Student Name: Thanh Tam Vo
Student ID: 103487596

COS30082 - Applied Machine Learning
Face Recognition Attendance System with Liveness Detection

Contents

1 Introduction

The project is Attendance System with Liveness Detection. Therefore, there are several problems that need to be solved:

- Face Detecting: locating the human face
- Verifying that whether the person is physically present
- Face Classification: Identifying individuals from a set of known faces

Releasing a product, which satisfies 3 criteria above, is challenging. It requires a lot of computing resources and infrastructure to ensure the high performance and smooth operation. Currently, I have already deployed an application that can detect the face of the person using their laptop webcam as Figure 1. In this report, I will only cover the methodology and result for liveness detection and face classification tasks. Finally, the link to the final product is provided in the last section.

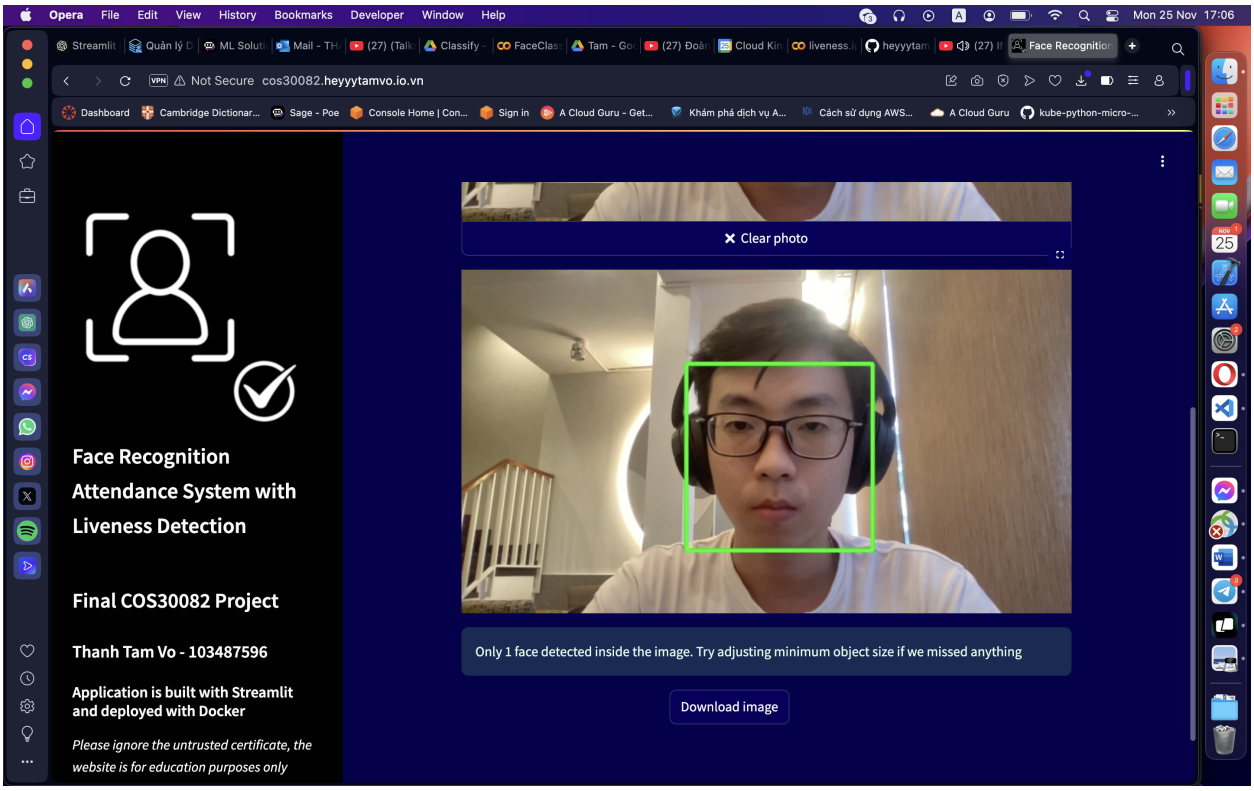

Figure 1: My Web Application for Face Detecting

2 Methodology

2.1 Dataset

The input for two models (Liveness and Classification) is an image with the size of $(224, 224)$. Before the training process, we need to scale the entire dataset. Below is the code base to scale the dataset.

Listing 1: Scale the preprocessing the dataset before training

```
import tensorflow as tf

# Define image size and other constants
real_path = '/content/drive/My Drive/COS30082/Assignment2/Liveness/Real'
fake_path = '/content/drive/My Drive/COS30082/Assignment2/Liveness/Fake'
IMAGE_SIZE=224
resized_real_path = '/content/drive/My Drive/COS30082/Assignment2/Liveness/Resized/Real'
resized_fake_path = '/content/drive/My Drive/COS30082/Assignment2/Liveness/Resized/Fake'

# Loop through all files in the source directory
for filename in os.listdir(real_path):
    if filename.endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif')):  # Check for image file
        extensions
        # Open an image file
        with Image.open(os.path.join(real_path, filename)) as img:
            # Resize the image to IMAGE_SIZE x IMAGE_SIZE
            img = img.resize((IMAGE_SIZE, IMAGE_SIZE))
            # Save the image to the output directory with the same name
            img.save(os.path.join(resized_real_path, filename))

for filename in os.listdir(fake_path):
    if filename.endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif')):  # Check for image file
        extensions
        # Open an image file
        with Image.open(os.path.join(fake_path, filename)) as img:
            # Resize the image to IMAGE_SIZE x IMAGE_SIZE
            img = img.resize((IMAGE_SIZE, IMAGE_SIZE))
            # Save the image to the output directory with the same name
            img.save(os.path.join(resized_fake_path, filename))
```

2.2 Image augmentation

To ensure the diversity of the training data, I applied the Image augmentation technique before training to increase the dataset size. This technique will help us to avoid overfitting situation as much as possible. Below is the code base showing my Data augmentation technique for this project.

Listing 2: Applying Image Augmentation Technique

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 20,
    width_shift_range = 0.1,
    height_shift_range = 0.2,
    horizontal_flip = True,
    shear_range = 0.2,
    zoom_range = 0.2,
    fill_mode = 'nearest'
)

validation_datagen = ImageDataGenerator(
    rescale = 1./255,
)

train_generator = train_datagen.flow_from_directory(
```

```
    DATASET_PATH,
    target_size = (IMAGE_SIZE, IMAGE_SIZE),
    class_mode = 'binary'
)
```

2.3 Model Architecture

Before going further, I need to declare: Due to computing resources, the Face Classification only only recognizes my face, meaning it can determine whether the detected face is mine or not.

I decided to use the same architecture for 2 models as Figure 2 below:

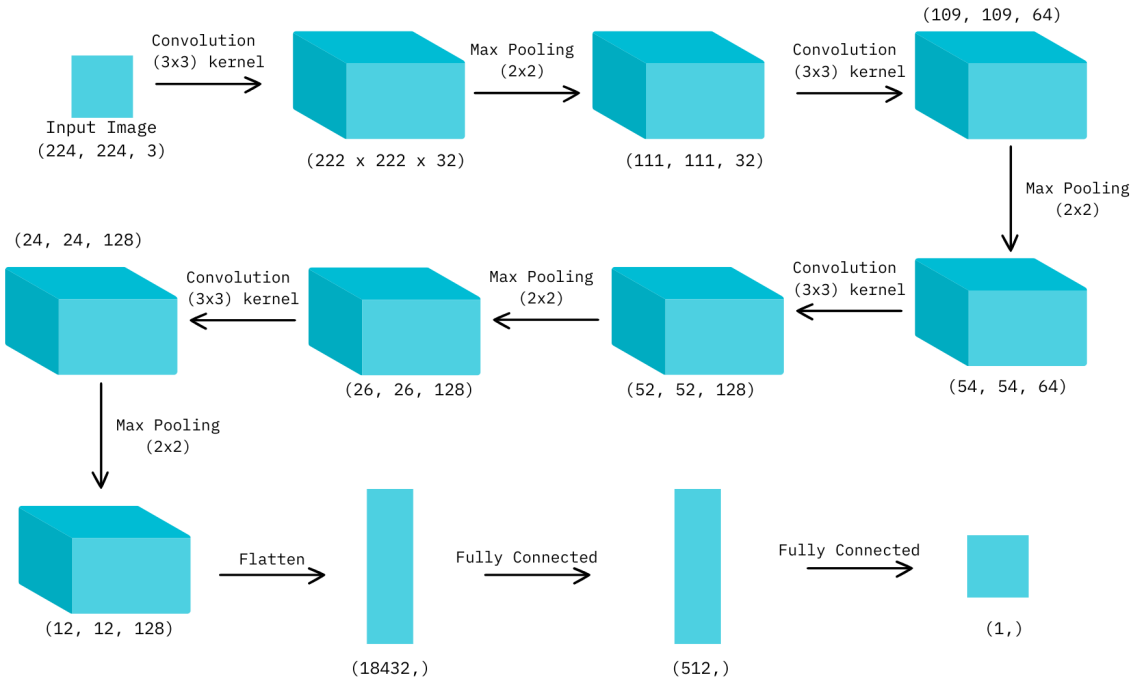

Figure 2: Model Architecture

The output from two models is a probability value: if the value is close to 1, it indicates that the detected face is likely to be mine. In order to obtain the weight for the model, I used Adam Optimizer Algorithm and Binary Cross Entropy as the Loss Function.

3 Result

3.1 Loss and Accuracy

Within 32 epochs, Figure 3 and 4 are the evaluation.

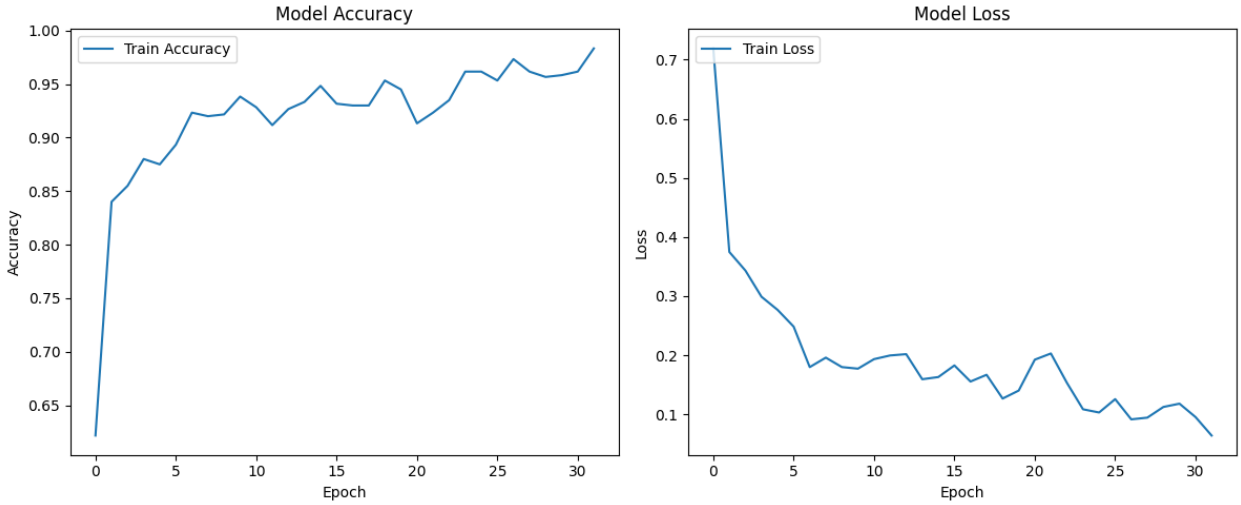

Figure 3: Liveness Model Evaluation

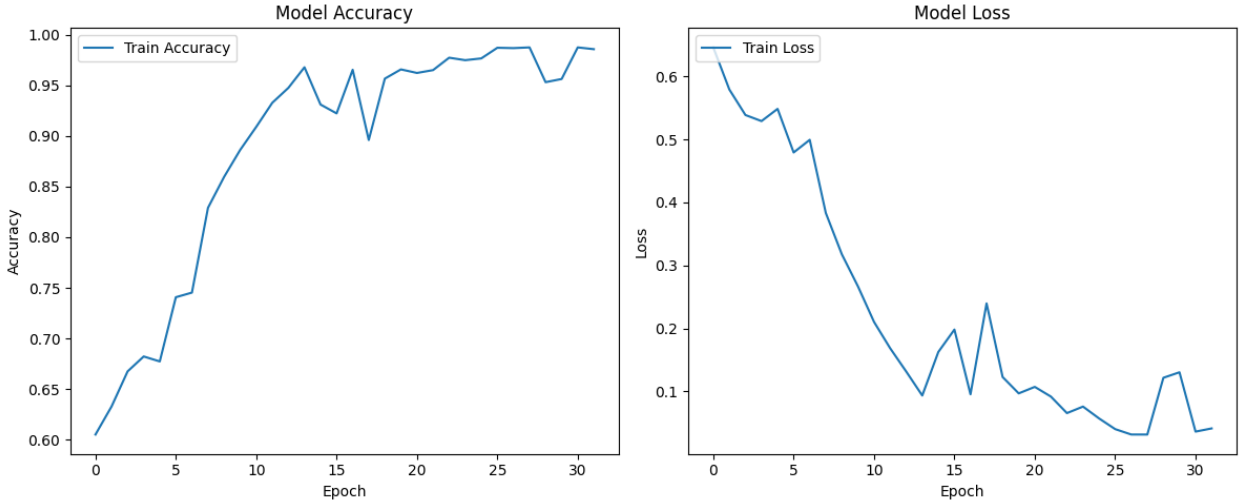

Figure 4: Face Classification Model Evaluation

3.2 Testing the Model

3.2.1 Liveness Model

To test this model, I used two images as Figure 5 and 6 below.

Figure 5: The image with my real face (`tam_real_crop.png`)

Figure 6: The image with my spoofed face (`tam_fake_crop.png`)

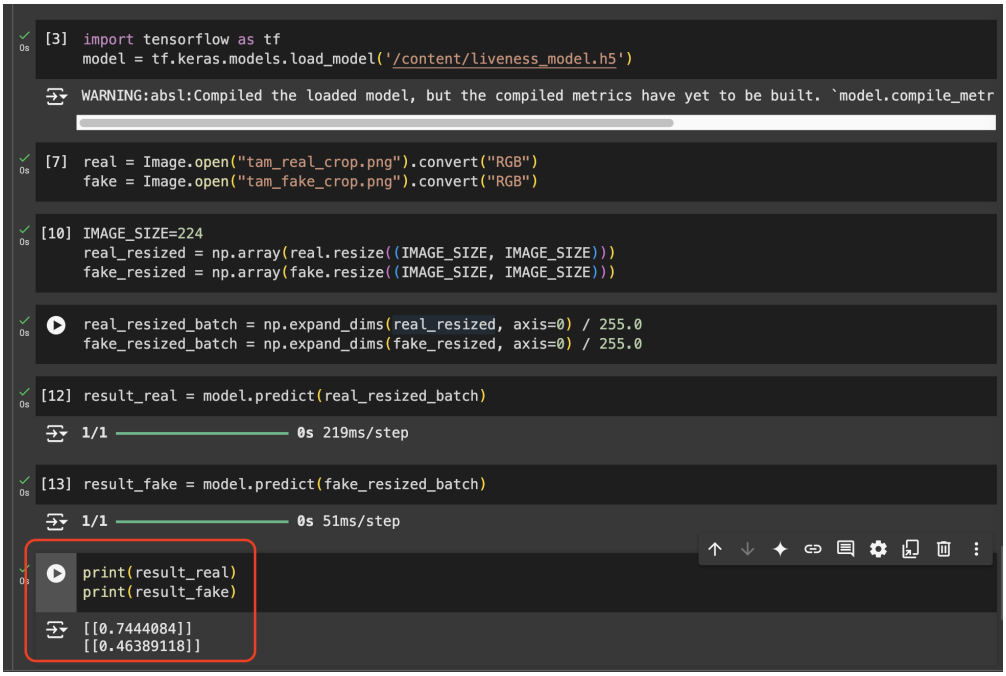

Figure 7: Testing Result for Liveness Model (`liveness.ipynb`)

3.2.2 Face Classification Model

To test this model, I used two images as Figure 8 and 9 below.

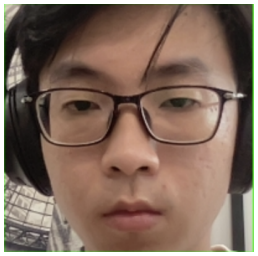

Figure 8: The image with my real face (`tam.png`)

Figure 9: The image without my face (`obama.png`)

```
[4]  import tensorflow as tf
     model = tf.keras.models.load_model("/content/classification.h5")

     WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metr

[6]  from PIL import Image
     import numpy as np

     IMAGE_SIZE = 224
     real = Image.open("tam.png").convert("RGB")
     real_resized = np.array(real.resize((IMAGE_SIZE, IMAGE_SIZE)))
     real_resized_batch = np.expand_dims(real_resized, axis=0) / 255.0
     result_real = model.predict(real_resized_batch)
     print(result_real)

     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 212ms/step
     [[0.99192774]]

     real = Image.open("obama.png").convert("RGB")
     real_resized = np.array(real.resize((IMAGE_SIZE, IMAGE_SIZE)))
     real_resized_batch = np.expand_dims(real_resized, axis=0) / 255.0
     result_real = model.predict(real_resized_batch)
     print(result_real)

     1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
     [[3.672695e-16]]
```

Figure 10: Testing Result for Classification Model (`FaceClassify.ipynb`)

3.3 Production Deployment

Please refer to this link to experience the production. There will be a warning before access the website because of untrusted certificate. Please ignore the untrusted certificate, this website is for education purposes only. To do that, please click on "Advanced" then "Proceed" to the website.

Note: Using Google Chrome for better experience

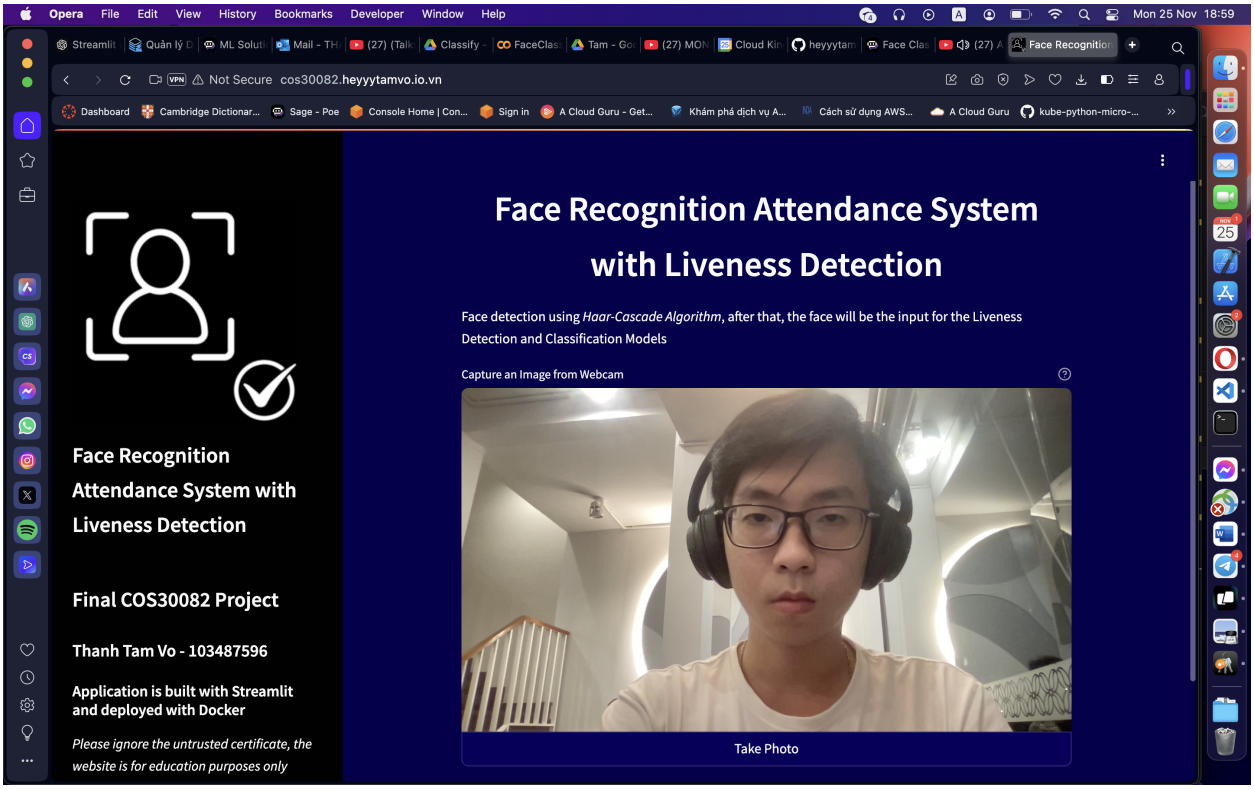

Figure 11: Face Detection Web Application

4 Submission Materials

Submission Materials will be found out via this link including:

- Liveness Model
- Classification Model
- Jupyter Notebook files
- Testing Images