In [29]:

```
import cv2
import os
import random
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import image as img
```

In [30]:

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Layer, Conv2D, Dense, MaxPooling2D, Input, Flatten
import tensorflow as tf
```

In [31]:

```
POS_PATH = os.path.join('E:/Face_Recognition' , 'positive1')
NEG_PATH = os.path.join( 'E:/Face_Recognition', 'negative')
ANC_PATH = os.path.join( 'E:/Face_Recognition', 'anchor1')
```

In [32]:

```
import uuid
```

In [ ]:

In [45]:

```
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()

    frame = frame[120:120+250,200:200+250, :]

    if cv2.waitKey(1) & 0XFF == ord('a'):

        imgname = os.path.join(ANC_PATH, '{}.jpg'.format(uuid.uuid1()))

        cv2.imwrite(imgname, frame)

    if cv2.waitKey(1) & 0XFF == ord('p'):

        imgname = os.path.join(POS_PATH, '{}.jpg'.format(uuid.uuid1()))

        cv2.imwrite(imgname, frame)

    cv2.imshow('Image Collection', frame)

    if cv2.waitKey(1) & 0XFF == ord('q'):
        break

cap.release()

cv2.destroyAllWindows()
```

In [53]:

```
def data_aug(img):
    data = []
    for i in range(9):
        img = tf.image.stateless_random_brightness(img, max_delta=0.02, seed=(1,2))
        img = tf.image.stateless_random_contrast(img, lower=0.6, upper=1, seed=(1,3))
        img = tf.image.stateless_random_flip_left_right(img, seed=(np.random.randint(100),np.random.randint(1
        img = tf.image.stateless_random_jpeg_quality(img, min_jpeg_quality=90, max_jpeg_quality=100, seed=(np
        img = tf.image.stateless_random_saturation(img, lower=0.9,upper=1, seed=(np.random.randint(100),np.ra

        data.append(img)

    return data
```

In [33]:

```
anchor = tf.data.Dataset.list_files(ANC_PATH +'/*.jpg').take(600)
positive = tf.data.Dataset.list_files(POS_PATH +'\*.jpg').take(600)
negative = tf.data.Dataset.list_files(NEG_PATH+'\*.jpg').take(600)
print(anchor)
```

```
<TakeDataset element_spec=TensorSpec(shape=(), dtype=tf.string, name=None)>
```

In [ ]:

In [34]:

```
def preprocess(file_path):

    byte_img = tf.io.read_file(file_path)
    img = tf.io.decode_jpeg(byte_img)
    img = tf.image.resize(img, (100,100))
    img = img / 255.0

    return img
```

In [35]:

```
positives = tf.data.Dataset.zip((anchor , positive , tf.data.Dataset.from_tensor_slices(tf.ones(len(anchor)))
negatives = tf.data.Dataset.zip((anchor, negative, tf.data.Dataset.from_tensor_slices(tf.zeros(len(anchor))))
data = positives.concatenate(negatives)
```

In [36]:

```
print(data)
```

```
<ConcatenateDataset element_spec=(TensorSpec(shape=(), dtype=tf.string, name=None), TensorSpec(shape=(), dty
pe=tf.string, name=None), TensorSpec(shape=(), dtype=tf.float32, name=None))>
```

In [37]:

```
def preprocess_twin (input_img, validation_img, label):
    return(preprocess(input_img), preprocess(validation_img), label)
```

In [38]:

```
data = data.map(preprocess_twin)
data = data.shuffle(buffer_size=1200)
```

In [ ]:

In [39]:

```
train_data = data.take(round(len(data)*.7))
train_data = train_data.batch(16)
train_data = train_data.prefetch(8)
```

In [40]:

```
test_data = data.skip(round(len(data)*.7))
test_data = test_data.take(round(len(data)*.3))
test_data = test_data.batch(16)
test_data = test_data.prefetch(8)
```

In [ ]:

In [ ]:

In [41]:

```
def make_cnn_network():
    inp = Input(shape=(100,100,3), name='input_image')

    c1 = Conv2D(64, (10,10), activation='relu')(inp)
    m1 = MaxPooling2D(64, (2,2), padding='same')(c1)

    c2 = Conv2D(128, (7,7), activation='relu')(m1)
    m2 = MaxPooling2D(64, (2,2), padding='same')(c2)

    c3 = Conv2D(128, (4,4), activation='relu')(m2)
    m3 = MaxPooling2D(64, (2,2), padding='same')(c3)

    c4 = Conv2D(256, (4,4), activation='relu')(m3)
    f1 = Flatten()(c4)
    d1 = Dense(4096, activation='sigmoid')(f1)


    return Model(inputs=[inp], outputs=[d1], name='cnn_network')
```

In [42]:

```
cnn_network = make_cnn_network()
cnn_network.summary()
```

```
Model: "cnn_network"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_image (InputLayer)    [(None, 100, 100, 3)]     0

 conv2d_4 (Conv2D)           (None, 91, 91, 64)        19264

 max_pooling2d_3 (MaxPooling  (None, 46, 46, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 40, 40, 128)       401536

 max_pooling2d_4 (MaxPooling  (None, 20, 20, 128)      0
 2D)

 conv2d_6 (Conv2D)           (None, 17, 17, 128)       262272

 max_pooling2d_5 (MaxPooling  (None, 9, 9, 128)        0
 2D)
```

In [ ]:

In [ ]:

In [43]:
```python
def make_face_model():


    input_image = Input(name='input_img', shape=(100,100,3))

    validation_image = Input(name='validation_img', shape=(100,100,3))

    Face_layer = L1Dist()
    distances = Face_layer(cnn_network(input_image),  cnn_network(validation_image))

    classifier = Dense(1, activation='sigmoid')(distances)

    return Model(inputs=[input_image, validation_image], outputs=classifier, name='Face_model')
```

In [ ]:
```python
Face_model = make_face_model()
Face_model.summary()
```

In [ ]:
```python

```

In [59]:
```python
binary_cross_loss = tf.losses.BinaryCrossentropy()
opt = tf.keras.optimizers.Adam(1e-4)
```

In [60]:
```python
checkpoint_dir = 'E:\\training_checkpoints\\'

checkpoint_prefix = os.path.join(checkpoint_dir)

checkpoint = tf.train.Checkpoint(opt=opt, Face_model= Face_model)
```

In [61]:

```
def train_step(batch):
    with tf.GradientTape() as tape:

        X = batch[:2]

        y = batch[2]

        yhat = Face_model(X, training=True)

        loss = binary_cross_loss(y, yhat)
    print(loss)

    grad = tape.gradient(loss, Face_model.trainable_variables)

    opt.apply_gradients(zip(grad, Face_model.trainable_variables))

    return loss
```

In [ ]:

In [62]:

```
def train(data, EPOCHS):

    for epoch in range(1, EPOCHS+1):
        print('\n Epoch {}/{}'.format(epoch, EPOCHS))
        progbar = tf.keras.utils.Progbar( len(data) )

        r = tf.keras.metrics.Recall()
        p = tf.keras.metrics.Precision()
        idx=0
        for batch in  data:
            loss = train_step(batch)
            yhat = Face_model.predict(batch[:2])
            r.update_state(batch[2], yhat)
            p.update_state(batch[2], yhat)
            progbar.update(idx+1)
        print(loss.numpy(), r.result().numpy(), p.result().numpy())

        if( epoch % 10 == 0):
            checkpoint.save( file_prefix=checkpoint_prefix )
```

In [63]:

```
numpy_iterator = train_data.as_numpy_iterator()
for d in numpy_iterator:
    print(d[2:])
    break
```

```
(array([0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 1., 1., 1., 0., 1., 0.],
      dtype=float32),)
```

In [68]:

```
len(train_data)
```

Out[68]:

```
53
```

In [65]:

```
len(data)
```

Out[65]:

```
1200
```

In [67]:
```
len(train_data)
```

Out[67]:
```
53
```

In [ ]:
```

from tensorflow.keras.metrics import Precision

```

In [151]:
```
s = tf.keras.models.load_model('E:/attendenceModel5.h5',
                                   custom_objects={'L1Dist':L1Dist, 'BinaryCrossentropy':tf.losses.BinaryCros
```

```
WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compil
e it manually.
```

In [1]:
```
def verify(model, detection_threshold, verification_threshold):

    results = []
    for image in os.listdir(os.path.join('E:\data', 'verification_images' , 'anant' )):
        input_img = preprocess(os.path.join('E:\data', 'input_image', 'input_image.jpg'))
        validation_img = preprocess(os.path.join('E:\data', 'verification_images' , 'anant' , image ))

        result = s.predict(list(np.expand_dims([input_img, validation_img], axis=1)))
        results.append(result)


    detection = np.sum(np.array(results) > detection_threshold)

    verification = detection / 100
    verified = verification > verification_threshold

    return results, verified ,input_img

```

In [183]:

```
results, verified , input_image = verify(s , 0.5, 0.8)
print(verified)
```

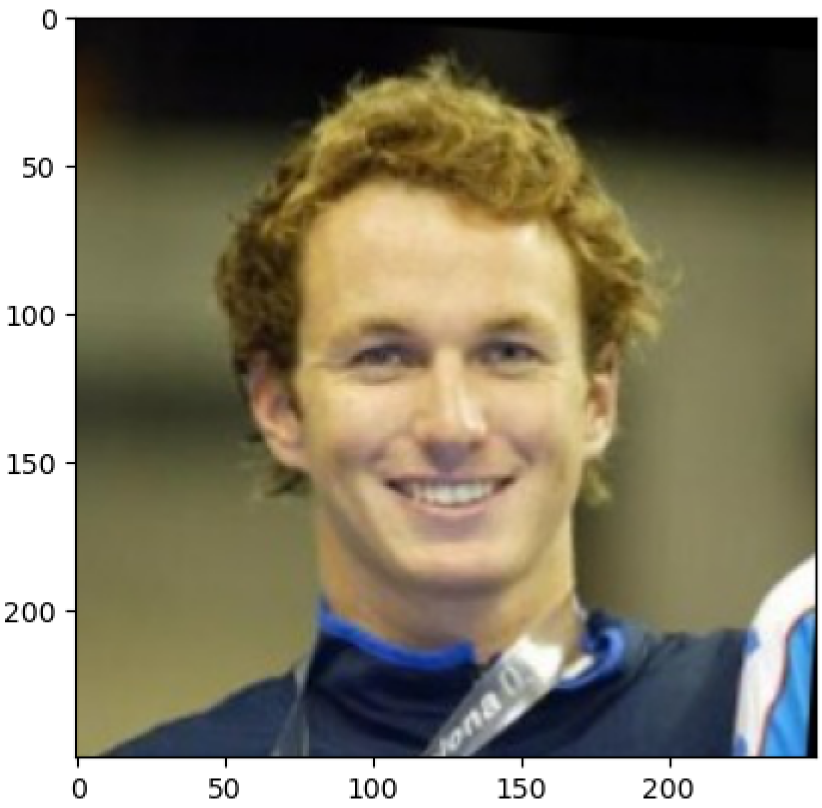
```
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 107ms/step
1/1 [==============================] - 0s 119ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 115ms/step
1/1 [==============================] - 0s 104ms/step
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 121ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 119ms/step
1/1 [==============================] - 0s 105ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 116ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 118ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 107ms/step
```

```
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 117ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 138ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 119ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 118ms/step
1/1 [==============================] - 0s 117ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 116ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 102ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 112ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 99ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 118ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 120ms/step
1/1 [==============================] - 0s 125ms/step
```

```
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 106ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 125ms/step
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 110ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 108ms/step
1/1 [==============================] - 0s 107ms/step
1/1 [==============================] - 0s 111ms/step
False
```

In [182]:
```
testImage = img.imread('E:\data\input_image\input_image.jpg')
plt.imshow(testImage)
```

Out[182]:
```
<matplotlib.image.AxesImage at 0x206eea06880>
```



In [ ]:

In [25]:

```
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    frame = frame[120:120+250,200:200+250, :]

    cv2.imshow('Verification', frame)

    if cv2.waitKey(10) & 0xFF == ord('v'):

        cv2.imwrite(os.path.join( 'E:\data' , 'input_image' , 'input_image.jpg' ), frame)

        results, verified = verify( s  , 0.5, 0.8)
        print(verified)


    if cv2.waitKey(10) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

In [ ]:

In [9]:

```
model = tf.keras.models.load_model('attendenceModel5.h5' , custom_objects={'L1Dist':L1Dist})
```

```
WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compil
e it manually.
```

In [ ]:

```
y_hat = model.predict([test_input, test_val])
y_hat
```

In [28]:

```
test_input, test_val, y_true = test_data.as_numpy_iterator().next()
```

In [ ]:

In [31]:

```
y_true
```

Out[31]:

```
array([0., 0., 1., 1., 0., 1., 1., 0., 0., 0., 0., 1., 0., 0., 1., 1.],
      dtype=float32)
```

In [32]:

```
[1 if p > 0.5

 else 0 for p in y_hat ]
```

Out[32]:

```
[0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1]
```

In [ ]:

```
m = Precision()
m.update_state(y_true, y_hat)

print("Precission of this model is :")

m.result().numpy()*100
```

In [34]:

```
len(test_data)
```

Out[34]:

```
12
```

In [39]:

```
 test_input , test_val , y_true = test_data.as_numpy_iterator().next()
```

In [ ]:

In [ ]:

In [ ]:

In [21]:

In [ ]:

```
In [ ]:
```