Real-Time Face Mask Detection System

A PROJECT REPORT

Submitted by

Thandayam Vamsi Govind (19MIC0056)

Harshita Sorout (19MIC0047)

MINI PROJECT

Under the guidance of

BHULAKSHMI BONTHU

Assistant Professor (Senior), SCOPE, VIT , Vellore.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April 2023

Table of Contents

ABSTRACT

In this project, we are aiming to make a model that detects whether the person is wearing a mask or not by monitoring them via a webcam. The foundation is laid upon training the model by two datasets with faces wearing masks and without face masks respectively and testing whether it classifies the image frames during video stream. This can be achieved by using TensorFlow library for pre-processing the frames obtained, use of dependencies on OpenCV library which is popular for face detection, and also using MobileNetV2 as an efficient classifier. And at last, we evaluate the model using appropriate validation techniques. Another perspective is that COVID-19, the virus which is infectious and contagious, has caused a pandemic early 2020, which still continues until today. Many medical personnel and scientists at WHO staunchly believe wearing masks can prevent contracting and transmitting the infection. This pandemic caused shutting down businesses both corporate and local businesses. The Indian Government is taking measures to reopen establishments to save the economy by putting forth rules and regulations that citizens must while in crowded areas which also includes wearing masks. Many businesses need to hire overseers to ensure that all customers wear masks while in the premises, so that protocol is followed. This might come into effect at huge cost for the owners. Hence, this project takes on the job of detecting whether a person is wearing a mask or not in a video stream. Two crucial components makeup this project; face detection and mask detection. It is achieved by effectively using concepts of Machine Learning, TensorFlow, Keras and MobilenetV2.

1. INTRODUCTION

Problem Statement Formation and Analysis

1.1 Problem Statement: To survive in this pandemic of coronavirus, strict measures have to be taken in order to restrict the spread and protect the people. Wearing a mask has always been a very important protocol in protecting ourselves from covid. Despite the downfall of cases, countries are already preparing for future waves that could possibly indicate newer variants with a larger number of people getting affected. Additionally, many states in India including Kerala, J&K, Assam, Jharkhand and modern Cities like Chennai have imposed compulsory wearing of masks indicating its paramount importance. So, in order to automate the detection and utilize the power of machine learning, a mask detector using a trained model would be very much necessary.

1.2 Motivation to take up this Problem Statement

The motivation behind this project is to ensure ease in management of the spread of diseases not only pertained to COVID-19 but according to WHO 80 other viral diseases can be erased if the person is wearing a mask. Statistically, it has been shown that wearing a N95 Mask can reduce the chance of getting COVID-19 by 83%, a surgical mask reduces the chances by 66% and a cloth mask by 56%, adding to our motivation to have a proper system that caters to this. As people move out from their homes and go to work, school or for any other purpose, the enforcement and monitoring of people wearing masks is a very erratic and tiring process, and hence we aim to solve this problem from a technical point of view.

1.2 Objective of this Project

1) Thoroughly understand the various positionings of the face mask on a person's face and how we can digitally analyze these positions to create an effective labeling
2) Incorporate the power of Live Video Feed to enhance the product, as video trumps images in terms of efficacy, throughput as well as provides a first-hand understanding of the working of the product.
3) Create a product that caters to the needs of those who will utilize a Real-Time Face Mask Detection System, by making it extremely user friendly and at the same time intricate with the usage of various technologies.
4) To develop a Machine Learning Model that acts as the pivot of the Entire Cumulative Model and Product, as it effectively and efficiently maximizes the accuracy of the classification, but also keeps in mind the restricted resources for an embedded application or an on-device.
5) On top of this, we aim to have our classifier that will be developed using a custom and intricate ML Algorithm to ensure that the real-feed video stream can be analyzed and returns a valid result.

2. LITERATURE SURVEY

[1] Goyal, K., Agarwal, K., & Kumar, R. (2017, April). Face detection and tracking: Using OpenCV. In *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)* (Vol. 1, pp. 474-478). IEEE.

The above paper provides a detailed study on Face attribute detection using various algorithms. Algorithms such as Haar Cascades, Cam Shift Algorithm and Finding via motion have been used to detect faces and the outputs of these algorithms have been measured based on time and space paradigms. Based on the outputs produced, it has been concluded that Haar Cascades provides accurate results in face detection in terms of all aspects. A brief comparative study has also been carried out comparing MATLAB and OpenCV based on attributes such as Speed, Portability and Cost. It has been concluded that OpenCV provides accurate and efficient results. All in all, an overall view of OpenCV and various algorithms used in detection of face attributes has been provided in the paper.

[2] Saxen, F., Werner, P., Handrich, S., Othman, E., Dinges, L., & Al-Hamadi, A. (2019, September). Face attribute detection with mobilenetv2 and nasnet-mobile. In *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)* (pp. 176-180). IEEE.

This work provides a detailed comparative study on various neural network architectures which are used to detect the face attributes using images and videos. The authors provide a proposal of using a method where two architectures namely MobileNetV2 and NasNet-Mobile.The mentioned architectures consist of a Convolutional Neural Network (CNN) which get dataset processed already from ResNet model. Similarly, all the other architectures such as LNets+ANet, MCNN-AUX etc have been tested on the same processed dataset and the accuracy values of the respective architectures have been calculated. Evidently, after performing the testing, it was found that MobileNetV2 and NasNet-Mobile provide the best results in determining the face attributes through frames.

[3] Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement, 167*, 108288.

This paper provides a brief overview on the methods used in machine learning to detect face masks on human faces. A measured study on the accuracy in detection of face masks using various algorithms has also been provided. The proposed method consists of two sections, the first section is about feature extraction of facemasks using large datasets. The second part consists of three classification processes namely Decision trees, Support Vector Machines and ensemble algorithm. After training the model with the mentioned procedure, testing was carried out on all the three classification methods, later it was found that the Support Vector Machine provided the most accurate results in detecting face mask attributes. The SVM classification method provided an accuracy of 99.64% in Real-World Masked Face Dataset (RMFD) which concluded that Support Vector Machine can provide the best accurate results in detecting face masks.

[4] Ding, Y., Li, Z., & Yastremsky, D. (2021). Real-time face mask detection in video data. *arXiv preprint arXiv:2105.01816*.

They demonstrated a powerful deep learning process that can distinguish between correctly and incorrectly wearing a mask from real-time video feeds. They developed two distinct strategies to achieve this aim and assessed the effectiveness and run-time efficiency of each. The first method makes use of a face detector that has already been trained in addition to an image classifier that has been trained on a sizable synthetic dataset while wearing a mask.The second method makes use of a cutting-edge network for object detection to localize and classify faces in a single pass using a limited number of tagged real-world photos.

[5] Nowrin, A., Afroz, S., Rahman, M. S., Mahmud, I., & Cho, Y. Z. (2021). Comprehensive review on facemask detection techniques in the context of covid-19. *IEEE access*.

They developed a potent deep learning technique that can tell if someone is wearing a mask correctly or wrongly using real-time video streams. To accomplish this goal, they created two unique techniques and evaluated the efficacy and run-time effectiveness of each. The first technique uses a face recognition system that has previously been trained along with an image classification that has been dressed in a mask and trained on a significant synthetic dataset. The second technique uses a small number of annotated real-world photographs with a state-of-the-art network for object identification to locate and identify features in a single pass. This article basically talks about the efficacy of the above mentioned techniques. Even while there has been a lot of research done on creating an effective facemask detection algorithm, it has mostly focused on the same set of challenges, leaving out several other very important ones. This research emphasized many flaws, including the need for category classifications, a lack of rich datasets, and retaining picture resolution during the detection process. Additionally, it listed the future horizons, which cover a variety of datasets and facemask kinds, as well as various facemask wearing scenarios and the restoration of the mask face, among other things.

[6] Liang, M., Gao, L., Cheng, C., Zhou, Q., Uy, J. P., Heiner, K., & Sun, C. (2020). Efficacy of face mask in preventing respiratory virus transmission: A systematic review and meta-analysis. *Travel medicine and infectious disease, 36*, 101751.

This work involves the main channels for the transmission of respiratory virus infection (RVIs) are contact and droplet pathways. Furthermore, recent studies have shown that the SARS-CoV-2 coronavirus may

survive and spread in aerosol for hours. Therefore, while avoiding the transmission of respiratory illnesses, the usage of masks as suitable personal protection equipment (PPE) is frequently recommended.
According to experimental data, dust or pathogens that really are larger in size of the mask's micropores are blocked by them. For instance, the N95 mask materials' 8 m-diameter micropores may successfully block the passage of viruses. The current research study and meta-analysis demonstrated the overall effectiveness of masks in stopping RVI transmission.

[7] Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet50 for medical face mask detection Mohamed Loey, Gunasekaran Manogaran, Mohamed Hamed N. Taha, Nour Eldeen M. Khalifa Sustainable Cities and Society, June 2020

In this study, a deep learning approach based on YOLOv2 detector and ResNet 50 is used. The medical masks dataset published by Mikolaj Witkowski has been used here, which consists of 682 pictures. The Resnet50 is used for feature extraction here and then the YOLOv2 CNN is used for classification using adam optimizer. It achieves an average precision of 81%.

[8]]SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2 Preeti Nagrath, Rachna Jain, Agam Madan, Rohan Arora, Piyush Kataria, and Jude Hemanth Sustain Cities Soc, December 2020

Here in SSDMNV2, MobileNetv2 is used as a framework for the classifier due to its lightweight nature due to which it can be used in embedded systems along with a Single Shot Multibox face detector to detect faces, which uses the ResNet10 architecture. The average accuracy of this model is 93% to classify whether a person is wearing a mask or not.

[9] Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement, 167*, 108288.

A face mask detector was created in this study employing SSD architecture and transfer learning methods in neural networks. A dataset of 1916 masked faces photos and 1919 unmasked faces photos was utilized to train, validate, and test the model. The MobileNetV2 Architecture was used to implement parameters like Accuracy, Precision, and Recall. It was 80-90 percent accurate overall

[10] Singh, S., Ahuja, U., Kumar, M., Kumar, K., & Sachdeva, M. (2021). Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment. *Multimedia Tools and Applications, 80*(13), 19753-19768.

OpenCV, tensor flow, Keras, PyTorch, and CNN were utilized in this work to determine whether or not persons were wearing face masks. Images and real-time video streams were used to test the models. The model's correctness is reached, but only to a limited extent, and model optimization is a constant process.

[11] Yadav, S. (2020). Deep learning based safe social distancing and face mask detection in public areas for covid-19 safety guidelines adherence. *Int. J. Res. Appl. Sci. Eng. Technol, 8*(7), 1368-1375.

A HaarCascade classifier is used to detect the input from the video cam while a CNN is used to recognise the pattern in the images and hence classify them. The dataset used has 1,315 images in which 658 images contain people with face masks and 657 images containing people without face masks. For training purposes, 80% images of each class are used and the rest of the images are utilized for testing purposes. The architecture contains three pairs of convolution layers each 32 followed by one max pooling layer. The convolution layer contains 100 kernels of window size 3x3. Max pooling layer of window size 2x2.

[12] Abboah-Offei, M., Salifu, Y., Adewale, B., Bayuo, J., Ofosu-Poku, R., & Opare-Lokko, E. B. A. (2021).
A rapid review of the use of face mask in preventing the spread of COVID-19. This paper discusses a rapid review to investigate the impact face mask use has had in controlling transmission of respiratory viral infections. The paper reviewed around 53 papers out of which 13 were systematic reviews and 45 were quantitative studies. The paper concluded that regardless of the type, setting, or who wears the face mask, wearing a face mask serves primarily a dual preventive purpose; protecting oneself from getting viral infection and protecting others. Therefore, if everyone wears a face mask in public, it offers a double barrier against COVID-19 transmission.

[13] Su, X., Gao, M., Ren, J., Li, Y., Dong, M., & Liu, X. (2022). Face mask detection and classification via deep transfer learning. *Multimedia Tools and Applications, 81*(3), 4475-4494.
This paper proposes a new algorithm for face mask detection that integrates transfer learning and Efficient-Yolov3, using EfficientNet as the backbone feature extraction network, and choosing CIoU as the loss function to reduce the number of network parameters and improve the accuracy of mask detection. Secondly, this paper divides the mask into two categories of qualified masks (N95 masks, disposable medical masks) and unqualified masks (cotton masks, sponge masks, scarves, etc.), creates a mask classification data set, and proposes a new mask classification algorithm that combines transfer learning and MobileNet, enhances the generalization of the model and solves the problem of small data size and easy overfitting. The overall performance of the transfer learning model is better and also consumes less time on training dataset.

[14] Song, Z., Nguyen, K., Nguyen, T., Cho, C., & Gao, J. (2021, August). Camera-Based Security Check for Face Mask Detection Using Deep Learning. In 2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService) (pp. 96-106). IEEE.

An innovative IoT-based face mask detection system for use in buses and other forms of public transportation is presented in this article. This technology would use facial recognition to gather data in real-time. The primary goal of the study is to use deep learning, machine learning, and image processing techniques to detect the existence of face masks in a real-time video stream. A hybrid deep and machine learning model was created and put into use to accomplish this goal. In addition to publicly available datasets, a novel dataset was used to evaluate the model. The results demonstrated that the Convolution Neural Network (CNN) classifier performs better than the Deep Neural Network (DNN) classifier; it also has an error rate of only 1.1 percent and almost complete face-identification capabilities with respect to people's presence when they are wearing masks. Overall, the suggested model performed better than the traditional models AlexNet, Mobinet, and You Only Look Once (YOLO).Additionally, the trials demonstrated that the suggested model can accurately detect faces and masks with little inference time or memory, satisfying the IoT's resource limitations.

[15] Bansal, A., Dhayal, S., Mishra, J., & Grover, J. (2022). COVID-19 Outbreak: Detecting face mask types in real time. *Journal of Information and Optimization Sciences*, *43*(2), 357-370.

One of the best non-pharmaceutical methods for halting the spread of infectious diseases is wearing a mask. Therefore, an automatic real-time mask detection and classification solution is urgently needed to help with the prevention of a public epidemic. Due to inadequate mask choice, the effectiveness of facemasks has been questioned. In jobs where there is a high risk of getting the virus, N95 masks must be worn. N95, surgical, and DIY masks are all effective to varying degrees. We developed a deep learning model that can categorize various kinds of masks and identify whether there is no mask in real-time video or photos in order to protect public safety. Our framework is divided into three sections: utilizing the object detection API, detect faces

3. TECHNICAL SPECIFICATIONS

Hardware Specifications:

- OS Name: Microsoft Windows
- Total Physical Memory: 16GB

Software Specifications:

Software Used:

- IDLE Python 3.7
- Language Used: Python 3
- Frameworks Used : Flask
- Libraries - TensorFlow

4. GAP ANALYSIS

1) Some systems use the ResNet50 architecture or the derivatives of it like the YOLOv2, this has its drawbacks because, ResNet-50 is a deep residual network and is 50 layers deep, it is computationally and heavy and not apt for deployment in real-time conditions.

2) A long-term impact of a lot of the existing work would be bias evaluation and terms like risk analysis which limit the utilization of the product in the real-world scenario. To cater to this, the dataset would have to be tweaked to ensure high accuracy and at the same time no demographic or color-based discrimination.

3) Some existing systems have concerns with Google Colab Notebook's excessive data loading latency when loading a dataset into it. Because restarting the runtime refreshes all of the cells, the cell for dataset loading takes the longest to run.

4) Many of the Existing works utilize ML Models that prevent webcam access, which makes it difficult to test photos and live video streams using these. As a result, the code must be run only when all system requirements are kept in sync with the person utilizing the program.

5) Most studies done already have only catered to either surgical, medical or plain masks and their various positionings- with mask, w/o mask, improper positioning, but we know that people have shifted to wearing N95 and Cloth Masks post the arrival of the Pandemic. Thus, the Accuracy of the model in these situations is skeptical and needs improvement.

5. DATASET COLLECTION

The image dataset is a crucial part of this project. Two folders named with_mask and without_mask consisting of 1915 and 1919 images respectively are collected from Real World Masked Face Dataset **[16]** and Labeled Faces in the Wild **[17]**. These datasets help in training the model for face mask detection.

With Mask Images:

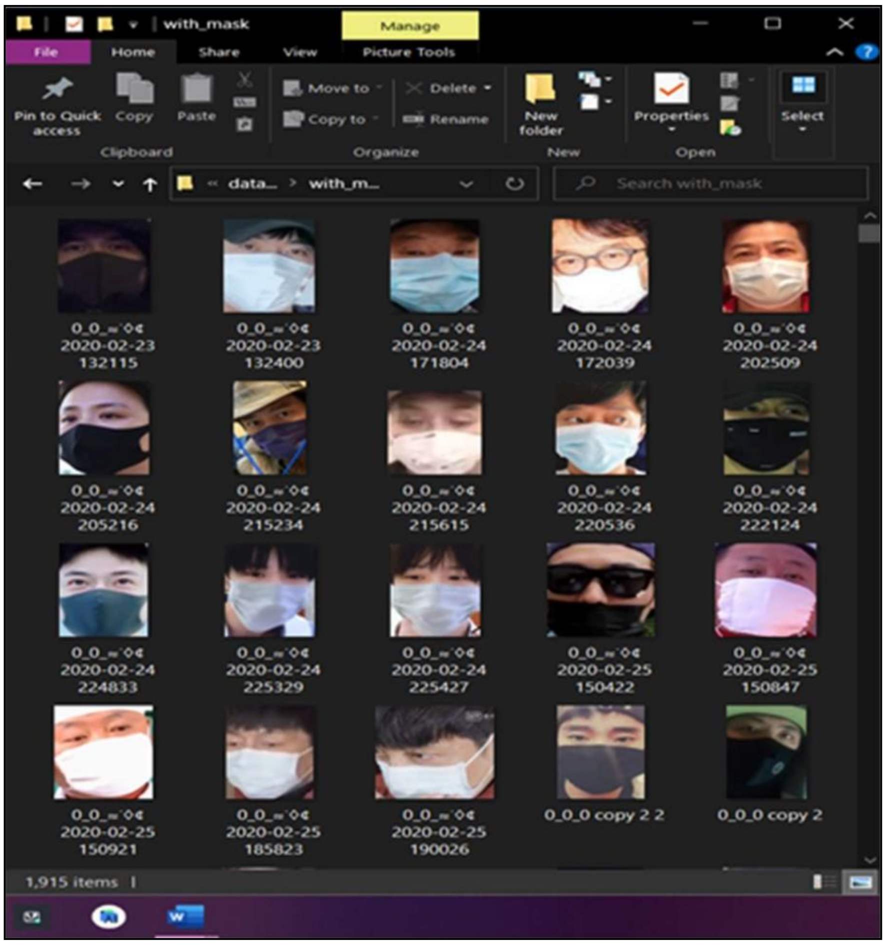


Figure 1: With Mask Images

Without Mask Images:

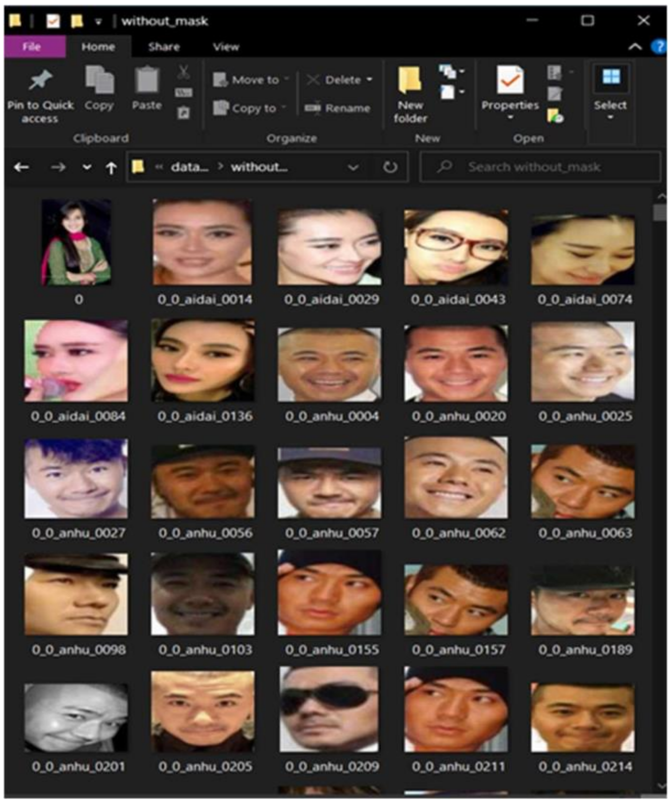


Figure 2: Without Mask Images

The two datasets are later categorized in the python script and image generator is used for creating more transformed images

6. ARCHITECTURE

The above figure is of a typical Mixed CNN architecture that is the core of our Model Architecture. It coupled with a MobileNet V2 Model to ensure a lightweight model, and helps in the implementation of Data pre-processing, which includes the steps of:

- Transforming
- Encode the labels
- Train and test splitting of the data
- Using image generator to increase data

After this there is a Max-Pooling Process, that helps in identifying the brighter pixels from an image that might be located in a dark background, followed by a Flattening process and then sending the output.

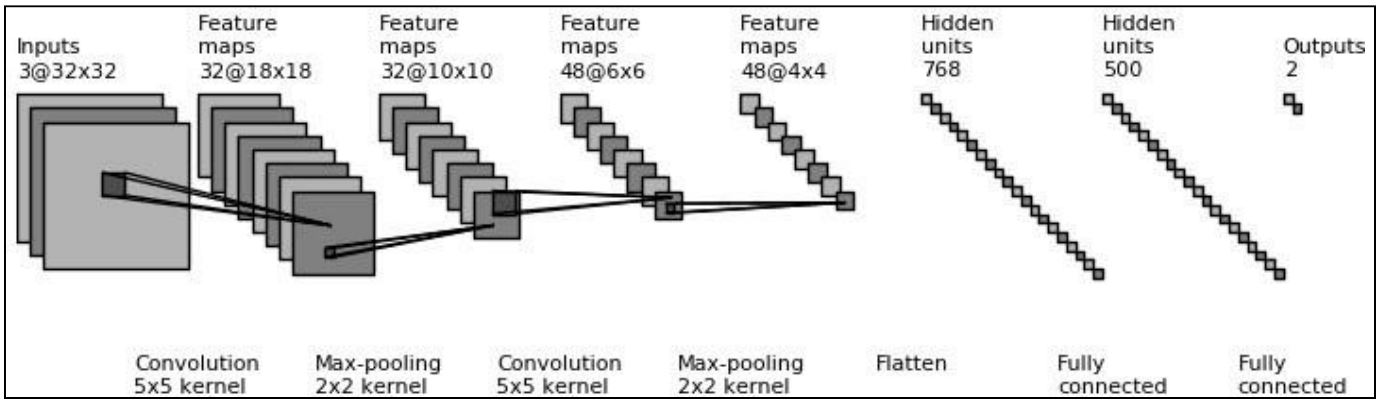
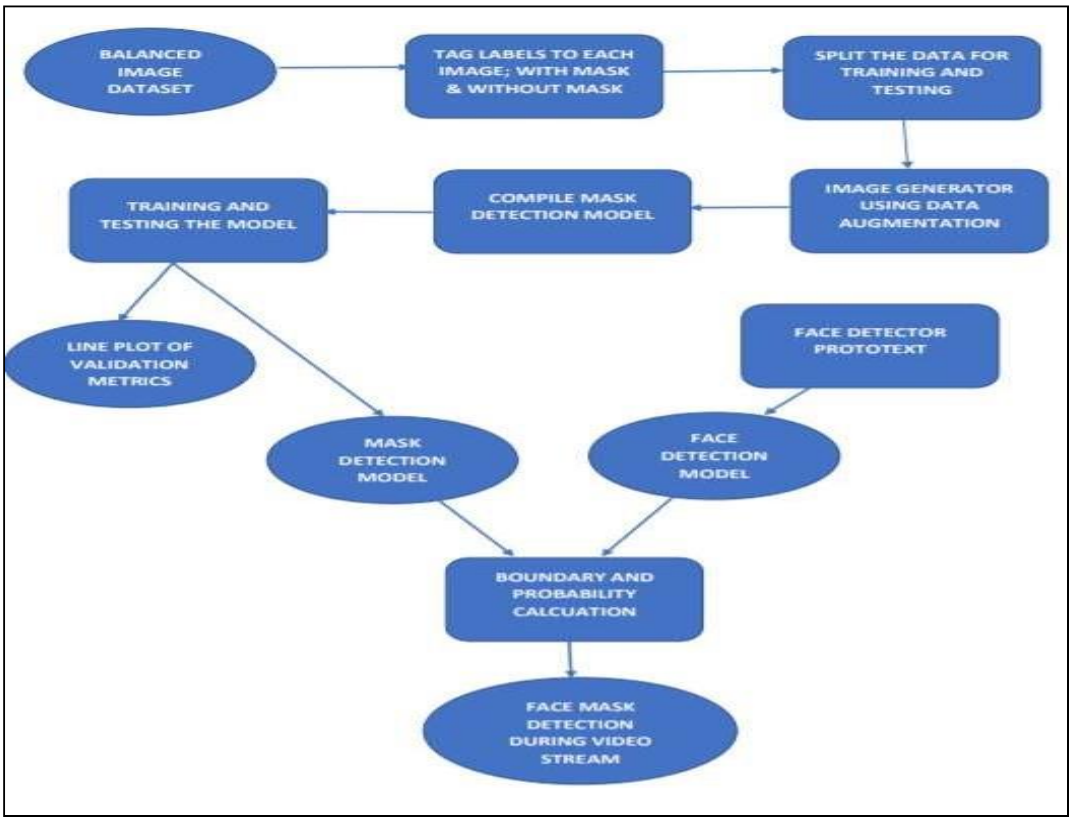


Figure 3: Architecture Diagram

7. PROPOSED SYSTEM

This project uses Convolutional Neural Network (CNN) as its main architecture to detect and classify the images according to the trained model. The Comprehensive Flowchart is as follows:



BALANCED IMAGE DATASET
TAG LABELS TO EACH IMAGE; WITH MASK & WITHOUT MASK
SPLIT THE DATA FOR TRAINING AND TESTING
IMAGE GENERATOR USING DATA AUGMENTATION
COMPILE MASK DETECTION MODEL
TRAINING AND TESTING THE MODEL
LINE PLOT OF VALIDATION METRICS
FACE DETECTOR PROTOTEXT
MASK DETECTION MODEL
FACE DETECTION MODEL
BOUNDARY AND PROBABILITY CALCUATION
FACE MASK DETECTION DURING VIDEO STREAM

Figure 4: Flowchart Depicting the process of the System

Functions Used

- MobileNetV2
- Average Pooling
- Flatten
- Dense Dropout

i) Adam Optimization Algorithm

The word is derived from Adaptive moment Estimation. The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. The algorithm calculates an exponential moving average of the gradient and the squared gradient values of the input the dataset attributes. It is derived from Adaptive Generic Algorithm and Root mean Square algorithm.

ii) Average Pooling

Average Pooling is a pooling operation that calculates the average value for patches of a feature map, and uses it to create a down sampled (pooled) feature map. It is usually used after a convolutional layer. It adds a small amount of translation invariance - meaning translating the image by a small amount does not significantly affect the values of most pooled outputs. It extracts features more smoothly than Max Pooling, whereas max pooling extracts more pronounced features like edges.

Example:

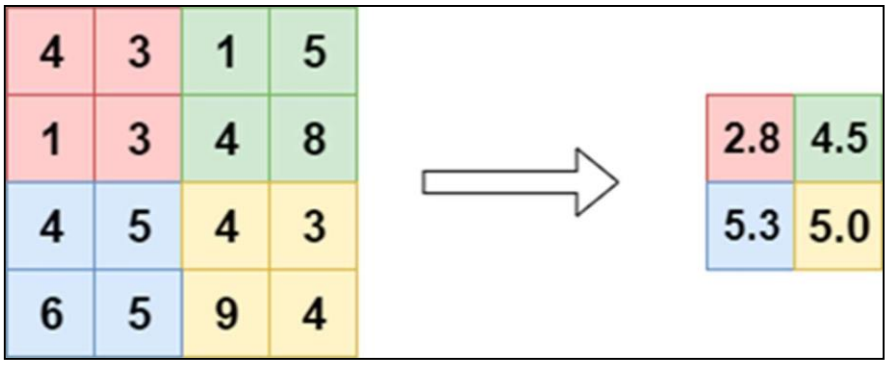


Figure 5: Example of Average Pooling Function

iii) Flatten Layer

It is used to convert the data into 1D arrays to create a single feature vector. After flattening we forward the data to a fully connected layer for final classification.

Example:

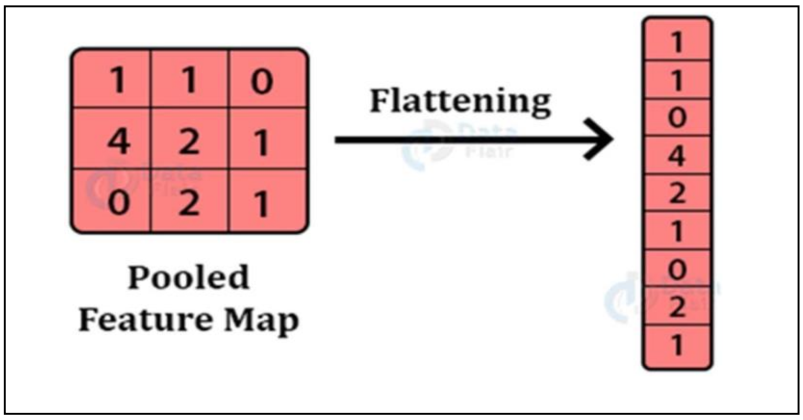


Figure 6: Example of Flatten Layer Function

iv) Dense Layer

It is a fully connected layer. Each node in this layer is connected to the previous layer i.e densely connected. This layer is used at the final stage of CNN to perform classification.

Example:

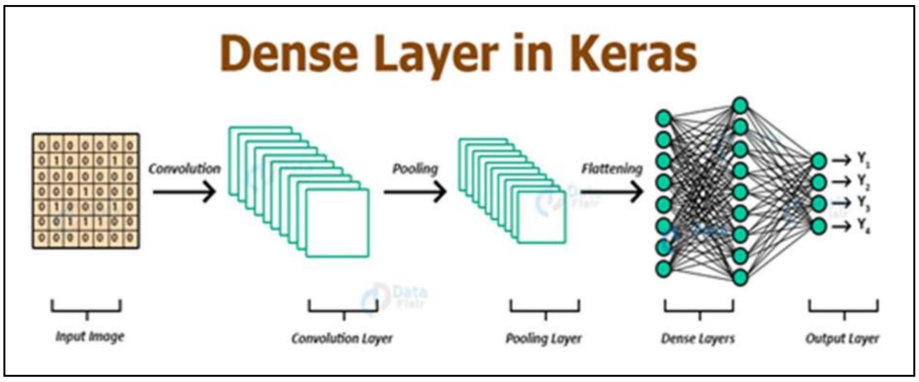


Figure 7: Example of Dense Layer Function in Keras

v) Dropout Layer

It is used to prevent the network from overfitting. In this layer, some fraction of units in the network is dropped in training such that the model is trained on all the units.

Example:

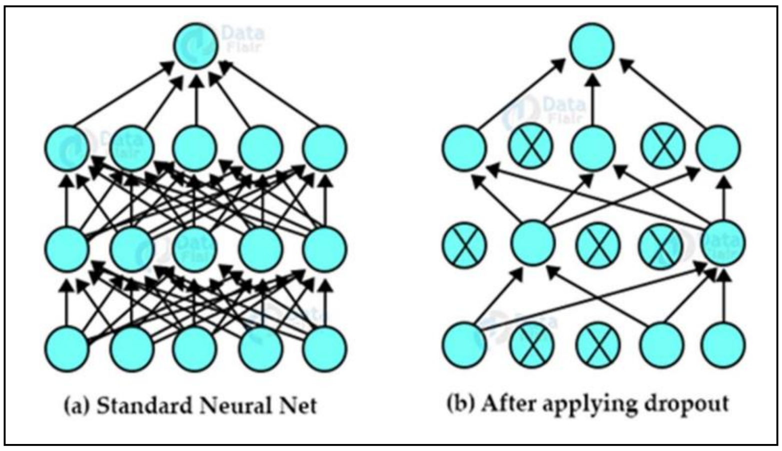


Figure 8: Example of Dropout Layer Function

Modules Used with Explanation

Module-1: Mask Detection

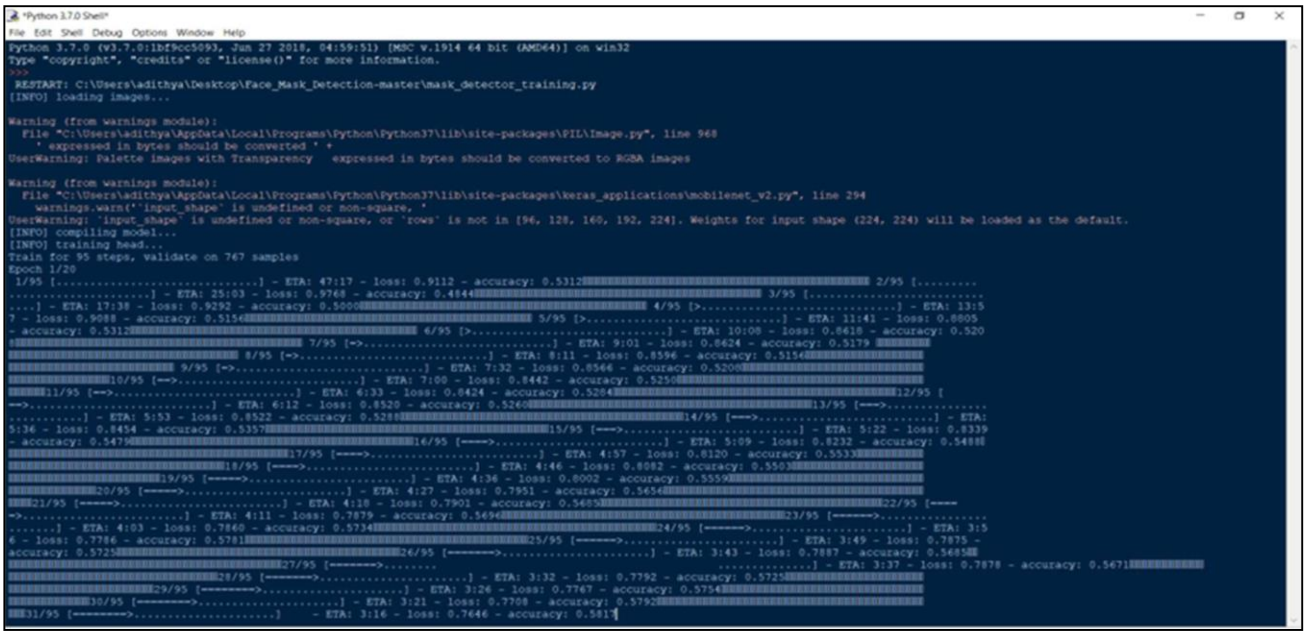
Implementation:

In this module, we basically are loading the image collected from the folder as shown in the images section below then categorizing them into with_mask and without_mask. Later a base model is created upon an ImageGenerator and freezes all the layers. Now, the mask detector model is compiled using Adam optimizer. Training of the model is done using the train data via fit method. Now, it is ready for predicting the test data whose results are printed in the form of classification validation metrics like precision, recall, f1-score and support. The model is saved and a line graph of the trend of validation measures according to the number of Epochs is plotted.

Procedure:

1. Import all the required libraries
2. Initialize number of epochs, batch size and learning rate.
3. Make a looping structure to load all the input images and append them to a data list and accordingly append the category that image belongs to; with mask or without mask in a label list.
4. Fit transforms the labels into a LabelBinarizer and makes them categories.
5. Load the MobileNetV2 network as base model.
6. Then make head over the base model output using AveragePooling2D, Dropout, Flatten and Dense.
7. Now, make a model using Model where input is base model and output is head model.
8. Using Adam Optimizer, compile the model with metrics as accuracy.
9. Train the head of model the by fitting augmented train data set according the batch size.
10. Predict the test data using the model which predicts the label of the image whether with a mask or without a mask.
11. Print the classification report
12. Save the model as maskdetector.model in h5 format.

Demonstrations:

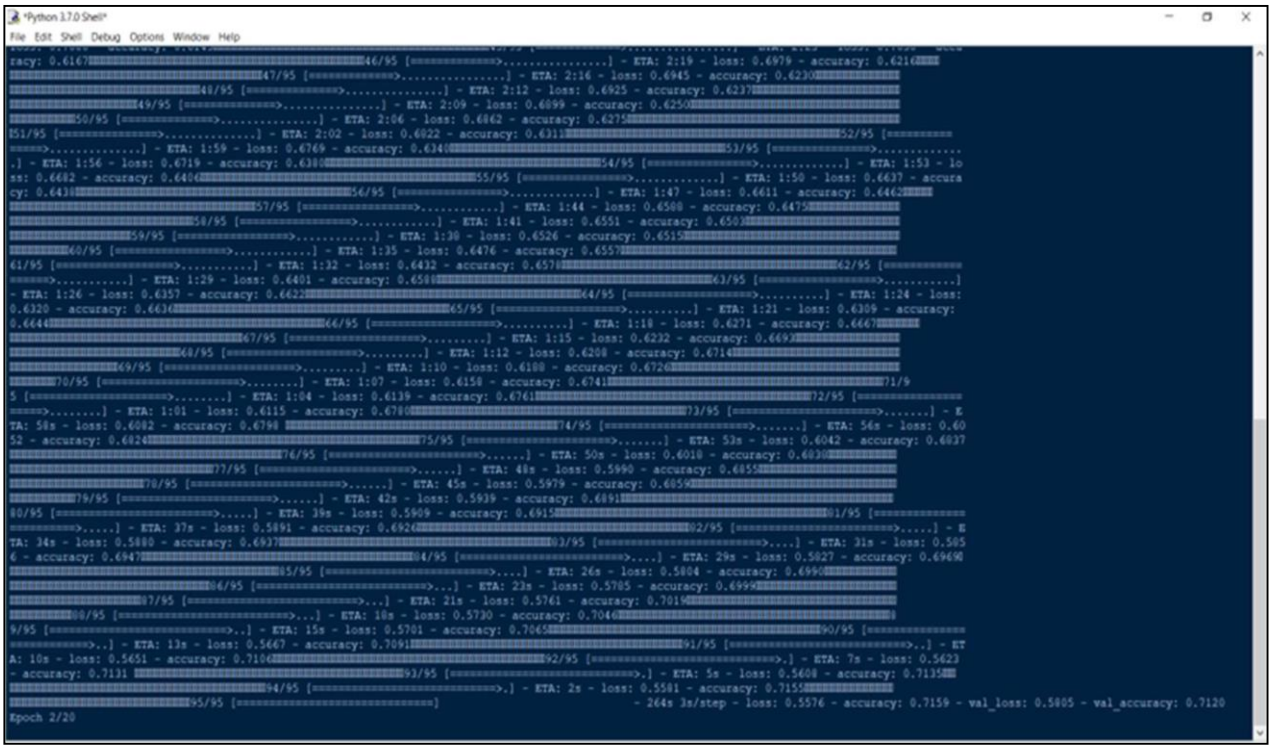


```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
 RESTART: C:\Users\adithya\Desktop\Face_Mask_Detection-master\mask_detector_training.py
[INFO] loading images...

Warning (from warnings module):
  File "C:\Users\adithya\AppData\Local\Programs\Python\Python37\lib\site-packages\PIL\Image.py", line 968
    ' expressed in bytes should be converted ' +
UserWarning: Palette images with Transparency   expressed in bytes should be converted to RGBA images

Warning (from warnings module):
  File "C:\Users\adithya\AppData\Local\Programs\Python\Python37\lib\site-packages\keras_applications\mobilenet_v2.py", line 294
    warnings.warn('`input_shape` is undefined or non-square, '
UserWarning: `input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
[INFO] compiling model...
[INFO] training head...
Train for 95 steps, validate on 767 samples
Epoch 1/20
 1/95 [..............................] - ETA: 47:17 - loss: 0.9112 - accuracy: 0.5312▯▯▯▯▯▯▯▯▯▯▯▯ 2/95 [.........
.....................] - ETA: 25:03 - loss: 0.9768 - accuracy: 0.4844▯▯▯▯▯▯▯▯▯▯▯▯ 3/95 [..........................
....] - ETA: 17:38 - loss: 0.9292 - accuracy: 0.5000▯▯▯▯▯▯▯▯▯▯▯▯ 4/95 [>.............................] - ETA: 13:5
7 - loss: 0.9088 - accuracy: 0.5156▯▯▯▯▯▯▯▯▯▯▯▯ 5/95 [>.............................] - ETA: 11:41 - loss: 0.8805
- accuracy: 0.5312▯▯▯▯▯▯▯▯▯▯▯▯ 6/95 [>.............................] - ETA: 10:08 - loss: 0.8618 - accuracy: 0.520
8▯▯▯▯▯▯▯▯▯▯▯▯ 7/95 [=>............................] - ETA: 9:01 - loss: 0.8624 - accuracy: 0.5179 ▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯ 8/95 [=>............................] - ETA: 8:11 - loss: 0.8596 - accuracy: 0.5156▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯ 9/95 [=>............................] - ETA: 7:32 - loss: 0.8566 - accuracy: 0.5208▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯10/95 [==>...........................] - ETA: 7:00 - loss: 0.8442 - accuracy: 0.5250▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯11/95 [==>...........................] - ETA: 6:33 - loss: 0.8424 - accuracy: 0.5284▯▯▯▯▯▯▯▯▯▯▯▯12/95 [
==>...........................] - ETA: 6:12 - loss: 0.8520 - accuracy: 0.5260▯▯▯▯▯▯▯▯▯▯▯▯13/95 [===>..............
...........] - ETA: 5:53 - loss: 0.8522 - accuracy: 0.5288▯▯▯▯▯▯▯▯▯▯▯▯14/95 [===>..........................] - ETA:
5:36 - loss: 0.8454 - accuracy: 0.5357▯▯▯▯▯▯▯▯▯▯▯▯15/95 [===>..........................] - ETA: 5:22 - loss: 0.8339
- accuracy: 0.5479▯▯▯▯▯▯▯▯▯▯▯▯16/95 [====>.........................] - ETA: 5:09 - loss: 0.8232 - accuracy: 0.5488
▯▯▯▯▯▯▯▯▯▯▯▯17/95 [====>.........................] - ETA: 4:57 - loss: 0.8120 - accuracy: 0.5533▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯18/95 [====>.........................] - ETA: 4:46 - loss: 0.8082 - accuracy: 0.5503▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯19/95 [=====>........................] - ETA: 4:36 - loss: 0.8002 - accuracy: 0.5559▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯20/95 [=====>........................] - ETA: 4:27 - loss: 0.7951 - accuracy: 0.5656▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯21/95 [=====>........................] - ETA: 4:18 - loss: 0.7901 - accuracy: 0.5685▯▯▯▯▯▯▯▯▯▯▯▯22/95 [====
=>........................] - ETA: 4:11 - loss: 0.7879 - accuracy: 0.5696▯▯▯▯▯▯▯▯▯▯▯▯23/95 [======>...............
.......] - ETA: 4:03 - loss: 0.7860 - accuracy: 0.5734▯▯▯▯▯▯▯▯▯▯▯▯24/95 [======>.......................] - ETA: 3:5
6 - loss: 0.7786 - accuracy: 0.5781▯▯▯▯▯▯▯▯▯▯▯▯25/95 [======>.......................] - ETA: 3:49 - loss: 0.7875 -
accuracy: 0.5725▯▯▯▯▯▯▯▯▯▯▯▯26/95 [=======>......................] - ETA: 3:43 - loss: 0.7887 - accuracy: 0.5685▯
▯▯▯▯▯▯▯▯▯▯▯▯27/95 [=======>.........                   ...............] - ETA: 3:37 - loss: 0.7878 - accuracy: 0.5671▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯28/95 [========>.....................] - ETA: 3:32 - loss: 0.7792 - accuracy: 0.5725▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯29/95 [========>.....................] - ETA: 3:26 - loss: 0.7767 - accuracy: 0.5754▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯30/95 [========>.....................] - ETA: 3:21 - loss: 0.7708 - accuracy: 0.5792▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯31/95 [=========>....................]     - ETA: 3:16 - loss: 0.7646 - accuracy: 0.5817
```

Figure 9: Start of Training Model through epoch



```
racy: 0.6167▯▯▯▯▯▯▯▯▯▯▯▯46/95 [==============>...............] - ETA: 2:19 - loss: 0.6979 - accuracy: 0.6216▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯47/95 [==============>...............] - ETA: 2:16 - loss: 0.6945 - accuracy: 0.6230▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯48/95 [===============>..............] - ETA: 2:12 - loss: 0.6925 - accuracy: 0.6237▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯49/95 [===============>..............] - ETA: 2:09 - loss: 0.6899 - accuracy: 0.6250▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯50/95 [===============>..............] - ETA: 2:06 - loss: 0.6862 - accuracy: 0.6275▯▯▯▯▯▯▯▯▯▯▯▯
▯51/95 [================>.............] - ETA: 2:02 - loss: 0.6822 - accuracy: 0.6311▯▯▯▯▯▯▯▯▯▯▯▯52/95 [==========
=====>.............] - ETA: 1:59 - loss: 0.6769 - accuracy: 0.6340▯▯▯▯▯▯▯▯▯▯▯▯53/95 [================>.............
.] - ETA: 1:56 - loss: 0.6719 - accuracy: 0.6380▯▯▯▯▯▯▯▯▯▯▯▯54/95 [=================>............] - ETA: 1:53 - lo
ss: 0.6682 - accuracy: 0.6406▯▯▯▯▯▯▯▯▯▯▯▯55/95 [=================>............] - ETA: 1:50 - loss: 0.6637 - accura
cy: 0.6438▯▯▯▯▯▯▯▯▯▯▯▯56/95 [=================>............] - ETA: 1:47 - loss: 0.6611 - accuracy: 0.6462▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯57/95 [==================>...........] - ETA: 1:44 - loss: 0.6588 - accuracy: 0.6475▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯58/95 [==================>...........] - ETA: 1:41 - loss: 0.6551 - accuracy: 0.6503▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯59/95 [==================>...........] - ETA: 1:38 - loss: 0.6526 - accuracy: 0.6515▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯60/95 [===================>..........] - ETA: 1:35 - loss: 0.6476 - accuracy: 0.6557▯▯▯▯▯▯▯▯▯▯▯▯
61/95 [===================>..........] - ETA: 1:32 - loss: 0.6432 - accuracy: 0.6578▯▯▯▯▯▯▯▯▯▯▯▯62/95 [============
======>..........] - ETA: 1:29 - loss: 0.6401 - accuracy: 0.6588▯▯▯▯▯▯▯▯▯▯▯▯63/95 [===================>..........]
- ETA: 1:26 - loss: 0.6357 - accuracy: 0.6622▯▯▯▯▯▯▯▯▯▯▯▯64/95 [====================>.........] - ETA: 1:24 - loss:
0.6320 - accuracy: 0.6636▯▯▯▯▯▯▯▯▯▯▯▯65/95 [====================>.........] - ETA: 1:21 - loss: 0.6309 - accuracy:
0.6644▯▯▯▯▯▯▯▯▯▯▯▯66/95 [====================>.........] - ETA: 1:18 - loss: 0.6271 - accuracy: 0.6667▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯▯67/95 [=====================>........] - ETA: 1:15 - loss: 0.6232 - accuracy: 0.6693▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯68/95 [=====================>........] - ETA: 1:12 - loss: 0.6208 - accuracy: 0.6714▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯69/95 [=====================>........] - ETA: 1:10 - loss: 0.6188 - accuracy: 0.6726▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯70/95 [======================>.......] - ETA: 1:07 - loss: 0.6158 - accuracy: 0.6741▯▯▯▯▯▯▯▯▯▯▯▯71/9
5 [======================>.......] - ETA: 1:04 - loss: 0.6139 - accuracy: 0.6761▯▯▯▯▯▯▯▯▯▯▯▯72/95 [=================
=====>.......] - ETA: 1:01 - loss: 0.6115 - accuracy: 0.6780▯▯▯▯▯▯▯▯▯▯▯▯73/95 [=======================>......] - E
TA: 58s - loss: 0.6082 - accuracy: 0.6798 ▯▯▯▯▯▯▯▯▯▯▯▯74/95 [=======================>......] - ETA: 56s - loss: 0.60
52 - accuracy: 0.6824▯▯▯▯▯▯▯▯▯▯▯▯75/95 [=======================>......] - ETA: 53s - loss: 0.6042 - accuracy: 0.6837
▯▯▯▯▯▯▯▯▯▯▯▯76/95 [========================>.....] - ETA: 50s - loss: 0.6018 - accuracy: 0.6838▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯77/95 [========================>.....] - ETA: 48s - loss: 0.5990 - accuracy: 0.6855▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯78/95 [========================>.....] - ETA: 45s - loss: 0.5979 - accuracy: 0.6859▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯79/95 [=========================>....] - ETA: 42s - loss: 0.5939 - accuracy: 0.6891▯▯▯▯▯▯▯▯▯▯▯▯
80/95 [=========================>....] - ETA: 39s - loss: 0.5909 - accuracy: 0.6915▯▯▯▯▯▯▯▯▯▯▯▯81/95 [==============
===========>.....] - ETA: 37s - loss: 0.5891 - accuracy: 0.6926▯▯▯▯▯▯▯▯▯▯▯▯82/95 [=========================>.....] - E
TA: 34s - loss: 0.5880 - accuracy: 0.6937▯▯▯▯▯▯▯▯▯▯▯▯83/95 [==========================>....] - ETA: 31s - loss: 0.585
6 - accuracy: 0.6947▯▯▯▯▯▯▯▯▯▯▯▯84/95 [==========================>....] - ETA: 29s - loss: 0.5827 - accuracy: 0.6969
▯▯▯▯▯▯▯▯▯▯▯▯85/95 [==========================>....] - ETA: 26s - loss: 0.5804 - accuracy: 0.6990▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯86/95 [===========================>...] - ETA: 23s - loss: 0.5785 - accuracy: 0.6999▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯87/95 [===========================>...] - ETA: 21s - loss: 0.5761 - accuracy: 0.7019▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯88/95 [===========================>...] - ETA: 18s - loss: 0.5730 - accuracy: 0.7046▯▯▯▯▯▯▯▯▯▯▯▯8
9/95 [============================>..] - ETA: 15s - loss: 0.5701 - accuracy: 0.7065▯▯▯▯▯▯▯▯▯▯▯▯90/95 [==============
==============>..] - ETA: 13s - loss: 0.5667 - accuracy: 0.7091▯▯▯▯▯▯▯▯▯▯▯▯91/95 [============================>..] - ET
A: 10s - loss: 0.5651 - accuracy: 0.7106▯▯▯▯▯▯▯▯▯▯▯▯92/95 [=============================>.] - ETA: 7s - loss: 0.5623
- accuracy: 0.7131 ▯▯▯▯▯▯▯▯▯▯▯▯93/95 [=============================>.] - ETA: 5s - loss: 0.5608 - accuracy: 0.7135▯▯
▯▯▯▯▯▯▯▯▯▯▯▯94/95 [=============================>.] - ETA: 2s - loss: 0.5581 - accuracy: 0.7155▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯95/95 [==============================]                        - 264s 3s/step - loss: 0.5576 - accuracy: 0.7159 - val_loss: 0.5805 - val_accuracy: 0.7120
Epoch 2/20
```

Figure 10: End of Training Model through epoch

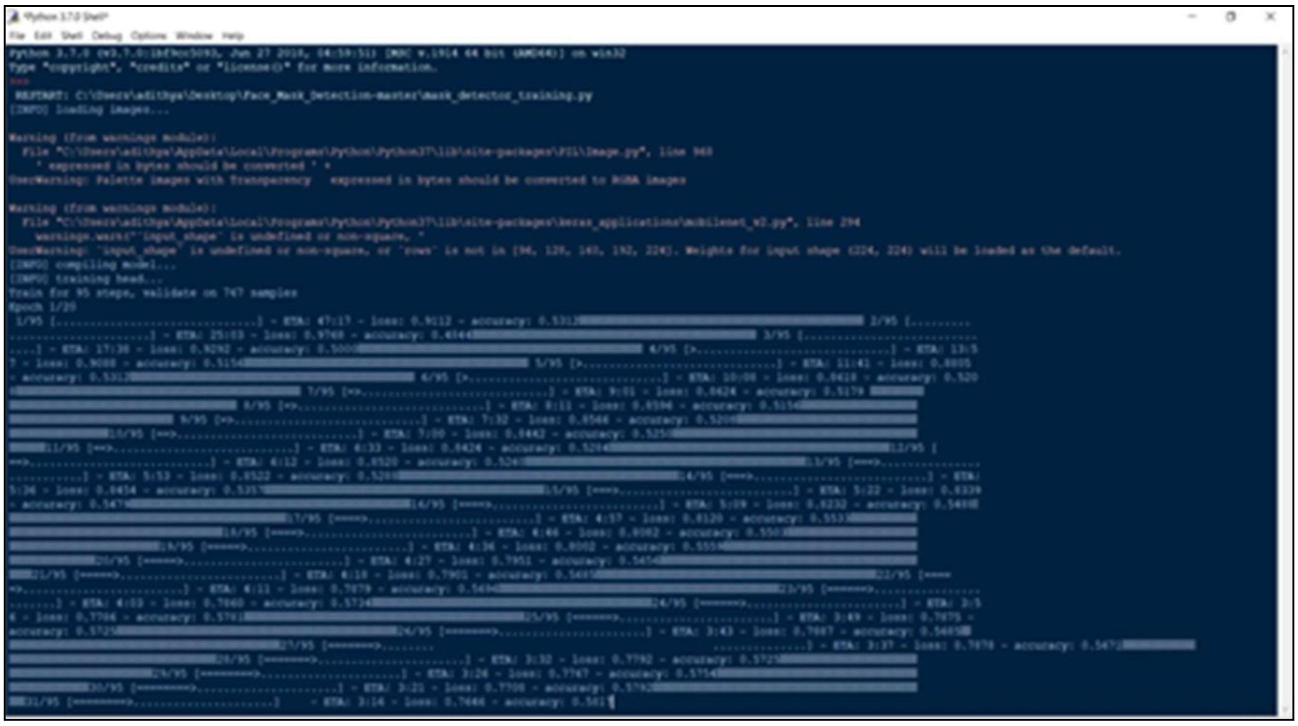
<u>Module-2 : Face Detection</u>

Implementation:
OpenCV is a library used for computer vision and also supports deep learning frameworks like caffe, TensorFlow, torch. Using OpenCV we can perform a trained deep learning face detection model. In this project we used OpenCV for face detection.

Procedure:

1. Face acknowledgment takes a picture from a video or a camera as info and yields the recognized picture point.
2. Facial highlights may remember regions of the face, varieties in the face structure, face cuts and points that have been formatted and styled.
3. Face extraction incorporates getting highlights from the camera.
4. Face identification remembers evacuation of the foundation and center for the frontal area dispensing with some other components separated from the face region.
5. Here we have prototxt file containing architecture model of OpenCV face detection which executed gives caffe model file which has the weights of the actual layers. Both files are required while training caffe model and face mask Detection.

Demonstration



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\adithya\Desktop\Face_Mask_Detection-master\mask_detector_training.py
[INFO] loading images...

Warning (from warnings module):
  File "C:\Users\adithya\AppData\Local\Programs\Python\Python37\lib\site-packages\PIL\Image.py", line 960
    " expressed in bytes should be converted " +
UserWarning: Palette images with Transparency   expressed in bytes should be converted to RGBA images

Warning (from warnings module):
  File "C:\Users\adithya\AppData\Local\Programs\Python\Python37\lib\site-packages\keras_applications\mobilenet_v2.py", line 294
    warnings.warn('`input_shape` is undefined or non-square, '
UserWarning: `input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
[INFO] compiling model...
[INFO] training head...
Train for 95 steps, validate on 747 samples
Epoch 1/20
 1/95 [..............................] - ETA: 47:17 - loss: 0.9112 - accuracy: 0.5312
[illegible]
```

Figure 11: Training of the Model

Module-3: Face Mask Detection in Video Stream

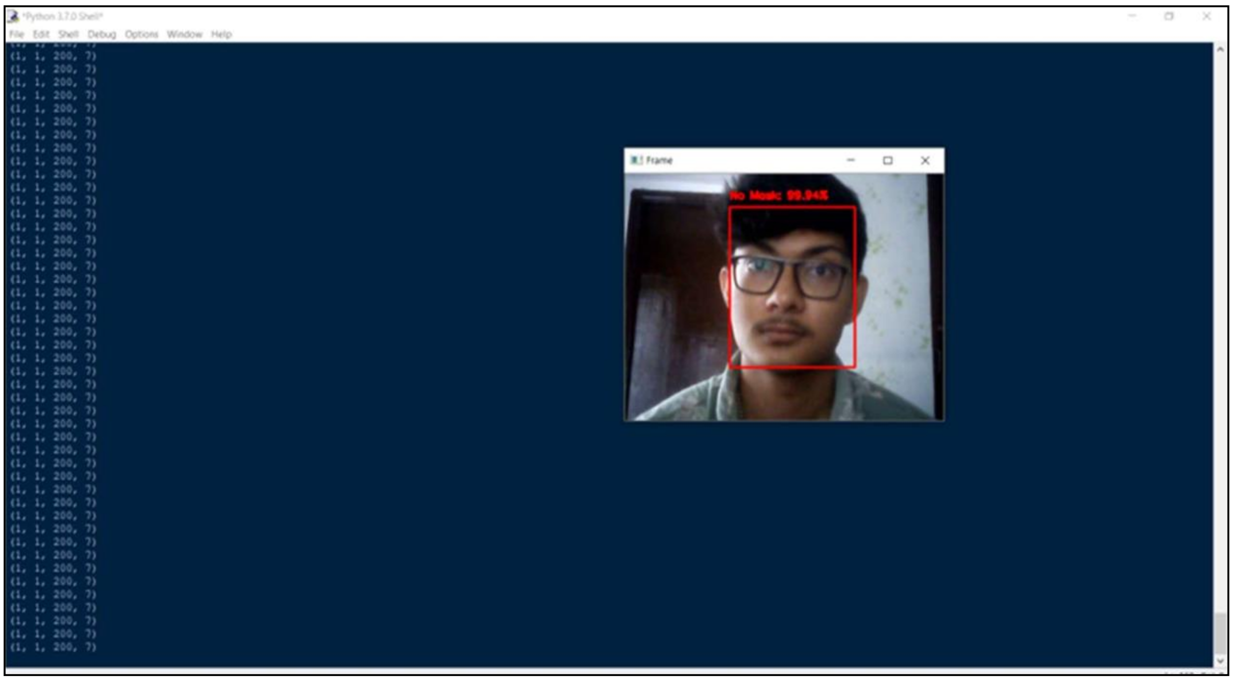
Implementation:

We are loading the face detection and mask detection models using os.path. Then starting live camera video stream, where each frame is sent to a function that calculates the box boundary coordinates and prediction of mask detection which is then printed upon the boundary window enclosing the face. The boundary window highlights in red if they are not wearing mask else highlights green if they are wearing a mask.

Procedure:

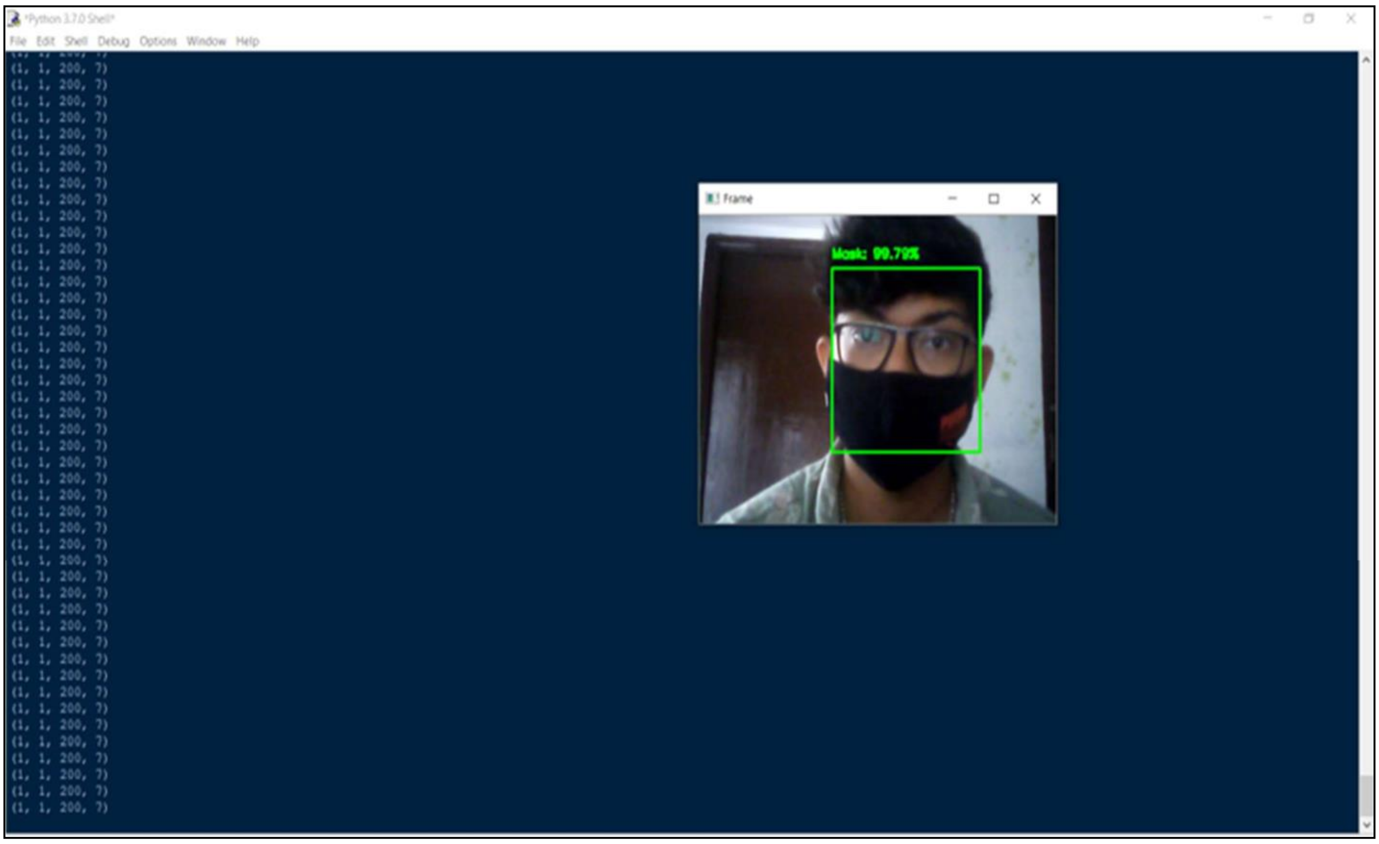
1. Importing required libraries and dependencies.
2. Load OpenCV face detector caffemodel as DNN model
3. Load the maskdetector.model.
4. Start a video stream.
5. Initialize a loop and get the box boundary coordinates for face and prediction of mask detection.
6. Get prediction rate for with mask and without mask.
7. If rate for with mask is greater than that of the without mask, then boundary box color must change to green while printing the percentage of Mask.
8. Else, boundary box colour must change to red while printing percentage of No Mask.
9. If the 'c' key is pressed, stop the video stream.

Demonstration



```
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
```

Figure 12: Without Mask Input



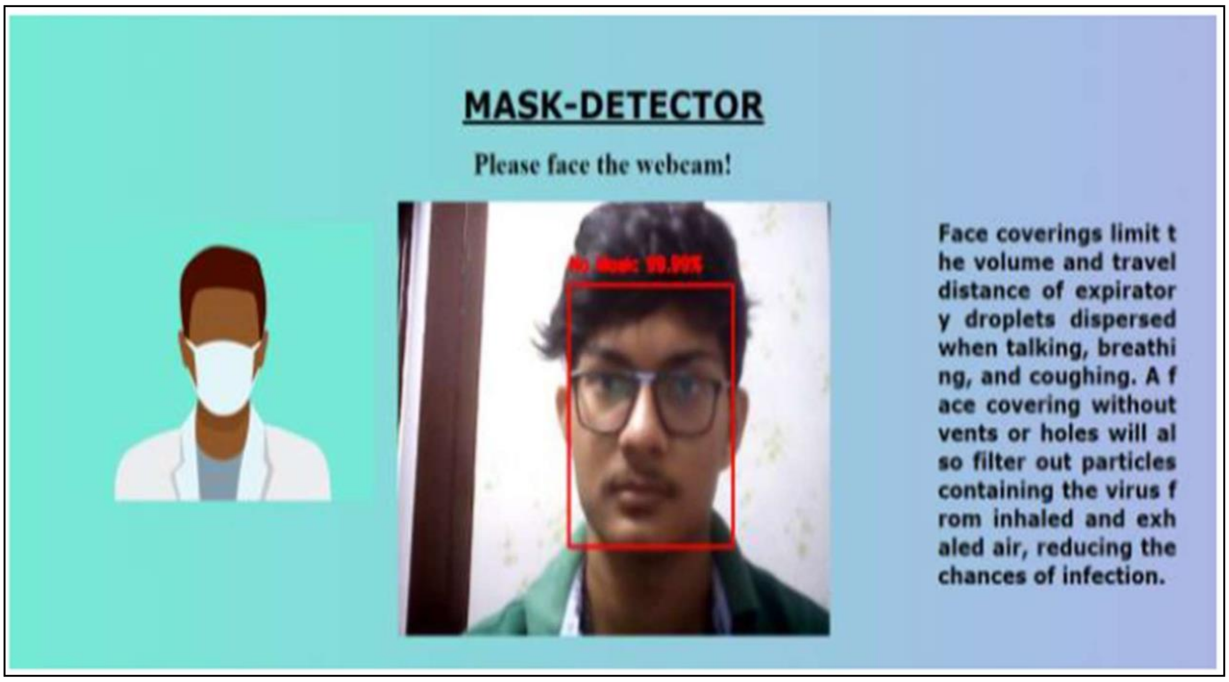
```
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
```

Figure 13: With Mask Input

8. ANALYSIS REPORT AND RESULTS

1) Without Mask Output from the Website

The Below Image shows the Model running on the Flask-Based Website, and gives the output for when the person is not wearing the mask.

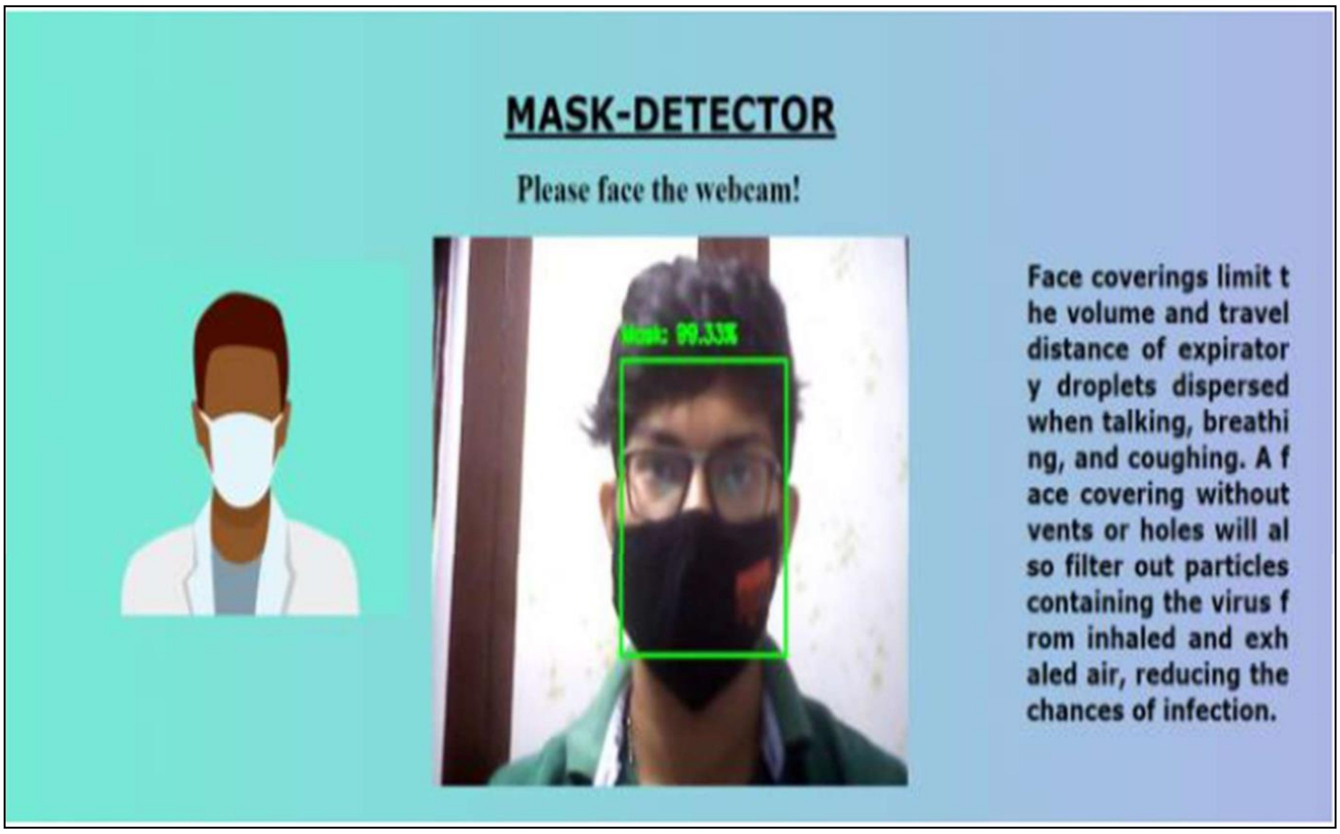


Face coverings limit the volume and travel distance of expiratory droplets dispersed when talking, breathing, and coughing. A face covering without vents or holes will also filter out particles containing the virus from inhaled and exhaled air, reducing the chances of infection.

Figure 14: Without Mask Input from Website

2) With Mask Output from the Website

The Below Image shows the Model running on the Flask-Based Website, and gives the output for when the person is wearing the mask.



MASK-DETECTOR

Please face the webcam!

Face coverings limit t
he volume and travel
distance of expirator
y droplets dispersed
when talking, breathi
ng, and coughing. A f
ace covering without
vents or holes will al
so filter out particles
containing the virus f
rom inhaled and exh
aled air, reducing the
chances of infection.

Figure 15: With Mask Input from Website

3) Training Loss and Accuracy Graph

The Below Image depicts the Training Loss as well as the Accuracy Loss in the form of a line graph. Apart from these 2 line graphs, we also have the val_loss and val_acc parameters which are standard in Keras and depict if the model is preventing overfitting and the performance of the model on validation data respectively.

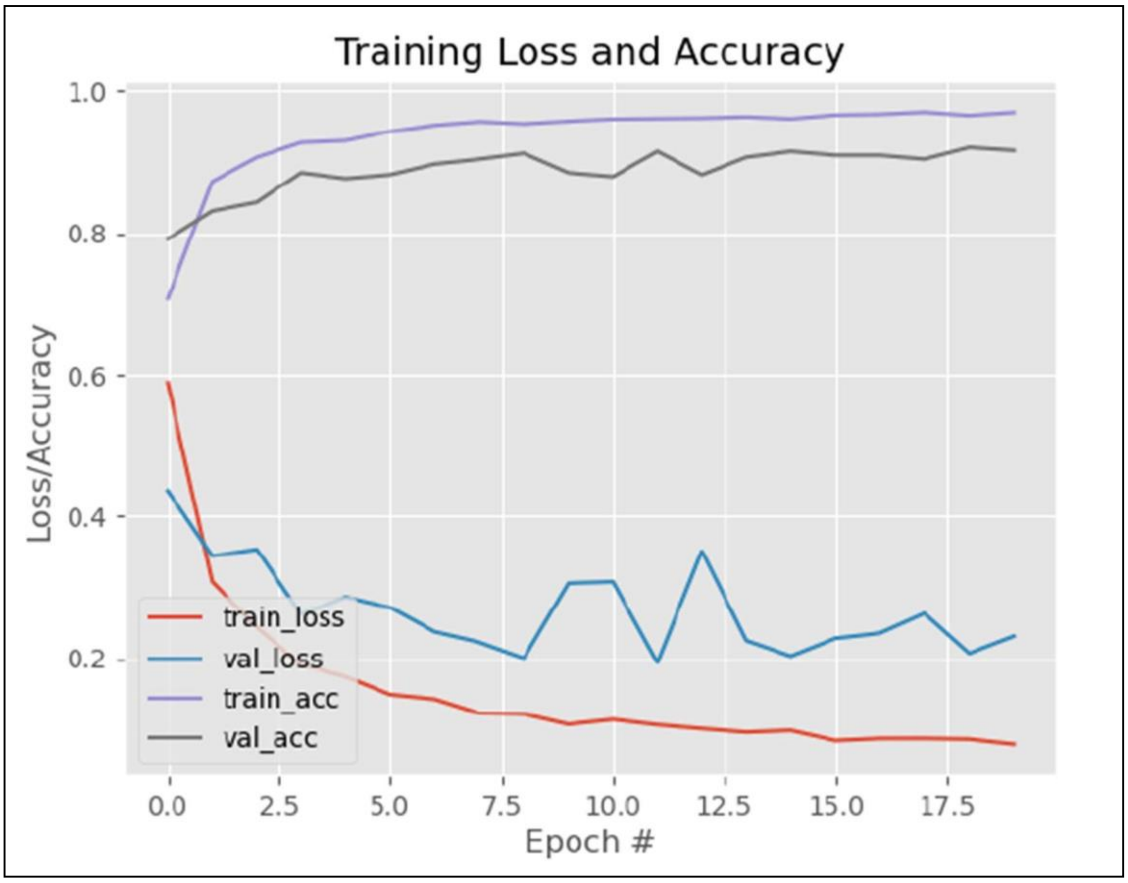
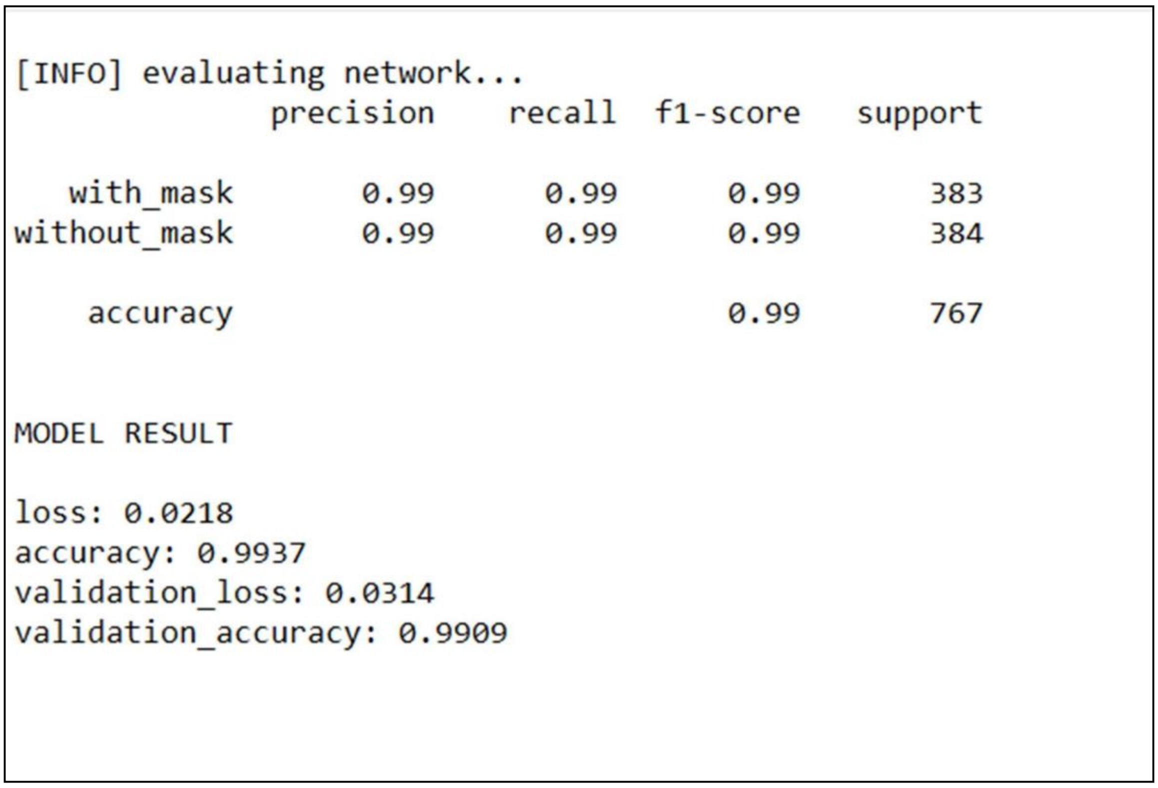


Figure 16: Training Loss and Accuracy Graph

4) Validation Results

The below image depicts the classification matrix for the two models, i.e with mask and without_mask and also numerically represents the loss, accuracy, validation_loss and validation_accuracy.



```
[INFO] evaluating network...
              precision    recall  f1-score   support

   with_mask       0.99      0.99      0.99       383
without_mask       0.99      0.99      0.99       384

    accuracy                           0.99       767


MODEL RESULT

loss: 0.0218
accuracy: 0.9937
validation_loss: 0.0314
validation_accuracy: 0.9909
```

Figure 17: Validation Results

9. CONCLUSION AND FUTURE SCOPE

The objective of this project i.e., to detect whether a person on live camera video stream is wearing a mask or not is achieved with evidence from validation measure results. The proposed method effectively detects face wearing masks with the help of TensorFlow, MobileNetV2 and OpenCV. The speed of evaluating each of the 20 Epochs initialized is increased by using Adam Optimizer, which efficiently increased the performance and decreased the time consumption for training and testing the model. The Future Scope of this Project includes implementing this in all public places initially within our college, but also in collaboration with various Non-Profit Organizations or Government Organizations, we can set up a hardware device in addition to the Model that can be used to cohesively and efficiently detect if a person is wearing a mask or not and to what accuracy, thus helping officials to ease the process of enforcing strict rules and regulations.

10. REFERENCES

[1] Goyal, K., Agarwal, K., & Kumar, R. (2017, April). Face detection and tracking: Using OpenCV. In *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)* (Vol. 1, pp. 474-478). IEEE.

[2] Saxen, F., Werner, P., Handrich, S., Othman, E., Dinges, L., & Al-Hamadi, A. (2019, September). Face attribute detection with mobilenetv2 and nasnet-mobile. In *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)* (pp. 176-180). IEEE.

[3] Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement*, *167*, 108288.

[4] Ding, Y., Li, Z., & Yastremsky, D. (2021). Real-time face mask detection in video data. *arXiv preprint arXiv:2105.01816*.

[5] Nowrin, A., Afroz, S., Rahman, M. S., Mahmud, I., & Cho, Y. Z. (2021). Comprehensive review on facemask detection techniques in the context of covid-19. *IEEE access*.

[6] Liang, M., Gao, L., Cheng, C., Zhou, Q., Uy, J. P., Heiner, K., & Sun, C. (2020). Efficacy of face mask in preventing respiratory virus transmission: A systematic review and meta-analysis. *Travel medicine and infectious disease*, *36*, 101751.

[7] Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet50 for medical face mask detection Mohamed Loey, Gunasekaran Manogaran, Mohamed Hamed N. Taha, Nour Eldeen M. Khalifa Sustainable Cities and Society, June 2020

[8]]SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2 Preeti Nagrath, Rachna Jain, Agam Madan, Rohan Arora, Piyush Kataria, and Jude Hemanth Sustain Cities Soc, December 2020

[9] Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement*, *167*, 108288.

[10] Singh, S., Ahuja, U., Kumar, M., Kumar, K., & Sachdeva, M. (2021). Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment. *Multimedia Tools and Applications*, *80*(13), 19753-19768.

[11] Yadav, S. (2020). Deep learning based safe social distancing and face mask detection in public areas for covid-19 safety guidelines adherence. *Int. J. Res. Appl. Sci. Eng. Technol*, *8*(7), 1368-1375.

[12] Abboah-Offei, M., Salifu, Y., Adewale, B., Bayuo, J., Ofosu-Poku, R., & Opare-Lokko, E. B. A. (2021).

[13] Su, X., Gao, M., Ren, J., Li, Y., Dong, M., & Liu, X. (2022). Face mask detection and classification via deep transfer learning. *Multimedia Tools and Applications*, *81*(3), 4475-4494.

[14] Song, Z., Nguyen, K., Nguyen, T., Cho, C., & Gao, J. (2021, August). Camera-Based Security Check for Face Mask Detection Using Deep Learning. In 2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService) (pp. 96-106). IEEE.

[15] Bansal, A., Dhayal, S., Mishra, J., & Grover, J. (2022). COVID-19 Outbreak: Detecting face mask types in real time. *Journal of Information and Optimization Sciences*, *43*(2), 357-370.

[16] Real-world masked face recognition dataset (RMFD) collected from below link
https://drive.google.com/file/d/1UlOk6EtiaXTHylRUx2mySgvJX9ycoeBp/view

[17] Labelled Faces in the Wild (LFW) dataset along with Simulated Masked Faced Dataset collected from
http://vis-www.cs.umass.edu/lfw/

11. APPENDIX

Sample Code:

1) video_mask_detection.py

Code:

```
import cv2
import imutils
import numpy as np
from imutils.video import VideoStream
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import img_to_array

def detect_and_predict_mask(frame, faceNet, maskNet):

        (h, w) = frame.shape[:2]
        blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
                (104.0, 177.0, 123.0))

        faceNet.setInput(blob)
        detections = faceNet.forward()
        print(detections.shape)

        faces = []
        locs = []
        preds = []

        for i in range(0, detections.shape[2]):
                confidence = detections[0, 0, i, 2]
                if confidence > 0.5:
                        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                        (startX, startY, endX, endY) = box.astype("int")
                        (startX, startY) = (max(0, startX), max(0, startY))
                        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
                        face = frame[startY:endY, startX:endX]
                        face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
                        face = cv2.resize(face, (224, 224))
                        face = img_to_array(face)
                        face = preprocess_input(face)
                        faces.append(face)
                        locs.append((startX, startY, endX, endY))

        if len(faces) > 0:
                faces = np.array(faces, dtype="float32")
                preds = maskNet.predict(faces, batch_size=32)
        return (locs, preds)

prototxtPath = r"C:\Users\adithya\Desktop\Face_Mask_Detection-
master\face_detector\deploy.prototxt"
weightsPath = r"C:\Users\adithya\Desktop\Face_Mask_Detection-
master\face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

maskNet = load_model("mask_detector.model")

print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()

while True:
        frame = vs.read()
        frame = imutils.resize(frame, width=400)
        (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
        for (box, pred) in zip(locs, preds):
                (startX, startY, endX, endY) = box
                (mask, withoutMask) = pred
                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, 0.45, color, 2)
                cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("c"):
                break

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

2) mask_detection_training.py

<u>Code:</u>

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
```

```
import matplotlib.pyplot as plt
import numpy as np
import os

INIT_LR = 1e-4
EPOCHS = 20
BS = 32

DIRECTORY = r"C:\Users\adithya\Desktop\Face_Mask_Detection-master\dataset"
CATEGORIES = ["with_mask", "without_mask"]

print("[INFO] loading images...")
data = []
labels = []

for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):
                                                img_path = os.path.join(path, img)
                                                image = load_img(img_path,
target_size=(224, 224))
                                                image = img_to_array(image)
                                                image = preprocess_input(image)

                                                data.append(image)
                                                labels.append(category)
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

data = np.array(data, dtype="float32")
labels = np.array(labels)

(trainX, testX, trainY, testY) = train_test_split(data, labels,
        test_size=0.20, stratify=labels, random_state=42)
```

```
aug = ImageDataGenerator(
	rotation_range=20,
	zoom_range=0.15,
	width_shift_range=0.2,
	height_shift_range=0.2,
	shear_range=0.15,
	horizontal_flip=True,
	fill_mode="nearest")

baseModel = MobileNetV2(weights="imagenet", include_top=False,
	input_tensor=Input(shape=(224, 224, 3)))

headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
model = Model(inputs=baseModel.input, outputs=headModel)

for layer in baseModel.layers:
	layer.trainable = False
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
	metrics=["accuracy"])
print("[INFO] training head...")
H = model.fit(
	aug.flow(trainX, trainY, batch_size=BS),
	steps_per_epoch=len(trainX) // BS,
	validation_data=(testX, testY),
	validation_steps=len(testX) // BS,
	epochs=EPOCHS)
```

```
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)
predIdxs = np.argmax(predIdxs, axis=1)
print(classification_report(testY.argmax(axis=1), predIdxs,
        target_names=lb.classes_))

print("[INFO] saving mask detector model...")
model.save("mask_detector.model",  save_format="h5")

N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")
```