Social Distancing System for Public Spaces

Chintan Jethva[1], Arya Kasulla[2], Yajnesh Shetty[3], Suruchi Singh[4], SonuTejwani[5]

[1]Assistant Professor, Dept. of Electronics and Telecommunication Engineering, Vivekanand Education Society's Institute of Technology, Maharashtra, India

[2,3,4,5]Student, Dept. of Electronics and Telecommunication Engineering, Vivekanand Education Society's Institute of Technology, Maharashtra, India

***

Abstract - *As the world struggles with the devastating effects of the SARS-CoV-2, our country witnesses' new horrors each day, it all boils down to taking efforts on an individual level to ensure the well-being of ourselves. Our project aims at alleviating the effects of this pandemic by putting forth a model that helps curb minuscule issues which when put together has the potential to solve a colossal problem. Social Distancing during COVID-19 encompasses four parts. In the first part "Automatic queue monitoring system" we have designed a model that monitors the entry and exit of the people entering and leaving the area in perspective. The second part, "Contactless Temperature Sensing" enables recording the temperature of the people without any human intervention. The third part "Mask detection system" is instrumental in ensuring that everyone in the designated area wears a mask all the time and finally the last part "Social distance monitoring system" is the model that works on maintaining a constant distance of 1 ft. (as mentioned by the health authorities of the country) between the people at all costs. Through this project, we aim at demonstrating the best of our technical skills to help our community and the nation at large to tackle this life-threatening virus.*

Keywords— Social distancing system, Mask Detection system, Queue Monitoring, Temperature Sensing

1. INTRODUCTION

Coronavirus or novel coronavirus which is taxonomically termed as SARS-CoV-2 and named by the World Health Organization (WHO) as COVID-19 which emerged from Wuhan city, Hubei Province of China by the end of 2019 has caused unprecedented panic across the world. The rapid transmission of this virus from human to human-made the World Health Organization (WHO) declare this as a public health emergency of international concern and called it a global pandemic. As of May 14, 2020, globally 42,48,389 COVID-19 cases have been reported and caused 2,92,046 deaths. The highest human casualty reported was from the USA with 1,09,121 deaths.[1]

The second wave did not do any favours to the prevalent devastations of the virus. Despite many measures being undertaken such as travel bans, curfews, and even the nascent vaccinations, the impediment of this virus did not subside. The world once again witnessed the horror of lack of medication and vaccines, hospitals, and the increase in the average mortality rate of the disease. India has had some of the worst repercussions of this second wave. India ranked first in the highest number of cases daily and third in the number of deaths.[2].

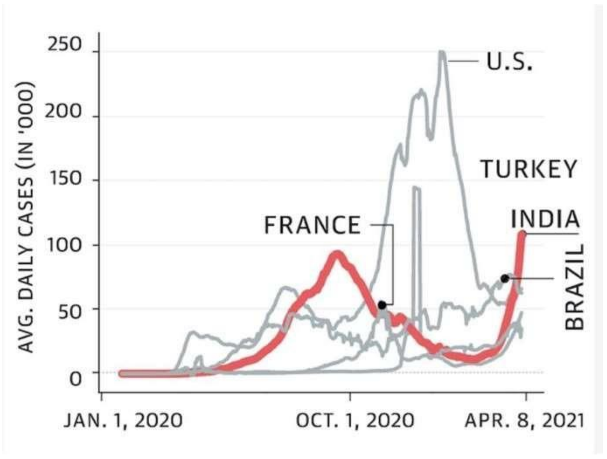


Fig. 1 Average daily cases in highly affected countries. [2]

Building upon the preceding point, we figured that in our capacity we could ameliorate the problem of mass mobility. With cases of violation of the social distance rule by more than half the population of India, it seemed necessary to take efforts in the direction of social distancing. The lax attitude of the general public in particular the lack of social distancing and lack of wearing masks has only exacerbated the situation. Thus we aim at making a model that counteracts this impediment and ensures that no matter what, people should maintain a minimum safe distance amongst themselves and wear masks. In the succeeding chapters, we talk about the research, execution, and prospects of our product.

2. LITERATURE REVIEW

We referred to the work of researchers who worked on a similar aim. In June 2020, Rucha Visal and her co-authors proposed a system for Monitoring Social Distancing for Covid19 using OpenCV and Deep Learning, their survey paper emphasizes a surveillance method that is designed to keep a track of the pedestrians and avoid overcrowding.[3] A concept called Visual Social Distancing (VSD) problem was introduced, It is defined as the automatic estimation of the interpersonal distance from an image, and the characterization of related people aggregations, to figure out the reasons for the possible breaks of such distance limitations and understand if this implies a potential threat.[4]

Deep Learning-based Safe Social Distancing and Face Mask Detection in Public Areas for COVID-19 Safety Guidelines Adherence is a proposed efficient computer vision-based approach focused on the real-time automated monitoring of people to detect both safe social distancing and face masks in public places by implementing the model on raspberry pi4 [5]. Also, an Automated Social Distancing Gate with Non-Contact Body Temperature Monitoring using Arduino Uno was proposed to develop an automatic social distancing gate and body temperature detection sensor [6]

3. BLOCK DIAGRAM

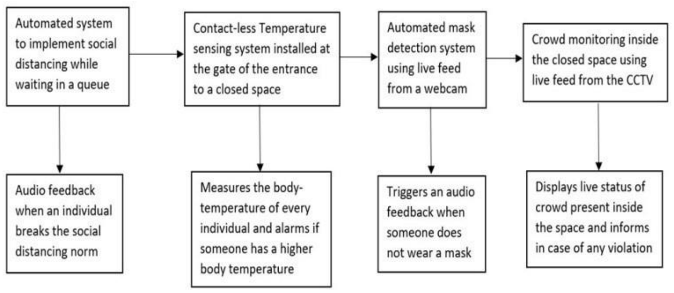
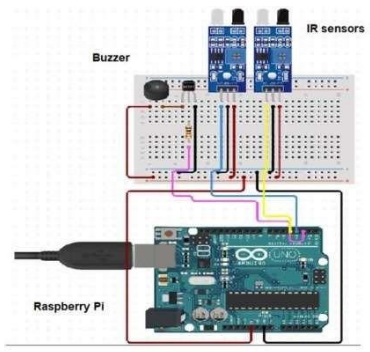

Automated system to implement social distancing while waiting in a queue
Contact-less Temperature sensing system installed at the gate of the entrance to a closed space
Automated mask detection system using live feed from a webcam
Crowd monitoring inside the closed space using live feed from the CCTV
Audio feedback when an individual breaks the social distancing norm
Measures the body-temperature of every individual and alarms if someone has a higher body temperature
Triggers an audio feedback when someone does not wear a mask
Displays live status of crowd present inside the space and informs in case of any violation

Fig. 2 Block Diagram of the system

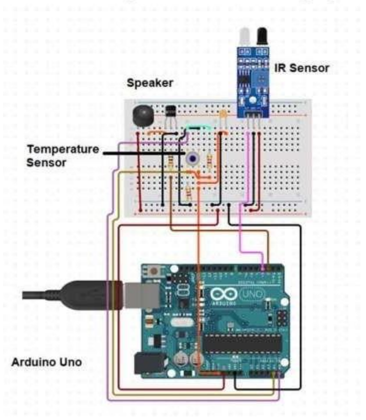
3.1AUTOMATIC QUEUE MONITORING SYSTEM AND TEMPERATURE SENSING SYSTEM

3.1.1 Circuit Diagram:

1) Automatic Queue Monitoring:



2) Contactless Temperature Sensing System:



3.1.2 Components Used:

1) Raspberry Pi:

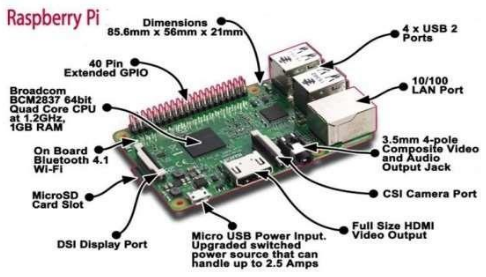

Raspberry Pi
Dimensions 85.6mm x 56mm x 21mm
4 x USB 2 Ports
40 Pin Extended GPIO
10/100 LAN Port
Broadcom BCM2837 64bit Quad Core CPU at 1.2GHz, 1GB RAM
3.5mm 4-pole Composite Video and Audio Output Jack
On Board Bluetooth 4.1 Wi-Fi
MicroSD Card Slot
CSI Camera Port
DSI Display Port
Micro USB Power Input. Upgraded switched power source that can handle up to 2.5 Amps
Full Size HDMI Video Output

Fig. 3 Raspberry Pi

We have used this single-board computer device to easily compute the input data received by the sensors and give out audio feedback depending on the use case. The original Pi had a single-core 700MHz CPU and just 256MB RAM, and the latest model has a quad-core CPU clocking in at over 1.5GHz, and 4GB RAM. It allows us to control hardware components for computing and explore the world of the Internet of Things (IoT).

2) Arduino Uno R3:

Fig. 4 Arduino uno R3

We have implemented our Temperature detection model by using the r3 version of arduino. This board includes 14 digital input/output pins, Analog inputs-6, a USB connection, quartz crystal-16 MHz, a power jack, a USB connection, resonator16Mhz, a power jack, an ICSP header, and an RST button.

3) Infrared Thermometer Sensor:

Fig. 5 Infrared Thermometer Sensor

MLX90614 is an infrared temp sensor that measures infrared radiation. It consists of a lens to focus the infrared thermal radiation onto a detector, which converts the radiant energy into an electric signal. This configuration facilitates temperature measurement from a distance, without the need for contact with the object to be measured.

4) IR Proximity Sensor:

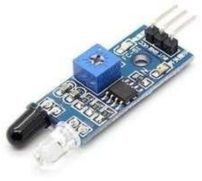


Fig. 6 IR proximity sensor

We use IR proximity sensors in both of our systems. In the case of the temperature detection system, it is placed below the thermometer sensor to detect the presence of an individual to start the temperature reading. Whereas in the case of the queue monitoring system we used it to detect the presence of individuals entering the gate.

5) LCD Module:

Fig. 7 LCD Module

We made use of a 16x2 LCD module. It will be integrated with the thermometer sensor to display the appropriate temperature of an individual.

6) Buzzer:

Fig. 8 Buzzer

The Buzzer is used in the case of the automatic temperature detection system to send an alarm when the person's temperature detected is above normal (Greater than 37.5 degrees Celsius).

3.1.3 Methodology

1) Stage 1:

We assume a two-door entry system to execute our system. Door 1 is the outer door and Door 2 is the one where the individual will have to measure his/her temperature and will be checked if wearing a mask before entering the premises. As the first person reaches Door 1, he/she will hear audio feedback instructing him/her to move forward towards Door 2.

2) Stage 2:

Once the individual moves up from Door 1 to Door 2, the system is designed so that the person behind him in the queue can't move in. So, if any person enters Door 1 while there is still someone standing at Door 2, an audio warning will be generated asking him/her to wait.

3) Stage 3:

When the person at Door 2 is cleared off for entry, he/she moves in the premise and the person waiting at Door 1 gets intimidated to move up to Door 2, again via audio feedback.

4) Role of Raspberry Pi:

The system uses IR sensors at both Door 1 and Door 2 to detect the presence of any person and feed this data continuously to the Raspberry pi, which in turn controls the speaker output.

5) Integration of temperature sensor with Arduino:

We first integrated the temperature sensor with the arduino using the library provided by arduino IDE. We made appropriate connections as shown in the figure above to achieve so. This helped detect the temperature of the object in front of the temperature sensor. An IR sensor ensured that only the detection of a human in perspective was recorded. The recording was then displayed on an LCD and if the threshold temperature was reached it led to audio feedback.

3.2 MASK DETECTION SYSTEM

3.2.1 Dataset:

We have used an intensive dataset provided by Kaggle which included a good amount of both with mask and without mask images. Along with this, we used a Real-world masked face dataset (RMFD).

3.2.2 Methodology and Implementation:

Since the problem we are trying to solve is a classification problem, wherein we predict whether the person is wearing a mask or not, it is practical to use a transfer learning approach. Transfer learning is a process wherein we use a model which has been pre-trained on a problem, to solve another problem having a common ground. This is typically understood in a supervised learning context, where the input is the same but the target may be different. For example, we may learn about one set of visual categories, such as cats and dogs, in the first set, then learn about a different set of visual categories, such as ants and wasps, in the second set. [7] Fig.9 shows a general way to use a pre-trained model.

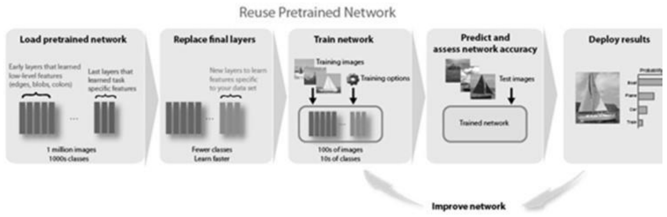


Fig.9 Using a Pre-trained Model [8]

1) Image Classification:

As we discussed the importance of using a pre-trained network earlier, we decided to use ResNet-18 to develop our model. ResNet-18 is a convolutional neural network that is 18 layers deep. It was evaluated on the ImageNet 2012 classification dataset that had over 1000 classes. The model was trained over 1.28 million training images and evaluated on 50k validation images[9].

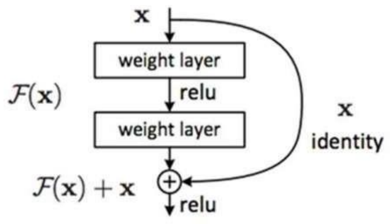

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig. 10 ResNet-18 Logic Flow [9]

ResNet is an artificial neural network of some sort that builds on constructs known from pyramidal cells in the cerebral cortex. It does this by utilizing skip connections to jump over some layers. It can be visualized with the help of Fig.10

2) Data Preparation:

When it comes to training the model, pre-processing is one of the most important steps as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn. We have used the "torchvision. transforms" module for this purpose. Transforms are a collection of common image transformations that can be chained together using "compose". We then split our dataset into train/validation sets keeping the proportion between them constant. We used 60% of our dataset for training, 20% for validation, and the remaining 20% for testing.

Since we were dealing with an imbalanced dataset, we needed to pass this information to the loss function to avoid unproportioned step sizes of the optimizer. We did this by assigning a weight to each class, according to its representability in the dataset. We assigned more weight to classes with a small number of samples so that the network will be penalized more if it makes mistakes predicting the label of these classes. While classes with large numbers of samples, we assign to them a smaller weight. This makes our network training agnostic to the proportion of classes.

3) Configuring the optimizer:

It is important to consider the setbacks offered by our model. One way to visualize this was by taking into account the loss factor. We used this information to improve our model and overcome the shortcomings. This is called optimization. Optimizers are algorithms or methods that change certain attributes such as weights and learning rate to reduce the losses. We used an Adaptive Moment Estimation (Adam) optimizer for our model. Adam optimizer (Adaptive Moment Estimation) looks like a combination of RMSprop and stochastic gradient descent momentum. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance).

Training the Model:

We train our model simply by initializing our MaskDetector object and fitting it into the model. We calculated training loss, validation loss, and accuracy and saved the model with the highest accuracy and the lowest loss. We trained our model for 20 epochs and saved our model at the epoch where we found it to be most accurate.

4) Testing the models on real videos:

To test our model we used a face detection model called Multi-Task Cascaded Convolutional Neural Network (MTCNN). MTCNN or Multi-Task Cascaded Convolutional Neural Network is a neural network that detects faces and facial landmarks on images. MTCNN is one of the most accurate and widely used face detection models that consists of 3 neural networks connected in a cascade. Having 3 networks — each with multiple layers — allows for higher precision, as each network can fine-tune the results of the previous one. For testing and using our model on a live video we extracted frames from the video feed, passed them to the mask detector model, and drew bounding boxes around the detected faces (green border in cases where mask was detected and red border in cases where mask was not detected). We also used a playground library provided by python to provide audio feedback.

3.3 SOCIAL DISTANCING MONITORING SYSTEM

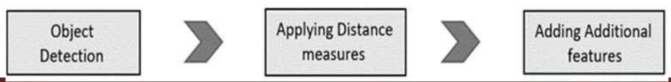


Fig. 10 Flow diagram of social distancing system.

3.3.1 Object Detection:

To detect whether individuals are maintaining social distancing our first hurdle was to detect the individual in the frame. For this purpose, we needed to select not only an accurate and dependable model but also one that would provide results at a high speed in real-time. YOLOv3 proved to have significant improvements over the previous versions of YOLO (You only live once). At 320 by 320 YOLOv3 runs in 22 ms at 28.2 mAP which is as accurate as an SSD but is three times faster. You only look once (YOLO) is a state-of-the-art, realtime object detection system. On a Pascal Titan X, it processes images at 30 FPS and has a mean average precision (mAP) of 57.9% on COCO test-dev. The performance of various other object detectors vs the YOLO detector can be seen in the figure below. In comparison to other detectors, YOLOv3 is extremely fast and accurate. In mAP measured at .5 IOU YOLOv3 is on par with Focal Loss but about 4 times faster. Moreover, one can easily trade-off between speed and accuracy simply by changing the size of the model without any retraining [13].

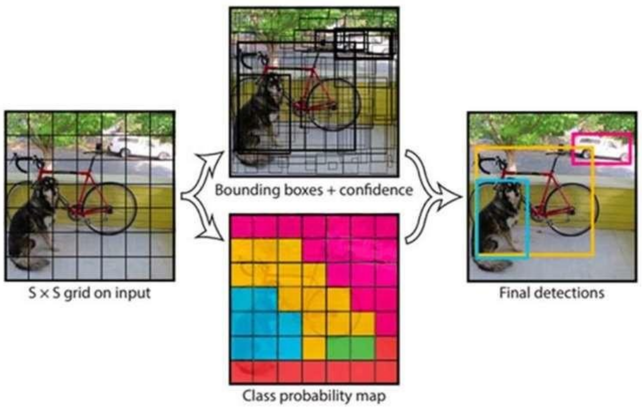


Fig. 11 YOLOv3 Logic Flow [11]

3.3.2 Methodology and Implementation:

We loaded YOLO from the disk using a special DNN function provided by OpenCV named “cv2.dnn.readNetFrom DarkNet()”. Since we intended on making a real-time detector, we needed a tool to extract frames from the live video feed and pre-process them before feeding them to the model. For this purpose we used OpenCV. There are several advantages of using OpenCV with YOLOv3 such as easy integration, faster implementation, and python-support. We determined the output layer names from the YOLO model and constructed a blob from the given input. A blob is just a collection of images with the same spatial dimensions, same depth and have all been preprocessed in the same manner. Following this, we performed a forward pass through our YOLO network and showed the inference time for YOLO. We then created three lists to visualize our result.

- Boxes: This list contained the bounding boxes around the object.
- Confidence: The confidence value that YOLO assigns to an object. Lower confidence values indicate that the object might not be what the network thinks it is. Remember from our command-line arguments above that we’ll filter out objects that don’t meet the 0.5 threshold.
- ClassId: It contains the classID of the detected object with reference to the COCO dataset.

Using the confidence value we extract all the weak detections. Further, we witnessed overlapping of bounding boxes, to overcome that we applied Non-maxima

Suppression. NMS shows good results and also ensures that we do not have any redundant or extraneous bounding boxes.

3.3.3 Applying Distance Measures:

Once we constructed the bounding boxes around all individuals present in the frame, we calculated the center of each such bounding box. We considered two objects at a time, stored their centers in two arrays, and calculated the distance between them. In cases where the distance was smaller than the threshold value, we drew a red box around the object indicating that the person is violating social distancing rules. Whereas when the distance was large enough to be considered safe, we drew a green box around the object indicating that rules were followed. Following is the formula we used to calculate the distance between the objects, p1 indicates person 1 and p2 indicates person 2. p1 and p2 are arrays containing (x,y) coordinates of the center of the object.

((p1[0] - p2[0]) ** 2 + 550 / ((p1[1] + p2[1]) / 2) * (p1[1] - p2[1]) ** 2) ** 0.5

Additionally using python libraries we created a live dashboard that not only displayed the live feed of the video input but also displayed intricate details such as how many individuals are present in the room, how many of them were following social distancing and how many were violating the rules and also provided audio feedback to make the system more user friendly and easy to execute.

4. RESULTS AND ANALYSIS

4.1 Queue Monitoring System:

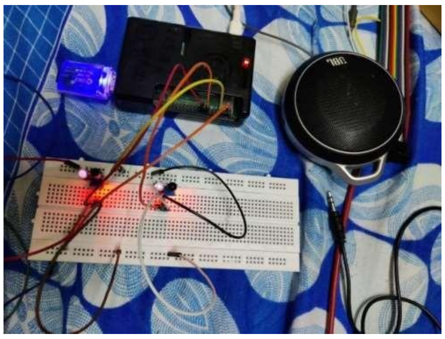


Fig. 11 Implementation of Automatic Queue Monitoring System

Fig.11 shows the implementation of this system. The two IR sensors attached to the doors were working in synchronization with each other. When the IR sensor at door 1 detected the presence of an individual it let the person inside for further procedure of temperature detection as mentioned earlier. However, during the time interval in which the first person was getting his/her temperature checked and leaving the second door if a second person tried to enter the area he/she was warned by audio feedback that stated “Please wait for the previous person to leave.” After the first person leaving the second person was allowed and a similar procedure was carried out for the third person.

4.2 Contactless Temperature Sensing System:

After connecting the components, interfacing it with arduino, and then executing the program we were able to achieve our objective. The temperature sensor appropriately sensed the temperature of an individual with a deviation of +- 0.5 degree celsius.

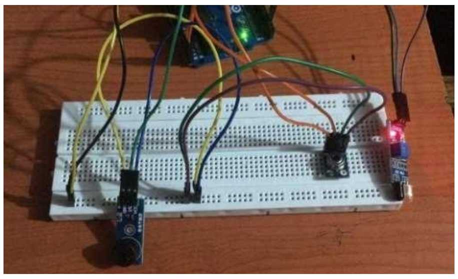
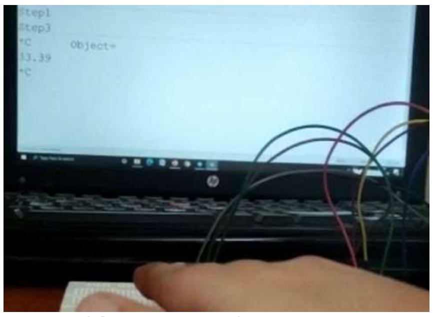


Fig. 12 Implementation of Contactless Temperature Sensing System

Owing to the continuous nature of detections, an IR sensor was placed. The sensor ensured the temperature detection only when it came in contact with a human arm instead of noting ubiquitous temperatures in the surroundings. The readings are then successfully displayed on the LCD along with audio feedback. We had kept a threshold of 38 degrees celsius over which a buzzer was initiated.

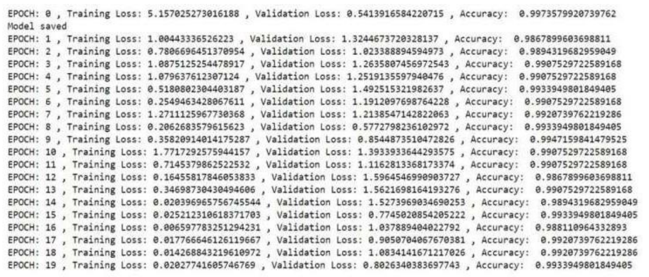


```
Step1
Step3
*C         Object=
33.39
*C
```

Fig. 13 Snippet of the execution of temperature sensing system

1.1 Mask Detection System:

We were able to create an accurate and effective mask detection system using the technologies mentioned in the earlier chapters. To visualize the accuracy and effectiveness of the model we calculated training loss, validation loss, and accuracy parameters of the model which can be seen in the below figure.



```
EPOCH: 0 , Training Loss: 5.157025273016188 , Validation Loss: 0.5413916584220715 , Accuracy:  0.9973579920739762
Model saved
EPOCH: 1 , Training Loss: 1.00443336526223 , Validation Loss: 1.3244673720328137 , Accuracy:  0.9867899603698811
EPOCH: 2 , Training Loss: 0.7806696451370954 , Validation Loss: 1.023388894594973 , Accuracy:  0.9894319682959049
EPOCH: 3 , Training Loss: 1.0875125254478917 , Validation Loss: 1.2635807456972543 , Accuracy:  0.9907529722589168
EPOCH: 4 , Training Loss: 1.079637612307124 , Validation Loss: 1.2519135597940476 , Accuracy:  0.9907529722589168
EPOCH: 5 , Training Loss: 0.5180802304403187 , Validation Loss: 1.492515321982637 , Accuracy:  0.9933949801849405
EPOCH: 6 , Training Loss: 0.2549463428067611 , Validation Loss: 1.1912097698764228 , Accuracy:  0.9907529722589168
EPOCH: 7 , Training Loss: 1.2711125967730368 , Validation Loss: 1.2138547142822063 , Accuracy:  0.9920739762219286
EPOCH: 8 , Training Loss: 0.2062683579615623 , Validation Loss: 0.5772798236102972 , Accuracy:  0.9933949801849405
EPOCH: 9 , Training Loss: 0.35820914014175287 , Validation Loss: 0.8544873510472826 , Accuracy:  0.9947159841479525
EPOCH: 10 , Training Loss: 1.7717292575944157 , Validation Loss: 1.3933933644293575 , Accuracy:  0.9907529722589168
EPOCH: 11 , Training Loss: 0.7145379862522532 , Validation Loss: 1.1162813368173374 , Accuracy:  0.9907529722589168
EPOCH: 12 , Training Loss: 0.16455817846053833 , Validation Loss: 1.5964546990903727 , Accuracy:  0.9867899603698811
EPOCH: 13 , Training Loss: 0.34698730430494606 , Validation Loss: 1.5621698164193276 , Accuracy:  0.9907529722589168
EPOCH: 14 , Training Loss: 0.020396965756745544 , Validation Loss: 1.5273969034690253 , Accuracy:  0.9894319682959049
EPOCH: 15 , Training Loss: 0.025212310618371703 , Validation Loss: 0.7745020854205222 , Accuracy:  0.9933949801849405
EPOCH: 16 , Training Loss: 0.006597783251294231 , Validation Loss: 1.037889404022792 , Accuracy:  0.988110964332893
EPOCH: 17 , Training Loss: 0.017766646126119667 , Validation Loss: 0.9050704067670381 , Accuracy:  0.9920739762219286
EPOCH: 18 , Training Loss: 0.014268843219610972 , Validation Loss: 1.0834141671217026 , Accuracy:  0.9920739762219286
EPOCH: 19 , Training Loss: 0.020277416057 46769 , Validation Loss: 0.8026340383697743 , Accuracy:  0.9933949801849405
```

Fig. 14 Snippet of Loss and accuracy parameters of the mask detection system

We saved the model which provided the highest accuracy and lowest loss. The accuracy of our model thus was above 99%. We then integrated our model into the system with the help of some common python libraries. Our end product then alerted the individual not wearing the mask with audio feedback. The snippets of the result can be seen below.

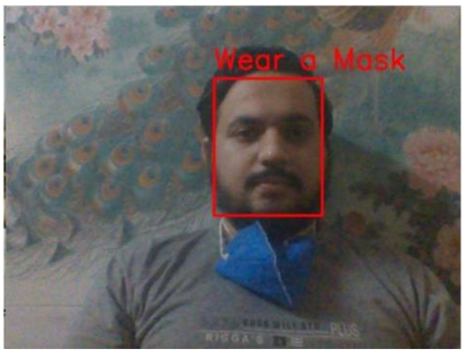


Fig. 15 (a) Output of mask detection system (No Mask Detected)

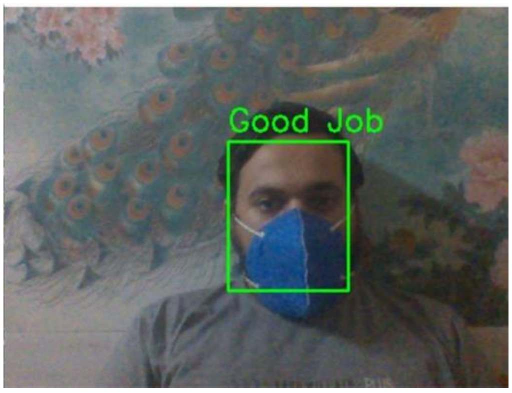


Fig. 15 (b) Output of mask detection system (Mask Detected)

As evident in Fig.15 (a) and Fig.15 (b), the model detects the partial covering of the face by the masks and thus the digital frame turns red, and subsequently as the person wears the masks appropriately, the frame turns green.

1.2 Social Distancing Monitoring System:

We developed the social distance monitoring system with the help of deep learning concepts and python tools. Since we used YOLOv3 for object detection we were able to obtain 45 FPS on a GPU. We then successfully were able to measure the distance between two individuals and accordingly provide our feedback. We used python's playsound library here as well to provide audio feedback. As soon as the distance norm between two individuals was violated our system provided an alert and the dashboard would indicate the statistics of this violation. The result can be seen in the snippets below.

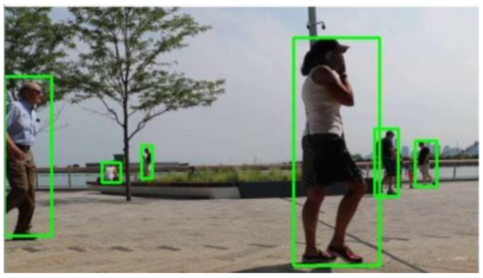


Fig. 16 (a) Output of social distancing monitoring system (No Violation)

As the people in Fig.16 (a) are maintaining appropriate distance, the digital frames are green and the dashboard does not show any changes.

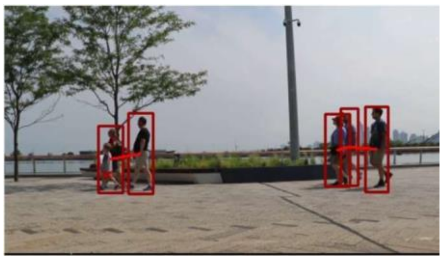


Fig. 16 (b) Output of social distancing monitoring system (Violation detected)

The people as and when are breaking the protocol, the frames become red and the dashboard shows the statistics of the number of safe people and those who are not. Can be seen in Fig.16 (b)

2. CONCLUSION

The project presented in this report tries to ameliorate the problems our country and the world at large are suffering through. As the cases show stark rises in current months and promise to increase further in the upcoming months, our project seeks a pathway to avoid any harm at affordable prices. We talk about four parts of the project that help in small ways and cumulatively address a colossal problem. The collaboration of different utilities, components, and technologies across various deep learning, electronics, and machine learning technologies is solely to put an end to a catastrophic pandemic and thus save the lives of many. We hope our project provides some assistance to the community even if it's in the smallest way possible.

REFERENCES

[1] N, V. R., & Patil, S. B. (2020). Indian Publications on SARSCoV-2: A bibliometric study of WHO COVID-19 database. Diabetes & metabolic syndrome, 14(5),1171–1178. https://doi.org/10.1016/j.dsx.2020.07.007

[2] Vignesh Radhakrishnan, Sumant Sen, Naresh Singaravelu Dissecting (2021). India's Second COVID-19 Wave. https://www.thehindu.com/data/dissecting-indias-second-covid-19wave/article34305418.ece

[3] Rucha Visal, Atharva Theurkar, Bhairavi Shukla "Monitoring Social Distancing for Covid-19 Using OpenCV and Deep Learning" Department of Information Technology, R.M.D Sinhgad School of Engineering, Warje, Pune, Maharashtra, India https://www.irjet.net/archives/V7/i6/IRJET-V7I6422.pdf

[4] M. Cristani, A. D. Bue, V. Murino, F. Setti and A. Vinciarelli, "The Visual Social Distancing Problem," in IEEE Access, vol. 8, pp. 126876126886, 2020, doi: 10.1109/ACCESS.2020.3008370.

[5] Yadav, Shashi. (2020). Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID-19 Safety Guidelines Adherence.

International Journal for Research in Applied Science and Engineering Technology. 8. 1368-1375. 10.22214/ijraset.2020.30560.

[6] Alcoran-Alvarez, Giselle Ann & Garcia, Marc & Alvarez, Dave. (2020). Automated Social Distancing Gate with Non-Contact Body Temperature Monitoring Using Arduino Uno.

[7] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning, 2016.

[8] Pre-trained Deep Neural Networks, MathWorks. https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Deep Residual Learning for Image Recognition. arXiv:1512.03385v1

[10] Ross Girshick. Fast R-CNN. arXiv:1504.08083v2

[11] You Only Look Once: Unified, Real-Time Object Detection. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi arXiv:1506.02640v5

[12] YOLO9000: Better, Faster, Stronger. Joseph Redmon, Ali Farhadi. arXiv:1612.08242v1

[13] YOLOv3: An Incremental Improvement. Joseph Redmon, Ali Farhadi arXiv:1804.02767v1