Face Mask Detection Using OpenCV in Python

Tushar Singh

June 04, 2021

Abstract

In this project, I tried to experiment the concepts of machine learning and OpenCV, and to explore how machine learning algorithms can be used in real world applications. After successfully completing the project here is the final report.

Keywords: Machine Learning, Open CV, Classification, Supervised learning, Support Vector Machine

Introduction

Introduction to Image Processing

Before implementing face mask detection problem, first we need to understand that how to handle images. Images are simply a collection of colors in red, green and blue format. As a human we see an image with some object or shape in it, but for computer it is just an array with color values range from 0 to 255.

The way computer sees anything is different from the way human see an image. But that's the good news for us because if we got an array of the image than it becomes simple for us to implement any algorithm on that array.

Steps to Perform Image Processing :

- Load images using Python or any other programming you are working on.
- Convert images into array
- And finally apply some algorithm on that array

Another good thing is that we have a library known as OpenCV which will help us to read the image and return array of color pixels.

Introduction to OpenCV

- OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.
- The library has more than 2000 optimized algorithms.
- It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS.
- Will help us to load images in Python and convert them into array.
- Each index of array represents (red, green, blue) color pixel which ranges from 0 to 255.
-

Features of OpenCV

- Face Detection
- Geometric Transformations
- Image Thresholding
- Smoothing Images
- Canny Edge Detection
- Background Removals
- Image Segmentation

Code:- 1) File1.py

```
import cv2
img = cv2.imread('Tushar.jpg')
img.shape
img[0]
import matplotlib.pyplot as plt
while True:
    cv2.imshow('result',img)
    if cv2.waitKey(2) == 27:
        break
cv2.destroyAllWindows()
haar_data = cv2.CascadeClassifier('data.xml')
haar_data.detectMultiScale(img)
while True:
    faces = haar_data.detectMultiScale(img)
    for x,y,w,h in faces:
        cv2.rectangle(img,(x,y) , (x+w, y+h) , (255,0,255) , 4)
    cv2.imshow('result',img)
    #27-ASCII of Escape
    if cv2.waitKey(2) == 27:
        break
cv2.destroyAllWindows()
plt.imshow(img)
capture = cv2.VideoCapture(0)
data = []
while True:
```

```
    flag, img = capture.read()
    if flag:
        faces = haar_data.detectMultiScale(img)
        for x,y,w,h in faces:
            cv2.rectangle(img,(x,y) , (x+w, y+h) , (255,0,255) , 4)
            face = img[y:y+h , x:x+w, :]
            face = cv2.resize(face,(50,50))
            print(len(data))
            if len(data) < 400:
                data.append(face)
        cv2.imshow('result', img)
        if cv2.waitKey(2) == 27 or len(data) >= 200:
            break
capture.release()
cv2.destroyAllWindows()
import numpy as np
x = np.array([3,4,5])
x
 p.save('withouut_mask.npy', data)
 np.save('with_mask.npy', data)
```

Explanation(file1.py):-

So first load OpenCV package, after importing OpenCV first let's read an image. So, when we read any image using OpenCV it returns object of numpy array by default and using img.shape we are checking the height and width of image and also it returns 3 which is the color channel of image. Now we can see the values of array are the color values actually. The image will open in same file so to open in new window we have written a separate code. Ie first we started a while loop then using cv2.waitKey. Now what cv2.waitKey is doing. 2 is the time in milliseconds and 27 is ASCII number of escape key. So it means if you press escape then loop is going to break and in last line it will destroy or close all the windows that are opened by OpenCV.

Face Detection Using OpenCV
So now we are going to see how to detect face from an image. Face detection algorithm was introduced by Viola and Jones in 2001. They divided this algorithm in four stages :

1. Haar Features Selection
2. Integral Images
3. AdaBoost
4. Cascading Classifier

We are using the haar cascade data XML file which is going to help us to detect faces from the image.

We will iterate over the array returned to us by detectMultiScale method and put x,y,w,h in cv2.rectangle After concatenating all the codes we get a desired output.

Face Mask Detection
Now we are going to start with face mask detection. First let's understand the process of it

- Collect face data with and without mask.
- Train data using machine learning.
- Do prediction on live data using camera.

To implement this case study, we need a lot of images of people wearing a mask and not wearing a mask. So first we need to collect data and we are going to collect data using our own camera

Save the data in a numpy file. Now we can load the data anywhere and start processing it to apply machine learning on it.

2) File2.py

```
import numpy as np
import cv2
with_mask = np.load('with_mask.npy')
without_mask = np.load('withouut_mask.npy')
with_mask.shape
without_mask.shape
with_mask = with_mask.reshape(200,50 * 50 * 3)
without_mask = without_mask.reshape(200,50 * 50 * 3)
with_mask.shape
without_mask.shape
X = np.r_[with_mask, without_mask]
X.shape
labels = np.zeros(X.shape[0])
labels[200:] = 1.0
names = {0 : 'Mask', 1 : 'NO Mask'}
#svm = Support vector machine
#SVC - Support vector classification
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,labels, test_size=0.25)
 x_train.shape
 #PCA - principal component analysis
from sklearn.decomposition import PCA
 # 3 - 3D data
pca = PCA(n_components = 3)
x_train = pca.fit_transform(x_train)
```

```
 x_train[0]
 x_train.shape
 x_train, x_test, y_train, y_test = train_test_split(X,labels, test_size=0.25)
 svm = SVC()
svm.fit(x_train, y_train)
 #x_test = pca.transform(x_test)
y_pred = svm.predict(x_test)
accuracy_score(y_test, y_pred)
haar_data = cv2.CascadeClassifier('data.xml')
capture = cv2.VideoCapture(0)
data = []
font = cv2.FONT_HERSHEY_COMPLEX
while True:
    flag, img = capture.read()
    if flag:
        faces = haar_data.detectMultiScale(img)
        for x,y,w,h in faces:
            cv2.rectangle(img,(x,y) , (x+w, y+h) , (255,0,255) , 4)
            face = img[y:y+h , x:x+w, :]
            face = cv2.resize(face,(50,50))
            face = face.reshape(1,-1)
            # face = pca.transform(face)
            pred= svm.predict(face)
            n = names[int(pred)]
            cv2.putText(img, n, (x,y), font, 1, (244,250,255), 2)


          print(n)
```

```
        cv2.imshow('result', img)
        if cv2.waitKey(2) == 27:
            break
capture.release()
cv2.destroyAllWindows()
```

Explanation(File2.py):-

After importing the cv2 and numpy files we can see that data is loaded with the shape in my case it is 200, 50, 50, 3. Here 200 is the number of images we have collected, 50, 50 is the size of each image, 3 is the color channel (red, green, blue) We can reshape the data to make it 2D, And we will concatenate the data into a single array. Using NPR will help you to store data row wise. So our features are ready. Now we need target variable. So let's create one array of zeros and assign first 200 indexes as zero and next 200 indexes as one. Because first 200 images belong to faces with mask and next 200 images belong to faces without mask. Now we can apply machine learning on our data after dividing it into train and test. The algorithm we are using is SVM here. And after training this data on SVM we are getting accuracy of 98%.

So finally, we can test our faces with or without mask and check whether this algorithm is able to identity you that you are wearing a mask or not.

OUTPUT