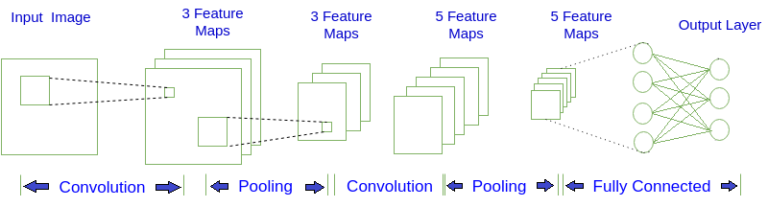
**卷积神经网络**

常用CNN网络（AlexNet，GoogleNet，VGG，ResNet，DenseNet，inceptionV4）适合初学者

<https://mp.weixin.qq.com/s?__biz=MzA4MTk3ODI2OA==&mid=2650343088&idx=1&sn=19e45c3a3ed2fedb7425b9649fd0161b&chksm=87811043b0f6995525dfba71abc667038229890bb14988f41319a9c3d6d5b48c5c41f9f9630a&mpshare=1&scene=1&srcid=&key=fb1dd35c5489928a87849b93dfab6d46191e46bf357ae89fd0782afd7bf86683c2afd7344a982505e20d6a6c1c697cc32a3066ddce4a97b2f37752dee2c2825b81baa2b66d57e11dd269ea62801183db&ascene=1&uin=MTYwOTEyNjk4Mw%3D%3D&devicetype=Windows+10&version=62060739&lang=zh_CN&pass_ticket=peTmuiErf8GOtfF7VDbxifRVe9pda7Gn%2FU5qAH5Uv7IUFRvscUgI3tQfgPSAAron>



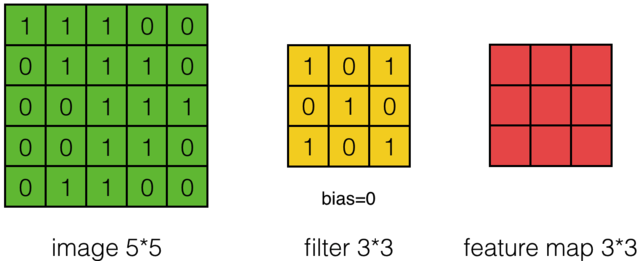
卷积神经网络基本结构

**传统神经网络对图片分类的不适用性**

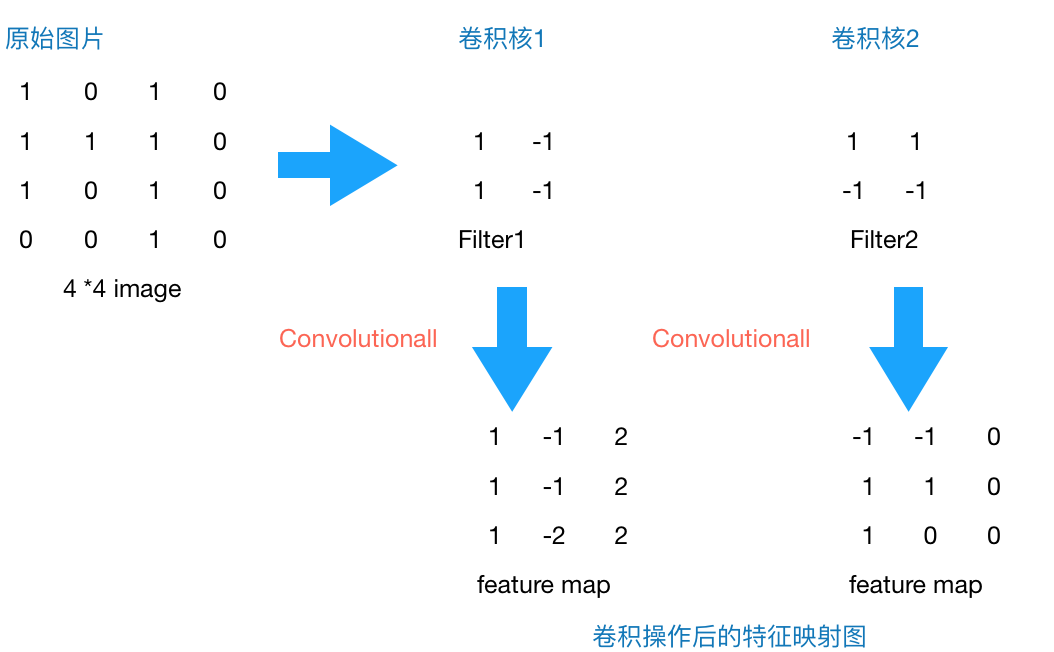
传统的三层神经网络需要大量的参数，原因在于每个神经元都和相邻层的神经元相连接，但是思考一下，这种连接方式是必须的吗？**全连接层**的方式对于图像数据来说似乎显得不这么友好，因为图像本身具有“**二维空间特征**”，通俗点说就是**局部特性**。

譬如我们看一张猫的图片，可能看到猫的眼镜或者嘴巴就知道这是张猫片，而不需要说每个部分都看完了才知道这个是猫。所以如果我们可以用某种方式对一张图片的某个**典型特征识别**，那么这张图片的类别也就知道了。这个时候就产生了**卷积**的概念。

**卷积**



举个例子，现在有一个4\*4的图像，我们设计两个卷积核，看看运用卷积核后图片会变成什么样。



4\*4 image与两个2\*2的卷积核操作结果

由上图可以看到，原始图片是一张灰度图片,每个位置表示的是像素值，0表示白色，1表示黑色，（0，1）区间的数值表示灰色。对于这个4\*4的图像，我们采用两个2\*2的卷积核来计算。设定步长为1，即每次以2\*2的固定窗口往右滑动一个单位。以第一个卷积核filter1为例，计算过程如下：

1 feature\_map1(1,1) = 1\*1 + 0\*(-1) + 1\*1 + 1\*(-1) = 1

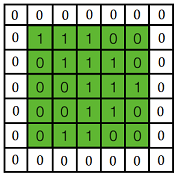
2 feature\_map1(1,2) = 0\*1 + 1\*(-1) + 1\*1 + 1\*(-1) = -1

3 …

4 feature\_map1(3,3) = 1\*1 + 0\*(-1) + 1\*1 + 0\*(-1) = 2

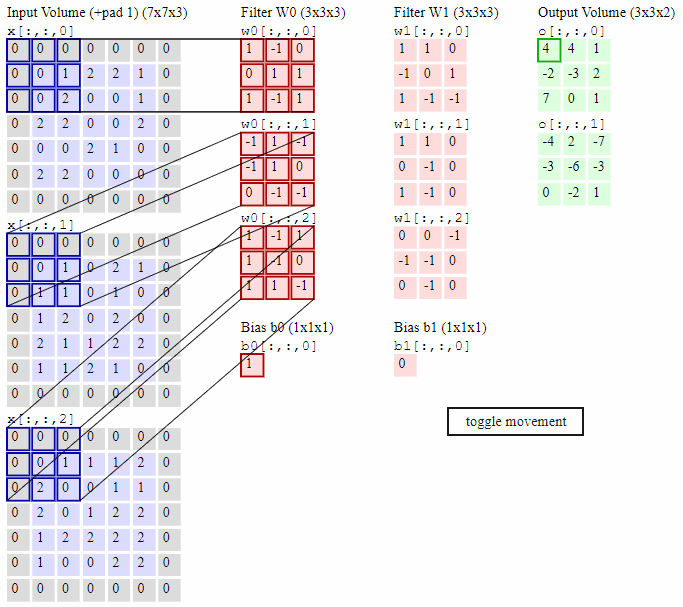
feature\_map尺寸计算公式：[ (原图片尺寸 -卷积核尺寸)/ 步长 ] + 1

如果滑动步幅大于 1, 则卷积核有可能无法恰好滑到边缘, 针对这种情况, 可在矩阵最外层补零, 补一层零后的矩阵如下图所示:



可根据需要设定补零的层数. 补零层称为 **Zero Padding**, 是一个可以设置的超参数, 但要根据卷积核的大小, 步幅, 输入矩阵的大小进行调整, 以使得卷积核恰好滑动到边缘.

上图是对一个特征图采用一个卷积核卷积的过程, 为了提取更多的特征, 可以采用多个卷积核分别进行卷积, 这样便可以得到多个特征图. 有时, 对于一张三通道彩色图片, 或者如第三层特征图所示, 输入的是一组矩阵, 这时卷积核也不再是一层的, 而要变成相应的深度.

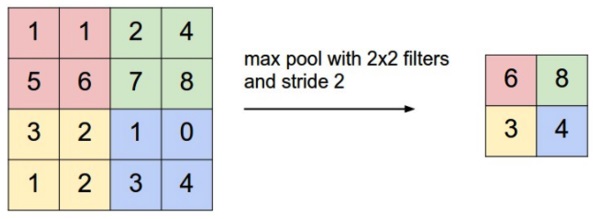


<https://github.com/vdumoulin/conv_arithmetic> 更多动态图

上图中, 最左边是输入的特征图矩阵, 深度为 3, 补零(Zero Padding)层数为 1, 每次滑动的步幅为 2. 中间两列粉色的矩阵分别是两组卷积核, 一组有三个, 三个矩阵分别对应着卷积左侧三个输入矩阵, 每一次滑动卷积会得到三个数, 这三个数的和作为卷积的输出. 最右侧两个绿色的矩阵分别是两组卷积核得到的特征图.

**池化**

池化又叫下采样(Dwon sampling), 与之相对的是上采样(Up sampling). 卷积得到的特征图一般需要一个池化层以降低数据量. 池化的操作如下图所示:



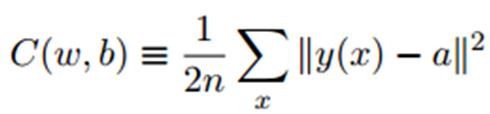
和卷积一样, 池化也有一个滑动的核, 可以称之为滑动窗口, 上图中滑动窗口的大小为 2×2, 步幅为 2, 每滑动到一个区域, 则取最大值作为输出, 这样的操作称为 **Max Pooling**. 还可以采用输出均值的方式, 称为 **Mean Pooling**.

**全连接**

经过若干层的卷积, 池化操作后, 将得到的特征图依次按行展开, 连接成向量, 输入全连接网络.

**深层神经网络的反向传播过程**

首先，为了量化预测结果的好坏。我们使用损失函数这样一个评价指标，来衡量预测结果与真实标签值之间的误差情况。

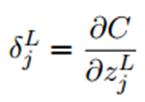


这里给出的是范数形式的损失函数，损失函数当然还可以有其他形式，例如交叉熵形式的等等。但损失函数的自变量都是网络结构中的参数，也就是说只与网络结构中的参数有关。

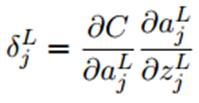
例如在这里w表示深层神经网络中所有权重参数的集合，b是每一层神经网络中的偏差，n是样本数量，x是神经网络的输入量，y是预测值，a是标签。我们用 \textstyle z^{(l)}_i 表示第 \textstyle l 层第 \textstyle i 单元输入加权和（包括偏置单元），比如， \textstyle  z_i^{(2)} = \sum_{j=1}^n W^{(1)}_{ij} x_j + b^{(1)}_i ，我们用 \textstyle a^{(l)}_i 表示第 \textstyle l 层第 \textstyle i 单元的激活值（输出值），则。

反向传播过程的推导与偏导数有关，并且都是基于链式法则进行的，接下来进行梯度下降法的推导：

1.首先是误差函数C关于输出层的L中每一个元素的偏导数：

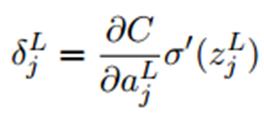


根据链式法则：

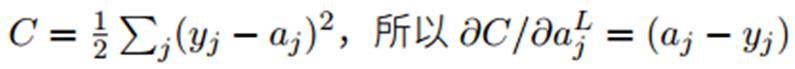


由于在前向传播过程中，激活函数https://private.codecogs.com/gif.latex?y%3D%5Csigma%20%28z%29

所以可以简化成：



上式左边项为损失函数C关于输出层L激活值也就是预测值的偏导数。例如，如果损失函数C是二次项的形式，那么关于预测值的偏导数的值：

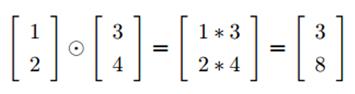


实际过程中这一项都是由损失函数决定的。

用矩阵形式来表示输出层L所有元素的偏导数：

https://img-blog.csdn.net/20170805223807123

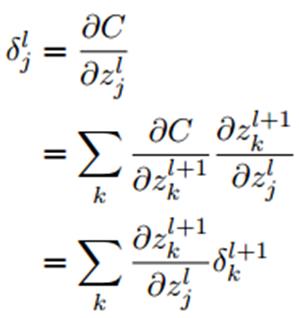
上式第一项是损失函数C关于预测值的梯度向量，中间这个运算符叫做哈达玛（Hadamard）乘积，用于矩阵或向量之间点对点的乘法运算：



第二项是激活函数关于输出层各元素导数构成的向量

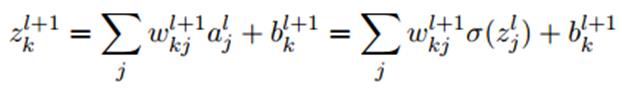
2.中间层各层的偏导数：

第l层第j个元素的偏导数：



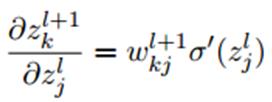
由于第l+1层中的k个输出值{z_{k}}^{l+1}都含有{z_{j}}^{l}这一项，所以在第二行展开过程中根据函数求导法则，需要对{z_{k}}^{l+1}的k个偏导数进行求和运算。

又由于：

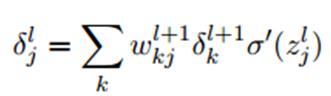


求中{w_{kj}}^{l+1}代表第l+1层参数矩阵w的第k行第j列的元素

求导后得到：



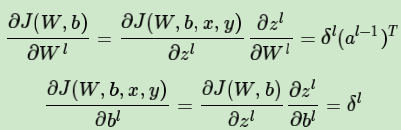
代入得到：



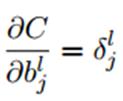
写成向量形式：

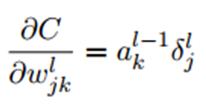
https://img-blog.csdn.net/20170806145343453

3.参数w及偏置b的偏导数：

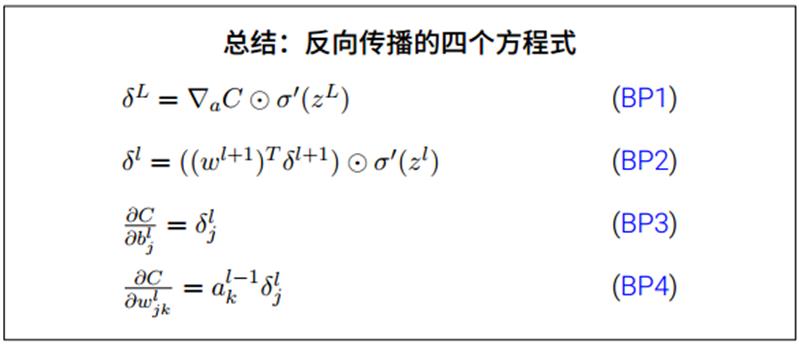


写成单个元素的形式：





4.总结



**卷积神经网络的反向传播过程**

由于CNN的运算过程与DNN不一样，所以CNN的反向传播过程也有所不同。

首先简单介绍一下CNN的计算过程：

（1）卷积运算过程

之后的反向传播的推导也需要用到这个图，记为图1

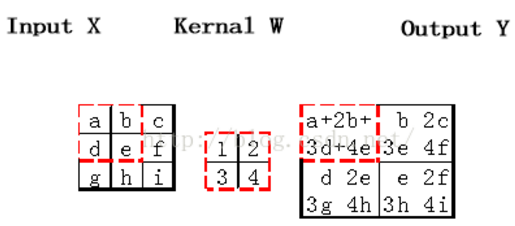
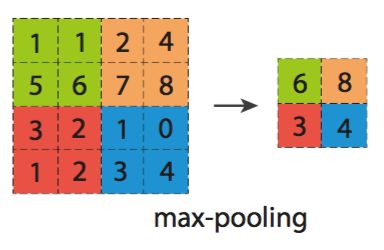


图1描述的是CNN卷积核进行卷积的过程，卷积核与输入矩阵对应位置求积再求和，作为输出矩阵对应位置的值。如果输入矩阵inputX为M\*N大小，卷积核为a\*b大小，那么输出Y为（M-a+1）\*（N-b+1）大小，这里假设步长为1。

（2）池化运算过程



上图所示的是最大池化，类似于卷积核在图片上的移动，这里是不断取核中4个值最大的那个值最为输出。平均池化与之类似，只不过输出的是四个位置的平均值。

（3）全连接层网络

全连接层的网络计算与之前DNN中的计算一样

**CNN与DNN的不同**

1.池化层在前向传播的时候，对输入进行了压缩，那么我们现在需要向前反向推导\delta ^{l-1}，这个推导方法和DNN完全不同。

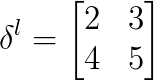
2.卷积层是通过张量卷积，或者说若干个矩阵卷积求和而得的当前层的输出，这和DNN很不相同，DNN的全连接层是直接进行矩阵乘法得到当前层的输出。这样在卷积层反向传播的时候，上一层的\delta ^{l-1}递推计算方法肯定有所不同。

3.对于卷积层，由于W使用的运算是卷积，那么从\delta ^{l}推导出该层的所有卷积核的W,b的梯度方式也不同。

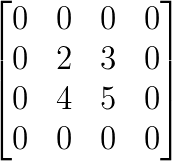
对于这三个问题，我们一个一个来解决

**已知池化层的\delta ^{l}推导上一层的\delta ^{l-1}，**这个过程一般称为upsample

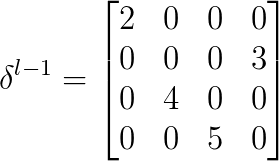
假设池化的size为2\*2，\delta ^{l}：



由于池化size为2\*2，首先将size还原：

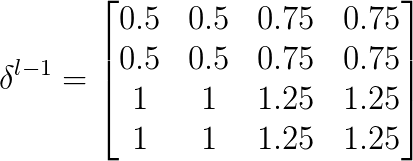


假设是最大池化，并且之前记录的最大值的位置为左上，右下，右上，左下。那么https://private.codecogs.com/gif.latex?%5Cdpi%7B100%7D%20%5Cdelta%20%5E%7Bl-1%7D:



解释下为什么要这么做，在正向传播的时候，池化之前的四个最大值位置左上，右下，右上，左下，都以比例为1的系数传递到下一层。而其他位置对输出的贡献都为0，也就是说对池化输出没有影响，因此比例系数可以理解为0。所以在正向传播的过程中，最大值所在位置可以理解为通过函数f(x)=x传递到下一层，而其他位置则通过f(x)=0传递到下一层，并且把这些值相加构成下一层的输出，虽然f(x)=0并没有作用，但这样也就不难理解反向传播时，把的各个值移到最大值所在位置，而其他位置为0了。因为由f(x)=x，最大值位置的偏导数为1，而f(x)=0的偏导数为0。

如果平均池化，那么https://private.codecogs.com/gif.latex?%5Cdpi%7B100%7D%20%5Cdelta%20%5E%7Bl-1%7D：



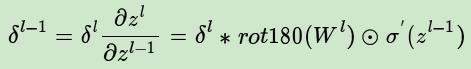
平均池化的话，池化操作的四个位置传递到下一层的作用可以等价为f(x)=x/4,所以在方向传播过程中就相当于把\delta ^{l}每一个位置的值乘1/4再还原回去。

所以由\delta ^{l}推导\delta ^{l-1}可以总结为：

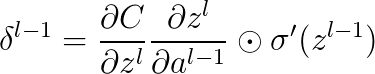
https://img-blog.csdn.net/20170826144238437

等式右边第一项表示上采样，第二项是激活函数的导数，在池化中可以理解为常数1（因为池化过程的正向传播过程中没有激活函数）。

**已知卷积层的**\delta ^{l}**推导上一层的**\delta ^{l-1}：



首先由链式法则：

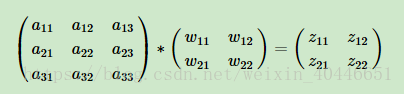


rot180({w^{l}})代表对卷积核进行翻转180°的操作，{\sigma }'(z^{l-1})为激活函数的导数。这里比较难理解的是为什么要对卷积核进行180°的翻转。

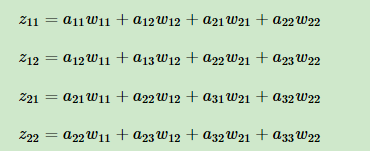
假设我们l-1层的输出a^{l-1}是一个3x3矩阵，第l层的卷积核W是一个2x2矩阵，采用1像素的步幅，则输出z^{l}是一个2x2的矩阵。这里暂时不考虑偏置项b的影响。

那么可得：

https://img-blog.csdn.net/20180809161136597?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3dlaXhpbl80MDQ0NjY1MQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70



展开：



求的梯度：

https://img-blog.csdn.net/20180809161338886?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3dlaXhpbl80MDQ0NjY1MQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

又由展开式：

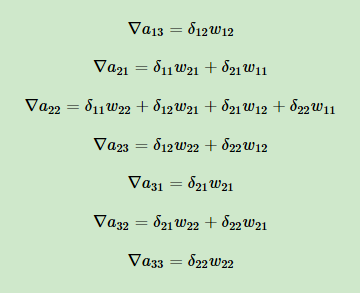
^{a_{11}}只与^{z_{11}}有关，并且系数为^{w_{11}}，所以：

https://img-blog.csdn.net/20180809162009377?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3dlaXhpbl80MDQ0NjY1MQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

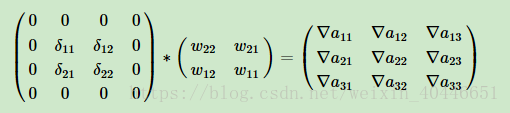
^{a_{12}}只与^{z_{11}}和^{z_{12}}有关，并且系数分别为^{w_{11}}，^{w_{12}}所以：

https://img-blog.csdn.net/20180809162328246?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3dlaXhpbl80MDQ0NjY1MQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

同理：



使用矩阵形式表示就是：



这就解释了为什么在反向传播时需要将卷积核进行180°的翻转操作了。

**已知卷积层的https://private.codecogs.com/gif.latex?%5Cdpi%7B100%7D%20%5Cdelta%20%5E%7Bl%7D推导w,b的梯度：**

全连接层中的w,b的梯度与DNN中的推导一致，池化层没有w,b参数，所以不用进行w,b梯度的推导。

对于卷积层正向传播过程：

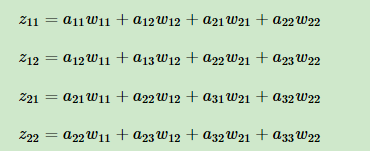
https://img-blog.csdn.net/20180809163156228?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3dlaXhpbl80MDQ0NjY1MQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

所以参数w的梯度：

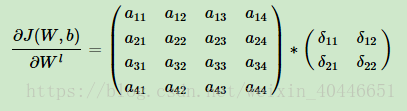
https://img-blog.csdn.net/20180809163221729?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3dlaXhpbl80MDQ0NjY1MQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

注意到这里并没有翻转180°的操作：

因为由之前的展开式：



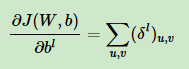
所以w的梯度：



这也就是没有进行翻转的原因。

b的梯度：

这里假设w=0，那么z=b，梯度\delta ^{^{l}}是三维张量，而b只是一个向量，不能像普通网络中那样直接和\delta ^{^{l}}相等。通常的做法是将误差δ的各个子矩阵的项分别求和，得到一个误差向量所以这里b的梯度就是\delta ^{^{l}}的各个通道对应位置求和：



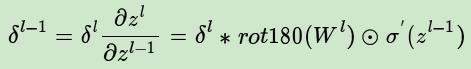
得到的是一个误差向量。

**总结一下CNN的反向传播过程：**

1 池化层的反向传播：

https://img-blog.csdn.net/20170826144238437

2 卷积层的反向传播



3 参数更新

https://img-blog.csdn.net/20180809163221729?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3dlaXhpbl80MDQ0NjY1MQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

