Summer Project report on

Face Mask Detection

by

Siddhi Gavande [2020510023]

Anupam Tiwari [2020510065]

Under the guidance of

Internal Supervisor

[Prof. Pallavi Thakur]

Department of Master in Computer Applications

Sardar Patel Institute of Technology

Autonomous Institute Affiliated to Mumbai University

2021-22

CERTIFICATE OF APPROVAL

This is to certify that the following students

Siddhi Gavande [2020510023]

Anupam Tiwari [2020510065]

Have satisfactorily carried out work on the project entitled

"Face Mask Detection"

Towards the fulfillment of summer term project, as laid down by University of Mumbai

during year 2021-22.

Project Guide 2

(Prof. Pallavi Thakur)

PROJECT APPROVAL CERTIFICATE

This is to certify that the following student

Siddhi Gavande [2020510023]
Anupam Tiwari [2020510065]

Have successfully completed the Project report on **"Face Mask Detection"**, which is found to be satisfactory and is approved

At

SARDAR PATEL INSTITUTE OF TECHNOLOGY,

ANDHERI (W), MUMBAI.

INTERNAL EXAMINER

EXTERNAL EXAMINER

Head of Department

(Dr. Pooja Raundale)

Principal

(Dr. B. N. Chaudhari)

Table of Contents

Abstract

Coronavirus disease has a serious impact throughout the world since 2019. Some of the precautions which people should take in these times is to wear masks in public areas. There is a need of the accurate mask detector system which will ensure that people are following the COVID-19 guidelines and in this way, it would hopefully help to decrease the number of corona cases.

The Face Mask Detection will be a system which detects the person wearing the mask or not during the COVID-19 situation. It will also detect what is the location of the face and provide the live percentage of whether a person is wearing mask correctly. This will be helpful in the entrance of the stores, airports, railway stations, offices, schools or can be used in any public areas by the authorities where there is a need. In this project, we propose a mask detector using image processing. The technique used for segmentation and classification of images is the light architectures of Deep Convolutional Neural Network – MobileNet v2.

Objectives

- To propose a method that will reduce the effort of the authorities or the officials to detect people not wearing the facemask.
- The main objective of developing this system is to create a training model which learns the face mask features of the dataset with deep learning algorithms.
- Provide the accuracy of percentage of the person is wearing / not wearing a mask.

List of Figures

List of Tables

1. Introduction

Face mask detection project is to detect whether a person wearing a mask or not and what is the location of the face. The problem is closely related to general object detection to detect the classes of objects and face detection is to detect a particular class of objects, i.e. face. Applications of object and face detection can be found in many areas, such as, airports, offices, railway stations, schools, surveillance and so on.

This project of "Face Mask Detection" basically aims to detect the person wearing mask or not. Further, it provides the accuracy percentage of how much percent the person is wearing / not wearing a mask. Here, the input is through a video from a webcam and processed through OpenCV, hence it will detect and capture the image from the frame and process the image and give the accuracy.

The face mask recognition in this study is developed with a machine learning algorithm through the image classification method: MobileNetv2. MobileNetV2 is a method based on Convolutional Neural Network (CNN) that developed by Google with improved performance and enhancement to be more efficient. This study conducted its experiments on two original datasets namely with mask and without mask. The first dataset was taken as the names says contains the images of person wearing mask and second without wearing a mask. These datasets are collected from the Kaggle dataset, few open sources image libraries and google images which are further used for the training, validation, and testing phase so the model can be implemented to the dataset.

1.1. Problem Definition

Since the declaration of the COVID-19 virus as a pandemic by WHO stated that efforts have been made by various parties to reduce the spread of the virus. Physical distancing and wearing a face mask in the public place to impede COVID-19 transmission is being maintained. However, some difficulties are faced by the authorities in the process of monitoring a large population that has a different habit in the country. The authorities need a solution to be able to validly control the implementation of the law, which begins with the availability of the data quickly and accurately. One of the solutions is to use a regionally automated face mask detection to differentiate between people who wear masks and those who do not.

1.2. Objective and Scope

- To detect the person wearing mask or not.
- Provide the accuracy of percentage of the person is wearing / not wearing a mask.

1.3. System Requirement

Hardware Requirements:

	Minimum	Recommended
Graphic card	AMD Radeon R5 M435	NVIDIA GeForce GTX 1650
Processor	Intel Core i5-8265U 160GHz	AMD Ryzen 53550H with Radeon Vega
Disk Capacity	1 Terabytes	1 Terabytes
RAM Capacity	8 Gigabytes	16 Gigabytes

Software Requirements:

Operating System	Microsoft Windows 10
Documentation Tool	Microsoft Word, Web Browser (https://draw.io/).
Software	Python 3.8.7, Anaconda3
Coding Languages	Python
Library	TensorFlow, Keras

Feasibility Study:

1. Technical Feasibility:

Technical: Python 3.8.7, Anaconda3
Requirements of the Libraries:

1) tensorflow>=1.15.2
2) keras==2.3.1
3) imutils==0.5.3
4) numpy==1.18.2
5) opencv-python==4.2.0.*
6) matplotlib==3.2.1
7) scipy==1.4.1

Operational Resources:
1 Machine
2 Developers
1 Tester

HP Pavilion Ryzen 5 Hexa Core 4600H - (8 GB/1 TB HDD/Windows 10 Home/NVIDIA GeForce GTX 1650)

2. Economic Feasibility:

Development Cost: Rs 45000

Total Budget: Rs 45000

2. SRS and Design

2.1. Introduction

2.1.1. Purpose

The purpose of this document is to provide a debriefed view of requirements and specifications of the project called Face Mask Detection System using deep learning.

The goal of this project is to automatically detect the face masks.

2.1.2. Scope

The Face Mask Detection system will allow the authorities to detect the people wearing mask or not who all are not following the guidelines. It will also provide the accuracy percentage of the person is wearing / not wearing a mask.

2.1.3. References

IEEE Standard

- https://ieeexplore.ieee.org/document/9325631

2.2. Overall Description

2.2.1. Product Perspective

Coronavirus disease 2019 has affected the world seriously. One major protection method for people is to wear masks in public areas. Furthermore, many public service providers require customers to use the service only if they wear masks correctly. However, there are only a few research studies about face mask detection based on image analysis.

2.2.2. Product Functions

- Pre-processing
- Training the Images
- Face Mask Detection

2.3. Assumptions and Dependencies

Assumptions

The datasets which we will be used for the initial entries of the training of the system is assumed to be the correct input for the system. The training dataset is taken from the online repositories such as Kaggle.

Dependencies

- Python
- Tensorflow – keras

2.4. Non-functional Requirements

2.4.1. Performance Requirements

1. The user must not move his/her face out of camera's sight in order to get correct results.
2. The background must not be too bright or too dark while detecting the face mask.
3. The system will be implemented in Python script with an accuracy of the model of over 90%.

3. Project Analysis and Design

3.1. Methodology Accepted

The best suitable model for this project is Iterative model.

Iterative Model :
In this project, we iterate through the training and the testing phase during the development.
With Iterative development, there are changes on each iteration, evolves and grows.
As each iteration builds on the previous one, software design remains consistent.

3.2. Algorithm

The major contributions of this system are:
Create a training model which learns the face mask features of the dataset with MobileNetV2 and Machine learning algorithms.

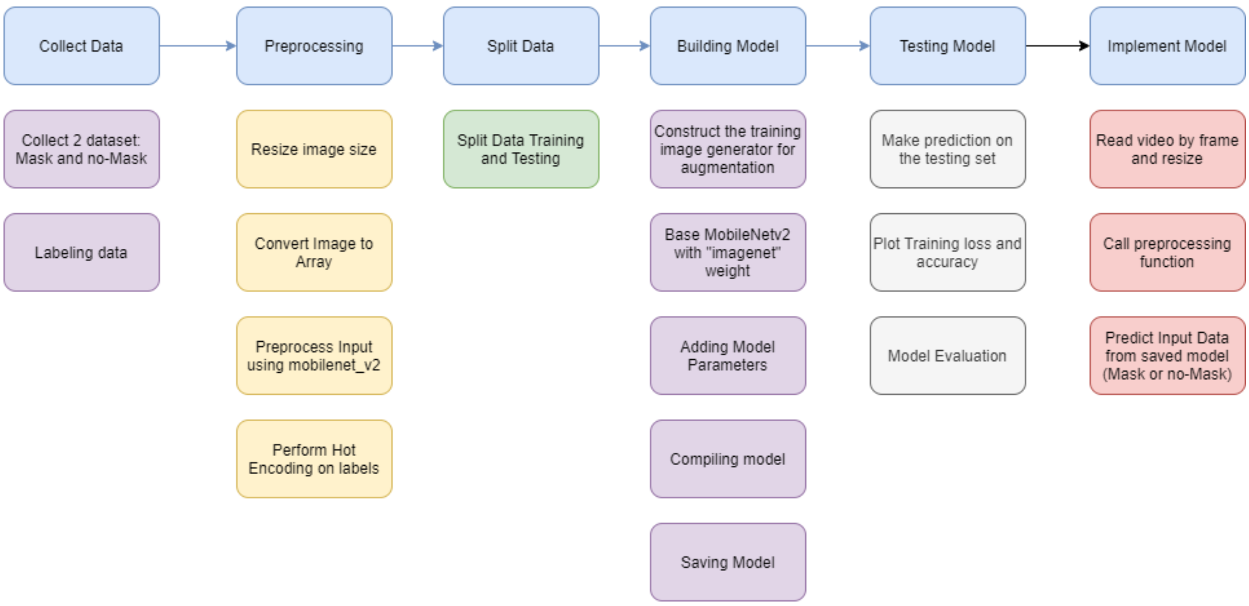


Fig 3.2.1. Steps in building the model

Convolutional Neural Network:

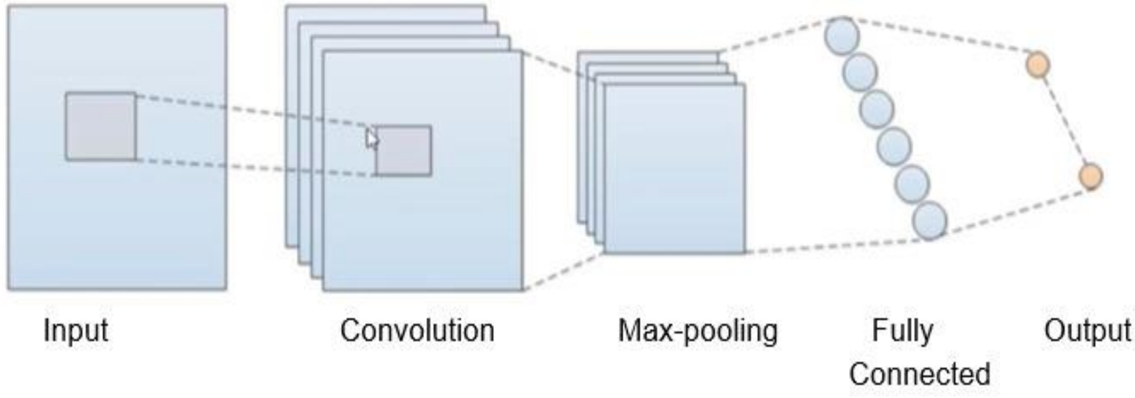


Fig 3.2.2. CNN

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

Convolutional neural network (CNN or ConvNet) is class of neural networks that specializes in processing data that has a grid-like topology e.g. digital image is a binary representation of visual data.

3.3. Flowchart Diagram

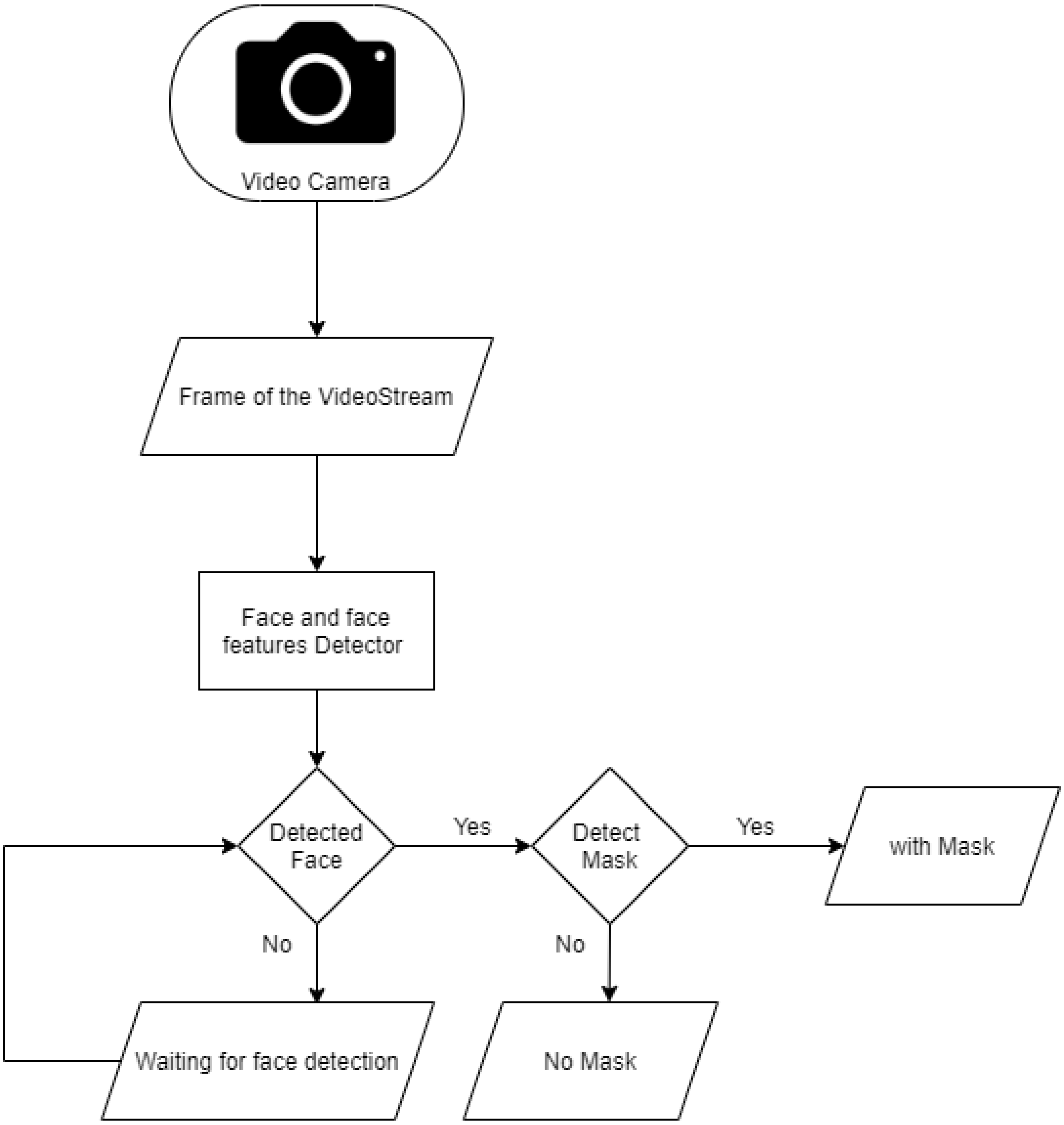


Fig 3.3.1. Flowchart of Face Mask Detection

3.4. Gantt Table

Table 3.4.1 Gantt Table

ID	Name	Begin date	End date	Duration	Predecessors	Priority	Resources
0	Requirement Gathering (FMD)	07/06/21	21/06/21	11			
3	Project selection research	07/06/21	07/06/21	1			Siddhi,Anupam
5	Gather Information	07/06/21	09/06/21	3			Siddhi,Anupam
7	Dataset Collection	09/06/21	21/06/21	9			Siddhi
8	Data Preprocessing (FMD)	21/06/21	25/06/21	5			
12	Image resizing	21/06/21	22/06/21	2			Anupam
10	Conversion image to array	22/06/21	23/06/21	2	7		Siddhi
11	Labelling data	24/06/21	25/06/21	2	7,10		Siddhi
13	Design (FMD)	25/06/21	26/07/21	22			
14	Building model	25/06/21	08/07/21	10			Siddhi,Anupam
15	Train model	29/06/21	16/07/21	14			Siddhi,Anupam
16	Compiling model	19/07/21	26/07/21	6	15		Anupam
17	Testing model (FMD)	27/07/21	04/08/21	7	13		
18	Prediction on testing data	27/07/21	02/08/21	5			Anupam
19	Plot training loss, accuracy	03/08/21	04/08/21	2	18		Siddhi
20	Evaluation (FMD)	05/08/21	11/08/21	5			
21	Model evaluation	05/08/21	11/08/21	5	17		Siddhi,Anupam
22	Model implementation (FMD)	12/08/21	13/08/21	2			
23	Live Detection Mask / No Mask	12/08/21	13/08/21	2	20		Siddhi,Anupam

3.5. Gantt Chart

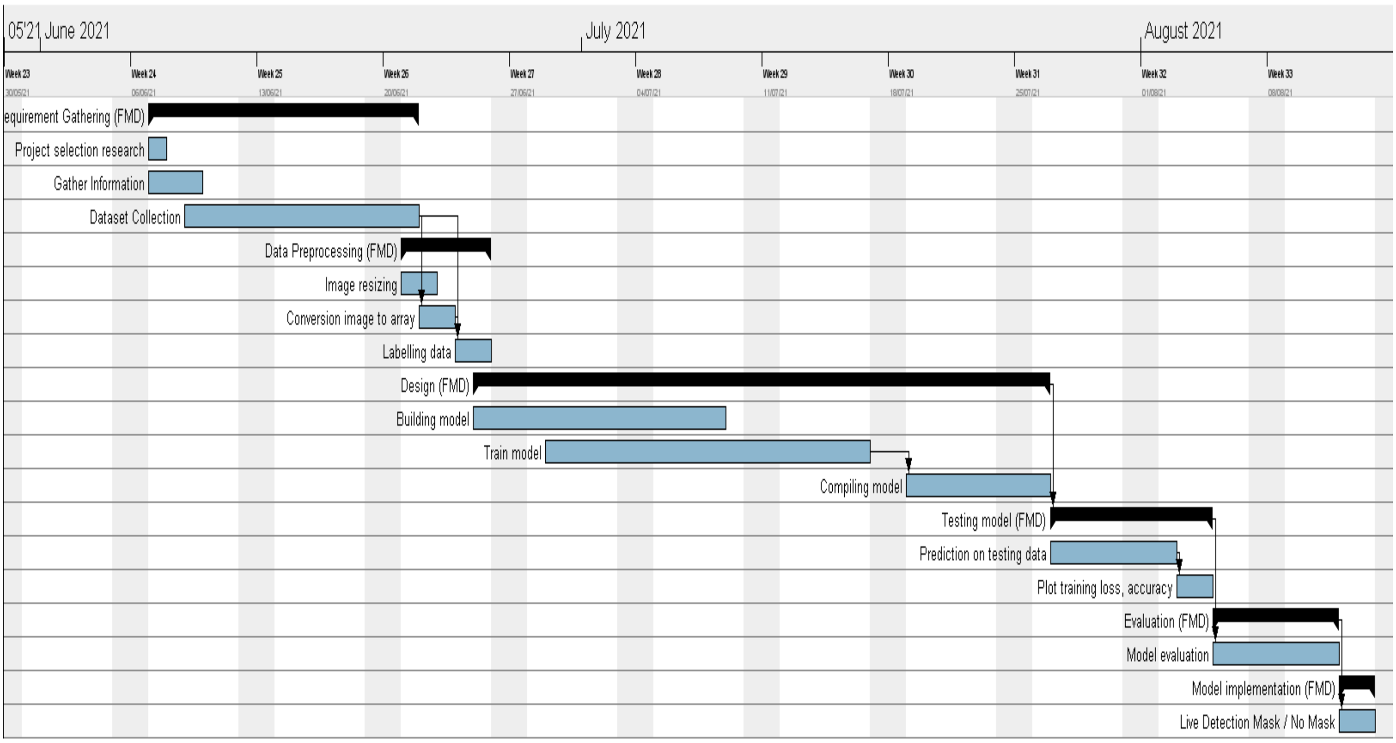


Fig 3.5.1 Gantt Chart in weeks

3.6. Architectural Diagram (Pipe-Filter)

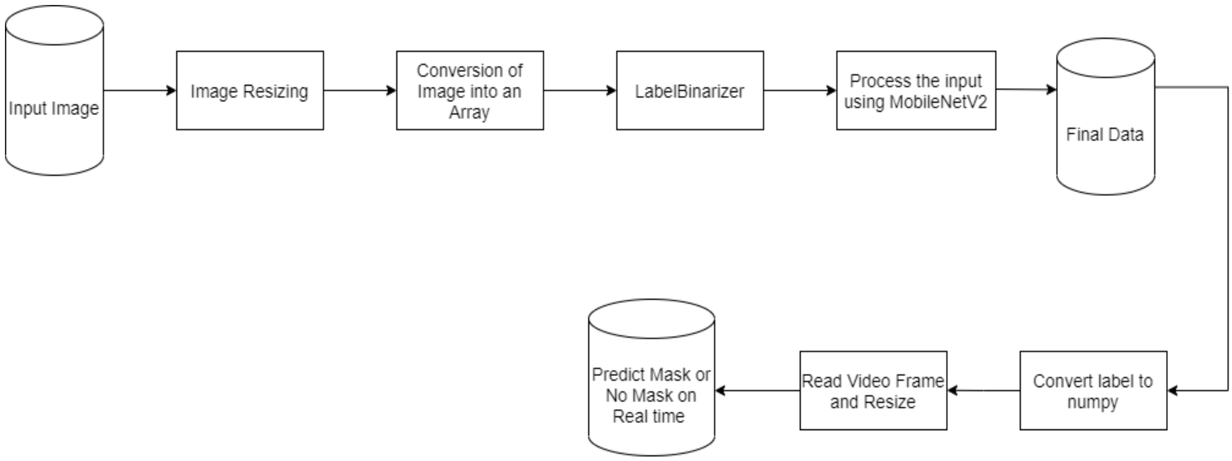


Fig 3.6.1. Pipe – Filter Architectural Diagram for Face Mask Detection

4.System Design

4.1 Users of the System

User: The user who handles the System.

4.2 Modularity criteria

The proposed system has following modules:

- data collecting
- pre-processing
- split the data
- building the model
- testing the model
- implement the model

4.3 Design Methodologies

Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in
input image, assign importance (learnable weights and biases) to various aspects/objects in the image
and be able to differentiate one from the other.

- The availability of GPUs that make model training process faster
- The availability of dataset.

For image captioning task, CNN is widely used because of the success of CNN to be used in image annotation problems. CNN has successfully solved image annotation problems with high accuracy.

4.4 User Interface Layouts

User interface design (UI) or user interface engineering is the design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing usability and the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centered design).

Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to itself. Graphic design and typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the Interface. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

5. Implementation and Testing

5.1 Tools/Scripts for Implementation

- **Anaconda**

  Anaconda is a free open source software and it supports free GPU. we can improve your Python programming language coding skills. Develop machine learning and deep learning applications using popular libraries such as Keras , TensorFlow and OpenCV. We use Anaconda for developing the training model.

- **TensorFlow**

  TensorFlow is an end-to-end open source platform for machine learning. It's a comprehensive and flexible ecosystem of tools, libraries and other resources that provide workflows with high- level APIs. The framework offers various levels of concepts for you to choose the one you need to build and deploy machine learning models. For instance, if you need to do some large machine learning tasks, you can use the Distribution Strategy API in order to perform distributed hardware configurations and if you need a full production machine learning pipeline, you can simply use TensorFlow Extended (TFX).

- **Keras**

  Keras on the other hand, is a high-level neural networks library which is running on the top of TensorFlow, CNTK, and Theano. Using Keras in deep learning allows for easy and fast prototyping as well as running seamlessly on CPU and GPU. This framework is written in Python code which is easy to debug and allows ease for extensibility.

- **OpenCV**

  OpenCV (**Open Source Computer Vision Library**) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

5.2 Module hierarchy

- **Dataset Collection**

  Dataset Collected From COCO Dataset which gives free and lively images.

- **Preprocessing**

  Preprocessing on images is done to convert image objects into RGB arrays. Then the array is resized to (299, 299, 3). Preprocessing in the caption is performed to make sentences that were previously in the form of the word into a sequence of tokens based on a unique word index in the dictionary. At the training phase, the model has two inputs. The first input is an image feeding into the pre-trained Inception-v3 model with the removed output layer and will outputting extracted images features. The second input is a description that has been done by preprocessing so that it becomes a sequence index of tokens.

- **Training the Images**

  The inception V3 model is employed as the encoderto extract CNN features of the target image and the training images. During the training stage, CNN features of the training images under the proposed weighted likelihood objective is identified. In the generation stage, the trained GRU plays as a decoder role, which takes the CNN features of the target image as input and generates identification of the images.

- **Face Mask Detection**

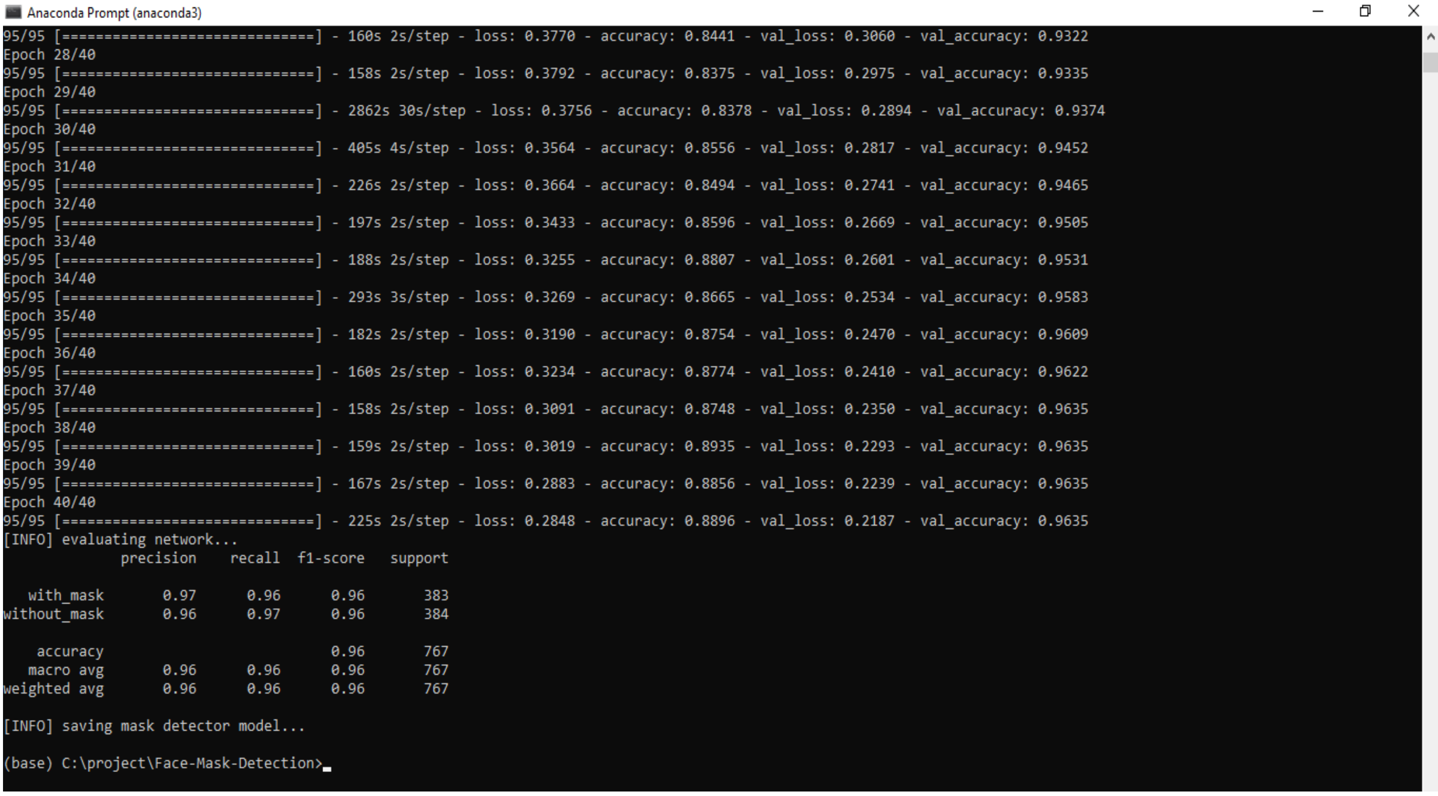
  In order to design an effective network for face mask detection, we adopt the object detector framework in which it suggests a detection network with a backbone, a neck and heads. The backbone refers to a general feature extractor made up of convolutional neural networks to extract information in images to feature maps.

5.3 Testing

5.3.1 Testing the Model

To make sure the model can predict well, there are steps in testing the model. The first step is making predictions on the testing set. The result for 20 iterations in checking the loss and accuracy when training the model is show in below figure.

```
Anaconda Prompt (anaconda3)
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-07-22 19:21:22.509314: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x24d48c6ce60 initialized for platform Host (this does not guarantee that XLA
will be used). Devices:
2021-07-22 19:21:22.509643: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
[INFO] compiling model...
[INFO] training head...
Epoch 1/40
2021-07-22 19:22:11.676638: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 51380224 exceeds 10% of free system memory.
2021-07-22 19:22:16.754082: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 51380224 exceeds 10% of free system memory.
2021-07-22 19:22:18.126514: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 154140672 exceeds 10% of free system memory.
2021-07-22 19:22:18.271971: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 156905472 exceeds 10% of free system memory.
2021-07-22 19:22:18.598774: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 57802752 exceeds 10% of free system memory.
95/95 [==============================] - 375s 4s/step - loss: 0.8758 - accuracy: 0.5267 - val_loss: 0.7644 - val_accuracy: 0.5098
Epoch 2/40
95/95 [==============================] - 189s 2s/step - loss: 0.8572 - accuracy: 0.5422 - val_loss: 0.7265 - val_accuracy: 0.5398
Epoch 3/40
95/95 [==============================] - 182s 2s/step - loss: 0.8262 - accuracy: 0.5593 - val_loss: 0.6930 - val_accuracy: 0.5658
Epoch 4/40
95/95 [==============================] - 162s 2s/step - loss: 0.7583 - accuracy: 0.5798 - val_loss: 0.6648 - val_accuracy: 0.5893
Epoch 5/40
95/95 [==============================] - 160s 2s/step - loss: 0.7379 - accuracy: 0.5985 - val_loss: 0.6376 - val_accuracy: 0.6219
Epoch 6/40
95/95 [==============================] - 158s 2s/step - loss: 0.7262 - accuracy: 0.5910 - val_loss: 0.6125 - val_accuracy: 0.6571
Epoch 7/40
95/95 [==============================] - 158s 2s/step - loss: 0.7037 - accuracy: 0.6177 - val_loss: 0.5886 - val_accuracy: 0.6793
Epoch 8/40
95/95 [==============================] - 157s 2s/step - loss: 0.6887 - accuracy: 0.6249 - val_loss: 0.5663 - val_accuracy: 0.6988
Epoch 9/40
95/95 [==============================] - 159s 2s/step - loss: 0.6669 - accuracy: 0.6440 - val_loss: 0.5451 - val_accuracy: 0.7275
Epoch 10/40
95/95 [==============================] - 312s 3s/step - loss: 0.6247 - accuracy: 0.6674 - val_loss: 0.5255 - val_accuracy: 0.7510
Epoch 11/40
95/95 [==============================] - 332s 3s/step - loss: 0.6125 - accuracy: 0.6753 - val_loss: 0.5070 - val_accuracy: 0.7718
Epoch 12/40
95/95 [==============================] - 308s 3s/step - loss: 0.5977 - accuracy: 0.6744 - val_loss: 0.4891 - val_accuracy: 0.7914
Epoch 13/40
95/95 [==============================] - 324s 3s/step - loss: 0.5595 - accuracy: 0.7179 - val_loss: 0.4727 - val_accuracy: 0.8057
Epoch 14/40
95/95 [==============================] - 236s 2s/step - loss: 0.5665 - accuracy: 0.7175 - val_loss: 0.4571 - val_accuracy: 0.8201
Epoch 15/40
95/95 [==============================] - 210s 2s/step - loss: 0.5468 - accuracy: 0.7274 - val_loss: 0.4419 - val_accuracy: 0.8383
Epoch 16/40
95/95 [==============================] - 157s 2s/step - loss: 0.5393 - accuracy: 0.7251 - val_loss: 0.4273 - val_accuracy: 0.8540
Epoch 17/40
```

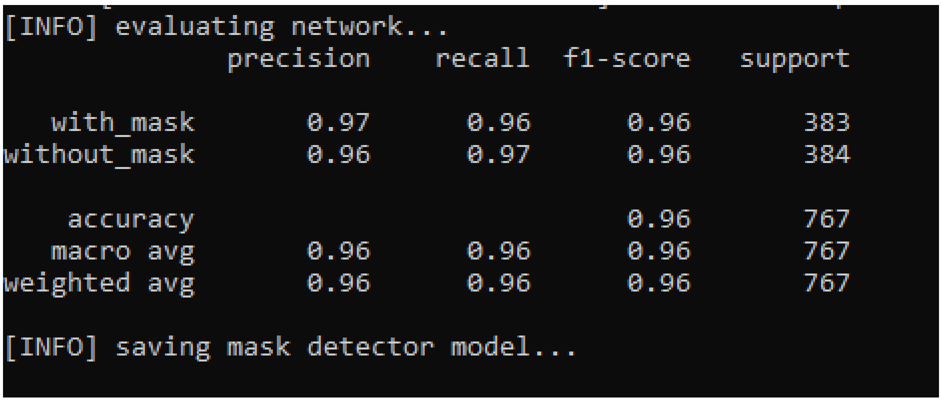


```
Anaconda Prompt (anaconda3)
95/95 [==============================] - 160s 2s/step - loss: 0.3770 - accuracy: 0.8441 - val_loss: 0.3060 - val_accuracy: 0.9322
Epoch 28/40
95/95 [==============================] - 158s 2s/step - loss: 0.3792 - accuracy: 0.8375 - val_loss: 0.2975 - val_accuracy: 0.9335
Epoch 29/40
95/95 [==============================] - 2862s 30s/step - loss: 0.3756 - accuracy: 0.8378 - val_loss: 0.2894 - val_accuracy: 0.9374
Epoch 30/40
95/95 [==============================] - 405s 4s/step - loss: 0.3564 - accuracy: 0.8556 - val_loss: 0.2817 - val_accuracy: 0.9452
Epoch 31/40
95/95 [==============================] - 226s 2s/step - loss: 0.3664 - accuracy: 0.8494 - val_loss: 0.2741 - val_accuracy: 0.9465
Epoch 32/40
95/95 [==============================] - 197s 2s/step - loss: 0.3433 - accuracy: 0.8596 - val_loss: 0.2669 - val_accuracy: 0.9505
Epoch 33/40
95/95 [==============================] - 188s 2s/step - loss: 0.3255 - accuracy: 0.8807 - val_loss: 0.2601 - val_accuracy: 0.9531
Epoch 34/40
95/95 [==============================] - 293s 3s/step - loss: 0.3269 - accuracy: 0.8665 - val_loss: 0.2534 - val_accuracy: 0.9583
Epoch 35/40
95/95 [==============================] - 182s 2s/step - loss: 0.3190 - accuracy: 0.8754 - val_loss: 0.2470 - val_accuracy: 0.9609
Epoch 36/40
95/95 [==============================] - 160s 2s/step - loss: 0.3234 - accuracy: 0.8774 - val_loss: 0.2410 - val_accuracy: 0.9622
Epoch 37/40
95/95 [==============================] - 158s 2s/step - loss: 0.3091 - accuracy: 0.8748 - val_loss: 0.2350 - val_accuracy: 0.9635
Epoch 38/40
95/95 [==============================] - 159s 2s/step - loss: 0.3019 - accuracy: 0.8935 - val_loss: 0.2293 - val_accuracy: 0.9635
Epoch 39/40
95/95 [==============================] - 167s 2s/step - loss: 0.2883 - accuracy: 0.8856 - val_loss: 0.2239 - val_accuracy: 0.9635
Epoch 40/40
95/95 [==============================] - 225s 2s/step - loss: 0.2848 - accuracy: 0.8896 - val_loss: 0.2187 - val_accuracy: 0.9635
[INFO] evaluating network...
              precision    recall  f1-score   support

   with_mask       0.97      0.96      0.96       383
without_mask       0.96      0.97      0.96       384

    accuracy                           0.96       767
   macro avg       0.96      0.96      0.96       767
weighted avg       0.96      0.96      0.96       767

[INFO] saving mask detector model...

(base) C:\project\Face-Mask-Detection>_
```

Fig 5.3.1. Testing the Model

5.3.2 Model Evaluation

When the accuracy line is being stable, it means that there is no need for more iteration for increasing the accuracy of the model. So then , the next step is making the model evaluation as show in below figure.



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

   with_mask       0.97      0.96      0.96       383
without_mask       0.96      0.97      0.96       384

    accuracy                           0.96       767
   macro avg       0.96      0.96      0.96       767
weighted avg       0.96      0.96      0.96       767

[INFO] saving mask detector model...
```

Fig 5.3.2. Model Evaluation

5.4 Testing Types

5.4.1 Unit testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. In our system,

- Test the preprocessing module work properly to preprocess the dataset
- Test to check whether the training of the images work properly.
- Test to check whether the model recognizes the face mask Accurately.

5.4.2 Integration testing

Integration testing (sometimes called integration and testing) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

- Check whether the model takes the input image.
- Check whether the System plot the features of given image.
- Check whether the model recognizes the face mask accurately.

5.4.3 System testing

System testing of software or hardware is testing conducted on a complete, integrated system toevaluate the system's compliance with its specified requirements.

5.5 Snapshot of UI

5.5.1. Without mask:

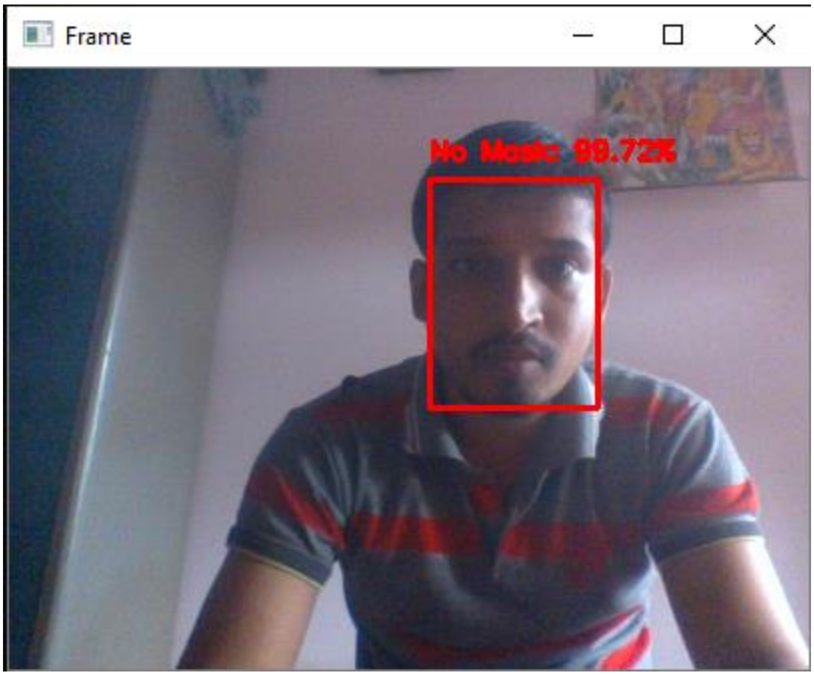


Fig 5.5.1 (a) Without mask

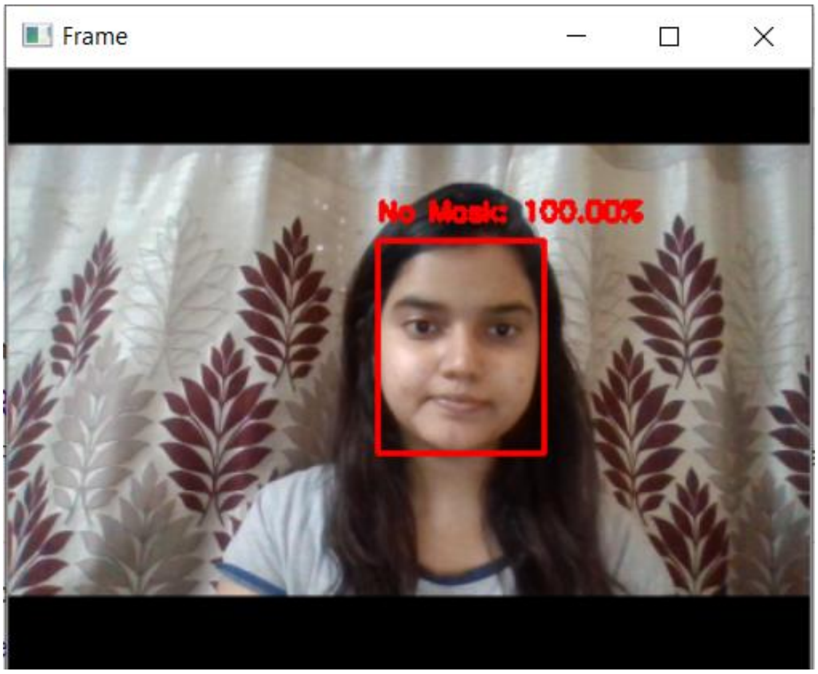


Fig 5.5.1 (b) Without mask

5.5.2. With Mask:

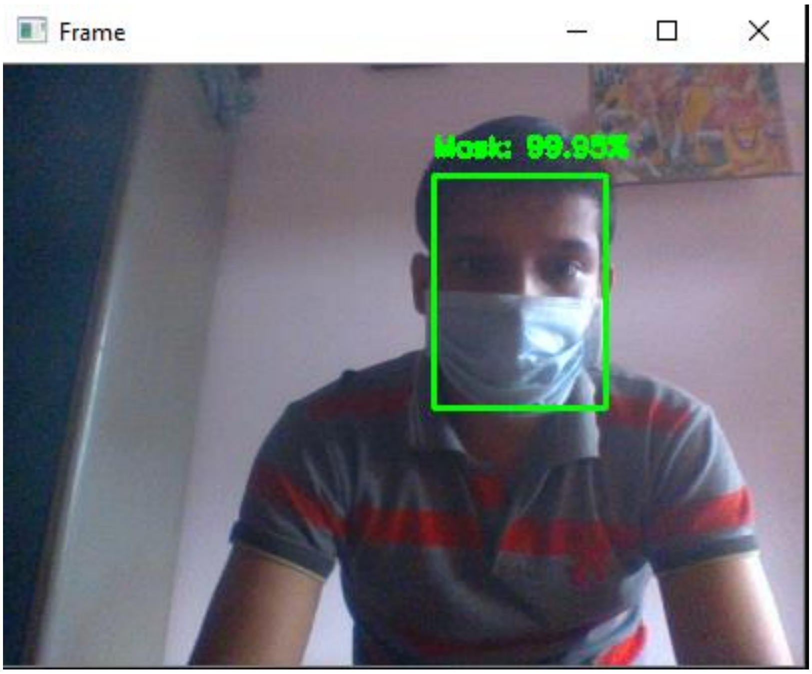


Fig 5.5.2 (a) With Mask

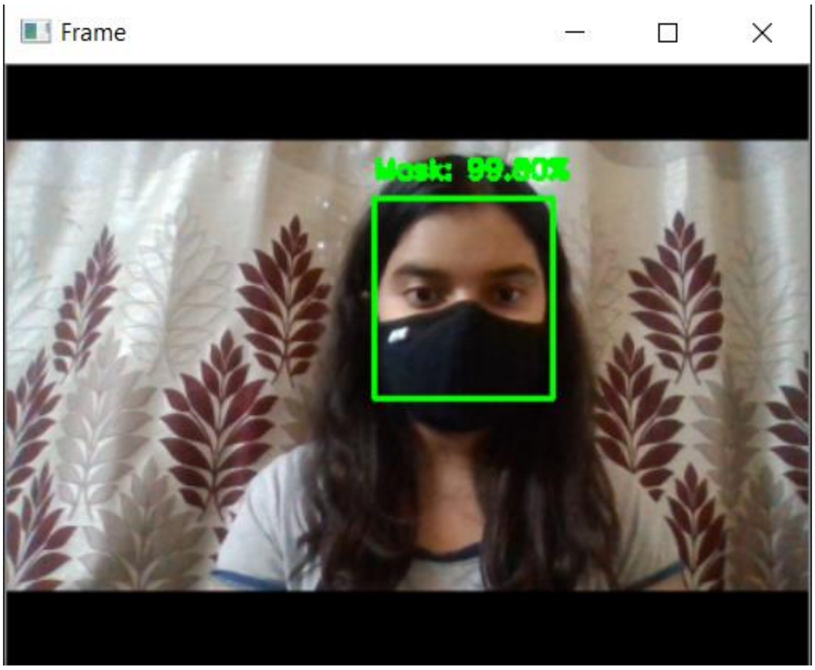


Fig 5.5.2 (b) With Mask

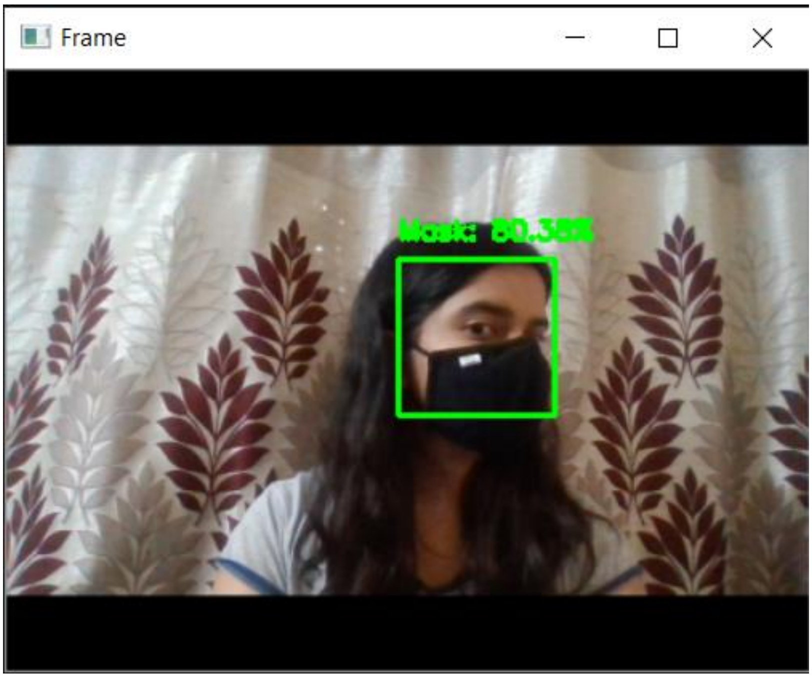


Fig 5.5.2. (c) With Mask

6. Future Enhancement

- We can use Internet of Things (IoT) technology like Arduino to detect temperature, mask wearing to keep a count of the number of people or an alert system if a person is not wearing a mask, while mask detection is used to identify individuals near the camera are wearing a mask or not.

- Enhance the accuracy of this model by more image Training and usage of efficient cameras the accuracy of the system can be improved.

7. Limitations

- Detection is vulnerable. While face detection provides more accurate results than manual identification processes, it can also be more easily thrown off by changes in appearance or camera angles.
- Massive data storage burden. The ML technology used in face detection requires powerful data storage that may not be available to all users.
- A potential breach of privacy. Face detection's ability to help the government track down criminals creates huge benefits; however, the same surveillance can allow the government to observe private citizens. Strict regulations must be set to ensure the technology is used fairly and in compliance with human privacy rights.

8. Conclusion

In Conclusion, this study presents a model using machine learning for face mask detection. After the training, Validation, and testing phase, the model can provide the percentage of people using face mask in some cities with high accuracy.

In the name of the statistical organization that needs to move quickly to adopt and take advantage of machine learning and new digital data resources, this study can be an, easy move for authorities to use more unstructured data resources for more data-based mitigation, evaluation, prevention, and action planning against COVID-19.

9. Bibliography

1. https://www.kaggle.com/andrewmvd/face-mask-detection
2. https://ieeexplore.ieee.org/document/9325631
3. https://www.pyimagesearch.com/
4. https://stackoverflow.com/
5. https://scikit-learn.org/
6. https://keras.io/api/layers/pooling_layers/average_pooling2d/
7. https://keras.io/api/layers/reshaping_layers/flatten/
8. https://keras.io/api/layers/core_layers/dense/
9. https://keras.io/api/layers/activations/
10. https://keras.io/api/applications/mobilenet/
11. https://www.geeksforgeeks.org/matplotlib-pyplot-figure-in-python/