

# Báo cáo demo 1

## THÔNG TIN NHÓM

**Thành viên 1:** Trần Quang Huy - 1512211

**Thành viên 2:** Phùng Tiến Hào - 1612174

**Thành viên 3:** Võ Quốc Huy – 1612269

Đây là phiên bản đầu nhóm sử dụng khi báo cáo ở lớp.

Ở phiên bản này, nhóm tham khảo [1], [2] và [3]. Nhóm đã cải tiến và bổ sung thêm các phần sau:

- Bổ sung thêm bộ phân lớp SVM để phân lớp các vector 128 chiều ở khâu nhận dạng.
- Nhóm bổ sung thêm việc tìm facial landmark và cân chỉnh ảnh để giúp ảnh mặt người nằm giữa khung hình và khuôn mặt không bị nghiêng. Bước này giúp tăng độ chính xác trong khâu nhận dạng.
- Cuối cùng, nhóm dùng thêm source code của Image Super-Resolution (ISR) bởi GAN [10] (**Generative Adversarial Network**) để giúp tăng cường chất lượng ảnh và tăng độ nét của ảnh. Để giúp khâu nhận diện đối với các ảnh kém chất lượng đạt kết quả tốt hơn.

**Lưu ý:** Nhóm không tích hợp ISR vào hệ thống vì ISR thời gian xử lý khá lâu. Do đó, nhóm chỉ sử dụng riêng lẻ đối với các ảnh có độ phân giải thấp.

### Các thư viện cần cài đặt:

- OpenCV
- Dlib
- imutils: chứa các hàm để xử lý ảnh và các tiện ích như video stream, đọc thư mục,... bởi Adrian.

**Đường dẫn để download:** [Demo1.rar](#)

## I. Giới thiệu về các tập tin và thư mục sử dụng:

Thư mục face\_detection\_model: chứa 2 files config và model của SSD dựa trên kiến trúc ResNet dùng để phát hiện khuôn mặt.

Thư mục output:

- Embedding.pickle: Lưu trữ các embedding image hay còn gọi là vector đặc trưng 128 chiều của dữ liệu huấn luyện.
- Le.pickle: Chứa nhãn dán tương ứng với các embedding image.

- Recognizer.pickle: Chứa model được train bởi SVM dùng để phân lớp ảnh đầu vào vào định danh tương ứng.

Tập tin extract\_embeddings.py: Dùng để mã hóa các khuôn mặt của một người thành vector đặc trưng 128 chiều.

Tập tin face\_detection.py: Dùng để phát hiện khuôn mặt trên ảnh tĩnh 2D.

Tập tin face\_detection\_video.py: Dùng để phát hiện mặt người bằng video-stream.

Tập tin facial\_landmark.py: Dùng để tìm facial landmark của mỗi khuôn mặt và cân chỉnh ảnh.

Tập tin openface\_nn4.small2.v1.t7: Đây là model của OpenFace được huấn luyện sẵn. Dùng để tạo các embedding image.

Tập tin recognize.py: Dùng để nhận diện mặt người qua ảnh tĩnh 2D.

Tập tin recognize\_face\_video.py: Dùng để nhận diện mặt người qua video-stream.

Tập tin shape\_predictor\_68\_face\_landmarks.dat: Chứa model được train sẵn, dùng để tìm facial landmark trên khuôn mặt.






Tập tin train\_model.py: Dùng để huấn luyện tập dữ liệu bằng SVM và sau đó đem model đi dự đoán ảnh khuôn mặt.

Thư mục ISR: chứa hệ thống tăng cường chất lượng ảnh bằng GAN.

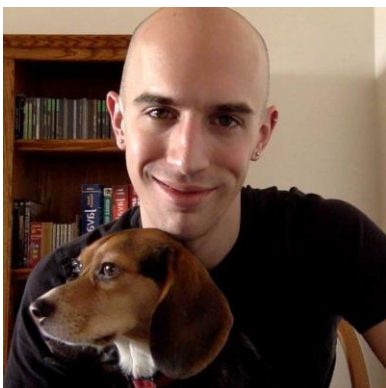
- image\_super\_resolution.py: Dùng để chạy và tăng cường chất lượng của ảnh yêu cầu.
- rdn-C6-D20-G64-G064-x2\_ArtefactCancelling\_epoch219.hdf5: Đây là model đã được huấn luyện sẵn và nhóm tận dụng để tăng cường chất lượng ảnh.

## II. Giới thiệu về tập dữ liệu huấn luyện và tập kiểm tra:

Dữ liệu huấn luyện thì gồm 5 nhân vật: (Ở thư mục dataset)

 adrian	6/9/2019 6:32 PM	File folder
 justin_bieber	6/9/2019 6:32 PM	File folder
 obama	6/9/2019 6:32 PM	File folder
 taylor	6/9/2019 6:32 PM	File folder
 unknown	6/9/2019 6:32 PM	File folder

- Adrian: 6 ảnh



- Taylor: 10  nh



- Justin Bieber: 10  nh



- Obama: 10  nh



- Unknown: 6 ảnh



Thư mục test\_images chứa các ảnh test.



Các ảnh có thêm chữ “hi” tức là ảnh đã qua tăng cường độ phân giải bằng mô hình GAN.

### III. Một số kết quả thực nghiệm:

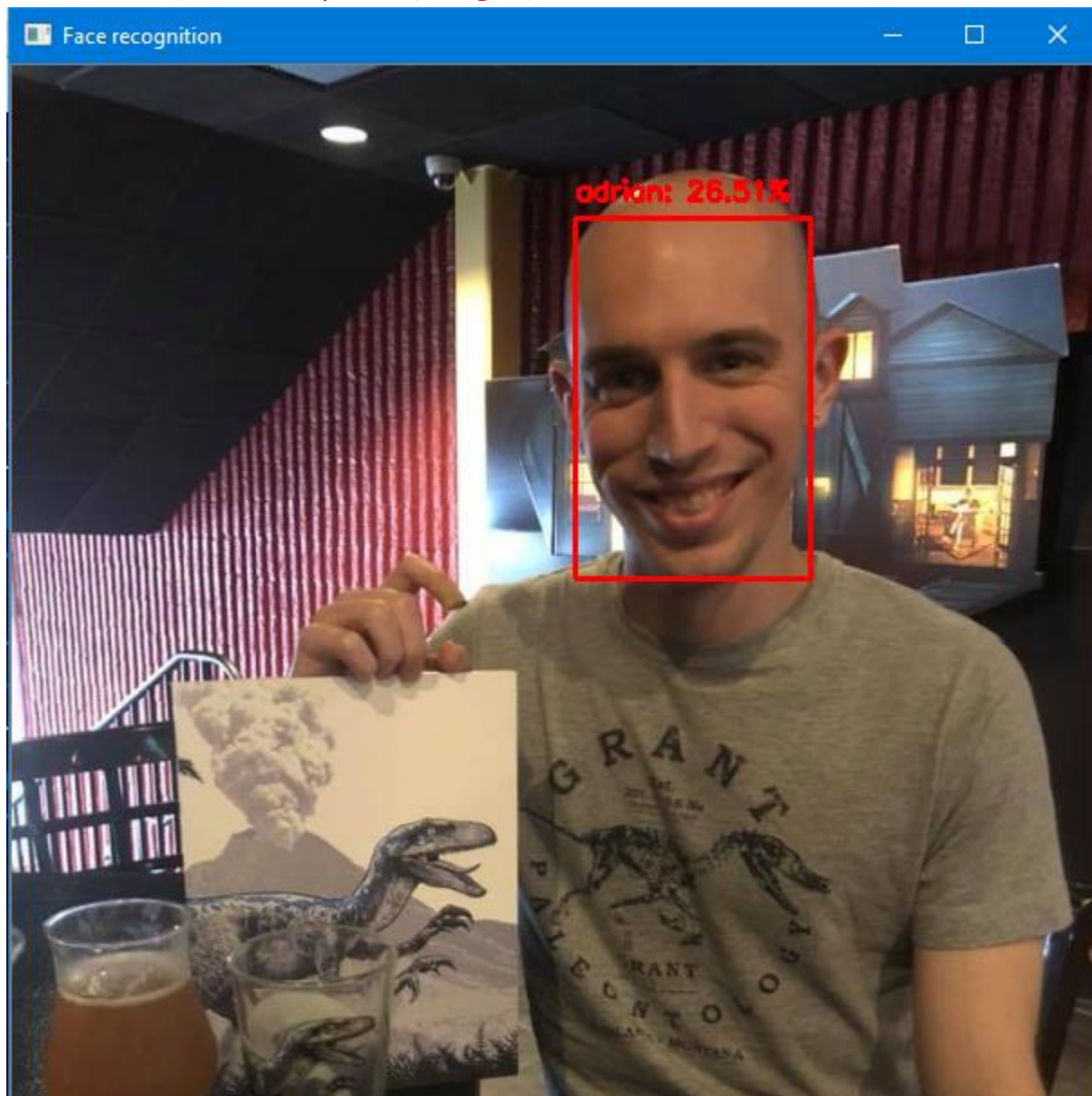


Figure 1: Nhận diện chính xác nhưng độ cậy của nhận dạng không cao chỉ có 26.51%



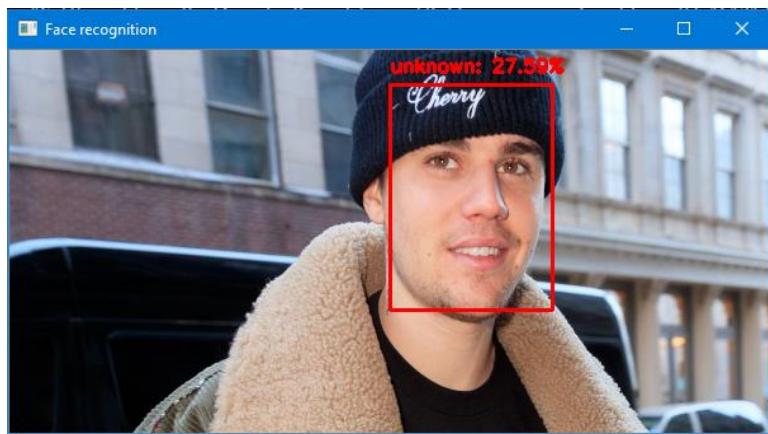


Figure 2: Nhận diện không chính xác

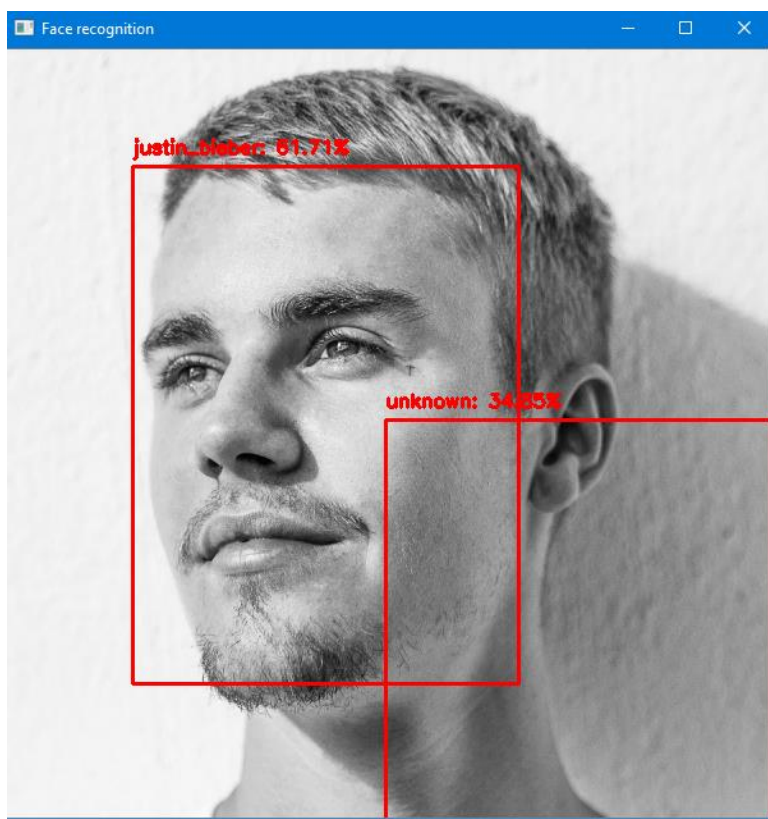


Figure 3: Trường hợp này ảnh nhận diện đúng nhưng có thừa thêm bounding box. Độ tin cậy của nhận dạng khá cao: cỡ 62%



Figure 4: Obama bị nhận diện nhầm thành Adrian. Có thể thấy ảnh khá nhòe có lẽ vì thế mà nhận dạng sai.



Figure 5: Đây là ảnh Obama sau khi tăng cường độ phân giải bằng model GAN. Kết quả nhận diện chính xác với độ tin cậy 42.25%

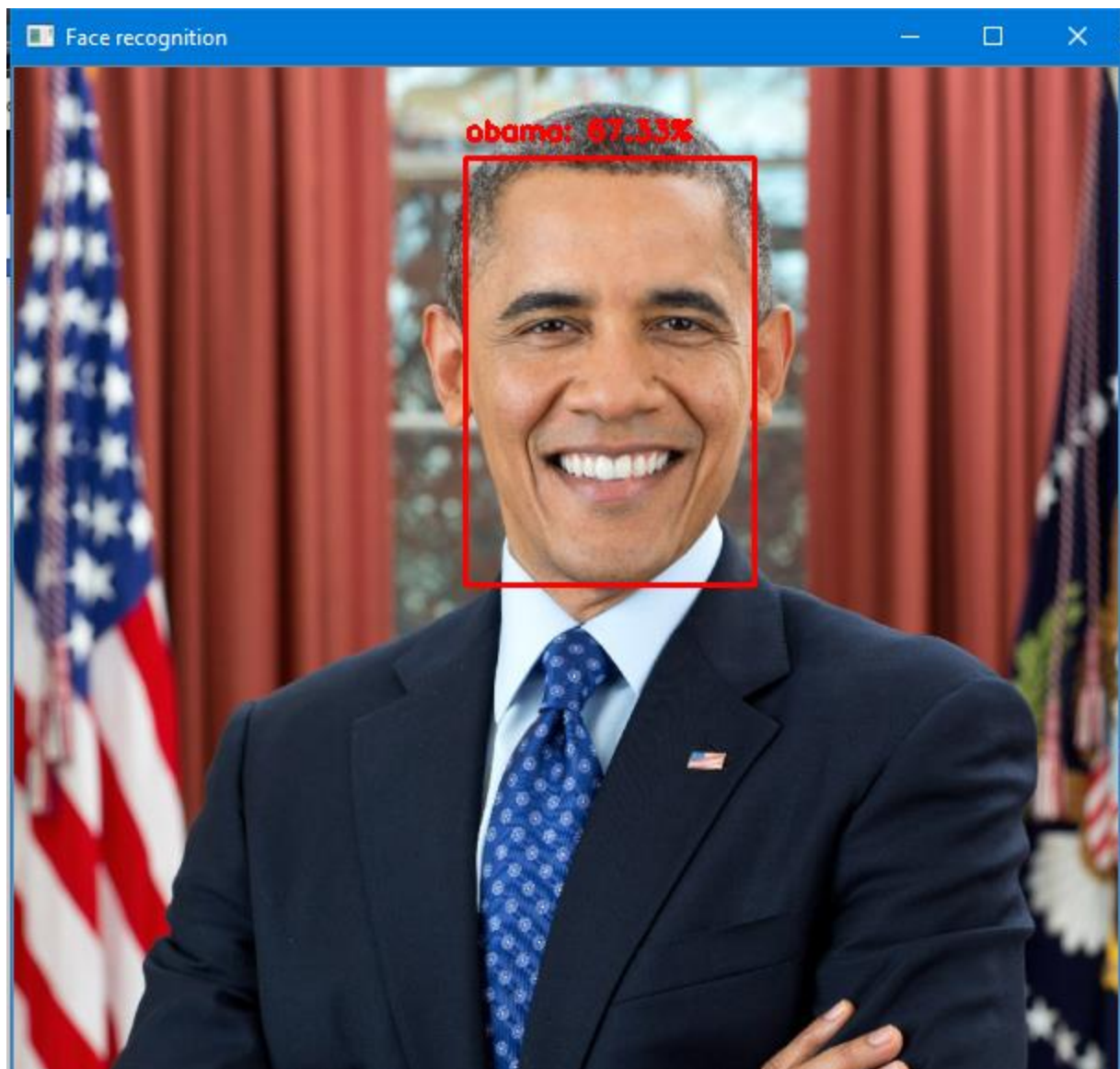


Figure 6: Độ chính xác cao và đạt mức tin cậy 67%



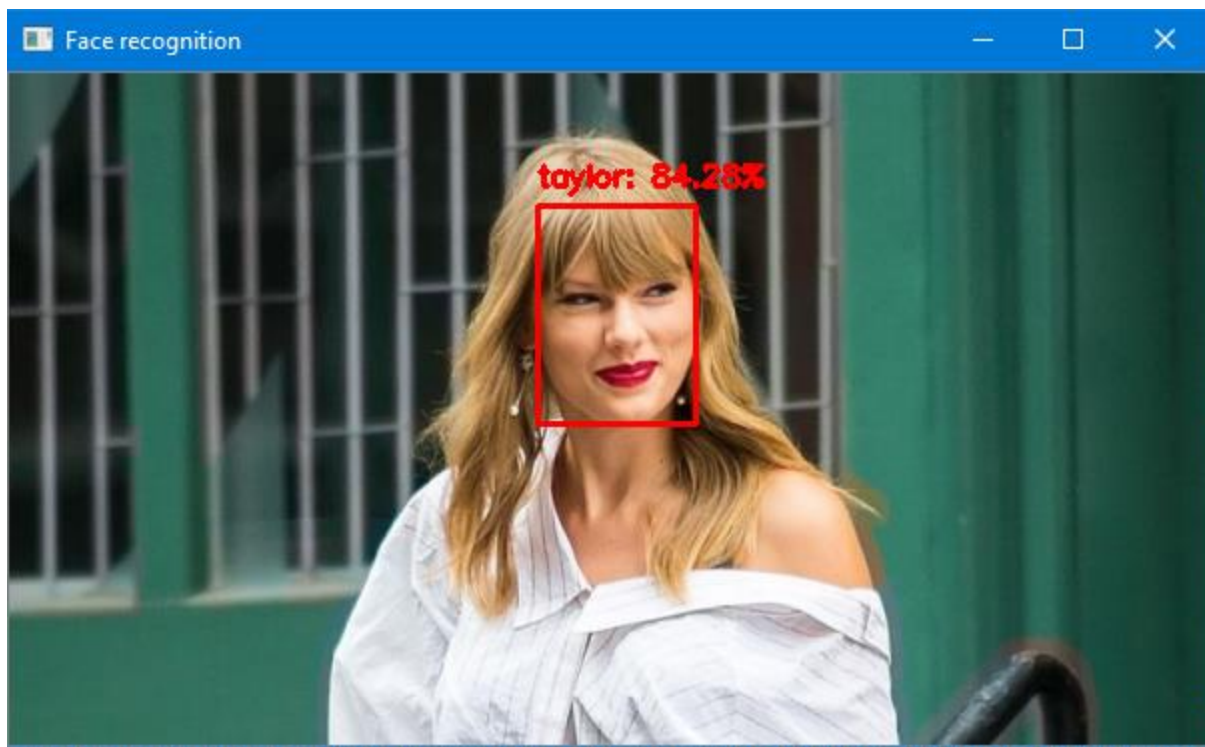


Figure 7: Ảnh tượng với độ tin cậy nhận diện tới 84.28%

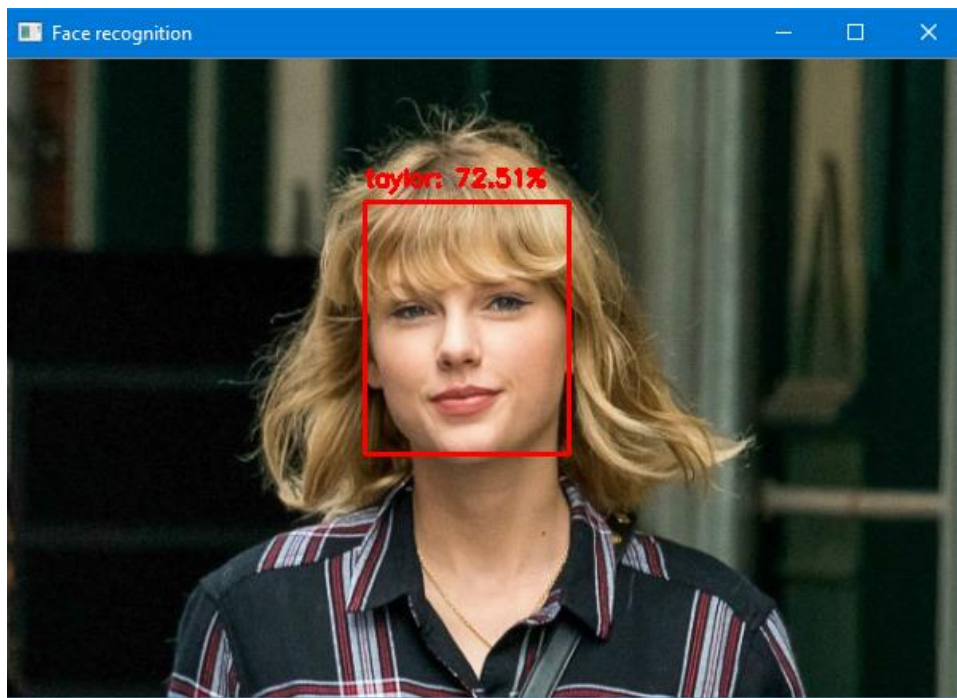


Figure 8: Lần này vẫn rất cao



Figure 9: Nhận dạng chính xác. Đây là ảnh của nhân vật không có trong huấn luyện do đó gán nhãn Unknown

#### IV. Kết quả thực nghiệm:

Độ chính xác trong phát hiện khuôn mặt là **98-99%**.

Độ chính xác trong nhận diện khuôn mặt là **80%**.

##### Nhận xét:

- Do số lượng tập training set khá ít do đó độ chính xác chưa cao trong khâu nhận dạng.
- Độ chính xác của phát hiện rất cao, phát hiện được cả mặt người bị khuất hay độ sáng thấp. Đồng thời bắt được khuôn mặt ở nhiều hướng khác nhau.
- Thời gian xử lý nhanh phù hợp cho các ứng dụng real-time.
- Phát hiện mặt ở nhiều scale khác nhau.
- Nếu khuôn mặt có scale bé hoặc tối quá, bị che khuất nhiều thì khả năng sẽ bị nhận dạng sai.

## V. Hướng cải tiến:

- Huấn luyện trên tập dữ liệu lớn hơn.
- Sử dụng nhiều frame để tăng cường chất lượng ảnh.
- Cải tiến ở việc tìm facial landmark và cân chỉnh ảnh để giúp nâng tốc độ xử lý nhanh hơn.

## VI. Hướng dẫn sử dụng chương trình:

### 1. Nhận diện ảnh tĩnh: `recognize.py`

```
python recognize.py
```

```
-i <Path to image>
```

```
-d "path to OpenCV's deep learning face detector"
```

```
-m "path to OpenCV's deep learning face embedding model"
```

```
-r "path to model trained to recognize faces"
```

```
-l "path to label encoder"
```

```
-c "minimum probability to filter weak detections"
```

Ví dụ:

```
python recognize.py -i E:\K16\Junior\Face-  
recognition\1612174\test_images\obama_1_hi.jpg -d E:\K16\Junior\Face-  
recognition\1612174\face_detection_model -m E:\K16\Junior\Face-  
recognition\1612174\openface_nn4.small2.v1.t7 -r E:\K16\Junior\Face-  
recognition\1612174\output\recognizer.pickle -l E:\K16\Junior\Face-  
recognition\1612174\output\le.pickle -c 0.5
```

Để nhận sự trợ giúp thì gõ lệnh: `python recognize.py -h`

```
(opencv-dlib-fr) C:\Users\Tien Hao\Google Drive\Junior\Pattern recognition\Team H3-1612174_1612269_1512211-Seminar\Demo1>python recognize.py -h
usage: recognize.py [-h] -i IMAGE -d DETECTOR -m EMBEDDING_MODEL -r RECOGNIZER
                  -l LE [-c CONFIDENCE]

optional arguments:
  -h, --help            show this help message and exit
  -i IMAGE, --image IMAGE
                        path to input image
  -d DETECTOR, --detector DETECTOR
                        path to OpenCV's deep learning face detector
  -m EMBEDDING_MODEL, --embedding-model EMBEDDING_MODEL
                        path to OpenCV's deep learning face embedding model
  -r RECOGNIZER, --recognizer RECOGNIZER
                        path to model trained to recognize faces
  -l LE, --le LE        path to label encoder
  -c CONFIDENCE, --confidence CONFIDENCE
                        minimum probability to filter weak detections
```

Remove items in "Goog  
Drive?  
You moved "iron\_chic.inn" ar

## 2. Nhận diện video-stream: `recognize_face_video.py`

`python recognize_face_video.py`

- d "path to OpenCV's deep learning face detector"
- m "path to OpenCV's deep learning face embedding model"
- r "path to model trained to recognize faces"
- l "path to label encoder"
- c "minimum probability to filter weak detections"

Để được trợ giúp, nhập lệnh: `python recognize_face_video.py -h`



```
(opencv-dlib-fr) C:\Users\Tien Hao\Google Drive\Junior\Pattern recognition\Team H3-1612174_1612269_1512211-Seminar\Demo1>python recognize_face_video.py -h
usage: recognize_face_video.py [-h] -d DETECTOR -m EMBEDDING_MODEL -r
                               RECOGNIZER -l LE [-c CONFIDENCE]

optional arguments:
  -h, --help            show this help message and exit
  -d DETECTOR, --detector DETECTOR
                        path to OpenCV's deep learning face detector
  -m EMBEDDING_MODEL, --embedding-model EMBEDDING_MODEL
                        path to OpenCV's deep learning face embedding model
  -r RECOGNIZER, --recognizer RECOGNIZER
                        path to model trained to recognize faces
  -l LE, --le LE        path to label encoder
  -c CONFIDENCE, --confidence CONFIDENCE
                        minimum probability to filter weak detections
```

## V. Tham khảo

- [1] Blog: Face detection. Adrian. Source: <https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/>
- [2] Blog: Face recognition using dlib. Adrian. Source: <https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/>
- [3] Blog: OpenCV face recognition. Adrian. Source: <https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/>
- [4] Dlib library by Davis King. Source: <http://dlib.net/>
- [5] Blog: Facial landmark. Adrian. Source: <https://www.learnopencv.com/facemark-facial-landmark-detection-using-opencv/>
- [6] Face alignment with OpenCV. Source: <https://www.pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/>
- [7] OpenFace library. Source: <https://cmusatyalab.github.io/openface/>
- [8] Face recognition by Adam Geitgey. Source: [https://github.com/ageitgey/face\\_recognition](https://github.com/ageitgey/face_recognition)
- [9] Blog: Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning. Adam Geitgey. Source: <https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78>
- [10] Image Super-Resolution (ISR). Source: <https://github.com/idealo/image-super-resolution>