远程科研总结报告

史心怡

2018.3.3

目录

1.项目背景

最早提出“大数据”时代到来的是全球知名咨询公司麦肯锡，麦肯锡称：“数据，已经渗透到当今每一个行业和业务职能领域，成为重要的生产因素。人们对于海量数据的挖掘和运用，预示着新一波生产率增长和消费者盈余浪潮的到来。”正如其所言，随着近些年计算机和信息技术的迅猛发展和普及应用，大数据一词越来越多地被提及，不仅一度登上了《纽约时报》，《华尔街日报》的专栏封面，相关大学也纷纷成立了大数据专业。这都是因为人们越发地认识到了数据对于学术界，商界，政府的不可取代的重大意义。

大数据巨大的商业价值在 2016 年麦肯锡发表的《分析的时代：在大数据的世界竞争》中被详细地解释了。这份报告指出来自网络、智能手机、传感器、相机、支付系统以及其他途径的数据形成了一项资产，产生了巨大的商业价值。体现在企业当中，梅西百货的 SAS 系统运用大数据技术根据 7300 种货品的需求和库存实现实时定价，充分满足了顾客需求的同时也使得自己利润最大化；零售业寡头摩尔玛通过最新的搜索引擎 Polaris，利用语义数据技术使得在线购物的完成率提升了 10%到 15%。

大数据充斥着人类经济社会的角角落落，海量和多样化的信息资产使得大数据需要新的处理模式，才能为数据信息使用者提供有效的信息，使得企业洞察危险的能力增强，流程得以优化，决策更加准确。

正是因为大数据的 5V 特点，即 Volume(大量)、Velocity(高速)、Variety(多样)、Value(低价值密度)、Veracity(真实性)，它得以走上世界的重要舞台，成为新技术发展的焦点。但同时由于国内外对于大数据人才培养的时间并不长，因此技术市场上对于掌握大数据技术的人才一直是供不应求。这也正凸显出了我们对于这门学科进行学习与研究的重要价值。

这一次的项目我们就将运用大数据方面的知识与技能实现 mnist 数据集在容器中的运行与数据的保存。相信学以致用才能让我们更深刻地理解相关理论知识。

2.MNIST 的原理及实现

2.1 研究背景

MNIST 手写数字分类问题是在机器学习领域中的一个经典问题，我们试图从一张手写体数字的图片入手,将它识别出来并分类为 0 至 9 十个数字中的一个。在本项目中我们使用 MNIST 官方的数据集,它提供 28*28 的手写体数字灰度图片。

MNIST 数据集的官网是 Yann LeCun's website。我们从这里下载数据集到项目代码文件中。MNIST 官方数据集来自美国国家标准与技术研究所 National Institute of Standards and Technology (NIST),它包含了共 70,000 个样本(每个样本都由一张手写数字图片和一个标签),其中训练集 (training set) 包含样本 60,000 个,由 250 个不同的人手写的数字构成，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) 包含样本 10,000 个,也是同样比例的手写数字数据。这种训练-测试的样本切分非常重要,使得我们训练和测试机器学习模型时使用的数据相互更独立，使得我们的模型更有说服力,从而更加容易把设计的模型推广到其他数据集上,即所谓的"泛化"。

2.2 结构与原理

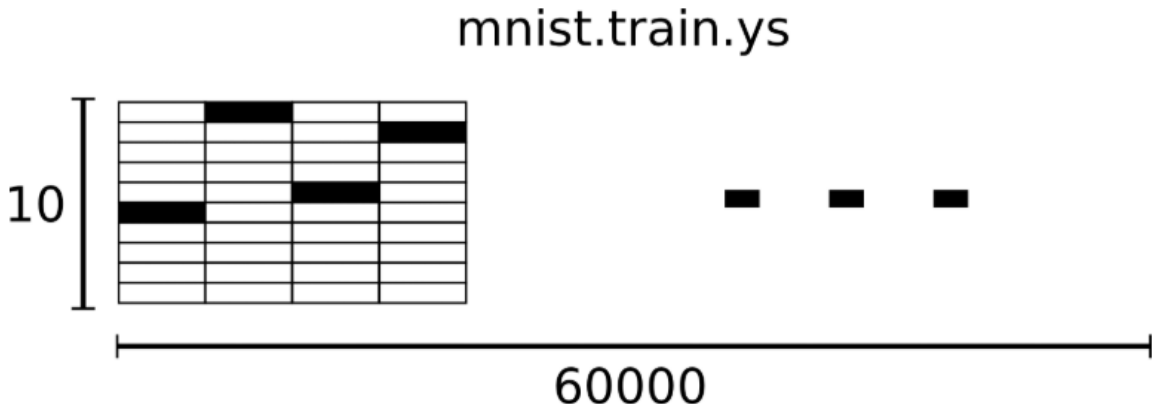
我们使用 python 语言来完成本次项目，运用 tensorflow 神经网络，softmax 方法来训练，调用我们的模型。

从官网上下载下来的数据集被分成两部分：60000 行的训练数据集（mnist.train）和 10000 行的测试数据集（mnist.test）。正如前面提到的一样，每一个 MNIST 数据单元有两部分组成：一张包含手写数字的图片和一个对应的标签。我们把这些图片设为"xs"，把这些标签设为"ys"。训练数据集和测试数据集都包含 xs 和 ys，比如训练数据集的图片是 mnist.train.images ，训练数据集的标签是 mnist.train.labels。

数据集中每一张图片大小都是 28X28 平方像素,我们可以用一个二维的数字数组来表示这张图片:

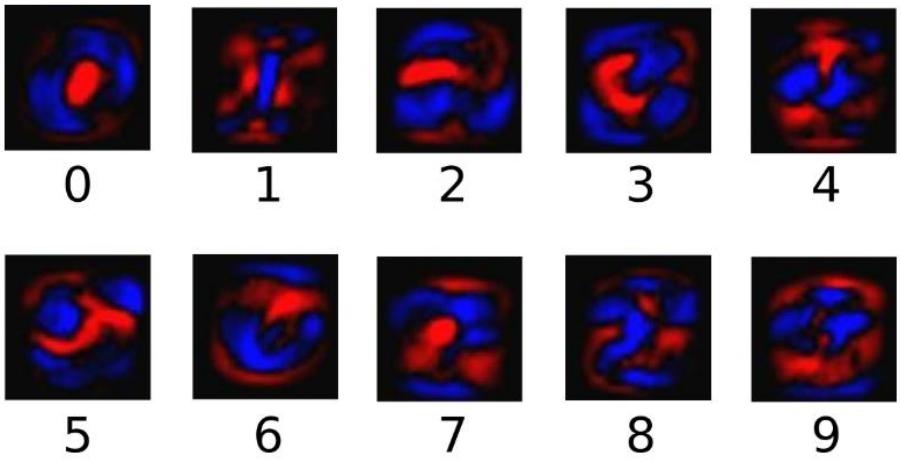
每个值的大小表明了对应像素点的灰度.由于手写体数字图片的结构比较复杂,像素的强度值会比较广泛地介于 0 和 1 之间，越到边缘灰度越低，值越趋近为 0。因此为了分析识别图片中的数字，我们需要将图片二度化，即低于一定灰度的值将它变为 0，而高于界线的就为 1。之后我们把这个数组展开成一个向量，使像素点们反映在这些数组展开的向量点上。Softmax 模型不关心这个数组如何展开（即数字间的顺序如何），但它要求我们保持各个图片采用相同的方式展开。因此我在 mnist_app 文件中加入了使提交图片二度化的代码。这样每次有图片上传之后，都将经过相同的二度化方式展开，也能使得模型正常运行。

数据集中图片的标签是介于 0 到 9 的数字，用来描述给定图片里表示的数字。在本项目中我们使标签数据是"one-hot vectors"即一个向量除了某一位的数字是 1 以外其余各维度数字都是 0。具体来说,数字 n 将表示成一个只有在第 n 维度（从 0 开始）数字为 1 的 10 维向量:



举个例子来说，标签 5 将表示成([0,0,0,0,0,1,0,0,0,0,0]),因此 mnist.train.labels 是一个 [60000, 10] 的数字矩阵。

MNIST 的每一张图片都表示一个数字，从 0 到 9。我们要实现手写数字的识别，就希望得到给定图片代表每个数字的概率，概率最高的自然就是识别的结果了。比如说，我们的模型可能推测一张包含 5 的图片代表数字 5 的概率是 80%但是判断它是 6 的概率是 10%（因为 5 和 6 在连笔或较为潦草的手写数字中十分相像），然后给予它代表其他数字的概率更小的值。这是一个使用 softmax 回归（softmax regression）模型的经典案例。第一步时我们为了得到一张给定图片属于某个特定数字类的证据（evidence），我们对图片像素值进行加权求和。如果这个像素具有很强的证据说明这张图片不属于该类，那么相应的权值为负数，相反如果这个像素拥有有利的证据支持这张图片属于这个类，那么权值是正数。这张图用红色表示负权值,蓝色表示正权值,通过 10 个具体例子展示了这点：



从图片中可以看出，这样的权值是会有一定的模糊与偏差的。因为输入往往会带有一些无关的干扰量

我们需要加入一个额外的偏置量（bias），因此对于给定的输入图片 x 它代表的是数字 i 的证据可以表示为：

$$\text{evidence}_i = \sum_j W_{i,j} x_j + b_i$$

其中 Wi 代表权重， bi 代表数字 i 类的偏置量，j 代表给定图片 x 的像素索引用于像素求和。然后用 softmax 函数可以把这些证据转换成概率 y：

$$y = \text{softmax}(\text{evidence})$$

这里的 softmax 可以看成是一个激励（activation）函数或者链接（link）函数，把我们定义的线性函数的输出转换成我们想要的格式，也就是关于 10 个数字类的概率分布。因此，给定一张图片，它对于每一个数字的吻合度可以被 softmax 函数转换成为一个概率值。softmax 函数可以定义为：

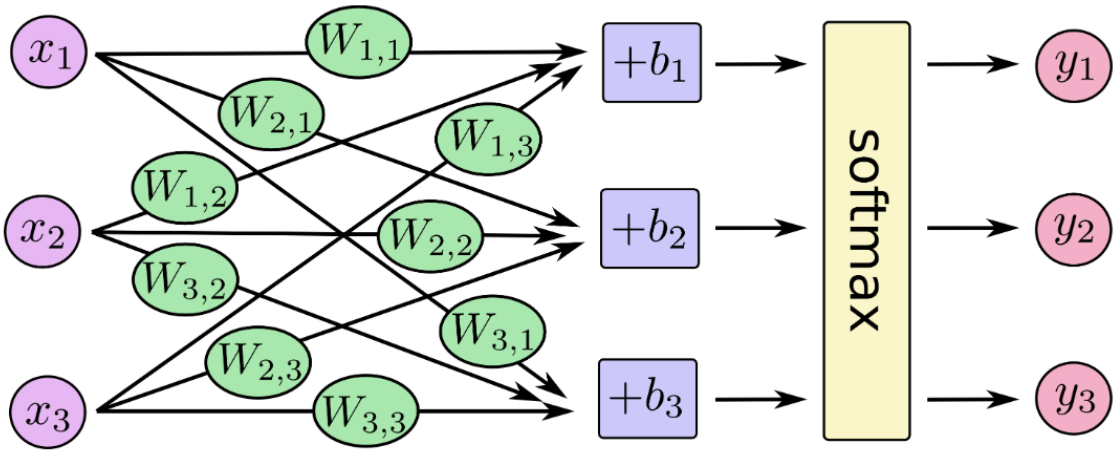
$$\text{softmax}(x) = \text{normalize}(\exp(x))$$

展开等式右边的子式，可以得到：

$$\mathrm{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

但是更多的时候把 softmax 模型函数定义为前一种形式：把输入值当成幂指数求值，再正则化这些结果值。这个幂运算表示，更大的证据对应更大的假设模型（hypothesis）里面的乘数权重值。反之，拥有更少的证据意味着在假设模型里面拥有更小的乘数系数。假设模型里的权值不可以是 0 值或者负值。Softmax 然后会正则化这些权重值，使它们的总和等于 1，以此构造一个有效的概率分布。

对于 softmax 回归模型可以用下面的图解释，对于输入的 xs 加权求和，再分别加上一个偏置量，最后再输入到 softmax 函数中：



如果把它写成一个等式，可以得到：

$$y = \mathrm{softmax}(Wx + b)$$

我们也可以用向量表示这个计算过程：用矩阵乘法和向量相加。这有助于提高计算效率。（也是一种更有效的思考方式）

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \mathrm{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

2.3 代码实现和相关结果

2.3.1 模型预处理

```
import tensorflow as tf

INPUT_NODE = 784
OUTPUT_NODE = 10
LAYER_NODE = 500

# 定义神经网络参数,传入两个参数,一个是 shape 一个是正则化参数大小
def get_weight(shape,regularizer):
    # tf.truncated_normal 截断的正态分布函数,超过标准差的重新生成
    w = tf.Variable(tf.truncated_normal(shape,stddev=0.1))
    if regularizer != None:

    # 将正则化结果存入 losses 中
                    tf.add_to_collection("losses",tf.contrib.layers.l2_regula
rizer(regularizer)(w))
    return w

# 定义偏置 b,传入 shape 参数
def get_bias(shape):
```

```
    # 初始化为0
    b = tf.Variable(tf.zeros(shape))
    return b

# 定义前向传播过程,两个参数,一个是输入数据,一个是正则化参数
def forward(x,regularizer):
    # w1 的维度就是[输入神经元大小,第一层隐含层神经元大小]
    w1 = get_weight([INPUT_NODE,LAYER_NODE],regularizer)
    # 偏置 b 参数,与 w 的后一个参数相同
    b1 = get_bias(LAYER_NODE)
    # 激活函数
    y1 = tf.nn.relu(tf.matmul(x,w1)+b1)

    w2 = get_weight([LAYER_NODE,OUTPUT_NODE],regularizer)
    b2 = get_bias(OUTPUT_NODE)
    y = tf.matmul(y1,w2)+b2

    return y
```

这一段代码将后面所需要用的模型进行了初步的处理，定义了一系列的参数与模型的基本结构，神经网络的搭建等，方便后面的调用。

2.3.2 训练模型

```
#coding:utf-8
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import mnist_forward
import os


BATCH_SIZE = 200
#学习率衰减的原始值
LEARNING_RATE_BASE = 0.1
# 学习率衰减率
LEARNING_RATE_DECAY = 0.99
# 正则化参数
REGULARIZER = 0.0001
# 训练轮数
STEPS = 50000
#这个使用滑动平均的衰减率
MOVING_AVERAGE_DECAY = 0.99
MODEL_SAVE_PATH = "./model/"
MODEL_NAME = "mnist_model"

def backward(mnist):
    #一共有多少个特征,784 行,一列
    x = tf.placeholder(tf.float32, [None,mnist_forward.INPUT_NODE])
    y_ = tf.placeholder(tf.float32, [None,mnist_forward.OUTPUT_NODE])
    # 给前向传播传入参数 x 和正则化参数计算出 y 的值
```

```
    y = mnist_forward.forward(x,REGULARIZER)
    # 初始化 global—step,它会随着训练轮数增加
    global_step = tf.Variable(0,trainable=False)

    # softmax 和交叉商一起运算的函数，logits 传入是 x*w,也就是 y
    ce =
tf.nn.sparse_softmax_cross_entropy_with_logits(logits=y,labels=tf.argmax(y_,
1))
    cem = tf.reduce_mean(ce)
    loss = cem + tf.add_n(tf.get_collection("losses"))

    learning_rate = tf.train.exponential_decay(LEARNING_RATE_BASE,
                                               global_step,
                                  mnist.train.num_examples/BATCH_SIZE,
                                              LEARNING_RATE_DECAY,
                                               staircase = True)

    train_step =
tf.train.GradientDescentOptimizer(learning_rate).minimize(loss,global_step =
global_step)

     # 滑动平均处理,可以提高泛化能力
    ema = tf.train.ExponentialMovingAverage(MOVING_AVERAGE_DECAY,global_step)
    ema_op = ema.apply(tf.trainable_variables())
    # 将 train_step 和滑动平均计算 ema_op 放在同一个节点
    with tf.control_dependencies([train_step,ema_op]):
        train_op = tf.no_op(name="train")

    saver = tf.train.Saver()

    with tf.Session() as sess:

        init_op = tf.global_variables_initializer()
        sess.run(init_op)

        for i in range(STEPS):
            # mnist.train.next_batch()函数包含一个参数 BATCH_SIZE,表示随机从训练集中抽
取 BATCH_SIZE 个样本输入到神经网络
            # next_batch 函数返回的是 image 的像素和标签 label
            xs,ys = mnist.train.next_batch(BATCH_SIZE)
            # _,表示后面不使用这个变量
            _,loss_value,step =
sess.run([train_op,loss,global_step],feed_dict={x:xs,y_:ys})

            if i % 1000 == 0:
                print("Ater {} training step(s),loss on training batch is {}
".format(step,loss_value))

saver.save(sess,os.path.join(MODEL_SAVE_PATH,MODEL_NAME),global_step=global_
step)

def main():

    MNIST_data_folder="/home/thea/mnist" #数据集保存的本地地址
    Mnist=input_data.read_data_sets(MNIST_data_folder,one_hot=True)
    backward(mnist)

if __name__ == "__main__":
    main()
```

运行结果：（训练 50000 轮时）

```
nist) is deprecated and will be removed in a future version.
nstructions for updating:
lease use alternatives such as official/mnist/dataset.py from tensorflow/models.
019-03-09 16:19:16.007973: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
ter 1 training step(s),loss on training batch is 2.957552909851074
ter 1001 training step(s),loss on training batch is 0.38177695870399475
ter 2001 training step(s),loss on training batch is 0.25153878331184387
ter 3001 training step(s),loss on training batch is 0.27313360571861267
ter 4001 training step(s),loss on training batch is 0.27087917923927307
ter 5001 training step(s),loss on training batch is 0.21519321203231812
ter 6001 training step(s),loss on training batch is 0.24082133173942566
ter 7001 training step(s),loss on training batch is 0.18190732598304749
ter 8001 training step(s),loss on training batch is 0.20831450819969177
ter 9001 training step(s),loss on training batch is 0.18545642495155334
ter 10001 training step(s),loss on training batch is 0.18489259481430054
ter 11001 training step(s),loss on training batch is 0.1577823609113693 2
ter 12001 training step(s),loss on training batch is 0.15980225801467896
ter 13001 training step(s),loss on training batch is 0.16054190695285797
ter 14001 training step(s),loss on training batch is 0.15831787884235382
ter 15001 training step(s),loss on training batch is 0.15356799960136414
ter 16001 training step(s),loss on training batch is 0.16002488136291504
ter 17001 training step(s),loss on training batch is 0.1480199694633484
ter 18001 training step(s),loss on training batch is 0.1436876505613327
ter 19001 training step(s),loss on training batch is 0.14518389105796814
ter 20001 training step(s),loss on training batch is 0.16442367434501648
ter 21001 training step(s),loss on training batch is 0.14464515447616577
ter 22001 training step(s),loss on training batch is 0.15730705857276917
ter 23001 training step(s),loss on training batch is 0.1669822782278061
ter 24001 training step(s),loss on training batch is 0.15965887904167175
ter 25001 training step(s),loss on training batch is 0.14546772837638855
ter 26001 training step(s),loss on training batch is 0.14833202958106995
ter 27001 training step(s),loss on training batch is 0.14087770879268646
ter 28001 training step(s),loss on training batch is 0.14518055319786072
ter 29001 training step(s),loss on training batch is 0.16646593809127808
ter 30001 training step(s),loss on training batch is 0.1328040361404419
ter 31001 training step(s),loss on training batch is 0.13077113032341003
ter 32001 training step(s),loss on training batch is 0.13635599613189697
ter 33001 training step(s),loss on training batch is 0.13366036117076874
ter 34001 training step(s),loss on training batch is 0.1348063349723816
ter 35001 training step(s),loss on training batch is 0.13583828508853912
ter 36001 training step(s),loss on training batch is 0.1297684758901596
ter 37001 training step(s),loss on training batch is 0.1374141275882721
ter 38001 training step(s),loss on training batch is 0.13869298994541168
ter 39001 training step(s),loss on training batch is 0.13478069007396698
ter 40001 training step(s),loss on training batch is 0.1281266212463379
ter 41001 training step(s),loss on training batch is 0.13120421767234802
ter 42001 training step(s),loss on training batch is 0.13729842007160187
ter 43001 training step(s),loss on training batch is 0.12780506908893585
ter 44001 training step(s),loss on training batch is 0.12838385999202728
ter 45001 training step(s),loss on training batch is 0.1336909681558609
ter 46001 training step(s),loss on training batch is 0.1281001716852188
ter 47001 training step(s),loss on training batch is 0.1279028356075287
ter 48001 training step(s),loss on training batch is 0.1278633326292038
ter 49001 training step(s),loss on training batch is 0.12330781668424606
```

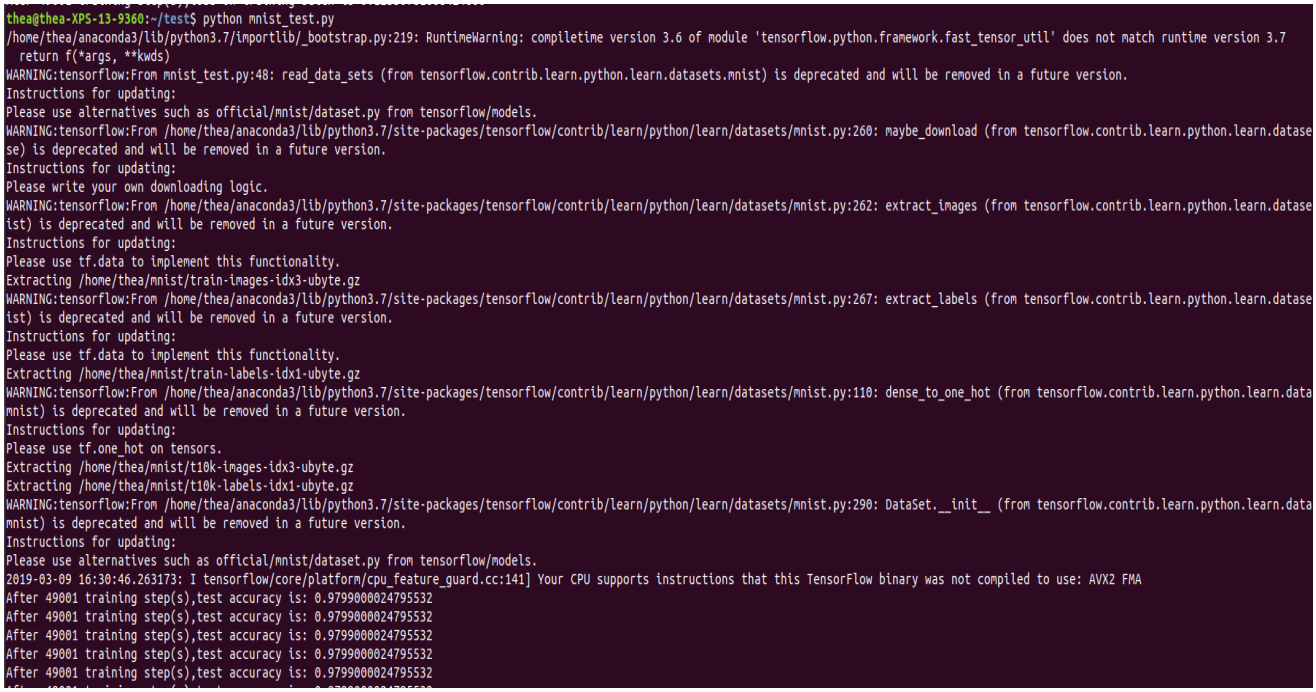
训练模型正是 mnist 手写数字识别的重要步骤。当调用官方下载的数据集中的训练集进行训练后，自己的模型就可以在不断的预测结果-比对标签值-判断正误的过程中，提高识别数字的准确度。根据我个人的训练经验，若只训练 10000 轮，该模型对于数字 5 和 7 的识别是有误的，正如我上文的例子中所言。而训练到 50000 轮后，数字 0 到 9 就都能准确的识别出来了！

2.3.3 利用数据集中的测试数据进行模型检测

```
#coding:utf-8
import time
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import mnist_forward
import mnist_backward

TEST_INTERVAL_SECS = 5

def test(mnist):
    with tf.Graph().as_default() as g:
        # 占位符,第一个参数是 tf.float32 数据类型,第二个参数是 shape,shape[0]=None 表
示输入维度任意,shpe[1]表示输入数据特征数
        x = tf.placeholder(tf.float32,shape =
[None,mnist_forward.INPUT_NODE])
        y_ = tf.placeholder(tf.float32,shape =
[None,mnist_forward.OUTPUT_NODE])
        """注意这里没有传入正则化参数,需要明确的是,在测试的时候不要正则化,不要
dropout"""
        y = mnist_forward.forward(x,None)

        # 实例化可还原的滑动平均模型
        ema =
tf.train.ExponentialMovingAverage(mnist_backward.MOVING_AVERAGE_DECAY)
        ema_restore = ema.variables_to_restore()
        saver = tf.train.Saver(ema_restore)

        # y 计算的过程：x 是 mnist.test.images 是 10000×784 的，最后输出的 y 仕
10000×10 的，y_:mnist.test.labels 也是 10000×10 的
        correct_prediction = tf.equal(tf.argmax(y,1),tf.argmax(y_,1))
        # tf.cast 可以师兄数据类型的转换,tf.equal 返回的只有 True 和 False
        accuracy = tf.reduce_mean(tf.cast(correct_prediction,tf.float32))

        while True:
            with tf.Session() as sess:
                # 加载训练好的模型
                ckpt =
tf.train.get_checkpoint_state(mnist_backward.MODEL_SAVE_PATH)
                if ckpt and ckpt.model_checkpoint_path:
                    # 恢复模型到当前会话
                    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})
                    print("After {} training step(s),test accuracy is: {}
".format(global_step,accuracy_score))
                else:
                    print("No chekpoint file found")
                    print(sess.run(y,feed_dict={x:mnist.test.images}))
                    return
            time.sleep(TEST_INTERVAL_SECS)

def main():
```

```
    MNIST_data_folder="/home/thea/mnist"
    mnist=input_data.read_data_sets(MNIST_data_folder,one_hot=True)
    test(mnist)

if __name__== "__main__":
    main()
```

运行结果：



在调用了之前的预处理与训练结果以后，我们已经可以加载模型使用了。这段代码是用于测试训练完毕的模型是否可以正常调用以及准确率多少的。更直观也更符合人类语言的习惯。

2.3.4 调用模型

```
import tensorflow as tf
import numpy as np
from PIL import Image
import mnist_forward
import mnist_backward
from redis import Redis, RedisError
import os
import socket
from flask import Flask, request, redirect, url_for
from werkzeug import secure_filename
from flask import send_from_directory
```

```
# 定义加载使用模型进行预测的函数
def restore_model(testPicArr):

    with tf.Graph().as_default() as tg:

        x = tf.placeholder(tf.float32,[None,mnist_forward.INPUT_NODE])
        y = mnist_forward.forward(x,None)
        preValue = tf.argmax(y,1)
        # 加载滑动平均模型
        variable_averages =
tf.train.ExponentialMovingAverage(mnist_backward.MOVING_AVERAGE_DECAY)
        variables_to_restore = variable_averages.variables_to_restore()
        saver = tf.train.Saver(variables_to_restore)

        with tf.Session() as sess:

            ckpt =
tf.train.get_checkpoint_state(mnist_backward.MODEL_SAVE_PATH)
            if ckpt and ckpt.model_checkpoint_path:
                # 恢复当前会话,将 ckpt 中的值赋值给 w 和 b
                saver.restore(sess,ckpt.model_checkpoint_path)
                # 执行图计算
                preValue = sess.run(preValue,feed_dict={x:testPicArr})
                return preValue
            else:
                print("No checkpoint file found")
                return -1
# 图片预处理函数
def pre_pic(picName):
    # 先打开传入的原始图片
    img = Image.open(picName)
    # 使用消除锯齿的方法 resize 图片
    reIm = img.resize((28,28),Image.ANTIALIAS)
    # 变成灰度图，转换成矩阵
    im_arr = np.array(reIm.convert("L"))
    threshold = 50#对图像进行二值化处理，设置合理的阈值，可以过滤掉噪声，让他只有纯白色
的点和纯黑色点
    for i in range(28):
        for j in range(28):
            im_arr[i][j] = 255-im_arr[i][j]
            if (im_arr[i][j]<threshold):
                im_arr[i][j] = 0
            else:
                im_arr[i][j] = 255
    # 将图像矩阵拉成 1 行 784 列，并将值变成浮点型（像素要求的仕 0-1 的浮点型输入）
    nm_arr = im_arr.reshape([1,784])
    nm_arr = nm_arr.astype(np.float32)
    img_ready = np.multiply(nm_arr,1.0/255.0)

return img_ready
```

这段代码调用了训练好，知道准确率的模型。并将使用者上传的图片进行预处理，以满足文件展开方式一样的需求。图片二值化的代码使得该 mnist 模型在应用中更加方便操作。

3、docker 容器的原理及运用

3.1 研究背景

无论是国内还是国外，大企业如亚马逊，谷歌，微软，百度，腾讯，小公司如摩拜，ofo，美团。“如果放弃了容器技术，那只有死路一条。”

容器不是一种工具，说它是一种微服务生态可能更合适一些。它运行时和底层磁盘交互，文件系统如何做？容器间的通讯，网络如何部署？容器还能进行如何的优化？都是一种深刻的联系，也是如今学术界在大量大数据的文章中想要说明的内容。

Docker 是一个新的 containerization technology。是 PaaS 提供商 dotCloud 开源的一个基于 LXC 的高级容器引擎。他的源代码托管在 Github 上，基于 go 语言并遵从 Apache2.0 协议开源。Docker 被发明出来是为了在这样日益复杂的开发环境下寻找出路:

如今的环境管理更复杂 - 从各种 OS 到各种中间件到各种 app，一款产品能够成功作为开发者需要关心的东西太多，且难于管理，这个问题几乎在所有现代 IT 相关行业都需要面对。

云计算时代的到来 - AWS 的成功，引导开发者将应用转移到 cloud 上，解决了硬件管理的问题，然而中间件相关的问题依然存在。Docker 为开发者思路变化提供了可能性。

虚拟化手段的变化 - cloud 时代采用标配硬件来降低成本，采用虚拟化手段来满足用户按需使用的需求以及保证可用性和隔离性。然而无论是 KVM 还是 Xen 在 docker 看来,都在浪费资源，因为用户需要的是高效运行环境而非 OS，GuestOS 既浪费资源又难于管理，更加轻量级的 LXC 更加灵活和快速。

LXC 的移动性 - LXC 在 linux 2.6 的 kernel 里就已经存在了，但是其设计之初并非为云计算考虑的。缺少标准化的描述手段和容器的可迁移性，决定了其构建出的环境难于迁移和标准化管理(相对于 KVM 之类 image 和 snapshot 的概念)。docker 就在这个问题上做出实质性的革新。这是 docker 最独特的地方。

面对上述几个问题，docker 设想是成为一个海运时的集装箱。就像我们把货物装进集装箱，等到达目的地时再打开。我们会把程序放进 docker 容器里封存起来，当他到达所需使用程序的电脑时，该电脑只需要安装了 docker,就可以将里面的程序跑起来。

容器有三个特点：light-weight，portable，self-sufficient。轻量是因为容器的大小就是你程序的大小，而不用存放系统等多余的东西。也正因为它的体积小，所以它在不同机器上可以轻松地迁移，十分便携。同时它又是自我完整的，在盒子里是有跑程序所需要的依赖的。这些特点也使得容器的地位愈发重要。

3.2 结构与原理

(1)dockerfile:

```
# Use an official Python runtime as a parent image
FROM tensorflow/tensorflow:1.9.0-devel-py3

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 8080

# Run app.py when the container launches
CMD ["python", "mnist_app.py"]
```

（2）requirements:

```
Flask
Redis
Pillow
Werkzeug
```

(3) mnist_app.py:（上传文件以及开放端口部分）

```
# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)

UPLOAD_FOLDER = '/app'
ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'])

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1] in ALLOWED_EXTENSIONS

@app.route('/', methods=['GET', 'POST'])
def upload_file():
    createKeySpace()
    if request.method == 'POST':
        file = request.files['file']
        if file and allowed_file(file.filename):
            global filename
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            return redirect(url_for('uploaded_file',
                                    filename=filename))
```

```
    return '''
    <!doctype html>
    <title>Upload new File</title>
    <h1>Upload new File</h1>
    <form action="" method=post enctype=multipart/form-data>
      <p><input type=file name=file>
         <input type=submit value=Upload>
    </form>
    '''

@app.route('/uploads/<filename>')
def uploaded_file(filename):
        testPic = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        # 将图片路径传入图像预处理函数中
        testPicArr = pre_pic(testPic)
        # 将处理后的结果输入到预测函数最后返回预测结果
        global prevalue
        prevalue= int(restore_model(testPicArr))
        insertInformation()
        return("The prediction number is :%d"%(prevalue))




if __name__ == "__main__":

    app.run(host='0.0.0.0', port=8080, debug=True)
```

创建容器：

```
Docker build -t mnistapp .
```

运行容器：

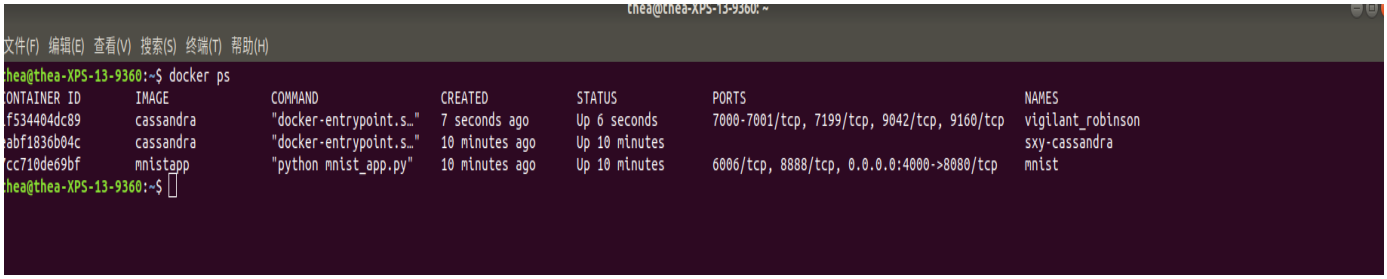
```
docker run -p 4000:8080 mnistapp
docker run -d -p 4000:8080 mnistapp
```

注意：实际操作时，为了使 mnistapp 容器程序中的 9042 端口与 cassandra 容器的 9042 端口指向一处，让信息互通有无，我使用了 network container 命令将两个容器连接起来：

docker run -d --name mnist -p 4000:8080 mnistapp
docker run -d --name sxy-cassandra --network container:mnist Cassandra

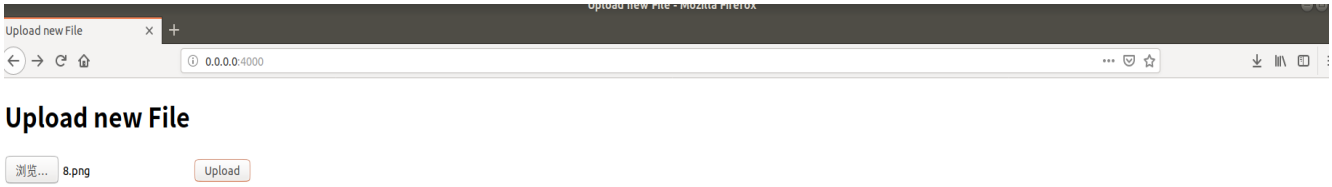
3.3 相关结果

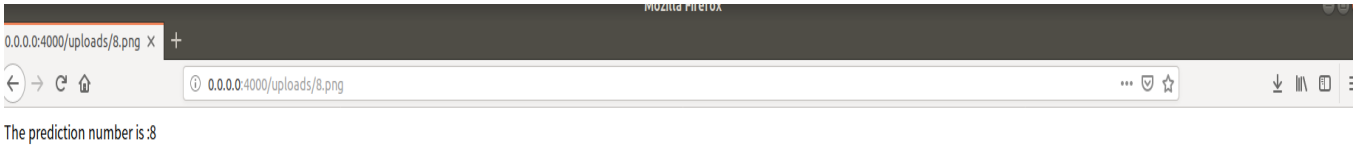
运行时查看运行的容器:



可以看见 mnistapp 正在运行，由宿主机的 4000 端口映射到容器里的 8080 端口。

访问 http://0.0.0.0:4000 网页时：





可以看见在该网页上实现了手写数字识别的功能!

4、cassandra 的原理及运用

4.1 研究背景

Cassandra 是一套开源分布式 NoSQL 数据库系统。它最初由 Facebook 开发，用于储存收件箱等简单格式数据，集 GoogleBigTable 的数据模型与 AmazonDynamo 的完全分布式的架构于一身。它是一个混合型的非关系的数据库，类似于 Google 的 BigTable。

Cassandra 最初由 Facebook 开发，后转变成了开源项目。它是一个网络社交云计算方面理想的数据库。以 Amazon 专有的完全分布式的 Dynamo 为基础，结合了 GoogleBigTable 基于列族（ColumnFamily）的数据模型。P2P 去中心化的存储。很多方面都可以称之为 Dynamo2.0。

4.2 代码实现和相关结果

打开 Cassandra 的命令行:

```
docker run -it --link sxy-cassandra:cassandra --rm cassandra cqlsh
cassandra
```

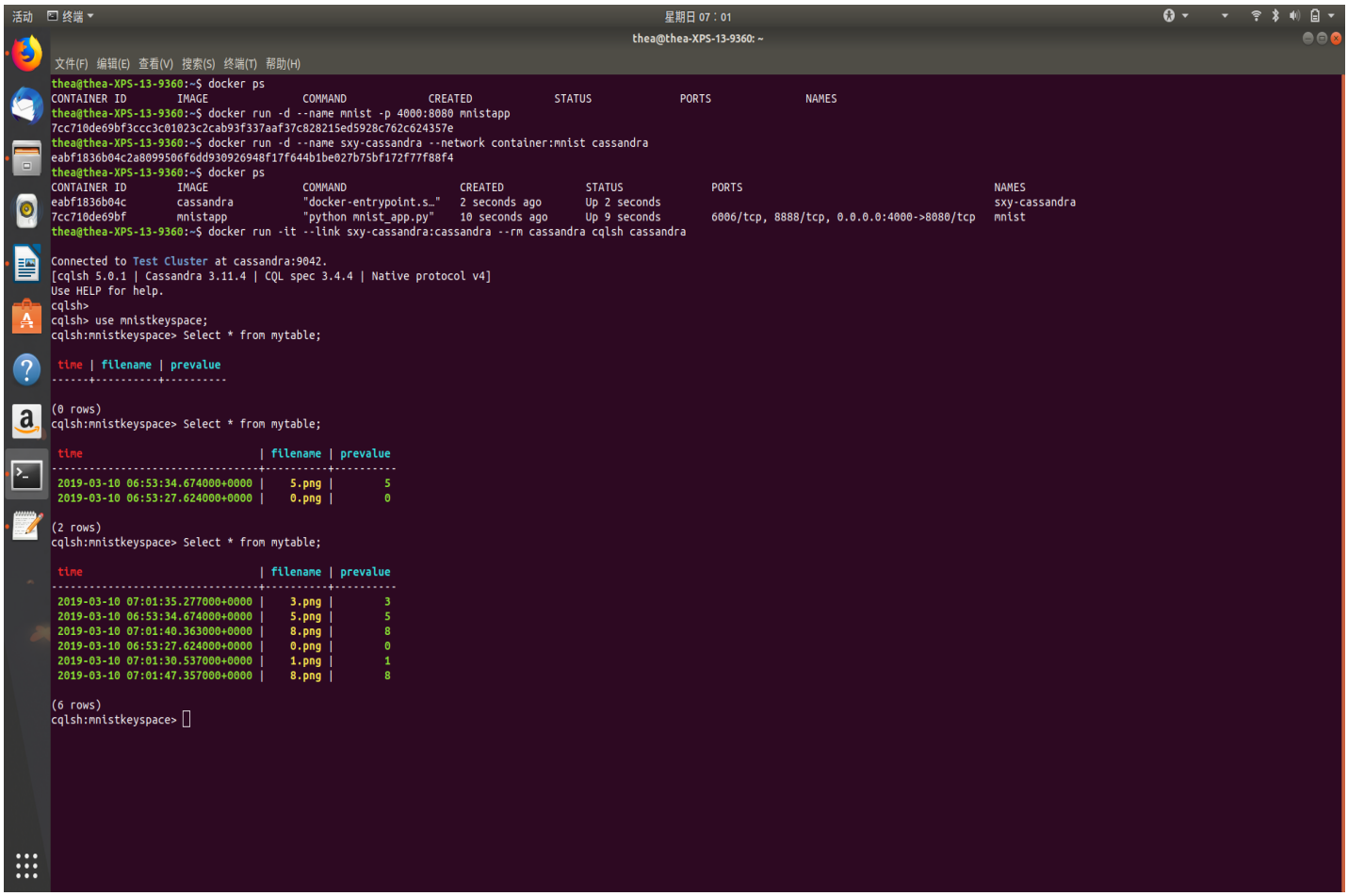
写在 mnist_app.py 中的代码:

```
prevalue = 0
filename = "123"


import logging
import datetime

log = logging.getLogger()
log.setLevel('INFO')

handler = logging.StreamHandler()
handler.setFormatter(logging.Formatter("%(asctime)s
[%(levelname)s] %(name)s: %(message)s"))

log.addHandler(handler)

#from cassandra.cluster import Cluster

#from cassandra import ConsistencyLevel

from cassandra.cluster import Cluster

from cassandra.query import SimpleStatement

KEYSPACE = "mnistkeyspace"


def createKeySpace():
```

```
        cluster = Cluster(contact_points=['0.0.0.0'],port=9042)

        session = cluster.connect()

        log.info("Creating keyspace...")

        try:

           session.execute("""

           CREATE KEYSPACE %s

           WITH replication = { 'class': 'SimpleStrategy',
'replication_factor': '2' }

           """ % KEYSPACE)

           log.info("setting keyspace...")

           session.set_keyspace(KEYSPACE)

           log.info("creating table...")

           session.execute("""

           CREATE TABLE mytable (

               time timestamp,

               filename text   ,

               prevalue int,

               PRIMARY KEY (time, filename, prevalue)

           )

           """)



        except Exception as e:

           log.error("Unable to create keyspace")

           log.error(e)

def insertInformation():
        global prevalue
        global filename

        cluster = Cluster(contact_points=['0.0.0.0'],port=9042)

        session = cluster.connect()

        try:

           log.info("inserting information...")
           session.execute("""
           INSERT INTO %smytable (
           time, filename, prevalue
```

```
        )
        VALUES (dateof(now()), %s , %d)
        """ %(KEYSPACE+".","\'"+filename+"\'",prevalue))

    except Exception as e:

        log.error("Unable to create keyspace")
        log.error(e)
```

相关结果:



该部分实现了将上传图片之后的上传时间、文件名称、识别数字存放入 cassandra 表格的功能。作为此次项目的最后一个模块，与 mnistapp 容器相连接，存放了所需要的数据，使得用户在查看时更加完整，方便查询。

5.完整代码:

```
import tensorflow as tf
import numpy as np
from PIL import Image
import mnist_forward
import mnist_backward
from redis import Redis, RedisError
import os
import socket
from flask import Flask, request, redirect, url_for
from werkzeug import secure_filename
from flask import send_from_directory

prevalue = 0
filename = "123"


import logging
import datetime

log = logging.getLogger()

log.setLevel('INFO')

handler = logging.StreamHandler()

handler.setFormatter(logging.Formatter("%(asctime)s
[%(levelname)s] %(name)s: %(message)s"))

log.addHandler(handler)

#from cassandra.cluster import Cluster

#from cassandra import ConsistencyLevel

from cassandra.cluster import Cluster

from cassandra.query import SimpleStatement

KEYSPACE = "mnistkeyspace"


def createKeySpace():


       cluster = Cluster(contact_points=['0.0.0.0'],port=9042)

       session = cluster.connect()

       log.info("Creating keyspace...")

       try:

          session.execute("""

          CREATE KEYSPACE %s

          WITH replication = { 'class': 'SimpleStrategy',
'replication_factor': '2' }
```

```
            """ % KEYSPACE)

            log.info("setting keyspace...")

            session.set_keyspace(KEYSPACE)

            log.info("creating table...")

            session.execute("""

            CREATE TABLE mytable (

                time timestamp,

                filename text   ,

                prevalue int,

                PRIMARY KEY (time, filename, prevalue)

            )

            """)



        except Exception as e:

            log.error("Unable to create keyspace")

            log.error(e)

def insertInformation():
        global prevalue
        global filename

        cluster = Cluster(contact_points=['0.0.0.0'],port=9042)

        session = cluster.connect()

        try:

            log.info("inserting information...")
            session.execute("""
            INSERT INTO %smytable (
            time, filename, prevalue
            )
            VALUES (dateof(now()), %s , %d)
            """ %(KEYSPACE+".","\'"+filename+"\'",prevalue))
        except Exception as e:

            log.error("Unable to create keyspace")

            log.error(e)

# 定义加载使用模型进行预测的函数
def restore_model(testPicArr):

    with tf.Graph().as_default() as tg:
```

```
        x = tf.placeholder(tf.float32,[None,mnist_forward.INPUT_NODE])
        y = mnist_forward.forward(x,None)
        preValue = tf.argmax(y,1)
        # 加载滑动平均模型
        variable_averages =
tf.train.ExponentialMovingAverage(mnist_backward.MOVING_AVERAGE_DECAY)
        variables_to_restore = variable_averages.variables_to_restore()
        saver = tf.train.Saver(variables_to_restore)

        with tf.Session() as sess:

            ckpt =
tf.train.get_checkpoint_state(mnist_backward.MODEL_SAVE_PATH)
            if ckpt and ckpt.model_checkpoint_path:
                # 恢复当前会话,将 ckpt 中的值赋值给 w 和 b
                saver.restore(sess,ckpt.model_checkpoint_path)
                # 执行图计算
                preValue = sess.run(preValue,feed_dict={x:testPicArr})
                return preValue
            else:
                print("No checkpoint file found")
                return -1
# 图片预处理函数
def pre_pic(picName):
    # 先打开传入的原始图片
    img = Image.open(picName)
    # 使用消除锯齿的方法 resize 图片
    reIm = img.resize((28,28),Image.ANTIALIAS)
    # 变成灰度图，转换成矩阵
    im_arr = np.array(reIm.convert("L"))
    threshold = 50#对图像进行二值化处理，设置合理的阈值，可以过滤掉噪声，让他只有纯白色
的点和纯黑色点
    for i in range(28):
        for j in range(28):
            im_arr[i][j] = 255-im_arr[i][j]
            if (im_arr[i][j]<threshold):
                im_arr[i][j] = 0
            else:
                im_arr[i][j] = 255
    # 将图像矩阵拉成 1 行 784 列，并将值变成浮点型（像素要求的仕 0-1 的浮点型输入）
    nm_arr = im_arr.reshape([1,784])
    nm_arr = nm_arr.astype(np.float32)
    img_ready = np.multiply(nm_arr,1.0/255.0)

    return img_ready



# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)

UPLOAD_FOLDER = '/app'
ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'])

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def allowed_file(filename):
    return '.' in filename and \
```

```
           filename.rsplit('.', 1)[1] in ALLOWED_EXTENSIONS

@app.route('/', methods=['GET', 'POST'])
def upload_file():
    createKeySpace()
    if request.method == 'POST':
        file = request.files['file']
        if file and allowed_file(file.filename):
            global filename
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            return redirect(url_for('uploaded_file',
                                    filename=filename))
    return '''
    <!doctype html>
    <title>Upload new File</title>
    <h1>Upload new File</h1>
    <form action="" method=post enctype=multipart/form-data>
      <p><input type=file name=file>
         <input type=submit value=Upload>
    </form>
    '''

@app.route('/uploads/<filename>')
def uploaded_file(filename):
        testPic = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        # 将图片路径传入图像预处理函数中
        testPicArr = pre_pic(testPic)
        # 将处理后的结果输入到预测函数最后返回预测结果
        global prevalue
        prevalue= int(restore_model(testPicArr))
        insertInformation()
        return("The prediction number is :%d"%(prevalue))




if __name__ == "__main__":

    app.run(host='0.0.0.0', port=8080, debug=True)
```

6、学习感受与收获

1

大数据产业方兴未艾，其中多项技术正在经历不断的创新与变革。在未来的几十年，相信大数据行业都是社会热切关注的焦点，是推动信息时代发展的重要动力。

这次的项目也让我受益匪浅。从一个对大数据不甚了解的学生，到对如今的大数据发展有了些许概念，并能用容器等重要知识完成一些实际的应用，这进步不可谓不大。虽然学习的时间不是很长，使得我对于很多老师上课所提及的内容了解不够深入，运用指令也不够熟练，但是相信在这一次项目的基础上，我将来的学习会更有方向性，更有效率!

这次的项目分为几大块：将 mnist 程序读通并跑起来，将该程序放入容器中运行，程序先将客户的图片识别出来然后将识别结果存入 cassandra 数据库。完成其中每一个环节都曾出现大大小小的错误，有时候一个格式错误也要反复修改几遍，查阅大量相关资料。但每解决一个问题之后，带来的成就感与喜悦也是能让人沉迷其中的。在完成这个项目以后，我做事的耐心，查阅资料的能力，辨别筛选有用信息的能力都有了大幅的提高。也为完成下一个项目积累了一定的经验。

这次项目上课的老师真的帮助了我很多。他不仅讲课的时候给到的知识点非常全面，有效地帮助我理解大数据技术，也为我后续的学习提供方向与方法。还在解决问题时给与我适当的指导，因为很多时候出现了问题我都无法知道是哪方面的原因，查询时自然也无从下手，但是老师可以帮我讲解相关的原理，启发我应当从哪个方面入手。

当然，这个项目还有很多改进的空间。比如上传文件的时间可以更符合平时阅读的习惯，实现应用的代码可以更加简练，将手写数字窗口放在网页上更方便做测试与使用等等。在完成这个项目以后，我还会继续钻研完善相关功能。

。