

MNIST Digit Classification with MLP

袁无为 计预0

Abstract

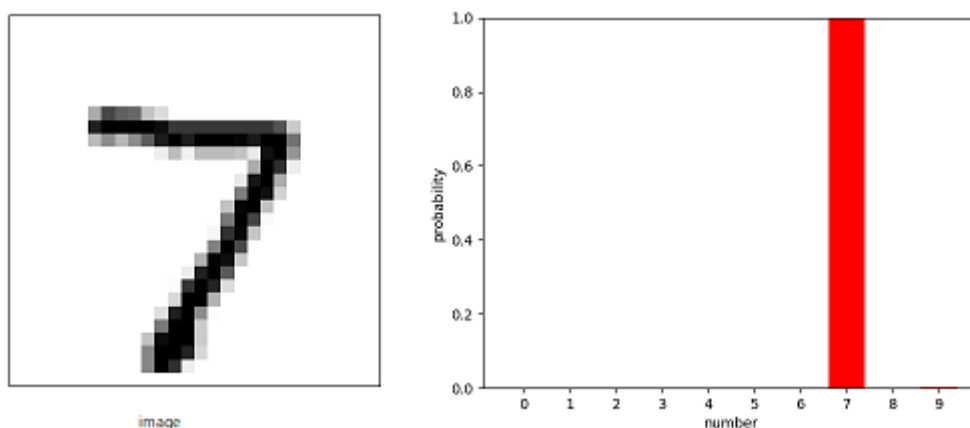
图像识别一直是深度学习中的一个基础又重要的内容。我们对手写数字识别和分类进行了一些研究，搭建了 MLP 模型并使用 MNIST 数据集对我们的模型进行了一些测试。

1. Introduction

手写数字识别是一个非常简单的问题。对于给定的一张 28×28 个像素的只包含一个数字的灰度图，只需要识别出这个图像对应的数字。而且这个数字往往会出现在图像的中央。

把一个图像输入到 MLP 中，会得到预测出的这个图像分别是每个数字的概率。

下图展示了一个手写数字的图像以及识别结果：



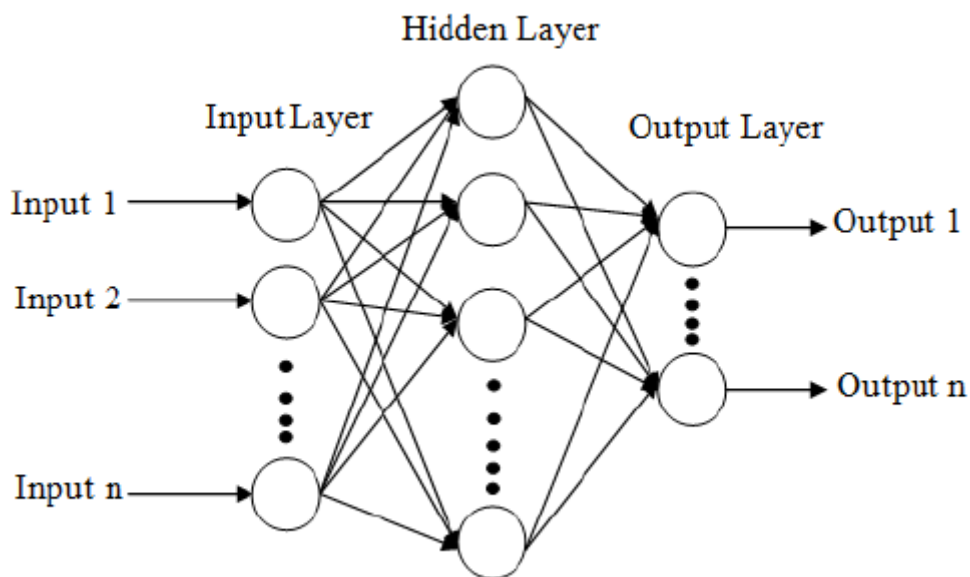
该图为 MNIST 中一个数字 7 的图像以及 MLP 网络的预测结果。

2. Approach

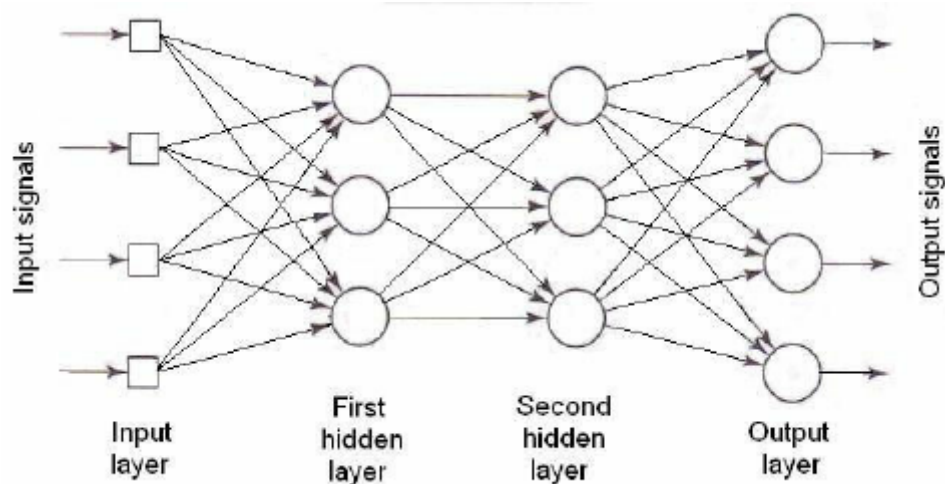
我使用的 MLP 模型主要包含以下几部分内容：

2.1 Basic Structure

我搭建了几个包含一个隐藏层的和几个包含两个隐藏层的模型，它们的结构如下：



包含一个隐藏层的网络



包含两个隐藏层的网络

所有网络的输入层节点个数为 784，输出层节点个数为 10。边为全连接的。没有 Dropout。

对于有一个隐藏层的网络，隐藏层的节点数为 100。

对于有两个隐藏层的网络，第一个隐藏层的节点个数为 200，第二个隐藏层的节点数为 100。

2.2 Activation Function

我选择了两个激活函数：

$$\text{ReLU}(x) = \begin{cases} x & , x > 0 \\ 0 & , \text{otherwise} \end{cases}$$

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

它们的导数分别为：

$$\text{ReLU}'(x) = \begin{cases} 1 & , x > 0 \\ 0 & , x < 0 \end{cases}$$

$$\text{Sigmoid}'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = \text{Sigmoid}(x)(1 - \text{Sigmoid}(x))$$

2.3 Loss Function

我选择了两个损失函数：

EuclideanLoss(MeanSquareError)：对于预测结果 $y^{(n)}$ 和实际的结果（人工标注的结果） $t^{(n)}$ ，损失为：

$$E = \frac{1}{2N} \sum_{n=1}^N \|t^{(n)} - y^{(n)}\|_2^2$$

，其中 N 为 batch size。它的导数为

$$\frac{\partial E}{\partial y_k^{(n)}} = \frac{1}{N} (y_k^{(n)} - t_k^{(n)})$$

SoftmaxCrossEntropy：对于预测结果 $y^{(n)}$ 和实际的结果 $t^{(n)}$ ，损失 E 为：

$$E = \frac{1}{N} \sum_{n=1}^N E^{(n)}$$
$$E^{(n)} = - \sum_{k=1}^K t_k^{(n)} \ln h_k^{(n)}$$
$$h_k^{(n)} = \frac{\exp(y_k^{(n)})}{\sum_{j=1}^K \exp(y_j^{(n)})}$$

其中 K 是分类的类别数。它的导数为：

$$\frac{\partial E}{\partial y_k^{(n)}} = \frac{1}{N} (h_k^{(n)} - t_k^{(n)})$$

2.4 Parameters

我使用了 Xavier 初始化方法，权值矩阵初始化为 $\sigma^2 = \frac{1}{n_i}$ 的正态分布的随机变量。其中 n_i 为前一层节点个数。

偏置矩阵初始化为 0。

2.5 Optimizer

使用了经典的 SGD 方法。

2.6 Normalization

在输入层前先对数据做一次归一化，即 $x'_i = \frac{x_i - \bar{x}}{\sigma}$ 。其中 \bar{x} 为 x_i 的平均值， σ 为 x_i 的标准差。这样做能把 x_i 调整为平均值为 0，方差为 1 的分布。能够减少微小扰动带来的影响。

3. Experiments

3.1 Datasets

使用了经典的 MNIST 手写数字数据集进行测试。

3.2 Implementation Details

不同结构的网络之间的对比

分别搭建了八个不同的网络，使用了相同的参数进行对比（其中误差函数为 SoftmaxCorssEntropy 的网络中输出层没有激活函数）。使用的参数为：learning rate = 0.1, momentum = 0, weight decay = 0, batch size = 100, without normalization。

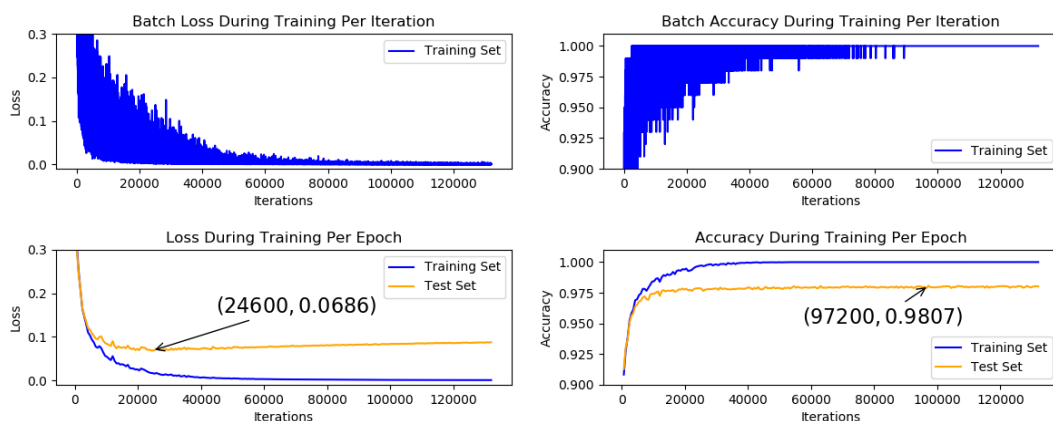
归一化的影响

搭建了八个模型，其中四个在输入层对数据做一遍归一化处理，另外四个没有。使用的参数为：hider layer = 2, learning rate = 0.1, momentum = 0, weight decay = 0, batch size = 100。

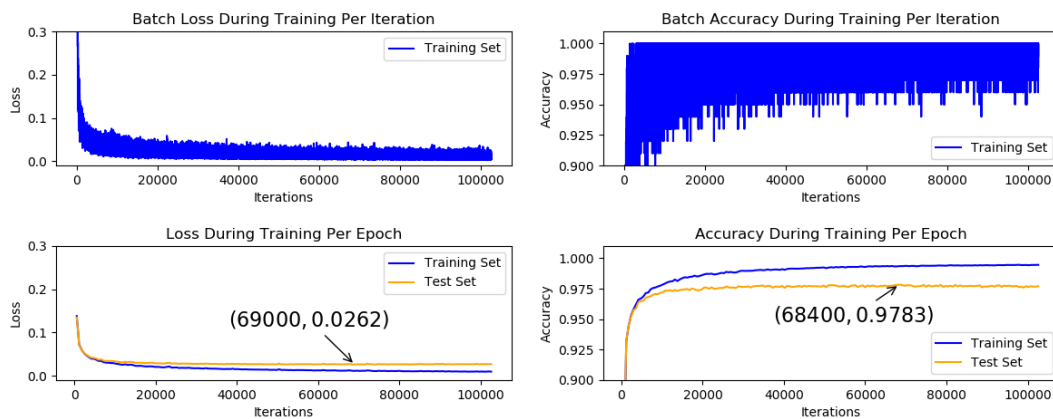
3.3 Quantitative Results

图像

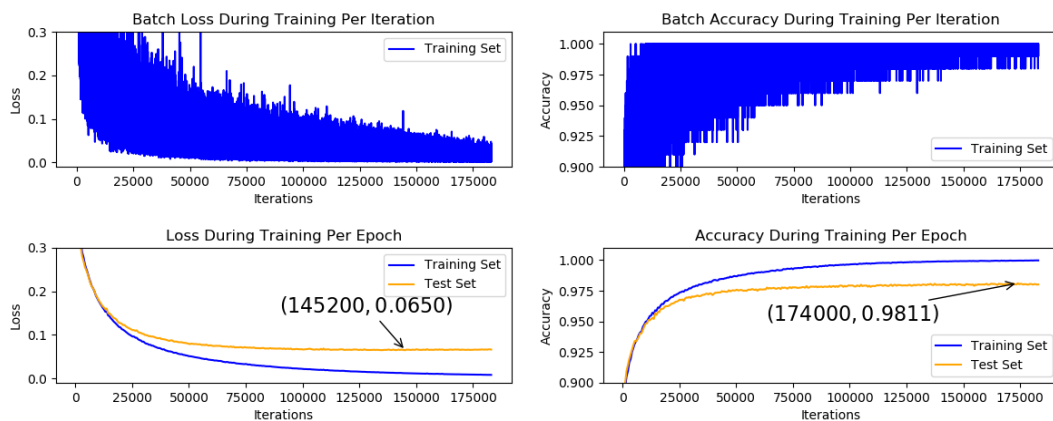
由于时间以及设备原因，我记录下了每次迭代中 Mini Batch 的正确率和损失以及每个 Epoch 后整个训练集以及测试集的正确率和损失。



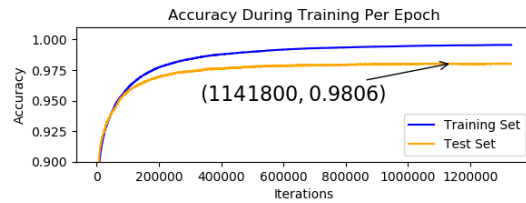
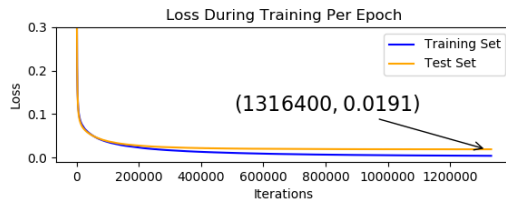
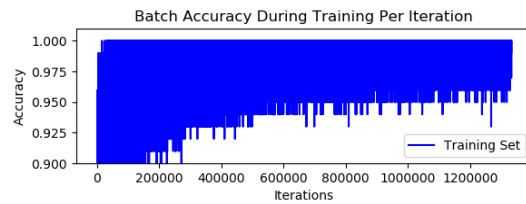
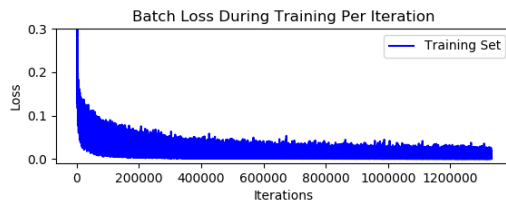
one hidden layer, ReLU, Softmax Cross-Entropy Loss, Without Normalization



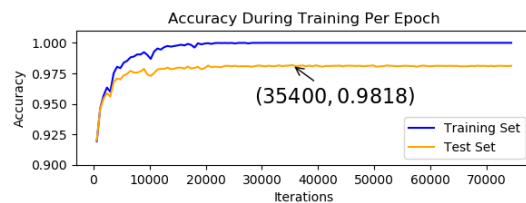
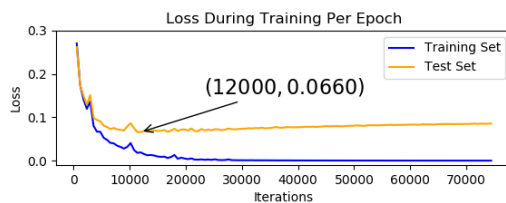
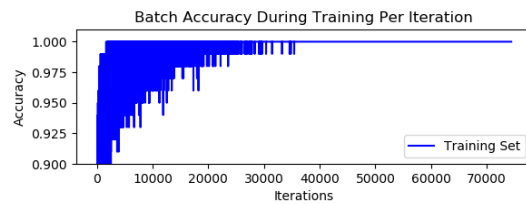
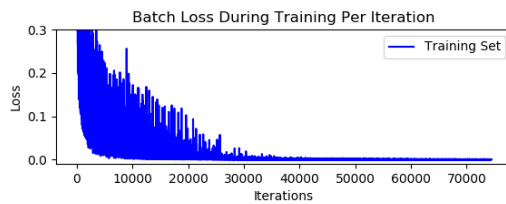
one hidden layer, ReLU, Mean Square Error, Without Normalization



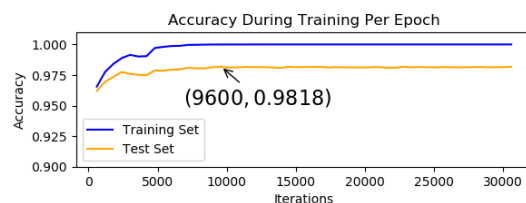
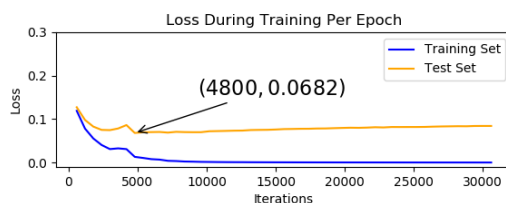
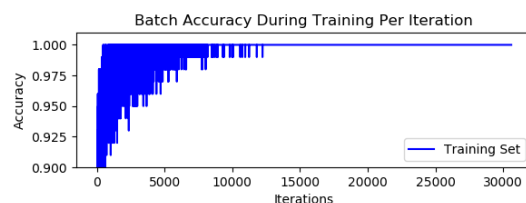
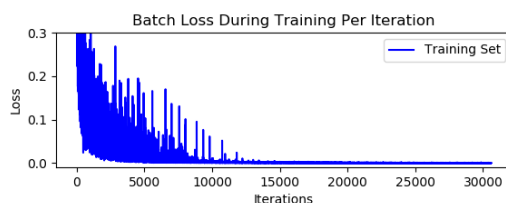
one hidden layer, Sigmoid, Softmax Cross-Entropy Loss, Without Normalization



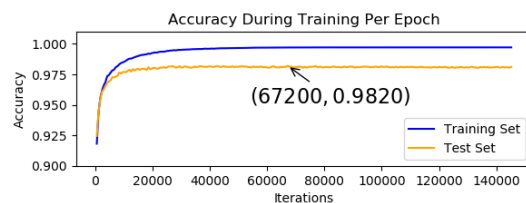
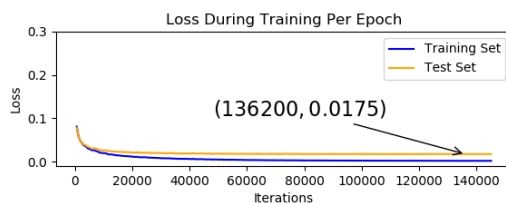
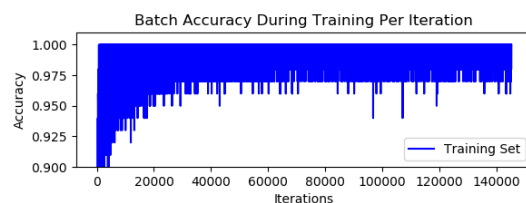
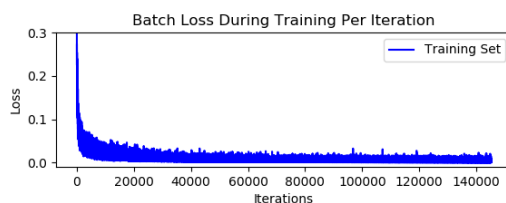
one hidden layer, Sigmoid, Mean Square Error, Without Normalization



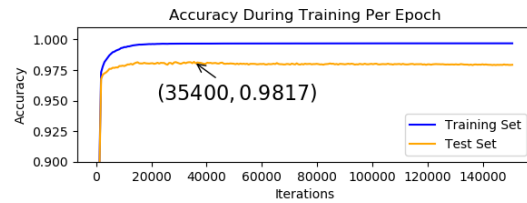
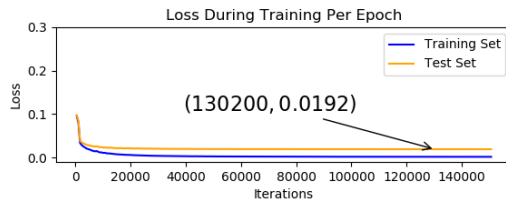
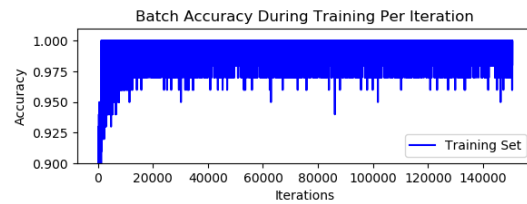
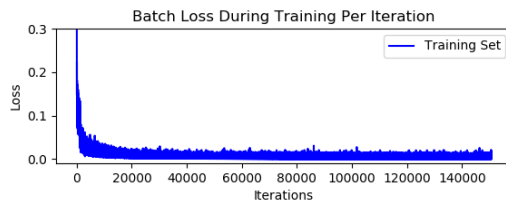
two hidden layer, ReLU, Softmax Cross-Entropy Loss, Without Normalization



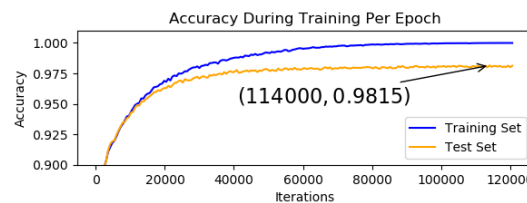
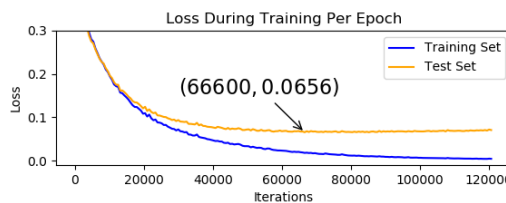
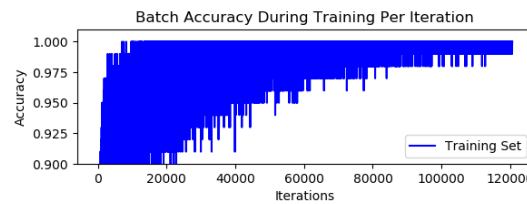
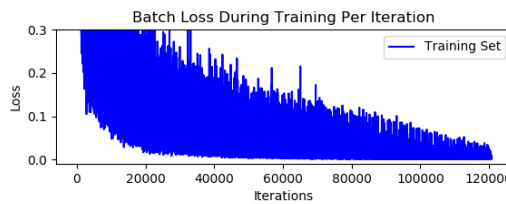
two hidden layer, ReLU, Softmax Cross-Entropy Loss, With Normalization



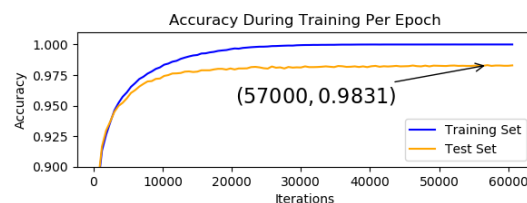
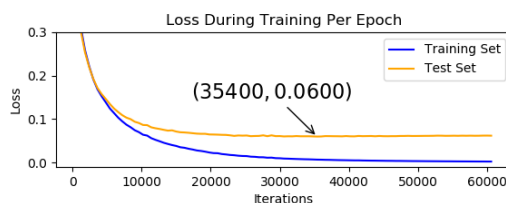
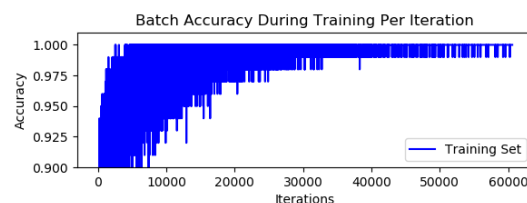
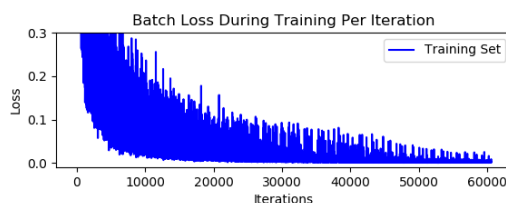
two hidden layer, ReLU, Mean Square Error, Without Normalization



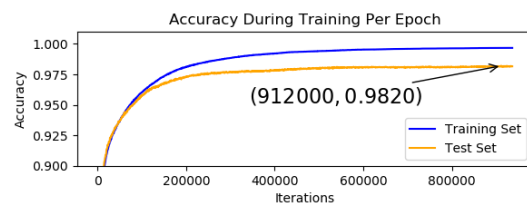
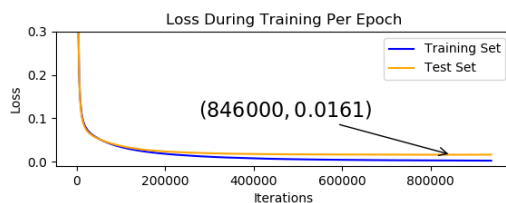
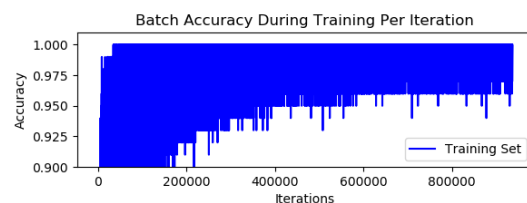
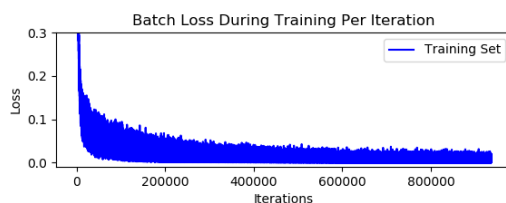
two hidden layer, ReLU, Mean Square Error, With Normalization



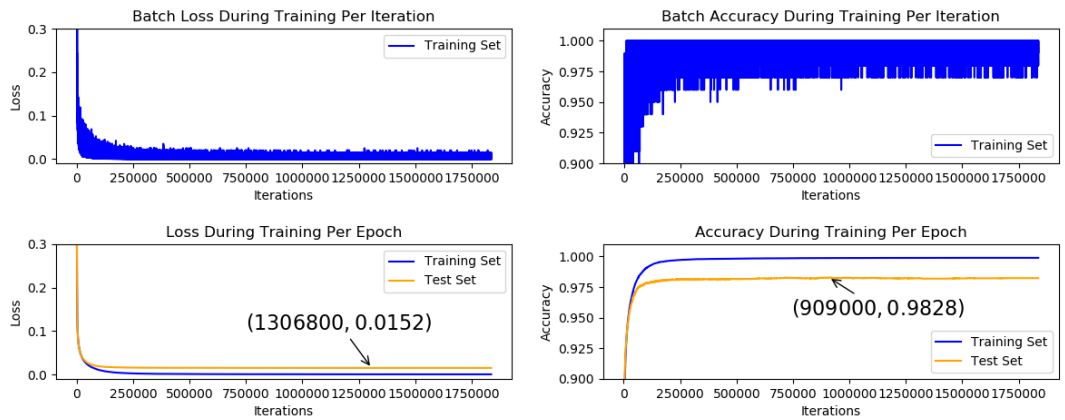
two hidden layer, Sigmoid, Softmax Cross-Entropy Loss, Without Normalization



two hidden layer, Sigmoid, Softmax Cross-Entropy Loss, With Normalization



two hidden layer, Sigmoid, Mean Square Error, Without Normalization



two hidden layer, Sigmoid, Mean Square Error, With Normalization

不同结构的网络之间的对比：

整理出的表格如下：

模型	隐藏层个数	激活函数	损失函数	测试集正确率	测试集达到最大正确率需要的迭代次数	每秒进行的迭代次数	测试集达到95%正确率需要的迭代次数	测试集达到98%正确率需要的迭代次数
模型1	1	ReLU	Softmax Cross-Entropy Loss	98.07%	97200	234	2400	54600

模型	隐藏层个数	激活函数	损失函数	测试集正确率	测试集达到最大正确率需要的迭代次数	每秒进行的迭代次数	测试集达到95%正确率需要的迭代次数	测试集达到98%正确率需要的迭代次数
模型 2	1	ReLU	Mean Square Error	97.83%	68400	237	2400	-----
模型 3	1	Sigmoid	Softmax Cross-Entropy Loss	98.11%	174000	192	11400	118200
模型 4	1	Sigmoid	Mean Square Error	98.06%	1141800	200	72600	848400
模型 5	2	ReLU	Softmax Cross-Entropy Loss	98.18%	35400	88	1800	16800
模型 6	2	ReLU	Mean Square Error	98.20%	67200	80	1800	18600
模型 7	2	Sigmoid	Softmax Cross-Entropy Loss	98.15%	114000	67	13200	66600
模型 8	2	Sigmoid	Mean Square Error	98.20%	912000	68	79800	476400

进行对比之后，可以得到以下结论：

对于有一个隐藏层的模型，用 Sigmoid 作激活函数的模型相对于用 ReLU 作激活函数的模型来说，在测试集上的正确率会更高，但是每次迭代的用时较长，收敛速度前者远远慢于后者。

对于有一个隐藏层的模型，用 Mean Square Error 作损失函数的模型相对于用 Softmax Cross-Entropy 作损失函数的模型来说，准确率相对较低，每次迭代的用时较长，收敛速度较慢。

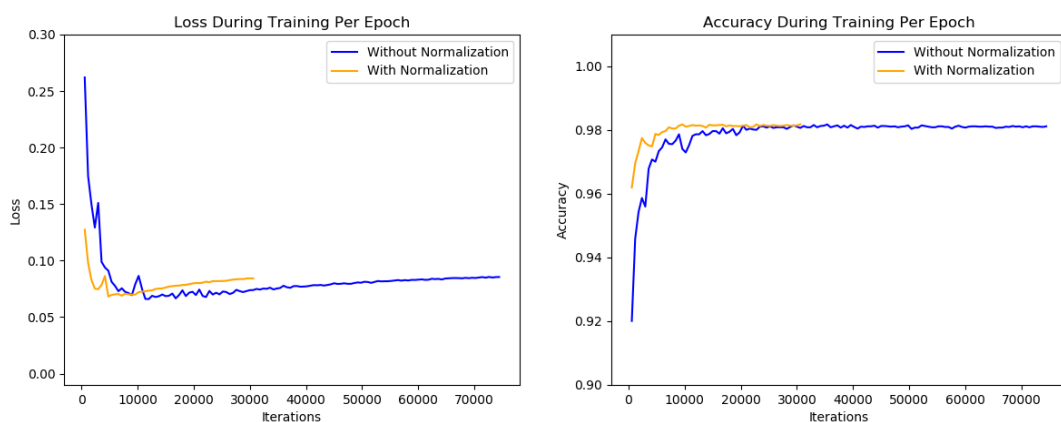
对于有两个隐藏层的模型，用 Sigmoid 作激活函数的模型相对于用 ReLU 作激活函数的模型来说，在测试集上的正确率差不多相同，但是每次迭代的用时较长，收敛速度前者远远慢于后者。

对于有两个隐藏层的模型，用 Mean Square Error 作损失函数的模型相对于用 Softmax Cross-Entropy 作损失函数的模型来说，准确率相对较高，每次迭代的用时较长，收敛速度较慢。

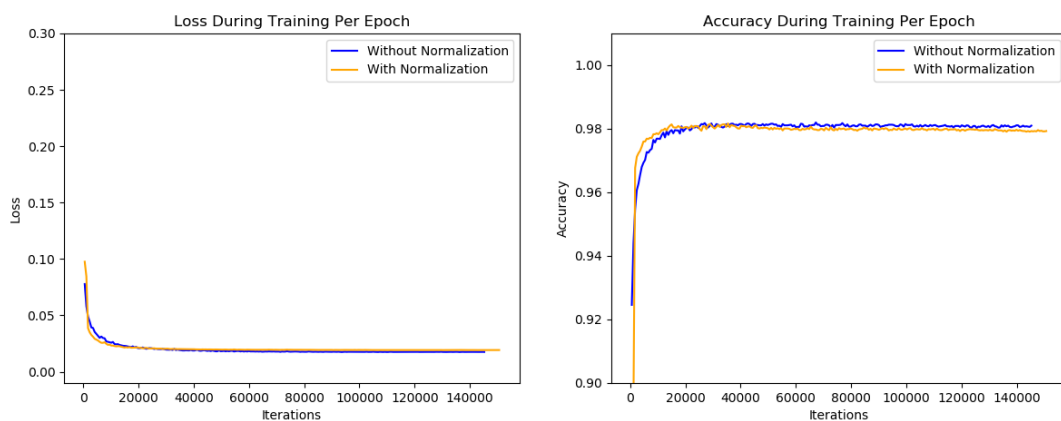
对于损失函数、激活函数相同的两个模型，含有一个隐藏层的模型相对于含有两个隐藏层的模型来说，在测试集上的正确率较低，每次迭代的用时较短，收敛速度较快。

归一化：

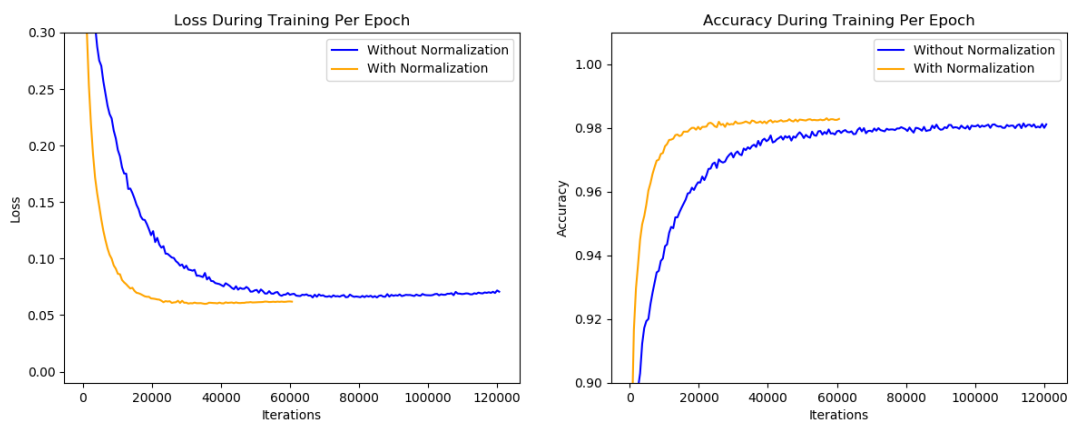
测试集的损失和正确率随迭代次数的图像如下：



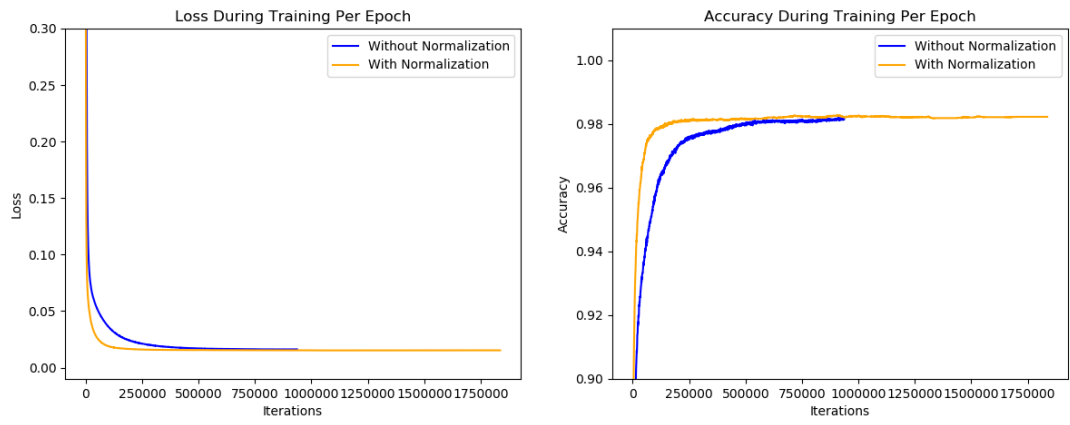
two hidden layer, ReLU, Softmax Cross-Entropy Loss



two hidden layer, ReLU, Mean Square Error



two hidden layer, Sigmoid, Softmax Cross-Entropy Loss



two hidden layer, Sigmoid, Mean Square Error

整理出的表格如下：

模型	激活函数	损失函数	是否有归一化	测试集正确率	测试集达到最大正确率需要的迭代次数	测试集达到95%正确率需要的迭代次数	测试集达到98%正确率需要的迭代次数
模型1	ReLU	Softmax Cross-Entropy Loss	否	98.18%	35400	1800	16800

模型	激活函数	损失函数	是否有归一化	测试集正确率	测试集达到最大正确率需要的迭代次数	测试集达到95%正确率需要的迭代次数	测试集达到98%正确率需要的迭代次数
模型 2	ReLU	Softmax Cross-Entropy Loss	是	98.18%	9600	600	7200
模型 3	ReLU	Mean Square Error	否	98.20%	67200	1800	18600
模型 4	ReLU	Mean Square Error	是	98.17%	35400	1800	12600
模型 5	Sigmoid	Softmax Cross-Entropy Loss	否	98.15%	114000	13200	66600
模型 6	Sigmoid	Softmax Cross-Entropy Loss	是	98.31%	57000	4200	18000
模型 7	Sigmoid	Mean Square Error	否	98.20%	912000	79800	476400
模型 8	Sigmoid	Mean Square Error	是	98.28%	909000	22200	142200

进行对比之后，可以得到以下结论：

归一化对模型的收敛速度有很大的提升，对使用 Sigmoid 函数作为激活函数的模型的正确率有微小的提升。

4. Conclusion

在这次作业中，我对手写数字识别以及 MLP 进行了一些研究。包括 MLP 的结构、激活函数、损失函数、初始化以及一些其他的内容。实验证明 MLP 能较好的完成手写数字识别的工作。尽管如此，MLP 也有一些不尽人意地方，会有一小部分的数字识别不出或识别错误。

References

[Exploring Context and Visual Pattern of Relationship for Scene Graph Generation](#)

[深度学习——Xavier初始化方法](#)

[Batch Normalization原理与实战](#)

[含有一个隐藏层的神经网络图片](#)

[含有两个隐藏层的神经网络图片](#)