人工神经网络 CNN报告

董又铭 计53 2015011294

1. **代码补全**

本次作业需要补全两个文件：loss.py和layers.py，需要复制作业MLP中的欧几里得损失、Linear层、Sigmoid层、Relu层的代码。需要自己实现的主要是三个部分，分别是Softmax交叉熵损失函数、卷积层Conv2D，平均值池化层AvgPool2D。由于计算过程相对比较复杂，不合理的利用numpy会大幅降低性能，所以代码实现时需要格外的注意效率。

* 1. Softmax交叉熵

在正向传播时，需要根据input和target算出损失函数的数值loss。张量input和target的形状都是[n, c]。根据定义，计算过程分为两步，第一步是Softmax函数

第二步是交叉熵

但是，直接计算exp和ln函数很容易导致中间结果过大或过小，超出浮点数的标识范围。注意到Softmax函数是关于exp的齐次分式，所以给每个input加上或减去相同的实数，计算结果不会发生变化。考虑到exp的值域，可以将input限制在负数范围内，也就是

这时就需要将两个步骤合并在一起

设

化简得到

反向传播的梯度

可以看到，在正向传播的过程中会计算出的多个中间变量都会在反向传播中使用到，这在写程序时值得注意。

* 1. Conv2D卷积层

为了方便，这里对input和output放弃较为常见的“NCHW”存储模式，采用“NHWC”存储模式，而卷积核weight也从原来的“OIHW”变成“HWIO”模式。

首先处理pad，得到新的input，然后卷积可以表示为

对于每一组，计算涉及到input都是一个子矩阵。将output的下标合并为一个维度，input子块中的下标也同样是，卷积核weight消去两个维度，则卷积可以表示为

可以看到中间部分变成了一个矩阵乘法。使用np.dot可以避免多重for循环语句，而剩下的for循环只有两层，每层的大小是kernel\_size，而kernel\_size一般都是比较小的，所以可以放心大胆的使用。

反向传播时，要分别计算input，weight和bias的梯度，其中bias最简单

只需要对grad\_output的三个维度求和，用np.sum实现。

卷积核weight的梯度依然在for循环中枚举消去weight中的这两个维度

只需要将input转置，仍然是一个矩阵乘法，用np.dot实现。

输入input的梯度仍然枚举，算出的梯度张量也是处理pad之后的input梯度张量，最原始的input的梯度张量需要再进行切片

只需要将weight转置，中间部分仍然是一个矩阵乘法，用np.dot实现。

可以发现，处理pad之后的input在正向和反向传播中都起着至关重要的作用，而且所有的维度大小信息也都会反复利用，注意保存这些信息可以提升代码的执行效率。

* 1. AvgPool2D平均值池化层

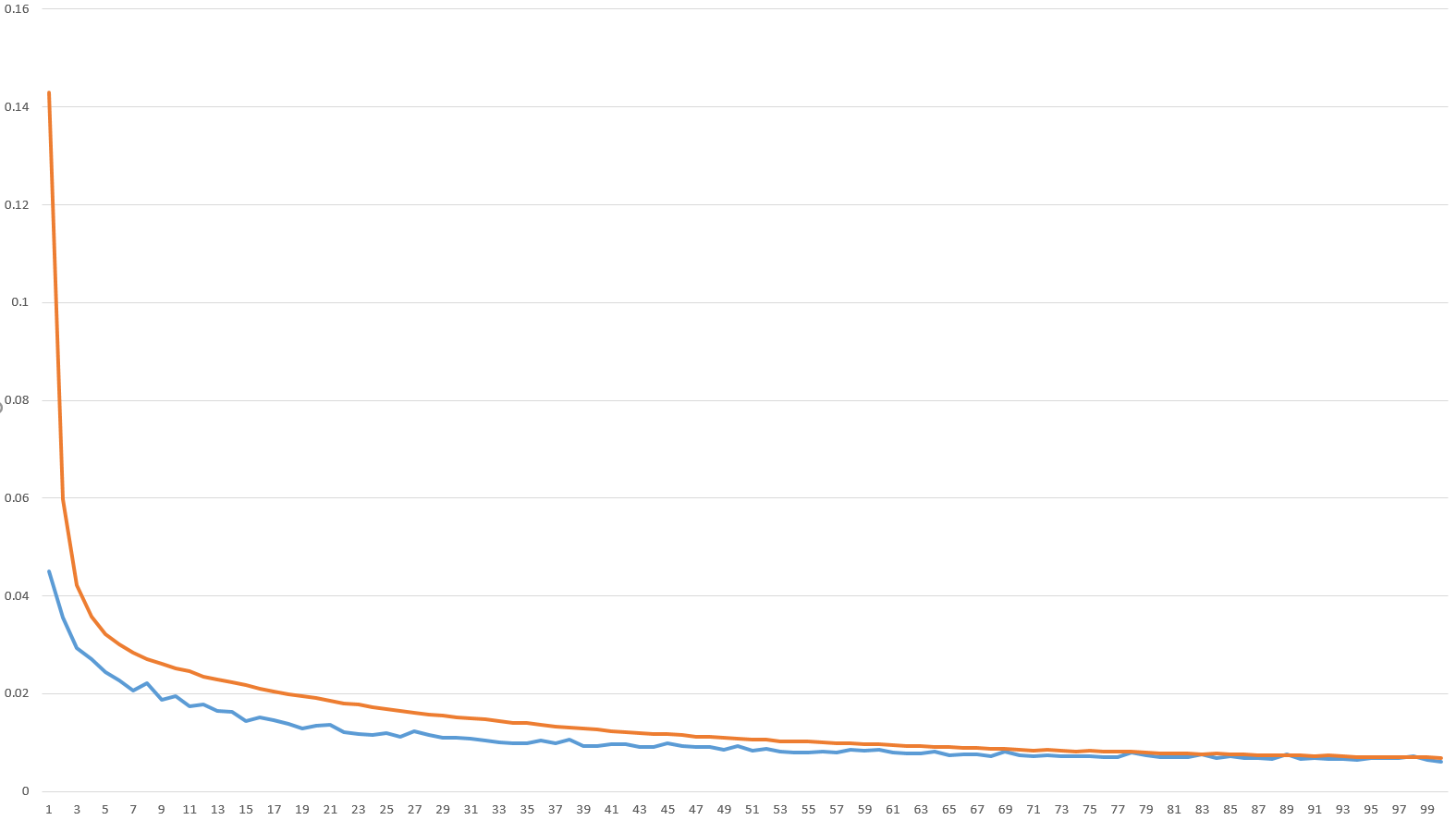
这个层非常简单，合理利用Python的数组下标切片方式，就能直接写出来，无论是正向传播还是反向传播，在此不做过多赘述。

* 1. 检验

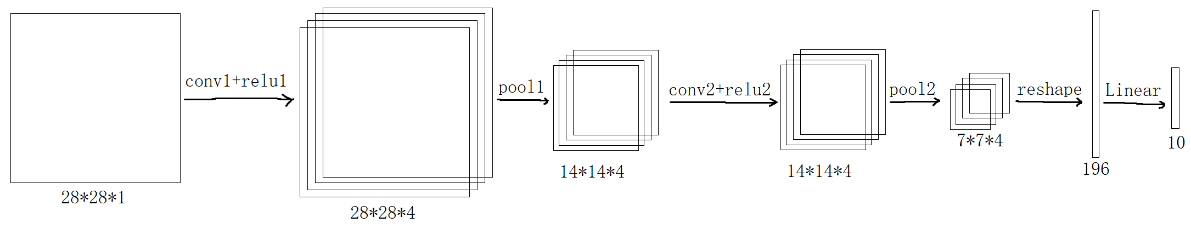
使用了上述各种手段提升代码效率之后，在作业项目给出的最原始网络结构下，每个epoch只需要10秒。

1. **正式开始炼丹**
   1. 看看CNN的效果

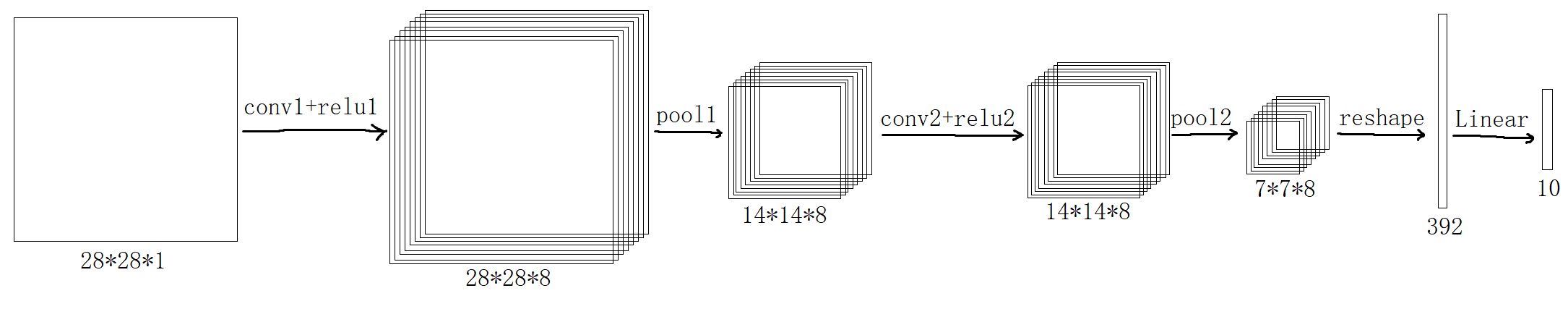
按原始网络训练，以及适当增加通道数训练，迭代100个epoch的loss函数值如下图所示



其中蓝色曲线是作业中给出的最初始4通道CNN的网络模型



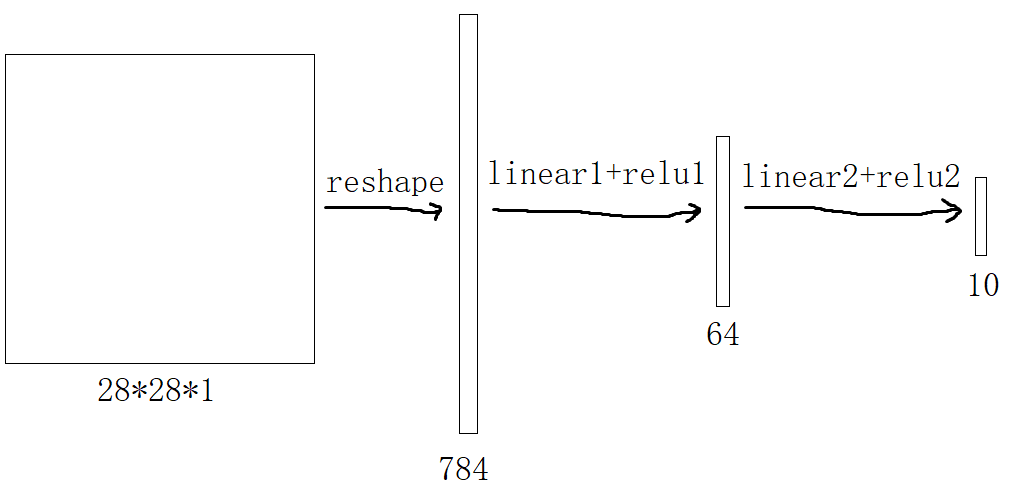
橙色曲线是将原始网络的所有通道数翻倍之后的8通道CNN网络模型



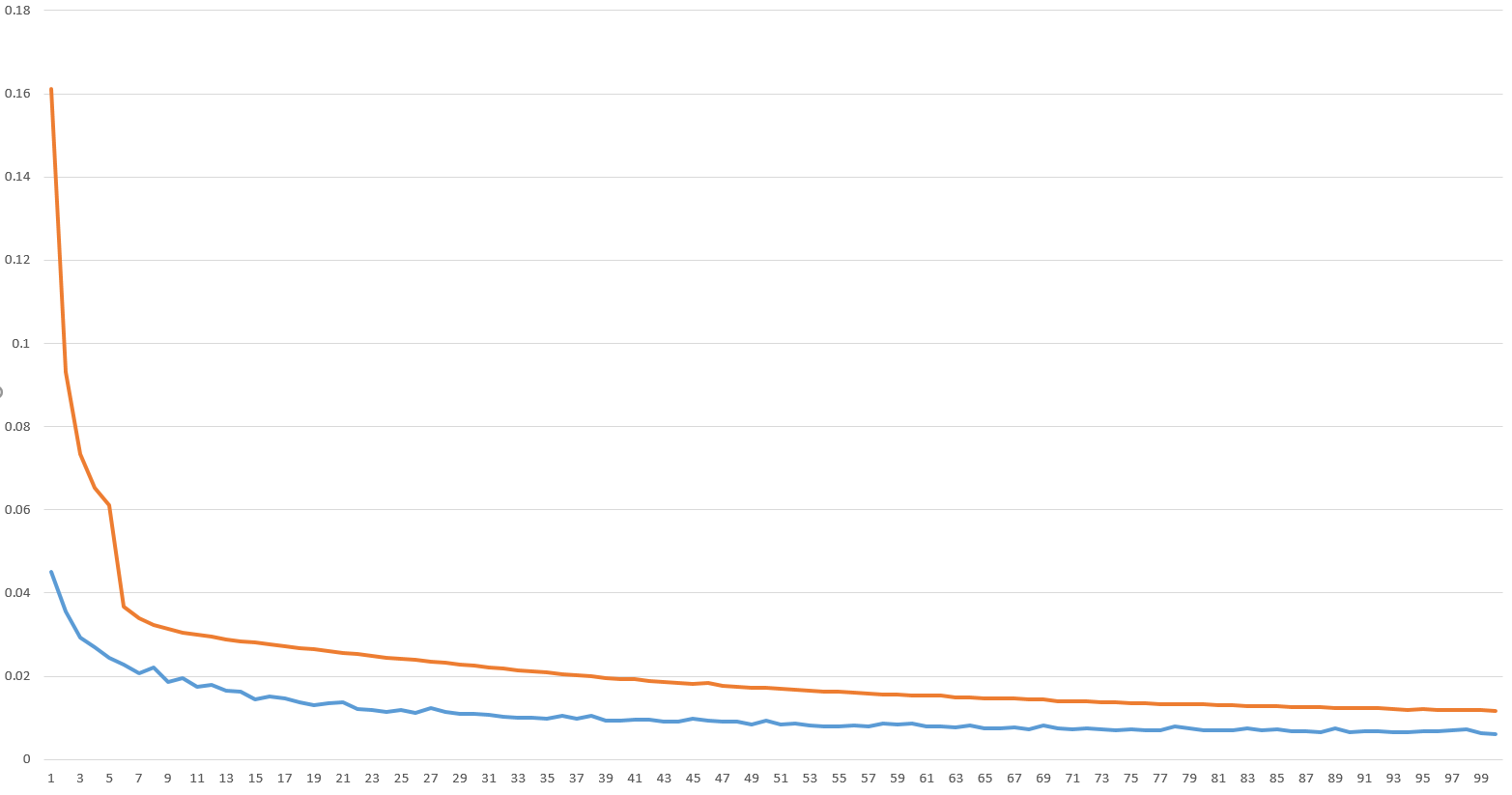
可以看出，通道数较少的情况下，一开始会大幅优于通道数较多的情况，但最终两条曲线趋于重叠，说明最终的网络效果是相近的。从直观的上来看，通道数增多可以让网络模型更加精确，但训练速度回更慢。这里训练速度确实更慢，但并没有看出多通道模型更优，这说明训练时间还不够或者训练数据还不够，没有体现出第二个网络模型的通道数量优势。

* 1. 与双线性层MLP + Softmax的效果对比

选择最好的一个CNN效果，与最好的一个双线性层MLP + Softmax效果进行对比。其中CNN就使用最初的4通道网络，而双线性层MLP的网络结构如图所示



将loss函数绘制在同一个图像中，其中蓝色是CNN，橙色是MLP

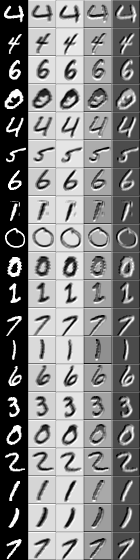


可以看出，CNN参数少所以一开始收敛快，MLP参数多以至于直到最后都没有趋于稳定， 潜力可观。

考虑到CNN虽然参数远少于MLP，核心计算量也少于MLP，但是网络模型训练时间却远高于MLP，可以肯定的得出结论，numpy实现卷积层依然进行了大量的冗余计算，没有达到极致的数据。

* 1. 第一个卷积层的输出。

选择最初的4通道CNN网络模型，抽取100个输入图像，将第一层卷积后的4通道图像和原始图像进行可视化处理，如下。其中每组的第一列都是原图，后面四列分别是第一个卷积层输出的四个通道的数据转化成灰度图

可以看出，原来的数字通过肉眼仍然能够比较清晰的分辨出来，还是很不错的。

1. **总结**

本次作业相比上一次作业，程序运行的时间大幅增加，主要是numpy限制了CPU的计算能力，一方面Python的多线程是假的导致多核CPU没有得到利用，另一方面numpy也不会去主动调用GPU计算，最重要的一方面是，numpy中对张量的切片、转置等操作其实消耗的时间并不少，所以非常的难受。

但是，本次作业让所有学生都清楚的理解了卷积层和池化层的本质，对实际运算过程有了较为深刻的理解，体现了社会主义核心价值观，是一个思想觉悟比较高的方案。