TRƯỜNG ĐẠI HỌC ĐIỆN LỰC

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO CHUYÊN ĐỀ HỌC PHẦN

HỌC MÁY NÂNG CAO

**ĐỀ TÀI: TÌM HIỂU OPEN VINO**

PHƯƠNG PHÁP GIẢM CHIỀU PCA VÀ ỨNG DỤNG

|  |  |
| --- | --- |
|  |  |
| |  |  | | --- | --- | | **Sinh viên thực hiện** | **: LƯU ĐÌNH KIÊN**  **TRƯƠNG VIỆT HOÀNG** | | **Giảng viên hướng dẫn** | **: PHẠM THỊ KIM DUNG** | | | **Ngành** | **: CÔNG NGHỆ THÔNG TIN** | | | **Chuyên ngành** | **: CÔNG NGHỆ PHẦN MỀM** | | | **Lớp** | **: D12CNPM2** | | |  |

**Hà Nội, 01/2021**

PHIẾU CHẤM ĐIỂM

Sinh viên thực hiện:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ tên** | **Nội dung thực hiện** | **Điểm** | **Chữ kí** |
| 1 | Lưu Đình Kiên |  |  |  |
| 2 | Trương Việt Hoàng |  |  |  |

Giảng viên chấm:

|  |  |  |
| --- | --- | --- |
| **Họ và tên** | **Chữ ký** | **Ghi chú** |
| Giảng viên chấm 1: |  |  |
| Giảng viên chấm 2: |  |  |

MỤC LỤC

[Chương 1:Tổng quan về Open VINO](#_Toc54244799)

1.1.Khái niệm OpenVINO**…………………………………………………....** 1.2. Quy trình suy luận căn bản……………………………………………….  
 1.2.1: Tối ưu hóa Model………………………………………………..  
 1.2.2: Đánh giá hiệu năng Benchmarks………………………………..  
1.2.3: OpenVINO và OpenCV………………………………………...  
 1.2.4: OpenVINO model server……………………………………….  
 1.2.5: Điểm hạn chế……………………………………………………

[Chương 2 : Phương pháp giảm chiều PCA](#_Toc54244811)

[2.1. Tổng quan về PCA](#_Toc54244812)

[2.2. Đặc tính của PCA](#_Toc54244813)

[2.3. Mô hình PCA](#_Toc54244814)

[2.4. Phương pháp phân tích thành phần chính – Principle Component Analysis (PCA)](#_Toc54244815)

[2.5. Giải pháp nhận diện khuôn mặt](#_Toc54244816)

[2.5.1. Tổng quan về nhận diện khuôn mặt](#_Toc54244817)

[2.5.2. Lĩnh vực ứng dụng](#_Toc54244818)

[2.5.3. Phương thức hoạt động](#_Toc54244819)

[Chương 3 : Thực nghiệm](#_Toc54244820)

[3.1. Ý tưởng bài toán.](#_Toc54244821)

[3.2. Tính toán các Eigenfaces](#_Toc54244822)

[3.3. Biểu diễn các khuôn mặt có sẵn (tập huấn luyện) vào trong không gian vector này](#_Toc54244823)

[3.4. Nhận dạng khuôn mặt bằng các EIGENFACES](#_Toc54244824)

[3.5. Kết quả thực nghiệm](#_Toc54244825)

[3.5.1. Tập dữ liệu huấn luyện](#_Toc54244826)

[3.5.2. Kết quả thực nghiệm](#_Toc54244827)

DANH MỤC HÌNH ẢNH

[Hình 2.1: Biểu diễn dữ liệu trên trục tọa độ 15](#_Toc54139428)

[Hình 2.2: Lựa chọn trục tọa độ mới để biểu diễn dữ liệu 16](#_Toc54139429)

[Hình 2.3: Dữ liệu sau khi thực hiện giảm chiều 16](#_Toc54139430)

[Hình 2.4: Ứng dụng tại Chung cư, trường học 19](#_Toc54139431)

[Hình 2.5: Ứng dụng tại Văn phòng, an ninh. 19](#_Toc54139432)

[Hình 2.6: Nhận diện khuôn mặt 20](#_Toc54139433)

[Hình 3.1:Mô hình Thuật toán PCA 21](#_Toc54139434)

[Hình 3.2: Dữ liệu khuôn mặt 23](#_Toc54139435)

[Hình 3.3: Mô hình Eigenfaces 30](#_Toc54139436)

[Hình 3.4: Biểu diễn Eigenfaces 31](#_Toc54139437)

[Hình 3.5: Diễn giải hình học phương pháp PCA 32](#_Toc54139438)

[Hình 3.6: Ví dụ giảm chiều dữ liệu 32](#_Toc54139439)

LỜI NÓI ĐẦU

Ngày nay, với cuộc cách mạng 4.0 cùng sự bùng nổ và phát triển của mạng 5G, AI (Artificial Intelligence)hay còn gọi là Trí tuệ nhân tạo đã trở thành một ngành khoa học máy tính được nhiều công ty công nghệ lớn trên thế giới rất quan tâm phát triển. Trí tuệ nhân tạo là một bộ phận của khoa học máy tính liên quan đến việc tự động hóa và thông minh, do đó nó được xây dựng dựa trên những nguyên lý lý thuyết vứng chắc, có khả năng ứng dụng cao len lỏi vào mọi lĩnh vực trong đời sống chúng ta.

Hầu hết mọi ngành công nghệ đang làm việc với lượng dữ liệu lớn ngày nay đều nhận ra tầm quan trọng của học máy, bởi học máy là một phương pháp để huấn luyện dữ liệu, giúp máy tính trở nên thông minh hơn thông qua các phương pháp học và cụ thể hơn là các thuật toán. Vậy nên, các ứng dụng của học máy đã dần trở nên quen thuộc với mọi người: xe tự hành của Google và Tesla, hệ thống tự tag khuôn mặt trên Facebook, hệ thống gợi ý sản phẩm của Amazon, …. Sau một thời gian tìm hiểu về học máy với mong muốn áp dụng được những kiến thức đã học vào thực tiễn, chúng em đã quyết định chọn đề tài “**PHƯƠNG PHÁP GIẢM CHIỀU PCA VÀ ỨNG DỤNG**”

***Chúng em xin chân thành cảm ơn!***

**Sinh viên thực hiện**

**Lưu Đình Kiên**

**Trương Việt Hoàng**

**Chương 1: Tổng quan về Open VINO**

**1.1: Khái niệm OpenVINO**

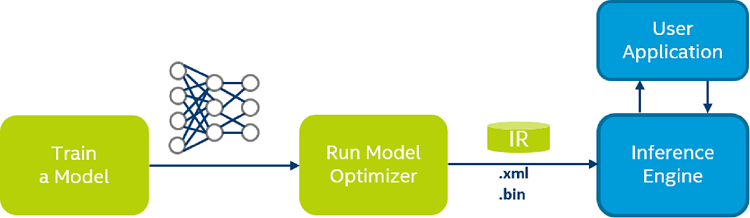
OpenVINO toolkit được xây dựng và phát triển bởi Intel, được tạo ra nhằm mục đích tối ưu hóa hiệu năng của model trên chính các bộ vi xử lý của Intel, cải thiện thời gian inference khi tiến hành deploy model trên rất nhiều các platform khác nhau (CPU / GPU / VPU / FPGA).

OpenVINO provides developers with improved neural network performance on a variety of Intel® processors and helps them further unlock cost-effective, real-time vision applications

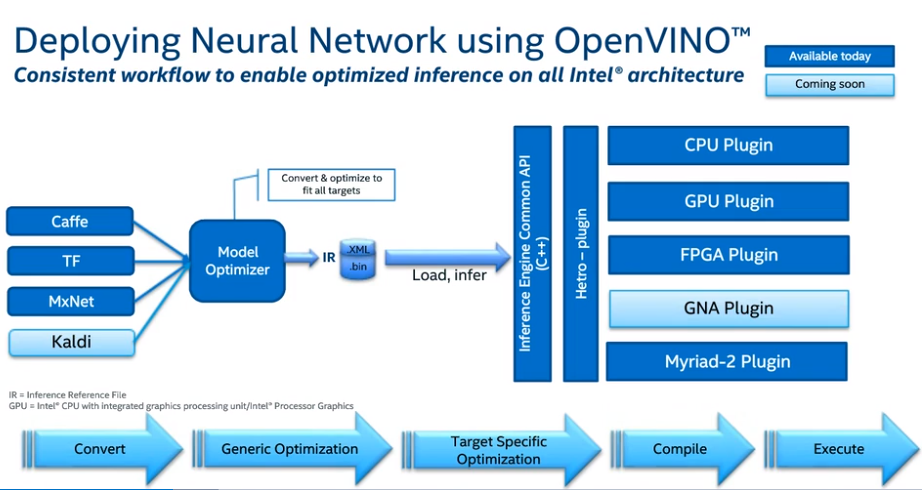
OpenVINO là viết tắt của Open Visual Inference and Neural network Optimization toolkit. Như các bạn nhìn vào tên cũng có thể đoán được OpenVINO được phát triển với mục đích hướng tới cải thiện khả năng Inference của model, đặc biệt liên quan tới các bài toán về Visual, computer vision như: Image Classification, Object Detection, Object Tracking, ...

* 1 số điểm đáng lưu ý của OpenVINO toolkit:
  + Cải thiện performance, khả năng inference time của model
  + Chính vì do Intel phát triển nên support đa platform khác nhau, từ CPU / GPU tới các thiết bị nhúng, dạng edge device như: VPU (Vision Processing Unit), Myriad, Modivius, hay FPGA, ....
  + Sử dụng chung 1 API cho công đoạn inference, nghĩa là bạn chỉ cần thay đổi mode đầu vào là có thể sử dụng các định dạng IR (Intermediate Representation của OpenVINO) trên nhiều các platform khác nhau.
  + Cung cấp khá nhiều các optimized model, việc convert sang định dạng trung gian IR của OpenVINO cũng khá dễ thực hiện.
  + Hỗ trợ việc gọi các file định dạng IR format (OpenVINO) bằng chính các thư viện phổ biến về xử lí ảnh / computer vision như: OpenCV và OpenVX
  + Gồm 2 phần chính:
    - Deep Learning deployment toolkit: <https://github.com/opencv/dldt/>
    - OpenVINO model zoo: <https://github.com/opencv/open_model_zoo> . Cung cấp các optimized model của OpenVINO cho các task cơ bản và các mô hình phổ biến. Tương tự như Tensorflow Model Zoo vậy.

**1.2: Quy trình suy luận căn bản**



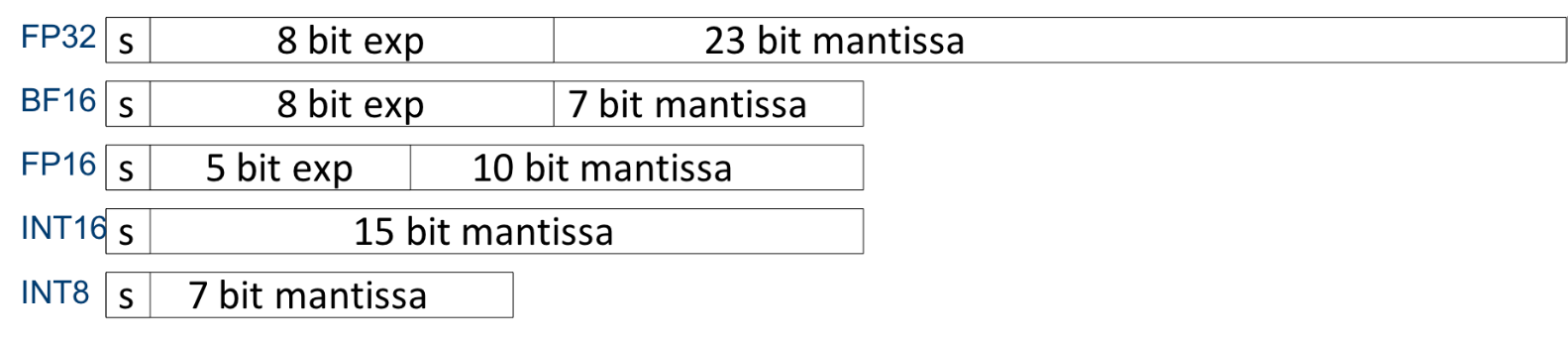
* Ta cần convert pretrained model sang định dạng IR hay Intermediated Representation của OpenVINO. IR format gồm 1 số file như sau:
  + frozen-\*.xml: network topology, là 1 file xml định nghĩa các layer của model, hay network graph.
  + frozen-\*.bin: contains the weights and biases binary data, file chứa trọng số của model, có thể convert dưới các định dạng: FP32, FP16, INT8
* OpenVINO cũng hỗ trợ việc convert sang IR format cho đa số các framework phổ biến hiện nay như:
  + Caffe
  + Tensorflow
  + MXNet
  + Kaldi
  + ONNX
  + [Keras / Pytorch]
* Ngoài ra, 1 vài các framework khác như: Keras và Pytorch tuy không hỗ trợ convert trực tiếp từ các pretrained model nhưng hoàn toàn có thể thông qua các định dạng trung gian khác:
  + Keras: bạn có thể convert file .h5 của keras sang dạng format frozen graph .pb của tensorflow, rồi từ đó convert frozen model đó sang định dạng IR của OpenVINO. Link tham khảo chuyển đổi từ h5 model sang frozen model của tensorflow:
    - <https://stackoverflow.com/questions/45466020/how-to-export-keras-h5-to-tensorflow-pb>
    - <https://www.dlology.com/blog/how-to-convert-trained-keras-model-to-tensorflow-and-make-prediction/>
  + Pytorch: hiện tại thì OpenVINO chưa thực sự support Pytorch nhưng file model .pth của pytorch có thể convert sang dạng format .onnx của ONNX, rồi từ đó convert sang IR format.



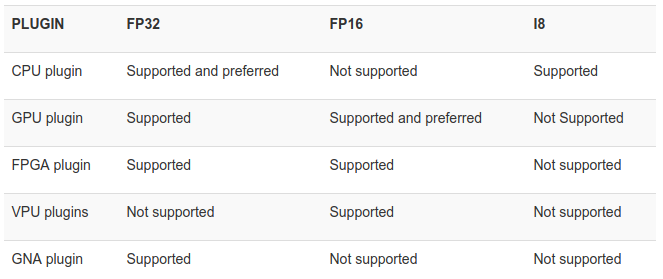
* 1 luồng xử lý với OpenVINO được mô tả qua hình trên.

**1.2.1: Tối ưu hóa Model**

* OpenVINO hỗ trợ việc convert sang IR format với các định dạng FP32, FP16, INT8, cũng là 1 dạng của model quantization (hay lượng tử hóa model)



* Việc convert sang các định dạng như FP16, INT8 giúp giảm thiểu dung lượng của model, giảm thiểu bộ nhớ khi tiến hành request, đồng thời giúp handle được nhiều request hơn, tăng tốc độ inference time trong khi độ chính xác của model thay đổi không đáng kể.
* Tuy nhiên, việc hỗ trợ các định dạng còn phụ thuộc vào các thiết bị khác nhau (CPU / GPU / VPU / FPGA)



Command mẫu, khá ngắn gọn:

python3 /opt/intel/openvino/deployment\_tools/model\_optimizer/mo.py --input\_model INPUT\_MODEL

# tensorflow model

python3 mo.py --framework tf --input\_model /user/models/model.pb --data\_type FP32

python3 mo\_tf.py --input\_model /user/models/model.pb --data\_type FP16

Ví dụ 1 đoạn command export YoloV3 model: <https://docs.openvinotoolkit.org/latest/_docs_MO_DG_prepare_model_convert_model_tf_specific_Convert_YOLO_From_Tensorflow.html>

python3 /opt/intel/openvino/deployment\_tools/model\_optimizer/mo\_tf.py \

--input\_shape [1,416,416,3] \

--input\_model ./1/frozen\_darknet\_yolov3\_model.pb \

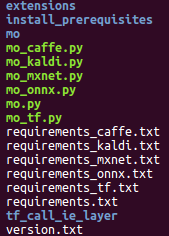
--tensorflow\_use\_custom\_operations\_config /opt/intel/openvino/deployment\_tools/model\_optimizer/extensions/front/tf/yolo\_v3.json \

--batch 1 \

--data\_type FP16 \

--output\_dir ./openvino \

Mặc định, OpenVINO cung cấp sẵn 1 số các file \*.py cho từng framework khác nhau.



Có thể tham khảo thêm qua các tài liệu đọc thêm về các cách export sang IR format khác nhau tại link sau:

* + <https://docs.openvinotoolkit.org/latest/_docs_MO_DG_prepare_model_convert_model_Converting_Model.html>
  + <https://docs.openvinotoolkit.org/latest/_docs_MO_DG_prepare_model_convert_model_Convert_Model_From_TensorFlow.html>

Hiện tại thì OpenVINO support khá nhiều các model và layer phổ biến, tuy nhiên 1 số model được viết thêm các custom layer kèm theo, các bạn có thể đọc thêm phần hướng dẫn tại 1 số link sau:

* + <https://docs.openvinotoolkit.org/latest/_docs_MO_DG_prepare_model_customize_model_optimizer_Customize_Model_Optimizer.html>
  + <https://github.com/david-drew/OpenVINO-Custom-Layers>

**Inference mode**

* Code mẫu thực hiện inference từ IR format

from openvino import inference\_engine as ie

from openvino.inference\_engine import IENetwork, IEPlugin

plugin\_dir = None

model\_xml = './model/frozen\_model.xml'

model\_bin = './model/frozen\_model.bin'

# Devices: GPU (intel), CPU, MYRIAD

plugin = IEPlugin("CPU", plugin\_dirs=plugin\_dir)

plugin.add\_cpu\_extension(

"/opt/intel/openvino/deployment\_tools/inference\_engine/lib/intel64/libcpu\_extension\_avx2.so",

)

# Read IR

net = IENetwork.from\_ir(model=model\_xml, weights=model\_bin)

input\_blob = next(iter(net.inputs))

out\_blob = next(iter(net.outputs))

# Load network to the plugin

exec\_net = plugin.load(network=net)

del net

# Run inference

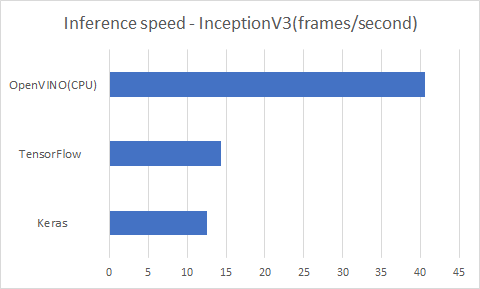
img\_fp = 'example.jpg'

image, processed\_img, image\_path = pre\_process\_image(img\_fp)

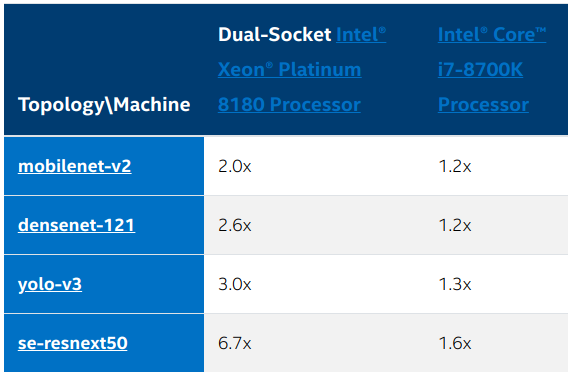
res = exec\_net.infer(inputs={input\_blob: processed\_img})

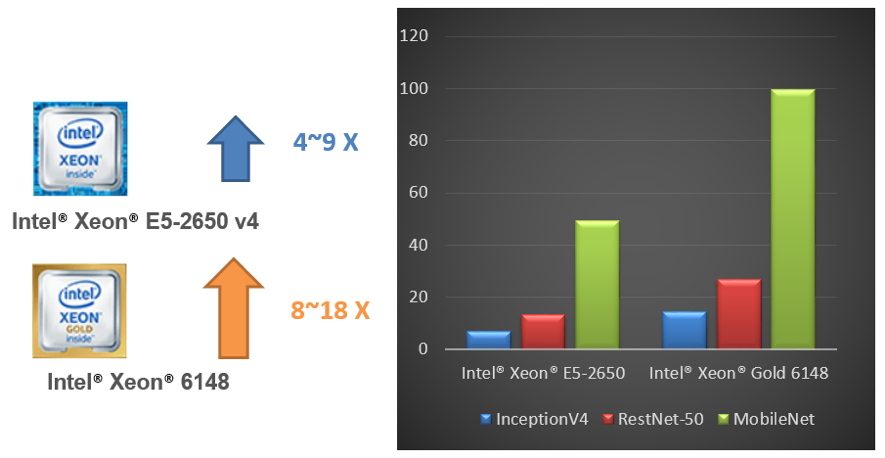
**1.2.2: Đánh giá hiệu năng Benchmarks**

Dưới đây là 1 số benchmarks với các định dạng, thiết bị khác nhau:

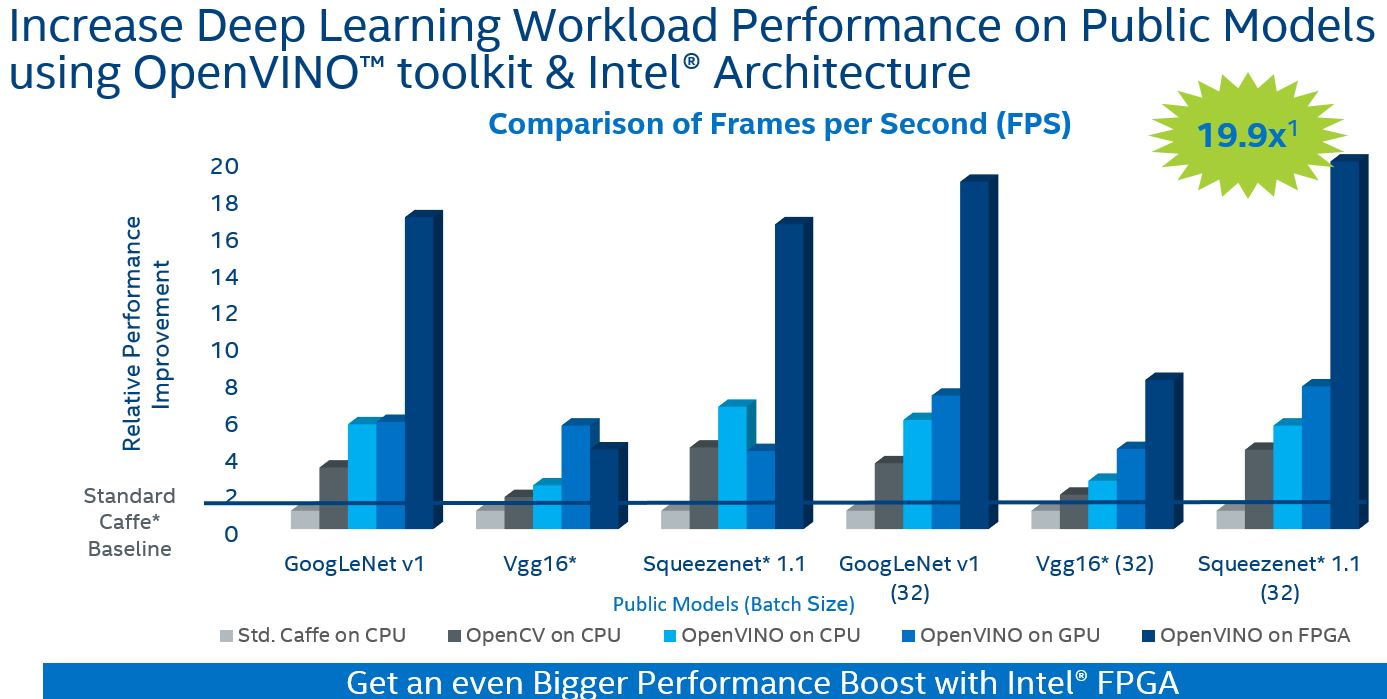


Số frame/second inference khi thực hiện test với các định dạng khác nhau trên InceptionV3 model: .h5 (keras), frozen .pb (tensorflow), .bin (IR-OpenVINO)



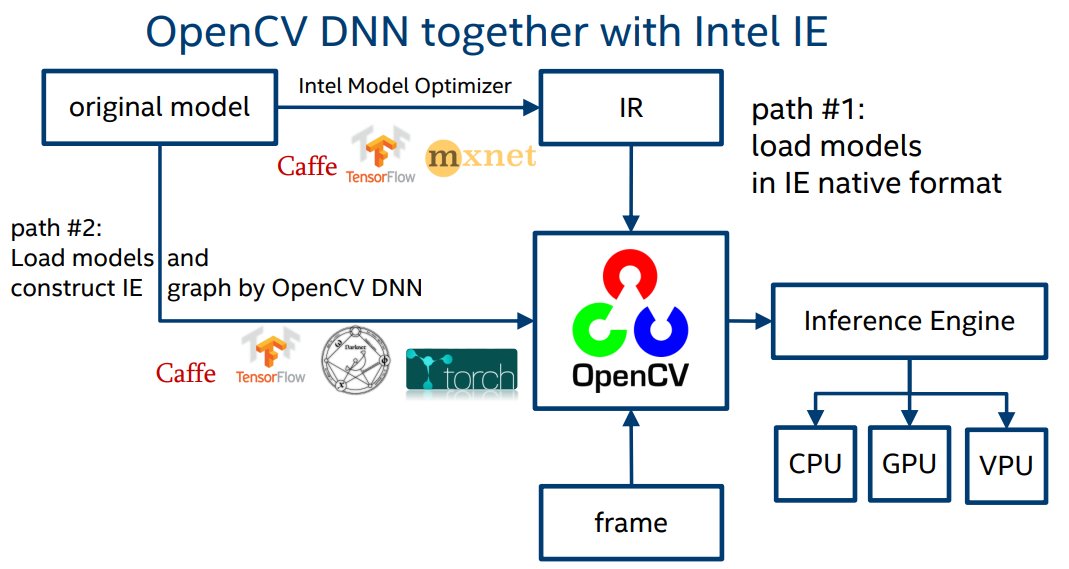


Tốc độ xử lý trên các con chip khác nhau của OpenVINO

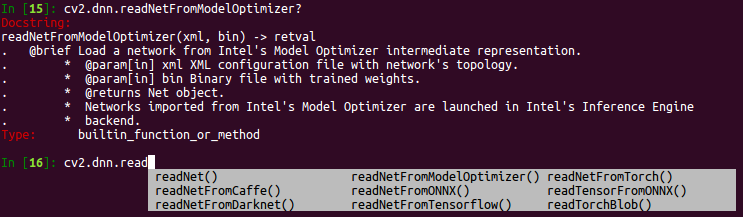


FPS khi thực hiện inference trên 1 số Model phổ biến và các thiết bị khác nhau: CPU, GPU, FPGA

**1.2.3: OpenVINO và OpenCV**



1 điểm đáng chú ý của OpenVINO là các file IR format sau khi đã được optimize thì hoàn toàn có thể đọc bởi OpenCV, 1 thư viện khá phổ biến về xử lý ảnh. Trong OpenVINO cung cấp sẵn method readNetFromModelOptimizer như hình bên dưới, 2 params truyền vào chính là 2 file .xml và .bin do OpenVINO tạo ra. Từ đó, các bạn có thể thực hiện predict như bình thường.



Ngoài ra, OpenCV cũng hỗ trợ đọc 1 số định dạng khác từ 1 vài framework phổ biến: Caffe, Tensorflow, Torch, ONNX, .. Ví dụ với Tensorflow thì có thể truyền params như sau:

net = cv.dnn.readNetFromTensorflow('frozen\_inference\_graph.pb', 'prototxt.pbtxt')

**1.2.4: OpenVINO model server**

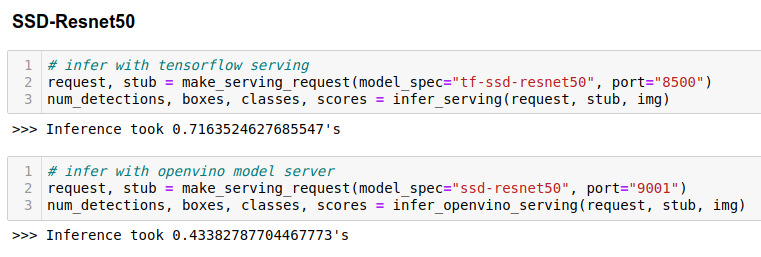
Thông thường, sau khi đã huấn luyện và kiểm thử xong mô hình, mình thường sử dụng tensorflow serving để deploy và serving model 1 cách hiệu quả nhất. 1 vài ưu điểm nổi bật của Tensorflow Serving có thể kể tới như sau:

* + Thuộc TFX (Tensorflow Extended) - có thể coi như là 1 hệ sinh thái end-to-end cho việc deploy các ML pipelines.
  + Auto-reload và cập nhật mới nhất các version của model.
  + Serving cùng lúc nhiều model chỉ với 1 file config duy nhất.
  + Handle được lượng truy cập lớn hơn.
  + Hỗ trợ expose cho 2 kiểu giao tiếp gRPC và RestfulAPI
  + Hỗ trợ nhiều định dạng dữ liệu khác nhau: text, image, embedding, ....
  + Dễ dàng đóng gói và tùy chỉnh tách biệt với phần request tới model

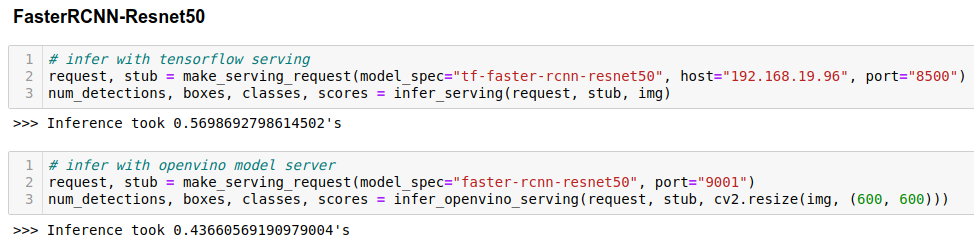
OpenVINO cũng cung cấp 1 OSS cho việc dễ dàng deploy và serving model dưới dạng IR format. Điểm hay của OpenVINO model server là vẫn giữ được các ưu điểm nổi bật của Tensorflow Serving (serving nhiều model với 1 file config duy nhất, support gRPC + RestfulAPI, ...), cùng với đó là inference time của model cũng được cải thiện đáng kể do model đã được convert dưới dạng IR format tối ưu hơn về mặt hiệu năng.

Mình có tiến hành 1 số thử nghiệm với 1 số Model + Backbone thông dụng như các model object detection (ssd / faster-rcnn), các mạng feature extraction phổ biến (mobilenet / resnet) thì OpenVINO model server trong mọi trường hợp đều cho kết quả tốt hơn so với Tensorflow Serving, thời gian inference nhanh hơn khoảng 1.3 --> 1.6 lần với OpenVINO.

* Với SSD-Resnet50



* Với Faster-RCNN-Resnet50



Ngoài ra, OpenVINO còn tích hợp được với 1 số nền tảng thông dụng khác như: Kubernetes, Sagemaker. Chi tiết các bạn có thể xem thêm tại mục Usage Examples: <https://github.com/IntelAI/OpenVINO-model-server#usage-examples>

**1.2.5: Điểm hạn chế**

* 1 vài "hạn chế" của OpenVINO có thể kể tới như:
  + Các custom layer của 1 số model có thể không được support bởi OpenVINO, các bạn có thể thực hiện convert sau như tutorial sau: <https://docs.openvinotoolkit.org/latest/_docs_MO_DG_prepare_model_customize_model_optimizer_Customize_Model_Optimizer.html> nhưng nhìn chung khá khó khăn.
  + Không phải dạng optimized model (FP16, FP32, INT8) nào của OpenVINO cũng support đa nền tảng

# Chương 2 : Phương pháp giảm chiều PCA

## 2.1. Tổng quan về PCA

Principal Components Analysis – PCA  là một [thuật toán](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n" \o "Thuật toán) [thống kê](https://vi.wikipedia.org/wiki/Th%E1%BB%91ng_k%C3%AA" \o "Thống kê) sử dụng phép biến đổi [trực giao](https://vi.wikipedia.org/w/index.php?title=Tr%E1%BB%B1c_giao&action=edit&redlink=1" \o "Trực giao (trang chưa được viết)) để biến đổi một [tập hợp](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_h%E1%BB%A3p" \o "Tập hợp) [dữ liệu](https://vi.wikipedia.org/wiki/D%E1%BB%AF_li%E1%BB%87u" \o "Dữ liệu) từ một [không gian](https://vi.wikipedia.org/wiki/Kh%C3%B4ng_gian" \o "Không gian) nhiều chiều sang một không gian mới ít chiều hơn (2 hoặc [3 chiều](https://vi.wikipedia.org/wiki/Kh%C3%B4ng_gian_ba_chi%E1%BB%81u)) nhằm tối ưu hóa việc thể hiện [sự biến thiên](https://vi.wikipedia.org/w/index.php?title=S%E1%BB%B1_bi%E1%BA%BFn_thi%C3%AAn&action=edit&redlink=1" \o "Sự biến thiên (trang chưa được viết)) của dữ liệu, là phương pháp biến đổi giúp giảm số lượng lớn các biến có tương quan với nhau thành tập ít các biến sao cho các biến mới tạo ra là tổ hợp tuyến tính của những biến cũ không có tương quan lần nhau. Ví dụ, chúng ta có 100 biến ban đầu có tương quan tuyến tính với nhau, khi đó chúng ta sử dụng phương pháp PCA xoay chiều không gian cũ thành chiều không gian mới mà ở đó chỉ còn 5 biến không có tương quan tuyến tính mà vẫn dữ được nhiều nhất lượng thông tin từ nhóm biến ban đầu.

## 2.2. Đặc tính của PCA

Một số đặc tính của PCA được kể đến như:

1. Giúp giảm số chiều dữ liệu - Giúp visualization khi dữ liệu có quá nhiều chiều thông tin.
2. Do dữ liệu ban đầu có số chiều lớn (nhiều biến) thì PCA giúp chúng ta xoay trục tọa độ xây một trục tọa độ mới đảm bảo độ biến thiên của dữ liệu và giữ lại được nhiều thông tin nhất mà không ảnh hưởng tới chất lượng của các mô hình dự báo. (Maximize the variability).
3. Do PCA giúp tạo 1 hệ trục tọa độ mới nên về mặt ý nghĩa toán học, PCA giúp chúng ta xây dựng những biến factor mới là tổ hợp tuyến tính của những biến ban đầu.
4. Trong không gian mới, có thẻ giúp chúng ta khám phá thêm những thông tin quý giá mới khi mà tại chiều thông tin cũ những thông tin quý giá này bị che mất (Điển hình cho ví dụ về chú lạc đà phía trên).

## 2.3. Mô hình PCA

Xét tập không gian (dữ liệu) k biến, k biến này được biểu qua j thành phần chính sao cho (j < k). Xét thành phần chính đầu tiên có dạng:

PC1=a1X1+a2X3+a4X5+...akXkPC1=a1X1+a2X3+a4X5+...akXk

Thành phần chính đầu tiên chứa đựng hầu hết thông tin từ k biến ban đầu (được hình thành là 1 tổ hợp tuyến tính của các biến ban đầu) và lúc này tiếp tục xét thành phần chính thứ 2 được biểu diễn tuyến tính từ k biến ban đầu tuy nhiên thành phần chính thứ 2 phải không trực giao với thành phần chính ban đầu hay (thành phần chính thứ 2 không có mối tương quan tuyến tính với thành phần chính đầu tiên). Về lý thuyết chúng ta có thể xây dựng nhiều thành phần chính từ nhiều biến ban đầu. Tuy nhiên chúng ta cần tìm được trục không gian sao cho ít thành phần nhất mà có thể biểu diễn được hầu hết thông tin từ những biến ban đầu

## 2.4. ****Phương pháp phân tích thành phần chính – Principle Component Analysis (PCA)****

Principle Component Analysis (PCA), đây là một kết quả rất đẹp của đại số mà ngày nay được ứng dụng trong rất nhiều lĩnh vực: Công nghệ thông tin, Sinh học,… và cả trong Tài chính.

Với dữ liệu cần phân tích ban đầu phụ thuộc nhiều biến, vấn đề là các biến này thường có tương quan với nhau sẽ bất lợi cho việc áp dụng các biến này để xây dựng các mô hình tính toán ví dụ: hồi quy… và với số biến giải thích lớn chúng ta sẽ rất khó để có cái nhìn trực quan về dữ liệu ví dụ: thị trường ta quan tâm có hàng ngàn mã cổ phiếu làm cách nào để khi quan sát dữ liệu từ hàng ngàn cổ phiếu này ta hình dung được xu hướng của toàn thị trường…

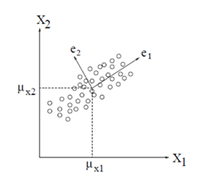
Phương pháp PCA sẽ “chiếu” (biễu diễn) dữ liệu đa chiều lên một không gian có cơ sở trực giao, tức nếu ta xem mỗi cơ sở trong không gian mới là một biến thì hình ảnh của dữ liệu gốc trong không gian mới này sẽ được biểu diễn thông qua các biến độc lập (tuyến tính). Vấn đề: nếu chuyển dữ liệu ban đầu sang không gian mới thì những thông tin đáng quan tâm của dữ liệu ban đầu liệu có bị mất? Để giải quyết vấn đề này phương pháp PCA sẽ tìm không gian mới với tiêu chí cố gắng phản ánh được càng nhiều thông tin gốc càng tốt, và thước đo cho khái niệm “thông tin” ở đây là phương sai. Một điểm hay nữa là: do các biến trong không gian mới độc lâp, nên ta có thể tính toán được tỷ lệ giải thích phương sai của từng biến mới đối với dữ liệu, điều này cho phép ta cân nhắc việc chỉ dùng số ít các biến để giải thích dữ liệu.

Các ứng dụng tự nhiên mà ta có thể nhận ra là:

* Giảm kích thước của dữ liệu.
* Nếu ta có thể giảm số chiều về 2 hoặc 3 chiều, ta có thể dùng các loại đồ thị để hiểu thêm về dữ liệu mà mình đang có, giúp ta nhìn dữ liệu trực quan hơn.
* Sử lý vấn đề tương quan giữa các biến trong dữ liệu ban đầu bằng cách sử dụng các biến mới trong không gian mà phương pháp PCA tìm được để mô tả dữ liệu.

\* Chúng ta sẽ bắt  đầu bằng một ví dụ mang tính chất minh họa trực quan:

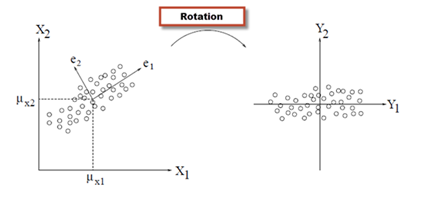
Chúng ta có 2 biến X1 và X2 có tương quan (tuyến tính), được biểu diển bằng đồ thị sau:



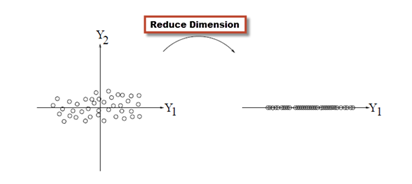
Hình 2.1: Biểu diễn dữ liệu trên trục tọa độ

Ta biết rằng khi thực hiện các phân tích đa biến mà trong đó các biến có tương quan với nhau là rất khó chịu!

Ta sẽ loại bỏ sự tương quan này bằng cách xoay trục (cơ sở)



Hình 2.2: Lựa chọn trục tọa độ mới để biểu diễn dữ liệu



Hình 2.3: Dữ liệu sau khi thực hiện giảm chiều

Ta thấy rằng dữ liệu trên trục mới đã giảm sự tương quan đáng kể (biến Y1 và Y2 gần như không tương quan), và sự thay đổi của dữ liệu phụ thuộc phần lớn vào biến Y1, ta có thể chỉ dung một biến Y1 để biểu diễn dữ liệu, điều này giúp ta giảm số chiều dữ liệu mà không làm giàm quá nhiều “phương sai” của dữ liệu. Đây cũng chính là tư tưởng của phương pháp PCA. Sau đây chúng ta sẽ làm rõ hơn về mặt lý thuyết của phương pháp này

\* Nền tảng toán học cho phương pháp PCA : Trong phần này chúng tôi sẽ giới thiệu cơ sở toán học của PCA một cách ngắn gọn và vắn tắt.

Cho biến ngẫu nhiên X có p chiều, X=(x1,…,xp) mô tả dữ liệu ban đầu và ta có ma trận hiệp phương sai là:

S=(sij)1≤i≤p, 1≤j≤p

\* Ý tưởng chính :

Ta tìm các biến ngẫu nhiên mới y có dạng là tổ hợp tuyến tính của các thành phần của X sao cho y có phương sai càng lớn càng tốt (chứa nhiều thông tin), về mặt hình thức ta viết lại vấn đề trên: Tìm y có dạng y=∑pi=1aixi sao cho Var(y) lớn nhất có thể. Không mất tính tổng quát và để đơn giản tính toán, ta giả sử kỳ vọng của các thành phần trong X là 0 và phương sai bằng 1, lúc đó ta có:

Var(y)=∑pi=1∑pj=1aiajsij=aTSa

với a=(a1,..,ap), và aTa=1. Như vậy, ta cần giải bài toán tối ưu với điều kiện biên:

max aTSa với điều kiện biên aTa=1

bằng phương pháp nhân tử Largrange ta chuyển bài toán này thành:

max aTSa−λ(1−aTa)

nên ta có a là nghiệm của Sa=λa, dễ thấy: a chính là vecto riêng, còn λ chính là trị riêng tương ứng của ma trận S.

Chú ý: S là ma trận đối xứng nửa xác định dương (symmetric positive semidefinite), và do đó có các trị riêng không âm và các vecto riêng lập thành họ trực giao.

Giả sử S có p trị riêng (λi)1≤i≤p và p vecto riêng (a(i))1≤i≤p tương ứng, do điều kiện aTa=1 nên ta thấy (a(i))1≤i≤p là họ cơ sở trực chuẩn trong không gian mới.

Ta thấy phương sai của hình chiếu của dữ liệu ban đầu lên cơ sở thứ i  (a(i)) chính là trị riêng thứ i (λi), như vậy ta xác định các thành phần chính dựa theo độ lớn của giá trị riệng. Ví dụ: thành phần quan trọng nhất, là hình chiếu của dữ liệu lên vecto riêng ứng với trị riêng lớn nhất…

Ngoài ra ta có: ∑pi=1λi=∑pi=1sii là tổng phương sai, người ta hay dùng λi∑pi=1λi để thể hiện mức độ phản ánh dữ liệu gốc theo thành phần thứ i, ví dụ: ta có tổng phương sai là 2 (∑pi=1λi), trị riêng của thành phần thứ 1 là 0.4, có thể hiểu là thành phần thứ 1 thể hiện được 0.4/2\*100 phần trăm phương sai của dữ liệu, trị riêng của thành phần thứ 2 là 0.3, có thể hiểu là thành phần thứ 2 thể hiện được 0.3/2\*100 phần trăm phương sai của dữ liệu, có thể nhận định thành phần thứ 1 quan trọng hơn thành phần thứ 2.

## 2.5. Giải pháp nhận diện khuôn mặt

### 2.5.1. Tổng quan về nhận diện khuôn mặt

Công nghệ nhân diện khuôn mặt (Facial Recognition Technology) hiện là một công nghệ đang được sử dụng khá phổ biến tại các quốc gia phát triển. Công nghệ này có khả năng xác định hoặc xác nhận một người từ hình ảnh kỹ thuật số được lấy mẫu trước đó hoặc từ một khung hình trong một nguồn video khác, nguồn này có thể là trực tiếp hoặc gián tiếp.

Công nghệ nhân diện khuôn mặt là công nghệ sinh trắc học ít tác động đến người dùng nhất và là công nghệ sinh trắc học nhanh nhất. Nhà quản lý có thể chủ động hơn trong việc kiểm soát, hệ thống sẽ nhận diện khuôn mặt một cách kín đáo bằng cách chụp ảnh khuôn mặt của những ai bước vào một khu vực được xác định từ camera giám sát. Công nghệ này, không cần sự tương tác (của người muốn vào khu vực) và không có thời gian trễ. Trong nhiều trường hợp đối tượng hoàn toàn không hay biết gì về quá trình này, vì vậy họ sẽ không cảm thấy mình đang “bị giám sát” hoặc sự riêng tư của họ bị xâm phạm.

### 2.5.2. Lĩnh vực ứng dụng

* Áp dụng tại khu vực lễ tân của tòa nhà, bệnh viện, khách sạn với mục đích phát hiện sớm và phân biệt khách vip, nhân viên, khách vãng lai…
* Áp dụng tại các khu vực kiểm soát vào ra của tòa nhà khu chung cư phân biệt giữa cư dân của tòa nhà và khách đến thăm…

[](https://homesecurity.com.vn/chung-cu/)[](https://homesecurity.com.vn/truong-hoc/)

Hình 2.4: Ứng dụng tại Chung cư, trường học

[](https://homesecurity.com.vn/van-phong/)[](https://homesecurity.com.vn/an-ninh/)

Hình 2.5: Ứng dụng tại Văn phòng, an ninh.

### 2.5.3. Phương thức hoạt động

Quá trình mà sinh trắc học nhận diện khuôn mặt hoạt động bao gồm:

* Lấy mẫu: Bằng các camera giám sát hay một thiết bị có cùng chức năng.
* Phân tích: Các dữ liệu (được nhắc tới ở trên) sẽ được trích xuất từ các hình ảnh, các mẫu.
* So sánh: Các dữ liệu vừa thu được sẽ được so sánh với các dữ liệu trong cơ sở dữ liệu.
* Kết quả: Hệ thống sau đó sẽ quyết định kết quả so sánh có phù hợp hay không.



Hình 2.6: Nhận diện khuôn mặt

Khi đối tượng lọt vào tầm quét của camera, hệ thống sẽ tự động xác định vị trí khuôn mặt và thực hiện các thuật toán để trích xuất các dữ liệu trên khuôn mặt. Trong quá trình di chuyển, đối tượng sẽ giúp hệ thống dễ dàng xác nhận khuôn mặt hơn. Tất cả các quá trình trên diễn ra trong vòng chưa đầy 2 giây.

# Chương 3 : Thực nghiệm

## 3.1. ****Ý tưởng bài toán.****

Mục tiêu của phương pháp PCA là *“giảm số chiều”* của 1 tập vector sao cho vẫn đảm bảo được *“tối đa thông tin quan trọng nhất”* . Tức Feature extraction (giữ k thuộc tính “mới”) chứ không phải Feature selection (giữ lại k thuộc tính nguyên gốc ban đầu).



Không gian N chiều với hệ cơ sở *v1, v2,…,vn*

Không gian K chiều (K<<N) với hệ cơ sở *u1, u2,…,un*

Ánh xạ tuyến tính T cần tìm (phép chiếu)

Vector x ban đầu có N chiều

Vector ***y*** chỉ còn ***K***



Hình 3.1:Mô hình Thuật toán PCA

Tất nhiên, khi loại bỏ 1 số thành phần của x để thu được y thì sẽ gây ra sai số (lỗi) !!

🡪 Phương pháp PCA sẽ cố gắng tìm phép biến đổi tuyến tính *T* thỏa:

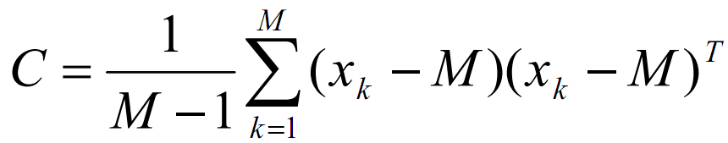
*y = T.x*

sao cho trung bình bình phương lỗi (MSE) là bé nhất.

Cách để tìm được T:

Gọi M là vector trung bình của tất cả các vector x trong tập mẫu.

Gọi ma trận hiệp phương sai của các phần tử x trong tập mẫu là C. C được tính theo công thức sau:



Người ta chứng minh được rằng:

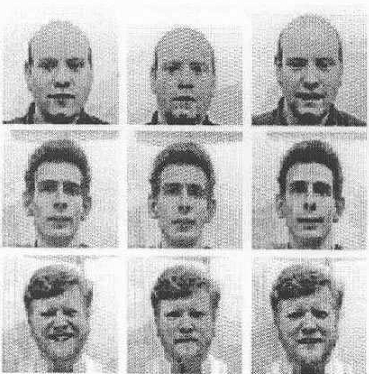
“Nếu T là ma trận *m* hàng, mỗi hàng là 1 vector riêng của C, đồng thời *m* vector riêng này phải ứng với *m* trị riêng lớn nhất. Khi đó T chính là phép biến đổi thỏa MSE nhỏ nhất”.

🡪 Nói tóm lại, phương pháp PCA quy về việc đi tìm trị riêng (eigenvalues) và vector riêng (eigenvectors) của ma trận hiệp phương sai C của tập mẫu X. Sau đó, ta chỉ giữ lại K vector riêng ứng với K trị riêng lớn nhất để làm cơ sở cho không gian mới này.

## 3.2. Tính toán các Eigenfaces

**Bước 1:**

Sử dụng các ảnh khuôn mặt *I1, I2, … In*(tập các khuôn mặt huấn luyện) với khuôn mặt phải *chính diện* & tất cả ảnh phải *cùng kích thước.*

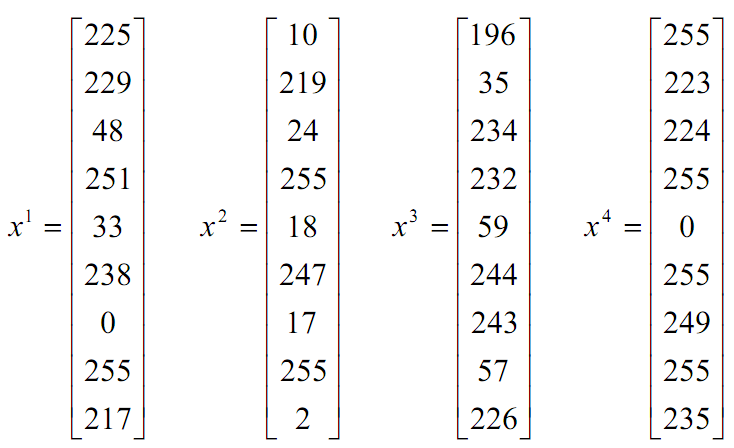


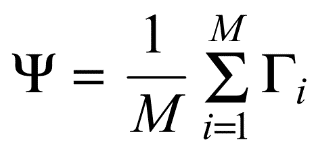
Hình 3.2: Dữ liệu khuôn mặt

**Bước 2:**

Biểu diễn mọi ảnh Ii thành vector

*Ví dụ:* Để đơn giản ta giả sử chỉ có 4 *ảnh* trong tập huấn luyện (kích thước *3x3*). Ta tính toán được:

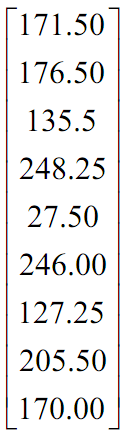
****

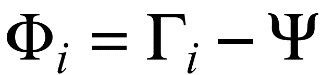
**Bước 3:**

Tính vector khuôn mặt trung bình  theo công thức:

Cụ thể ta có:

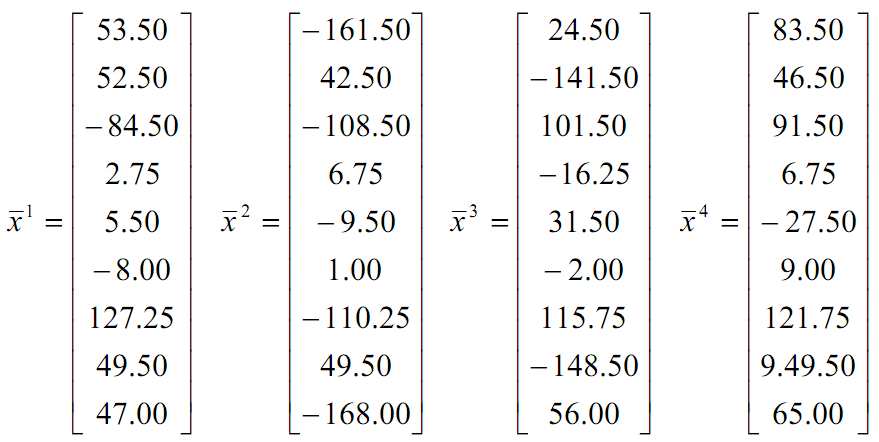
Ψ=



**Bước 4:**

Trừ vector khuôn mặt trung bình:

Cụ thể ta có:



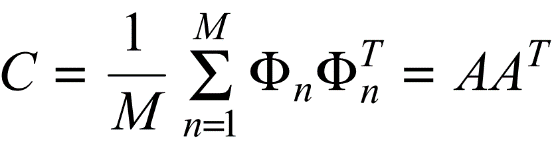
Φ4

Φ3

Φ1

Φ2

Bước 5:

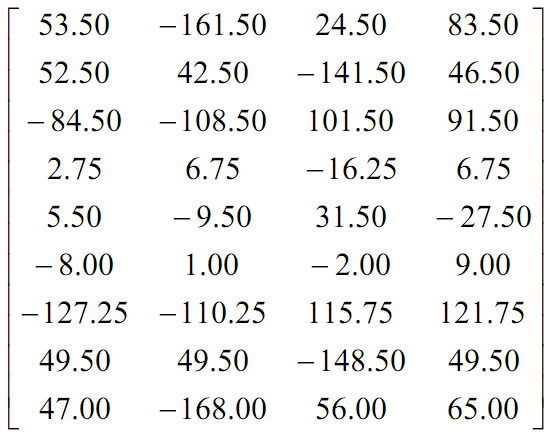
Tính ma trận hiệp phương sai (covariance) C: 

C sẽ có kích thước *N2xN2*

Trong đó:  A sẽ có kích thước là N2xM

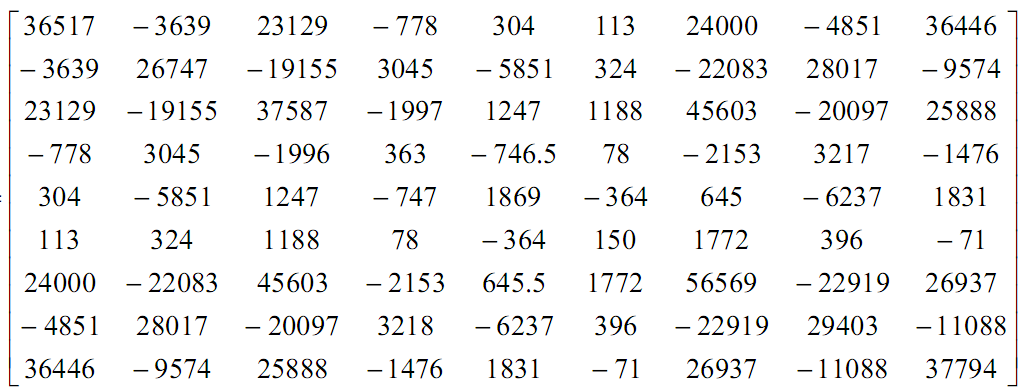
Cụ thể ta có:

Ma trận A kích thước N2xM (9x4) tạo bởi ghép các làm các cột



A=

Từ đó ta dễ dàng tính được ma trận hiệp phương sai C, kết quả như sau:



C=

Ma trận C kích thước *N2x N2 (9x9)***.** Quá lớn !!

**Bước 6:**

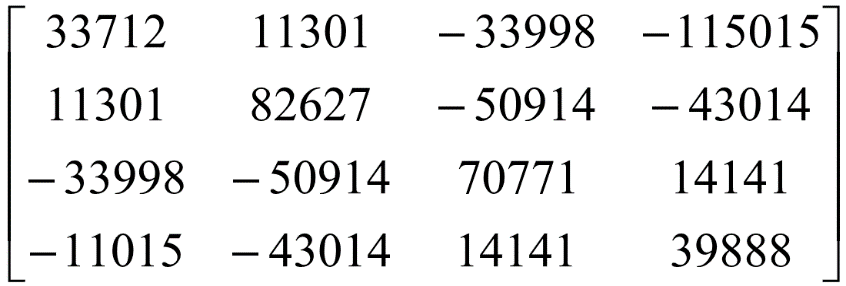
Tính các Eigenvector *ui*(*“vector riêng”*) của ma trận vuông *A.AT* (C có kích thước *N2xN2*).

Ma trận này có kích thước quá lớn 🡪 không khả thi 🡪 phải khôn khéo đi đường vòng !!

**Bước 6.1**:

Xét ma trận *AT.A* (chú ý ma trận này chỉ có kích thước là *MxM*)

Cụ thể ta có:



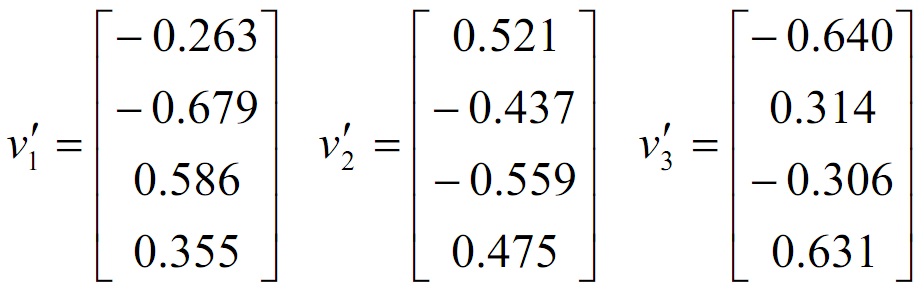
AT.A =

Kích thước *MxM (4x4)***.** Giảm hẳn !!

**Bước 6.2:**

Tính các vector riêng *vi* (eigenvectors) của ma trận vuông *AT.A* này.

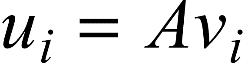
Ở đây ta sẽ tìm được 4 trị riêng của ma trận *AT.A,* tuy nhiên ta sẽ sắp xếp lại theo thứ tự giảm dần, và chỉ lấy các trị riêng “non-zero”. Kết quả ta được 3 trị riêng (từ đó tính ra 3 vector riêng tương ứng):

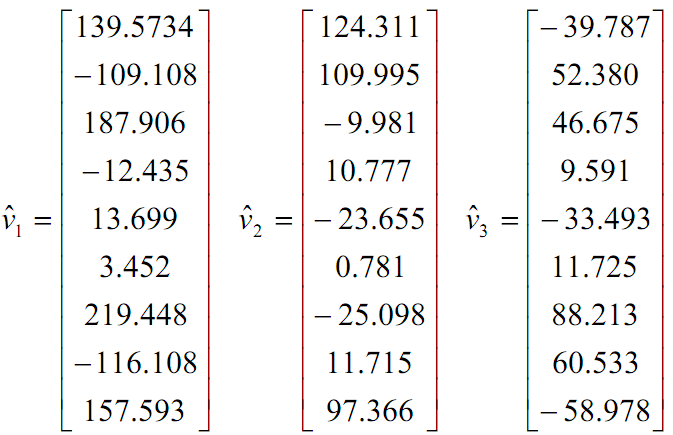


Các eigenvector của *AT.A* tương ứng với các eigenvalues. Kích thước *Mx1*

Các eigenvalues của *AT.A.*

Sau khi đã tính được các vector vi (có kích thước là *Mx1*), ta sẽ dễ dàng suy ra được các vector riêng *ui (*kích thước *N2x1)* mong muốn cần tìm, theo công thức:

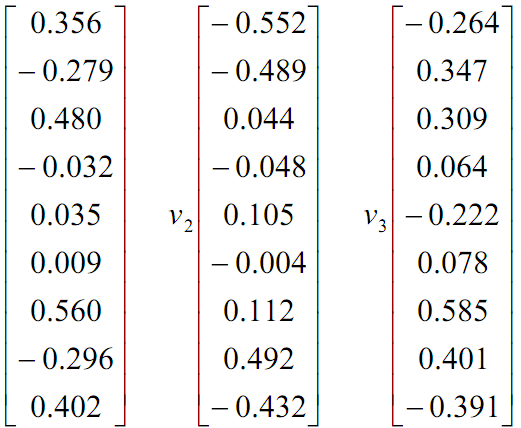




Các eigenvector của ma trận hiệp phương sai C (*AT.A)* cần phải tìm. Kích thước *N2x1*

Chú ý nên chuẩn hóa các vector *ui* (), nghĩa là: 

Sau khi chuẩn hóa ta thu được các vector *ui* cuối cùng như sau:



Đây chính là các vector cơ sở của không gian mới. Ta gọi các vector này là các ***EIGENFACES***

Ta rút ra được các chú ý sau (theo TOÁN HỌC):

* Ma trận A.AT có thể có đến N2 trị riêng (mỗi trị riêng sẽ ứng với vô số vector riêng – nó được gọi là “không gian riêng ứng với trị riêng nào đó”).
* Ma trận AT.A có thể có đến M trị riêng.
* M trị riêng của ma trận AT.A (kèm với các vector riêng tương ứng), sẽ ứng với M trị riêng lớn nhất của A.AT (cái này quan trọng bởi ta đang đi tìm các hướng biến thiên quan trọng nhất).

**Bước 6.3:**

Tính *M* vector riêng *ui* tốt nhất của *A.AT* theo công thức (\*)

**Bước 7:**

Chỉ giữ lại *K* vector riêng trong số M vector nói trên (ứng với *K* trị riêng *lớn nhất*), tất nhiên *K<<N2*.

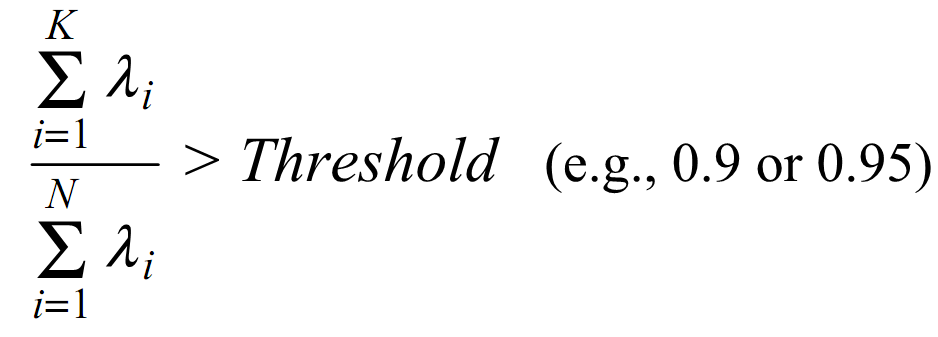
Có 2 cách để xác định *K*.

Cách 1:

* Sắp xếp theo thứ tự dãy *giảm dần* các eigenvalues tìm được.
* Theo dõi sự biến thiên của dãy trên, khi không còn biến thiên (hoặc xấp xỉ bằng không) thì lúc đó ta đã chọn đủ *K*.

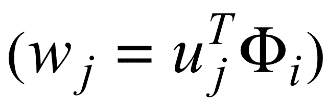
Cách 2:

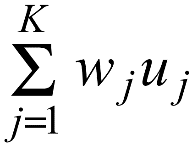
Chọn K theo công thức sau:



N(Alpaydin)

## 3.3. Biểu diễn các khuôn mặt có sẵn (tập huấn luyện) vào trong không gian vector này

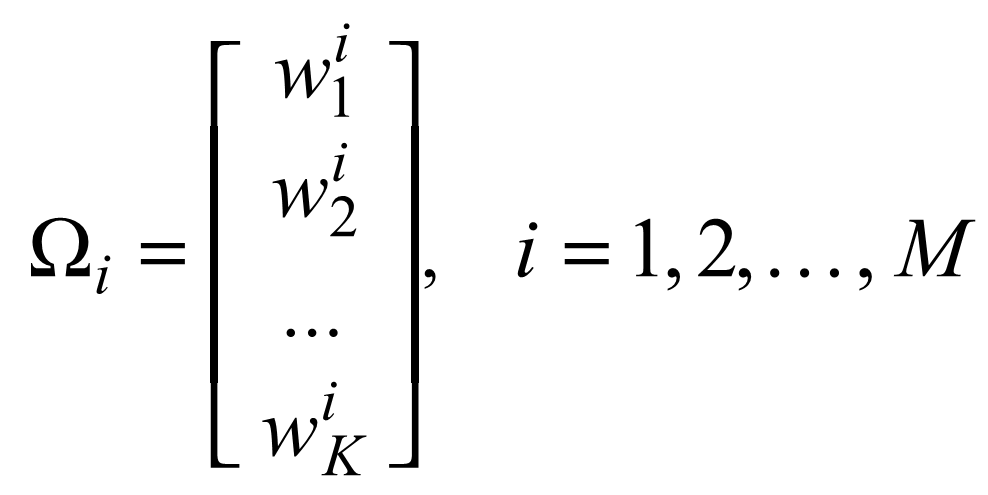
Mỗi khuôn mặt trong tập huấn luyện có thể được biểu diễn lại là 1 tổ hợp tuyến tính của K vector riêng lớn nhất:

Trong đó:

Ta thống nhất gọi các ***“vector riêng ui”*** là các **“EIGENFACES”** (khuôn mặt riêng).Như vậy, bây giờ, mỗi khuôn mặt huấn luyệnsẽ được biểu diễn trong không gian mới là 1 vector như sau:

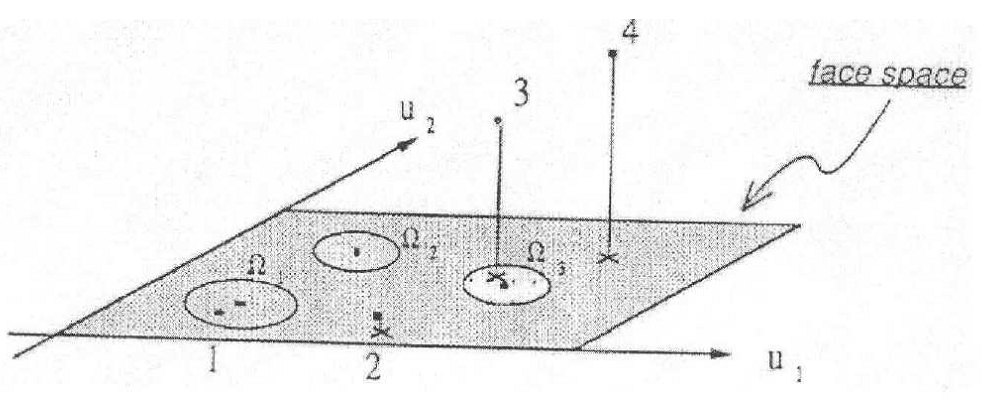


Kích thước vector này chỉ còn là Kx1



Thật ra, đây chính là TỌA ĐỘ của vector

Xem hình sau để hiểu rõ hơn những gì mình đang làm:



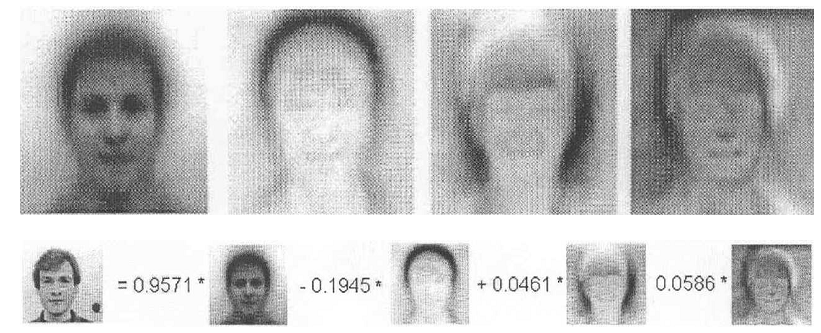
Đây là các vector (ảnh khuôn mặt “gốc” ban đầu)

Chiếu cácvào trong *“không gian khuôn mặt”* ta được các vector . Tọa độ của trong không gian này là vector

Đây là các vector cơ sở của không gian khuôn mặt (được gọi là *“EIGENFACE”)*

Hình 3.3: Mô hình Eigenfaces

Hình này “tả thực” hơn. Thật ra nó không thật sát với tổ chức dữ liệu, chỉ là mô tả cho dễ hình dung. *Ví dụ: Eigenface có kthước N2x1, hình của , hình eigenfaces….*



Các Eigenface (vector cơ sở ui của không gian khuôn mặt)

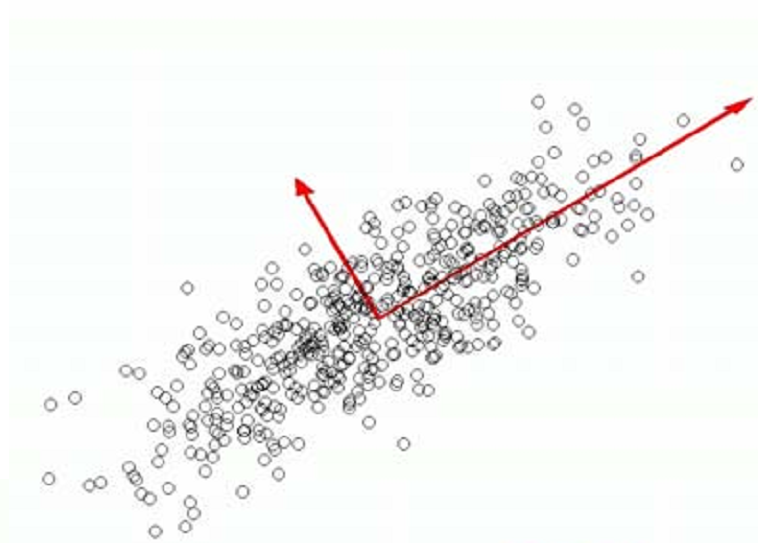
Khuôn mặt có thể biểu diễn là tổ hợp tuyến tính của các vector cơ sở

Các ***wj*** của vector **Ωi**(đây chính là tọa độ trong không gian khuôn mặt )

Hình 3.4: Biểu diễn Eigenfaces

Diễn giải hình học của phương pháp PCA (tìm các eigenvalue/vectors):

* + PCA chiếu dữ liệu theo hướng mà ở đó dữ liệu khác nhau nhiều nhất.
  + Các hướng này được xác định bằng các eigenvectors của ma trận hiệp phương sai *(covariance matrix)*.



Mặt Ellipsoid định nghĩa bởi covariance matrix (các sample vector tập trung chủ yếu ở trong này)

Trục dài nhất ám chỉ hướng biến thiên quan trọng nhất của dữ liệu (có variance – *“phương sai”* - cao nhất). Đây chính là hướng của Eigenvector ứng với eigenvalues lớn nhất.

Gốc tọa độ chính là Ψ

Hình 3.5: Diễn giải hình học phương pháp PCA

Ví dụ dễ hiểu về khả năng tại sao PCA lại có thể *“giảm được chiều dữ liệu”*:

x1

x2

y1

Biến đổi (chiếu)

Vector ***2 chiều***, trong không gian với các cơ sở là x1, x2

Vector chỉ còn ***1 chiều***, trong không gian với cơ sở là y1

Hình 3.6: Ví dụ giảm chiều dữ liệu

## 3.4. Nhận dạng khuôn mặt bằng các EIGENFACES

Cho 1 ảnh khuôn mặt chưa biết là ai T tất nhiên cũng phải giống tập mẫu – *chính diện & cùng kích thước*). Chú ý, giai đoạn nhận dạng này giống hệt giai đoạn biểu diễn ảnh khuôn mặt trong tập mẫu. Ta thực hiện lần lượt các bước sau:

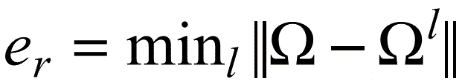
**Bước 1:**

Chuẩn hóa 

**Bước 2:**

Biểu diễn  thành như sau:



**Bước 3:**

Tìm:

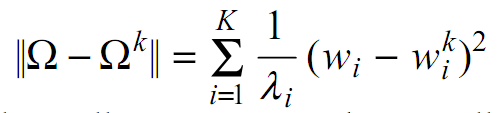
Tức ta tìm khuôn mặt thứ *l* trong tập mẫu có khoảng cách gần nhất với khuôn mặt cần nhận dạng.

**Bước 4:**

Nếu er < Tr (Tr là 1 ngưỡng chấp nhận được nào đó):

Tức ảnh khuôn mặt cần xác định “đủ gần” với ảnh của người thứ *l* trong tập mẫu. Khi đó, ta kết luận đó chính là khuôn mặt của người thứ *l*.

Chú ý:Ta có thể sử dụng khoảng cách Euclid để tính er. Tuy nhiên, người ta chứng mình được rằng, kết quả sẽ tốt hơn nếu dùng khoảng cách *“Mahalanobis”*:



Nhược điểm của Eigenfaces

Trong những trường hợp sau, PCA Eigenfaces sẽ nhận dạng sai:

* Khác nhau về điều kiện ánh sáng



* Khác nhau về điệu bộ (nghiêng đầu chẳng hạn…)
* Cảm xúc (cười to, há miệng…)

## 3.5. Kết quả thực nghiệm

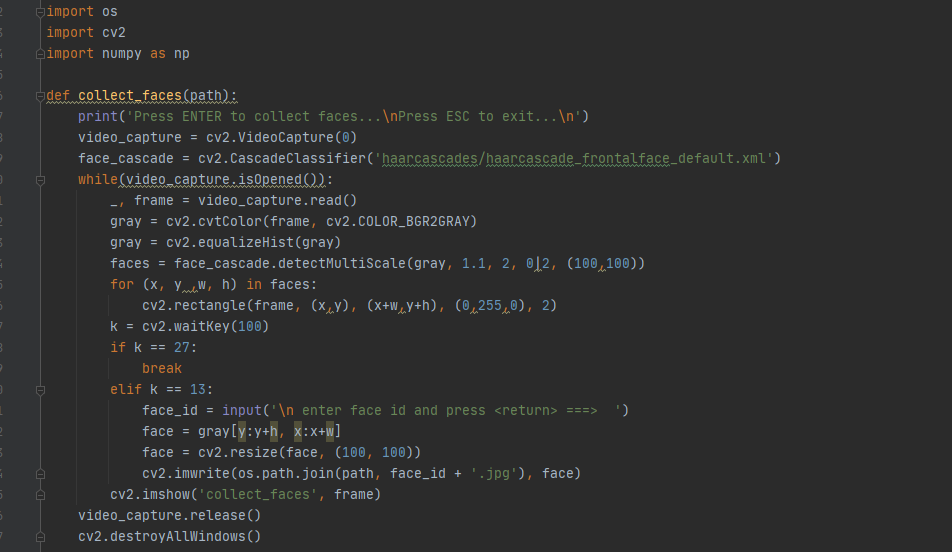
### 3.5.1. Tập dữ liệu huấn luyện

Tập dữ liệu huấn luyện bao gồm 100 bức ảnh khác nhau của 100 người khác nhau, dữ liệu được lấy từ Internet và do người dùng tự thêm vào

Hình 3.7: Hình ảnh tập dữ liệu huấn luyện

### 3.5.2. Kết quả thực nghiệm

Chương trình thêm khuôn mặt vào bộ dữ liệu



Chương trình nhận diện khuôn mặt

Chạy chương trình



Kết luận

Demo nhận dạng với phương pháp trình bày trong bài toán này đã đạt được một số kết quả sau:

* Nắm được các vấn đề cơ bản của ảnh số và xử lý ảnh số
* Sử dụng tương đối tốt thư viện Opencv cùng với tập dữ liệu
* Nắm được các đặc điểm của một bài toán nhận dạng và lọc ảnh nói chung
* Tìm hiểu được một số phương pháp nhận dạng, nắm được ưu nhược điểm của từng phương pháp.
* Demo và Test thử thành công trên tập dữ liệu

Dù cố gắng hết sức tối ưu hóa, song do khả năng còn có hạn, chúng em không thể tránh khỏi những hạn chế và thiếu sót khi thực hiện đề tài này. Chúng em kính mong nhận được ý kiến đóng góp của các thầy cô để chúng em có thể rút ra nhiều kinh nghiệm quý giá hơn.

Một lần nữa chúng em xin chân thành cảm ơn các thầy, cô giáo, đặc biệt là thầy Phạm Đức Hồng – giảng viên bộ môn Học máy nâng cao đã tận tình hướng dẫn, tạo điều kiện tối đa để chúng em hoàn thành đề tài này.

Tài liệu tham khảo

1. Koheri Arai and Ali Ridho Barakbah, Heirarchical K-means: An algorithm for Centroid initialization for K-means, Saga University, (2007)
2. C.Bishop, Pattern Recognition and Machine Learning (<https://goo.gl/pjgqRr>), Springer, 2006 [Bis06].
3. Sunil Arya and David M Mount. Algorithms for fast vector quan-tization. In Data Compression Conference, pages 381–390. IEEE, 1993.
4. Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin.Learning from data, volume 4. AMLBook New York, NY, USA, 2012.
5. Artem Babenko and Victor Lempitsky. Additive quantization for extreme vector compression. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition, pages 931–938, 2014.
6. David M Blei. Hierarchical clustering. 2008.
7. Alan Jose, S. Ravi and M. Sambath, Brain Tumor Segmentation using K-means Clustering and Fuzzy C-means Algorithm and its Area Calculation. In International Journal of Innovative Research in Computer and Communication Engineering, vol. 2, issue 2, March (2014).