**人神作业2——CNN**

**实验报告**

姓名：范国杰

班级：计64

学号：2015011619

目录

[一、 实验要求 3](#_Toc527395998)

[二、 实现思路 4](#_Toc527395999)

[1、 loss使用softmax交叉熵 4](#_Toc527396000)

[2、 relu、linear全连接使用和mlp相同的做法 4](#_Toc527396001)

[3、 卷积实现思路 4](#_Toc527396002)

[scipy.signal.convolve2d+for循环 4](#_Toc527396003)

[im2col 4](#_Toc527396004)

[经典图像卷积处理——快速卷积 4](#_Toc527396005)

[4、 参数更新实现思路 8](#_Toc527396006)

[grad\_b: 8](#_Toc527396007)

[grad\_w: 8](#_Toc527396008)

[5、 池化实现思路 8](#_Toc527396009)

[6、池化梯度实现发现 8](#_Toc527396010)

[三、 实验结果 9](#_Toc527396011)

[1、loss和accuracy rate随epoch的变化 9](#_Toc527396012)

[2、4个指定类别 9](#_Toc527396013)

[3、训练的最后结果 10](#_Toc527396014)

[四、 对比MLP 11](#_Toc527396015)

[1、 运行速度 11](#_Toc527396016)

[2、 准确率 11](#_Toc527396017)

[3、 tradeoff 11](#_Toc527396018)

[五、 总结 12](#_Toc527396019)

**图表目录**

[Figure 1 带padding的lena 5](#_Toc527396147)

[Figure 2 把9张图片加权相加 7](#_Toc527396148)

[Figure 3 grad\_w标准公式 8](#_Toc527396149)

[Figure 4 局部梯度递推公式 8](#_Toc527396150)

[Figure 5 loss和accuracy rate随epoch的变化 9](#_Toc527396151)

[Figure 6 标签3 10](#_Toc527396152)

[Figure 7 标签2 10](#_Toc527396153)

[Figure 8 标签1 10](#_Toc527396154)

[Figure 9 标签4 10](#_Toc527396155)

[Figure 10 训练的最后结果 10](#_Toc527396156)

# 实验要求

1. 使用numpy实现cnn对mnist进行分类
2. 将loss和准确率随epoch变化输出
3. 输出4个不同的指定类别的图片
4. 对比mlp和cnn

# 实现思路

## loss使用softmax交叉熵

softmax公式：exp(xi)/ Σj exp(xj)

softmax可以视为多分类的sigmoid实现。从定义上满足了归一性，本身可以作为预测的结果。

交叉熵的公式- Σiyilog(pi),yi是第i个数据的标签，pi是对第i个标签的预测概率，标签是onehot编码的，预测概率也是一个向量，当二者越接近的时候，值越小，因而可以作为损失函数使用。

二者公式看起来都比较复杂， 但求导的时候会发现只有留下yi-pi这一项，因而计算简单

## relu、linear全连接使用和mlp相同的做法

这一部分在上一份报告里面已经详细论述了，这里不做细致讨论

## 卷积实现思路

### scipy.signal.convolve2d+for循环

开始的时候使用scipy.signal.convolve2d+for循环的方案，有结果，但训练一个epoch大概要18s\*600=3h的时间，我最长让它训练了13个epoch，能有95的准确率，但速度慢的惊人。

### im2col

网上的方案大多用im2col，但网上的实现感觉都太繁琐了.同学说用这个方案一个epoch可以达到最快35s。是上面的两个数量级

### 经典图像卷积处理——快速卷积

经过同学的启发，上学期听了数字图像处理的课，想起来可以用经典图像处理的方法来做卷积，以3x3的卷积核为例，见下图

|  |  |  |
| --- | --- | --- |
| a11 | a12 | a13 |
| a21 | a22 | a23 |
| a31 | a32 | a33 |

第一步：我这里的卷积用的都是same，因而先生成带有padding的图像



Figure 带padding的lena

第二步：根据卷积核相应元素，反方向移动图片，说的比较抽象，见下表

|  |  |
| --- | --- |
| a11 |  |
| a12 |  |
| a13 |  |
| a21 |  |
| a22 |  |
| a23 |  |
| a31 |  |
| a32 |  |
| a33 |  |

第三步：把对应系数和相应的图片相乘，而后全部加起来，这样得到的就是full模式的卷积。

我们现在只要same，故只把中间部分取出来。

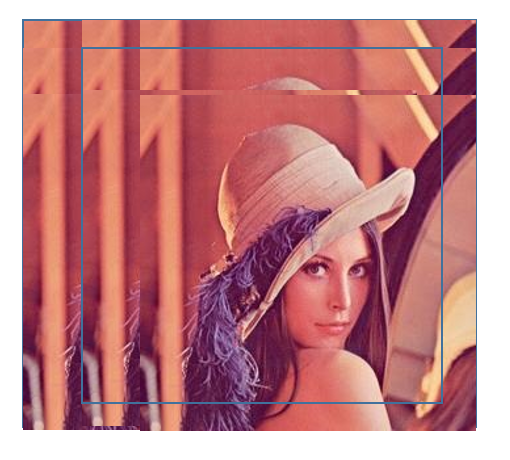


Figure 把9张图片加权相加

如此计算，一个卷积操作只要把9张图片加起来的的工作量，大大减小了计算量。

一个epoch用时28s，更快，并且卷积代码共3行，其中两行在写for循环，可以说是非常的的简洁了

## 参数更新实现思路

这里只说明卷积层的参数更新：

### grad\_b:

这个比较简单，把梯度矩阵除了输入层的channel以外其他维度全部相加即可

值得注意的是，我把grad\_b恒定为0，也就是在卷积时没有b的影响，在20个epoch也可以达到99%的准确率了，猜测这里它的影响很有限

### grad\_w:

引用课件上的结论

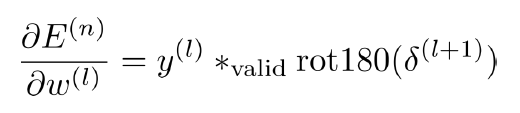


Figure grad\_w标准公式

局部梯度的递推公式也引用课件上的例子：

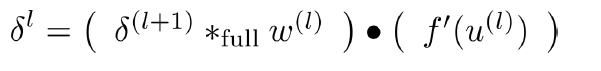


Figure 局部梯度递推公式

但我们这里把relu单独作为一层提取出来了，因而局部梯度只需要被乘数一项即可

## 池化实现思路

没考虑padding ,个人觉得池化的时候padding没什么用，按0处理了，平均池化很直观，用切片就可以实现

## 6、池化梯度实现发现

理论上说，池化梯度应该是输入的梯度除以池化的大小。但实验中发现，如果是乘以池化核的的大小收敛速度大大加快

就算迭代了300个epoch，用乘的方法能到99%左右，可是用除以的方法只有将近98，差距还是有的

和助教讨论的结果是，这一层的lr可能偏小了，但我这里的lr都是全局的，如果是caffe就可以对每一层的lr都给一个值来

但至少，事实证明池化层这里乘除一个常数对网络模型没有本质影响，对收敛速度会有差别

# 实验结果

## 1、loss和accuracy rate随epoch的变化

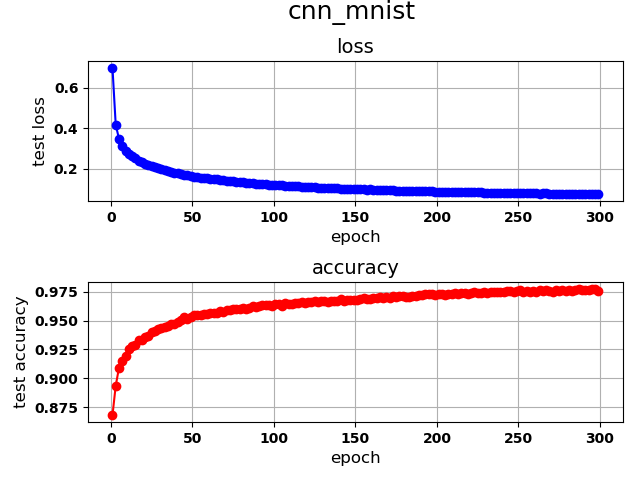


Figure loss和accuracy rate随epoch的变化

## 2、4个指定类别



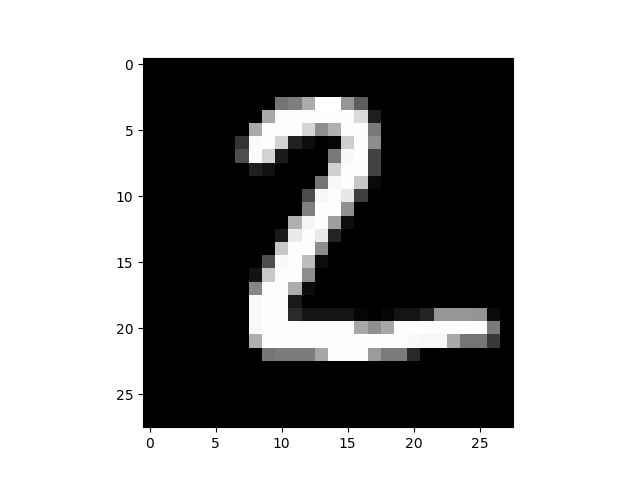
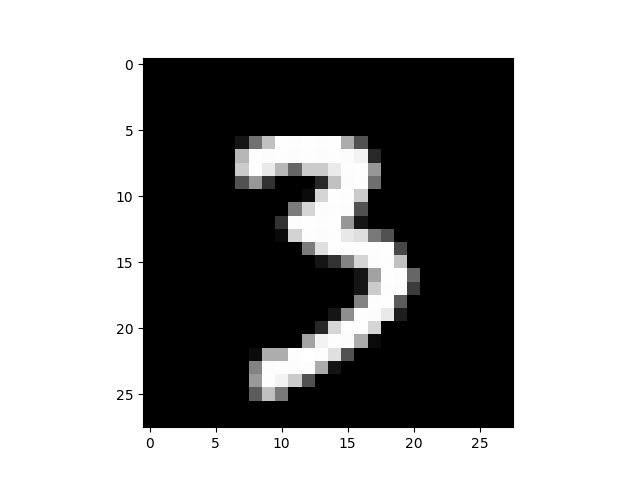


Figure 标签3

Figure 标签2

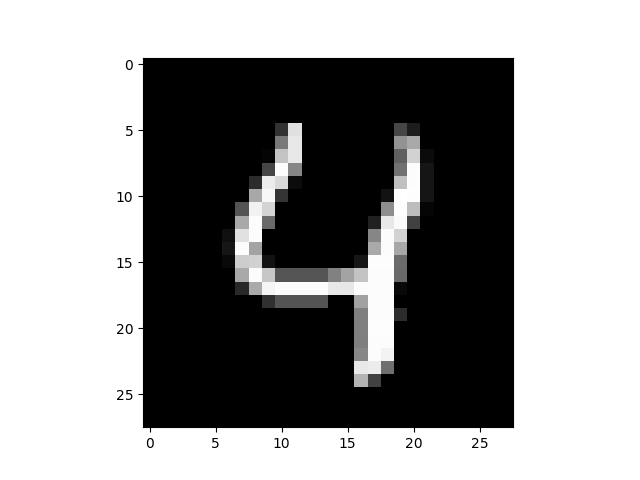
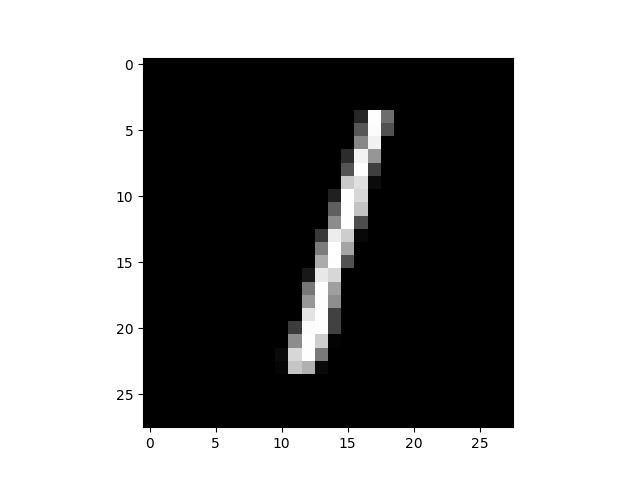


Figure 标签1

Figure 标签4

## 3、训练的最后结果

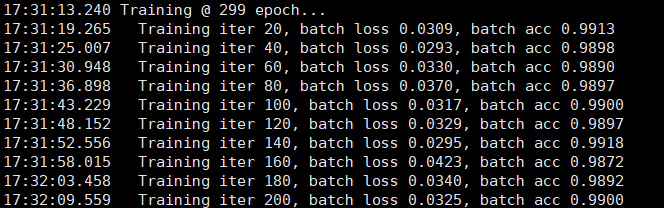


Figure 训练的最后结果

# 对比MLP

## 运行速度

cnn网络显然比mlp一层两层深，计算量由于卷积的存在也更大，速度自然更慢。

但不得不说，这慢的的不是一点两点

## 准确率

cnn网络更“高级”,而且深度也也更深，自然效果要更好，我训练了200~300个，只到99%左右，网上说，最好的的有99.8%。但这显然已经不本质了，因为有些错误就算是人也很容易认错

## tradeoff

现实情况是是需要权衡利弊的。mlp可以轻松到98%+，上了cnn，参数合理99%+,需要调参加上训练本身时间就更长。

结论：cnn是经典的图像处理方法，但对于像mnist这种简单的图像，mlp就足够好了。像素级别大的可以降低像素训练效果也不会太差，毕竟在图像中的冗余信息还是太多了。

# 总结

手写完CNN，感觉整个人都升华了。特别是看到它能跑的时候，还是非常的开心的。尽管中间有不少坎坷，但结果是满意的，也因此学习了不少编程技巧。比如softmax的时候要先统一减去一个最大值，不然可能会越界等神奇事情发生。计算log的时候，输入量用np.clip限制一下，虽然正常计算是不会出问题的，但特殊情况会有奇怪的的事情发生干扰debug。

实现过程有和同学讨论和向Internet取经，如实回答不是全部自己拍脑袋想出来的，但总体框架肯定都是自己手写出来的。

终于可以用TensorFlow了，开心一下。