TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

KHOA CƠ KHÍ CHẾ TẠO MÁY

~~~~~~*~~~~~~

BÁO CÁO CUỐI KỲ

MÔN HỌC: TRÍ TUỆ NHÂN TẠO

GVHD: Thầy Nguyễn Trường Thịnh

Sinh viên thực hiện	:	Nguyễn Phúc Dũng
MSSV	:	20146486
Lớp	:	ARIN337629_Nhóm09 (Thứ 5,tiết 1-4)
Giảng viên hướng dẫn	:	Thầy Nguyễn Trường Thịnh

TP.HCM – 2023
~~~~~~

CHƯƠNG 1: GIỚI THIỆU CHUNG

1.1. Giới thiệu chung về ngành AI nói chung

Trí tuệ nhân tạo hay trí thông minh nhân tạo (Artificial Intelligence – viết tắt là AI) là một ngành thuộc lĩnh vựa khoa học máy tính (Computer Science). Là trí tuệ do con người lập trình tạo nên với mục tiêu giúp máy tính có thể tự động hóa các hành vi thông minh như con người. Cụ thể ta có thể thấy trí tuệ nhân tạo trong những ngôi nhà thông minh giúp kiểm soát các thông số cũng như ra lệnh cho các thiết bị khi nhận lệnh từ con người và nó cũng có thể đề xuất những các thức tối ưu cho tình huống cụ thể. Hay ta cũng có thể thấy AI ứng dụng trong việc nhận diện.

Trong bài này ta sẽ tiến hành sử dụng mô hình mạng Convolution Neural Network (CNN) để tiến hành huấn luyện các model có thể nhận diện được độ tuổi và giới tính của con người thông qua các phương tiện ảnh và camera theo thời gian thực.

1.2. Mục tiêu nghiên cứu

Xây dựng mô hình huấn luyện các model có thể nhận diện được khuôn mặt, độ tuổi và giới tính của con người để từ đó có thể thu thập được các dữ liệu người dùng và đưa ra các giải pháp công nghệ giúp đỡ cho đời sống con người trở nên dễ dàng hơn. Cụ thể là những công nghệ này cung cấp các giải pháp cửa hàng trực tuyến và cửa hàng thực, phương pháp tiếp thị, cải tiến dịch vụ và phát triển sản phẩm hiệu quả cao.

1.3. Phương pháp nghiên cứu

- Tìm hiểu các phương pháp hiện có thông qua Github
- Tìm kiếm khối lượng Data thông qua Google, Kaggle, hình ảnh thực tế,…
- Xây dựng “Model Training” và tiến hành chạy thử để kiểm chứng độ chính xác
- Xây dựng webapp thân thiện với người dùng

1.4. Nội dung báo cáo

Bài báo cáo tập trung vào việc nghiên cứu phương pháp tối ưu để tạo model train bằng CNN sao cho độ chính xác đạt cao nhất. Từ đó đưa ra các kết quả nhận diện với độ chính xác cao khoảng 94%.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ PHƯƠNG PHÁP THỰC HIỆN

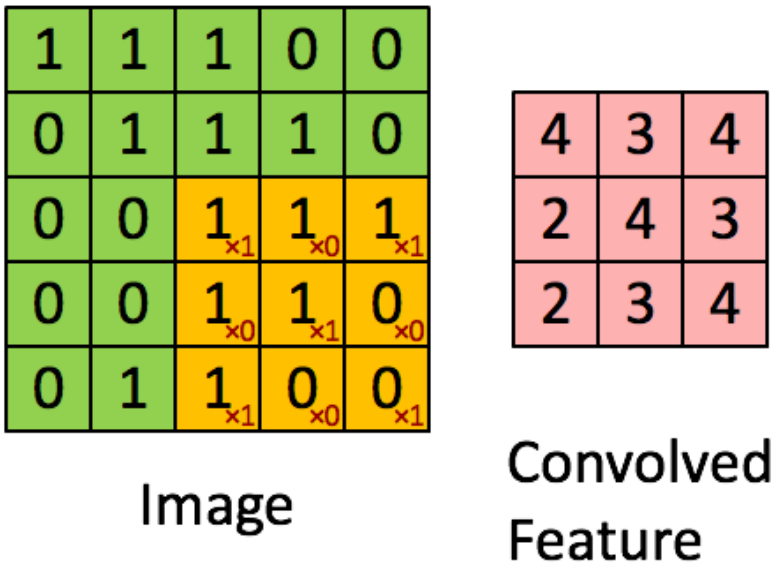
2.1. Thuật toán CNN - Convolutional Neural Network

2.1.1. Convolutional Neural Network là gì

Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh. Để tìm hiểu tại sao thuật toán này được sử dụng rộng rãi cho việc nhận dạng (detection).

Convolutional là một cửa sổ trượt (Sliding Windows) trên một ma trận (xem hình)



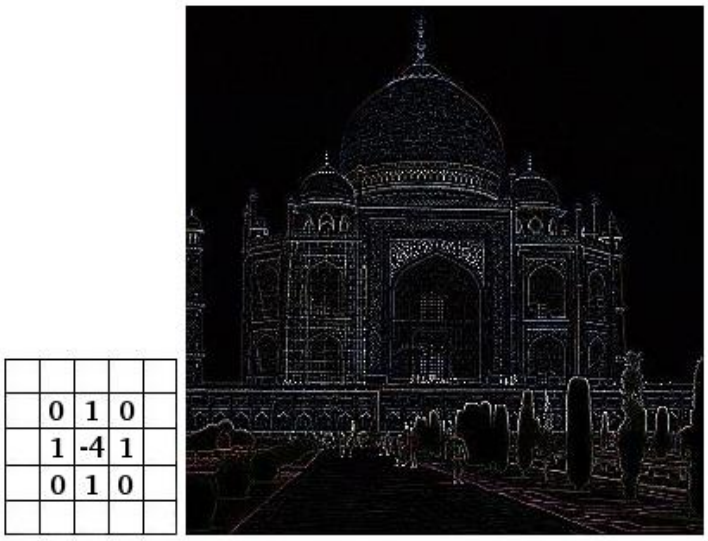
Hình 2.1. Ma trận Convolutional

Các convolutional layer có các parameter(kernel) đã được học để tự điều chỉnh lấy ra những thông tin chính xác nhất mà không cần chọn các feature.

Trong hình ảnh trên, ma trận bên trái là một hình ảnh trắng đen được số hóa. Ma trận có kích thước 5×5 và mỗi điểm ảnh có giá trị 1 hoặc 0 là giao điểm của dòng và cột.

Convolution hay tích chập là nhân từng phần tử trong ma trận 3. Sliding Window hay còn gọi là kernel, filter hoặc feature detect là một ma trận có kích thước nhỏ như trong ví dụ trên là 3×3.

Convolution hay tích chập là nhân từng phần tử bên trong ma trận 3×3 với ma trận bên trái. Kết quả được một ma trận gọi là Convoled feature được sinh ra từ việc nhận ma trận Filter với ma trận ảnh 5×5 bên trái.



	0	1	0	
	1	-4	1	
	0	1	0	

Hình 2.2. Hình ảnh trắng đen được số hóa

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Hình 2.3. Hình ảnh được Convoled feature

2.1.2. Cấu trúc của mạng CNN

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

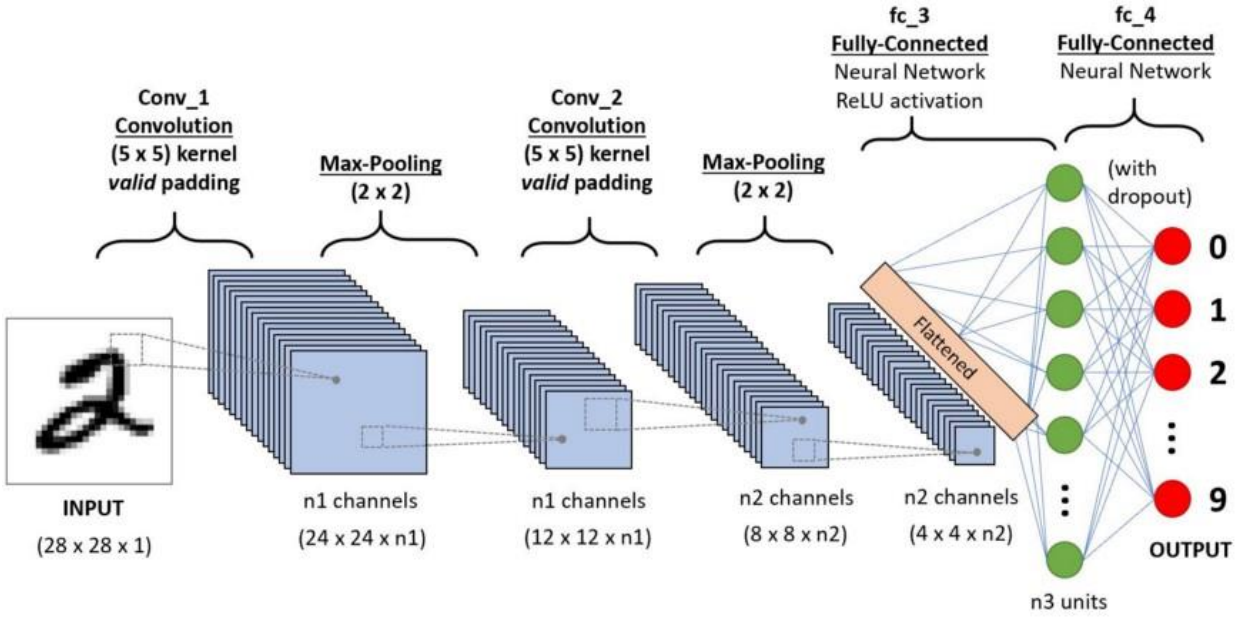
Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution.

Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong quá trình huấn luyện mạng (traning) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



Hình 2.4. Cấu trúc mạng CNN

Trong mô hình CNN có 2 khía cạnh cần quan tâm là tính bất biến (Location Invariance) và tính kết hợp (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các gốc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể.

Pooling layer sẽ cho bạn tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter. Đó là lý do tại sao CNNs cho ra mô hình với độ chính xác rất cao. Cũng giống như cách con người nhận biết các vật thể trong tự nhiên.

2.1.3. Cách thức hoạt động của các lớp trong mạng CNN

Cấu trúc của CNN bao gồm các lớp sau:

- *Lớp nhập liệu (Input Layer)*: Lớp này nhận dữ liệu đầu vào là ảnh hoặc bất kỳ đối tượng nào có kích thước đầu vào xác định.
- *Lớp tích chập (Convolutional Layer)*: Lớp này sử dụng các bộ lọc tích chập (convolutional filter) để trích xuất các đặc trưng từ ảnh đầu vào. Mỗi bộ lọc sẽ di chuyển trên toàn bộ ảnh và thực hiện phép tích chập để tạo ra một feature map mới.

- *Lớp kích hoạt (Activation Layer)*: Lớp này sử dụng một hàm kích hoạt như ReLU (Rectified Linear Unit) để giúp mạng học được các đặc trưng phi tuyến tính.
- *Lớp giảm mẫu (Pooling Layer)*: Lớp này được sử dụng để giảm kích thước của feature map bằng cách lấy giá trị trung bình hoặc giá trị lớn nhất của một vùng nhỏ của feature map.
- *Lớp bỏ (Dropout Layer)*: Lớp này sẽ ngẫu nhiên loại bỏ một số lượng neuron để tránh overfitting.
- *Lớp kết nối đầy đủ (Fully Connected Layer)*: Lớp này sử dụng các neuron để kết nối đến tất cả các neuron trong lớp trước đó và tính toán ra các kết quả cuối cùng của mạng.
- *Lớp đầu ra (Output Layer)*: Lớp này đưa ra kết quả dự đoán cho mạng CNN.

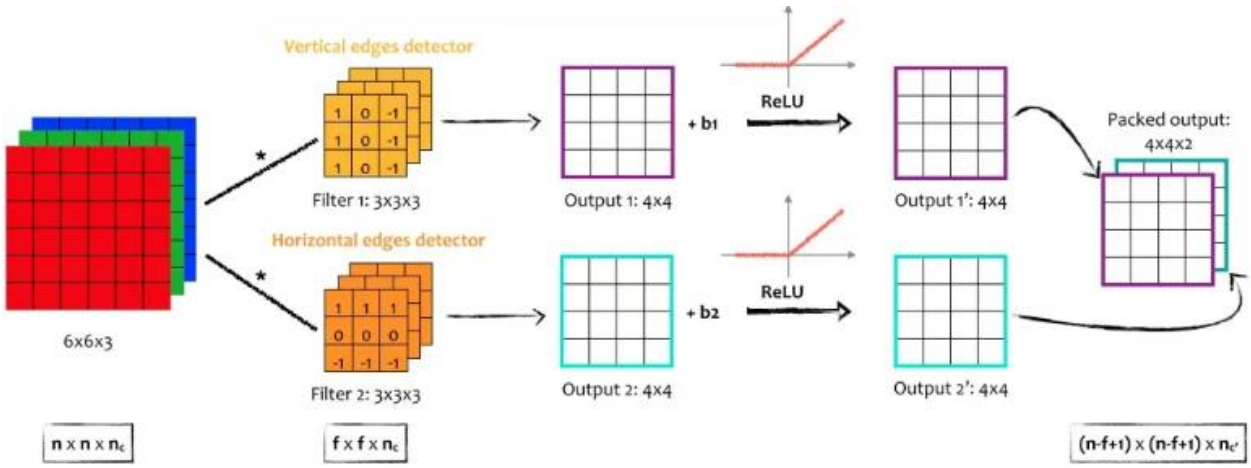
Khi được huấn luyện đầy đủ, mạng CNN có khả năng tự động nhận diện các đối tượng trong ảnh hoặc video, và thực hiện các tác vụ như phân loại ảnh, nhận diện khuôn mặt, phân tích tín hiệu, và nhiều ứng dụng khác.

2.2. Phương pháp thực hiện

2.2.1. Cách chọn tham số cho CNN

Nhằm lựa chọn được tham số phù hợp nhất cho CNN, nên lưu ý đến số lượng các yếu tố sau đây: Filter size, pooling size, số convolution và số lần train test.

- **Convolution layer:** Nếu lớp này có số lượng lớn hơn,chương trình chạy sẽ càng được cải thiện và tiến bộ. Sử dụng layer với số lượng nhiều có thể giúp các tác động được giảm một cách đáng kể. Trong đa phần các trường hợp, chỉ cần khoảng 3 đến 5 lớp là sẽ thu về kết quả như mong đợi.
- **Filter size:** Thông thường, các filter size sẽ có kích thước là 3×3 hoặc 5×5.
- **Pooling size:** Với các hình ảnh thông thường, nên sử dụng loại kích thước 2×2. Nếu đầu vào xuất hiện dạng hình ảnh lớn hơn thì nên chuyển sang dùng loại 4×4.
- **Train test:** Càng thực hiện nhiều lần train test, càng có nhiều khả năng thu được các parameter tốt nhất, giúp mô hình "thông minh" và hiệu quả hơn.



Hình *2.5*. Cách chọn tham số cho mạng CNN

CHƯƠNG 3: XÂY DỰNG MÔ HÌNH

3.1. Xây dựng mô hình trainning trên Google Colab cho nhận dạng độ tuổi

3.1.1. Khai báo và sử dụng các thư viện cần thiết

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
import os
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv2D, MaxPooling2D,
Flatten, BatchNormalization
from keras.optimizers import Adam
from keras.layers import Activation
from os import listdir
from numpy import asarray,save
from keras.utils import load_img, img_to_array
import random
from keras import backend as K
from keras.optimizers import Adam
```

3.1.2. Thay đổi tỉ lệ ảnh và tăng cường data

```
train = ImageDataGenerator(rescale =1.0/255.0,
                    rotation_range=25, width_shift_range=0.1,
                    height_shift_range=0.1, shear_range=0.2,
                    zoom_range=0.2,vertical_flip=True,
                    horizontal_flip=True, fill_mode="nearest")
test = ImageDataGenerator(rescale=1.0/255.0,
                    rotation_range=25, width_shift_range=0.1,
                    height_shift_range=0.1, shear_range=0.2,
                    zoom_range=0.2,vertical_flip=True,
                    horizontal_flip=True, fill_mode="nearest")
```

3.1.3. Tìm các ảnh và tự động chia thành các lớp trong đường dẫn

```
train_data = train.flow_from_directory('/content/drive/MyDrive/Gender
        and Age Detection Final Project Data/Age Dataset Main Final
Project/Age Detection', target_size=(96,96), class_mode='categorical')
test_data = test.flow_from_directory('/content/drive/MyDrive/Gender
        and Age Detection Final Project Data/Age Dataset Main Final
Project/Storage Test', target_size=(96,96), class_mode='categorical')
print('train_index :', train_data.class_indices)
print('test_index:', test_data.class_indices)
```

```
Found 5080 images belonging to 10 classes.
Found 100 images belonging to 10 classes.

train_index : {'eightyone_to_ninety': 0, 'eleven_to_twenty': 1,
'fiftyone_to_sixty': 2, 'fourtyone_to_fifty': 3,
'ninetyone_to_onehundred': 4, 'seventyone_to_eighty': 5,
'sixtyone_to_seventy': 6, 'thirtyone_to_fourty': 7,
'twentyone_to_thirdty': 8, 'zero_to_ten': 9}


test_index: {'eightyone_to_ninety': 0, 'eleven_to_twenty': 1,
'fiftyone_to_sixty': 2, 'fourtyone_to_fifty': 3,
'ninetyone_to_onehundred': 4, 'seventyone_to_eighty': 5,
'sixtyone_to_seventy': 6, 'thirtyone_to_fourty': 7,
'twentyone_to_thirdty': 8, 'zero_to_ten': 9}
```

3.1.4. Tạo ra mạng CNN để train mô hình

```
model= Sequential()
chanDim = -1
if K.image_data_format() == "channels_first": #Returns a
string, either 'channels_first' or 'channels_last'
    chanDim = 1
#CNN1
model.add(Conv2D(32, kernel_size=(3,3),
activation='relu',input_shape=(96,96,3), padding='same'))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3,3), padding='same'))
model.add(Dropout(0.25))
#CNN2
model.add(Conv2D(64,(3,3),activation='relu', padding='same'))
model.add(BatchNormalization(axis=chanDim))
#CNN3
model.add(Conv2D(128,(3,3),activation='relu', padding='same'))
```

```
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2,2), padding='same'))
model.add(Dropout(0.25))
#CNN4
model.add(Conv2D(128, kernel_size=(3,3), activation='relu',
padding="same"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2,2), padding='same'))
model.add(Dropout(0.25))

# initial parameters
epochs = 100
lr = 1e-3
batch_size = 64
img_dims = (96,96,3)
```

3.1.5. Lớp Fully Connected Layers

```
# Fully connected layers
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(classes))
model.add(Activation("sigmoid"))

model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_14 (Conv2D)          (None, 96, 96, 32)        896

 batch_normalization_17 (Bat  (None, 96, 96, 32)       128
 chNormalization)
```

```
max_pooling2d_11 (MaxPoolin  (None, 32, 32, 32)       0
g2D)

dropout_14 (Dropout)         (None, 32, 32, 32)       0

conv2d_15 (Conv2D)           (None, 32, 32, 64)       18496

batch_normalization_18 (Bat  (None, 32, 32, 64)       256
chNormalization)

conv2d_16 (Conv2D)           (None, 32, 32, 128)      73856

batch_normalization_19 (Bat  (None, 32, 32, 128)      512
chNormalization)

max_pooling2d_12 (MaxPoolin  (None, 16, 16, 128)      0
g2D)

dropout_15 (Dropout)         (None, 16, 16, 128)      0

conv2d_17 (Conv2D)           (None, 16, 16, 128)      147584

batch_normalization_20 (Bat  (None, 16, 16, 128)      512
chNormalization)

max_pooling2d_13 (MaxPoolin  (None, 8, 8, 128)        0
g2D)

dropout_16 (Dropout)         (None, 8, 8, 128)        0

flatten_3 (Flatten)          (None, 8192)             0

dense_6 (Dense)              (None, 1024)             8389632
```

```
activation_6 (Activation)    (None, 1024)              0
batch_normalization_21 (Bat  (None, 1024)              4096
chNormalization)

dropout_17 (Dropout)         (None, 1024)              0

dense_7 (Dense)              (None, 10)                10250

activation_7 (Activation)    (None, 10)                0

=================================================================
Total params: 8,646,218
Trainable params: 8,643,466
Non-trainable params: 2,752
```

3.1.6. Biên dịch và training cho mô hình

```
opt = Adam(learning_rate=lr, decay=lr/epochs)
model.compile(loss='binary_crossentropy', optimizer=opt,
              metrics=['accuracy'])

his=model.fit(train_data,epochs=epochs,batch_size=128,verbose=1,
              validation_data=test_data)
value=model.evaluate(train_data,verbose=0)
print('loss', value[0])
print('accuracy', value[1])
```

```
Epoch 100/100
159/159 [==============================] - 26s 161ms/step - loss: 0.1903 -
accuracy: 0.8482 - val_loss: 0.1719 - val_accuracy: 0.8600
loss 0.1894519031047821
accuracy 0.8450787544250488
```

3.1.7. Lưu lại model

```
model.save('/content/drive/MyDrive/Model Save/Age_Detection.h5')
```

3.2. Xây dựng mô hình trainning trên Google Colab cho nhận dạng giới tính

3.2.1. Khai báo và sử dụng các thư viện cần thiết

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
import os
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv2D, MaxPooling2D,
Flatten, BatchNormalization
from keras.optimizers import Adam
from keras.layers import Activation
from os import listdir
from numpy import asarray,save
from keras.utils import load_img, img_to_array
import random
from keras import backend as K
from keras.optimizers import Adam
```

3.2.2. Thay đổi tỉ lệ ảnh và tăng cường data

```
train = ImageDataGenerator(rescale =1.0/255.0,
                    rotation_range=25, width_shift_range=0.1,
                    height_shift_range=0.1, shear_range=0.2,
                    zoom_range=0.2,vertical_flip=True,
                    horizontal_flip=True, fill_mode="nearest")
test = ImageDataGenerator(rescale=1.0/255.0, rotation_range=25,
                          width_shift_range=0.1,
                          height_shift_range=0.1,
                          shear_range=0.2,
                          zoom_range=0.2,vertical_flip=True,
                          horizontal_flip=True,fill_mode="nearest")
```

3.2.3. Tìm các ảnh và tự động chia thành các lớp trong đường dẫn

```
train_data = train.flow_from_directory('/content/drive/MyDrive/Gender
         and Age Detection Final Project Data/Gender Dataset Main Final
         Project/train', target_size=(96,96), class_mode='categorical')
test_data = test.flow_from_directory('/content/drive/MyDrive/Gender and
           Age Detection Final Project Data/Gender Dataset Main Final
          Project/test', target_size=(96,96), class_mode='categorical')
print('train_index :', train_data.class_indices)
print('test_index:', test_data.class_indices)
```

```
Found 5645 images belonging to 2 classes.
Found 346 images belonging to 2 classes.
train_index : {'man': 0, 'woman': 1}
test_index: {'man': 0, 'woman': 1}
```

3.2.4. Tạo ra mạng CNN để train mô hình

```
model= Sequential()
chanDim = -1
if K.image_data_format() == "channels_first": #Returns a string,
either 'channels_first' or 'channels_last'
    chanDim = 1
#CNN1
model.add(Conv2D(32, kernel_size=(3,3),
activation='relu',input_shape=(96,96,3), padding='same'))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3,3), padding='same'))
model.add(Dropout(0.25))
#CNN2
model.add(Conv2D(64,(3,3),activation='relu', padding='same'))
model.add(BatchNormalization(axis=chanDim))
#CNN3
model.add(Conv2D(128,(3,3),activation='relu', padding='same'))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2,2), padding='same'))
model.add(Dropout(0.25))
#CNN4
model.add(Conv2D(128, kernel_size=(3,3), activation='relu',
padding="same"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2,2), padding='same'))
model.add(Dropout(0.25))
```

```
# initial parameters
epochs = 100
lr = 1e-3
batch_size=128
classes = 2
img_dims = (96,96,3)
```

3.2.5. Lớp Fully Connected Layers

```
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(classes))
model.add(Activation("sigmoid"))

model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_12 (Conv2D)          (None, 96, 96, 32)        896

 batch_normalization_20 (Bat  (None, 96, 96, 32)       128
 chNormalization)

 max_pooling2d_9 (MaxPooling  (None, 32, 32, 32)       0
 2D)

 dropout_17 (Dropout)        (None, 32, 32, 32)        0

 conv2d_13 (Conv2D)          (None, 32, 32, 64)        18496

 batch_normalization_21 (Bat  (None, 32, 32, 64)       256
 chNormalization)

 conv2d_14 (Conv2D)          (None, 32, 32, 128)       73856

 batch_normalization_22 (Bat  (None, 32, 32, 128)      512
 chNormalization)

 max_pooling2d_10 (MaxPoolin  (None, 16, 16, 128)      0
 g2D)
```

```
 dropout_18 (Dropout)         (None, 16, 16, 128)       0

 conv2d_15 (Conv2D)           (None, 16, 16, 128)       147584

 batch_normalization_23 (Bat  (None, 16, 16, 128)       512
 chNormalization)

 max_pooling2d_11 (MaxPoolin  (None, 8, 8, 128)         0
 g2D)

 dropout_19 (Dropout)         (None, 8, 8, 128)         0

 flatten_9 (Flatten)          (None, 8192)              0

 dense_17 (Dense)             (None, 1024)              8389632

 activation_16 (Activation)   (None, 1024)              0

 batch_normalization_24 (Bat  (None, 1024)              4096
 chNormalization)

 dropout_20 (Dropout)         (None, 1024)              0

 dense_18 (Dense)             (None, 2)                 2050

 activation_17 (Activation)   (None, 2)                 0

=================================================================
Total params: 8,638,018
Trainable params: 8,635,266
Non-trainable params: 2,752
_________________________________________________________________
```

3.2.6. Biên dịch và training cho mô hình

```
opt = Adam(learning_rate=lr, decay=lr/epochs)
model.compile(loss='binary_crossentropy', optimizer=opt,
metrics=['accuracy'])
his=model.fit(train_data,epochs=epochs,batch_size=128,
              verbose=1,validation_data=test_data)
value=model.evaluate(train_data,verbose=0)
print('loss', value[0])
print('accuracy', value[1])
```

```
Epoch 100/100
177/177 [==============================] - 31s 173ms/step - loss: 0.1623 -
accuracy: 0.9398 - val_loss: 0.1178 - val_accuracy: 0.9364
loss 0.14075706899166107
accuracy 0.9482728242874146
```

3.2.7. Lưu lại model

```
model.save('/content/drive/MyDrive/Model Save/Gender_Detection.h5')
```

3.3. Tiến hành dự đoán bằng hình ảnh trên Google Colab

3.3.1. Khai báo và sử dụng các thư viện cần thiết

```
from keras.models import load_model
import os
from os import listdir
from numpy import asarray,save
from keras.utils import load_img, img_to_array
import numpy as np
import matplotlib.pyplot as plt
```

3.3.2. Gán model vào biến

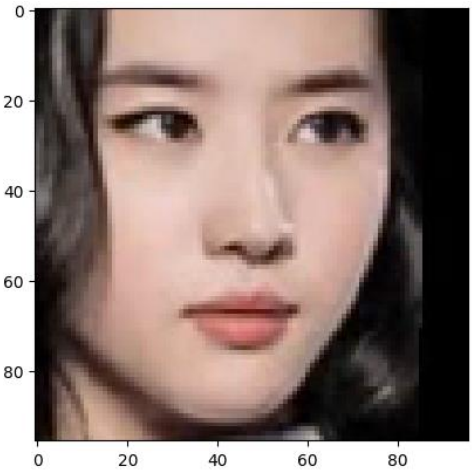
```
# load gender model save
gender_model= load_model('/content/drive/MyDrive/Model
                          Save/Gender_Detection.h5')
age_model = load_model('/content/drive/MyDrive/Model
                         Save/Age_Detection.h5')
```

3.3.3. Quét hết các hình trong đường dẫn và dự đoán giới tính và độ tuổi

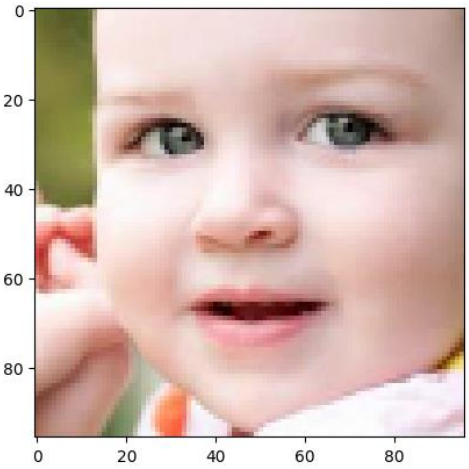
```
# Đường dẫn thư mục test
test="/content/drive/MyDrive/Gender and Age Detection Final
Project Data/Age Dataset Main Final Project/Identify_test_age"

# Tạo tên hiện của giới tính và độ tuổi
classes = ['man','woman']
age_classes = ['0-10','11-20','21-30','31-40','41-50',
               '51-60','61-70','71-80','81-90','91-120']
age_classes_reversed = list(reversed(age_classes))
# Quét hết các hình trong đường dẫn và tiến hành dự đoán các
loại tiền
for i in os.listdir(test):
  img=load_img(test+'/'+i,target_size=(96,96),
               color_mode = 'rgb')
  plt.imshow(img)

  img=img_to_array(img)
```

```
  img=img.astype('float32')
  img=img/255
  img=np.expand_dims(img,axis=0)
  result=(gender_model.predict(img).argmax())
  result1=(age_model.predict(img).argmax())

  print("Giới tính là {} và độ tuổi là
    {}".format(classes[result],age_classes_reversed[result1]))
  plt.show()
```

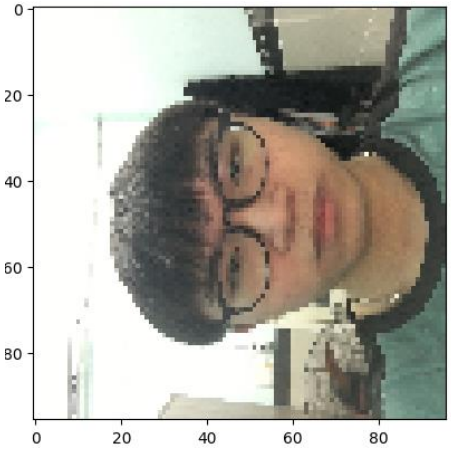
```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
Giới tính là woman và độ tuổi là 11-20
```



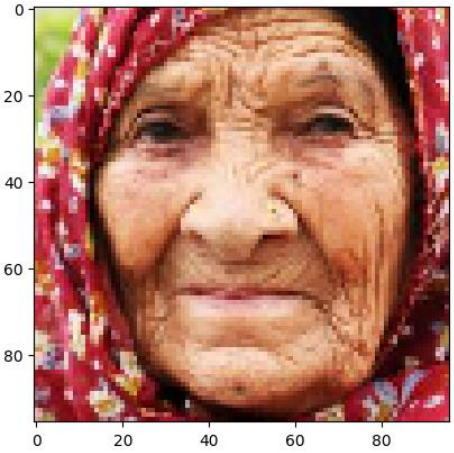

```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 21ms/step
Giới tính là woman và độ tuổi là 0-10
```



```
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
Giới tính là man và độ tuổi là 11-20
```

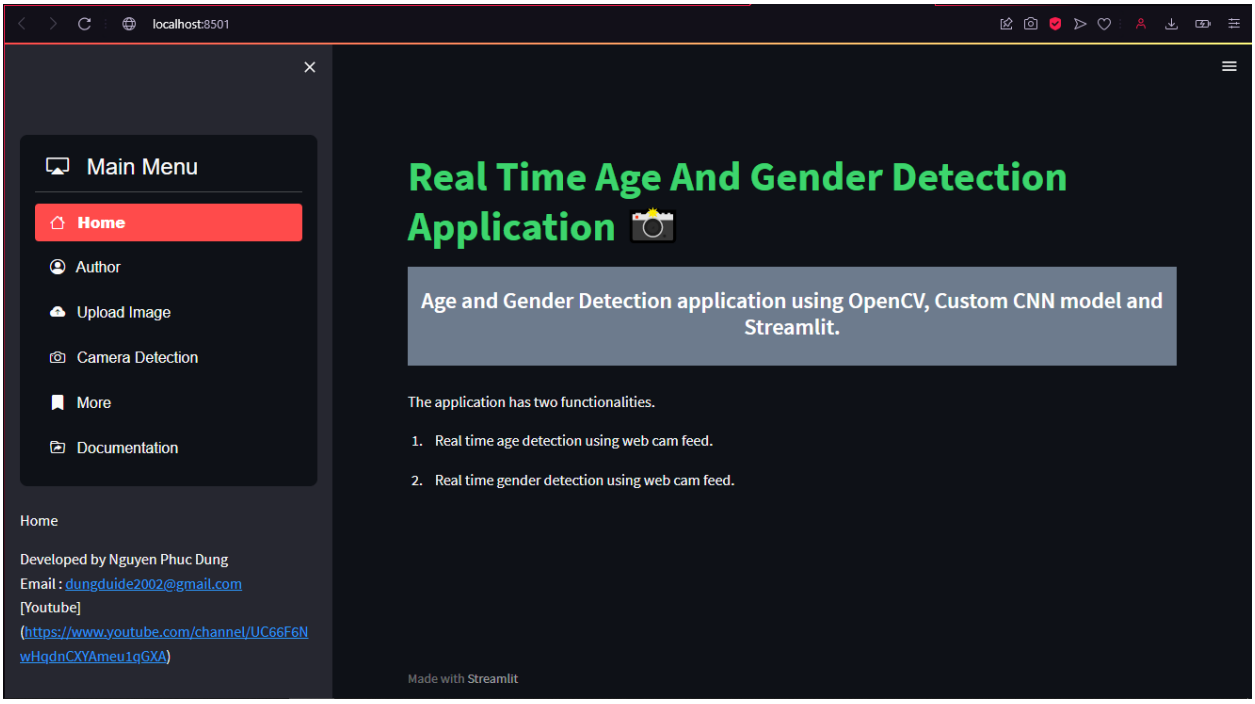


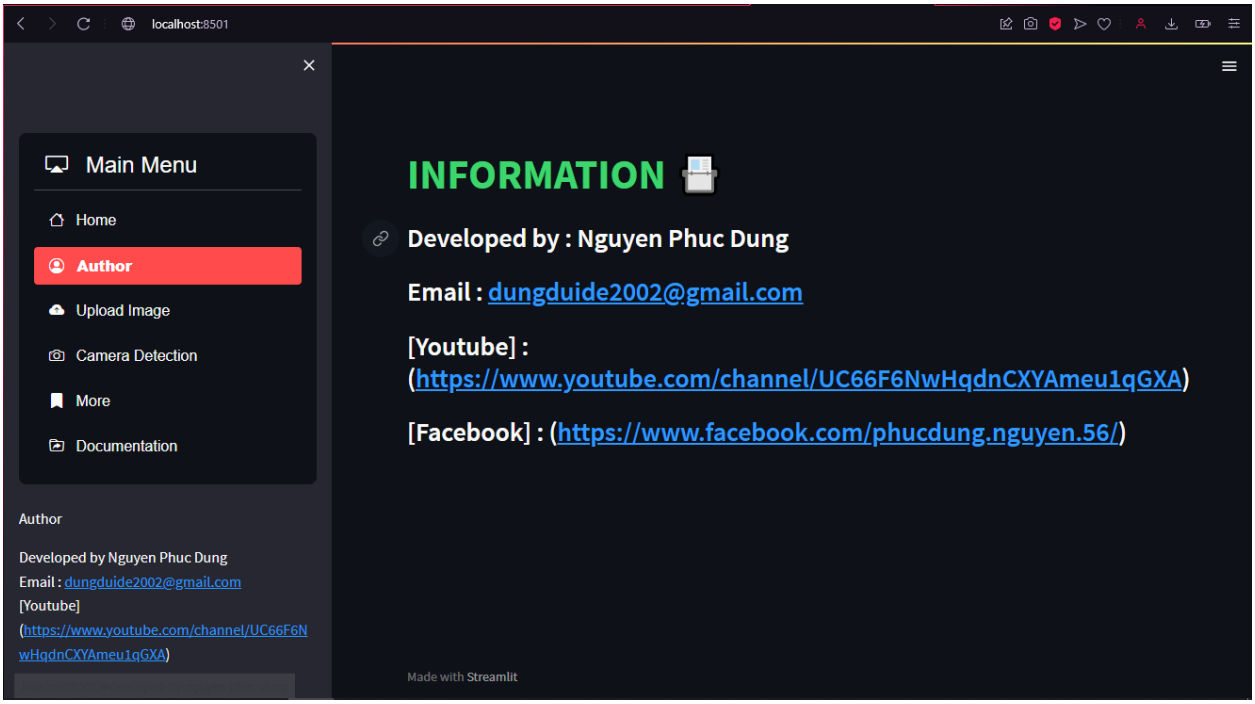

```
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 17ms/step
Giới tính là woman và độ tuổi là 91-120
```



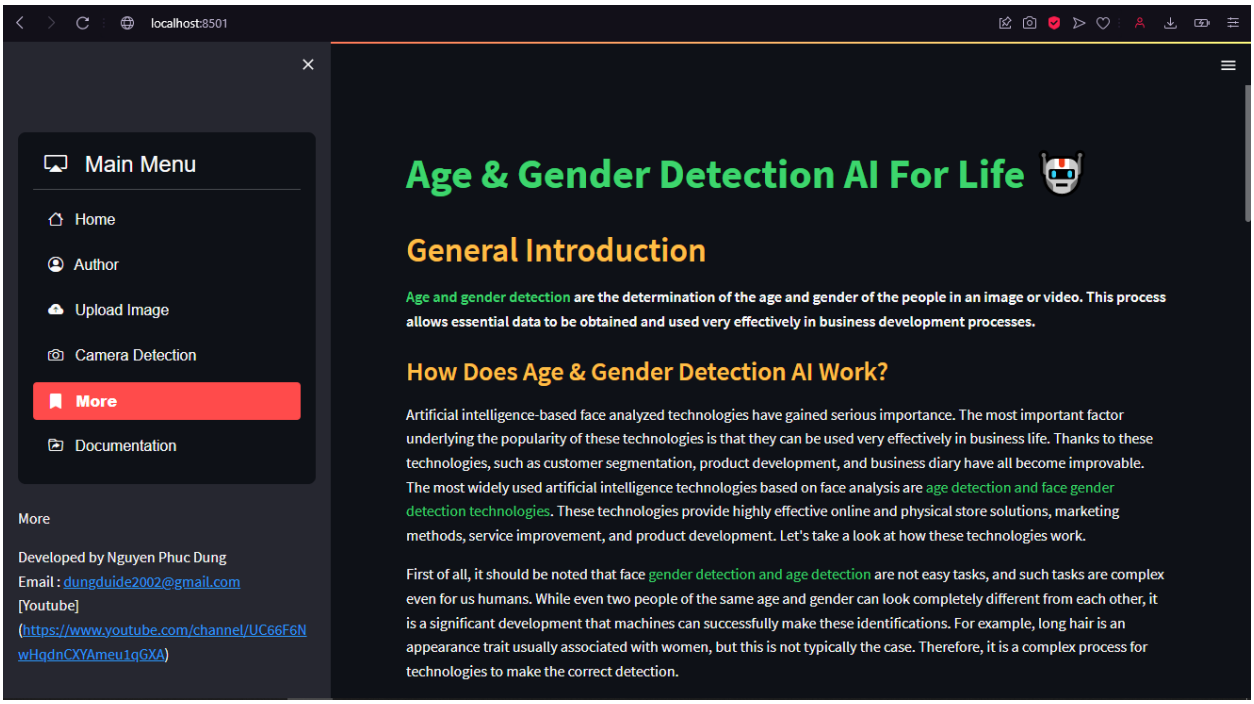
CHƯƠNG 4 : XÂY DỰNG WEBAPP VỚI STREAMLIT LIBRARY VÀ OPENCV

4.1. Giao diện tổng quát của webapp

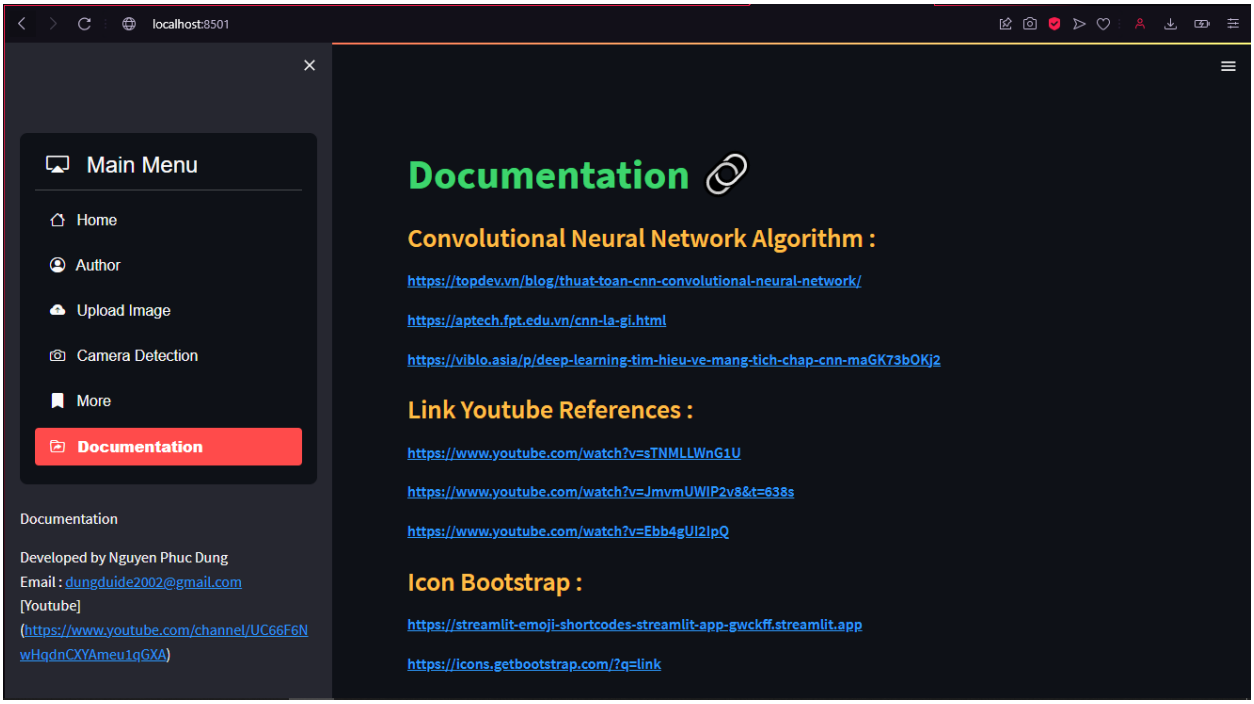




Hình 4.1 và 4.2. Giao diện Home và Thông tin cá nhân của webapp



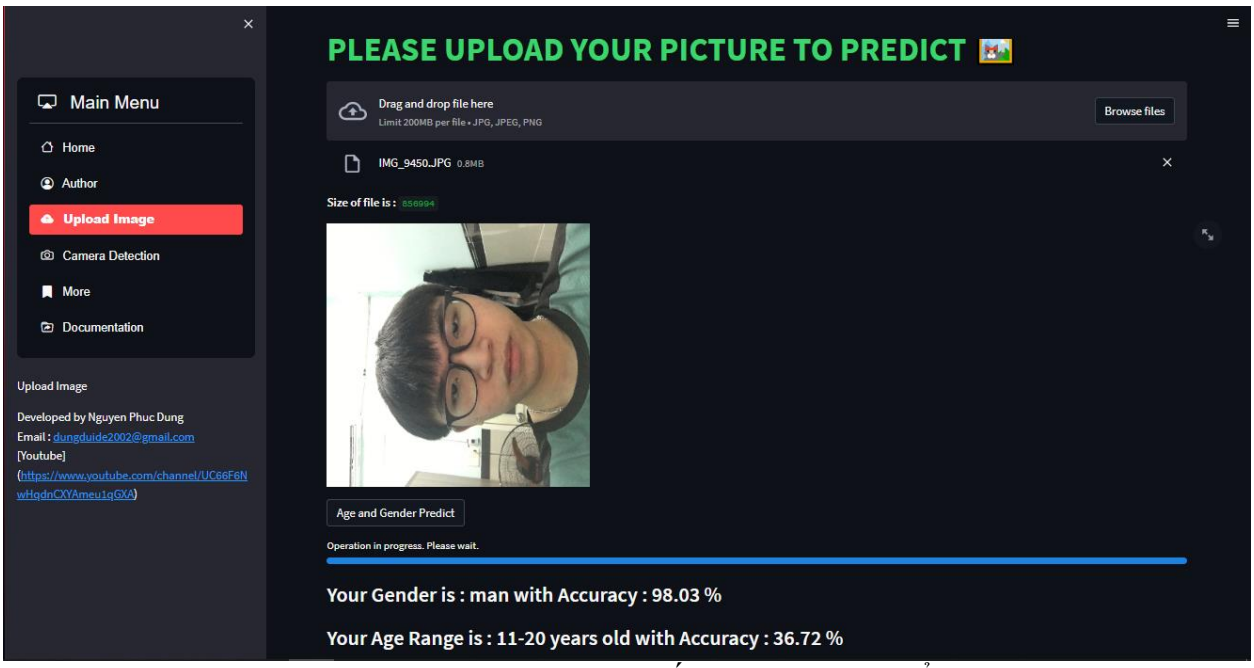
Hình 4.3. Giao diện Thông tin thêm của webapp



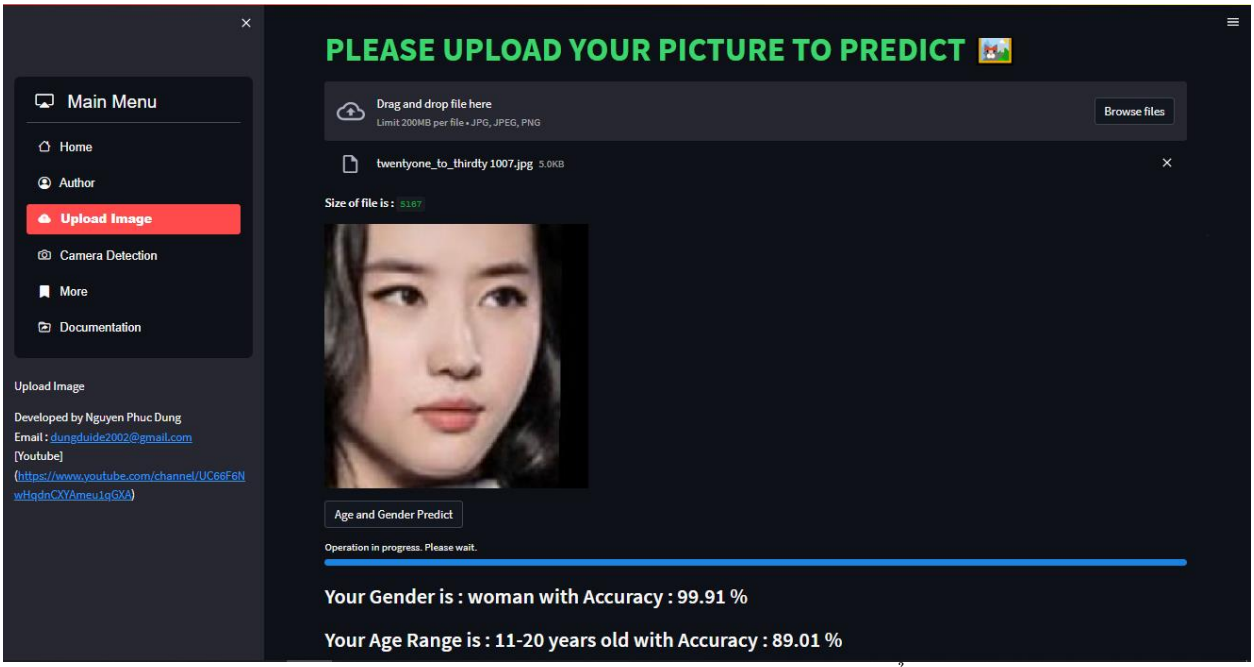
Hình 4.4. Giao diện Tài liệu tham khảo của webapp

4.2. Upload Image lên webapp để dự đoán

Webapp cho phép tải lên tấm ảnh khuôn mặt để web tiến hành xử lý, hiển thị lên trên web, sau khi nhấn nút *"Age and Gender Predict"* thì sẽ đưa ra kết quả dự đoán được.

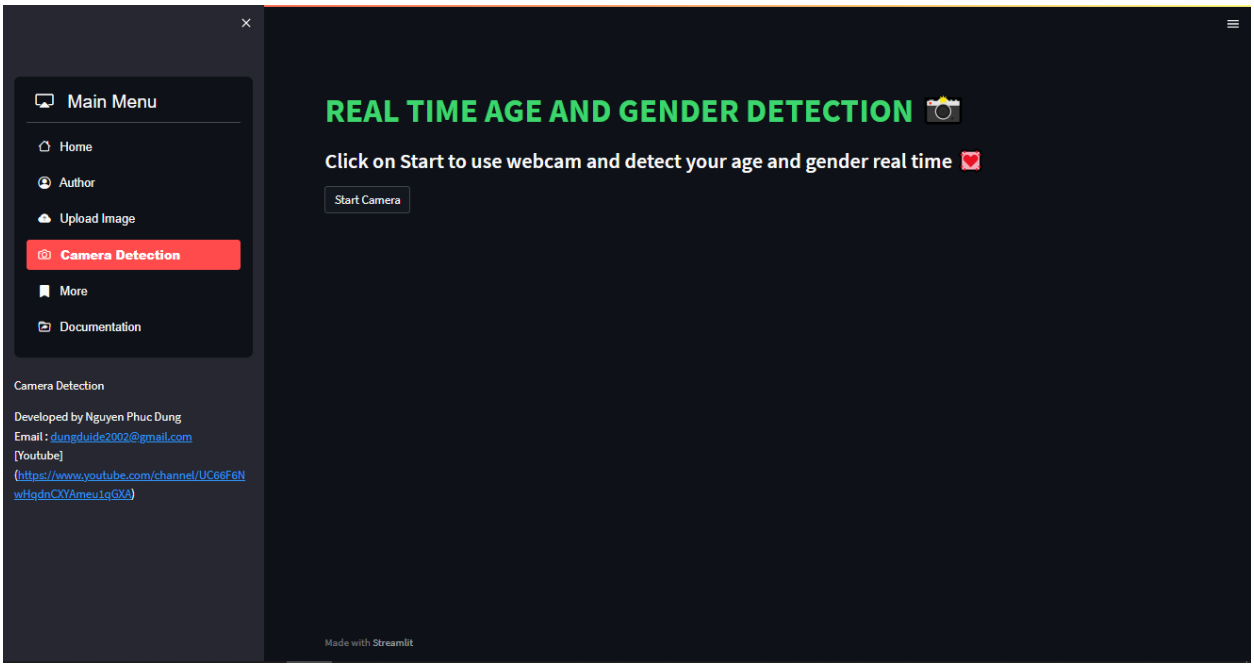


Hình 4.5. Hình ảnh thứ nhất được đưa vào để dự đoán



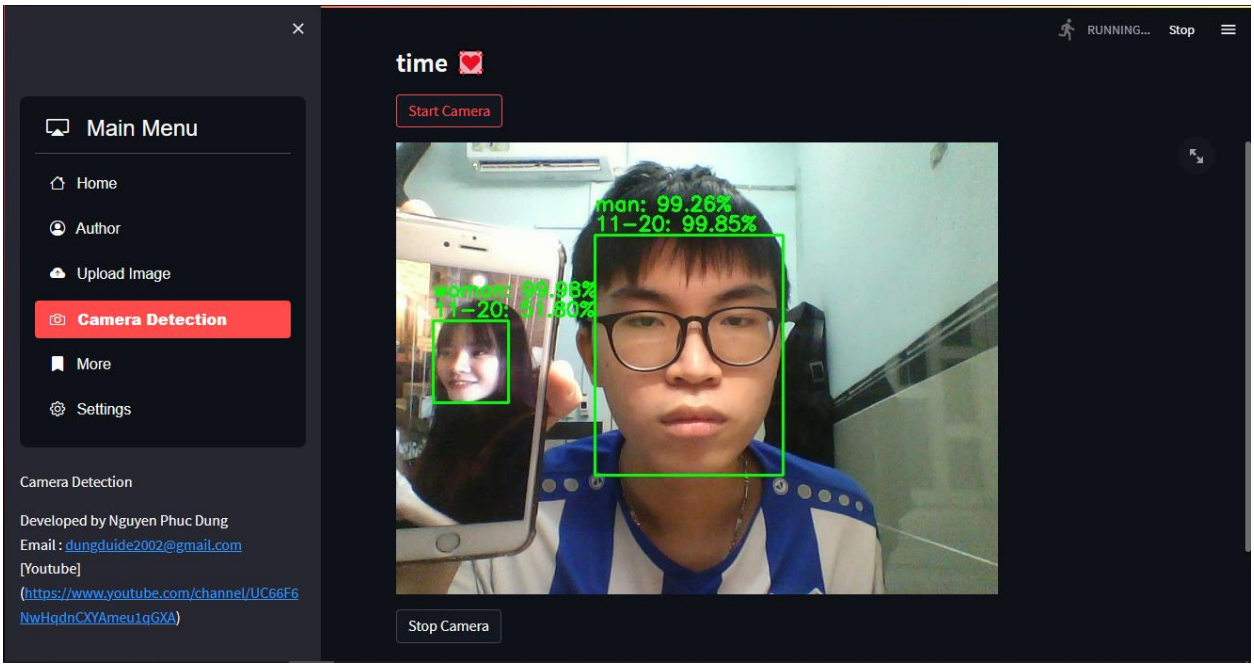
Hình 4.6. Hình ảnh thứ hai được đưa vào để dự đoán

4.3. Camera Detection Real Time kết hợp thư viện openCV



Hình 4.7. Giao diện Camera Real Time trên webapp

Khi ta nhấn nút *"Start Camera"* thì webapp sẽ sử dụng webcam trên máy tính bàn hoặc laptop để tiến hành các quá trình xử lý trực tiếp trong thời gian thực từ đó đưa ra những dự đoán ngay trên webcam hiển thị trên webapp.



Hình 4.8. Quá trình xử lý Camera Real Time trên webapp

KẾT QUẢ ĐẠT ĐƯỢC

Các mô hình áp dụng trong các bài nhận diện hình ảnh sử dụng thuật toán CNN đạt được độ chính xác khá cao (hầu như là trên 94%). Mô hình đã có thể nhận diện được gần như chính xác các bức ảnh đầu vào cũng như sử dụng camera theo thời gian thực nhận diện chính xác giới tính và độ tuổi của con người. Độ chính xác của mô hình phụ thuộc vào số lượng và chất lượng của dữ liệu đầu vào dùng để train mô hình. Dữ liệu ảnh càng đa dạng (các góc quay khác nhau, độ nghiêng, màu sắc, số lượng ảnh,..) thì độ chính xác của mô hình càng cao.

Tuy nhiên, dù mô hình có độ chính xác khá cao nhưng không có nghĩa là mô hình sẽ nhận diện đúng được 100% các bức ảnh đầu vào, sẽ vẫn còn những trường hợp nhận diện sai. Lí do là vì chất lượng dữ liệu dùng để train mô hình chất lượng kém, số lượng bị hạn chế chính vì thế nên mô hình vẫn còn những trường hợp sai, không thể hoàn hảo 100% được. Nhưng nhìn chung, kết quả vẫn chấp nhận được.

KẾT LUẬN

CNN là một phương thức rất hữu hiệu trong việc xử lý các dữ liệu ảnh. Thông thường sử dụng CNN giúp cho việc nhận dạng các đối tượng có trong ảnh có độ chính xác khá cao (khoảng trên 80%, có thể lên tới gần 100%) tùy thuộc vào các dữ liệu đầu vào và cách chúng ta cho máy học theo mô hình CNN như thế nào.

Thế nhưng sử dụng mô hình CNN không đồng nghĩa với việc chắc chắn sẽ nhận dạng đúng. Việc sai sót vẫn là không thể tránh khỏi. Việc này phụ thuộc khá nhiều vào số lượng, chất lượng dữ liệu và kinh nghiệm chọn phương pháp học của người dạy.

Như có thể thấy, các công nghệ phát hiện tuổi và giới tính có thể rất quan trọng đối với các doanh nghiệp của chúng ta. Việc sử dụng những công nghệ này giúp chúng ta có thể đầu tư và phát triển doanh nghiệp của mình ở nhiều cấp độ khác nhau. Nếu chúng ta muốn tận dụng khả năng của công nghệ đầu tư phù hợp vào doanh nghiệp của mình, ta có thể bắt đầu sử dụng Camera Real-time. Để dễ dàng sử dụng thì mô hình huấn luyện đã được tích hợp sẵn vào webapp để người dùng có thể dễ dàng sử dụng.

Việc sử dụng mô hình CNN sẽ giúp cho việc xử lý trên dữ liệu ảnh tốt hơn chính xác hơn, điều này giúp cho các dự án hoặc các ứng dụng có liên quan đến lĩnh vực này sẽ dễ dàng phát triển hơn.