**Pytorch实现卷积神经网络二-卷积神经网络卷积层和池化层**

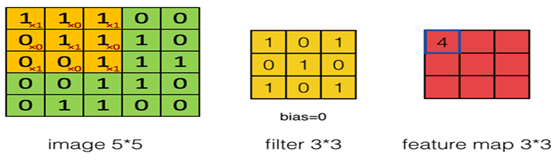
上次次推文我们讲述了卷积神经网络的卷积意义，用实例和图解的方式讲述了卷积神经网络中的卷积操作，之后讲述了卷积神经网络中各个层的实际作用。本次文章我们进一步详细的讲述卷积层、池化层相关参数的设置，如步长Stride,填充Padding等。

1. **特征图**

特征图即Feature Map，在一些教程中提及到了特征图的概念，那么这里进行简单的介绍一些，帮助大家进行理解。首先我们要知道的是在[CNN](https://so.csdn.net/so/search?q=CNN&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/xiaomai1996/article/details/_blank)中，Feature Map是在卷积操作下产生的，也就是说我们使用卷积核与原图进行卷积运算就会得到各种各样的feature map。当然了基于feature map进行卷积操作也会得到次层级的feature map。

很显然当我们的输入的数据是灰度图片，那就只有1个feature map，当我们输入数据是彩色图片，那么就有3个feature map（红绿蓝）。一般来说层与层之间会有若干个卷积核（kernel，有的称为顾虑器），上一层的每个feature map跟特定的卷积核做卷积，都会产生下一层的一个feature map。

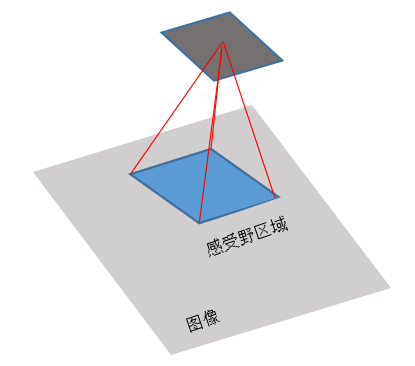
**一般来说在卷积的过程中有m个卷积核，下层就会产生m个feather map。**



1. **感受野**

感受野的相关知识在深度学习中还是比较重要的，也是一个老生常谈的话题。先说一个人类眼中感受野（Receptive Field）的事情，我们在使用手机进行浏览新闻时，当我们的被手机左上角的信息吸引时，在不转动视线的情况下我们是无法看清手机右下角的信息的，这个区域是模糊的。这就是人类感受野的这个这么一个认识。

那么在深度卷积神经网络中，感受野（Receptive Field）的定义是：每一个神经元节点都对应着输入图像的某一个特定的区域，该区域的图像的内容能对相应的神经元的激活具有绝对的话语权，那么这个区域我们称之为该神经元的感受野。如下所示：



红色线段指出的神经元是否激活只受到蓝色区域的图像的影响

那么为什么会使用感受野呢？主要是实际工程中处理图像这样的高维度数据的时候，会使用各种各样的手段降低神经网络中的参数，以加快神经网络的训练速度。基于这个前提，那么下层神经元与上层神经元之间进行**全连接（full connect）**就变得不在现实（计算和调整这些连接权重使得网路达到最优太耗费时间和资源），那么有什么办法解决这个问题呢？还好我们聪明，我们让下层的神经元与上层有限局部区域的神经元进行连接，使用权值共享的这个策略就可以解决训练难的问题，从这个角度上可以降低参数量。

在黄文坚中的《Tensorflow实战》中对权值共享的描述为：

“一个卷积层中可以有多个不同的卷积核，而每一个卷积核都对应于一个滤波后映射出的新图像，同一个图像中每一个像素都来自相同的卷积核，这就是卷积核的参数共享”

参数共享的理解很简单，我这里就不在赘述，大家随便百度一下就可以找到答案。可以学习一下这两篇论文:

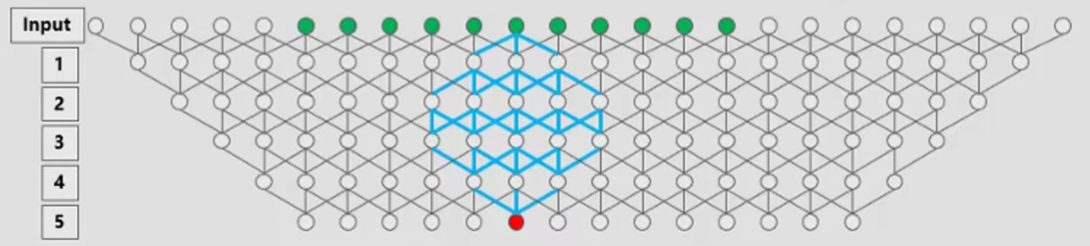
1.Generalization and Network Design Strategies,Y.le Cun 1989

2.Gradient-Based Learning Applied to Document Recognition,Yann LeCun 1998 (P42)

感受野有什么性质呢？参考网路上其他的文章大家赞同的性质为：

1. 越靠近感受野中间区域的数据越重要
2. 重要程度由中心向两边进行递减

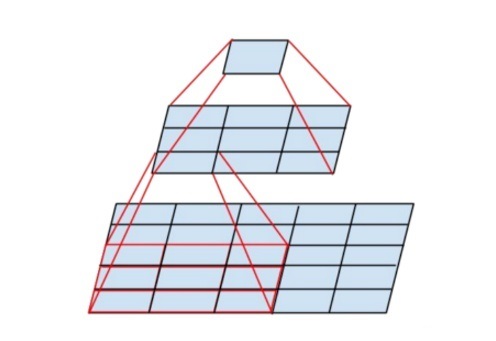
大家可以参考下图网络连接图示，中间的绿点对红点的影响权重最大：



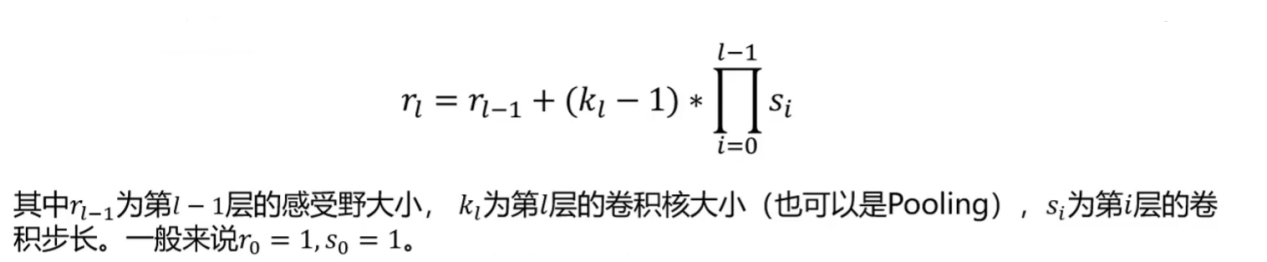
1. **感受野的计算**

很自然的我们想到感受野是一个区域，自然而然也就出现了感受野区域大小计算的问题。 那么感受野的计算该怎么进行呢？

假设我们输入一个尺寸大小是5\*5图像，经过两次3\*3的卷积核运算之后（其中我们固定参数stride=1,padding=0）得到1\*1尺寸的数据，如下所示：



这里我们先给出感受野的计算公式：



另外这里有几个注意点：

（1）最后一层（卷积层或池化层）输出特征图感受野的大小等于卷积核的大小。

（2）第i层卷积层的感受野大小和第i层的卷积核大小以及步长有关，而且与第（i+1）层感受野大小有关。

（3）计算感受野size时不用考虑padding大小。

至此我们来看看上述举例中的感受野的大小该怎么进行计算：

**第一次卷积操作**：5\*5大小的原始图像，经过kernal\_size=3,stride=1,padding=0的卷积后图像大小为3\*3，其感受野为： 1 + (3-1) \*1 = 3

**第二次卷积操作**：3\*3大小的特征图，经过kernal\_size=3,stride=1,padding=0的卷积后图像大小为1\*1，其感受野为： 3+(3-1)\*1\*1 = 5

**顺便得出一个小结论，在网络深度越深，感受野也越来越大。这样我们就完成了感受野的计算，更多的关于感受野的知识大家可以参考其他博客或者相关文献。**

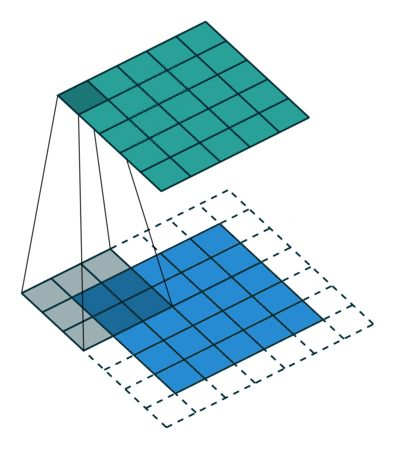
1. **卷积层**

卷积层的主要工作就是将输入图像与卷积核进行卷积的操作，那么我们自然而然的就会有一个疑问，我们有哪些常见的卷积呢？

关于有哪些常用的卷积以及相关的具体介绍，大家可以参考这篇论文：

[A Comprehensive Introduction to Different Types of Convolutions in Deep Learning](https://link.zhihu.com/?target=https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215" \t "https://www.zhihu.com/question/_blank)

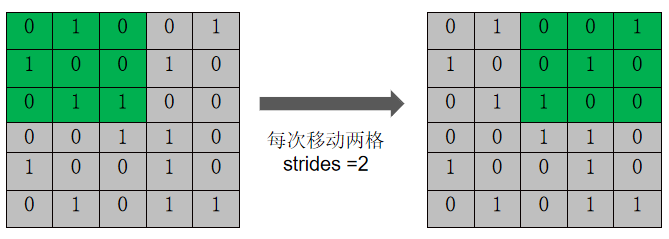
对于卷积层中的卷积操作，我们首先要明白几个专业的术语，理解好这几个术语可以更好的把握卷积操作的实际运行机制，我们以二维图像的一个卷积过程为例：



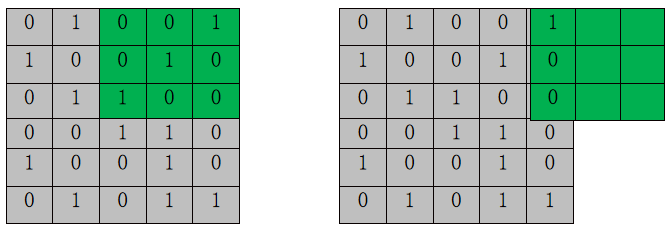
**动图**

1. Kernel Size：卷积核的大小，上图中灰色区域的长或者宽部分，显然Kernel Size = 3
2. Stride：步长，即卷积核在特征图像上的移动，上述图像中Stride=1表示在特征图上进行逐图像像素滑动，当Stride=2时表示每间隔一个像素进行滑动，显然当我们的Stride越大的时候得到的feature map将会变小（或者说输出的特征张量将会变小），大家自行体会。

Stride=2如下所示：

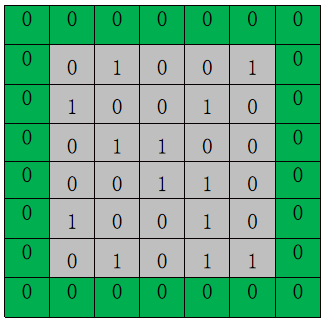


（3）Padding：边界填充，卷积核如果继续往右移动2格，卷积核窗口部分在输入矩阵之外，如下图所示此时我们就需要使用到填充这个技巧。



也就是说当输入图片与卷积核不匹配时或卷积核超过图片边界时，可以采用边界填充（padding）的方法。即以补0的方式将图片尺寸进行扩展，当然我们也可以选择不扩展，主要看网络设计人员的想法了。当我们选在填充的时候可以这样：

我们虽上述图片采用padding方法，使用0补充的结果如下：



这样当每一次滑动2格的时候就不会出现滑出原始特征图的“意外”。

Padding有两种可选的参数：Same和Valid，当使用Same方式时，对图片扩展并补0；采用Valid方式时，对图片不扩展。在实际训练过程中，一般选择Same，而且一个优点就是使用Same不会丢失信息。

这里大家可能有一个疑问，我怎么知道填充多少圈呢？很好，经前人的研究得到一下计算逻辑：假设我们需要填充的圈数为P,输入特征图的大小为N\*N，卷积的大小为K\*K，步幅大小为S，那么我们有：



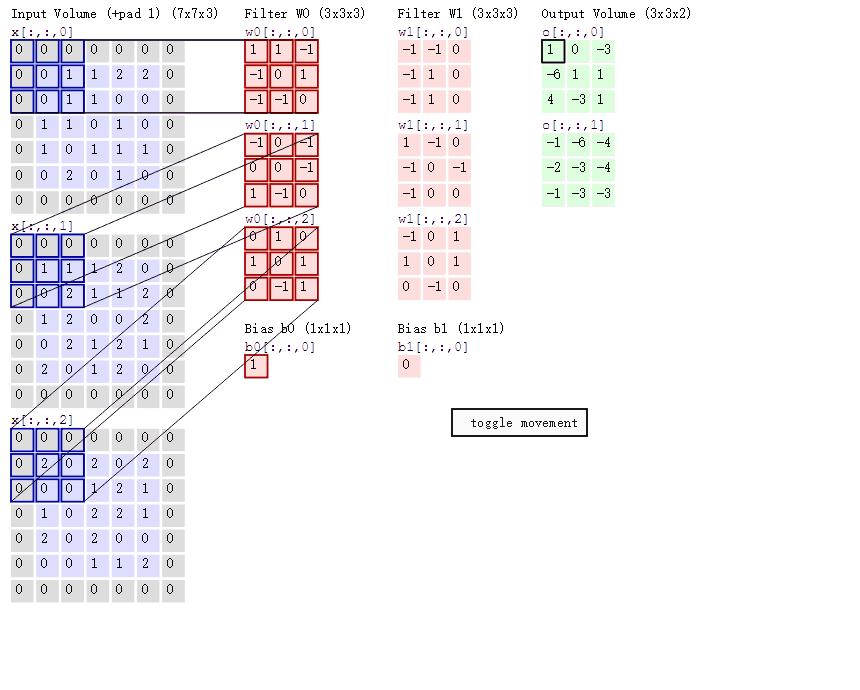
通过卷积操作之后，我们得到的特征图的大小为：



至此，我们就基本上了解到卷积层中的常用概念了。

1. **多通道的卷积**

多通道比如彩色图像3通道，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。卷积之前我们已经稍微的了解过，比如说这张动态的图：



例如一个7 x 7 x 3的图片，分别表示图片的高度（height）、宽度（weight）和通道（channel）。过程是将每个单通道（RGB）与对应的卷积核进行卷积运算求和，然后再将3通道的和相加，得到输出图片的一个像素值，（有时候会加上一定的偏置项）。

上述动图中使用了0进行padding，这对于图像边缘部分的特征提取是很有帮助的，可以防止信息丢失。

1. **池化层**

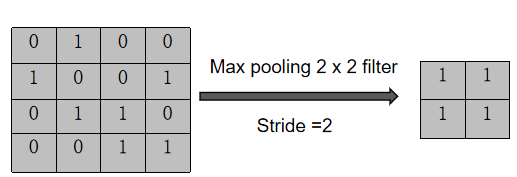
知道了卷积层之后，再来看池化层就非常简单了。池化层本质上可以认为是一个下采样层。

在构建网络的时候通常会在卷积层之间插入一个池化层。它的作用作用是降低数据体的空间尺寸，这样的话就能不断的减少网络中参数的数量，使得计算资源耗费变少，当然了参数变少了也就能在一定的程度上控制模型的过拟合。

常用的池化操作主要有两种：最大池化(max-pooling) 和平均池化(mean-pooling)。还有全局平均池化、全局最大池化、重叠池化和空金字塔池化。

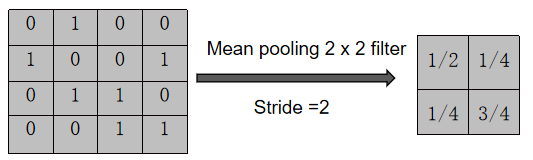
（1）最大池化层

最大池化是以一定的步长对特征图的特定区域内的像素点求得最大值。一般情况下是选取步长为 2，尺寸为 2×2 的滤波器，如果使用最大池化则有下图：



（2）平均池化层

平均池化是以一定的步长对特征图的特定区域内的像素点求其平均值。一般情况下也是选取步长为 2，尺寸为 2×2 的滤波器，如果使用平均池化则有下图：



池化层的输出通道跟输入通道数相同，平均池化历史上比较常用，但是现在已经很少使用了。因为实践证明，最大池化的效果比平均池化要好，当然我们也可以使用其他形式的[非线性](https://so.csdn.net/so/search?q=%E9%9D%9E%E7%BA%BF%E6%80%A7&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/Chen_Swan/article/details/_blank)池化函数。

可以看出无论最大还是平均池化，都没有需要学习的参数。因此在卷积神经网络的训练中进行反向传播的时候，Pooling层需要做的仅仅是将误差项传递到上一层，而没有梯度的计算。

**七、卷积层和池化层的作用总结**

接下来我们总结一下卷积神经网络中的卷积层和池化层的相关作用：

1. 卷积层的作用
2. 卷积核也被称为滤波器，卷积的深度和输入数据一致，可以进行特征的提取。

2.  降低参数的数量。这个由于卷积具有“权值共享”这样的特性，可以降低参数数量，在模型层面可以防止由于参数过多而造成过拟合。

（2）池化层的作用

1. 特征不变性：池化操作是模型更加关注是否存在某些特征而不是特征具体的位置。其中不变形性包括平移不变性、旋转不变性和尺度不变性。关于这三种性质可以参考其他材料进行学习。

2. 特征降维：池化的过程中减小了下一层的输入大小，进而减少计算量和参数个数，这在一定程度上防止过拟合。

3. 某一些池化的操作具有非线性的功能非线性。

1. **参考资料**

*https://cs231n.github.io/convolutional-networks/*

*https://www.zhihu.com/question/30888762*

*https://www.jianshu.com/p/9997c6f5c01e*

*<https://zhuanlan.zhihu.com/p/47184529/>*

*https://www.bilibili.com/video/BV17p4y1B78h?spm\_id\_from=333.880.my\_history.page.click*