

```

import csv
import os
from random import seed
from random import random
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
from PIL import Image
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, MaxPooling2D, Flatten, Conv2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.activations import softmax, sigmoid
from tensorflow.python.keras.optimizers import SGD
import tensorflow as tf

from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode

def take_photo(filename='photo.jpg', quality=0.8):
    js = Javascript('''
        async function takePhoto(quality) {
            const div = document.createElement('div');
            const capture = document.createElement('button');
            capture.textContent = 'Capture';
            div.appendChild(capture);

            const video = document.createElement('video');
            video.style.display = 'block';
            const stream = await navigator.mediaDevices.getUserMedia({video: true});

            document.body.appendChild(div);
            div.appendChild(video);
            video.srcObject = stream;
            await video.play();

            // Resize the output to fit the video element.
            google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

            // Wait for Capture to be clicked.
            await new Promise((resolve) => capture.onclick = resolve);

            const canvas = document.createElement('canvas');
            canvas.width = video.videoWidth;
            canvas.height = video.videoHeight;
            canvas.getContext('2d').drawImage(video, 0, 0);
            stream.getVideoTracks()[0].stop();
            div.remove();
            return canvas.toDataURL('image/jpeg', quality);
        }
    ''')
    display(js)
    data = eval_js('takePhoto({})'.format(quality))
    binary = b64decode(data.split(',')[1])

```

```

with open(filename, 'wb') as f:
    f.write(binary)
return filename

from IPython.display import Image
# try:
#     filename = take_photo()
#     print('Saved to {}'.format(filename))
#     # Show the image which was just taken.
#     display(Image(filename))
# except Exception as err:
#     # Errors will be thrown if the user does not have a webcam or if they do not
#     # grant the page permission to access it.
#     print(str(err))

import cv2
from google.colab.patches import cv2_imshow
import numpy as np
labels_dict={0:'without_mask',1:'with_mask'}
color_dict={0:(0,0,255),1:(0,255,0)}
predict_path = "/content/drive/My Drive/prediction_images/all_classes/"
model = tf.keras.models.load_model("/content/drive/My Drive/saved_model/model_keras_150.h5")
size = 4
classifier = cv2.CascadeClassifier('/content/drive/My Drive/haarcascade_frontalface_default.xml')

while True:
    filename = take_photo()
    im = cv2.imread(filename)
    im=cv2.flip(im,1,1) #Flip to act as a mirror

    # Resize the image to speed up detection
    mini = cv2.resize(im, (im.shape[1] // size, im.shape[0] // size))

    # detect MultiScale / faces
    faces = classifier.detectMultiScale(mini)

    # Draw rectangles around each face
    for f in faces:
        (x, y, w, h) = [v * size for v in f] #Scale the shapesize backup
        #Save just the rectangle faces in SubRecFaces
        face_img = im[y:y+h, x:x+w]
        resized=cv2.resize(face_img,(150,150))
        normalized=resized/255.0
        reshaped=np.reshape(normalized,(1,150,150,3))
        reshaped = np.vstack([reshaped])
        result=model.predict(reshaped)
        #print(result)

        label=np.argmax(result,axis=1)[0]

        cv2.rectangle(im,(x,y),(x+w,y+h),color_dict[label],2)
        cv2.rectangle(im,(x,y-40),(x+w,y),color_dict[label],-1)
        cv2.putText(im, labels_dict[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255))

```

```
# Show the image
cv2.imshow(im)
key = cv2.waitKey(10)
# if Esc key is press then break out of the loop
if key == 27: #The Esc key
    break
# Stop video
webcam.release()

# Close all started windows
cv2.destroyAllWindows()
```



