

AI 825 Visual Recognition
Mini Project 1 – Face Mask Detection

Akshay Mahesh Gudiyawar (MT2020137)

Vikrant Punia (MT2020049)

Bhashkar Yadav (MT2020136)

Problem statement: To build a vision based automatic door entry system, to allow door open only if a human wears a mask. It should make use of 3 modules:

- Human Detection using YOLO/RCNN
- Face Detection using VIOLA JONES
- Mask Detection – any neural network method

Solution: To detect the mask in an image, the system has to first identify person in it, which we have done using YOLO network. Then, we make use of Viola Jones algorithm which is also called as haar-cascade to detect faces in the image. Finally, we make use of Densenet201, with re-training the last layers with mask dataset, we identify the masks in a person's face.

We have built the program as to optimize the three modules and avoid unnecessary computation. We feed the entire image frames to YOLO network first to identify any humans in it. It is configured to identify only 'person' classes out of its 80 classes, so we are avoiding unnecessary computation here. Next when YOLO detects a person in image, we capture the region of interest (human part of the image), and feed it to Viola Jones (Cascade classifier) to detect face in it as shown in Fig 1.

```
while True:
    ret, img = cam.read()
    img2, preds = yolo.detectCustomObjectsFromImage(input_image=img,
                                                    custom_objects=yolo.CustomObjects(person=True), input_type="array",
                                                    output_type="array",
                                                    minimum_percentage_probability=70,
                                                    display_percentage_probability=True,
                                                    display_object_name=True)

    for each in preds:
        if each["percentage_probability"] > 70: ## threshold for accepting person class
            rc = each["box_points"]
            x1, y1, x2, y2 = rc[0], rc[1], rc[2], rc[3]
            roi = img[y1:y2, x1:x2]

            gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
            faces = faceCascade.detectMultiScale(
                gray,
                scaleFactor=1.15,
                minNeighbors=5,
                minSize=(30, 30),
                flags=cv2.CASCADE_SCALE_IMAGE
            )
```

Fig. 1 Cropping ROI out of yolo detected persons to Cascade classifier

Here, we have set the minimum percentage probability for it to be detected as person as 70. Yolo.detectCustomObjects function returns a prediction dictionary which contains info about probability with which the class was detected, co-ordinates of bounding box. We are using these coordinates to crop out the region of interest (roi), which is the human part of frame and pass it to Cascade classifier (Viola Jones).

We have set the parameters of face classifier as, 'scaleFactor', which specifies how much the image size is reduced at each image scale as 1.15 which means the image is reduced 15% at each step to detect the face. 'minNeighbours', which will affect the quality of detected faces. Higher value results in less detections but with higher quality. 'minSize',

specifies the threshold size of images. Objects smaller than it are ignored. We found out the recommended optimal values for these parameters from this attached reference [1].

After detecting faces in the image, we next pass the image to densenet network, of which the last layer was re-trained with mask dataset [2].

```
for (x, y, w, h) in faces:
    im=cv2.resize(img,(224, 224))
    im=im/255.0
    im=np.reshape(im,(1,224,224,3))
    im = np.array(im)
    res=model.predict(im)
    nm = res[0][0]
    m = res[0][1]
    if m> nm:
        print("Mask Detected, Door Open")
        label = 'MASK'
        box_color = (0, 255, 0)
    elif nm >m:
        print("No mask Detected, Door closed")
        label = 'NO MASK'
        box_color = (255, 0, 0)

    cv2.rectangle(img2, (x+x1, y+y1), (x+w+x1, y+h+y1), box_color, 2)
    cv2.rectangle(img2,(x+x1, y+y1-40), (x+w+x1, y+y1), box_color, -1)
    cv2.putText(img2, label, (x+x1, y+y1-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,255,255), 2)
```

Fig. 2 Using densenet model to detect masked faces and putting bounding box

We take the prediction of model and compare the probabilities of mask and no-mask classes and accordingly print the output on console and set the label and bounding box color. Lastly, we use OpenCV rectangle and putText functions to draw bounding box and class label on the output frame.

By cropping out region of interest when yolo detects the person in frame, it avoids unnecessary computation of Cascade face classifier and mask detector.

Modules Explanation:

1) YOLO (You Only Look Once) for Real Time Object for human detection

YOLO is an effective real-time object detection algorithm, first described in 2015 by Joseph Redmon. In our code we will use the YOLOv3 to detect humans. The network architecture of YOLO is as shown in Figure 3.

YOLOv3 uses a variant of Darknet, a framework to train neural networks, which originally has 53 layers. For the detection task another 53 layers are stacked onto it, accumulating to a total of a 106-layer fully convolutional architecture.

	Type	Filters	Size	Output
	Convolutional	32	3×3	256×256
	Convolutional	64	$3 \times 3 / 2$	128×128
1x	Convolutional	32	1×1	128×128
	Convolutional	64	3×3	
	Residual			
	Convolutional	128	$3 \times 3 / 2$	64×64
2x	Convolutional	64	1×1	64×64
	Convolutional	128	3×3	
	Residual			
	Convolutional	256	$3 \times 3 / 2$	32×32
8x	Convolutional	128	1×1	32×32
	Convolutional	256	3×3	
	Residual			
	Convolutional	512	$3 \times 3 / 2$	16×16
8x	Convolutional	256	1×1	16×16
	Convolutional	512	3×3	
	Residual			
	Convolutional	1024	$3 \times 3 / 2$	8×8
4x	Convolutional	512	1×1	8×8
	Convolutional	1024	3×3	
	Residual			
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig 3 YOLO Network Architecture

YOLOv3 regards target detection as a regression problem i.e. It divides the image into $S \times S$ grids. If the center of the object falls into a grid, then this grid is responsible for detecting the object and the positions of these bounding boxes, Confidence, category probability. It then uses logistic regression to predict a score for each bounding box. Fig 4 below explains how YOLO algorithm works.

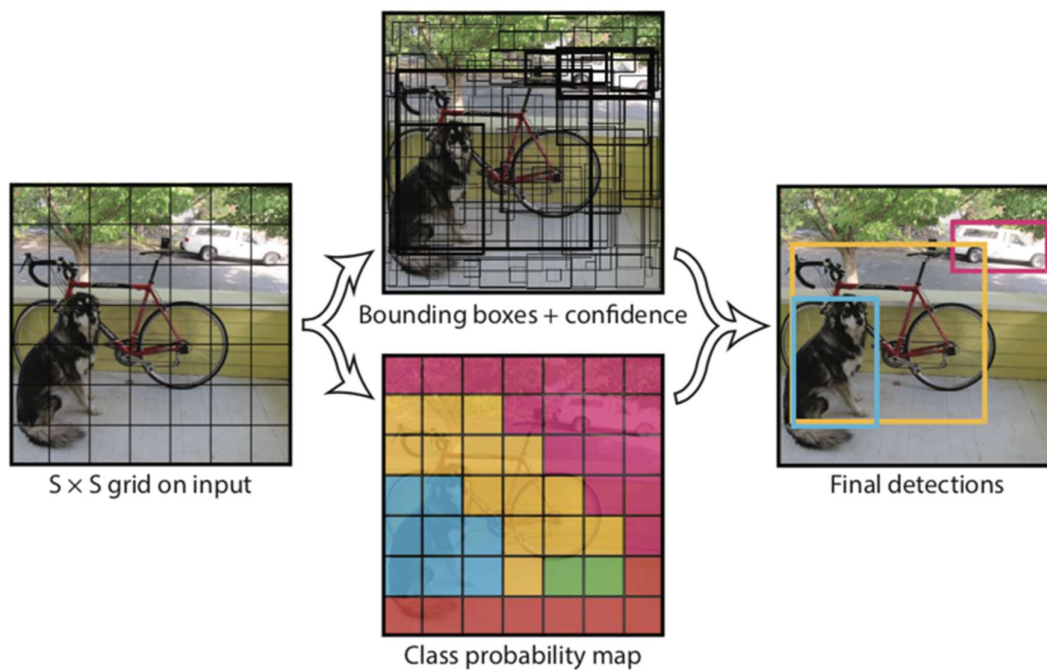


Fig 4. YOLO bounding box Detection

In our use case since we have to only track of humans, we have modified our YOLO code to only detect humans as shown in Figure 1 (Yolo detectCustomObject function).

Performance comparison of YOLO with other popular model object detection networks Fig 5

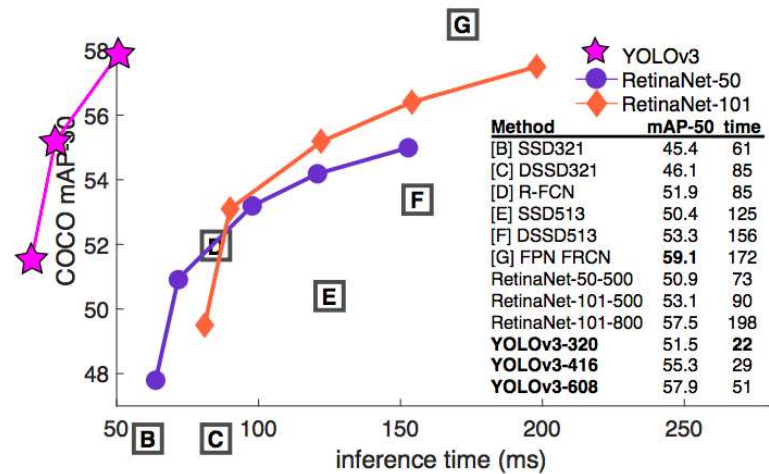


Fig 5 YOLO performance comparison

As we can see, YOLO takes relatively less time for detection i.e. its fast and has good mAP also. mAP (mean average precision) compares ground truth bounding box to detected box and returns a score. The higher the score, better the models predictions are. The biggest advantage of YOLO over other networks is its speed.

Performance/Accuracy results of YOLO:

YOLO was able to detect person class in every case with probability > 90%

2) Viola Jones (Cascade Classifier) for Face detection

Viola Jones is an object-recognition framework that allows the detection of image features in real-time, its application has proven to be notable in real-time face detection. Its primarily designed for detection of frontal faces. Its downside it doesn't perform well in detecting faces sideways. It searches for 3 types of haar-like features in images: Edge features, Line features, Four-sided features as shown in diagram below Fig 6.

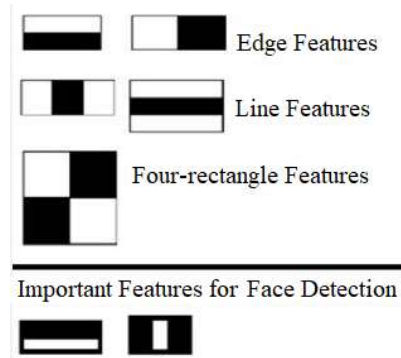



Fig 6. Haar-like features used by Viola Jones algorithm

The horizontal and the vertical features describe what eyebrows and the nose, respectively. The value of a feature is calculated with the help of an Integral Image which subtracts the value of white area from black in a haar-like feature. Next, for detection, it uses Adaptive Boosting (Ensemble model) which combines the features, which are analogous to weak classifiers into one strong classifier with weights as shown in following Figure 7.



$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \dots$$

Fig 7. combination of features using Ensemble

Viola Jones algorithm uses Cascading concept to boost the speed and accuracy of the model as shown in Fig 8. The concept is that, in a sub-region it first checks for its best feature if it is there in that region, if it is not present, then the entire sub-region is discarded, reasoning being it cannot contain face if it doesn't have the best feature. Similarly, if it detects the first best feature, it checks for second-best feature in that sub-region and discards if it is not found. This way, it avoids unnecessary computation and speeds up the model.

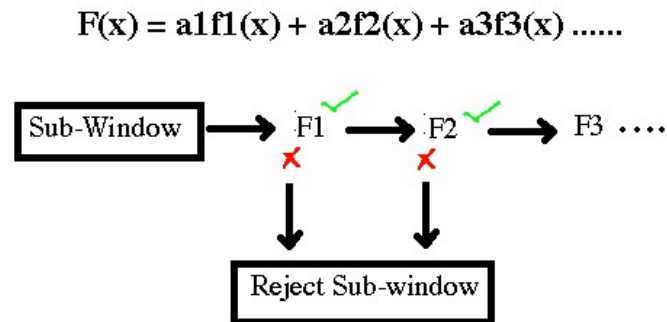


Fig 8. Cascading to boost speed and accuracy of the model

Performance Results of Viola Jones (Cascade) module:

Viola Jones algorithm was able to detect straight faces, but it didn't perform well and was not able to detect faces in sideways, up or down direction.

3) Mask detection using Transfer learning of densenet201

For mask detection task, we took densenet121 network and added a fully connected layer with two nodes, since we have two output class mask / no-mask.

The densenet architecture is as shown in Figure 9. An output of the previous layer acts as an input of the second layer by using composite function operation.

This composite operation consists of the convolution layer, pooling layer, batch normalization, and non-linear activation

layer. These connections mean that the network has $L(L+1)/2$ direct connections. L is the number of layers in the architecture.

The DenseNet has different versions, like DenseNet-121, DenseNet-160, DenseNet-201, etc. The numbers denote the number of layers in the neural network. For our case, we used DenseNet201 model. The DenseNet is divided into Dense-Blocks, where a number of filters are different, but dimensions within the block are the same.

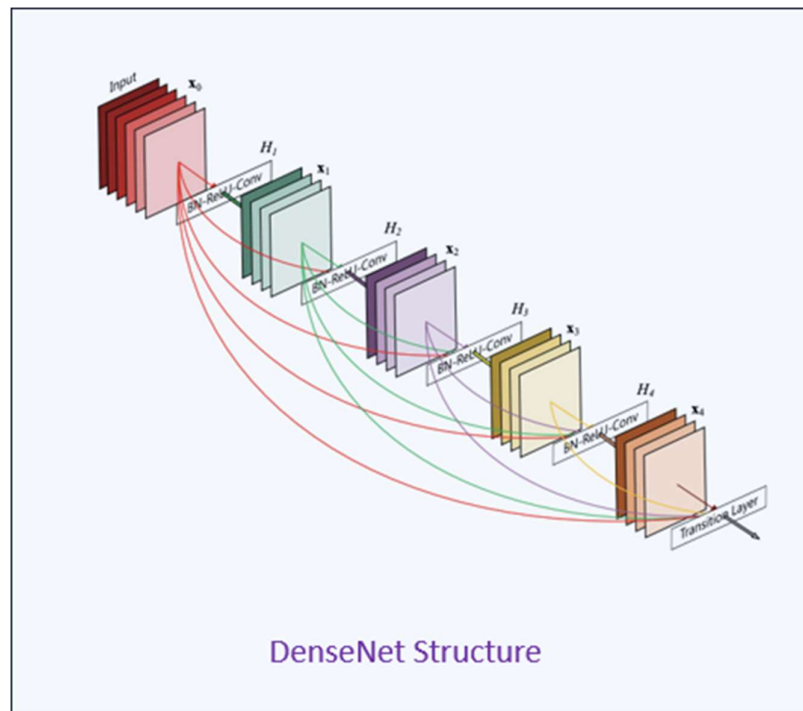


Fig 9. Densenet architecture

We retrained the last added layer with mask dataset [2]. It contained 2452 train images and 768 test images consisting of mask and no-mask images. We used this model to predict the face sub-region with mask/no-mask labels.

Performance results of mask detection module:

The model performed far better in detecting masked faces than non-masked faces. It correctly predicted 330 masked faces, and detected 19 out of total 384 test images of each class respectively. The F1 score for mask class was 0.85 and for no-mask it was 0.049.


```
jupyter mask_detection_transfer_training Last Checkpoint: 14 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Pyt

print("Total test images: {}".format(total))

print("Mask recall: {}".format(mask_recall))
print("No-mask recall: {}".format(no_mask_recall))

print("Mask precision: {}".format(mask_precision))
print("No-mask precision: {}".format(no_mask_precision))

print("Mask F1 score: {}".format(mask_f1))
print("No-mask F1 score: {}".format(no_mask_f1))

print("Classification accuracy: {}".format(accuracy))
print("Weighted recall: {}".format(recall))
print("Weighted precision: {}".format(precision))
print("Weighted F1 score: {}".format(f1))

Mask correct predictions: 330
Mask incorrect predictions: 54
No-mask correct predictions: 19
No-mask incorrect predictions: 365
Mask total images: 384
No-mask total images: 384
Total test images: 768
Mask recall: 0.859375
No-mask recall: 0.049479166666666664
Mask precision: 0.859375
No-mask precision: 0.049479166666666664
Mask F1 score: 0.859375
No-mask F1 score: 0.049479166666666664
Classification accuracy: 0.4544270833333333
Weighted recall: 0.4544270833333333
Weighted precision: 0.4544270833333333
Weighted F1 score: 0.4544270833333333
```

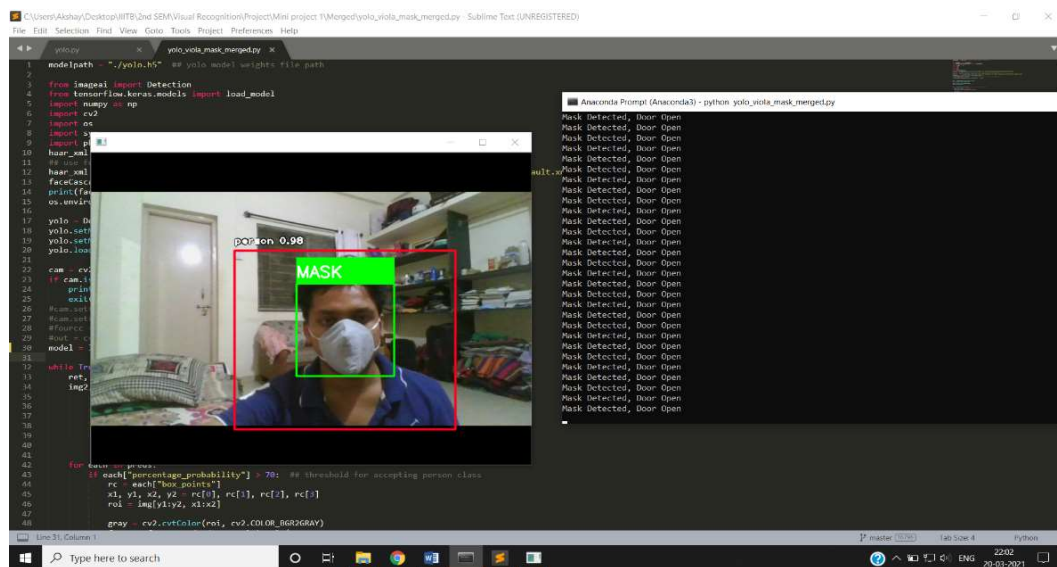
Overall System Performance:

We tested the system using live webcam feed and video file input. It was able to detect masks properly, but there were some instances where mask was not detected correctly. We put a print statement to display in console to signal door open when the mask is detected in face, and door-closed when mask wasn't detected. The output results are uploaded to below drive link on four different people with and without mask. For visualization purpose, we have set red box for yolo-human detection, green box for mask-class detection by Viola Jones algo, and blue box for non-mask class.

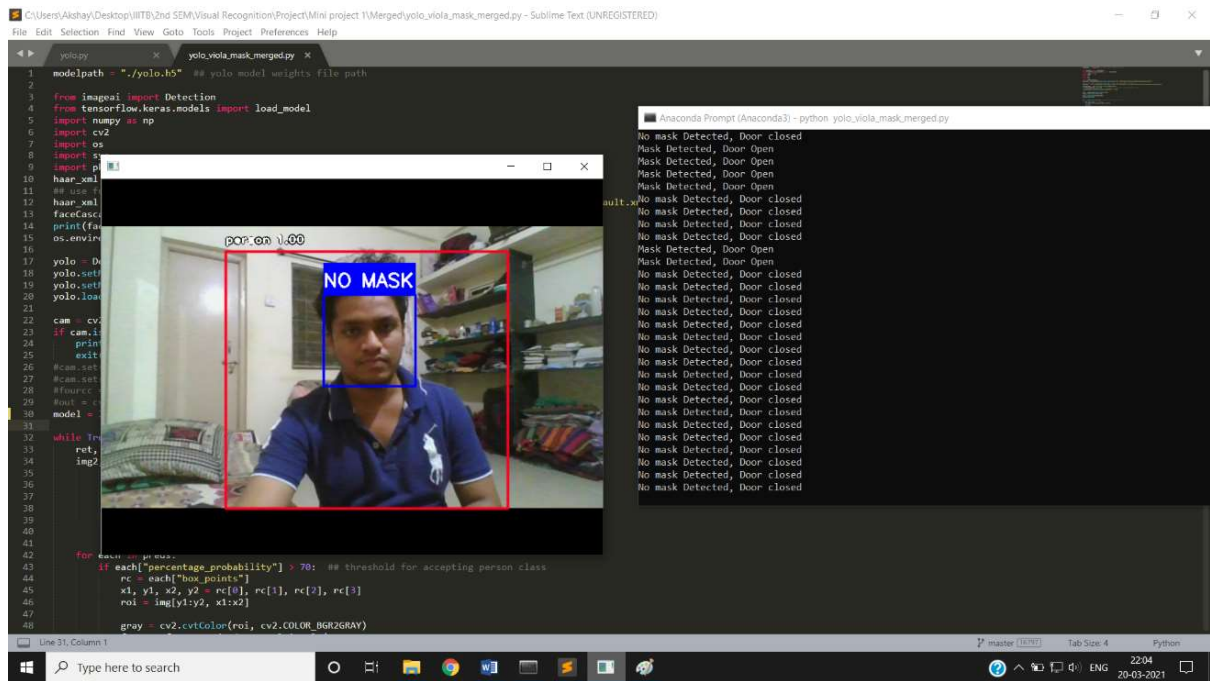
The video output files of test cases are available here:

<https://drive.google.com/folderview?id=1UGTnCTJiZRs7s6BzA6x8f78V3zOGY46h>

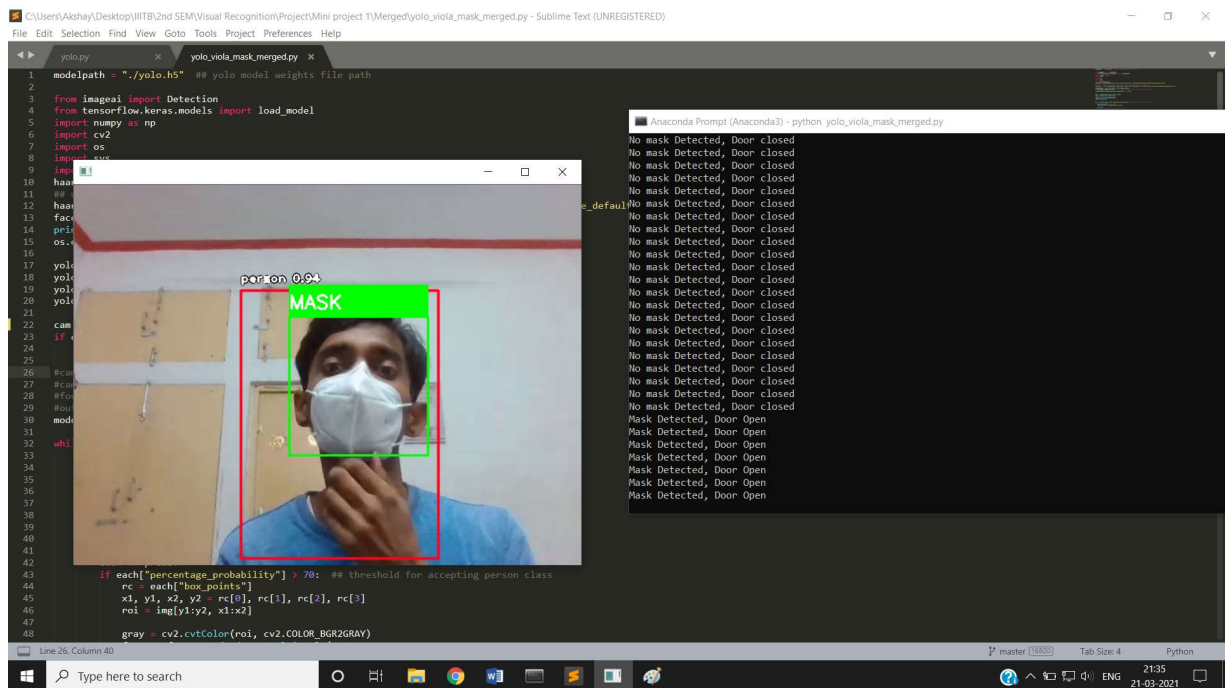
Some of the screenshots from video output are attached below:



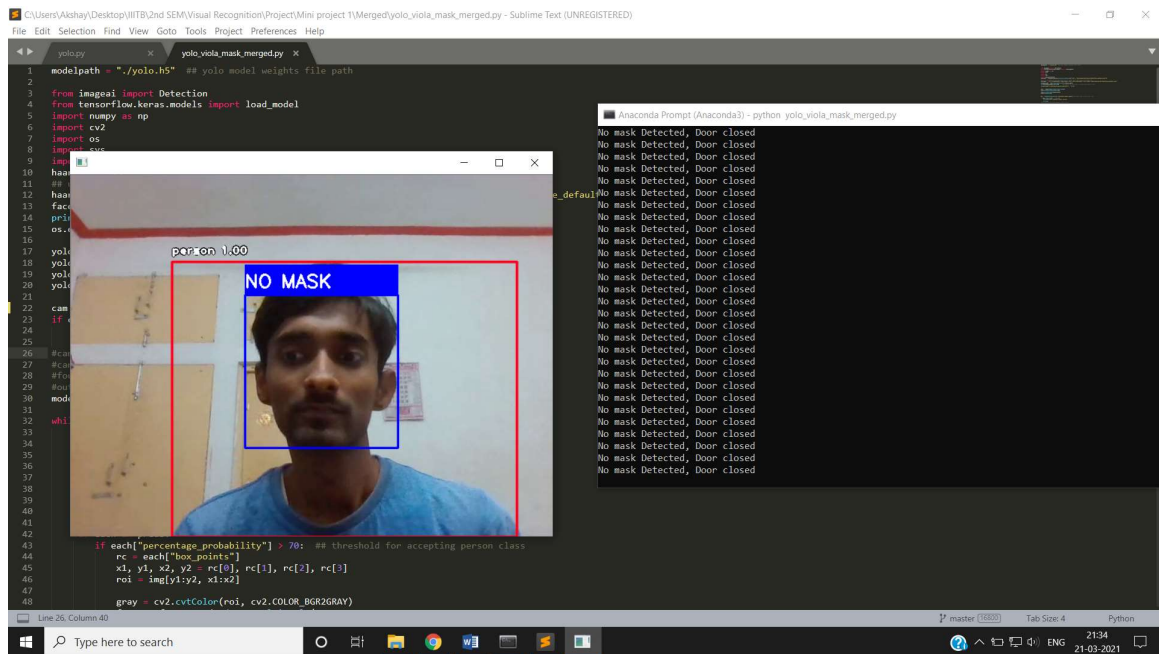
(No-Mask):



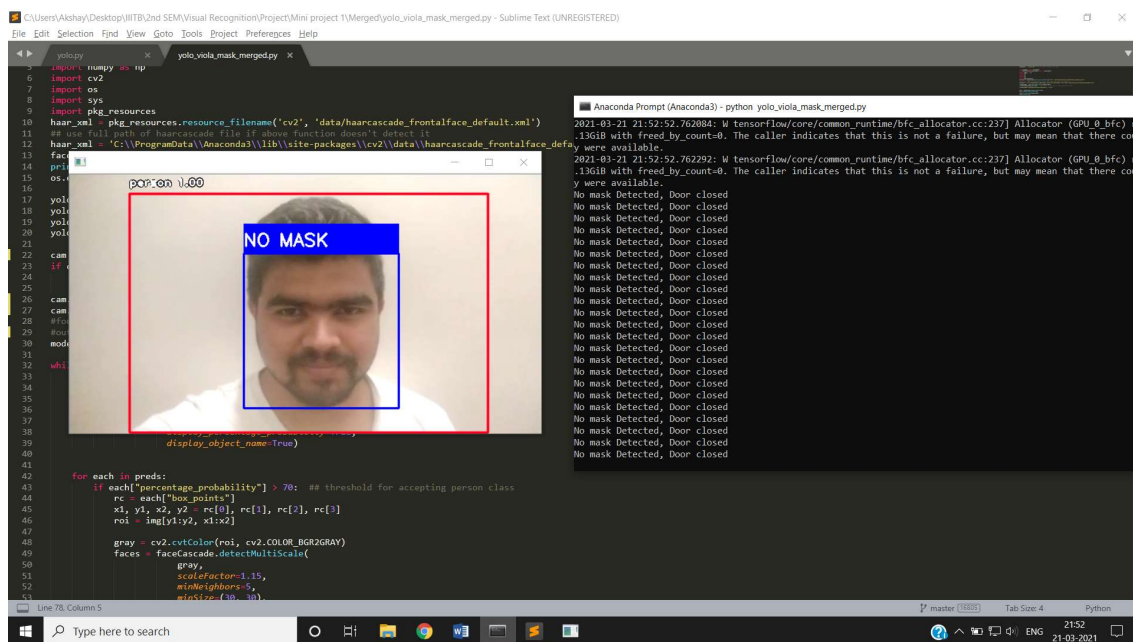
(Mask):



(No-Mask):



(No-Mask):



References:

[1] <https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmultiscale-parameters>

[2] <https://github.com/J-Douglas/Face-Mask-Detection/tree/master/datasets/compiled>

[3] <https://drive.google.com/folderview?id=1UGTnCTJiZRs7s6BzA6x8f78V3zOGY46h>

(Output video files of test cases)