CS331.I11.KHTN

THỊ GIÁC MÁY TÍNH NÂNG CAO

HỆ THỐNG PHÂN LOẠI ẢNH CÓ MẶT NGƯỜI DỰA TRÊN MẠNG VGG16 VÀ MODEL SVM

SVTH:

NGUYỄN CAO MINH – 14520529

ĐOÀN TRÍ ĐỨC – 14520178

GVDH:

NGÔ ĐỨC THÀNH

NGUYỄN VINH TIỆP

TP HCM, 20/1/2018

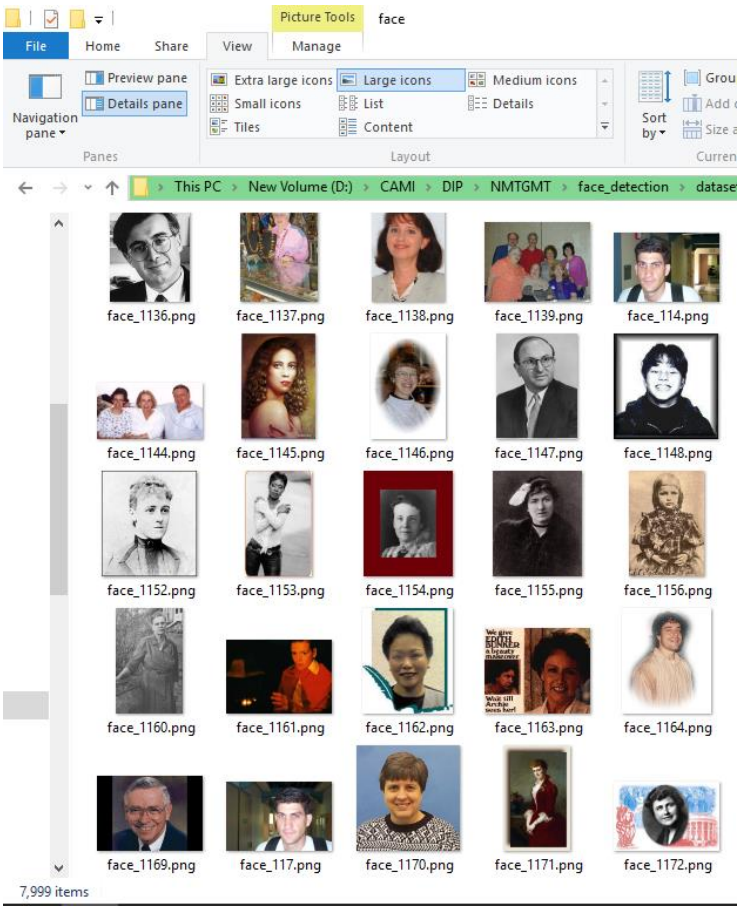
MỤC LỤC

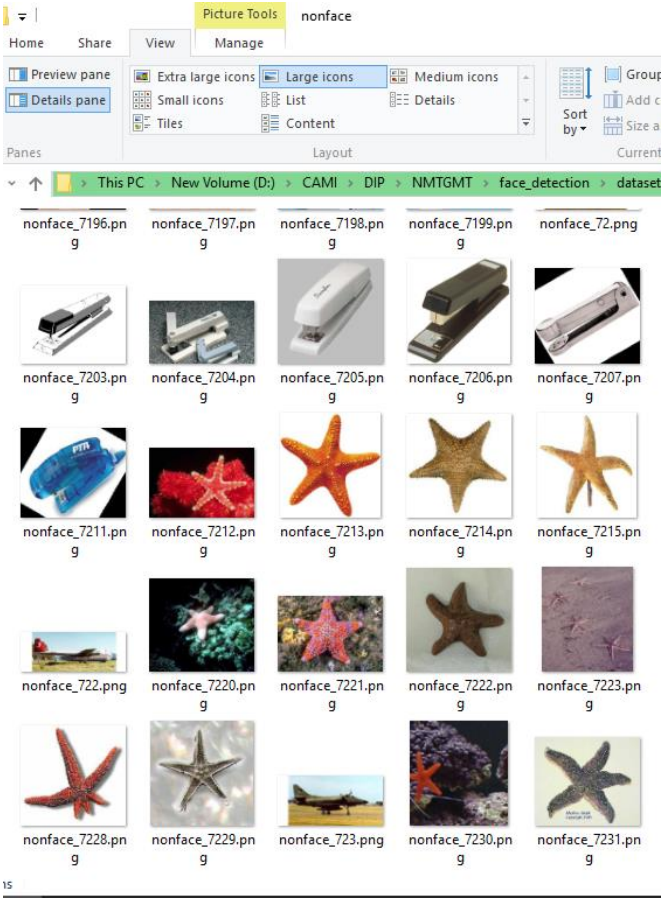
I. GIỚI THIỆU CHUNG:

- Hệ thống sử dụng 2 bộ dataset của Caltech:
  - Caltech-101[1]:>8000 ảnh của 101 loại object khác nhau.
  - Caltech 10,000 web faces[2]: >7000 ảnh khác nhau của các gương mặt con người được thu thập từ web.
- Hệ thống sẽ dùng thư viện Keras để sử dụng mạng neural network VGG16 (OxfordNet) rút trích các đặc trưng ảnh, sau đó dùng để đưa vào các bộ classifer (Linear classification, SVM) để xây dựng model phân lớp ảnh có chứa mặt người.

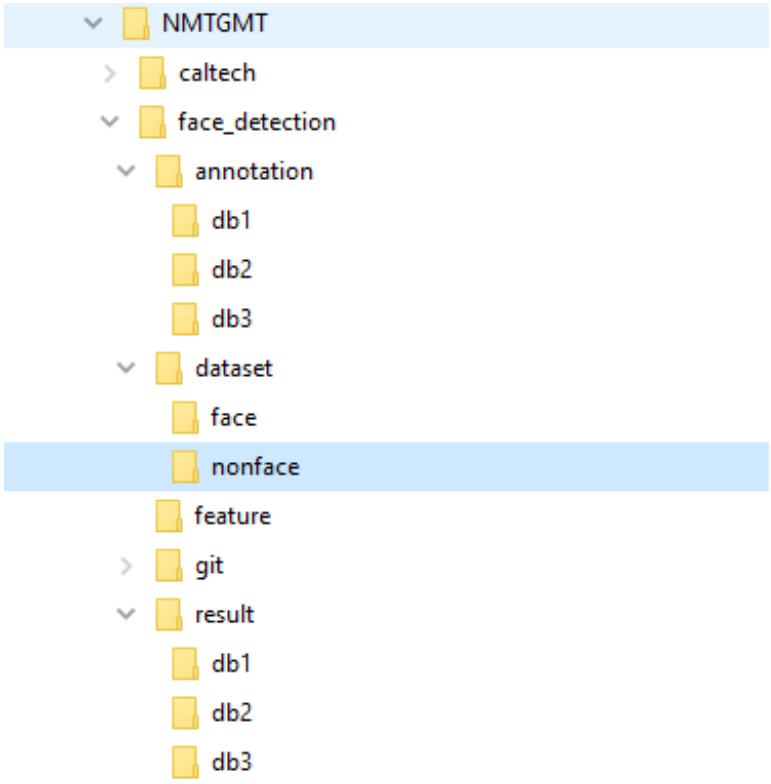
II. PHƯƠNG PHÁP THỰC HIỆN:

1. Khởi tạo bộ dataset:
   - Bộ dataset bao gồm 16273 tấm ảnh được tổng hợp từ 2 bộ dataset của Caltech: Caltech-101 và Caltech 10,000 web faces. Với mục đích của hệ thống bộ dataset có thể được chia thành 2 phần:
     - Phần có chứa mặt người: (7999 ảnh) Bộ ảnh từ Caltech 10,000 web faces.
     - Phần ảnh không tồn tại mặt người (8274 ảnh) Bộ ảnh từ Caltech-101.

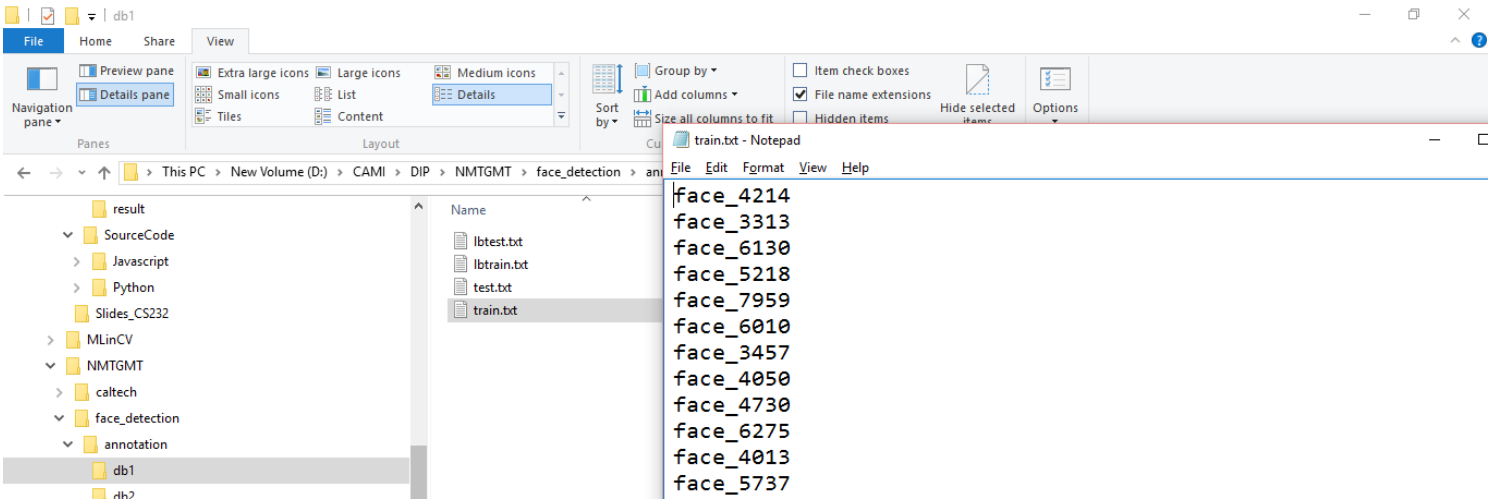




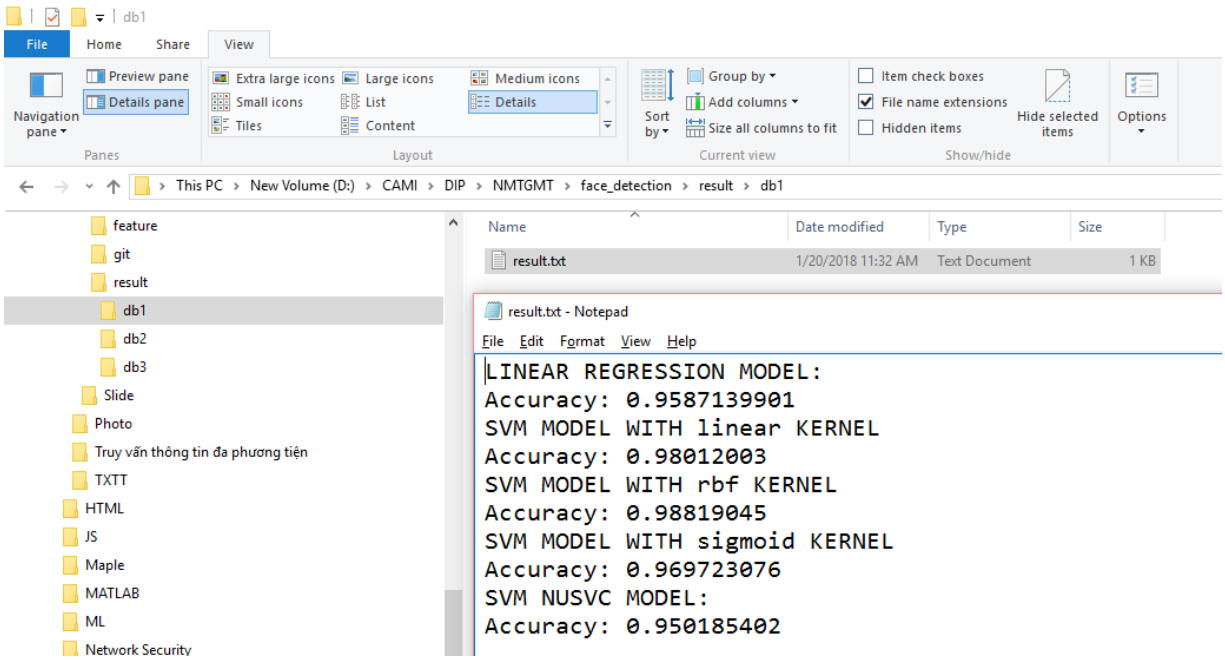
2. Xây dựng cấu trúc thư mục lưu trữ:
   - Thư mục lưu trữ hệ thống có dạng như sau:



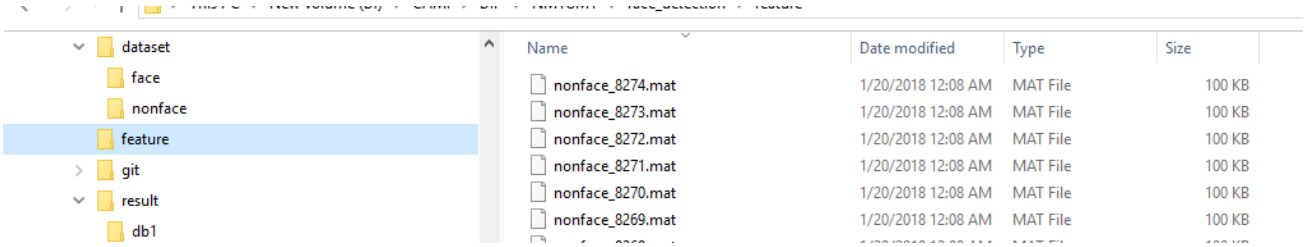
   - Face_detection: Thư mục chính của project
     - annotation: Thư mục lưu trữ các bộ dữ liệu bao gồm các tập train và test dưới dạng file .txt



- result: Thư mục lưu trữ kết quả so sánh độ chính xác khi áp dụng bộ phân loại ứng với các bộ dữ liệu trong thư mục annotation.



- Feature: thư mục lưu trữ ma trận feature của các bức ảnh được rút trích thông qua mạng neural VGG16.



- dataset: Thư mục chứa hình ảnh (bộ dataset)

3. Rút trích feature:
   - Sử dụng thư viện của Keras để dùng mạng neural VGG16 rút trích đặc trưng từ các tấm ảnh trong bộ dataset và lưu lại thành ma trận feature. (file .mat)
   - Kết quả rút trích feature được lấy ra tại output của layer “fc2” trong mô hình VGG16, vì ta chỉ cần lấy feature đặc trưng của tấm ảnh, nên

không cần phải đi qua lớp softmax của mô hình VGG16 (sigmoid function).
- Nền của thư viện Keras được sử dụng là TensorFlow.

4. Xây dựng bộ dữ liệu train và test:
   - Bộ dữ liệu train và test được chạy bằng hàm random theo tỷ lệ 7:3, tức mỗi 10 ảnh thì có 7 ảnh dùng để train và 3 ảnh dùng để test.
   - Cụ thể đối với 1 bộ dữ liệu ~16000 ảnh thì ta có ~11200 ảnh dùng để train và ~4800 ảnh dùng để test bộ phân loại được xây dựng từ bộ dữ liệu train.
   - Ứng với mỗi bộ dữ liệu train và test luôn có bộ label tương ứng dùng để đối chiếu trong việc training và evaluating trong hệ thống.
5. Xây dựng chương trình:
   - Chương trình được xây dựng dựa trên các bộ phân lớp sau:
     - Linear Regression.
     - Supported Vector Machine (SVM):
       - SVC:
         - Linear Kernel.
         - Rbf Kernel.
         - Sigmoid Kernel.
       - NuSVC.
6. Kiểm thử và so sánh:
   - Các mô hình đều được tính độ chính xác khi so sánh giữa label mà chương trình dự đoán ra và label mà người dùng mong đợi. Các kết quả sẽ được lưu vào file result.txt của mỗi bộ dữ liệu trong folder exps.
   - Sau đó sẽ dùng phần mềm excel để tính toán độ lệch chuẩn và vẽ biểu đồ so sánh độ chính xác giữa các mô hình và đưa ra kết luận.

III. CÁC KHÁI NIỆM CƠ BẢN:

- **VGG16[3]:**
  - VGG16 hay còn được gọi là OxfordNet là một mạng di truyền dạng xoắn (convolutional neural network) được đặt tên và xây dựng bởi nhóm Visual Geometry Group từ Oxford.
  - Mô hình này đã giành chiến tháng trong cuộc thi ILSVR (Image Net) năm 2014.

    - Cho đến nay, nó vẫn được xem là một mô hình tuyệt vời khi áp dụng vào các bài toán cái đầu vào là hình ảnh, mặc dù rằng đã có nhiều mô hình cải tiến từ nó như ResNet hay Inception.
- Keras[4]**:**
    - Keras là một API mạng di truyền cao cấp, là một thư viện neural network được viết cho ngôn ngữ python.
    - Được xây dựng dựa trên Tensorflow, CNTK hoặc Theano.
    - Được thiết kế để thực hiện các hàm tính toán thực thi với mạng di truyền sâu (deep neural network) một cách nhanh chóng với các cú pháp đơn giản.
- TensorFlow [5]:
    - Là một thư viện mã nguồn mở dùng để tính toán dựa trên việc biến đổi và xử lý dữ liệu như một đồ thị.
    - Mỗi nodes trong đồ thị tương ứng với một phép toán, và các cạnh biểu thị cho số chiều của dữ liệu sử dụng cũng như thể hiện sự liên kết giữa các dữ liệu.
    - Kiến trúc của TensorFlow thích hợp cho cả CPU và GPU, có thể thực hiện cài đặt sử dụng trên các loại thiết bị và platform khác nhau như desktop, mobile, server, ... chỉ thông qua 1 API duy nhất.
    - TensorFlow được phát triển và nghiên cứu bởi tổ chức nghiên cứu Google's Machine Learning với mục đích giải quyết và thực hiện các bài toán liên quan đến máy học cũng như là các mạng di truyền sâu.
    - TensorFlow hiện tại đang được áp dụng trong rất nhiều lĩnh vực khác nhau để giải quyết các bài toán liên quan đến Machine Learning và Deep Neural Network.
- Linear Regression[6]**:**
    - Trong thống kê, hồi qui tuyến tính là một cách tiếp cận tuyến tính để mô hình hóa mối quan hệ giữa một biến phụ thuộc vô hướng y và một hoặc nhiều biến giải thích (hoặc các biến độc lập) biểu thị X.

    - Trong hồi quy tuyến tính, các mối quan hệ được mô hình bằng các hàm dự báo tuyến tính mà các tham số mô hình không xác định là ước tính từ dữ liệu.
- Supported Vector Machine (SVM) [7]:
    - Là một tập các phương pháp học tập có giám sát (supervised), được sử dụng trong việc phân lớp (classification), hồi quy (regression), và phát hiện các điểm ngoại biên (outliers detection).
    - SVC là một mô hình classification trong SVM[7].
- NuSVC[9]:
    - Nu-Support Vector Classification, tương tự như SVC nhưng sử dụng thêm tham số Nu để điều chỉnh độ lỗi có thể chấp nhận được của các support vectors.

IV. CÁC HÀM THỰC THI:

1. Môi trường, ngôn ngữ và các thư viện sử dụng:
    - Ngôn ngữ lập trình được sử dụng: Python 3.
    - Các thư viện phục vụ:
        - Tensorflow
        - Keras
        - Sklearn
2. Hàm khởi tạo ngẫu nhiên bộ dữ liệu train và test:
    - Hàm khởi tạo ngẫu nhiên 1 tập hơn 7000 số bao gồm các chữ số hoàn toàn ngẫu nhiên và khác nhau.
    - Sử dụng hàm chuyển đổi tập ~7000 số thành tên của ~7000 file ứng với mỗi folder face và folder nonface (tổng số ~16000 ảnh), trong đó lấy 70% số ảnh của mỗi tập làm bộ ảnh train, 30% còn lại được sử dụng trong việc test.

- Mỗi bộ dữ liệu được thực hiện theo trình tự như trên, để đảm bảo tính khách quan của bộ dữ liệu, chương trình thực hiện khởi tạo ngẫu nhiên 3 bộ dữ liệu và tiến hành tính toán các giá trị trung bình khi áp dụng từng bộ dữ liệu vào các mô hình khác nhau.

```
25  #FUNCTION: RANDOM TRAINING DATA
26  def random_file(path_file):
27      import random
28      num_rand = random.sample(range(1,8000),7999)
29      #FACE
30      file_train_txt = open(path_file+"/train.txt","w")
31      file_train_lb = open(path_file+"/lbtrain.txt","w")
32      file_test_txt = open(path_file+"/test.txt","w")
33      file_test_lb = open(path_file+"/lbtest.txt","w")
34      for i in num_rand[0:5600]:
35          file_train_txt.write("face_"+str(i)+"\n")
36          file_train_lb.write("1\n")
37      for i in num_rand[5600:7999]:
38          file_test_txt.write("face_"+str(i)+"\n")
39          file_test_lb.write("1\n")
40      file_train_txt.close()
41      file_train_lb.close()
42      file_test_txt.close()
43      file_test_lb.close()
44
45      #NONFACE
46      num_rand = random.sample(range(1,8275),8274)
47      file_train_txt = open(path_file+"/train.txt","a")
48      file_train_lb = open(path_file+"/lbtrain.txt","a")
49      file_test_txt = open(path_file+"/test.txt","a")
50      file_test_lb = open(path_file+"/lbtest.txt","a")
51      for i in num_rand[0:5792]:
52          file_train_txt.write("nonface_"+str(i)+"\n")
53          file_train_lb.write("0\n")
54      for i in num_rand[5792:8274]:
55          file_test_txt.write("nonface_"+str(i)+"\n")
56          file_test_lb.write("0\n")
57      file_train_txt.close()
58      file_train_lb.close()
59      file_test_txt.close()
60      file_test_lb.close()
```

3. Hàm rút trích feature:
   - Sử dụng mạng VGG16 full-connected để rút trích các đặc trưng từ ảnh, kết quả được lấy từ layer "fc2" với output là ma trận chứa 4096 vector đặc trưng của bức ảnh (1x4096).
   - Weight được sử dụng trong mạng VGG16 là bộ dataset của ImageNet.

```
#FUNCTION: EXTRACT FEATURE VGG16
def extract_fea_vgg16(path_fea,path_img):
    from vgg16 import VGG16
    from keras.preprocessing import image
    from keras.models import Model
    from imagenet_utils import preprocess_input
    import numpy as np
    import glob
    import os
    vgg16model = VGG16(weights='imagenet', include_top=True);
    model = Model(inputs=vgg16model.input, outputs=vgg16model.get_layer('fc2').output)
    i=1
    for filename in glob.glob(os.path.join(path_img, '*.png')):
        name_save=(filename.replace(path_img,"")).replace(".png","")
        file_feature = open(path_fea+"/"+name_save+".mat","wb")
        img_path=filename
        img = image.load_img(img_path, target_size=(224, 224))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        x = preprocess_input(x)
        features = model.predict(x)
        np.savetxt(file_feature, features)
        file_feature.close()
        print("Step "+str(i)+" done")
        i=i+1
    print("DONE EXTRACT FEATURE")
```

4. Hàm xây dựng chương trình và kết quả so sánh:
   - Trước khi thực hiện việc đưa dữ liệu input vào các bộ phân loại để training, cần khởi tạo một ma trận chứa tất cả các feature đặc trưng của các ảnh sẽ dùng để training và test, cũng như ma trận chứa label tương ứng của các bộ ảnh đó.
   - Hàm get_training_input sẽ trả về các ma trận:
     - arr_test/arr_train: Ma trận chứa tất cả các feature của n ảnh dùng để test/train, ma trận có dạng (nx4096) – ma trận bao gồm n ảnh, mỗi ảnh có 4096 bộ feature đặc trưng.
     - arr_lbtest/arr_lbtrain: Ma trận chứa các nhãn tương ứng với các ảnh của bộ test/train, ma trận có dạng (nx1) – ma trận bao gồm n ảnh, mỗi ảnh chứa 1 giá trị là 0: nếu là nữ và 1: nếu là nam

```
#FUNCTION: MERGE DATA TO 1 MATRIX
def get_training_input(path_training,path_features):
    arr_test = initialize_input(path_training,path_features,"test.txt")
    arr_lbtest = initialize_input(path_training,path_features,"lbtest.txt")
    arr_train = initialize_input(path_training,path_features,"train.txt")
    arr_lbtrain = initialize_input(path_training,path_features,"lbtrain.txt")
    return [
        arr_test,
        arr_lbtest,
        arr_train,
        arr_lbtrain
    ]
```

- Các bộ phân loại áp dụng:
    - Linear Regression:
        - Sử dụng thư viện sklearn, gọi bộ phân loại Linear Regression trong classs linear bộ phân loại của sklearn.
        - Sau đó fit bộ phân loại với tập train và label train đã được khởi tạo từ trước thông qua hàm fit().
        - Sau khi bộ phân loại đã được training, ta sử dụng hàm predict() cho bộ dữ liệu test để bộ phân loại đưa ra các kết luận dựa trên dữ liệu đã được học tập.

```
#FUNCTION: LINEAR REGRESSION MODEL
def sklearn_linear_model(x_train,lbx_train,x_test,lbx_test,path_result):
    from sklearn import linear_model
    from sklearn.metrics import mean_squared_error
    x_train=x_train.tolist()
    x_test=x_test.tolist()
    lbx_train=(lbx_train.ravel()).tolist()
    lbx_test=(lbx_test.ravel()).tolist()

    file_result_txt = open(path_result+"/result.txt","w")
    # Create linear regression object
    regr = linear_model.LinearRegression()
    # Train the model using the training sets
    regr.fit(x_train, lbx_train)
    # Make predictions using the testing set
    x_pred = regr.predict(x_test)
    print("Mean squared error: %.2f"
              % mean_squared_error(lbx_test, x_pred))
    result = 1 - mean_squared_error(lbx_test, x_pred)

    file_result_txt.write("LINEAR REGRESSION MODEL:\n")
    file_result_txt.write("Accuracy: "+str(result)+"\n")
    file_result_txt.close()
```

- Ta thực hiện phép so sánh giữa kết quả label mà bộ phân loại dự đoán ra được với ma trận label chuẩn mà ta đã khởi tạo từ trước.

- SVM.SVC:
  - Cách thực hiện tương tự như bộ phân loại Linear Regression, tuy nhiên trong bộ phân loại SVC của thư viện SVM của sklearn. Ta có nhiều kernel để sử dụng: linear, rbf, sigmoid.

- Ngoài ra ta còn có thể áp dụng bộ phân loại NuSVC trong thư viện SVM, việc so sánh kết quả giữa bộ phân loại dự đoán từ dữ liệu học tập và kết quả chuẩn cũng tương tự như bộ phân loại Linear Regression.

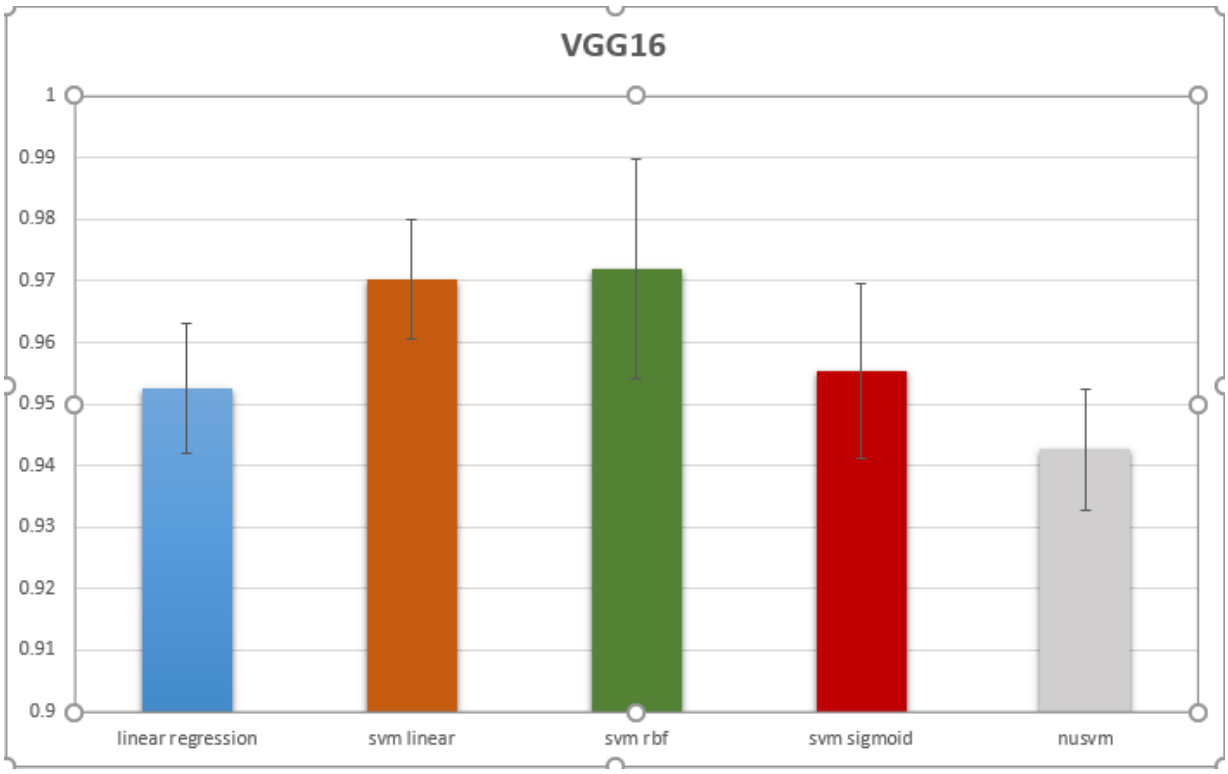
```
169
170  #FUNCTION: SVM MODEL
171  def sklearn_svm(x_train,lbx_train,x_test,lbx_test,kernel_type,kernel_input,path_result):
172      from sklearn import svm
173      from sklearn.metrics import mean_squared_error
174      x_train=x_train.tolist()
175      x_test=x_test.tolist()
176      lbx_train=(lbx_train.ravel()).tolist()
177      lbx_test=(lbx_test.ravel()).tolist()
178      file_result_txt = open(path_result+"/result.txt","a")
179
180      if kernel_type == "nonlinear":
181          clf = svm.NuSVC()
182          clf.fit(x_train, lbx_train)
183          x_pred = clf.predict(x_test)
184           # The mean squared error
185          print("NuSVC Kernel :")
186          print("Mean squared error: %.2f"
187                % mean_squared_error(lbx_test, x_pred))
188          result = 1 - mean_squared_error(lbx_test, x_pred)
189          # Explained variance score: 1 is perfect prediction
190          file_result_txt.write("SVM NUSVC MODEL:\n")
191          file_result_txt.write("Accuracy: "+str(result)+"\n")
192          file_result_txt.close()
193      else:
194          clf = svm.SVC(kernel=kernel_input)
195          clf.fit(x_train, lbx_train)
196          x_pred = clf.predict(x_test)
197          # The mean squared error
198          print("SVC Kernel " + kernel_input + " :")
199          print("Mean squared error: %.2f"
200                % mean_squared_error(lbx_test, x_pred))
201          result = 1 - mean_squared_error(lbx_test, x_pred)
202          file_result_txt.write("SVM MODEL "+"WITH "+kernel_input+" KERNEL"+"\n")
203          file_result_txt.write("Accuracy: "+str(result)+"\n")
204          file_result_txt.close()
205
```

V. ĐÁNH GIÁ KẾT QUẢ:

- Kết quả đạt được khi áp dụng các bộ phân loại trên 3 bộ dữ liệu được thống kê như sau:

2	Model	db1	db2	db3	Accuracy Score	Deviation
3	linear regression	0.958714	0.961371	0.937861	0.952648657	0.010512342
4	svm linear	0.98012	0.973776	0.957211	0.970368958	0.009657862
5	svm rbf	0.98819	0.980537	0.947211	0.971979485	0.017790344
6	svm sigmoid	0.969723	0.960664	0.936126	0.955504125	0.014193095
7	nusvm	0.950185	0.948986	0.9287	0.942623862	0.009857602

- Trong đó:
  - Accuracy Score: Độ chính xác trung bình của bộ phân loại (trung bình độ chính xác của 3 bộ dữ liệu)
  - Deviation: Độ lệch chuẩn của bộ phân loại. Sử dụng hàm STDEV.P của Excel để tính toán.
- Ta có biểu đồ thống kê độ chính xác và độ lệch chuẩn giữa các phương pháp:



VI. KẾT LUẬN:

- Từ các số liệu và biểu đồ thống kê khi thực hiện trên các bộ dữ liệu khác nhau, ta thấy rằng bộ phân loại SVM với kernel linear và SVM với kernel rbf cho ra kết quả tốt nhất (>97%), tuy nhiên độ lệch chuẩn của kernel linear lại tốt hơn so với rbf, nguyên do vì kết quả phân loại của chúng ta chỉ có 2 nhãn là 1 hoặc 0. Do đó những bộ phân loại có kernel là linear sẽ cho lại kết quả tốt hơn.
- Dù linear regression bộ phân loại vẫn sử dụng phương pháp áp dụng linear classify trong việc training và xây dựng bộ phân loại tuy nhiên, do input đầu vào của chúng là bộ feature của ảnh (1x4096), do đó linear regression không phát huy hết khả năng tối ưu của nó (vốn được áp dụng rất nhiều đối với dữ liệu 1 hoặc 2 chiều), trong khi đó input đầu vào của chúng ta là một ảnh có 4096 chiều vector đặc trưng.
- Material và Source Code: https://github.com/bigredbug47/FACE_CLASSIFICATION

VII. TÀI LIỆU THAM KHẢO:

[1] http://www.vision.caltech.edu/Image_Datasets/Caltech101/Caltech101.html

[2] http://www.vision.caltech.edu/Image_Datasets/Caltech_10K_WebFaces/

[3] https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

[4] https://en.wikipedia.org/wiki/Keras - https://keras.io/

[5] https://www.tensorflow.org/

[6] http://www.statisticssolutions.com/what-is-linear-regression/

[7][8] https://en.wikipedia.org/wiki/Support_vector_machine

[9] https://www.ncbi.nlm.nih.gov/pubmed/11516360