FACE MASK DETECTION SYSTEM

A PROJECT REPORT
Submitted By

SHIKHA CHAUDHARY

(2000290140112)

ARSHI GOEL

(2000290140026)

SHREYA GAUR
(2000290140115)

Submitted in partial fulfillment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Dr. Akash Rajak
Supervisor

Submitted to
DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

CERTIFICATE

Certified that Shikha Chaudhary (2000290140112), Arshi Goel (2000290140026), Shreya Gaur (2000290140115) have carried out the project work having "Face Mask Detection" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:25 May 2022

Shikha Chaudhary (2000290140112)

Arshi Goel (2000290140026)

Shreya Gaur (2000290140115)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:25 May 2022

Dr. Akash Rajak
Supervisor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner

Signature of External Examiner

Dr. Ajay Shrivastava
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, Dr. Akash Rajak for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot in many critical conditions. Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work wouldn't have been possible in time. They keep my life filled with enjoyment and happiness.

Shikha Chaudhary (2000290140112)

Arshi Goel (2000290140026)

Shreya Gaur (2000290140115)

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.

Table of Contents

1. Introduction

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, Tensorflow and Keras. We will achieve high accuracy while consuming least time in the process of training and detection in our model.

Scope:

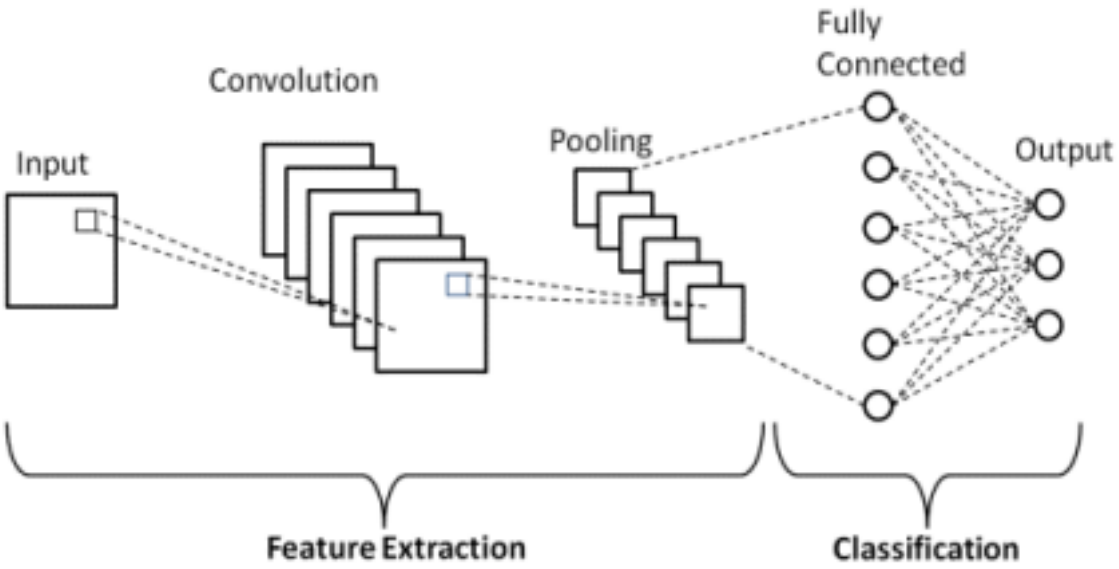
Video is captured and then it is converted into frames, then it will automatically recognize the faces with masks and without masks. Whenever a face is detected without a mask a message will be sent to the concerned authority. It is base on computer vision technology, enabling computers to understand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

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

Methodology:

Machine learning techniques are used in our system with CNN. Convolutional Neural Networks is used for image extraction and its correspondent labeling.

1.3.1.1 Convolution Neural Network(CNN)



Convolution Layers:

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, a CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function which are defined below.

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.

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 the method used, there are several types of Pooling operations.In Max Pooling, the largest element is taken from the feature map. Average Pooling calculates the average of the elements in a predefined size 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.

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

Dropout:

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 new data.

To overcome this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during the 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

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 functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred and for a multi-class classification, generally softmax is used.

CNN Models

AlexNet(2012)

AlexNet 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.

AlexNet won the ImageNet Large Scale Visual Recognition Challenge 2012 by a phenomenally large margin. It consists of eight layers: five convolutional layers, two fully-connected hidden layers, and one fully-connected output layer. It was the first max-pooling layer, 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.

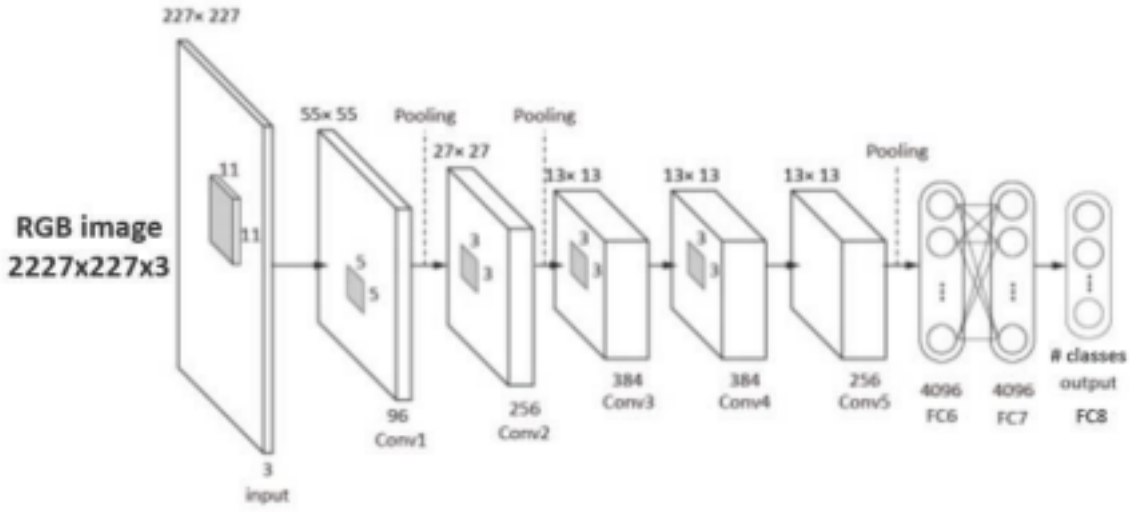


Figure 1 Alexnet Architecture

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.

VGG 16 (2014):

VGG 16 was proposed by Karen Simonyan 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 in the above categories in the 2014 ILSVRC challenge. It is considered to be one of the excellent vision model architectures till date.

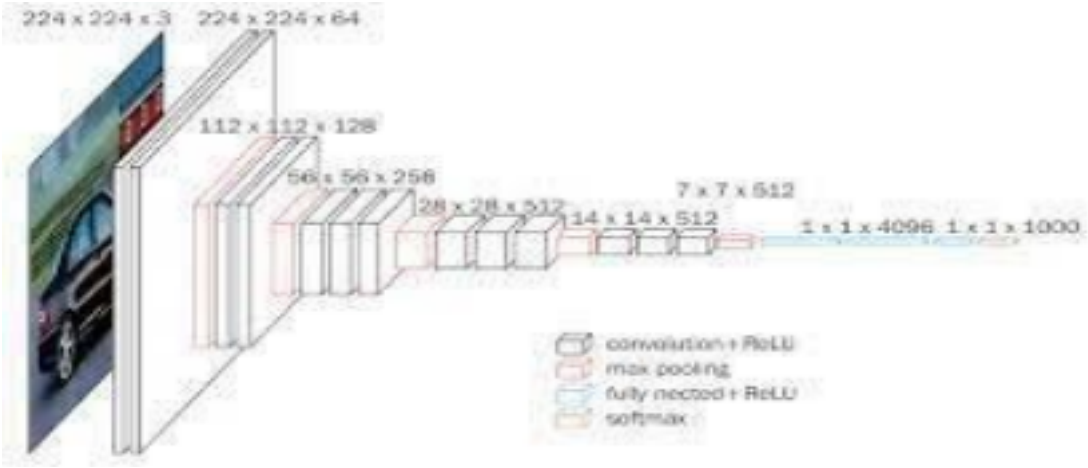


Figure 2 VGG 16 Architecture

It was the runner up in classification task with top-5 classification error of 7.32% (only behind GoogLeNet 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 GoogleNet, appearing in 2014. The InceptionNet 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.

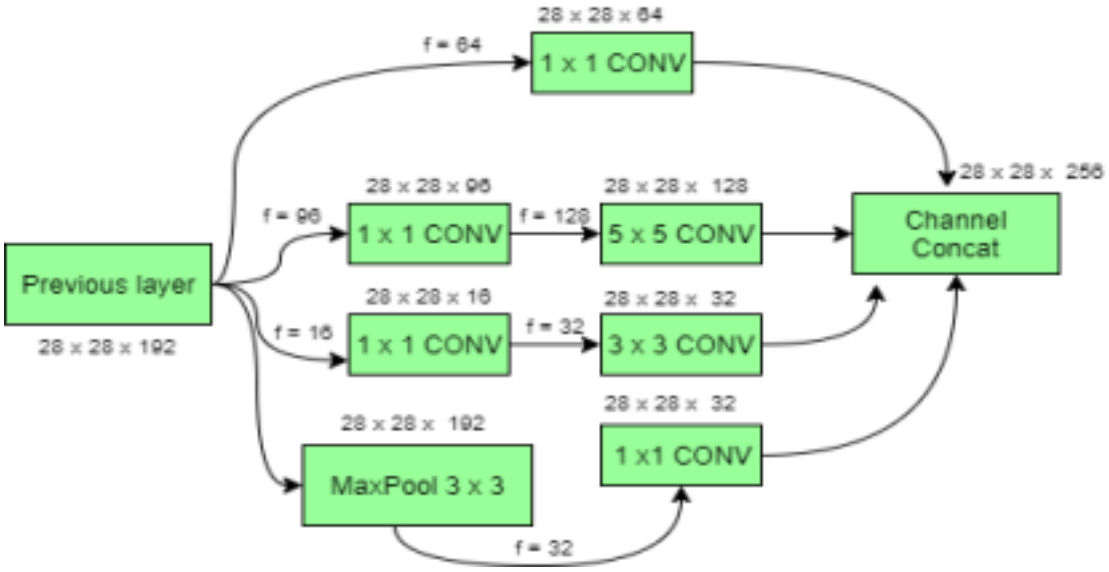


Figure 3 Inception V1 Architecture

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 ResNet model is one of the popular and most successful deep learning models so far.

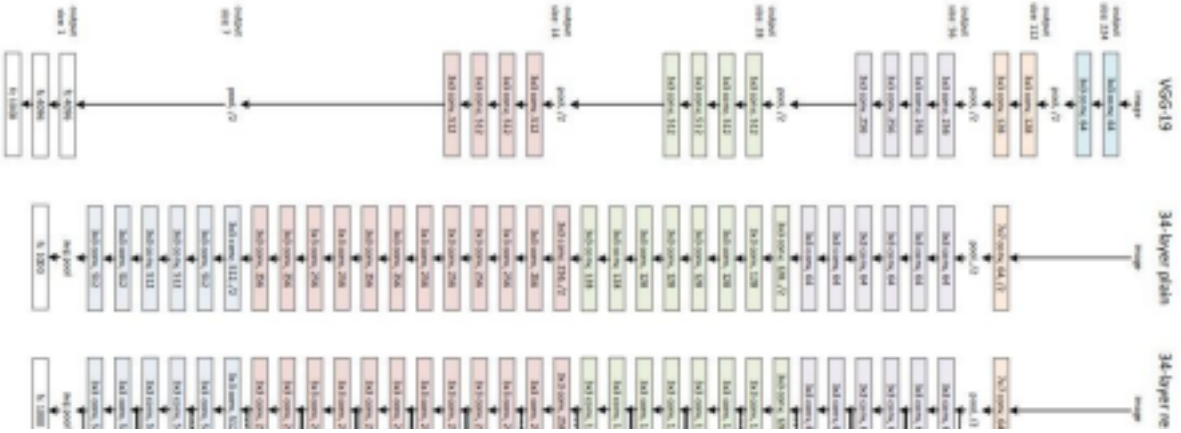


Figure 4 ResNet -34 Architecture

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.

DenseNet: Densely Connected Convolutional Networks (2017): A DenseNet 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.

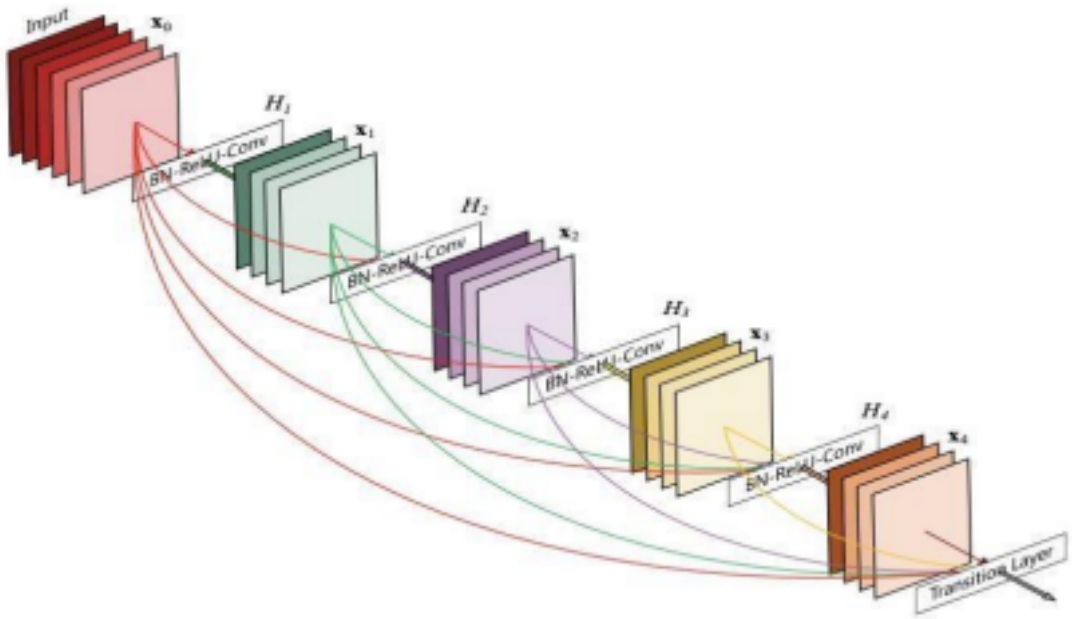


Figure 5 DenseNet Architecture

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

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (2019):

EfficientNet 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 EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients.

EfficientNet is all about engineering and scale. It proves that if you carefully design your architecture you can achieve top results with reasonable parameters.

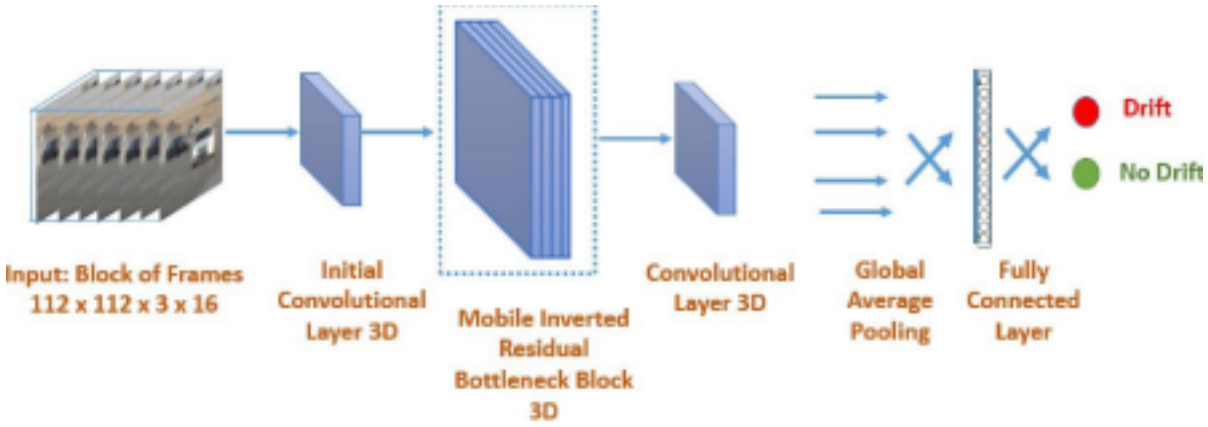


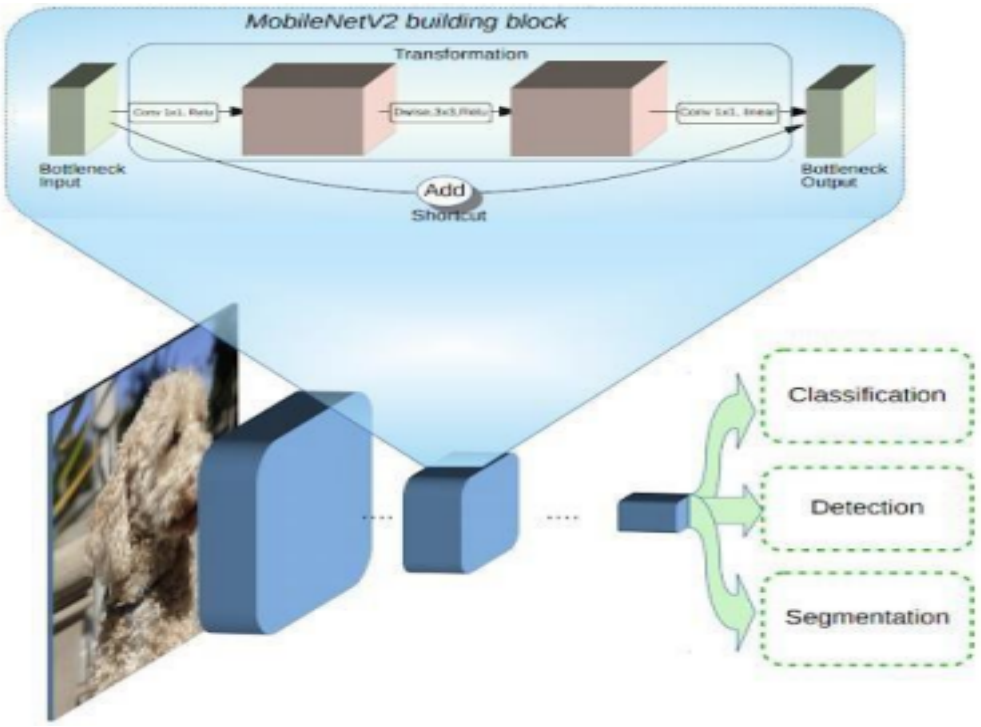
Figure 6 EfficientNet Architecture

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 7 CNN Models

1.3.1.2 MobileNetV2 Architecture

MobileNetV2 is a powerful image classification tool. TensorFlow 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's Caffemodel 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.



2. Background & Literature Survey

Literature Review

Retina Facemask[1] is about a single stage face mask detector for assisting control of the Covid-19 Pandemic. They have used the 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 humans ability to transfer knowledge from the face detection task to improve face mask detection.

Another model[2], has been proposed to detect whether the face mask is put on or not for offices, or any other work place with a lot of people coming to work. They used Convolutional Neural Network and MobileNetV2 architecture. They have used python libraries such as TensorFlow, Keras 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 uses SVM for proposing a machine learning based face detection and recognition system. It has been proposed for the students to monitor their activities during online examinations. This model is trained, validated, tested upon two datasets. Corresponding to 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 States. 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 models 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 safe working environment. Datasets were collected from the public domain along with some data scraped from the internet. They used only pretrained 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 detectors 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 streams. 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 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 images of a person wearing a face mask and no masks. Our proposed study uses the architectural features of CNN as the foundation network for face detection. It shows accuracy in detecting a person wearing a face mask and not wearing a face mask .This study presents a useful tool in fighting the spread of covid 19 virus.

Requirement Specification

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.

Definitions, Acronyms, and Abbreviations –
The acronyms that are constantly used in documents include the following. COVID- Coronavirus Disease.
API- Application Programming Interface.
IEEE- Institute of Electrical and Electronics Engineers.
UML- Unified Modeling Language.
CNN- Convolutional Neural Network.

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 Specifications.

Overview

The rest of the SRS document describes various system requirements, interfaces, features, and functionalities in detail.

Overall Description

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 of 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. Product Perspective

Functional Requirements:

DATASET:

R1. The system must have an 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 streams**.**

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 webcam frame and must be closer to the camera.

R6. Correctly able to detect face masks in 'png', 'jpg' , and 'gif' format images.

R7. The system must be able to detect masks on human faces on every frame in a live video.

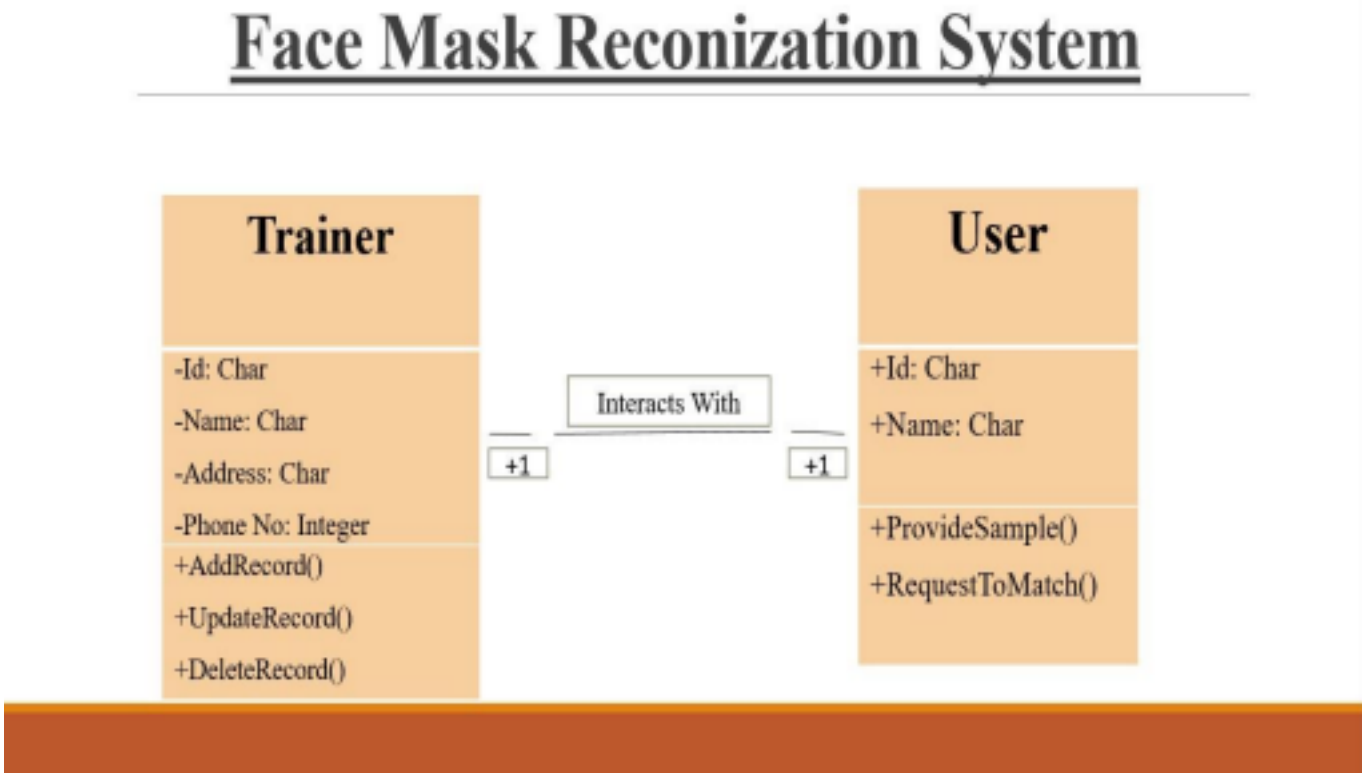
R8. The results must be viewed by showing the probability along with the output of 'mask' or No 'mask'**.**

NON-FUNCTIONAL REQUIREMENTS:

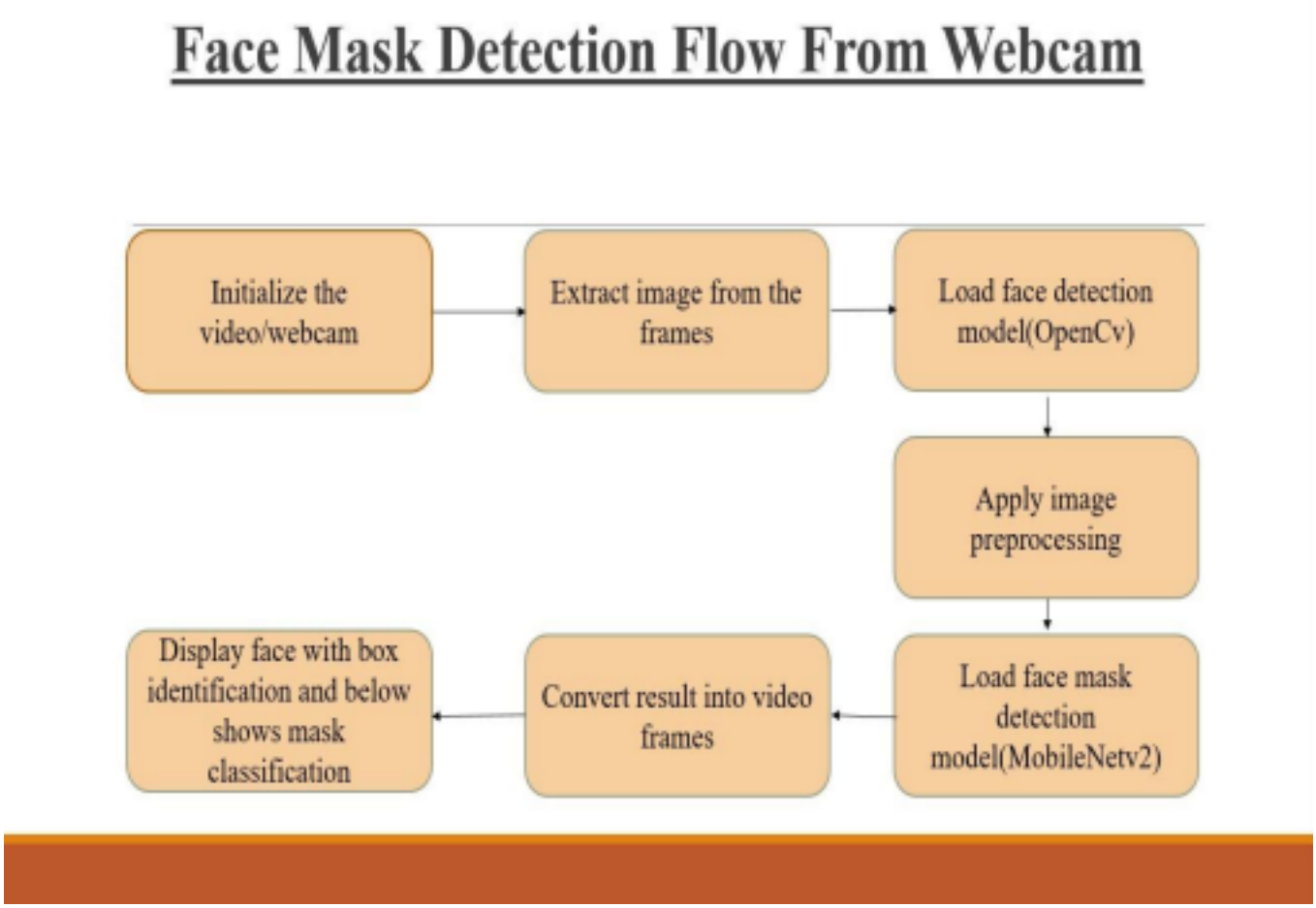
- **Usability:**
  The user experience should be very smooth in using any provided feature.
- **Reliability:**
  Face Mask Detection should be available and satisfy every user's needs.
- **Performance:**
  The application should work efficiently so that there might not be any problem in using this platform.
- **Availability:**
  Face Mask Detection should be always available for the users. The server uptime should be around 99% allowing for a small 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:

Class Diagram



Flow Diagram



4. Technology & Tools Used :

Front-end Tools

4.1.1.1. 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.

Back-end Tools

Tensor Flow:

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 it to implement interfaces for expressing machine learning algorithms. For example:- Handwritten digit recognition, face recognition. Some important API like: FAST API , CNTK (Microsoft Cognitive Toolkit)**.**

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 TensorFlow. All the layers used in the CNN model are implemented using Keras.

OpenCV:

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

Operating System

Windows 8,10 & Windows 11,MAC OS, etc.

Additional Software Requirement (Web Browser)

Support Web Browser- Firefox, Mozilla Google Chrome, Microsoft edge.

5. Hardware Requirement:

System Configuration (CPU, RAM, HDD)

- Ram- 8GB
- Storage- 512GB
- Graphics: Integrated graphics
- Processor- Any

6. Source Code:

Back End Code

Source Code Mask Detection:

```
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import
preprocess_input from tensorflow.keras.preprocessing.image import
img_to_array from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os

def detect_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
for i 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
if len(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)
if cv2.waitKey(1) & 0xFF ==27 :
        break
```

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

Source Code Model Training:

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

```
import numpy as np
import os

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

```
    data.append(image)
    labels.append(category)

# 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 # the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel) 33
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
```

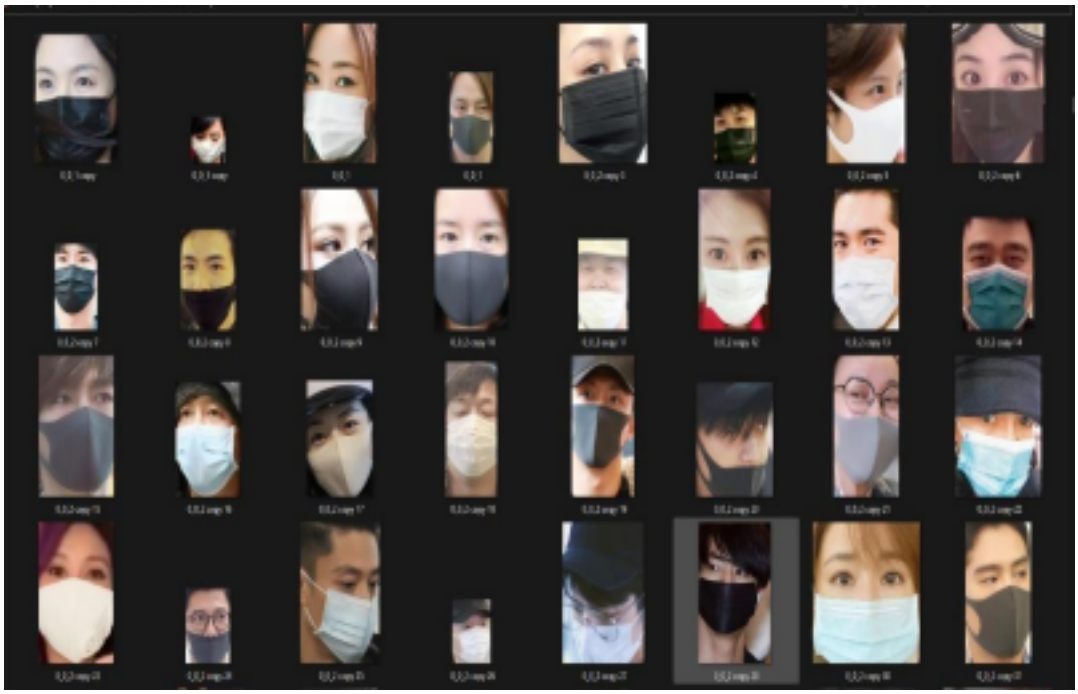
```
# compile our model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["accuracy"])

# 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:

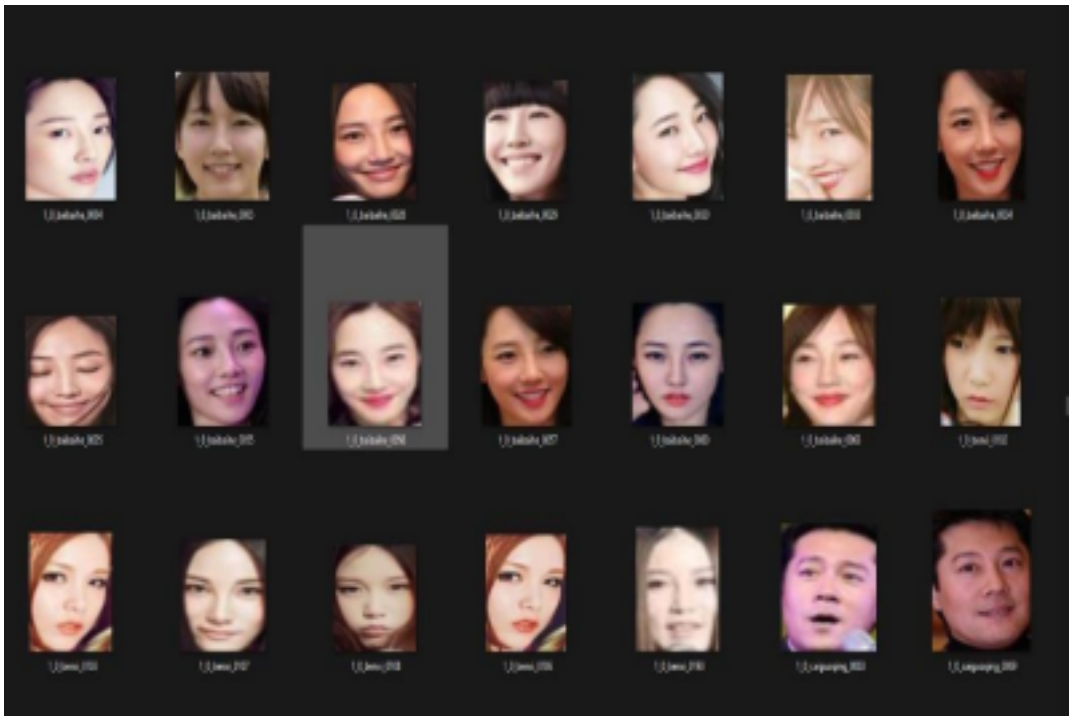
Data Collection:

Snapshots below illustrate an example of faces wearing and not wearing masks. The experiments of this research are conducted on one dataset taken from **Kaggle**. It consists 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

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

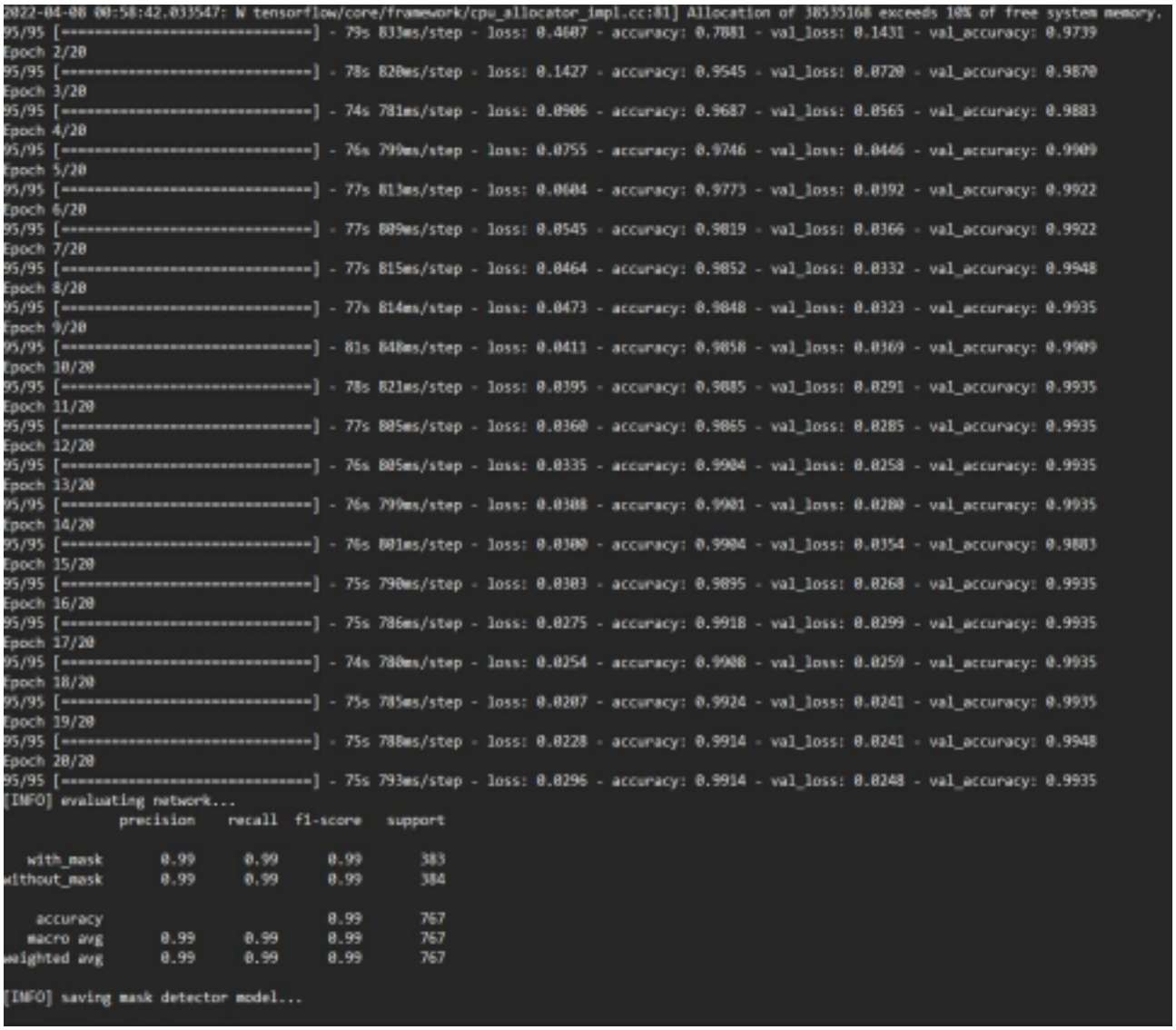


Snapshot 2 : Faces Without Masks

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

Model Evaluation:

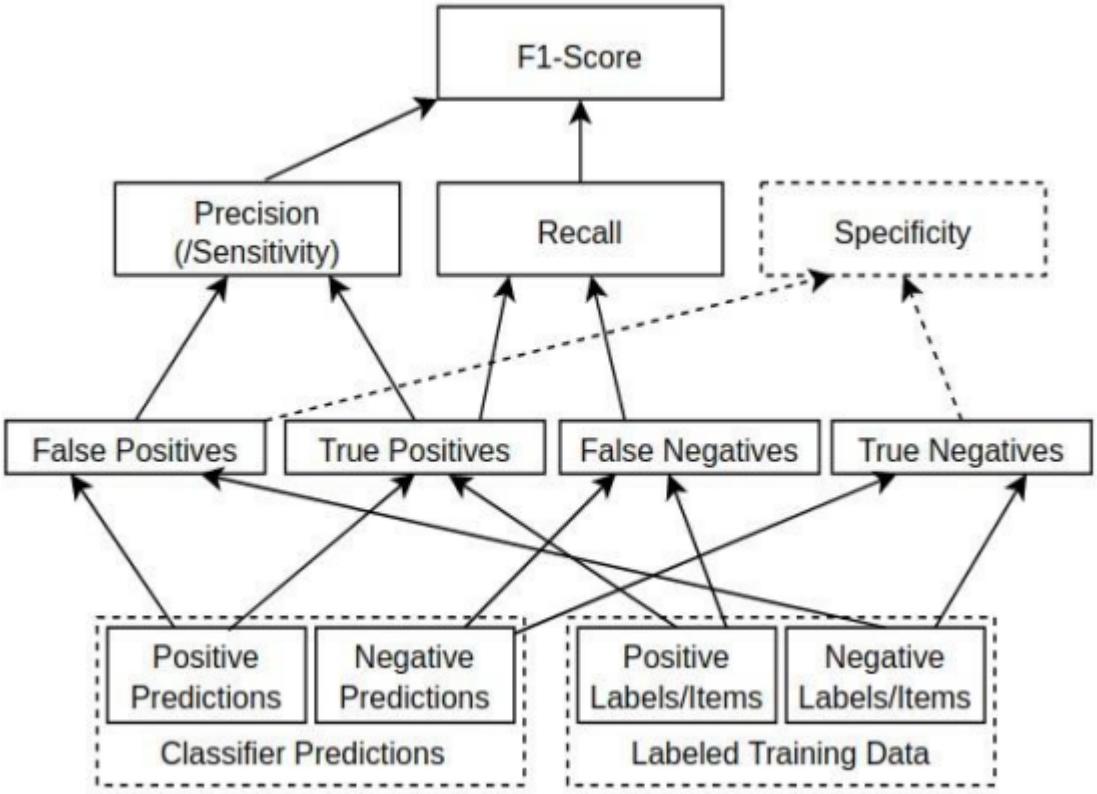
The Training model is evaluated using 20 Epochs each containing 95 data elements. During the Model Evaluation significant improvement on loss, accuracy, value loss and value accuracy can be observed using the methods such as accuracy, precision, recall, fi-score and support.



Snapshot 1

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.

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 a 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)

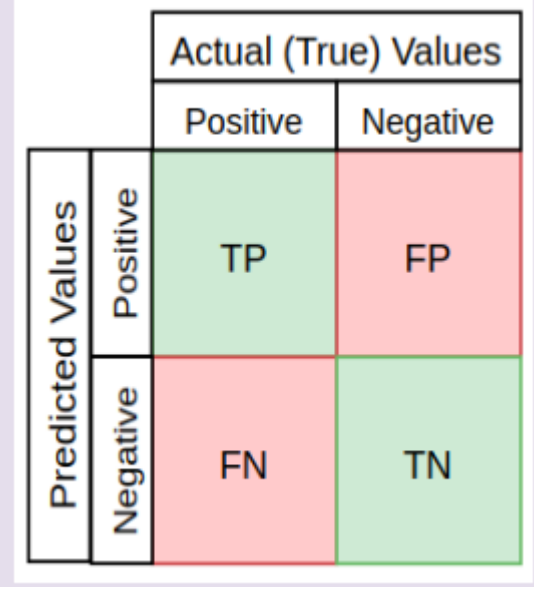
In the False Negative (FN) case, the classifier has predicted a Negative result, while the actual result was positive. Like no cat when there is a cat. So the prediction was Negative and wrong (False). Thus it is a False Negative.

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.

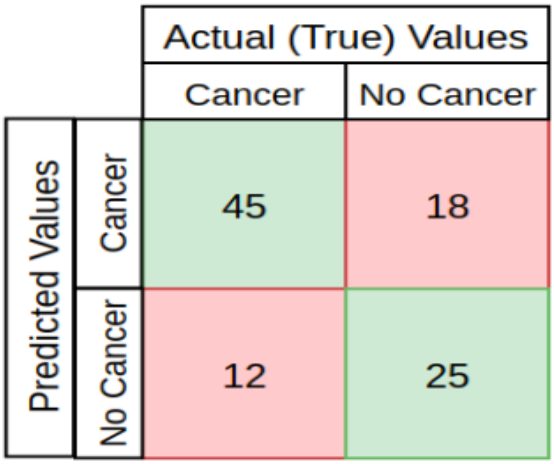
Confusion Matrix

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:



Confusion Matrix. Image by Author.

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



Confusion matrix for the cancer example. Image by Author.

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:

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

Accuracy Formula

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:

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

Precision formulas.

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:

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

Recall formulas.

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 generally described 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

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

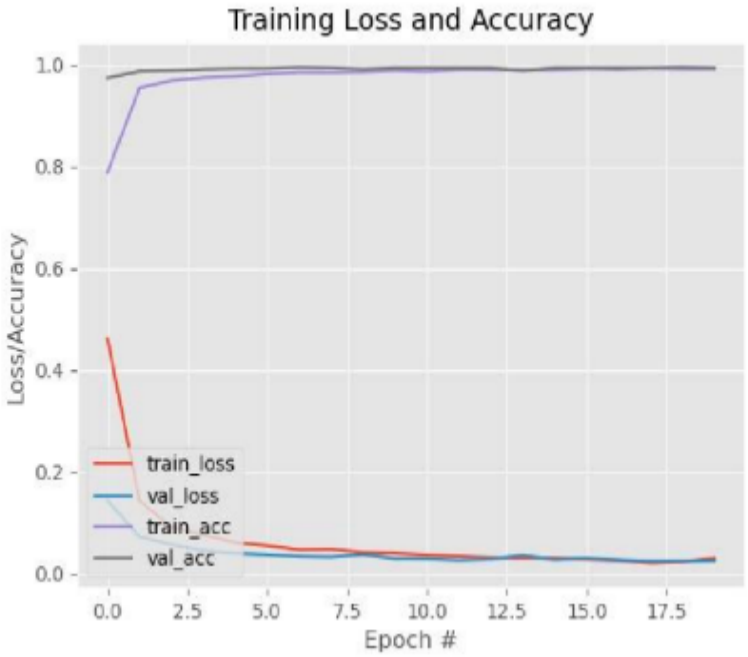
• F1-score score (formula above) of 2*(0.01*1.0)/(0.01+1.0)=~0.02.

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.

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.[45]

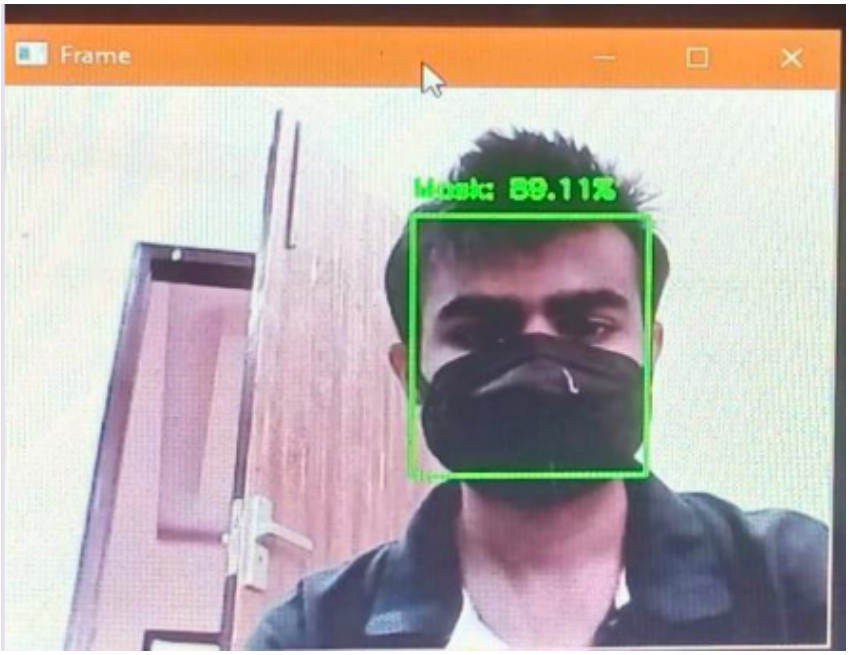
(a) Fig: Result after training once VS



(b) Fig: Improvement over training twice

Face Mask Detection on Live WebCam

Snapshot of actual working Mask Detector Frame window



8. Bibliography

[1] Xingi Fan, Mingjie Jiang “Retina Facemask: A Single Stage FaceMask Detector for Assisting Control of the Covid-19 Pandemic”

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

[3] Susanto Susanto,Febri Alwan Putra,Riska Analia, Ika Karlina Laila Nur Suciningtyas The Face Mask Detection For Preventing the Spread of COVID-19

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

[5] Amit Chavda, Jason Dsouza, Sumeet Badgujar 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