学号：201822090530 姓名：李欣芮

前向传播

#当如相应模块

import tensorflow as tf

#输入的图片训练集为784维

INPUT\_NODE=784

#训练集的标签为10维，10分类，10个数索引号的概率

OUTPUT\_NODE=10

#定义了隐藏的节点个数

LAYER1\_NODE=500

#定义生成权重函数

def get\_weight(shape,regularizer,name):

w=tf.Variable(tf.truncated\_normal(shape,stddev=0.1),name=name)

#加入正则化，参数惩罚

if regularizer !=None:tf.add\_to\_collection('losses',tf.contrib.layers.l2\_regularizer(regularizer)(w))

return w

#定义生成偏置函数

def get\_bias(shape,name):

b=tf.Variable(tf.zeros(shape),name=name)

return b

def forward(x,regularizer):

#第一层隐藏层

w1=get\_weight([INPUT\_NODE,LAYER1\_NODE],regularizer,'w1')

b1=get\_bias([LAYER1\_NODE],'b1')

y1=tf.nn.relu(tf.matmul(x,w1)+b1)

#第二层隐藏层

w2=get\_weight([LAYER1\_NODE,OUTPUT\_NODE],regularizer,'w2')

b2=get\_bias([OUTPUT\_NODE],'b2')

y=tf.matmul(y1,w2)+b2

#返回结果

return y

后向传播：

#导入相应的模块

import tensorflow as tf

#导入并下载mnist数据集

from tensorflow.examples.tutorials.mnist import input\_data

#导入前向传播模块

import minist\_forward

#导入系统模块，系统调用（例如打开关闭写入读取文件等操作）

import os

#一次喂如多少张图片

BATCH\_SIZE=200

#初始学习率

LEARNING\_RATE\_BASE=0.1

#学习衰减率

LEARNING\_RATE\_DECAY=0.99

#正则化系数

REGULARIZER=0.0001

#训练的轮数

STEPS=50000

#滑动平均衰减率

MOVING\_AVERAGE\_DECAY=0.99

#模型存储路径

MODEL\_SAVE\_PATH="./model/"

#模型名称

MODEL\_NAME="mnist\_model"

def backward(mnist):

#生成输入占位，标准输出占位

x=tf.placeholder(tf.float32,shape=[None,minist\_forward.INPUT\_NODE])

y\_=tf.placeholder(tf.float32,shape=[None,minist\_forward.OUTPUT\_NODE])

#前向计算

y=minist\_forward.forward(x,REGULARIZER)

#轮数计数器

global\_step=tf.Variable(0,trainable=False)

#损失函数(这里求交叉熵，对标准值)

ce=tf.nn.sparse\_softmax\_cross\_entropy\_with\_logits(logits=y,labels=tf.arg\_max(y\_,1))

#求标平均值

cem=tf.reduce\_mean(ce)

with tf.name\_scope('name'):

#正则化后的losse值

loss=cem+tf.add\_n(tf.get\_collection("losses"))

tf.summary.scalar('loss',loss)

#学习率的指数衰减

learning\_rate=tf.train.exponential\_decay(LEARNING\_RATE\_BASE,global\_step,mnist.train.num\_examples/BATCH\_SIZE,LEARNING\_RATE\_DECAY,staircase=True)

#定义训练过程

train\_step=tf.train.GradientDescentOptimizer(learning\_rate).minimize(loss,global\_step=global\_step)

#定义滑动平均

ema=tf.train.ExponentialMovingAverage(MOVING\_AVERAGE\_DECAY,global\_step)

ema\_op=ema.apply(tf.trainable\_variables())

with tf.control\_dependencies([train\_step,ema\_op]):

train\_op=tf.no\_op(name='train')

#实例化Saver

saver=tf.train.Saver()

#建立计算会话

with tf.Session() as sess:

#初始化所有变量

init\_op=tf.global\_variables\_initializer()

sess.run(init\_op)

##############################################################################

#断点续接功能

ckpt=tf.train.get\_checkpoint\_state(MODEL\_SAVE\_PATH)

if ckpt and ckpt.model\_checkpoint\_path:

saver.restore(sess,ckpt.model\_checkpoint\_path)

##############################################################################

#开始训练

for i in range(STEPS):

#文件中读取的mnist从它的下一组训练样本开始

xs,ys=mnist.train.next\_batch(BATCH\_SIZE)

#训练结果赋值

\_, loss\_value, step, b=sess.run([train\_op ,loss ,global\_step,'b2:0'], feed\_dict={x:xs,y\_:ys})

#每训练1000轮打印一次训练的轮数和误差值，并且保存当前训练的情况

if i % 1000 == 0:

print("After %d training step(s), loss on training batch is %g." % (step,loss\_value))

#存储器，进行存储，将当前的计算模型

saver.save(sess,os.path.join(MODEL\_SAVE\_PATH, MODEL\_NAME), global\_step=global\_step)

writer=tf.summary.FileWriter("logs/",sess.graph)

#定义主函数

def main():

mnist=input\_data.read\_data\_sets("./data/",one\_hot=True)

backward(mnist)

#定义函数入口

if \_\_name\_\_=='\_\_main\_\_':

main()

测试：

import time

import tensorflow as tf

from tensorflow.examples.tutorials.mnist import input\_data

import minist\_forward

import minist\_backward

TEST\_INTERVAL\_SECS=10

def test(mnist):

#使用到Graph.as\_default() 的上下文管理器（ context manager），它能够在这个上下文里面覆盖默认的图

with tf.Graph().as\_default() as g:

#生成测试集的占位

x=tf.placeholder(tf.float32,shape=[None,minist\_forward.INPUT\_NODE])

y\_=tf.placeholder(tf.float32,shape=[None,minist\_forward.OUTPUT\_NODE])

y=minist\_forward.forward(x,None)

#实例化影子容器，容器中存储着影子值得地址

ema=tf.train.ExponentialMovingAverage(minist\_backward.MOVING\_AVERAGE\_DECAY)

#将影子值和本身变量进行绑定，本身变量名指向影子值的地址

ema\_restore=ema.variables\_to\_restore()

#saver加载器加载的是变量的内存

saver=tf.train.Saver(ema\_restore)

#准确预测值当相

correct\_prediction=tf.equal(tf.argmax(y,1),tf.argmax(y\_,1))

with tf.name\_scope('accuracy'):

#tf.cast转换数据类型

accuracy=tf.reduce\_mean(tf.cast(correct\_prediction,tf.float32))

tf.summary.scalar('accuracy',accuracy)

while True:

with tf.Session() as sess:

#加载训练好的模型

ckpt=tf.train.get\_checkpoint\_state(minist\_backward.MODEL\_SAVE\_PATH)

if ckpt and ckpt.model\_checkpoint\_path:

print(ckpt.model\_checkpoint\_path)

#恢复会话

saver.restore(sess,ckpt.model\_checkpoint\_path)

#恢复轮数

global\_step=ckpt.model\_checkpoint\_path.split('/')[-1].split('-')[-1]

#计算准确率

accuracy\_score=sess.run(accuracy,feed\_dict={x:mnist.test.images,y\_:mnist.test.labels})

#打印提示

print("after %s training step(s), test accuracy = %g" % (global\_step,accuracy\_score))

# print("The current training steps' w2 is:", sess.run(g.get\_tensor\_by\_name("w2:0")))

# print("The current training steps' b2 is:", sess.run(g.get\_tensor\_by\_name("b2:0")))

writer=tf.summary.FileWriter("./logs1/",sess.graph)

else:

print("No checkpoint file found!")

return

#引入时间线程，睡眠5秒

time.sleep(TEST\_INTERVAL\_SECS)

#定义主函数

def main():

#对于数据集采用one\_hot格式

mnist=input\_data.read\_data\_sets("./data/",one\_hot=True)

test(mnist)

#定义函数入口

if \_\_name\_\_=='\_\_main\_\_':

main()