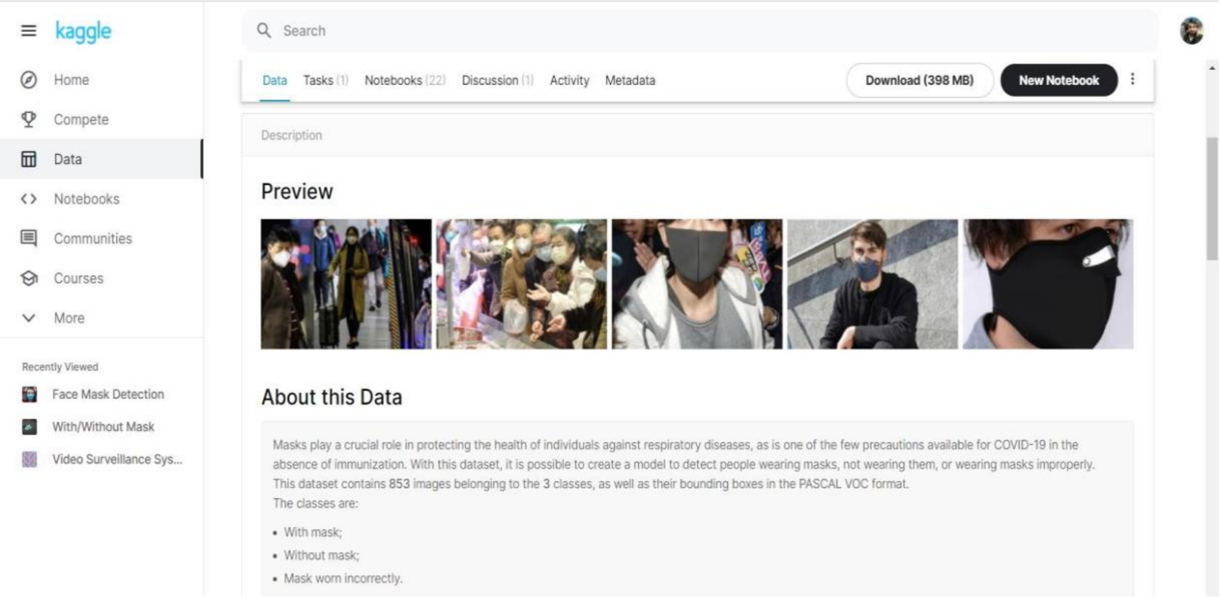
Face Mask detection using Azure Custom Vision

Dataset for training the model

Source for training the model has been taken in the **kaggle.com** for training the model in custom vision azure platform for images. The data for mask and without mask has been inputted equally in form of 1:1 ratio. These datasets satisfy certain kind of constraints divided across classes such as:

- ✓ One or more individual in image with or without mask
- ✓ One or more individual in image some with mask and some without mask
- ✓ Mask worn incorrectly

The above varieties of data in form of image has been used to train the model in custom vision.



About this Data

Masks play a crucial role in protecting the health of individuals against respiratory diseases, as is one of the few precautions available for COVID-19 in the absence of immunization. With this dataset, it is possible to create a model to detect people wearing masks, not wearing them, or wearing masks improperly.
This dataset contains 853 images belonging to the 3 classes, as well as their bounding boxes in the PASCAL VOC format.
The classes are:

- With mask;
- Without mask;
- Mask worn incorrectly.

Fig 1. Kaggle dataset

The picture represents the **repository** for the face mask detection containing the dataset. In this dataset there are hundreds of images, but we chose specific images as dataset which satisfies the constraints for the most varieties of dataset to achieve the successful training of the model to predict the source in the live environment with more **precise** output.

Training the model using the Custom Vision

After the collection of the dataset now we enter the step to train the model using the dataset. Certain steps have been undergone to achieve training the model. They are:

- ✓ Create the project in the custom vision platform of azure
- ✓ Uploading the images in the custom vision
- ✓ Splitting the images as tags ie…, with and without mask
- ✓ Train the model using the service in custom vision
- ✓ Test the model with any input image in the custom vision
- ✓ Publish the model and retrieve the API key, resource ID, endpoint key for the implementation using the python

Project on custom vision named FaceDetection2020

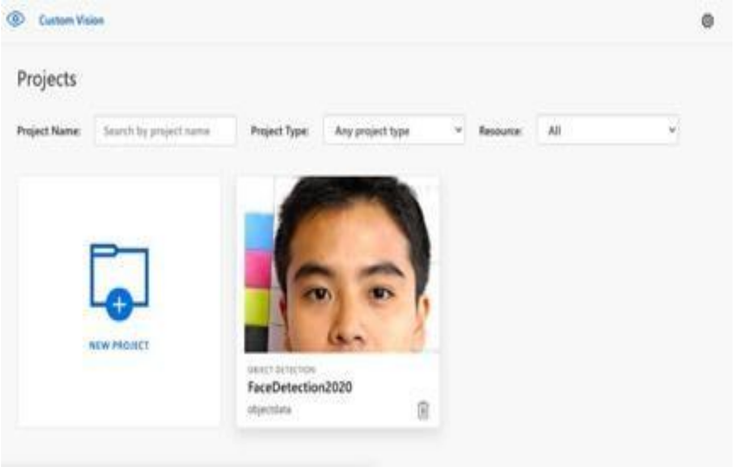


Fig 2.Custom vision project

Dataset in the custom vision azure platform

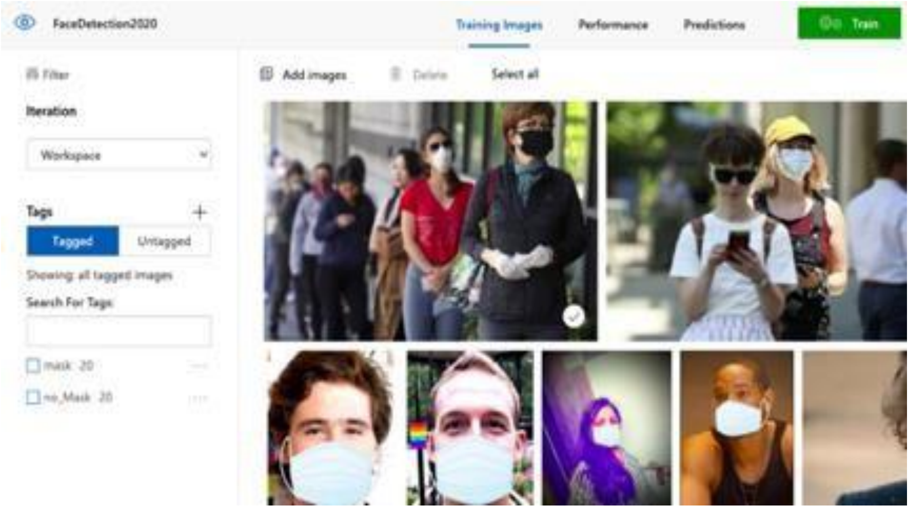
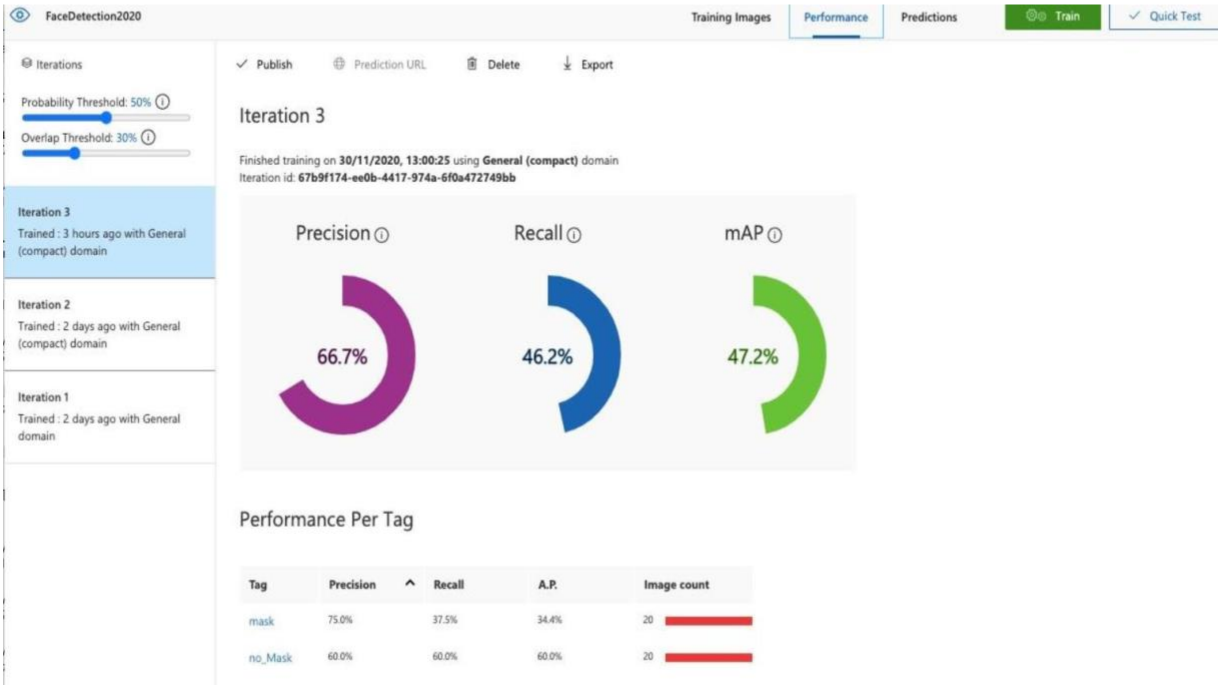


Fig 3. Custom vision dataset

After the model has been trained the prediction analysis looks as follows:



Probability Threshold: 50%

Overlap Threshold: 30%

Iteration 3
Trained : 3 hours ago with General (compact) domain

Iteration 2
Trained : 2 days ago with General (compact) domain

Iteration 1
Trained : 2 days ago with General domain

Iteration 3

Finished training on **30/11/2020, 13:00:25** using **General (compact)** domain
Iteration id: **67b9f174-ee0b-4417-974a-6f0a472749bb**

Precision 66.7%

Recall 46.2%

mAP 47.2%

Performance Per Tag

Tag	Precision	Recall	A.P.	Image count
mask	75.0%	37.5%	34.4%	20
no_Mask	60.0%	60.0%	60.0%	20

Fig 4. Custom vision model performance

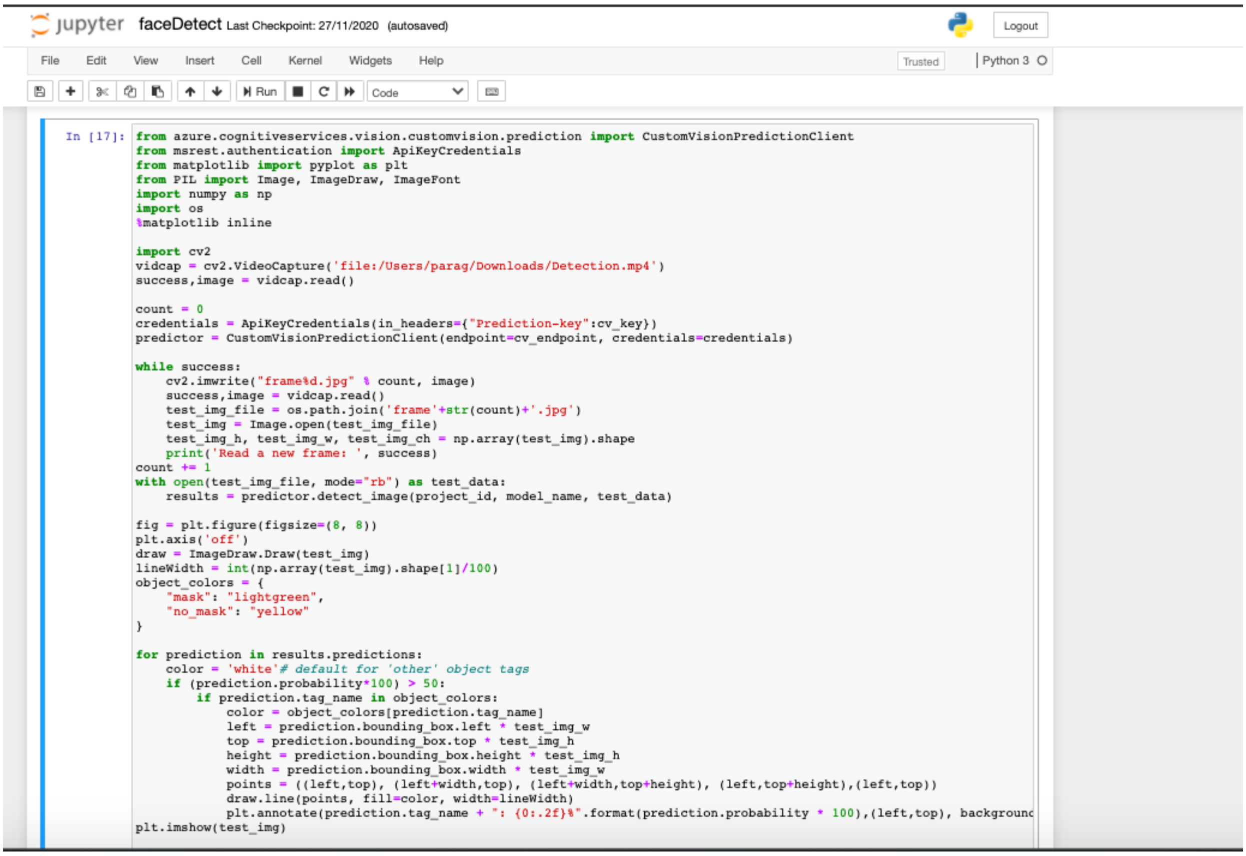
Implementing the trained model in the application:

After the model has been trained we implement it in the application using the trained custom vision model using the API credentials provided in the custom vision platform.

Python and **opencv** has been used in order to analyse the individuals has face mask o not when video has been inputted. The pathway followed up in the implementation are listed in steps as below:

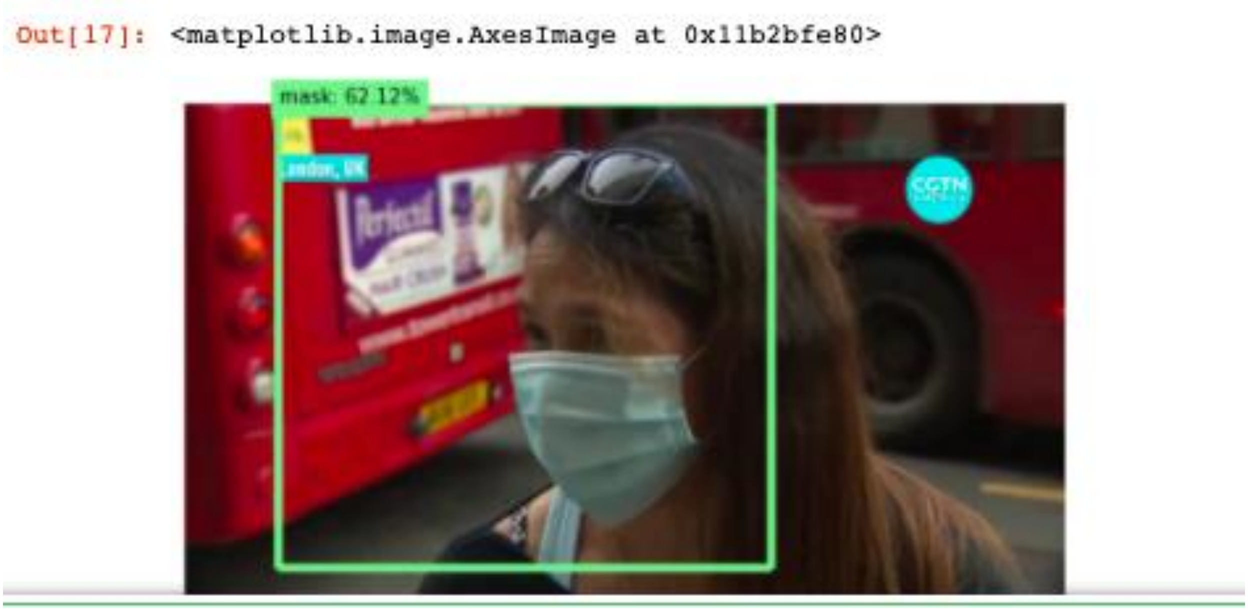
- ✓ Input video and read it using opencv
- ✓ Convert the video as images by dividing as frames and stores as JPG format
- ✓ Using the methods of opencv and the credentials of custom vision trained model we identify the faces
- ✓ Identified faces are drawn a rectangular box and classify as with mask or without mask
- ✓ The drawn rectangular box in the image frames are stored as JPG

The python implementation is as follows:



```
In [17]: from azure.cognitiveservices.vision.customvision.prediction import CustomVisionPredictionClient
from msrest.authentication import ApiKeyCredentials
from matplotlib import pyplot as plt
from PIL import Image, ImageDraw, ImageFont
import numpy as np
import os
%matplotlib inline

import cv2
vidcap = cv2.VideoCapture('file:/Users/parag/Downloads/Detection.mp4')
success,image = vidcap.read()

count = 0
credentials = ApiKeyCredentials(in_headers={"Prediction-key":cv_key})
predictor = CustomVisionPredictionClient(endpoint=cv_endpoint, credentials=credentials)

while success:
    cv2.imwrite("frame%d.jpg" % count, image)
    success,image = vidcap.read()
    test_img_file = os.path.join('frame'+str(count)+'.jpg')
    test_img = Image.open(test_img_file)
    test_img_h, test_img_w, test_img_ch = np.array(test_img).shape
    print('Read a new frame: ', success)
count += 1
with open(test_img_file, mode="rb") as test_data:
    results = predictor.detect_image(project_id, model_name, test_data)

fig = plt.figure(figsize=(8, 8))
plt.axis('off')
draw = ImageDraw.Draw(test_img)
lineWidth = int(np.array(test_img).shape[1]/100)
object_colors = {
    "mask": "lightgreen",
    "no_mask": "yellow"
}

for prediction in results.predictions:
    color = 'white'# default for 'other' object tags
    if (prediction.probability*100) > 50:
        if prediction.tag_name in object_colors:
            color = object_colors[prediction.tag_name]
            left = prediction.bounding_box.left * test_img_w
            top = prediction.bounding_box.top * test_img_h
            height = prediction.bounding_box.height * test_img_h
            width = prediction.bounding_box.width * test_img_w
            points = ((left,top), (left+width,top), (left+width,top+height), (left,top+height),(left,top))
            draw.line(points, fill=color, width=lineWidth)
            plt.annotate(prediction.tag_name + ": {0:.2f}%".format(prediction.probability * 100),(left,top), backgrounc
plt.imshow(test_img)
```

Result:-



```
Out[17]: <matplotlib.image.AxesImage at 0x11b2bfe80>
```

Conclusion

To relieve the spread of COVID-19 pandemic, measures must be taken. We have demonstrated a face mask detector. To train, validate and test the model, we utilized the dataset that comprised of mask faces pictures and unmasked faces pictures. These pictures were taken from different assets like Kaggle datasets. The model was construed on pictures and video streams. To choose a base model, we assessed the measurements like exactness, accuracy, and review with the best exhibition. This face mask detector can be sent in numerous zones like shopping centres, airports and other substantial traffic spots to screen the general population and to evade the spread of the infection by checking who is observing essential principles and who isn't.