P3:卷积层的目的是提取输入的不同的特征，前面的卷积层提取相对简单的特征，如边缘，线条等。随着卷积层数的增加，从低级特征中迭代提取更复杂的特征。具体卷积的算法在此不再赘述，大家可以看斯坦福的教程自行学习

P4:CONV:卷积神经网络的一大优点就是参数少。接触过MLP的同学都知道，MLP的参数特别多，CNN有两大神器，有效的减少了参数数目。局部感知是受生物的视觉系统的启发，像我们的视野也是局部的。这样会大大减少参数的数量，不再像MLP那样，每层网络，每个节点的值都要传给下一层的每个节点。其次是参数共享，采用共用卷积核进行参数共享，也就是说图像的一部分统计特性和其他部分是一样的，这样的话，我们就可以采用同样的卷积核。

P5:RELU。RELU是激活函数，对比sigmoid函数，RELU的梯度更容易求解。其次，sigmoid易出现梯度消失情况，造成信息丢失。RELU会使一部分神经元输出为0，这样网络具有一定的稀疏性，也减小了参数相互依存的关系，缓解过拟合问题。

P6: pool层，即池化层，又被成为下采样层，现有两种池化方式：Max和mean，分别是采用最大值和平均值作为采样值。有图就是Max-pooling的方式。Pool的降维效果很明显，如右图，维度降为原来的1/4，同样也会提高系统的鲁棒性。（鲁棒性是指算法在遭遇输入、运算等异常时继续正常运行的能力）。接下来，我以CNN中经典网络LeNet-5为例，介绍一下我学会的定义网络结构部分。

P7: 编写网络结构的配置文件。Lenet-5的配置文件caffe里有。首先是数据层，数据层是每个网络的最底层，类型为Data，则证明数据来自高效数据库（如LevelDB和LMDB），每一层用top来输出数据，用bottom输入数据。多个top代表多个输出。**include**: 一般训练的时候和测试的时候，模型的层是不一样的。该层（layer）是属于训练阶段的层，还是属于测试阶段的层，需要用include来指定。如果没有include参数，则表示该层既在训练模型中，又在测试模型中。**Transformations**: 数据的预处理，可以将数据变换到定义的范围内。如设置scale为0.00390625，实际上就是1/255, 即将输入数据由0-255归一化到0-1之间。

r\_mult: 学习率的系数，最终的学习率是这个数乘以solver.prototxt配置文件中的base\_lr。如果有两个lr\_mult, 则第一个表示权值的学习率，第二个表示偏置项的学习率。一般偏置项的学习率是权值学习率的两倍。

num\_output: 卷积核（filter）的个数

kernel\_size: 卷积核的大小。如果卷积核的长和宽不等，需要用kernel\_h和kernel\_w分别设定

stride: 卷积核的步长，默认为1。

weight\_filler: 权值初始化。 默认为“constant",值全为0，很多时候我们用"xavier"算法来进行初始化，也可以设置为”gaussian"

 bias\_filler: 偏置项的初始化。一般设置为"constant",值全为0。

全连接层：num\_output: 过滤器（filfter）的个数

accuracy: 输出分类精确度，只有test阶段才有，因此需要加入include参数

Slover: Slover算是caffe的核心的核心，它协调着整个模型的运作。Solver的主要作用就是交替调用前向（forward）[**算法**](http://lib.csdn.net/base/datastructure)和后向（backward）算法来更新参数，从而最小化loss，实际上就是一种迭代的优化算法。Solver配置文件的编写：test\_iter: 这个要与test layer中的batch\_size结合起来理解。mnist数据中测试样本总数为10000，一次性执行全部数据效率很低，因此我们将测试数据分成几个批次来执行，每个批次的数量就是batch\_size。假设我们设置batch\_size为100，则需要迭代100次才能将10000个数据全部执行完。因此test\_iter设置为100。执行完一次全部数据，称之为一个epoch

test\_interval:500 测试间隔。也就是每训练500次，才进行一次测试。

base\_lr: 0.01

lr\_policy: "inv"

gamma: 0.0001

power: 0.75

base\_lr：用于设置基础学习率，在迭代的过程中，可以对基础学习率进行调整。

inv:如果设置为inv,还需要设置一个power, 返回base\_lr \* (1 + gamma \* iter) ^ (- power)

P9:在将网络结构和solver的配置文件都写好之后，训练网络模型就很简单了。执行**./train\_lenet.sh**就可以开始跑这个模型了，然后你就会看到它会给出网络结构的细节，然后根据咱们自己的设置，它会每迭代100次再输出一次loss和lr。然后几分钟后，会输出最后的准确率和损失值。