

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN MÔN:  
TÍNH TOÁN ĐA PHƯƠNG TIỆN

ĐỀ TÀI  
CHƯƠNG TRÌNH NHẬN DẠNG MESSI và RONALDO



Lớp : **CS232.J11**, Tính toán đa phương tiện  
Khoa : Khoa học máy tính  
GVLT : **Mai Tiến Dũng**  
GVTH : **Mai Tiến Dũng**  
Thành viên : Võ Huy Nam , mssv: 15520528  
Phùng Tấn Lợi, mssv: 15520437

TP.Hồ Chí Minh, tháng 12, năm 2018

## LỜI CẢM ƠN

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến toàn thể quý thầy cô Trường Đại học Công nghệ Thông tin- Đại học Quốc gia TP.HCM và quý thầy cô khoa Khoa Học Máy Tính đã tận tình truyền đạt những kiến thức cơ bản làm nền tảng cho chúng em thực hiện đề tài này.

Đặc biệt nhóm em xin gửi lời cảm ơn tới thầy Mai Tiến Dũng (giảng viên lý thuyết và thực hành môn tính toán đa phương tiện ). Quý thầy cô đã trực tiếp hướng dẫn tận tình, sửa chữa và góp ý giúp nhóm hoàn thành tốt đồ án của mình.

Trong thời gian một học kỳ thực hiện đề tài, nhóm đã vận dụng những kiến thức tiếp thu được kết hợp với việc học hỏi và nghiên cứu những kiến thức mới để hoàn thiện đồ án một cách tốt nhất. Chính vì vậy nhóm em rất mong nhận được những sự góp ý chân thành từ phía thầy cô nhằm cải thiện đồ án cũng như kiến thức của nhóm trong tương lai.

Xin chân thành cảm ơn quý Thầy Cô !

## This image shows a full page of white paper designed for handwriting practice. It features approximately 20 evenly spaced horizontal dotted lines running from left to right across the entire width of the page. There are no margins, text, or other markings present.

## Mục lục

I Tổng quan về đề .....	6
II Giới thiệu về mạng tích chập (CNN) .....	7
III Support Vector Machine .....	10
IV Các bước nhận dạng Ronaldo && Messi.....	13
V Chi tiết các bước thực hiện .....	14

## BẢNG PHÂN CÔNG CÔNG VIỆC

Họ và tên	Mã số sinh viên	Công việc chính	Công việc cụ thể
Võ Huy Nam	15520528	- Code chính	- Xây dựng bộ phân lớp - Xây dựng chương trình để nhận dạng từng khuôn mặt trong một bức ảnh - Viết báo cáo
Phùng Tấn Lợi	15520437	- Thu thập dữ liệu - Xây dựng dataset	- Thu thập dữ liệu - Xây dựng dataset - Viết báo cáo

## ĐÁNH GIÁ NỘI BỘ

- Các thành viên hoạt động rất tích cực, năng nổ. Tất cả đều hoàn hành công việc được giao đúng thời hạn
- Tận dụng đúng thế mạnh của từng thanh viên để phân chia công việc hợp lý
- Đồ án hoàn thành đúng như dự kiến

## BẢNG ĐÁNH GIÁ PHẦN TRĂM ĐÓNG GÓP

Họ và tên	Mã số sinh viên	Phần trăm đóng góp
Võ Huy Nam	15520528	100%
Phùng Tấn Lợi	15520437	80%

## I. Tổng quan về đề tài

### 1. Phát biểu bài toán

- Có 1 ảnh bất kỳ, làm thế nào để nhận diện đâu là Ronaldo, đâu là Messi ở trong bức ảnh. Nếu không phải 2 người này thì là Unknown
- Input: Bức ảnh
- Output: Bức ảnh và khuôn mặt + tên của từng người trên bức ảnh

### 2. Ý tưởng

- Đưa bài toán về bài toán phân lớp một bức ảnh dựa trên 3 lớp đã biết

### 3. Các Module được sử dụng

- Thư viện Opencv
- Mạng tích chập VGG-FACE
- Bộ phân lớp SVC (Linear Support Vector Classification)
- Ngôn ngữ: Python

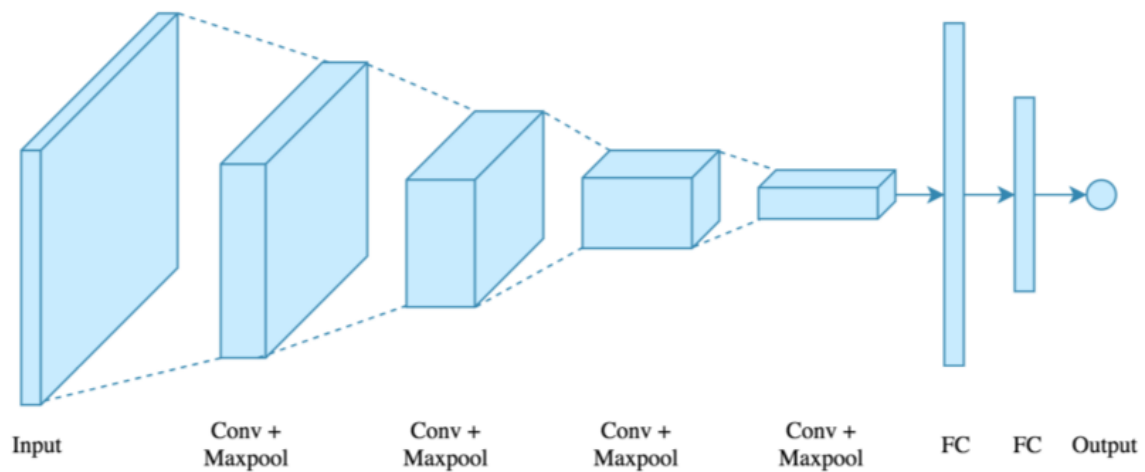
## II. Giới thiệu về mạng tích chập (CNN)

### 1. Khái quát

- Với sự phát triển phần cứng mạnh mẽ cho phép tính toán song song hàng tỉ phép tính, tạo tiền đề cho Mạng tích chập trở nên phổ biến và đóng vai trò quan trọng trong sự phát triển của trí tuệ nhân tạo nói chung và xử lý ảnh nói riêng. Một trong các ứng dụng quan trọng của mạng nơ-ron tích chập đó là cho phép các máy tính có khả năng “nhìn” và “phân tích”, nói 1 cách dễ hiểu, Convnets được sử dụng để nhận dạng hình ảnh bằng cách đưa nó qua nhiều layer với một bộ lọc tích chập để sau cùng có được một điểm số nhận dạng đối tượng. Mạng tích chập được lấy cảm hứng từ vỏ não thị giác.
- Mỗi khi chúng ta nhìn thấy một cái gì đó, một loạt các lớp tế bào thần kinh được kích hoạt, và mỗi lớp thần kinh sẽ phát hiện một tập hợp các đặc trưng như đường thẳng, cạnh, màu sắc, v.v.v của đối tượng. lớp thần kinh càng cao sẽ phát hiện các đặc trưng phức tạp hơn để nhận ra những gì chúng ta đã thấy.

### 2. Ứng dụng

- Mạng tích chập được ứng dụng nhiều trong Computer Vision, Recommender System, Natural Language Processing, ... Ví dụ như tự động nhận diện khi chúng ta up một ảnh lên Facebook, hay khi search một từ bất kì lên google search, ví dụ "Mèo" thì trong tab "Hình ảnh", google sẽ hiển thị rất nhiều ảnh có mèo trong đó



### 3. Mô hình mạng tích chập

- Có ảnh input đầu vào. Qua hàng loạt các Convolutional Layer cùng Max Pool Layer (thường pooling sẽ theo ngay sau 1 convolutional layer), cuối cùng là 2 fully connected.

#### 3.1. Convolution

- convolution gồm 2 khái niệm khác là Convolution Filter và Convolutional Layer. Trong mạng neural network thông thường, từ input, ta cho qua các hidden layer rồi ra được output. Với CNN, Convolutional Layer cũng chính là hidden layer, khác ở chỗ, Convolutional Layer là một tập các feature map và mỗi feature map này là một bản scan của input ban đầu, nhưng được trích xuất ra các feature/đặc tính cụ thể. Scan như thế nào thì lại dựa vào Convolution Filter hay kernel. Đây là một ma trận sẽ quét qua ma trận dữ liệu đầu vào, từ trái qua phải, trên xuống dưới, và nhân tương ứng từng giá trị của ma trận đầu vào mà ma trận kernel rồi cộng tổng lại, đưa qua activation function (sigmoid, relu, elu, ...), kết quả sẽ là một con số cụ thể, tập hợp các con số này lại là 1 ma trận nữa, chính là feature map.

#### 3.2. Pooling



- Mục đích của pooling rất đơn giản, nó làm giảm số hyperparameter mà ta cần phải tính toán, từ đó giảm thời gian tính toán, tránh overfitting. Loại pooling ta thường gặp nhất là *max pooling*, lấy giá trị lớn nhất trong một *pooling window*. Pooling hoạt động gần giống với convolution, nó cũng có 1 cửa sổ trượt gọi là *pooling window*, cửa sổ này trượt qua từng giá trị của ma trận dữ liệu đầu vào (thường là các feature map trong convolutional layer), chọn ra một giá trị từ các giá trị nằm trong cửa sổ trượt (với max pooling ta sẽ lấy giá trị lớn nhất)

### 3.3. Fully Connected

- Thường thì sau các lớp Conv+Pooling thì sẽ là 2 lớp Fully connected, 1 layer để tập hợp các feature layer mà ta đã tìm ra, chuyển đổi dữ liệu từ 3D, hoặc 2D thành 1D, tức chỉ còn là 1 vector. Còn 1 layer nữa là output, số neuron của layer này phụ thuộc vào số output mà ta muốn tìm ra. Giả sử với tập dữ liệu MNIST chẳng hạn, ta có tập các số viết tay từ 0 -> 9. Vậy output sẽ có số neuron là 10.

## 4. Mô hình VGG Face

- VGG Face được đào tạo để phân loại hình ảnh chuyên về nhận dạng khuôn mặt người. Nó có 22 lớp và 37 đơn vị sâu.



layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
type	input	conv	relu	conv	relu	mpool	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv
name	-	conv1_1	relu1_1	conv1_2	relu1_2	pool1	conv2_1	relu2_1	conv2_2	relu2_2	pool2	conv3_1	relu3_1	conv3_2	relu3_2	conv3_3	relu3_3	pool3	conv4_1
support	-	3	1	3	1	2	3	1	3	1	2	3	1	3	1	3	1	2	3
filt dim	-	3	-	64	-	-	64	-	128	-	-	128	-	256	-	256	-	-	256
num filts	-	64	-	64	-	-	128	-	128	-	-	256	-	256	-	256	-	-	512
stride	-	1	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	2	1
pad	-	1	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1

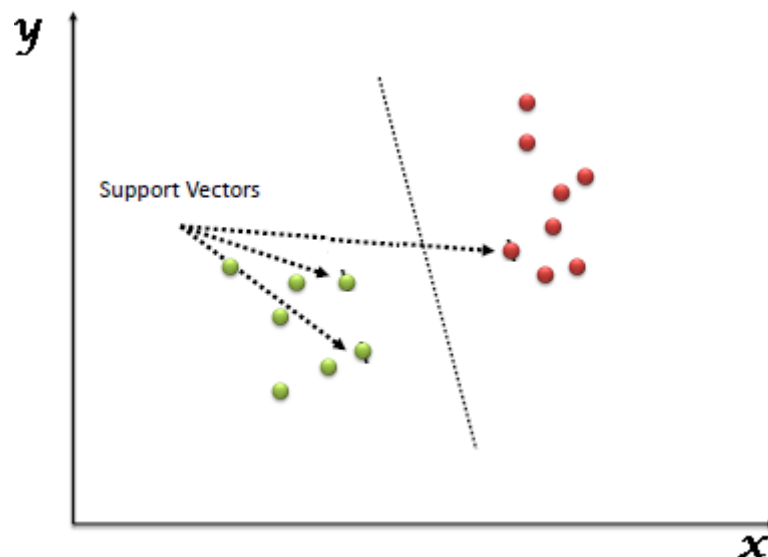
  

layer	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
type	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	softmax
name	relu4_1	conv4_2	relu4_2	conv4_3	relu4_3	pool4	conv5_1	relu5_1	conv5_2	relu5_2	conv5_3	relu5_3	pool5	fc6	relu6	fc7	relu7	fc8	prob
support	1	3	1	3	1	2	3	1	3	1	3	1	2	7	1	1	1	1	1
filt dim	-	512	-	512	-	-	512	-	512	-	512	-	-	512	-	4096	-	4096	-
num filts	-	512	-	512	-	-	512	-	512	-	512	-	-	4096	-	4096	-	2622	-
stride	1	1	1	1	1	2	1	1	1	1	1	1	2	1	1	1	1	1	1
pad	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

### III. Support Vector Machine

#### 1. Giới thiệu

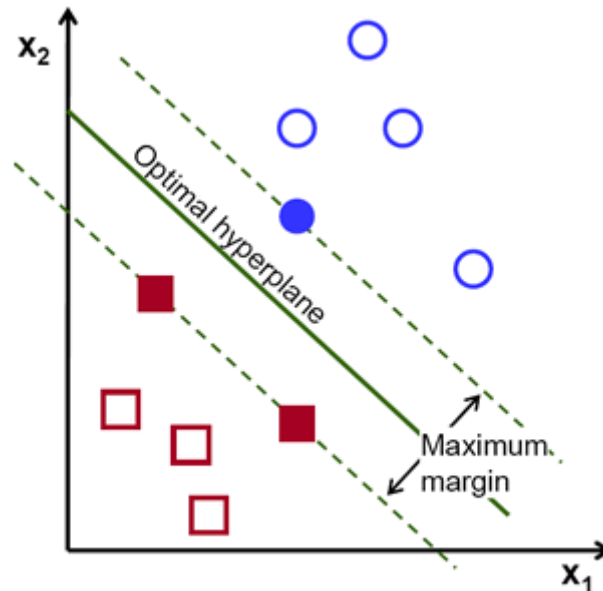
- SVM là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại. Trong thuật toán này, chúng ta vẽ đồ thị dữ liệu là các điểm trong  $n$  chiều (ở đây  $n$  là số lượng các tính năng bạn có) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó chúng ta thực hiện tìm "đường bay" phân chia các lớp. Đường bay - nó chỉ hiệu đơn giản là 1 đường thẳng có thể phân chia các lớp ra thành hai phần riêng biệt.



- Support Vectors hiểu một cách đơn giản là các đối tượng trên đồ thị tọa độ quan sát, Support Vector Machine là một biên giới để chia hai lớp tốt nhất.
- #### 2. SVM thực hiện điều này như thế nào?
- Bản chất của phương pháp SVM là chuyển không gian dữ liệu ban đầu thành một không gian mới hữu hạn chiều mà ở đó cho khả năng phân lớp dễ dàng hơn. Một quả táo nằm trên mặt bàn sẽ được gắn với một tọa độ cụ thể. Ví dụ, quả táo nằm cách mép trái 2cm và cách mép dưới 5cm được thể hiện trên trục tọa độ  $(x, y)$  tương ứng là  $(2, 5)$ .  $x$  và  $y$  chính là tọa độ trong không gian hai chiều của quả táo. Khi đưa lên chiều thứ 3 là  $z(x, y)$ , ta có thể tính được tọa độ của  $z$  trong không gian 3 chiều dựa vào tọa độ  $x, y$  ban đầu. Điểm làm SVM hiệu quả hơn các phương pháp khác chính là việc sử dụng Kernel

Method giúp cho SVM không còn bị giới hạn bởi việc phân lớp một cách tuyến tính, hay nói cách khác các siêu phẳng có thể được hình thành từ các hàm phi tuyến.

### 3. Margin trong SVM là gì?



- Margin là khoảng cách giữa siêu phẳng đến 2 điểm dữ liệu gần nhất tương ứng với các phân lớp.
- Điều quan trọng ở đây đó là phương pháp SVM luôn cố gắng cực đại hóa margin này, từ đó thu được một siêu phẳng tạo khoảng cách xa nhất so với 2 đối tượng. Nhờ vậy, SVM có thể giảm thiểu việc phân lớp sai (misclassification) đối với điểm dữ liệu mới đưa vào.

### 4. Ưu và nhược điểm

#### 4.1. Ưu điểm

- Xử lý trên không gian số chiều cao: SVM là một công cụ tính toán hiệu quả trong không gian chiều cao, trong đó đặc biệt áp dụng cho các bài toán phân loại văn bản và phân tích quan điểm nơi chiều có thể cực kỳ lớn
- Tiết kiệm bộ nhớ: Do chỉ có một tập hợp con của các điểm được sử dụng trong quá trình huấn luyện và ra quyết định thực tế cho các điểm dữ liệu mới

nên chỉ có những điểm cần thiết mới được lưu trữ trong bộ nhớ khi ra quyết định

- Tính linh hoạt - phân lớp thường là phi tuyến tính. Khả năng áp dụng Kernel mới cho phép linh động giữa các phương pháp tuyến tính và phi tuyến tính từ đó khiến cho hiệu suất phân loại lớn hơn.

#### 4.2. Nhược điểm

- Bài toán số chiều cao: Trong trường hợp số lượng thuộc tính (**p**) của tập dữ liệu lớn hơn rất nhiều so với số lượng dữ liệu (**n**) thì SVM cho kết quả khá tồi
- Chưa thể hiện rõ tính xác suất: Việc phân lớp của SVM chỉ là việc cố gắng tách các đối tượng vào hai lớp được phân tách bởi siêu phẳng SVM. Điều này chưa giải thích được xác suất xuất hiện của một thành viên trong một nhóm là như thế nào. Tuy nhiên hiệu quả của việc phân lớp có thể được xác định dựa vào khái niệm margin từ điểm dữ liệu mới đến siêu phẳng phân lớp mà chúng ta đã bàn luận ở trên.

#### 5. Kết luận

- SVM là một phương pháp hiệu quả cho bài toán phân lớp dữ liệu. Nó là một công cụ đặc lực cho các bài toán về xử lý ảnh, phân loại văn bản, phân tích quan điểm. Một yếu tố làm nên hiệu quả của SVM đó là việc sử dụng Kernel function khiến cho các phương pháp chuyển không gian trở nên linh hoạt hơn.

#### IV. CÁC BƯỚC NHẬN DIỆN RONALDO VÀ MESSI

##### 1. Ý tưởng bài toán

- Đưa bài toán về bài toán phân lớp ảnh dựa trên ba lớp đã có
- Xây dựng 3 lớp ảnh đã biết Messi, Ronaldo, Unknown
- Lớp unknown để nhận diện cho các khuôn mặt không phải của Ronaldo và messi

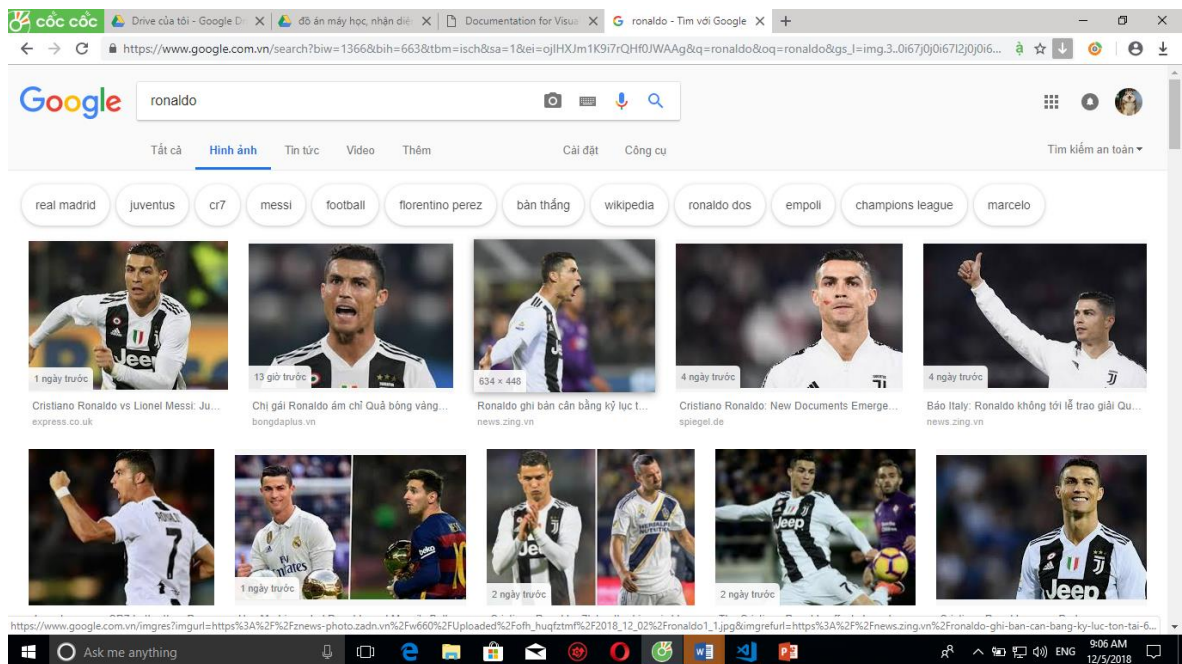
##### 2. Các bước thực hiện

- Bước 1: Thu thập ảnh cho 3 lớp Ronaldo, Messi, Unknown (ảnh của lớp Unknown sẽ gồm các ảnh của Neymar)
- Bước 2: Trích xuất ảnh khuôn mặt cho 3 lớp để xây dựng dataset
- Bước 3: Chia tập dataset thành 2 tập gồm tập test và tập train
- Bước 4: Trích xuất đặc trưng cho từng tập dataset
- Bước 5: Xây dựng bộ phân lớp từ đặc trưng của tập train
- Bước 6: Kiểm tra độ chính xác từ tập test

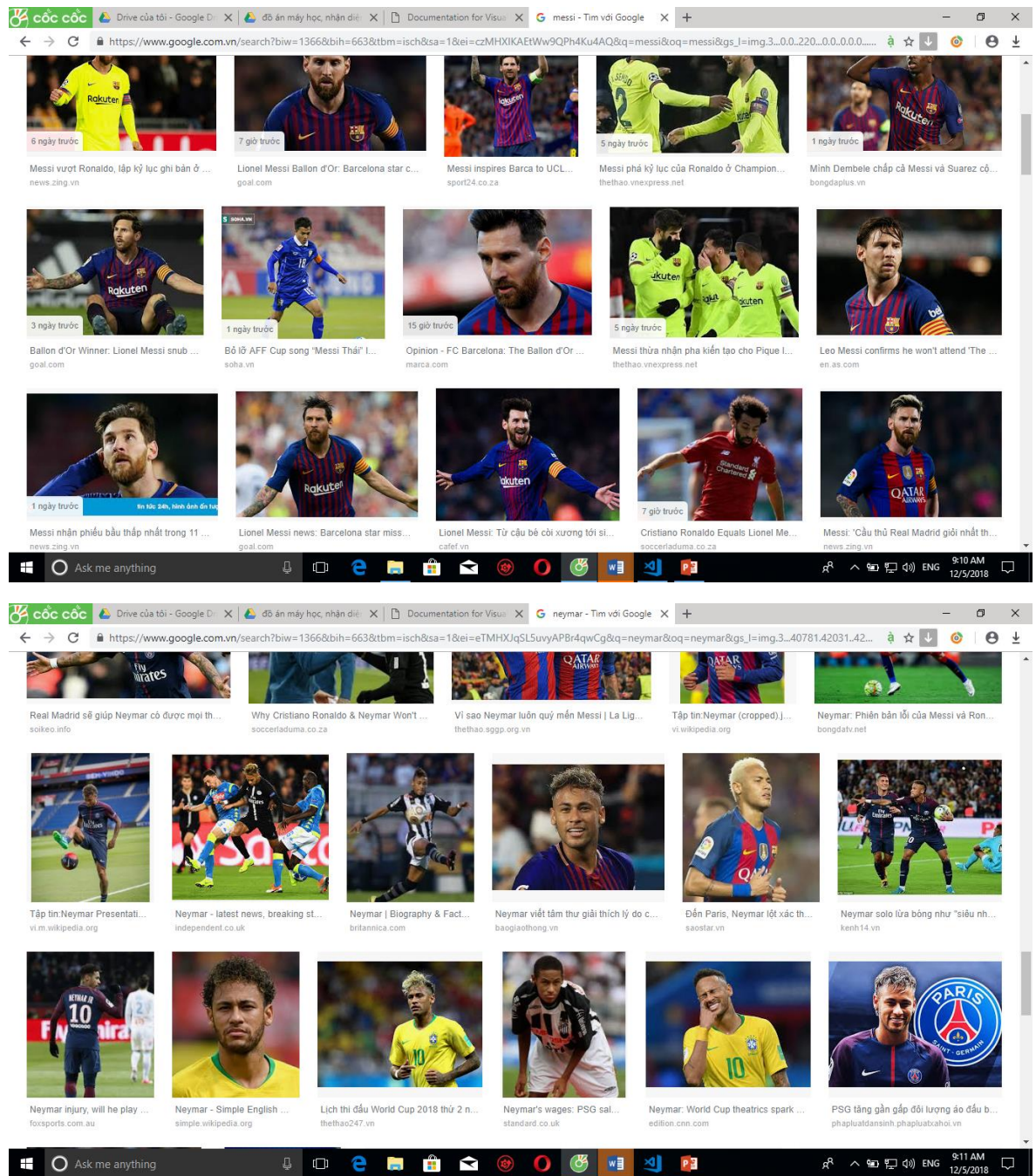
## V. CHI TIẾT CÁC BƯỚC THỰC HIỆN

### 1. Thu thập ảnh cho 3 lớp Ronaldo, Messi, Unknown

- Nhóm thu thập ảnh của 3 lớp trên từ google hình ảnh
- Thao tác để lấy là sử dụng tool zip có sẵn để lấy hình ảnh về







- Sau khi thu thập thì sẽ lưu vào folder imageSource
- Folder images Source sẽ gồm 2 thư mục Ronaldo, Messi và Unknown, mỗi thư mục sẽ chứa ảnh của cầu thủ tương ứng với mỗi thư mục

## 2. Trích xuất ảnh khuôn mặt cho 3 lớp để xây dựng dataset

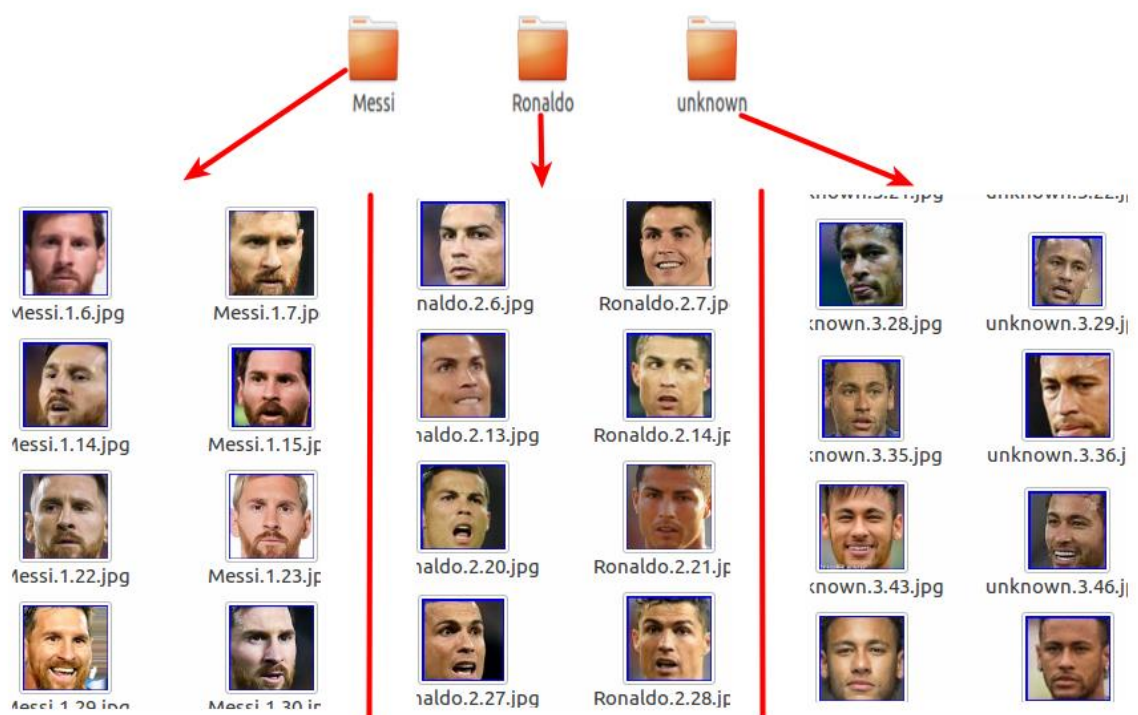
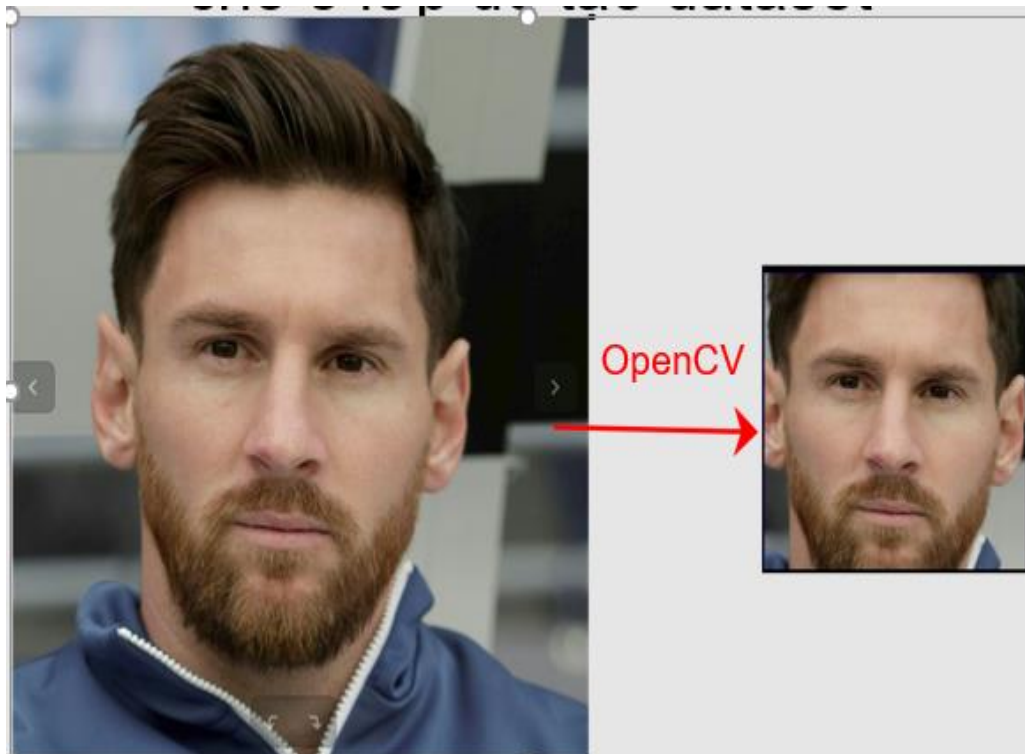


- Để trích xuất khuôn mặt cho từng lớp, nhóm sử dụng thư viện opencv để phát hiện khuôn mặt, sau đó lưu lại khuôn mặt được detect và lưu vào thư mục tương ứng
- Trong thư mục project file. File face\_recognition.py sẽ là file dùng để trích xuất khuôn mặt từng cá thể trong tập imageSource và lưu vào trong thư mục images

```

1  import cv2,os
2  import numpy as np
3  from PIL import Image
4  cam = cv2.VideoCapture(0)
5  detector = cv2.CascadeClassifier('/home/namvh/Documents/do an cuoi c
6
7  #Id = input('enter your id')
8  Id = '3'
9  sampleNum = 0
10 imgs = []
11 path = "/home/namvh/Documents/do an cuoi cung/file/imageSource/unkno
12 valid_images = [".jpg",".jpeg"]
13 for f in os.listdir(path):
14     imgs.append(Image.open(os.path.join(path,f)))
15 for img in imgs:
16     print(img.filename)
17     img1 = cv2.imread(img.filename)
18     gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
19
20     faces = detector.detectMultiScale(gray, 1.3, 5)
21     for (x, y, w, h) in faces:
22         cv2.rectangle(img1, (x, y), (x + w, y + h), (255, 0, 0), 2)
23
24         # incrementing sample number
25         sampleNum = sampleNum + 1
26         # saving the captured face in the dataset folder
27
28         cv2.imwrite("/home/namvh/Documents/do an cuoi cung/file/imag
29 cv2.destroyAllWindows()

```



- Sau khi rút trích xong ảnh khuôn mặt cho từng lớp, ta có được bộ dataset gồm 3 lớp lưu trong 3 folder Messi, Ronaldo, Unkown

- Tổng số ảnh trong tập dataset là 750 ảnh

### 3. Chia bộ dataset gồm 2 tập test và train

- Để thuận tiện cho việc chia dataset thành 2 tập test và train, nhóm dung một file test.txt để lưu những ảnh dùng cho việc test, một file train.txt dung cho training
- Tập test sẽ chiếm 30% dataset : gồm 226 ảnh
- Tập train sẽ chiếm 70% dataset: gồm 524 ảnh
- File generate\_db.py sẽ xử lý việc chia bộ dataset vào 2 file test.txt và train.txt trong thư mục dataset

```

1  import sys
2  import os
3  import random
4  import re
5
6  def generate_train(files, n):
7      random.shuffle(files)
8      return random.sample(files, n)
9
10 def generate_test(files, train_files):
11     return list(set(files) - set(train_files))
12
13 ...
14 def get_trailing_numbers(s):
15     m = re.search(r'\d+$', s)
16     return m.group()
17 ...
18
19 def write_file(path, files):
20     print("[+]Write ", path)
21     with open(path, "w") as f:
22         for file in files:
23             f.write(file)
24             f.write("\n")
25
26 def generate_data(src, db):
27     train_files = []
28     test_files = []
29     for folder in os.listdir(src):
30         print("[+]Access folder ", folder)
31         folder_path = os.path.join(src, folder)
32         files = [os.path.join(folder_path, file) for file in os.listdir(folder_path)]
33         n = len(files)

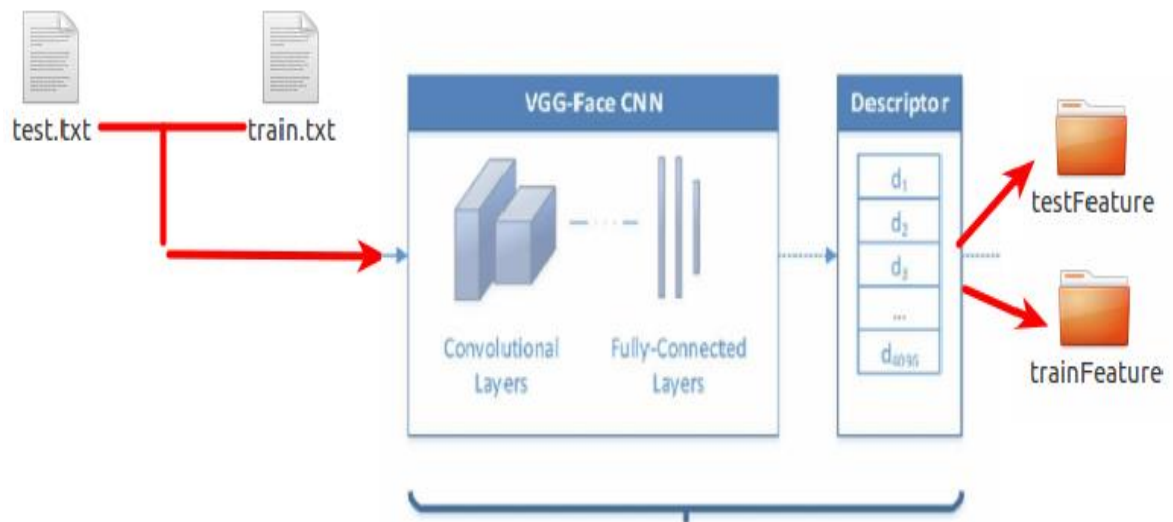
```

his PC > Downloads > do\_an\_cuoi\_cung(final) > file > dataSet

Name	Date modified	Type	Size
test	11/29/2018 6:01 PM	Text Document	18 KB
train	11/29/2018 6:01 PM	Text Document	42 KB

### 4. Trích xuất đặc trưng cho từng tập dataset

- Nhóm sử dụng VGG-face trong thư viện keras để trích xuất đặc trưng cho ảnh trong 2 tập test và train
- Sau khi quá trình kết thúc, đặc trưng của ảnh test sẽ được lưu vào thư mục testFeature , đặc trưng của ảnh train sẽ được lưu vào thư mục trainFeature

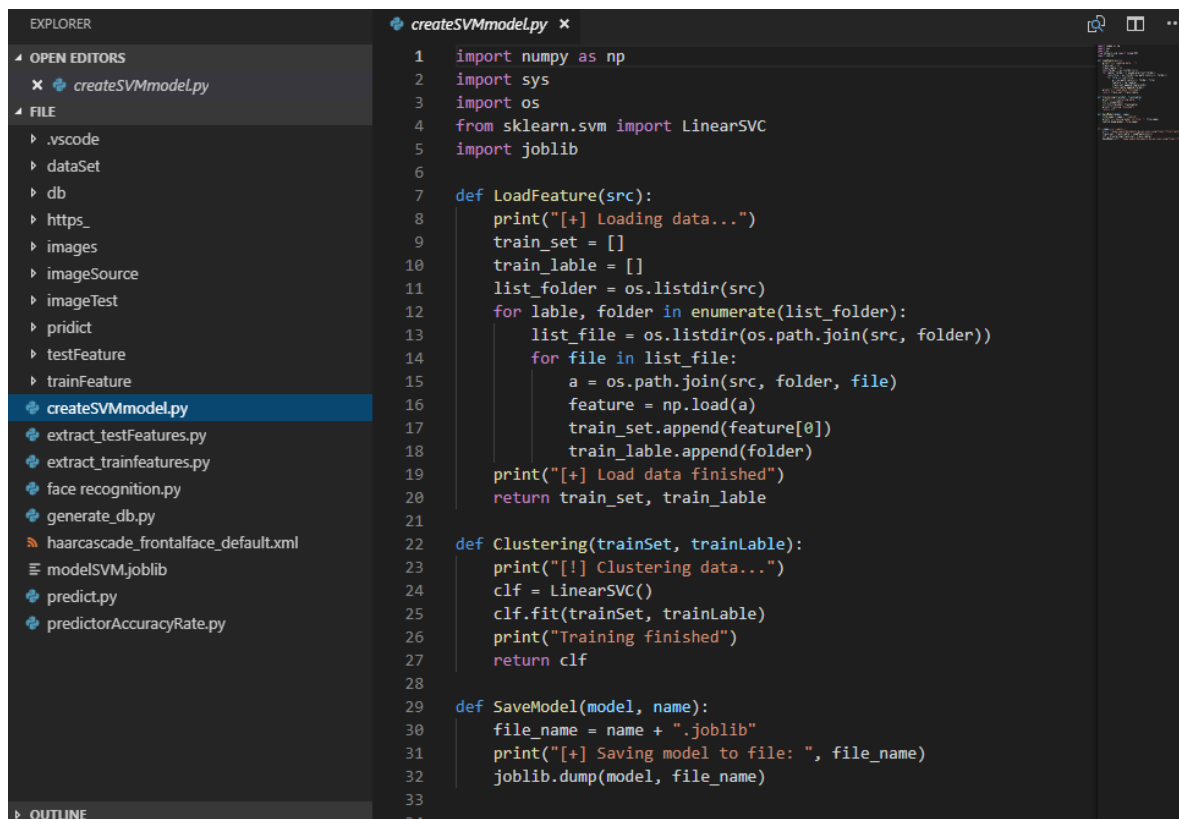


- Chạy file `extrac_testFeatures.py` và `extrac_trainFeatures.py` để trích xuất đặc trưng cho tập test và tập train

```
1 from keras.engine import Model
2 from keras.layers import Input
3 from keras_vggface.vggface import VGGFace
4 from keras_vggface import utils
5 from keras.preprocessing import image
6 import numpy as np
7 import sys
8 import os
9 from PIL import ImageFile
10
11 '''
12 ImageFile.LOAD_TRUNCATED_IMAGES=True
13 print("[+] Setup model")
14 layer_name = 'fc1' # edit this line
15 vgg_model = VGGFace()
16 out = vgg_model.get_layer(layer_name).output
17 model = Model(vgg_model.input, out)
18 '''
19
20 # tensorflow
21 model = VGGFace() # default : VGG16 , you can use model='resnet50' o
22
23 def save_feature(save_path, feature):
24     os.makedirs(os.path.dirname(save_path), exist_ok=True)
25
26     print("[+] Save extracted feature to file : ", save_path)
27     np.save(save_path, feature)
28
29 def extract_features(src):
30     with open(src, "r") as file:
31         for i, line in enumerate(file):
32             img_path = line[:-1]
33             print("[+] Read image : ", img_path, " id : ", i)
34             if os.path.isfile(img_path) and img_path.find(".jpg") != -1:
```

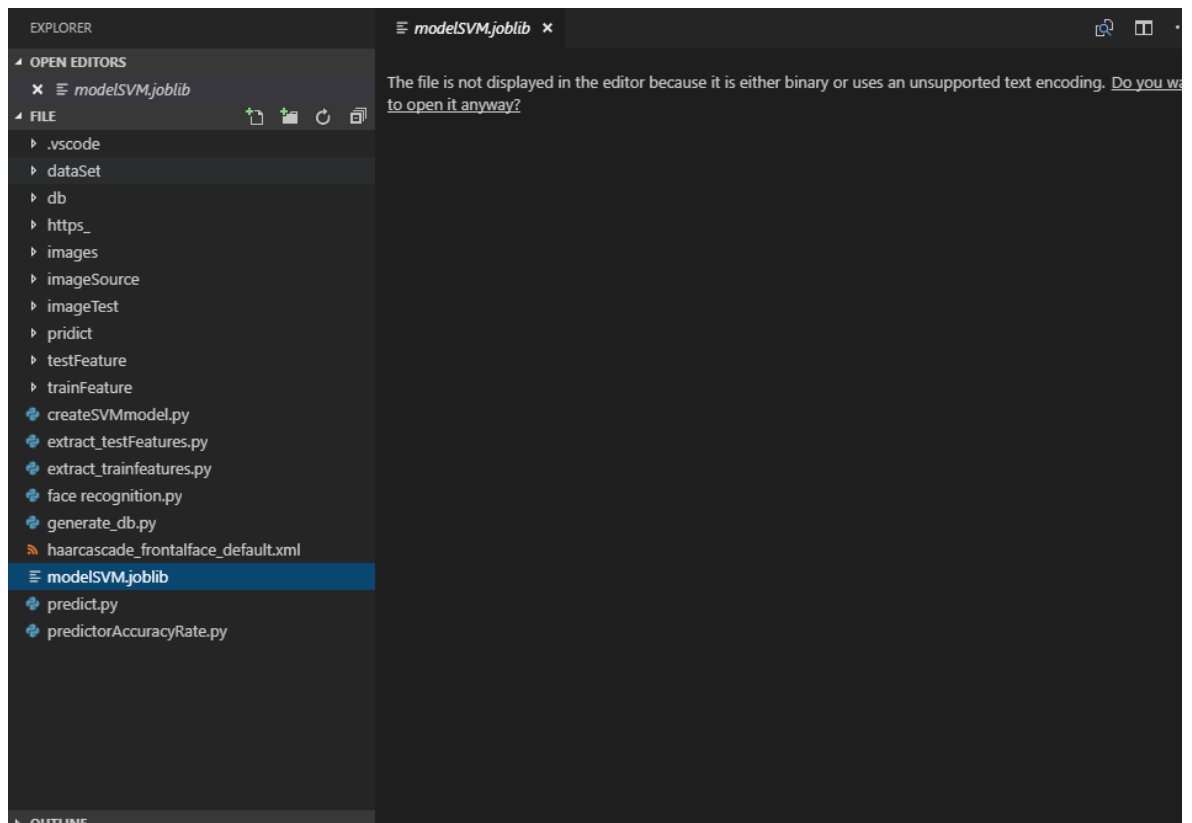
## 5. Xây dựng bộ phân lớp

- Nhóm sử dụng những đặc trưng được rút trích từ tập train mà đã có ở bước 4
- Sử dụng module LinearSVC của thư viện keras để xây dựng bộ phân lớp
- Chạy file createSVMmodel.py để xây dựng bộ phân lớp



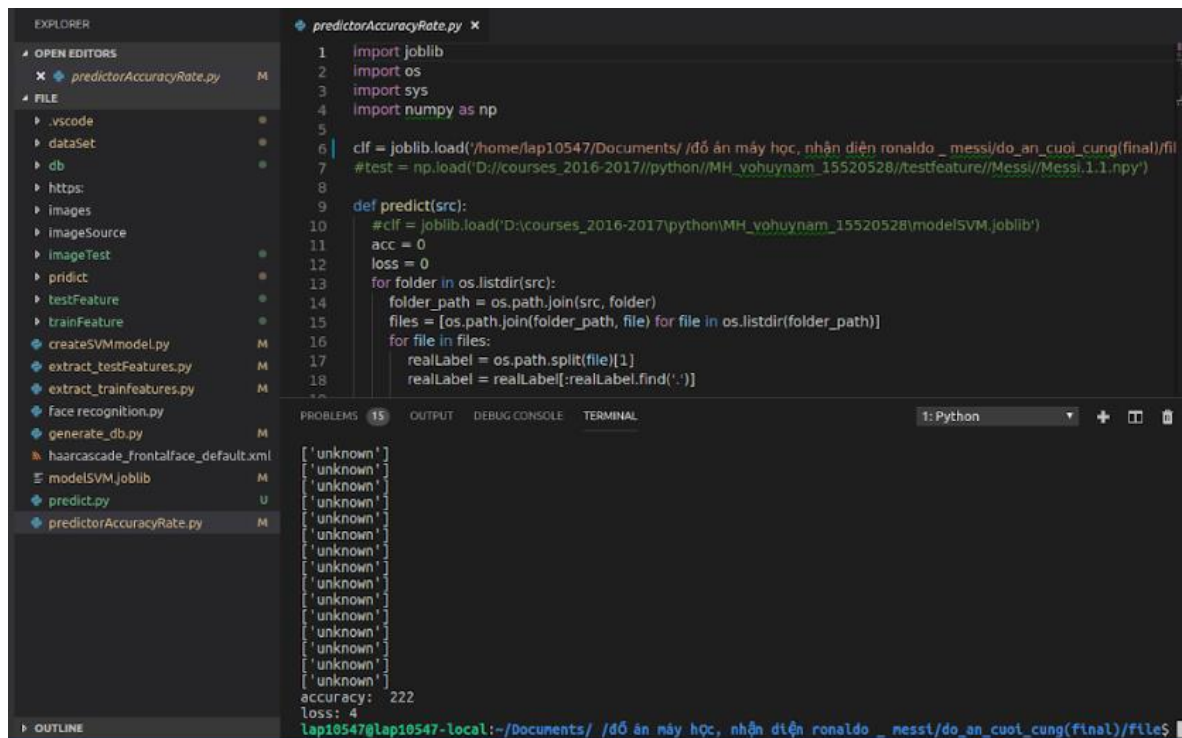
```
1 import numpy as np
2 import sys
3 import os
4 from sklearn.svm import LinearSVC
5 import joblib
6
7 def LoadFeature(src):
8     print("[+] Loading data...")
9     train_set = []
10    train_label = []
11    list_folder = os.listdir(src)
12    for label, folder in enumerate(list_folder):
13        list_file = os.listdir(os.path.join(src, folder))
14        for file in list_file:
15            a = os.path.join(src, folder, file)
16            feature = np.load(a)
17            train_set.append(feature[0])
18            train_label.append(folder)
19    print("[+] Load data finished")
20    return train_set, train_label
21
22 def Clustering(trainSet, trainLabel):
23     print("[!] Clustering data...")
24     clf = LinearSVC()
25     clf.fit(trainSet, trainLabel)
26     print("Training finished")
27     return clf
28
29 def SaveModel(model, name):
30     file_name = name + ".joblib"
31     print("[+] Saving model to file: ", file_name)
32     joblib.dump(model, file_name)
33
34
```

- Kết quả của quá trình xây dựng bộ phân lớp là ta sẽ có một file module có tên modelSVM.joblib



## 6. Kiểm tra độ chính xác

- Sau khi đã có bộ phân lớp và tập test, ta sẽ lần lượt test từng đặc trưng trong tập testFeature và kiểm tra xem độ chính xác của bộ phân lớp này
- Chạy file predictorAccuracyRate.py để kiểm tra độ chính xác



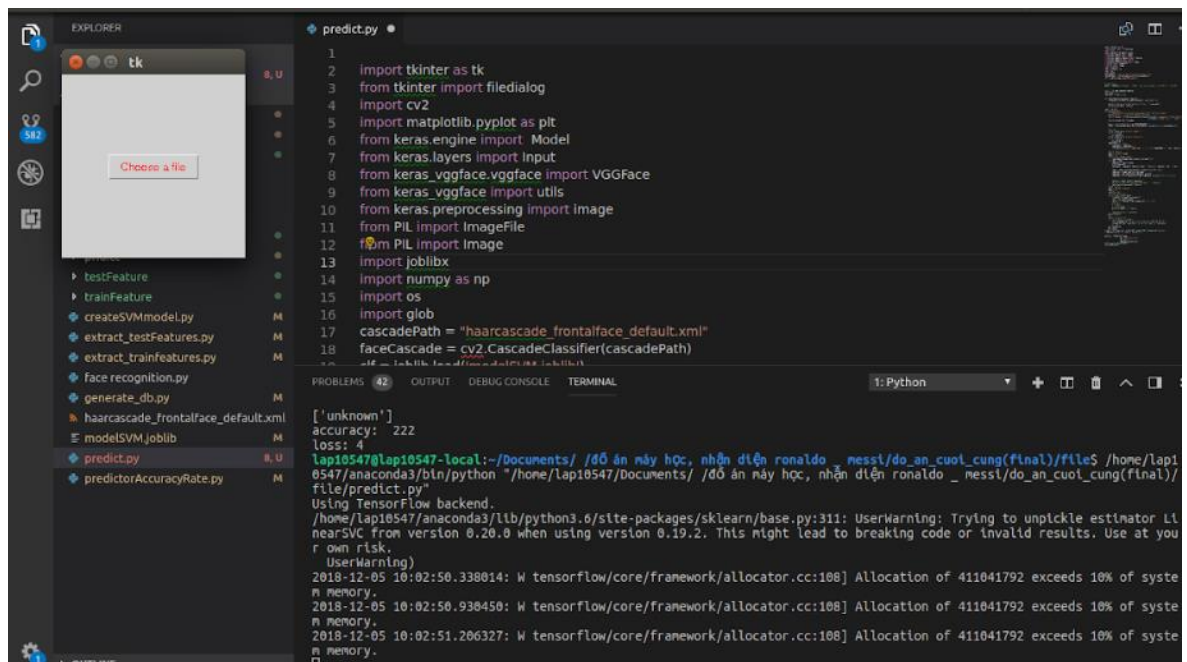
```
1 import joblib
2 import os
3 import sys
4 import numpy as np
5
6 clf = joblib.load('/home/lap10547/Documents/ /đồ án máy học, nhận diện ronaldo _ messi/do_an_cuoi_cung(final)/fi
7 #test = np.load('D:/courses_2016-2017/python/MH_vohuynam_15520528/testfeature/Messi/Messi.1.1.npy')
8
9 def predict(src):
10     #clf = joblib.load('D:/courses_2016-2017/python/MH_vohuynam_15520528/modelSVM.joblib')
11     acc = 0
12     loss = 0
13     for folder in os.listdir(src):
14         folder_path = os.path.join(src, folder)
15         files = [os.path.join(folder_path, file) for file in os.listdir(folder_path)]
16         for file in files:
17             realLabel = os.path.splitext(file)[1]
18             realLabel = realLabel[:realLabel.find('.')]
19
20     accuracy: 222
21     loss: 4
```

- Kết quả trả về là trong 226 ảnh được test, thì có 222 ảnh khuôn mặt được nhận dạng đúng, 4 ảnh bị nhận dạng sai.
- Nhận xét của nhóm: tuy kết quả đúng khá cao, nhưng vì dữ liệu của dataset để train và test còn khá ít nên kết quả này khá chủ quan.

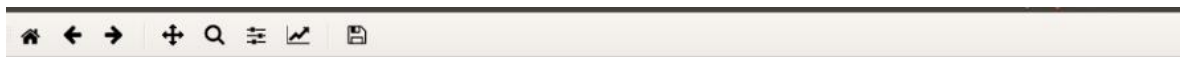
## 7. Kiểm tra thực tế

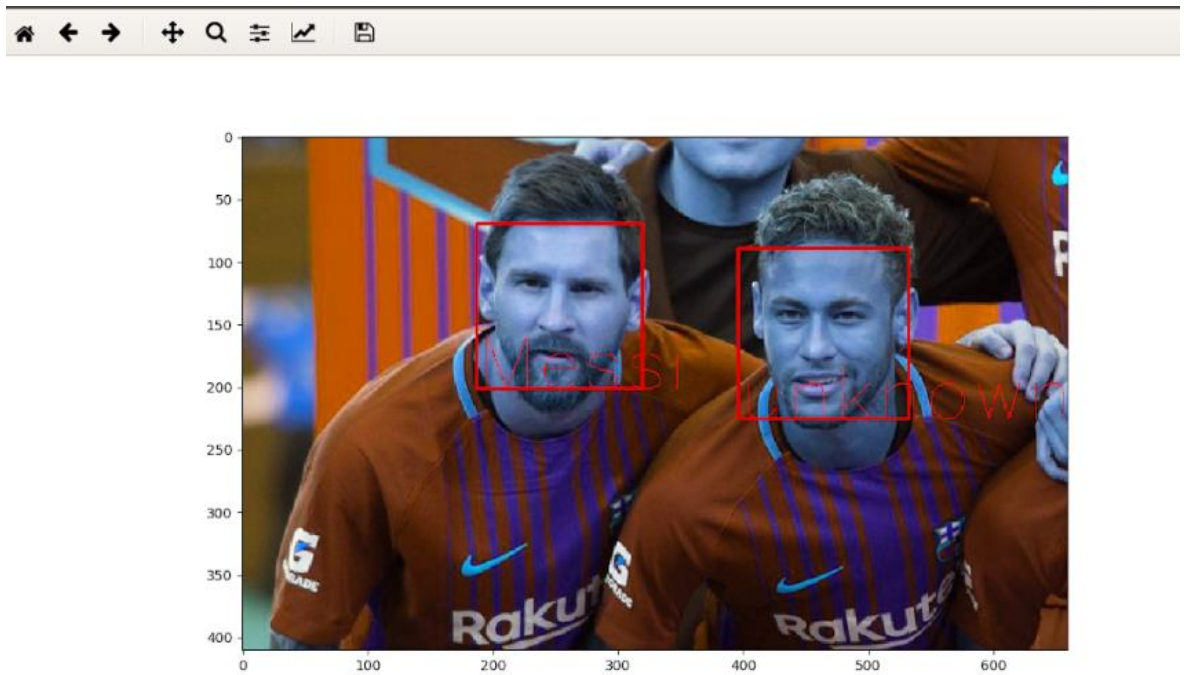
- Để nhận dạng khuôn mặt Ronaldo và Messi trong một bức ảnh, nhóm đã viết ột chương trình predict.py
- Chương trình predict.py sẽ cho phép bạn chọn một bức ảnh trong thư mục, load nó và sẽ đóng khung từng khuôn mặt và tên ứng với mỗi khuôn mặt
- Chạy file predict.py, giao diện hiện lên



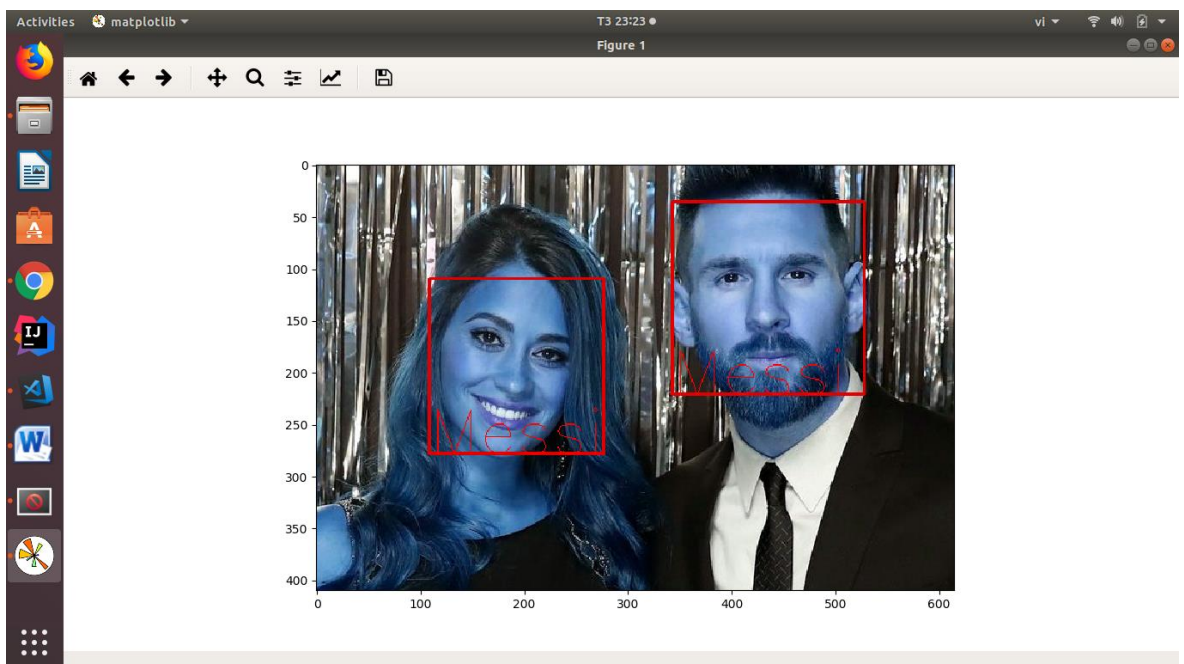


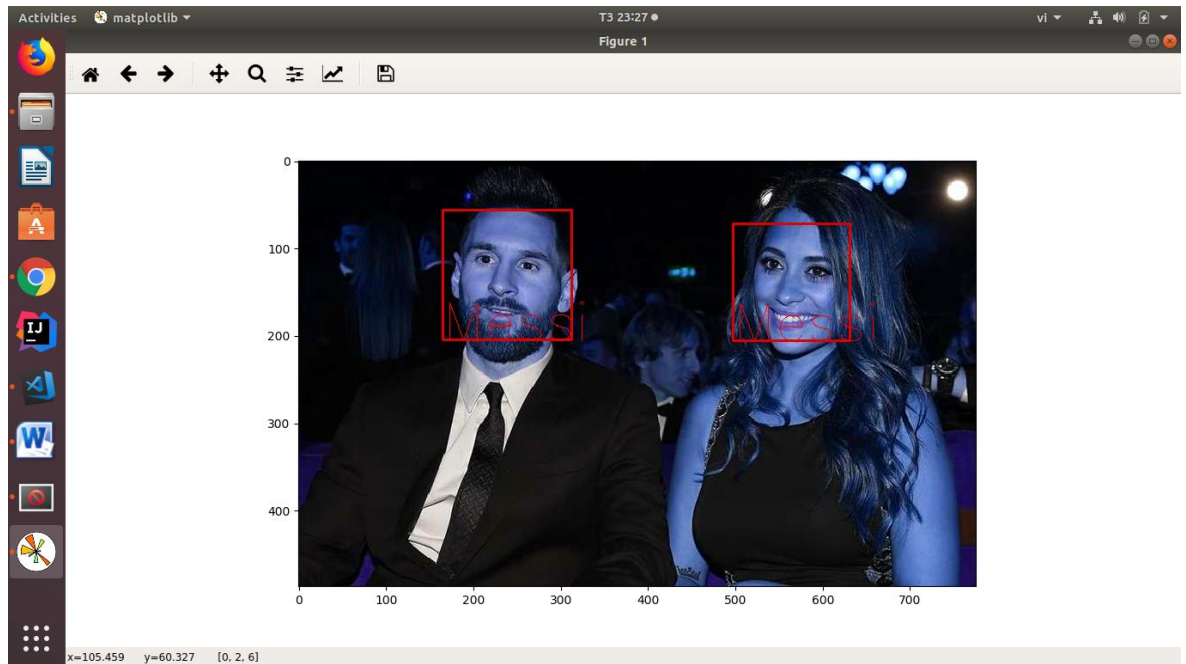
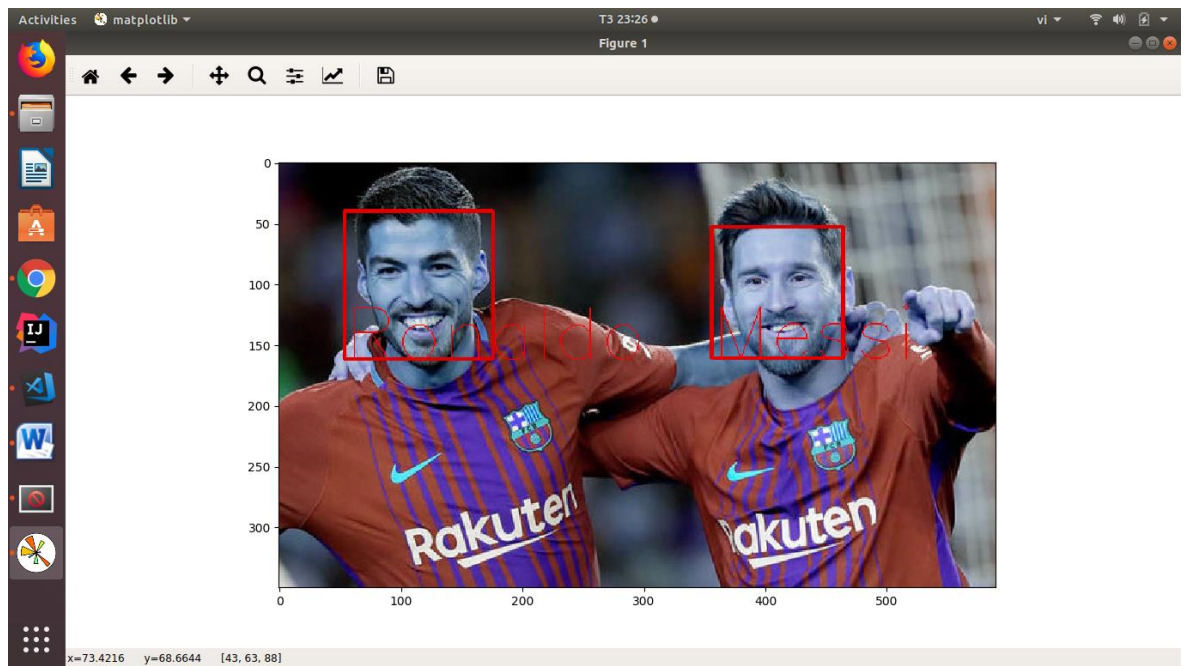
Chọn Choose a file để chọn 1 ảnh bất kỳ



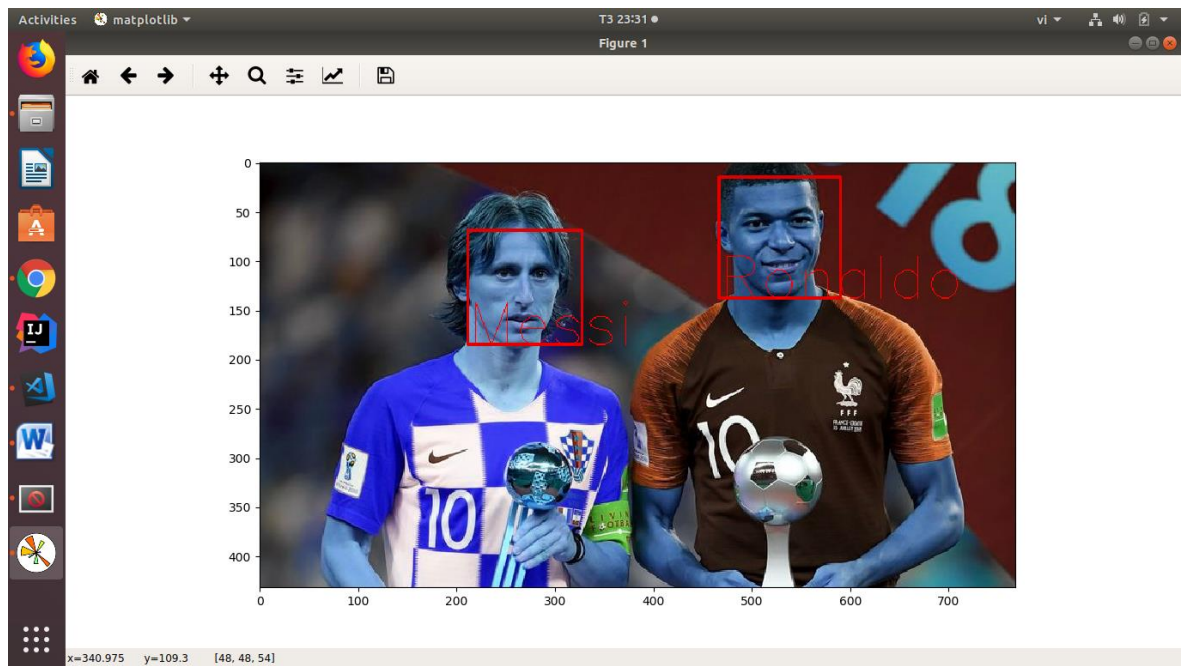


Nhưng trường hợp sai









Nguyên nhân sai: Do dữ liệu dùng để xây dựng bộ phân lớp ít.

Cách khắc phục : Tăng dữ liệu xây dựng bộ phân lớp

Tài liệu tham khảo

[Detection khuôn mặt dựa trên opencv:](#)

[VGGFACE](#)