CODING™ DOJO

COMPUTER VISION

Avengers

Introduction

Early experiments in computer vision took place in the 1950s, using some of the first neural networks to detect the edges of an object and to sort simple objects into categories like circles and squares. In the 1970s, the first commercial use of computer vision interpreted typed or handwritten text using optical character recognition. This advancement was used to interpret written text for the blind.

OUR DATASET

Our dataset is Face Mask Detection Dataset

The dataset contains about 1006 equally distributed images of 2 distinct types.

-Mask

-No mask

STEPS

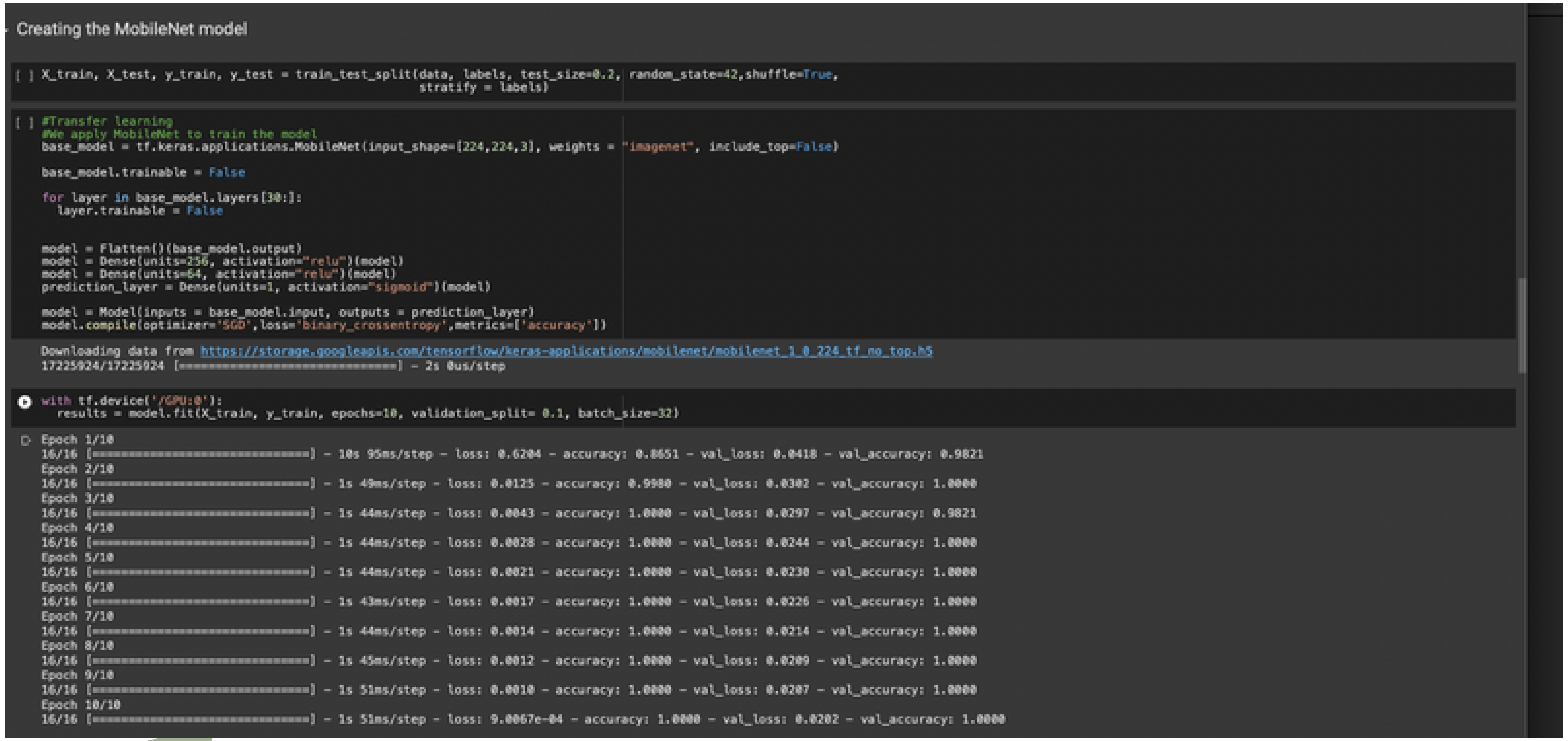
✦ step 1

Creating a MobileNet model to detect the face masks

✦ step 2

Using our saved model and build the detection system, via video or real time camera

Creating a MobileNet model to detect the face masks



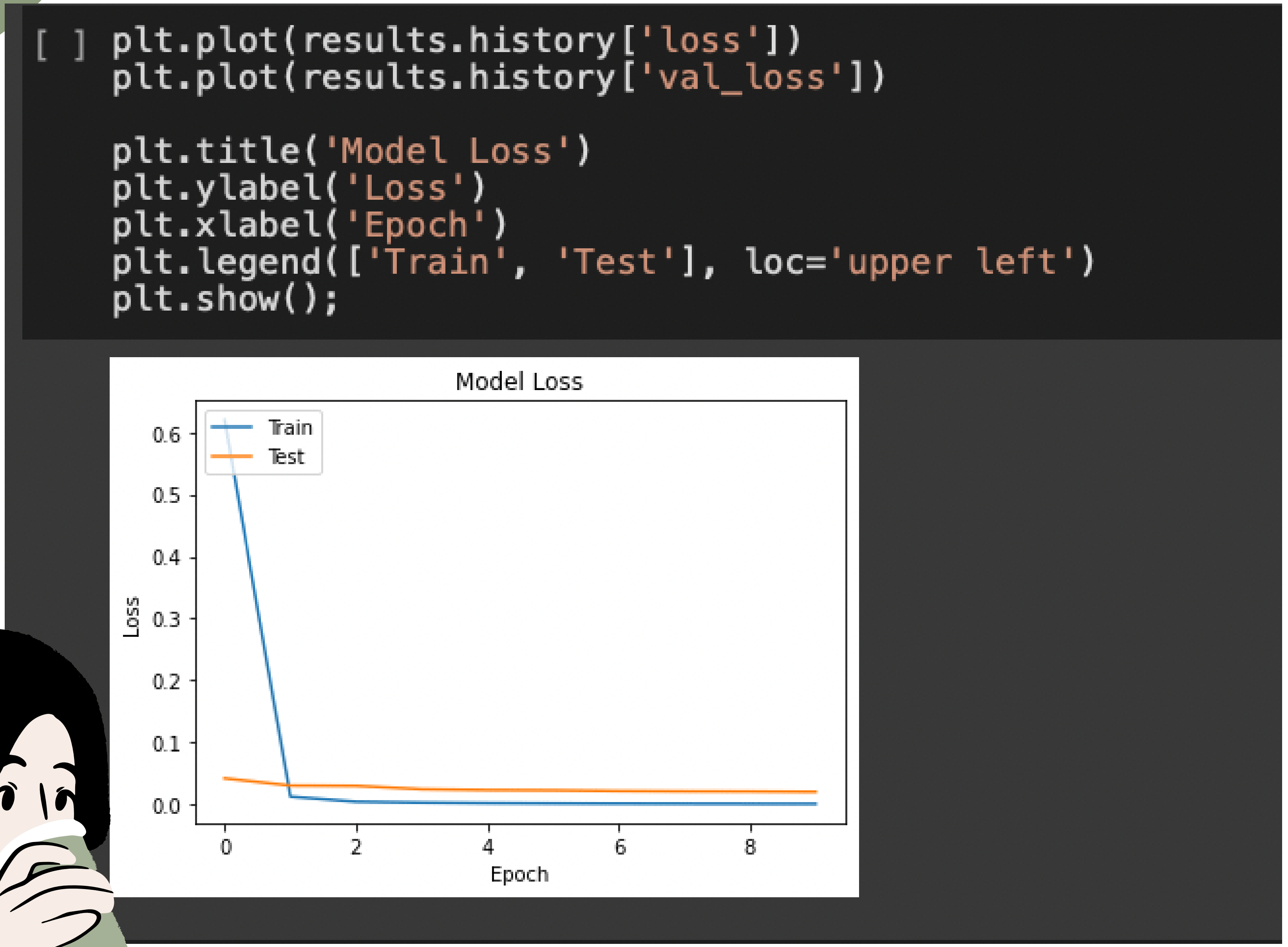
Creating the MobileNet model

```
[ ] X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42,shuffle=True,
                                                         stratify = labels)
```

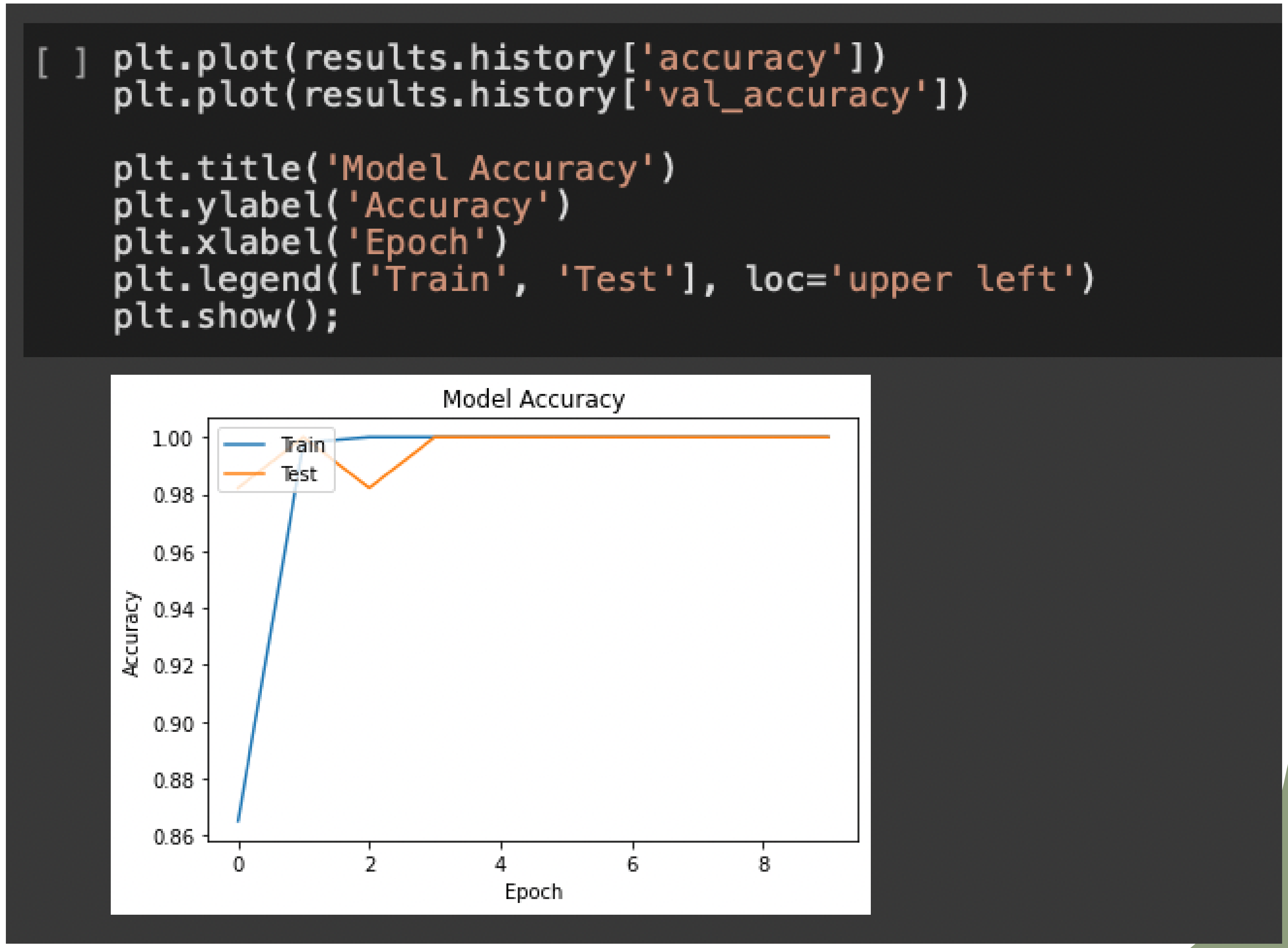
```
[ ] #Transfer learning
    #We apply MobileNet to train the model
    base_model = tf.keras.applications.MobileNet(input_shape=[224,224,3], weights = "imagenet", include_top=False)

    base_model.trainable = False

    for layer in base_model.layers[30:]:
      layer.trainable = False


    model = Flatten()(base_model.output)
    model = Dense(units=256, activation="relu")(model)
    model = Dense(units=64, activation="relu")(model)
    prediction_layer = Dense(units=1, activation="sigmoid")(model)

    model = Model(inputs = base_model.input, outputs = prediction_layer)
    model.compile(optimizer='SGD',loss='binary_crossentropy',metrics=['accuracy'])

    Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet/mobilenet_1_0_224_tf_no_top.h5
    17225924/17225924 [==============================] - 2s 0us/step
```

```
    with tf.device('/GPU:0'):
      results = model.fit(X_train, y_train, epochs=10, validation_split= 0.1, batch_size=32)

    Epoch 1/10
    16/16 [==============================] - 10s 95ms/step - loss: 0.6204 - accuracy: 0.8651 - val_loss: 0.0418 - val_accuracy: 0.9821
    Epoch 2/10
    16/16 [==============================] - 1s 49ms/step - loss: 0.0125 - accuracy: 0.9980 - val_loss: 0.0302 - val_accuracy: 1.0000
    Epoch 3/10
    16/16 [==============================] - 1s 44ms/step - loss: 0.0043 - accuracy: 1.0000 - val_loss: 0.0297 - val_accuracy: 0.9821
    Epoch 4/10
    16/16 [==============================] - 1s 44ms/step - loss: 0.0028 - accuracy: 1.0000 - val_loss: 0.0244 - val_accuracy: 1.0000
    Epoch 5/10
    16/16 [==============================] - 1s 44ms/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.0230 - val_accuracy: 1.0000
    Epoch 6/10
    16/16 [==============================] - 1s 43ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0226 - val_accuracy: 1.0000
    Epoch 7/10
    16/16 [==============================] - 1s 44ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0214 - val_accuracy: 1.0000
    Epoch 8/10
    16/16 [==============================] - 1s 45ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0209 - val_accuracy: 1.0000
    Epoch 9/10
    16/16 [==============================] - 1s 51ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0207 - val_accuracy: 1.0000
    Epoch 10/10
    16/16 [==============================] - 1s 51ms/step - loss: 9.0067e-04 - accuracy: 1.0000 - val_loss: 0.0202 - val_accuracy: 1.0000
```

Creating a MobileNet model to detect the face masks

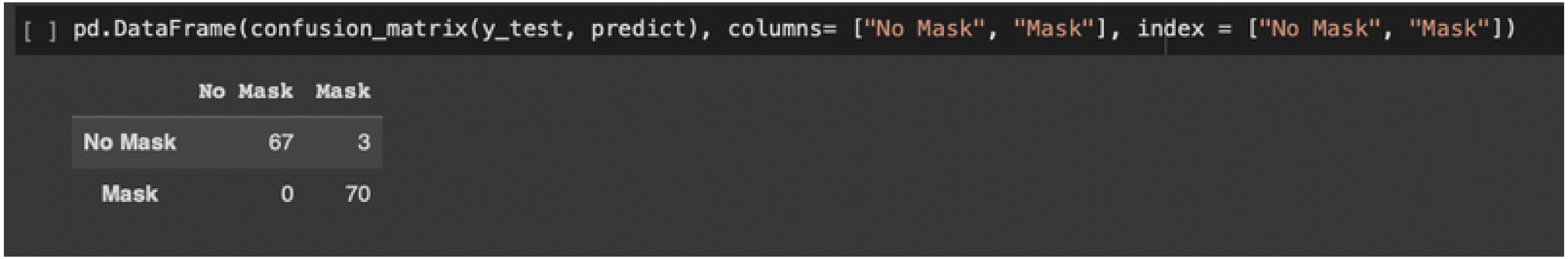


```
[ ] plt.plot(results.history['loss'])
    plt.plot(results.history['val_loss'])

    plt.title('Model Loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show();
```



```
[ ] plt.plot(results.history['accuracy'])
    plt.plot(results.history['val_accuracy'])

    plt.title('Model Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show();
```

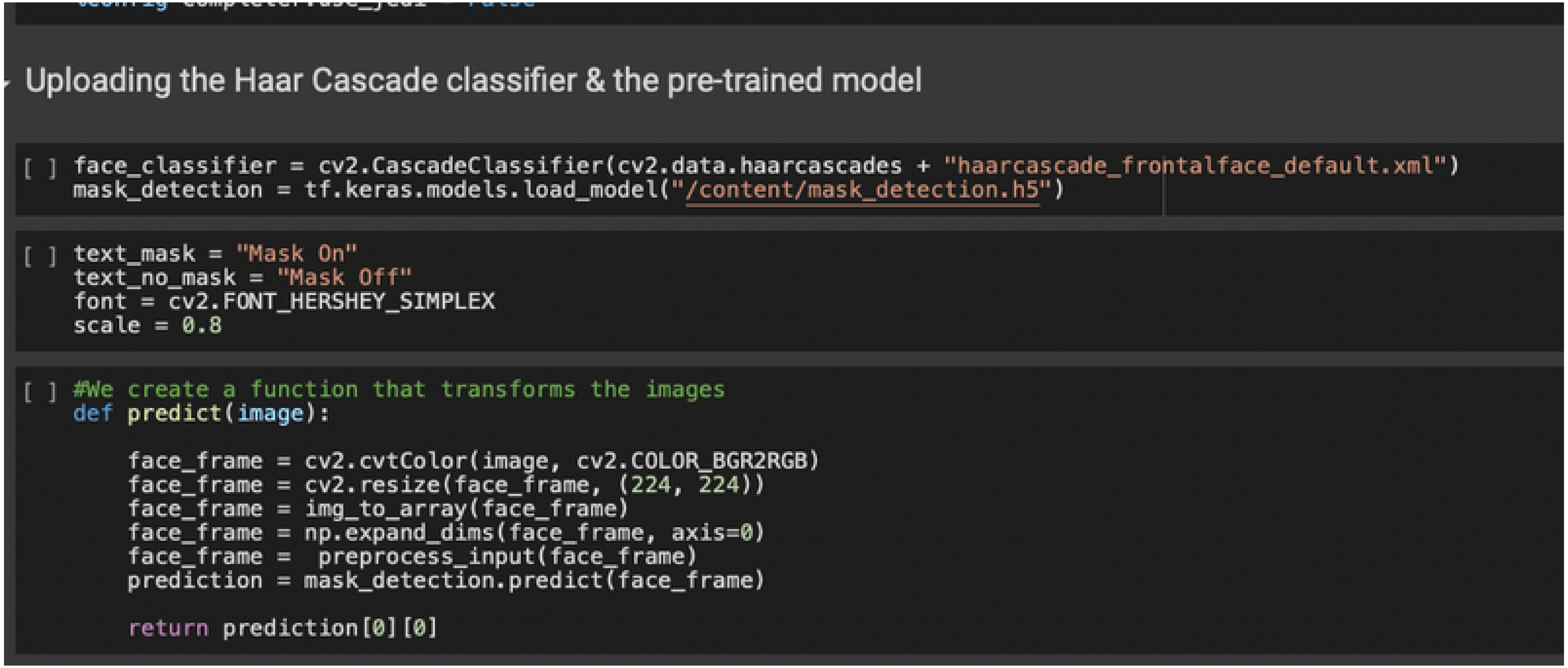
Creating a MobileNet model to detect the face masks



```
[ ] pd.DataFrame(confusion_matrix(y_test, predict), columns= ["No Mask", "Mask"], index = ["No Mask", "Mask"])
```

```
         No Mask  Mask
No Mask       67     3
   Mask        0    70
```

Building the detection system



```
%config Completer.use_jedi = False
```

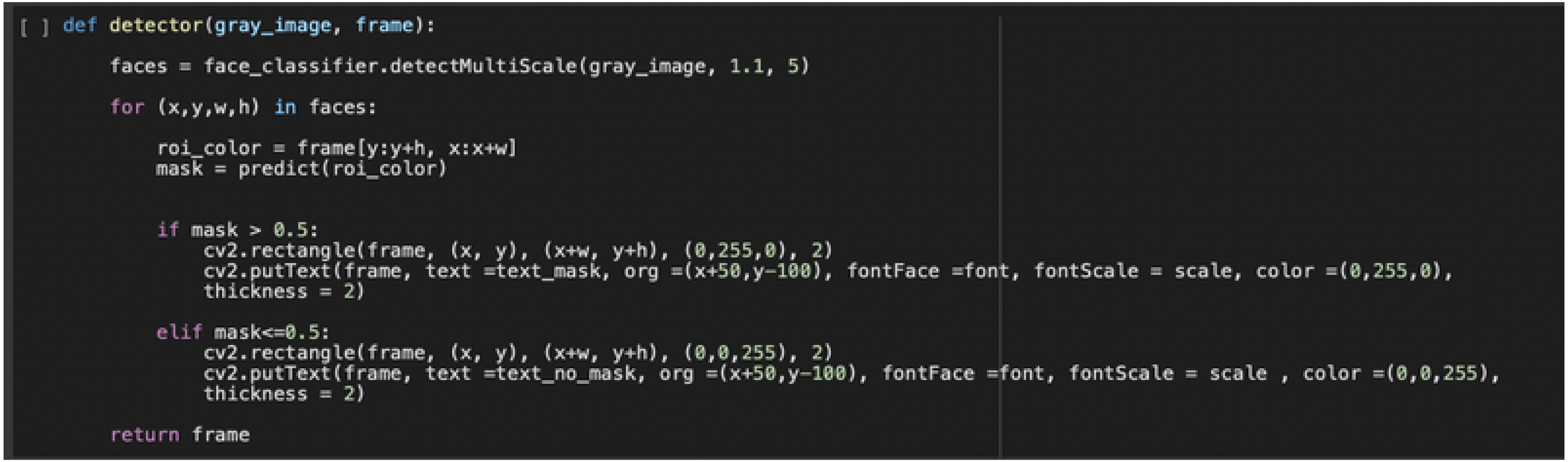
Uploading the Haar Cascade classifier & the pre-trained model

```
[ ] face_classifier = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
    mask_detection = tf.keras.models.load_model("/content/mask_detection.h5")
```

```
[ ] text_mask = "Mask On"
    text_no_mask = "Mask Off"
    font = cv2.FONT_HERSHEY_SIMPLEX
    scale = 0.8
```

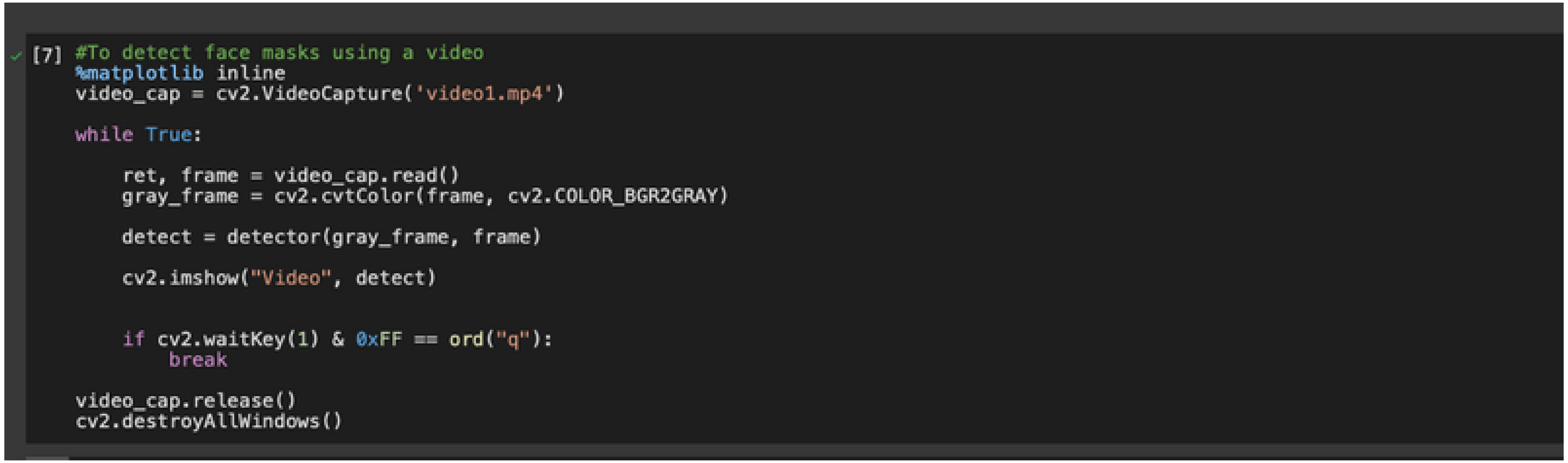
```
[ ] #We create a function that transforms the images
    def predict(image):

        face_frame = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        face_frame = cv2.resize(face_frame, (224, 224))
        face_frame = img_to_array(face_frame)
        face_frame = np.expand_dims(face_frame, axis=0)
        face_frame =  preprocess_input(face_frame)
        prediction = mask_detection.predict(face_frame)

        return prediction[0][0]
```

Building the detection system



```
[ ] def detector(gray_image, frame):

        faces = face_classifier.detectMultiScale(gray_image, 1.1, 5)

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

            roi_color = frame[y:y+h, x:x+w]
            mask = predict(roi_color)


            if mask > 0.5:
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,0), 2)
                cv2.putText(frame, text =text_mask, org =(x+50,y-100), fontFace =font, fontScale = scale, color =(0,255,0),
                thickness = 2)

            elif mask<=0.5:
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0,0,255), 2)
                cv2.putText(frame, text =text_no_mask, org =(x+50,y-100), fontFace =font, fontScale = scale , color =(0,0,255),
                thickness = 2)

        return frame
```

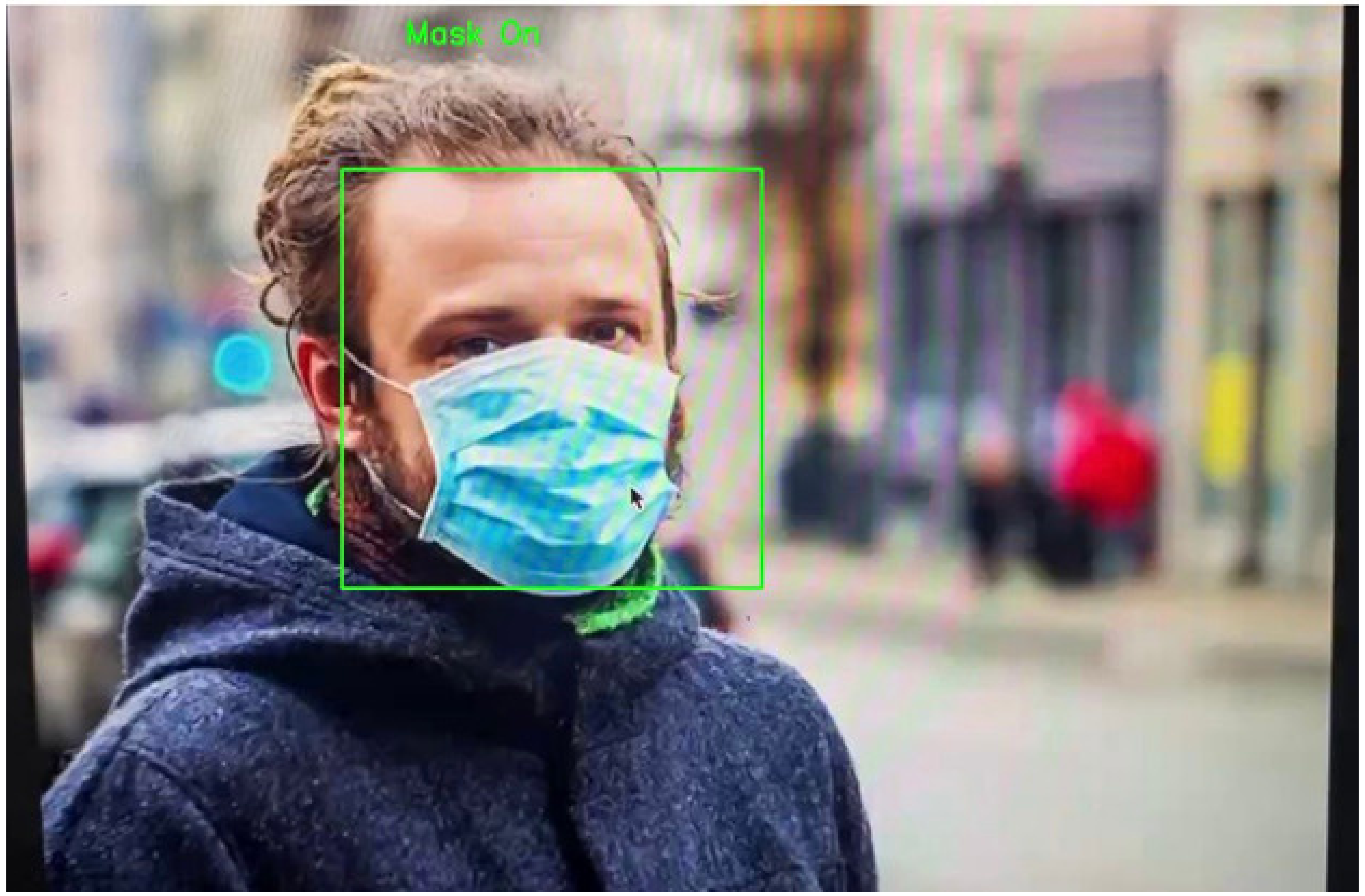
Building the detection system

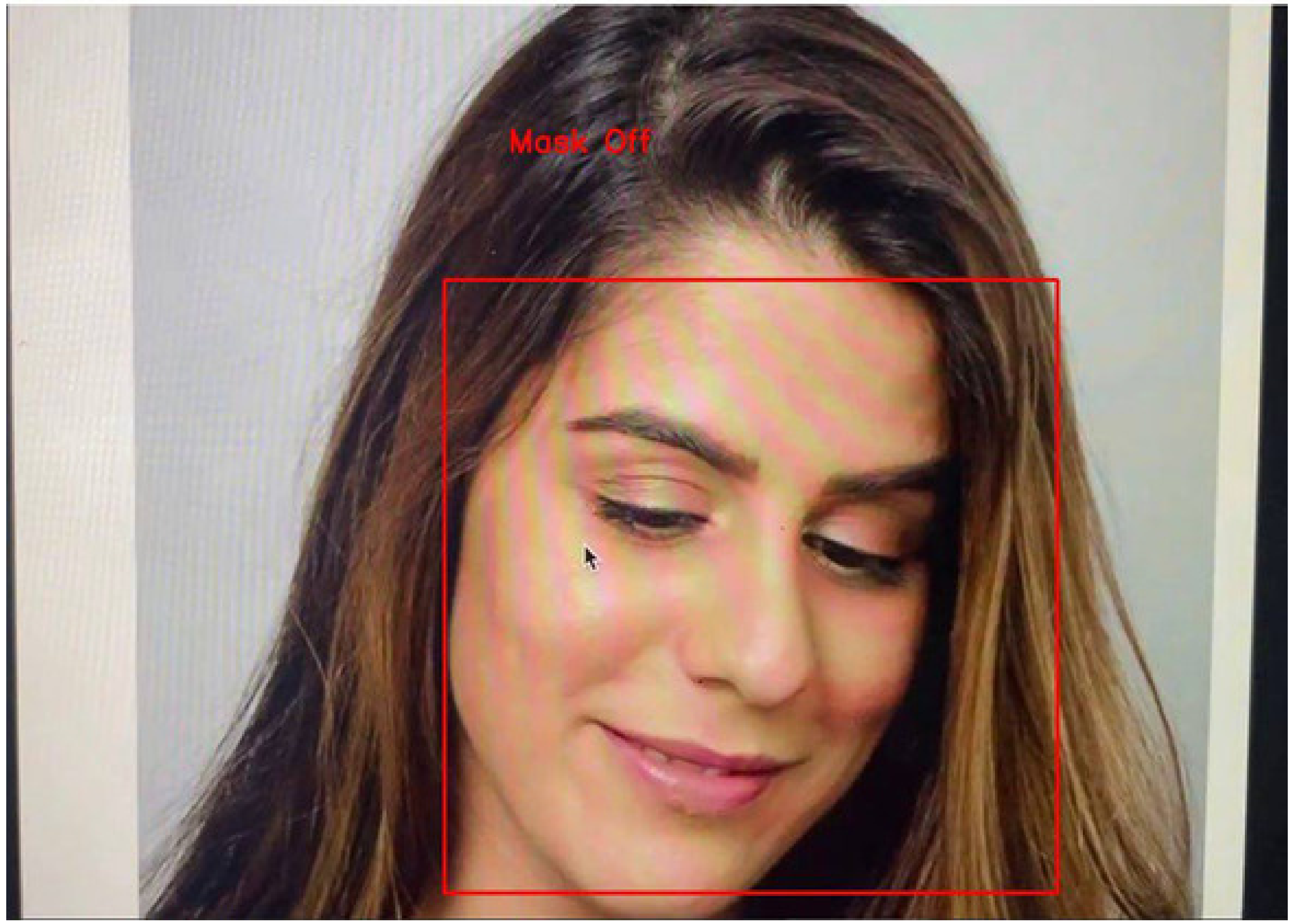


```
#To detect face masks using a video
%matplotlib inline
video_cap = cv2.VideoCapture('video1.mp4')

while True:

    ret, frame = video_cap.read()
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    detect = detector(gray_frame, frame)

    cv2.imshow("Video", detect)


    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

video_cap.release()
cv2.destroyAllWindows()
```

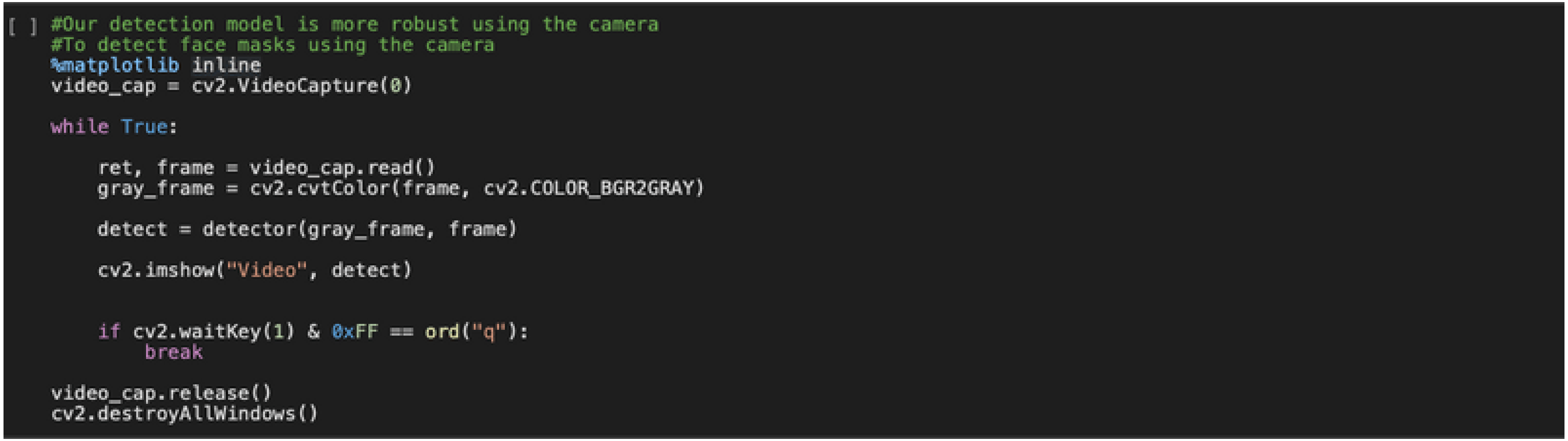
Building the detection system

The result of detecting face masks using a video





Building the detection system



```
[ ] #Our detection model is more robust using the camera
    #To detect face masks using the camera
    %matplotlib inline
    video_cap = cv2.VideoCapture(0)

    while True:

        ret, frame = video_cap.read()
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        detect = detector(gray_frame, frame)

        cv2.imshow("Video", detect)


        if cv2.waitKey(1) & 0xFF == ord("q"):
            break

    video_cap.release()
    cv2.destroyAllWindows()
```

Real-world mask detection application.

Recently and after the spreading of COVID-19 worldwide Lockdowns happened due to the COVID-19 outbreak, Face masks became mandatory for everyone while roaming outside. Using Computer vision to detect whether a person is wearing a mask or not. can help to keep him safe and everybody in the place. and in our crowded world having something like COVID-19 again is expected.

THANK YOU