问题

CNN网络在反向传播中需要逐层向前求梯度，然而**pooling层没有可学习的参数**，那它是如何进行反向传播的呢？

此外，CNN中为什么要加pooling层，它的作用是什么？

Pooling层

CNN一般采用average pooling或max pooling来进行池化操作，而池化操作会改变feature map的大小，例如大小为64×64的feature map使用2×2的步长池化后，feature map大小为32×32。因此，这会使得在反向传播中，pooling层的梯度无法与前一层相对应。

那怎么解决这个问题呢？其实也很简单，可以理解为就是pooling操作的一个逆过程，把一个像素的梯度传递给4个像素，保证传递的loss（或梯度）总和不变。下面分别来看average pooling和max pooling的反向传播操作过程。

average pooling

average pooling在前向传播中，就是把一个patch中的值取平均传递给下一层的一个像素。因此，**在反向传播中，就是把某个像素的值平均分成 n 份分配给上一层**。（！！注意这里是分成 n 份，而不是将该元素的值复制 n 份，不然会使得loss之和变为原来的 n 倍，造成梯度爆炸。）

max pooling

max pooling在前向传播中，把一个patch中最大的值传递给下一层，其他值会被舍弃掉。因此，**在反向传播中，就是将当前梯度直接传递给前一层的某个像素，而让同一个patch中的其他像素值为0**。

所以，max pooling和average pooling不同的是，**max pooling在前向传播的时候要记录池化操作时哪个像素的值是最大的**，即max_id，在反向传播中才能将其对应起来。

> 总结： pooling层没有可学习的参数，在CNN的反向传播中，pooling层需要做的仅仅是将误差传递到上一 层，而没有计算梯度的过程。

Pooling层的作用

两种pooling层的原理其实很容易就理解了，那它的作用又是什么呢， CNN中为什么要加pooling层？下面汇总一下几位大佬的解释：

1、**增加非线性**

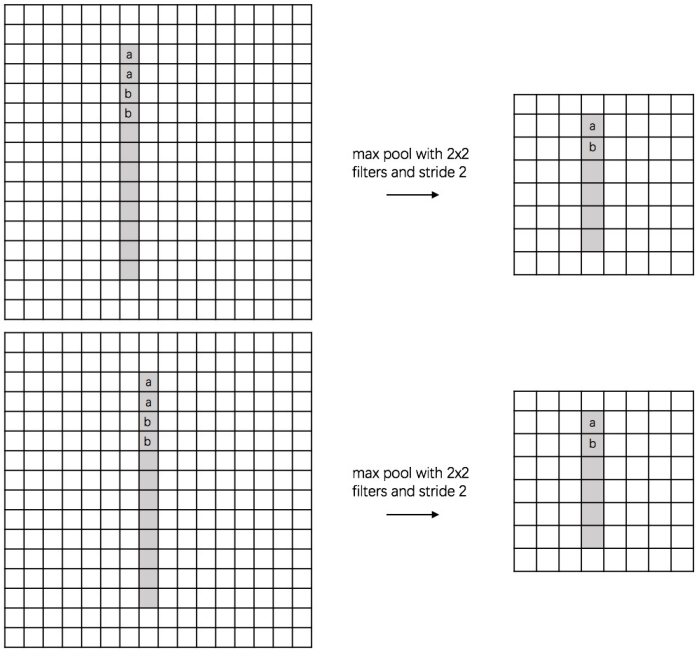
2、**保留主要的特征同时减少参数(降维，效果类似PCA)和计算量，防止过拟合，提高模型泛化能力**

3、**invariance(不变性)，这种不变性包括translation(平移)，rotation(旋转)，scale(尺度)**

①translation invariance（平移不变性）：

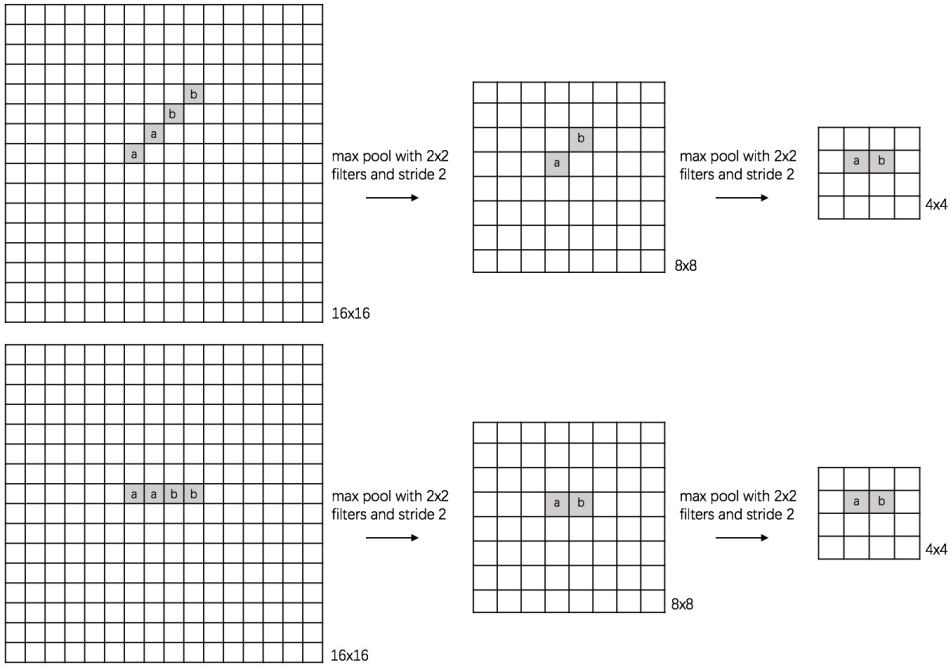
例如下面一个数字识别的例子，左边下图（大小为16×16）中的数字 1 比上图中的向右偏了一个单位，但是经过max pooling层之后，都变成了8×8的feature map。**平移不变性体现在，max pooling之后，原图中的a(或b)最终都会映射到相同的位置**（这句话的应该可以理解为原来feature map中的特征保持不变？比如a和b的位置不会错开，而是保持了相对位置从而保持了原来的主要特征）。

此外，图像主要的特征捕获到了，同时又将问题的规模从16×16降到了8×8（降维）。



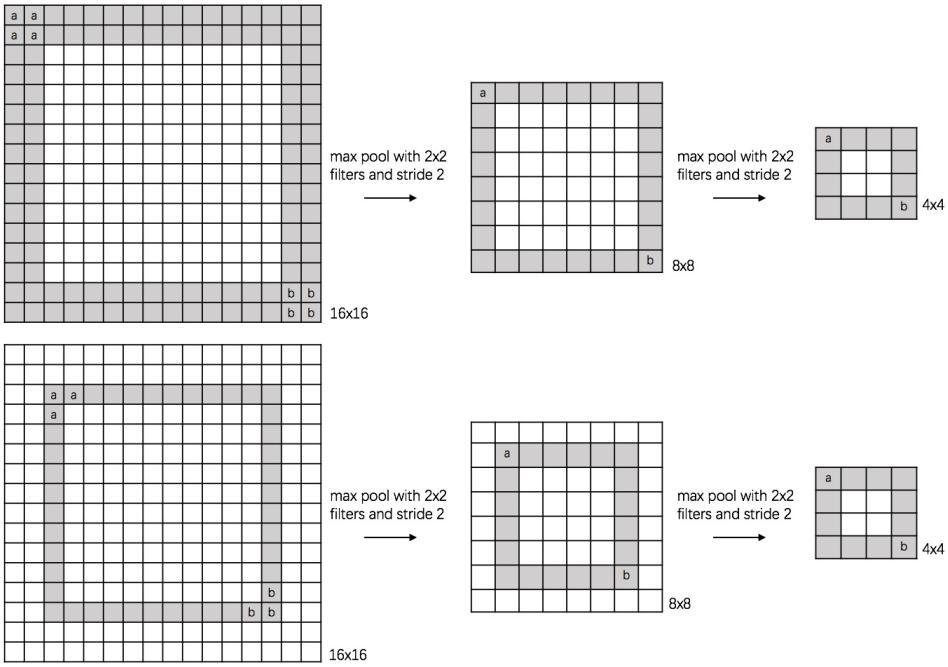
②rotation invariance（旋转不变性）：

下图表示汉字“一”的识别，第一张相对于x轴有倾斜角，第二张是平行于x轴，两张图片相当于做了旋转，经过多次max pooling后具有相同的特征。



③scale invariance（尺度不变性）：

下图表示数字“0”的识别，第一张的“0”比较大，第二张的“0”进行了较小，相当于作了缩放，同样地，经过多次max pooling后具有相同的特征。



对③scale invariance（尺度不变性）的补充理解：（来自另一位大佬，作为参考）

增大了感受野！！！ 怎么理解？比如上图中16×16的“0”，经过max pooling之后，可以用4×4的图表示了。

另外我们知道，CNN中利用卷积核进行卷积操作后，图像的的感受野会增大，那是不是一开始就用和图像大小一样的卷积核，获得的感受野更大，这样就更好呢？不是。因为卷积层越深模型的表征能力越强，如果直接用图像大小的卷积核就会得到1×1的feature map，一下子降维这么多，会导致很多重要信息丢失。

那如果多次卷积到最后也是要降维到1×1大小，信息不是一样丢失了吗？跟直接一次降维到1×1有什么区别吗？有区别的。因为如果每次只降维一些，逐渐降维虽然信息每次都会丢失一些，但每次卷积后表征的能力就会更强一些，到最后降到1×1的时候相比于直接降到1×1还是会强一些的。

pooling的缺点：

pooling能够增大感受野，让卷积能看到更多的信息，但是在降维的过程中也会丢失一部分信息（只留下了它认为重要的信息）。比如对segmentation要求的精度location会有一定的影响。

其他的pooling方法

overlapping pooling（重叠池化）

重叠池化，就是相邻池化窗口之间会有重叠，即窗口大小大于步长sizeX>stride。

Spatial Pyramid Pooling（空间金字塔池化）

空间金字塔池化的思想来源于SPPNet，用大小不同的池化窗口来作用于feature map，得到1×1、2×2和4×4的池化结果，如下图所见，假设卷积层有256个filter，那么可以得到1个256维的特征、4个256维的特征、16个256维的特征。

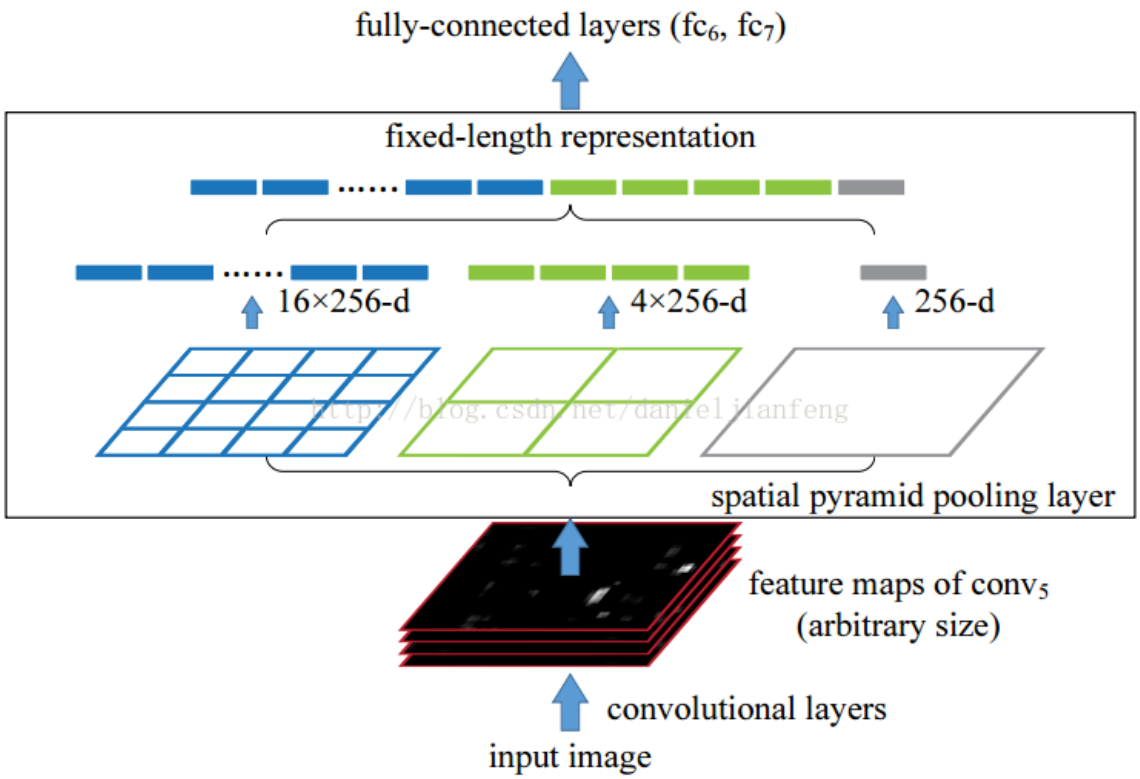
注意：这里的1×1、2×2和4×4不是池化窗口本身的大小，而是池化后将feature map分别划分为1×1、2×2和4×4个相同大小的子区域，而要得到这样的结果，就**需要根据图像的大小动态地计算出池化窗口的大小和步长**。

计算方法：假设$conv$层输出为$a*a$，要得到$n*n$的池化结果，则有：

$$sizeX = \frac{a}{n},\ stride = \frac{a}{n} \tag{1}$$

> 若 $\frac{a}{n}$ 刚好取得整数，自然没有问题，例如假设a=13，要得到1×1pooling结果，只需令sizeX=13，stride=13即可。
>
> 但是当 $\frac{a}{n}$ 不能取整时，例如要得到2×2pooling结果，论文中给的sizeX=7，stride=6。（应该是对窗口大小sizeX稍作调整吧，然后采用重叠池化overlapping pooling的方法进行操作）

作用：CNN中加入SPP层之后，可以让CNN处理任意大小的输入，因而模型可以变得更加灵活。



参考资料

深度学习笔记（3）——CNN中一些特殊环节的反向传播

CNN网络的pooling层有什么用？

深度学习---之pooling层的作用与缺陷

池化方法总结