# Deep-Learning实验报告

## （MNIST）实验

日期：2017.6.23

学号：24320142202455

姓名：林金鹏

## 实验目标：

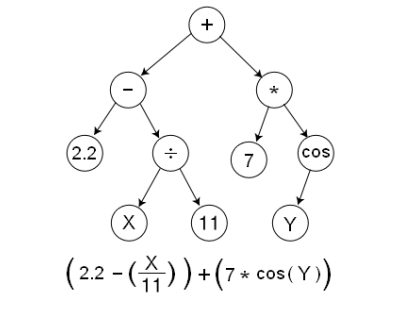
* 熟悉 tensorflow 的使用，以及运作方式
* 了解 jupyter 的使用
* 学会基本的MNIST数据集识别

## 实验内容：

* Anaconda3 + python3.5 + spyder + jupyter 实验环境的安装
* 通过不同的损失函数和激活函数进行MNIST 数据集的训练识别

## 实验相关知识：

* Tensorflow 的运作方式：



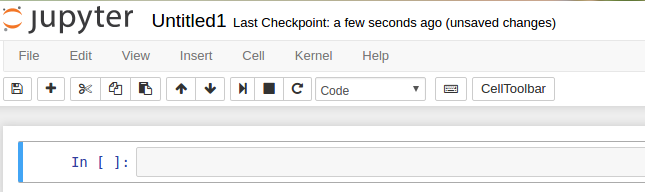
TensorFlow是用数据流图(data flow graphs)技术来进行数值计算的。

[数据流图](http://bears.ece.ucsb.edu/research-info/DP/dfg.html)是描述有向图中的数值计算过程。

有向图中，节点通常代表数学运算，边表示节点之间的某种联系，它负责传输多维数据(Tensors)。

节点可以被分配到多个计算设备上，可以异步和并行地执行操作。因为是有向图，所以只有等到之前的入度节点们的计算状态完成后，当前节点才能执行操作。

* Jupyter 的使用



notebook 界面的组成部分

1）notebook 的名称

2）主工具栏提供了保存、导出、重载 notebook，以及重启内核等选项

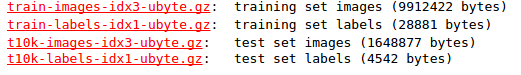
3）快捷键

4）notebook 编辑区

最下面的哪个 In [ ]: 的框叫做单元格，你可以把你的代码分成一段段的单元格输入，然后可以逐个单元格地运行。注意，这个功能是非常友好的，有时候只修改了中间的一小段代码，又不想全部代码都要重新运行的时候这个功能就非常有用了。另外，单元格是可以改变顺序的。而且可以输出图片和绘图！

* MNIST数据集

这个数据集由四部分组成，分别是图示：



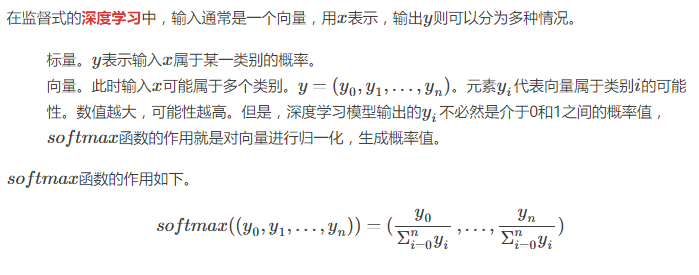
也就是一个训练图片集，一个训练标签集，一个测试图片集，一个测试标签集；我们可以看出这个其实并不是普通的文本文件或是图片文件，而是一个压缩文件，下载并解压出来，我们看到的是二进制文件，其中训练图片集的内容部分如此。

包含有60000 行的训练数据集，mnist.train

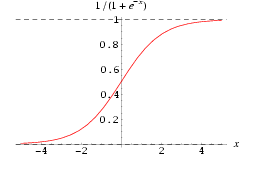
还有10000 行的测试数据集,mnist.test

有一个单独的测试数据集不用于训练而是用来评估这个模型的性能，从而更加容易把设计的模型推广到其他数据集上。图片设置为 xs ,标签设置为 ys ,图片为 28X28 = 784 pi。所以 mnist.train.images 是一个 [60000,784] 的张量。所以 mnist.train.labels 是一个 [60000,10] 的张量。

* Softmax 激活函数



* Sigmoid 函数

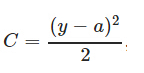


（1）对于深度神经网络，中间的隐层的输出必须有一个激活函数。否则多个隐层的作用和没有隐层相同。这个激活函数不一定是sigmoid，常见的有sigmoid、tanh、relu等。

（2）对于二分类问题，输出层是sigmoid函数。这是因为sigmoid函数可以把实数域光滑的映射到[0,1]空间。函数值恰好可以解释为属于正类的概率（概率的取值范围是0~1）。另外，sigmoid函数单调递增，连续可导，导数形式非常简单，是一个比较合适的函数

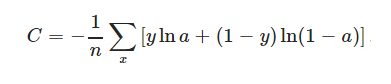
（3）神经网络中的激活函数，其作用就是引入非线性。具体的非线性形式，则有多种选择。sigmoid的优点在于输出范围有限，所以数据在传递的过程中不容易发散。当然也有相应的缺点，就是饱和的时候梯度太小。sigmoid还有一个优点是输出范围为(0, 1)，所以可以用作输出层，输出表示概率。

* 方差代价函数



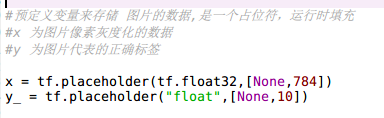
用于计算两组数据值之间的距离

* 交叉熵 cross\_entropy

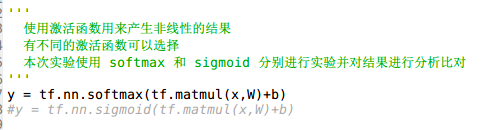


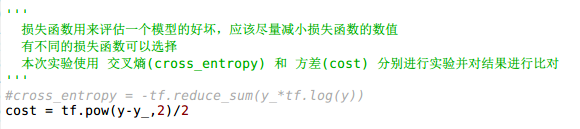
交叉熵可在神经网络(机器学习)中作为损失函数，p表示真实标记的分布，q则为训练后的模型的预测标记分布，交叉熵损失函数可以衡量p与q的相似性。交叉熵作为损失函数还有一个好处是使用sigmoid函数在梯度下降时能避免均方误差损失函数学习速率降低的问题，因为学习速率可以被输出的误差所控制。

## 关键代码：

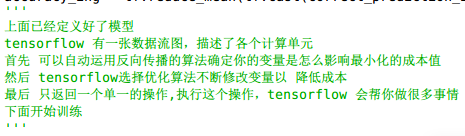


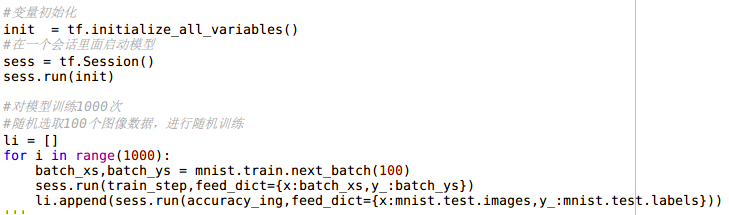


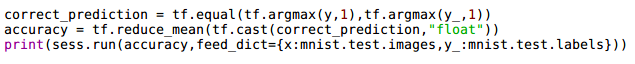








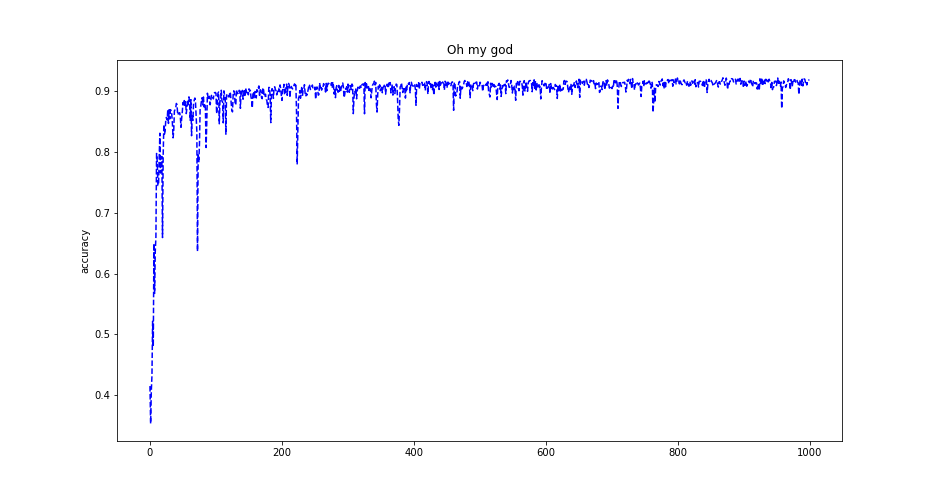




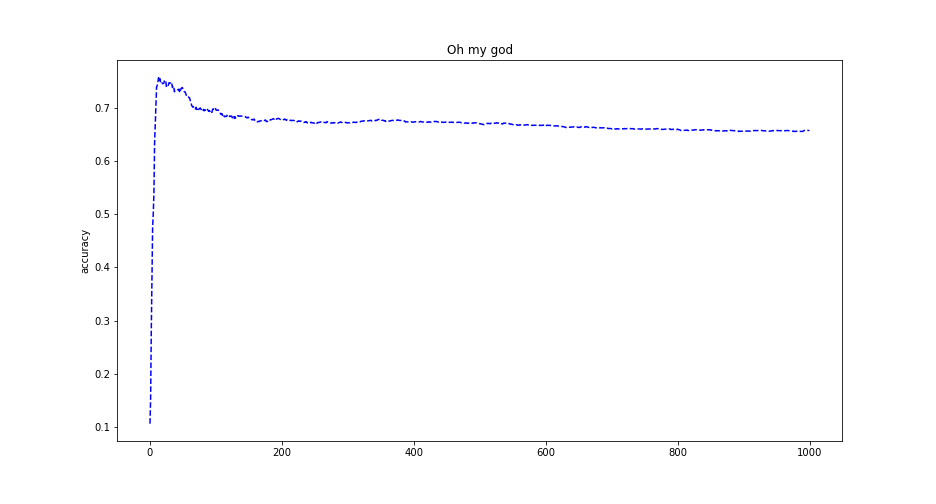
## 实验结果分析：

采用不同损失函数和不同的激活函数 会导致不一样的结果。

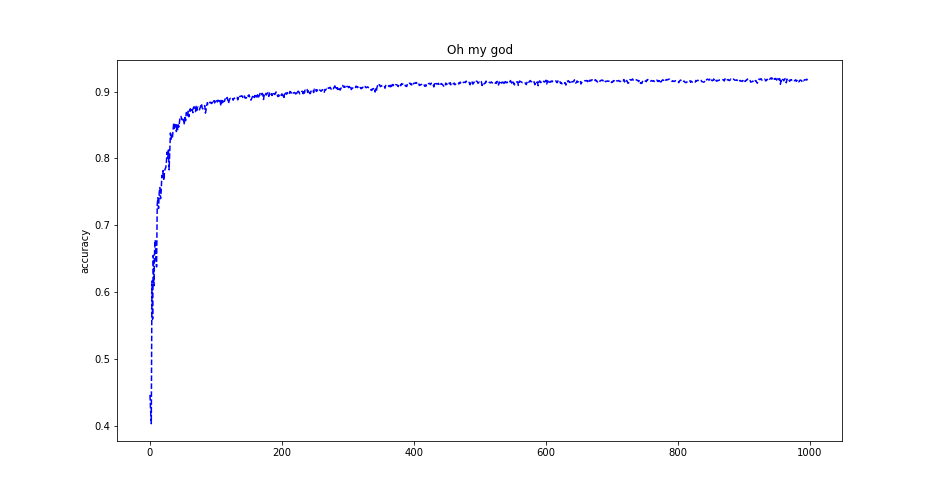
使用cross\_entropy + softmax



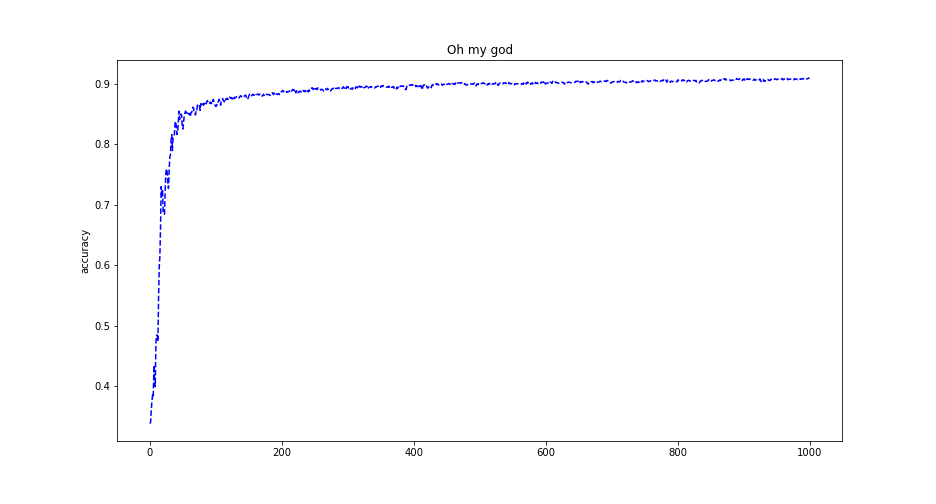
使用 cross\_entropy+sigmoid



使用 cost + softmax

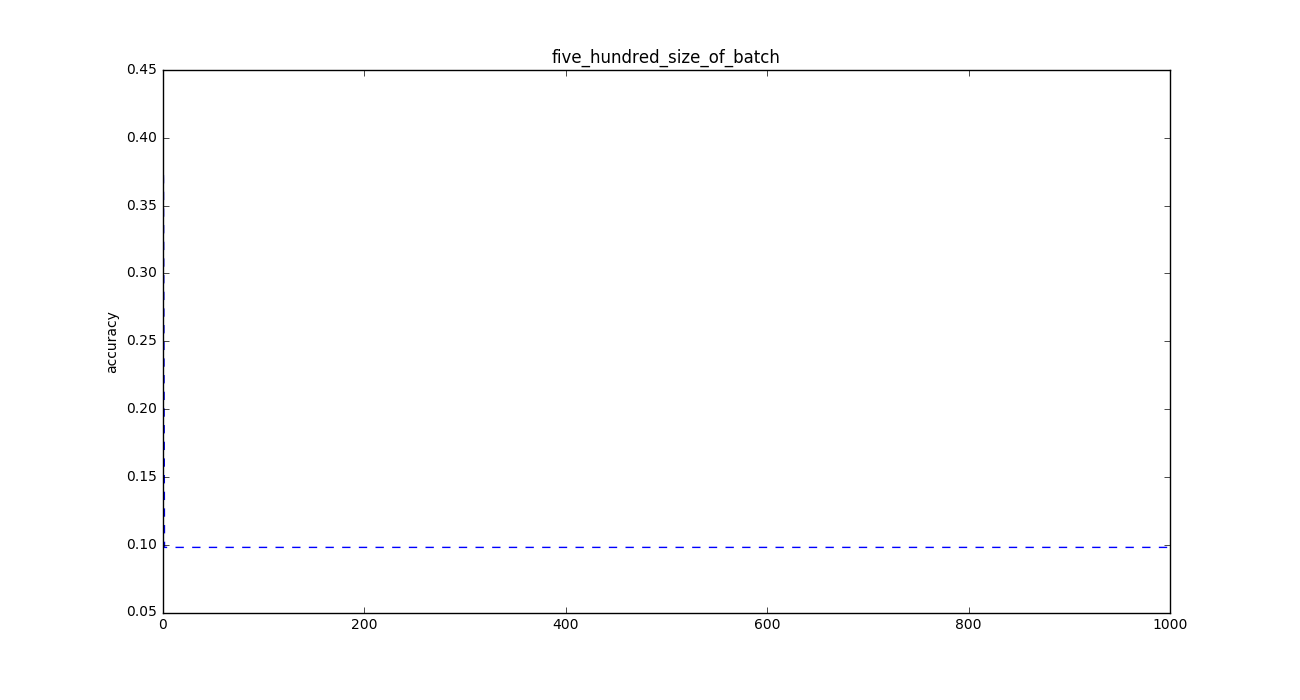
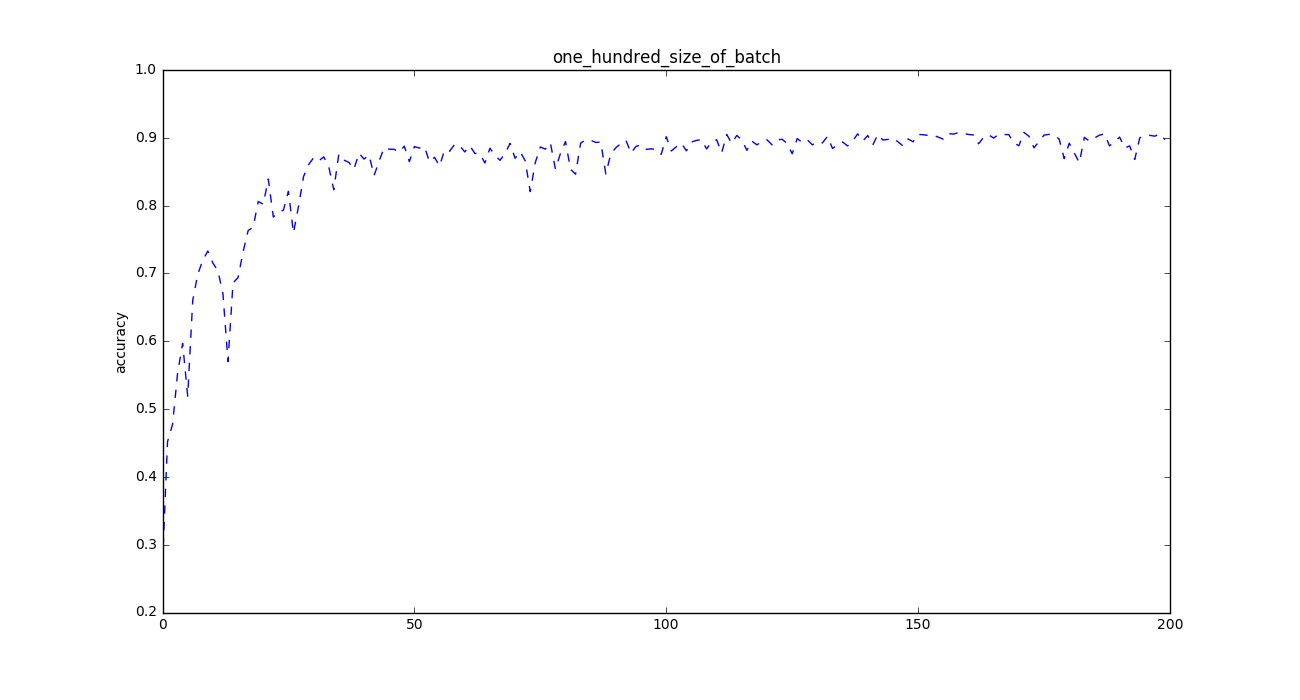
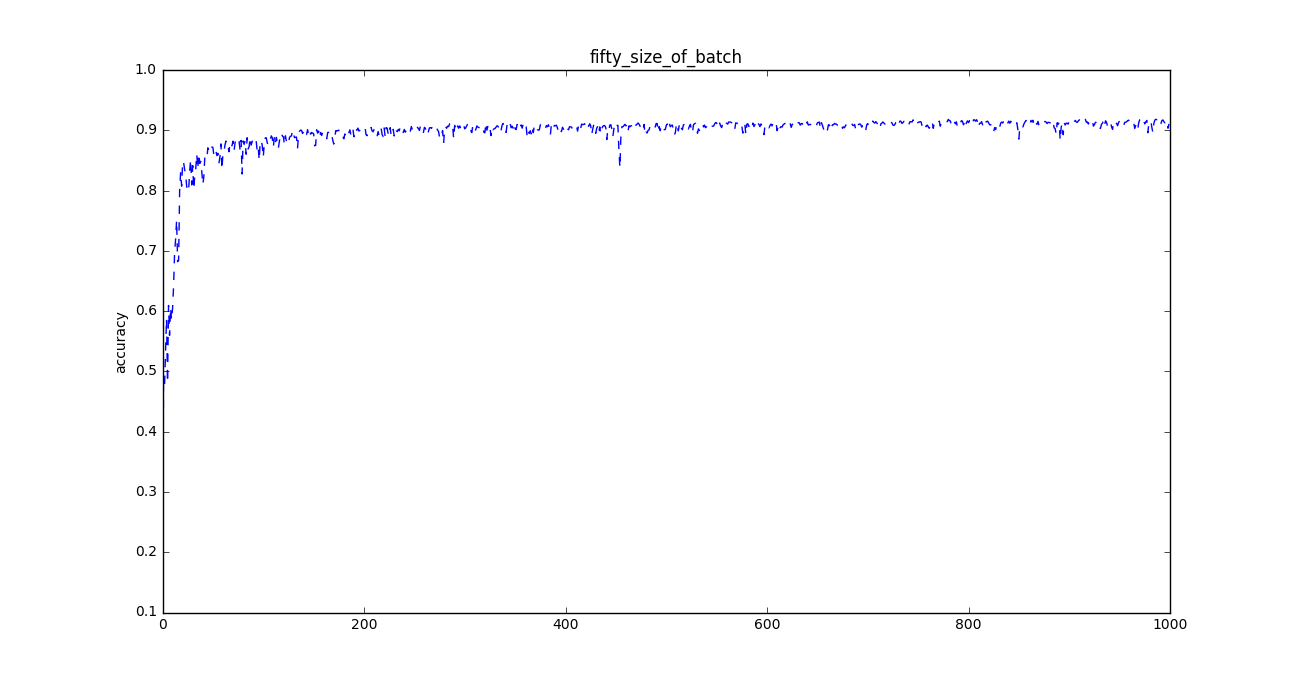
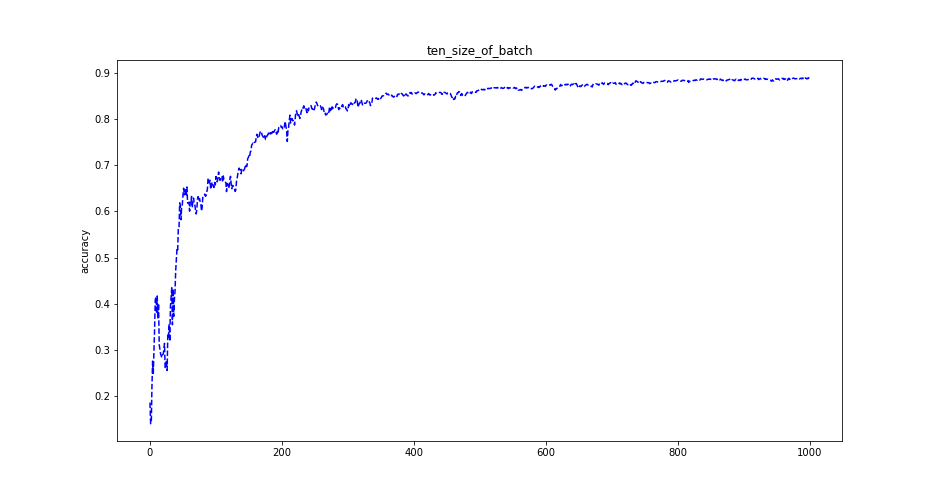


使用 cost + sigmoid



可以看出选择不同的损失函数和激活函数的组合会产生不一样的结果。交叉熵和sigmoid的组合导致最后正确率不高，其他的可以保持90%的准确率。使用cross\_entropy + softmax的收敛速度是最快的，但同时收敛后也是最不稳定的。

对于每次训练所使用的batch 的大小也会对训练的收敛速度还有精确度有重要的影响。如下面四张图，分别选用10,50,100,500的batch 大小进行1000次的训练。



可以看出，batch = 50 的收敛速度是最快的，同时准确率是最高的。当batch过小的时候，可以看出当batch=10 的时候，学习比较慢，而且准确率也不够好，没超过0.9，可以得出原因是数据量太小导致学习出的模型不够准确。当batch过大的时候，可以看出当batch=500 的时候，已经完全偏离了轨道，准确率只有10%相当于10个数中随机选择一个，完全没有效果，原因可能是batch=500，导致每次训练相同的数据概率很大，导致不具有广泛性，导致不能很好的识别测试数据。

最后，对于学习率，训练的步数等参数的调整对于模型训练的优化具有重要的意义，如果学习率过小，会导致学习速度较慢，学习率太大会导致难以调到正确的参数。

## 实验总结：

Tensorflow 是一个很好的开发深度学习的框架，独特的图计算可以提高计算的效率，可以方便理解计算的过程。封装的一系列函数库可以很方便的使用。几句简短的代码就实现了MINST数据集的识别。

在深度学习中，计算结构的构建、各种函数的选择，还有找到合适的参数取值对于训练的模型的好坏具有重大影响。