FACE MASK RECOGNITION SYSTEM

A project report submitted

to

Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY,LUCKNOW

For Partial Fulfilment of Requirement for the

Award of the Degree

of

MASTER OF COMPUTER APPLICATION

by

KRASHN TRIPATHI (2000270140023)

RISHABH MAHESHWARI (2000270140044)

SHEFALIMANSHANI(2000270140047)

SHIVAM SINGH(2000270140049)

VICKY KUMAR (2000270140060)

AJAYKUMARGARGENGINEERINGCOLLEGE,GHAZIABAD

2021-22

FACE MASK RECOGNITION SYSTEM

A project report submitted

to

Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY,LUCKNOW

For Partial Fulfilment of Requirement for the

Award of the Degree

of

MASTER OF COMPUTER APPLICATION

by

KRASHN TRIPATHI (2000270140023)

RISHABH MAHESHWARI (2000270140044)

SHEFALIMANSHANI(2000270140047)

SHIVAM SINGH(2000270140049)

VICKY KUMAR (2000270140060)

Under the guidance of

Dr. SAROJ BALA (Ajay Kumar Garg Engineering College)

AJAYKUMARGARGENGINEERINGCOLLEGE-MCA,GHAZIABAD

2021-22

COLLEGE CERTIFICATE

This is to certify that the project report entitled “Face Mask Recognition System” which is submitted by **KrashnTripathi** (2000270140023), **RishabhMaheshwari** (2000270140044), **ShefaliManshani** (2000270140047), **Shivam Singh** (2000270140049), **Vicky Kumar** (2000270140060) in partial fulfillment of the requirement for the award of degree Master of Computer Application of Dr. A.P.J. Abdul Kalam Technical University, Uttar Pradesh (AKTU), is a record of the candidates work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Dr. B. K. Sharma

Principal,

Ajay Kumar Garg Engineering College-MCA, Ghaziabad

ACKNOWLEDGEMENT

Withanoverwhelmingsenseofgratitude,weacknowledgethatthevaluableguidanceandconsistentencouragementextendedtousbyourknowledgeablefacultymembers with whose guidance, we are able to accomplish this endeavor. Theirtechnical acumen and yearsof experience has provided us with crucial inputs at acriticalstage.

We would also like to thank the HOD of the MCA department **Dr. B.K Sharma** and all the faculty members **Dr.SarojBala, Mrs. Mani Dwivedi, Mrs. Indu Verma,** and **Mrs. ArpanaSaxena**for their guidance.We would also like to thank the lab assistants for permitting us to use computers in the lab as and when required.

KRASHN TRIPATHI (2000270140023)

RISHABH MAHESHWARI (2000270140044)

SHEFALIMANSHANI(2000270140047)

SHIVAM SINGH(2000270140049)

VICKY KUMAR (2000270140060)

LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE

COVID- Coronavirus Disease.

API- Application Programming Interface.

IEEE- Institute of Electrical and Electronics Engineers.

UML- Unified Modelling Language.

CNN- Convolution Neural Network. CNN is a deep learning neural network sketched for processing structured arrays of data such as portrayal.

DNN- A deep neural network (DNN) is an artificial neural network(ANN) with multiple layers between the input and output layers.

MobileNetV2- MobileNetV2 is a powerful image classification tool. TensorFlow provides the image weights in MobileNetV2, a lightweight CNN-based deep learning model.

OpenCV- OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.

CCTV – Closed Circuit Television.

CNTK- Microsoft Cognitive Toolkit.

LIST OF FIGURES

ABSTRACT

In recent decades, facial recognition has become the object of research worldwide. In addition, with the advancement of technology and the rapid development of artificial intelligence, very significant advances have been made. For this reason, public and private companies use facial recognition systems to identify and control the access of people in airports, schools, offices, and other places. On the other hand, with the spread of the COVID-19 pandemic, government entities have established several biosafety regulations to limit infections. Among them is the mandatory use of face masks in public places, as they have proven to be an effective measure in protecting users and those around them. Corona virus has globally infected over 20 million people causing over 0.7 million deaths.

INDEX

Contents **Page No.**

Contents **Page No.**

1. Introduction:

1.1 Objective:

In recent decades, facial recognition has become the object of research worldwide.
In addition, with the advancement of technology and the rapid development of artificial intelligence, very significant advances have been made. For this reason, public and private companies use facial recognition systems to identify and control the access of people in airports, schools, offices, and other places. On the other hand, with the spread of the COVID-19 pandemic, government entities have established several biosafety regulations to limit infections. Among them is the mandatory use of face masks in public places, as they have proven to be an effective measure in protecting users and those around them. Corona virus has globally infected over 20 million people causing over 0.7 million deaths.

People are forced by laws to wear face masks in public in many countries. These rules and laws were developed as an action to the exponential growth in cases and deaths in many areas. However, the process of monitoring large groups of people is becoming more difficult in public areas. So we will create an automation process for detecting the faces.

Here we introduce a facemask detection model that is based on computer vision and deep learning. The proposed model can be integrated with Surveillance Cameras to impede the COVID-19 transmission by allowing the detection of people who are wearing masks not wearing face masks. The model is integration between deep learning and classical machine learning techniques with OpenCV, Tensor flow and Keras. We will achieve high accuracy while consuming least time in the process of training and detection in our model.

1.2 Scope:

Video is captured and then it is converted into frames, then it will automatically recognize the faces with masks and without masks. Whenever a face is detected without mask a message will be send to the concerned authority. It is based on computer vision technology, enabling computers to understand images, which can be exploited in wide applications such as,

- Covid-19 Control.
- Other epidemic Control.
- Image analysis.
- Other industrial areas.
- Face Mask Detection.

1.3 Methodology:

Machine learning technique is used in our system with CNN. Convolutional Neural Networks is used for image extraction and its correspondent labelling.

1.3.1Convolution Neural Network(CNN)

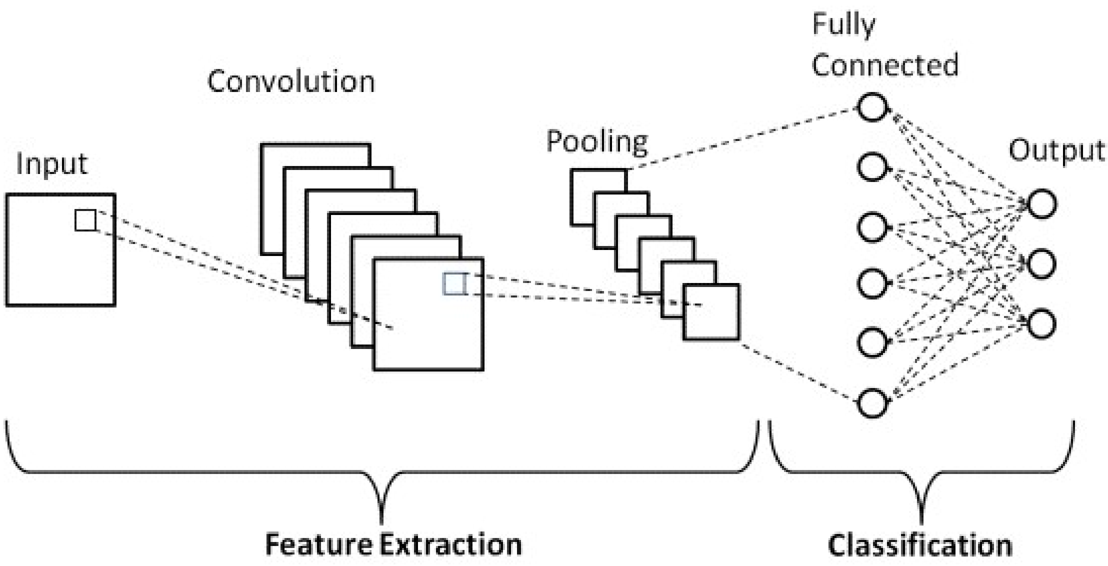


Figure 1CNN Architecture

1.3.2 Convolutional Layer:

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

1.3.3 Pooling Layer:

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of pooling operations. In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.

1.3.4 Fully Connected Layer:

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.
In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place

1.3.5Dropout:

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data.
To overcome this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

1.3.6 Activation Functions:

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network.
It adds non-linearity to the network. There are several commonly used activation functions such as the **ReLU, Softmax, tanH** and the Sigmoid functions. Each of these f unctions have a specific usage. For a binary classification CNN model, **sigmoid** and**softmax** functions are preferred an for a multi-class classification, generally **softmax** is used.

CNN Models

Alex Net(2012)

Alex Net was primarily designed by Alex Krizhevsky. It was published with Ilya Sutskever and Krizhevsky's doctoral advisor Geoffrey Hinton, and is a Convolution Neural Network or CNN.

Alex Net won the ImageNet Large Scale Visual Recognition Challenge 2012 by a phenomenally large margin. It consist of eight layers: five convolutional layers, two fully-connected hidden layers, and one fully-connected output layer. It was the first max-pooling layers, ReLu activation functions, and dropout for the 3 enormous linear layers. The network was used for image classification with 1000 possible classes, which for that time was madness.

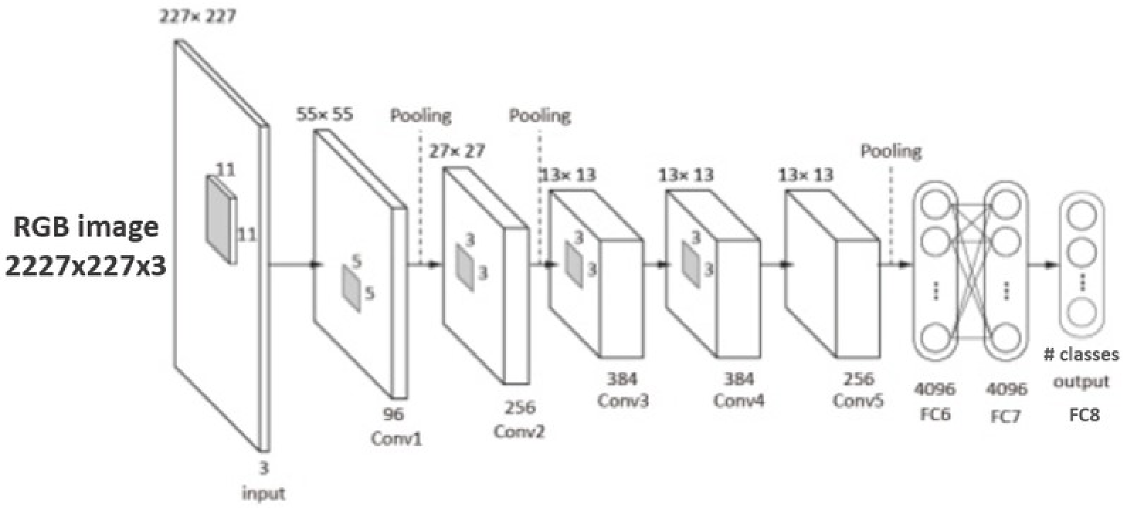


Figure 2Alex net Architecture

The primary result of the original paper was that the depth of the model was absolutely required for its high performance. This was quite expensive computationally but was made feasible due to GPUs or Graphical Processing Units, during training.
Modified Alex Net.

VGG 16 (2014):

VGG 16 was proposed by Karen Simony and Andrew Zisserman of the Visual Geometry Group Lab of Oxford University in 2014 in the paper “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION”. This model won the 1st and 2nd place on the above categories in 2014 ILSVRC challenge. It is considered to be one of the excellent vision model architectures till date.

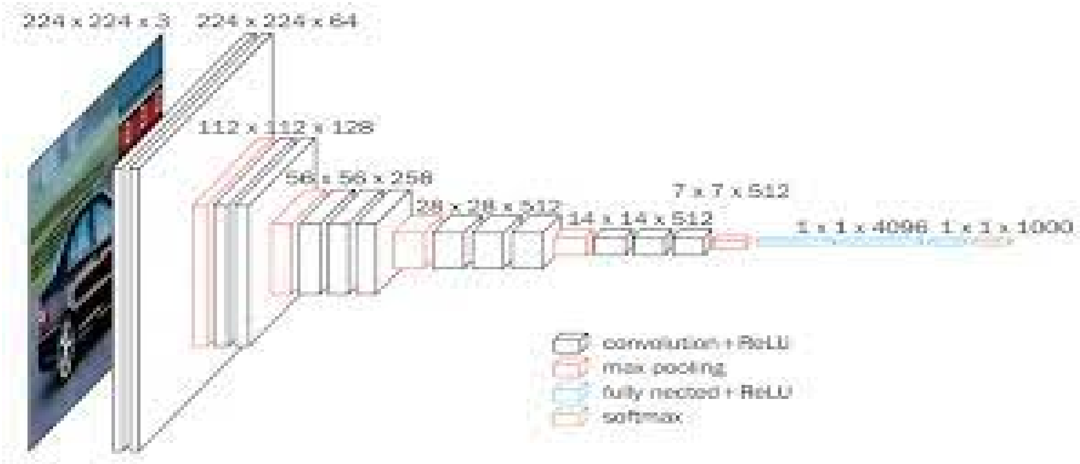


Figure 3VGG 16 Architecture

It was the runner up in classification task with top-5 classification error of 7.32% (only behind Google Net with classification error 6.66%). It was also the winner of localization task with 25.32% localization error.

Inception V1(2014):

Inception V1 is the earliest version of Google Net, appearing in 2014. The Inception Net architecture consists of 9 inception modules stacked together, with max-pooling layers between (to halve the spatial dimensions). It consists of 22 layers (27 with the pooling layers). It uses global average pooling after the last inception module. This indeed dramatically declines the total number of parameters.

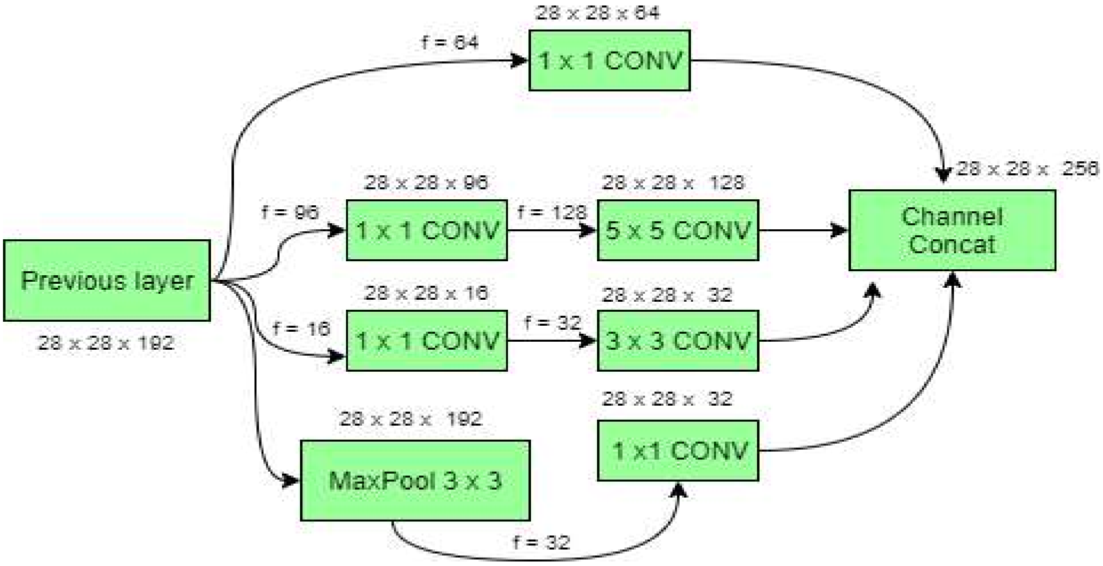


Figure 4Inception V1 Architecture

Thus, Inception Net is a victory over the previous versions of CNN models. It achieves an accuracy of top-5 on ImageNet, it reduces the computational cost to a great extent without compromising the speed and accuracy.

ResNet: Deep Residual Learning for Image Recognition (2015):

Residual Network (ResNet) is one of the famous deep learning models that was introduced by Shaoqing Ren, Kaiming He, Jian Sun, and Xiangyu Zhang in their paper. The paper was named "Deep Residual Learning for Image Recognition" [1] in 2015. The ResNetmodel is one of the popular and most successful deep learning models so far.

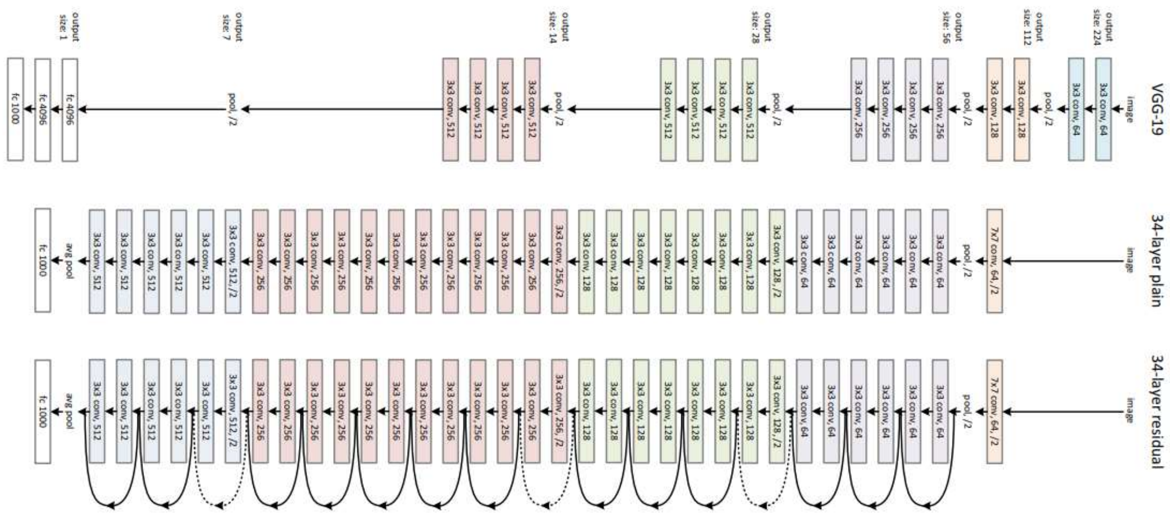


Figure 5Rest Net-34 Architecture

There is a 34-layer plain network in the architecture that is inspired by VGG-19 in which the shortcut connection or the skip connections are added. These skip connections or the residual blocks then convert the architecture into the residual network.

Dense Net: Densely Connected Convolutional Networks (2017):

A Dense Net is a type of convolutional neural network that utilizes dense connections between layers, through Dense Blocks, where we connect all layers (with matching feature-map sizes) directly with each other.To preserve the feed-forward nature, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers.

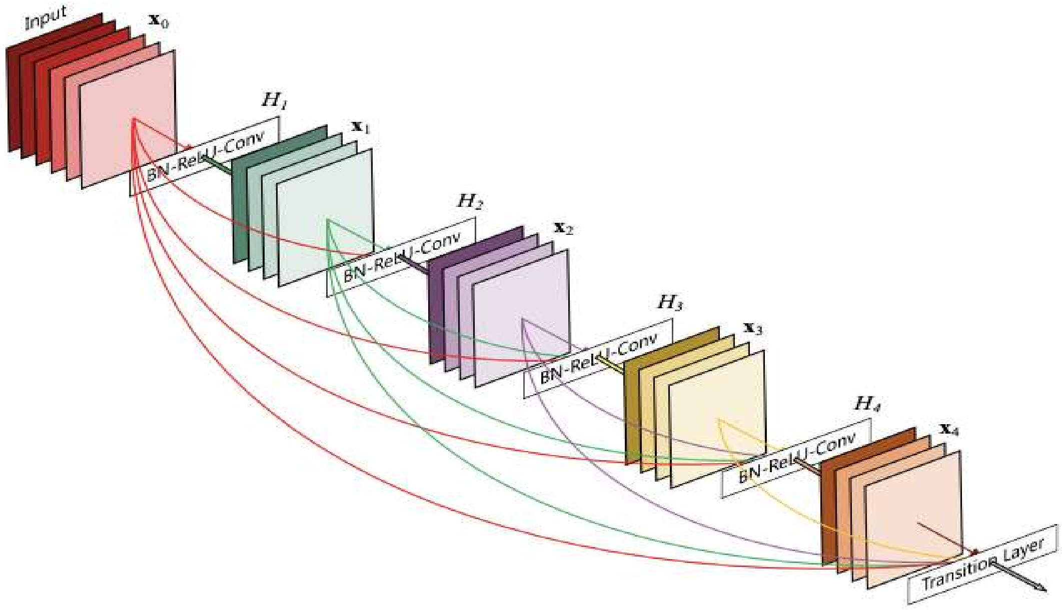


Figure 6Dense Net Architecture

With dense connection, fewer parameters and high accuracy are achieved compared with ResNetand Pre-Activation ResNet.

Efficient Net: Rethinking Model Scaling for Convolutional NeuralNetworks (2019):

Efficient Net is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. Unlike conventional practice that arbitrary scales these factors, the Efficient Net scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. Efficient Net is all about engineering and scale. It proves that if you carefully design your architecture you can achieve top results with reasonable parameters.

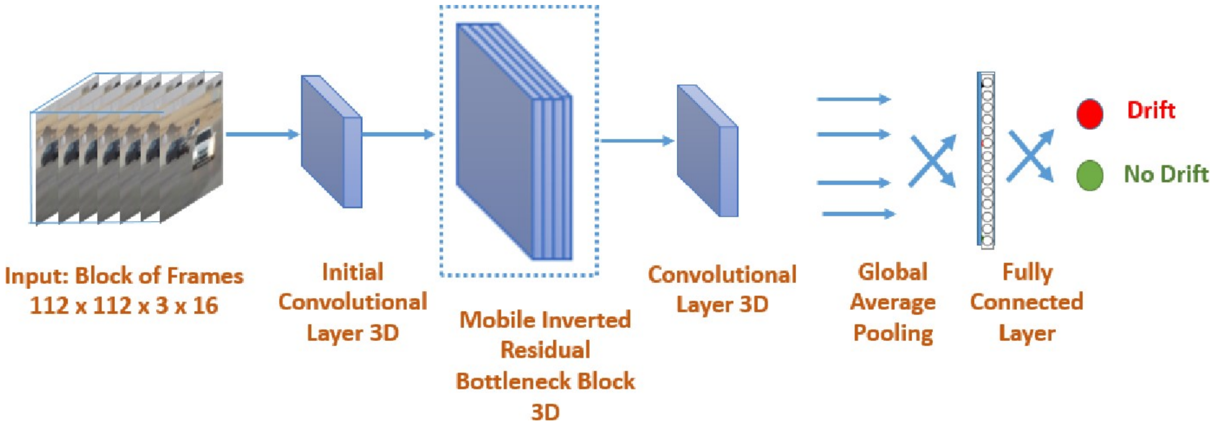


Figure 7Efficient Net Architecture

Model name	Number of parameters [Millions]	ImageNet Top 1 Accuracy	Year
AlexNet	60 M	63.3 %	2012
Inception V1	5 M	69.8 %	2014
VGG 16	138 M	74.4 %	2014
VGG 19	144 M	74.5 %	2014
Inception V2	11.2 M	74.8 %	2015
ResNet-50	26 M	77.15 %	2015
ResNet-152	60 M	78.57 %	2015
Inception V3	27 M	78.8 %	2015
DenseNet-121	8 M	74.98 %	2016
DenseNet-264	22M	77.85 %	2016
BiT-L (ResNet)	**928 M**	87.54 %	2019
NoisyStudent EfficientNet-L2	**480 M**	**88.4** %	2020
Meta Pseudo Labels	**480 M**	**90.2** %	2021

Figure 8 CNN Model

1.3.3. **MobileNetV2 Architecture**

MobileNetV2 is a powerful image classification tool. Tensor Flow provides the image weights in MobileNetV2, a lightweight CNN-based deep learning model. First, the MobileNetV2 base layer is removed, and a new trainable layer is added. The model analyzes the data and extracts the most relevant features from our images. There are 19 bottleneck layers in MobileNetV2. In the base model, we used OpenCV, which is based on the ResNet-10 architecture. To detect the face and mask from an image and a video stream, **OpenCV'sCaffemodel** is used. The mask detecting classifier receives the output face detected image. It allows for faster and more accurate detection of masks in video streaming.

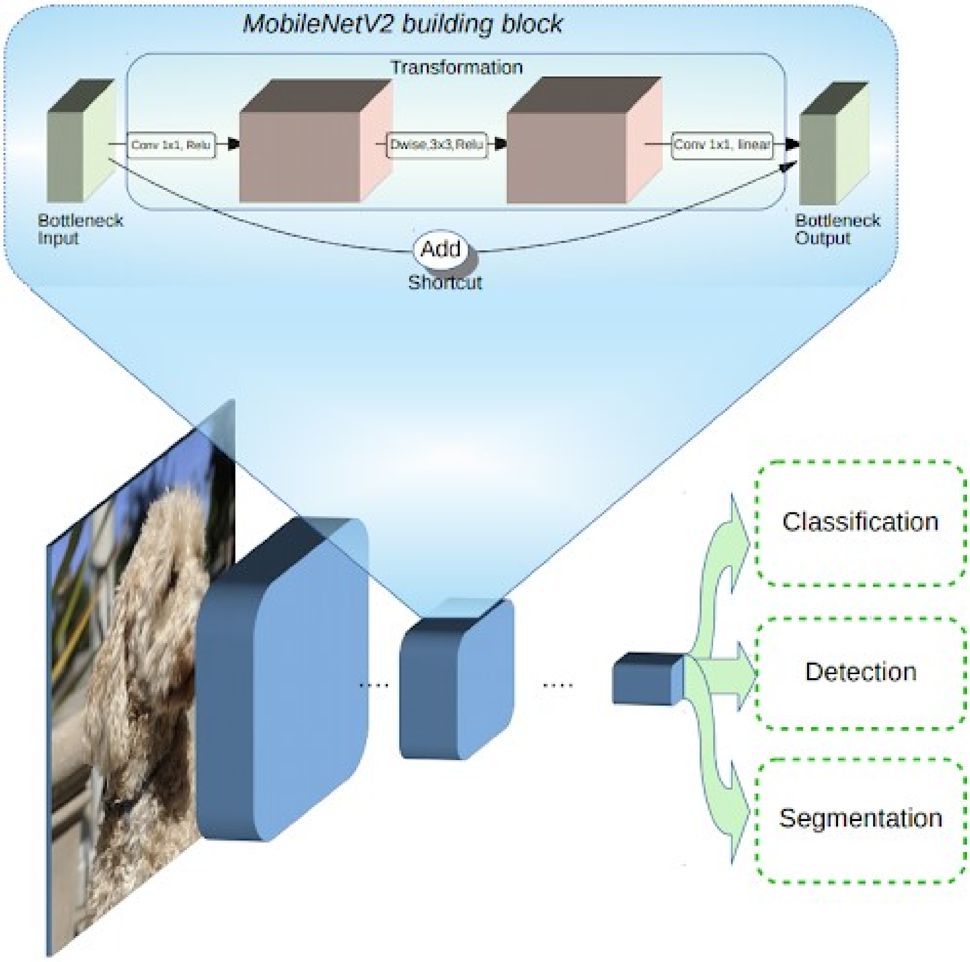


Figure 9Mobile NetV2 Architecture

2. Background & Literature Survey:

2.1. Literature Review:

Retina Facemask [1] is about a single stage face mask detector for assisting control of the Covid-19 Pandemic. They have used MAFA-FMS dataset. The annotations of the dataset have locations of faces, mask types, etc. The methodologies used in this projects are Network Architecture, Context Architecture Module, Transfer Learning, Training and Inference. They have also solved the issue to distinguish between correct and incorrect mask wearing states by establishing a new dataset containing these annotations. They also have emulated human'sability to transfer knowledge from the face detection task to improve face mask detection.

Another model[2], has been proposed to detect the face mask is put on or not for offices, or any other work place with a lot of people coming to work. They have used Convolutional Neural Network and MobileNetV2 architecture. They have used python libraries such as Tensor Flow, Karas and OpenCV. It can be used with any surveillance system. They also have the system that sends an alert message to the authorized person if someone has entered without a face mask. The accuracy rate with a face mask is 95-97% depending on the digital capabilities.

Deep Learning model [3] is a simple and effective model for real-time monitoring using the Convolution Neural Network (CNN) to detect whether an individual wears a mask or not. This model uses the Deep Learning DeBNet approach for feature extraction and classification and used SVM for proposing a machine learning based face detection and recognition system. It has been proposed for the students for monitoring their activities during online examination. This model is trained, validated, tested upon two datasets. Corresponding to the dataset 1, the accuracy of the model was 95.77% and it was 94.58% for dataset 2.

Another model has been proposed for face mask detection webcam based real world face mask detection[4] dataset of over 1TB of images collected across different regions of the United States by implementing state-of-the-art object detection algorithms to understand their effectiveness in such a real-world application. It provides the bigger picture of face mask usage in the United Sates. They have used concepts such as Machine Learning, Computer Vision and Neural Network. They have tested 12 different models to understand their efficacy and also utilized three model to label the remaining data to compare predicted mask usage trends and with another source of data.

Multi-Stage Architecture system[5] consists of a dual stage CNN architecture capable of detecting masked and unmasked faces and can be integrated with pre-installed CCTV cameras. This will help track safety violations, promote the use of face masks and ensure a safety working environment. Datasets were collected from public domain along with some data scraped from the internet. They used only pertained datasets for detection. It can be implemented by using any cameras to detect faces. It will be very useful for society and for peoples to prevent them from virus transmission. Here they have used live video detection using OpenCV.

Single Shot Detector architecture[6] is used for the object detection purpose. In this system face mask detector can be deployed in many areas like shopping malls, airports and other heavy traffic places to monitor the public and to avoid the spread of the disease by checking who is following basic rules and who is not. It takes excessive time for data loading in Google Collab Notebook. It did not allow the access of webcam which posed a hurdle in testing images and video stream. We have modeled a facemask detector using Deep learning. We are processed a system computationally efficient using MobileNetV2 which makes it easier to Extract the data sets. We use CNN architecture for better performance.

Raspberry pi based real time face mask recognition [7] that captures the facial image. This process gives a precise and speedily results for facemask detection. This system uses the architectural features of VGG-16 as the foundation network for face recognition. Deep learning techniques are applied to construct a classifier that will collect image of a person wearing a face mask and no masks. Our proposed study areuses the architectural features of CNN as the foundation network for face detection. It shows accuracy in detecting person wearing a face mask and not wearing a face mask.This study presence a useful tool in fighting the spread of covid 19 virus.

2.2. Requirement Specification:

2.2.1. Introduction:

In recent decades, facial recognition has become the object of research worldwide. In addition, with the advancement of technology and the rapid development of artificial intelligence, very significant advances have been made. For this reason, public and private companies use facial recognition systems to identify and control the access of people in airports, schools, offices, and other places. On the other hand, with the spread of the COVID-19 pandemic, government entities have established several biosafety regulations to limit infections. Among them is the mandatory use of face masks in public places, as they have proven to be an effective measure in protecting users and those around them. Corona virus has globally infected over 20 million people causing over 0.7 million deaths.

2.2.2. Definitions,Acronyms,andAbbreviations:

The acronyms that are constantly used in document include the following.
COVID- Coronavirus Disease.
API- Application Programming Interface.
IEEE- Institute of Electrical and Electronics Engineers.
UML- Unified Modelling Language.
CNN- Convolution Neural Network.
DNN- Deep Neural Network.

2.2.3. References:

Wikipedia: https://www.wikipedia.org/
Tutorials point: https://www.tutorialspoint.com/
Geeks for geeks: https://www.geeksforgeeks.org/
• IEEE Standard 830-1998 a Recommended Practice for Software Requirements Secifications

2.2.4. Overview:

The rest of the SRS document describes various system requirements, interfaces, features, andfunctionalitiesindetail.

2.2.5. OverallDescription:

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. In this paper, we propose Face Mask Recon, which is a high-accuracy and efficient face mask detector. The proposed Face Mask Recon is a one-stage detector, which consists of a feature pyramid network to fuse high-level semantic information with multiple feature maps, and a novel context attention module to focus on detecting face masks. It will help the system to run the overall system to prevent the spreading the Covid 19 and easy to control the mob in a cost effective way. An Iot Component will send a message to the concerned authority that it will help the entire system to function very smoothly.

2.2.6. Functional Requirements:

DATASET:

R1. Thesystem must have unbiased ‘with mask’ dataset.
R2. The dataset must have over 1500+ images in both ‘with mask’ and ‘without mask’ classes.
R3. The dataset must not re-use the same images in training and phase.

MASK DETECTOR:

R1. The system must be correctly able to load the face mask classifier model.
R2. The system must be able to detect faces in images or video stream**.**
R3. The system must be able to extract each face’s Region of interest.
R4. There must not be any object between the face of the user for a successful face detection and hence the face mask detection.
R5. The end position of the face must be fit inside the web came frame and must be closer to the camera.
R6. Correctly able to detect face mask in ‘png’, ‘jpg’, and ‘gif’ format images.
R7. The system must able to detect masks on human faces on every frame in a live video.
R8. The results must be viewed by showing the probability along with the output of ‘mask’ or No ‘mask’**.**

2.2.7. NON-FUNCTIONAL REQUIREMENTS:

- **Usability:**
  The user experience should be very smooth in using any provided feature.
- **Reliability:**
  FaceMask Detection should be available and satisfy every user’s need.
- **Performance:**
  The application should work efficiently so that there might not be any problem in using this platform.
- **Availability:**
  Face Mask Detection should be always available for the users. The server uptime should be around 99% allowing for a small time of down time for database upgrade and maintenance(if necessary).
- **Modularity:**
  The system will be designed in such a way that the algorithms for the four main units will be able to be easily swapped out.
- **Accuracy**:
  The overall accuracy of the Web API’s response will be measured using a developer-made testing set.
- **Fast Response:**
  The average time for the server to respond, over the question testing set, will be less than or equal to 2 seconds.
- **Security:**
  The connection between the Web API and the programs will use HTTPS, for security.

3. Design:

3.1 Class Diagram:

Class Diagram for the proposed system.

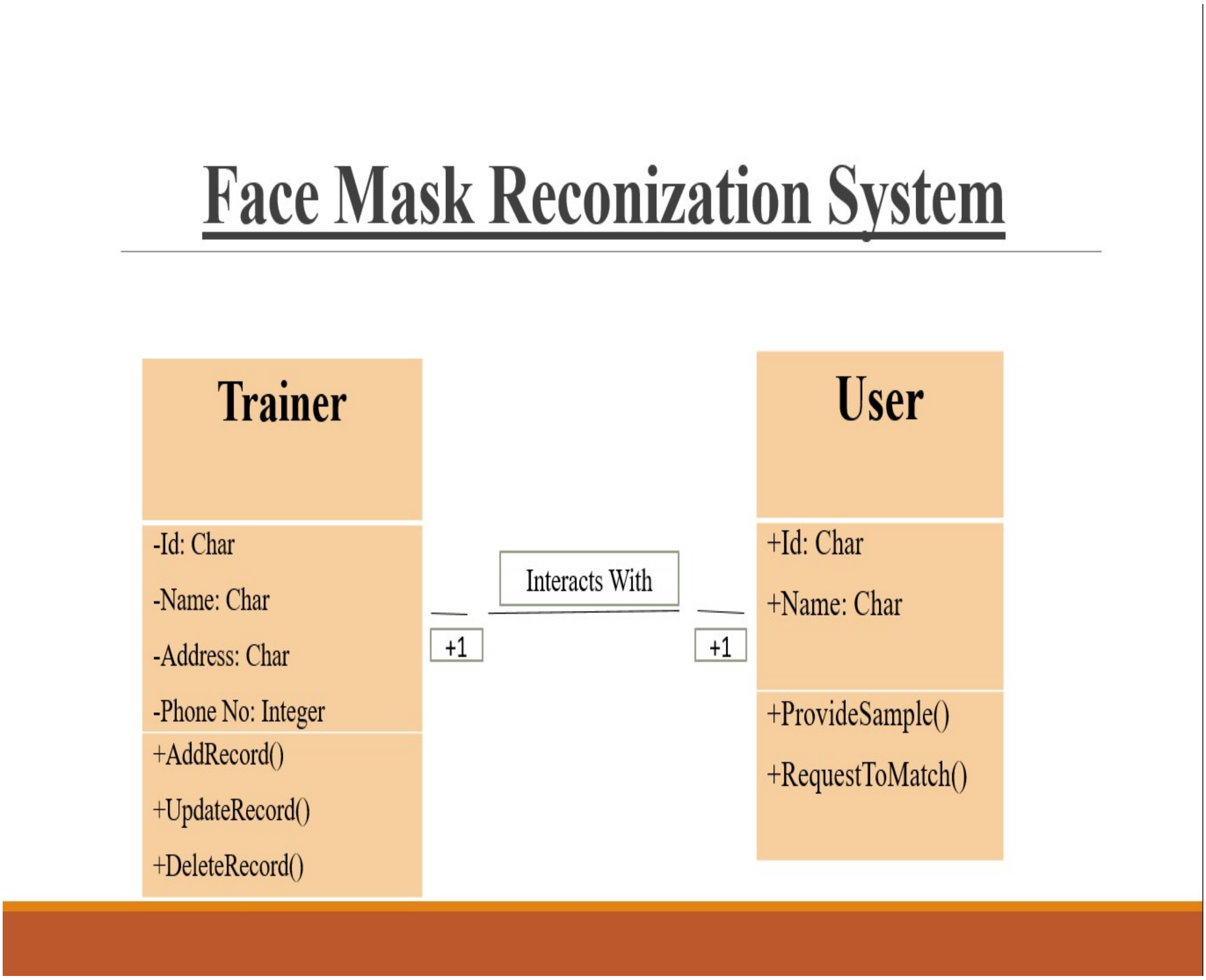


Figure 10Class Diagram

3.2 Flow Diagram

Flow diagram for the proposed Face Mask Detection system.

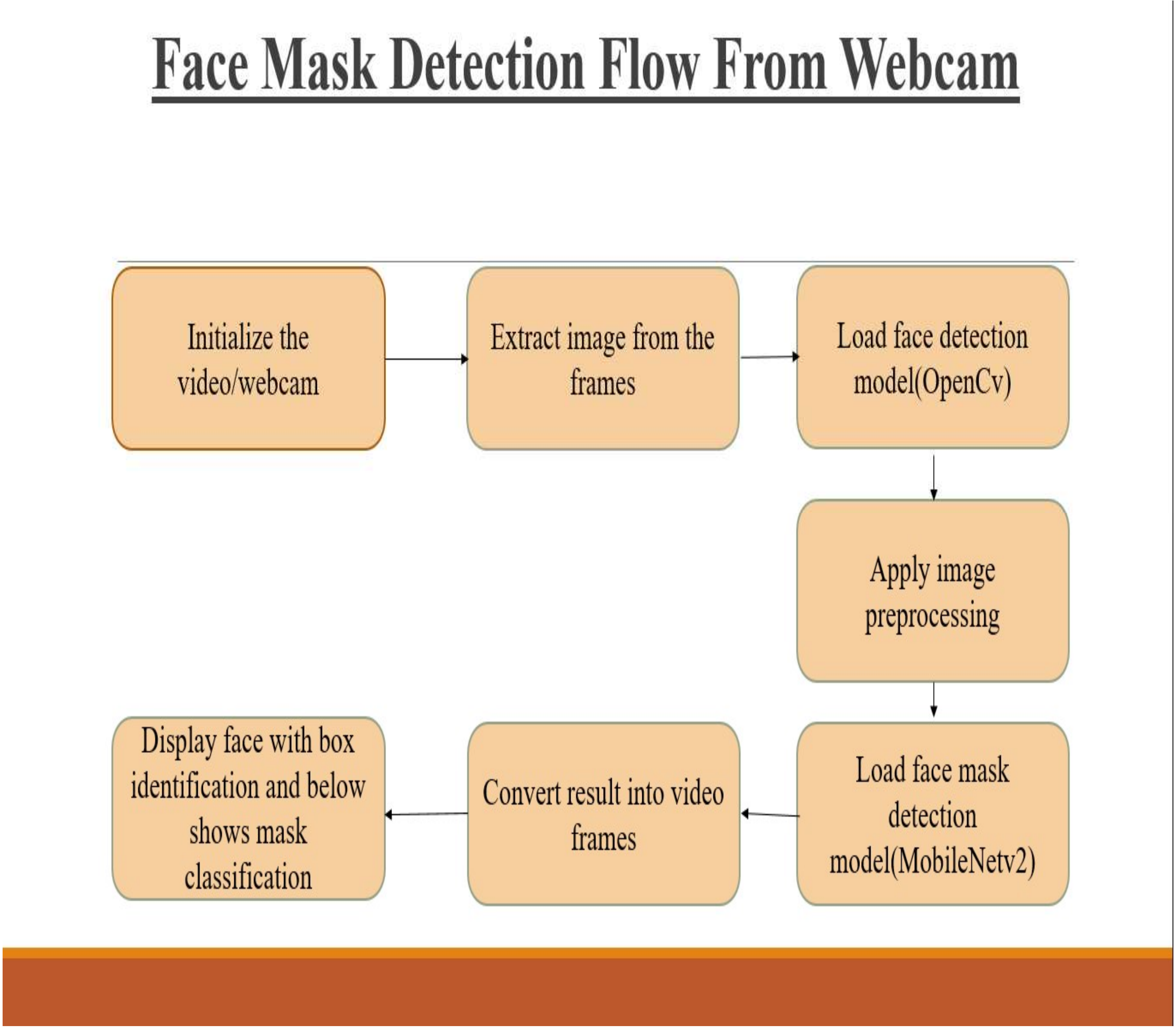


Figure 11 Flow Diagram

3.3 Proposed Model over Base Model:

```python
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)
```

Figure 12Base Model

3.3.1. Average Pooling:

Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

3.3.2. Flatten:

The flatten () function is used to get a copy of a given array collapsed into one dimension.

3.3.3. Dense:

A Dense layer feeds all outputs from the previous layer to all its neurons, each neuron providing one output to the next layer.

3.3.4. Dropout:

The term "dropout" is used for a technique which drops out some nodes of the network.

3.3.5. ReLU(Rectified Linear Unit) Activation Function:

The ReLU is the most used activation function in the world right now. It is almost used in CNN.

$$R(z) = \max(0, z)$$

Range $[0, \infty)$

The function and its derivative both are monotonic.

4. Technology & Tools Used :

The programming language used to develop this application is Python and the IDE used is Jupyter Notebook.

- Programming Language – Python
- Python IDE – VS Code, Jupyter Notebook
- Deep Learning Framework - Tensorflow.

4.1. Front-end Tools:

4.1.1. Voila:

Voila, an open-source python library. That is used to turn the**Jupyter** notebook into a standalone web application It supports widgets to create interactive dashboards, reports, etc. Voila converts the **Jupyter**notebook into **HTML** and returns it to the user as a dashboard or report with all the inputs excluded and the outputs included. Voila supports all the python libraries for widgets such as **bqplot**, **plotly**, **ipywidgets**, etc.

4.2. Back-end Tools:

4.2.1. TensorFlow:

Tensor Flow is an end-to-end open source platform for machine learning. It has an ecosystem of tools, libraries and community resources that helps is to implementing interfaces for expressing machine learning algorithms.
For example: - Handwritten digit recognition, face recognition. Some important API like: FAST API, CNTK (Microsoft Cognitive Toolkit)**.**

4.2.2. Keras:

Keras gives fundamental reflections and building units for creation and transportation of ML arrangements with high iteration velocity. It takes full advantage of the scalability and cross-platform capabilities of Tensor Flow. All the layers used in the CNN model are implemented using Keras.

4.2.3. OpenCV:

OpenCV(Open-Source Computer Vision Library) is an open source computer vision and ML software library, is utilized to differentiate and recognize faces, recognize objects, group movements in recordings, follow eye gesture, take camera actions etc. This method makes use of these features of OpenCV in resizing and color conversion of data images.

5. Hardware Requirement:

5.1. System Configuration (CPU, RAM, HDD)

- RAM - 8GB
- Storage- 512 GB
- Graphics - Integrated graphics
- Processor - AMD Ryzen 5 5500U
- Processor Speed - 2.10 GHz
- Operating System – Windows 10 & above.
- Web Browser – Google Chrome, Mozilla Firefox and Microsoft Edge

6. Source Code:

6.1. FRONT END

****

6.1.1 Home Page Code :

****

```
fromIPython.core.display import display, HTML

display(HTML('<head><meta name="viewpoint" content="width=device-width, initial-scale=1.0"
    /></head>'))

display(HTML('<h1><center>Face Mask Recognition System</center></h1>'))

display(HTML('<marquee style="color:red; font-size:15px; font-family: Book Antiqua"
    behavior="alternate">Wear your mask properly.</marquee>'))

display(HTML('<ul style="overflow:hidden; list-style-type: none; background-color:#36454F; text-
    color:white;"><li style=" float:left; padding: 16px;"><a href="Face_mask2.ipynb">
    Home</a></li><li style= "float:left; padding: 16px;"><a href="About2.ipynb">About
    Us</a></li></ul>'))

display(HTML('<center><img
    src="https://miro.medium.com/max/1200/1*fyfSOSKswsmV0n7Wdy6R4Q.jpeg"width="500"
    height="400"></center>'))

display(HTML('<center><h1>Live Face Mask Detection</h1></center>'))

display(HTML('<center><a href="detect.ipynb"><button>Start
    Camera</button></a></center><br>'))

display(HTML('<center><h5>To stop the camera, Press Esc.</h5></center><br><br>'))

display(HTML('<center><h1>How to Wear a Mask Correctly?</h1></center>'))

display(HTML('<div style="display: flex; align-items: center; flex-wrap: wrap; justify-content:
    space-around;"><img style="border: 5px solid #555;" src="https://encrypted-
    tbn0.gstatic.com/images?q=tbn:ANd9GcTxEysRHGAp1lmCYd4LUtY0eH0d8GG0MhW5XA&
    usqp=CAU"><p>According to the guidelines set by the World Health Organization (WHO)<br>
    a face mask needs to cover the face fully, including the<br> nose and the chin. Therefore, our
    detector only classifies <br>someone as wearing a mask if these conditions are satisfied. Our
    <br>software is also spoof proof which means that it understands<br> if you're covering your
    face with a hand or an object other than a mask.</p></div>'))

display(HTML('<center><h1>Features of Face Mask Recognition System</h1></center>'))
```

```
display(HTML('<div style="display: flex; align-items: center; flex-wrap: wrap; justify-content: space-around;"><div><h3>Multi-Channel Recognition</h3><br>Attach multiple cameras in a few minutes and enable<br> all the cameras to access the AI capability <br>of recognizing faces.
</div><div><h3>No new hardware to install</h3><br> The system can work on any existing RTSP camera<br> without the installation of any new cameras. Most of<br> the hospitals and airports have IP cameras installed<br> and RTSP-enabled.</div></div>'))
```

```
display(HTML('<br><br><br><center><h1>Use Cases of Face Mask Recognition System</h1>
        </center>'))
```

****

6.1.2 About Us Code :

****

```
fromIPython.core.display import display, HTML

display(HTML('<head><meta name="viewpoint" content="width=device-width, initial-scale=1.0" /></head>'))

display(HTML('<h1><center>Face Mask Recognition System</center></h1>'))

display(HTML('<ul style="overflow:hidden; list-style-type: none; background-color:#36454F; color:white;"><li style=" float:left; padding: 16px;"><a href="Face_mask2.ipynb"> Home</a></li><li style= "float:left; padding: 16px;"><a href="About2.ipynb">About Us</a></li></ul>'))

display(HTML('<p><center>In the face of the COVID-19 pandemic, the World Health Organization (WHO) declared the use of a face mask as a mandatory<br> biosafety measure. Wearing a mask is among the non-pharmaceutical measures that can be used to cut the primary source of COVID<br> droplets expelled by an infected individual. To contribute towards communal health, this project aims to devise a highly <br>accurate and real-time technique that can efficiently detect non-mask faces in public and thus, enforce them to wear masks. </center></p>'))

display(HTML('<p><center>The implementation is done in Python, and the python script implementation will train our face mask detector on our <br>selected dataset using TensorFlow and Keras. We have added more robust features and trained our model on various variations, we made<br> sure to have large varied and augmented dataset so that the model is able to clearly identify and detection the face masks in<br> real time videos.</center></p>'))

display(HTML('<p></p>'))

display(HTML('<center><h2>Graphical of Training Loss and Accuracy</h2></center>'))

    display(HTML('<center><img src="plot-MSI.png"></center>'))
```

6.2. BACK END

****

6.2.1 Live Mask Detection :

****

```
#import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
fromtensorflow.keras.preprocessing.image import img_to_array
fromtensorflow.keras.models import load_model
fromimutils.video import VideoStream
importnumpy as np
importimutils
import time
import cv2
importos
defdetect_and_predict_mask(frame, faceNet, maskNet):
    # grab the dimensions of the frame and then construct a blob from it
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
        (104.0, 177.0, 123.0))
    # pass the blob through the network and obtain the face detections
    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)
    # initialize our list of faces, their corresponding locations,
    # and the list of predictions from our face mask network
    faces = []
    locs = []
    preds = []
```

```
# loop over the detections
fori in range(0, detections.shape[2]):
    # extract the confidence (i.e., probability) associated with the detection
    confidence = detections[0, 0, i, 2]
    # filter out weak detections by ensuring the confidence is
    # greater than the minimum confidence
    if confidence > 0.5:
        # compute the (x, y)-coordinates of the bounding box for the object
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")
        # ensure the bounding boxes fall within the dimensions of the frame
        (startX, startY) = (max(0, startX), max(0, startY))
        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
        # extract the face ROI, convert it from BGR to RGB channel
        # ordering, resize it to 224x224, and preprocess it
        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)
        # add the face and bounding boxes to their respective lists
        faces.append(face)
        locs.append((startX, startY, endX, endY))
# only make a predictions if at least one face was detected
iflen(faces) > 0:


# for faster inference we'll make batch predictions on all
```

```
# faces at the same time rather than one-by-one predictions
# in the above `for` loop
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)
# return a 2-tuple of the face locations and their corresponding locations
    return (locs, preds)
# load our serialized face detector model from disk
prototxtPath = r"C:/Users/risha/Desktop/Final_Project/Face-Mask-
    Detection/face_detector/deploy.prototxt"
weightsPath = r"C:/Users/risha/Desktop/Final_Project/Face-Mask-
    Detection/face_detector/res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
# load the face mask detector model from disk
maskNet = load_model("C:/Users/risha/Desktop/Final_Project/Face-Mask-
    Detection/mask_detector.model")
# initialize the video stream
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)
    # detect faces in the frame and determine if they are wearing a
    # face mask or not
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
```

```
# loop over the detected face locations and their corresponding locations
    for (box, pred) in zip(locs, preds):
        # unpack the bounding box and predictions
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred
        # determine the class label and color we'll use to draw the bounding box and text
        label = "MASK" if mask >withoutMask else "NO MASK"
        color = (0, 255, 0) if label == "MASK" else (0, 0, 255)
        # include the probability in the label
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
        # display the label and bounding box rectangle on the output frame
        cv2.putText(frame, label, (startX, startY - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
    # show the output frame
    cv2.imshow("Frame", frame)
# set Escape key as exit key
    if cv2.waitKey(1) & 0xFF ==27 :
break
cv2.destroyAllWindows()
    vs.stop()
```

****

6.2.2 Model Training:

****

 # import the necessary packages

```
fromtensorflow.keras.preprocessing.image import ImageDataGenerator
fromtensorflow.keras.applications import MobileNetV2
fromtensorflow.keras.layers import AveragePooling2D
fromtensorflow.keras.layers import Dropout
fromtensorflow.keras.layers import Flatten
fromtensorflow.keras.layers import Dense
fromtensorflow.keras.layers import Input
fromtensorflow.keras.models import Model
fromtensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
fromtensorflow.keras.preprocessing.image import img_to_array
fromtensorflow.keras.preprocessing.image import load_img
fromtensorflow.keras.utils import to_categorical
fromsklearn.preprocessing import LabelBinarizer
fromsklearn.model_selection import train_test_split
fromsklearn.metrics import classification_report
fromimutils import paths
importmatplotlib.pyplot as plt
importnumpy as np
importos
```

 # initialize the initial learning rate, number of epochs to train for, and batch size

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

```
DIRECTORY = r"C:/Users/risha/Desktop/Final_Project/Face-Mask-Detection/dataset"
CATEGORIES = ["with_mask", "without_mask"]
# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")
data = []
labels = []
for category in CATEGORIES:
path = os.path.join(DIRECTORY, category)
forimg 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)
# perform one-hot encoding on the labels
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)
```

```
# construct the training image generator for data augmentation
    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")
  # load the MobileNetV2 network, ensuring the head FC layer sets are left off
  baseModel = MobileNetV2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))
  # construct the head of the model that will be placed on top of the base model
  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)
  # place the head FC model on top of the base model (this will become
  # the actual model we will train)
  model = Model(inputs=baseModel.input, outputs=headModel)
  # loop over all layers in the base model and freeze them so they will
  # not be updated during the first training process
  for layer in baseModel.layers:
    layer.trainable = False
```

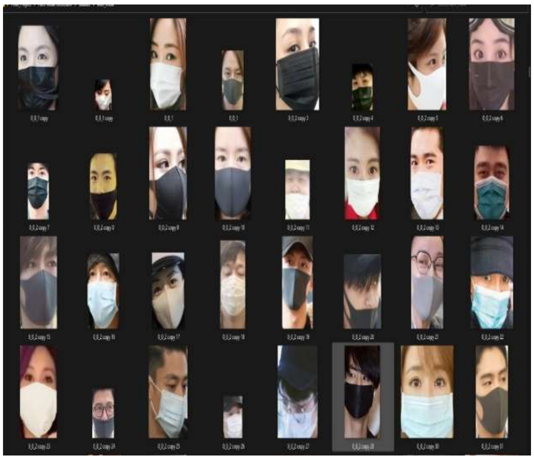
```
# compile our model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["accuracy"])
# train the head of the network
print("[INFO] training head...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)
# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)
# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
    target_names=lb.classes_))
# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")
```

```
# plot the training loss and accuracy
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")
```

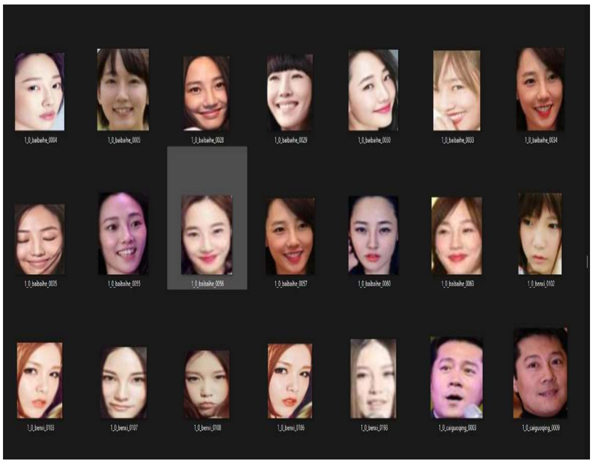
7. Snapshots:

7.1. Data Collection:

Snapshots below illustrates an example of faces wearing and not wearing masks. The experiments of this research are conducted on one dataset taken from **Kaggle**.The dataset used in this research was collected in various picture formats such as JPEG, PNG, and others. It consists totally of **3833** images. This is a balanced dataset containing two categories, faces with masks (**1915** images) and without masks (**1918** images) with a mean height of 283.68 and mean width of 278.77. It comprises two categories. This dataset is used not only for training and validation, but also for testing, and if an individual is wearing a mask or not.



Snapshot 1: Faces with Masks



Snapshot 2: Faces without Masks

[Directory 1 containing images of Faces with Masks. Size 1900 images approximately]

[Directory 2 containing images of Faces without Masks. Size 1900 images approximately]

7.2. Model Training:

The training model is trained using 20 Epochs each containing 95 data elements. During the Model Evaluation significant improvement on loss, accuracy, value loss and value accuracy can be observed using the methods such as accuracy, precision, recall, fi-score and support.

```
Epoch 1/20
2022-04-08 00:58:41.766500: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 51380224 exceeds 10% of free system memory.
2022-04-08 00:58:41.835425: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 51380224 exceeds 10% of free system memory.
2022-04-08 00:58:41.908047: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 154140672 exceeds 10% of free system memory.
2022-04-08 00:58:41.959875: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 156905472 exceeds 10% of free system memory.
2022-04-08 00:58:42.033547: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 38535168 exceeds 10% of free system memory.
95/95 [==============================] - 79s 833ms/step - loss: 0.4607 - accuracy: 0.7881 - val_loss: 0.1431 - val_accuracy: 0.9739
Epoch 2/20
95/95 [==============================] - 78s 820ms/step - loss: 0.1427 - accuracy: 0.9545 - val_loss: 0.0720 - val_accuracy: 0.9870
Epoch 3/20
95/95 [==============================] - 74s 781ms/step - loss: 0.0906 - accuracy: 0.9687 - val_loss: 0.0565 - val_accuracy: 0.9883
Epoch 4/20
95/95 [==============================] - 76s 799ms/step - loss: 0.0755 - accuracy: 0.9746 - val_loss: 0.0446 - val_accuracy: 0.9909
Epoch 5/20
95/95 [==============================] - 77s 813ms/step - loss: 0.0604 - accuracy: 0.9773 - val_loss: 0.0392 - val_accuracy: 0.9922
Epoch 6/20
95/95 [==============================] - 77s 809ms/step - loss: 0.0545 - accuracy: 0.9819 - val_loss: 0.0366 - val_accuracy: 0.9922
Epoch 7/20
95/95 [==============================] - 77s 815ms/step - loss: 0.0464 - accuracy: 0.9852 - val_loss: 0.0332 - val_accuracy: 0.9948
Epoch 8/20
95/95 [==============================] - 77s 814ms/step - loss: 0.0473 - accuracy: 0.9848 - val_loss: 0.0323 - val_accuracy: 0.9935
Epoch 9/20
95/95 [==============================] - 81s 848ms/step - loss: 0.0411 - accuracy: 0.9858 - val_loss: 0.0369 - val_accuracy: 0.9909
Epoch 10/20
95/95 [==============================] - 78s 821ms/step - loss: 0.0395 - accuracy: 0.9885 - val_loss: 0.0291 - val_accuracy: 0.9935
Epoch 11/20
95/95 [==============================] - 77s 805ms/step - loss: 0.0360 - accuracy: 0.9865 - val_loss: 0.0285 - val_accuracy: 0.9935
Epoch 12/20
95/95 [==============================] - 76s 805ms/step - loss: 0.0335 - accuracy: 0.9904 - val_loss: 0.0258 - val_accuracy: 0.9935
Epoch 13/20
95/95 [==============================] - 76s 799ms/step - loss: 0.0308 - accuracy: 0.9901 - val_loss: 0.0280 - val_accuracy: 0.9935
Epoch 14/20
95/95 [==============================] - 76s 801ms/step - loss: 0.0300 - accuracy: 0.9904 - val_loss: 0.0354 - val_accuracy: 0.9883
Epoch 15/20
95/95 [==============================] - 75s 790ms/step - loss: 0.0303 - accuracy: 0.9895 - val_loss: 0.0268 - val_accuracy: 0.9935
Epoch 16/20
95/95 [==============================] - 75s 786ms/step - loss: 0.0275 - accuracy: 0.9918 - val_loss: 0.0299 - val_accuracy: 0.9935
Epoch 17/20
95/95 [==============================] - 74s 780ms/step - loss: 0.0254 - accuracy: 0.9908 - val_loss: 0.0259 - val_accuracy: 0.9935
Epoch 18/20
95/95 [==============================] - 75s 785ms/step - loss: 0.0207 - accuracy: 0.9924 - val_loss: 0.0241 - val_accuracy: 0.9935
Epoch 19/20
95/95 [==============================] - 75s 788ms/step - loss: 0.0228 - accuracy: 0.9914 - val_loss: 0.0241 - val_accuracy: 0.9948
Epoch 20/20
95/95 [==============================] - 75s 793ms/step - loss: 0.0296 - accuracy: 0.9914 - val_loss: 0.0248 - val_accuracy: 0.9935
```

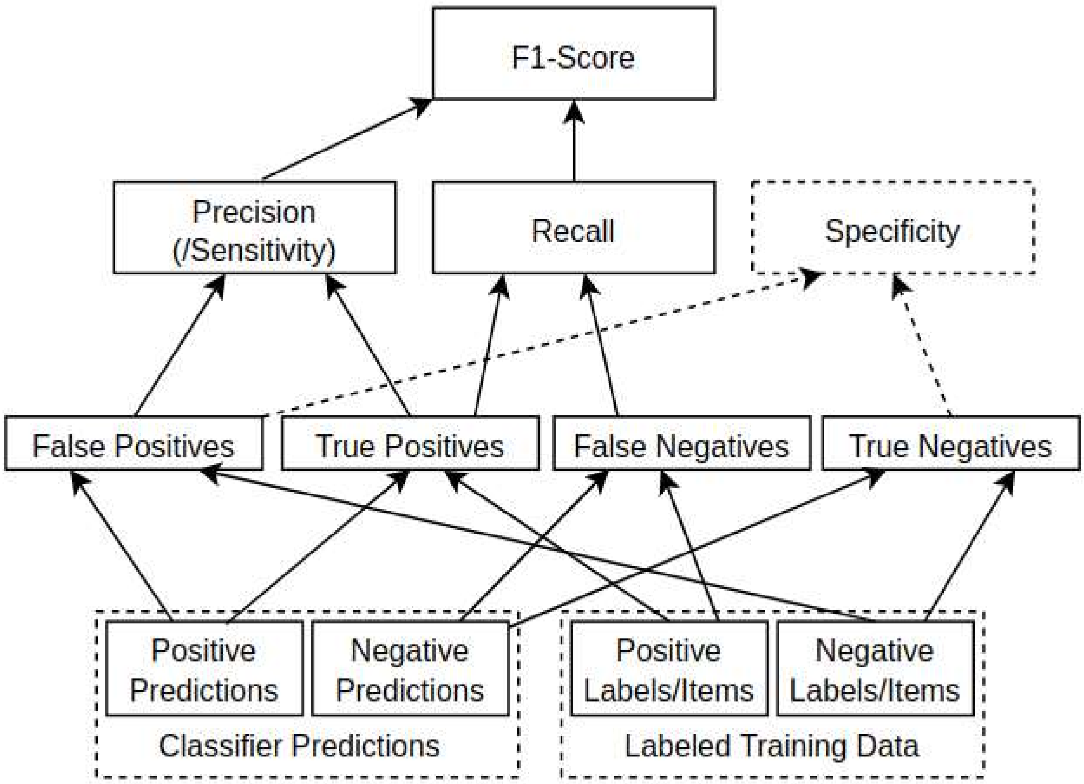
Snapshot 3 : Model Trained for 20 Epochs

```
[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
   macro avg       0.99      0.99      0.99       767
weighted avg       0.99      0.99      0.99       767
```

Snapshot 4 : Classification Report

7.3. Methods:

Methods used for Model Evaluation are Accuracy, Precision, Recall, FI- score & Support. These relate to getting a finer-grained idea of how well a classifier is doing, as opposed to just looking at overall accuracy.



Hierarchy of Metrics from raw measurements / labeled data to F1-Score.

Snapshot 4

True/False Positives and Negatives

A binary classifier can be viewed as classifying instances as positive or negative:

7. **Positive**: The instance is classified as a member of the class the classifier is trying to identify. For example, a classifier looking for cat photos would classify photos with cats as positive (when correct).
8. **Negative**: The instance is classified as not being a member of the class we are trying to identify. For example, a classifier looking for cat photos should classify photos with dogs (and no cats) as negative.

The basis of precision, recall, and F1-Score comes from the concepts of *True Positive*, *True Negative*, *False Positive*, and *False Negative*.

Prediction	Actual value	Type	Explanation
1	1	True Positive	Predicted Positive and was Positive
0	0	True Negative	Predicted Negative and was Negative
1	0	False Positive	Predicted Positive but was Negative
0	1	False Negative	Predicted Negative but was Positive

Examples of True/False Positive and Negative

True Positive (TP)

The following table shows 3 examples of a True Positive (TP). The first row is a generic example, where 1 represents the Positive prediction. The following two rows are examples with labels. Internally, the algorithms would use the 1/0 representation, but I used labels here for a more intuitive understanding.

Prediction	Actual value	Type	Explanation
1	1	True Positive	Predicted Positive and was Positive
Cat	Cat	True Positive	Cat classifier: Predicted a Cat and it was an actual cat
Cancer	Cancer	True Positive	Cancer classifier: Predicted Cancer and patient really had Cancer

Examples of True Positive (TP) relations.

False Positive (FP)

These False Positives (FP) examples illustrate making wrong predictions, predicting Positive samples for an actual Negative samples. Such failed prediction is called False Positive.

Prediction	Actual value	Type	Explanation
1	0	False Positive	Predicted Positive but was Negative
Cat	No Cat	False Positive	Cat classifier: Predicted a Cat but it was not a cat (maybe it was a dog)
Cancer	No Cancer	False Positive	Cancer classifier: Predicted Cancer but the patient did not have cancer

Examples of False Positive (FP) relations

True Negative (TN)

For the True Negative (TN) example, the cat classifier correctly identifies a photo as not having a cat in it, and the medical image as the patient having no cancer. So, the prediction is Negative and correct (True).

Prediction	Actual value	Type	Explanation
0	0	True Negative	Predicted Negative and was Negative
No Cat	No Cat	True Negative	Cat classifier: Predicted No Cat and it was not a cat (maybe it was a dog)
No Cancer	No Cancer	True Negative	Cancer classifier: Predicted No Cancer and patient had no cancer

Examples of True Negative (TN) relations.

False Negative (TN)

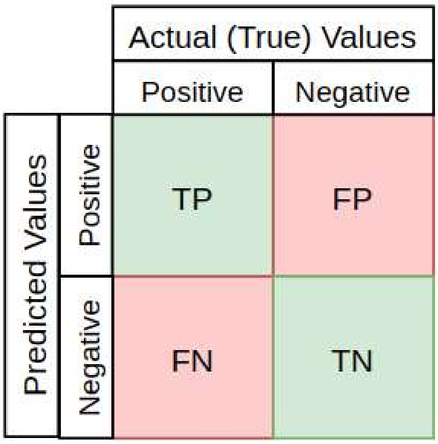
In the False Negative (FN) case, the classifier has predicted a Negative result, while the actual result was positive. Like no cat when there is a cat. So the prediction was Negative and wrong (False). Thus it is a False Negative.

Prediction	Actual value	Type	Explanation
0	1	False Negative	Predicted Negative and was Positive
No Cat	Cat	False Negative	Cat classifier: Predicted No Cat but there actually was a cat
No Cancer	Cancer	False Negative	Cancer classifier: Predicted No Cancer but the patient really had cancer

Examples of False Negative (FN) relations.

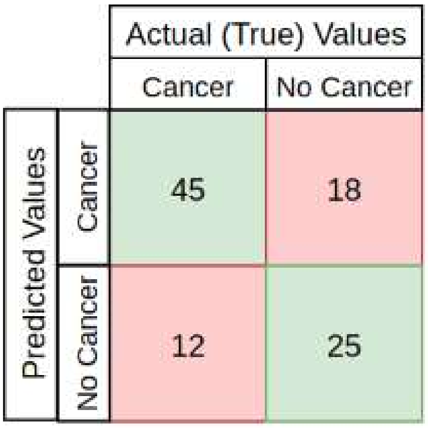
Confusion Matrix

A confusion matrix is sometimes used to illustrate classifier performance based on the above four values (TP, FP, TN, FN). These are plotted against each other to show a confusion matrix:



Snapshot 5: Confusion Matrix. Image by Author.

Using the cancer prediction example, a confusion matrix for 100 patients might look something like this



Snapshot 6: Confusion matrix for the cancer example.

This example has:

9. TP: 45 positive cases correctly predicted
10. TN: 25 negative cases correctly predicted
11. FP: 18 negative cases are misclassified (wrong positive predictions)
12. FN: 12 positive cases are misclassified (wrong negative predictions)

Accuracy

The base metric used for model evaluation is often *Accuracy*, describing the number of correct predictions over all predictions:

$$\frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}} = \frac{\text{N. of Correct Predictions}}{\text{N. of all Predictions}} = \frac{\text{N. of Correct Predictions}}{\text{Size of Dataset}}$$

Accuracy Formula

Snapshot 7

These three show the same formula for calculating accuracy, but in different wording. From more formalized to more intuitive (my opinion). In the above cancer example, the accuracy would be:

- (TP+TN)/Dataset Size = (45+25)/100=0.7=70%.

This is perhaps the most intuitive of the model evaluation metrics, and thus commonly used. But often it is useful to also look a bit deeper.

Precision

Precision is a measure of how many of the positive predictions made are correct (true positives). The formula for it is:

$$\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} = \frac{\text{N. of Correctly Predicted Positive Instances}}{\text{N. of Total Positive Predictions you Made}} = \frac{\text{N. of Correctly Predicted People with Cancer}}{\text{N. of People you Predicted to have Cancer}}$$

Precision formulas.

Snapshot 8

All three above are again just different wordings of the same, with the last one using the cancer case as a concrete example. In this cancer example, using the values from the above example confusion matrix, the precision would be:

- (TP)/ (TP + FP) = 45/ (45+18)=45/63=0.714=71.4%.

Recall / Sensitivity

Recall is a measure of how many of the positive cases the classifier correctly predicted, over all the positive cases in the data. It is sometimes also referred to as *Sensitivity*. The formula for it is:

$$\frac{\text{True Positives}}{\text{True Positives + False Negatives}} = \frac{\text{N. of Correctly Predicted Positive Instances}}{\text{N. of Total Positive Instances in the Dataset}} = \frac{\text{N. of Correctly Predicted People with Cancer}}{\text{N. of People with Cancer in the Dataset}}$$

Recall formulas.

Snapshot 9

Once again, this is just the same formula worded three different ways. For the cancer example, using the confusion matrix data, the recall would be:

- (TP)/ (TP + FN) = (45/ (45+12)=45/57=0.789=78.9%.

F1-Score

F1-Score is a measure combining both precision and recall. It is generallydescribed as the harmonic mean of the two. Harmonic mean is just another way to calculate an “average” of values, generally described as more suitable for ratios (such as precision and recall) than the traditional arithmetic mean. The formula used for F1-score in this case is:

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

F1-Score formula

Snapshot 10

The idea is to provide a single metric that weights the two ratios (precision and recall) in a balanced way, requiring both to have a higher value for the F1-score value to rise. For example, a Precision of 0.01 and Recall of 1.0 would give:

- an arithmetic mean of (0.01+1.0)/2=0.505,
- F1-score score (formula above) of 2*(0.01*1.0)/ (0.01+1.0) =~0.02.

This is because the F1-score is much more sensitive to one of the two inputs having a low value (0.01 here). Which makes it great if you want to balance the two.

Macro Average

The method is straightforward. Just take the average of the precision and recall of the system on different sets. The Macro-average will be simply the average mean of Macro-average precision and macro-average recall.

Weighted Average

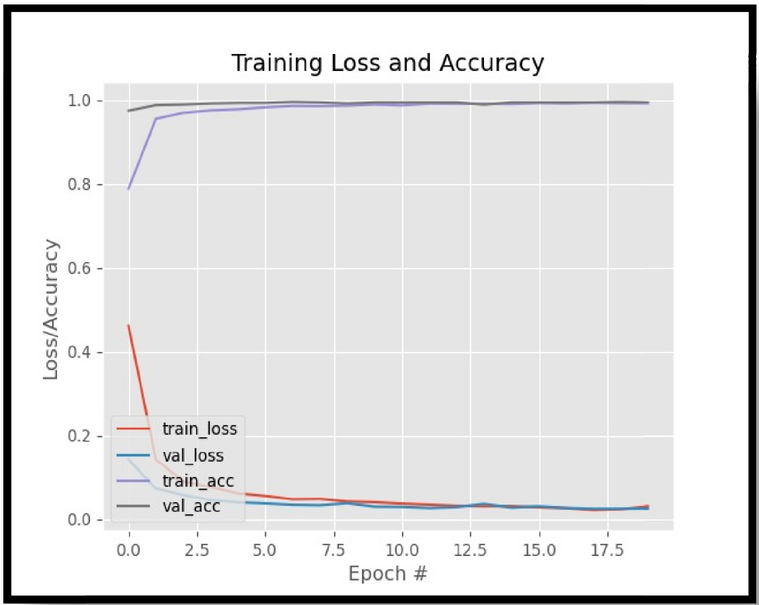
The F1 Scores are calculated for each label and then their average is weighted by support - which is the number of true instances for each label. It can result in an F1Score that is not between precision and recall.

7.4. Graph Plotted Against Data

Here the results from **8.2** training the model evaluation are plotted and stored in a graphical format using the Python library **matplotlib.pyplot**twice and improvement on training model once versus training the model twice is clearly observable to us.

Snapshot 11 : Result after training once

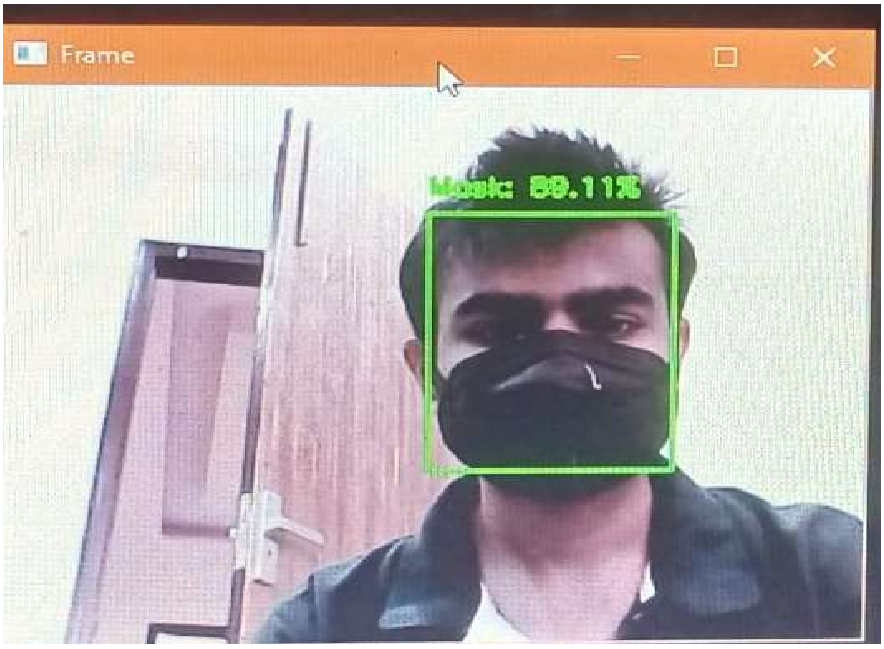
VS



Snapshot 12 :Improvement over training twice

7.5. Face Mask Detection on Live Webcam:

Snapshot of actual working Mask Detector Frame window



Snapshot 13

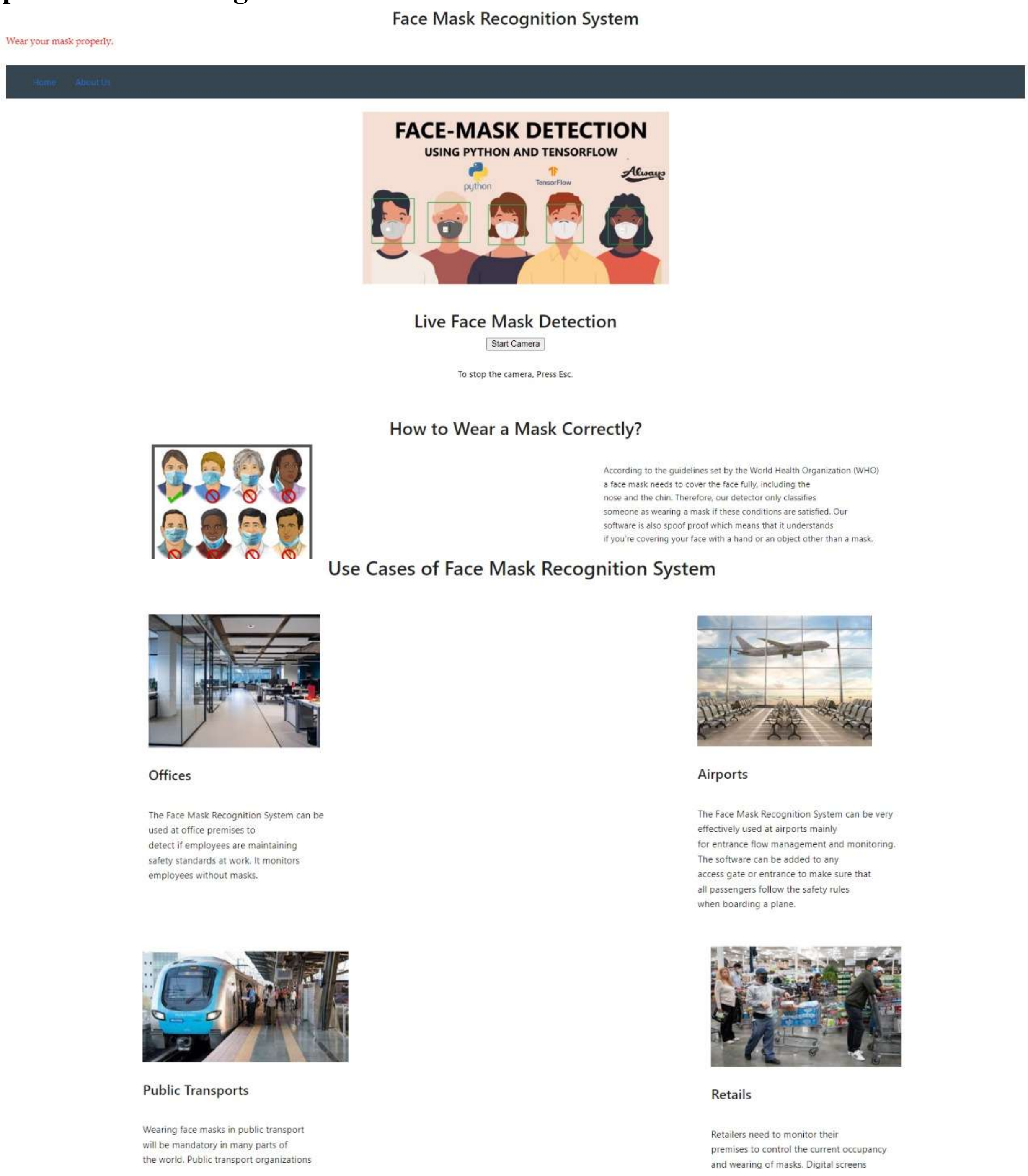
Snapshot of Frontend code for Home Page

```
from IPython.core.display import display, HTML
display(HTML('<head><meta name="viewpoint" content="width=device-width, initial-scale=1.0" /></head>'))
display(HTML('<h1><center>Face Mask Recognition System</center></h1>'))
display(HTML('<marquee style="color:red; font-family: Book Antiqua" behavior="alternate">Wear your mask properly.</marquee>'))
display(HTML('<ul style="overflow:hidden; list-style-type: none; background-color:#36454F; text-color:white;"><li style=" float:

#display(HTML('<img src="https://sightcorp.com/wp-content/uploads/2021/01/Sightcorp-x11-min-768x512.png.webp">'))
#display(HTML('<img src="https://d3lkc3n5th01x7.cloudfront.net/wp-content/uploads/2020/03/27034046/offices.jpg">'))

display(HTML('<center><h1>Live Face Mask Detection</h1></center>'))
display(HTML('<center><a href="detect.ipynb"><button>Start Camera</button></a></center><br>'))
display(HTML('<center><h5>To stop the camera, Press Esc.</h5></center><br><br>'))
display(HTML('<center><h1>How to Wear a Mask Correctly?</h1></center>'))
display(HTML('<div style="display: flex; align-items: center; flex-wrap: wrap; justify-content: space-around;"><img style="border
display(HTML('<center><h1>Features of Face Mask Recognition System</h1></center>'))
display(HTML('<div style="display: flex; align-items: center; flex-wrap: wrap; justify-content: space-around;"><div><h3>Multi-Cha
display(HTML('<br><br><br><center><h1>Use Cases of Face Mask Recognition System</h1></center>'))
display(HTML('<div style="display: flex; align-items: center; flex-wrap: wrap; justify-content: space-around;"><div><h3>Offices</
display(HTML('<div style="display: flex; align-items: center; flex-wrap: wrap; justify-content: space-around;"><div><h3>Public Tr
```

Snapshot 14

Versus

Snapshot of Home Page



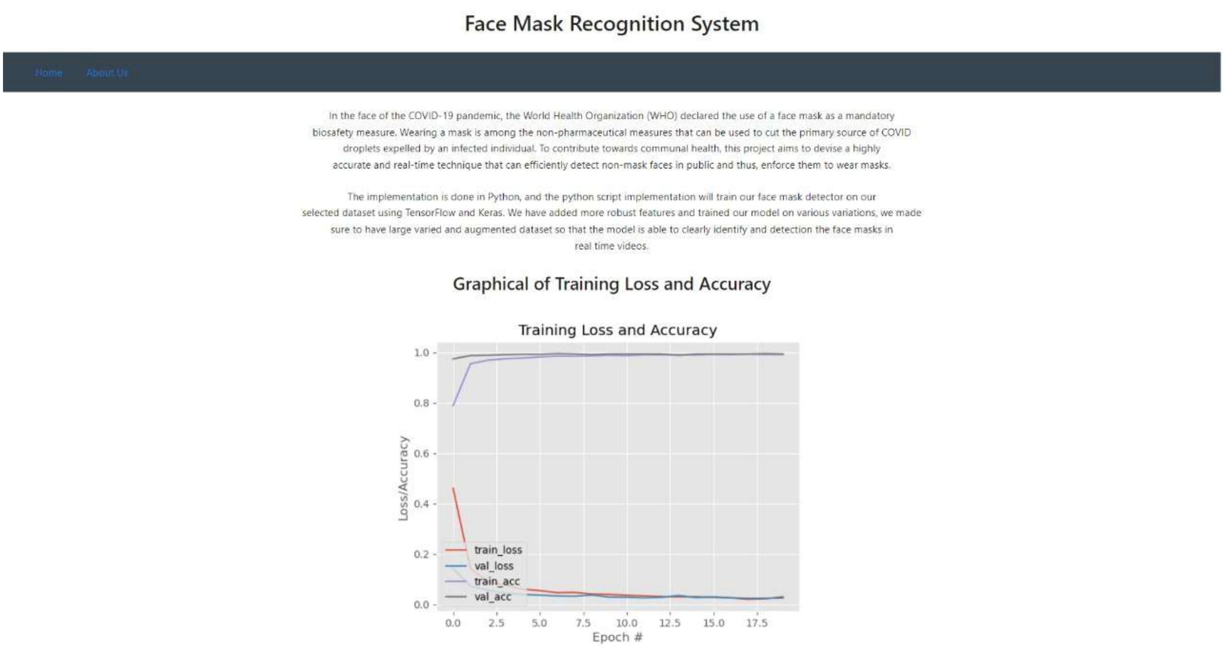
Snapshot 15

Snapshot of Frontend code for About Us

```
from IPython.core.display import display, HTML
display(HTML('<head><meta name="viewpoint" content="width=device-width, initial-scale=1.0" /></head>'))
display(HTML('<h1><center>Face Mask Recognition System</center></h1>'))
display(HTML('<ul style="overflow:hidden; list-style-type: none; background-color:#36454F; color:white;"><li style=" float:left;
display(HTML('<p><center>In the face of the COVID-19 pandemic, the World Health Organization (WHO) declared the use of a face mas
display(HTML('<p><center>The implementation is done in Python, and the python script implementation will train our face mask dete
display(HTML('<p></p>'))
display(HTML('<center><h2>Graphical of Training Loss and Accuracy</h2></center>'))
display(HTML('<center><img src="plot-MSI.png"></center>'))
```

Snapshot 16

Snapshot of About Us



Snapshot 17

8. Limitations:

8.1. Highly dependent on camera:

8.1.1. Range of camera.

8.1.2. Visibility issue in dark environment.

8.1.3. Capacity of camera to monitor large base of crowd.

8.2. Can only classify as 0 or 1 i.e. mask worn or not warn nothing in middle.

8.3. Cannot identify the credentials of people due to lack of people's database.

9. Conclusion:

To moderate the spread of the COVID-19 pandemic, measures should be taken. Wearing mask is the safety precaution that has to be implemented by all to avoid getting infected but still people are avoiding the use of masks so we are developing the face mask detection that can detect if the person is wearing mask or not. The accuracy of the model will be achieved and the optimization of the model is a continuous process and so we are building a highly accurate solution. We are mainly focusing this to implement this in schools and colleges and to ensure the safety of all and prevent them from virus transmission.

10. Future Scope:

The future of facial mask recognition technology is bright. Forecasters opine that this technology is expected to grow at a formidable rate and will generate huge revenues in the coming years. Security and surveillances are the major segments which will be deeply influenced. Other areas that are now welcoming it with open arms are private industries, public buildings, and schools. It is estimated that it will also be adopted by retailers and banking systems in coming years to keep fraud in debit/credit card purchases and payment especially the ones that are online. This technology would fill in the loopholes of largely prevalent inadequate password system. In the long run, robots using facial recognition technology may also come to foray. They can be helpful in completing the tasks that are impractical or difficult for human beings to complete.

11. Bibliography:

[1] Xingi Fan, MingjieJiang "Retina Facemask: A Single Stage Facemask Detector for Assisting Control of the Covid-19 Pandemic"

[2]Riya Chiragkumar Shah, RutvaJigneshShah Detection of Face Mask using Convolutional Neural Network

[3] SusantoSusanto,FebriAlwanPutra,Riska Analia, IkaKarlina Laila NurSuciningtyas The Face Mask Detection For Preventing the Spread of COVID-19

[4]EashanAdhikarla, Brian D. Davidson Face Mask Detection on Real-World Webcam Images 2021

[5] Amit Chavda, Jason Dsouza, SumeetBadgujar Multi-Stage CNN Architecture for Face Mask Detection September 2020

[6]Raza Ali, Saniya Adeel, Akhyar Ahmed Face Mask Detector July 2020

[7]Sammy v. militante,Nanettev.dionisioReal time face mask recognition with alarm system using deep learning 2020