Face Mask Detection System – Siddhartha Sen

What

1. It is a computer vision application which can first detect the faces into an image and then detect whether the face has a mask or not and then save the faces of no mask. (Computer Vision Application is Related to applying ML on Video or image data .)
2. This Application can get the image from different sources such as image , video , web cam , IP camera.
3. This Application will be used using a browser on a Local Area Network.

Why

1. This system can be used in many places such as hospitals, research labs, Nuclear Power Plants, Air Borne Disease areas, High AQI Areas (Air Quality Index).
2. This kind of project shows our Machine Learning Knowledge, programming skills , project development skills.

How

Backend : Code , Functionalities

Face Detection : OpenCV (CV stands for Computer Vision)

Mask Detection : Tensorflow

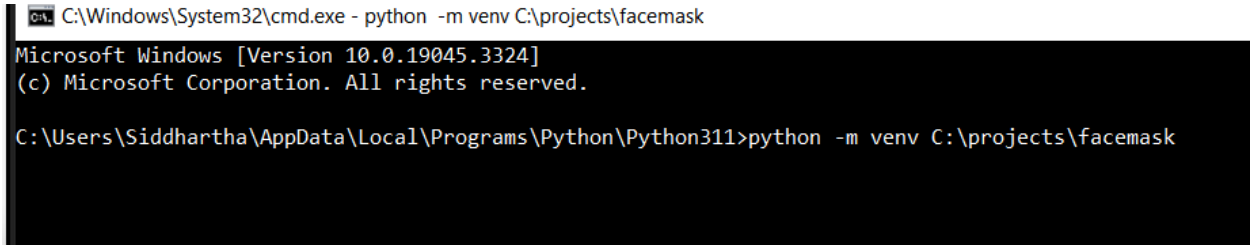
Frontend : Interface

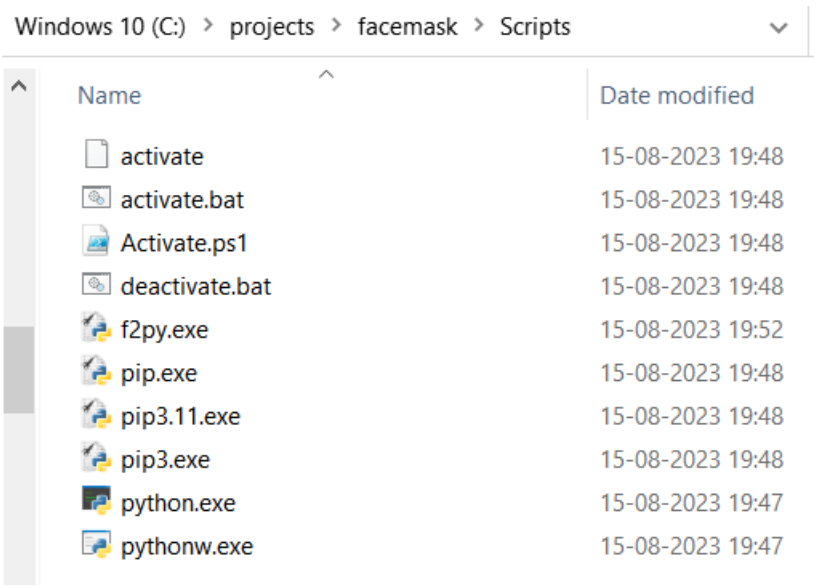
Web Application : Streamlit

Steps Under How part

1. **Overview and OpenCV**
2. **Face Detection**
3. **Mask Detection**
4. **Frontend Development**
5. **Feature Integration**

1.We Create the virtual environment.





2.We Activate the environment everytime we use it.

OpenCV

import cv2 # cv2 is the open cv package

img = cv2.imread("woman.jpg") # To read a basic image and show it.

cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)# To customise the window so that we can stretch the screen size .

cv2.imshow("sidd window",img) # img will be shown in sidd window

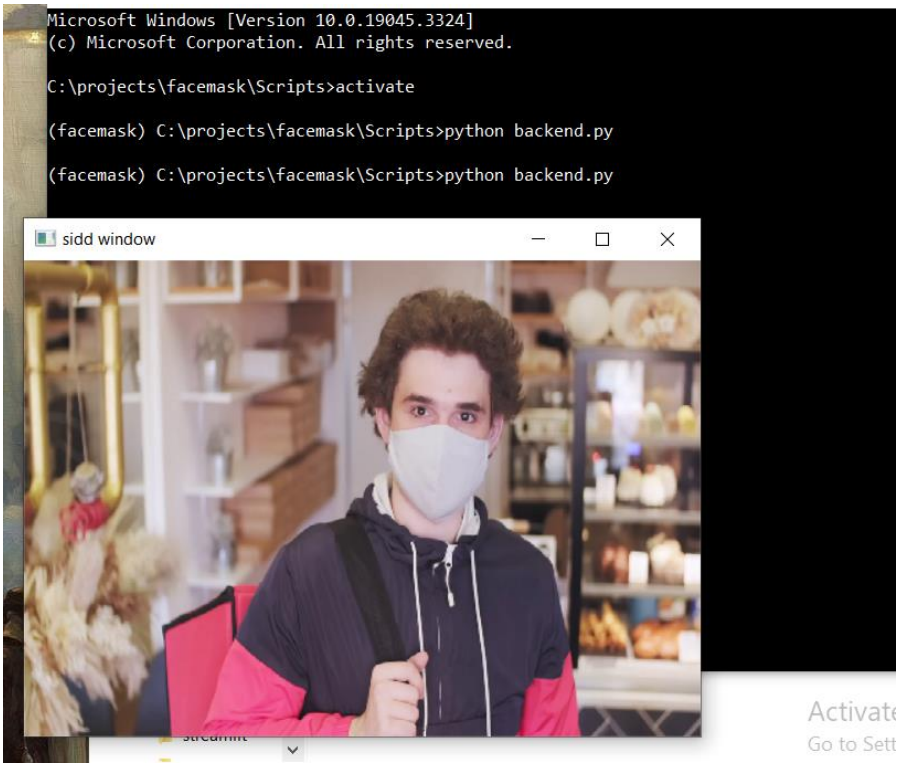
 # cv2.waitKey(100) , It waits the user to press a key for 100 milliseconds

cv2.waitKey(0) # 0 means the window will wait till we press a key

cv2.destroyAllWindows()

We downloaded a video from pexels.com and stored it in the Scripts folder of our environment.

```
# To read a video and show it
import cv2
vid=cv2.VideoCapture("mask.mp4")
#reading video means reading multiple images from that video
#So we will run a loop.
while(vid.isOpened()):
    flag,frame = vid.read()
    if(flag): # means if flag == True
        cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
        cv2.imshow("sidd window",frame)
        cv2.waitKey(0)
    else:
        break
cv2.destroyAllWindows()
```



```
# flag will store a boolean value indicating
# whether the video has a readable image or not.
# frame will contain the actual images .

# We will show the video as video now .
# For that we will have to put the value of how many milliseconds in
# waitKey(? milliseconds) .
# So if it is 24 FPS . Means 24 frames in 1 second i.e. 1000 milliseconds
# Therefore, 1 frame in 1000/24 milliseconds.

# 1000/24 is 41 nearly .


import cv2
vid=cv2.VideoCapture("mask.mp4")
#reading video means reading multiple images from that video
#So we will run a loop.
while(vid.isOpened()):
    flag,frame = vid.read()
    if(flag): # means if flag == True
        cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
        cv2.imshow("sidd window",frame)
        cv2.waitKey(41)
    else:
        break
cv2.destroyAllWindows()

|
# we can alter and see the different ffp values like 30,40 etc.
```

By the above code , the video runs and stops at the end. But how to stop the video at any time between the starting and ending of the video ?

When we press a key on keyboard , there is a Unicode value according to that key. That value is stored inside the waitKey function. We will store that Unicode value in a variable say k and then we will make a condition according to that. If the value of k matches with Unicode value of x , then we will break the loop.

‘ord’ function will give the Unicode value of x.

```
import cv2
vid=cv2.VideoCapture("mask.mp4")
#reading video means reading multiple images from that video
#So we will run a loop.
while(vid.isOpened()):
    flag,frame = vid.read()
    if(flag): # means if flag == True
        cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
        cv2.imshow("sidd window",frame)
        k = cv2.waitKey(33)
        if(k == ord('x')):
            break
    else:
        break
cv2.destroyAllWindows()
```

For webcam image , we need to set the value inside VideoCapture as 0 . 0 is for the web cam of laptop that we Use in meetings and all .

```
import cv2
# To read a webcam and show it.
vid=cv2.VideoCapture(0)
while(vid.isOpened()):
   flag,frame = vid.read()
   if(flag): # means if flag == True
      cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
      cv2.imshow("sidd window",frame)
      k = cv2.waitKey(33)
```

```
        if(k == ord('x')):
            break
    else:
        break
cv2.destroyAllWindows()
```

IP camera launches a hotspot and those with access to that hotspot can see the IP camera footage . This is hence a wireless process unlike a surveillance camera which transmits footage through wires.

Since we don't have a IP camera or a hardware camera, we will simulate that using an app through our smart phone. We will use our android phone as an IP camera for testing purpose to see how to access it .

To access footage on laptop through a phone , install ip webcam application from playstore on that phone , click on 3 dots above and start server , copy the first ipv4 address and paste that link ending with '/video' for eg: http://ipv4address/video on a browser.

Now we will do the same through OpenCV.

The code is

```
# accessing footage from phone through opencv
import cv2
vid=cv2.VideoCapture("http://192.168.0.108:8080/video")
#reading video means reading multiple images from that video
#So we will run a loop.
while(vid.isOpened()):
    flag,frame = vid.read()
    if(flag): # means if flag == True
        cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
        cv2.imshow("sidd window",frame)
        k = cv2.waitKey(33)
        if(k == ord('x')):
            break
    else:
        break
cv2.destroyAllWindows()
```

Now we will detect mask on faces . For that we will need to make a rectangle around the faces available. To make that rectangle , we need 4 parameters , x1,y1,l (length of the rectangle), h (height of the rectangle).

For colouring the rectangle , we need color codes . We take help of this site for that .

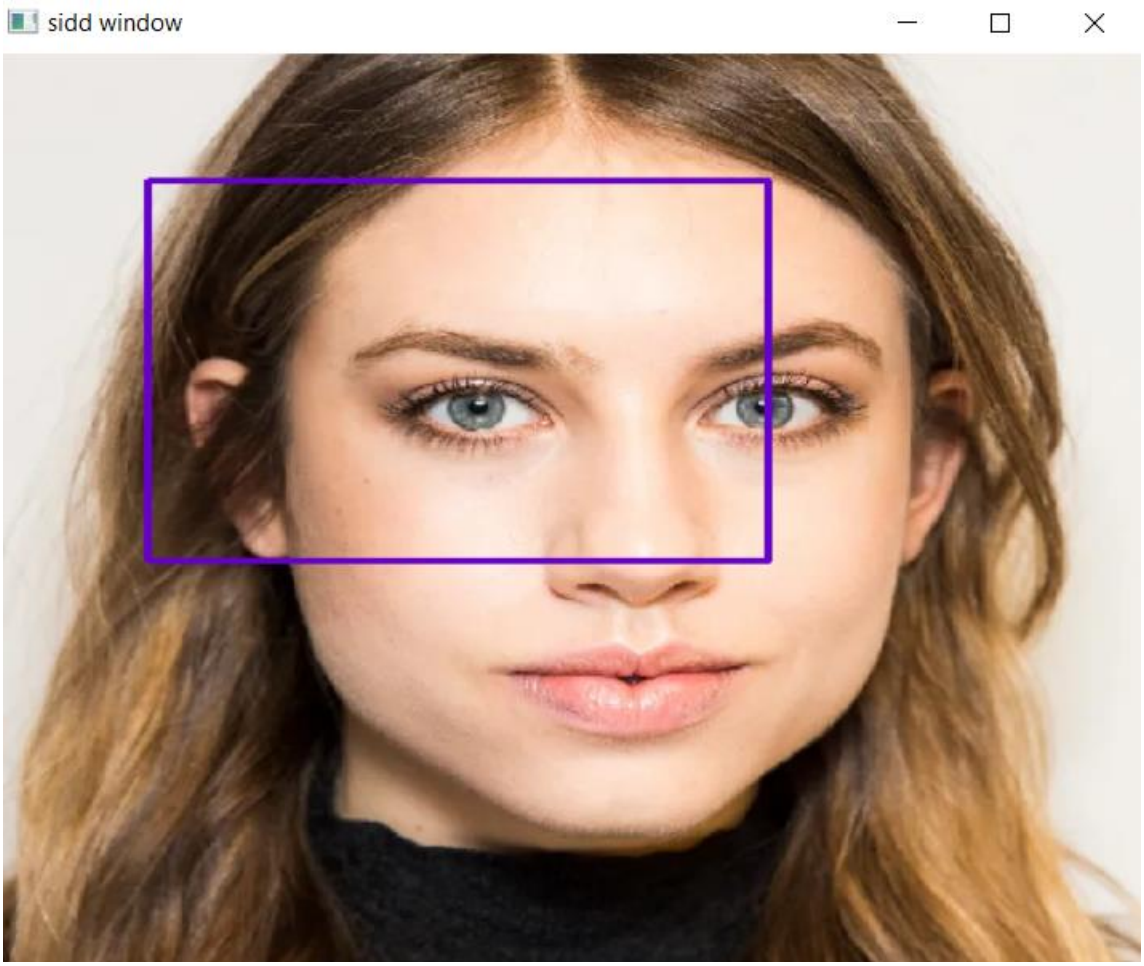
https://www.rapidtables.com/web/color/RGB_Color.html

Face Detection

The code for manually making the rectangle :

```
import cv2 # cv2 is the open cv package
img = cv2.imread("woman.jpg")
# To read an image and draw a rectangle on it and show it.
# general command
# cv2.rectangle(img,(x1,y1),(l,h),(blue,green,red color codes),rectangle outline width)
# l=200 , h = 50 here , x1=100,y1=100 , initially we took these
# to judge the recatngle area so that later we can gauge and use more precise
# parameter values.
cv2.rectangle(img,(100,100),(500,400),(204,0,102),4) # for purple color
# To customise the window so that we can stretch the screen size .
cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
cv2.imshow("sidd window",img) # img will be shown in sidd window
# cv2.waitKey(100) , It waits the user to press a key for 100 milliseconds
cv2.waitKey(0) # 0 means the window will wait till we press a key
cv2.destroyAllWindows()
```

```
C:\projects\facemask\Scripts>activate

(facemask) C:\projects\facemask\Scripts>python facedet.py
```



We will paste the face.xml file from FMD folder into our virtual environment scripts folder . That xml file contains data about the picture .

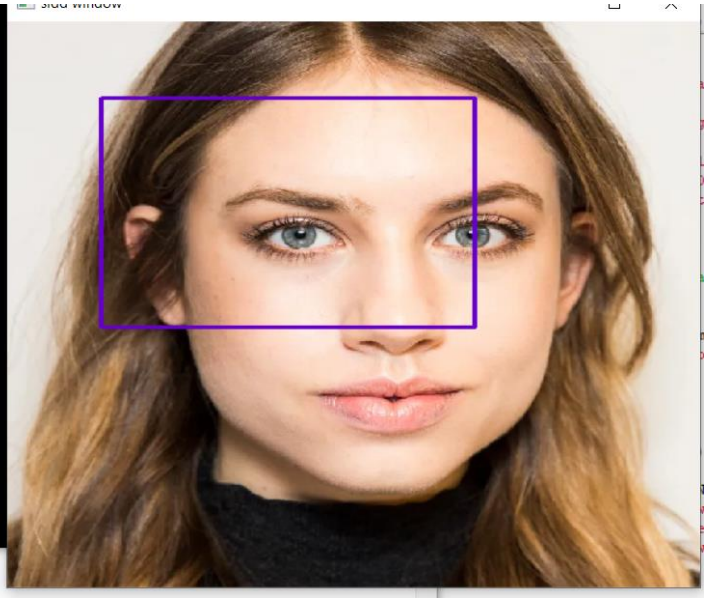
For Detecting one face through some functions , the code :

facedetxml.py - C:\projects\facemask\Scripts\facedetxml.py (2.7.10)

File Edit Format Run Options Window Help

```
# for detecting one face.

import cv2 # cv2 is the open cv package
img = cv2.imread("woman.jpg")
# To read an image and draw a rectangle on it and show it.
# general command
# cv2.rectangle(img,(x1,y1),(l,h),(blue,green,red color codes),rectangle outline width)
# l=200 , h = 50 here , x1=100,y1=100 , initially we took these
# to judge the recatngle area so that later we can gauge and use more precise
# parameter values.

# we will make a facemodel to detect face.
facemodel = cv2.CascadeClassifier("face.xml") # model got created and trained
# with this command.
# Now detection
faces = facemodel.detectMultiScale(img)
# MultiScale function will return coordinates of all faces inside the image.
# faces will store the coordinates.
print(faces)
cv2.rectangle(img,(100,100),(500,400),(204,0,102),4) # for purple color
# To customise the window so that we can stretch the screen size .
cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
cv2.imshow("sidd window",img) # img will be shown in sidd window
# cv2.waitKey(100) , It waits the user to press a key for 100 milliseconds
cv2.waitKey(0) # 0 means the window will wait till we press a key
cv2.destroyAllWindows()
```

```
(facemask) C:\projects\facemask\Scripts>python facedetxml.py
[[159  95 494 494]]
```



The values in the list within the bigger list are the x,y,x+l,y+h values of this particular face in this image . But we haven't used those coordinates while showing the image which is why the rectangle is not properly covering the face. We do that in

the next code below which is applicable for multiple images also.

P.T.O.

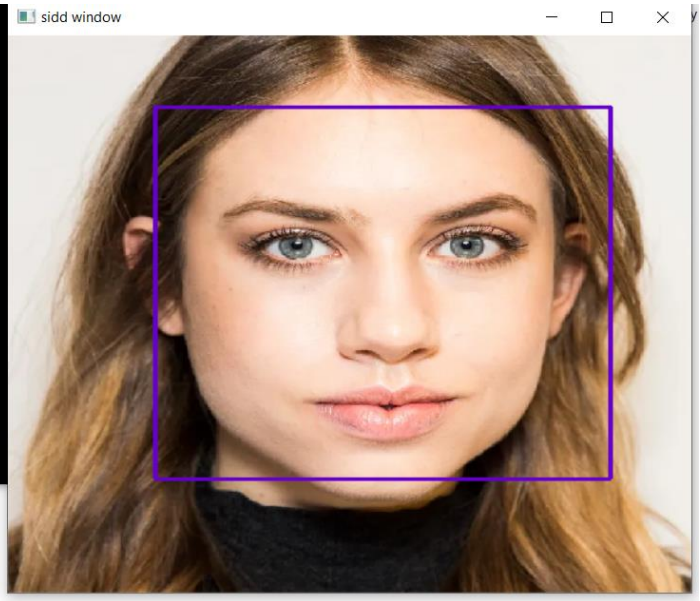
For multiple face detection , the code :

multiplefaces.py - C:\projects\facemask\Scripts\multiplefaces.py (2.7.10)

File Edit Format Run Options Window Help

```
# for detecting multiple faces.

import cv2 # cv2 is the open cv package
img = cv2.imread("woman.jpg")
# To read an image and draw a rectangle on it and show it.
# general command
# cv2.rectangle(img,(x1,y1),(l,h),(blue,green,red color codes),rectangle outline width)
# l=200 , h = 50 here , x1=100,y1=100 , initially we took these
# to judge the recatngle area so that later we can gauge and use more precise
# parameter values.

# we will make a facemodel to detect face.
facemodel = cv2.CascadeClassifier("face.xml") # model got created and trained
# with this command.
# Now detection
faces = facemodel.detectMultiScale(img)
# MultiScale function will return coordinates of all faces inside the image.
# faces will store the coordinates.
print(faces)
# for multiple faces
for (x,y,l,h) in faces:
    cv2.rectangle(img,(x,y),(x+l,y+h),(204,0,102),4) # for purple color
# To customise the window so that we can stretch the screen size .
cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
cv2.imshow("sidd window",img) # img will be shown in sidd window
# cv2.waitKey(100) , It waits the user to press a key for 100 milliseconds
cv2.waitKey(0) # 0 means the window will wait till we press a key
cv2.destroyAllWindows()
```

```
(facemask) C:\projects\facemask\Scripts>python multiplefaces.py
[[159  95 494 494]]
```



Here we have used x,y,x+l,y+h(which is 159,95,494,494 here) within cv2.rectangle within for loop for detecting all faces within the image .

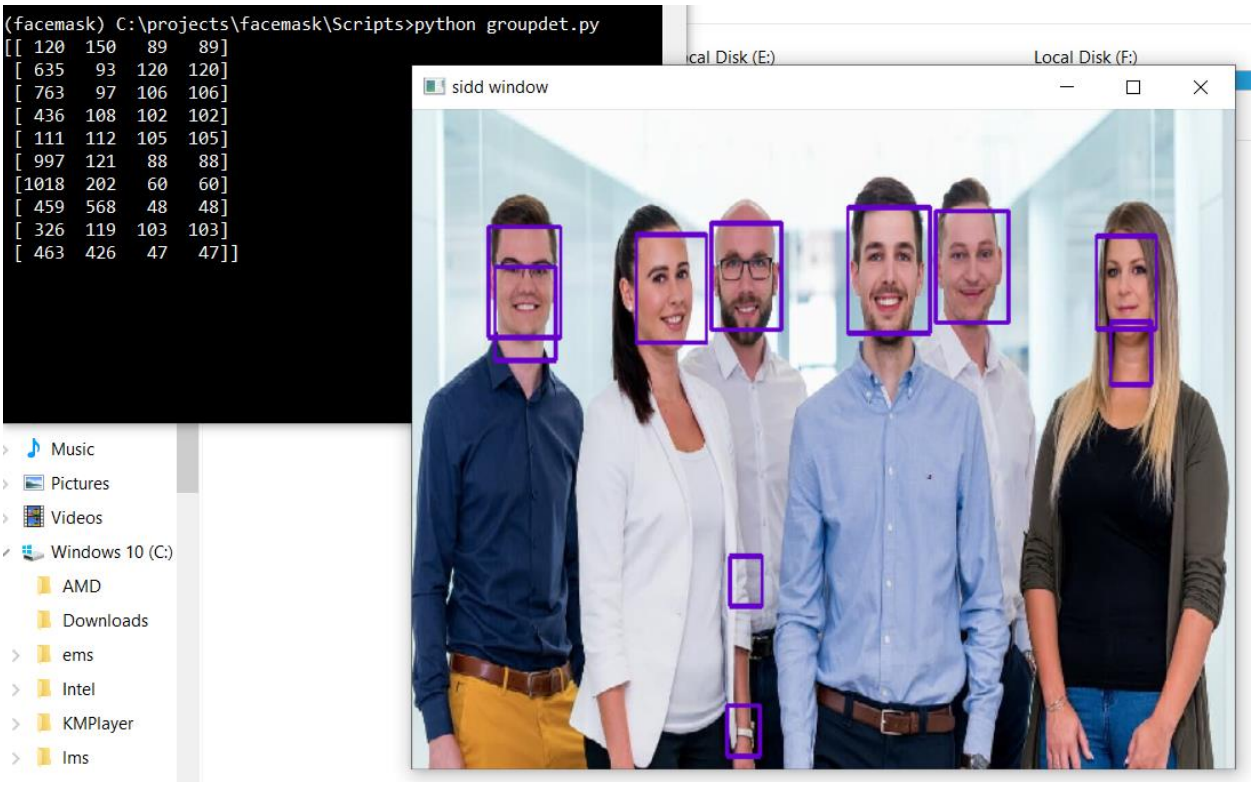
Now we will detect multiple faces within a group pic , the code :

groupdet.py - C:\projects\facemask\Scripts\groupdet.py (2.7.10)

File Edit Format Run Options Window Help

```
# for detecting multiple faces in a group.

import cv2 # cv2 is the open cv package
img = cv2.imread("group5.jpg")
# To read an image and draw a rectangle on it and show it.
# general command
# cv2.rectangle(img,(x1,y1),(l,h),(blue,green,red color codes),rectangle outline width)
# l=200 , h = 50 here , x1=100,y1=100 , initially we took these
# to judge the recatngle area so that later we can gauge and use more precise
# parameter values.

# we will make a facemodel to detect face.
facemodel = cv2.CascadeClassifier("face.xml") # model got created and trained
# with this command.
# Now detection
faces = facemodel.detectMultiScale(img)
# MultiScale function will return coordinates of all faces inside the image.
# faces will store the coordinates.
print(faces)
# for multiple faces
for (x,y,l,h) in faces:
    cv2.rectangle(img,(x,y),(x+l,y+h),(204,0,102),4) # for purple color
# To customise the window so that we can stretch the screen size .
cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
cv2.imshow("sidd window",img) # img will be shown in sidd window
# cv2.waitKey(100) , It waits the user to press a key for 100 milliseconds
cv2.waitKey(0) # 0 means the window will wait till we press a key
cv2.destroyAllWindows()
```

P.T.O.



The 10 lists within one bigger list signifies the x,y,x+l,y+h values of all 10 rectangles in the group picture.

Now , it is seen that sometimes the model is not being able to detect a face or it is making rectangles where there are no faces or multiple rectangles near a face . That is beacause of the limitation of the training data ‘face.xml’. Those are just inaccuracies.

For detecting a single face within a video , the code :

facedetvid.py - C:\projects\facemask\Scripts\facedetvid.py (2.7.10)

File Edit Format Run Options Window Help

```
import cv2
facemodel=cv2.CascadeClassifier("face.xml")
vid=cv2.VideoCapture("mask.mp4")
#reading video means reading multiple images from that video
#So we will run a loop.
while(vid.isOpened()):
    flag,frame = vid.read()
    if(flag):# means if flag == True
        faces=facemodel.detectMultiScale(frame) # this command detects the faces within every frame of the video .
        for (x,y,l,h) in faces:
            cv2.rectangle(frame,(x,y),(x+l,y+h),(204,0,102),4)
        cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
        cv2.imshow("sidd window",frame)
        k = cv2.waitKey(10) # this value is a measure of how long to stop before moving to the nexr frame
        if(k == ord('x')):
            break
    else:
        break
cv2.destroyAllWindows()
```

We activate the environment from cmd and type the command python facedetvid.py

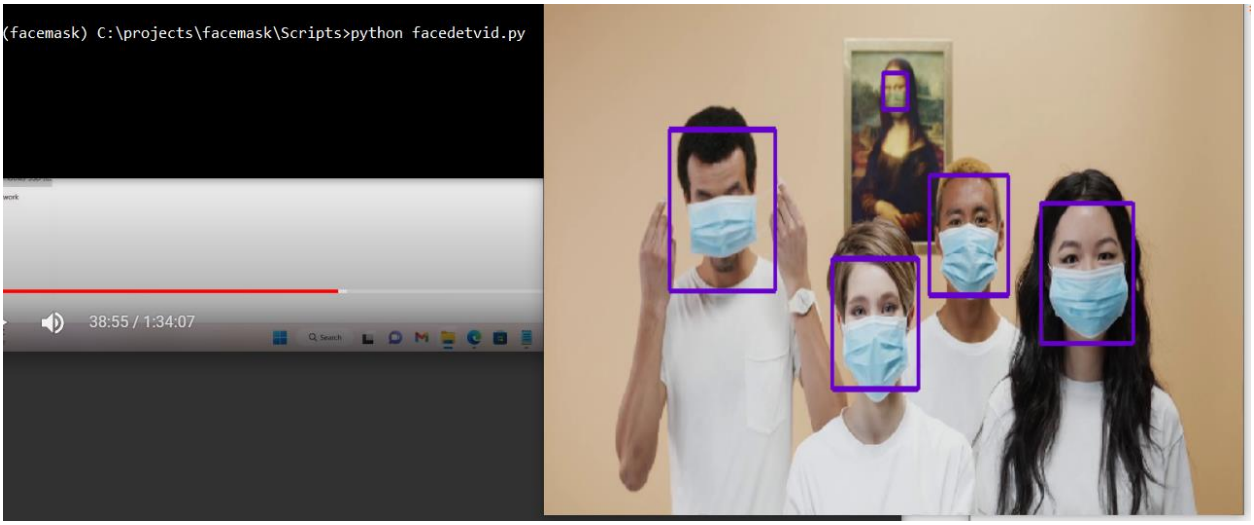
We will see the rectangular box covering face and also some other boxes due to inaccuracies.

For Detecting multiple faces from a video , we use the following code :

facedetvid.py - C:\projects\facemask\Scripts\facedetvid.py (2.7.10)

File Edit Format Run Options Window Help

```
import cv2
facemodel=cv2.CascadeClassifier("face.xml")
vid=cv2.VideoCapture("groupvidface.mp4")
#reading video means reading multiple images from that video
#So we will run a loop.
while(vid.isOpened()):
    flag,frame = vid.read()
    if(flag):# means if flag == True
        faces=facemodel.detectMultiScale(frame) # this command detects the
        for (x,y,l,h) in faces:
            cv2.rectangle(frame,(x,y),(x+l,y+h),(204,0,102),4)
        cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
        cv2.imshow("sidd window",frame)
        k = cv2.waitKey(10) # this value is a measure of how long to stop b
        if(k == ord('x')):
            break
    else:
        break
cv2.destroyAllWindows()
```



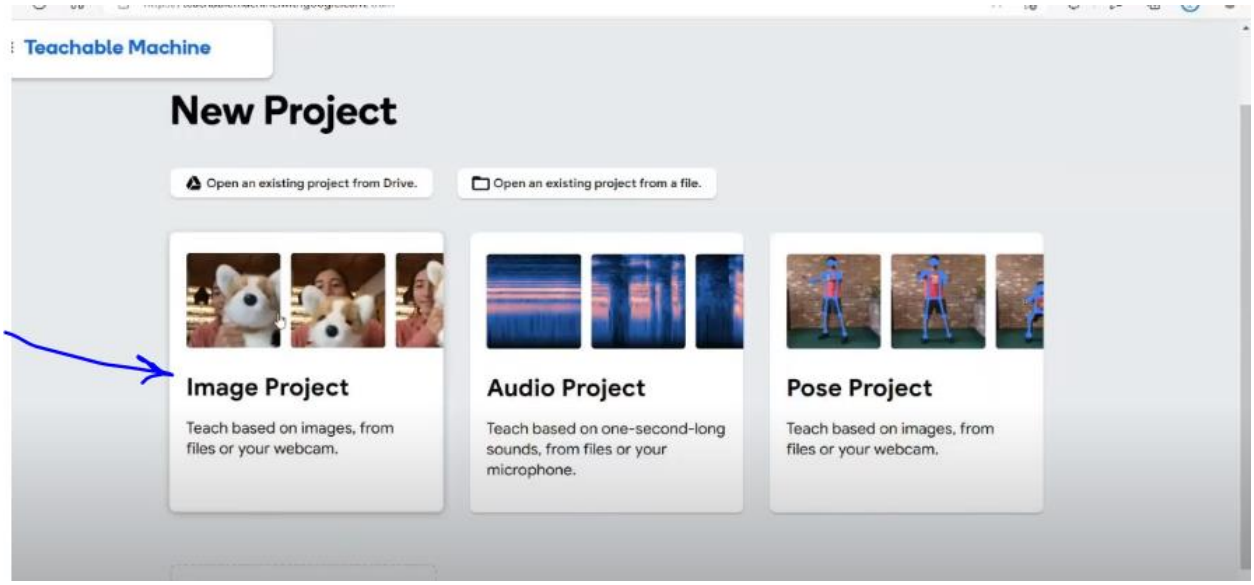
Mask Detection

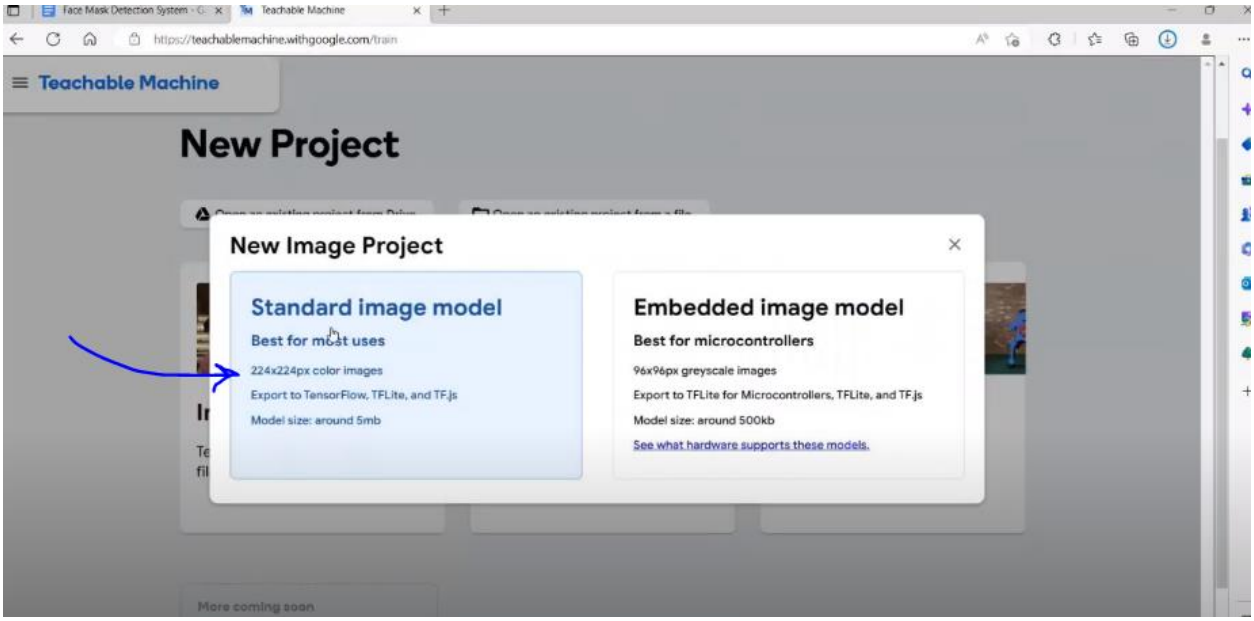
We will train a model online according to our image data and then work on it . We will use an online tool to train the model.

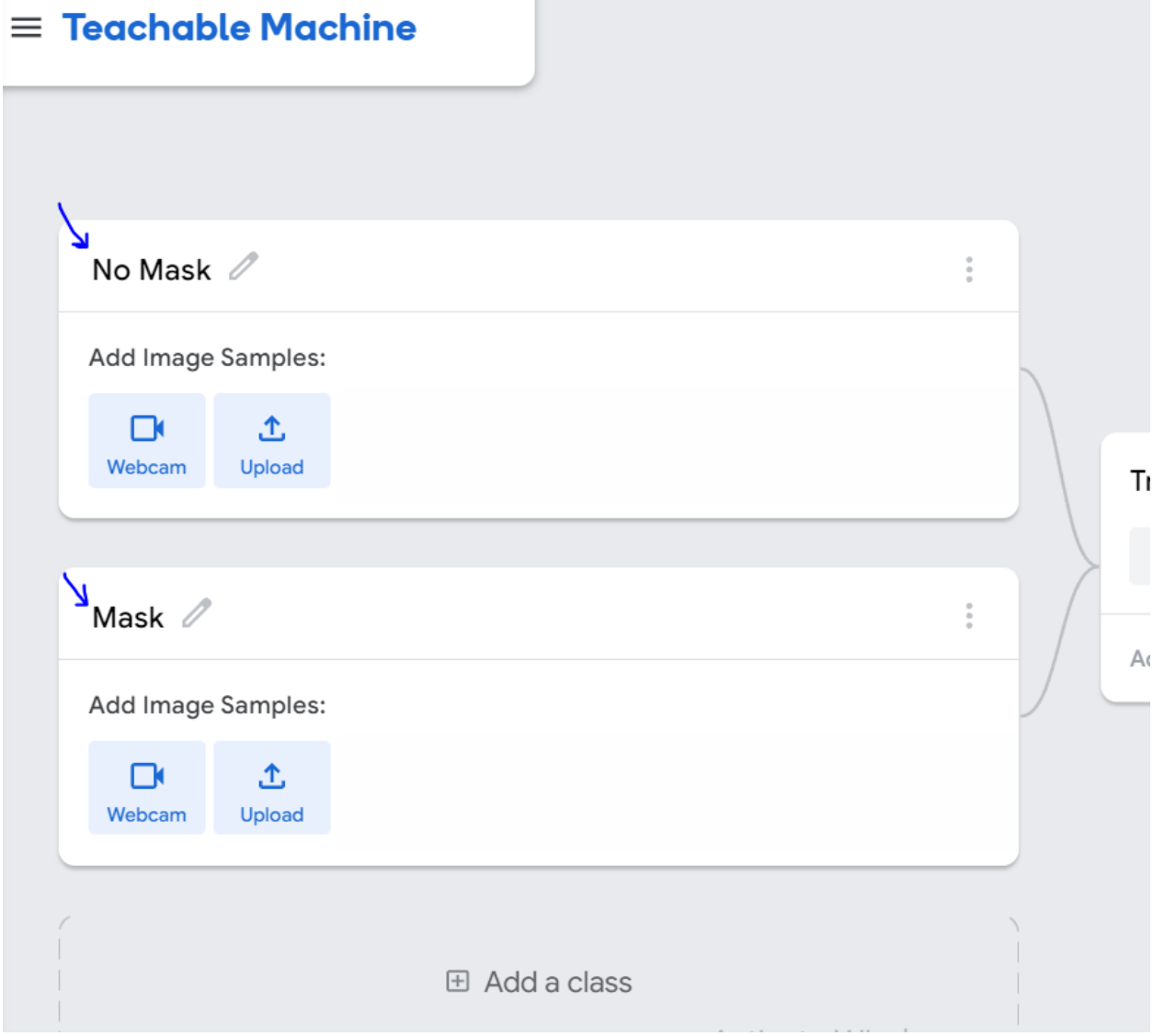
We will pass the data to Teachable Machine and do the training there (only the training part , nothing else).

This is the link to that site
https://teachablemachine.withgoogle.com/

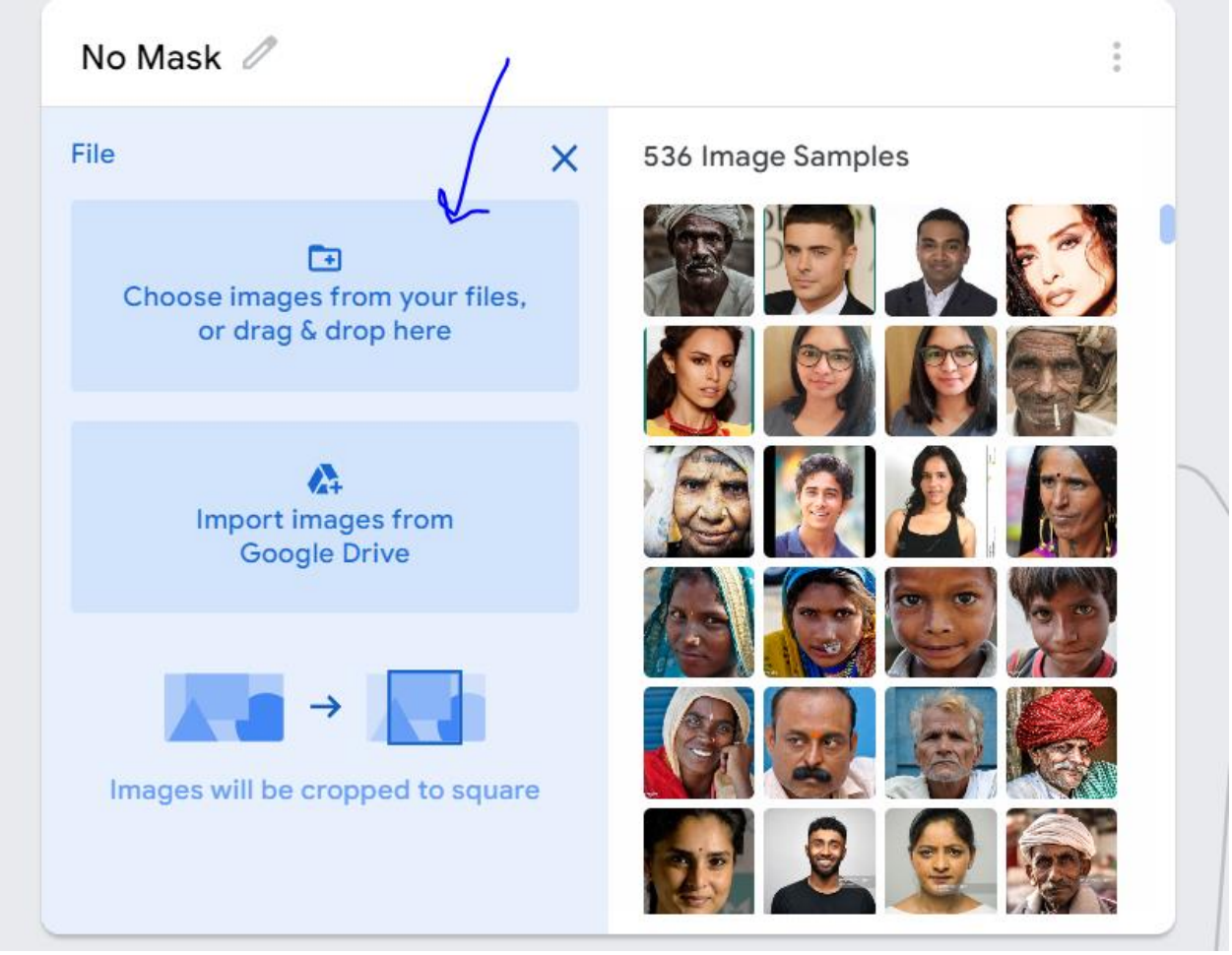
Go there and click on Get Started. And then click here .


Teachable Machine
New Project
Open an existing project from Drive.
Open an existing project from a file.
Image Project
Teach based on images, from files or your webcam.
Audio Project
Teach based on one-second-long sounds, from files or your microphone.
Pose Project
Teach based on images, from files or your webcam.

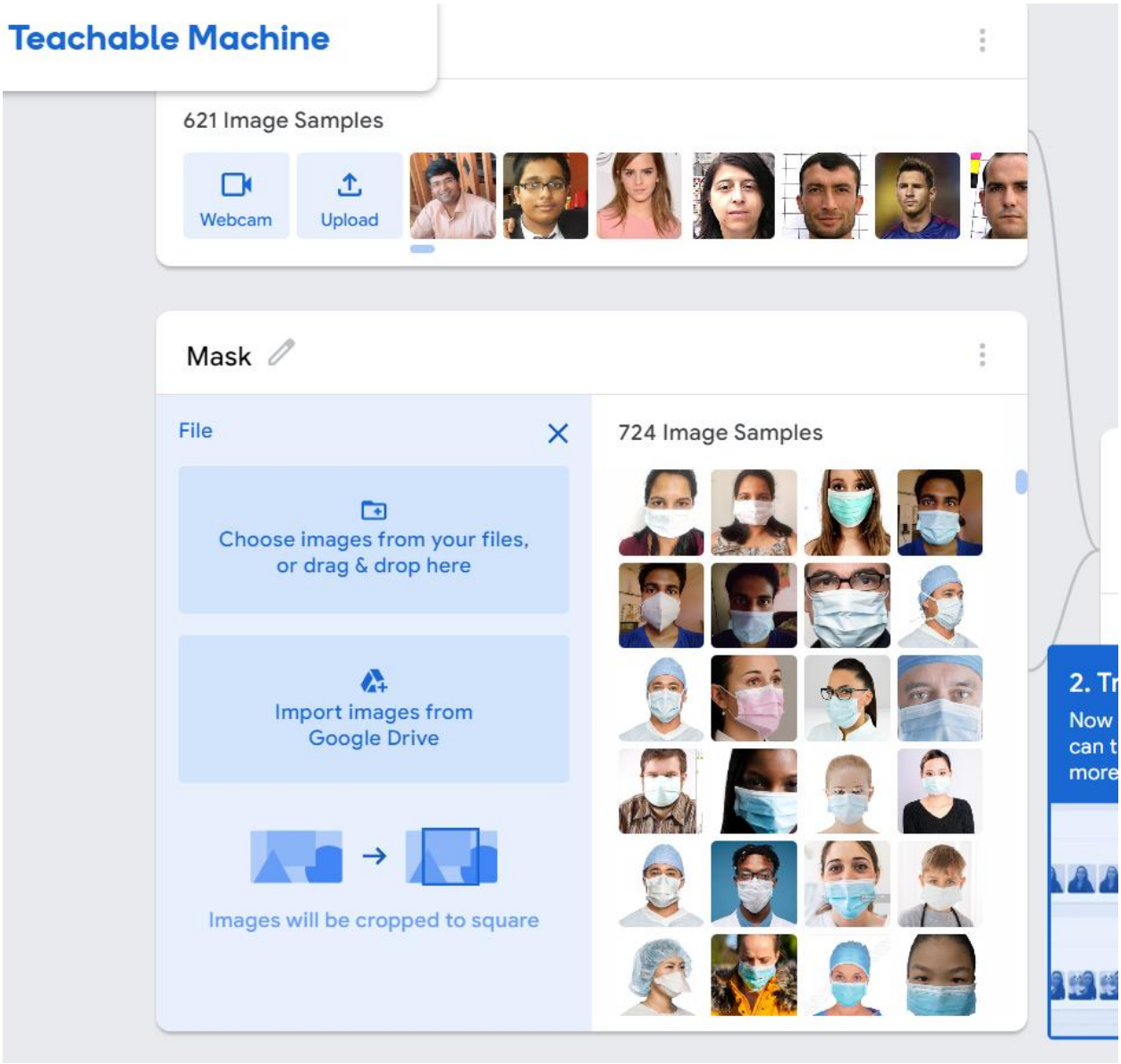

Face Mask Detection System
Teachable Machine
https://teachablemachine.withgoogle.com/train
Teachable Machine
New Project
New Image Project
Standard image model
Best for most uses
224x224px color images
Export to TensorFlow, TFLite, and TF.js
Model size: around 5mb
Embedded image model
Best for microcontrollers
96x96px greyscale images
Export to TFLite for Microcontrollers, TFLite, and TF.js
Model size: around 500kb
See what hardware supports these models.
More coming soon



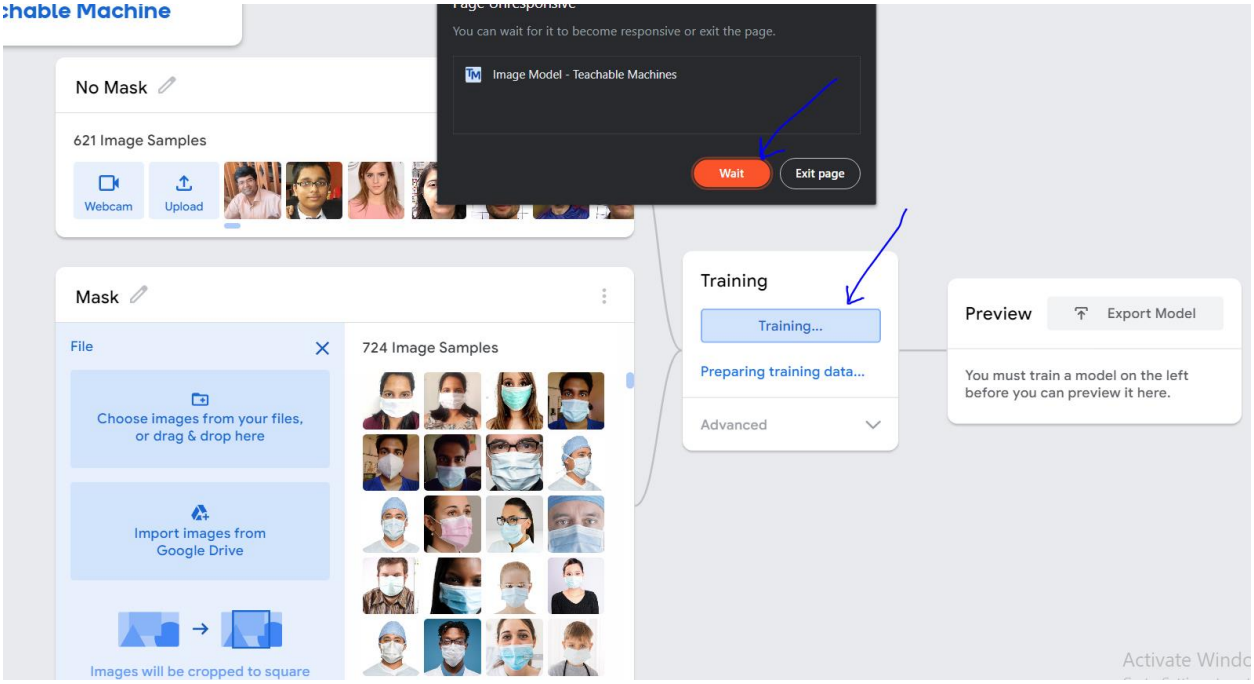
Change the class names as shown above from class 1 , class 2 to No Mask and Mask and then click on upload . Click on Choose images from your files or drag and drop here.

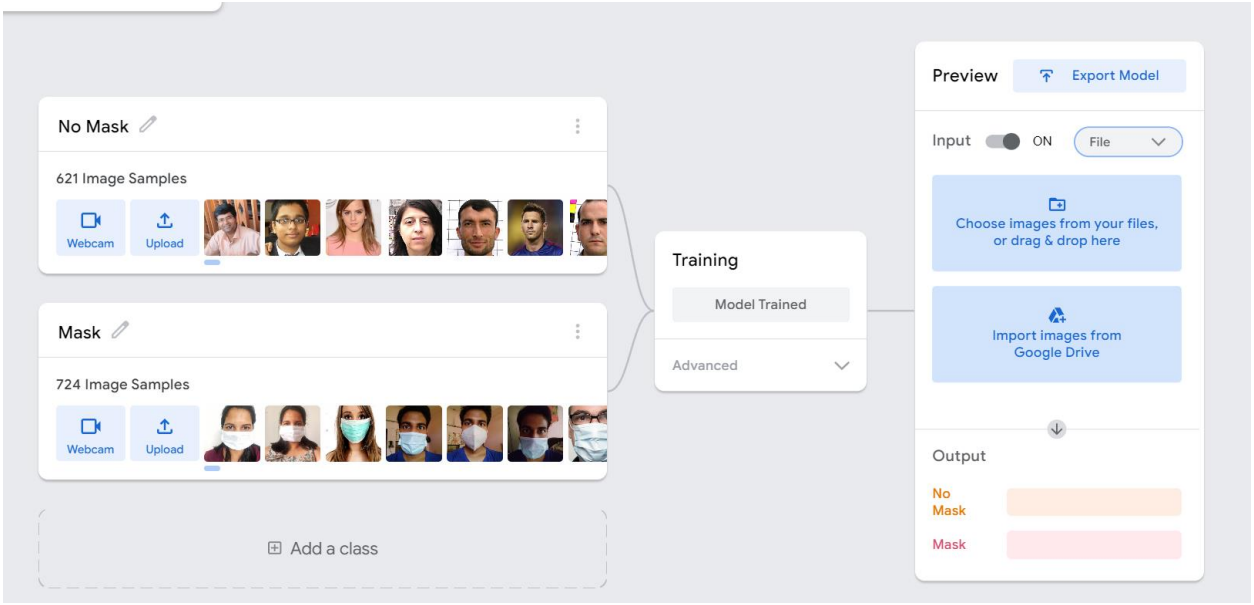


We uploaded here masked and unmasked images from training as well as some from the testing folder. (for No Mask we took all images from training without mask and first 427 images from test without mask . for Mask we took all with mask images from both training and testing folders.)

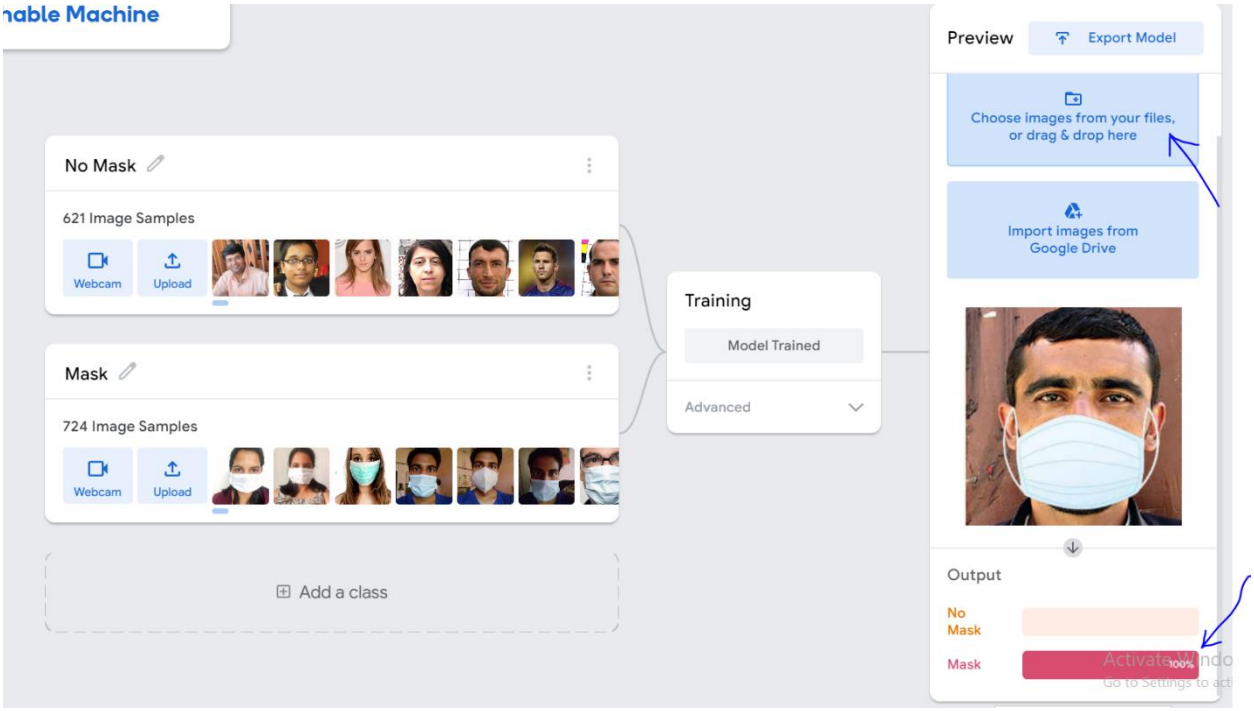


Click on Train . And browser might show wait or leave . Click on Wait everytime it asks so .

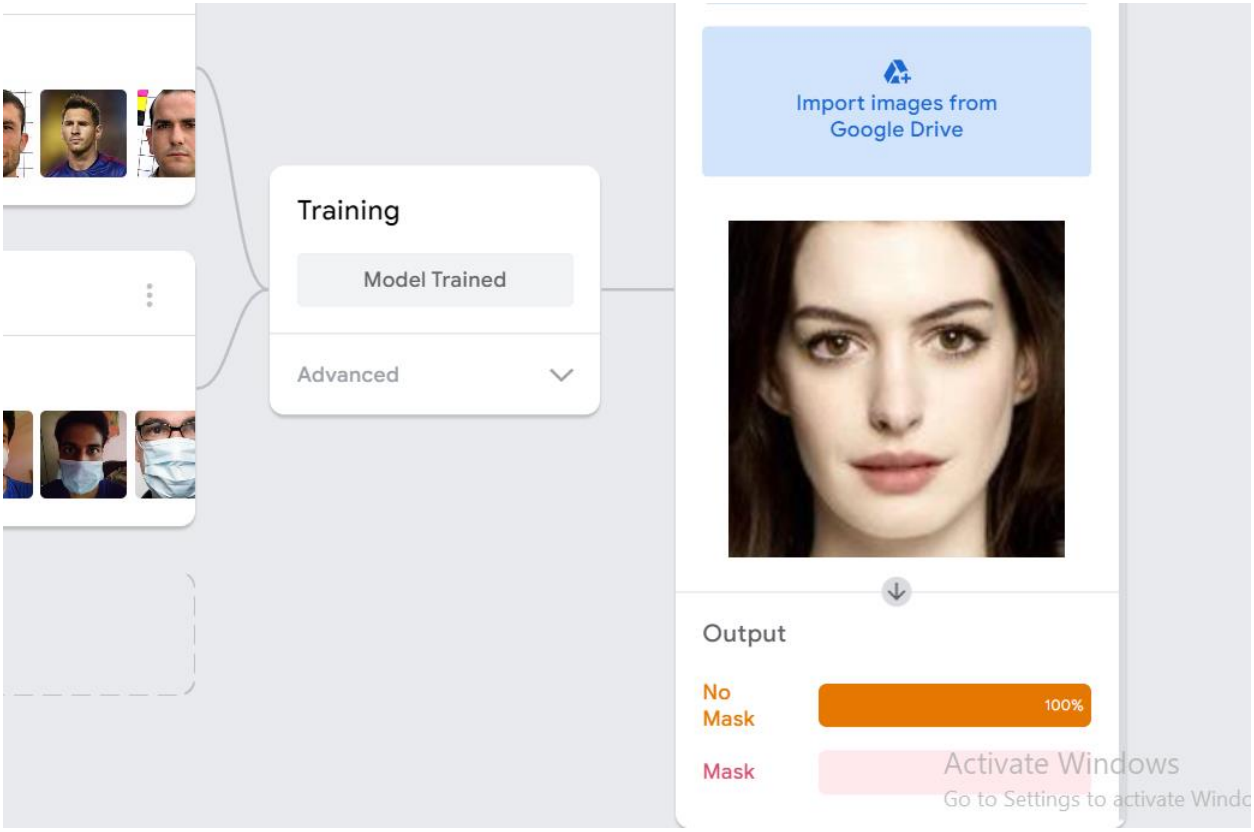




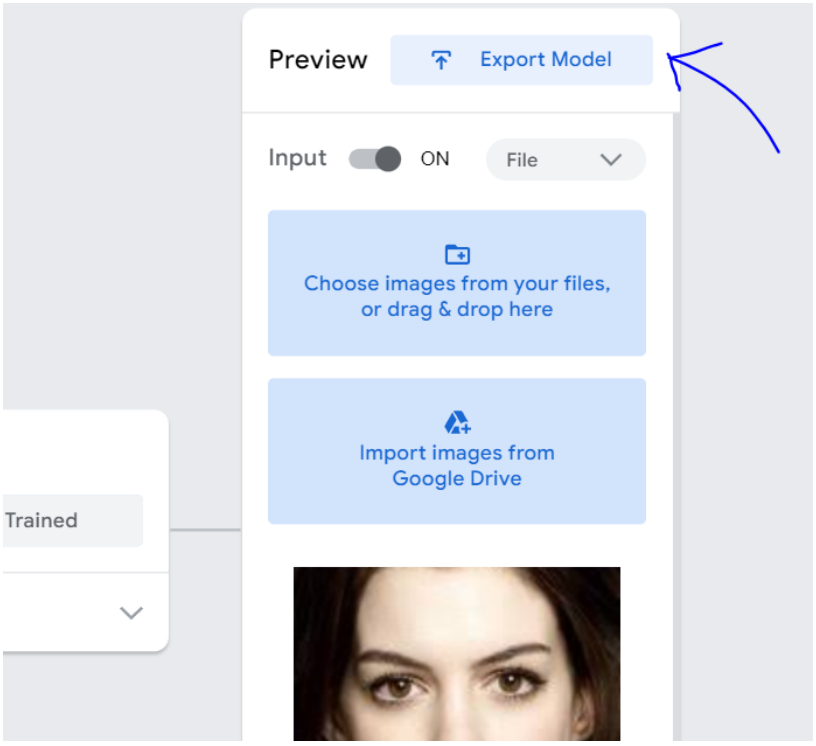
Our model got trained . The website asks for allow or block the webcam. We block it . And then choose File on the right and click on choose images and upload an image from the testing folder which was not given into training.

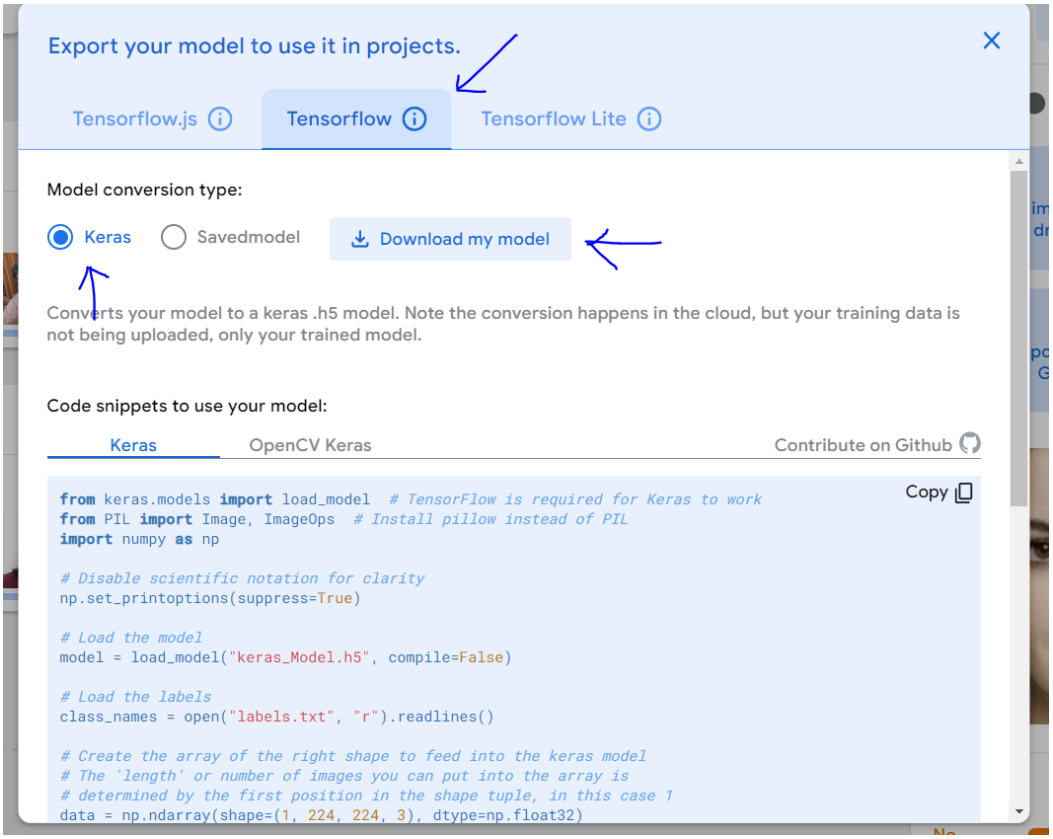


We see below that for the chosen image , the output shows 100% for Mask which is the correct prediction.

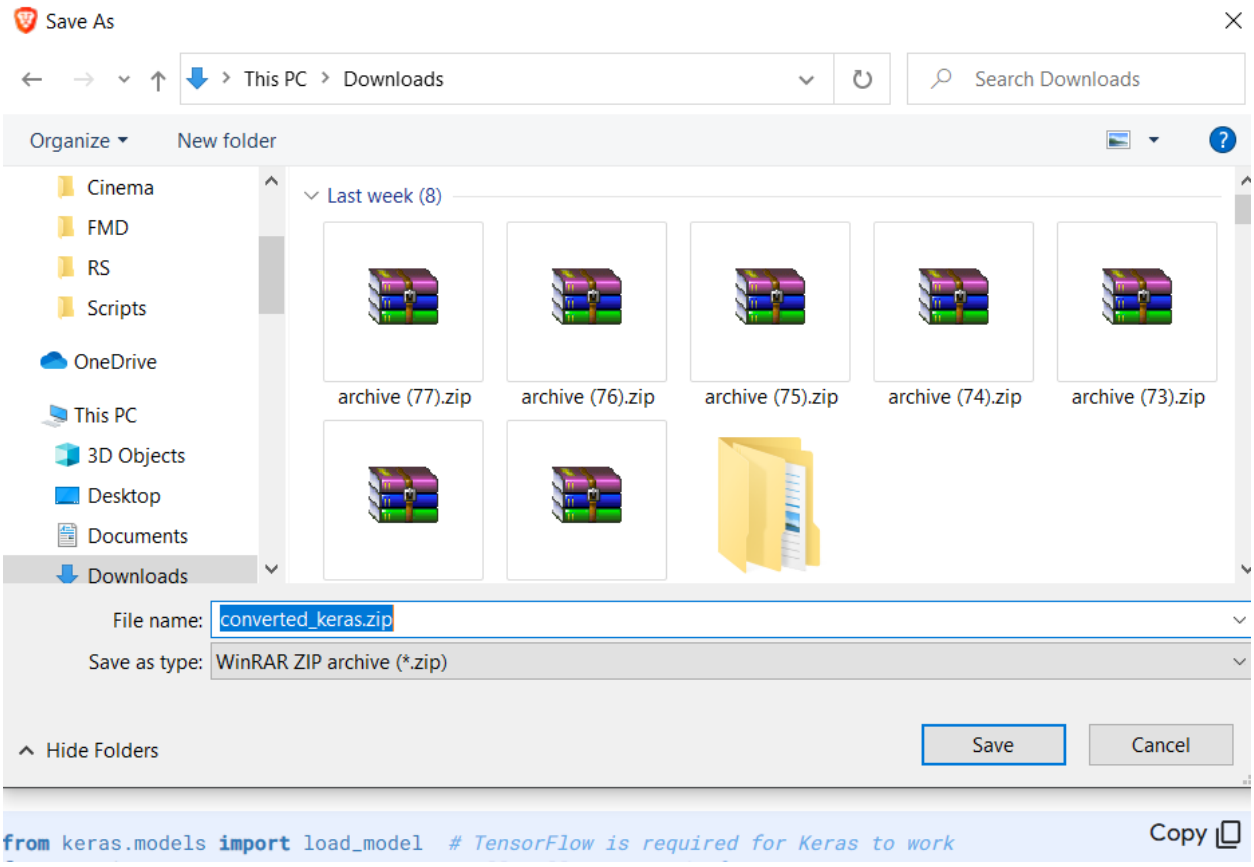


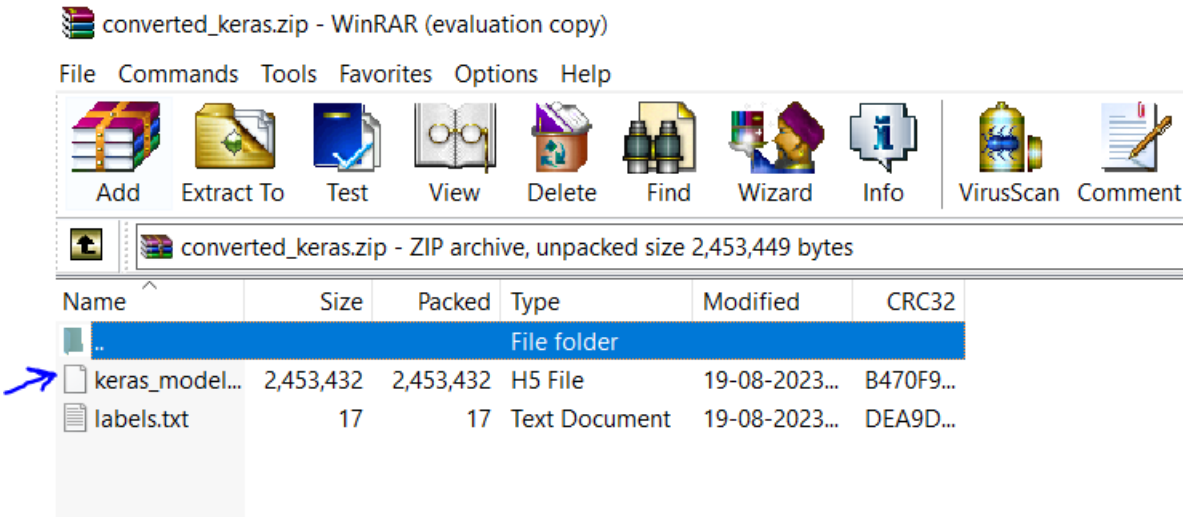
Now click on Export Model.





Now click on Tensorflow , select Keras and click on Download my model .





Paste this keras_model file in the scripts folder of the virtual environment. We rename the file as ‘mask.h5’ .

Now we will create a mask model and use this mask.h5 file(which is the model we created and downloaded to detect masked faces or not) to detect mask on faces.

We also need to install some new packages at this moment.

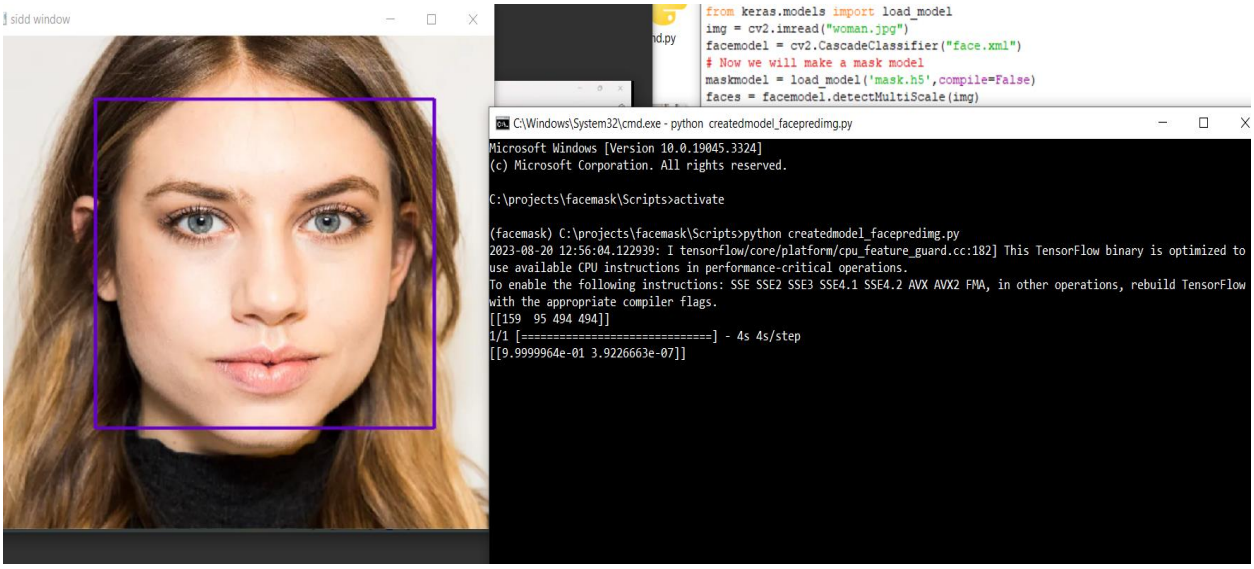
We need to do pip install pillow , pip install scipy , pip install tensorflow . These 3 packages are needed . tensorflow is needed for mask detection.

We get Tensorflow models sometimes as compiled and sometimes not compiled. If we get a compiled model , we can use that model in different places. But here our model is not compiled . So we set that to False.

createdmodel_facepredimg.py - C:\projects\facemask\Scripts\createdmodel_facepredimg.py (2.7.10)

File Edit Format Run Options Window Help

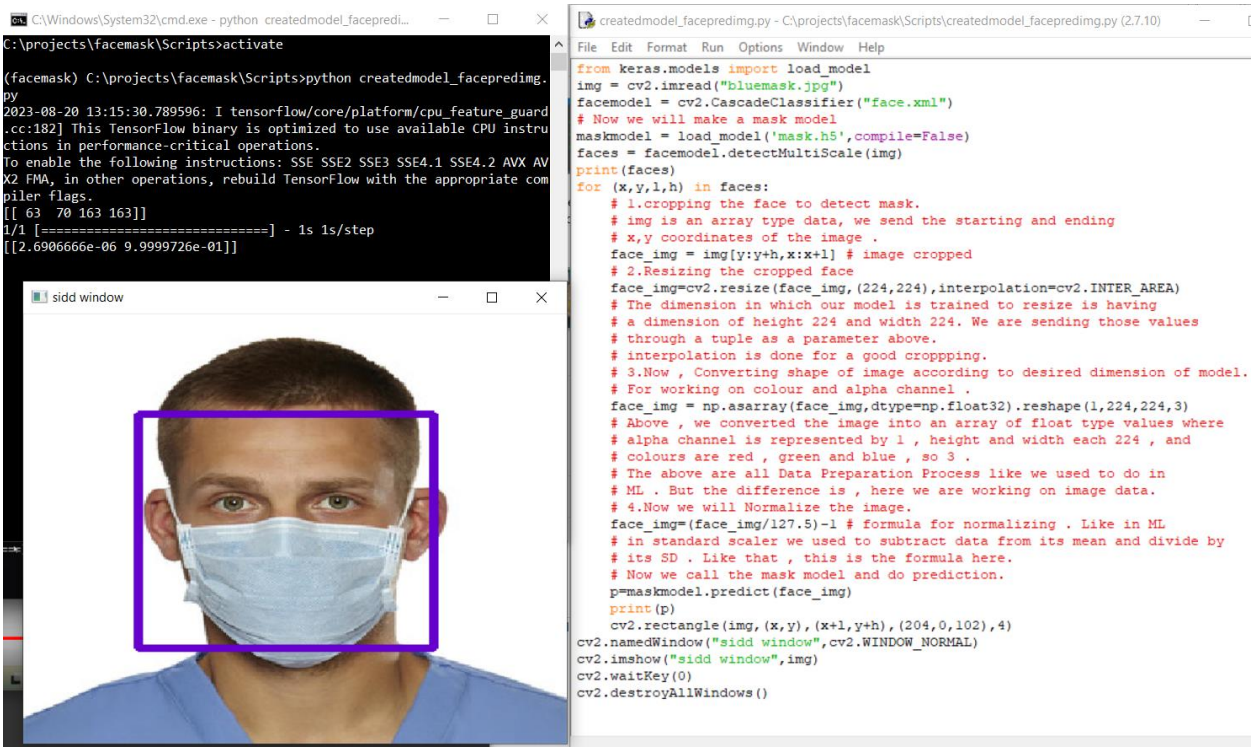
```
from keras.models import load_model
img = cv2.imread("woman.jpg")
facemodel = cv2.CascadeClassifier("face.xml")
# Now we will make a mask model
maskmodel = load_model('mask.h5',compile=False)
faces = facemodel.detectMultiScale(img)
print(faces)
for (x,y,l,h) in faces:
    # 1.cropping the face to detect mask.
    # img is an array type data, we send the starting and ending
    # x,y coordinates of the image .
    face_img = img[y:y+h,x:x+l] # image cropped
    # 2.Resizing the cropped face
    face_img=cv2.resize(face_img,(224,224),interpolation=cv2.INTER_AREA)
    # The dimension in which our model is trained to resize is having
    # a dimension of height 224 and width 224. We are sending those values
    # through a tuple as a parameter above.
    # interpolation is done for a good croppping.
    # 3.Now , Converting shape of image according to desired dimension of model.
    # For working on colour and alpha channel .
    face_img = np.asarray(face_img,dtype=np.float32).reshape(1,224,224,3)
    # Above , we converted the image into an array of float type values where
    # alpha channel is represented by 1 , height and width each 224 , and
    # colours are red , green and blue , so 3 .
    # The above are all Data Preparation Process like we used to do in
    # ML . But the difference is , here we are working on image data.
    # 4.Now we will Normalize the image.
    face_img=(face_img/127.5)-1 # formula for normalizing . Like in ML
    # in standard scaler we used to subtract data from its mean and divide by
    # its SD . Like that , this is the formula here.
    # Now we call the mask model and do prediction.
    p=maskmodel.predict(face_img)
    print(p)
    cv2.rectangle(img,(x,y),(x+l,y+h),(204,0,102),4)
cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
cv2.imshow("sidd window",img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```




```
(facemask) C:\projects\facemask\Scripts>python createdmodel
2023-08-20 12:56:04.122939: I tensorflow/core/platform/cpu_
use available CPU instructions in performance-critical oper
To enable the following instructions: SSE SSE2 SSE3 SSE4.1
with the appropriate compiler flags.
[[159  95 494 494]]
1/1 [==============================] - 4s 4s/step
[[9.9999964e-01 3.9226663e-07]]
```

Inside the bigger list , we have a smaller list of 2 values , the first value is for no mask prediction which is 99% here and the second value is for mask prediction which is nearly 0 % here . So the prediction is right .

For a masked image named ‘bluemask.jpg’ we again found a correct prediction.



The values nearly got reversed here. So it has predicted a mask on the face which is right .

We will now do a binary prediction based on mask or no mask as red rectangle for no mask and green rectangle for masked faces. Let us do this judgement based on the first value inside the smaller list which is within the bigger list.

So we will call the first value for no mask prediction .

p=maskmodel.predict(face_img)[0][0]

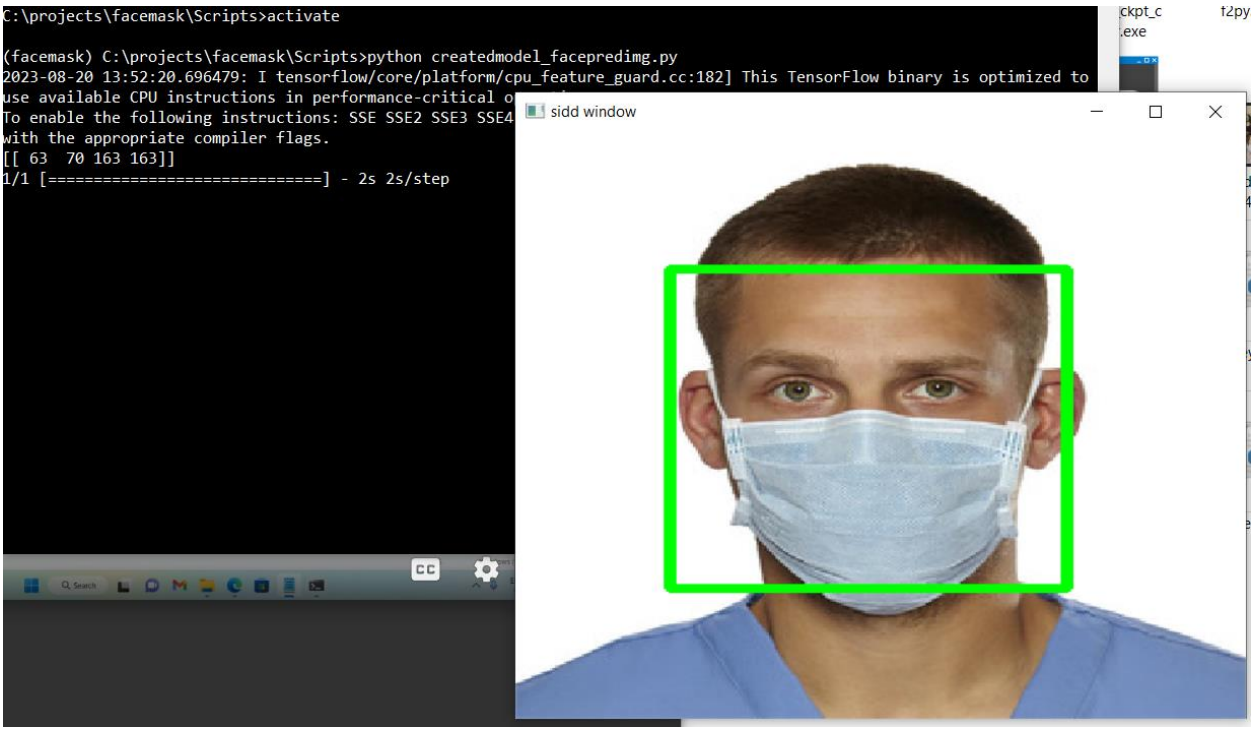
The first 0 calls the first list (the only list) inside the bigger list. The second 0 calls the first value (no mask prediction value) within that smaller list.

The code is given below :

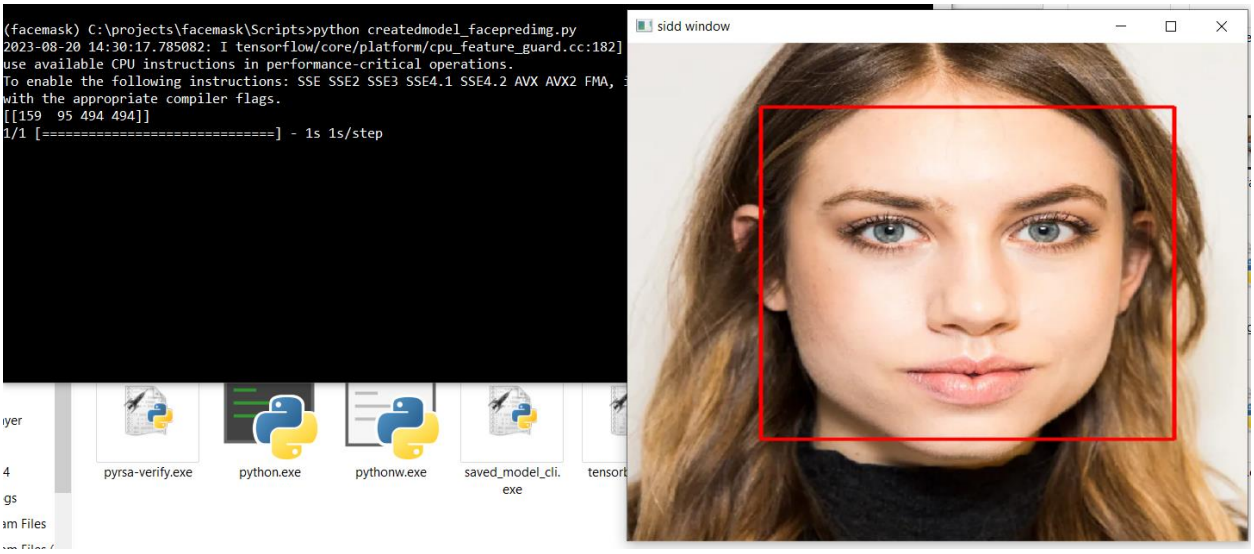
createdmodel_facepredimg.py - C:\projects\facemask\Scripts\createdmodel_facepredimg.py (2.7.10)

File Edit Format Run Options Window Help

```
import cv2 # cv2 is the open cv package
import numpy as np
from keras.models import load_model
img = cv2.imread("bluemask.jpg")
facemodel = cv2.CascadeClassifier("face.xml")
# Now we will make a mask model
maskmodel = load_model('mask.h5',compile=False)
faces = facemodel.detectMultiScale(img)
print(faces)
for (x,y,l,h) in faces:
    # 1.cropping the face to detect mask.
    # img is an array type data, we send the starting and ending
    # x,y coordinates of the image .
    face_img = img[y:y+h,x:x+l] # image cropped
    # 2.Resizing the cropped face
    face_img=cv2.resize(face_img,(224,224),interpolation=cv2.INTER_AREA)
    # The dimension in which our model is trained to resize is having
    # a dimension of height 224 and width 224. We are sending those values
    # through a tuple as a parameter above.
    # interpolation is done for a good croppping.
    # 3.Now , Converting shape of image according to desired dimension of model.
    # For working on colour and alpha channel .
    face_img = np.asarray(face_img,dtype=np.float32).reshape(1,224,224,3)
    # Above , we converted the image into an array of float type values where
    # alpha channel is represented by 1 , height and width each 224 , and
    # colours are red , green and blue , so 3 .
    # The above are all Data Preparation Process like we used to do in
    # ML . But the difference is , here we are working on image data.
    # 4.Now we will Normalize the image.
    face_img=(face_img/127.5)-1 # formula for normalizing . Like in ML
    # in standard scaler we used to subtract data from its mean and divide by
    # its SD . Like that , this is the formula here.
    # Now we call the mask model and do prediction.
    p=maskmodel.predict(face_img)[0][0]
    if(p>0.9): # if no mask prediction is greater than 0.9 then red rectangle.
        cv2.rectangle(img,(x,y),(x+l,y+h),(0,0,255),4) # red color rectangle
        # (b,g,r) set to (0,0,255).
    else:
        cv2.rectangle(img,(x,y),(x+l,y+h),(0,255,0),4) # green color rectangle
        # (b,g,r) set to (0,255,0).
cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
cv2.imshow("sidd window",img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Lets run it for a no mask face and see the result. In the same code we change the image file name to ‘woman.jpg’. The output is :



Hence our model is working correctly .

We will now do the same work on a video. The code is :

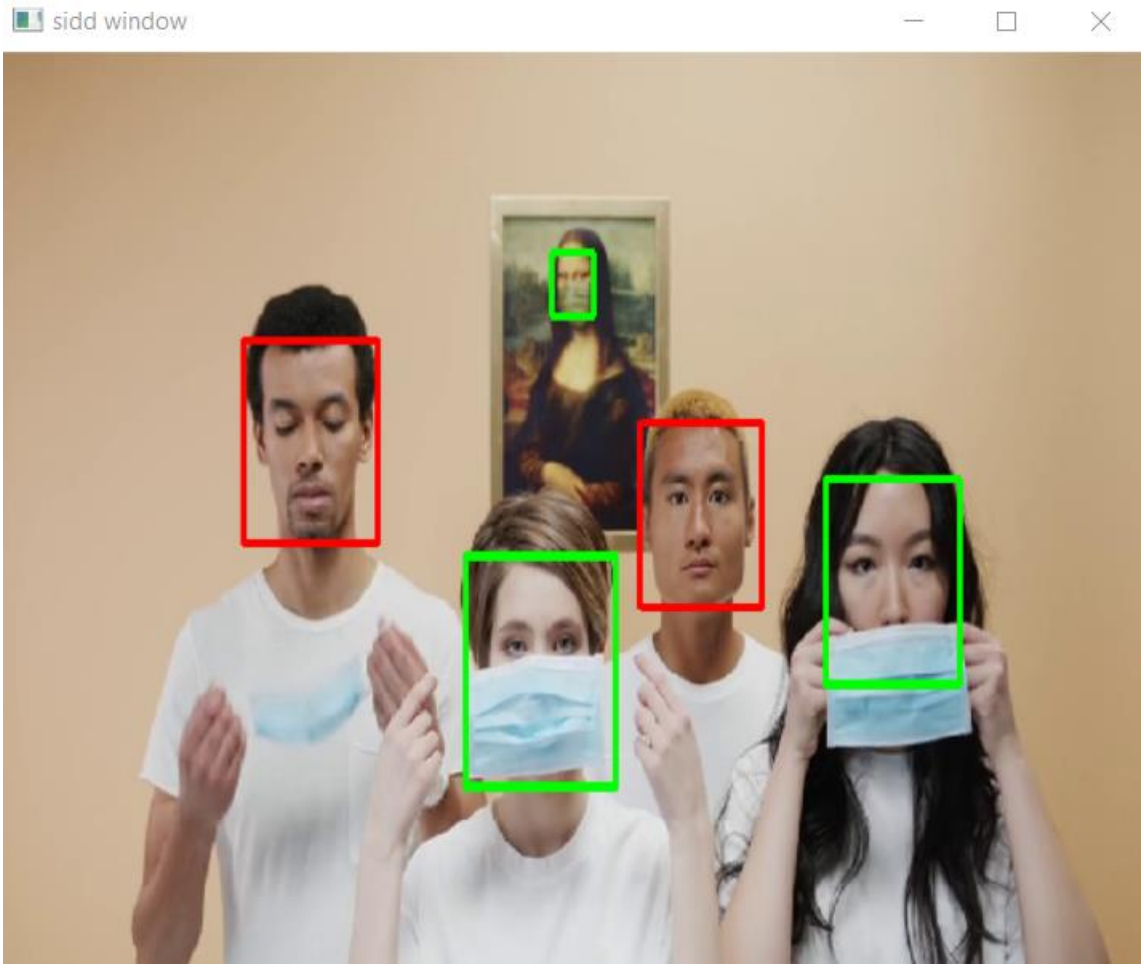
videorectanglepred.py - C:\projects\facemask\Scripts\videorectanglepred.py (2.7.10)

File Edit Format Run Options Window Help

```
import cv2 # cv2 is the open cv package
import numpy as np
from keras.models import load_model
# To read a video and detect mask on it and show it .
facemodel = cv2.CascadeClassifier("face.xml")
# Now we will make a mask model
maskmodel = load_model('mask.h5',compile=False)
vid=cv2.VideoCapture("groupvidface.mp4")
#reading video means reading multiple images from that video
#So we will run a loop.
while(vid.isOpened()):
    flag,frame = vid.read()
    if(flag):# means if flag == True
        faces=facemodel.detectMultiScale(frame) # this command detects the faces within every frame of the video .
        for (x,y,l,h) in faces:
            # 1.cropping the face to detect mask.
            # img is an array type data, we send the starting and ending
            # x,y coordinates of the image .
            face_img = frame[y:y+h,x:x+l] # image cropped
            # 2.Resizing the cropped face
            face_img=cv2.resize(face_img,(224,224),interpolation=cv2.INTER_AREA)
            # The dimension in which our model is trained to resize is having
            # a dimension of height 224 and width 224. We are sending those values
            # through a tuple as a parameter above.
            # interpolation is done for a good croppping.
            # 3.Now , Converting shape of image according to desired dimension of model.
            # For working on colour and alpha channel .
            face_img = np.asarray(face_img,dtype=np.float32).reshape(1,224,224,3)
            # Above , we converted the image into an array of float type values where
            # alpha channel is represented by 1 , height and width each 224 , and
            # colours are red , green and blue , so 3 .
            # The above are all Data Preparation Process like we used to do in
            # ML . But the difference is , here we are working on image data.
            # 4.Now we will Normalize the image.
            face_img=(face_img/127.5)-1 # formula for normalizing . Like in ML
            # in standard scaler we used to subtract data from its mean and divide by
            # its SD . Like that , this is the formula here.
            # Now we call the mask model and do prediction.
            p=maskmodel.predict(face_img)[0][0]
            if(p>0.9): # if no mask prediction is greater than 0.9 then red rectangle.
                cv2.rectangle(frame,(x,y),(x+l,y+h),(0,0,255),4) # red color rectangle
                # (b,g,r) set to (0,0,255).
            else:
                cv2.rectangle(frame,(x,y),(x+l,y+h),(0,255,0),4) # green color rectangle
                # (b,g,r) set to (0,255,0).
        cv2.namedWindow("sidd window",cv2.WINDOW_NORMAL)
        cv2.imshow("sidd window",frame)
        k = cv2.waitKey(10) # this value is a measure of how long to stop before moving to the nexr frame
        if(k == ord('x')):
            break
    else:
        break
cv2.destroyAllWindows()
```

```
(facemask) C:\projects\facemask\Scripts>python videorectanglepred.py
```

The Output is shown in the next page .



The output is showing green retangles for masked faces and red rectangles for unmasked faces.

For doing the same on a real time video through an IP camera , we need to replace this command line below

vid = cv2.VideoCapture(“groupvidface.mp4”)

with

vid=cv2.VideoCapture("http://192.168.0.108:8080/video")

The URL is the first ipv4 address in the IP webcam app after starting its server.

The rest of the code will be same as the code above.

Frontend

```
C:\projects\facemask\Scripts>activate

(facemask) C:\projects\facemask\Scripts>pip install streamlit
```

The code for the entire front end design is :

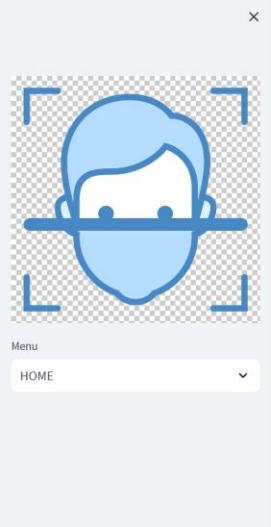
```
import streamlit as st

import cv2

import numpy as np

from keras.models import load_model

st.title("FACE MASK DETECTION SYSTEM")

st.sidebar.image("https://cdn.imgbin.com/1/6/15/imgbin-facial-recognition-system-computer-icons-face-detection-iris-recognition-scanner-y78VcgcD5aV3BTReGGiizifnQ.jpg")

choice=st.sidebar.selectbox("Menu",("HOME","URL","CAMERA"))

st.header(choice) # st.header is used for printing the choice

if(choice=="HOME"):

   st.image("https://5.imimg.com/data5/PI/FD/NK/SELLER-5866466/images-500x500.jpg")

elif(choice=="URL"):
```

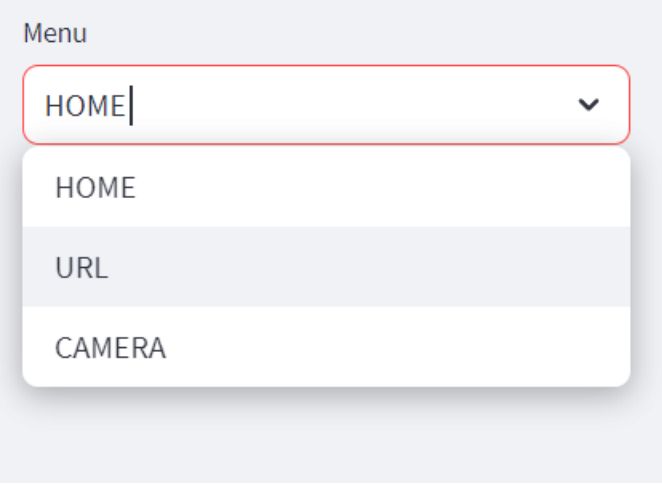
```
    url=st.text_input("Enter your URL")
    btn=st.button("Start Detection")
    window=st.empty() # for
    if btn:
        i=1
        btn2 = st.button("Stop Detection")
        if btn2:
            st.experimental_rerun() # for rereun.
        facemodel=cv2.CascadeClassifier("face.xml")
        maskmodel=load_model("mask.h5")
        vid=cv2.VideoCapture(url)
        while(vid.isOpened()):
            flag,frame=vid.read()
            if flag:
                faces=facemodel.detectMultiScale(frame)
                for (x,y,l,w) in faces: # w or h is same.
                    face_img=frame[y:y+w,x:x+l] # cropping

face_img=cv2.resize(face_img,(224,224),interpolation=cv2.INTER_AREA)

face_img=np.asarray(face_img,dtype=np.float32).reshape(1,224,224,3)
                    face_img=(face_img/127.5)-1
```

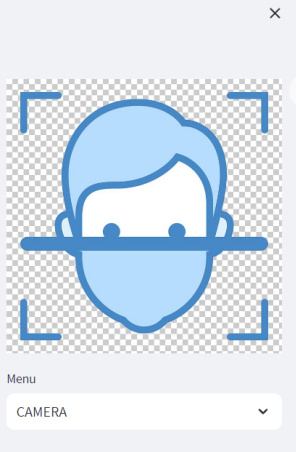
```
            p=maskmodel.predict(face_img)[0][0]
            if(p>0.9):
               path="nomask/"+str(i)+".jpg" # saving without mask faces in nomask folder of our environment as 1.jpg , 2.jpg etc
               cv2.imwrite(path,frame[y:y+w,x:x+l]) # saving the cropped unmasked faces.
               i=i+1
               cv2.rectangle(frame,(x,y),(x+l,y+w),(0,0,255),4)
            else:
               cv2.rectangle(frame,(x,y),(x+l,y+w),(0,255,0),4)
         window.image(frame,channels='BGR')
elif(choice=="CAMERA"):
  cam = st.selectbox("Choose Camera",("None","Primary","Secondary"))
  btn=st.button("Start Detection")
  window=st.empty() # for
  if btn:
    i=1
    btn2 = st.button("Stop Detection")
    if btn2:
      st.experimental_rerun() # for rereun.
    facemodel=cv2.CascadeClassifier("face.xml")
    maskmodel=load_model("mask.h5")
```

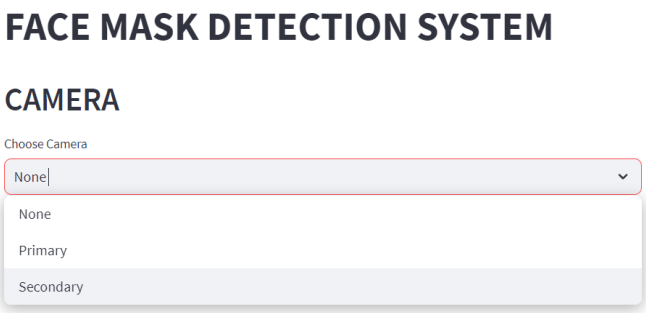
```
    if cam=="Primary":
       cam=0 # for primray cam i.e. front cam
    else:
       cam=1
    vid=cv2.VideoCapture(cam)
    while(vid.isOpened()):
       flag,frame=vid.read()
       if flag:
          faces=facemodel.detectMultiScale(frame)
          for (x,y,l,w) in faces: # w or h is same.
             face_img=frame[y:y+w,x:x+l] # cropping

face_img=cv2.resize(face_img,(224,224),interpolation=cv2.INTER_AREA)

face_img=np.asarray(face_img,dtype=np.float32).reshape(1,224,224,3)
             face_img=(face_img/127.5)-1
             p=maskmodel.predict(face_img)[0][0]
             if(p>0.9):
                path="nomask/"+str(i)+".jpg" # saving without mask faces in
nomask folder of our environment as 1.jpg , 2.jpg etc
                cv2.imwrite(path,frame[y:y+w,x:x+l]) # saving the cropped
unmasked faces.
                i=i+1
```

```
            cv2.rectangle(frame,(x,y),(x+l,y+w),(0,0,255),4)
        else:
            cv2.rectangle(frame,(x,y),(x+l,y+w),(0,255,0),4)
    window.image(frame,channels='BGR')
```









We have access to footage of primary and secondary camera.

Here is a glimpse of our secondary camera real time footage.

As we can see it is an unmasked face . Hence it is detected within a red rectangle. This image also got saved in our nomask folder of our virtual environment so that we can later take action on it . This is a good usage of this system.

If there is a camera connected to a USB then that acts as a secondary camera . In that case also we can access that footage if we click on the 'Secondary' option under our Choose Camera selectbox.

Same happens for URL footage also. On giving the appropriate ipv4 address on the URL box we get access to its footage. On clicking 'Stop Detection' , the program stops the entire detection process.