TensorFlow 实现基于 CNN 数字识别的代码

任务时间：时间未知

前期准备

TensorFlow 相关 API 可以到在实验 [TensorFlow - 相关 API](https://cloud.tencent.com/developer/labs/lab/10187) 中学习。

训练数据下载：

wget https://devlab-1251520893.cos.ap-guangzhou.myqcloud.com/t10k-images-idx3-ubyte.gz

wget https://devlab-1251520893.cos.ap-guangzhou.myqcloud.com/t10k-labels-idx1-ubyte.gz

wget https://devlab-1251520893.cos.ap-guangzhou.myqcloud.com/train-images-idx3-ubyte.gz

wget https://devlab-1251520893.cos.ap-guangzhou.myqcloud.com/train-labels-idx1-ubyte.gz

CNN 模型构建

**示例代码：**

现在您可以在 /home/ubuntu 目录下创建源文件 mnist\_model.py，内容可参考：

**示例代码：/home/ubuntu/mnist\_model.py**

#!/usr/bin/python

# -\*- coding: utf-8 -\*

from \_\_future\_\_ import absolute\_import

from \_\_future\_\_ import division

from \_\_future\_\_ import print\_function

import argparse

import sys

import tempfile

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

import tensorflow as tf

FLAGS = None

def deepnn(x):

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

x\_image = tf.reshape(x, [-1, 28, 28, 1])

#第一层卷积层，卷积核为5\*5，生成32个feature maps.

with tf.name\_scope('conv1'):

W\_conv1 = weight\_variable([5, 5, 1, 32])

b\_conv1 = bias\_variable([32])

h\_conv1 = tf.nn.relu(conv2d(x\_image, W\_conv1) + b\_conv1) #激活函数采用relu

# 第一层池化层，下采样2.

with tf.name\_scope('pool1'):

h\_pool1 = max\_pool\_2x2(h\_conv1)

# 第二层卷积层，卷积核为5\*5，生成64个feature maps

with tf.name\_scope('conv2'):

W\_conv2 = weight\_variable([5, 5, 32, 64])

b\_conv2 = bias\_variable([64])

h\_conv2 = tf.nn.relu(conv2d(h\_pool1, W\_conv2) + b\_conv2)#激活函数采用relu

# 第二层池化层，下采样2.

with tf.name\_scope('pool2'):

h\_pool2 = max\_pool\_2x2(h\_conv2)

#第一层全连接层，将7x7x64个feature maps与1024个features全连接

with tf.name\_scope('fc1'):

W\_fc1 = weight\_variable([7 \* 7 \* 64, 1024])

b\_fc1 = bias\_variable([1024])

h\_pool2\_flat = tf.reshape(h\_pool2, [-1, 7\*7\*64])

h\_fc1 = tf.nn.relu(tf.matmul(h\_pool2\_flat, W\_fc1) + b\_fc1)

#dropout层，训练时候随机让某些隐含层节点权重不工作

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

keep\_prob = tf.placeholder(tf.float32)

h\_fc1\_drop = tf.nn.dropout(h\_fc1, keep\_prob)

# 第二层全连接层，1024个features和10个features全连接

with tf.name\_scope('fc2'):

W\_fc2 = weight\_variable([1024, 10])

b\_fc2 = bias\_variable([10])

y\_conv = tf.matmul(h\_fc1\_drop, W\_fc2) + b\_fc2

return y\_conv, keep\_prob

#卷积

def conv2d(x, W):

return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

#池化

def max\_pool\_2x2(x):

return tf.nn.max\_pool(x, ksize=[1, 2, 2, 1],

strides=[1, 2, 2, 1], padding='SAME')

#权重

def weight\_variable(shape):

initial = tf.truncated\_normal(shape, stddev=0.1)

return tf.Variable(initial)

#偏置

def bias\_variable(shape):

initial = tf.constant(0.1, shape=shape)

return tf.Variable(initial)

训练 CNN 模型

**示例代码：**

现在您可以在 /home/ubuntu 目录下创建源文件 train\_mnist\_model.py，内容可参考：

**示例代码：/home/ubuntu/train\_mnist\_model.py**

#!/usr/bin/python

# -\*- coding: utf-8 -\*

from \_\_future\_\_ import absolute\_import

from \_\_future\_\_ import division

from \_\_future\_\_ import print\_function

import argparse

import sys

import tempfile

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

import tensorflow as tf

import mnist\_model

FLAGS = None

def main(\_):

mnist = input\_data.read\_data\_sets(FLAGS.data\_dir, one\_hot=True)

#输入变量，mnist图片大小为28\*28

x = tf.placeholder(tf.float32, [None, 784])

#输出变量，数字是1-10

y\_ = tf.placeholder(tf.float32, [None, 10])

# 构建网络，输入—>第一层卷积—>第一层池化—>第二层卷积—>第二层池化—>第一层全连接—>第二层全连接

y\_conv, keep\_prob = mnist\_model.deepnn(x)

#第一步对网络最后一层的输出做一个softmax，第二步将softmax输出和实际样本做一个交叉熵

#cross\_entropy返回的是向量

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

cross\_entropy = tf.nn.softmax\_cross\_entropy\_with\_logits(labels=y\_,

logits=y\_conv)

#求cross\_entropy向量的平均值得到交叉熵

cross\_entropy = tf.reduce\_mean(cross\_entropy)

#AdamOptimizer是Adam优化算法：一个寻找全局最优点的优化算法，引入二次方梯度校验

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

train\_step = tf.train.AdamOptimizer(1e-4).minimize(cross\_entropy)

#在测试集上的精确度

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

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

correct\_prediction = tf.cast(correct\_prediction, tf.float32)

accuracy = tf.reduce\_mean(correct\_prediction)

#将神经网络图模型保存本地，可以通过浏览器查看可视化网络结构

graph\_location = tempfile.mkdtemp()

print('Saving graph to: %s' % graph\_location)

train\_writer = tf.summary.FileWriter(graph\_location)

train\_writer.add\_graph(tf.get\_default\_graph())

#将训练的网络保存下来

saver = tf.train.Saver()

with tf.Session() as sess:

sess.run(tf.global\_variables\_initializer())

for i in range(5000):

batch = mnist.train.next\_batch(50)

if i % 100 == 0:

train\_accuracy = accuracy.eval(feed\_dict={

x: batch[0], y\_: batch[1], keep\_prob: 1.0})#输入是字典，表示tensorflow被feed的值

print('step %d, training accuracy %g' % (i, train\_accuracy))

train\_step.run(feed\_dict={x: batch[0], y\_: batch[1], keep\_prob: 0.5})

test\_accuracy = 0

for i in range(200):

batch = mnist.test.next\_batch(50)

test\_accuracy += accuracy.eval(feed\_dict={x: batch[0], y\_: batch[1], keep\_prob: 1.0}) / 200;

print('test accuracy %g' % test\_accuracy)

save\_path = saver.save(sess,"mnist\_cnn\_model.ckpt")

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

parser = argparse.ArgumentParser()

parser.add\_argument('--data\_dir', type=str,

default='./',

help='Directory for storing input data')

FLAGS, unparsed = parser.parse\_known\_args()

tf.app.run(main=main, argv=[sys.argv[0]] + unparsed)

**然后执行:**

cd /home/ubuntu;

python train\_mnist\_model.py

训练的时间会较长，可以喝杯茶耐心等待。

**执行结果：**

step 3600, training accuracy 0.98

step 3700, training accuracy 0.98

step 3800, training accuracy 0.96

step 3900, training accuracy 1

step 4000, training accuracy 0.98

step 4100, training accuracy 0.96

step 4200, training accuracy 1

step 4300, training accuracy 1

step 4400, training accuracy 0.98

step 4500, training accuracy 0.98

step 4600, training accuracy 0.98

step 4700, training accuracy 1

step 4800, training accuracy 0.98

step 4900, training accuracy 1

test accuracy 0.9862

测试 CNN 模型

**下载测试图片**

下载 test\_num.zip

cd /home/ubuntu

wget https://devlab-1251520893.cos.ap-guangzhou.myqcloud.com/test\_num.zip

**解压测试图片包**

解压 test\_num.zip，其中 1-9.png 为 1-9 数字图片。

unzip test\_num.zip

**实现 predict 代码**

现在您可以在 /home/ubuntu 目录下创建源文件 predict\_mnist\_model.py，内容可参考：

**示例代码：/home/ubuntu/predict\_mnist\_model.py**

#!/usr/bin/python

# -\*- coding: utf-8 -\*

from \_\_future\_\_ import absolute\_import

from \_\_future\_\_ import division

from \_\_future\_\_ import print\_function

import argparse

import sys

import tempfile

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

import tensorflow as tf

import mnist\_model

from PIL import Image, ImageFilter

def load\_data(argv):

grayimage = Image.open(argv).convert('L')

width = float(grayimage.size[0])

height = float(grayimage.size[1])

newImage = Image.new('L', (28, 28), (255))

if width > height:

nheight = int(round((20.0/width\*height),0))

if (nheigth == 0):

nheigth = 1

img = grayimage.resize((20,nheight), Image.ANTIALIAS).filter(ImageFilter.SHARPEN)

wtop = int(round(((28 - nheight)/2),0))

newImage.paste(img, (4, wtop))

else:

nwidth = int(round((20.0/height\*width),0))

if (nwidth == 0):

nwidth = 1

img = grayimage.resize((nwidth,20), Image.ANTIALIAS).filter(ImageFilter.SHARPEN)

wleft = int(round(((28 - nwidth)/2),0))

newImage.paste(img, (wleft, 4))

tv = list(newImage.getdata())

tva = [ (255-x)\*1.0/255.0 for x in tv]

return tva

def main(argv):

imvalue = load\_data(argv)

x = tf.placeholder(tf.float32, [None, 784])

y\_ = tf.placeholder(tf.float32, [None, 10])

y\_conv, keep\_prob = mnist\_model.deepnn(x)

y\_predict = tf.nn.softmax(y\_conv)

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

saver = tf.train.Saver()

with tf.Session() as sess:

sess.run(init\_op)

saver.restore(sess, "mnist\_cnn\_model.ckpt")

prediction=tf.argmax(y\_predict,1)

predint = prediction.eval(feed\_dict={x: [imvalue],keep\_prob: 1.0}, session=sess)

print (predint[0])

if \_\_name\_\_ == "\_\_main\_\_":

main(sys.argv[1])

然后执行:

cd /home/ubuntu;

python predict\_mnist\_model.py 1.png

**执行结果：**

1

你可以修改 1.png 为 1-9.png 中任意一个

完成

任务时间：时间未知