BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

HCMUTE

Artificial Intelligence
ARIN337629E

AI final project

EMOTION, GENDER, AGE RECOGNITION BASE ON HUMAN FACE USING CONVOLUTIONAL NEURAL NETWORK AND BOUNDING BOX REGRESSION.

GVHD: PGS.TS Nguyễn Trường Thịnh
SVTH: Nguyễn Công ThànH 19146120

Hồ Chí Minh, ngày 23 tháng 6, 2022

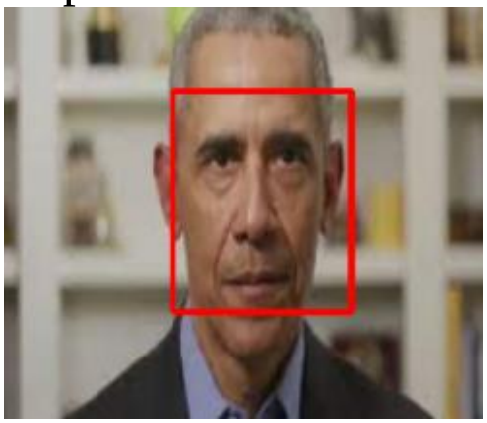
LINK YOUTUBE: https://youtu.be/TH-r1eoEoZQ

Bounding Box regression training

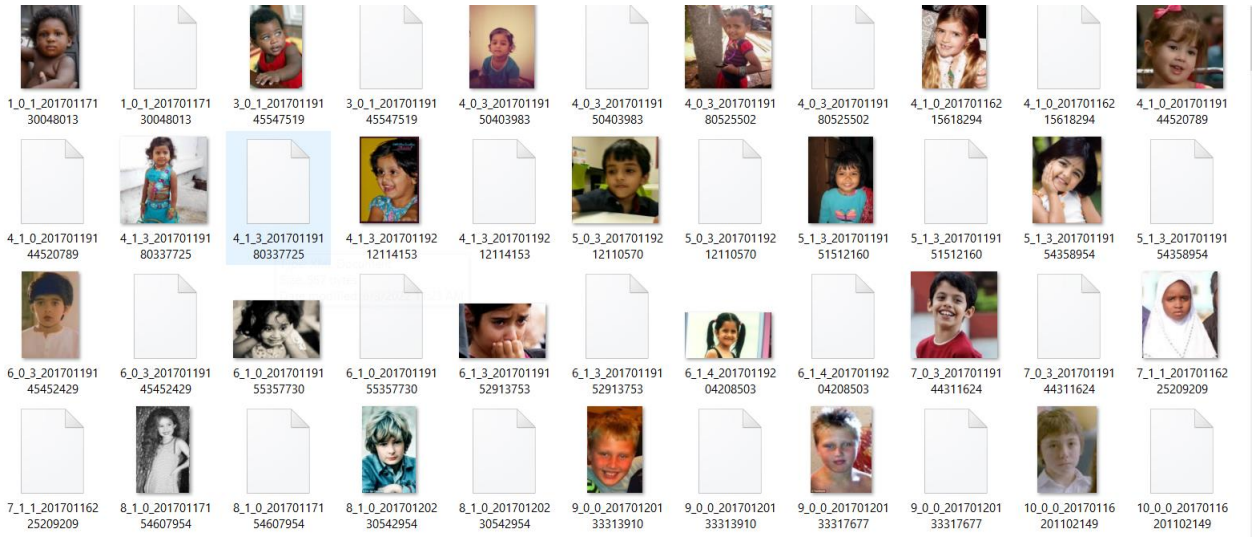
Goal of model:

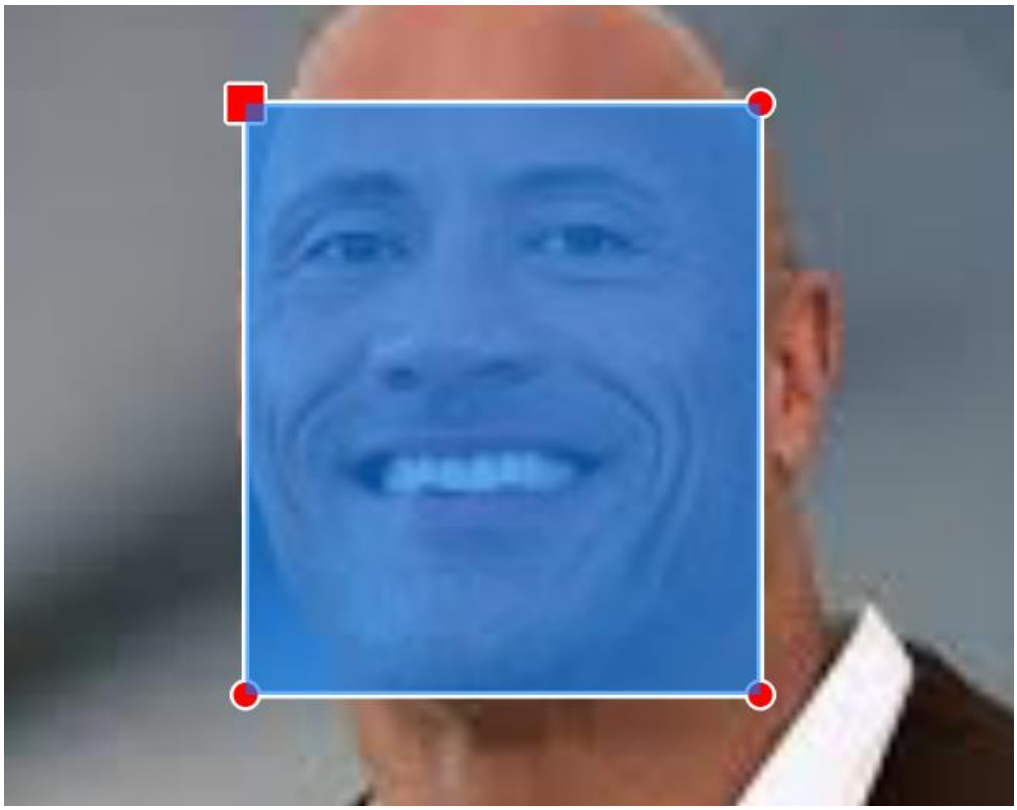
Original:

Expected result:



Dataset preparation:





Import library

Tensorflow for deep learning
Xml lib for xml file processing
OpenCV for image processing

```
import tensorflow as tf
import pandas as pd
import os
import numpy as np
from xml.dom import minidom
import cv2
import matplotlib.pyplot as plt
import glob
from sklearn.model_selection import train_test_split
from skimage import io
```

Make a data and annotation loading function

```python
def load_img(path) :
    X = []
    for i in sorted(glob.glob(path)) :
        img = cv2.imread(i,cv2.COLOR_BGR2RGB)
        img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
        img = cv2.resize(img,(225,225))
        X.append(img)
    return np.asarray(X)
```

```python
def load_xml(path) :
    y = []
    for i in sorted(glob.glob(path)) :

        obj = minidom.parse(i)
        folder = obj.getElementsByTagName('folder')[0].firstChild.nodeValue
        width = get_value(obj,'width')
        height = get_value(obj,'height')
        xmin = get_value(obj,'xmin')/width
        ymin = get_value(obj,'ymin')/height
        xmax = get_value(obj,'xmax')/width
        ymax = get_value(obj,'ymax')/height


        y.append([xmin,ymin,xmax,ymax])
    return np.asarray(y)
```

Load data and label

```
X = load_img('gdrive/MyDrive/facetest/*.jpg')
X.shape
```

```
(304, 225, 225, 3)
```

```
Y = load_xml('gdrive/MyDrive/facetest/*.xml')
Y.shape
```

```
(304, 4)
```

Build Model CNN + Regression
Input is the RGB image with tensor shape (225,225,3)
The output is 4 equivalent to 4 corner of bounding box: xmin, ymin, xmax, ymax

```
model = Sequential()
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same',input
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(64,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Conv2D(128,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(128,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Conv2D(256,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(256,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(512,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(512,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
#model.add(Conv2D(512,(1,1),activation='relu',kernel_initializer='he_uniform',padding='same'))
#model.add(Conv2D(512,(1,1),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
#model.add(Conv2D(512,(1,1),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dropout(0.2))
#model.add(Dense(512,activation='relu'))
model.add(Dense(252,activation='relu'))
model.add(Dense(128,activation='relu'))
model.add(Dense(64,activation='relu'))

#model.compile(loss='mse',optimizer=keras.optimizers.Adam(),metrics=['mae'])
model.add(Dense(4))
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_108 (Conv2D)          (None, 225, 225, 32)      896

conv2d_109 (Conv2D)          (None, 225, 225, 32)      9248

max_pooling2d_59 (MaxPoolin  (None, 112, 112, 32)      0
g2D)

batch_normalization_39 (Bat  (None, 112, 112, 32)      128
chNormalization)

conv2d_110 (Conv2D)          (None, 112, 112, 64)      18496

conv2d_111 (Conv2D)          (None, 112, 112, 64)      36928

max_pooling2d_60 (MaxPoolin  (None, 56, 56, 64)        0
g2D)

batch_normalization_40 (Bat  (None, 56, 56, 64)        256
chNormalization)

conv2d_112 (Conv2D)          (None, 56, 56, 128)       73856
...
Total params: 11,078,496
Trainable params: 11,077,024
Non-trainable params: 1,472
```

Compile CNN model with optimizer is Adam, loss function is ‘Huber’

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$

```
model.compile(optimizer='adam',loss='huber')
```

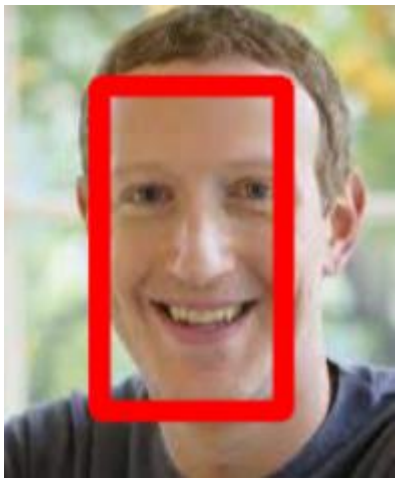
Loss result of model

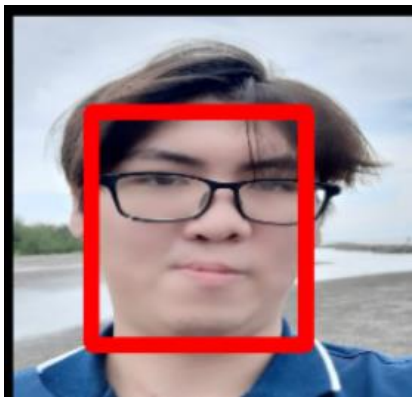
```
Epoch 199/200
10/10 [==============================] - 2s 161ms/step - loss: 4.9854e-04
Epoch 200/200
10/10 [==============================] - 2s 161ms/step - loss: 4.3969e-04

<keras.callbacks.History at 0x7fb728525150>
```

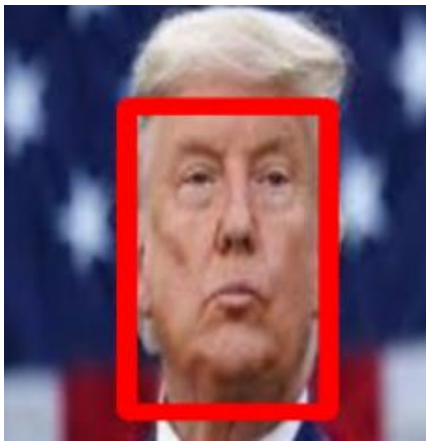
Save model

```
model.save('face.h5')
```

Result:







Human Age regression training

A person's age cannot be precisely determined by facial appearance, instead, we estimate a person's age in a certain age range by using regression with CNN for image feature extraction, the loss function is mse

Build model

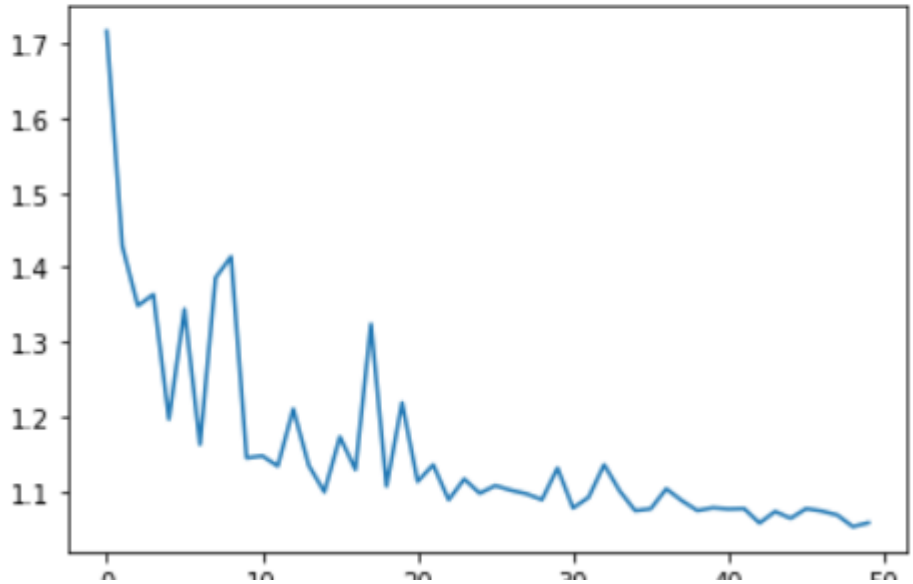
```
model = Sequential()
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same',input_shape=(100,100,3)))
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(64,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Conv2D(128,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(128,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Conv2D(256,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(256,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(512,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(512,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(252,activation='relu'))
model.add(Dense(128,activation='relu'))
model.add(Dense(1,activation='linear'))
model.compile(loss='mse',optimizer=keras.optimizers.Adam(),metrics=['mae'])
model.summary()
```

Loss result of model:

```
Epoch 46/50
646/646 [==============================] - 77s 119ms/step - loss: 0.1639 - mae: 0.2837 - val_loss: 2.4702 - val_mae: 1.0763
Epoch 47/50
646/646 [==============================] - 77s 119ms/step - loss: 0.1600 - mae: 0.2780 - val_loss: 2.5382 - val_mae: 1.0730
Epoch 48/50
646/646 [==============================] - 77s 119ms/step - loss: 0.1591 - mae: 0.2784 - val_loss: 2.4481 - val_mae: 1.0677
Epoch 49/50
646/646 [==============================] - 77s 119ms/step - loss: 0.1550 - mae: 0.2736 - val_loss: 2.3827 - val_mae: 1.0521
Epoch 50/50
646/646 [==============================] - 77s 119ms/step - loss: 0.1418 - mae: 0.2614 - val_loss: 2.4635 - val_mae: 1.0573
```

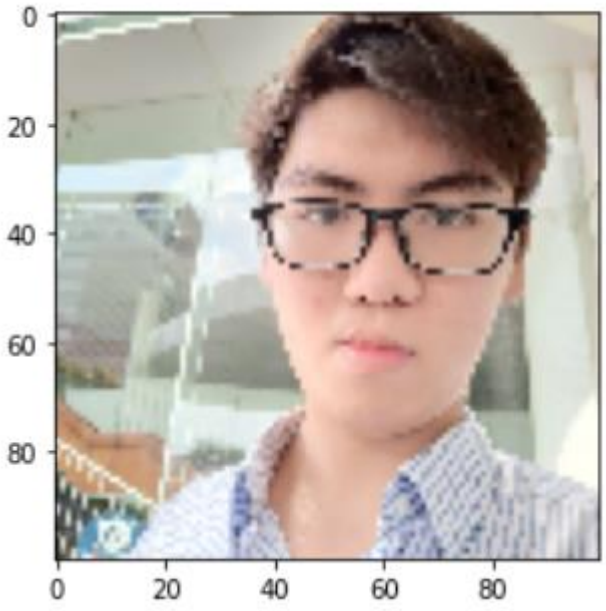
```
[ ]  plt.plot(hist.history['val_mae'])
```

```
[<matplotlib.lines.Line2D at 0x7fc24eee0c50>]
```

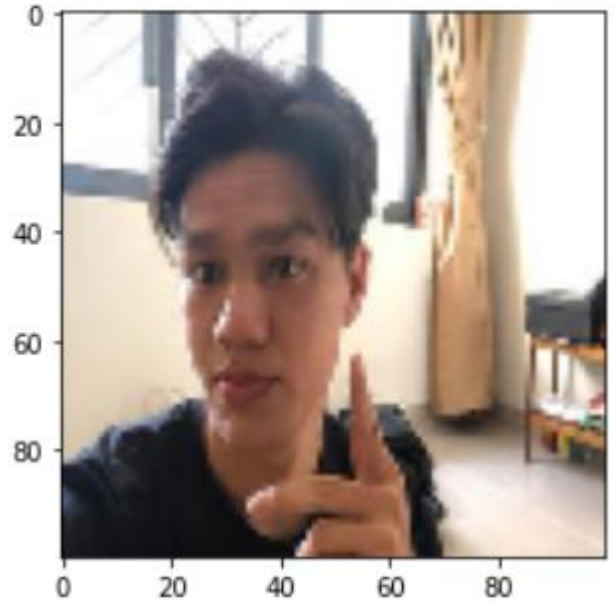


Testing result

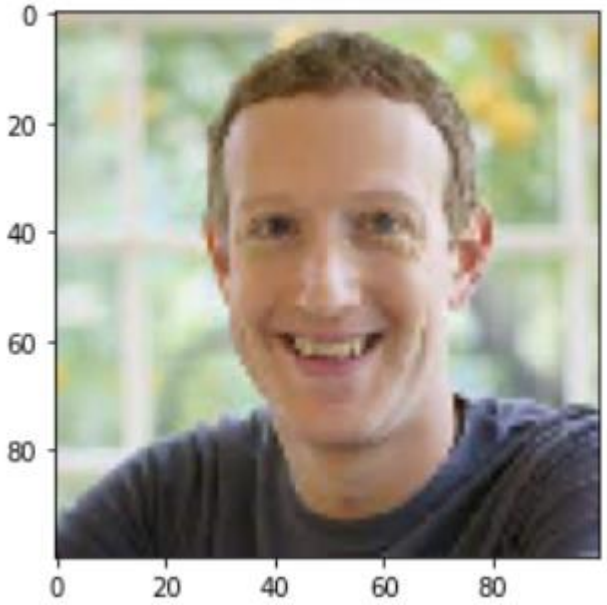
```
expectation 21-25 tuoi
```



expectation 21-25 tuoi



expectation 36-40 tuoi



Sex classification training

There ‘re only 2 biological genders: male and female

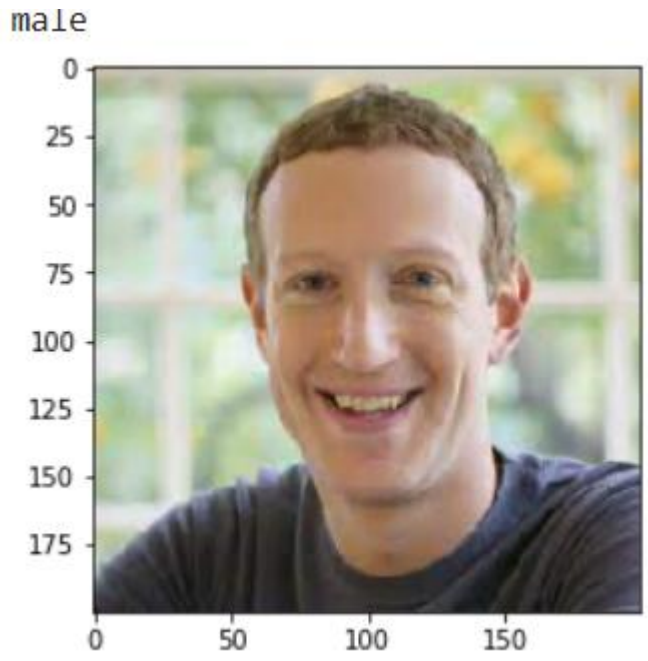
The label of genders data is binary data which mean only 0 and 1

The output of model for sex classification is 1 with the activation function sigmoid

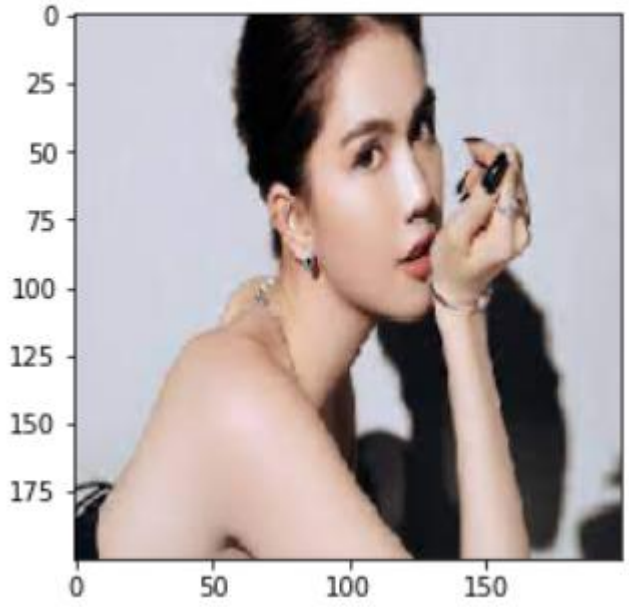
Which mean we are gonna use logistic classication with CNN model

```
model = Sequential()
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same',input_shape=(200,200,3)))
model.add(Conv2D(32,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(64,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Conv2D(128,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(128,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Conv2D(256,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(256,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(512,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(Conv2D(512,(3,3),activation='relu',kernel_initializer='he_uniform',padding='same'))
model.add(MaxPooling2D((2,2)))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(252,activation='relu'))
model.add(Dense(128,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer=keras.optimizers.Adam(),metrics=['accuracy'])
model.summary()
```

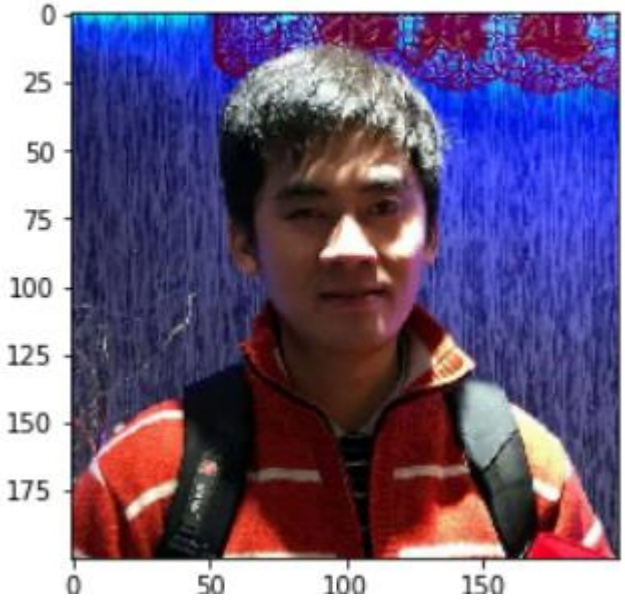
Testing result:



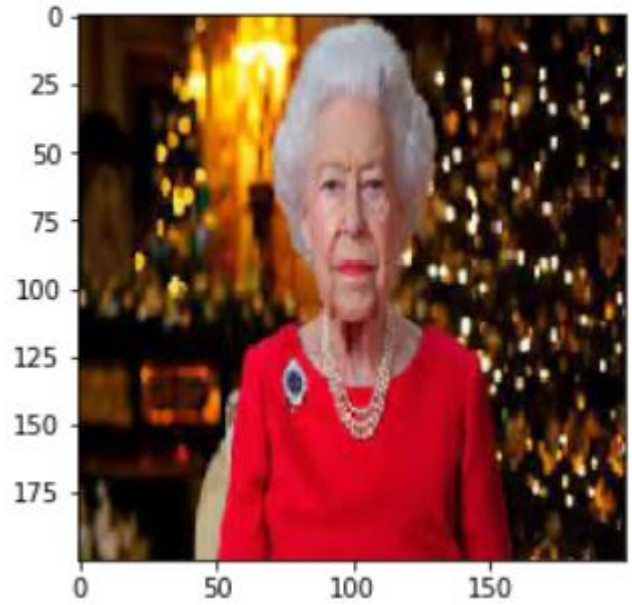
```
female
[[0.9980172]]
```




```
male
[[2.3027935e-06]]
```



```
female
[[0.98984843]]
```



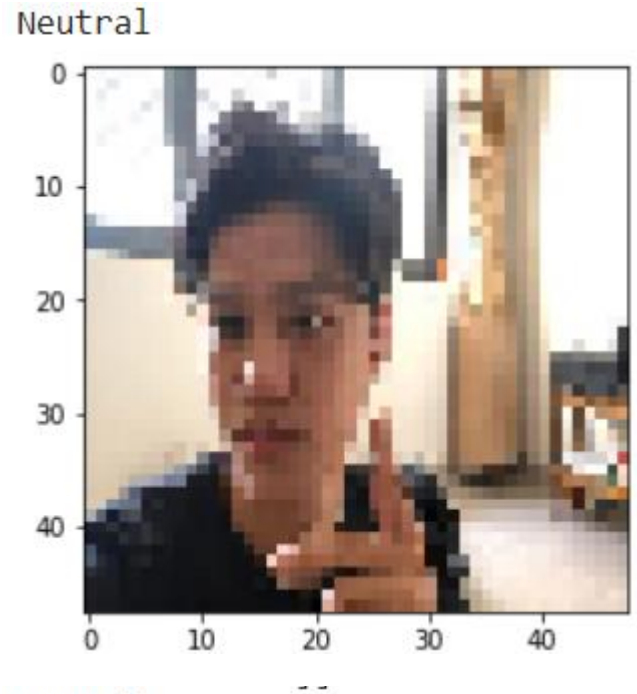
Facial emotion classification training

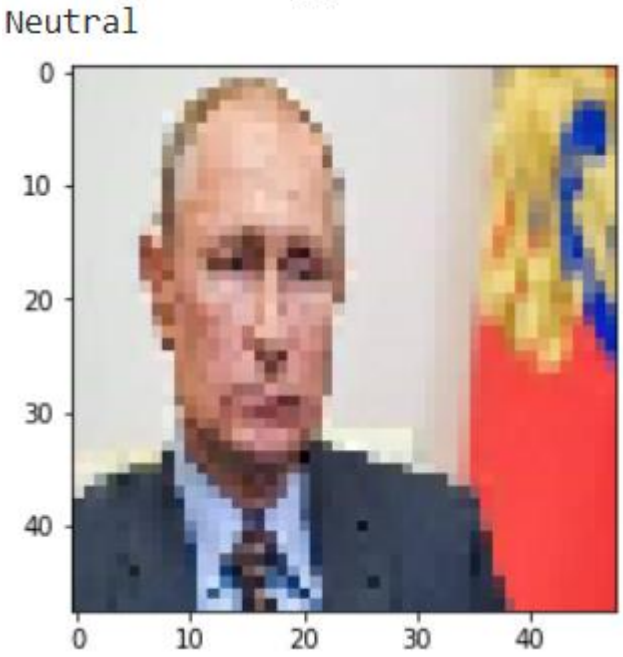
There ar 6 facial emotion of human, so we use CNN with sofmaxt function to classify them

```
# Block-4
model.add(Conv2D(256,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Conv2D(256,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

# Block-5
model.add(Flatten())
model.add(Dense(64,kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# Block-6
model.add(Dense(64,kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# Block-7
model.add(Dense(6,kernel_initializer='he_normal'))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy',optimizer=keras.optimizers.Adam(),metrics=['accuracy'])
```

```
693/693 [==============================] - 9s 14ms/step - loss: 0.1560 - accuracy: 0.9512 - val_loss: 2.9832 - val_accuracy: 0.5556
Epoch 46/50
693/693 [==============================] - 9s 13ms/step - loss: 0.1587 - accuracy: 0.9499 - val_loss: 2.2803 - val_accuracy: 0.6552
Epoch 47/50
693/693 [==============================] - 10s 14ms/step - loss: 0.1480 - accuracy: 0.9535 - val_loss: 2.2947 - val_accuracy: 0.6470
Epoch 48/50
693/693 [==============================] - 9s 13ms/step - loss: 0.1544 - accuracy: 0.9523 - val_loss: 1.8658 - val_accuracy: 0.7015
Epoch 49/50
693/693 [==============================] - 9s 13ms/step - loss: 0.1446 - accuracy: 0.9531 - val_loss: 2.6135 - val_accuracy: 0.6060
Epoch 50/50
693/693 [==============================] - 9s 13ms/step - loss: 0.1467 - accuracy: 0.9526 - val_loss: 2.5204 - val_accuracy: 0.5983
```

Testing result:





Realtime

Load model

```
model = load_model('face.h5')
model1 = load_model('linear.h5')
model2 = load_model('gender.h5')
model3 = load_model('emotion.h5')
```

Make label

```
label = {0: '1-5 ',
         1: '6-10 ',
         2: '11-15 ',
         3: '16-20 ',
         4: '21-25 ',
         5: '26-30 ',
         6: '31-35 ',
         7: '36-40 ',
         8: '41-45 ',
         9: '46-50 ',
         10: '51-55 ',
         11: '56-60 ',
         12: '61-65 ',
         13: '65-70 ',
         14: '71 -75 ',
         15: '75-80 ',
         16: '80-85 ',
         17: '85 tro len'}
gender_labels = ['Male', 'Female']
emotion_labels=['Angry','Disgust','Happy','Neutral','sad','Surprise']
```

‘face’ is the location of facial bouding box

extract the corner xmin ymin xmax ymax from ‘face’

draw the bouding box with cv.rectangle() function

```
face = model.predict(face.reshape(1,225,225,3))
gray=cv.cvtColor(frame,cv.COLOR_BGR2GRAY)

for (x1,y1,x2,y2) in face:
    #cv.rectangle(frame,(int(x1*680),int(y1*480)),(int(x2*680),int(y2*480)),(255,0,0),2)
    cv.rectangle(frame,(int(x1*670),int(y1*680)),(int(x2*680),int(y2*680)),(255,0,0),2)
    #roi_color=frame[int(y1*480):int(y2*480),int(x1*680):int(x2*680)]
```

Predict the age of human

```
roi_color=frame[int(y1*680):int(y2*680),int(x1*680):int(x2*680)]
roi_color=cv.resize(roi_color,(100,100),interpolation=cv.INTER_AREA)
result = model1.predict(np.array(roi_color).reshape(-1,100,100,3))
labels=label[(int(result))]
label_position=(int(x1*225),int(y2*225)+10) #50 pixels below to move the label
cv.putText(frame,labels,label_position,cv.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)
```

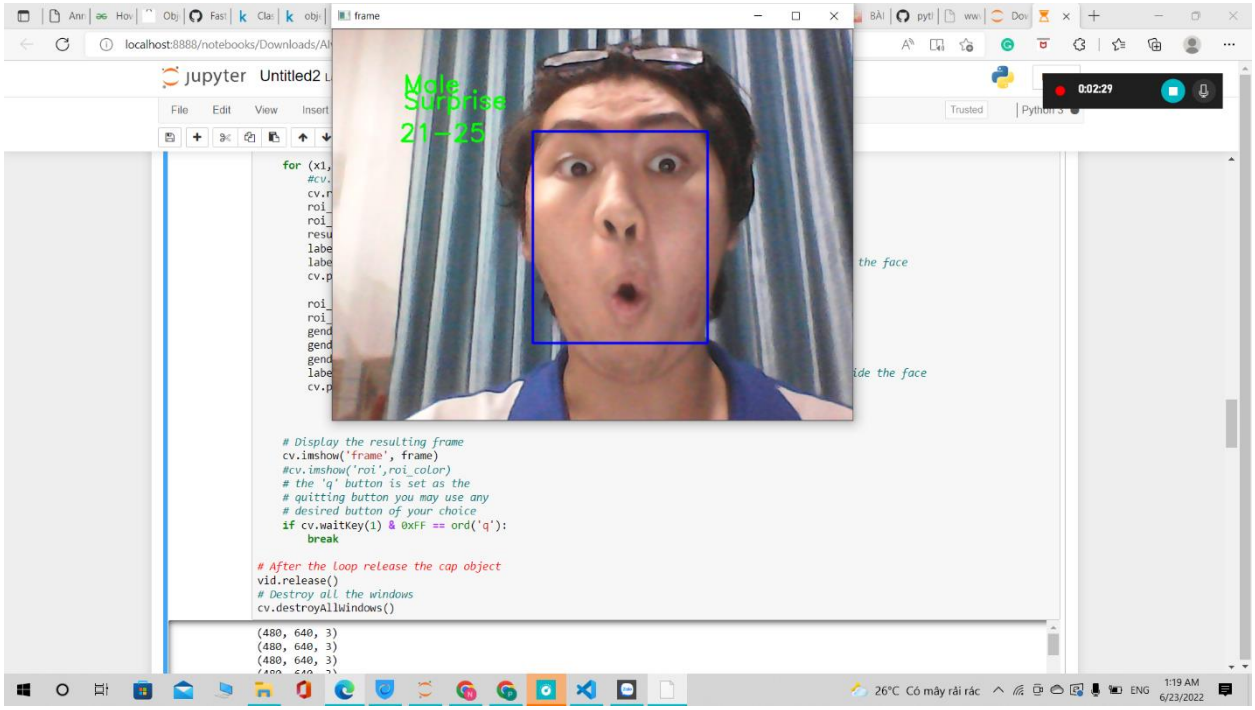
Predict the gender of human

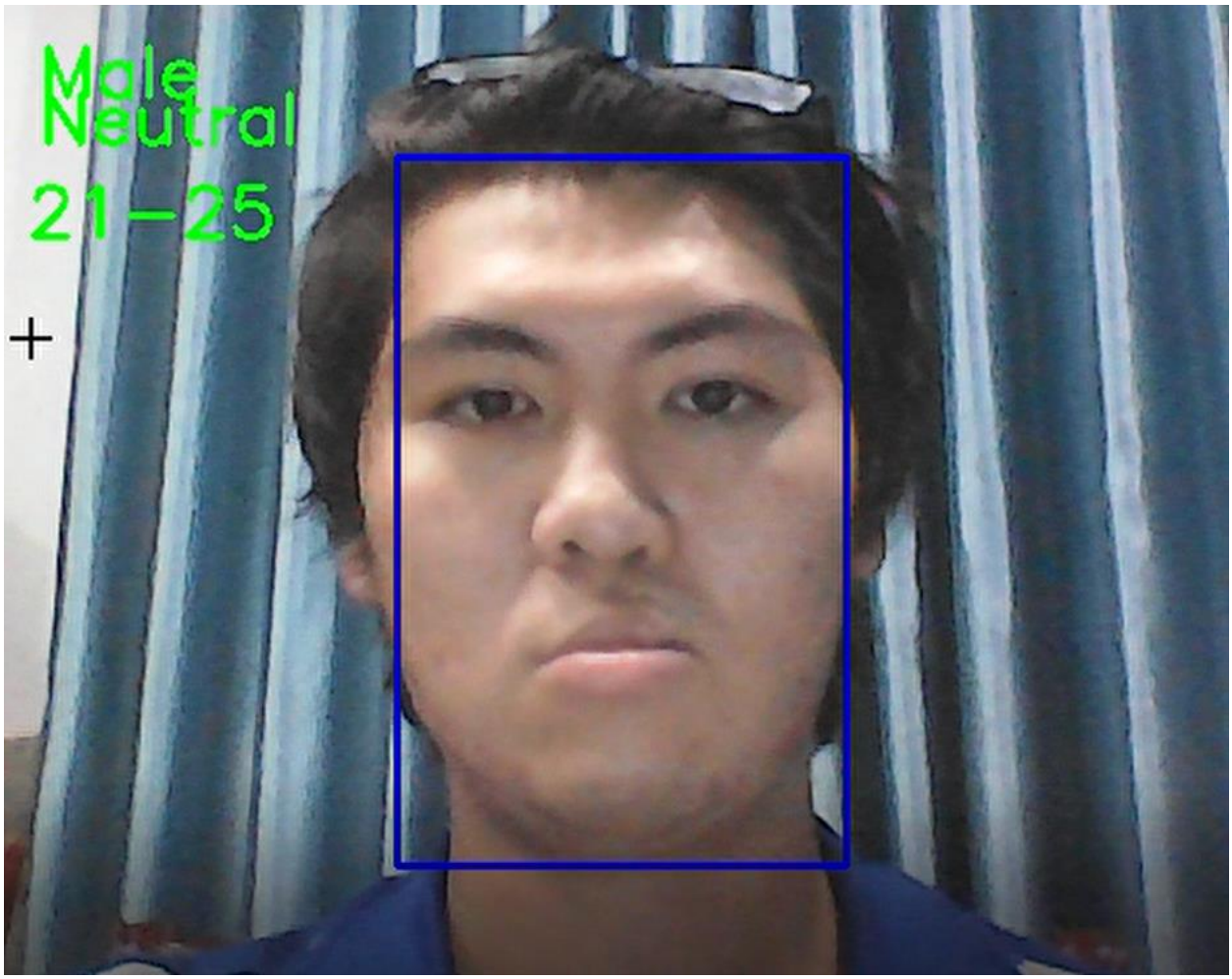
```
roi_gender=frame[int(y1*680):int(y2*680),int(x1*680):int(x2*680)]
roi_gender=cv.resize(roi_gender,(200,200),interpolation=cv.INTER_AREA)
gender = model2.predict(np.array(roi_gender).reshape(-1,200,200,3))
gender = (gender>= 0.5).astype(int)[:,0]
gender_label=gender_labels[gender[0]]
label_position1=(int(x1*225)+5,int(y2*225)-50) #50 pixels below to move the label oqut
cv.putText(frame,gender_label,label_position1,cv.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)
```

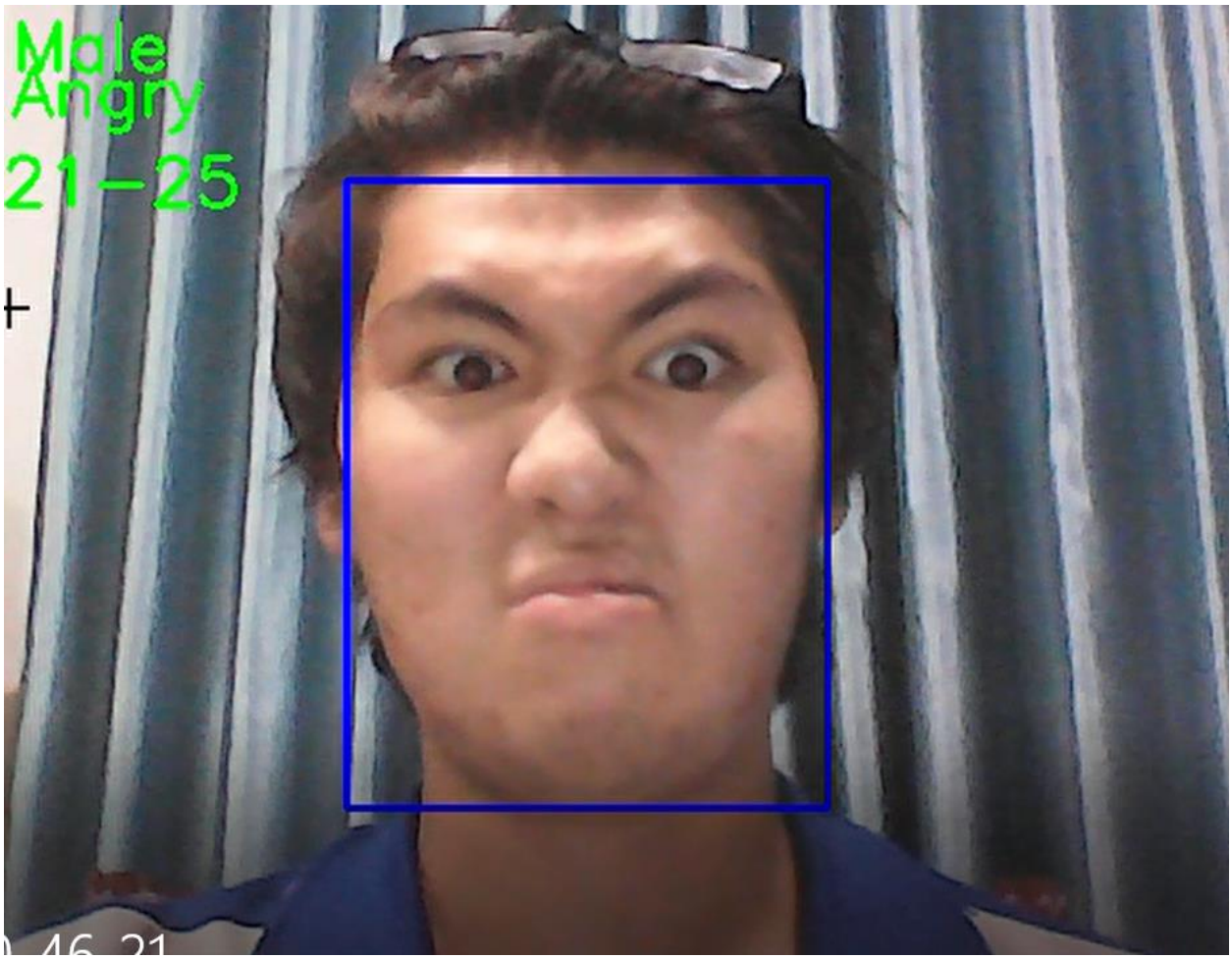
Predict the facial emotion

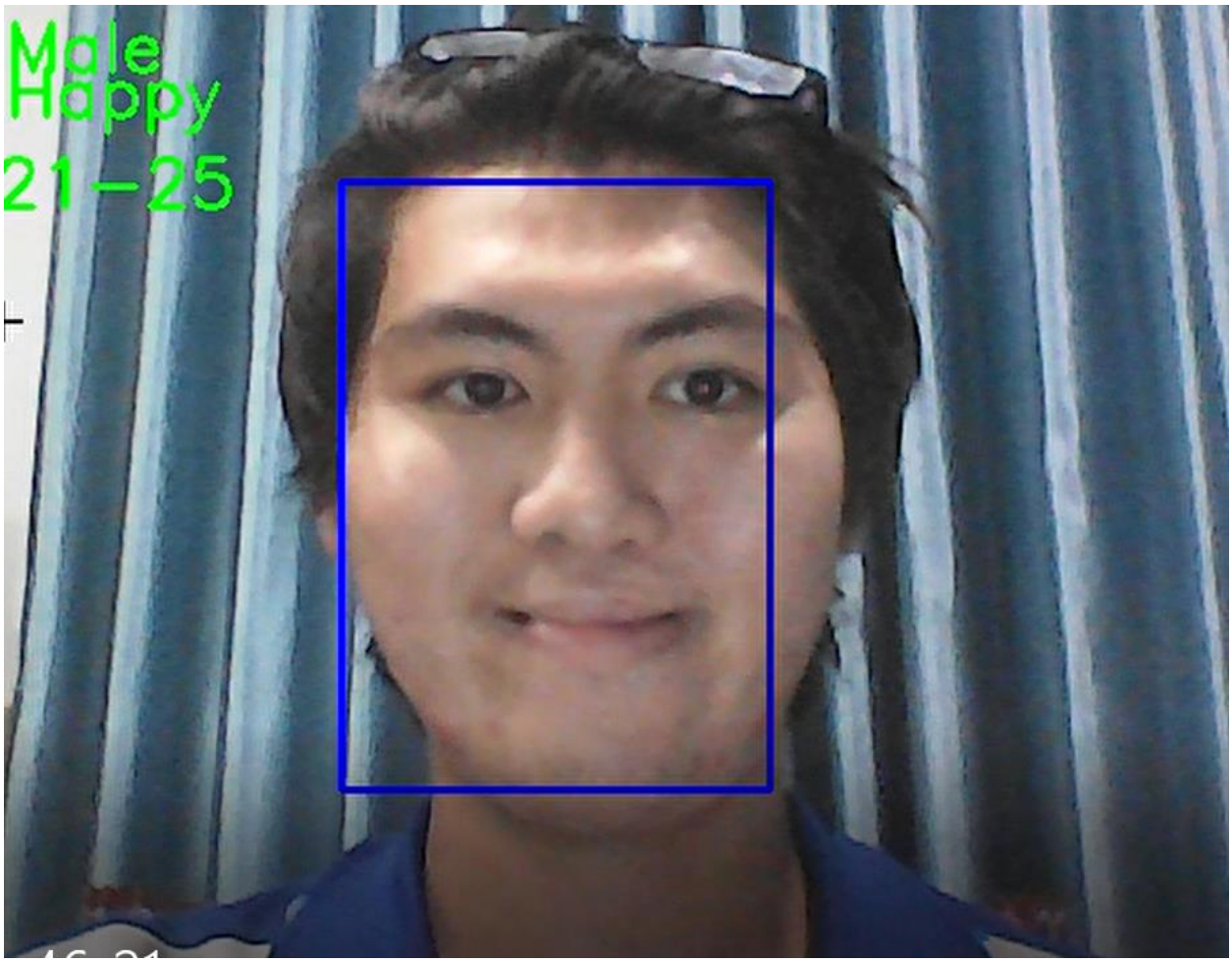
```
roi_emo=gray[int(y1*680):int(y2*680),int(x1*680):int(x2*680)]
roi_emo=cv.resize(roi_emo,(48,48),interpolation=cv.INTER_AREA)
emo = model3.predict(np.array(roi_emo).reshape(-1,48,48,1))
```

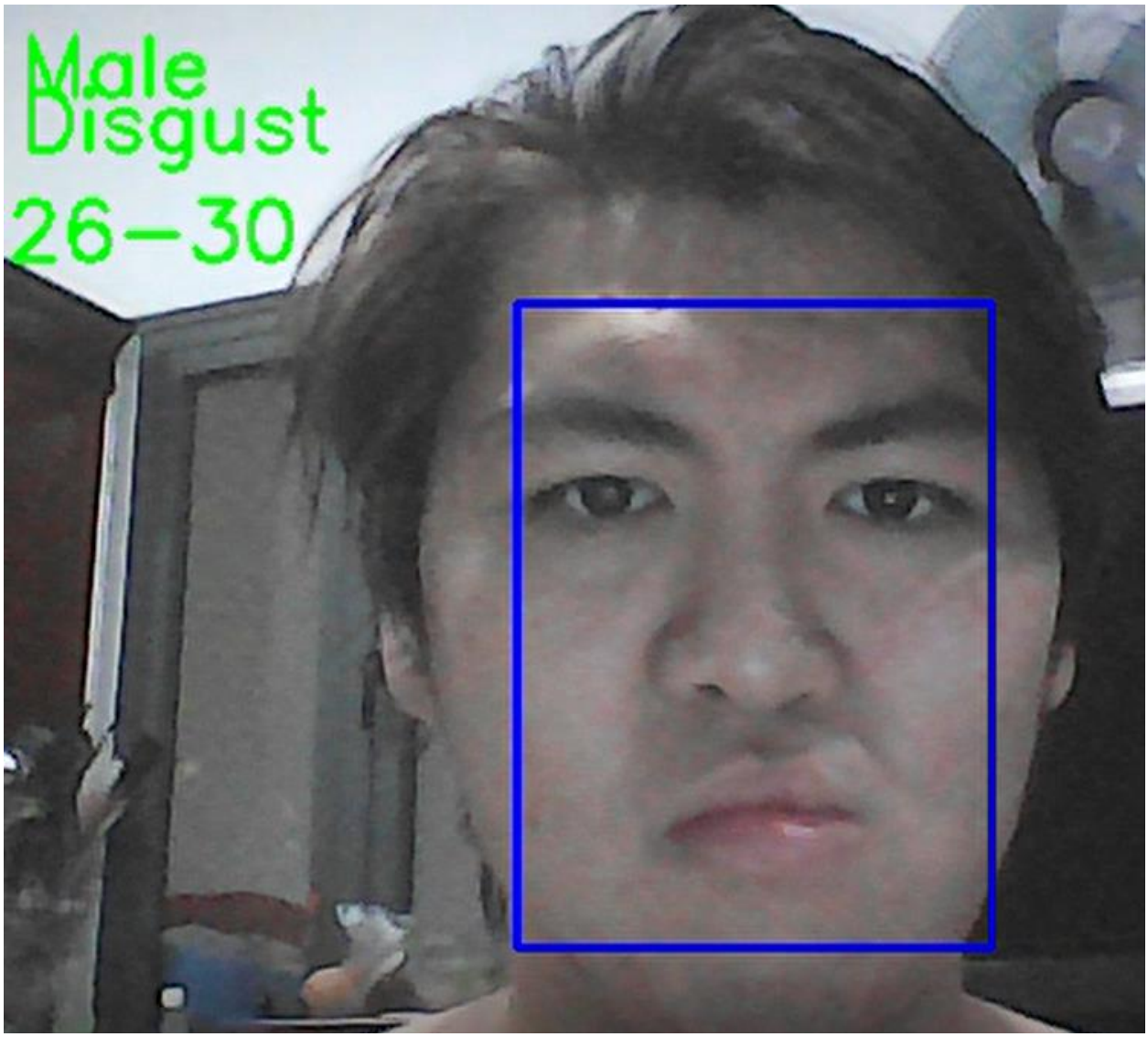
Real-time result




Male
Neutral
21-25


Male
Angry
21-25


Male
Happy
21-25


Male
Disgust
26-30