手写数字识别

撰写人：微电子所-杨登天-202028015926089

目 录

手写数字识别

杨登天-微电子所-202028015926089

1、实验要求

1. 搭建 Tensorflow（或其他框架）环境;
2. 构建一个规范的卷积神经网络结构;
3. 在 MNIST 手写数字数据集上进行训练和评估，实现测试集准确率达到 98%及以上;

2、环境配置

Anaconda Navigator 1.10.0
Python 3.7.10 64-bit
Tensorflow 2.3.0
Visual Studio Code 1.56.2

3、实验分析

3.1 预处理模块

1) 数据导入——将数据从 tensorflow.examples.tutorials.mnist 模块中引入 mnist 数据，这是为了方便运用 read_data_sets 函数将图片显示结果调整为 10 位独热码形式，以便于在网络输出层设置 10 个输出节点进行对比，通过对比 10 个节点得到训练用误差等。

```
import numpy as np
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data",one_hot=True)
```

2) 图像分割——设定尺寸为 28 个单位长度、数据通道为 1、标签数目为 10（对应预测结果）

```
IMAGE_SIZE = 28
NUM_CHANNELS = 1   #黑白
BATCH_SIZE = None
NUM_LABELS = 10
```

3.2 网络层模块

1) 张量喂养部分的设置——训练数据、训练标签和用于降低过拟合 dropout 层的神经元失活概率

```
image = tf.placeholder(tf.float32, [None, 784])
label = tf.placeholder(tf.float32, [None, 10])
keep_prob = tf.placeholder(tf.float32)
x_image = tf.reshape(image,[-1,28,28,1])
```

2) 网络层内变量的设置——包括卷积层的权重、偏置和全连接层的权重、偏置。
3) 网络层的设置

第一层卷积层：二维卷积，卷积核为 5×5，步长为[1, 1, 1, 1]，采用“SAME”采样；用 ReLu 函数激活；最大池化，步长为[1, 2, 2, 1]，池化层[1, 2, 2, 1]，采用”SAME”采样。

第二层卷积层：二维卷积，卷积核为 5×5，步长为[1, 1, 1, 1]，采用“SAME”采样；用 ReLu 函数激活；最大池化，步长为[1, 2, 2, 1]，池化层[1, 2, 2, 1]，采用”SAME”采样。

第三层全连接层：1024 个节点，且用 ReLu 函数激活。

第四层输出层：先用 dropout 失活部分神经元，以避免出现过拟合的情况；之后采用 softmax 函数结算并预测结果。同时这部分完成误差的计算，此处采用交叉熵。

3.3 精度计算和训练模块

1) 精度计算模块

交叉熵

2) 训练模块

使用 Adam 优化器的误差反向传播方法

3.4 模型保存模块

模型保存模块负责将训练好的神经网络参数保存到 save，其程序如下

```
saver = tf.train.Saver()
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(15000):
        batch = mnist.train.next_batch(100)
        if i % 300 == 0:
            train_accuracy = accuracy.eval(feed_dict={image: batch[0], label: batch[1], keep_prob: 0.5})
            print('Step %d, training accuracy %g' % (i, train_accuracy))
            train_step.run(feed_dict={image: batch[0], label: batch[1], keep_prob: 0.5})
saver.save(sess, 'D:/AI computer system Lab/MyLab/DLcourse/Lab1/NewLab1_1/SAVE/model.ckpt')
```

```
acc = accuracy.eval(feed_dict={image: mnist.test.images, label: mnist.test.labels, keep_prob: 0.5})
    print('testing accuracy %g' % acc)
```

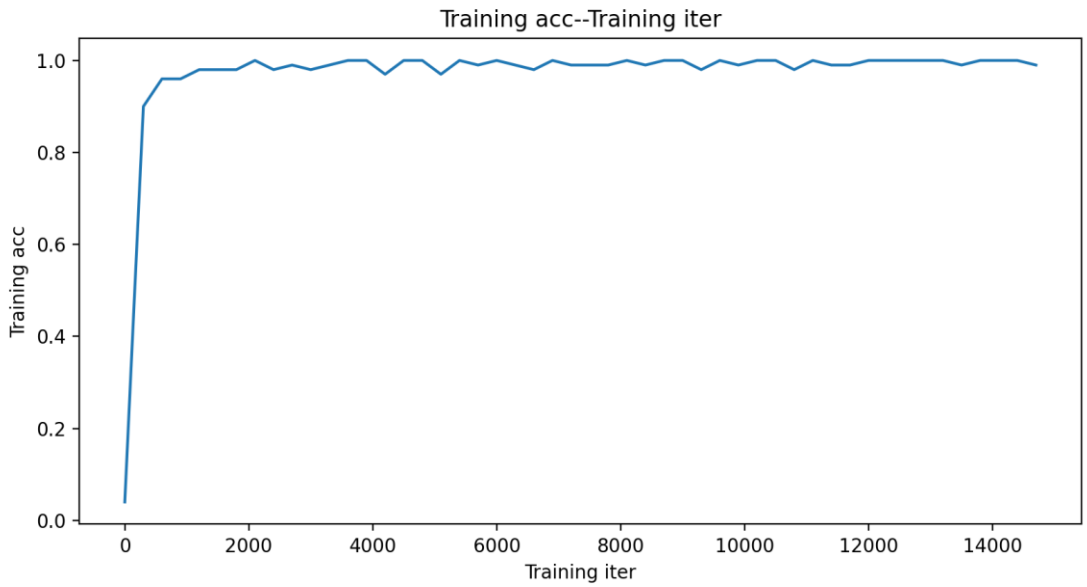
3.5 模型加载模块

通过模型加载对 mnist.validation 数据集进行验证

```
logs_train_dir = "D://AI computer system Lab/MyLab/DLcourse/Lab1/NewLab1_1/SAVE/"
saver = tf.train.Saver()
with tf.Session() as sess:
    ckpt = tf.train.get_checkpoint_state(logs_train_dir)
    #if ckpt and ckpt.model_checkpoint_path:
    global_step = ckpt.model_checkpoint_path.split('/')[-1].split('-')[-1]
    saver.restore(sess, ckpt.model_checkpoint_path)
    accuracy_score = sess.run(accuracy,feed_dict={image: mnist.validation.images,label: mnist.validation.labels,keep_prob: 0.5})
    print("validation accuracy is ",accuracy_score)
```

4. 实验结果

训练过程情况:



测试集准确率

```
2021-05-29 12:01:14.825051: W tensorflow/core/framework/allocator.cc:116] Allocation of 1003520000 exceeds 10% of system memory.
test accuracy is  0.9888
```

0.9888 > 0.98

完成实验!

5. 实验心得

随便唠叨两句吧，整个程序前前后后跑了 4 次，每次都是运行 3 天到 5 天，第一次体会用 tensorflow 跑程序的艰辛，不过也是刻意去追求这种体验，想看看这么大的计算量只用 CPU 运行会怎么样。过程很艰难，心也一直都吊着，但是结果起码在失败了三次以后获得了成功，好在有两台笔记本在同时运行。

本次实验针对手写数据识别问题，本人参照所给的实验指导文件并借助网上资料完成对程序的编写，并获得成功，前后五次分别得到 95.31%、96.72%、97.14%、99.12%，除了最后一次不再期待继续运行以得到更高一点的正确率，其余都是在上下波动之后取得相对中间的数值。主要是并不清楚偌大的卷积神经网络参数可调节关键参数都包括哪些，我原先使用梯度下降优化器，但是取得的效果无论是正确率还是收敛速度都并不理想，后来想到改成 Adam 优化器试试，神奇的事情总是这样发生的！

本次实验所获心得是大体上学懂网络层的设定、体验了单纯用 CPU 运行张量计算程序的过程，尤其是学习网络层内各类函数及其参数选取，特别感谢 tensorflow 官网给出的教程示例。