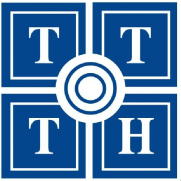
**PROGRAM & NETWORK DEPARTMENT**

**COMPUTER CENTER**

**UNIVERSITY OF NATURAL SCIENCE**

**⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯**

****

**PROJECT REPORT**

**Machine Learning Specialist**

**(Applied Machine Learning Certificate)**

**REALTIME FACE DETECTION**

Instructor: Nguyen Quan Liem

Performed by: Nguyen Xuan Minh Khoi - 0932649647

***City. Ho Chi Minh, date ... month ... year ...***

***TEACHER'S COMMENTS***

***⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯***

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**………………………………………………………………………………………………. ....**

**Day …. month…. year …**

**Instructors**

*(Sign and write full name)*

***PROJECT INTRODUCTION***

⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯

In 2018, Apple began to integrate facial recognition function in the iPhone X product line. After that, a trend of smart phones with facial recognition applications was born. Some banks in Australia have started using facial authentication in ATM transactions.

In China, facial recognition systems have been rolled out across the country to help score citizens and at the same time verify many fugitives.

However, it is quite strange that many places have not yet applied facial recognition to their offices. Building a facial recognition system is not difficult.

The project aims to build real face recognition systems in schools and agencies by CCTV for timekeeping and security monitoring.

Table of Contents

[**I. Overview** 5](#_Toc51207328)

[**1. Introduction** 5](#_Toc51207329)

[**2. Situation and Solutions** 5](#_Toc51207330)

[**II. Data Preparation and Modeling** 6](#_Toc51207331)

[**1. Selection of Data and Environment for Training** 6](#_Toc51207332)

[**2. Multi-Task Cascaded Convolutional Neural Network (MTCNN)** 7](#_Toc51207333)

[**2.1 Concepts** 7](#_Toc51207334)

[**2.2 Face Positioning Using MTCNN** 8](#_Toc51207335)

[**3. Siamese Networks** 9](#_Toc51207336)

[**3.1 Concepts** 9](#_Toc51207337)

[**3.2 Facenet Algorithm** ten](#_Toc51207338)

[**3.3 Triplet Loss** 11](#_Toc51207339)

[**3.4 Pre-train Model Facenet** 11](#_Toc51207340)

[**III. Model Deployment** 14](#_Toc51207341)

[**IV. Conclude** 16](#_Toc51207342)

[**V. Suggestions for Improvement** 16](#_Toc51207343)

[**REFERENCES** 16](#_Toc51207344)

# **I. Overview**

## **1. Introduction**

* **Project introduction:**

In 2018, Apple began to integrate facial recognition function in the iPhone X product line. After that, a trend of smart phones with facial recognition applications was born.

Some banks in Australia have started using facial authentication in ATM transactions.

In China, a nationwide facial recognition system has been deployed to help score citizens and at the same time verify many criminals on the run.

At my office, the facial recognition system is applied in-house to timekeeping employees.

I think facial recognition is highly applicable. Many companies, businesses and countries are in need.

However, it is quite strange that many places have not yet applied facial recognition to their offices. Building a facial recognition system is not difficult.

* **Project scope:**

In this project, we will focus on face recognition and identification of the person being recognized using MTCNN and Siamese Network.

## **2. Situation and Solutions**

* **Reality:**

Currently, the model commonly used for face recognition is Haar Cascade. However, Haar Cascade has a disadvantage that it is difficult to recognize faces that are too close to the frame of the image and only accurately detect faces in the front.

A popular supervised learning algorithm that is often used to learn about a person's face image when the data is too small is One-shot learning. This is an algorithm where each person only needs a few or even a single photo.

Since the input is a picture of a person, we use a simple CNN algorithm architecture to predict who that person is.

However, the disadvantage of this method is that we have to retrain the algorithm often when a new person appears because the shape of the output changes by 1. Obviously not good for face recognition systems. of a company because the number of people fluctuates over time.

* **Solution:**

Use MTCNN (Multi-Task Cascaded Convolutional Neural Network) to solve the Face Regconition step.

To overcome the problem of One-shot learning, we use Siamese Network to recognize identified faces using only an original image.

The input is an image of a person, the output is to identify that person's face in the image and determine that person's identity based on the previously trained model.

Using Siamese Network to recognize faces

Face detection using MTCNN

# **II. Data Preparation and Modeling**

## **1. Selection of Data and Environment for Training**

The model is trained on a data set of five people, each with a unique picture.

Images are saved as .jpg or .png.

Install model development environment (Environment):

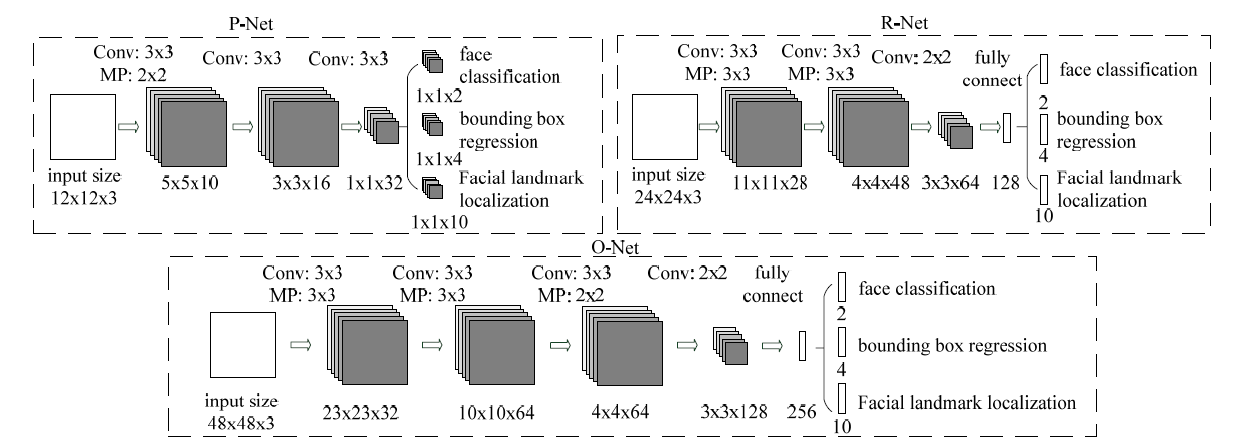
Operating System: Microsoft Windows 10 Home Version 2004

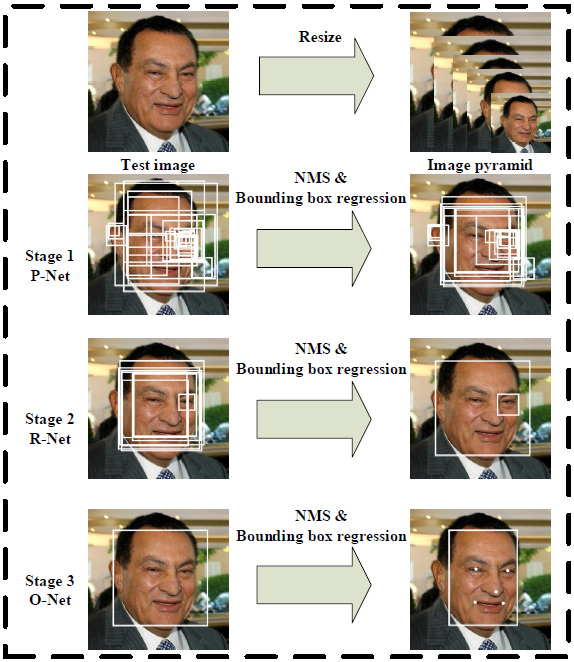
Package: Python 3.6

Tool: PyCharm Community Edition 2020.2.1

## **2. Multi-Task Cascaded Convolutional Neural Network (MTCNN)**

### **2.1 Concepts**

MTCNN works in 3 steps, each step has its own neural network: P-Net, R-Net and O-net.



For each input picture, it will create multiple copies of that image with different sizes as input for the next three steps.

**Step 1:** At P-Net, the algorithm uses a 12x12 kernel that runs through each image to find faces.

**Step 2:** At R-Net, there is a similar structure to P-Net but using more layers. Here, the network will use the bounding boxes provided from P-Net and fine-tune the coordinates.

**Step 3:** O-Net takes the bounding boxes from R-Net as input and marks the five coordinates of the landmarks on the face. In this step, the algorithm gives 3 different outputs including: probability of the face in the bounding box, coordinates of the bounding box and coordinates of landmarks on the face (position of eyes, nose, mouth). ).

Based on the UTK dataset of human faces consisting of 24,111 images, we have the following comparison between Haar Cascade and MTCNN from datawow.

|  |  |
| --- | --- |
| **Haarcascade** | **MTCN** |
| Number of face recognition electric: 19,915  Total faces detected from a single photo: 947  Recall = (19915/24111)\*100 = 82.60%  Precision = (18968/19915)\*100 = 95.24% | Number of face recognition electric: 21,666  Total faces detected from a single photo: 428  Recall = (21666 / 24111)\*100 = 89.85%  Precision = (21238/21666)\*100 = 98.02% |

From the above comparison, it can be seen that MTCNN has higher accuracy than Haarcascade and that is the reason why MTCNN is applied in this project.

### **2.2 Face Positioning Using MTCNN**

MTCNN is quite difficult to implement, fortunately, Python has MTCNN support. First, you need to import OpenCV library to read, write and display images, then import MTCNN.

from mtcnn.mtcnn import MTCNNimport cv2

For live video or camera, we use the function cv2.VideoCapture(0) of Open CV ddeh? ddread videos. Create a detector from class MTCNN. function detect\_facesddTo detect faces in the frame we use.

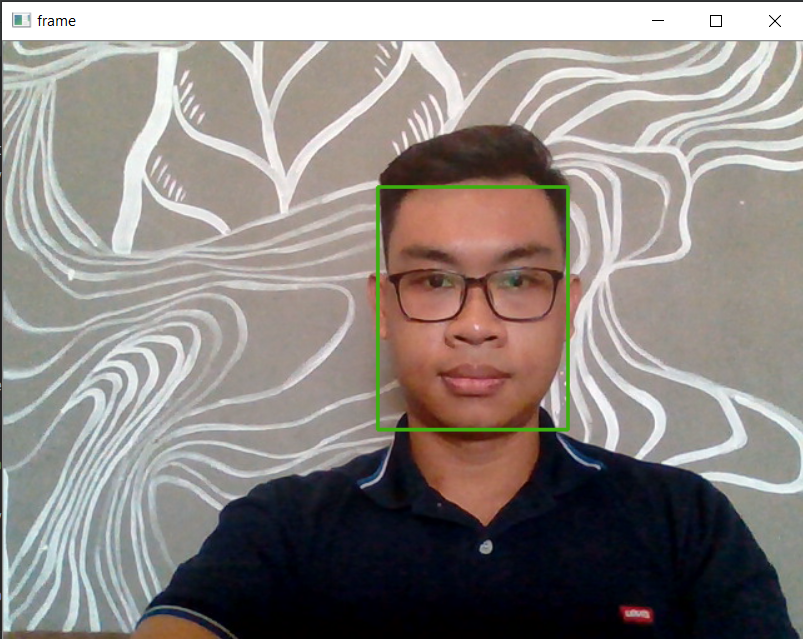
detector = MTCNN()cap = cv2.VideoCapture(0)while True: ret, frame = cap.read() result = detector.detect\_faces(frame)

All results returned include a list of boding boxes in there have coordinates dd2 eyes, nose, mouth. And with itddI believehuh?of these bounding boxes as well.

[{'confidence': 0.9969762563705444, 'keypoints': {'mouth\_right': (314, 186), 'nose': (303, 176), 'left\_eye': (280, 167), 'right\_eye': (305, 157), 'mouth\_left': (290, 195)}, 'box': [260, 131, 61, 81]}, {'confidence': 0.9960743188858032, 'keypoints': {'mouth\_right': (927, 218) , 'nose': (915, 203), 'left\_eye': (905, 184), 'right\_eye': (933, 186), 'mouth\_left': (905, 216)}, 'box': [888, 151 , 61, 83]}, {'confidence': 0.9958836436271667, 'keypoints': {'mouth\_right': (158, 212), 'nose': (149, 198), 'left\_eye': (134, 180), ' right\_eye': (163, 179), 'mouth\_left': (136, 212)}, 'box': [115, 148, 61, 81]}, {'confidence': 0.9955955147743225, 'keypoints':{'mouth\_right': (454, 228), 'nose': (449, 214), 'left\_eye': (436, 193), 'right\_eye': (463, 200), 'mouth\_left': (431, 223) }, 'box': [416, 164, 56, 79]}, {'confidence': 0.9930985569953918, 'keypoints': {'mouth\_right': (741, 236), 'nose': (726, 221), ' left\_eye': (713, 203), 'right\_eye': (744, 204), 'mouth\_left': (715, 235)}, 'box': [696, 174, 65, 82]}, {'confidence': 0.9929332733154297, 'keypoints': {'mouth\_right': (591, 262), 'nose': (580, 250), 'left\_eye': (567, 231), 'right\_eye': (596, 231), 'mouth\_left' : (569, 261)}, 'box': [551, 202, 59, 77]}, {'confidence': 0.9779009222984314, 'keypoints': {'mouth\_right': (1068, 203), 'nose' :( 1050, 191),'left\_eye': (1042, 173), 'right\_eye': (1070, 173), 'mouth\_left': (1043, 203)}, 'box': [1031, 144, 61, 79]}]

From this return, we can draw, rockmark the positions that MTCNN is already geta ra ddto check if the location detection algorithm has rightchhuh?a.

for face in result: x, y, w, h = face['box'] cv2.rectangle(frame, (x, y), (x+w, y+h), (15, 175, 61), 2 )



## **3. Siamese Networks**

### **3.1 Concepts**

Network architectures that when you put in two pictures and the model will tell if they belong to the same person or not are collectively known as Siamese Networks. Siamese Networks was first introduced by DeepFace: Closing the Gap to Human-Level - Yaniv Taigman. elt.

The architecture of Siamese Networks is based on a base network which is a Convolutional neural network with the output layer removed which encodes the image into vector embedding. The input to the siam network is two random images selected from the image data. Output of Siamese Networks is 2 vectors corresponding to the representation of 2 input images. We then feed the two vectors into the loss function to measure the difference between them. Usually the loss function is a standard 2nd order norm function.

When using Siamese Networks we will not need to worry about the output shape changing because the base network has been removed the last layer.

The main goal of Siamese Networks is to find the representation of an image in n-dimensional space, so it is not necessary to choose a loss function that is a binary cross entropy function like binary classification problems. In fact, choosing the loss function as binary cross entropy still finds a good representation of the image.

### **3.2 Facenet Algorithm**

Facenet is a form of Siamese Network that represents images in an n-dimensional eucledean space (usually 128) such that the smaller the distance between the embedding vectors, the greater the degree of similarity between them.

Most of the pre-facenet face recognition algorithms try to represent the face with an embedding vector through a bottle neck layer which has the effect of reducing the data dimension.

* However, the limitation of these algorithms is that the number of embedding dimensions is relatively large (usually >= 1000) and affects the speed of the algorithm. Often we have to apply the PCA algorithm to reduce the data dimension to reduce the calculation speed.
* The loss function only measures the distance between two images. Thus, in a training input, only one of two possibilities can be learned which is the similarity if they are of the same class or the difference if they are of different classes without learning the similarity and the difference at the same time on the same training run. training.

Facenet has solved both of these problems with small but highly effective tweaks:

* The base network applies a convolutional neural network and reduces the data dimension to only 128 dimensions. Therefore, the inference and prediction process is faster and at the same time the accuracy is still guaranteed.
* Using the loss function is a triplet loss function capable of simultaneously learning the similarity between two photos in the same group and distinguishing the images that are not in the same group. It is therefore much more efficient than previous methods.

### **3.3 Triplet Loss**

In facenet, the encoding process of the convolutional neural network helps us to encode the image in 128 dimensions. These vectors are then used as input to a loss function that evaluates the distance between the vectors.

To apply triple loss, we need to take out 3 pictures in which one is the anchor image. Of the 3 images, the anchor image is fixed first. We will select the remaining 2 images so that one is negative (of another person with the anchor) and the other is positive (the same person with the anchor).

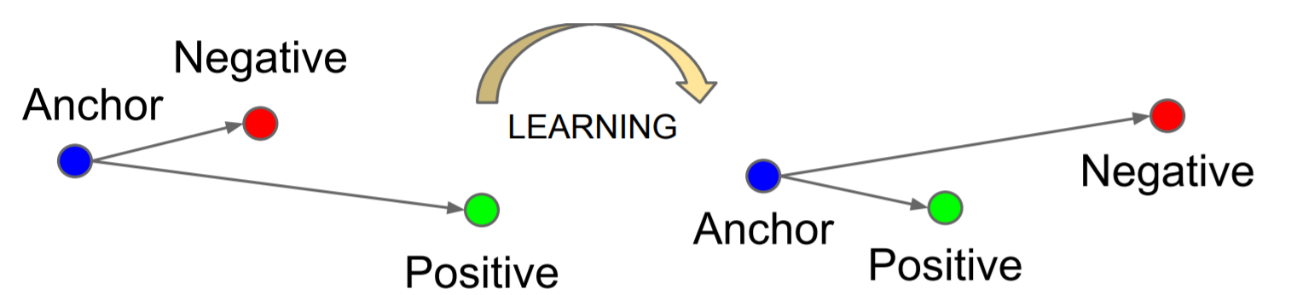
Anchor, Positive, Negative image symbols are A, P, N . respectively

The goal of the loss function is to minimize the distance between two images when they are negative and maximize the distance when they are positive. So we need to select sets of 3 images such that:

* Anchor and Positive images are the most different: it is necessary to choose so that the distance d(A,P) is large. This is similar to you choosing a photo of yourself when you were younger than you are now so that the algorithm learns more difficult. But if it is aware, it will be smarter.
* Anchor and Negative images are most similar: it is necessary to choose so that the distance d(A,N) is small. This is similar to the algorithm that distinguishes a photo of a brother who looks like you from you.

The triplot loss function always takes 3 images as input and in all cases we expect, the distance between Anchor and Positive must be less than the distance between Anchor and Negative:

*d*(A,P)<d(A,N)

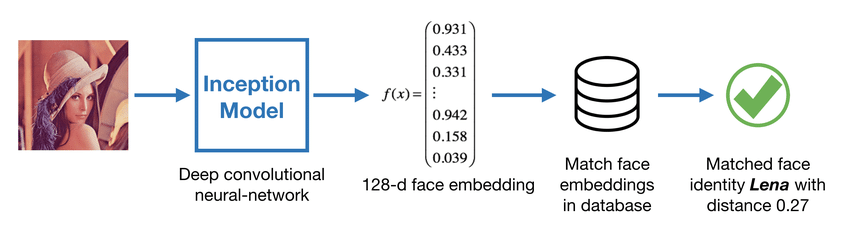


### **3.4 Pre-train Model Facenet**

Most of us when building a face recognition algorithm will not need to retrain the facenet model, but take advantage of the existing pretrain models. You won't need to waste time and effort if you don't have oneddincubate resources and data. There This is also the reason why I think that building facial recognition models in ddCurrent score is very easy.

Pretrain models gettrained on data up to millions of images. Dothere possibleăng encodes images very well in 128 dimensions. All that remains for us is to reuse the model, compute the vector embeddingeh? and train embedding vectorseh? by a classifier pausesimple ddto classify classes.

Facenet model.



* **The working steps of the Facenet model:**

**Step 1:** Encode the face image into a 128-dimensional vector

The FaceNet model takes a lot of data and time momentumo create long term. So, it is customary to use the weights that have been trained before. The Facenet network architecture follows the Inception model of Szegedy et al.

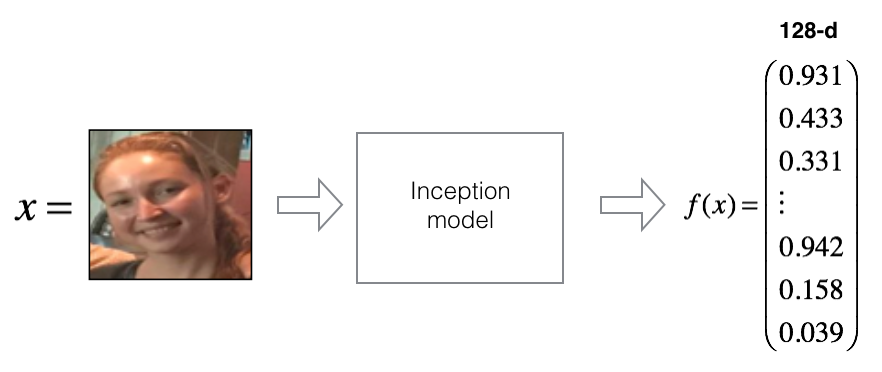
This model uses a 96x96 dimensional RGB image as input. When inputting a face image or a series of face images as a tensor whose shape is (𝑚,𝑛𝐶,𝑛𝐻,𝑛𝑊)=(𝑚,3,96,96), it will output a matrix of the form (𝑚,128).

FRmodel = faceRecoModel(input\_shape=(3, 96, 96))print("Total Params:", FRmodel.count\_params())

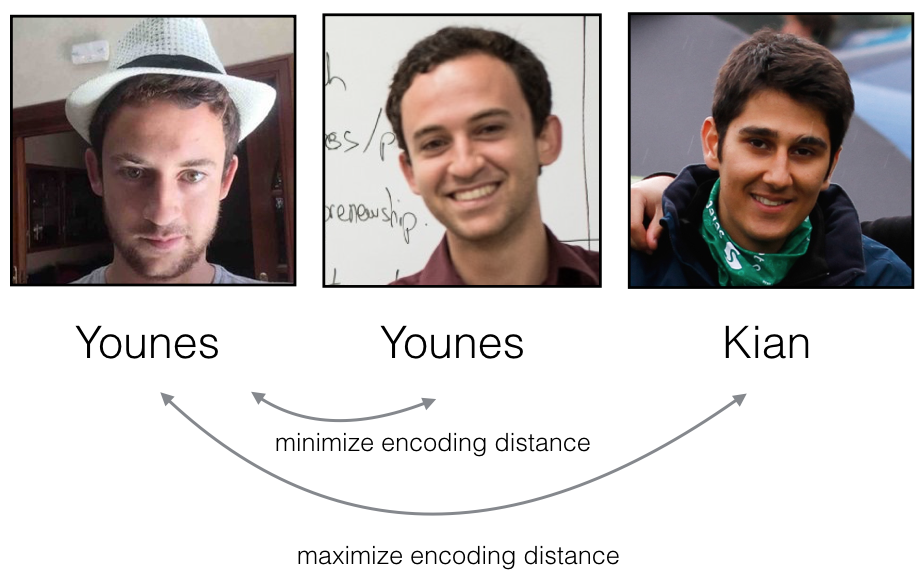
Total output params:



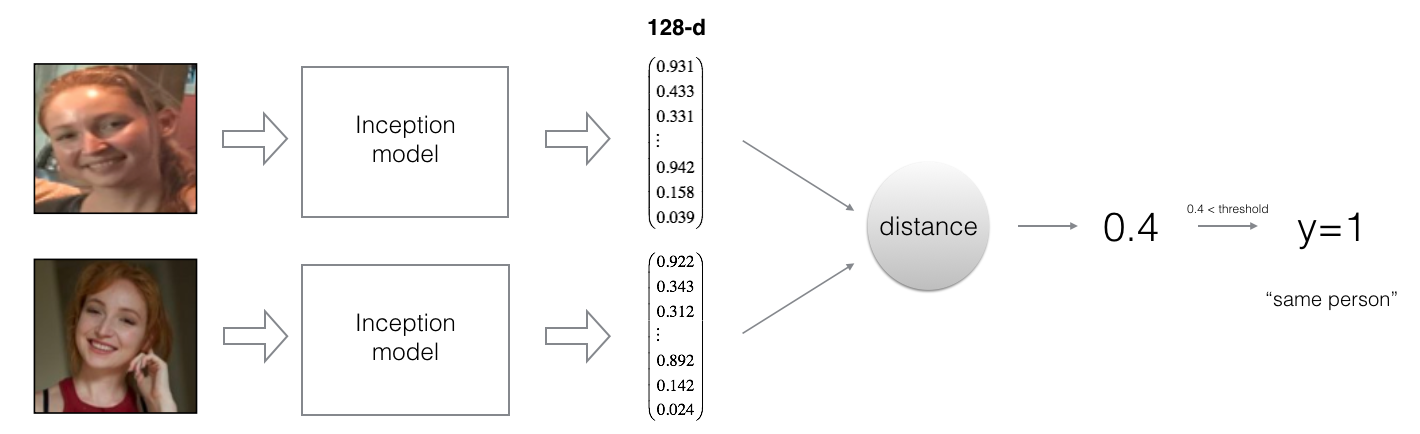
**Step 2:** The Inception model will be used to encode each input face image into a 128-dimensional vector.



Next, the triplet loss function will try to push the encodings of the same person (Anchor and Positive) closer together, while pulling the encodings of two different people images (Anchor and Negative) further apart.



Then compare the distance between the two encodings of the two pictures, if the distance is less than the threshold, then this is the same person.



* **Instructions to retrain the Facenet model pretrain:**

**Step 1:** Build a triplet loss function for training.

def triplet\_loss(y\_true, y\_pred, alpha=0.2): # triplet formula components anchor, positive, negative = y\_pred[0], y\_pred[1], y\_pred[2]  
  
 # Step 1: Calculate the distance between the anchor and the positive, we will need to sum on the axis equal to -1.

pos\_dist = tf.reduce\_sum(tf.square(tf.subtract(anchor, positive)), axis=-1)

# Step 2: Calculate the distance between anchor and negative, we will need to sum on the axis equal to -1.

neg\_dist = tf.reduce\_sum(tf.square(tf.subtract(anchor, negative)), axis=-1)

# Step 3: Subtract the previous two distances and add alpha.

basic\_loss = tf.add(tf.subtract(pos\_dist, neg\_dist), alpha)

# Step 4: Get the maximum value of basic\_loss and 0.0. Sum the training values.

loss = tf.reduce\_sum(tf.maximum(basic\_loss, 0.))  
  
 return loss

**Step 2:** Facenet is trained by minimizing the triplet loss function.

with tf.compat.v1.Session() as test: tf.compat.v1.set\_random\_seed(1) y\_true = (None, None, None) y\_pred = (tf.random.normal([3, 128], mean=6 , stddev=0.1, seed=1), tf.random.normal([3, 128], mean=1, stddev=1, seed=1), tf.random.normal([3, 128], mean=3 , stddev=4, seed=1)) loss = triplet\_loss(y\_true, y\_pred)  
  
 print("loss = " + str(loss.eval()))  
  
  
FRmodel = faceRecoModel(input\_shape=(3, 96, 96))print("Total Params:", FRmodel.count\_params())FRmodel.compile(optimizer = 'adam', loss = triplet\_loss, metrics = ['accuracy']) load\_weights\_from\_FaceNet(FRmodel)FRmodel.save('models/model.h5')

After training, we will save the model as a .h5 file. The model will then be used to identify the face.



* Model Rating:
* Model accuracy: 93.8%
* Area Under Curve (AUC): 97.9%
* Equal Error Rate (EER): 6.2%

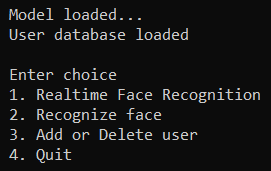
# **III. Model Deployment**

Build user interfaces capable of:

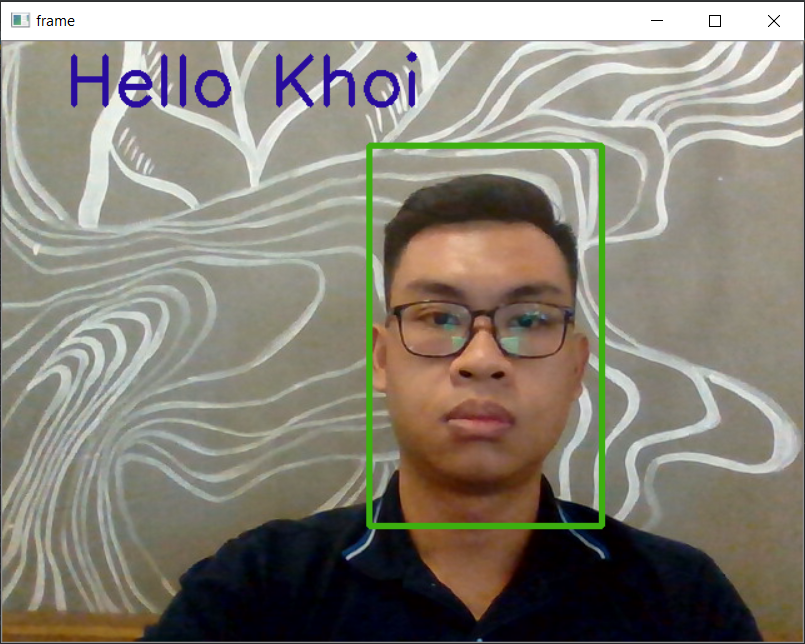
Real-time user face recognition.

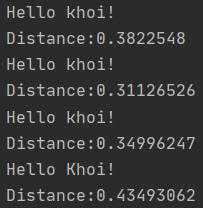
Capture the user's face and identify the user.

Allows users to add or remove recognized faces in the database.



When selecting 1, the webcam will record the user in real time and determine the user's identity along with the distance between the user's face and the face in the database.

****

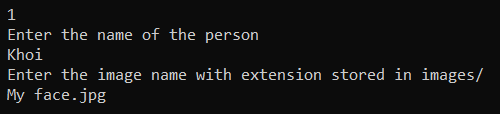
****

When selecting 3, the user will be asked to choose whether to add or remove the image of the trained user.

If you choose to add users, you will have two options: add directly from the webcam or via previously saved photos.



If 1 is selected, the user will be asked to enter the name and name of the image file that was previously saved in the 'images' folder.



If you choose 2, the user will be asked to enter a name and the webcam will automatically take a picture of the user and save it as a database.

# **IV. Conclude**

The combination of MTCNN and Siamese Networks produces a highly accurate face recognition model.

The results show an accuracy of 98.02% when using MTCNN to identify faces in the image and 93.8% when recognizing previously trained faces.

# **V. Suggestions for Improvement**

As a premise to build real face recognition systems in schools and agencies by CCTV for timekeeping and security monitoring.

# **REFERENCES**

[1] Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks, Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, Senior Member, IEEE, and Yu Qiao, Senior Member, IEEE.

[2] One Shot Learning – deeplearning.ai | Coursera, Course 4 of 5 in the Deep Learning Specilization,<https://www.coursera.org/lecture/convolutional-neural-networks/one-shot-learning-gjckG>

[3] Lesson 27 - Facenet model in face recognition, Data science - Khanh's blog

Pham Dinh Khanh, Data Scientist at VinID, March 12, 2020.