A Mini Project Report

on

DETECTION OF FACE MASK

Submitted for the partial fulfillment of the Semester-VI BE degree as per the requirements of Osmania University, Hyderabad
Degree in **Information Technology**

Under the Guidance of

Mr. Sandeep, Assistant Professor, IT Department

By

M. Shravani	160618737034
J. Sai Sruthi	160618737021
N. Alekya Reddy	160618737038

Department of information technology & Engineering
Stanley College of Engineering and Technology for Women
(Autonomous & Approved by AICTE)

Chapel Road, Abids, Hyderabad - 500001

INSTITUTE VISION

•Empowering girl students through professional education integrated with values and character to make an impact in the world.

INSTITUTE MISSION

•Enabling quality engineering education for girl students to make them competent and confident to succeed in professional practice and advanced learning.

•Providing state-of-art-facilities and resources towards world class education. Integrating qualities like humanity, social values, ethics, and leadership towards their contribution to society.

DEPARTMENT VISION

•Empowering girl students with the contemporary knowledge in Information Technology for their success in life.

DEPARTMENT MISSION

•Quality education and excellent environments for students to learn and practice various hardware, software and firmware platforms prevalent in industry, integrated with opportunities for teamwork, leadership, values, ethics and social activities.

•Providing state-of-art-facilities and resources towards world class education. Integrating qualities like humanity, social values, ethics, and leadership towards their contribution to society.

PROGRAMOUTCOMES

• **Engineering knowledge:** Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the conceptualization of engineering models

• **Problem Analysis**: Identify, formulate, research literature and solve complex engineering problems reaching substantiated conclusions using first principles of mathematics and engineering sciences.

• **Design/development of solutions:** Design solutions for complex engineering problems and design systems, components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations.

• **Modern Tool Usage**: Create, select and apply appropriate techniques, resources, and modern engineering tools, including prediction and modeling, to complex engineering activities, with an understanding of the limitations.

• **The engineer and society:** Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.

• **Environment & sustainability**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

• **Ethics**: Demonstrate understanding of the societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to engineering practice.

• **Conduct investigations of complex problems**: Conduct investigations of complex problems including design of experiments, analysis and interpretation of data, and synthesis

• Individual and Team work: Understand and commit to professional ethics and responsibilities and norms of engineering practice.

• **Communication**: Understand the impact of engineering solutions in a societal context and demonstrate knowledge of and need for sustainable development.

• **Project Management and Finance**: Demonstrate a knowledge and understanding of management and business practices, such as risk and change management, and understand their limitations.

• **Lifelong Learning**: Recognize the need for, and have the ability to engage in independent and life-long learning

PROGRAM SPECIFIC OUTCOME

PSO1: Skilled Professional: Ability to apply technical skills and involve in the creation, maintenance and use of Computer, Computer Networks and Computer Information Systems.

PSO2: Research Capability: Ability to pursue research with academic excellence and core competence skills.

Stanley College of Engineering & Technology for Women Chapel Road, Hyderabad

(Autonomous & Approved by AICTE)

Ref No: SCETW/IT Dept/III Year 2021/Mini Project Date: 20-10-2021

CERTIFICATE

This is to certify that the mini project titled "Detection Of Face Mask" is a bonafide work carried over by Ms. M.Shravani (H.T. No 160618737034), Ms. J. Sai Sruthi (H.T. No 160618737021) and Ms. N. Alekya Reddy (H.T. No. 160618737038) in partial fulfillment of the requirements for the award of the degree Bachelor of Engineering in Information Technology from Osmania University during the III/IV Semester-II of their B.E. course during the academic year 2020-2021.

Internal Examiner **HOD** **External Examiner**

ACKNOWLEDGEMENT

I take this opportunity to convey my heartfelt gratitude to each and every one who has supported us in every way or the other during the course of our project.

From the very core of my heart, I would like to express our sincere gratitude to our internal guide Mr. Sandeep Assistant Professor, IT Department for his supervisory guidance. I express my profound gratitude to project coordinator Dr. K. Ramakrishna and Mr. Srinivas, HOD of Information Technology department. We are always grateful to our peers and friends who have always encouraged us and guided us whenever we needed assistance.

TABLE OF CONTENTS

ABSTRACT

COVID-19 pandemic has rapidly affected our day-to-day life disrupting the world trade and movements. Wearing a protective face mask has become a new normal. In the near future, many public service providers will ask the customers to wear masks correctly to avail of their services. Nowadays we are facing a pandemic, there is a situation where people are not ready to wear face masks, or they do not wear them properly, so, in this research, we are introducing an automatic mask detection system using image processing and soft computing techniques to tackle this problem. In the midst of the pandemic, covering our faces with a mask has become a new normal, as face masks are active in preventing the spread of the virus. Other precautionary measures are also advocated by the government apart from covering faces, to ensure protection and hygiene. In addition, because of the limited supply of masks in the industry, millions of people are learning to make their face masks. On the contrary, identifying faces with masks on any surveillance devices would be demanding while ensuring less access control in buildings. Face coverage with masks is a problem for algorithms and success in face detection. Currently, the authorities have to manually ask people to wear masks even then they tend to fool the authorities, to avoid that we are proposing a face Machine learning-based model of recognition. In the field of computer vision, this is a common research direction by extracting features directly from the detection region and then using machine detection learning algorithms to identify and recognize. In this Face mask detection-based attendance System, people will be only able to mark their attendance only if they wear a mask, besides if they do not wear a mask, they are given an alert and they would have to wear a mask. Therefore, face mask detection has become a crucial task to help global society. This paper presents a simplified approach to achieve this purpose using some Python and Deep Learning CNN. The proposed method detects the face from the image correctly and then identifies if it has a mask on it or not. As a surveillance task performer, it can also detect a face along with a mask in motion. The method attains accuracy up to 95.77% and 94.58% respectively on two different datasets. We explore optimized values of parameters using the Sequential Convolutional Neural Network model to detect the presence of masks correctly without causing over-fitting.

Guide Signature

1.INTRODUCTION

According to the World Health Organization (WHO)’s official Situation Report on coronavirus disease 2021 (COVID-19) has globally infected over 177 Million people causing over 3.8 Million deaths.

Individuals with COVID-19 have had a wide scope of symptoms reported – going from mellow manifestations to serious illness. Respiratory problems like shortness of breath or difficulty in breathing is one of them. Elder people having lung disease can possess serious complications from COVID-19 illness as they appear to be at higher risk. Some common human coronaviruses that infect public around the world are 229E, HKU1, OC43, and NL63. Before debilitating individuals, viruses like 2019-nCoV, SARS-CoV, and MERS-CoV infect animals and evolve to human coronaviruses. Persons having respiratory problems can expose anyone (who is in close contact with them) to infective beads. Surroundings of a tainted individual can cause contact transmission as droplets carrying virus may withal arrive on his adjacent surfaces.

To curb certain respiratory viral ailments, including COVID-19, wearing a clinical mask is very necessary. The public should be aware of whether to put on the mask for source control or aversion of COVID-19. Potential points of interest of the utilization of masks lie in reducing vulnerability of risk from a noxious individual during the "pre-symptomatic" period and stigmatization of discrete persons putting on masks to restraint the spread of virus. WHO stresses on prioritizing medical masks and respirators for health care assistants. Therefore, face mask detection has become a crucial task in present global society.

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 stream.

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.

2. LITERATURE REVIEW

2.1 LITERATURE SURVEY

MAMATA S. KALAS, REAL TIME FACE DETECTION AND TRACKING USING OPENCV,

International Journal of Soft Computing and Artificial Intelligence, ISSN: 2321-404X, Volume-2, Issue-1, May- 2014

2.2RELATED WORK

Face detection is defined as the procedure that has many applications like face tracking, pose estimation or compression. Face detection is a two-class problem where we have to decide if there is a face or not in a picture. This approach can be seen as a simplified face recognition problem.

AdaBoost: Adaboost is an algorithm for constructing a strong classifier as a linear combination. Adaboost, short for Adaptive Boosting, is a machine learning algorithm. It is a meta-algorithm and can be used in conjunction with many other learning algorithms to improve their performance. Adaboost is adaptive in the sense that subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifiers. Adaboost generates and calls a new weak classifier in each of a series of rounds. For from a set of training images. This method can be used for both face detection and face locations. In this method, a standard face (such as frontal) can be used. The advantages of this method are that it is very simple to implement the algorithm, and it is easily to determine the face locations such as nose, eyes, mouth, etc. based on the correlation values.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Face Mask Detection is the process of detecting faces, from an image or video to find whether a person is wearing a mask or not. Face mask detection feature uses visible stream from the camera combined with machine learning techniques to detect, generate and alert for people not wearing face mask. A user-friendly interface allows monitoring and review of alerts generated by system.

Detection of face masks is an extremely challenging task for the present proposed models of face detectors. This is because faces with masks have varied accommodations, various degrees of obstructions, and diversified mask types. They are used to facilitate self-focusing. Even after having such extraordinary and exceptional results in the existing face detectors, there is still high rising scrutiny in the development of more advanced face detectors as for existing models, event analysis and video surveillance is always a challenging job. Several reasons were found for the poor achievement of existing face mask detection model as compared to the normal ones, two of them were First due to lack of suitable datasets with properly masked faces and facial recognition. Secondly, the presence of masks on the face brings a certain kind of noise, which further deteriorates the detection process. There is an excellent challenge for a vast dataset so that an efficient face mask detection model can be easily developed.

3.2 PROPOSED SYSTEM

In our proposed system we will use a live video stream or an image and finally in output, it alerts a red color square box when a person is not wearing a mask. To predict whether a person has worn a mask correctly, the initial stage would be to train the model using a proper dataset. Within the dataset consist of After training the classifier, an accurate face detection model is required to detect faces, so that the SSDMNV2 model can classify whether the person is wearing a mask or not. The task in this paper is to raise the accuracy of mask detection without being too resource-heavy.

3.3 MODULES

3.3.1. DATA COLLECTION

Collect the dataset, Two folders with mask and without mask. With mask is the photos of people who are wearing the mask. Without mask is the collection of photos of people who are not wearing the mask.

3.3.1. TRAINING THE DATASET

We need to load the dataset in the disk and train the dataset using tensor flow and keras libraries in the anaconda prompt. We use the CNN (convolutional neural network) to train the data set. Lastly serializing the face mask classifier to disk.
Accuracy in image recognition problems. This helps us to get the results accurate and differentiate between mask and no mask.
CNN automatically detects the important features without any human supervision.
If there is no good GPU they are quite slow to train (for complex tasks). They use to need a lot of training data.
It is computationally very expensive and time consuming to train with traditional CPUs.
It is computationally very expensive and time consuming to train with traditional CPUs.

3.3.2. DEPLOYMENT

To detect the faces in a live video stream an extract the ROI (region of interest) for each face using the OpenCV library.
We can apply a face mask classifier of each region of interest to determine whether a person has worn a mask or not and at last display the result.
In result, if the ROI turns into a red color box then the person is not wearing a mask and if ROI turns into a green color box that means the person is wearing a mask.

3.3.3. TESTING

In the first path that is in the training face we will write a python script using tensorflow and keras into the face mask detection model.
In the second part that is in the deployment phase, we test the results in a real time webcam using OpenCV and mobileNetV2.
Send alerts to the faces which are recognized, also set the rate of sending the alerts and detection of faces.
Attach multiple cameras in a few minutes and enable all the cameras to access the AI capability of recognizing faces.
The system can work on any existing web camera without the installation of any new cameras. Most of the hospitals and airports have IP cameras installed and webcam-enabled.

4.SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

Processor	**:** AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx
Motherboard	**:** RYZEN Chipset Motherboard
Ram	**:** 8.00 GB

Cache	**:** 512 KB
Hard Disk	**:** 20.00 GB
Disk Drive	**:** 1.44 MB Floppy Disk recommended
Monitor	**:** 1024 x 720 Display
Speed	**:**2.7 GHZ

4.2 SOFTWARE REQUIREMENTS

Operating system	**:** Windows 10
Back End	**:** Python 3.7
IDE	**:** Anaconda Prompt
Tools And Libraries	**:** OpenCV, TensorFlow, Keras, MobileNet V2.

5. SYSTEM DESIGN

5.1 ALGORITHM

We have two main phases to complete this project. Which are mentioned below.

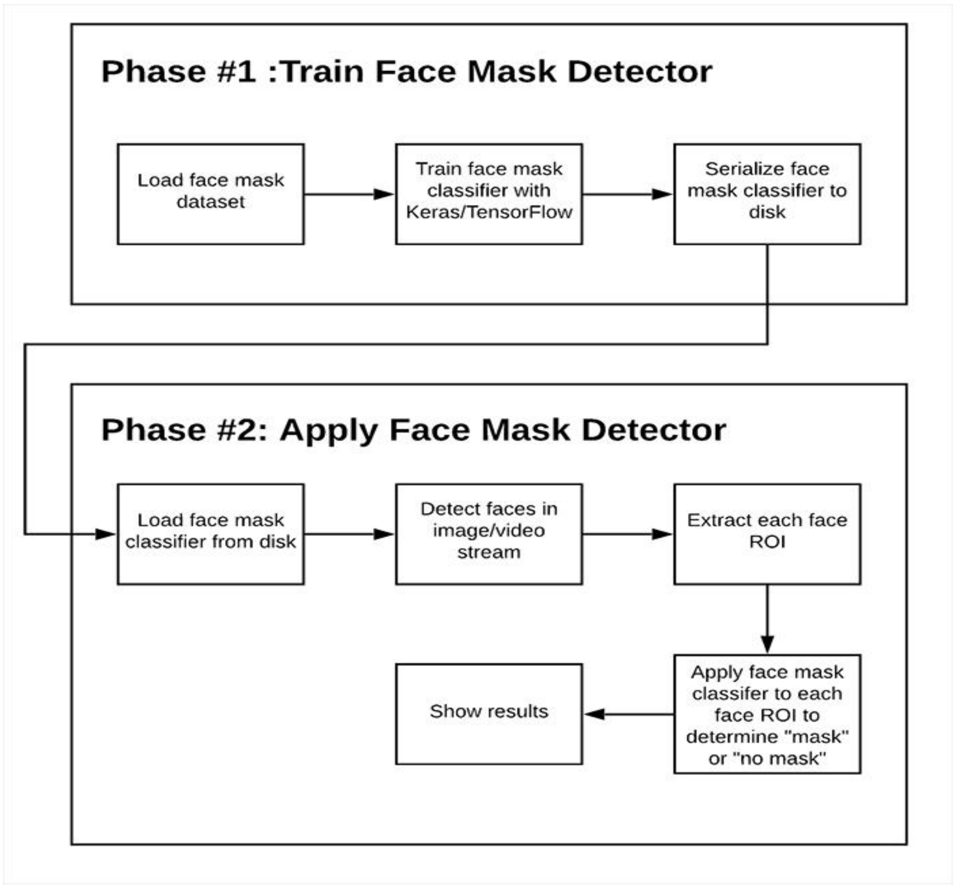
PHASE1: Load the dataset and train the dataset.

PHASE2: Apply the trained dataset to detects the rate of interest.

PSEUDO CODE

```
STEP-1: We need to load face mask dataset having images of with and without mask.
STEP-2: Train the dataset using the Keras/TensorFlow.
STEP-3: Then serialize the face mask classifier to the disk.
STEP-4: To apply face mask detector, load the face mask classifier from the disk.
STEP-5: It detects the faces in image or video stream.
STEP-6: Then extract each face from ROI ( Region Of Interest ).
STEP-7: Apply face mask classifier to each face ROI to determine “Mask” or “No Mask”.
STEP-8:  Finally it shows the result.
```

5.2 FLOW CHART



Phase #1 :Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifer to each face ROI to determine "mask" or "no mask"
Show results

Fig.5.2.1

5.3 ER DIAGRAM

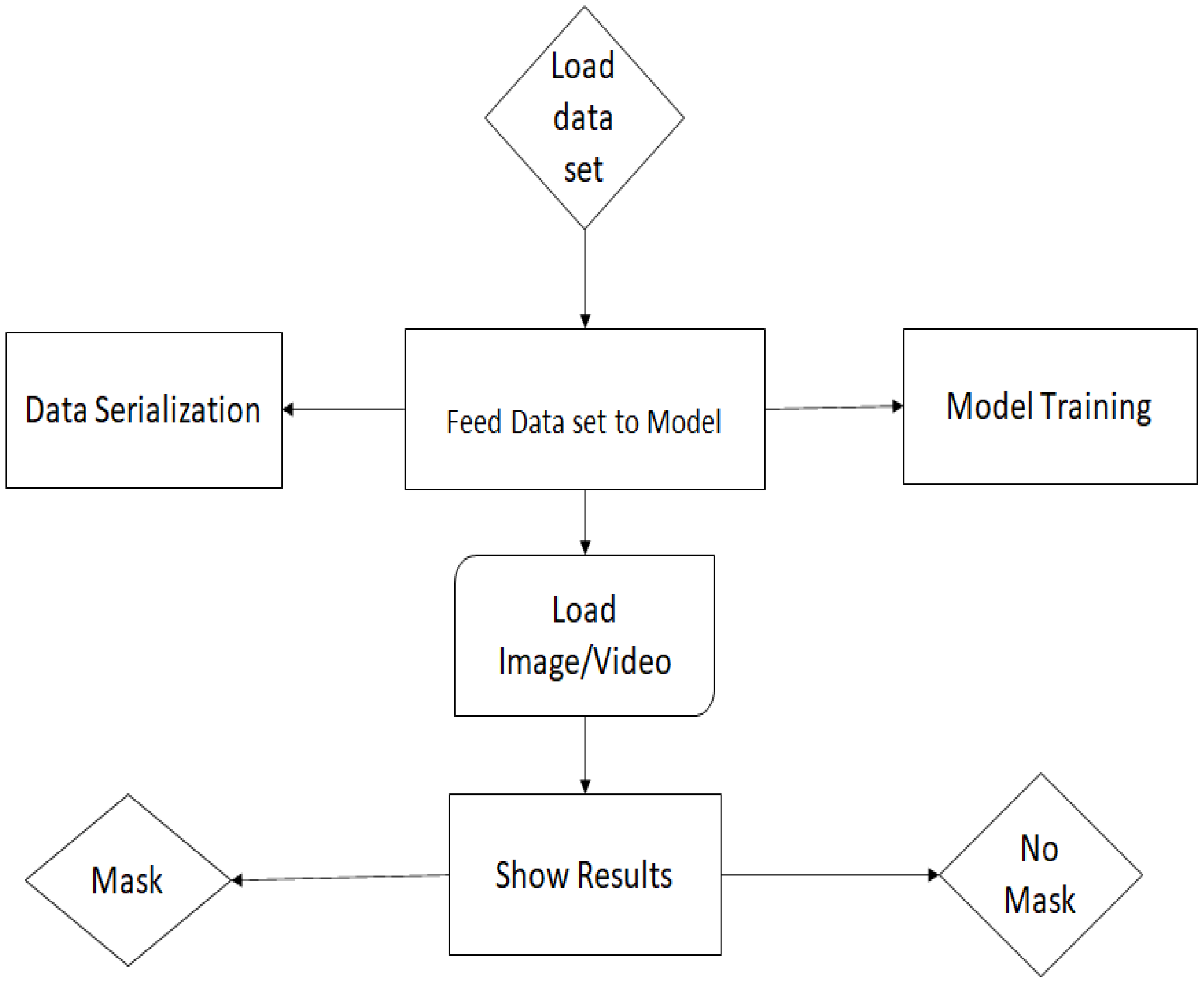


Fig.5.3.1

5.3 SYSTEM MODEL

5.3.1 USE CASE DIAGRAM

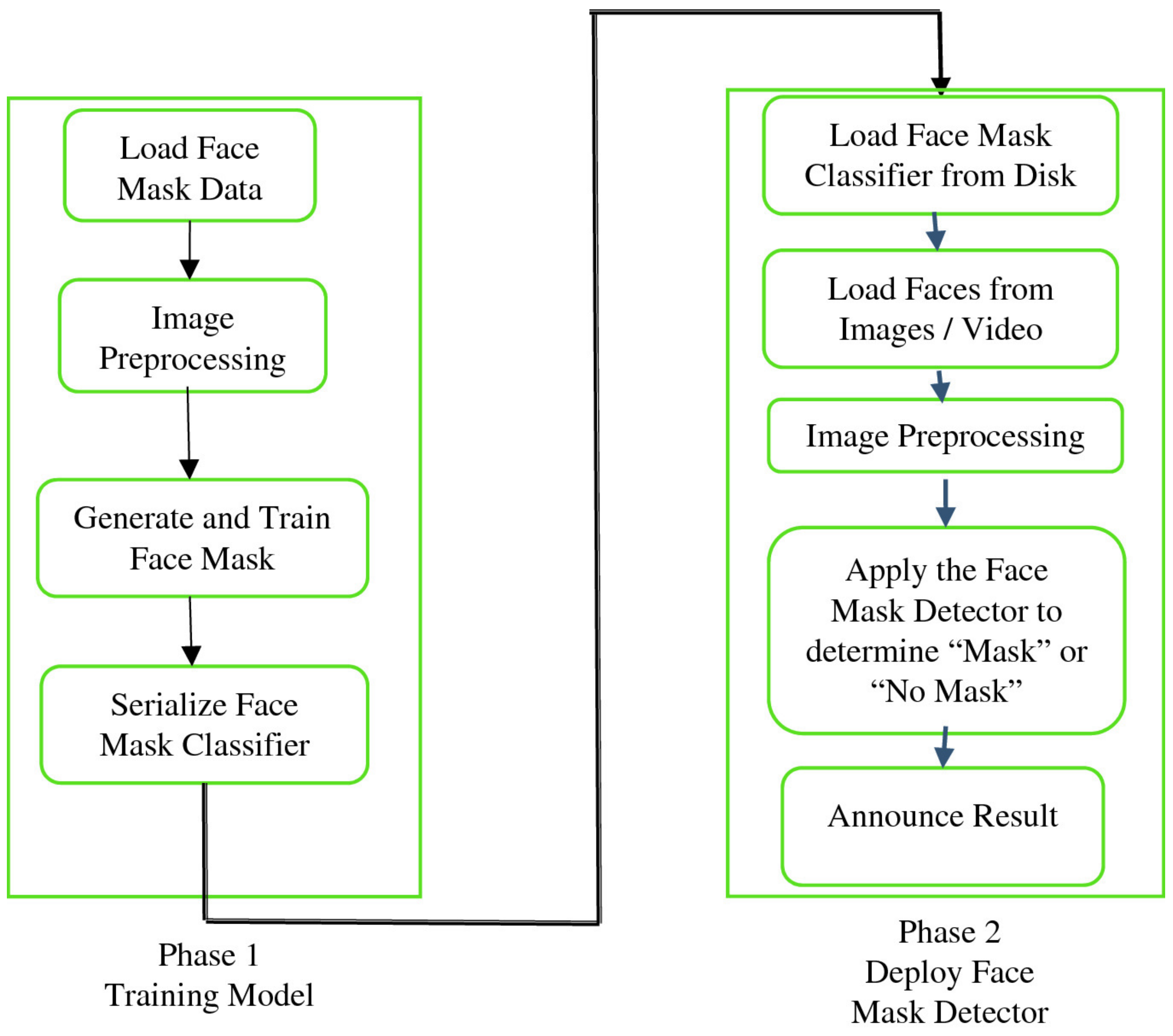

Load Face Mask Data
Image Preprocessing
Generate and Train Face Mask
Serialize Face Mask Classifier
Phase 1 Training Model
Load Face Mask Classifier from Disk
Load Faces from Images / Video
Image Preprocessing
Apply the Face Mask Detector to determine "Mask" or "No Mask"
Announce Result
Phase 2 Deploy Face Mask Detector

Fig.3.2.2

5.3.2 CNN (CONVOLUTIONAL NEURAL NETWORK)

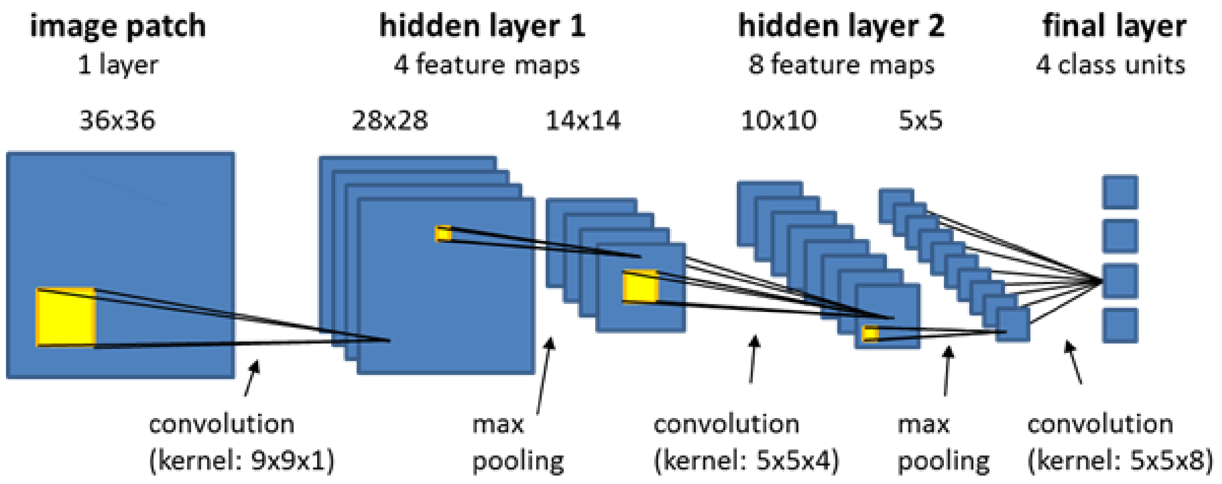


Fig.3.2.3

6.IMPLEMENTATION

6.1 CODE

TRAINING THE MASK DETECTOR

```
# 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\Siddartha\OneDrive\Desktop\MASK\dataset\data"

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)
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")
```

DETECTING MASK VIDEO

```
# 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"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()

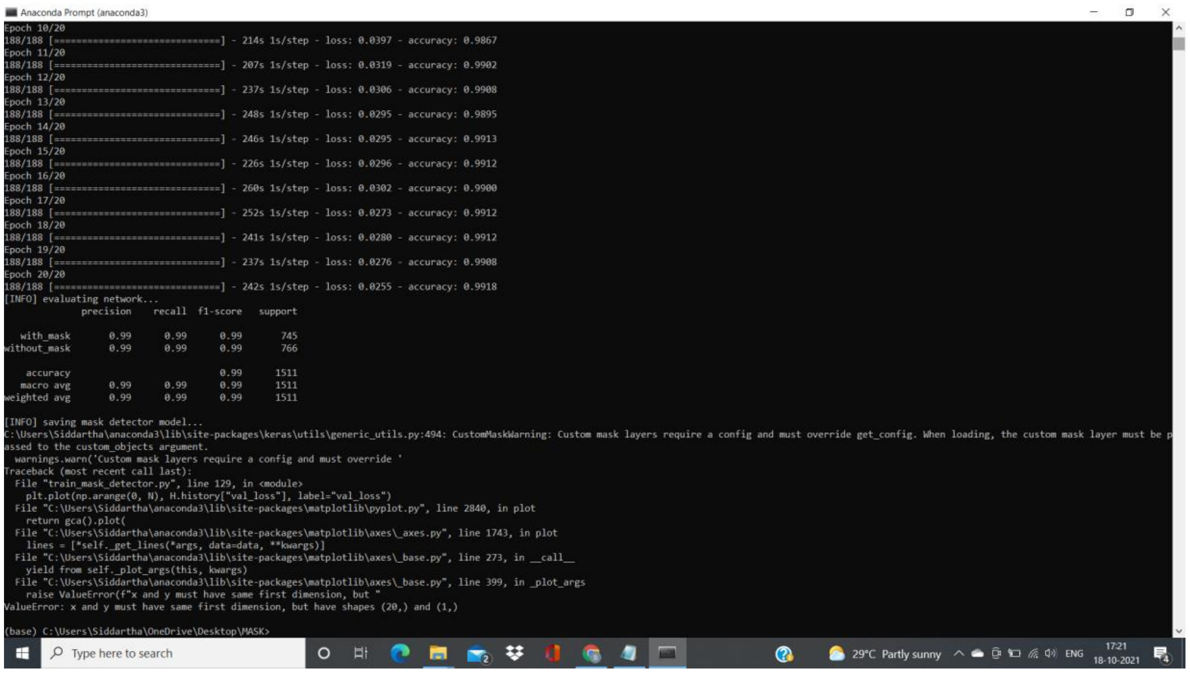
vs.stop()

7. RESULT

7.1 OUTPUT SCREENSHOT1

Training the dataset using the tensorflow and keras.

Fig.7.1.1



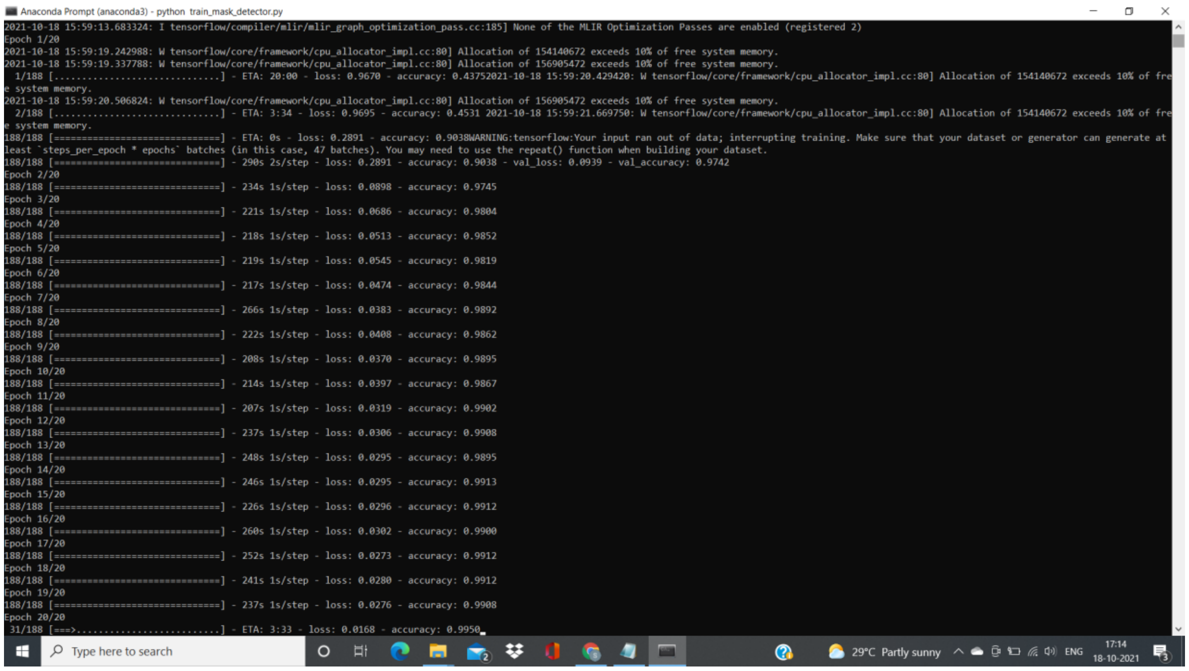
```
Anaconda Prompt (anaconda3)
Epoch 10/20
188/188 [==============================] - 214s 1s/step - loss: 0.0397 - accuracy: 0.9867
Epoch 11/20
188/188 [==============================] - 207s 1s/step - loss: 0.0319 - accuracy: 0.9902
Epoch 12/20
188/188 [==============================] - 237s 1s/step - loss: 0.0306 - accuracy: 0.9908
Epoch 13/20
188/188 [==============================] - 248s 1s/step - loss: 0.0295 - accuracy: 0.9895
Epoch 14/20
188/188 [==============================] - 246s 1s/step - loss: 0.0295 - accuracy: 0.9913
Epoch 15/20
188/188 [==============================] - 226s 1s/step - loss: 0.0296 - accuracy: 0.9912
Epoch 16/20
188/188 [==============================] - 260s 1s/step - loss: 0.0302 - accuracy: 0.9900
Epoch 17/20
188/188 [==============================] - 252s 1s/step - loss: 0.0273 - accuracy: 0.9912
Epoch 18/20
188/188 [==============================] - 241s 1s/step - loss: 0.0280 - accuracy: 0.9912
Epoch 19/20
188/188 [==============================] - 237s 1s/step - loss: 0.0276 - accuracy: 0.9908
Epoch 20/20
188/188 [==============================] - 242s 1s/step - loss: 0.0255 - accuracy: 0.9918
[INFO] evaluating network...
              precision    recall  f1-score   support

   with_mask       0.99      0.99      0.99       745
without_mask       0.99      0.99      0.99       766

    accuracy                           0.99      1511
   macro avg       0.99      0.99      0.99      1511
weighted avg       0.99      0.99      0.99      1511

[INFO] saving mask detector model...
C:\Users\Siddartha\anaconda3\lib\site-packages\keras\utils\generic_utils.py:494: CustomMaskWarning: Custom mask layers require a config and must override get_config. When loading, the custom mask layer must be p
assed to the custom_objects argument.
  warnings.warn('Custom mask layers require a config and must override '
Traceback (most recent call last):
  File "train_mask_detector.py", line 129, in <module>
    plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
  File "C:\Users\Siddartha\anaconda3\lib\site-packages\matplotlib\pyplot.py", line 2840, in plot
    return gca().plot(
  File "C:\Users\Siddartha\anaconda3\lib\site-packages\matplotlib\axes\_axes.py", line 1743, in plot
    lines = [*self._get_lines(*args, data=data, **kwargs)]
  File "C:\Users\Siddartha\anaconda3\lib\site-packages\matplotlib\axes\_base.py", line 273, in __call__
    yield from self._plot_args(this, kwargs)
  File "C:\Users\Siddartha\anaconda3\lib\site-packages\matplotlib\axes\_base.py", line 399, in _plot_args
    raise ValueError(f"x and y must have same first dimension, but "
ValueError: x and y must have same first dimension, but have shapes (20,) and (1,)

(base) C:\Users\Siddartha\OneDrive\Desktop\MASK>
```

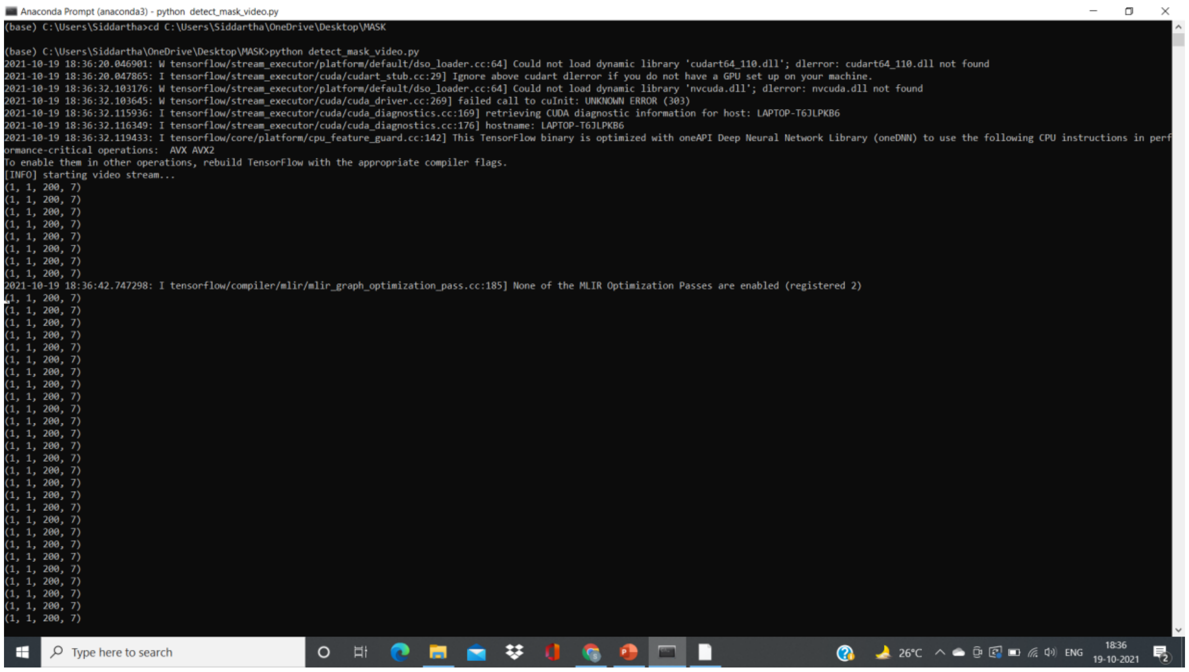


```
2021-10-18 15:59:13.683324: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)
Epoch 1/20
2021-10-18 15:59:19.242988: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 154140672 exceeds 10% of free system memory.
2021-10-18 15:59:19.337788: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 156905472 exceeds 10% of free system memory.
  1/188 [..............................] - ETA: 20:00 - loss: 0.9670 - accuracy: 0.43752021-10-18 15:59:20.429420: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 154140672 exceeds 10% of fre
e system memory.
2021-10-18 15:59:20.506824: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 156905472 exceeds 10% of free system memory.
  2/188 [..............................] - ETA: 3:34 - loss: 0.9695 - accuracy: 0.4531 2021-10-18 15:59:21.669750: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 154140672 exceeds 10% of fre
e system memory.
188/188 [==============================] - ETA: 0s - loss: 0.2891 - accuracy: 0.9038WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at
least `steps_per_epoch * epochs` batches (in this case, 47 batches). You may need to use the repeat() function when building your dataset.
188/188 [==============================] - 290s 2s/step - loss: 0.2891 - accuracy: 0.9038 - val_loss: 0.0939 - val_accuracy: 0.9742
Epoch 2/20
188/188 [==============================] - 234s 1s/step - loss: 0.0898 - accuracy: 0.9745
Epoch 3/20
188/188 [==============================] - 221s 1s/step - loss: 0.0686 - accuracy: 0.9804
Epoch 4/20
188/188 [==============================] - 218s 1s/step - loss: 0.0513 - accuracy: 0.9852
Epoch 5/20
188/188 [==============================] - 219s 1s/step - loss: 0.0545 - accuracy: 0.9819
Epoch 6/20
188/188 [==============================] - 217s 1s/step - loss: 0.0474 - accuracy: 0.9844
Epoch 7/20
188/188 [==============================] - 266s 1s/step - loss: 0.0383 - accuracy: 0.9892
Epoch 8/20
188/188 [==============================] - 222s 1s/step - loss: 0.0408 - accuracy: 0.9862
Epoch 9/20
188/188 [==============================] - 208s 1s/step - loss: 0.0370 - accuracy: 0.9895
Epoch 10/20
188/188 [==============================] - 214s 1s/step - loss: 0.0397 - accuracy: 0.9867
Epoch 11/20
188/188 [==============================] - 207s 1s/step - loss: 0.0319 - accuracy: 0.9902
Epoch 12/20
188/188 [==============================] - 237s 1s/step - loss: 0.0306 - accuracy: 0.9908
Epoch 13/20
188/188 [==============================] - 248s 1s/step - loss: 0.0295 - accuracy: 0.9895
Epoch 14/20
188/188 [==============================] - 246s 1s/step - loss: 0.0295 - accuracy: 0.9913
Epoch 15/20
188/188 [==============================] - 226s 1s/step - loss: 0.0296 - accuracy: 0.9912
Epoch 16/20
188/188 [==============================] - 260s 1s/step - loss: 0.0302 - accuracy: 0.9900
Epoch 17/20
188/188 [==============================] - 252s 1s/step - loss: 0.0273 - accuracy: 0.9912
Epoch 18/20
188/188 [==============================] - 241s 1s/step - loss: 0.0280 - accuracy: 0.9912
Epoch 19/20
188/188 [==============================] - 237s 1s/step - loss: 0.0276 - accuracy: 0.9908
Epoch 20/20
 31/188 [===>..........................] - ETA: 3:33 - loss: 0.0168 - accuracy: 0.9950
```

Fig.7.1.2

Detecting the mask in the live video stream using MobileNetV2

Fig.7.1.3



```
(base) C:\Users\Siddartha>cd C:\Users\Siddartha\OneDrive\Desktop\MASK

(base) C:\Users\Siddartha\OneDrive\Desktop\MASK>python detect_mask_video.py
2021-10-19 18:36:20.046901: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2021-10-19 18:36:20.047865: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2021-10-19 18:36:32.103176: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found
2021-10-19 18:36:32.103645: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2021-10-19 18:36:32.115936: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: LAPTOP-T6JLPKB6
2021-10-19 18:36:32.116349: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: LAPTOP-T6JLPKB6
2021-10-19 18:36:32.119433: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in perf
ormance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
[INFO] starting video stream...
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
2021-10-19 18:36:42.747298: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
```

7.2 RESULT ANALYSIS

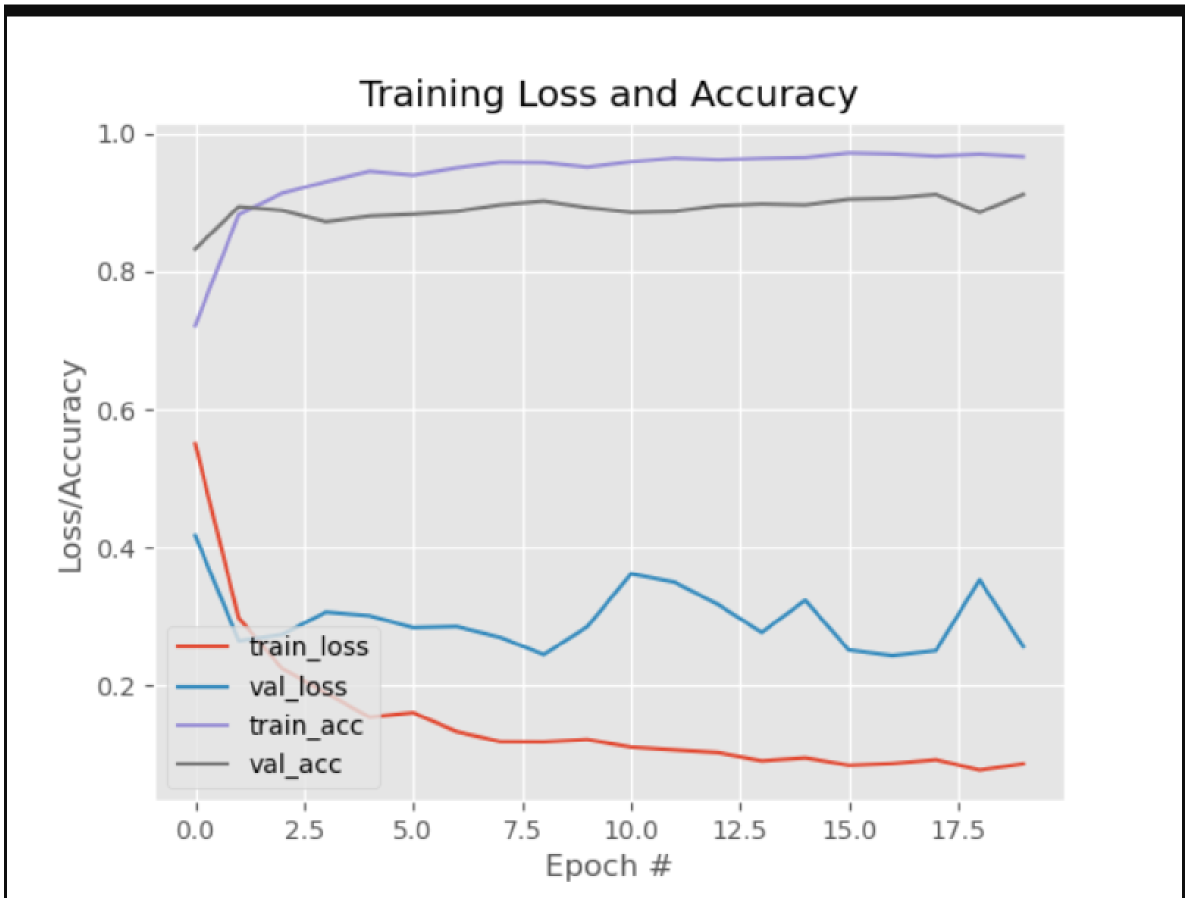
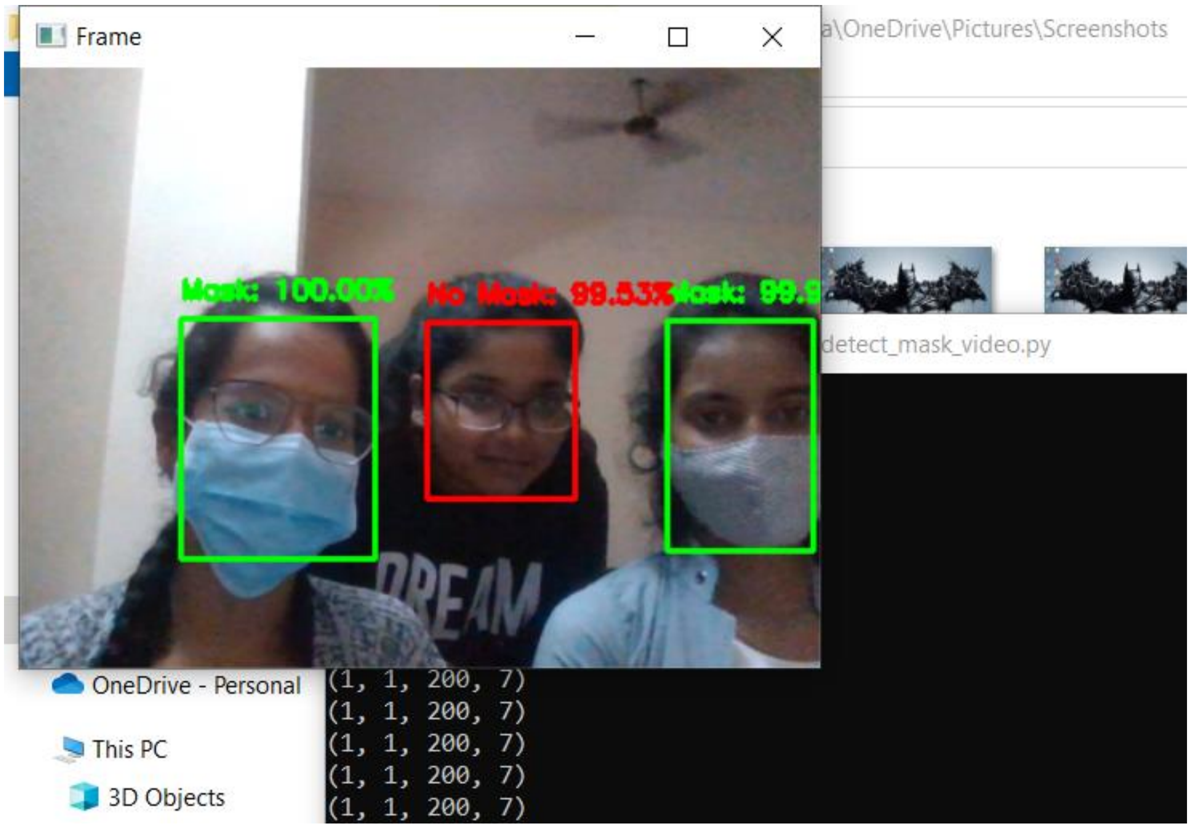


Fig.7.2.1



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

Fig.7.2.2

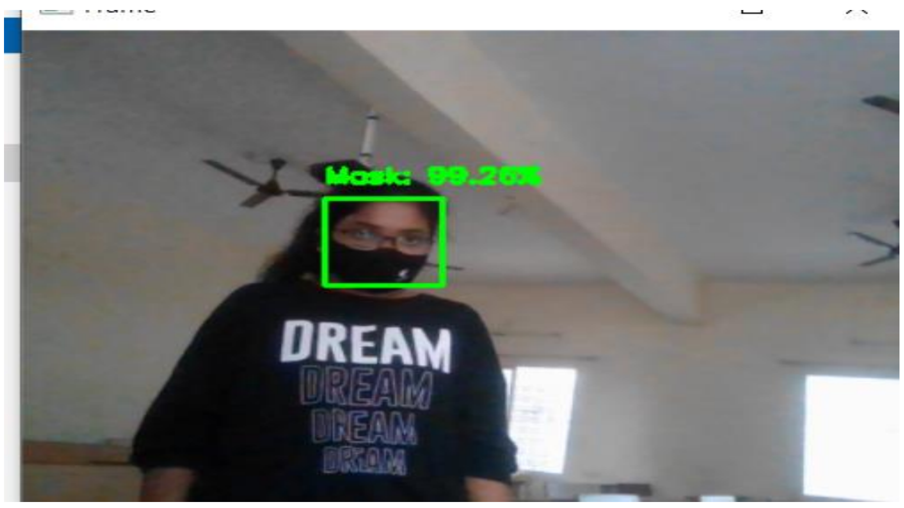


Fig.7.2.3

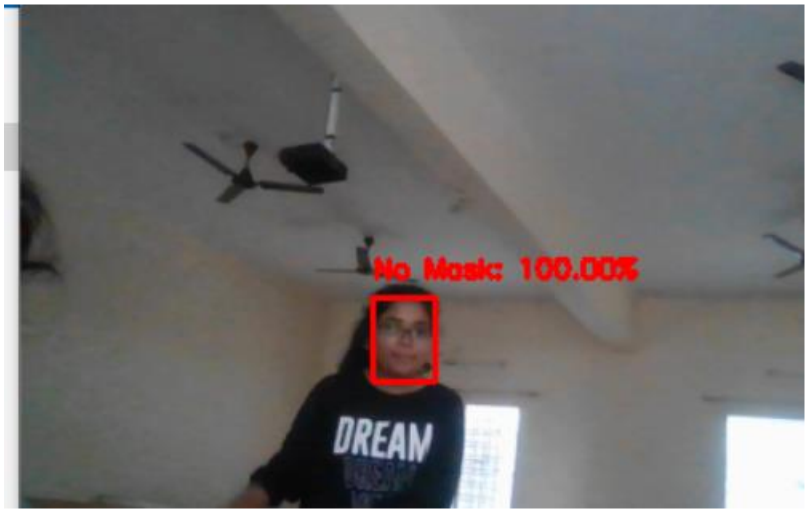


Fig.7.2.3

8. CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

Due to the urgency of controlling COVID-19, the application value and importance of real-time mask and social distancing detection are increasing. This work reviewed, firstly, many research works that seek to surround COVID-19 outbreak. Then, it clarified the basic concepts of deep CNN models. After that, this paper reproduced the training and testing of the most used deep pretrained-based CNN models (MobileNet, MobileNetV2) on the face mask dataset. Finally and after evaluating the numerical results, best models are tested on an embedded vision system consisting of live video stream and webcam where efficient real-time deep learning-based techniques are implemented with a social distancing task to automate the process of detecting masked faces and maintained distance between peoples.

This embedded vision-based application can be used in any working environment such as public place, station, corporate environment, streets, shopping malls, and examination centers, Hospitals, where accuracy and precision are highly desired to serve the purpose. It can be used in smart city innovation, and it would boost up the development process in many developing countries. Our framework presents a chance to be more ready for the next crisis or to evaluate the effects of huge scope social change in respecting sanitary protection rules.

In future works, we will exploit this methodology on smart sensors or connected RP nodes that will be considered as an Edge Cloud to collect multimedia data, e.g., an autonomous drone system, which can provide capture (by the camera) of the detected objects from different angles and send them to the Edge Cloud system to be analyzed.

8.2 FUTURE ENHANCEMENT

The work opens interesting future directions for researchers.
Firstly, the proposed technique can be integrated into any high-resolution video surveillance devices and not limited to mask detection only.
Secondly, the model can be extended to detect facial landmarks with a facemask for biometric purposes.
This project has been developed to come up with an efficient way for detecting and notifying officials when a person does not follow the COVID 19 safety protocols in a workplace, business establishments etc. In this work, we have trained a model for face mask detection using TensorFlow and Keras and used YOLO Object detection for detecting social distancing. The proposed CNN architecture comprises two convolutional layers followed by relu activation function and a max pooling layer. YOLOv3 was used to detect people in a frame and find the Euclidean distance between them. With the help of OpenCV we were able to capture the video feed from different sources like webcam, video file or an IP camera. An android app was developed which will get notified every time a violation is detected, and the detected images can also be viewed through the app. This was achieved with the help of Firebase service. As a future study, we can work on finding a pattern to detect or

predict the time at which it gets crowded the most and the heat map can be plotted in a more accurate manner.

9. REFERENCES

DATASET

https://www.kaggle.com/akashguna/lfw-dataset-with-masks/tasks?taskId=554

1. M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, “A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic,” *Measurement*, vol. 167, Article ID 108288, 2021.
2. View at: Publisher Site | Google Scholar
3. B. Qin and D. Li, “Identifying facemask-wearing condition using image super-resolution with classification network to prevent COVID-19,” *Sensors*, vol. 20, no. 18, p. 5236, 2020.
4. View at: Publisher Site | Google Scholar
5. Z. Wang, P. Wang, P. C. Louis, L. E. Wheless, and Y. Huo, “Wearmask: fast In-browser face mask detection with serverless edge computing for COVID-19,” 2021, https://arxiv.org/abs/2101.00784.