

# 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 [ ]: 1 #ShengpingJiang- Face recognition model as a flask application
2
3 import pickle
4 import os
5 import numpy as np
6 from flask import Flask, flash, request, redirect, url_for, send_fr
7 from werkzeug.utils import secure_filename
8 from PIL import Image
9 import face_recognition as frg
10
11 #model = None
12 app = Flask(__name__)
13
14
15 def load_model():
16     global model
17     # model variable refers to the global variable
18     with open('face_model_file_frg', 'rb') as f:
19         model = pickle.load(f)
20
21
22 @app.route('/')
23 def home_endpoint():
24     return 'Hello World!'
25
26
27 @app.route('/predict', methods=['GET', 'POST'])
28 def get_prediction():
29     dist_threshold = 0.4
30     name=''
31     # Works only for a single sample
32     if request.method == 'POST':
33         data = request.get_json() # Get data posted as a json
34         #data[0] means 1st {} in the JSON data [{..},{..}]. data[0]
35         #the value of key 'encoding' in data[0]
36         #print(type(data[0]['encoding']))
37         #print(data[0]['encoding'])
38         #The value of the key 'encoding' is a string '[-0.17077433
39         str1 = data[0]['encoding']
40         # str1[1:-1] from '[-0.17077433 0.086519...]' to '-0.17077
41         # np.fromstring changes a string '-0.17077433 0.086519...'
42         # [-0.17077433 0.086519...]
43         encoding = np.fromstring(str1[1:-1], dtype=float, sep=' ')
44         #print("encoding type:", type(encoding))
45         #print(encoding)
46
47         # reshape(1,-1) change [-0.17077433 0.086519...] to [[-0.1
48         xt = encoding.reshape(1,-1)
49         #print('xt:', xt)
50         closest_distance = model.kneighbors(xt, n_neighbors=1, retu
51         #print("closest_distance[0][0][0]:",closest_distance[0][0][
52         if closest_distance[0][0][0] <= dist_threshold :
53             # model.predict(xt) returns a string list ['name']
54             # model.predict(xt)[0] returns 'name'
55             name = model.predict(xt)[0]
56             print('name:', name)

```

```

57         else:
58             name = "Unknown"
59     elif request.method == 'GET':
60         print("Shengping")
61
62     return name
63
64 ALLOWED_EXTENSIONS = ['jpg', 'jpeg', 'gif']
65 UPLOAD_FOLDER = './uploads'
66 app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
67
68 def allowed_file(filename):
69     return '.' in filename and \
70         filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
71
72 @app.route('/uploads', methods=['GET', 'POST'])
73 def upload_file():
74     name = ""
75     if request.method == 'POST':
76         # check if the post request has the file part
77         if 'file' not in request.files:
78             flash('No file part')
79             return redirect(request.url)
80         file = request.files['file']
81         # if user does not select file, browser also
82         # submit an empty part without filename
83         if file.filename == '':
84             flash('No selected file')
85             return redirect(request.url)
86         if file and allowed_file(file.filename):
87             filename = secure_filename(file.filename)
88             # Uncomment below two lines will save uploaded file in './upload
89             # fpath = os.path.join(app.config['UPLOAD_FOLDER'], file
90             # file.save(fpath)
91             # file.stream is a file-like object. And load_image_file() needs
92             # or file-like object
93             image = frg.load_image_file(file.stream, mode='RGB')
94             print("type of image1:", type(image))
95             name = predict_file(image)
96             return render_template('prediction.html', value=name)
97
98     elif request.method == 'GET':
99         print("Shengping")
100
101     return '''
102     <!doctype html>
103     <title>Upload new File</title>
104     <h1>Upload new File</h1>
105     <form method=post enctype=multipart/form-data>
106         <input type=file name=file>
107         <input type=submit value=Upload>
108     </form>
109     '''
110 @app.route('/uploads/<filename>')
111 def uploaded_file(filename):
112     return send_from_directory(app.config['UPLOAD_FOLDER'], filename)
113

```

```

114 def predict_file(image):
115     dist_threshold = 0.4
116     print("type of image2:", type(image))
117     name=''
118     # face_location: (top, right, bottom, left)
119     f_location = frg.face_locations(image, model='cnn')
120     if len(f_location) != 1:
121         return 'Incorrect face image!'
122     print("type of f_location:", type(f_location))
123     print("f_location:", f_location)
124     encoding = frg.face_encodings(image, known_face_locations=f_loc
125     if len(encoding) == 0:
126         return 'No face encoding'
127     else:
128         encoding = encoding[0]
129         print("encoding type:", type(encoding))
130         print(encoding)
131         # reshape(1,-1) change [-0.17077433  0.086519...] to [[-0.17077
132         xt = encoding.reshape(1,-1)
133         #print('xt:', xt)
134         closest_distance = model.kneighbors(xt, n_neighbors=1, return_d
135         #print("closest_distance[0][0][0]:",closest_distance[0][0][0])
136         if closest_distance[0][0][0] <= dist_threshold :
137             # model.predict(xt) returns a string list ['name']
138             # model.predict(xt)[0] returns 'name'
139             name = model.predict(xt)[0]
140             print('name:', name)
141         else:
142             name = "Unknown"
143         return name
144
145
146 if __name__ == '__main__':
147     load_model() # load model at the beginning once only
148     app.run(host='0.0.0.0', port=3000)
149

```

## 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 compiled 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 .

```

simon@jspace: ~/faceprod
File Edit View Search Terminal Help
faceprod:latest 8cd00e033003 5 days ago 345MB
app-iris:latest 20f0a9e2701c 4 weeks ago 352MB
python:3.6-slim c36a97a24d09 5 weeks ago 111MB
hello-world:latest bf756fb1ae65 9 months ago 13.3kB
simon@jspace:~/faceprod$ docker build -f ./Dockerfile3 -t faceprod2 .
Sending build context to Docker daemon 51.41MB
Step 1/14 : FROM python:3.6-slim-stretch
3.6-slim-stretch: Pulling from library/python
babf97a3f00a: Pull complete
6dadf980e6ea: Pull complete
12ad5cdaba79a: Pull complete
146c9fc2f1338: Pull complete
a6850a4c295e: Pull complete
Digest: sha256:f61d4e32e02466945142d42d8f85ca53f4fba1d0052357b68af45d7dcbd3cc24
Status: Downloaded newer image for python:3.6-slim-stretch
--> 7aba2ec65d6d
Step 2/14 : RUN apt-get -y update
--> Running in 64456ece0604
Get:1 http://security.debian.org/debian-security stretch/updates InRelease [53.0 kB]
Ign:2 http://deb.debian.org/debian stretch InRelease
Get:3 http://deb.debian.org/debian stretch/updates InRelease [93.6 kB]
Get:4 http://deb.debian.org/debian stretch Release [118 kB]
Get:5 http://deb.debian.org/debian stretch Release.gpg [2410 B]
Get:6 http://security.debian.org/debian-security stretch/updates/main amd64 Packages [604 kB]
Get:7 http://deb.debian.org/debian stretch/updates/main amd64 Packages [2596 B]
Get:8 http://deb.debian.org/debian stretch/main amd64 Packages [7080 kB]
Fetched 7953 kB in 7s (1024 kB/s)
Reading package lists...
Removing intermediate container 64456ece0604
--> 2d02d059c521
Step 3/14 : RUN apt-get install -y --fix-missing build-essential cmake gfortran git wget curl
graphicsmagick libgraphicsmagick1-dev libatlas-base-dev libavcodec-dev libavformat-dev libgdk-pixbuf2.0-dev
libjpeg-dev liblapack-dev libswscale-dev pkg-config python3-dev python3-numpy software-properties
es-common zip && apt-get clean && rm -rf /tmp/* /var/tmp/*
--> Running in 92c230b38cc8
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
  apt-utils autoconf automake autopoint autotools-dev binutils bsdmainutils
  bzip2 bzip2-doc cgmanager cmake-data cpp cpp-6 cron dbus debhelper
  dh-autoreconf dh-python dh-strip-nondeterminism dmccap distro-info-data

```

```

simon@jspace: ~/faceprod
File Edit View Search Terminal Help
Created wheel for tornado: filename=tornado-6.0.4-cp36-cp36m-linux_x86_64.whl size=417406 sha256=3c6453a429e636cc96caf0c
160e3e80d940769b81538392408bb63eb5bba625
Stored in directory: /root/.cache/pip/wheels/37/a7/db/2d592e44029ef817f3ef63ea991db34191cebaef087a96f505
Successfully built face-recognition-models pandocfilters pickle-mixin pyrsistent sklearn tornado
Installing collected packages: pycparser, cffi, six, argon2-cffi, async-generator, attrs, backcall, pyparsing, packaging,
webencodings, bleach, click, decorator, defusedxml, entrypoints, face-recognition-models, Pillow, numpy, face-recognition,
MarkupSafe, Jinja2, itsdangerous, Werkzeug, Flask, zipp, importlib-metadata, ipython-genutils, traitlets, tornado, python
-dateutil, jupyter-core, pyzmq, jupyter-client, parso, jedi, Pygments, wcwidth, prompt-toolkit, pickleshare, ptyprocess, p
expect, ipython, ipykernel, Send2Trash, pyrsistent, jsonschema, nbformat, prometheus-client, pandocfilters, nest-asyncio,
nbclient, mistune, testpath, jupyterlab-pygments, nbconvert, terminado, notebook, widgetsnbextension, ipywidgets, joblib,
jupyter-console, QtPy, qtconsole, jupyter, opencv-python, pytz, pandas, pickle-mixin, threadpoolctl, scipy, scikit-learn,
sklearn
Successfully installed Flask-1.1.2 Jinja2-2.11.2 MarkupSafe-1.1.1 Pillow-8.0.0 Pygments-2.7.1 QtPy-1.9.0 Send2Trash-1.5.0
Werkzeug-1.0.1 argon2-cffi-20.1.0 async-generator-1.10 attrs-20.2.0 backcall-0.2.0 bleach-3.2.1 cffi-1.14.3 click-7.1.2 de
corator-4.4.2 defusedxml-0.6.0 entrypoints-0.3 face-recognition-1.3.0 face-recognition-models-0.3.0 importlib-metadata-2.0
.0 ipykernel-5.3.4 ipython-7.16.1 ipython-genutils-0.2.0 ipywidgets-7.5.1 itsdangerous-1.1.0 jedi-0.17.2 joblib-0.17.0 jso
nschema-3.2.0 jupyter-1.0.0 jupyter-client-6.1.7 jupyter-console-6.2.0 jupyter-core-4.6.3 jupyterlab-pygments-0.1.2 mistun
e-0.8.4 nbclient-0.5.1 nbconvert-6.0.7 nbformat-5.0.8 nest-asyncio-1.4.1 notebook-6.1.4 numpy-1.19.2 opencv-python-3.4.3.1
packaging-20.4 pandas-1.1.2 pandocfilters-1.4.2 parso-0.7.1 pexpect-4.8.0 pickle-mixin-1.0.2 pickleshare-0.7.5 prometheu
s-client-0.8.0 prompt-toolkit-3.0.8 ptyprocess-0.6.0 pycparser-2.20 pyparsing-2.4.7 pyrsistent-0.17.3 python-dateutil-2.8.
1 pytz-2020.1 pyzmq-19.0.2 qtconsole-4.7.7 scikit-learn-0.23.2 scipy-1.5.2 six-1.15.0 sklearn-0.0 terminado-0.9.1 testpath
-0.4.4 threadpoolctl-2.1.0 tornado-6.0.4 traitlets-4.3.3 wcwidth-0.2.5 webencodings-0.5.1 widgetsnbextension-3.5.1 zipp-3.
1
Removing intermediate container f5b32d239ceb
--> cb636484d2fb
Step 13/14 : EXPOSE 3000
--> Running in 38dfb2660fa4
Removing intermediate container 38dfb2660fa4
--> e74e10f126ef
Step 14/14 : ENTRYPOINT ["python", "face_app2.py"]
--> Running in 34ae3668a119
Removing intermediate container 34ae3668a119
--> 812da403c72d
Successfully built 812da403c72d
Successfully tagged faceprod2:latest
simon@jspace:~/faceprod$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
faceprod2 latest 812da403c72d 38 seconds ago 2.08GB
python 3.6-slim-stretch 7aba2ec65d6d 30 hours ago 96MB
faceprod latest 8cd00e033003 5 days ago 345MB
app-iris latest 20f0a9e2701c 4 weeks ago 352MB

```

## 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)

```

simon@jspacer: ~/faceprod
File Edit View Search Terminal Help
s-client-0.8.0 prompt-toolkit-3.0.8 ptyprocess-0.6.0 pyparsing-2.2.0 pyparsing-2.4.7 pyrsistent-0.17.3 python-dateutil-2.8.1
1 pytz-2020.1 pyzmq-19.0.2 qtconsole-4.7.7 scikit-learn-0.23.2 scipy-1.5.2 six-1.15.0 sklearn-0.0 terminado-0.9.1 testpath-0.4.4
threadpoolctl-2.1.0 tornado-6.0.4 traitlets-4.3.3 wcwidth-0.2.5 webencodings-0.5.1 widgetsnbextension-3.5.1 zipp-3.1
Removing intermediate container f5b32d239ceb
--> cb636484d2fb
Step 13/14 : EXPOSE 3000
--> Running in 38dfb2660fa4
Removing intermediate container 38dfb2660fa4
--> e74e10f126ef
Step 14/14 : ENTRYPOINT ["python", "face_app2.py"]
--> Running in 34ae3668a119
Removing intermediate container 34ae3668a119
--> 812da403c72d
Successfully built 812da403c72d
Successfully tagged faceprod2:latest
simon@jspacer:~/faceprod$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
faceprod2           latest             812da403c72d       38 seconds ago     2.08GB
python              3.6-slim-stretch  7aba2ec65d6d       30 hours ago       96MB
faceprod            latest            8cd00e033003       5 days ago         345MB
app-iris            latest            20f0a9e2701c       4 weeks ago        352MB
python              3.6-slim          c36a97a24d09       5 weeks ago        111MB
hello-world         latest            bf756fb1ae65       9 months ago       13.3kB
simon@jspacer:~/faceprod$ docker run -p 5000:3000 faceprod2 .
* Serving Flask app "face_app2" (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://0.0.0.0:3000/ (Press CTRL+C to quit)
172.17.0.1 - - [22/Oct/2020 00:12:15] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [22/Oct/2020 00:12:28] "GET /uploads HTTP/1.1" 200 -
172.17.0.1 - - [22/Oct/2020 00:12:48] "POST /uploads HTTP/1.1" 200 -
172.17.0.1 - - [22/Oct/2020 00:12:57] "GET /uploads HTTP/1.1" 200 -
172.17.0.1 - - [22/Oct/2020 00:13:12] "POST /uploads HTTP/1.1" 200 -
172.17.0.1 - - [22/Oct/2020 00:13:25] "GET /uploads HTTP/1.1" 200 -
172.17.0.1 - - [22/Oct/2020 00:13:41] "POST /uploads HTTP/1.1" 200 -
172.17.0.1 - - [22/Oct/2020 00:13:51] "GET /uploads HTTP/1.1" 200 -
172.17.0.1 - - [22/Oct/2020 00:14:07] "POST /uploads HTTP/1.1" 200 -

```

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:

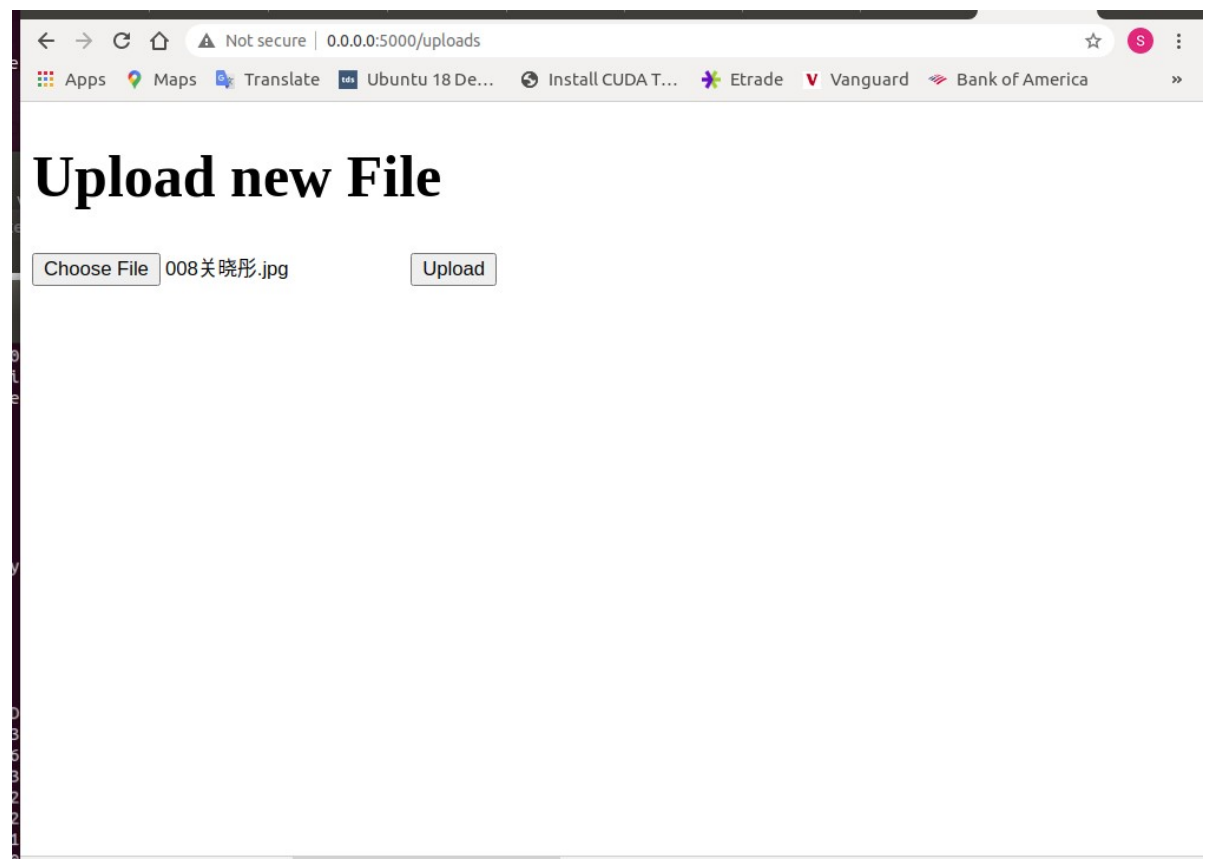
004郭坚

## 5.2 Test docker image from website

Go to <http://0.0.0.0:5000/uploads> (<http://0.0.0.0:5000/uploads>).

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'

