A Project Report

On

FACE MASK DETECTION USING PYTHON

Submitted in partial fulfilment of the requirements for the award of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

By

SREERAM MEGHANA (19BQ1A12F3)

SHAIK SUPHIYA NAWAZ BANU (19BQ1A12E9)

POOJITHA MANNE (19BQ1A12D2)

SINGAVARAPU RISHITHA (19BQ1A12F1)

Under the esteemed guidance of

Dr B.SAI JYOTHI

PROFESSOR

Department Of Information Technology

Vasireddy Venkatadri Institute of Technology

Jawaharlal Nehru Technological University, Kakinada, AP, India

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY: NAMBUR

BONAFIED CERTIFICATE

This is to certify that this project report is the Bonafide work of **SREERAM MEGHANA, SHAIK SUPHIYA AWAZ BANU, POOJITHA MANNE, SINGAVARAPU RISHTHA** Reg. No: **19BQ1A12F3,19BQ1A12E9,19BQ1A12D2,19BQ1A12F1** who carried out the project entitled **"FACE MASK DETECTION USING PYTHON"** under our supervision during the year 2020-2021.

PROJECT GUIDE
Dr B.SAI JYOTHI
Professor
Department of Information technology technology

HEAD OF DEPARTMENT
Dr. KALAVATHI ALLA
HOD Department of Information

External Viva voice conducted on

Internal Examiner **External Examiner**

VVIT
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY:

NAMBUR

CERTIFICATE OF AUTHENTICATION

I Solemnly declare that this project report "**FACE MASK DETECTION USNG PYTHON** "is the Bonafide work done by purely by us, carried out under the supervision of Ms. **Dr B. Sai Jyothi** towards partial fulfillment of the requirements of the requirements of Degree of **Bachelor of Technology** in Information Technology from Jawaharlal Nehru Technological University, Anantapur during the year 2020-2021.

It is further certified that this work has not been submitted, either in part of in full, to any department of the Jawaharlal Nehru Technological University, Institution or elsewhere, or for the publication in any form

Signature of the Student

DECLARATION

We **SREERAM MEGHANA, SHAIK SUPHIYA NAWAZ BANU, POOJITHA MANNE, SINGAVARAPU RISHITHA** hereby declare that the project report entitled "**FACE MASK DETECTION USING PYTHON**" done by us under the guidance of Dr B.SAI JYOTHI, M. TECH, Ph. D, PROFESSOR in **VASIREDDY VENKATADRI**
INSTITUTE OF TECHNOLOGY is submitted in fulfilment of their acquirements for the award of degree in **INFORMATION TECHNOLOGY**. The results embodied in this project have not been submitted to any other university or college for the award of any degree or diploma.

DATE:

PLACE: NAMBUR

SIGNATURE OF CANDIDATES:

(SREERAM MEGHANA-19BQ1A12F3)

(SHAIK SUPHIYA NAWAZ BANU-19BQ1A12E9)

(POOJITHA MANNE –19BQ1A12D2)

(SINGAVARAPU RISHITHA-19BQ1A12F1)

ACKNOWLEDGEMENT

First, we would like to express our sincere gratitude to our beloved Chairman, **Sri V. VIDYASAGAR**. We would be grateful to our beloved Principal, **Dr. Y. MALLIKARJUNA REDDY**, we would be grateful to our beloved Director, **Dr. NAVEEN RAVELA**. We would be grateful to our beloved Dean of Academics**, Dr. N. Kumara Swamy**. And people who made this project work easier with words of encouragement, motivation, discipline, and faith by offering different places to look forward in expanding our ideas and helped us towards the successful completion of this project work.

We take this opportunity to express our deepest gratitude and appreciation to our HOD**, Dr. A. KALAVATHI**, Professor, Guide, **Dr B.SAI JYOTHI, professor, Ph. D** and Project coordinators **Ms. M. Rajya Lakshmi, Assoc. Professor** and Ms. Sk. Mulla Almas, Asst Professor, Department of Information Technology, for their motivating suggestions, insightful advice, invaluable guidance, help and support in successful completion of this project and also for constant encouragement and advice throughout our B. Tech programme.

We would like to take this opportunity to express our thanks to the **teaching and non-teaching staff** of Department of Information Technology, VVIT for their valuable help and support in these four years of our study. We would like to thank Department of Information Technology, VVIT for providing all facilities to complete this project work. We are also grateful to all our classmates for their help, encouragement and invaluable suggestions. We would be grateful to our parents for their incessant support and cooperation.

Finally, we would like to thank all those whose direct and indirect support helped us in completing our project work in time.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

CNN - Convolutional Neural Network

IOU - Input Over Union

OpenCV - Open-Source Computer Vision Library

UML - Unified Modeling Language

ABSTRACT

The end of 2019 witnessed the outbreak of Corona virus Disease 2019 (COVID-19), which has continued to be the cause of plight for millions of lives and businesses even in 2020. As the world recovers from the pandemic and plans to return to a state of normalcy, there is a wave of anxiety among all individuals, especially those who intend to resume in person activity. Studies have proved that wearing a face mask significantly reduces the risk of viral transmission as well as provides a sense of protection. However, it is not feasible to manually track the implementation of this policy. Technology holds the key here. We introduce a Deep Learning based system that can detect instances where face masks are not used properly. Our system consists of a dual stage Convolutional Neural Network (CNN) architecture capable of detecting masked and unmasked faces and can be further integrated with pre-installed CCTV cameras. This will help track safety violations, promote the use of face masks, and ensure a safe working environment.

CHAPTER -1

INTRODUCTION

1.1 OVERVIEW:

The year 2020 has shown mankind some mind-boggling series of events amongst which the COVID19 pandemic is the most life-changing event which has startled the world since the year began. Affecting the health and lives of masses, COVID-19 has called for strict measures to be followed in order to prevent the spread of disease. From the very basic hygiene standards to the treatments in the hospitals, people are doing all they can for their own and the society's safety; face masks are one of the personal protective equipment. People wear face masks once they step out of their homes and authorities strictly ensure that people are wearing face masks while they are in groups and public places. To monitor that people are following this basic safety principle, a strategy should be developed. A face mask detector system can be implemented to check this. Face mask detection means to identify whether a person is wearing a mask or not. The first step to recognize the presence of a mask on the face is to detect the face, which makes the strategy divided into two parts: to detect faces and to detect masks on those faces. Face detection is one of the applications of object detection and can be used in many areas like security, biometrics, law enforcement and more. 12 There are many detector systems developed around the world and being implemented. However, all this science needs optimization; a better, more precise detector, because the world cannot afford any more increase in corona cases. In this project, we will be developing a face mask detector that is able to distinguish between faces with masks and faces with no masks. In this report, we have proposed a detector which employs SSD for face

detection and a neural network to detect presence of a face mask. The implementation of the algorithm is on images, videos and live video streams.

1.2 OBJECTIVE:

To develop a Face Mask Detector with OpenCV, TensorFlow, Keras and Deep Learning that helps to detect whether or not a person wears a mask.

1.3 MOTIVATION:

As the country starts going through various stages of reopening, face masks have become an important element of our daily lives and are here to stay. Wearing face masks (and wearing them correctly) will be required in order to socialize or conduct business. Shield yourself as well as other people from disease by washing your hands or utilizing a liquor-based rub regularly, not contacting your face and wearing a veil. The first three parts need to be 13 governed by ourselves but it can either urge people or motivate them to wear masks, the proposed project implementation has attempted to make people aware that face masks are essential for their own and other's safety.So we decided to create an application that utilizes a camera to detect if a person is wearing a mask and if the mask is being used correctly.

SCOPE:

The system can be used in the following places to identify people with or without masks:

• Offices – Manufacturers, retail, other SMEs and corporate giants

• Hospitals/healthcare organizations

• Airports and railway stations

• Sports venues

• Entertainment and hospitality industry

• Densely populated areas.

CHAPTER –2

REQUIREMENT ANALYSIS:

2.1 SOFTWARE REQUIREMENTS:

The libraries that we have used for developing this system are listed below.

• Tensor Flow and Keras:

Keras is a powerful and easy-to-use free open-source Python library for developing and evaluating deep learning. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in just a few lines of code.

• OpenCV:

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

• NumPy:

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. It is an open-source project and you can use it freely. NumPy stands for Numerical Python.

• Matplotlib:

It is probably the most used Python package for 2D-graphics. It provides both a quick way to visualize data from Python and publication-quality figures in many formats. We are going to explore matplotlib in interactive mode covering most common cases.

• SciPy library:

The SciPy library is one of the core packages that make up the SciPy stack. It provides many user-friendly and efficient numerical routines, such as routines for numerical integration, interpolation, optimization, linear algebra, and statistics. • imutils: A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much easier with OpenCV and both Python 2.7 and Python 3.

2.2HARDWARE REQUIREMENTS:

• Processor: Minimum 1 GHz; Recommended 2GHz or more.

• Hard Drive: Minimum 32 GB; Recommended 64 GB or more.

• Memory (RAM): Minimum 1 GB; Recommended 4 GB or above.

• Also require a camera.

2.3 FEASIBILITY ANALYSIS:

- **Technical Feasibility:** Technical feasibility assesses the current resources (hardware and software) and technologies, which are required to accomplish user requirements. It requires a computer with python anaconda installed. Today every organization has computer, so it is not an extra cost.
- **Economic Feasibility:** Economic feasibility is the most frequently used method for evaluating the effectiveness of proposed system. The proposed model is cost effective.
- **Operational feasibility:** The proposed system performs effective than the Existing system. The system recognizes a person without wearing a mask.

2.4 EXISTING SYSTEM:

The existing system is purely manual that if someone is found without face mask then he will be instructed wear it. There is no alternative for it. In order to check, whether people are using masks in a proper way, many measures are taken. In malls, theatres, markets, etc; few people are allotted only for this reason. Police do check on roads and public places. Volunteers are trying to bring awareness among people about the usage of masks. Though the government is investing lot of time, effort and man power, the outcome is not as expected. By implementing this system there will be two problems

1. No one will be there for controlling the passing of people without mask every moment and thus spreading of covid 19 will be easy.
2. Manual method is purely expensive.

3.It is hard to monitor the people, who are on moving vehicles etc.

2.5 PROPOSED SYSTEM:

As a part of 4.0 industry, every work of person must be replaced with a machine. In order to do this, we are proposing a new system which detects whether a person is using mask in a proper way or neglecting the pandemic and panicking others. The proposed system develops

classification and predictive model that can account for accurate classification grouping and prediction of Face masks on the face of a person. This system also has the ability to identify the persons who are not wearing the masks and mention the percentage of the mask. Hence the proposed system helps for the management of this pandemic situation in an effective way.

2.6ADVANTAGES OF PROPOSED SYSTEM:

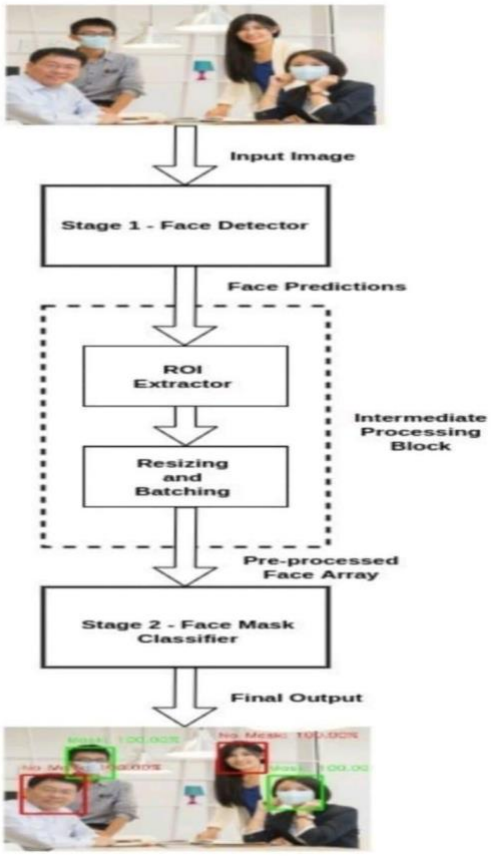
1. It detects multiple faces and face masks from all Angles in a small frame.
2. This model makes awareness without intervention of monitoring people (police), so lives of covid warriors are not at risk.
3. It uses the existing cameras to monitor the people so it is economically feasible.
4. The accuracy will be more and the time complexity will be less due to the MobileNet algorithm implementation.

CHAPTER-3

SYSTEM DESIGN

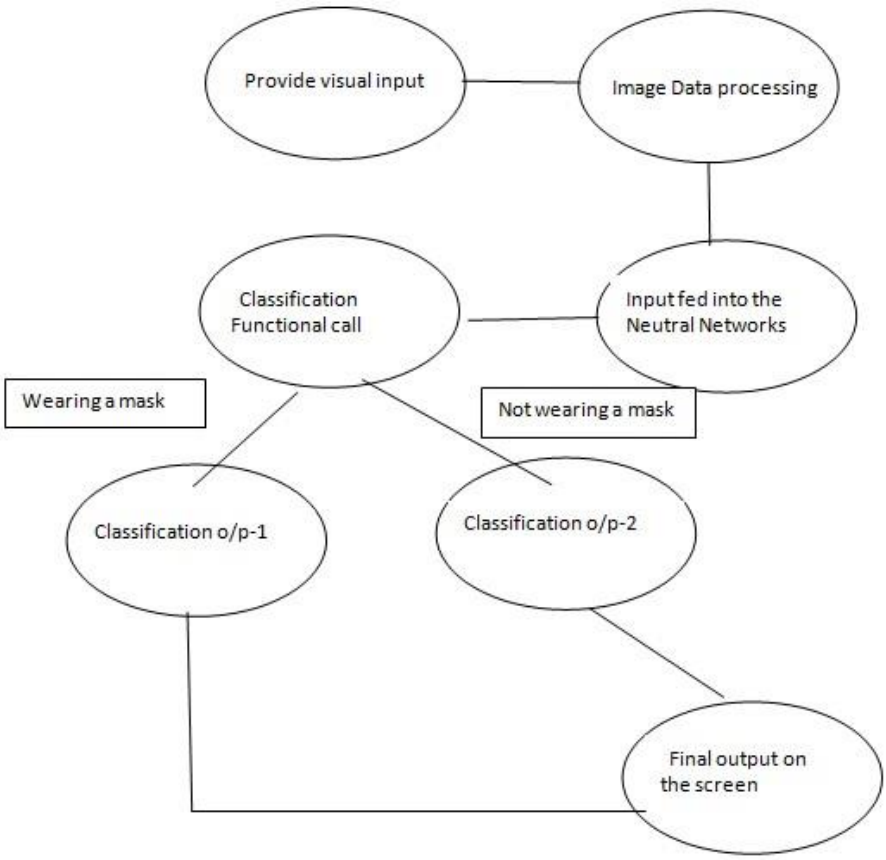
3.1 ARCHITECTURE OVERVIEW:

This system is a two staged architecture where the first stage detects human faces, while the second stage uses a lightweight image classifier to classify the faces detected in the first stage as either **'Mask'** or **'No Mask'** faces and draws bounding boxes around them along with the detected class name. This algorithm was further extended to videos as well. The detected faces are then tracked between frames using an object tracking algorithm, which makes the detections robust to the noise due to motion blur. This system can then be integrated with an image or video capturing device like a CCTV camera, to track safety violations, promote the use of face masks, and ensure a safe working.



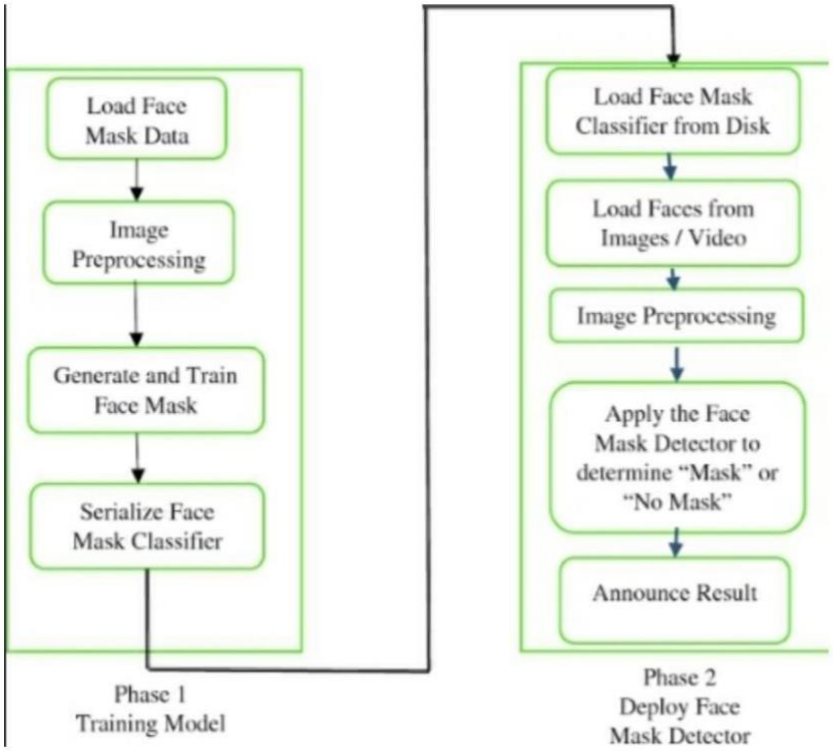
3.2USE CASE DIAGRAM:

Use case diagrams are a way to capture the system's functionality and requirements in UML diagrams. It captures the dynamic behavior of a live system. The below figure demonstrates the different ways that a user might interact with a face mask detector system



3.3DATA FLOW DIAGRAM

The below figure shows the two phases of face mask detection system. Phase-1 describes build and the train of the learning model and Phase-2 is deployment of the mask detector over images or live videos. Image processing steps such as resize the input image, color- filtering over the channels, scaling or normalization images, cropping the images and subdividing them into sub arrays are applied to all the raw input images to have augmented data, which could be fed to neutral network machine learning model. Then in the first phase, build a model and train the system with mask and without mask images with an appropriate algorithm. Once the model is trained, then in the second phase, load the face mask detector, perform face detection and then classify each face. Once on image has uploaded the classification happens automatically.



3.4DATASET CREATION:

The dataset used in this project is made with the help of a script in python which fetches free and open-source images from the internet from sites like Kaggle, Google images, etc. These images are put into a folder named without mask. An image of mask was then applied to these images in that dataset to make pictures of people with a mask and these pictures are stored in a folder named with mask. Our dataset structure is as follows:

• with mask [1915 entries]

• Without mask [1915 entries]

We need to split our dataset into three parts:

• training dataset

• test dataset

• Validation dataset.

The purpose of splitting data is to avoid over fitting which is paying attention to minor details, which is not necessary and only optimizes the training dataset accuracy. We need a model that performs well on a dataset that it has never seen (test data), which is called generalization.

• **TRAINING DATASET:**

The training set is the actual subset of the dataset that we use to train the model. The model observes and learns from 23 this data and then optimizes its parameters.

• **VALIDATION DATASET:**

The validation dataset is used to select hyperparameters (learning rate, regularization parameters). When the model is performing well enough on our validation dataset, we can stop learning using a training dataset.

• **TEST DATASET:**

The test set is the remaining subset of data used to provide an unbiased evaluation of a final model fit on the training dataset. Data is split as per a split ratio which is highly dependent on the type of model we are building and the dataset itself. If our dataset and model are such that a lot of training is required, then we use a larger chunk of the data just for training which is our case.

If the model has a lot of hyperparameters that can be tuned, then we need to take a higher amount of validation dataset. Models with a smaller number of hyperparameters are easy to tune and update, and so we can take a smaller validation dataset. In our approach, we have dedicated 80% of the dataset as the training data and the remaining 20% as the testing data, which makes the split ratio as 0.8:0.2 of train to test set. Out of the training data, we have used 20% as a

validation data set. Overall, 64% of the dataset is used for training, 16% for24 Validation and 20% for testing.

3.5 PREPROCESSING:

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. Preprocessing on images is done to convert image objects into RGB arrays. Then the array is resized to (224, 224, 3). Preprocessing in the caption is performed to make sentences that were previously in the form of the word into a sequence of tokens based on a unique word index in the dictionary. At the training phase, the model has two inputs. The first input is an image feeding into the pre-trained MobileNet-v3 model with the removed output layer and will outputting extracted images features. The second input is a description that has been done by preprocessing so that it becomes a sequence index of tokens.

Preprocessing steps as mentioned below was applied to all the raw input images to convert them into clean versions, which could be fed to a neural network Machine Learning model.

- Resizing the input image (256x 256)
- Applying the color filtering (RGB) over the channels (Our model MobileNetV2 supports 2D 3 channel image)
- Scaling / Normalizing images.
- Center cropping the image with the pixel value of 224x224x3
- Finally Converting them into tensors (Similar to NumPy array).

3.6 TRAINING

WITH MASK

WITHOUT MASK

Training the model include 3 steps:

1) augmenting the data

2) loading the MobileNetV2 classifier for tuning this mode finely, ImageNet weights are used to build a completely new FC head.

3) saving the trained detector model to the disk.

Before Training 2 steps are done...

• INIT_LR is used here to initialize the learning rate. 0.0001 is the initial learning rate, the lower the INIT_LR, the better the results.

• 18 EPOCHS are used so that the accuracy for training is high.

• EPOCHS are the number of passes; the entire training algorithm has completed which will trigger the end of training.

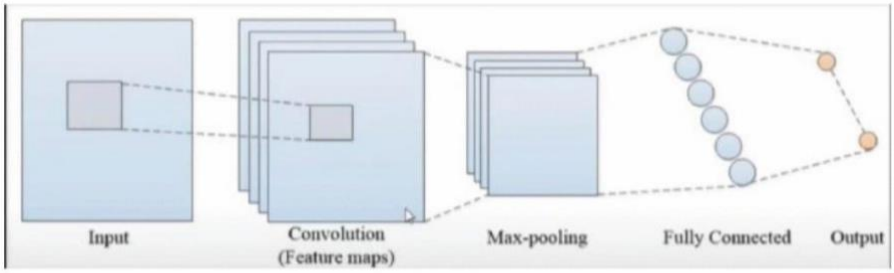
• BS is batch size and it means that the training algorithm will use 32 images at once for training.

The training Procedure involves many steps that are taken from the documentation for the classifier model.

- Sci-Kit-learn (sklearn) is used for binarizing class marks, dividing our dataset and printing a characterization report.

- Imutils will assist us with finding the rundown of pictures in our dataset.

This image shows how data is processed inside the convolutional neural network.



TRANING OVERVIEW:

- We also compare the default bounding boxes having different sizes and aspect ratios with ground truth boxes and finally use Intersection over Union (IOU) method to select the best matching box for each pixel.

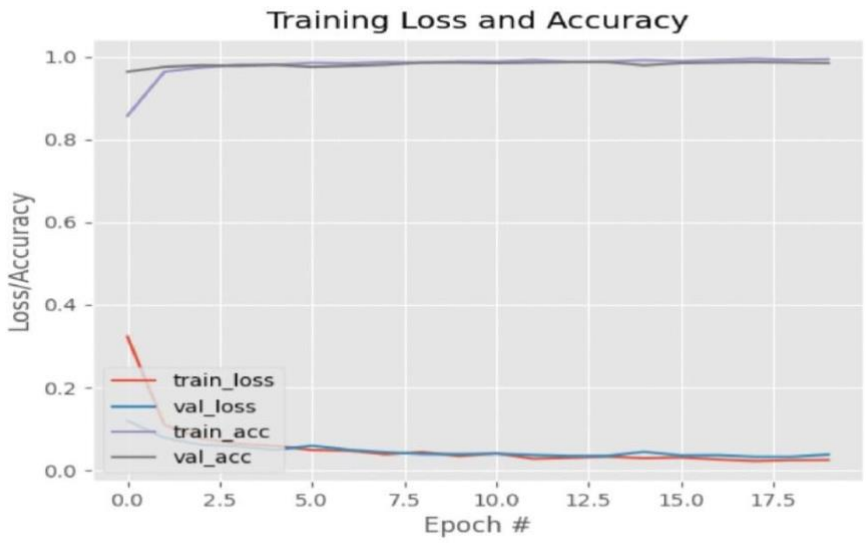
- IOU evaluates how much part of our predicted box match with the ground reality. The values range from 0 to 1 and increasing values of IOU determine the accuracies in the prediction. The best value being the highest value of IOU. The equation and pictorial descriptionof IOU is given as follow:

$$\mathbf{IOU(B1, B2) = B1 \cap B2 / B1 \cup B2}$$

RESULT AFTER TRAINING:

The Graph is plotted to analyze the training data in the dataset and it is plotted using the matplotlib library of python. A graph is designed to show data and value, loss and accuracy as

the epochs progress, here the epochs mean the number of passes that the training algorithm has completed. This graph was plotted using matplotlib. RESULT OF GRAPH The plot shows that the software plotted which shows the accuracy of the training algorithm. This curve shows that as EPOCHS passed the training accuracy has been constantly above 98% and the data loss has been much high in the initial EPOCHS but has been minimized to 5% and under



3.7 FACE MASK DETECTION:

The face mask detection involves the two stages the first stage is face recognition and second is the applying the face mask classifier.

• FACE DETECTION:

Like object detection, face detection adopts the same architectures as one-stage and two-stage detectors, but in order to improve face detection accuracy, more face-like features are being added. We perform the face detection for each frame in a video. So when it comes to detecting a face in still image and detecting a face in a real-time video stream, there is not much difference between them. Therefore, we will be using Haar Cascade algorithm, also known as Voila-Jones algorithm to detect faces. It is basically a machine learning object detection algorithm

which is used to identify objects in an image or video. This whole face detection is done in two steps The first step involves expanding the bounding boxes in height and width by 20%, which covers the required Region of Interest (ROI) with minimal overlap with other faces in most situations. The second step involves cropping out the expanded bounding boxes from the image to extract the ROI foreach detected face. The extracted faces are resized and normalized as required by Stage2. Furthermore, all the faces are batched together for batch inference.

• **FACE MASK CLASSIFIER:**

The second stage of our system is a facemask classifier. This stage takes the processed ROI from the Intermediate Processing Block and classifies it as either Mask or No Mask. A CNN based classifier for this stage was trained, based on the image classification model MobilenetV2.These models have a light weight architecture that offers high performance with low latency, which is suitable for video analysis. The output of this stage is an image (or video frame) with Localized faces, classified as masked or unmasked.

CHAPTER 4

SYSTEM

IMPLEMENTATION

4.1 TRAIN MASK DETECTOR:

```
# import the necessary packages

# 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:\Mask Detection\CODE\Face-Mask-Detection-master\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
```

```
# 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")
```

4.2 DETECT MASK VIDEO:

```
# import the necessary packages


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"face_detector\deploy.prototxt" weightsPath =
r"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("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)          key
= cv2.waitKey(1) & 0xFF
```

```
        # if the `q` key was pressed, break from the loop
if key == ord("q"):
                break


# do a bit of cleanup
cv2.destroyAllWindows() vs.stop()
```

4.3 OUTPUT SCREENS

We implemented our model on images containing one and more faces. We also implemented it on videos and live video streams by removing and wearing masks one by one. Some screenshots of the results are shown below:

Output...

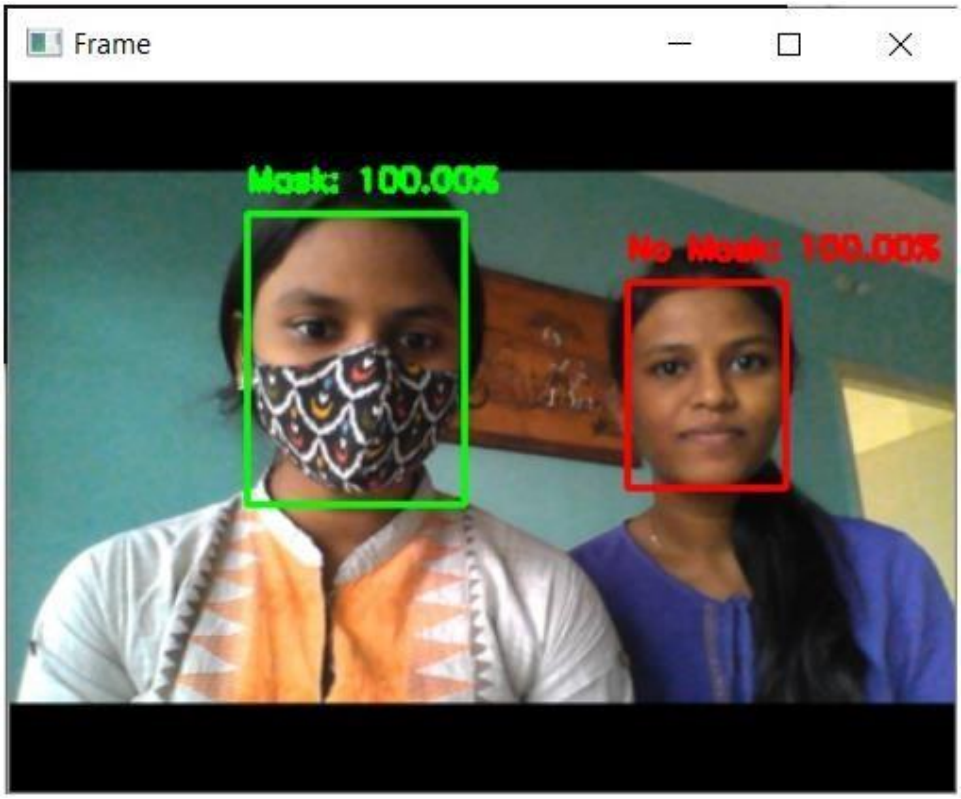


Fig 1 One person with mask and other person without mask

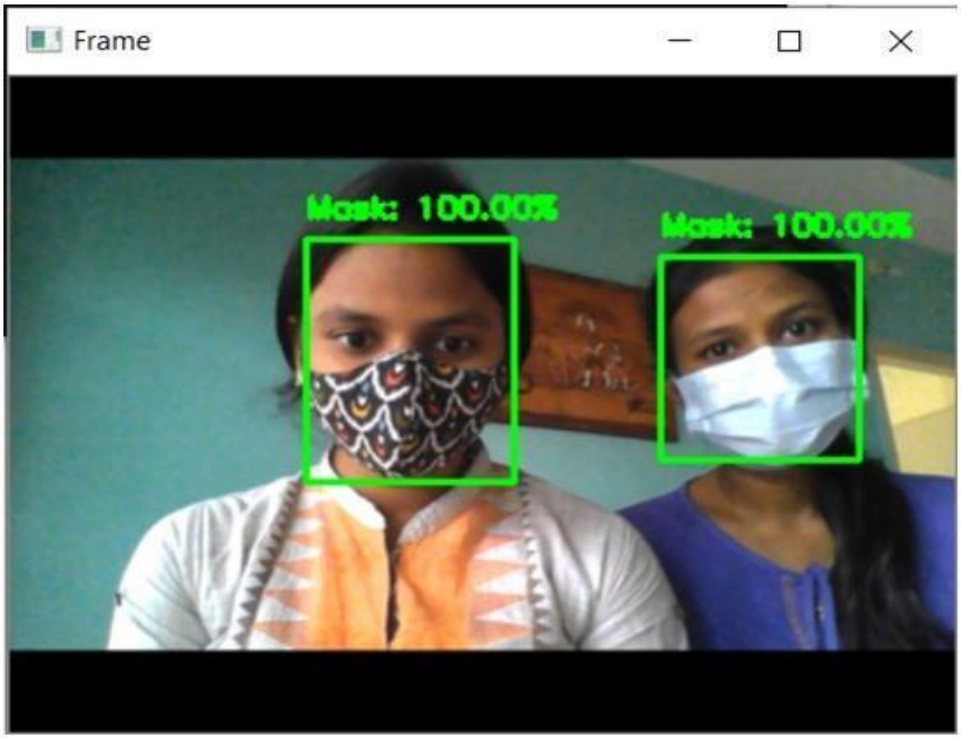


Fig 2 Two persons with mask

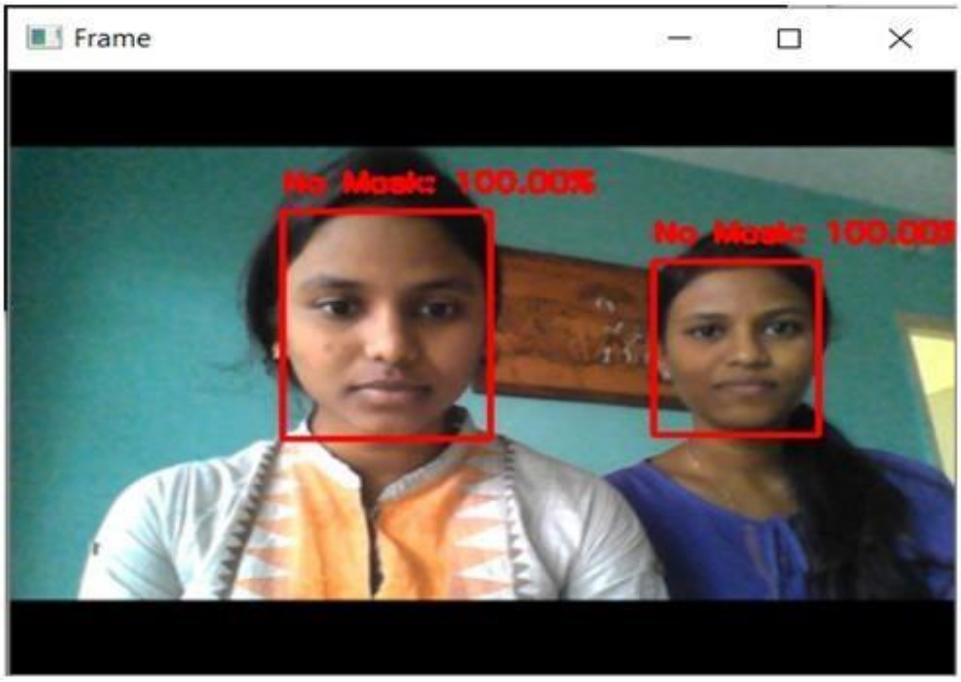


Fig 3 Two persons without mask

CHAPTER-5

TESTING

5.1 TEST PLANS:

A test plan documents strategy that will be used to verify and ensure that a product or system meets its design specification and other requirements. A test plan is usually prepared by or with significant input from the engineer. This document describes the plans for testing the architectural prototype of System. In this Project the Trained Model has to be tested to get the Desired Output. I use Different Classes of images for testing the system.

5.2 UNIT TESTING:

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

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

5.3 INTEGRATION TESTING: It takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

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

CONCLUSION AND FUTURE ENHANCEMENT

To mitigate the spread of COVID-19 pandemic, measures must be taken. We have modeled a face mask detector using SSD architecture and transfer learning methods in neural networks. To train, validate and test the model, we used the dataset that consisted of 1916 masked faces images and 1919 unmasked faces images. These images were taken from various resources like Kaggle and RMFD datasets. The model was inferred on images and live video streams. To select a base model, we evaluated the metrics like accuracy, precision and recall and selected MobileNetV2 architecture with the best performance having 100% precision and 99% recall. It is also computationally efficient using MobileNetV2 which makes it easier to install the model to embedded systems. This 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. By the development of face mask detection, we can detect if the person is wearing a face mask and allow their entry would be of great help to the society.

More than fifty countries around the world have recently initiated wearing face masks compulsory. People have to cover their faces in public, supermarkets, public transports, offices, and stores. Retail companies often use software to count the number of people entering their stores. They may also like to measure impressions on digital displays and promotional screens. We are planning to improve our Face Mask Detection tool and release it as an open-source project. Our software can be equated to any existing USB, IP cameras, and CCTV cameras to detect people without a mask. This detection live video feed can be implemented in web and desktop applications so that the operator can see notice messages. Software operators can also get an image in case someone is not wearing a mask. Furthermore, an alarm system can also be implemented to sound a beep when

someone without a mask enters the area. This software can also be connected to the entrance gates and only people wearing face masks can come in.

REFERENCE:

The Reference we took for Face Mask Detection project are given below:

1. et al. Coronavirus disease 2019 (covid-19): situation report 96 2020.
2. D. Chiang Detect faces and determine whether people are wearing mask 2020 [online] Available: https://github.com/AIZOOTech/Facemask Detection.
3. https://ieeexplore.ieee.org/document/9297399
4. /2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deeplearning/
5. Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms Book by Nicholas Locascio and Nikhil Buduma

• www.github.com

• www.stackoverflow.com

• www.codeacademy.com