# Handwriting CNN

写在前面

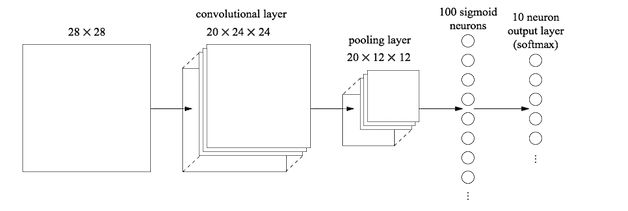
1. 简单理解卷积和池化

卷积就是取特征，而池化就是为了降维。比如max pooling，认为某个window中最大的那个值最能代表该patch的特征，那么就用一个值来代表而不是一个特征矩阵。

1. 简单理解局部连接层和全连接层

知乎下这两个答案个人感觉是最好理解的

*“以前的神经网络都是采取全连接的方式，认为下一层的输出与上一层所有输入都有关，但是这样很容易导致overfitting。利用convolution的局部连接和权值共享，pooling的下采样能够降低overfitting，增加层数，提取提取都具有鉴别力的特征”*

**

作者：晚晴也是晴  
链接：https://www.zhihu.com/question/41037974/answer/110588956  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

*（个人理解：在上图中 100个sigmoid neurons的weight是20\*12\*12\*100，bias是100\*1，最后的输出层weight是100\*10，bias是10\*1）*

*全连接的核心操作就是矩阵向量乘积*

*y=Wx+b*

*本质就是由一个特征空间线性变换到另一个特征空间。目标空间的任一维——也就是隐层的一个 cell——都认为会受到源空间的每一维的影响。不考虑严谨，可以说，目标向量是源向量的加权和。*

*在 CNN 中，全连接常出现在最后几层，用于对前面设计的特征做加权和。比如 mnist，前面的卷积和池化相当于做特征工程，后面的全连接相当于做特征加权。（卷积相当于全连接的有意弱化，按照局部视野的启发，把局部之外的弱影响直接抹为零影响；还做了一点强制，不同的局部所使用的参数居然一致。弱化使参数变少，节省计算量，又专攻局部不贪多求全；强制进一步减少参数。少即是多）*

[作者：张程序员  
链接：https://www.zhihu.com/question/41037974/answer/150546125  
来源：知乎]

CNN部分代码理解：

# 改变输入格式，将原输入784\*1变成是28\*28的二维形式，方便后面做卷积change the format of x to 4D [batch, in\_height, in\_width, in\_channels]

x\_image = tf.reshape(x,[-1,28,28,1])

**卷积和池化**

**#**5\*5窗口，32个卷积核，什么意思？就是相当于5\*5的滑动窗口在图像上做卷积，而这样的窗口有32个

W\_conv1 = weight\_variable([5,5,1,32]) # 5\*5windiw, 32 conv kernel on one channel

#每个特征图像一个bias，所以是32个bias

b\_conv1 = bias\_variable([32])

#完成第一次卷积

h\_conv1 = tf.nn.relu(conv2d(x\_image, W\_conv1) + b\_conv1)

#完成第一次池化，得到14\*14\*32

h\_pool1 = max\_pool\_2x2(h\_conv1)

**全连接层**

#上一层输出是7\*7（每层大小）\*64（层），全连接层有1024个神经元，所有这一层的weight的大小是(7\*7\*64\*1024)。（如果不好理解，想一下之前最简单的神经网络，假设有100个神经元，那么weight的大小是(784\*100)，这里道理是一样，只不过输入不是原始的图像而是经过卷积和池化后的。）

这里唯一和之前简单神经网络不同的就是多了一个扁平化处理h\_pool2\_flat，其实也和简单，就是将2d变成1d而已。（继续回忆之前的简单神经网络）

W\_fc1 = weight\_variable([7\*7\*64,1024])

b\_fc1 = bias\_variable([1024])

h\_pool2\_flat = tf.reshape(h\_pool2, [-1,7\*7\*64])

h\_fc1 = tf.nn.relu(tf.matmul(h\_pool2\_flat, W\_fc1)+b\_fc1)

**出结果啦**

# 神经元的利用率的feel，有时候数据少而网络复杂的时候很容易出现overfitting的情况，这里定义一个利用率，认为只有keep\_prob（~[0,1]）的神经元是实际起作用的

keep\_prob = tf.placeholder(tf.float32)

h\_fc1\_drop = tf.nn.dropout(h\_fc1, keep\_prob)

# 最终分10类（0~9）上层每个样本输出维度1024\*1，所以这里权值size1024\*10，bias 10\*1

W\_fc2 = weight\_variable([1024,10])

b\_fc2 = bias\_variable([10])

prediction = tf.nn.softmax(tf.matmul(h\_fc1\_drop, W\_fc2)+b\_fc2)

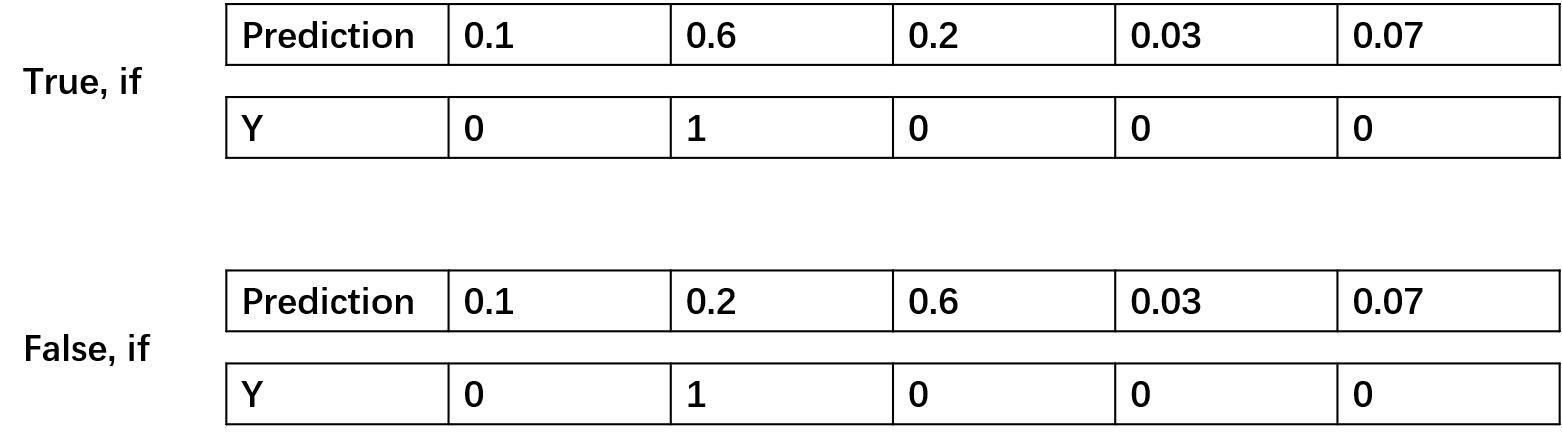
# 在训练过程中求预测值和真实值的差

cross\_entropy = tf.reduce\_mean(tf.nn.softmax\_cross\_entropy\_with\_logits(labels = y, logits = prediction))

# minimize真实值和预测值的差（通过这一步来调整每一层的weight和bias）

train\_step = tf.train.AdamOptimizer(1e-2).minimize(cross\_entropy)

# tf.argmax返回最大值的位置，当prediction最大值位置和真实prediction的位置相同是，证明预测正确，那么correct\_prediction=true



correct\_prediction = tf.equal(tf.argmax(prediction,1), tf.argmax(y,1))

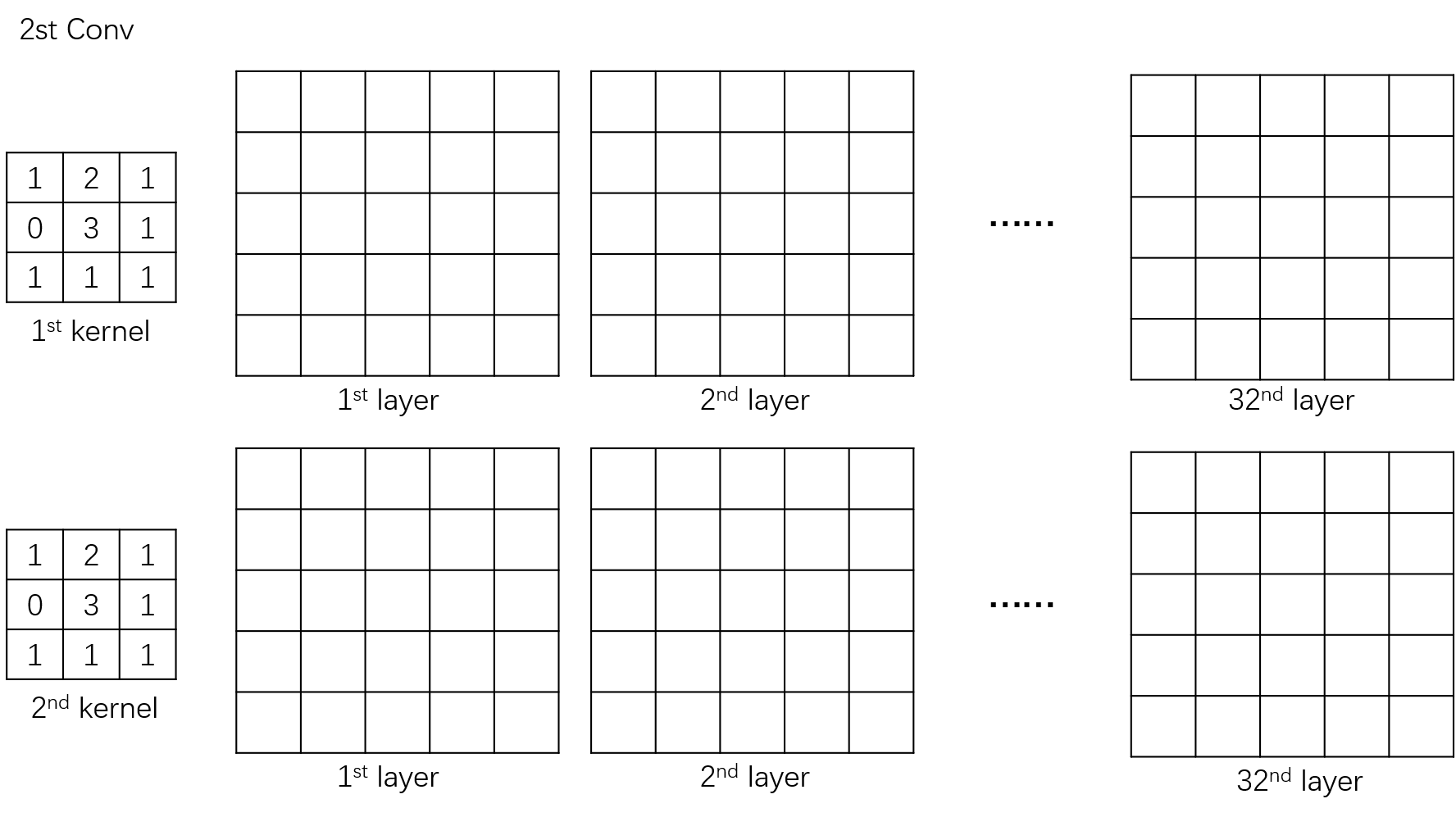
# 把true和false转化为0，1，所有的取平均则为准确率

accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))

常见疑问：

1. 第二次卷积有32个卷积核，卷积后为什么不生成32\*64个特征图而是64个？

在后记（2）中提到了convolution的权值共享，个人理解就是说，对每一层的卷积核是一样的，所以当有64个卷积核的时候，生成的就是64个特征图像。简单画个图：



# RNN

强烈推荐台大[李宏毅](https://www.youtube.com/channel/UC2ggjtuuWvxrHHHiaDH1dlQ)教授的教程，RNN课程

<https://www.youtube.com/watch?v=xCGidAeyS4M&list=LL0EMfBP6yInDg7NLFR2bd8w&index=2&t=0s>

1. 如果你只想把LSTM当黑盒子用，那么简单理解就是：

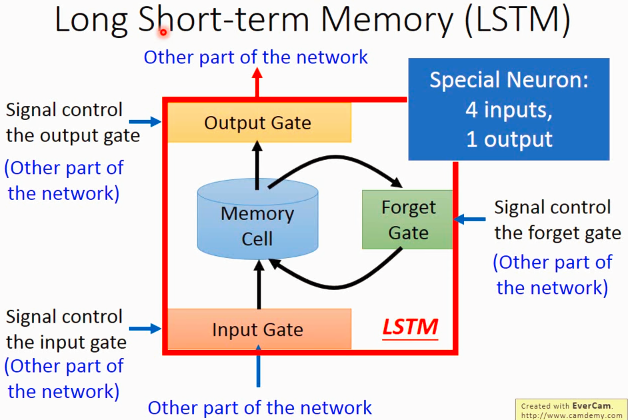
假设输入图像的28行(x1,x2,x3…x28)，有100个LSTM的神经元，那么输出的就是100\*1的向量

1. Why RNN？

在CNN中，每个图像可以单独地处理，但是在分析语音或者文字的时候，单独一个字的分析是没有意义的，需要结合前后进行分析，CNN无法完成，所以RNN

1. What’s RNN?

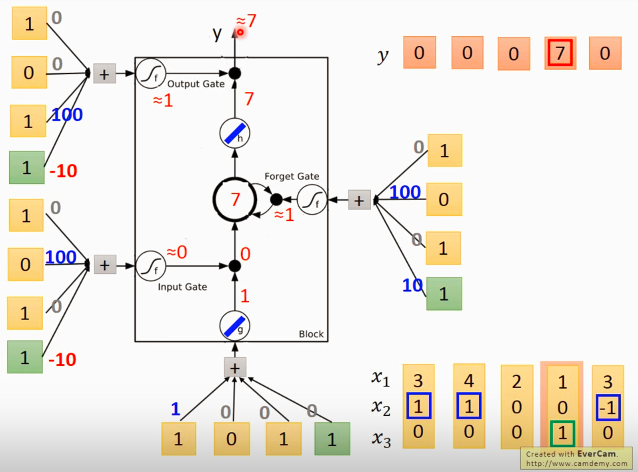
首先什么是LTSM cell



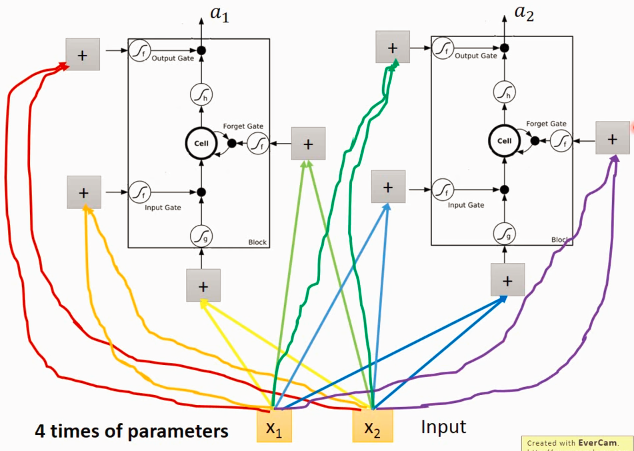
每个神经单元的4个input：

1. input（e.g. 假设原始输入为x，那么这里输入是g(x)）
2. Input gate（所谓gate就是门，只有开关两个状态，在这里表现为0和1,0代表我不使用你input的信息，1则代表我用）
3. Forget gate（和input gate同理，有0,1两个状态，为什么叫forget gate呢，是因为我们知道memory中储存的是之前所有信息的集合，但是有些信息里包含了或者更好地描述了底层的信息，那么就可以将底层的forget。比如六年级我知道了2\*3=6，我就不用按照一年级的2+2+2=6来计算，这个就被forget掉了）
4. Output gate（同理）

给一个例子，更好理解



1. 怎么把一个LSTM cell 和过去的神经网络结合起来



对于每个原始输入，我们利用学习到的四个值，得到了需要四个输入，最终得到输出a。需要注意的就是对于每一个输入，4个g, input gate, output gate 和forget gate是一样的，也是需要学习得到的参数！（简单理解，比如对x1的g=[1,0,0,0], 那么x2的g也是[1,0,0,0]）.

这个理解之后再来个复杂点的

