#pip install opencv-python

**IMPORT LIBRARIES**

import numpy as np

import pandas as pd

import os

import cv2

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras import layers

## IMPORT DATASET

# loading training data

train\_data= tf.keras.preprocessing.image\_dataset\_from\_directory("C:/Users/HP/Documents/train")

# loading testing data

test\_data = tf.keras.preprocessing.image\_dataset\_from\_directory('C:/Users/HP/Documents/test’)

Found 1234 files belonging to 2 classes.

Found 218 files belonging to 2 classes.

## CHECK LABELED NAME

class\_names = train\_data.class\_names

print(class\_names)

['Closed', 'Open']

## TO VISUALIZE LABELED DATA

plt.figure(figsize=(10,5))

for images, labels in train\_data.take(10):

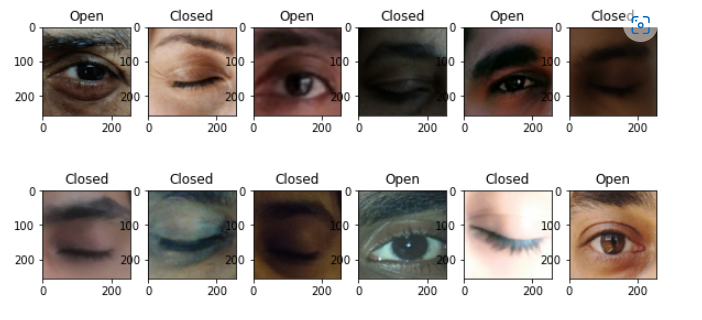
for i in range(12):

ax = plt.subplot(2,6,i+1)

plt.imshow(images[i].numpy().astype("uint8"))

plt.title(class\_names[labels[i]])

plt.axis("on")



## TUNING

AUTOTUNE = tf.data.experimental.AUTOTUNE

train\_data = train\_data.cache().prefetch(buffer\_size=AUTOTUNE)

test\_data = test\_data.cache().prefetch(buffer\_size=AUTOTUNE)

## DEFINE MODEL

Cnn = tf.keras.models.Sequential([

layers.experimental.preprocessing.Rescaling(1./255),

layers.Conv2D(32, 3, activation='relu'),

layers.MaxPooling2D(),

layers.Conv2D(64, 3, activation='relu'),

layers.MaxPooling2D(),

layers.Conv2D(128, 3, activation='relu'),

layers.MaxPooling2D(),

layers.GlobalAveragePooling2D(),

layers.Dense(256, activation='relu'),

layers.Dense(len(class\_names), activation= 'softmax')

])

Cnn.compile(optimizer='adam',loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

## TRAIN CNN

## Value = Cnn.fit(train\_data, validation\_data= test\_data, epochs =10 )

Epoch 1/10

39/39 [==============================] - 209s 5s/step - loss: 0.6772 - accuracy: 0.5989 - val\_loss: 0.6735 - val\_accuracy: 0.5459

Epoch 2/10

39/39 [==============================] - 142s 4s/step - loss: 0.6626 - accuracy: 0.6232 - val\_loss: 0.6344 - val\_accuracy: 0.7064

Epoch 3/10

39/39 [==============================] - 238s 6s/step - loss: 0.6109 - accuracy: 0.6888 - val\_loss: 0.5635 - val\_accuracy: 0.7706

Epoch 4/10

39/39 [==============================] - 226s 6s/step - loss: 0.6405 - accuracy: 0.6434 - val\_loss: 0.5831 - val\_accuracy: 0.7752

Epoch 5/10

39/39 [==============================] - 154s 4s/step - loss: 0.5976 - accuracy: 0.6888 - val\_loss: 0.6623 - val\_accuracy: 0.6743

Epoch 6/10

39/39 [==============================] - 163s 4s/step - loss: 0.5644 - accuracy: 0.7293 - val\_loss: 0.5780 - val\_accuracy: 0.7202

Epoch 7/10

39/39 [==============================] - 136s 3s/step - loss: 0.5426 - accuracy: 0.7520 - val\_loss: 0.5099 - val\_accuracy: 0.7936

Epoch 8/10

39/39 [==============================] - 132s 3s/step - loss: 0.5004 - accuracy: 0.7699 - val\_loss: 0.4775 - val\_accuracy: 0.8028

Epoch 9/10

39/39 [==============================] - 130s 3s/step - loss: 0.4626 - accuracy: 0.7974 - val\_loss: 0.4214 - val\_accuracy: 0.8349

Epoch 10/10

39/39 [==============================] - 170s 4s/step - loss: 0.4675 - accuracy: 0.8039 - val\_loss: 0.4537 - val\_accuracy: 0.8028

## CHECK FOR FITTING

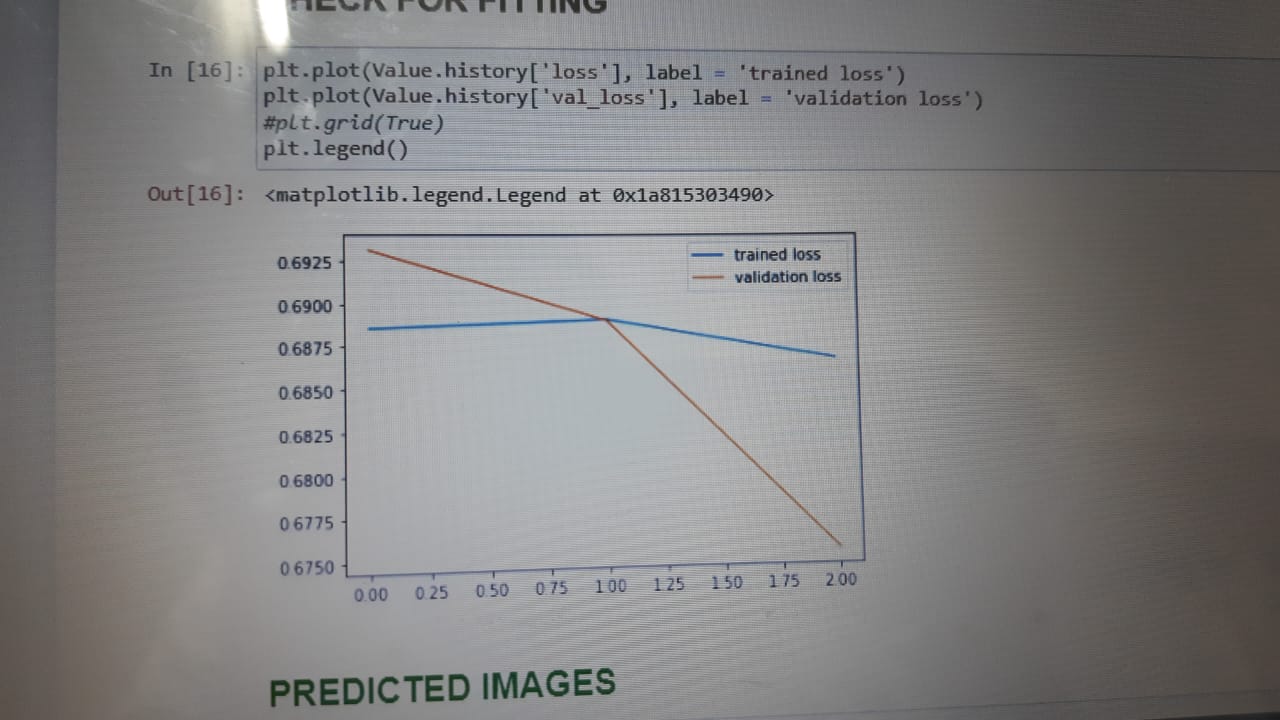
plt.plot(Value.history['loss'], label = 'trained loss')

plt.plot(Value.history['val\_loss'], label = 'validation loss')

#plt.grid(True)

plt.legend()

<matplotlib.legend.Legend at 0x1a815b0d6d0>



## PREDICTED IMAGES

## plt.figure(figsize=(10,5))

## for images, labels in test\_data.take(10):

## predictions = Cnn.predict(images)

## predlabel = []

## for mem in predictions:

## predlabel.append(class\_names[np.argmax(mem)])

## for i in range(12):

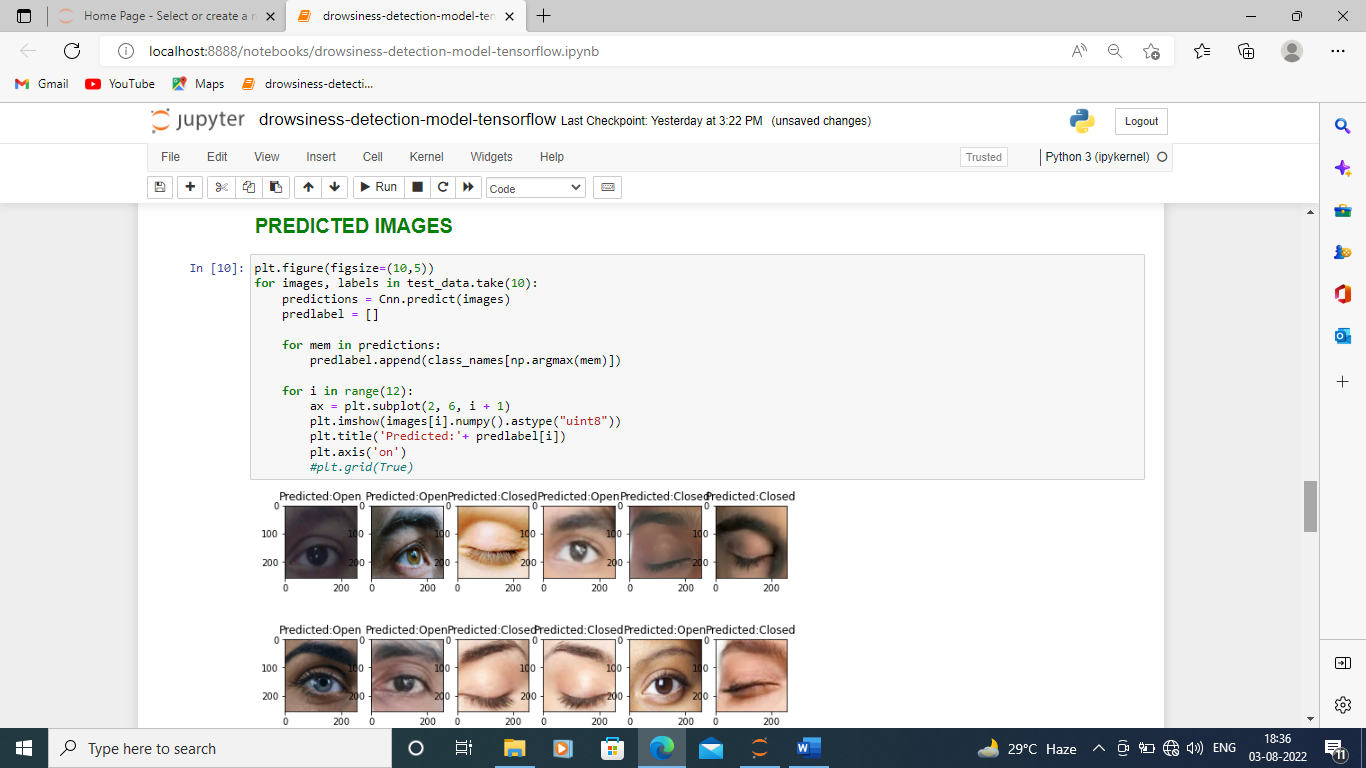
## ax = plt.subplot(2, 6, i + 1)

## plt.imshow(images[i].numpy().astype("uint8"))

## plt.title('Predicted:'+ predlabel[i])

## plt.axis('on')

## #plt.grid(True)



## LIVE IMLEMENTATION

#import cv2

#import os

from keras.models import load\_model

#import numpy as np

from pygame import mixer

#import time

mixer.init()

alarm = mixer.Sound('clock-alarm-8761.mp3')

face = cv2.CascadeClassifier('haarcascade\_frontalface\_alt.xml')

leye = cv2.CascadeClassifier('haarcascade\_lefteye\_2splits.xml')

reye = cv2.CascadeClassifier('haarcascade\_righteye\_2splits.xml')

lbl=['Close','Open']

model = load\_model('cnncat2.h5')

path = os.getcwd()

cap = cv2.VideoCapture(0)

font = cv2.FONT\_HERSHEY\_COMPLEX\_SMALL

count=0

score=0

thicc=2

rpred=[99]

lpred=[99]

while(True):

ret, frame = cap.read()

height,width = frame.shape[:2]

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face.detectMultiScale(gray,minNeighbors=5,scaleFactor=1.1,minSize=(25,25))

left\_eye = leye.detectMultiScale(gray)

right\_eye = reye.detectMultiScale(gray)

cv2.rectangle(frame, (0,height-50) , (200,height) , (0,0,0) , thickness=cv2.FILLED )

for (x,y,w,h) in faces:

cv2.rectangle(frame, (x,y) , (x+w,y+h) , (100,100,100) , 1 )

for (x,y,w,h) in right\_eye:

r\_eye=frame[y:y+h,x:x+w]

count=count+1

r\_eye = cv2.cvtColor(r\_eye,cv2.COLOR\_BGR2GRAY)

r\_eye = cv2.resize(r\_eye,(24,24))

r\_eye= r\_eye/255

r\_eye= r\_eye.reshape(24,24,-1)

r\_eye = np.expand\_dims(r\_eye,axis=0)

#rpred = model.predict\_classes(r\_eye)

predict\_x=model.predict(r\_eye)

rpred=np.argmax(predict\_x,axis=1)

if(rpred[0]==1):

lbl='Open'

if(rpred[0]==0):

lbl='Closed'

break

for (x,y,w,h) in left\_eye:

l\_eye=frame[y:y+h,x:x+w]

count=count+1

l\_eye = cv2.cvtColor(l\_eye,cv2.COLOR\_BGR2GRAY)

l\_eye = cv2.resize(l\_eye,(24,24))

l\_eye= l\_eye/255

l\_eye=l\_eye.reshape(24,24,-1)

l\_eye = np.expand\_dims(l\_eye,axis=0)

#lpred = model.predict\_classes(l\_eye)

predict\_y=model.predict(l\_eye)

lpred=np.argmax(predict\_y,axis=1)

if(lpred[0]==1):

lbl='Open'

if(lpred[0]==0):

lbl='Closed'

break

if(rpred[0]==0 and lpred[0]==0):

score=score+3

cv2.putText(frame,"Closed",(10,height-20), font, 1,(255,255,255),1,cv2.LINE\_AA)

if(rpred[0]==1 or lpred[0]==1):

score=score-5

cv2.putText(frame,"Open",(10,height-20), font, 1,(255,255,255),1,cv2.LINE\_AA)

if(score<0):

score=0

cv2.putText(frame,'Seconds:'+str(score),(100,height-20), font, 1,(255,255,255),1,cv2.LINE\_AA)

if(score>15):

#person is feeling sleepy so we beep the alarm

cv2.imwrite(os.path.join(path,'image.jpg'),frame)

try:

alarm.play()

except: # isplaying = False

pass

if(thicc<16):

thicc= thicc+2

else:

thicc=thicc-2

if(thicc<2):

thicc=2

cv2.rectangle(frame,(0,0),(width,height),(0,0,255),thicc)

cv2.imshow('output\_frame',frame)

if cv2.waitKey(1) & 0xFF == ord('s'):

break

cap.release()

cv2.destroyAllWindows()

**OUTPUT**

