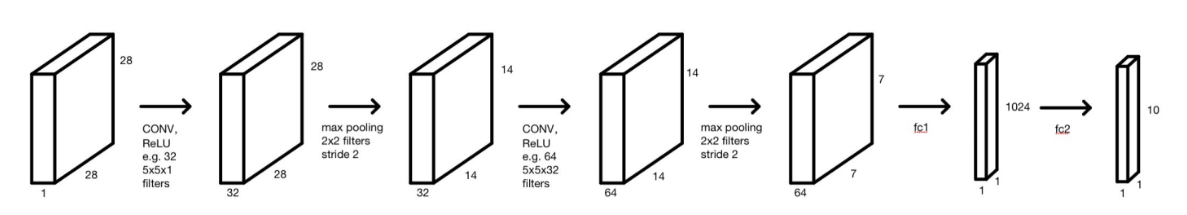
**CNN课程报告**

学号：20307110315

姓名：周训哲

# 1.任务描述

利用CNN卷积神经网络算法实现手写数字的识别与分类。解决了多样化模糊图片的识别分类问题。其中网络结构为：输入→卷积(32)→池化→卷积(64)→池化→全连接(1024)→输出(10)



损失函数主要运用了CrossEntropyLoss的损失函数，优化函数采用梯度下降和随机梯度下降算法。

# 2.数据集描述

**采用何种数据集开展实验？**

利用MNIST手写数字集作为学习以及测试的数据集。里面包含了图片集作为识别对象，标签集作为学习判断的标准。

**数据集有什么特点？**

MNIST数据集来自美国国家标准与技术研究所, National Institute of Standards and Technology (NIST)。训练集 (training set) 由来自 250 个不同人手写的数字构成, 其中 50% 是高中学生，50% 来自人口普查局 (the Census Bureau) 的工作人员。测试集(test set) 也是同样比例的手写数字数据。具有极大的测试数据量。

数据集分为训练集和测试集两个部分，其中又细分为图片集和标签集。

图片集主要存储的是图像信息，每一张图片包含28\*28个像素，在数据集中展开转换为一个长度为784的向量数据。在Mnist的训练数据集中mnist.train.images是一个形状为[60000, 784]的张量，第一个维度数字用来索引图片，第二个维度数字用来索引每张图片中的像素点，图片里的某个像素的强度值介于0-1之间。

标签集主要存储的是分类的信息，将每张图片所表示的数字进行存储。Mnist数据集中所有的标签mnist.train.labels是一个[60000, 10]的数字矩阵。在训练集中主要用来学习，以及kNN算法中的分类（利用距离）。在测试集中主要用来对比计算算法的识别准确率。

**数据集的各项统计量是多少？**

训练集中的数据量有60000个，测试集中数据量有10000个。

# 3.方法介绍

## 3.1数据预处理（如有）

**从原始数据到输入模型的数据之间，经过了哪些处理？这些处理有何作用？**

首先，原始数据为ubyte二进制文件，需要对二进制数据进行处理。需要利用open（）函数打开，其中参数为 ‘rb’，其含义为将文件以二进制文件形式打开。

对于图像数据（image）是一个3维的数据集（60000\*28\*28）。开头为信息标识，有四个变量，魔数（magic）为2051，数据集大小（size）为60000，图像的行、列都是28，均通过unpack（）函数读取。然后将图像数据从28\*28的二维数据进行转换为长度为748的一维向量images。

对于标签数据（label）是一个一维的数据集（60000），开头也是信息标识，有两个变量，魔数（magic）为2049，数据集大小（size）为60000，也是通过unpack（）函数读取，最后利用append（）函数，将标签存储在向量labels中。

## 3.2算法描述

**核心思想：**

卷积神经网络是一种特殊的深层的神经网络模型，它的特殊性体现在两个方面，一方面它的神经元间的连接是非全连接的， 另一方面同一层中某些神经元之间的连接的权重是共享的（即相同的）。它的非全连接和权值共享的网络结构使之更类似于生物神经网络，降低了网络模型的复杂度（对于很难学习的深层结构来说，这是非常重要的），减少了权值的数量。

经过卷积后的目标图像中的每一个像素点都对应了原图像中的局部特征，使用不同的卷积核，就可以从原图像中提取出不一样的局部特征了。计算目标图像的方式，是将卷积核在原图像上滑动，与各个部分的图像分别计算卷积直到遍历全部的原图像。接着，如果我们使用不止一个卷积核，那显而易见的，就可以提取出图像中不同的特征了。因此，计算得出的隐藏层的层数，与选择使用的卷积核的个数是一致的。

我们可以通过不同的卷积核来提取出图像中的不同特征。而我们使用卷积神经网络的过程，实际上是通过训练来确定各个卷积核的权重，从而提取出最适合的图像特征。使用不同长度大小的卷积核，在训练卷积神经网络的过程中，可以帮助我们提取出最优的特征来组成feature maps。之后再结合全连接神经网络来完成相关的图像分类聚合问题，就可以实现图像的分类。

**具体实现：**

**输入：** 28\*28的手写字体图片，这些手写字体包含0~9数字，也就是10个输出类

**输出：** 分类结果，0~9之间的一个数

因此我们可以知道，这是一个多分类问题，总共有十个类，因此神经网络的最后输出层必然是SoftMax问题，最后的神经元的个数是10个。

仿照LeNet-5网络结构的思路，可以设计一个卷积层为2\*2的卷积神经网络。

输入层：28\*28的图片，也就是相当于784个神经元

**C1层**：选择32个特征卷积核，然后卷积核大小选择5\*5，步幅为1，填充为“SAME”即与之前相同，这样我们可以得到32个特征图，然后每个特征图的大小为28，也就是神经元的个数为32\*28\*28=32\*784。

**S2层**： 下采样层，使用最大池化（选择最大的值）进行下采样，池化的size，选择(2,2)，也就是相当于对C1层28\*28的图片，进行分块，每个块的大小为2\*2，这样我们可以得到14\*14个块，然后我们统计每个块中，最大的值作为下采样的新像素，因此我们可以得到S1结果为：14\*14大小的图片，共有32个这样的图片。

**C3层**： 卷积层，这一层我们选择卷积核的大小依旧为5\*5，步幅仍然为1，填充为“SAME”，经过一次pooling，将图片个数翻倍，得到32\*14\*14的图片。

**S4层**：下采样层，对C3的64张14\*14的图片进行最大池化，池化块的大小为2\*2。因此最后S4层为64张大小为7\*7的图片。

**全连接层**：至此，神经元网络已被化简为64\*7\*7的图片，可以直接用全连接进行连接：

第一层为64\*7\*7->1024，第二层为1024->10，使用线性函数进行连接。

**损失函数**采用交叉熵，优化函数采用梯度下降和随机梯度下降分别进行不同实现方法的测试。

模型训练采用正向+反向+优化的bp算法，对模型进行训练优化。

反向bp的激活函数采用relu函数。

误差逆传播算法即BP算法。BP算法基于梯度下降策略，以目标的负梯度的方向对参数调整，给定学习率，控制着算法每一轮迭代中更新步长，若太大则容易振荡；若太小则收敛速度太慢）。

在BP算法训练的过程：

**1将输入示例提供给输入神经元；**

**2对输入数据进行计算，并逐层向前传播，产生输出结果；**

**3计算输出结果的误差；**

**4将误差逆向传播至隐层神经元；**

**5根据误差，调整参数（包括连接权和阈值），从2开始反复迭代，直到满足条件。**

**整体流程：**

总体来说，CNN分类算法包括以下4个步骤：

①准备数据，对数据进行预处理。

②定义CNN，设置网络结构

③训练神经网络，不断迭代得到最优参数

④对测试集进行回测，计算准确率

**缺点：**

**反向传播**

反向传播算法并不是一个深度学习中的高效算法，因为它对数据量的需求很大。

**平移不变性**

当我们说平移不变性时，我们意思是，稍微改变同一物体的朝向或位置，可能并不会激活那些识别该物体的神经元。假如一个神经元是用来识别一只猫的，其参数会随着猫的位置和转动的变化而变化。虽然数据扩增(data augmentation)在一定程度上解决了这个问题，但这个问题并没有得到彻底的解决。

**池化层**

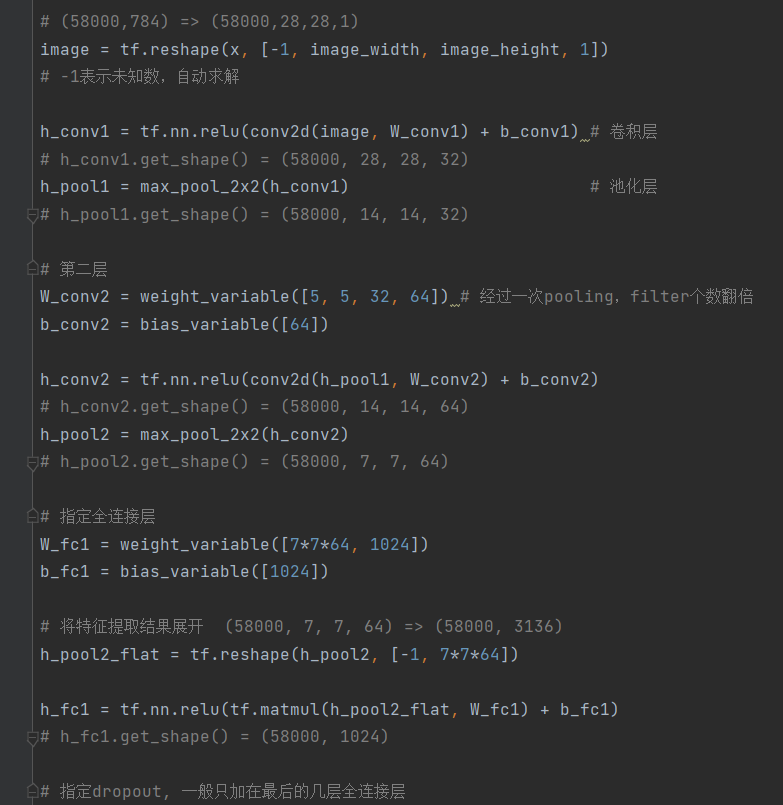
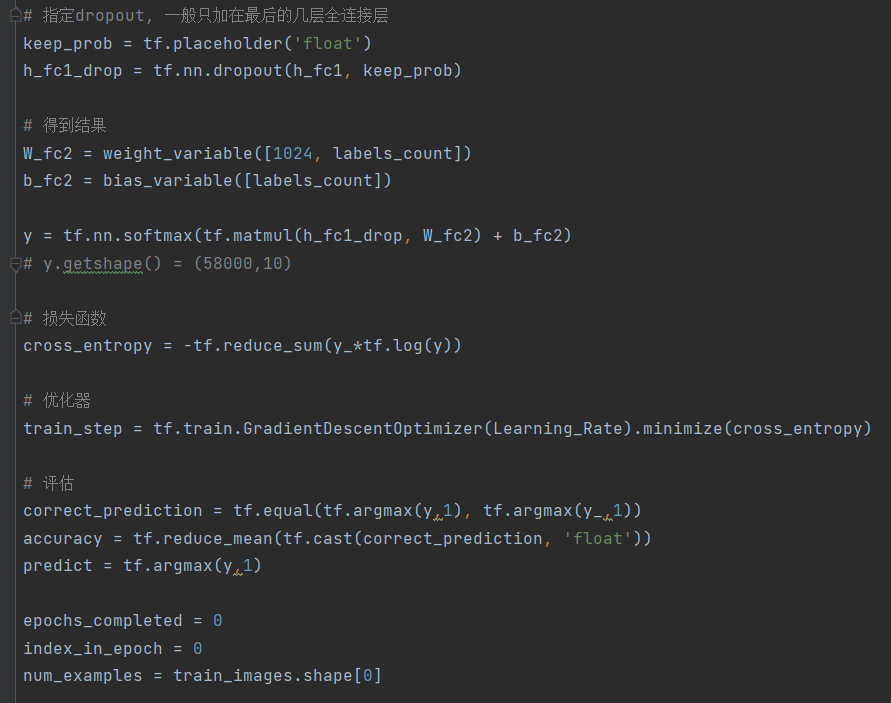
池化层的存在会导致许多非常有价值的信息的丢失，同时也会忽略掉整体与部分之间的关联。比如，当我们使用脸部探测器时，我们必须把一些特征结合在一起（嘴，双眼，脸的轮廓和鼻子）才能识别这张脸。CNN的判别方式是，如果这5个特征同时呈现的概率很高，那这就将是一张人脸。 合并图层，合并层是一个很大的错误，因为它丢失了大量有价值的信息，如果我们正在谈论一个人脸识别器，它会忽略部分和整体之间的关系，所以我们必须结合一些特征（嘴巴，双眼，椭圆的脸和鼻子）才能说明是一张脸。

**实现方式：**

**主要有两种实现方式：一种是用tensorflow实现，采用梯度下降优化；另一种是使用torch实现，采用随机梯度下降算法优化。**

**Tensorflow：**

****

**** ****

**Torch：**

**** ****

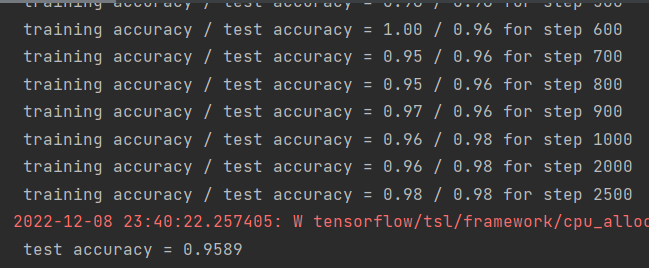
# 4.实验结果分析

## 4.1评价指标

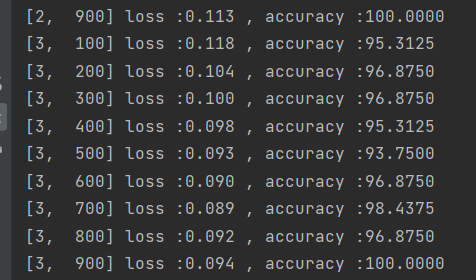
该算法训练集学习之后，对测试集样例的图像文件进行识别与预测，将预测结果与标签集比对，最后计算得到10000个测试数据的准确率。

## 4.2定量评价结果

**使用tensorflow库和梯度下降算法：**



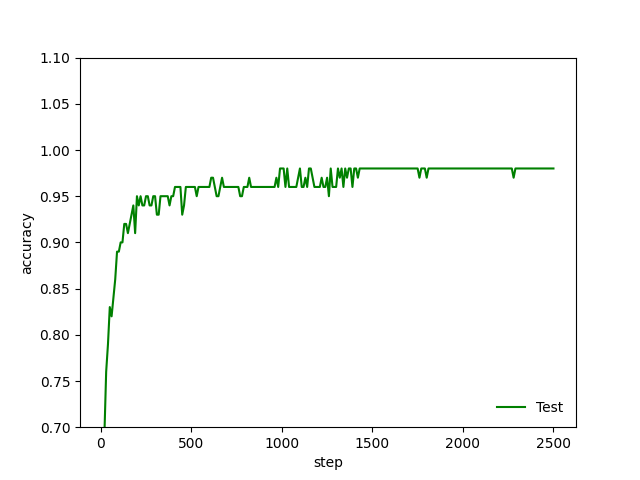
**使用torch库和随机梯度下降算法：**



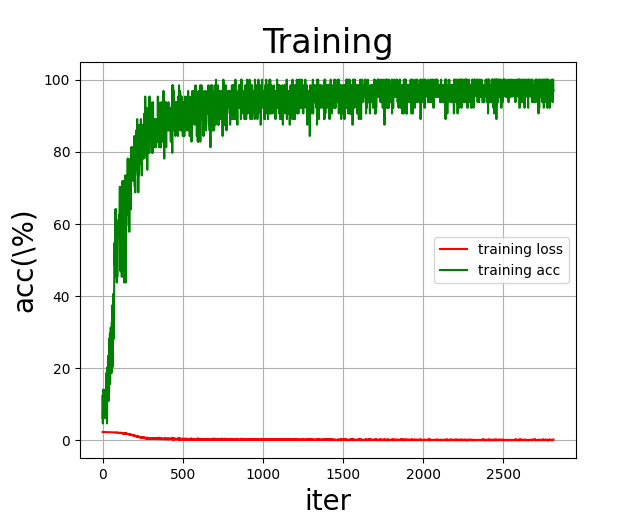
比较两种损失函数的损失以及其准确率，发现最终学习的结果准确率都可以达到96%但是随机梯度下降算法很好的体现了其特性，即随机性，即每次测试的结果准确率趋向增加，但是不是一个完全增的过程，而是有明显波动。而梯度下降算法则是测试准确率不断提升，直到最后达到96%的准确率。

## 4.3可视化结果

**使用tensorflow库和梯度下降算法：**



**使用torch库和随机梯度下降算法：**



可以看见，随着迭代次数的增加，准确率不断上升，误差损失不断下降，说明CNN学习有明显效果。

# 总结

本次实验主要学习了经典的卷积神经网络算法，在本次实验中，我学会了卷积神经网络算法中的卷积、池化等一系列的架构模型，以及其的具体工作原理以及如何使用代码实现。除此以外，我还尝试了解了卷积神经网络的不同损失函数以及不同库函数的不同实现方法所带来的结果差异。

本次实验主要采用relu作为激活函数，一个原因是因为为了计算方便，节约时间，采用最简单的激活函数可以提高算法效率，另一方面，采用relu函数作为线性的分类函数，对于简单的分类来说，可能效果还要比非线性的分类函数效果更好（具体参考svm中的sigmoid）。同时采用softmax作为输出函数的原因也是因为，对于非线性多分类模型来说，softmax这种非线性模型效果可能更好。