SWE3999 - Technical Answers to Real World Problems (TARP)
Project Report

Face Mask Detection and Social Distancing

By

17MIS1016	Yaswanth K
17MIS1044	PragnaPulipati
17MIS1089	Sai Monalisa K
17MIS1137	Harshini G
17MIS1179	ManojKarthik
17MIS1192	LaxmiShivani N
17MIS1194	Sruthi S

M.Tech Software Engineering
(5 Year Integrated)

Submitted to

Prof. Jayaram B

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

November 2020

DECLARATION

I hereby declare that the report titled "**Face Mask Detection and Social Distancing"** submitted by me to VIT Chennai is a record of bona-fide work undertaken by me under the supervision of **Prof.Jayaram B**, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

Signature of the Candidate

Yaswanth K
17MIS1016

PragnaPulipati
17MIS1044

Sai Monalisa K
17MIS1089

Harshini G
17MIS1137

ManojKarthik
17MIS1179

LaxmiShivani N
17MIS1192

Sruthi S
17MIS1194

CERTIFICATE

Certified that this project report entitled "**Face Mask Detection and Social Distancing"** is a bonafide work of **Yaswanth K (17MIS1016), PragnaPulipati (17MIS1044), Sai Monalisa K (17MIS1089), Harshini G (17MIS1137), ManojKarthik (17MIS1179), LaxmiShivani N (17MIS1192), Sruthi S (17MIS1194)** and they carried out the Project work under my supervision and guidance for SWE3999 - Technical Answers to Real World Problems (TARP).

Prof. Jayaram B
SCOPE, VIT Chennai

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof. Jayaram B**, School of Computer Science and Engineering for his/her consistent encouragement and valuable guidance offered to us throughout the course of the project work.

We are extremely grateful to **Dr. R. Jagadeesh Kannan**, Dean and **Dr. S. Geetha**, Associate Dean, School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, for extending the facilities of the School towards our project and for their unstinting support.

We express our thanks to **Dr. AsnathVictyPhamila,** Head of the Department, M.Tech. Software Engineering (5 year Integrated) for her support throughout the course of this project.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

Yaswanth K	**17MIS1016**
PragnaPulipati	**17MIS1044**
Sai Monalisa K	**17MIS1089**
Harshini G	**17MIS1137**
ManojKarthik	**17MIS1179**
LaxmiShivani N	**17MIS1192**
Sruthi S	**17MIS1194**

ABSTRACT

The end of 2019 witnessed the outbreak of Coronavirus Disease 2019 (COVID-19), which has continued to be the cause of plight for millions of lives and businesses even in 2020. As the world recovers from the pandemic and plans to return to a state of normalcy, there is a wave of anxiety among all individuals, especially those who intend to resume in-person activity. Studies have proved that wearing a face mask and maintaining social distance significantly reduces the risk of viral transmission as well as provides a sense of protection. However, it is not feasible to manually track the implementation of this policy. Technology holds the key here. We introduce a Deep Learning based system that can detect instances where face masks are not used properly. Our system consists of a dual-stage Convolutional Neural Network (CNN) architecture capable of detecting masked and unmasked faces and can be integrated with real time cameras. This will help track safety violations, promote the use of face masks, and ensure a safe working environment. Social distancing is a method to minimize crowd interactions and prevent the spread of disease within groups of people. This is a common practice which has been carried out over generations to minimize the spread of virus by limiting its reproduction rate (R0) among communities.The proposed framework utilizes the YOLO v3 object detection model to segregate humans from the background and Deepsort approach to track the identified people with the help of bounding boxes and assigned IDs.

CONTENTS

1. INTRODUCTION

1.1 OBJECTIVE AND GOAL OF THE PROJECT

COVID-19 belongs to the family of coronavirus caused diseases, initially reported at Wuhan, China, during late December 2019.Several healthcare organizations, medical experts and scientists are trying to develop proper medicines and vaccines for this deadly virus, but till date, no success is reported. This situation forces the global community to look for alternate ways to stop the spread of this infectious virus. Social distancing is claimed as the best spread stopper in the present scenario, and all affected countries are locked-down to implement social distancing. Face detection has emerged as a very interesting problem in image processing and computer vision. It has a range of applications from facial motion capture to face recognition which at the start needs the face to be detected with a very good accuracy. Face detection is more relevant today because it is not only used on images but also in video applications like real time surveillance and face detection in videos. This project is aimed to support and mitigate the coronavirus pandemic along with minimum loss of economic endeavours, and propose a solution to determine if a person is wearing a face mask and detect the social distancing among people gathered at any public place.

1.2 PROBLEM STATEMENT

The word social distancing is best practice in the direction of efforts through a variety of means, aiming to minimize or interrupt the transmission of COVID-19. It aims at reducing the physical contact between possibly infected individuals and healthy persons. As per the WHO norms it is prescribed that people should maintain at least 6 feet of distance among each other in order to follow social distancing.

A recent study indicates that social distancing is an important containment measure and essential to prevent COVID-19, because people with mild or no symptoms may fortuitously carry corona infection and can infect others.

Wearing a face mask will help prevent the spread of infection and prevent the individual from contracting any airborne infectious germs and so many people are ignorant of this information. A survey revealed that only 50 per cent respondents wear a mask during the entire duration while out of home; Around 30 per cent put on the mask only when someone is in close vicinity; Over 73 per cent respondents ensure their masks cover mouth and nose when stepping out; and that users belief handkerchief and face shield offered highest protection with regards to coverage of ears, mouth and nose.

1.3MOTIVATION

Social distancing stops the spread ofdisease.Social distancing puts space between individuals. If someone is sick and there are no people around, a virus cannot spread.we need a framework to automate the process of monitoring the social distancing via object detection and tracking approaches. Using this, organisations can screen if their workers are wearing masks during their work day or not. In the event that anyone is found without a mask, they will get a warning with a suggestion to wear a face mask. This system can therefore be used in real-time applications which require face-mask detection for safety purposes due to the outbreak of covid. This project can be integrated with embedded systems for application in airports, railway stations, offices, schools, and public places to ensure that public safety guidelines are followed.

1.4CHALLENGES

For detecting faces, the algorithms take cues from the landmarks around eyes, nose and mouth. Now since the bottom half of the facial features are hidden by the masks, facial recognition technology is facing new challenges. Size of the images, varying angles and lack of clarity of the images in the dataset were also a hindrance. When it comes to social distancing, Higher numbers of false positives may raise discomfort and panic situations among people being observed. There may also be genuinely raised concerns about privacy and individual rights which can be addressed with some additional measures such as prior consents for such working environments, hiding a person’s identity in general, and maintaining transparency about its fair uses within limited stakeholders.

2. LITERATURE SURVEY

S.no	Title	Methodology	Observation
1	Li T, Liu Y, Li M, Qian X, Dai SY (2020) Mask or no mask for COVID-19: A public health and market study. PLOS ONE 15(8): e0237691.	Investigating three factors that might influence the effectiveness of mask use and the COVID-19 transmission rate, including the mask aerosol reduction rate, mask availability, and mask population coverage.	In combination with social distancing and other measures, wearing a mask promises to replace shelter-in-place orders and significantly reduce the burden on society for COVID-19.
2	A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic.	The introduced model includes two main components, the first component is deep transferring learning (ResNet50) as feature extractor and the second component is a classical machine learning like decision trees, SVM, and ensemble. According to [20,21], ResNet-50 has achieved better results when it is used as a feature extractor	The novelty of this research is using a proposed feature extraction model that has an end-to-end structure without traditional techniques with three classifiers machine learning algorithms for mask face detection. The SVM classifier achieved the highest accuracy possible with the least time consumed in the training process. The SVM classifier in RMFD achieved 99.64% testing accuracy. In SMFD, it gained 99.49%, while in LFW, it reached 100% testing accuracy.
3	Monitoring COVID-19 social distancing with	The emergence of deep learning has brought the best performing techniques	An efficient real-time deep learning based framework to

	person detection and tracking via fine-tuned YOLO v3 and Deepsort techniques	for a wide variety of tasks and challenges including medical diagnosis, machine translation, speech recognition , and a lot more. Most of these tasks are centredaround object classification, detection, segmentation, tracking, and recognition. In recent years, the convolution neural network (CNN) based architectures have shown significant performance improvements that are leading towards the high quality of object detection.	automate the process of monitoring the social distancing via object detection and tracking approaches, where each individual is identified in the real-time with the help of bounding boxes. The generated bounding boxes aid in identifying the clusters or groups of people satisfying the closeness property computed with the help of pairwise vectorized approach.
4	Real-Time social Distancing Detector Using Social distancing net-19 Deep Learning Network	In this methodology ,load 295 images from the dataset, where each image had single or multiple labels inside it which were used for training the model. Further, more images and labels were generated using an auxiliary dataset. First, the image file path and the second is the corresponding label.We used the Social distancing Net-19 architecture for the training purpose. Box labels were used to create the data for training and evaluation purposes.	This work distinguishes the social distancing pattern and classifies them as a violation of social distancing or maintaining the social distancing norm.The classifier was then implemented for live video streams and images also. This system can be used in CCTV for surveillance of people during pandemics. Mass screening is possible and hence can be used in crowded places like railway stations, bus stops, markets, streets, mall entrances, schools, colleges, etc

5	Social distancing detection system with artificial intelligence using computer vision and deep learning	The proposed system focuses on how to identify the person on image/video stream whether the social distancing is maintained or not with the help of computer vision and deep learning algorithm by using theOpenCV,Tensorflow library. As the input video may be taken from an arbitrary perspective view,the first step is to transform perspective of view to a bird's-eye (top-down).	The model proposes an efficient real-time deep learning based framework to automate the process of monitoring the social distancing via object detection and tracking approaches, where each individual is identified in the real-time with the help of bounding boxes.The extensive trials were conducted with popular state-of-the-art object detection models Faster RCNN, SSD, and YOLO v3, since this approach is highly sensitive to the spatial location of the camera, the same approach can be fine tuned to better adjust with the corresponding field of view.
6	Monitoring Social Distancing for Covid-19 Using OpenCV and Deep Learning	Social Distancing would undoubtedly be the most important factor as COVID 19 spreads through close contact with infected ones. In order to supervise large mobs, an effective solution is important and this survey paper focuses on that. Using installed CCTV and drones, authorities can keep a track of human activities and control large crowds to come together and prevent violating the law.	As far as people are maintaining a safe distance they would be indicated with green light, and as the CCTV captures more and more crowd gathering, red light would pop-up and the allocated police of that area will be notified and the situation can come under control immediately [6]. As controlling a large mob is not an easy task, using this

			survey, conditions can be managed before the situation goes out of control.
7	Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID19 Safety Guidelines Adherence	The proposed system helps to ensure the safety of the people at public places by automatically monitoring them whether they maintain a safe social distance, and also by detecting whether or not an individual wears a face mask. This section briefly describes the solution architecture and how the proposed system will automatically function in an automatic manner to prevent the coronavirus spread.	The implementation of this solution was successfully tested in real-time by deploying a model in raspberry pi4. The solution has the potential to significantly reduce violations by real-time interventions, so the proposed system would improve public safety through saving time and helping to reduce the spread of coronavirus. This solution can be used in places like temples, shopping complexes, metro stations, airports, etc.

3. REQUIREMENTS SPECIFICATION

1.1 FACE MASK DETECTION

1. OpenCV - Open Source Computer Vision Library is an open-source software library for real-time computer vision and machine learning and is generally used for facial recognition and image processing.

2. Keras - Keras is an open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

3. Caffe based Face Detection Model - A pre-trained deep learning model based on the Caffe framework for face detection.

4. Jupyter Notebook - Python IDE

1.2 SOCIAL DISTANCING

1.OpenCV - Open Source Computer Vision Library is an open-source software library for real-time computer vision and machine learning and is generally used for facial recognition and image processing.

2.YOLO v3-YOLO(You Only Look Once) is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

3.anaconda3-world's most popular Python distribution platform .

4.Tensorflow-TensorFlow is a Python library for fast numerical computing created and released by Google.

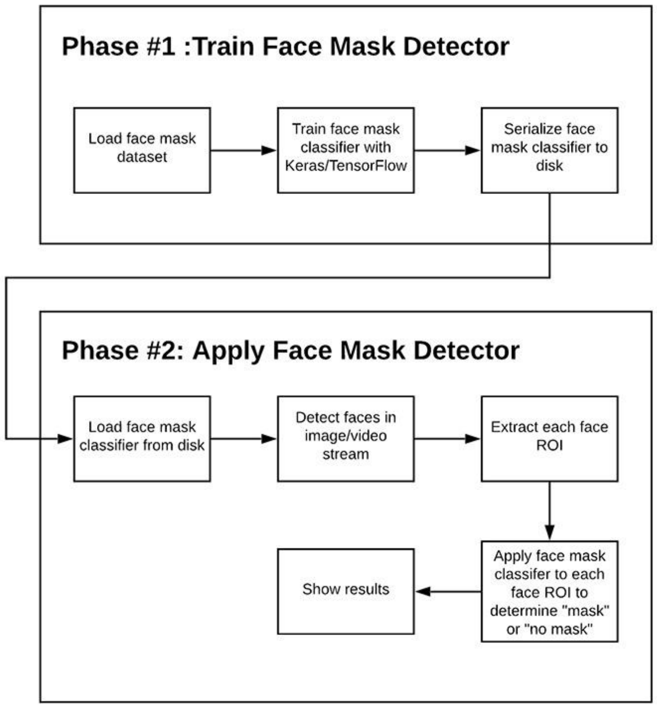
5.CMake-CMake is an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files.

6.Imutilis-A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

4. SYSTEM DESIGN

1.1 FACE MASK DETECTION

1. Detect COVID-19 face masks in images
2. Detect face masks in real-time video streams


Phase #1 :Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifer to each face ROI to determine "mask" or "no mask"
Show results

In order to train a custom face mask detector, we need to break our project into two distinct phases, each with its own respective sub-steps (as shown by the Figure above):

1. Training: Here we'll focus on loading our face mask detection dataset from disk, training a model (using Keras/TensorFlow) on this dataset, and then serializing the face mask detector to disk

2. Deployment: Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as with_mask or without_mask.

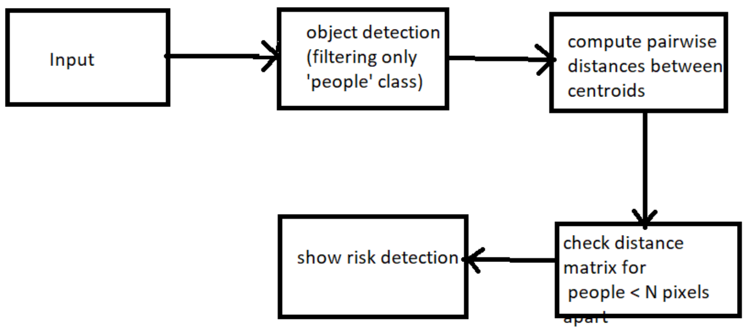
Proposed Methodology

We propose a two-stage architecture for detecting masked and unmasked faces and localizing them.

Architecture Overview :

The above Fig. represents our proposed system architecture. It consists of two major stages. The first stage of our architecture includes a Face Detector, which localizes multiple faces in images of varying sizes and detects faces even in overlapping scenarios. The detected faces (regions of interest) extracted from this stage are then batched together and passed to the second stage of our architecture, which is a CNN based Face Mask Classifier. The results from the second stage are decoded and the final output is the image with all the faces in the image correctly detected and classified as either masked or unmasked faces.

1.2 SOCIAL DISTANCING


Input
object detection (filtering only 'people' class)
compute pairwise distances between centroids
check distance matrix for people < N pixels apart
show risk detection

Architecture overview:

The above fig represents our system architecture. Its consists of 5 steps

1. First we need to give input and using the YOLO object detector to detect people in a video stream and
2. Determining the centroids for each detected person
3. Computing the pairwise distances between all centroids
4. Checking to see if any pairwise distances were < N pixels apart, and if so, indicating that the pair of people violated social distancing rules.

5 IMPLEMENTATION OF SYSTEMS

1.1 FACE MASK DETECTION

Step 1 : Data pre-processing

The dataset we use consists of images of multiple colors, varying sizes, and different orientations. Therefore, we need to be sure that colour should not be a crucial point for detecting masks. we need to convert all the images to grayscale using the function ImageDataGenerator from Keras's image preprocessing library.

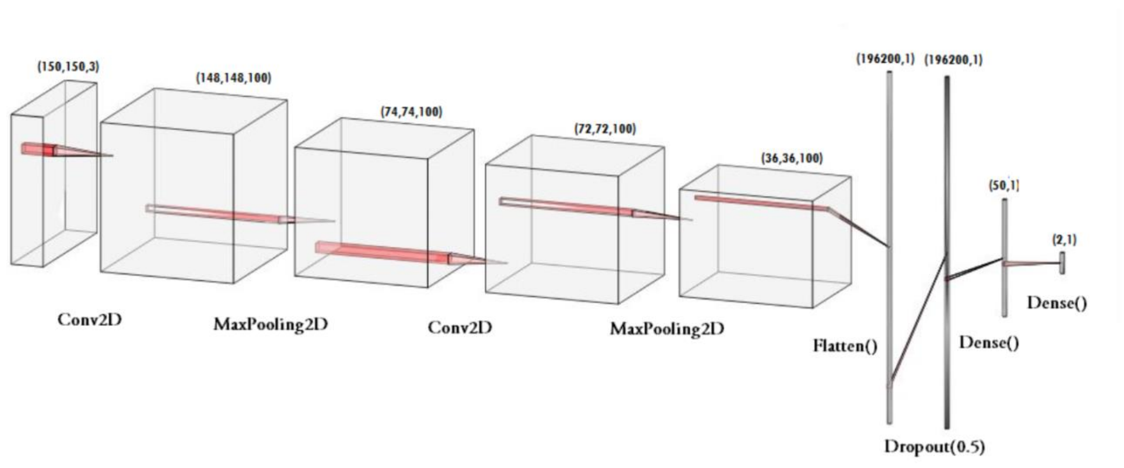
Step 2 : Data augmentation

We augment our dataset to include more number of images for our training. In this step of data augmentation, we rotate, shear, zoop and flip each of the images in our dataset.

Step 3 : Building the model

We build our Sequential CNN model with various layers such as Conv2D, MaxPooling2D, Flatten, Dropout and Dense.

In the last Dense layer, we use the ‘softmax’ function to output a vector that gives the probability of each of the two classes. We use the ‘adam’ optimizer and ‘binary_crossentropy’ as our loss function as there are only two classes.



Step 4 : Training the model

This step is the main step where we fit our images in the training set and the test set to our sequential model we built using Keras library. We have trained the model for 15 epochs. We see that after the 15th epoch, our model has an accuracy of 95.00% This implies that it is well trained without any over-fitting.

Step 5 : Testing the model on a single image

To test the model on a single image, we need to implement face detection. In this, We are using the Caffe based Face Detection Model for detecting the features of the face. This Face detection model is designed to detect the frontal face by training thousands of images. The .xml file for the same needs to be downloaded and used in detecting the face.

Step 6 : Loading the model and predicting on real-time input

In this step, we intend to use the model to detect if we are wearing a face mask using our PC’s webcam. We use the OpenCV library to run an infinite loop to use our web camera in which we detect the face using the Caffe based Face Detection Model .

The code webcam = cv2.VideoCapture(0) denotes the usage of webcam.
The model will predict the possibility of each of the two classes ([mask, withoutMask]). Based on which probability is higher, the label will be chosen and displayed around our faces.

Step 7 : Sending an email in case of violation

If the person detected is not wearing a mask, a warning dialog box is displayed using the tkinter library. This is determined by the probability of withoutMask> mask. In addition to the warning dialog box, an email is sent to the admin. This can be integrated with embedded systems for application in airports, railway stations, offices, schools, and public places to ensure that public safety guidelines are followed.

1.2 SOCIAL DISTANCING

Step 1: Object detection using yolo.

We are using yolov3 object detection to detect the class "person" in the video input.
Here a dataset with millions of images is taken. It contains all types of postures of a human walking,sitting, standing etc.. Now we have used an external software called "LabelImg". We open the file directory which contains our dataset and then check if the settings is set to YOLO and then we start labelling the images in our dataset. We create a rectangular box by selecting the area that contains the human and label it as a person. Similarly we perform the operation for all the images in our dataset. After performing the operation we can see for every image a text file will be created and stored in our dataset.

Step 2: Training the dataset.

We are training the dataset containing the image and corresponding text files. Here we are using Darknet53 as a convolutional neural network. It contains 53 convolutional layers and each image iterates more than 100 times through each layer and finally the yolo weight files and configuration files for our dataset are generated. The coc.names file contains the names of the classes that are identified using object detection.

Step 3: Object detection for the real time video.

With the help of Open cv we use a dnn module and read the configuration and weight files.From the given input video by dividing it into frames checking if the class id is equal to person if and only if the class id is person then the distance will be calculated between them.

Step 4: Social distance calculation.

For every detected person a bounding box is being formed and the centroid of the detected person is calculated and the calibrated distance between the centroids is being calculated using euclidean distance and then pairwise distance between every two persons is calculated if it is less than 15% of the calibrated distance then they are declared as close, if it is less than 20% of the calibrated distance then they are declared as second close else they are declared as safe.

Step 5: Risk detection

Once the distance is calculated if it is declared as close then it returns “1” and the bounding box colour is indicated “red” , if it is declared as second close then it returns “2” and the bounding box colour is indicated “yellow” else if it specified as safe then it returns “0” and the bounding box colour is indicated as “green”.

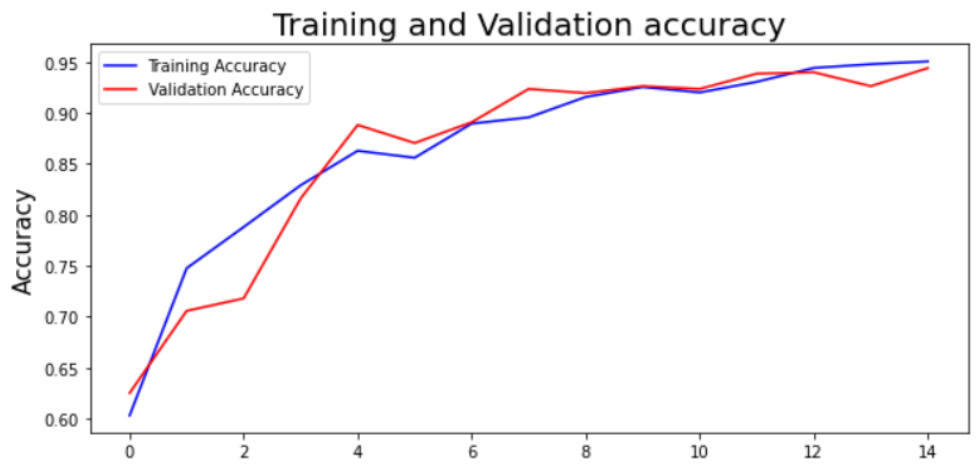
Step 6: Detecting real time video

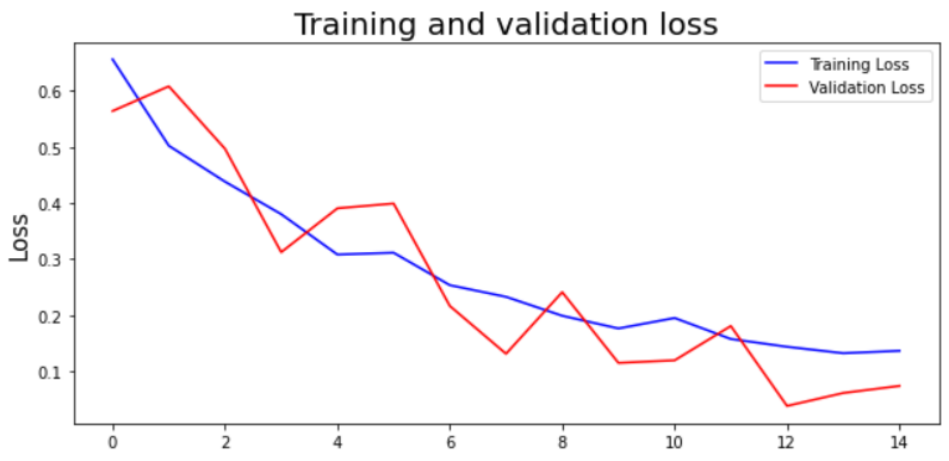
After specifying the video path the model takes the specified video as input and calculates the distance between the persons frame by frame and generates the output by changing the colours of the bounding boxes.

6. RESULTS AND DISCUSSIONS

1.1 FACE MASK DETECTION

Accuracy of the model





The face mask detector training accuracy/loss curves demonstrate high accuracy and little signs of overfitting on the data. The model worked perfectly with an accuracy score of ~95.

The threshold value and the confidence value have been set to 0.5 to acquire more precision and accuracy during the object detection phase.

1.2 SOCIAL DISTANCING

The SOCIAL DISTANCING DETECTOR model worked with a maximum precision for the 2D input videos given. The video calibration has made it easier for the distance calculation in this process.

7. CONCLUSION AND FUTURE WORK

We proposed an approach that uses computer vision and MobileNet V2 architecture to help maintain a secure environment and ensure individuals protection by automatically monitoring public places to avoid the spread of the COVID-19 virus and assist police or other communities by minimizing their physical surveillance work in containment zones and public areas where surveillance is required by means of camera feeds in real-time.
Thus, this proposed system will operate in an efficient manner in the current situation when the lockout is eased and helps to track public places easily in an automated manner. We have addressed in depth the tracking of social distancing and the identification of face masks that help to ensure human health.The solution has the potential to significantly reduce violations by real-time interventions, so the proposed system would improve public safety through saving time and helping to reduce the spread of coronavirus. This solution can be used in places like temples, shopping complex, metro stations, airports, etc

The above mentioned use cases are only some of the many features that were incorporated as part of this solution. We assume there are several other cases of usage that can be included in this solution to offer a more detailed sense of safety. Several of the currently under development features are listed below in brief:

- The use of Machine Learning in the field of mobile deployment is rising rapidly. Hence, we plan to port our models to their respective TensorFlow Lite versions.

- Our architecture can be made compatible with TensorFlowRunTime (TFRT), which will increase the inference performance on edge devices and make our models efficient on multithreading CPUs.

- The implementation of this solution could be tested in real-time by deploying the model in raspberry pi4.

When it comes to social distancing,since this project is intended to be used in any working environment; accuracy and precision are highly desired to serve the purpose. Higher numbers of false positives may raise discomfort and panic situations among people being observed. There may also be genuinely raised concerns about privacy and individual rights which can be addressed with some additional measures such as prior consents for such working environments, hiding a person's identity in general, and maintaining transparency about its fair uses within limited stakeholders.

This social distancing detector did not leverage a proper camera calibration, meaning that we could not (easily) map distances in pixels to actual measurable units (i.e., meters, feet, etc.).

Therefore, the first step to improving our social distancing detector is to utilize a proper camera calibration.Doing so will yield better results and enable us to compute actual measurable units (rather than pixels).Secondly, we should consider applying a top-down transformation of our viewing angle.

8. REFERENCES

I. Dong, C., Loy, C.C., Tang, X., Accelerating the Super-Resolution Convolutional Neural Network, in Proceedings of European Conference on Computer Vision (ECCV), 2016

II. He, K., Zhang, X., Ren, S., and Sun, J., Deep residual learning for image recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770– 778.

III. Ren, S., He, K., Girshick, R., & Sun, J., (2015), Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (pp. 91–99), MIT Press.

IV. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L., MobileNetV2: Inverted Residuals and Linear Bottlenecks, 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 4510- 4520, doi: 10.1109/CVPR.2018.00474

V. Li T, Liu Y, Li M, Qian X, Dai SY (2020) Mask or no mask for COVID-19: A public health and market study. PLOS ONE 15(8): e0237691

VI. W. H. Organization, “WHO corona-viruses (COVID-19),” https://www.who.int/emergencies/diseases/novel-corona-virus-2019, 2020.

VII. Glass RJ, Glass LM, Beyeler WE, Min HJ. Targeted social distancing architecture for pandemic influenza. Emerging Infectious Diseases. 2006;12:1671– 1681.

VIII. S. Ge, J. Li, Q. Ye and Z. Luo, "Detection of Masked Faces in the Wild with LLE-CNNs," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 426-434.

IX. https://towardsdatascience.com/monitoring-social-distancing-using-ai-c5b81da44c9f

X. https://www.analyticsvidhya.com/blog/2020/05/social-distancing-detection-tool-deep-learning/

CONTRIBUTION

Reg. no	Name	Contribution
17MIS1016	Yaswanth K	Accuracy and loss plotting, Data preprocessing and Augmentation, and Partial documentation (Face Mask Detection)
17MIS1044	PragnaPulipati	object detection, partial documentation,testing (Social distancing detection)
17MIS1089	Sai Monalisa K	Identifying problem,Yolo object detection and risk analysis and partial documentation (Social distancing detection).
17MIS1137	Harshini G	Defining Model architecture , MAsk detection in a Real time video and partial documentation (Face Mask Detection)
17MIS1179	ManojKarthik	Distance calculation,partial documentation (Social distancing detection)
17MIS1192	LaxmiShivani N	Real time video, Testing,partial documentation. (Social distancing detection)
17MIS1194	Sruthi S	Training and Validation, Dataset Visualisation, Mask detection in an image and Partial documentation (Face Mask Detection)

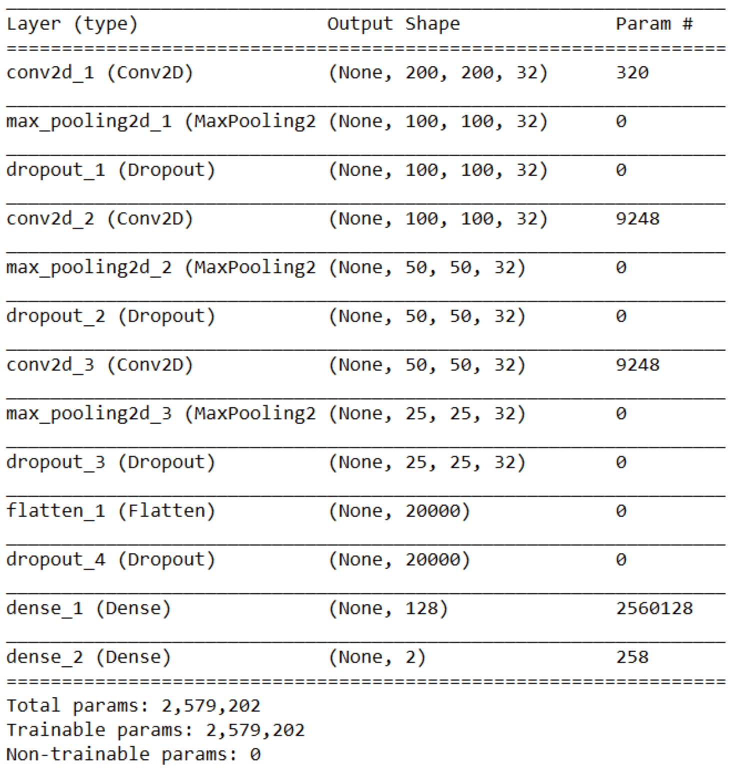
APPENDIX

1.1 FACE MASK DETECTION

Defining model architecture

```
defmodel_builder():
model = Sequential()
model.add(Conv2D(32, (3,3) ,input_shape=(200, 200, 1), activation='relu',
padding='same')) #Convolution
model.add(MaxPooling2D(pool_size= (2,2), strides=2)) #Pooling
model.add(Dropout(0.05))

model.add(Conv2D(32, (3, 3), activation='relu', padding='same')) #Convolution
model.add(MaxPooling2D(pool_size= (2,2), strides=2)) #Pooling
model.add(Dropout(0.05))

model.add(Conv2D(32, (3, 3), activation='relu', padding='same')) #Convolution
model.add(MaxPooling2D(pool_size= (2,2), strides=2)) #Pooling
model.add(Dropout(0.05))

model.add(Flatten())
model.add(Dropout(0.6))

model.add(Dense(units=128, activation='relu', kernel_initializer='uniform'))
model.add(Dense(units=2, activation='softmax'))

model.compile(optimizer='adam',
loss='categorical_crossentropy',
metrics=['accuracy'])

model.summary()
```

return model



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 200, 200, 32)      320       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 100, 100, 32)      0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 100, 100, 32)      0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 100, 100, 32)      9248      
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 50, 50, 32)        0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 50, 50, 32)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 50, 50, 32)        9248      
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 25, 25, 32)        0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 25, 25, 32)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 20000)             0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 20000)             0         
_________________________________________________________________
dense_1 (Dense)              (None, 128)               2560128   
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258       
=================================================================
Total params: 2,579,202
Trainable params: 2,579,202
Non-trainable params: 0
```

Visualizing data

import cv2

frommatplotlib import pyplot as plt

image1 = cv2.imread('Resources/Data/with_mask/93.jpg' ,1)

image2 = cv2.imread('Resources/Data/with_mask/96.jpg', 1)

image3 = cv2.imread('Resources/Data/without_mask/11.jpg', 1)

image4 = cv2.imread('Resources/Data/without_mask/16.jpg' , 1)

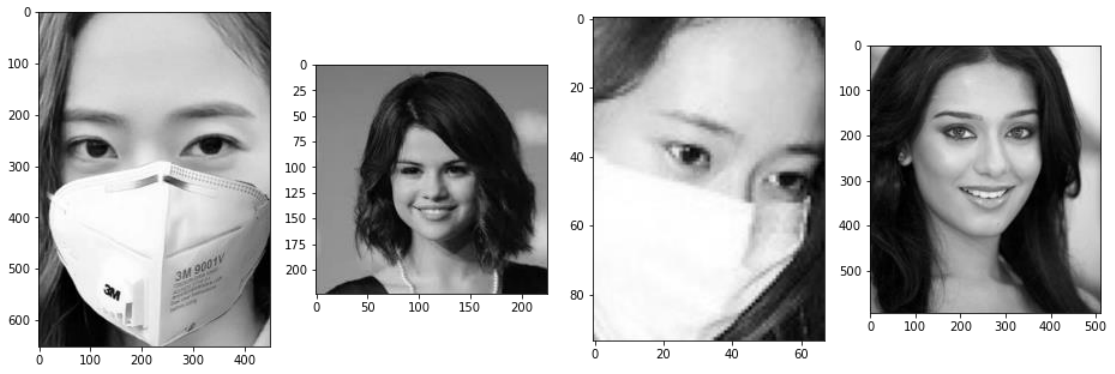
image1 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)

image2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)

image3 = cv2.cvtColor(image3, cv2.COLOR_BGR2GRAY)

image4 = cv2.cvtColor(image4, cv2.COLOR_BGR2GRAY)

```
fig = plt.figure(figsize=(20,10))

ax1 = fig.add_subplot(1,5,1)
ax2 = fig.add_subplot(1,5,3)
ax3 = fig.add_subplot(1,5,2)
ax4 = fig.add_subplot(1,5,4)

ax1.imshow(image1, cmap='gray')
ax2.imshow(image2, cmap='gray')
ax3.imshow(image3, cmap='gray')
ax4.imshow(image4, cmap='gray')

plt.show()
```



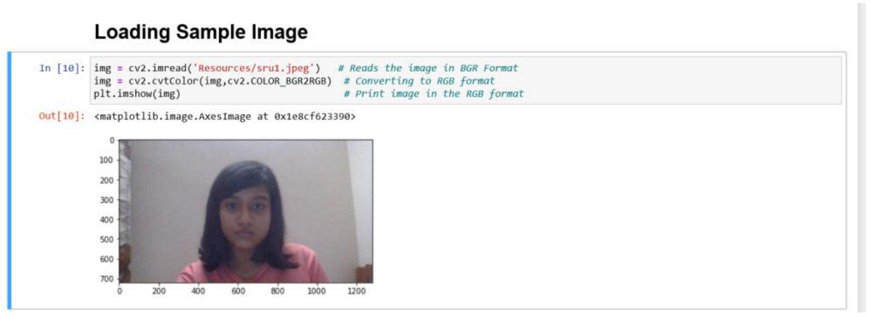
Loading Mask Detector Model and Predicting

```
# Loading Mask Detector Model and Predicting



mask_detector = load_model('Resources/mask_detector/detector-model')


# For Storing all the predictions by Mask Detector
prediction = []
```

```
# For Storing Image after Pre-Processing of all the faces detected
batch = []

for face in faces:
   # (StartX, StartY, EndX, EndY) respectively of the box
Sx,Sy,Ex,Ey = face

   # Croping only the face which will be feeded to the model
crop_img = img[Sy:Ey, Sx:Ex]

   # Converting Image to Grayscale as Mask-Detection model is trained on it
gray_img = np.dot(crop_img,[0.2989, 0.5870, 0.1140])

   # Resizing Gray Image ---> (200x200)
gray_img = skimage.transform.resize(gray_img, (200,200))

   # Scaling Gray-Values Between 0 and 1
gray_img = gray_img/255

   # Re-Dimensioning to 4D array that will be feeded to the CNN
gray_img = np.expand_dims(gray_img, -3)
gray_img = np.expand_dims(gray_img, -1)

batch.append(gray_img)



fori in range(faces_count):
   # Predicting if the person is wearing a mask or not
pred = mask_detector.predict(batch[i])
prediction.append(pred)
print(pred)
```

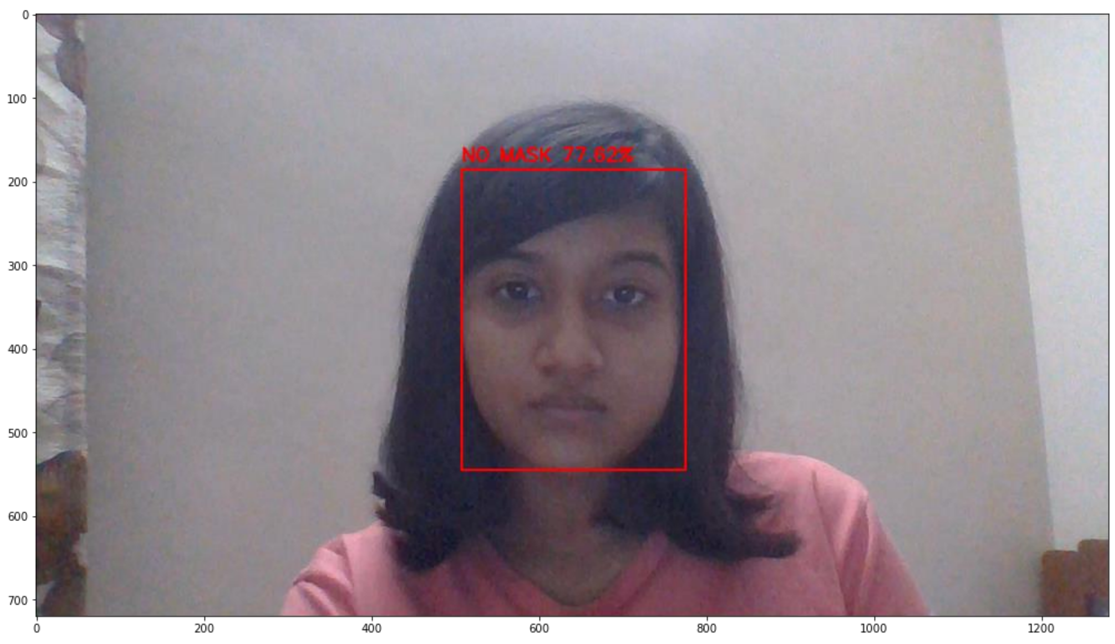
Drawing Boxes around Detected faces

```
i = 1
n = faces_count
fori in range(faces_count):
Sx,Sy,Ex,Ey = face
if prediction[i][0][0] > 0.5 :
label = 'MASK ' + str(round(prediction[i][0][0]*100,2)) + '%'
cv2.putText(img, label, (Sx,Sy-10), cv2.FONT_HERSHEY_SIMPLEX,
0.75,(0,255,0), 2)
cv2.rectangle(img,(Sx,Sy),(Ex,Ey),(0,255,0),2)
else :
label = 'NO MASK ' + str(round(prediction[i][0][1]*100,2)) + '%'
cv2.putText(img, label, (Sx,Sy-10), cv2.FONT_HERSHEY_SIMPLEX,
0.75,(255,0,0), 2)
cv2.rectangle(img,(Sx,Sy),(Ex,Ey),(255,0),2)

plt.show()
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111)
ax.imshow(img)
```



Loading Sample Image

```
In [10]: img = cv2.imread('Resources/sru1.jpeg')   # Reads the image in BGR Format
         img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)  # Converting to RGB format
         plt.imshow(img)                            # Print image in the RGB format

Out[10]: <matplotlib.image.AxesImage at 0x1e8cf623390>
```



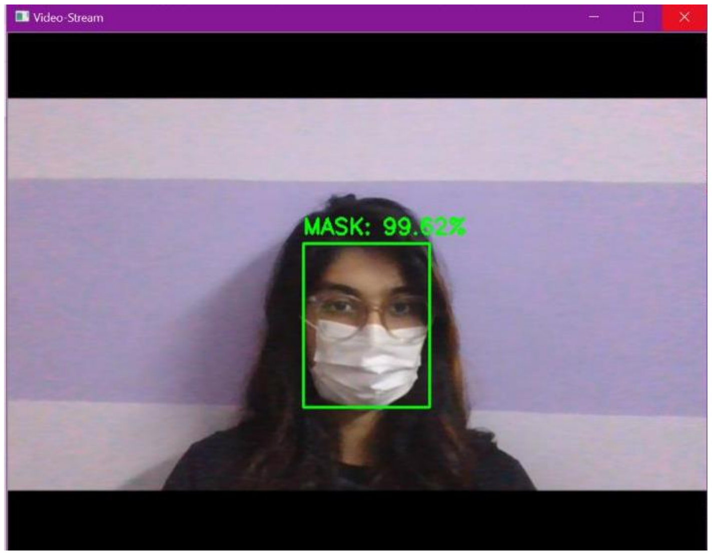
Looping over Frames in Stream and Detecting Faces With or Without Mask

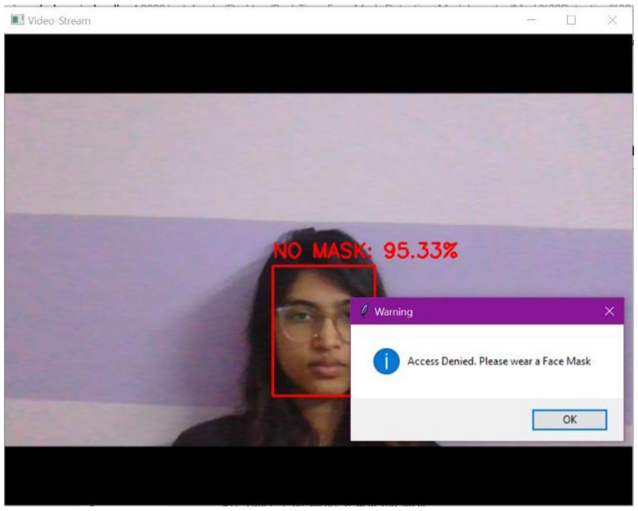
while True:

```
frame = vs.read()
frame = imutils.resize(frame, width=720)

    (locs, preds) = detect_and_predict_mask(frame, FaceNet, MaskNet)


    ## DRAWING BOXES AROUND DETECTED FACES ##

for (box, pred) in zip(locs, preds):

        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred[0]

label = "MASK" if mask >withoutMask else "NO MASK"
color = (0, 255, 0) if label == "MASK" else (0, 0, 255)
```

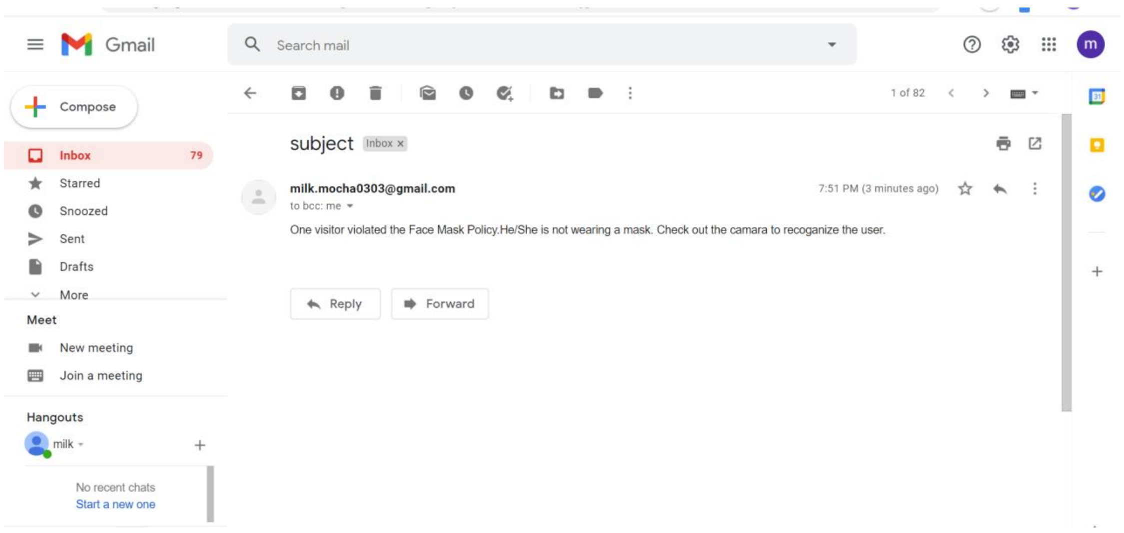
```
label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

cv2.putText(frame, label, (startX, startY - 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.75, color, 2)
cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)



        #If label 1-no mask, 0-wearing mask
ifwithoutMask> mask:
            #throw a warning saying the user to wear a mask. this will stay until the
user wears the mask
tkinter.messagebox.showinfo("Warning","Access Denied. Please wear a Face
Mask")

            #send an email to administrator if user is not wearing a mask
message='Subject : {}\n\n{}'.format(SUBJECT,TEXT)
mail=smtplib.SMTP('smtp.gmail.com',587)
mail.ehlo()
mail.starttls()
mail.login('milk.mocha0303@gmail.com','strong.password100')
mail.sendmail('harshiraina3@gmail.com','milk.mocha0303@gmail.com',message)
mail.close
else:
pass
break

cv2.imshow("Video-Stream", frame)

    # Closing the Video Stream when 'q' is pressed
if cv2.waitKey(1) & 0xFF == ord("q"):
```

```
break

cv2.destroyAllWindows()
vs.stop()
```





Access Denied. Please wear a Face Mask



subject

milk.mocha0303@gmail.com
to bcc: me
7:51 PM (3 minutes ago)

One visitor violated the Face Mask Policy.He/She is not wearing a mask. Check out the camara to recoganize the user.

1.2 SOCIAL DISTANCING

```
import time
import cv2
importnumpy as np

//Importing the required packages

confid = 0.5
thresh = 0.5

vname=input("Video name in videos folder:  ")
if(vname==""):
vname="Town.mp4"
vid_path = "./videos/"+vname



//Specifying the path for the input video and taking user input.
```

```
defcalibrated_dist(p1, p2):
return ((p1[0] - p2[0]) ** 2 + (p1[1] - p2[1]) ** 2) ** 0.5

// Calculating the calibrated distance using euclidean distance formula.



defisclose(p1, p2):
c_d = calibrated_dist(p1, p2)
calib = 576
if 0 <c_d< 0.15 * calib:
return 1
elif 0 <c_d< 0.2 * calib:
return 2
else:
return 0
//Detecting the risk returning 1 if  close, returning 2 if second close, returning 0 if
safe.

labelsPath = "./coco.names"
LABELS = open(labelsPath).read().strip().split("\n")
np.random.seed(42)

weightsPath = "./yolov3.weights"
configPath = "./yolov3.cfg"
```

//Specifying the path for the input files i.e weights and configuration files.We will load the YoloV3 weights and configuration file with the help of the dnn module of OpenCV. The cfg file describes the layout of the network, block by block.

```
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
```

//Now to read the files using the cv2.dnn module through the darknetfunction .

```
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

//net.getUnconnectedOutLayers() returns the indices of the output layers of the network, getLayerNames()returns the name of all layers of the network.

```
vs = cv2.VideoCapture(vid_path)
```

//capturing the videopath through the cv function

```
writer = None
(W, H) = (None, None)
fl = 0
q = 0
while True:
 (grabbed, frame) = vs.read()
if not grabbed:
break
if W is None or H is None:
        (H, W) = frame.shape[:2]
        q = W
```

//loop over the frames from the video stream and read the next frame from the file. If the frame was not grabbed, then we have reached the end of the stream then break the condition.Resize the frame accordingly.

```
frame = frame[0:H, 200:q]
   (H, W) = frame.shape[:2]
blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
swapRB=True, crop=False)
```

//using blobFromImage in a function called detect_objects() that accepts frame from video or webcam stream, model and output layers as parameters.

```
net.setInput(blob)
start = time.time()
layerOutputs = net.forward(ln)
end = time.time()
```

//net.forward() - Runs a forward pass to compute the net output.
i.enet.forward() will give Numpyndarray as output which we can use to plot the box on the given input .

```
boxes = []
confidences = []
classIDs = []
```

//Initializing the boxes, confidence and classIDs list

```
for output in layerOutputs:

for detection in output:

scores = detection[5:]
classID = np.argmax(scores)
confidence = scores[classID]
```

```
if LABELS[classID] == "person":
```

//From the output from the layerOutputs that are the layers in the network , we are considering the maximum of the scores as the classID and confidence is the score of the computed classID and also checking for the condition of the label should be person. Only if that condition satisfies , we need to compute the distances.

```
if confidence >confid:
box = detection[0:4] * np.array([W, H, W, H])
                    (centerX, centerY, width, height) = box.astype("int")

                    x = int(centerX - (width / 2))
                    y = int(centerY - (height / 2))

boxes.append([x, y, int(width), int(height)])
confidences.append(float(confidence))
classIDs.append(classID)
```

//Now , if the class label is person ,then checking the confidence value whether it is lower than confid that we have initialized as 0.5.If it is having lesser value , then it might not be indicating properly the object.that is why it should be greater than that value.Now extracting the values of the bounding box from the detection and appending the confidence list ,boxes and classID list.

```
idxs = cv2.dnn.NMSBoxes(boxes, confidences, confid, thresh)
```

//To remove the noise that may be created for one object , that is many boxes may be created around one object, it means the noise is being created. hence we use NMS(non-maxima suppression) to remove the noise.

```
iflen(idxs) > 0:

status = list()
idf = idxs.flatten()
close_pair = list()
s_close_pair = list()
center = list()
dist = list()
```

//Now , initializing the status , close_pair, second_close_pair(low_risk) , center and co_ordinate information lists . using the flatten function, we get a copy of the given array into one dimension.

```
fori in idf:
        (x, y) = (boxes[i][0], boxes[i][1])
        (w, h) = (boxes[i][2], boxes[i][3])
center.append([int(x + w / 2), int(y + h / 2)])
```

//Loop over the values in the idf(one dimension array) ,computing the center points and append in the center list and also append the values of the co_ordinates in the co_ordinate list.

```
status.append(0)
fori in range(len(center)):
for j in range(len(center)):
          g = isclose(center[i], center[j])

if g == 1:
```

```
close_pair.append([center[i], center[j]])
status[i] = 1
status[j] = 1
elif g == 2:
s_close_pair.append([center[i], center[j]])
if status[i] != 1:
status[i] = 2
if status[j] != 1:
status[j] = 2
```

//Now, based on the range of length of co_ordinates of the center, we need to compute the categorization of risk and safety using the function is_close. Finally g returns the value of either 0,1,2 and based on that append the co_ordinates into the close_pair and second_close_pair list.

```
total_p = len(center)
low_risk_p = status.count(2)
high_risk_p = status.count(1)
safe_p = status.count(0)
kk = 0

fori in idf:
         (x, y) = (boxes[i][0], boxes[i][1])
        (w, h) = (boxes[i][2], boxes[i][3])
if status[kk] == 1:
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 150), 2)

elif status[kk] == 0:
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

else:
```

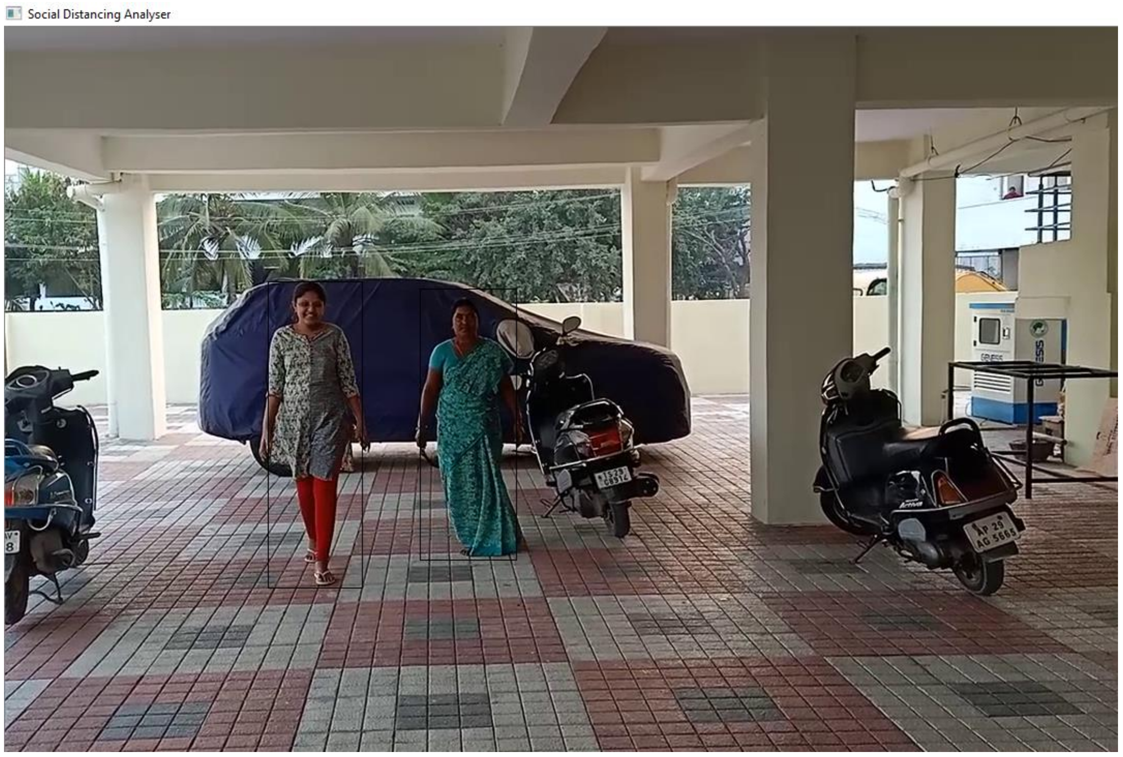
```
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 120, 255), 2)

kk += 1
```

//Here we are looping over the values in the array of the one -dimension ,checking the condition based on the co_ordinates of the risk in the status list and drawing the rectangle using the rectangle() function where we need to give the frame(input), start coordinates(x,y) , end coordinates(x+w,y+h),color (RGB format) and thickness(2). And categorization is expressed in the change of color (where red-high_risk, orange-low_risk and green -safe).

```
for h in close_pair:
cv2.line(frame, tuple(h[0]), tuple(h[1]), (0, 0, 255), 2)
for b in s_close_pair:
cv2.line(frame, tuple(b[0]), tuple(b[1]), (0, 255, 255), 2)

cv2.imshow('Social distancing analyser', frame)
cv2.waitKey(1)

if writer is None:
fourcc = cv2.VideoWriter_fourcc(*"MJPG")
writer = cv2.VideoWriter("output.mp4", fourcc, 30,
                         (frame.shape[1], frame.shape[0]), True)
writer.release()
vs.release()
```

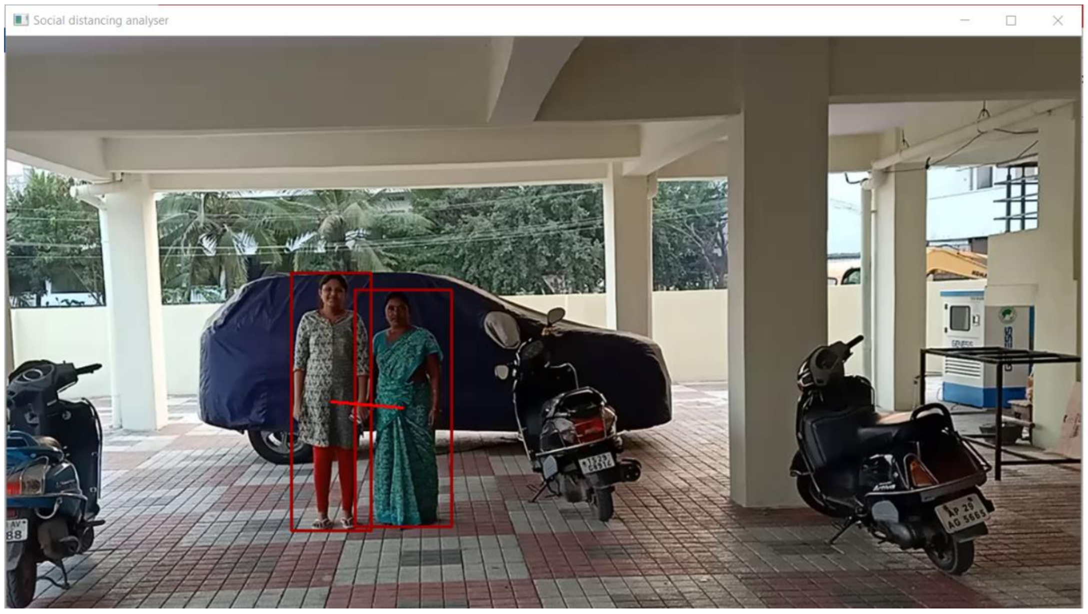
//for the video or webcam purposes we use the waitkey(1) which will wait for keyPress for just 1 millisecond and it will continue to refresh and read frames from your video or webcam. And finally release or close the video writer.

PLAIN OBJECT DETECTION FOR THE INPUT VIDEO

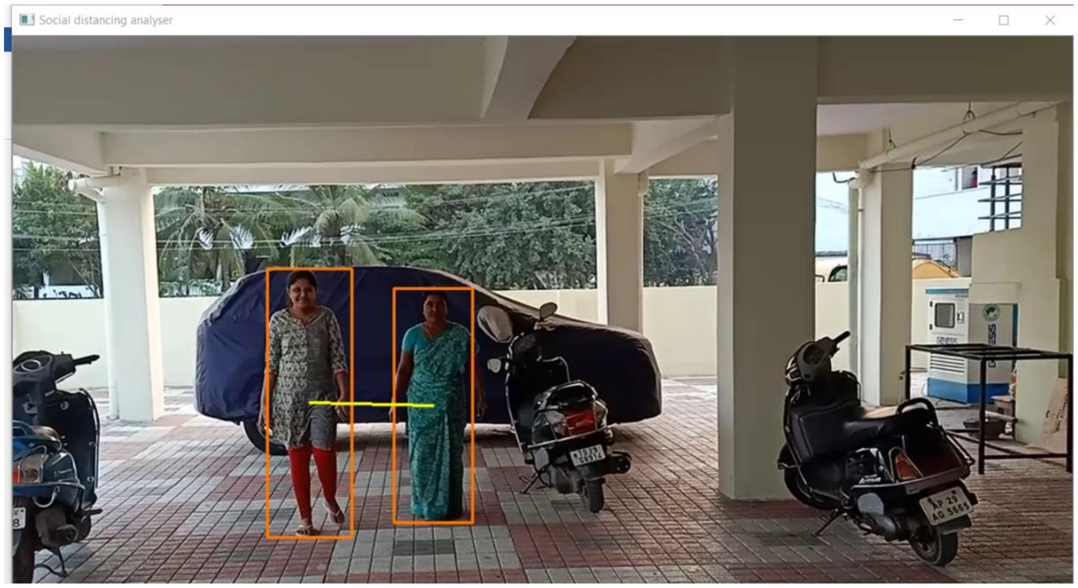


RISK DETECTION FOR THE INPUT VIDEO

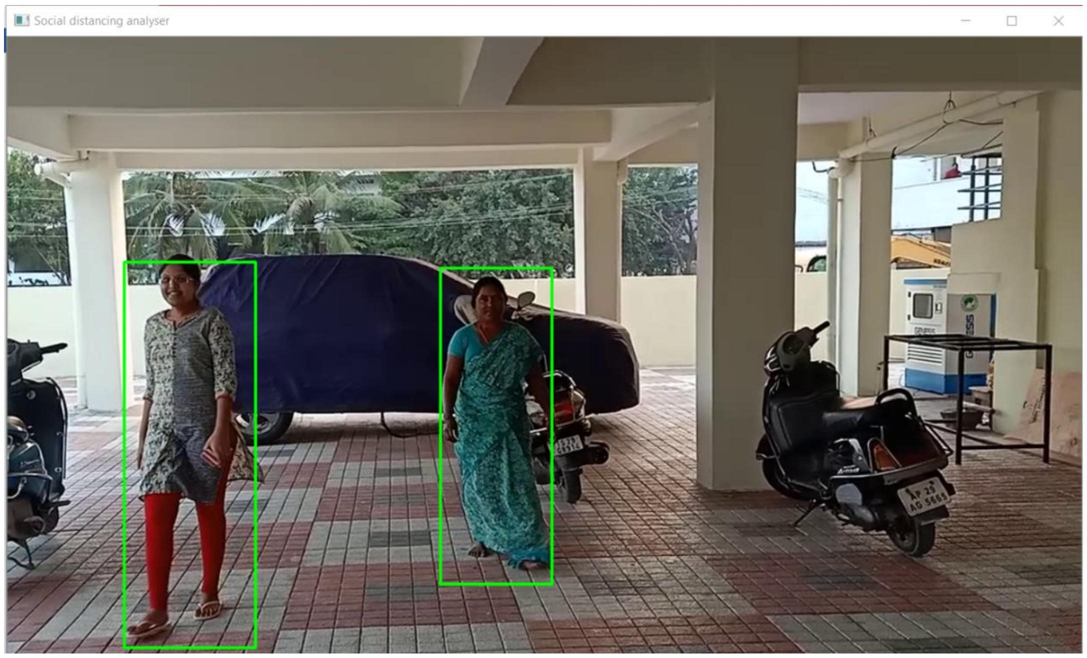
close:



second close:



safe:


Social distancing analyser