AI - Enabled Face Mask Detector

To read, write, and display Images & Videos

We will set up a basic understanding of how we can begin learning image processing with OpenCV by reading, writing, and displaying images as well videos using OpenCV methods like imshow(), imread(), etc.

OpenCV

Open-source library for the Computer Vision, Machine Learning & Image Processing.

Basic Functions

After we have established a basic understanding, we will learn about various methods on images to resize, blur, etc.

Shapes and Texts

We will learn about the various shapes that we can put the images into and with corresponding text values as their labels

Joining Images

This section particularly focuses on how we can join the images together.

- Face Detection on Images, Videos, & in Real-time

- Face and Eyes Detection in Real-time

- Pedestrians Detection

- Face Mask Detection - Real Time

In [1]:

```
#Import Libraries

import cv2
import numpy as np

print(cv2.__version__)
```

```
4.5.1
```

Read, Write & Display Image

In [2]:

```
#Reading & Displaying the Image

img = cv2.imread('D:/Project/dog.jpg') #imread = To read the image from specified path

cv2.imshow("Output", img) #Displaying the Image

cv2.waitKey(0) #Delay in Milliseconds for which we want to show the Image

cv2.destroyAllWindows()
```

In [3]:

```
cv2.imwrite('D:/Project/dog2.jpg', img)
```

Out[3]:

```
True
```

Read, Write & Display Videos

In [4]:

```
#Reading & Displaying the Image
cap = cv2.VideoCapture(0) #0 - WebCam & 1 -External Camera

#4 - byte identifier which specifies the format of a Video Stream
fourcc = cv2.VideoWriter_fourcc('D','I','V','X')

out = cv2.VideoWriter('D:/Project/myVideo2.avi',fourcc,20.0,(640,480)) #No. of frames/s,
Frame Size

#Videos are sequence of Images
#Will add a while loop to capture the frame continuously

while True:
    success, frame = cap.read()
    if success == True:

        out.write(frame)
        cv2.imshow("Video", frame)

        if cv2.waitKey(1) == ord('q'): #Delay & to Break the Loop
            break
    else:
        break
```

In [5]:

```
#Reading & Displaying the Video
cap = cv2.VideoCapture(0)

cap.set(3,640) #Width
cap.set(4,480) #Height
cap.set(10,255) #Brightness

#Videos are sequence of Images
#Will add a while loop to capture the frame continuously

while True:
    success, img = cap.read() # img variable will capture the Video & success variable wi
ll tell us whether it was captured successfully or not
    cv2.imshow("Video", img)

    if cv2.waitKey(1) & 0xFF == ord('q'): #Delay & to Break the Loop
        break

cap.release() #Releases the resourcing after recording
cv2.destroyAllWindows()
```

In [6]:

```
#Converting the Image into Gray Scale

#Reading the Image
img = cv2.imread('D:/Project/dog.jpg')
imgGray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY) #Converting Image into Gray Scale
cv2.imshow('GrayScale Image', imgGray)
cv2.imwrite('D:/Project/Graydog.png', imgGray)

cv2.waitKey(0) #Delay in Milliseconds for which we want to show the Image
cv2.destroyAllWindows()
```

In [7]:

```
#BLUR Function to the BLUR Image

img = cv2.imread('D:/Project/dog.jpg')
imgBlur = cv2.GaussianBlur(img, (59,59), 0)
#(7,7) is the Kernal Size (Amount of Blur) which is always Odd. (3,3),(5,5),..etc.
#0 is Sig function (Standard Deviation along X-Direction)

cv2.imshow('Blurred Image', imgBlur)

cv2.waitKey(0) #Delay in Milliseconds for which we want to show the Image
cv2.destroyAllWindows()
```

In [8]:

```
#Edge Detector - Canny

img = cv2.imread('D:/Project/dog.jpg')
imgCanny = cv2.Canny(img, 150, 200) #Threshold Values

kernel = np.ones((5,5), np.uint8)
'''Type of the object which is unsigned integer of 8 bits
which means the values can range from 0 to 255'''

#dilate functions are used when Edges are not properly connected
imgDilation = cv2.dilate(imgCanny, kernel, iterations = 1)

#Erosion functon is used when we want to thin(Erode) the Image
imgErosion = cv2.erode(imgDilation, kernel, iterations = 1)

# Iterations is for the requied Thickness
cv2.imshow('Canny Image', imgCanny)
cv2.imshow('Dilation Image', imgDilation)
cv2.imshow('Erosion Image', imgErosion)

cv2.waitKey(0)# Delay in Milliseconds for which we want to show the Image
cv2.destroyAllWindows()
```

In [9]:

```
#Resizing the Image

import cv2
import numpy as np

img = cv2.imread('D:/Project/dog.jpg') #Reading the Image

cv2.imshow("Output", img) #Displaying the Image

print(img.shape) #Shape of the Image (Height, Width, No. of Channels(RGB))

cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
(424, 283, 3)
```

In [10]:

```
img_resize = cv2.resize(img, (500,400)) #(Width, Height)

cv2.imshow("Original Image", img)
cv2.imshow("Re-sized Image", img_resize)

cv2.waitKey(0) # Delay in Milliseconds for which we want to show the Image
cv2.destroyAllWindows()
```

Shapes & Texts

In [11]:

```
#0 means Black

img = np.zeros((512, 512)) #Print a Black Image. It is Gray Scale Image
cv2.imshow("Output", img)
print(img.shape)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
(512, 512)
```

In [12]:

```
#0 means Black

img = np.zeros((512, 512, 3), np.uint8) #Print a Black Image with 3 Channels RGB.

#img[:] = 255, 0, 0

cv2.line(img, (50,30), (400,250), (231,255,9),3)
   #Starting Point, Ending Point, Color, Thickness
cv2.imshow("Output", img)

#print(img.shape)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [13]:

```
#Rectangle

cv2.rectangle(img, (20,20), (150,150), (255,136,2),cv2.FILLED)

#cv2.FILLED is used to Fill the Rectangles

cv2.imshow("Output", img)

#print(img.shape)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [14]:

```
#Circle

cv2.circle(img, (400,200), 30, (0,255,255),cv2.FILLED)

#cv2.FILLED is used to Fill the Rectangles

cv2.imshow("Output", img)

#print(img.shape)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [15]:

```
#Text

cv2.putText(img, "Text Here", (300, 100), cv2.FONT_HERSHEY_COMPLEX, 1, (0,150,0), 3)
#                           Starting Point,   Font,              Scale,  Color, Thickne
ss
cv2.imshow("Output", img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Face Detection on an Image

In [16]:

```
faceCascade = cv2.CascadeClassifier("D:/Project/haarcascade_frontalface_default.xml")
img = cv2.imread('D:/Project/Guru.JPG')

img = cv2.resize(img, (600,640))
imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = faceCascade.detectMultiScale(imgGray, 1.1, 4)

for (x, y, w, h) in faces:
    cv2.rectangle(img, (x,y),(x+w, y+h), (255,255,0),2)

cv2.imshow("Output", img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Face Detection in Videos

In [17]:

```
import cv2
cap = cv2.VideoCapture('D:/Project/myVideo.avi')

#Videos are just a sequence of Images
#So, WHILE LOOP has been added to capture the frames continuously

faceCascade = cv2.CascadeClassifier("D:/Project/haarcascade_frontalface_default.xml")

while True:
    success, frame = cap.read()

    #Fame variable will capture the Video & Success variable will tell us whether it was
captured successfully or not

    imgGray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(imgGray, 1.1, 4)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y),(x+w, y+h), (0,255,0),2)

    cv2.imshow("Video", frame)

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

cap.release() #Release the resources after Recording
cv2.destroyAllWindows()
```

Face Detection in Real Time

In [18]:

```
import cv2
cap = cv2.VideoCapture(0)

#Videos are just a sequence of Images
#So, WHILE LOOP has been added to capture the frames continuously

faceCascade = cv2.CascadeClassifier("D:/Project/haarcascade_frontalface_default.xml")

while True:
    success, frame = cap.read()
```

```
    #Fame variable will capture the Video & Success variable will tell us whether it was captured successfully or not

    imgGray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(imgGray, 1.1, 4)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y),(x+w, y+h), (0,255,0),2)

    cv2.imshow("Video", frame)

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

cap.release() #Release the resources after Recording
cv2.destroyAllWindows()
```

Face & Eyes Detection in Real Time

In [19]:

```
cap = cv2.VideoCapture(0)

faceCascade = cv2.CascadeClassifier("D:/Project/haarcascade_eye.xml")
faceCascade1 = cv2.CascadeClassifier("D:/Project/haarcascade_frontalface_default.xml")

while True:
    success, frame = cap.read()

    imgGray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    eyes = faceCascade.detectMultiScale(imgGray, 1.1, 4)
    faces = faceCascade1.detectMultiScale(imgGray, 1.1, 4)

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x,y), (x+w, y+h), (255,17,17), 2)

    for (x, y, w, h) in eyes:
        cv2.rectangle(frame, (x,y), (x+w, y+h), (0,0,0), 2)

    cv2.imshow("Video", frame)

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

cap.release() #Release the resources after Recording
cv2.destroyAllWindows()
```

Pedestrians Detection

In [20]:

```
cap = cv2.VideoCapture('D:/Project/Pedestrians.mp4')

faceCascade = cv2.CascadeClassifier("D:/Project/haarcascade_fullbody.xml")

while True:
    success, frame = cap.read()

    imgGray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(imgGray, 1.1, 4)

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,0,0),2)

    cv2.imshow("Video", frame)
```

```
    if cv2.waitKey(1) == ord('q'):
        break

cap.release() #Release the resources after Recording
cv2.destroyAllWindows()
```

MASK DETECTION in Real Time

In [21]:

```
import tensorflow as tf

from tensorflow.keras.models import load_model
detector = load_model(r'D:/Project/dummy.model')
```

Starting the stream for Mask Detection in Real Time

To detect the mask in a live stream we will follow the following steps:

1. **start the video stream**
2. **Capture the frame from the stream**
3. **Resize the frame**
4. **Detect faces in the frame using haarcascade classifier**
5. **Get the predictions using the saved model**
6. **Depending on the results draw rectangle and put text on the faces accordingly**

The following are some of the important functions that we will use for our process.
img_to_array() - Converts the image to a numpy array
detectmultiscale() - Detects objects in the image
tf.expand_dims() - Inserts a dimension of length 1 and returns a tensor
tf.nn.softmax() - used for computing softmax activations
numpy.argmax() - returns the indices of the values that are maximum along the x-axis

Alternatively, you can choose other face detection technique instead of haarcascade, since it is the most basic technique to detect faces. And sometimes the results are not very efficient. You can use the opencv caffe model for face detection for variation in your results

In [22]:

```
import tensorflow as tf
import cv2
import numpy

cap = cv2.VideoCapture(0)

classifier = cv2.CascadeClassifier(r"D:/Project/haarcascade_frontalface_default.xml")
```

In [23]:

```
#Using the Loops to watch the stream in real time

while True:
    (success, frame) = cap.read() #Reading the frame from Stream
    new_img = cv2.resize(frame, (frame.shape[1]//1, frame.shape[0]//1))

    #Resize the frame to speed up th process of detection
    face = classifier.detectMultiScale(new_img)

    #Detecting faces from the frame (ROI)
    for x,y,w,h in face:
        try:
            face_img = new_img[y:x+h, x:x+w]
            #getting the coordinates for the face detected
            resized = cv2.resize(face_img,(224,224))
```

```
            #resizing the face detected to fit into the model in the shape(224,224)
            img_array = tf.keras.preprocessing.image.img_to_array(resized)
            #converting the detected image into an array
            img_array = tf.expand_dims(img_array,0)
            #expanding the dimensions to fit in the model
            predictions = detector.predict(img_array)
            #making predictions on the ROI
            score = tf.nn.softmax(predictions[0])
            #getting the results
            label = numpy.argmax(score)
        except Exception as e:
            print('bad frame')

        if label == 0:
            cv2.rectangle(new_img,(x,y),(x+w,y+h),(0,255,0),2)
            cv2.putText(new_img,"mask",(x,y),cv2.FONT_HERSHEY_SIMPLEX,0.8,(0,255,0),2)
        elif label == 1:
            cv2.rectangle(new_img,(x,y),(x+w,y+h),(0,0,255),2)
            cv2.putText(new_img,"no_mask",(x,y),cv2.FONT_HERSHEY_SIMPLEX,0.8,(0,0,255),2
)
        else:
            None
    #Displaying the window after predicting the outcome
    cv2.imshow('face_window', new_img)
    print(numpy.argmax(score), 100*numpy.max(score))
    #waitkey to terminate the loop
    key = cv2.waitKey(10)
    if key == ord('q'):
        break
#Release the Stream
cap.release()
cv2.destroyAllWindows()
```

```
0 0.27133943513035774
0 0.27136080898344517
0 0.27136080898344517
0 0.27136080898344517
0 0.27136080898344517
0 0.27136080898344517
```