转，而是直接在原始图片上进行滑动计算。

其实，目前为止我们介绍的CNN卷积实际上计算的是相关系数，而不是数学意义上的卷积。但是，为了简化计算，我们一般把CNN中的这种“相关系数”就称作卷积运算。之所以可以这么等效，是因为滤波器算子一般是水平或垂直对称的，180度旋转影响不大；而且最终滤波器算子需要通过CNN网络梯度下降算法计算得到，旋转部分可以看作是包含在CNN模型算法中。总的来说，忽略旋转运算可以大大提高CNN网络运算速度，而且不影响模型性能。

### 1.4 卷积计算公式

我们用s表示stride长度，p表示padding长度，如果原始图片尺寸为n x n，filter尺寸为f x f，则卷积后的图片尺寸为：

（1.1）

### 1.5 多通道卷积

#### 1.5.1 多通道单核卷积

对于3通道的RGB图片，其对应的滤波器算子同样也是3通道的。例如一个图片是6 x 6 x 3，分别表示图片的高度（height）、宽度（weight）和通道（#channel）。3通道图片的卷积运算与单通道图片的卷积运算基本一致。过程是将每个单通道（R，G，B）与对应的filter进行卷积运算求和，然后再将3通道的和相加，得到输出图片的一个像素值，具体的计算如下图1.3

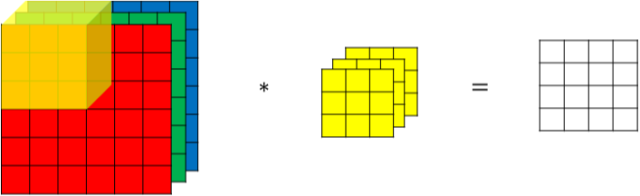


图1.3 RGB三通道彩色图像卷积示意图

不同通道的滤波算子可以不相同。例如R通道filter实现垂直边缘检测，G和B通道不进行边缘检测，全部置零，或者将R，G，B三通道filter全部设置为水平边缘检测。

为了进行多个卷积运算，实现更多边缘检测，可以增加更多的滤波器组。例如设置第一个滤波器组实现垂直边缘检测，第二个滤波器组实现水平边缘检测。这样，不同滤波器组卷积得到不同的输出，个数由滤波器组决定。

#### 1.5.2 多通道多核卷积

## 2. 池化（下采样）

池化是缩小高、长方向上的空间的运算。比如，如图所示，进行将 2×2的区域集约成1个元素的处理，缩小空间大小。一般来说，池化的窗口大小会 和步幅设定成相同的值。

#### 2.1 最大池化（max polling）

下图2.1为最大池化的示意图。

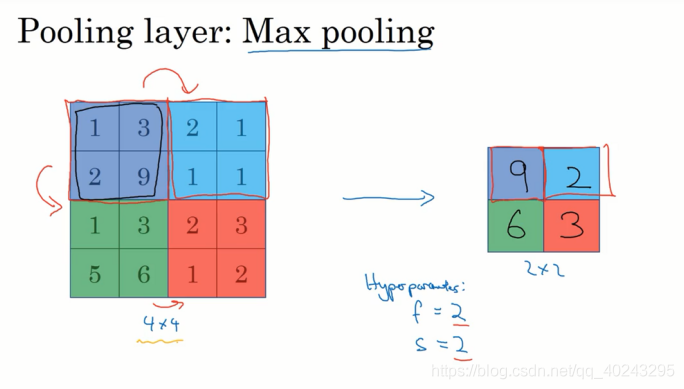


图2.1 最大池化示意图

#### 2.2 平均池化