目录

[1.数据输入层 3](#_Toc44755586)

[2.卷积计算层 5](#_Toc44755587)

[3.激励层 9](#_Toc44755588)

[4.池化层 10](#_Toc44755589)

[5.全连接层 11](#_Toc44755590)

[卷积神经网络之训练算法 12](#_Toc44755591)

[卷积神经网络之优缺点 12](#_Toc44755592)

[卷积神经网络之典型CNN 12](#_Toc44755593)

[卷积神经网络之 fine-tuning 13](#_Toc44755594)

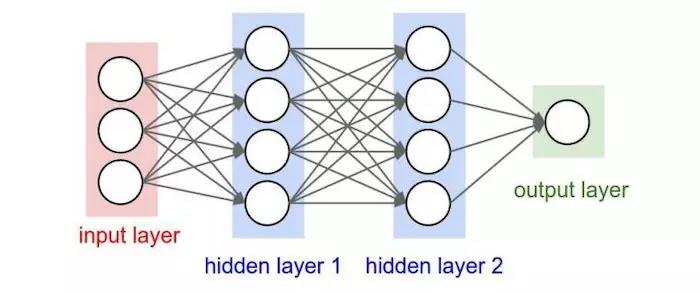
[卷积神经网络的常用框架 13](#_Toc44755595)

[总结 13](#_Toc44755596)

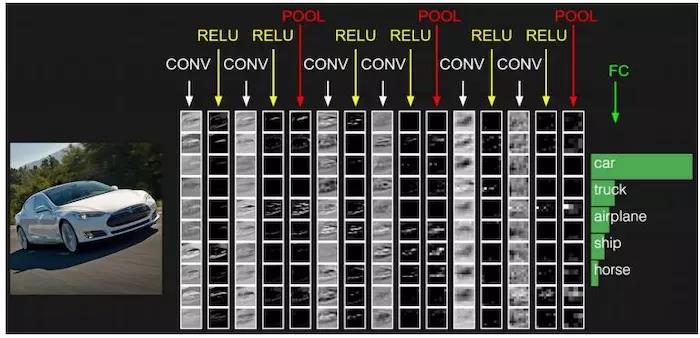
[nn.conv1d和nn.conv2d 14](#_Toc44755597)

**从神经网络到卷积神经网络（CNN）**

我们知道神经网络的结构是这样的：



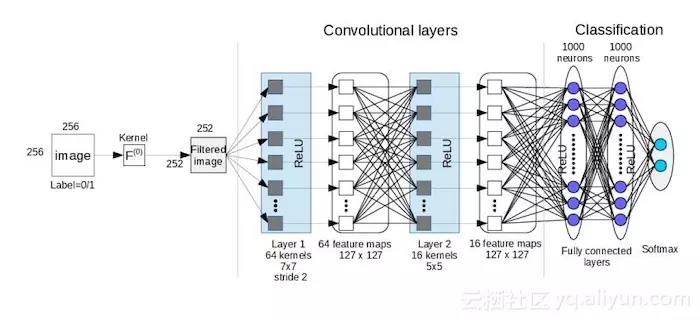
那卷积神经网络跟它是什么关系呢？  
其实卷积神经网络依旧是层级网络，只是**层的功能和形式做了变化**，可以说是传统神经网络的一个改进。比如下图中就多了许多传统神经网络没有的层次。



**定义**

简而言之，[卷积神经网络（Convolutional Neural Networks）](https://en.wikipedia.org/wiki/Convolutional_neural_network)是一种深度学习模型或类似于人工神经网络的多层感知器，常用来分析视觉图像。

**卷积神经网络的架构**

  
如上图所示，卷积神经网络架构与常规人工神经网络架构非常相似，特别是在网络的最后一层，即全连接。此外，还注意到卷积神经网络能够**接受多个特征图**作为输入，而不是向量。

**卷积神经网络的层级结构**

* 数据输入层/ Input layer
* 卷积计算层/ CONV layer
* ReLU激励层 / ReLU layer
* 池化层 / Pooling layer
* 全连接层 / FC layer

1.数据输入层

该层要做的处理主要是对原始图像数据进行预处理，其中包括：

* 去均值：把输入数据各个维度都中心化为0，如下图所示，其目的就是把样本的中心拉回到坐标系原点上。
* 归一化：幅度归一化到同样的范围，如下所示，即减少各维度数据取值范围的差异而带来的干扰，比如，我们有两个维度的特征A和B，A范围是0到10，而B范围是0到10000，如果直接使用这两个特征是有问题的，好的做法就是归一化，即A和B的数据都变为0到1的范围。
* PCA/白化：用PCA降维；白化是对数据各个特征轴上的幅度归一化

PCA详情：PCA的具有2个功能,一是维数约简（可以加快算法的训练速度，减小内存消耗等），一是数据的可视化。

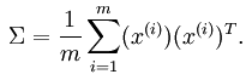
PCA并不是线性回归，因为线性回归是保证得到的函数是y值方面误差最小，而PCA是保证得到的函数到所降的维度上的误差最小。另外线性回归是通过x值来预测y值，而PCA中是将所有的x样本都同等对待。

在使用PCA前需要对数据进行预处理，首先是均值化，即对每个特征维，都减掉该维的平均值，然后就是将不同维的数据范围归一化到同一范围，方法一般都是除以最大值。但是比较奇怪的是，在对自然图像进行均值处理时并不是减去该维的平均值，而是减去这张图片本身的平均值。因为PCA的预处理是按照不同应用场合来定的。

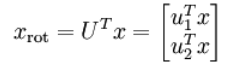
自然图像指的是人眼经常看见的图像，其符合某些统计特征。一般实际过程中，只要是拿正常相机拍的，没有加入很多人工创作进去的图片都可以叫做是自然图片，因为很多算法对这些图片的输入类型还是比较鲁棒的。在对自然图像进行学习时，其实不需要太关注对图像做方差归一化，因为自然图像每一部分的统计特征都相似，只需做均值为0化就ok了。不过对其它的图片进行训练时，比如手写字识别等，就需要进行方差归一化了。

PCA的计算过程主要是要求2个东西，一个是降维后的各个向量的方向，另一个是原先的样本在新的方向上投影后的值。

首先需求出训练样本的协方差矩阵，如公式所示（输入数据已经均值化过）：



求出训练样本的协方差矩阵后，将其进行SVD分解，得出的U向量中的每一列就是这些数据样本的新的方向向量了，排在前面的向量代表的是主方向，依次类推。用得到的就是降维后的样本值z了，即：



（其实这个z值的几何意义是原先点到该方向上的距离值，但是这个距离有正负之分），这样PCA的2个主要计算任务已经完成了。用U\*z就可以将原先的数据样本x给还原出来。

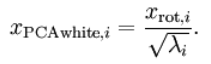
在使用有监督学习时，如果要采用PCA降维，那么只需将训练样本的x值抽取出来，计算出主成分矩阵U以及降维后的值z，然后让z和原先样本的y值组合构成新的训练样本来训练分类器。在测试过程中，同样可以用原先的U来对新的测试样本降维，然后输入到训练好的分类器中即可。

有一个观点需要注意，那就是PCA并不能阻止过拟合现象。表明上看PCA是降维了，因为在同样多的训练样本数据下，其特征数变少了，应该是更不容易产生过拟合现象。但是在实际操作过程中，这个方法阻止过拟合现象效果很小，主要还是通过规则项来进行阻止过拟合的。

并不是所有ML算法场合都需要使用PCA来降维，因为只有当原始的训练样本不能满足我们所需要的情况下才使用，比如说模型的训练速度，内存大小，希望可视化等。如果不需要考虑那些情况，则也不一定需要使用PCA算法了。

Whitening详情：白化的目的是去掉数据之间的相关联度，是很多算法进行预处理的步骤。比如说当训练图片数据时，由于图片中相邻像素值有一定的关联，所以很多信息是冗余的。这时候去相关的操作就可以采用白化操作。数据的whitening必须满足两个条件：一是不同特征间相关性最小，接近0；二是所有特征的方差相等（不一定为1）。常见的白化操作有PCA whitening和ZCA whitening。

PCA whitening是指将数据x经过PCA降维为z后，可以看出z中每一维是独立的，满足whitening白化的第一个条件，这是只需要将z中的每一维都除以标准差就得到了每一维的方差为1，也就是说方差相等。公式为：

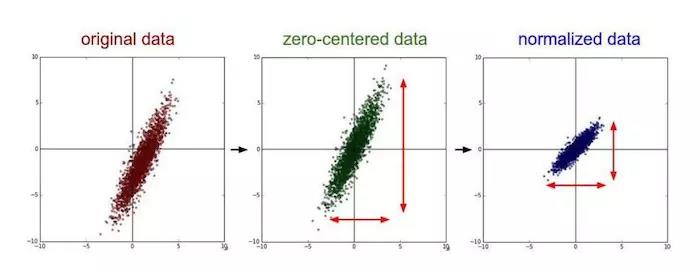


ZCA whitening是指数据x先经过PCA变换为z，但是并不降维，因为这里是把所有的成分都选进去了。这是也同样满足whtienning的第一个条件，特征间相互独立。然后同样进行方差为1的操作，最后将得到的矩阵左乘一个特征向量矩阵U即可。

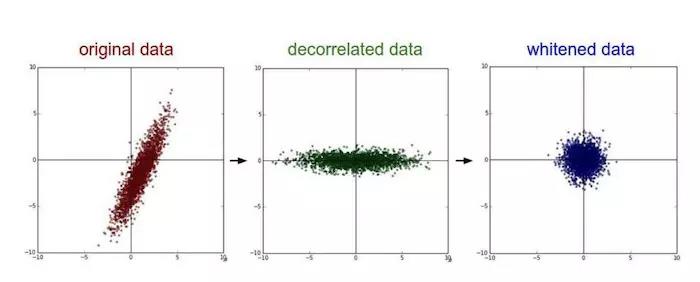
ZCA whitening公式为：



去均值与归一化效果图：



去相关与白化效果图：



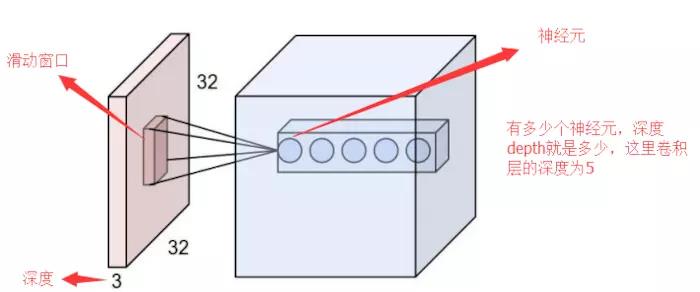
2.卷积计算层

这一层就是卷积神经网络最重要的一个层次，也是“卷积神经网络”的名字来源。  
在这个卷积层，有两个关键操作：

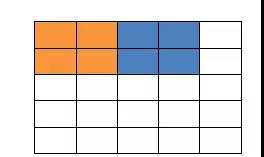
* 局部关联。每个神经元看做一个滤波器(filter)（滤波器即卷积核Kernel，各个卷积核提取一种特征）
* 窗口(receptive field)滑动， filter对局部数据计算

先介绍卷积层遇到的几个名词：

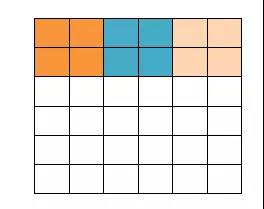
* 深度/depth（解释见下图）（一个神经元就代表一个卷积核/一个滤波器，提取一种特征，所以有多少神经元，深度就有多少层）
* 步长/stride （窗口一次滑动的长度）
* 填充值/zero-padding



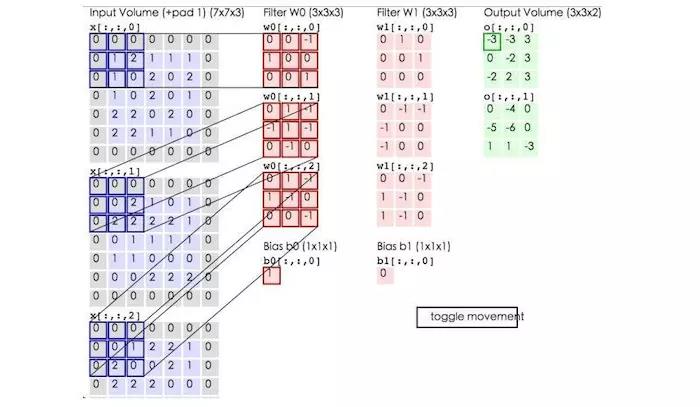
填充值是什么呢？以下图为例子，比如有这么一个5 \* 5的图片（一个格子一个像素），我们滑动窗口取2\*2，步长取2，那么我们发现还剩下1个像素没法滑完，那怎么办呢？

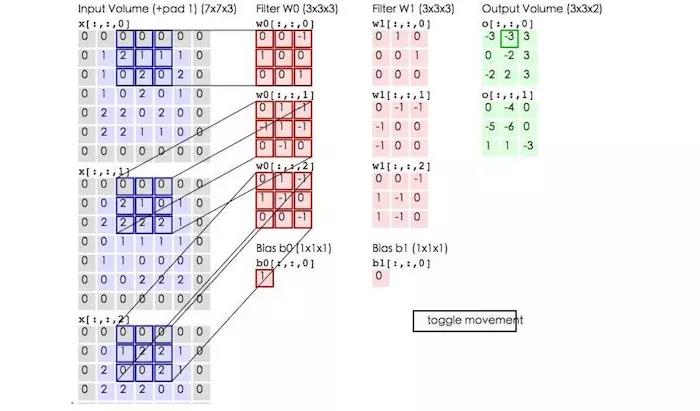


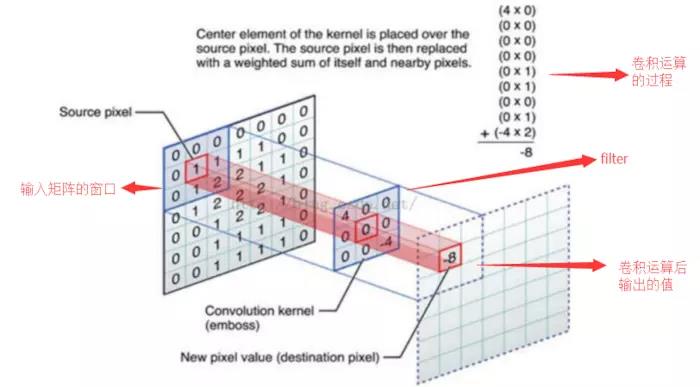
那我们在原先的矩阵加了一层填充值，使得变成6\*6的矩阵，那么窗口就可以刚好把所有像素遍历完。这就是填充值的作用。



卷积的计算（注意，下面蓝色矩阵周围有一圈灰色的框，那些就是上面所说到的填充值）

  
这里的蓝色矩阵就是输入的图像，粉色矩阵就是**卷积层的神经元**，这里表示了有两个神经元（w0,w1）。**绿色矩阵就是经过卷积运算后的输出矩阵**，这里的步长设置为2。



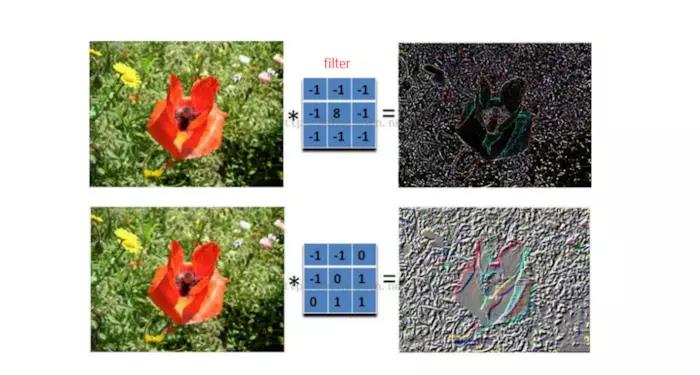
蓝色的矩阵(输入图像)对粉色的矩阵（filter）进行矩阵内积计算并将三个内积运算的结果与偏置值b相加（比如上面图的计算：2+（-2+1-2）+（1-2-2） + 1= 2 - 3 - 3 + 1 = -3），计算后的值就是绿框矩阵的一个元素。

下面的动态图形象地展示了卷积层的计算过程：

。。。见网页。。。https://www.jianshu.com/p/1ea2949c0056

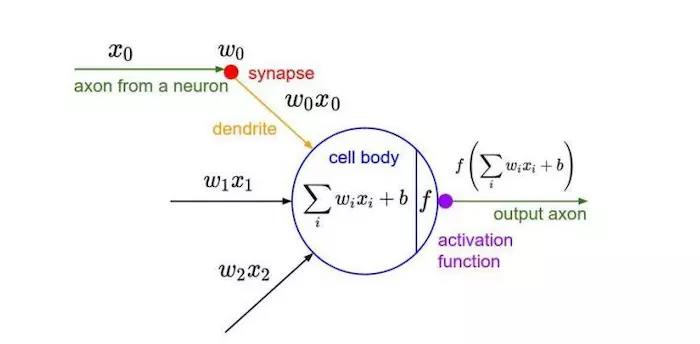
**参数共享机制**

* 在卷积层中每个神经元连接数据窗的权重是固定的，每个神经元只关注一个特性。神经元就是图像处理中的滤波器，比如边缘检测专用的Sobel滤波器，即卷积层的每个滤波器都会有自己所关注的一个图像特征，比如垂直边缘，水平边缘，颜色，纹理等等，所有这些神经元加起来好比就是整张图像的特征提取器集合。
* 需要估算的权重个数减少: AlexNet 1亿 => 3.5w
* 一组固定的权重和不同窗口内数据做内积: 卷积

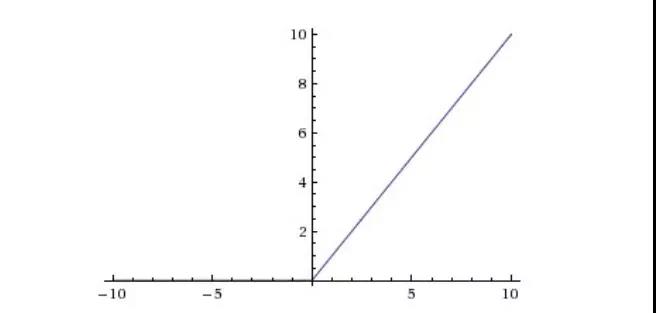


3.激励层

把卷积层输出结果做非线性映射(non-linear mapping,在保持高维空间某些集合结构的前提下将样本点集映射到较低维空间的方法)。



CNN采用的激励函数一般为ReLU(The Rectified Linear Unit/修正线性单元)，它的特点是收敛快，求梯度简单，但较脆弱，图像如下。



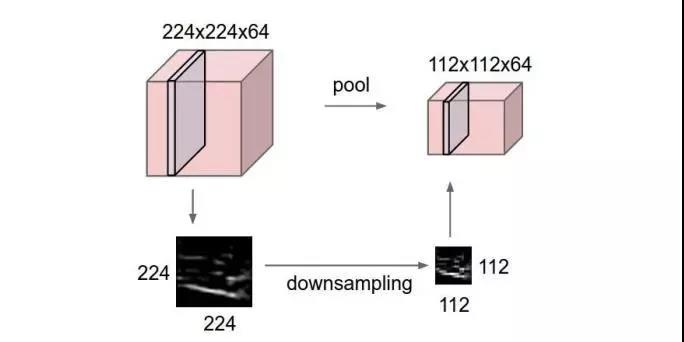
激励层的实践经验：  
①不要用sigmoid！不要用sigmoid！不要用sigmoid！  
② 首先试ReLU，因为快，但要小心点  
③ 如果2失效，请用Leaky ReLU或者Maxout  
④ 某些情况下tanh倒是有不错的结果，但是很少

4.池化层

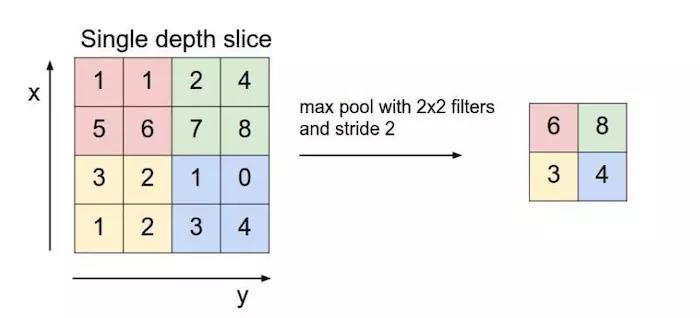
池化层夹在连续的卷积层中间， 用于压缩数据和参数的量，减小过拟合（overfitting，是指为了得到一致假设而使假设变得过度严格，避免过拟合是分类器设计中的一个核心任务，通常采用增大数据量和测试样本集的方法对分类器性能进行评价。判断方法：一个假设在训练数据上能够获得比其他假设更好的拟合， 但是在训练数据外的数据集上却不能很好地拟合数据，此时认为这个假设出现了过拟合的现象。出现这种现象的主要原因是训练数据中存在噪音或者训练数据太少）。  
简而言之，**如果输入是图像的话，那么池化层的最主要作用就是压缩图像**。

这里再展开叙述池化层的具体作用。

1. 特征不变性，也就是我们在图像处理中经常提到的特征的尺度不变性，池化操作就是图像的resize，平时一张狗的图像被缩小了一倍我们还能认出这是一张狗的照片，这说明这张图像中仍保留着狗最重要的特征，我们一看就能判断图像中画的是一只狗，图像压缩时去掉的信息只是一些无关紧要的信息，而留下的信息则是具有尺度不变性的特征，是最能表达图像的特征。（怎么判断哪些是有用信息，难道要依次去除无用信息判断？）
2. 特征降维，我们知道一幅图像含有的信息是很大的，特征也很多，但是有些信息对于我们做图像任务时没有太多用途或者有重复，我们可以把这类冗余信息去除，把最重要的特征抽取出来，这也是池化操作的一大作用。
3. 在一定程度上防止过拟合

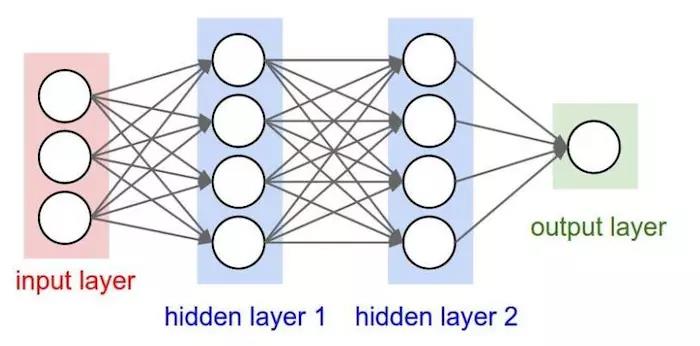


池化层用的方法有Max pooling 和 average pooling，而实际用的较多的是Max pooling。  
这里就说一下Max pooling，其实思想非常简单。

  
对于每个2 \* 2的窗口选出最大的数作为输出矩阵的相应元素的值，比如输入矩阵第一个2 \* 2窗口中最大的数是6，那么输出矩阵的第一个元素就是6，如此类推。

5.全连接层

两层之间所有神经元都有权重连接，通常全连接层在卷积神经网络尾部。也就是跟传统的神经网络神经元的连接方式是一样的：



**一般CNN结构依次为**

1.INPUT  
2.[[CONV -> RELU]*N -> POOL?]*M  
3.[FC -> RELU]\*K  
4.FC

卷积神经网络之训练算法

1.同一般机器学习算法，先定义Loss function，衡量和实际结果之间差距。  
2.找到最小化损失函数的W和b， CNN中用的算法是SGD（随机梯度下降）。

卷积神经网络之优缺点

优点  
　　•共享卷积核，对高维数据处理无压力  
　　•无需手动选取特征，训练好权重，即得特征分类效果好  
缺点  
　　•需要调参，需要大样本量，训练最好要GPU  
　　•物理含义不明确（也就说，我们并不知道没个卷积层到底提取到的是什么特征，而且神经网络本身就是一种难以解释的“黑箱模型”）

卷积神经网络之典型CNN

•LeNet，这是最早用于数字识别的CNN  
•AlexNet， 2012 ILSVRC比赛远超第2名的CNN，比  
•LeNet更深，用多层小卷积层叠加替换单大卷积层。  
•ZF Net， 2013 ILSVRC比赛冠军  
•GoogLeNet， 2014 ILSVRC比赛冠军  
•VGGNet， 2014 ILSVRC比赛中的模型，图像识别略差于GoogLeNet，但是在很多图像转化学习问题(比如object detection)上效果奇好

卷积神经网络之 fine-tuning

何谓fine-tuning？  
fine-tuning就是使用已用于其他目标、预训练好模型的权重或者部分权重，作为初始值开始训练。

那为什么我们不用随机选取选几个数作为权重初始值？原因很简单，第一，自己从头训练卷积神经网络容易出现问题；第二，fine-tuning能很快收敛到一个较理想的状态，省时又省心。

那fine-tuning的具体做法是？  
•复用相同层的权重，新定义层取随机权重初始值  
•调大新定义层的的学习率，调小复用层学习率

卷积神经网络的常用框架

Caffe  
　•源于Berkeley的主流CV工具包，支持C++,python,matlab  
　•Model Zoo中有大量预训练好的模型供使用  
Torch  
　•Facebook用的卷积神经网络工具包  
　•通过时域卷积的本地接口，使用非常直观  
　•定义新网络层简单  
TensorFlow  
　•Google的深度学习框架  
　•TensorBoard可视化很方便  
　•数据和模型并行化好，速度快

总结

卷积网络在本质上是一种输入到输出的映射，它能够学习大量的输入与输出之间的映射关系，而不需要任何输入和输出之间的精确的数学表达式，只要用已知的模式对卷积网络加以训练，网络就具有输入输出对之间的映射能力。

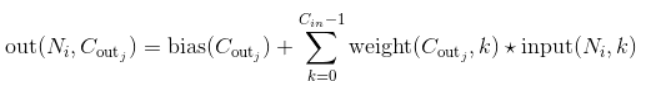
CNN一个非常重要的特点就是头重脚轻（越往输入权值越小，越往输出权值越多），呈现出一个倒三角的形态，这就很好地避免了BP神经网络中反向传播的时候梯度损失得太快。

卷积神经网络CNN主要用来识别位移、缩放及其他形式扭曲不变性的二维图形。由于CNN的特征检测层通过训练数据进行学习，所以在使用CNN时，避免了显式的特征抽取，而隐式地从训练数据中进行学习；再者由于同一特征映射面上的神经元权值相同，所以网络可以并行学习，这也是卷积网络相对于神经元彼此相连网络的一大优势。卷积神经网络以其局部权值共享的特殊结构在语音识别和图像处理方面有着独特的优越性，其布局更接近于实际的生物神经网络，权值共享降低了网络的复杂性，特别是多维输入向量的图像可以直接输入网络这一特点避免了特征提取和分类过程中数据重建的复杂度。

nn.conv1d和nn.conv2d

1.所谓一维卷积，就是卷积时只看纵列

class torch.nn.Conv1d(in\_channels, out\_channels, kernel\_size, stride=1, padding=0, dilation=1, groups=1, bias=True)

内部进行了卷积运算：

符号是互相关([cross-correlation](https://en.wikipedia.org/wiki/Cross-correlation)）操作符因为实现起来很简单，所以一般的深度学习框架都是用互相关操作取代真卷积）, N is a batch size, C 代表通道的数量, L 代表信号序列的长度。

(给定图像和滤波器,一般m<<M,n<<N

互相关Cross-correlation： loop: 从左到右，从上到下

卷积Convolution： loop:从右到左，从下到上

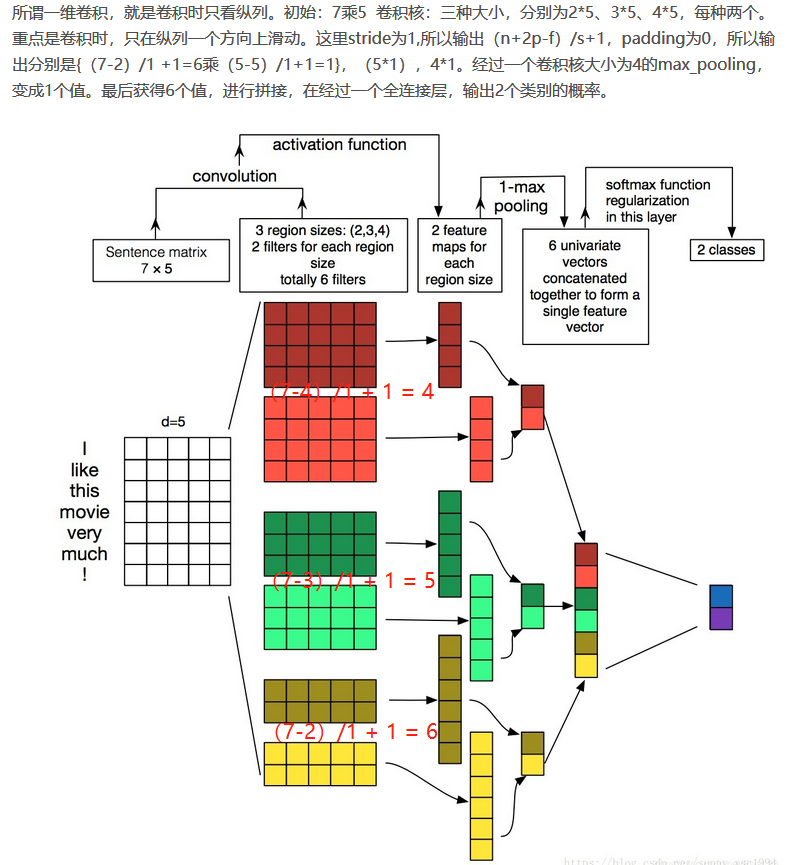
其实两者不必做过分区别，在CNN中卷积核的方向没有特别的规定，当filter关于x轴y轴对称时（包括Average/Gaussion filter）两者是一样的)



注意：互相关只保证正确的操作而不是满操作，所以要设置好padding保证余数列参与计算

理解：

初始：7乘5 卷积核：三种大小，分别为2\*5、3\*5、4\*5（f = 2,3,4），每种两个。重点是卷积时，只在纵列一个方向上滑动。这里stride为1,所以输出（n+2p-f）/s+1，padding为0，输出分别如图。经过一个卷积核大小为4的max\_pooling（选择出一个最大值），变成1个值。最后获得6个值，进行拼接，在经过一个全连接层，输出2个类别的概率。



2.给定一个4维的input和filter tensor，计算一个二维卷积，所谓二维卷积就是横向和纵向均卷积（就是常见的那种矩阵相乘的卷积运算）