UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER NETWORK AND COMMUNICATION


**REPORT**
Subject: Cryptography
Semester II (2021 – 2022)

# PROTECT THE PRIVACY FOR SMART DEVICE USING FACE AUTHENTICATION

Student: Võ Anh Kiệt
Student ID Number: 20520605
Student: Nguyễn Bảo Phương
Student ID Number: 20520704
Class: NT219.M21.ANTN
University of Information Technology
Lecturer: Nguyễn Ngọc Tự

**Hồ Chí Minh City, March 2022**

UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER NETWORK AND COMMUNICATION


**REPORT**
Subject: Cryptography
Semester II (2021 – 2022)

# PROTECT THE PRIVACY FOR SMART DEVICE USING FACE AUTHENTICATION

Student: Võ Anh Kiệt

Student ID Number: 20520605

Student: Nguyễn Bảo Phương

Student ID Number: 20520704

Class: NT219.M21.ANTN

University of Information Technology

Lecturer: Nguyễn Ngọc Tự

**Hồ Chí Minh City, March 2022**

**Acknowledgement**

To begin, we would like to thank our advisor, PhD Nguyễn Ngọc Tự, for his direction and consistent monitoring, as well as for providing important project information and for their help in finishing the research.

My gratitude and appreciation also extend to our colleagues and lecturers who assisted us in the development of the project, as well as to those who have volunteered their time and skills to assist us.

Võ Anh Kiệt – 20520605 – ANTN.2020

Nguyễn Bảo Phương – 20520704 – ANTN.2020

# Contents

# Part 1: Introduction

## 1.1. Overview

Biometrics, a method of automatically identifying a person based on physiological or behavioral characteristics instead of traditional authentication methods by granting them access to physical and virtual domains based on a password, PIN code, smart card, plastic card, token, key, and so on., has emerged as the most promising option for recognizing individuals in recent years. There are several biometric systems such as signature, fingerprint, voice, iris, retina, hand geometry, ear, and face geometry. Among these systems, facial recognition seems to be one of the most popular, collectible, and accessible. Facial recognition has several advantages over other biometric methods, as demonstrated here: most other biometric-based techniques require the user to voluntarily take an action, for example such as placing your palm up for fingerprints or holding it in a fixed position in front of the camera to identify iris or retina. On the contrary, face recognition can be done passively without any specific action as face images can be obtained by remote camera. Furthermore, there are many possible problems with other techniques during data collection in general such as the uselessness of fingerprints for a hypothetical person or the high cost of equipment during the data collection process. using iris and retina etc. However, facial images can be easily obtained with a few inexpensive still cameras. Good face recognition algorithms and proper image pre-processing can compensate for noise and small changes in orientation, scale, and lighting. Finally, while many techniques use the same device to capture biological characteristics that can lead to the transmission of germs and impurities by other users. In contrast, facial recognition is completely non-intrusive and poses no danger to the user's health.

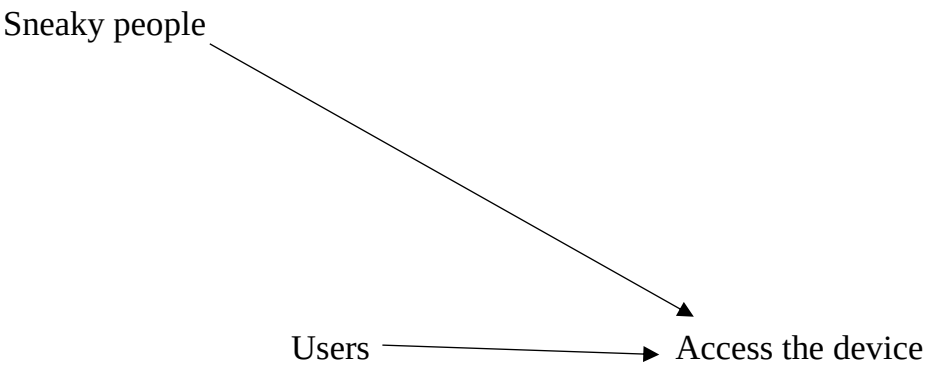
Face recognition has a long history dating back to the 1960s with manual measurement of the coordinate position of various facial features such as eyes, ears, nose and mouth, increasing accuracy by adding more markers are subjective but always have to be calculated manually.

Until 1991, the first version of automatic face recognition was released but was not ready until 2006, the performance of the latest facial recognition algorithms was evaluated using using the latest facial recognition algorithms available. Facebook is the first social media company to begin rolling out facial recognition that helps identify people whose faces may appear in photos Facebook users update daily. airport, financial authentication, etc. Originally a form of computer application, it has recently been used in a variety of platforms, from mobile devices to in-vehicle devices. But he still hasn't fully overcome the ongoing challenges with the quality of his delivery such as lighting, backdrops, naming a few, and importantly, the intricacies of facial recognition. relies on a 'too deep' architecture of complex neural networks (CNNs) that are complex and unsuitable for real-time performance on embedded devices.

## 1.2. Problem Statement

Technology has accelerated the speed of data access, acquisition and creation, users' devices are now rich repositories of their life memories and content. Many users use their devices for most browsing activities, the importance of improving security and privacy on smartphones is becoming more and more important.

Despite the development of technology, there are more and more cases of information disclosure, theft, data information, invasion, data loss on the device because some sneaky people try to access the phone and put the data on public place on the internet.




## 1.3. Scope

Face recognition

Users

Sneaky people

## 1.4. Objective

Understand the core concepts and algorithm using face recognition

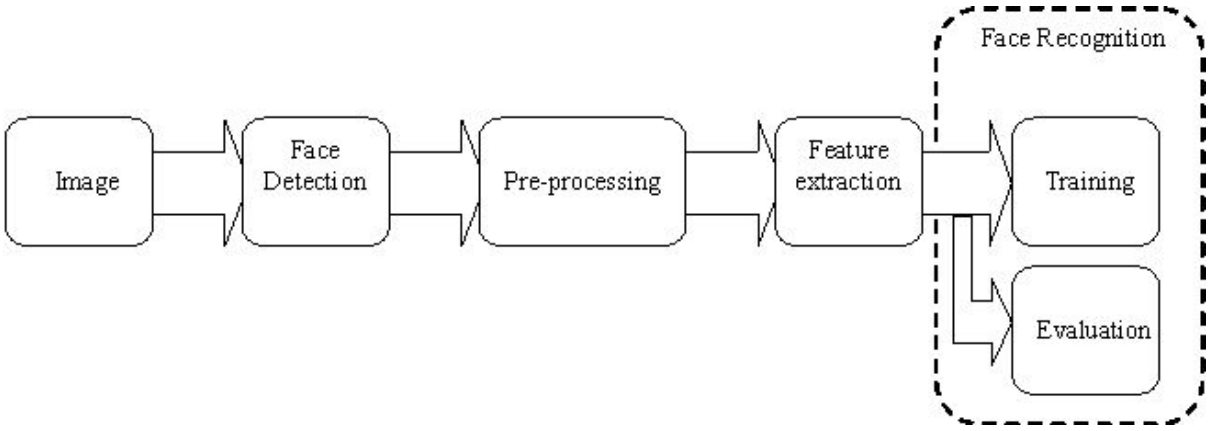
Build a simple face authentication mechanism to verify the identity quickly and protect the privacy of the users.

## Part 2: Background

### 2.1.1. Fundamental of face recognize

Facial recognition essentially works the same way as other forms of "biometric" identification such as voice, iris, or fingerprint recognition. Given a specific photo or other biometric data as input, the computer analyzes and looks for a very specific set of markers within it. Comparing facets in this way is conceptually like comparing fingerprints, although much more complex. If the computer finds a significant similarity threshold between the sample and the example samples, a match is declared.

The process of recognizing a face in an image has three phases:




- Face detection: Determines whether a human face appears in each image and detects pixels in the image that represent a face.
- Preprocessing: A variety of preprocessing techniques are used to minimize the effect of factors that can negatively affect the facial recognition algorithm.
- Feature extraction: the features used in the recognition phase are computed. These features vary depending on the automatic facial recognition system used. For example, the first and simplest features used in face recognition are the geometrical and distance relationships between key points on the face and the recognition "algorithm" corresponding to those intervals, this way.
- Face recognition: It consists of two separate steps:

- Training: Create a database of faces. For each person, several images are taken, and their characteristics are extracted and stored in a database.
- Evaluation: When a face image is available, face detection and feature extraction are used so that facial features can be obtained to compare with each face class available in the database The more appropriate the data and model, the more determined. If they are close enough, a recognition event is fired. In general, facial recognition is used for two main tasks:
  - Verification (1-1 match): When presented with an image of the face of an unidentified individual along with identity confirmation, verifying whether the individual is who he claims to be.
  - Identity (1-many match): given an image of an unknown individual, identify that person by comparing (possibly after encoding) that image with a database (can be encrypted) images of known individuals.

## 2.2. OpenCV

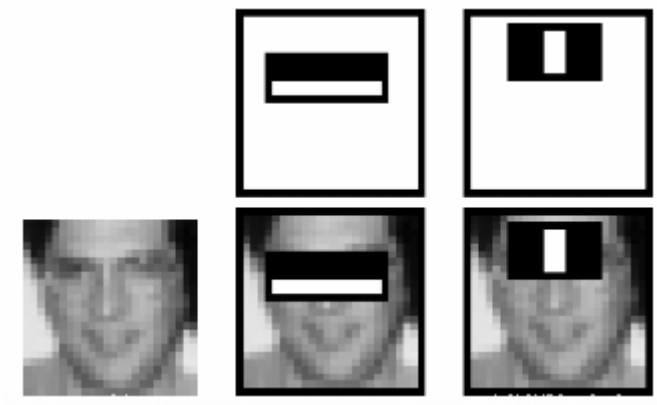
OpenCV is a library of programming functions primarily for real-time computer vision. Free and cross-platform library under the open-source BSD license. OpenCV supports TensorFlow, Torch/PyTorch, and Caffe deep learning frameworks. OpenCV launches with a pre-trained engine that can be used for face detection.

To detect face from camera stream, we use both the the suppliment from both the HaarCascade and Caffe to build the register and verifier. Finally, make the comparision.

## 2.3. OpenCV – HaarCascade Classification

The Haar Cascades classifier, which uses Haar's feature-based cascading classifier, is an efficient object detection method proposed by Paul Viola and Michael Jones in their paper, "Object Detection rapidly using simple feature

enhancement cascaders" in 2001. It is a machine learning-based approach in which a cascade function is formed from a large number of positive and negative images. It is then used to detect objects in other images. HaarFeatures has very good edge and line detection. This makes it especially good at detecting faces.



Different stages in visualization. Source: docs.opencv.org

However, since Haar Features must be defined manually, there is some limit to the types of things they can detect. If you feed the classifier (a network or any algorithm that detects faces) edge and line features, it will only be able to detect objects with obvious edges and lines. Even as a face detector, if we manipulate the face a little (e.g. cover our eyes with sunglasses or tilt our head to one side), the Haar-based classifier may not recognize it. face.

## 2.4. OpenCV – Single Shot Detector base ResNet network – Caffe Model

OpenCV's deep learning face detector is based on the Single Shot Detector (SSD) framework with the ResNet base framework.

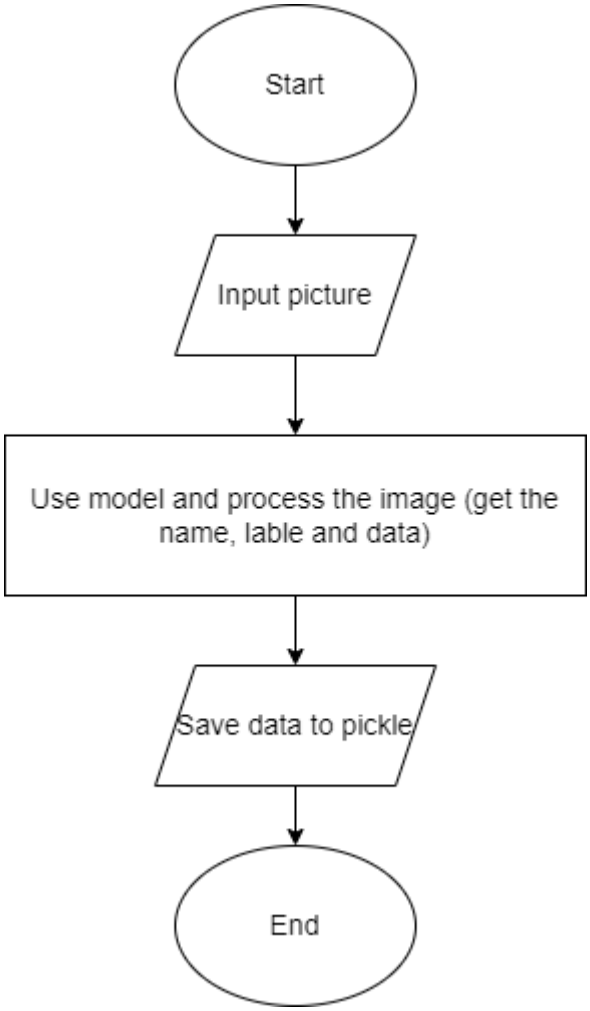
The SSD approach is based on a forward agglomeration network that generates a fixed-size set of bounding boxes and scores for the presence of feature class instances in those boxes, then zero-decimal deletion step to generate the final detection. The first network layers are based on a standard architecture used for high-quality image classification (truncated before any classifiers), which we will call the underlying ResNet network.

ResNet, the so-called Residual Neural Network by Kaiming He et al., introduced a new architecture with "jump connections" and provided heavy batch normalization at ILSVRC 2015. These hopping connections also known as 'closed unit' or closed repeat unit and bears strong resemblance to recent success factors applied in RNN. Thanks to this technique, they were able to generate a NN with 152 layers while having less complexity than VGGNet. It achieves a top5 error rate of 3.57%, which exceeds human performance on this dataset.

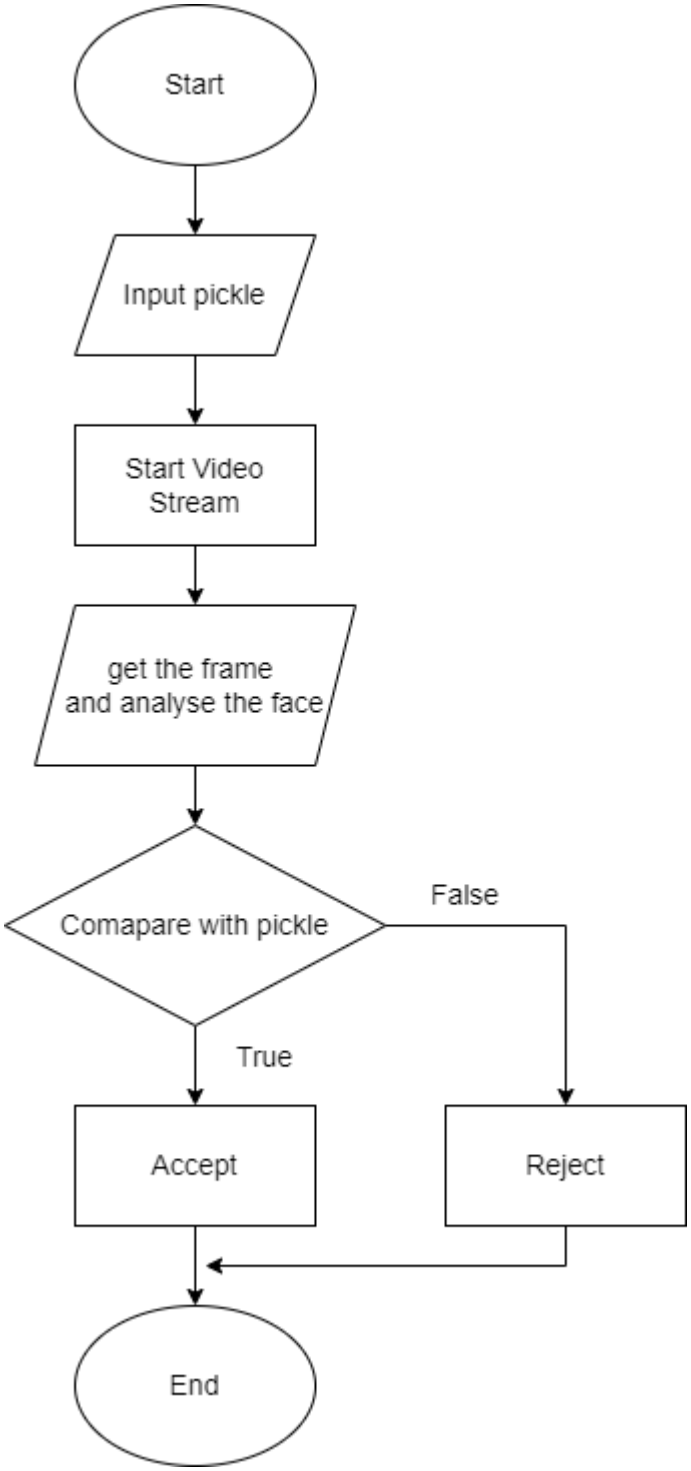
# Part 3: Design and Code

## 3.1. Design

### 3.1.1. Register



### 3.1.2. Recognize and verify



## 3.2. HaarCascade

### 3.2.1. Register

Get the data of the users from Images

```
#get the data from Images
imagePaths = list(paths.list_images('Images'))
knownEncodings = []
knownNames = []
```

Process the image

```
#start processing the images
for (i, imagePath) in enumerate(imagePaths):
    print("Processing image {}".format(imagePaths))

    #get the name of the person from the image
    name = imagePath.split(os.path.sep)[-2]

    #get the picture and transfer to RGB
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    #use face_recognition to get the face location
    boxes = face_recognition.face_locations(rgb,model='hog')

    #get the facial of the face
    encodings = face_recognition.face_encodings(rgb, boxes)

    #save the data to the knownEncodings and knownNames
    for encoding in encodings:
        knownEncodings.append(encoding)
        knownNames.append(name)
```

Save the data to pickle file

```
#save the encodings and names to data
data = {"encodings": knownEncodings, "names": knownNames}

#use pickle to save data into face.pickle and use to recognize
f = open('pickleFile/face.pickle', "wb")
f.write(pickle.dumps(data))
f.close()

#end of the program
print("Finished")
```

### 3.2.2. Recognize and Verify

Load the model and pickle file

```
# load the detection model
print("Loading the detection model")
cascPathface = os.path.dirname(cv2.__file__) + "/data/haarcascade_frontalface_alt.xml"

# load the recognition model
print("Loading the recognition model")
faceCascade = cv2.CascadeClassifier(cascPathface)

# load the pickle file
print("Loading the pickle file")
data = pickle.loads(open('pickleFile/face.pickle', "rb").read())
```

Load the video stream, start recognizing and verifying

```
while True:

    # get the frame from the threaded video stream
    ret, frame = video_capture.read()

    # convert the frame to grayscale and resize it to the width of the screen
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # detect faces in the grayscale frame
    faces = faceCascade.detectMultiScale(
        gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)

    # transfer the input frame from BGR to RGB
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # initialize the list of recognized faces
    encodings = face_recognition.face_encodings(rgb)
    names = []

    for encoding in encodings:

        # match the encoding against the known encodings
        matches = face_recognition.compare_faces(data["encodings"], encoding)

        # set name = unknown and reject if there are too many or too few matches or no encoding matches
        name = "Unknown_Rejected"

        # if the encoding matches, set the name to the name of the person
        if True in matches:

            # get the index of the first match
            matchedIdxs = [i for (i, b) in enumerate(matches) if b]

            # count the number of matches for each person
            counts = {}
```

```
            for i in matchedIdxs:

                # get the name of the person
                name = data["names"][i]

                # increase count for the name we got
                counts[name] = counts.get(name, 0) + 1

            # set name which has highest count
            name = max(counts, key=counts.get)

        # update the list of names
        names.append(name)

        for ((x, y, w, h), name) in zip(faces, names):

            # draw the rectangle around the face
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

            # draw the name of the person
            if (name != "Unknown_Rejected"):
                cv2.putText(frame, name + "_Accepted", (x, y),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 225, 0), 2)
            else:
                cv2.putText(frame, name, (x, y),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 225, 0), 2)

    # display the frame
    cv2.imshow("Frame", frame)

    # if q is pressed, break the loop
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# clean up the camera and close any open windows
video_capture.release()
cv2.destroyAllWindows()

# end of program
print("Finished")
```

## 3.3. Caffe

### 3.3.1. Register

Load the model, database and declare variable

```
import numpy as np
import cv2
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
import pickle
import os
import imutils

#get the current working directory
curr_path = os.getcwd()

#load the face detection model
print("Loading face detection model")
proto_path = os.path.join(curr_path, 'model', 'deploy.prototxt')
model_path = os.path.join(curr_path, 'model', 'res10_300×300_ssd_iter_140000.caffemodel')
face_detector = cv2.dnn.readNetFromCaffe(prototxt=proto_path, caffeModel=model_path)

#load the face recognition model
print("Loading face recognition model")
recognition_model = os.path.join(curr_path, 'model', 'openface_nn4.small2.v1.t7')
face_recognizer = cv2.dnn.readNetFromTorch(model=recognition_model)

#load the database
print("Loading the database")
data_base_path = os.path.join(curr_path, 'database')

#load the labels
filenames = []
for path, subdirs, files in os.walk(data_base_path):
    for name in files:
        filenames.append(os.path.join(path, name))

#declare the list of labels
face_embeddings = []
face_names = []
```

Process the image

```python
for (i, filename) in enumerate(filenames):
    #load the image and process it
    print("Processing image {}".format(filename))

    #load the image
    image = cv2.imread(filename)

    #resize the image
    image = imutils.resize(image, width=600)

    #get the image dimensions
    (h, w) = image.shape[:2]

    #detect the face with the face detector model - using ResNet Neural Network - Caffe Model
    image_blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 1.0, (300, 300), (104.0, 177.0, 123.0), False, False)

    #set the input to the face detector model
    face_detector.setInput(image_blob)

    #detect the face
    face_detections = face_detector.forward()

    #check if at least one face was detected
    i = np.argmax(face_detections[0, 0, :, 2])

    #check if the face was detected
    confidence = face_detections[0, 0, i, 2]

    #check if the confidence is greater than the threshold
    if confidence > 0.5:
        #get the coordinates of the face
        box = face_detections[0, 0, i, 3:7] * np.array([w, h, w, h])

        #get the face ROI
        (startX, startY, endX, endY) = box.astype("int")

        #extract the face ROI
        face = image[startY:endY, startX:endX]

        #resize the face ROI
        face_blob = cv2.dnn.blobFromImage(face, 1.0/255, (96, 96), (0, 0), True, False)

        #set the input to the face recognition model
        face_recognizer.setInput(face_blob)

        #recognize the face
        face_recognitions = face_recognizer.forward()

        #get the name of the label
        name = filename.split(os.path.sep)[-2]

        #add the embedding vector to the list
        face_embeddings.append(face_recognitions.flatten())

        #add the name to the list
        face_names.append(name)
```

Save the data to pickle file

```
#get the data with face embeddings and names
data = {"embeddings": face_embeddings, "names": face_names}

#get the encoded labels
le = LabelEncoder()

#transfer the names to the encoded labels
labels = le.fit_transform((data["names"]))

#get the recognizer
recognizer = SVC(C=1.0, kernel="linear", probability=True)

#get the recognizer with embeddings and labels
recognizer.fit(data["embeddings"], labels)

#save the recognizer to pickle file
print("Saving the pickle file")
f = open('pickleFile/recognizer.pickle', "wb")
f.write(pickle.dumps(recognizer))
f.close()

#save the label encoder to pickle file
f = open('pickleFile/le.pickle', "wb")
f.write(pickle.dumps(le))
f.close()

#end of the program
print("Finished")
```

### 3.3.2. Recognize and Verify

Load the model and data

```
import numpy as np
import pickle
import os
import cv2
import time
import datetime
import imutils

#get the current directory of the file
curr_path = os.getcwd()

#load the face detection model
print("Loading face detection model")
proto_path = os.path.join(curr_path, 'model', 'deploy.prototxt')
model_path = os.path.join(curr_path, 'model', 'res10_300x300_ssd_iter_140000.caffemodel')
face_detector = cv2.dnn.readNetFromCaffe(prototxt=proto_path, caffeModel=model_path)

#load the face recognition model
print("Loading face recognition model")
recognition_model = os.path.join(curr_path, 'model', 'openface_nn4.small2.v1.t7')
face_recognizer = cv2.dnn.readNetFromTorch(model=recognition_model)

#load the pickle file
print("Loading the pickle file")
recognizer = pickle.loads(open('pickleFile/recognizer.pickle', "rb").read())
le = pickle.loads(open('pickleFile/le.pickle', "rb").read())
```

Start recognizing and verify

```
print( Starting recognition )
vs = cv2.VideoCapture(0)
time.sleep(1)

while True:
    #read the frame from the video stream
    ret, frame = vs.read()

    #resize the frame
    frame = imutils.resize(frame, width=600)

    #get the image dimensions
    (h, w) = frame.shape[:2]

    #detect the face with the face detector model - using ResNet Neural Network - Caffe Model
    image_blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 1.0, (300, 300), (104.0, 177.0, 123.0), False, False)

    #set the input to the face detector model
    face_detector.setInput(image_blob)

    #detect the face
    face_detections = face_detector.forward()

    for i in range(0, face_detections.shape[2]):

        #check if the face was detected
        confidence = face_detections[0, 0, i, 2]

        #check if the confidence is greater than the threshold
        if confidence > 0.5:

            #get the coordinates of the face
            box = face_detections[0, 0, i, 3:7] * np.array([w, h, w, h])

            #get the face ROI
            (startX, startY, endX, endY) = box.astype("int")

            #extract the face ROI
            face = frame[startY:endY, startX:endX]

            #get the face shape
            (fH, fW) = face.shape[:2]

            #resize the face ROI
            face_blob = cv2.dnn.blobFromImage(face, 1.0/255, (96, 96), (0, 0, 0), True, False)

            #set the input to the face recognition model
            face_recognizer.setInput(face_blob)

            #recognize the face
            vec = face_recognizer.forward()

            #get the label for the face
            preds = recognizer.predict_proba(vec)[0]
            j = np.argmax(preds)
            proba = preds[j]
            name = le.classes_[j]
```

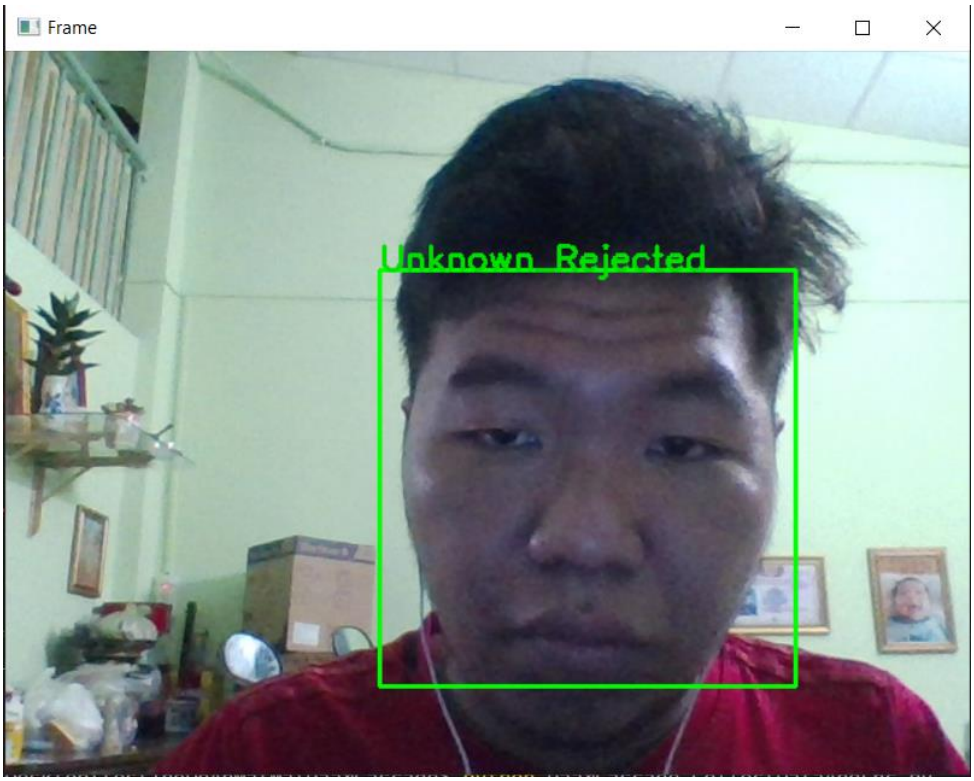
```
            if(name == 'Unknown'):
                name = 'Unknown_Rejected'
            else:
                name = name + '_Accepted'

            #set the value of the text
            text = "{}: {:.2f}".format(name, proba * 100)
            y = startY - 10 if startY - 10 > 10 else startY + 10

            #draw the rectangle around the face
            cv2.rectangle(frame, (startX, startY), (endX, endY), (0, 0, 255), 2)

            #draw the name of the person
            cv2.putText(frame, text, (startX, y), cv2.FONT_HERSHEY_SIMPLEX, 0.65, (0, 0, 255), 2)

    #display the frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    #if q is pressed, break the loop
    if key == ord('q'):
        break

#clean up the camera and close any open windows
cv2.destroyAllWindows()

#end of program
print("Finished")
```

# Part 4: Implement and Test

## 4.1. Implemet

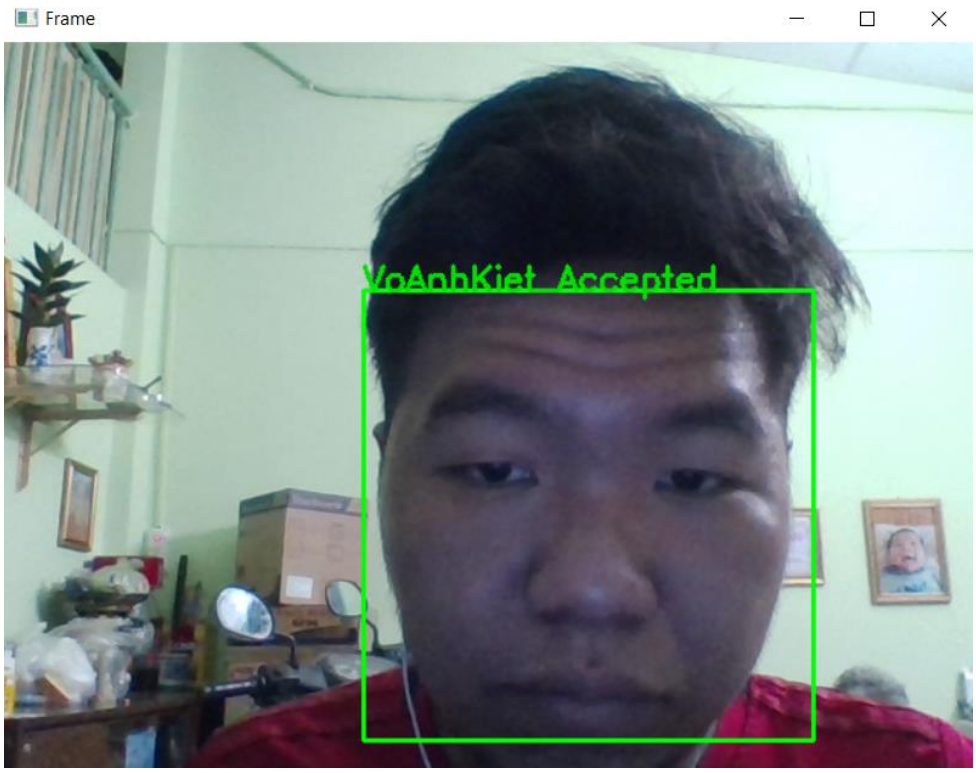
### 4.1.1. HaarCascade

Before registering



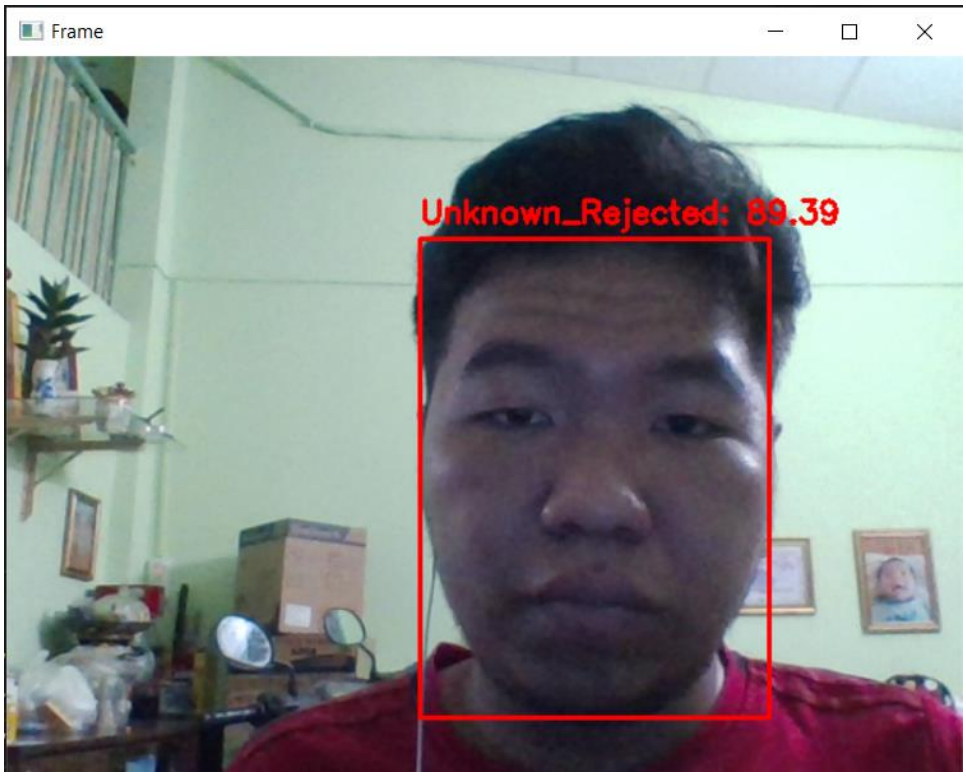

Register

Recognize and verify




### 4.1.2. Caffe

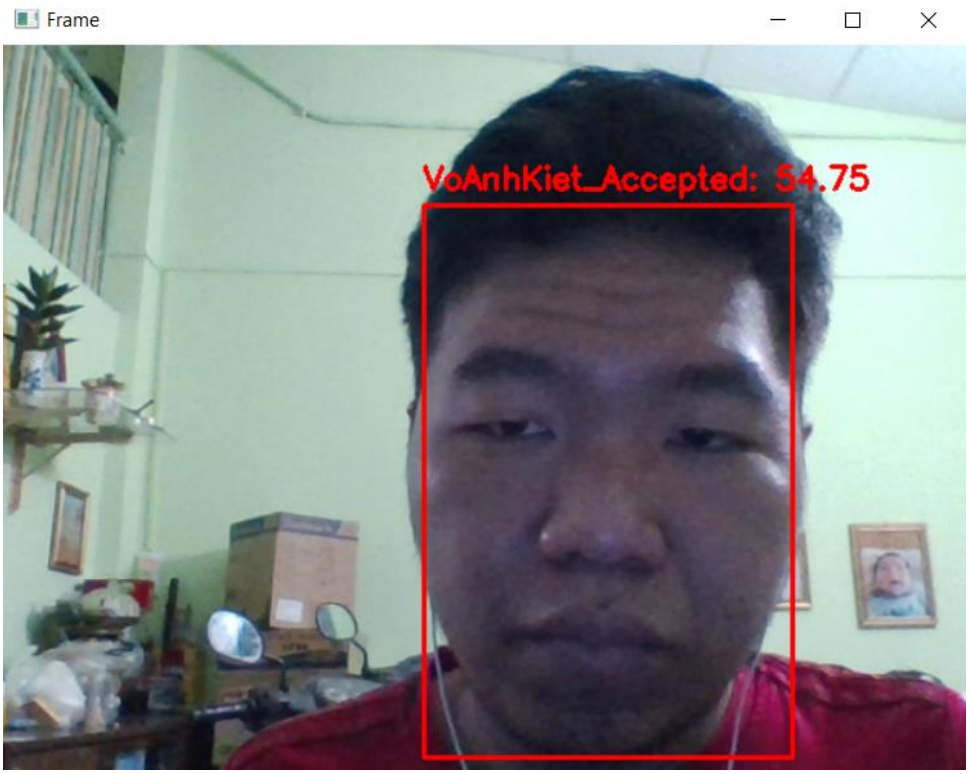
Before registering



## Register

```
PS C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe> python Caffe_CollectDataAndEnc.py
Loading face detection model
Loading face recognition model
Loading the database
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\278637350_4930545123665332_799118479497993888_n.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\278910487_520589669561806_5719928888139035809_n.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\HongTam1.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\HongTam3.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382048698714_0ddd450138a906428ce3f95ff568edab.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382048709574_e46eb5afc0b29983fd5cf454763df7be.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382051712686_76c6ea3545d3d9bdbceac0e6e592c2a3.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382051714089_224c95b33100bcd9480349732774ea26.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382055518867_e98a9f65ce9ef3a5f9d60045981de240.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382055519272_862d122b1c078f227d36197e3905cbf6.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382055521204_03f1c815ef564e4c549301451dda1e95.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382055521462_8eb69b2ba0a2d62093b8e5a161210f18.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382055526954_2d2e20f2a6b31bcf008010322166e31f.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382055530974_f5746888193268l6a21e4373b55c1fdb.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382056448114_3890f28a3d09a32e80c4e09fe75cafe7.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382056855802_5e05d39630823f8c54815f44b21dda8a.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\TranHongTam\z3382057090924_c75ac4eec0c77caf8eb9980e41a251b9.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\Unknown\alan_grant.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\Unknown\claire_dearing.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\Unknown\ellie_sattler.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\Unknown\ian_malcolm.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\Unknown\john_hammond.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\Unknown\own_grady.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\VoAnhKiet\107552911_2691735477780511_8839448572045971408_n.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\VoAnhKiet\32530691_2070049716615760_6476359929845252096_n.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\VoAnhKiet\48426229_2242591992694864_3335421617244209152_n.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\VoAnhKiet\72428896_2483111728642888_8708744769070694400_n.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\VoAnhKiet\WIN_20220501_16_34_59_Pro.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\VoAnhKiet\WIN_20220501_17_21_13_Pro.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\VoAnhKiet\WIN_20220501_17_21_20_Pro.jpg
Processing image C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe\database\VoAnhKiet\WIN_20220501_17_21_26_Pro.jpg
Saving the pickle file
Finished
PS C:\Users\ACER\Desktop\testingDoAnMatMa\Caffe>
```

## Recognize and verify



### 4.2. Test

| | HaarCascade | Caffe | Paper1 |
|---|---|---|---|
| Recognize in normal light | Pass | Pass | Pass |
| Recognize a part of face | Pass | Pass | Not mentioned |
| Detect fake face | Not Pass | Not Pass | Pass |
| Recognize in low light | Not Pass | Not Pass | Not mentioned |
| Respone time counter | 2.0052 second | 2.2138 second | 0.3496 second |
| Accuracy | 46.8547% | 71.8954% | 99.95% |

### 4.3. Review

This project provides the simple functions and features to register and verify the people with ficial recognition. In this project, people can regist with the pictures and verify with the video stream. Despite the verified testcase, it still faces a security problem and user experience.

# Part 5: Conclusion and Future Work

## 5.1. Conclusion

In this project, we demonstrated another authentication method using facial recognition and authentication with HaarCascade and Caffe (powered by ResNet). The program works well with a basic function for users to register and verify.

Due to the latest technology, we faced many problems due to our lack of knowledge about face recognition as well as multithreaded programming in Python. Besides, facing difficulties in developing and deploying Azure Function is also a challenge for us.

Although our project is not excellent enough for real-world application because of security issues with facial recognition, it is the foundation for development in the next steps. So, we will continue to develop it after this project.

During the implementation of this project, we had the opportunity to review and update the knowledge that we had acquired during our 2 years of study at the university. Furthermore, we have also gained valuable insights into new technology trends.

## 5.2. Future work

Various adaptations, test, and experiments were left for the future due to lack of time. Future work issues extend to more advanced features

- Develop the mobile app and web app allowing user to have more methods in registering and verifying
- Implement the program in the IOT device such as Raspberry Pi
- Build the training data system using the larger data to improve the register and verify system
- Improve the register and verify system algorithm to pop up the accuracy of the system

## Reference


1. Jong-Hyuk Im , Seong-Yun Jeon, & Mun-Kyu Lee (2020). Practical Privacy-Preserving Face Authentication for Smartphones Secure Against Malicious Clients. IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 15, 2020 (2386-2401)
2. Adrian Rosebrock, Face detection with OpenCV and deep learning, 2018, https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/
3. Chi-Feng Wang, What's the Difference Between Haar-Feature Classifiers and Convolutional Neural Networks? 2018, https://towardsdatascience.com/whats-the-difference-between-haar-feature-classifiers-and-convolutional-neural-networks-ce6828343aeb
4. OpenCV library documentation, https://opencv.org/
5. Ramiz Raja, Face Detection Using Opencv And Python: A Beginner's Guide, 2017 https://www.superdatascience.com/opencv-face-detection/
6. Renan Dias, The 5 Factors of Authentication, 2017, https://medium.com/@renansdias/the-5-factors-of-authentication-bcb79d354c13