**Numpy实现全连接神经网络**

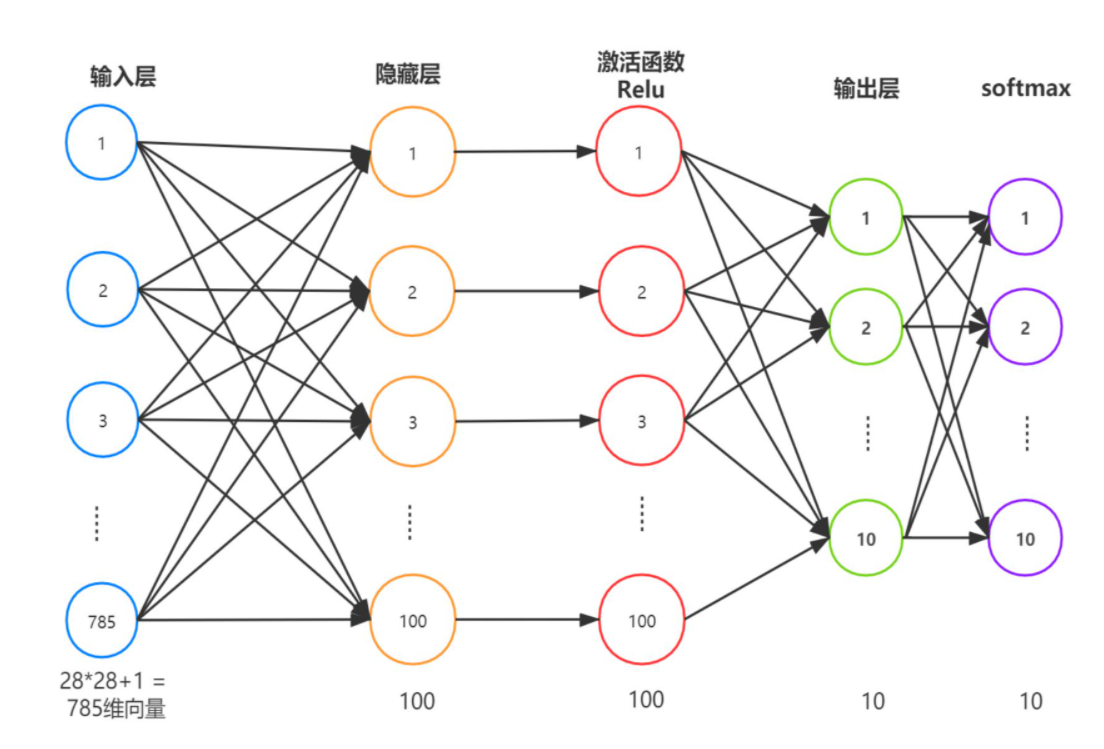
**姓名：胡畅**

**班级：大数据1801**

**学号：2018317230105**

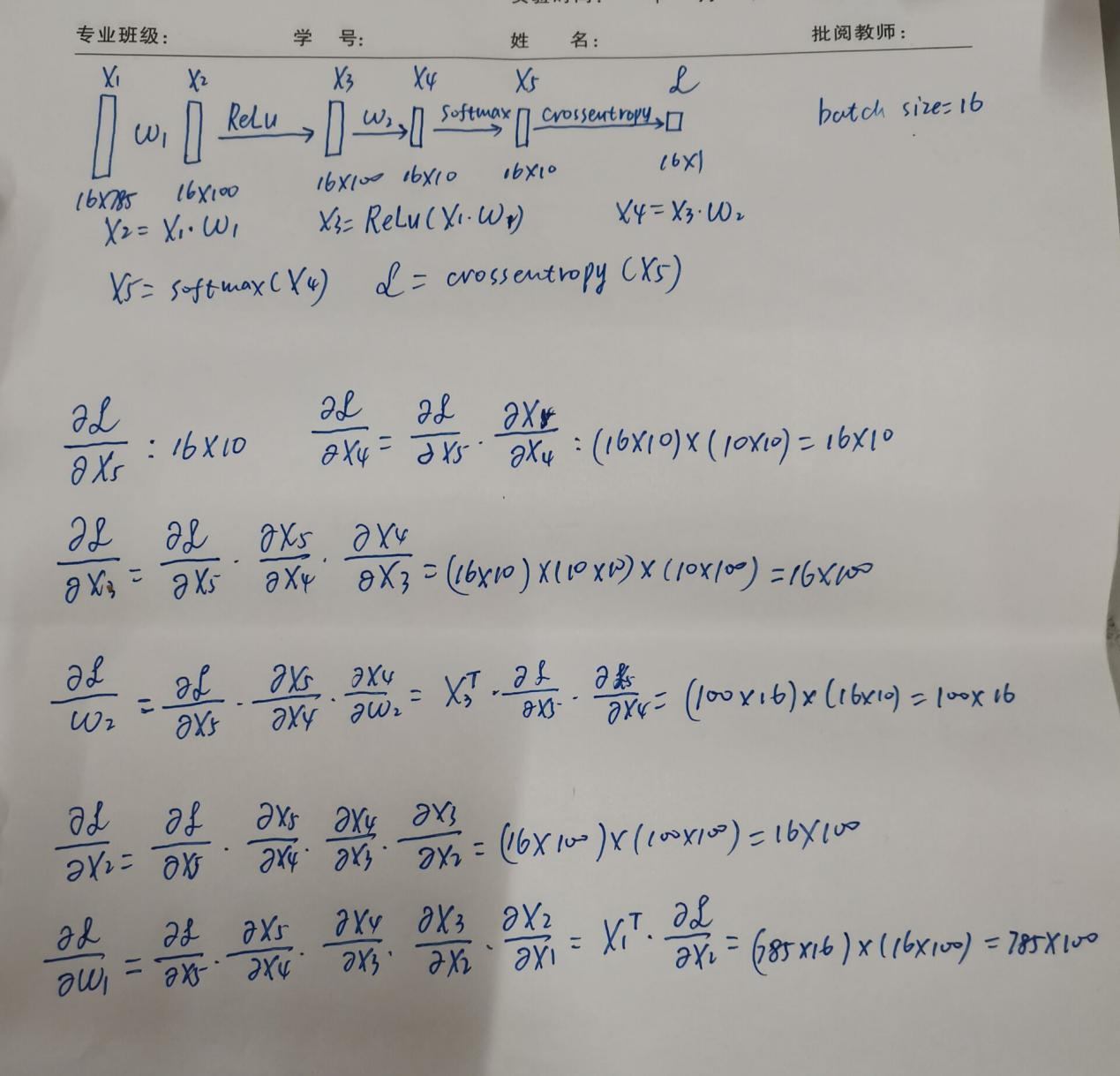
1. **实验要求**

用python中的numpy模块实现全连接神经网络。网络结构为一个输入层、一个隐藏层、一个输出层。隐藏层的激活函数为ReLu，输出层函数为Softmax，损失函数为交叉熵，具体网络结构如下图所示：



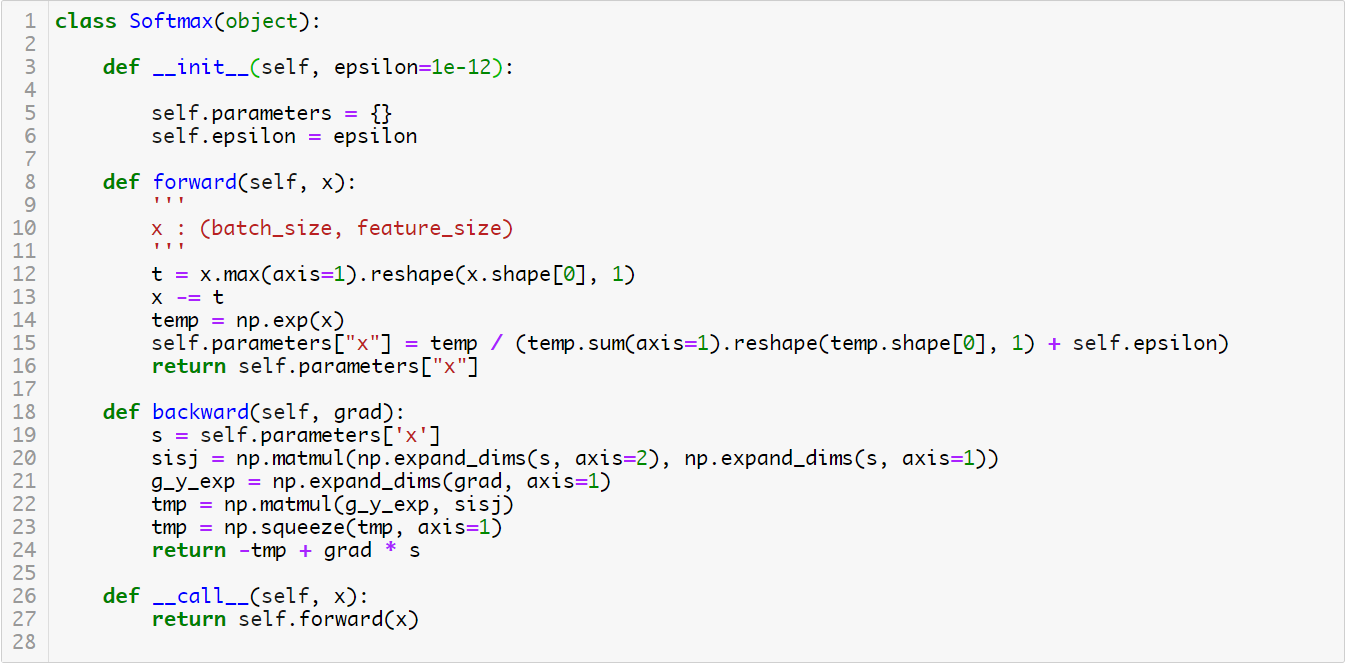
1. **实验方案**

本次实验最终要的不是在写代码，最终要的事搞清楚，数据在网络中的传递过程以及反向传播过程中的具体实现。实验中先确定网络参数的维度，然后进行前向传播，在进行梯度反向回归。下面以batch size = 16的图片为例子进行一次该过程。



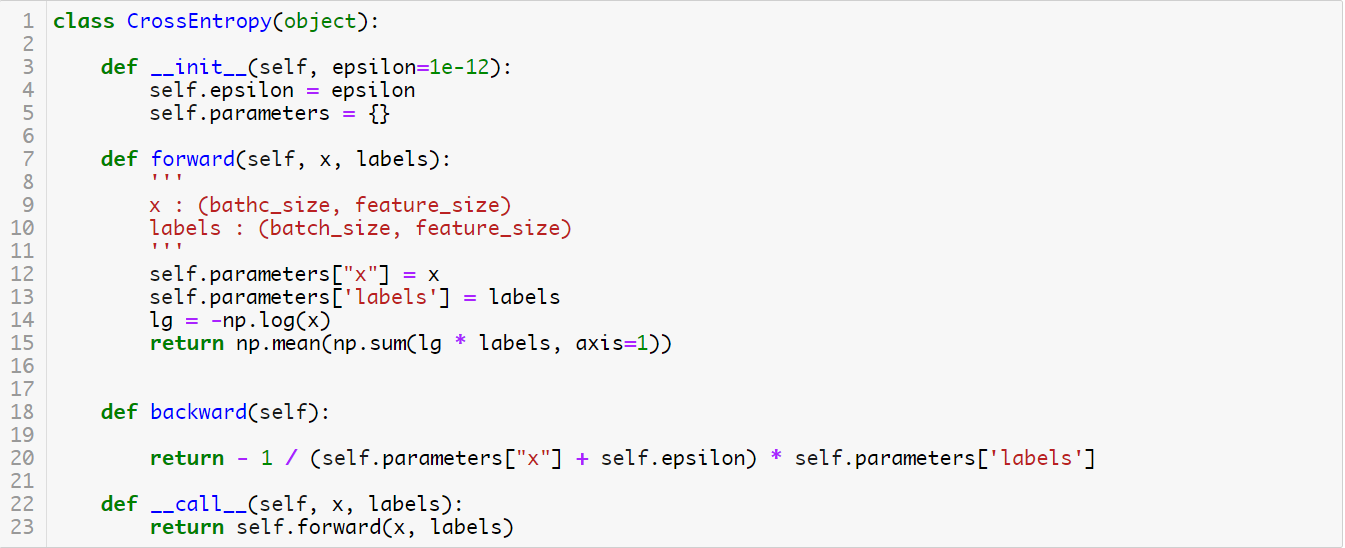
上面就是一组数据在网络中的传递和反向传播的过程的维度变化。根据上述的传播过程，实验过程的维度变化就很清楚了。

1. **实验代码**
2. **Softmax过程**

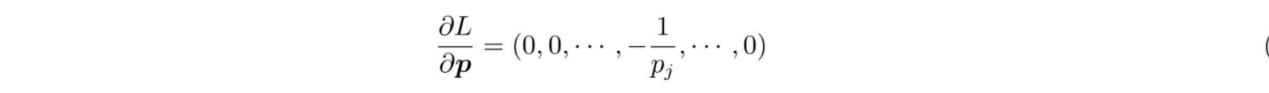


实验过程中要使用到`np.matmul`函数，这个函数使用时是用来计算三维矩阵cxmxn的点积，计算c个大小为mxn矩阵的点积，在反向传播过程中，到softmax需要计算batch size个大小为10x1 和 1x10的向量的点积。

1. **CrossEntropy过程**

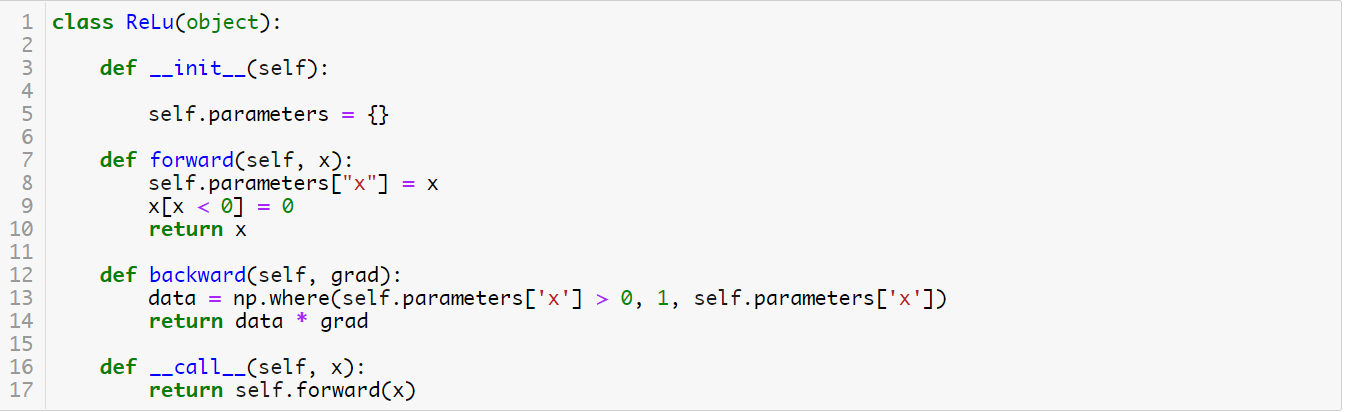


CrossEntropy过程比较简单，主要是求导的过程，根据下面公式计算：



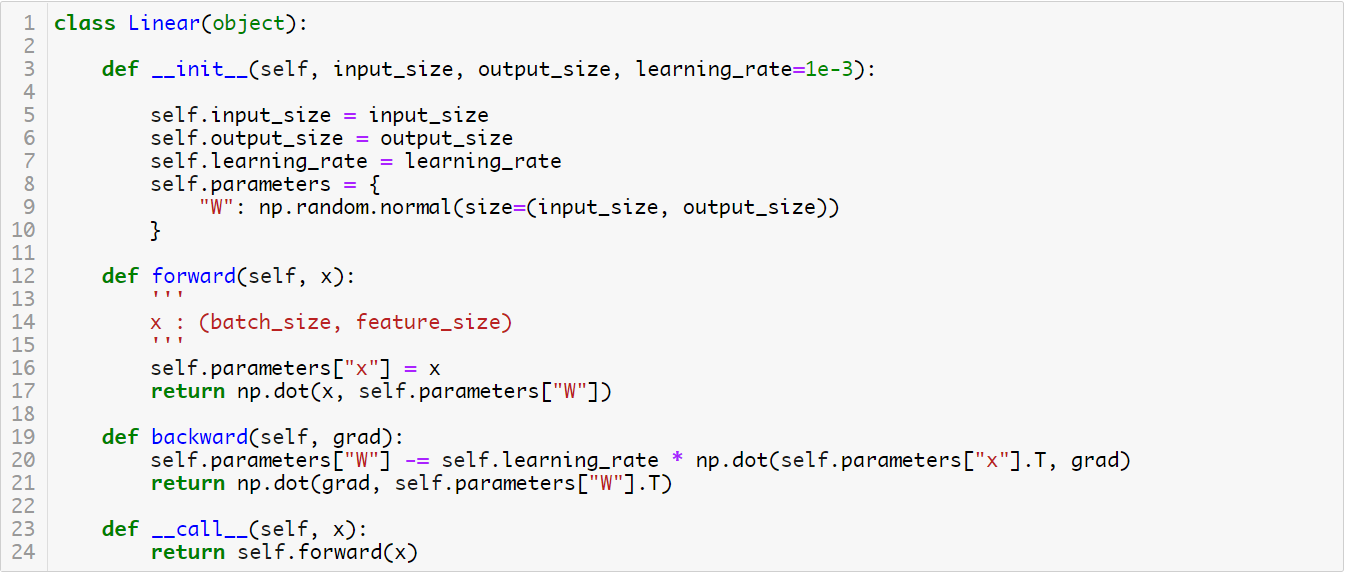
注意：需要放置分母为0，所以需要进行拉普拉斯修正，加一个` self.epsilon`

1. **ReLu过程**



激活函数这一层比较简单，因为ReLu函数的导数非常简单，原x>0的导数为1，其余处为0，x=0处不可导，因此在`backward`过程中直接data \* grad即可

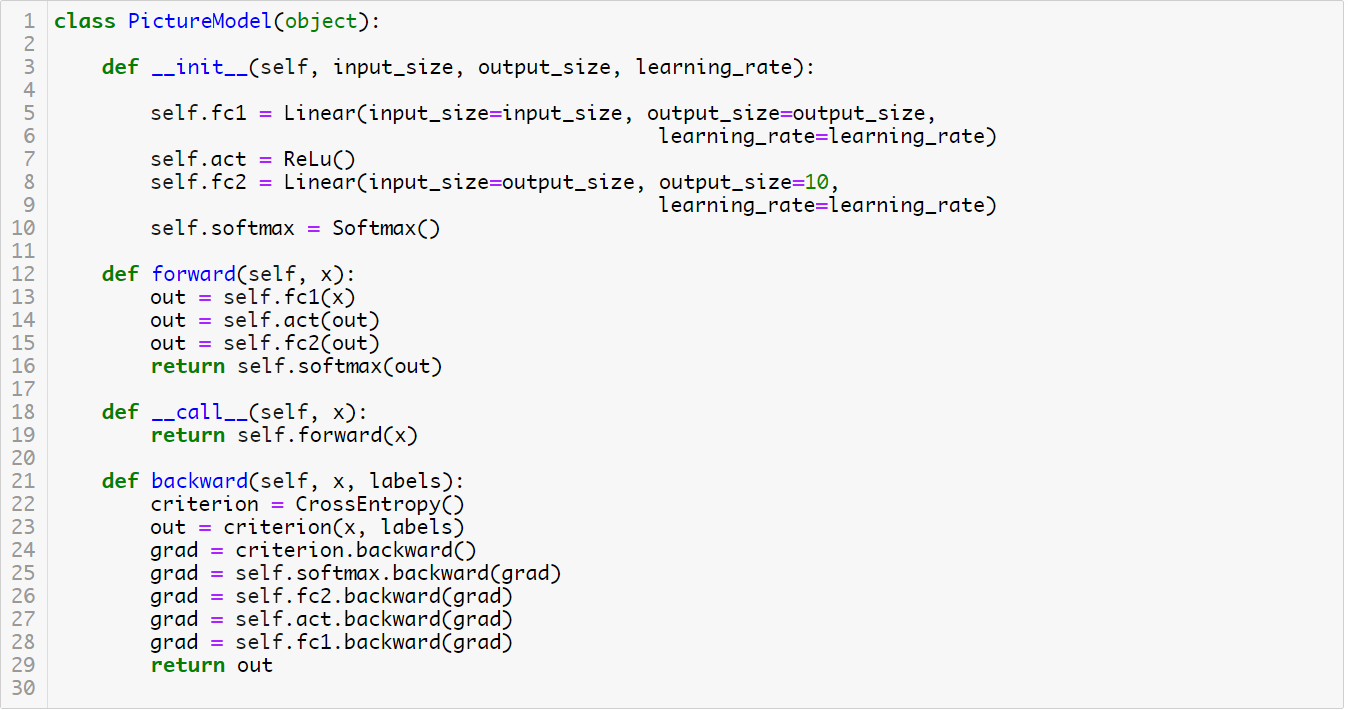
1. **Linear过程**



这一层是本次实验中唯一有参数的一层，这一层很重要，这一层的求导也比较复杂，根据前面的过程来看，本层的求导有两部分一部分是对X求导，另一部分是对W求导，具体过程上面部分已经展示。

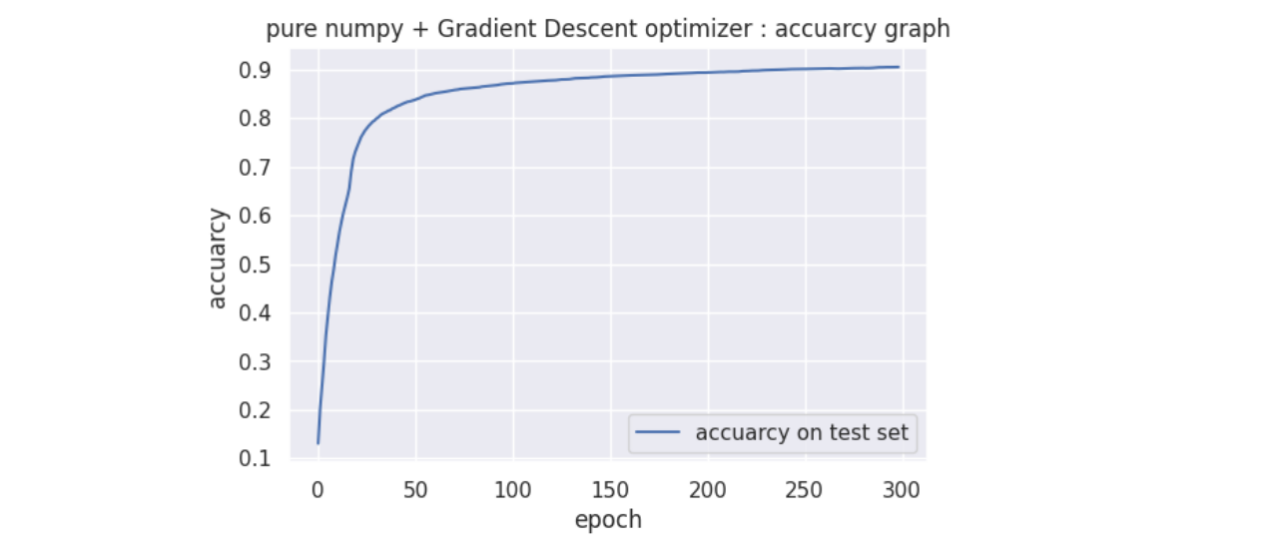
在`backward`过程中第一需要使用梯度下降法对参数进行更新，二是将本层对X的导数继续进行反向传播。

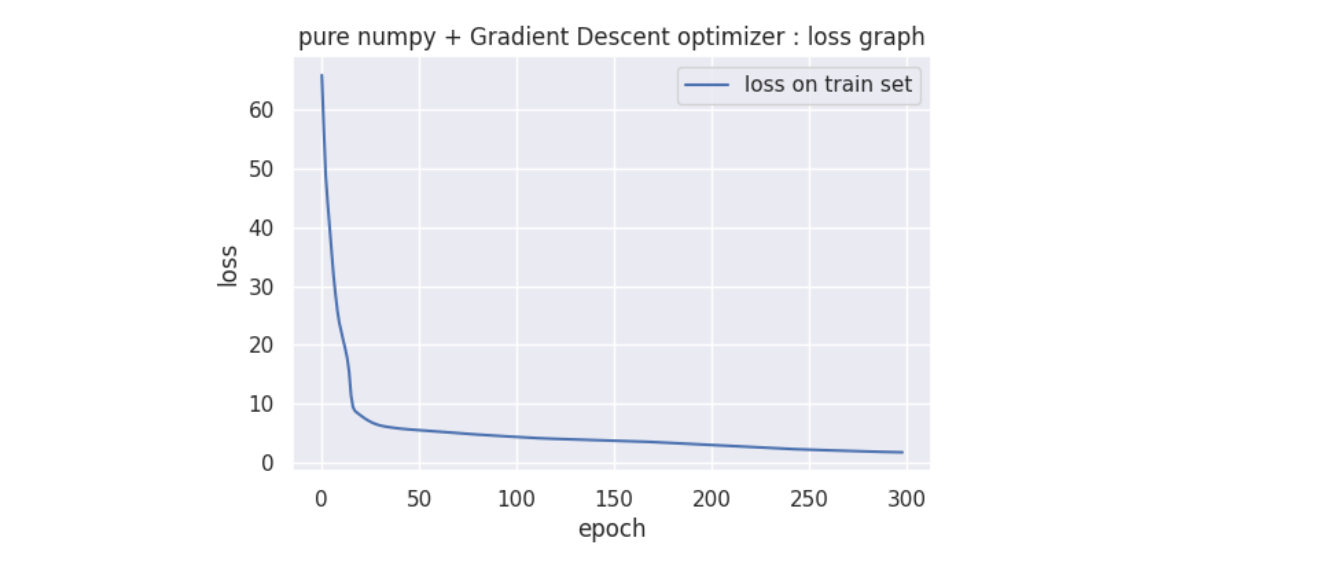
1. **模型构建**



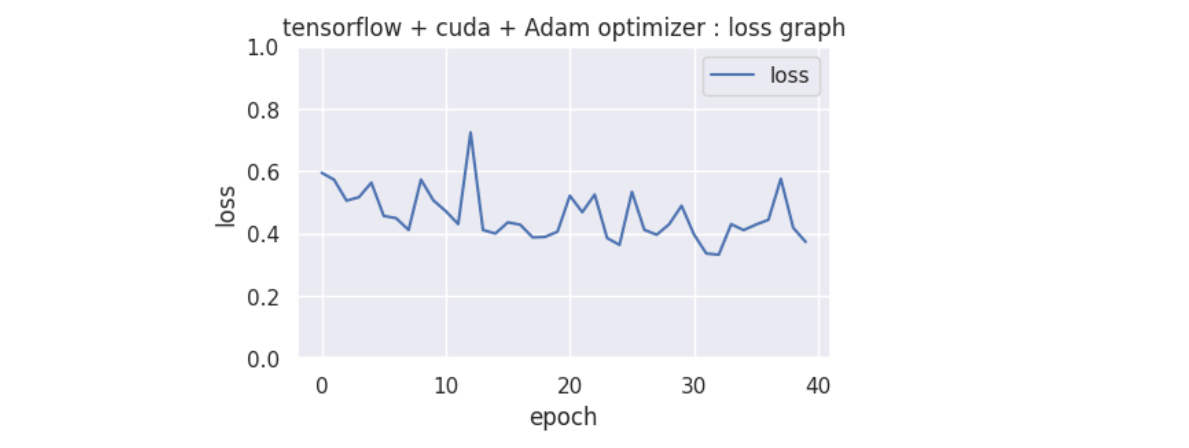
网络结构按顺序进行搭建，Linear + ReLu + Linear + Softmax + CrossEntropy，然后在`backward`部分进行反向传播过程。

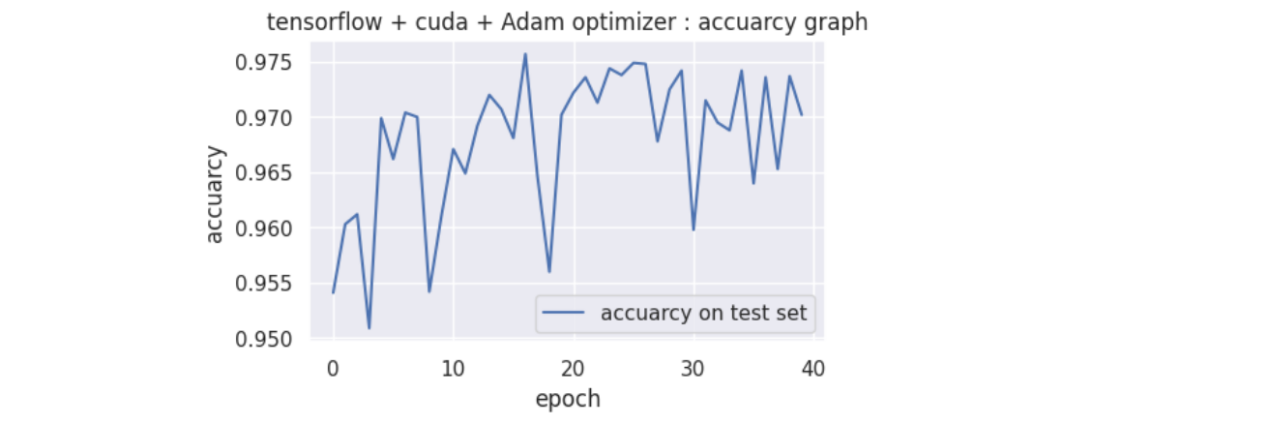
1. **实验结果**
2. **使用纯Numpy + 梯度下降的结果**





**2>直接使用 tensorflow + Adam 优化器的结果**





从结果实验结果来看用纯numpy实现的网络还没有完全收敛，如果继续进行训练，正确率可以达到更高，但是使用numpy 实现的梯度下降算法收敛很慢需要300个epoch才能达到90%的正确率，如果提高学习率的话中途就会遇到局部最优值，再也不能提高正确率了。而用tensorflow和Adam优化器在一个epoch后就能达到95%以上的正确率，而且比较收敛，这里主要原因是优化器的选择不同，所以在实验中优化器的和学习率的选择很重要。

1. **实验总结**

本次用numpy实现全神经网络的实验最重要的不是写代码，而是理解数据在网络中的传播过程，反向传播和数据求导，这些是神经网络的本质，只要了解了网络的本质，代码实现起来就非常简单了，只需要了解具体的API接口的调用即可，numpy有非常详细的参考文档，因此实现过程也比较快，同时网络的学习率，优化器等等对实验结果有很大的影响。

Source code is available at <https://github.com/Chang-LeHung/Deep-Learning-Tutorials/>