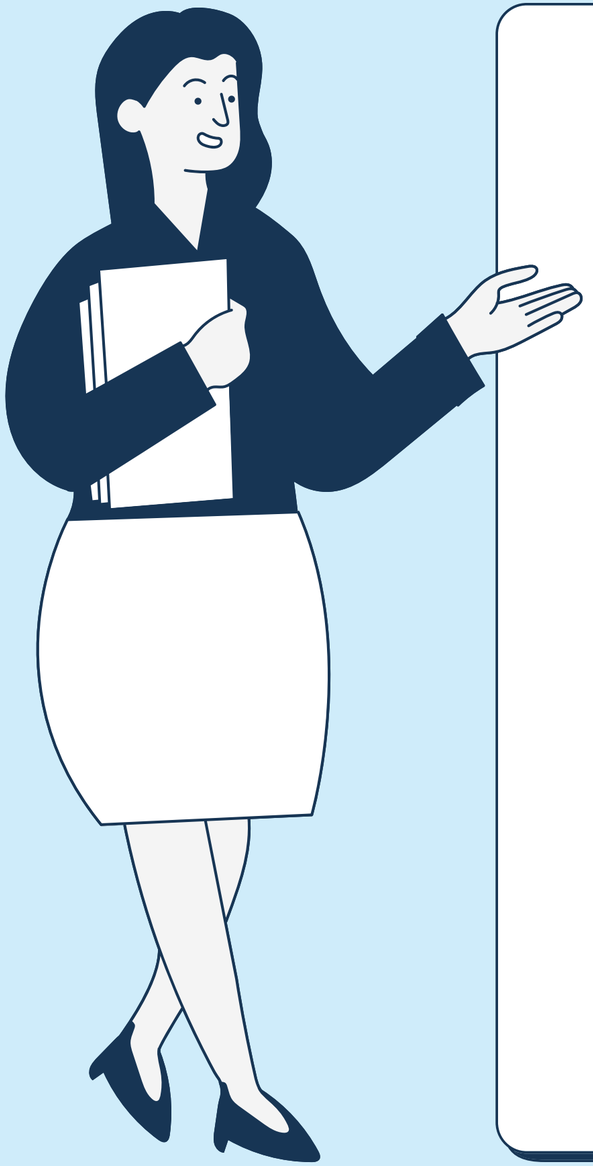
FACE MASK DETECTION SYSTEM USING YOLO ALGORITHM for BETTER IMPLEMENTATION of COVID-19 SAFETY PROTOCOLS

Table of Contents

What we'll discuss today

- Introduction
- Significance
- Goals and objectives
- Methodology
- Results
- Conclusion



Introduction

As of August 1, 2021, the number of COVID-19 cases in the Philippines reached approximately 1.6 million – with 63,000+ being considered active (CNN Philippines Staff, 2021). With the existence of transmissive variants like Delta that overwhelmed the healthcare system of countries like India and currently, Indonesia, Malaysia and Thailand, it’s more relevant than ever to implement COVID-19 safety protocols.

Goals and Objectives

The Program

Develop a system that can determine and classify in real-time if users are wearing a mask or not

Data Gathering

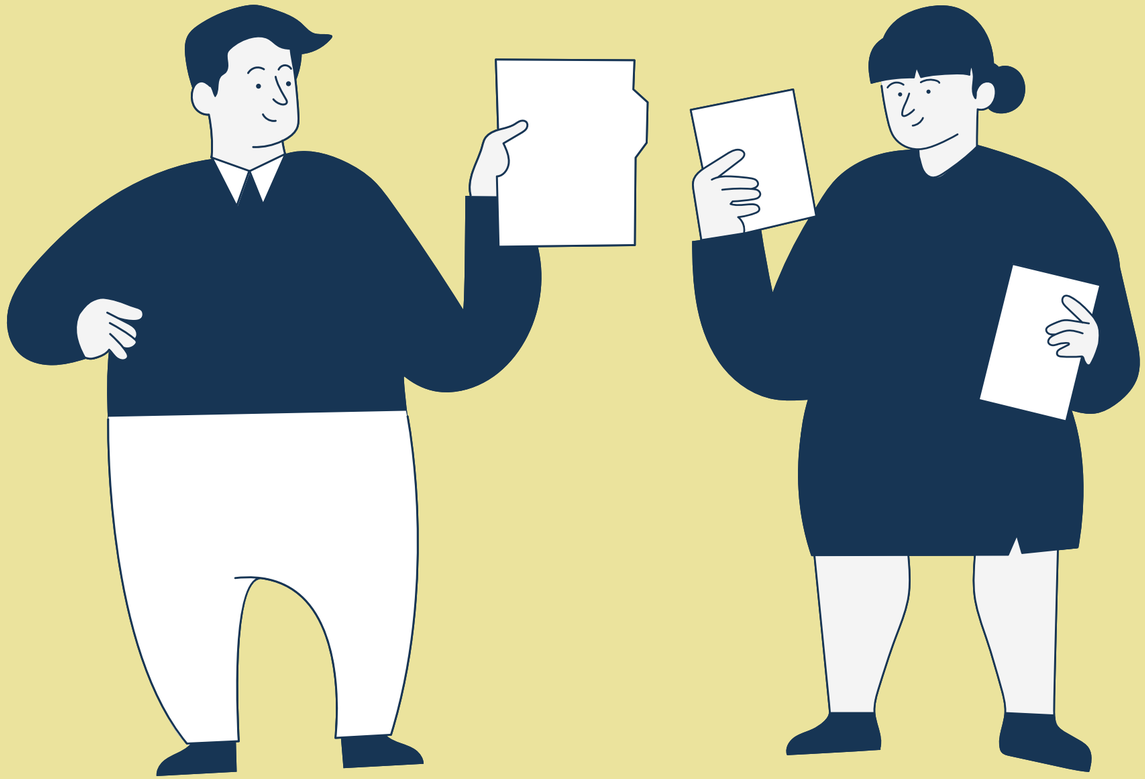
Develop an evaluation system that determine the accuracy of the image processing and classification.

Significance of the Study

This project will be greatly beneficial to:

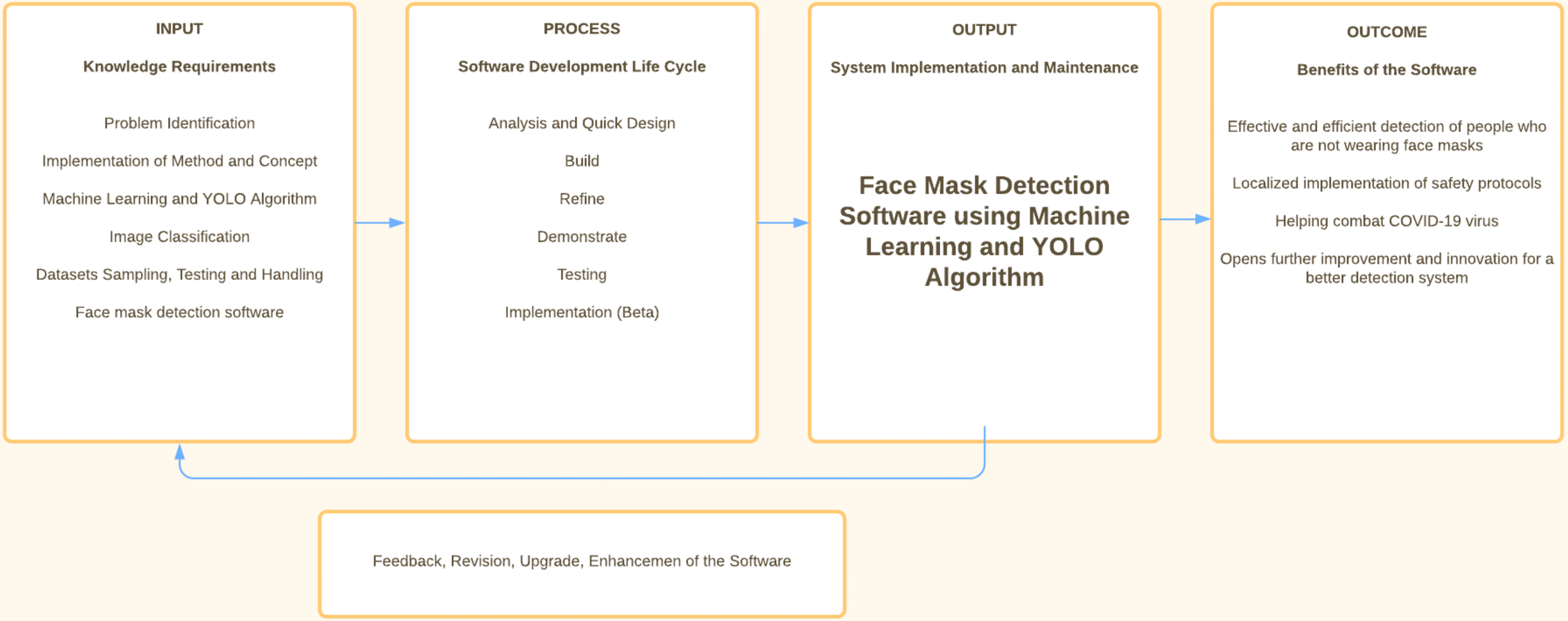
- Barangay
- Healthcare Sectors
- Police Force
- Future Researchers

Methodology

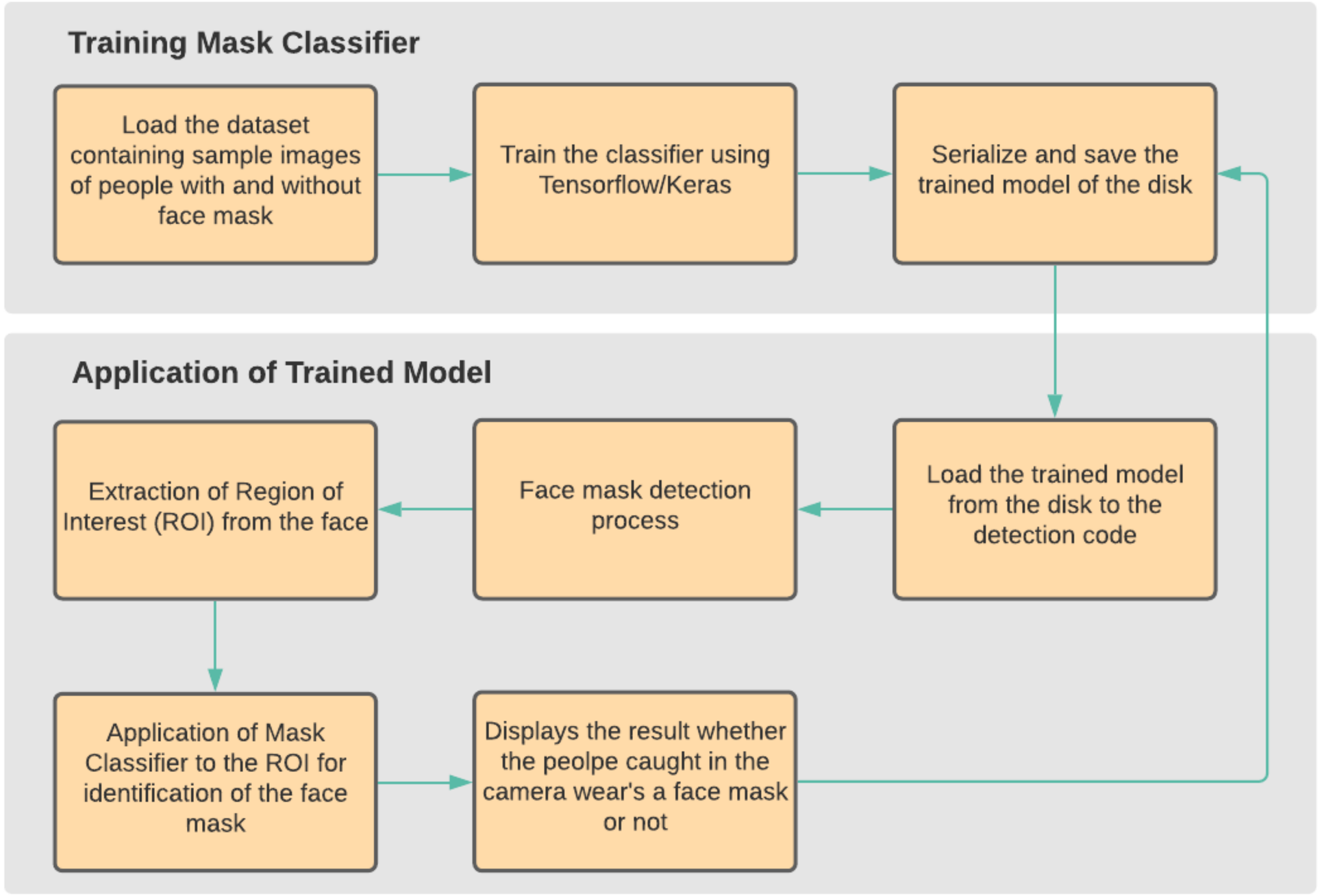


The Design and Methods

Conceptual Framework


INPUT
Knowledge Requirements
Problem Identification
Implementation of Method and Concept
Machine Learning and YOLO Algorithm
Image Classification
Datasets Sampling, Testing and Handling
Face mask detection software
PROCESS
Software Development Life Cycle
Analysis and Quick Design
Build
Refine
Demonstrate
Testing
Implementation (Beta)
OUTPUT
System Implementation and Maintenance
Face Mask Detection Software using Machine Learning and YOLO Algorithm
OUTCOME
Benefits of the Software
Effective and efficient detection of people who are not wearing face masks
Localized implementation of safety protocols
Helping combat COVID-19 virus
Opens further improvement and innovation for a better detection system
Feedback, Revision, Upgrade, Enhancemen of the Software

Research Design


Training Mask Classifier
Load the dataset containing sample images of people with and without face mask
Train the classifier using Tensorflow/Keras
Serialize and save the trained model of the disk
Application of Trained Model
Load the trained model from the disk to the detection code
Face mask detection process
Extraction of Region of Interest (ROI) from the face
Application of Mask Classifier to the ROI for identification of the face mask
Displays the result whether the peolpe caught in the camera wear's a face mask or not

Development Procedure

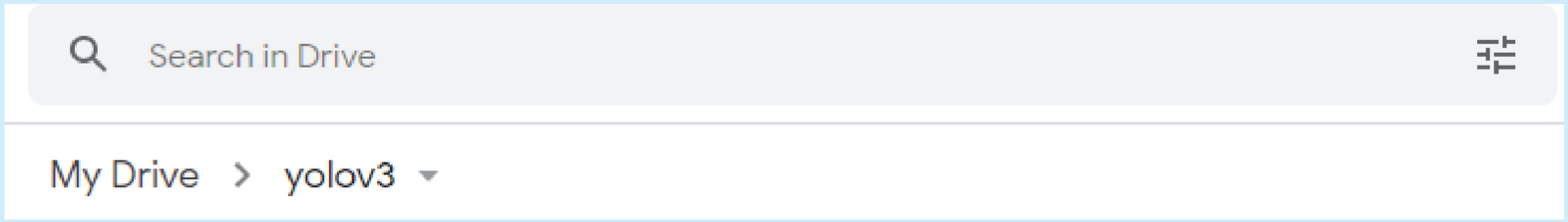


Figure 3 'yolov3' Google Drive Folder

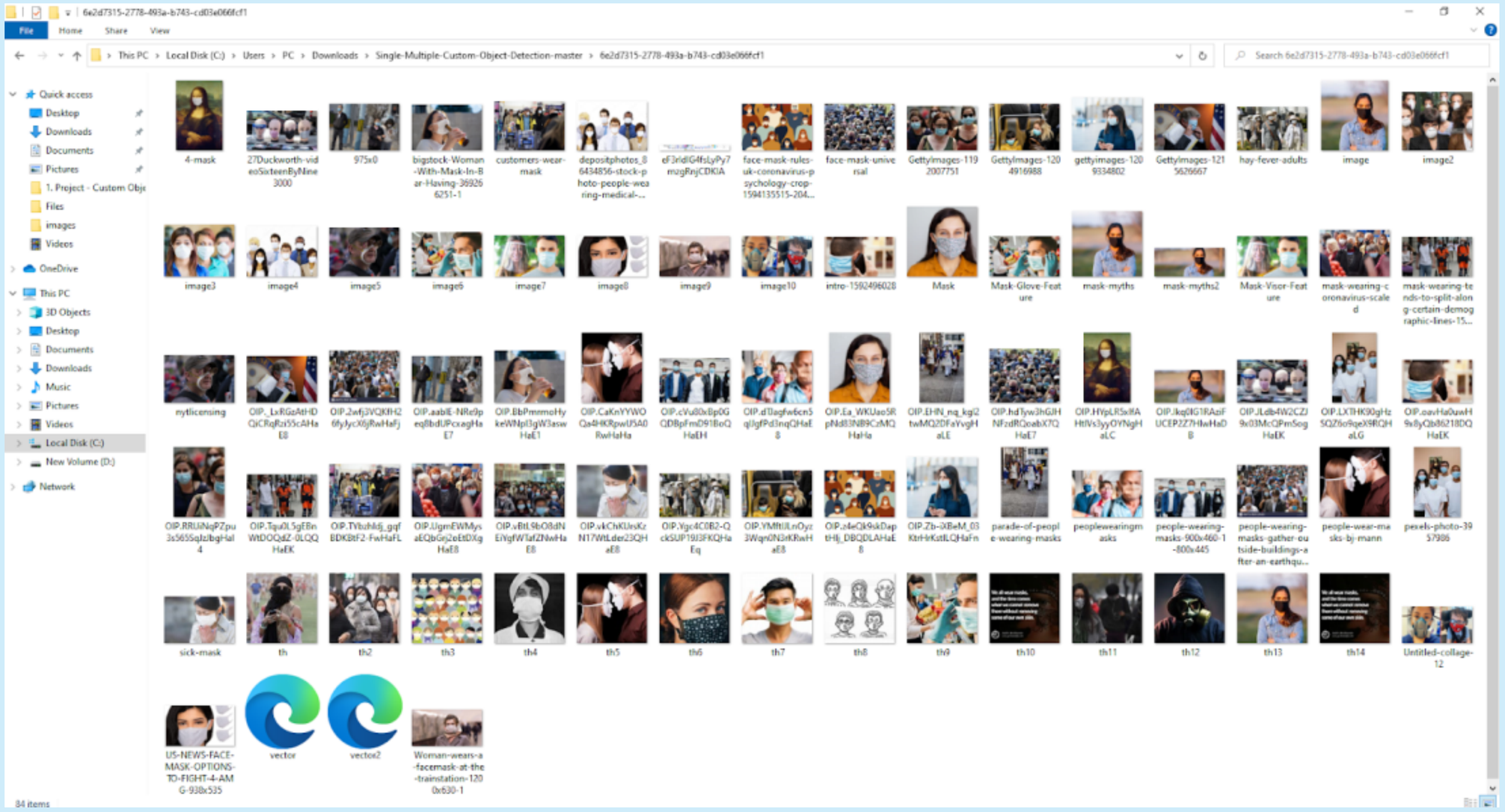


Figure 4 Screenshot of Downloaded Images Used in Training the Classifierr

Figure 5 Manually Setting Bounding Boxes and Labeling the Dataset

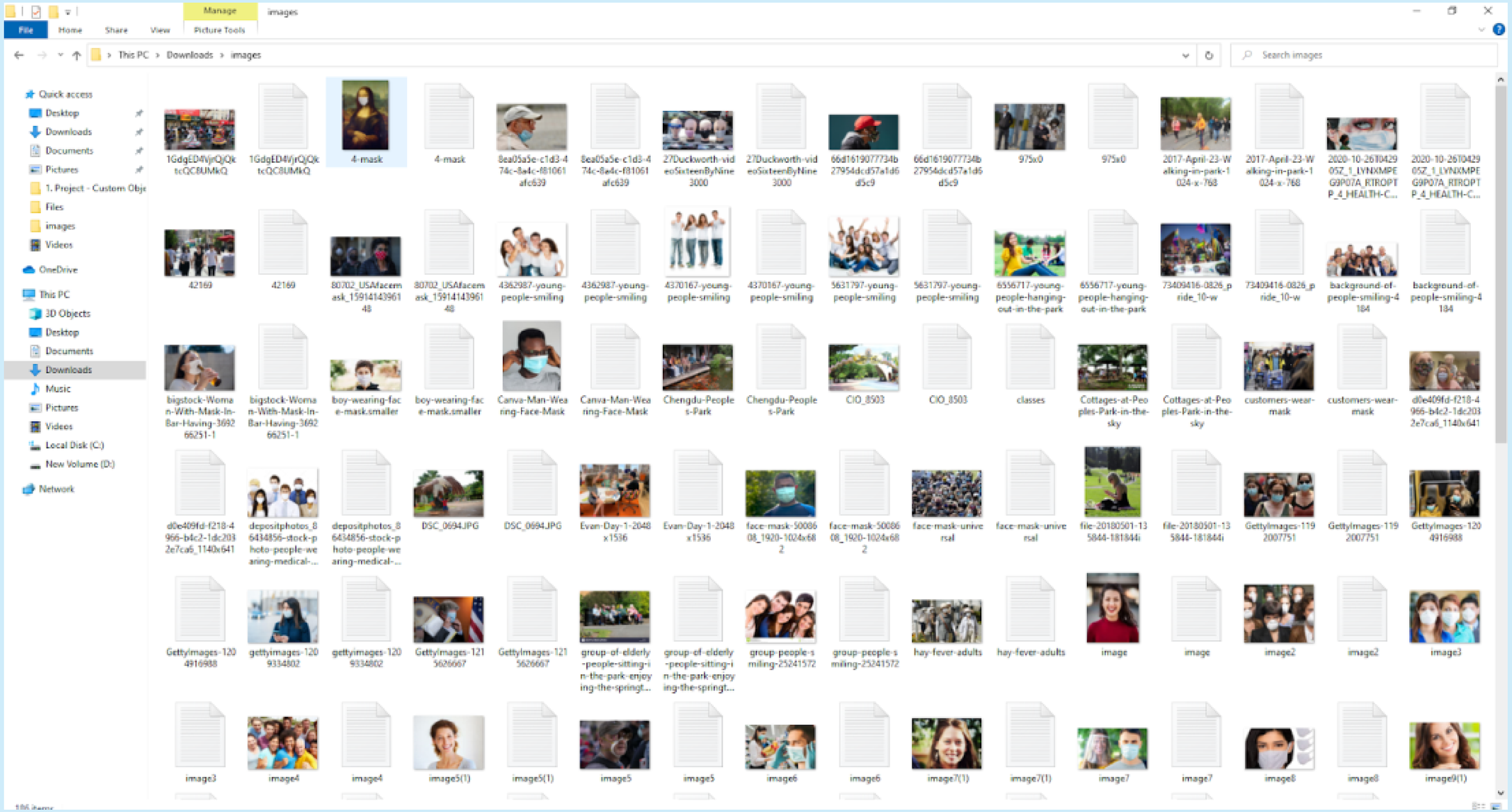


Figure 6 Screenshot of Images after Going through Pre-processing

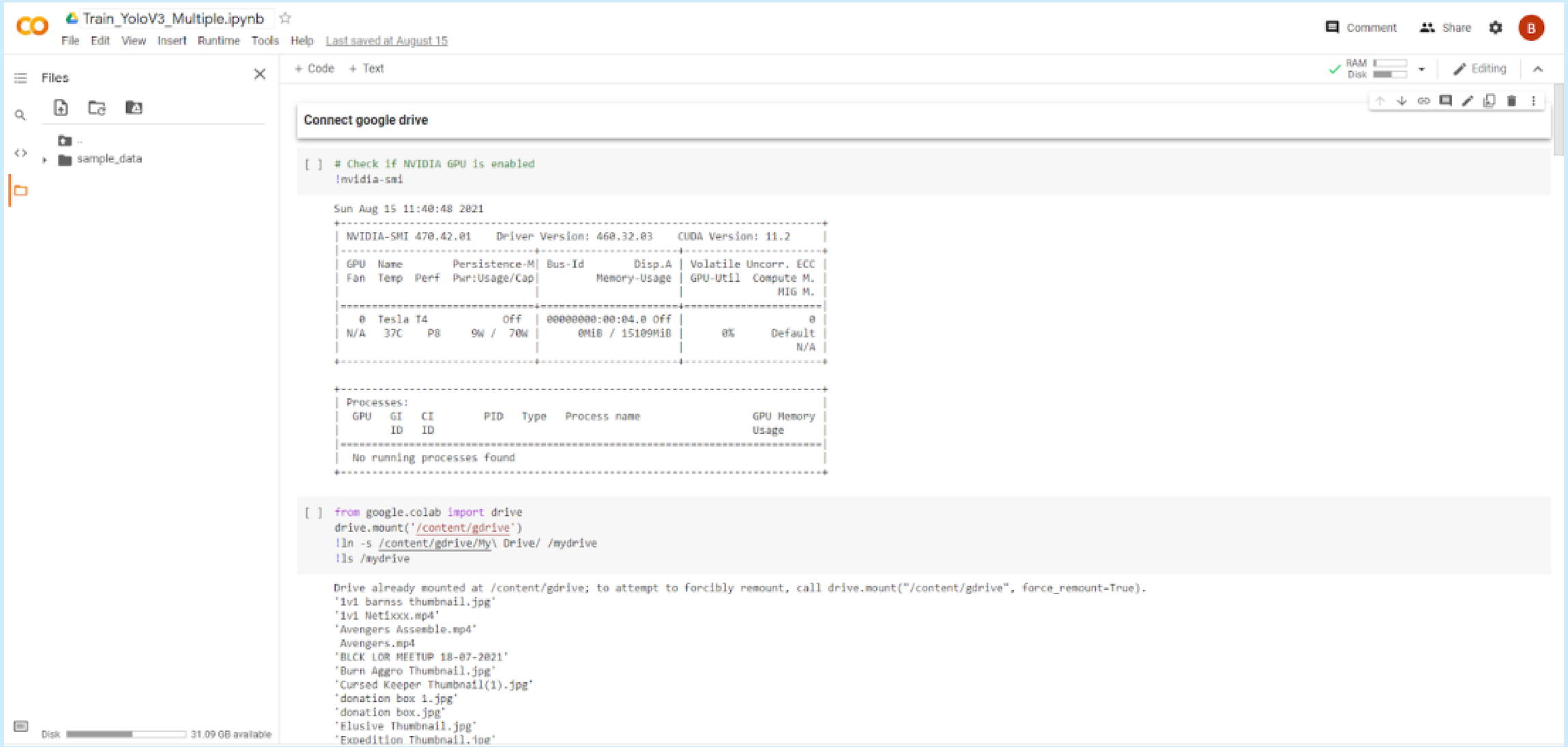


Figure 8 Checking the Availability of the GPU

1) Clone, configure & compile Darknet

```
[ ]  # Clone
     !git clone https://github.com/AlexeyAB/darknet
```

```
fatal: destination path 'darknet' already exists and is not an empty directory.
```

```
[ ]  # Configure
     %cd darknet
     !sed -i 's/OPENCV=0/OPENCV=1/' Makefile
     !sed -i 's/GPU=0/GPU=1/' Makefile
     !sed -i 's/CUDNN=0/CUDNN=1/' Makefile
     !make
```

```
     # Compile
     !make
```

2) Configure yolov3.cfg file

```
[ ]  # Make a copy of yolov3.cfg
     !cp cfg/yolov3.cfg cfg/yolov3_training.cfg
```

```
[ ]  # Change lines in yolov3.cfg file
     !sed -i 's/batch=1/batch=64/' cfg/yolov3_training.cfg
     !sed -i 's/subdivisions=1/subdivisions=16/' cfg/yolov3_training.cfg
     !sed -i 's/max_batches = 500200/max_batches = 4000/' cfg/yolov3_training.cfg
     !sed -i '610 s@classes=80@classes=2@' cfg/yolov3_training.cfg
     !sed -i '696 s@classes=80@classes=2@' cfg/yolov3_training.cfg
     !sed -i '783 s@classes=80@classes=2@' cfg/yolov3_training.cfg
     !sed -i '603 s@filters=255@filters=21@' cfg/yolov3_training.cfg
     !sed -i '689 s@filters=255@filters=21@' cfg/yolov3_training.cfg
     !sed -i '776 s@filters=255@filters=21@' cfg/yolov3_training.cfg
```

Figure 9 Darknet Integration to Enable Object Detection using Neural Networks

3) Create .names and .data files

```
[ ]  !echo -e 'Wearing Mask\nNot Wearing Mask' > data/obj.names
     !echo -e 'classes= 2\ntrain  = data/train.txt\nvalid  = data/test.txt\nnames = data/obj.names\nbackup = /mydrive/yolov3' > data/obj.data
```

Figure 10 Creation of .names and .data to Store Classes and Objects

4) Save yolov3_training.cfg and obj.names files in Google drive

```
[ ]  !cp cfg/yolov3_training.cfg /mydrive/yolov3/yolov3_testing.cfg
     !cp data/obj.names /mydrive/yolov3/classes.txt
```

Figure 11 Saving of YOLOv3 Configuration File

5) Create a folder and unzip image dataset

```
!mkdir data/obj
!unzip /mydrive/yolov3/images.zip -d data/obj
```

Figure 12 Creating a Folder to Access the Dataset

6) Create train.txt file

```
[ ] import glob
    images_list = glob.glob("data/obj/*.jpg")
    with open("data/train.txt", "w") as f:
        f.write("\n".join(images_list))
```

7) Download pre-trained weights for the convolutional layers file

```
[ ] !wget https://pjreddie.com/media/files/darknet53.conv.74
```

```
--2021-08-15 11:44:32--  https://pjreddie.com/media/files/darknet53.conv.74
Resolving pjreddie.com (pjreddie.com)... 128.208.4.108
Connecting to pjreddie.com (pjreddie.com)|128.208.4.108|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 162482580 (155M) [application/octet-stream]
Saving to: ‘darknet53.conv.74.2’

darknet53.conv.74.2 100%[===================>] 154.96M  67.2MB/s    in 2.3s

2021-08-15 11:44:34 (67.2 MB/s) - ‘darknet53.conv.74.2’ saved [162482580/162482580]
```

8) Start training

```
[ ] !./darknet detector train data/obj.data cfg/yolov3_training.cfg darknet53.conv.74 -dont_show
    # Uncomment below and comment above to re-start your training from last saved weights
    #!./darknet detector train data/obj.data cfg/yolov3_training.cfg /mydrive/yolov3/yolov3_training_last.weights -dont_show
```

Figure 13 Screenshot of Code Implementing Data Training and Testing

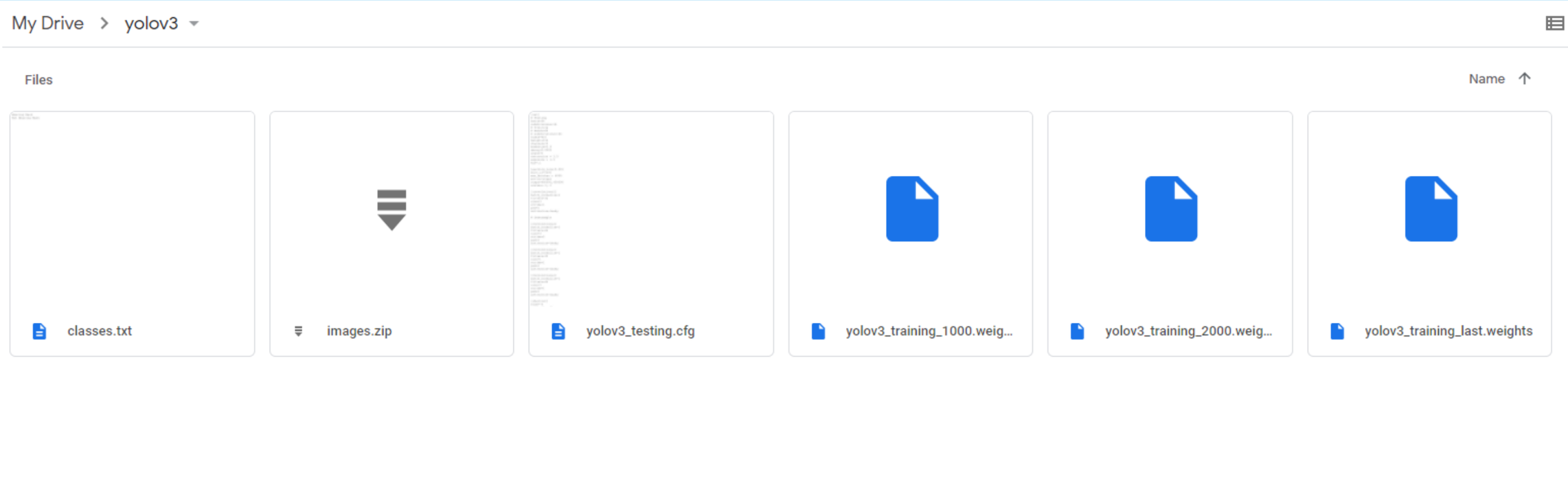


Figure 14 'yolov3' Google Drive Folder after Google Colab and Darknet

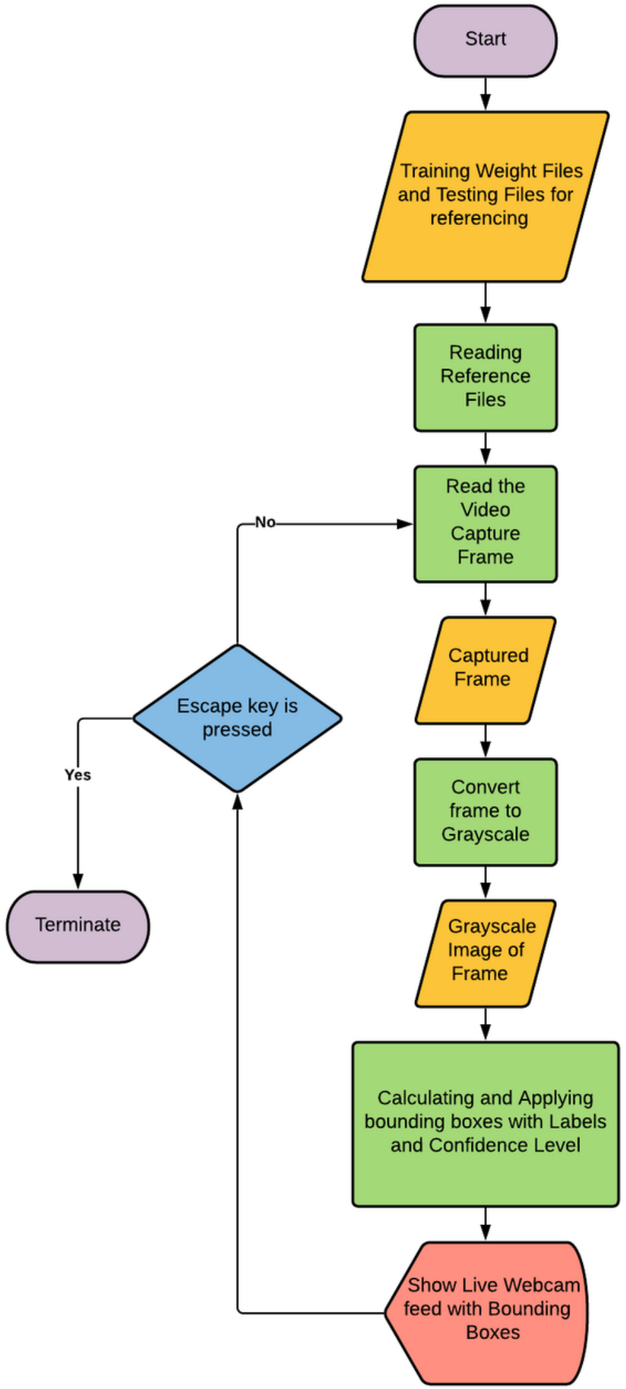


Figure 15 Program Flowchart

Results

Findings and Discussion

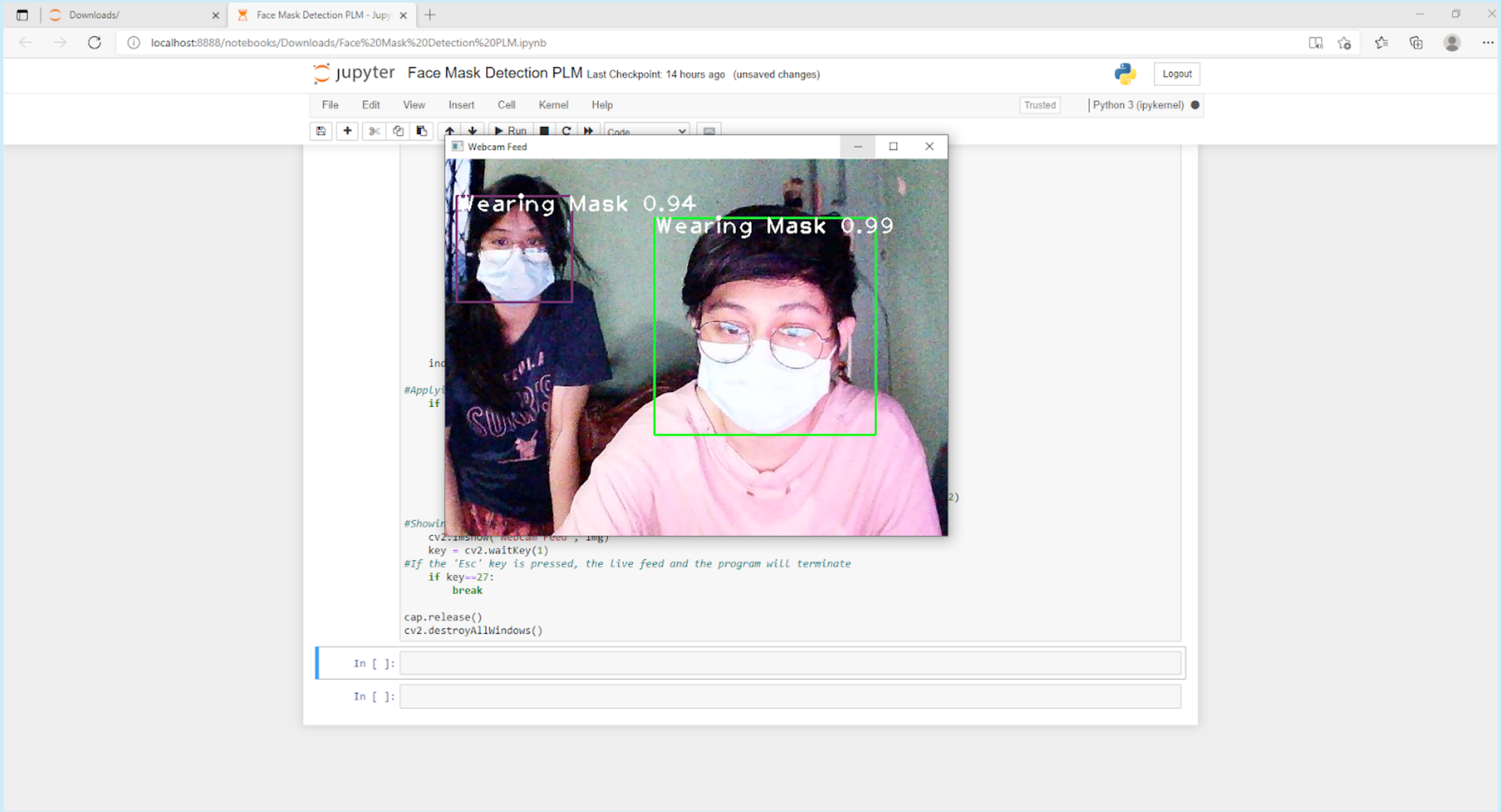


Figure 17 Output: 2 People in a Frame – Both Wearing Face Masks

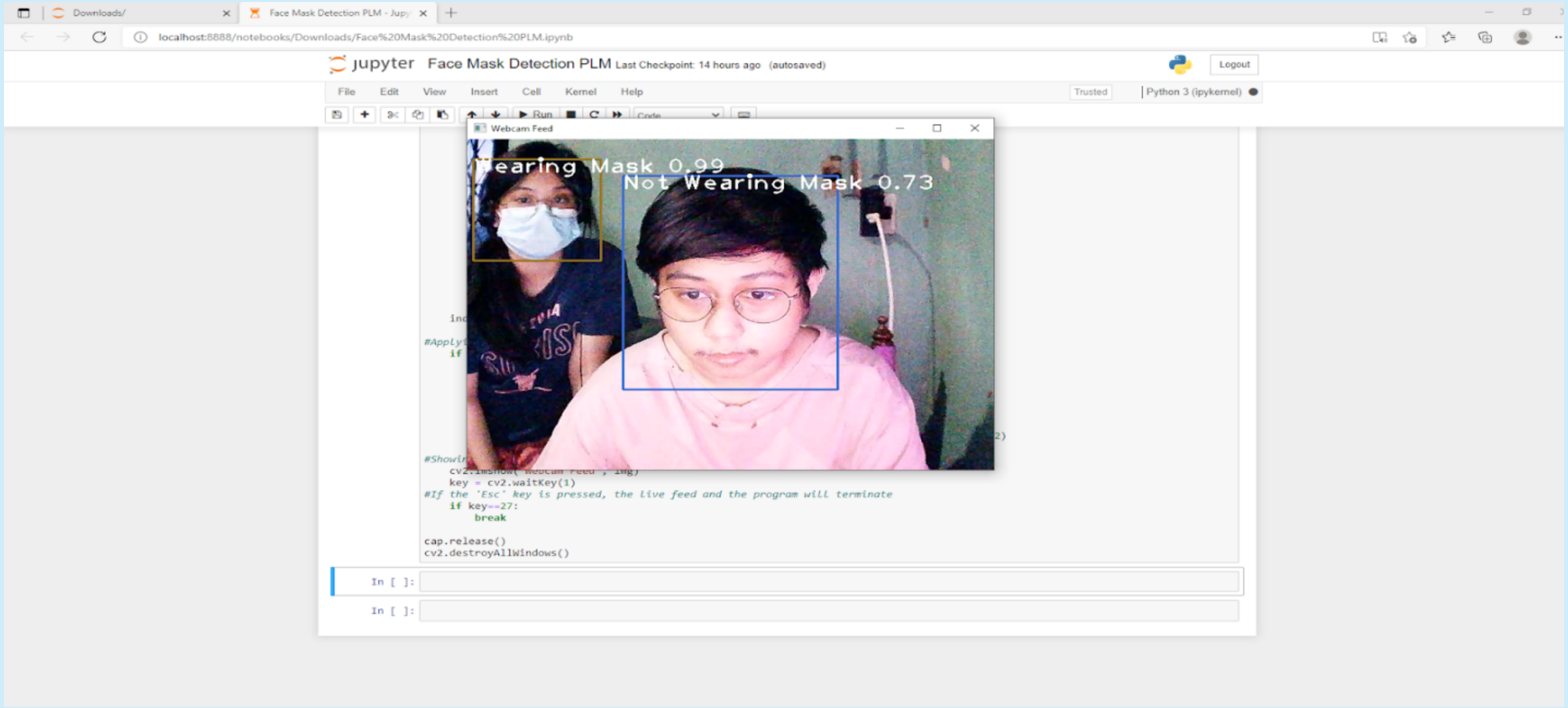


Figure 18 Output: 2 People in a Frame – Person Farther from the Webcam Wearing Face Mask

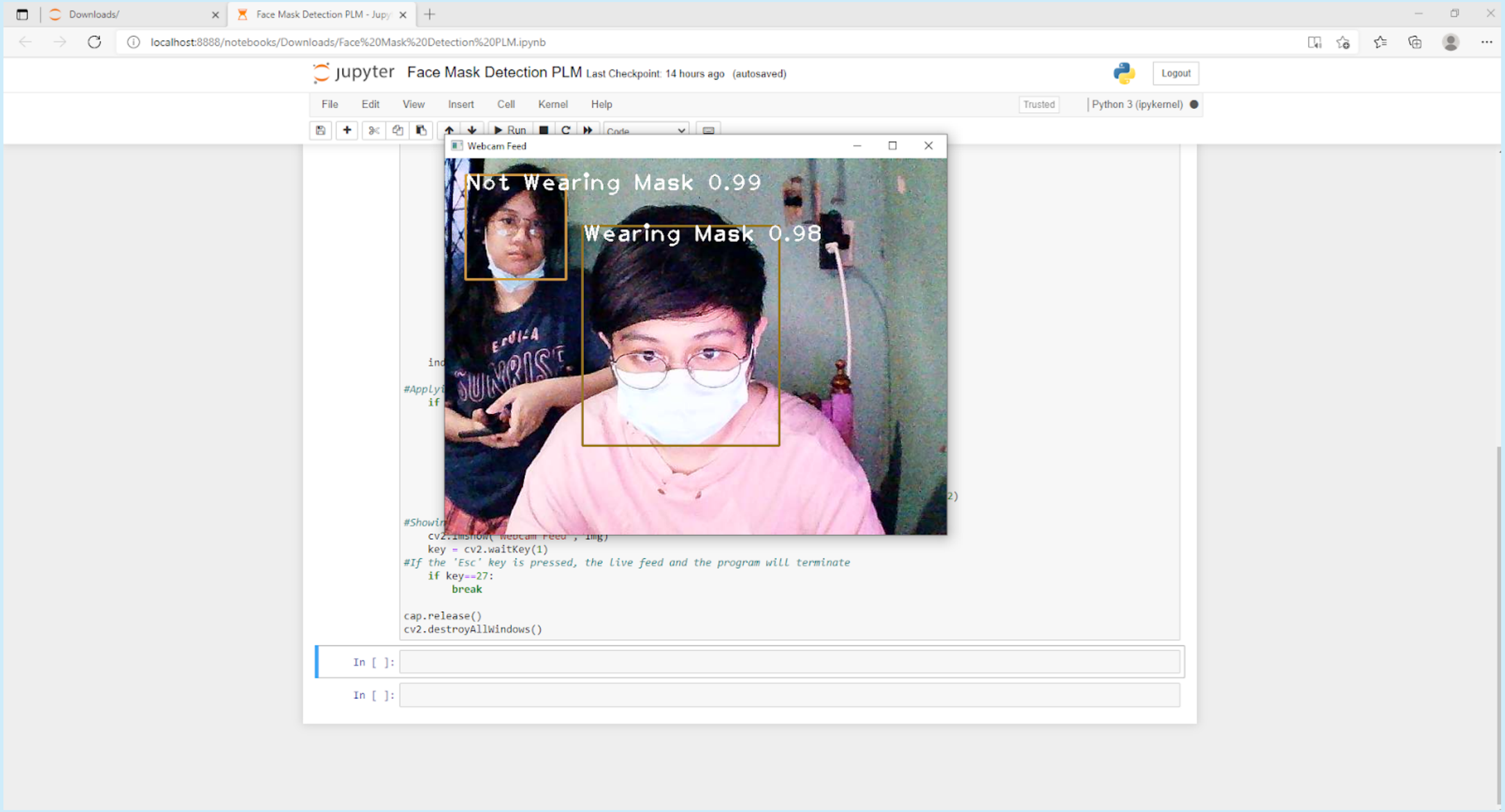


Figure 19 Output: 2 People in a Frame – Person Nearer from the Webcam Wearing Face Mask

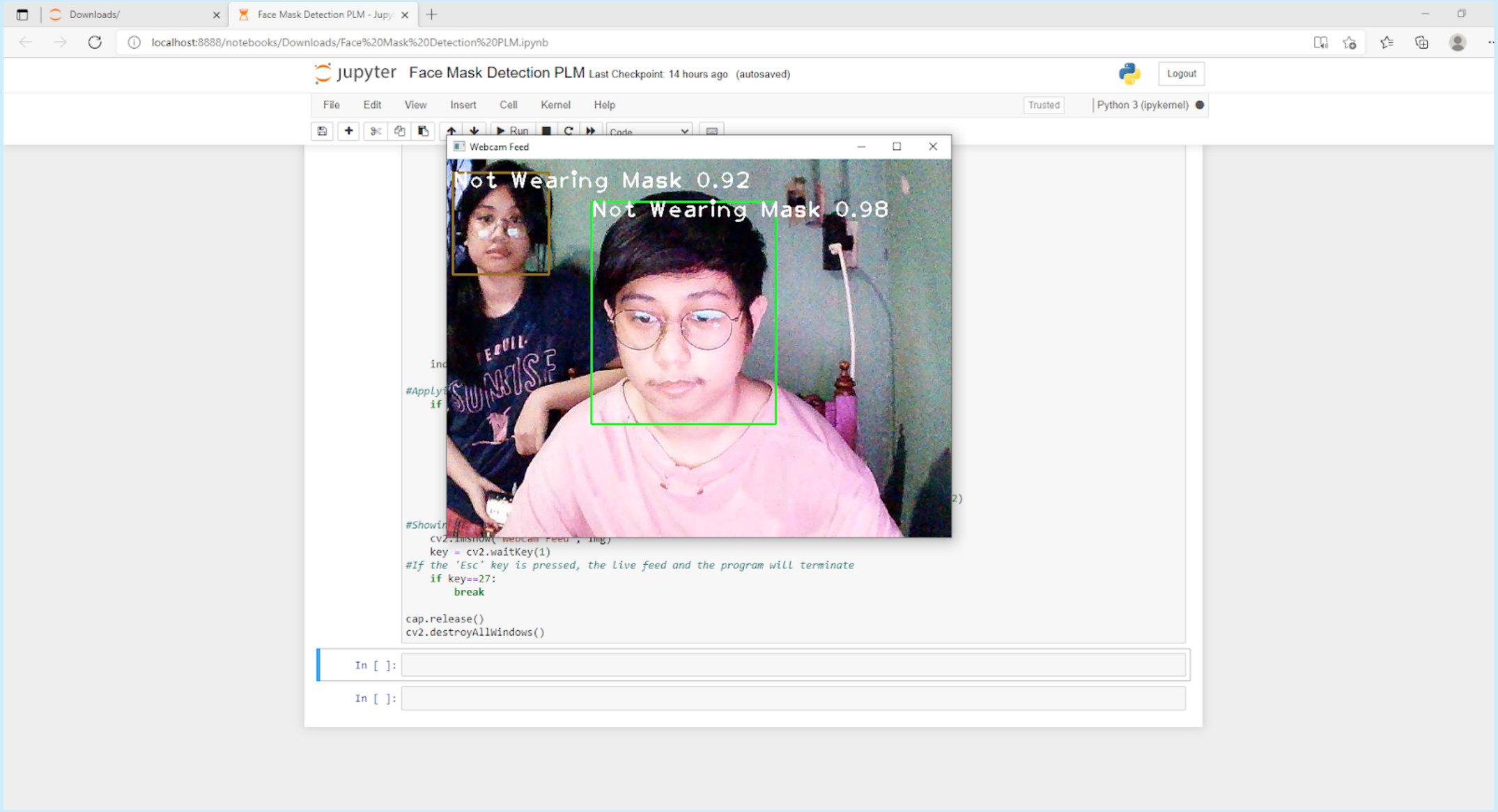


Figure 20 Output: 2 People in a Frame – Both Not Wearing Face Masks

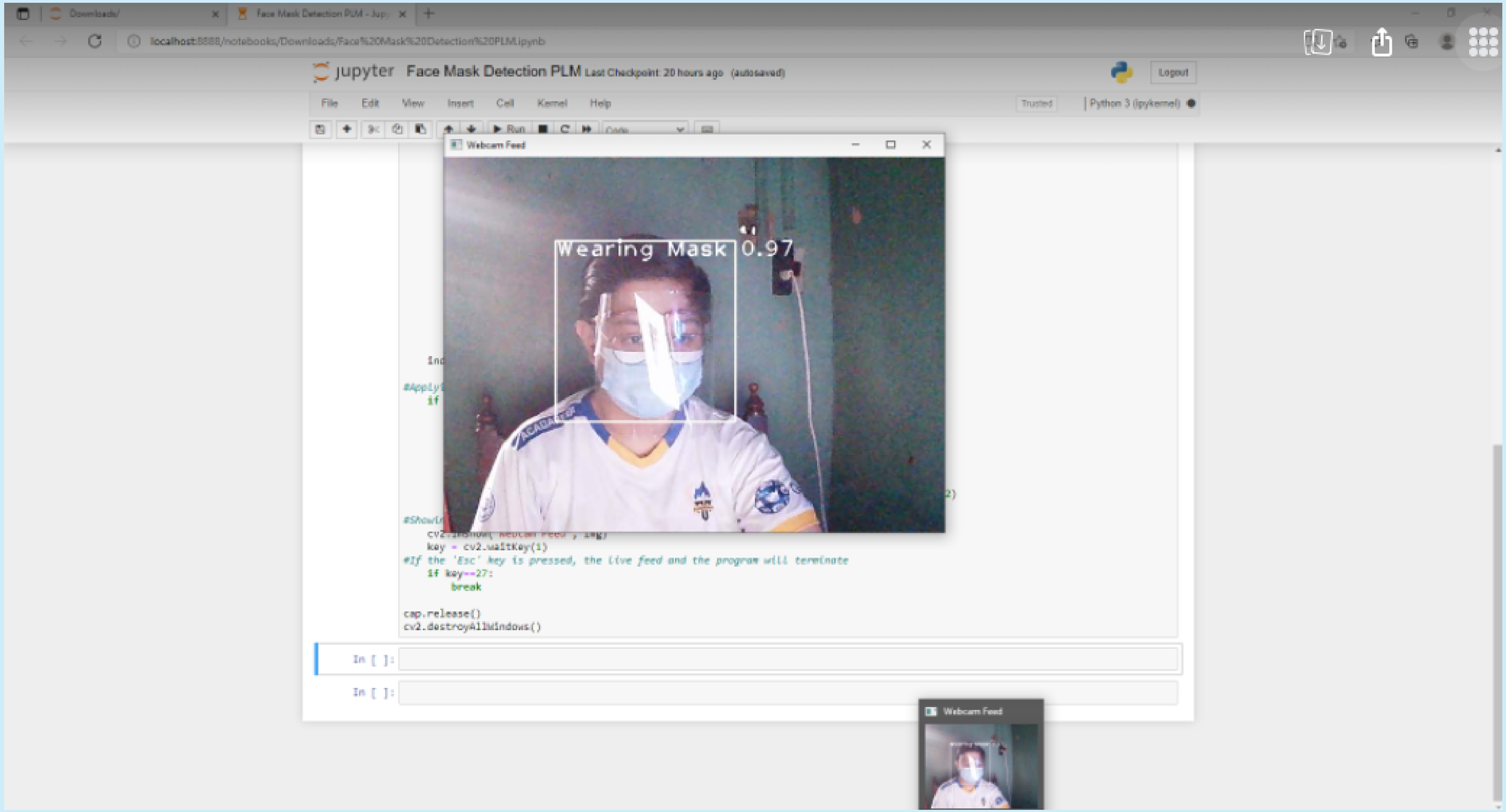


Figure 21 Output: Wearing a Face Shield with Mask

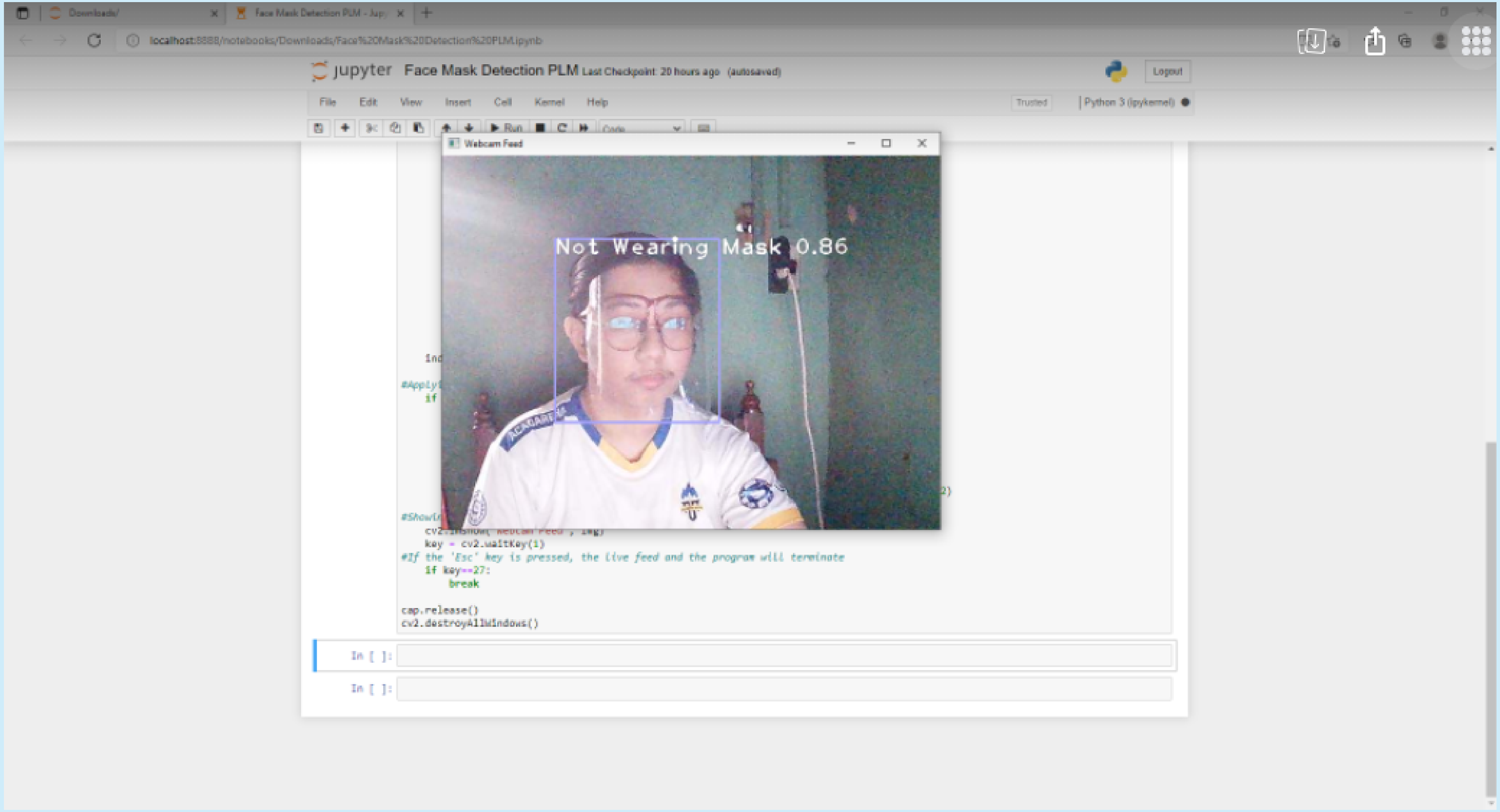


Figure 22 Output: Wearing a Face Shield without Mask

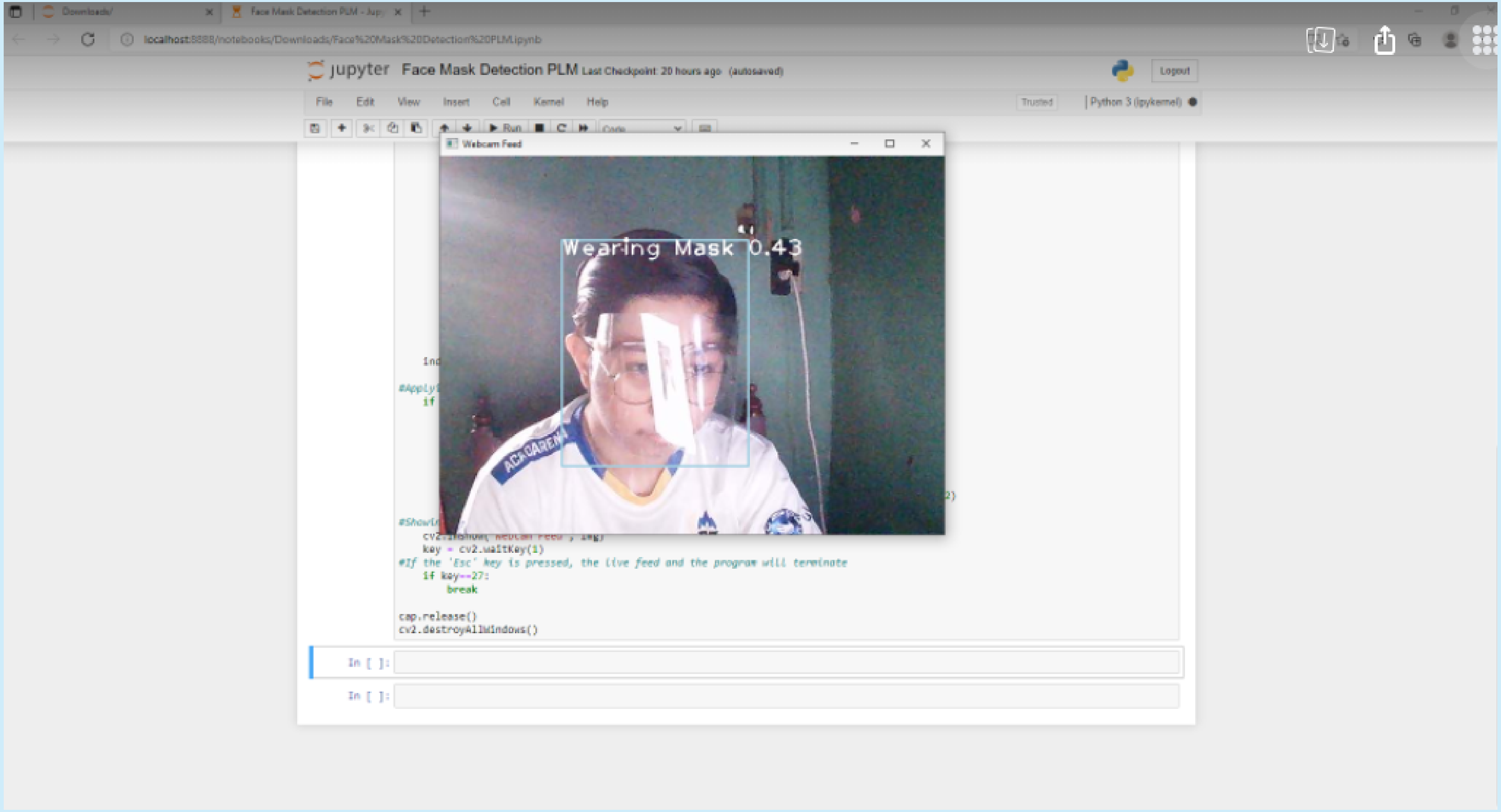


Figure 23 Output: Wearing a Face Shield showing Reflections

Figure 24 Output – Still Image: 3 People Walking in the Street

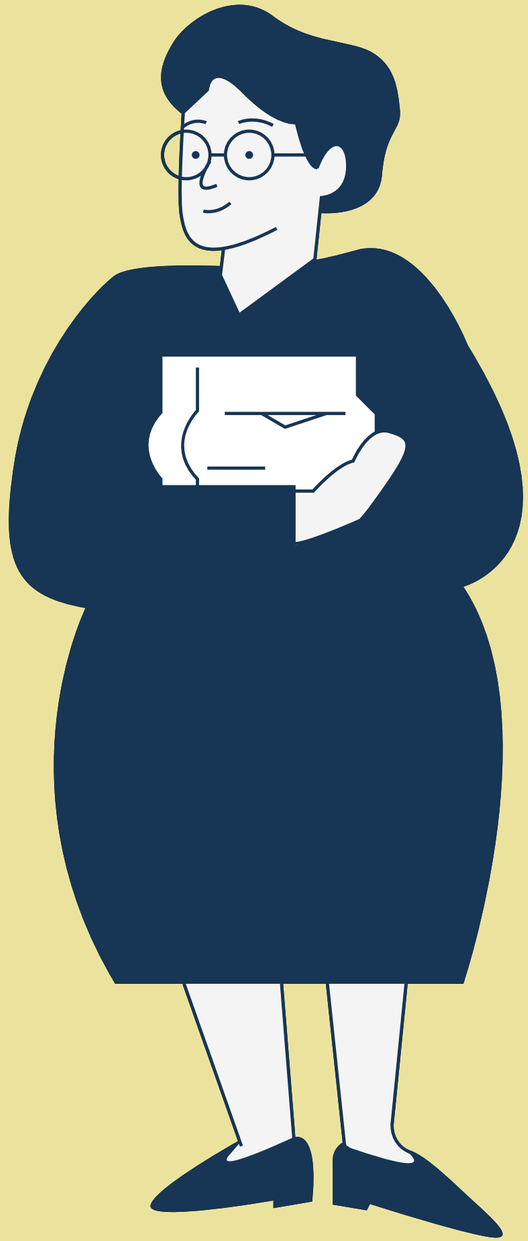
Figure 25 Output – Still Image: 20 People in a Gathering

Figure 26 Output – Still Image: Overlapping People Crossing the Pedestrian

Real Time Monitoring	
Figures	Confidence Level
Figure 17	0.94 – Person farther from the camera 0.99 – Person nearer the camera
Figure 18	0.99 – Person farther from the camera 0.73 – Person nearer the camera
Figure 19	0.99 – Person farther from the camera 0.98 – Person nearer the camera
Figure 20	0.92 – Person farther from the camera 0.98 – Person nearer the camera
Figure 21	0.97 – Person wearing face shield with mask
Figure 22	0.86 – Person wearing face shield without mask
Figure 23	0.43 – Person wearing face shield with reflections

Still Image	
Figures	Confidence Level
Figure 24	1.00 – Person in the left side of the image 1.00 – Person in the middle of the image 1.00 – Person in the right of the image
Figure 25	0.72 to 1.00 – 20 People in the image wearing a mask
Figure 26	0.5 to 0.80 – Overlapping people crossing a pedestrian; assorted

Conclusion



Using YOLO Algorithm, the researchers have been able to develop a system that can determine and classify real-time if users are wearing a mask or not.

Confidence level calculated by the program ranging from 0.0 to 1.0.

It works even with the presence of other people or groups with the confidence level not reaching less than 0.70

It also works on still images and the program were able to detect and classify as high as 20 with a confidence level between 0.73 to 1.0

The presence of faceshield with facemask can be still classified and identified, but it resulted to a lower confidence level, as low as 0.36

The researchers recommend finding a way to lessen the dependency of the algorithm on the GPU through code optimization techniques.

More detailed analyses can be integrated such as facial recognition and identification, social distancing of people from each other, and integrating face shields into the training model.

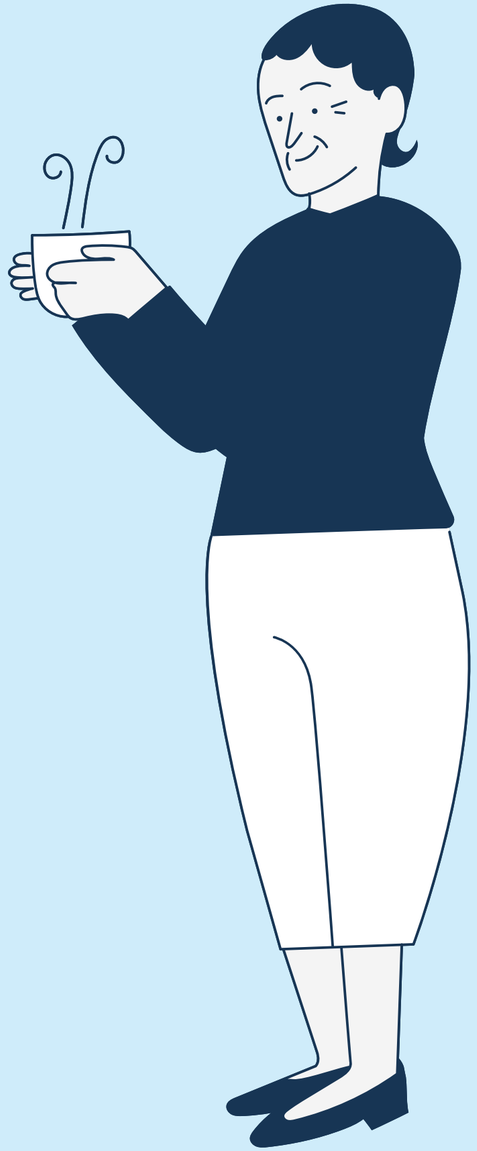
Recommendation

The researchers recommend finding a way to lessen the dependency of the algorithm on the GPU through code optimization techniques.

More detailed analyses can be integrated such as facial recognition and identification, social distancing of people from each other, and integrating face shields into the training model.

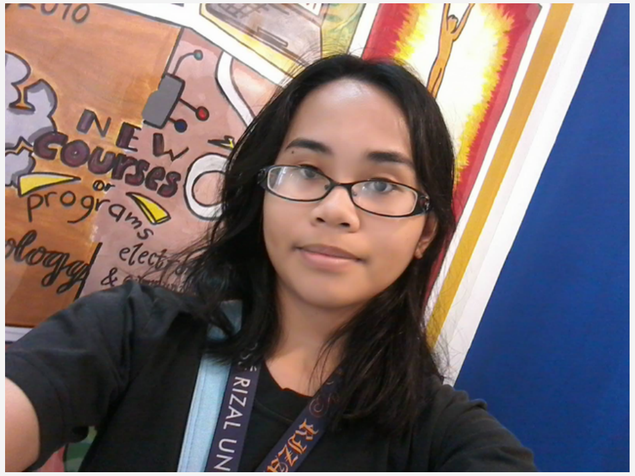
Demonstration

Thank You!



Members

Brix Relano



Kaye Calungsod

Marc Cadiz

Members

John Benedict Malabanan

Jewel Catchuela