深度学习系列（3）：卷积神经网络

本文将要介绍一种更适合图像、语音识别任务的神经网络结构——卷积神经网络(Convolutional Neural Network, CNN)。说卷积神经网络是最重要的一种神经网络也不为过，它在最近几年大放异彩，几乎所有图像、语音识别领域的重要突破都是卷积神经网络取得的。它在 2012 年崭露头角，Alex Krizhevsky 凭借它们赢得了那一年的 ImageNet 挑战赛（大体上相当于计算机视觉的年度奥林匹克），他把分类误差记录从 26% 降到了 15%，在当时震惊了世界。自那之后，大量公司开始将深度学习用作服务的核心。Facebook 将神经网络用于自动标注算法、谷歌将它用于图片搜索、亚马逊将它用于商品推荐、Pinterest 将它用于个性化主页推送、Instagram 将它用于搜索架构。打败李世石的AlphaGo也用到了这种网络。本文将详细介绍卷积神经网络的结构以及它的训练算法

一、初识卷积神经网络

1.1 全连接神经网络与卷积神经网络

全连接神经网络之所以不太适合图像识别任务，主要有三个方面的问题：

- 1）**参数数量太多：**考虑一个输入为1000×1000像素的图片（100万像素，现在已经不能算大图了），输入层有100万个节点。假设第一个隐藏层有100个节点（这个数量并不多），那么仅这一层就有（1000×1000+1）×100=1亿 的参数，这实在是太多了！我们看到图像只扩大一点，参数数量就会多很多，因此其扩展性很差。显而易见，这种全连接方式效率低下，大量的参数也很快会导致过拟合。

- 2）**没有利用像素之间的位置信息。**对于图像识别任务来说，每个像素和其周围像素的联系是比较紧密的，和离得很远的像素的联系就比较小了。若一个神经元和上一层所有神经元相连，那么就相当于对于一个像素来说，把图像的所有像素都等同对待，这不符合实际情况。当我们完成每个连接权值的学习之后，最终可能会发现，有大量的权值，它们的值都是很小的也就是这些连接其实都是无关紧要的。努力学习大量并不重要的权值，这样的学习必将是非常低效的。比如下面这个喵星人：

我们的任务就是想识别这只猫，按照之前的做法就是把这幅图片转换成一个一维向量，然后作为神经网络的输入。对于人眼的物体识别来说，虽然对人眼识别物体的原理并没有研究明白，但绝不是通过把物体转换为一维向量再做识别的。一张图片必然存在着一定的位置关系，比如猫的鼻子下面有嘴巴、鼻子上面有眼睛，这些都是很明确的位置关系，但要是转换成了一维向量，这些位置关系就被掩盖了。

- 3）**网络层数限制：**我们知道网络层数越多，其表达能力越强，但是通过梯度下降方法训练深度全连接神经网络很困难，因为全连接神经网络的梯度很难传递超过三层。因此，我们不可能得到一个很深的全连接神经网络，也就限制了它的能力。

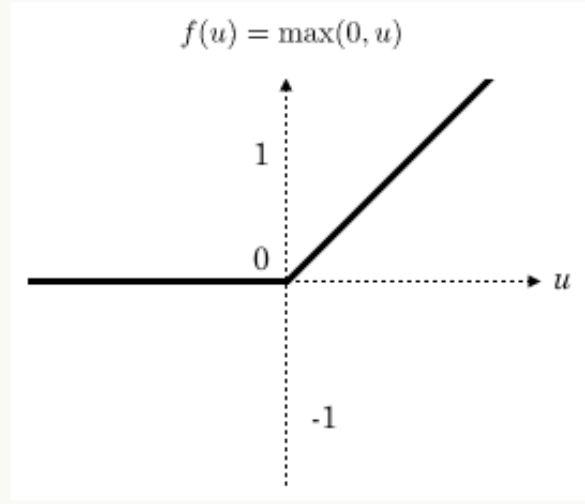
那卷积神经网络是怎么解决全连接神经网络的这些问题的呢？主要有以下几个方面。

1.2 激活函数——Relu

最近几年卷积神经网络中，激活函数往往不选择sigmoid或者tanh函数，而是选择relu函数。relu函数的定义是：

$$f(x) = \max(x, 0)$$

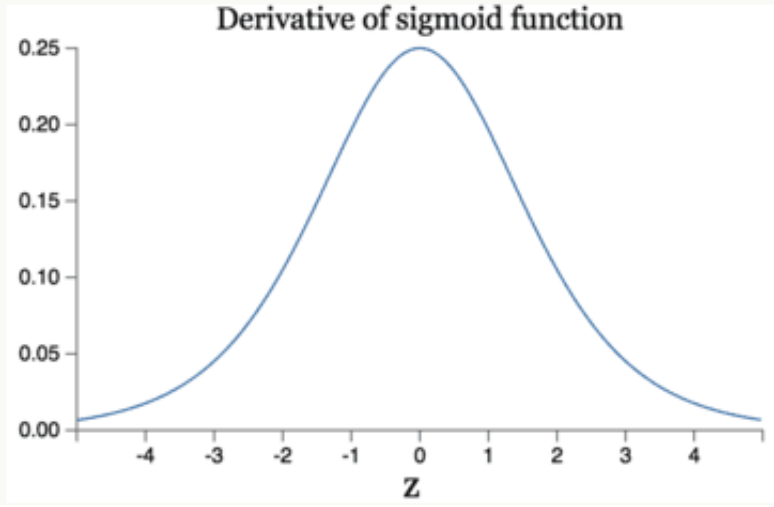
Relu函数图像如下所示：



Relu函数作为激活函数，有以下几大优势：

- 1）**速度快：**采用sigmoid等函数，算激活函数时（指数运算），计算量大，反向传播求误差梯度时，求导涉及除法，计算量相对大。而Relu函数其实就是一个max（x,0），整个过程的计算量节省很多。

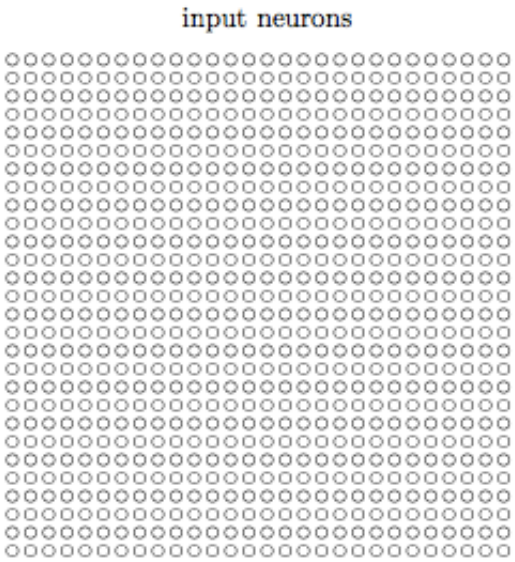
- 2）**减轻梯度消失问题：**回忆一下计算梯度的公式$\nabla = \sigma' \delta x$。其中$\sigma'$是sigmoid函数的导数。在使用反向传播算法进行梯度计算时，每经过一层sigmoid神经元，梯度就要乘上一个σ'。从下图可以看出，σ'函数最大值是$\frac{1}{4}$。因此，乘一个σ'会导致梯度越来越小，这对于深层网络的训练是个很大的问题。而Relu函数的导数是1，不会导致梯度变小。当然，激活函数仅仅是导致梯度减小的一个因素，但无论如何在这方面Relu的表现强于sigmoid。使用Relu激活函数可以让你训练更深的网络。



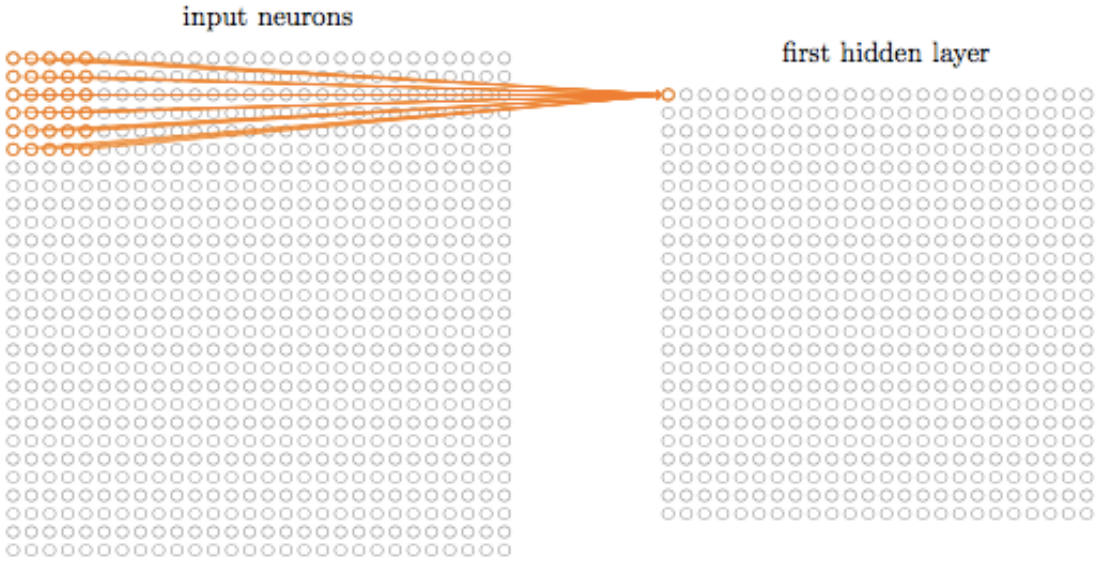
- 3）通过对大脑的研究发现，大脑在工作的时候只有大约5%的神经元是激活的，而采用sigmoid激活函数的人工神经网络，其激活率大约是50%。有论文声称人工神经网络在15%~30%的激活率时是比较理想的。因为Relu函数在输入小于0的时候是完全不激活的，因此可以获得一个更低的激活率。

1.3 局部感受野（local receptive fields）

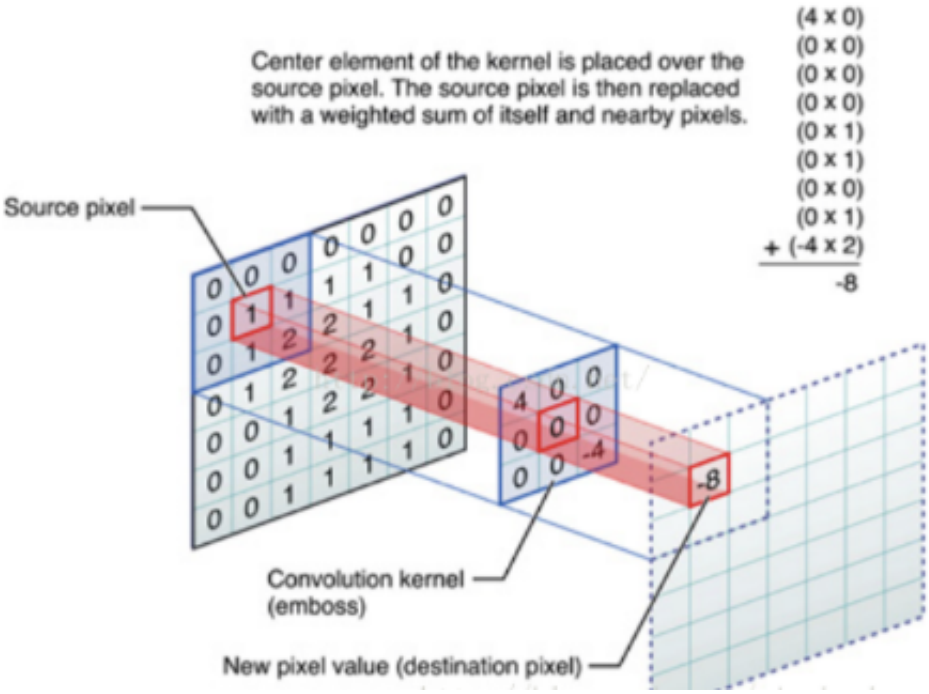
在之前的全连接神经网络中，一个样例的输入被转换为一个一维向量。但在一个卷积网络中，把输入看作是一个按照28×28排列的正方形，或者当有颜色通道的时候，比如28x28x3，就是宽高都是28，且有3个颜色通道。比如下图就代表了一个输入



然后，我们通常把输入像素连接到一个隐藏层的神经元，但和全连接神经网络那样每个输入都连接一个隐藏层神经元不同的是，这里我们只是把输入图像进行局部的连接。



如此不断地重复，构建起第一个隐藏层。注意如果我们有一个28×28的输入图像，5×5的局部感受野，那么隐藏层中就会有24×24个神经元。这是因为在抵达抵达最右边或最底部的输入图像之前，我们只能把局部感受野向右或向下移动23个神经元。



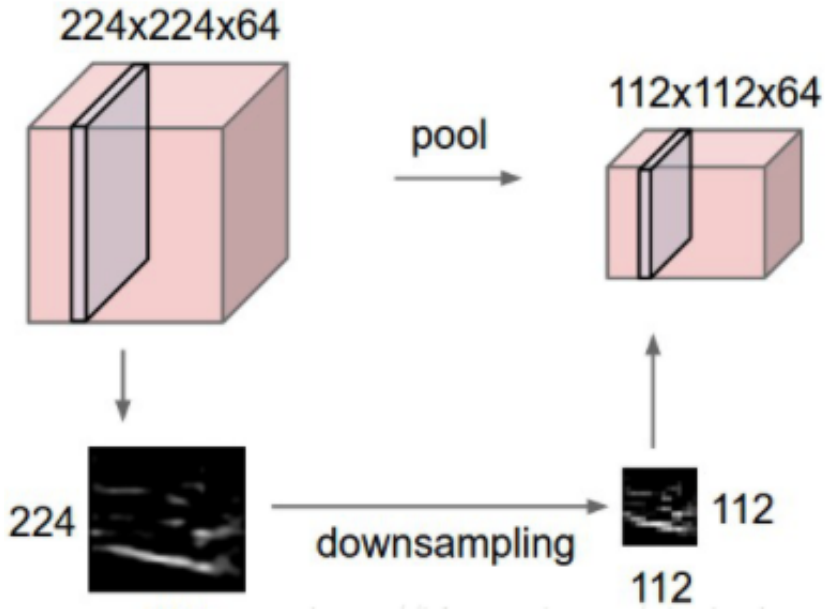
如上图所示，把图中间的那个看作是可以“滑动的窗口”，他的作用是和输入相应的“感受域”下的像素做运算得到新的值。这个运算就是“卷积”运算了。图上面有详细的运算过程。实际上就是每个相应元素的值相乘，然后把得到的都加起来。这个窗口的本质是其中的数字和一个偏置构成的，通常就把这个窗口叫做滤波器或者卷积核。上图是对于一个颜色通道的输入做卷积操作，但通常是三个颜色通道。中间那个“窗口”是可以滑动的，每次的滑动步长可以人为指定。

1.4 共享权值与偏置（Shared weights and biases）

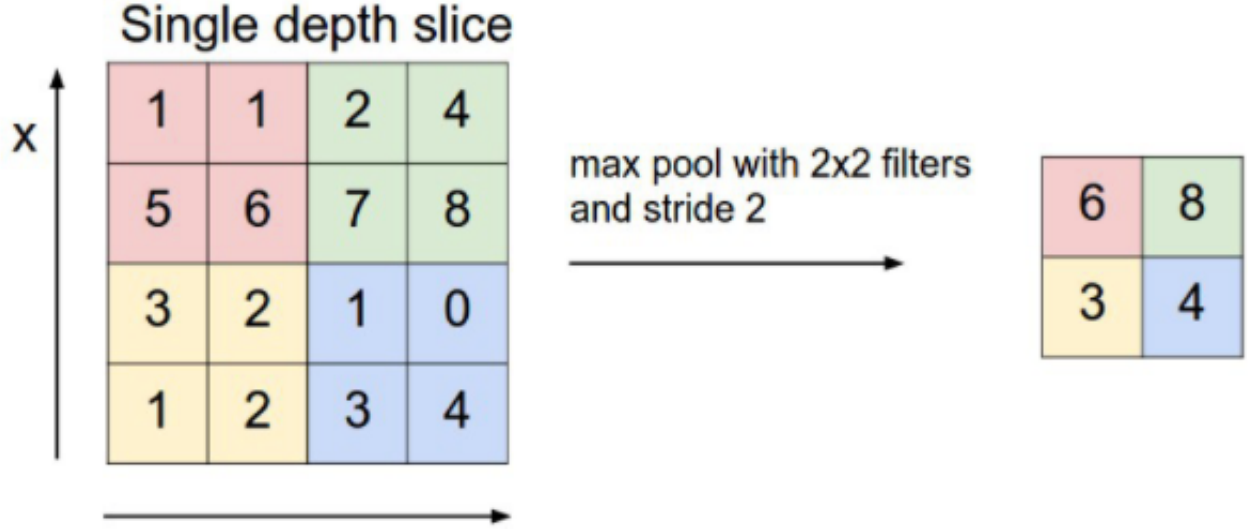
权值共享是指在一个模型的多个函数中使用相同的参数。 在传统的神经网络中，当计算一层的输出时，权值矩阵的每一个元素只使用一次， 当它乘以输入的一个元素后就再也不会用到了。而在卷积神经网络中，我们对24×24的隐藏层神经元的每一个使用相同的权重和偏置，这样可以很好地使用图像的平移不变性（例如稍稍移动一副猫的图像，它仍然是一副猫的图像）。因为这个原因，我们有时候把输入层到隐藏层的映射称为一个特征映射。把定义特征映射的权重称为共享权重，把以这种方式定义特征映射的偏置称为共享偏置。共享权值和偏置通常被称为一个卷积核或者滤波器。共享权值和偏置有一个很大的优点就是，它大大较少了参与卷积网络的参数，它的平移不变性将会使训练更快，有助于我们使用卷积层建立深度网络。

1.5 池化（pooling）

在连续的卷积层之间会周期性地插入一个池化层。经过池化层前后，发生的变化如下图所示：

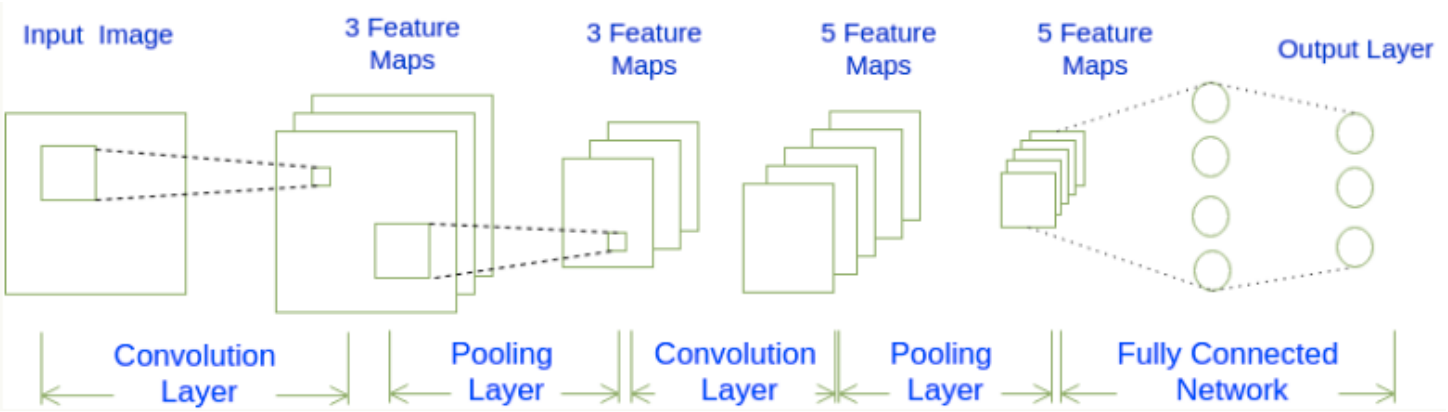


它的作用是逐渐降低数据体的空间尺寸，这样的话就能减少网络中参数的数量，使得计算资源耗费变少，也能有效控制过拟合。上图是一个MAX Pooling的过程，对输入数据体的每一个深度切片独立进行操作，改变它的空间尺寸。最常见的形式是汇聚层使用尺寸2x2的滤波器，以步长为2来对每个深度切片进行降采样，将其中75%的激活信息都丢掉。每个MAX操作是从4个数字中取最大值（也就是在深度切片中某个2x2的区域）。深度保持不变。



二、卷积神经网络的层

首先，让我们对卷积神经网络有一个感性的认识，下图就是一个卷积神经网络的示意图：



如上图所示，一个神经网络由若干卷积层（CONV）、Pooling层（POOL）、全连接层（FC）组成。你可以构建各种不同的卷积神经网络，它的常用架构模式为：

$$INPUT \rightarrow [[CONV] \times N \rightarrow POOL] \times M \rightarrow [FC] \times K$$

也就是N个卷积层叠加，然后叠加一个Pooling层（可选），重复这个结构M次，最后叠加K个全连接层。

对于上图来说，该卷积神经网络的架构为：

$$INPUT \rightarrow [[CONV] \times 1 \rightarrow POOL] \times 2 \rightarrow [FC] \times 2$$

也就是$N = 1, M = 2, K = 2$

从中我们可以发现卷积神经网络和全连接神经网络的层结构有很大不同。全连接网络每层的神经元是按照一维排列的，也就是排成一条线的样子；而卷积神经网络每层的神经元是按照三维排列的，也就是排成一个长方体的样子，有宽度、高度和深度。

我们看到输入层的宽度和高度对应于输入图像的宽度和高度，而他的深度为1。接着第一个卷积层对这幅图像进行了卷积操作，得到了三个Feature Map。实际上这个卷积层包含三个Filter，也就是三套参数，每个Filter都可以把原始输入图像卷积得到一个Feature Map，三个Filter就可以得到三个Feature Map。至于一个卷积层可以有多少个Filter，那是可以自由设定的。也就是说，卷积层的Filter个数也是一个超参数。我们可以把Feature Map可以看做是通过卷积变换提取到的图像特征，三个Filter就对原始图像提取出三组不同的特征，也就是得到了三个Feature Map，也称做三个通道(channel)。

在第一个卷积层之后，Pooling层对三个Feature Map做了下采样，得到了三个更小的Feature Map。接着，是第二个卷积层，它有5个Filter。每个Fitler都把前面下采样之后的3个Feature Map卷积在一起，得到一个新的Feature Map。这样，5个Filter就得到了5个Feature Map。接着，是第二个Pooling，继续对5个Feature Map进行下采样，得到了5个更小的Feature Map。

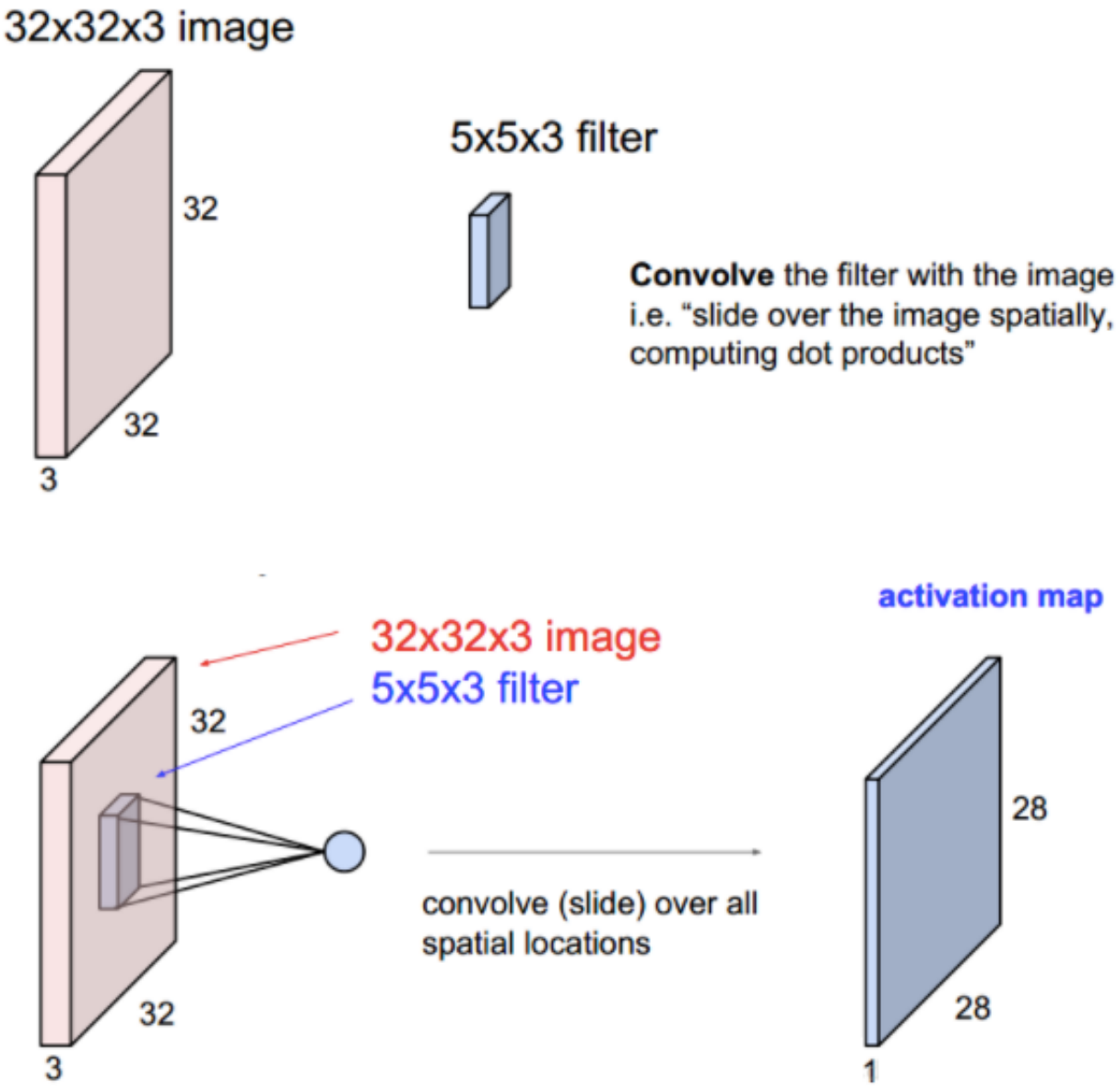
最后两层是全连接层。第一个全连接层的每个神经元，和上一层5个Feature Map中的每个神经元相连，第二个全连接层(也就是输出层)的每个神经元，则和第一个全连接层的每个神经元相连，

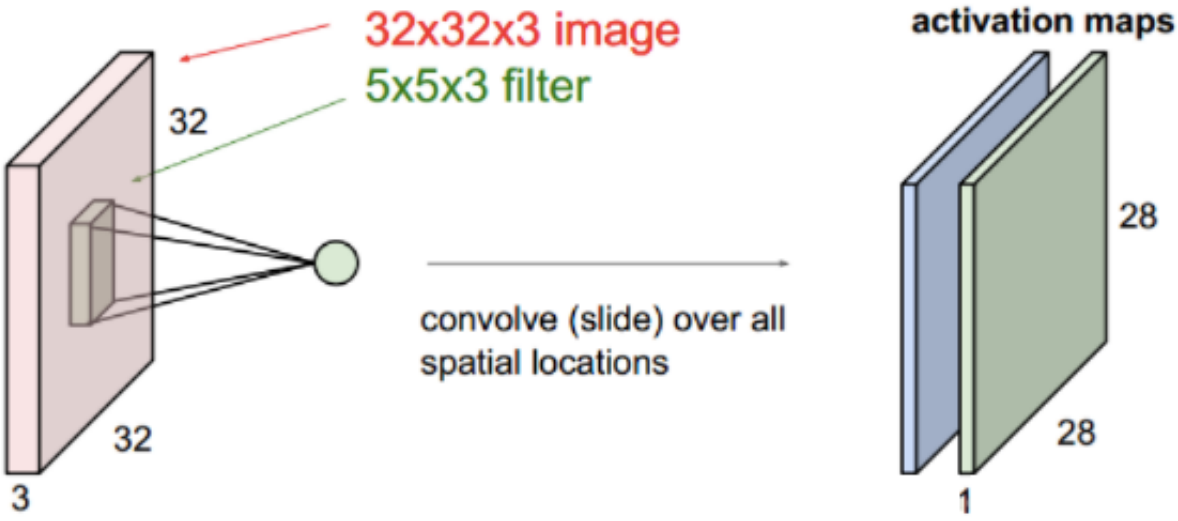
这样得到了整个网络的输出。

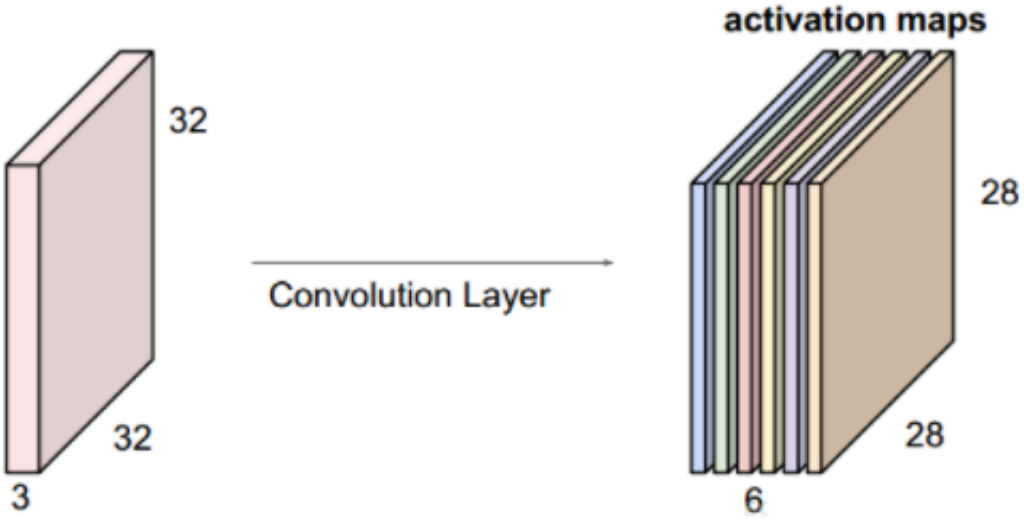
至此，我们对卷积神经网络有了最基本的感性认识。接下来，我们将介绍卷积神经网络中各种层的计算和训练。

2.1 卷积层

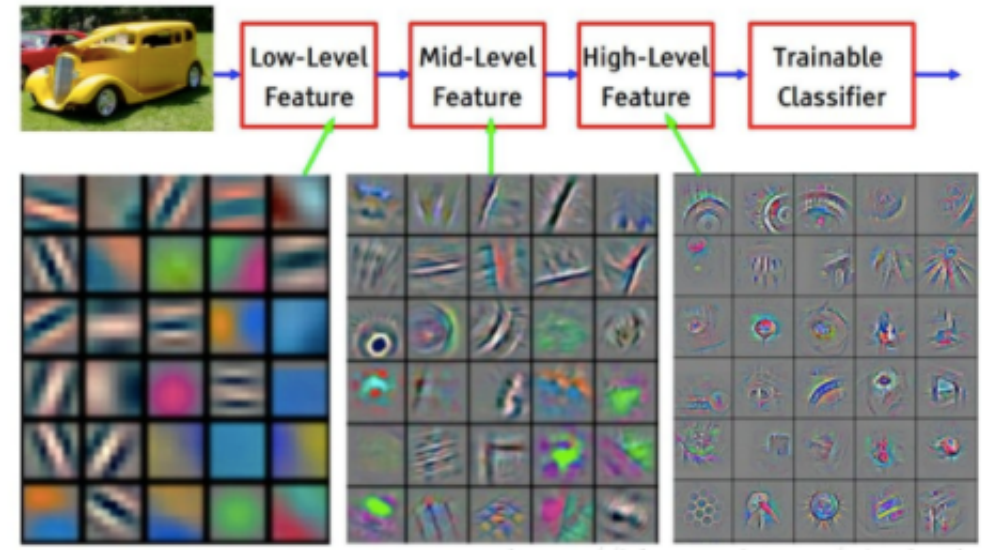
卷积层的参数是一些可学习的滤波器集合（卷积核）构成，滤波器的宽度和高度一般不大，深度与其输入数据保持一致。见下图：







在上面的过程中，原图像是32×32×3的图像，我们有一个滤波器（卷积核）为5×5×3，5×5的宽高相比起32×32来说，不怎么大，深度3和输入数据保持一致。一个卷积核在原图像上滑动，可以生成一个activation map，这里有6个不同的卷积核，得到的6个不同的activation map分别表示诸如边缘特征、形状特征等特征图，将这些activation map映射在深度方向上层叠起来就生成了输出数据。所以在用了6个过滤器（卷积层）之后，我们可以得到28×28×6的激活图。对各个activation map的直观感受可以看下图，其中每一个activation map代表着不同层次的特征。

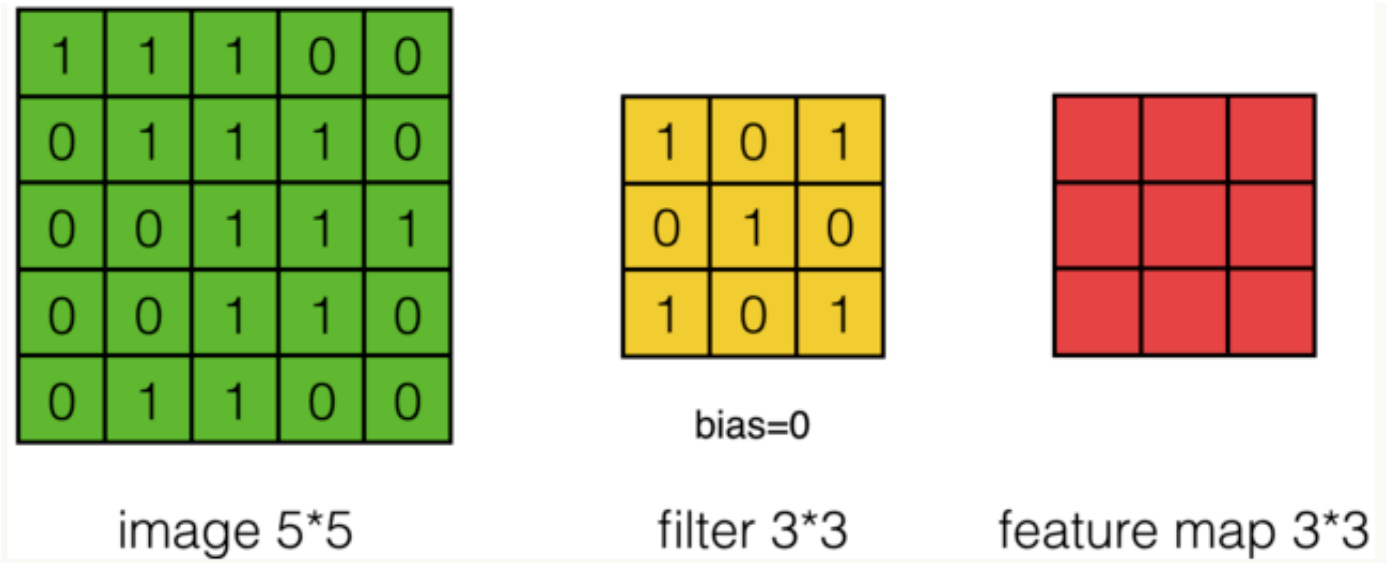


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

2.1.1 卷积层输出值的计算

我们使用一个简单的例子来讲述如何计算卷积，然后，抽象出卷积层的一些重要概念和计算方法。

假设有一个5×5的图像，使用一个3×3的滤波器进行卷积，想得到3×3的Feature Map，如下所示：



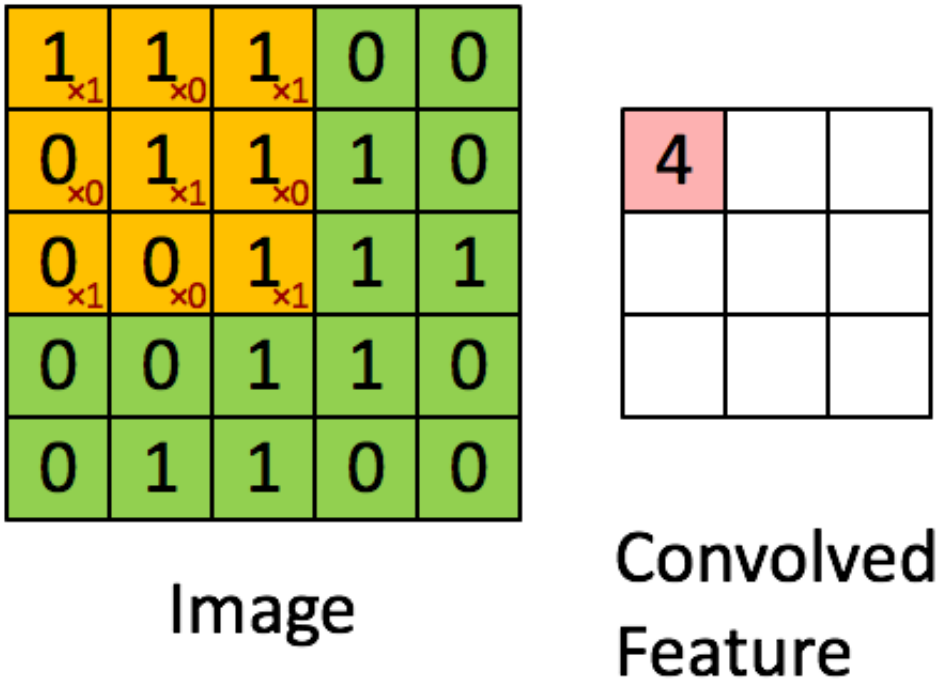
为了清楚地描述卷积的计算过程，我们首先对图像的每个像素进行编号，用$x_{i,j}$表示图像的第i行第j列元素；对filter的每个权重进行编号，用$w_{m,n}$表示第m行第n列权重，用w_b表示filter的偏置项；对Feature Map的每个元素进行编号，用$a_{i,j}$表示Feature Map的第i行第j列元素；用f表示激活函数（此处使用Relu函数作为激活函数）。然后使用下列公式计算卷积：

$$a_{i,j} = f\left(\sum_{m=0}^{2}\sum_{n=0}^{2} w_{m,n}x_{m+i,n+j} + w_b\right)$$

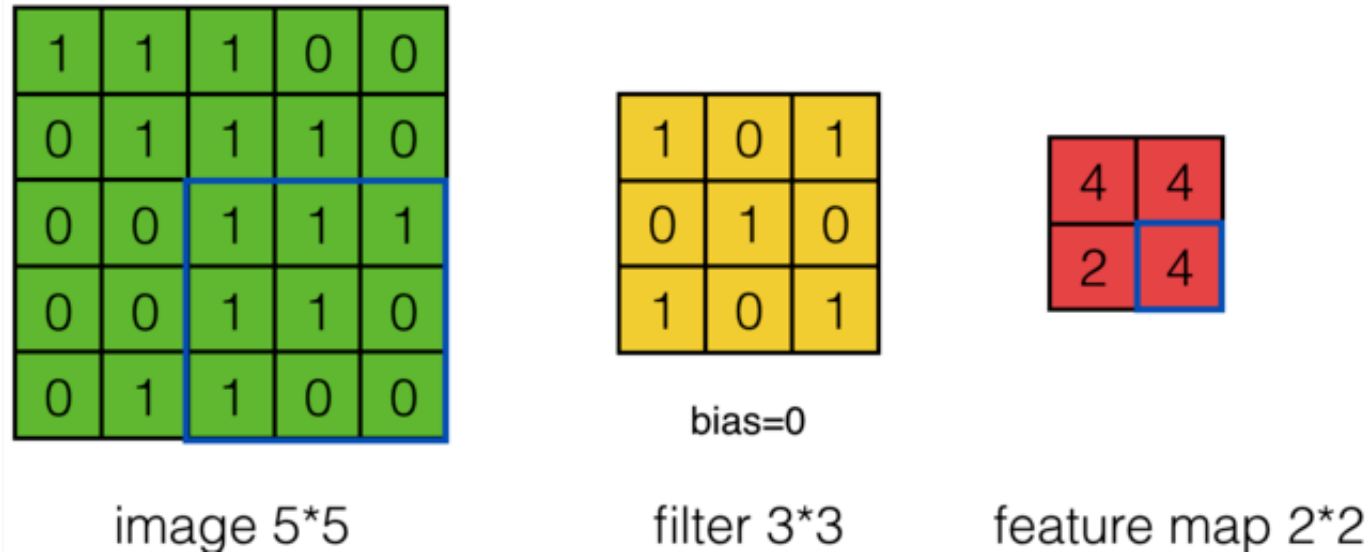
例如，对于Feature Map的左上角元素$a_{0,0}$来说，其卷积计算方法为：

$$a_{0,0} = f\left(\sum_{m=0}^{2}\sum_{n=0}^{2} w_{m,n}x_{m+0,n+0} + w_b\right) = Relu\,(4) = 4$$

按照这个公式可以依次计算出Feature Map中所有的值，下面的动画显示了整个Feature Map的计算过程：

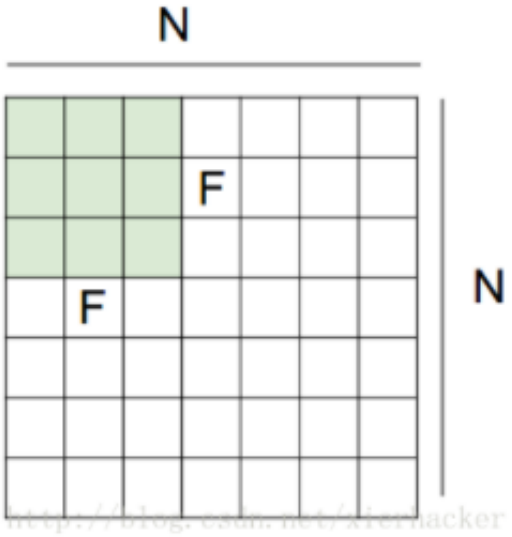


上面的计算过程中，步幅（stride）为1。当然步幅可以设为大于1的数。例如，当步幅为2时，Feature Map计算如下：



这里我们可以看到，当把步幅设置为2时，Feature Map就变成了2×2了。这说明图像大小、步幅

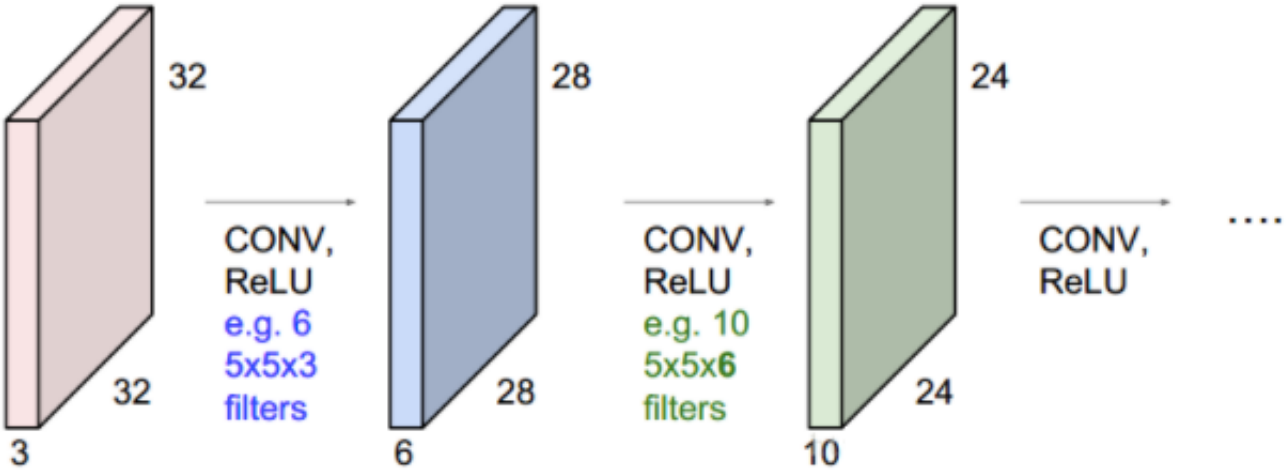
和卷积后的Feature Map大小是有关系的。我们设卷积前的图像宽度为N，步幅为S，filter的边长为F，卷积后Feature Map宽度为N_f，如图所示



则它们之间的关系为：

$$N_f = \frac{N - F}{S} + 1$$

但是这样持续卷积运算下去会出现一些问题，比如下图：



每经过一个filter，得到的激活图就会小一些，要是经过好几个，会导致最后消失殆尽。所以我们通过零填充（zero padding）的方法在原始图像周围补上几圈0，将补上的圈数设为P，则改写我们之前的关系式为：

$$N_f = \frac{N + 2P - F}{S} + 1$$

这里P乘以2是加了一圈之后两侧都加了1。例如上方那幅图中，$N = 5$，$F = 3$，$S = 1$，我们想保持卷积前后的尺寸保持不变，即$N_f = N = 5$，则零填充的P为：

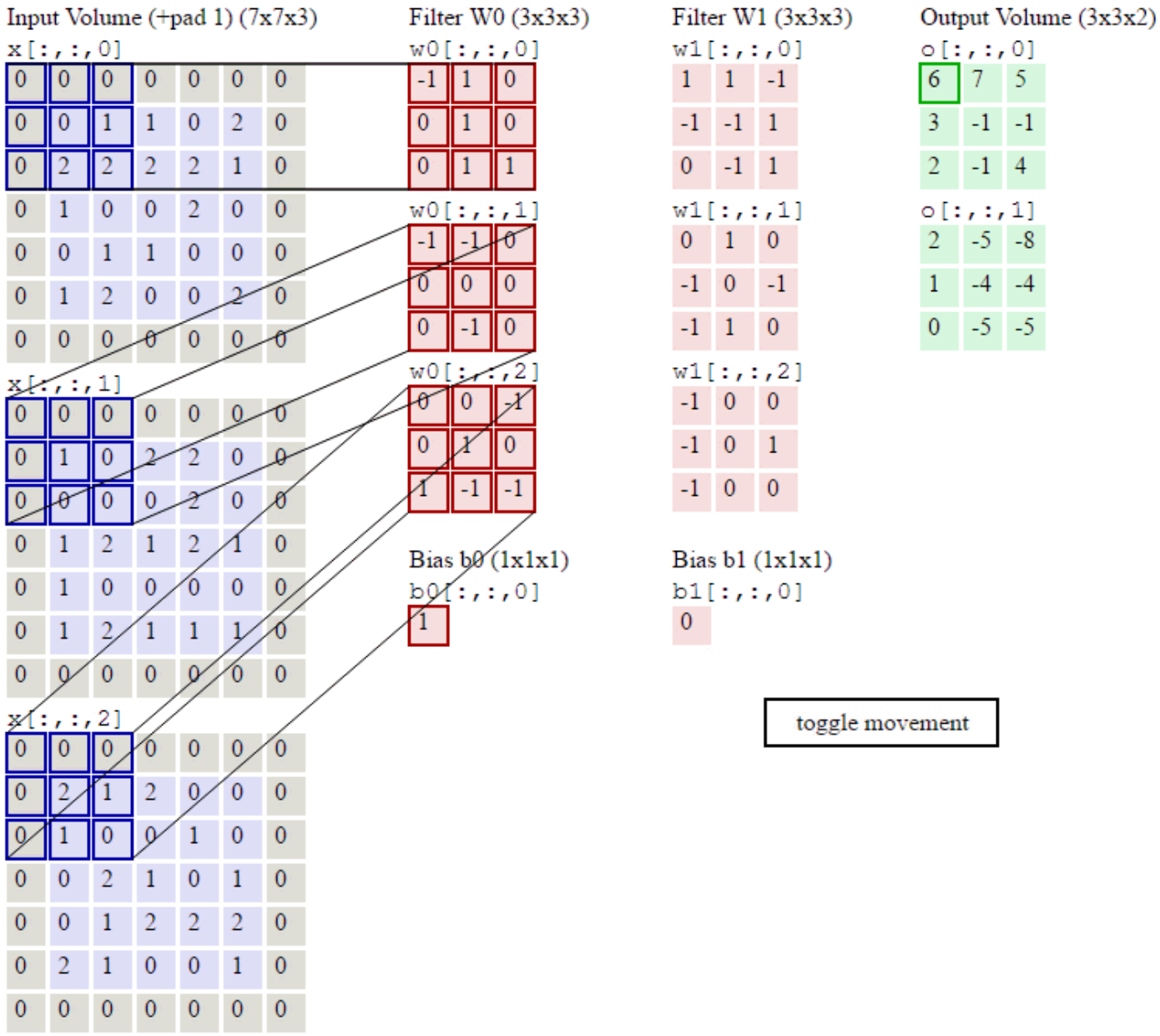
$$P = \frac{(N_f - 1)S + F - N}{2} = \frac{(5-1)\times 1 + 3 - 5}{2} = 1$$

到此我们讲了深度为1的卷积层的计算方法，如果深度大于1怎么计算呢？其实也是类似的。如果卷积前的图像深度为D，那么相应的filter的深度也必须为D。我们扩展一些之前的式子，得到深度为D的卷积计算公式：

$$a_{i,j} = f\left(\sum_{d=0}^{D-1}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} w_{d,m,n}x_{d,m+i,n+j} + w_b\right)$$

该式中，D为深度；$w_{d,m,n}$表示filter的第d层第m行第n列的权重；$a_{d,i,j}$表示图像的第d层第i行第j列像素。
我们前面还曾提到，每个卷积层可以有多个filterr。每个filter和原始图像进行卷积之后，都可以得到一个Feature Map。因此，卷积后Feature Map的深度和卷积层的filter个数是相同的。

下面的动画显示了包含两个filter的卷积层的计算。我们可以看到7×7×3的输入，经过两个3×3×3filter的卷积，其步幅为2，得到了3×3×2的输出，另外我们也会看到下图的Zero Padding是1，也就是在输入元素的周围补了一圈0。Zero Padding对图像边缘部分的特征提取是很有帮助的。



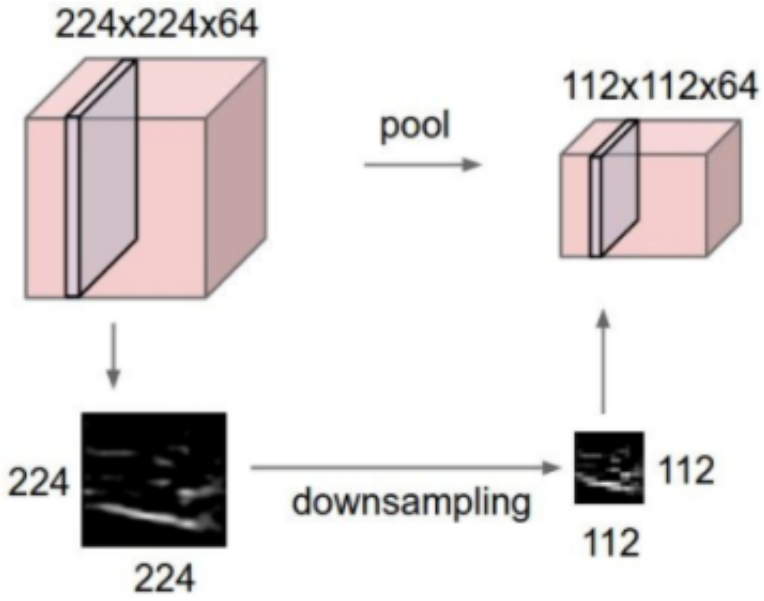
以上就是卷积层的计算方法。这里面体现了局部连接和权值共享：每层神经元只和上一层部分神经元相连（卷积计算规则），且filter的权值对于上一层所有神经元都是一样的。对于包含两个3×3×3的filter的卷积层来说，其参数数量仅有$(3 \times 3 \times 3 + 1) \times 2 = 56$个，且参数数量与上一层神经元个数无关。与全连接神经网络相比，其参数数量大大减少了。

2.1.2 用卷积公式来表达卷积层计算

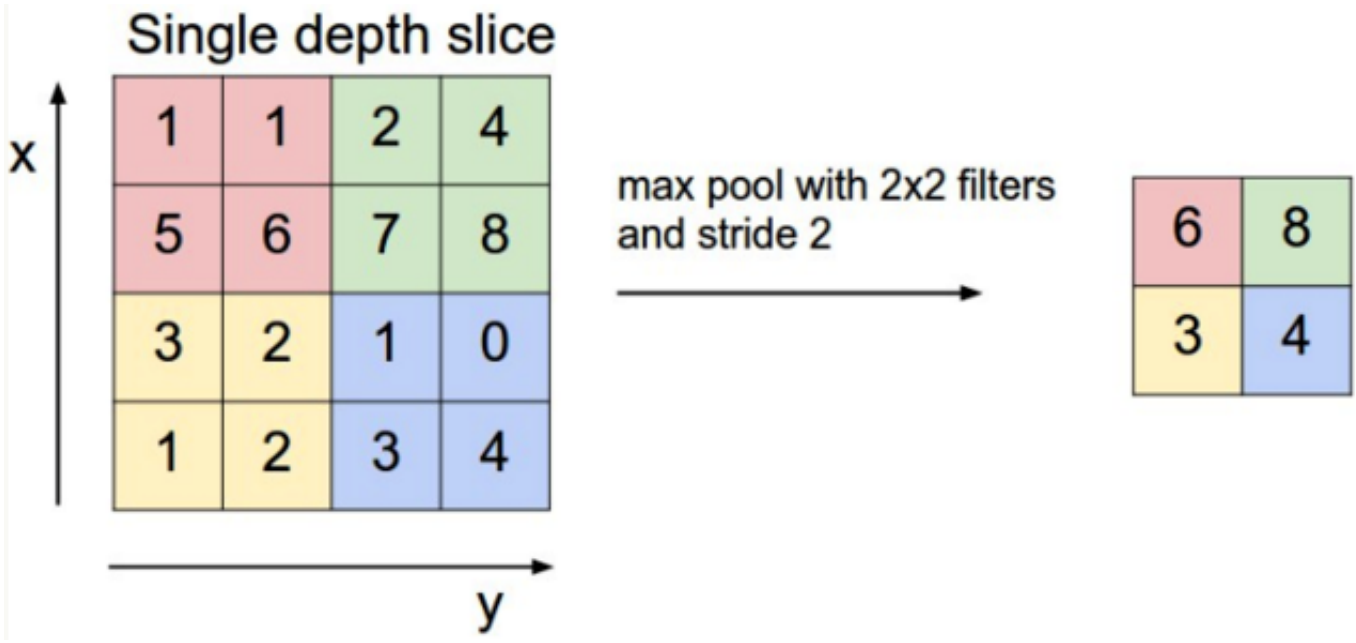
之前计算卷积层输出的式子很繁冗，最好可以简化一下

2.2 池化层（Pooling layer）

Pooling层的主要作用就是通过下采样，去掉Feature Map中不重要的样本，进一步减少参数数量，降低了计算成本，而且可以控制过拟合（overfitting）。池化层并不会对Feature map的深度有影响，即还是会保持原来的深度。



Pooling的方法很多，最常用的是Max Pooling，它实际上就是在$n \times n$的样本中取最大值，作为采样后的样本值。下图是2×2 Max Pooling：



此外，还有平均池化（average pooling）和L2-norm池化。平均汇聚历史上比较常用，但是现在已经很少使用了。因为实践证明，最大汇聚的效果比平均汇聚要好。池化层背后的直观推理是：一旦我们知道了原始输入中一个特定的特征，它与其他特征的相对位置就比它的绝对位置更重要。

很多人不喜欢汇聚操作，认为可以不使用它。比如在Striving for Simplicity: The All Convolutional Net一文中，提出使用一种只有重复的卷积层组成的结构，抛弃池化层。通过在卷积层中使用更大的步长来降低数据体的尺寸。有发现认为，在训练一个良好的生成模型时，弃用池化层也是很重要的。比如变化自编码器（VAEs：variational autoencoders）和生成性对抗网络（GANs：generative adversarial networks）。现在看起来，未来的卷积网络结构中，无池化层的结构不太可能扮演重要的角色。

2.3 归一化层

在卷积神经网络的结构中，提出了很多不同类型的归一化层，有时候是为了实现在生物大脑中观测到的抑制机制。但是这些层渐渐都不再流行，因为实践证明它们的效果即使存在，也是极其有限的。

2.4 全连接层

全连接层输出值的计算经网络和全连接神经网络是一样的，这里就不再赘述了。

三、卷积神经网络的训练

和全连接神经网络相比，卷积神经网络的训练要复杂一些。但训练的原理是一样的：利用链式求导计算损失函数对每个权重的偏导数（梯度），然后根据梯度下降公式更新权值。训练算法依然是反向传播算法。

我们知道神经网络和反向传播那一节中介绍的反向传播算法，它的整个算法分为三个基本步骤：

- 1）前向计算每个神经元的输出值a_j（j表示网络的第j个神经元，以下同）；
- 2）反向计算每个神经元的误差项δ_j，δ_j在有的文献中也叫作敏感度（sensitivity）。它实际上是网络的损失函数E_d对神经元加权输出net_j的偏导数，即$\delta_j = \frac{\partial E_d}{\partial net_j}$；
- 3）计算每个神经元连接权重w_{ij}的梯度（w_{ij}表示从神经元i连接到神经元j的权重，公式为$\frac{\partial E_d}{\partial w_{ji}} = a_i \delta_j$）,其中，$a_i$表示神经元$i$的输出。
- 4）根据梯度下降法更新每个权重即可

对于卷积神经网络，由于涉及到局部连接、下采样等操作，影响到了第二部误差项δ的具体计算方法，而权值共享影响了第三步权重w的梯度的计算方法。接下来，我们分别介绍卷积层和池化层的训练算法。

3.1 卷积层的训练

3.2 Pooling层的训练