Final report for the big data research

Yin Ni

2019.09.13

Content

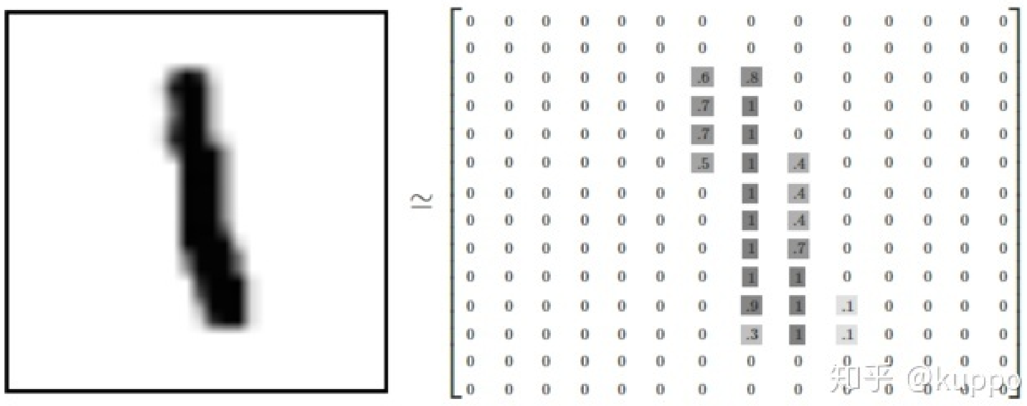
● Introduction

This program aims to recognize the handwriting picture through the MNIST data set, a classic dataset for the data mining, and softmax algorithm, an useful function for machine learning.

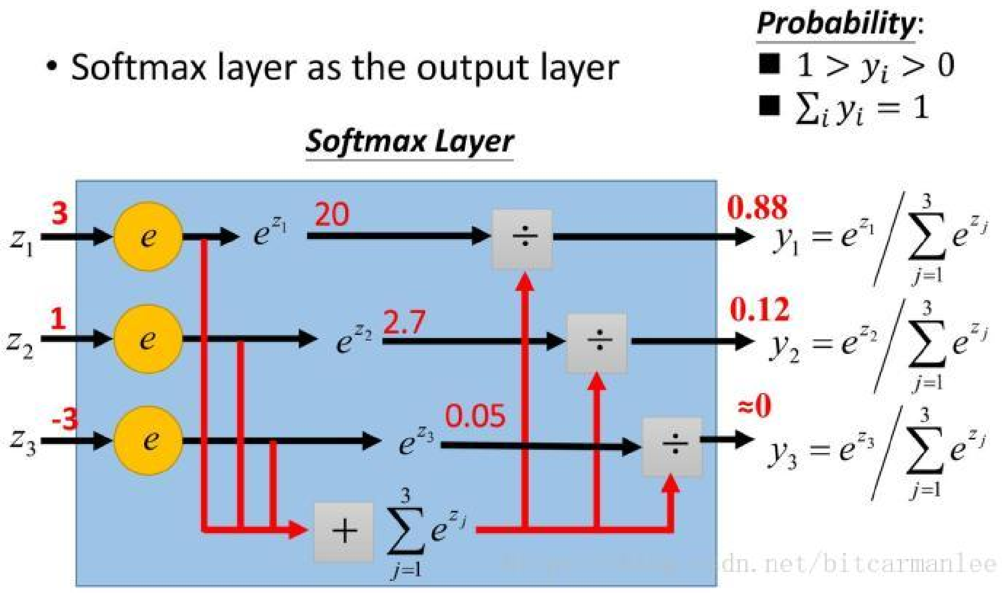
- The MNIST(Mixed National Institute of Standards and Technology database) data set is created by National Institute of Standards and Technology, which is a typical and special data set for handwriting number. This data set includes 60000 samples for training and 10000 samples for test.

  Some samples are similar to the picture :

  In this data set, all pictures are made up by 28*28 pixels and is marked by one gray value. This requirement is a must for the MNIST. No matter what the size or color a picture is, the first step we do is to turn it in to a standard format(28*28pixels and grayscale). So when we look at the pictures, actually we are looking at the matrix. Each pixel has its own gray value.



- Softmax algorithm is a kind of simple but useful function to calculate the probability of some situation and produce an evidence for judgment.


• Softmax layer as the output layer
Probability:
■ $1 > y_i > 0$
■ $\sum_i y_i = 1$
Softmax Layer
z_1 3
z_2 1
z_3 -3
e
e^{z_1} 20
e^{z_2} 2.7
e^{z_3} 0.05
+ $\sum_{j=1}^{3} e^{z_j}$
÷
0.88 $y_1 = e^{z_1} / \sum_{j=1}^{3} e^{z_j}$
0.12 $y_2 = e^{z_2} / \sum_{j=1}^{3} e^{z_j}$
≈0 $y_3 = e^{z_3} / \sum_{j=1}^{3} e^{z_j}$
dn.net/bitcarmanlee

Suppose we have a list V, Vi is the ith element in the list, the the softmax value of Vi is

$$S_i = \frac{e^i}{\sum_j e^j}$$

So the function can category data based on the value.

Then the process will be : we first use the MNIST and Softmax to get a training model, this model can recognize at least 98% of all pictures in the test samples, then based on the model, we can recognize any picture in local machine. Then, through flask, we can realize the function through local net server windows.

- **Research process**

◆ preparation

In this part, we will introduce the use of Docker.

Docker is a tech of Lightweight virtual technology project. It allows us to pack our environment and the program together, turned into an image in docker. We can push it up to the docker hub, which allows others to pull down the image on their machine and run it without the dependence of environment for each installation.

Docker has several advantages: High utilization rate of system resources: a host can run thousands of dockers without consuming resources. In the traditional virtual machine way,10 different applications will start 10 virtual machines, while Docker only needs to start 10 isolated applications;Docker containers can run on almost any platform and are easier to migrate…

In our porject, we need to claim some basic ideas about docker:

Image: it allows the creation of one container, as talked before, it is a sum of code and environment, each container is built on one image.

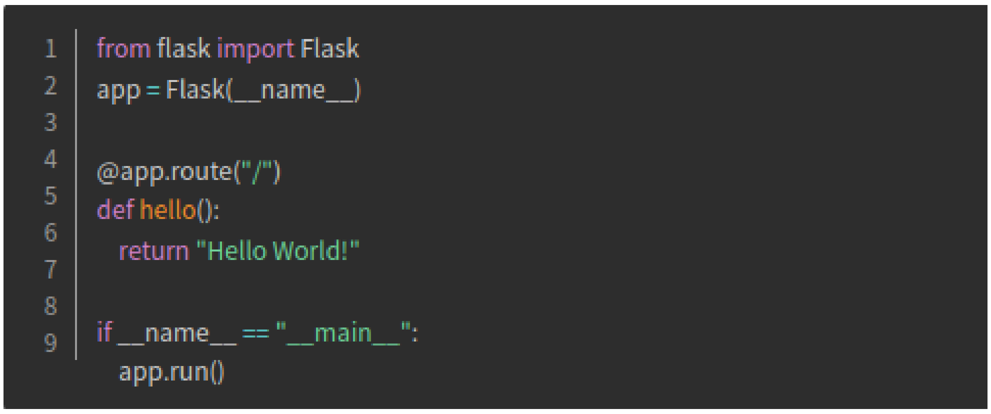
Container: it is a specific case for image to run. We can create, stop , restart, delete them. Each container is isolated. We can see one container as a simplified environment.

Repository: images are stored. The biggest repository is docker hub.

Besides that, we will also need Flask:

Flask is a lightweight web application framework written in Python. Based on Werkzeug WSGI (Python Web Server Gateway Interface (WSGI)) is an interface between Python applications or frameworks and Web servers. It has been widely accepted and it has basically achieved its portability aspects. The target) toolbox and the Jinja2 template engine. Flask uses BSD licenses. Flask is also known as "microframework" because it uses a simple core and adds other features with extension. Flask does not have a database or form validation tool that is used by default. However, Flask retains the flexibility of amplification, which can be added with Flask-extension: ORM, form validation tools, file uploads, and various open authentication technologies.

This is a simple code using flask.



```python
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
  return "Hello World!"

if __name__ == "__main__":
  app.run()
```

First we imported the Flask class.

Then we created an instance of this class. The first parameteris the name of the application module or package.

Then we use the route() decorator to tell Flask to trigger the URL of the function.

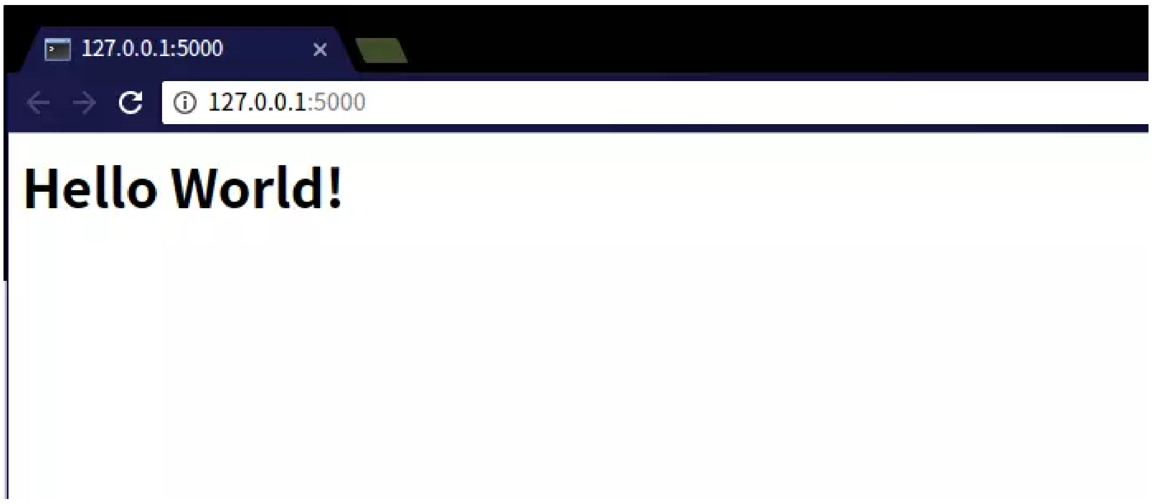
The function name can be used to generate an associated URL and return information that needs to be displayed in the user's browser.

Finally, use the run() function to run the local server and our application. If name =='main':

Press control-C to stop the server

After that, when we visit our local server (since the address is random, it always is 127.0.0.1:5000)

we can see :



Hello World!

Then we can add something more complex into this frame. And construct a page which could allow anyone to submit a picture and it will be automatically recognized and return a value.

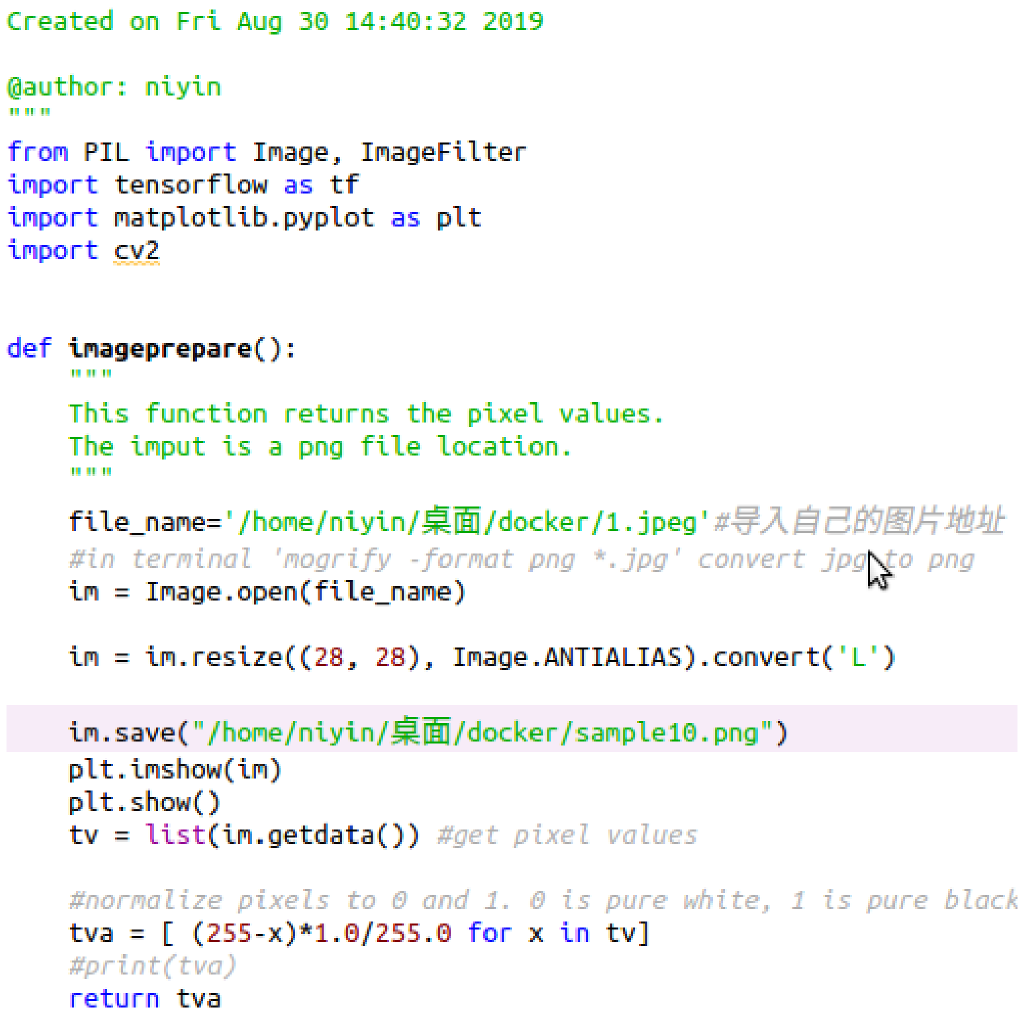
◆ Code analysis

The code for training and test can be found on

https://github.com/tensorflow/tensorflow/blob/r1.4/tensorflow/examples/tutorials/mnist/mnist_softmax.py

this is a typical training method and after install necessary modules, you will get a file named “model.ckpt”, this is the result of the training.

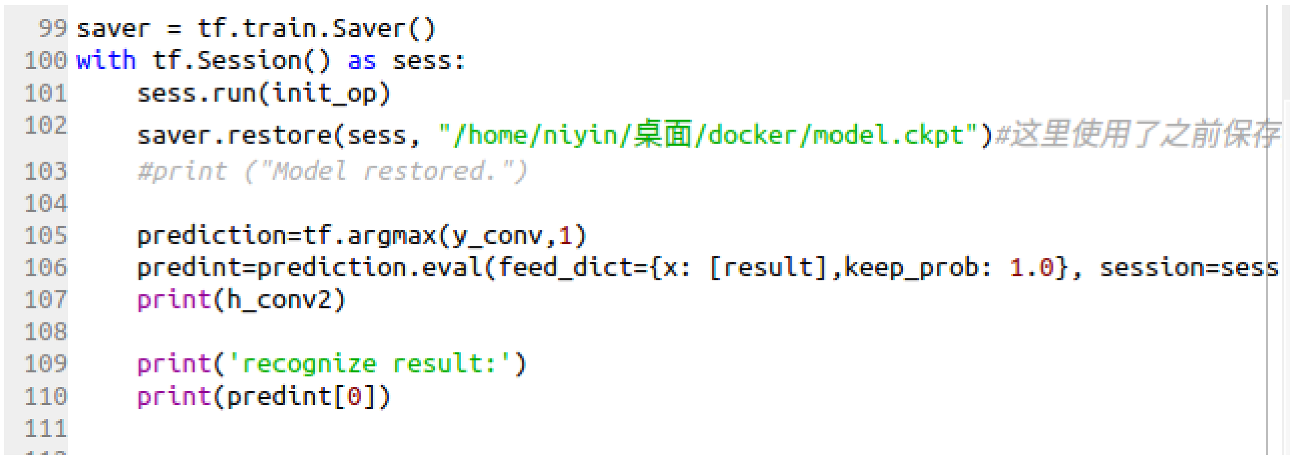
The part code for submit and use the model is listed :



```
Created on Fri Aug 30 14:40:32 2019

@author: niyin
"""
from PIL import Image, ImageFilter
import tensorflow as tf
import matplotlib.pyplot as plt
import cv2


def imageprepare():
    """
    This function returns the pixel values.
    The imput is a png file location.
    """
    file_name='/home/niyin/桌面/docker/1.jpeg'#导入自己的图片地址
    #in terminal 'mogrify -format png *.jpg' convert jpg to png
    im = Image.open(file_name)

    im = im.resize((28, 28), Image.ANTIALIAS).convert('L')

    im.save("/home/niyin/桌面/docker/sample10.png")
    plt.imshow(im)
    plt.show()
    tv = list(im.getdata()) #get pixel values

    #normalize pixels to 0 and 1. 0 is pure white, 1 is pure black
    tva = [ (255-x)*1.0/255.0 for x in tv]
    #print(tva)
    return tva
```

this part is for prepare the image into a standard format and store it.

This part is for us to use the model we trained and produce the final result.



```
saver = tf.train.Saver()
with tf.Session() as sess:
    sess.run(init_op)
    saver.restore(sess, "/home/niyin/桌面/docker/model.ckpt")#这里使用了之前保存
    #print ("Model restored.")

    prediction=tf.argmax(y_conv,1)
    predint=prediction.eval(feed_dict={x: [result],keep_prob: 1.0}, session=sess
    print(h_conv2)

    print('recognize result:')
    print(predint[0])

```

Since the route is in local machine. We try to run it on docker, two thing we have to do :

one is to create an environment on docker and the route above can be changed into route on docker

another is we have to combine the flask code with the recognizing code.

Consider the code on docker doc, we adapt it :

Dockerfile is :

FROM python:3.7.4-slim   # basic layer

WORKDIR /app # the workdir we create on docker

COPY . /app # copy files in local dir (the dockerfile lies in) into workdir we created

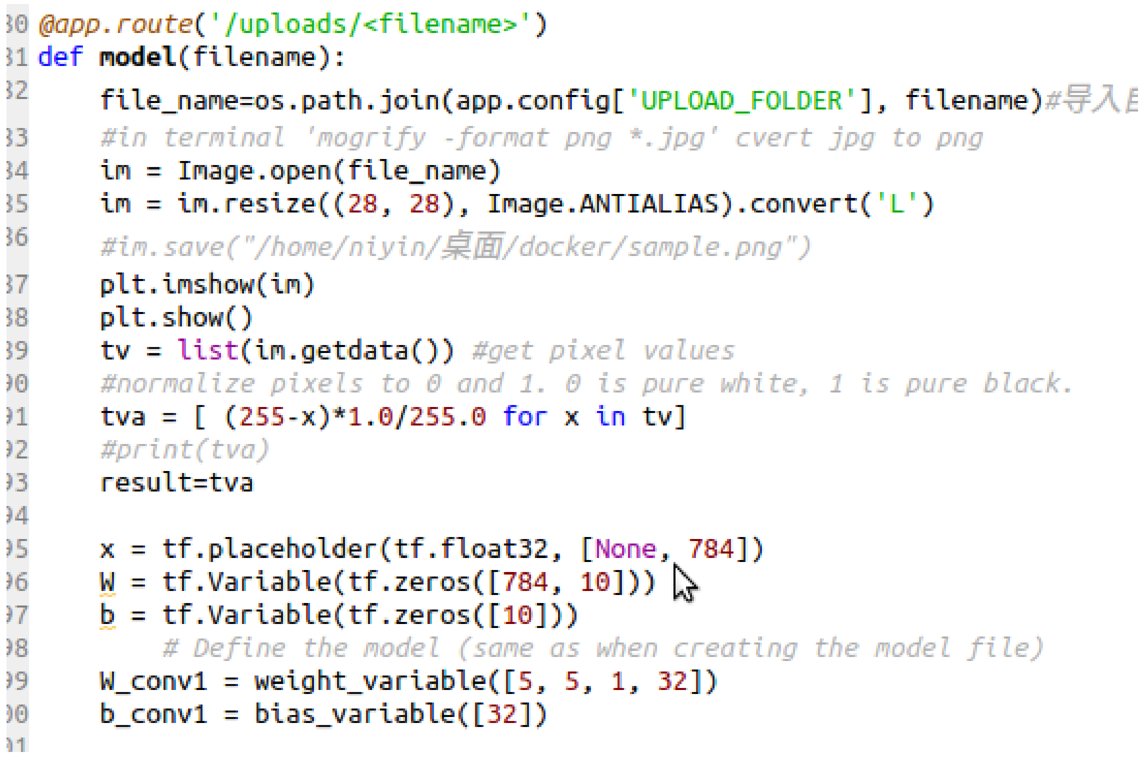
RUN pip install --trusted-host pypi.python.org -r requirements.txt #install necessary module in the image

EXPOSE 100 # expose 100 for outsiders to visit

CMD ["python", "real1.py"] # use python to run the main code

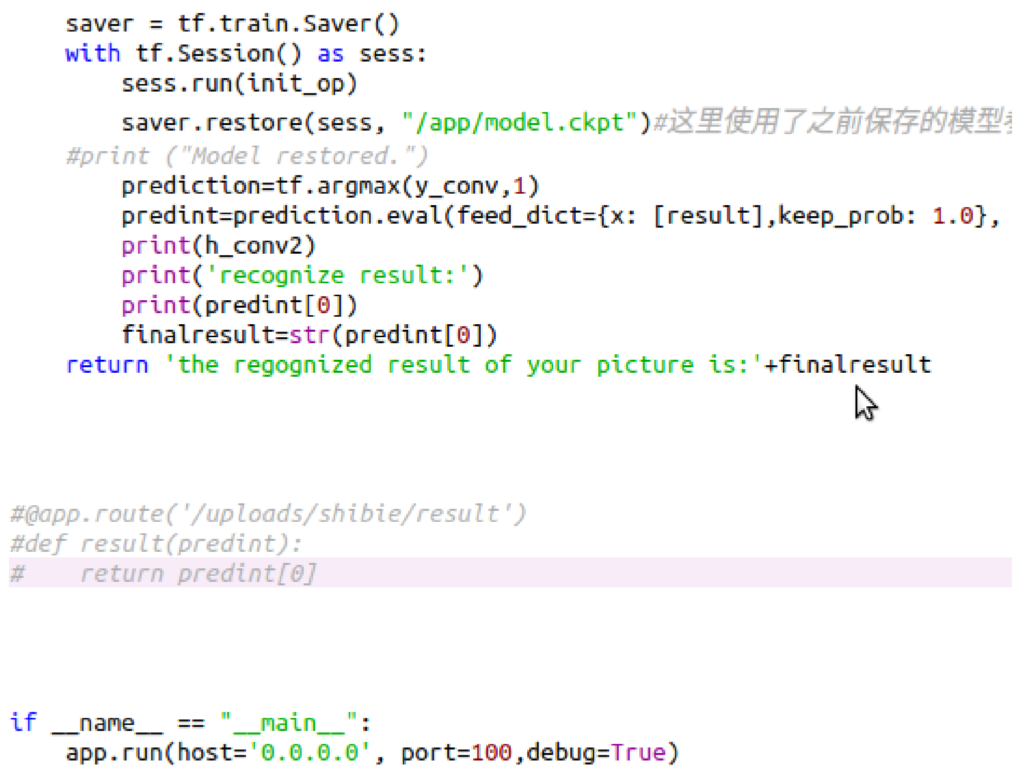
Thus we have the route app in docker, thus we can store pictures temporarily in it and for future use.

when we adapt the code in flask:



```
@app.route('/uploads/<filename>')
def model(filename):
    file_name=os.path.join(app.config['UPLOAD_FOLDER'], filename)#导入
    #in terminal 'mogrify -format png *.jpg' cvert jpg to png
    im = Image.open(file_name)
    im = im.resize((28, 28), Image.ANTIALIAS).convert('L')
    #im.save("/home/niyin/桌面/docker/sample.png")
    plt.imshow(im)
    plt.show()
    tv = list(im.getdata()) #get pixel values
    #normalize pixels to 0 and 1. 0 is pure white, 1 is pure black.
    tva = [ (255-x)*1.0/255.0 for x in tv]
    #print(tva)
    result=tva

    x = tf.placeholder(tf.float32, [None, 784])
    W = tf.Variable(tf.zeros([784, 10]))
    b = tf.Variable(tf.zeros([10]))
        # Define the model (same as when creating the model file)
    W_conv1 = weight_variable([5, 5, 1, 32])
    b_conv1 = bias_variable([32])
```

each route can only have one function, thus we have to define some necessary functions out off the @app.route

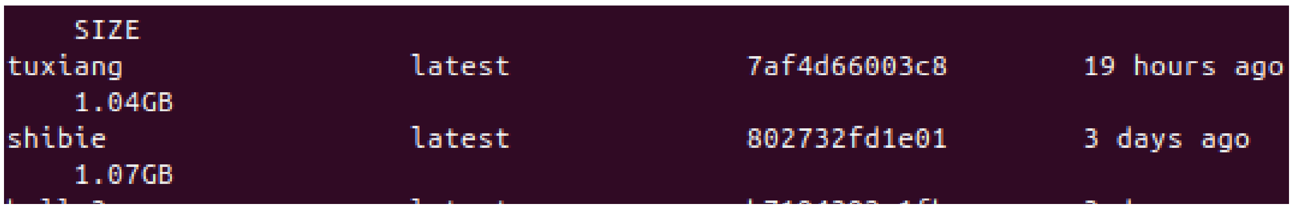


```
    saver = tf.train.Saver()
    with tf.Session() as sess:
        sess.run(init_op)
        saver.restore(sess, "/app/model.ckpt")#这里使用了之前保存的模型
    #print ("Model restored.")
        prediction=tf.argmax(y_conv,1)
        predint=prediction.eval(feed_dict={x: [result],keep_prob: 1.0},
        print(h_conv2)
        print('recognize result:')
        print(predint[0])
        finalresult=str(predint[0])
    return 'the regognized result of your picture is:'+finalresult



#@app.route('/uploads/shibie/result')
#def result(predint):
#    return predint[0]



if __name__ == "__main__":
    app.run(host='0.0.0.0', port=100,debug=True)
```

another thing we have to notice is that we change the address into “/app/model.ckpt”

this is an address that we can find in docker for the model file. Actually ,the “copy” order has copied all file in that local dir to docker workdir /app

then use the order “ docker build -t tuxiang” through the terminal.we can create one image called tuxiang

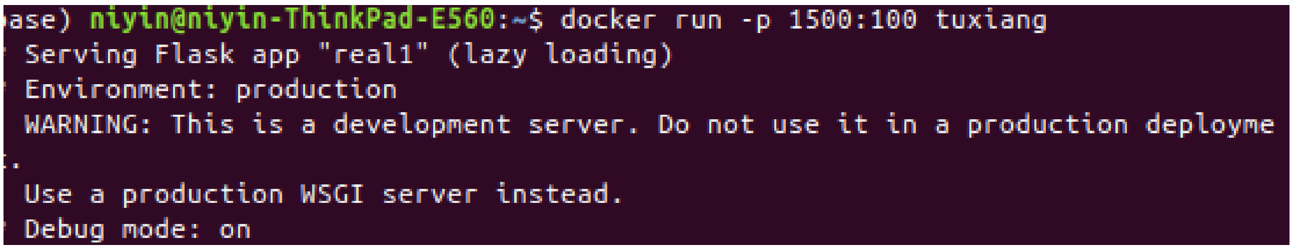


```
    SIZE
tuxiang                  latest              7af4d66003c8        19 hours ago
    1.04GB
shibie                   latest              802732fd1e01        3 days ago
    1.07GB
```

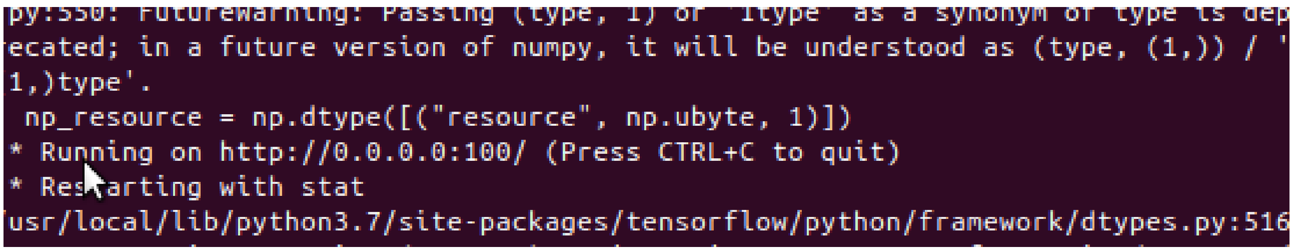
◆ Result presentation

we run this image:

docker run -p 1500:100 tuxiang



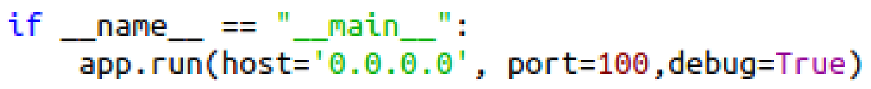
```
ase) niyin@niyin-ThinkPad-E560:~$ docker run -p 1500:100 tuxiang
 Serving Flask app "real1" (lazy loading)
 Environment: production
 WARNING: This is a development server. Do not use it in a production deployme
.
 Use a production WSGI server instead.
 Debug mode: on
```



```
py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is dep
ecated; in a future version of numpy, it will be understood as (type, (1,)) / '
1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
 * Running on http://0.0.0.0:100/ (Press CTRL+C to quit)
 * Restarting with stat
usr/local/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:516
```

we can see that the project can be visit through http://0.0.0.0:100/

this address is in our code



```
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=100,debug=True)
```

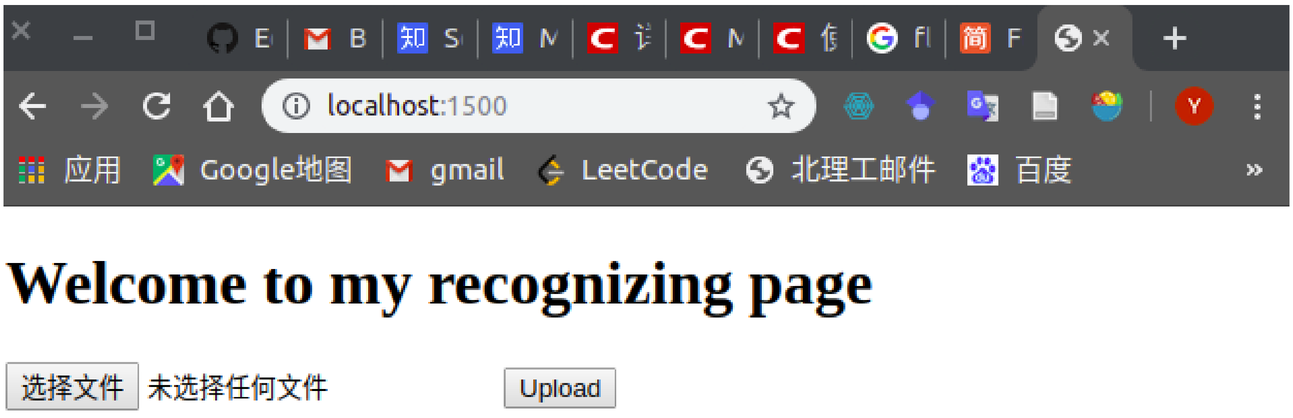
Commonplace is that we can visit through this address, but sometimes some mistakes of limitation will happen if we just visit it.

So we use the order

docker run -p 1500:100 tuxiang

that we can visit the same page through localhost:1500

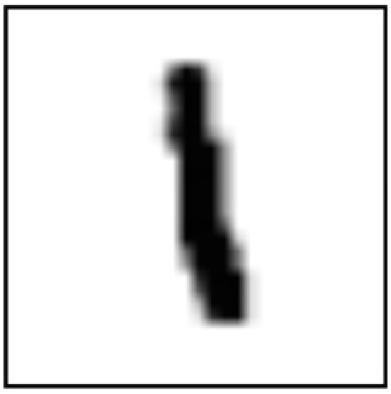
we notice that the 100 is written in our code and the dockerfile, it is what we want others to visit.



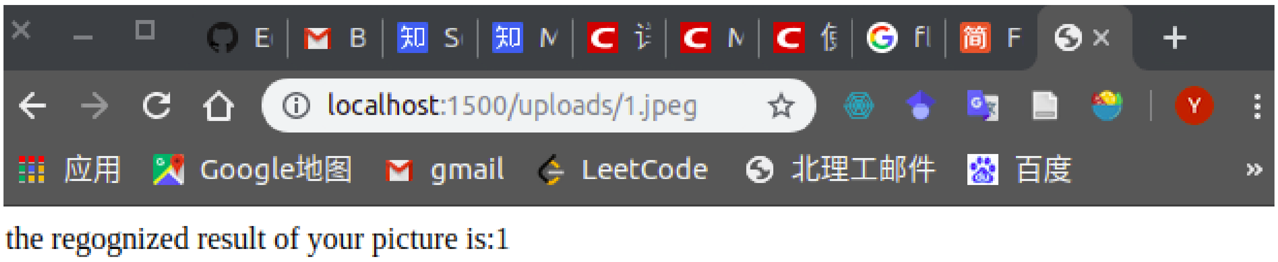
Welcome to my recognizing page

We can click the bottom “选择文件” and select local file from the user’s computer,then we can upload it

we upload the file 1.jpeg



and we can see



the regognized result of your picture is:1

it is correct!

And the route is correct:

localhost:1500/uploads/1.jpeg

since the 1.jpeg is the name of file we upload.

- **Conclusion and feedback**

In this program, I learned the mnist and softmax, which are basic knowledge for me to step into the data analysis field.

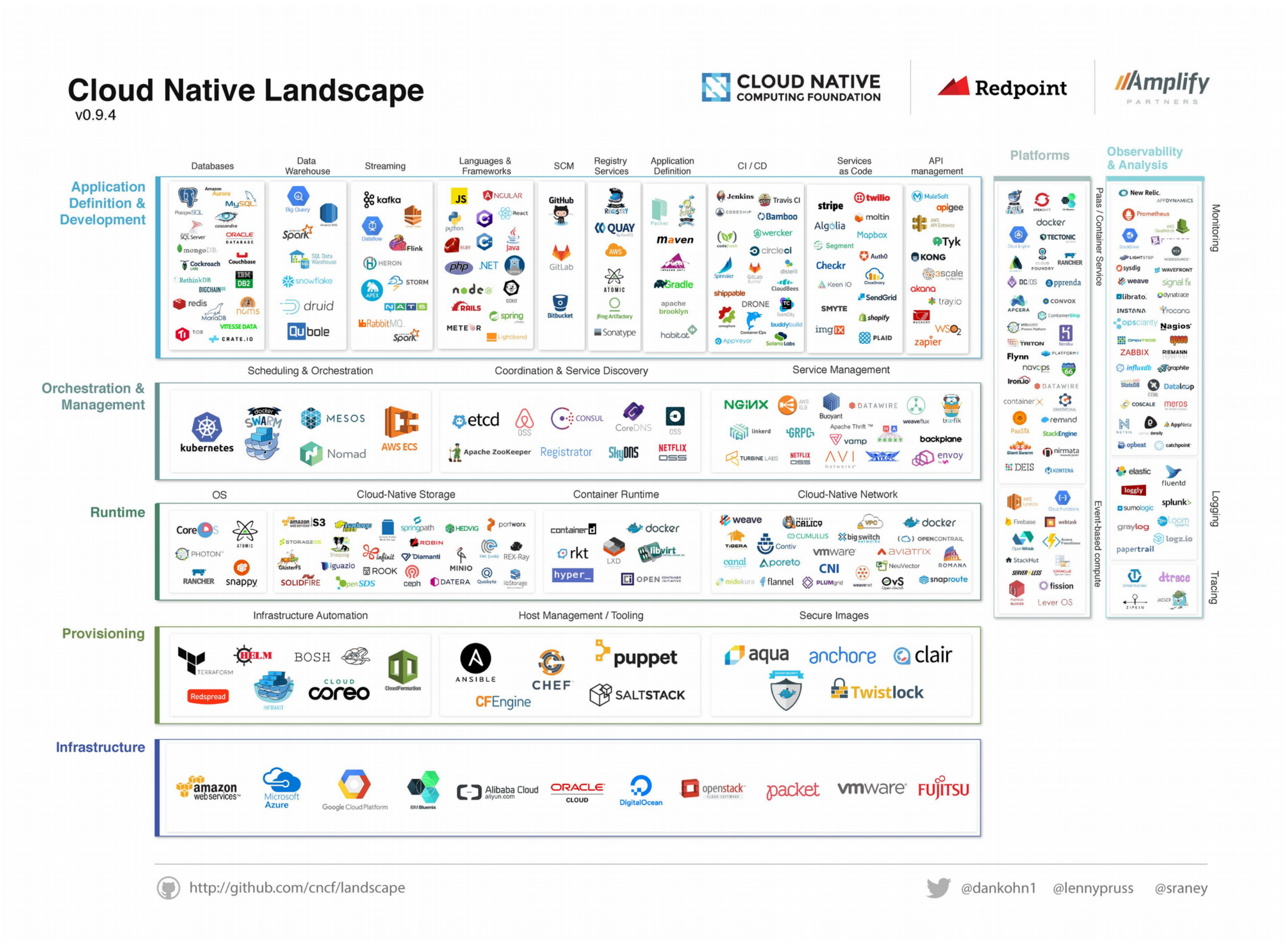
What's more, simple use of flask allows me to know what I probably do in the future, especially in the development of web frame.

In addition, the container tech, docker, is very essential for each one in the AI, CS, data science field. It is the most advanced tech in those fields.

However, the most fortune for me is that

in the last lecture, Dr. zhang introduce the whole knowledge frame for future learning. I consider it very important, since one month is definitely not enough to be professional, but if we know what we are going to do or to learn, we can spend the least and get our goal in the shortest way!

With sincere gratitude to Dr. zhang for showing such a beautiful world to me!


Cloud Native Landscape
v0.9.4
CLOUD NATIVE COMPUTING FOUNDATION
Redpoint
Amplify PARTNERS
Application Definition & Development
Databases
Data Warehouse
Streaming
Languages & Frameworks
SCM
Registry Services
Application Definition
CI / CD
Services as Code
API management
Orchestration & Management
Scheduling & Orchestration
Coordination & Service Discovery
Service Management
Runtime
OS
Cloud-Native Storage
Container Runtime
Cloud-Native Network
Provisioning
Infrastructure Automation
Host Management / Tooling
Secure Images
Infrastructure
Platforms
PaaS / Container Service
Event-based compute
Observability & Analysis
Monitoring
Logging
Tracing
http://github.com/cncf/landscape
@dankohn1
@lennypruss
@sraney