Deploy a face recognition ML project as Docker image

Shengping Jiang

This deployment will build a Docker image with install dlib, face_recognition and a lot of modules

Our face recognition model is based on Dlib and KNN. A set of face images were used to train the KNN model. The training program is face_model_train.py. The program can be executed as below:
(faceprod)$ python face_model_train.py
A trained model face_model_file_frg will be saved in current folder by using pickle

Deployment Process

1 Create a flask app
The load_model() loads a trained ML model
The get_prediction() receives JSON data. A face encoding 128D data is treated as a string in the JSON data. This function will extract the string and convert it to a numpy array. The 128 x 1 array is sent to model.predict() to get a prediction (name)
The upload_file() launches a form of upload file. User can upload a face image from http://localhost:5000/uploads (http://localhost:5000/uploads). The image will be loaded as a dlib image format
The predict_file(image) detects face area and generates a 128D face encoding. The trained ML model will test the encoding and return a name or 'Unknown'
The program is saved as face_app2.py

In [ ]:

```
#ShengpingJiang- Face recognition model as a flask application

import pickle
import os
import numpy as np
from flask import Flask, flash, request, redirect, url_for, send_fr
from werkzeug.utils import secure_filename
from PIL import Image
import face_recognition as frg

#model = None
app = Flask(__name__)


def load_model():
    global model
    # model variable refers to the global variable
    with open('face_model_file_frg', 'rb') as f:
        model = pickle.load(f)


@app.route('/')
def home_endpoint():
    return 'Hello World!'


@app.route('/predict', methods=['GET','POST'])
def get_prediction():
    dist_threshold = 0.4
    name=''
    # Works only for a single sample
    if request.method == 'POST':
        data = request.get_json()  # Get data posted as a json
        #data[0] means 1st {} in the JSON data [{..},{..}]. data[0]
        #the value of key 'encoding' in data[0]
        #print(type(data[0]['encoding']))
        #print(data[0]['encoding'])
        #The value of the key 'encoding' is a string '[-0.17077433
        str1 = data[0]['encoding']
        # str1[1:-1] from '[-0.17077433  0.086519...]' to '-0.17077
        # np.fromstring changes a string '-0.17077433  0.086519...'
        # [-0.17077433  0.086519...]
        encoding = np.fromstring(str1[1:-1], dtype=float, sep=' ')
        #print("ecoding type:", type(encoding))
        #print(encoding)

        # reshape(1,-1) change [-0.17077433  0.086519...] to [[-0.1
        xt = encoding.reshape(1,-1)
        #print('xt:', xt)
        closest_distance = model.kneighbors(xt, n_neighbors=1, retu
        #print("closest_distance[0][0][0]:",closest_distance[0][0][
        if closest_distance[0][0][0] <= dist_threshold :
    # model.predict(xt) returns a string list ['name']
    # model.predict(xt)[0] returns 'name'
            name = model.predict(xt)[0]
            print('name:', name)
```

```
        else:
            name = "Unknown"
    elif request.method == 'GET':
        print("Shengping")

    return name

ALLOWED_EXTENSIONS = ['jpg', 'jpeg', 'gif']
UPLOAD_FOLDER = './uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

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

@app.route('/uploads', methods=['GET', 'POST'])
def upload_file():
    name = ""
    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit an empty part without filename
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
    #Uncomment below two lines will save uploaded file in './upload
            #fpath = os.path.join(app.config['UPLOAD_FOLDER'], file
            #file.save(fpath)
    #file.stream is a file-like object. And load_image_file() needs
    # or file-like object
            image = frg.load_image_file(file.stream, mode='RGB')
            print("type of image1:", type(image))
            name = predict_file(image)
            return render_template('prediction.html', value=name)

    elif request.method == 'GET':
        print("Shengping")

    return '''
    <!doctype html>
    <title>Upload new File</title>
    <h1>Upload new File</h1>
    <form method=post enctype=multipart/form-data>
      <input type=file name=file>
      <input type=submit value=Upload>
    </form>
    '''
@app.route('/uploads/<filename>')
def uploaded_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filenam

```

```
def predict_file(image):
    dist_threshold = 0.4
    print("type of image2:", type(image))
    name=''
    # face_location: (top, right, bottom, left)
    f_location = frg.face_locations(image, model='cnn')
    if len(f_location) != 1:
      return 'Incorrect face image!'
    print("type of f_location:", type(f_location))
    print("f_location:", f_location)
    encoding = frg.face_encodings(image, known_face_locations=f_loc
    if len(encoding) == 0:
      return 'No face encording'
    else:
      encoding = encoding[0]
    print("encoding type:", type(encoding))
    print(encoding)
    # reshape(1,-1) change [-0.17077433  0.086519...] to [[-0.17077
    xt = encoding.reshape(1,-1)
    #print('xt:', xt)
    closest_distance = model.kneighbors(xt, n_neighbors=1, return_d
    #print("closest_distance[0][0][0]:",closest_distance[0][0][0])
    if closest_distance[0][0][0] <= dist_threshold :
    # model.predict(xt) returns a string list ['name']
    # model.predict(xt)[0] returns 'name'
      name = model.predict(xt)[0]
      print('name:', name)
    else:
      name = "Unknown"
    return name


if __name__ == '__main__':
    load_model()  # load model at the beginning once only
    app.run(host='0.0.0.0', port=3000)

```

2 Test face_app.py in faceprod virtualenv

2.1 Create a virtual env faceprod and install packages
mkvirtualenv faceprod -p python3
(faceprod) pip install numpy
(faceprod) pip install flask
(faceprod) pip install pickle-mixin
(faceprod) pip install sklearn
(faceprod) pip install dlib==19.21.0
(faceprod) pip install face-recognition==1.3.0
(faceprod) pip install opencv-python
....

(faceprod) pip freeze > faceprod_list2.txt

2.2 Launch the flask app face_app2.py
(faceprod)$ python face_app.py

2.3 Open another terminal. Send test data (dlib face encoding 128D vector) to web 0.0.0.0:3000/predict, and test the model

$ curl -X POST 0.0.0.0:3000/predict -H 'Content-Type: application/json' -d '[{"encoding": "
[-0.17077433 0.086519 0.04608656 0.02226515 -0.10071052 0.0246949 -0.09879136
-0.08271502 0.15330137 -0.1101086 0.2084657 0.0172283 -0.18812549 0.00964276
-0.06756912 0.11148367 -0.11918792 -0.07723383 -0.05200598 -0.01760992 0.0567386
0.04599836 0.03339319 0.04884979 -0.10915887 -0.33869374 -0.10735007 -0.11223182
0.08643846 -0.07478593 -0.05546422 -0.08678006 -0.11504613 0.01475477 0.01169325
0.15265159 -0.02465688 -0.06824835 0.21678171 -0.03042633 -0.19874264 -0.01212559
-0.02762683 0.26414317 0.13703299 0.0334272 0.01637992 -0.10932572 0.09580361
-0.21135658 0.11234938 0.1291863 0.0340074 0.03284376 0.09014399 -0.17272305
0.01153929 0.14709072 -0.14064969 0.02695761 0.03161349 0.01307983 -0.0100578
-0.05213601 0.20376676 0.14580815 -0.11039062 -0.15493403 0.11541102 -0.2119666
0.0013991 0.08922509 -0.11429761 -0.22043382 -0.28854343 0.04549009 0.44805536
0.20364918 -0.16662233 0.02062135 -0.00946902 -0.02268174 0.16432424 0.10247331
-0.08463222 0.0589206 -0.11151487 0.04075154 0.17744561 0.00353054 -0.0321093
0.19991624 0.01635876 0.06169297 0.05581587 0.04786064 -0.07188784 -0.04009981
-0.1177263 -0.01570286 0.08082893 -0.0241716 0.03095182 0.11278267 -0.16012146
0.1034444 -0.01475013 -0.01811141 0.03154366 0.02885633 -0.14979976 -0.0449345
0.21942021 -0.22967488 0.15503235 0.15902625 0.02446658 0.15540583 0.12920454
0.0752509 -0.01832712 -0.00534262 -0.19305748 -0.00229457 0.01291393 -0.05213701
0.07341617 0.01301993]"}]'

Note: above command is one line. No return is in the line

2.4 Test http://0.0.0.0:3000/uploads (http://0.0.0.0:3000/uploads)
Refresh http://0.0.0.0:3000/uploads (http://0.0.0.0:3000/uploads)
Upload an image from the web
Check the prediction

3 Create a Dockerfile3
Use text editor to create Dockerfile3 and put in lines below:
#The Dockerfile3 will be used to create a docker image

FROM python:3.6-slim-stretch

RUN apt-get -y update
RUN apt-get install -y --fix-missing \
build-essential \
cmake \
gfortran \
git \
wget \
curl \
graphicsmagick \
libgraphicsmagick1-dev \
libatlas-base-dev \
libavcodec-dev \

```
libavformat-dev \
libgtk2.0-dev \
libjpeg-dev \
liblapack-dev \
libswscale-dev \
pkg-config \
python3-dev \
python3-numpy \
software-properties-common \
zip \
&& apt-get clean && rm -rf /tmp/* /var/tmp/*

RUN cd ~ && \
mkdir -p dlib && \
git clone -b 'v19.9' --single-branch https://github.com/davisking/dlib.git
(https://github.com/davisking/dlib.git) dlib/ && \
cd dlib/ && \
python3 setup.py install --yes USE_AVX_INSTRUCTIONS

#Shengping's project

COPY ./face_app2.py /deploy/
COPY ./faceprod_list3.txt /deploy/
COPY ./face_model_file_frg /deploy/
COPY ./LICENSE /deploy/
COPY ./README2.md /deploy/
ADD ./templates /deploy/templates

WORKDIR /deploy/
RUN pip install -r faceprod_list3.txt
EXPOSE 3000
ENTRYPOINT ["python", "face_app2.py"]
```

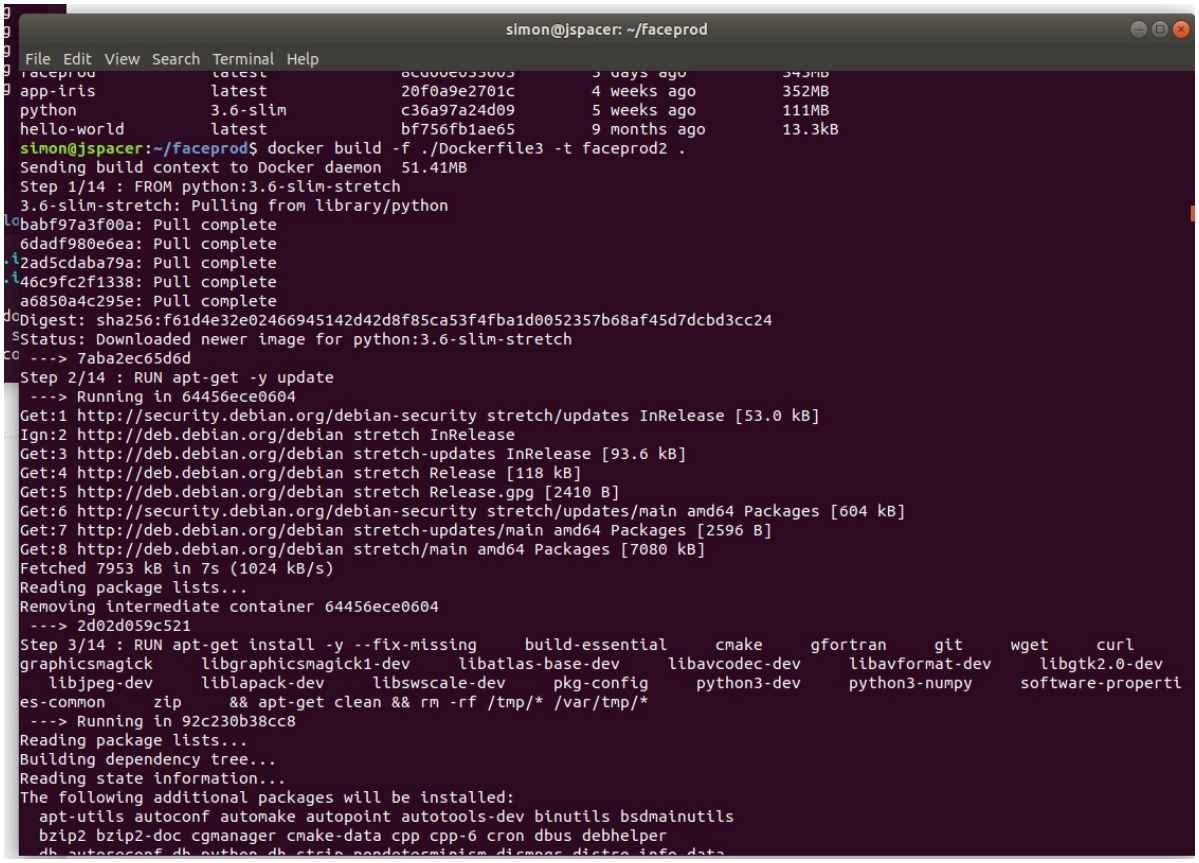
The faceprod_list3.txt was generated from the virtualenv faceprod, but removed pip install dlib as the dlib is complied and installed during building the docker image

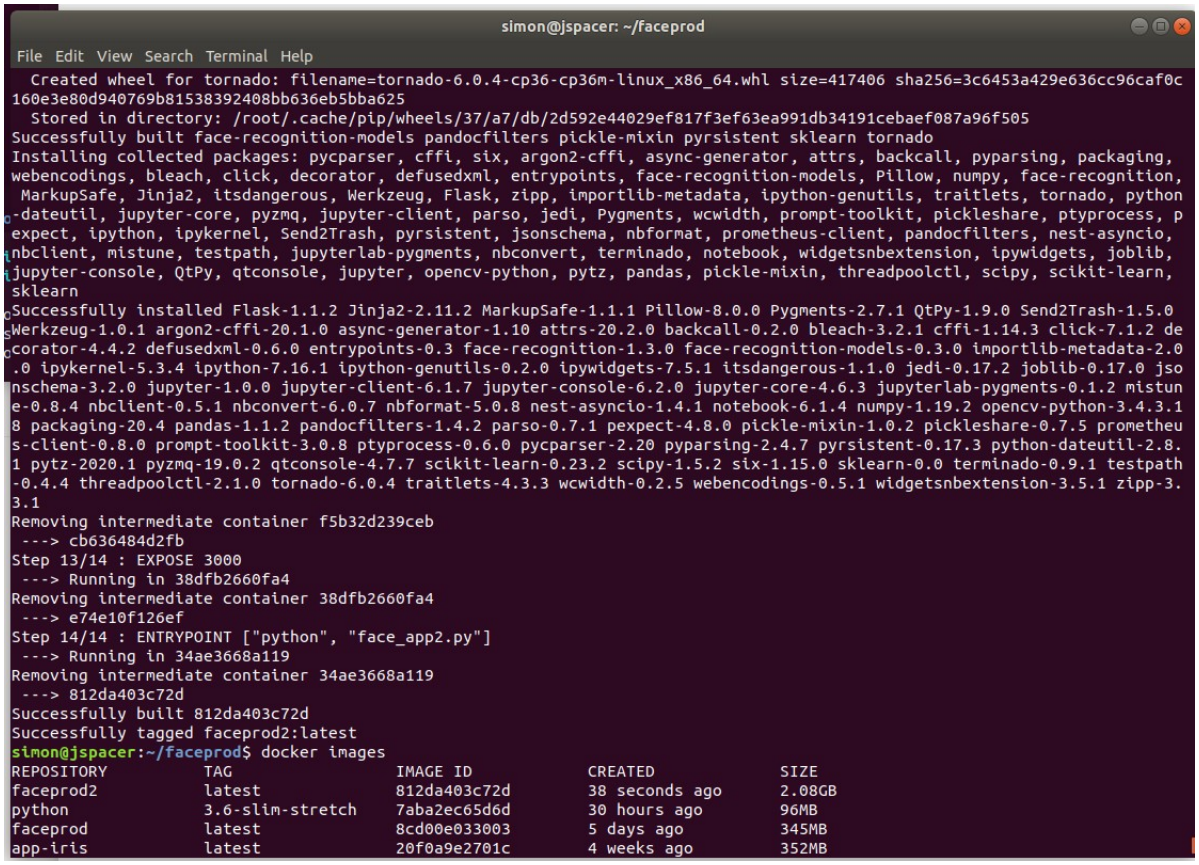
4 Create Docker image
Get out the virtual env faceprod. Check docker is running

```
$ docker run hello-world
Got permission denied...

$ sudo chmod 666 /var/run/docker.sock
--this command fix above error

$ docker run hello-world
Hello from Docker!
```

Create docker image

~/faceprod$ docker build -t faceprod .

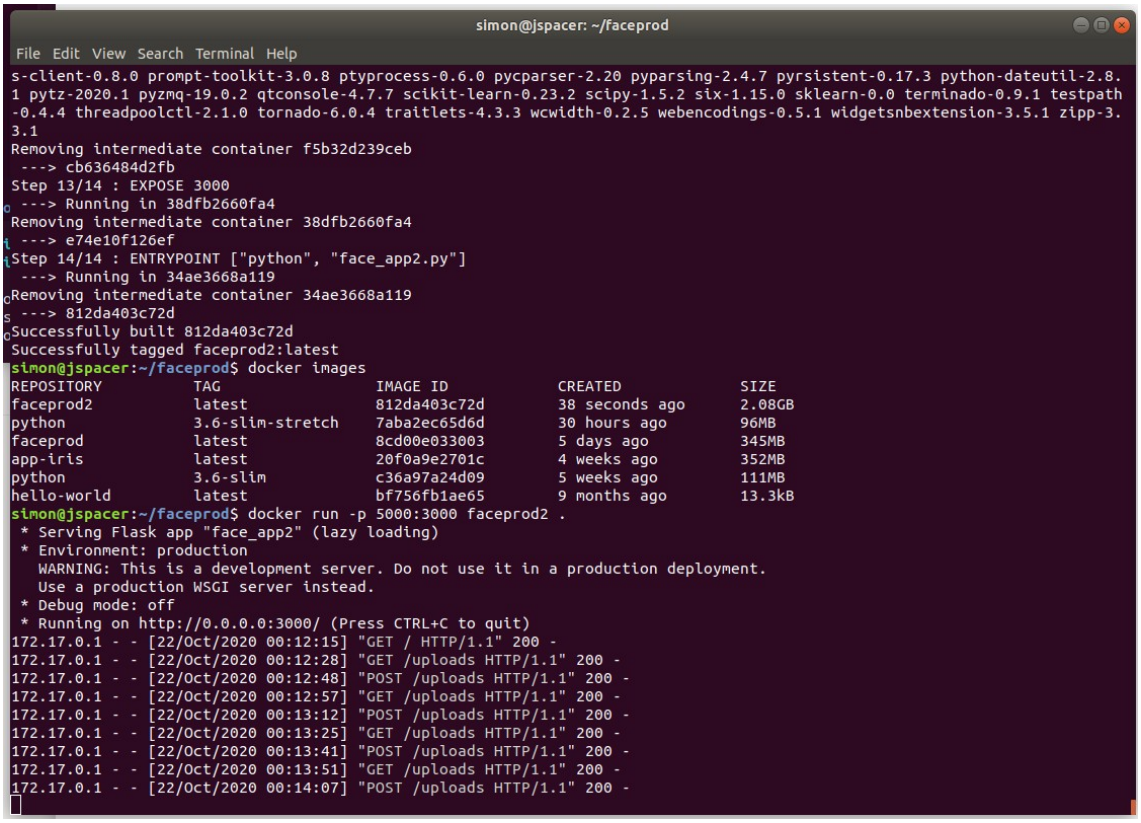




5 Launch and test docker image

5.1 Test docker image from command line
Run docker image. 1st 5000 is local machine port. 2nd 3000 is the port assigned in face_app2.py (it is inside docker image)
~/faceprod$ docker run -p 5000:3000 faceprod2 .

- Serving Flask app "face_app" (lazy loading)
- Environment: production
  WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
- Debug mode: off
- Running on http://0.0.0.0:5000/ (http://0.0.0.0:5000/) (Press CTRL+C to quit)

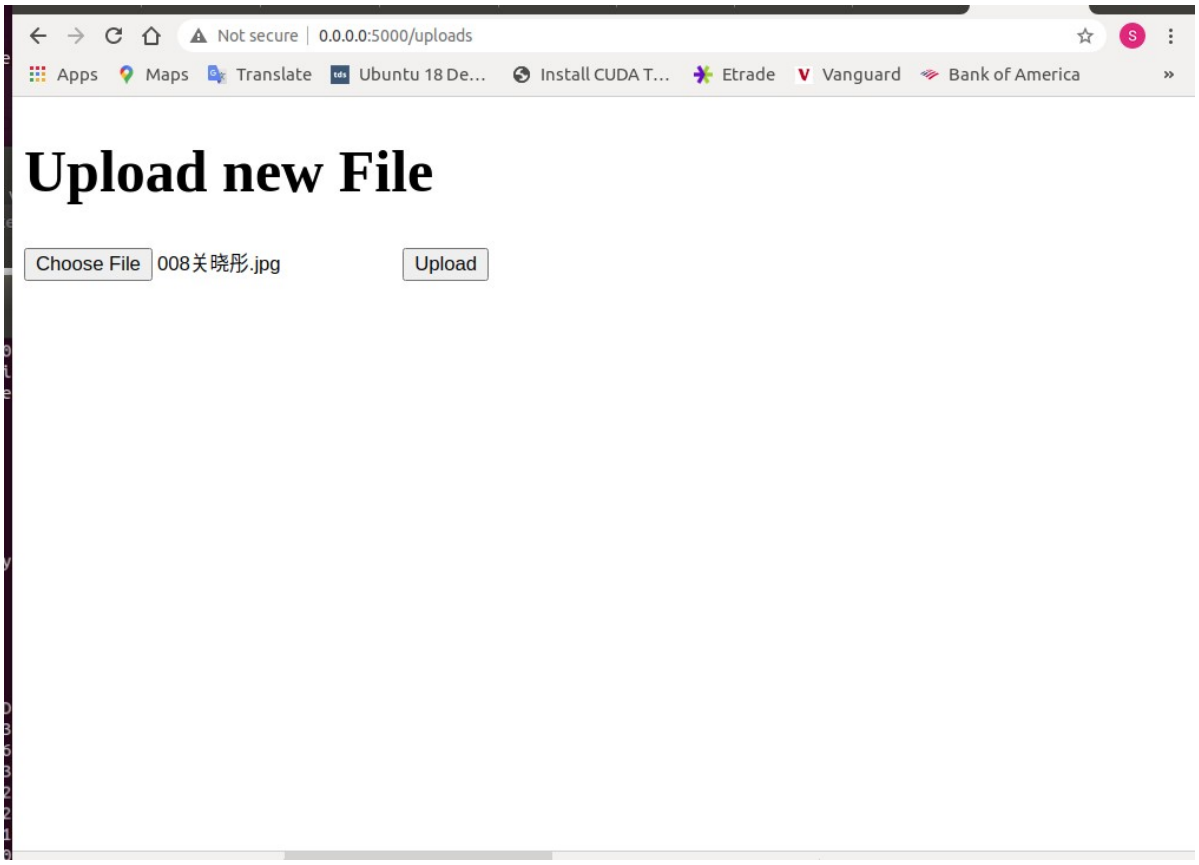


In another terminal, send test data and get response
$ curl -X POST 0.0.0.0:5000/predict -H 'Content-Type: application/json' -d '[{"encoding": "
[-0.17077433 0.086519 0.04608656 0.02226515 -0.10071052 0.0246949 -0.09879136
-0.08271502 0.15330137 -0.1101086 0.2084657 0.0172283 -0.18812549 0.00964276
-0.06756912 0.11148367 -0.11918792 -0.07723383 -0.05200598 -0.01760992 0.0567386
0.04599836 0.03339319 0.04884979 -0.10915887 -0.33869374 -0.10735007 -0.11223182
0.08643846 -0.07478593 -0.05546422 -0.08678006 -0.11504613 0.01475477 0.01169325
0.15265159 -0.02465688 -0.06824835 0.21678171 -0.03042633 -0.19874264 -0.01212559
-0.02762683 0.26414317 0.13703299 0.0334272 0.01637992 -0.10932572 0.09580361
-0.21135658 0.11234938 0.1291863 0.0340074 0.03284376 0.09014399 -0.17272305
0.01153929 0.14709072 -0.14064969 0.02695761 0.03161349 0.01307983 -0.0100578

-0.05213601 0.20376676 0.14580815 -0.11039062 -0.15493403 0.11541102 -0.2119666
0.0013991 0.08922509 -0.11429761 -0.22043382 -0.28854343 0.04549009 0.44805536
0.20364918 -0.16662233 0.02062135 -0.00946902 -0.02268174 0.16432424 0.10247331
-0.08463222 0.0589206 -0.11151487 0.04075154 0.17744561 0.00353054 -0.0321093
0.19991624 0.01635876 0.06169297 0.05581587 0.04786064 -0.07188784 -0.04009981
-0.1177263 -0.01570286 0.08082893 -0.0241716 0.03095182 0.11278267 -0.16012146
0.1034444 -0.01475013 -0.01811141 0.03154366 0.02885633 -0.14979976 -0.0449345
0.21942021 -0.22967488 0.15503235 0.15902625 0.02446658 0.15540583 0.12920454
0.0752509 -0.01832712 -0.00534262 -0.19305748 -0.00229457 0.01291393 -0.05213701
0.07341617 0.01301993]"}]'

An answer from the flask app:
004郭坚

5.2 Test docker image from website
Go to [http://0.0.0.0:5000/uploads (http://0.0.0.0:5000/uploads)](http://0.0.0.0:5000/uploads)
Click choose file, and upload a face image file from local
Screenshort for testing docker image and get an answer:



The ML model will return a prediction: a name or 'unknown'


Not secure | 0.0.0.0:5000/uploads
Apps
Maps
Translate
Ubuntu 18 De...
Install CUDA T...
Etrade
Vanguard
Bank of America
Prediction: 008关晓彤