

REAL TIME FACEMASK DETECTION USING PYTHON

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

PATAPATI MOURYA VARMA (Reg .No: 37110551)

PATHAN ARFATH KHAN (Reg. No:37110552)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC |
12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI – 600 119

MARCH - 2021



SATHYABAMA



INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC | 12B Status by UGC | Approved by AICTE
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **PATAPATI MOURYA VARMA (37110551)** and **PATHAN ARFATH KHAN (37110552)** who carried out the project entitled "**REAL TIME FACEMASK DETECTION USING PYTHON**" under my supervision from **November 2020** to **March 2021**.

Internal Guide

Dr. A. VIJI AMUTHA MARY, M.Tech., Ph.D,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Dr.L.LAKSHMANAN M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I PATAPATI MOURYA VARMA (37110551), PATHAN ARFATH KHAN (37110552) hereby declare that the Project Report entitled **“REALTIME FACEMASK DETECTION USING PYTHON”** done by me under the guidance of **Dr. A. VIJI AMUTHA MARY, M.Tech., Ph.D**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.SASIKALA M.E., Ph.D., Dean**, School of Computing , **Dr.S.VIGNESHWARI M.E., Ph.D.,** and **Dr.L.LAKSHMANAN M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A. VIJI AMUTHA MARY, M.Tech., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In order to effectively prevent the spread of COVID-19 virus, almost everyone wears a mask during coronavirus epidemic. This almost makes conventional facial recognition technology ineffective in many cases, such as community access control, face access control, facial attendance, facial security checks at train stations, etc. Therefore, it is very urgent to improve the recognition performance of the existing face recognition technology on the masked faces. Most current advanced face recognition approaches are designed based on deep learning, which depend on a large number of face samples. However, at present, there are no publicly available masked face recognition datasets. To this end, this work proposes three types of masked face datasets, including Masked Face Detection Dataset (MFDD), Real-world Masked Face Recognition Dataset (RMFRD) and Simulated Masked Face Recognition Dataset (SMFRD). Among them, to the best of our knowledge, RMFRD is currently the world's largest real-world masked face dataset. These datasets are freely available to industry and academia, based on which various applications on masked faces can be developed. The multigranularity masked face recognition model we developed achieves 95% accuracy, exceeding the results reported by the industry.

TABLE OF CONTENTS

CHAPTER.NO	TITLE	PAGE NO
	ABSTRACT	i
	LIST OF FIGURES	ii
1	INTRODUCTION	1
	1.1 OUTLINE OF THE PROJECT	1
	1.2 OVERVIEW	1
2	AIM AND SCOPE	3
	2.1 AIM	3
	2.2 SCOPE	3
	2.2.1 APPLICATIONS	4
	2.2.2 DATASET	4
	2.2.3 TRAINING PROCESS	4
	2.2.4 MASK DETECTION	4
3	SYSTEM IMPLEMENTATION	5
	3.1 INTRODUCTION	5
	3.2 SOFTWARE SPECIFICATION	5
	3.3 HARDWARE REQUIREMENTS	6
	3.4 INTRODUCTION TO PYTHON	6
	3.5 APLICATIONS	6
	3.6 DATASET	7
	3.7 TRAINING PROCESS	8
	3.8 MASK DETECTION	8
4	RESULTS AND DISSCUSSIONS	10
	4.1 RESULTS	10
5	CONCLUSION AND FUTURE WORKS	13
	5.1 CONCLUSION	13
	5.2 FUTURE WORKS	13

REFERENCES	14
APPENDIX	15
A.. SOURCE CODE	15
B. SCREENSHOTS	22

LIST OF FIGURES

FIG 3.1 – IMPORTING APPLICATIONS

FIG 3.2 – IMPORTING APPLICATIONS

FIG 3.3 – WITHMASK DATASET

FIG 3.4 – WITHOUTMASK DATASET

FIG 3.5 – FACE DETECTION

FIG 3.6 – FACE MARKING

FIG 3.7- WITHMASK DETECTION

FIG 3.8 – ARCHITETURE OF TRAINING

FIG 4.1 – TRAINING THE COMPUTER

FIG 4.2 – DETECT MASK CODE

FIG 4.3 – COMPILING OF CODE

FIG 4.4 – DETECTING THE PERSON FOR A MASK

FIG 4.5 – DETECTION RESULT

CH

APT

1.1 OUTLINE OF PROJECT

ER 1

INTR

ODU

CTIO

N

Considering the covid-19 outbreak, I think this is best project to work on python . Today everyone is aware of taking precaution and safety measures regarding covid-19, so face mask detection will play a huge role to avoid corona virus. This project helps us to spread the awareness among people using face mask properly. It detects the face mask on your face whether the person is hiding his/her face by mask or not. It also check the face mask is properly cover your face both nose and mouth. This is so helpful when there is a gathering of people it helps in detecting people in a real time aspect which is more helpful in a screening process of any event .

1.2 OVERVIEW

The World Health Organization (WHO) reports suggest that the two main routes of transmission of the COVID-19 virus are **respiratory droplets** and **physical contact** .Respiratory droplets are generated when an infected person coughs or sneezes. Any person in close contact (within 1 m) with someone who has respiratory symptoms (coughing, sneezing) is at risk of being exposed to potentially infective respiratory droplets. Droplets may also land on surfaces where the virus could remain viable; thus, the immediate environment of an infected individual can serve as a source of transmission (contact transmission). Wearing a medical mask is one of the prevention measures that can limit the spread of certain respiratory viral diseases, including COVID-19.

Face recognition techniques, the most important means of identification, have nearly failed ,which has brought huge dilemmas to authentication applications that rely on face recognition, such as community entry and exit, face access

control, face attendance, face gate at train stations, face authentication based mobile payment ,face recognition based social security investigation, etc. In particular, in the public security check like railway stations ,the gates based on traditional face recognition systems cannot effectively recognize the masked

faces, but removing masks for passing authentication will increase the risk of virus infection.

In this study, medical masks are defined as surgical or procedure masks that are flat or pleated (some are shaped like cups); they are affixed to the head with straps. They are tested for balanced high filtration, adequate breathability and optionally, fluid penetration resistance. The study analyses a set of video streams/images to identify people who are compliant with the government rule of wearing medical masks. This could help the government to take appropriate action against people who are non-compliant.

CHAPTER 2

AIM AND SCOPE

2.1 AIM

The aim of the project is to detect the person who is facing the camera whether he or she is wearing the mask in a real time. In order to do so we have to train the computer with use of some libraries in python

2.2 SCOPE

The corona virus disease 2019 (COVID-19) has globally infected over 2.7million people and caused over 180,000 deaths. There are several similar large scale serious respiratory diseases, such as severe acute respiratory syndrome (SARS) and the Middle East respiratory syndrome (MERS), which occurred in the past few years. Therefore, more and more people are concerned about their health, and public health is considered as the top priority for governments. Furthermore, many public service providers require customers to use the service only if they wear masks. Face mask detection has become a crucial computer vision task to help the global society, but research related to face mask detection is limited.

Corona virus 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. So, it is a very big issue and virus spreading problem. That's the reason for we are taking in this project. We propose Retina Face Mask, which is a high-accuracy and efficient face mask detector. The proposed Retina Face Mask 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. Face mask detection refers to detect whether a person wearing a mask or not and

what is the location of the face.

Applications
Data set
Training process
Face mask detection

2.2.1 APPLICATIONS

The applications are the library materials used in the project for the python .This includes Tensorflow,Kears , Imutils, Numpy , Opencv-python , Matplotlib and Scipy. These are to be imported or installed into the python inorder to work the program. Command prompt is opened to install all the applications which are needed as applications with a command .

2.2.2 DATA SET

First we collected a dataset form the internet. We are collecting 2 types of dataset with mask dataset and without mask dataset. These are the main training mods which help to train the program

2.2.3 TRAINING PROCESS

We are train the collected video frames using Deep leaning techniques. Because, it will train to multiple times and push a very fast results. Deep Learning is a one of the best algorithm for making a project and it is also give an accurate output.

2.2.4 MASK DETECTION

Finally, it will working in real time detection with mask or without mask. The detection will occur with a square box around the person face , if the mask is on the color of the box will be green and shows mask on, if not the color will be red and labiling will be no mask.

CHAPTER 3

SYSTEM IMPLEMENTATION

3.1 INTRODUCTION

In this we'll discuss our two-phase COVID-19 face mask detector, detailing how our computer vision/deep learning pipeline will be implemented. From there, we'll review the dataset we'll be using to train our custom face mask detector. I'll then show you how to implement a Python script to train a face mask detector on our dataset using Keras and TensorFlow.

We'll use this Python script to train a face mask detector and review the results.

Given the trained COVID-19 face mask detector, we'll proceed to implement two more additional Python scripts used to

1. Detect COVID-19 face masks in images
2. Detect face masks in real time video stream

We'll wrap up the post by looking at the results of applying our face mask detector.

I'll also provide some additional suggestions for further improvement.

3.2 SOFTWARE REQUIREMENTS

- Operating System : Windows 7 , 8, 10 (64 bit)
- Software : Python 3.7
- Tools : Comand Prompt

3.3 HARDWARE REQUIREMENTS

- Hard Disk : 500GB and Above
- RAM : 4GB and Above
- Processor : I3 and Above

3.4 INTRODUCTION TO PYTHON

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

3.5 APPLICATIONS

The applications are the library materials used in the project for the python .This includes Tensorflow,Kears , Imutils, Numpy , Opencv-python , Matplotlib and Scipy. These are to be imported or installed into the python inorder to work the program. Command prompt is opened to install all the applications which are needed as applications with a command .

- TENSORFLOW
- KEARS
- IMUTILS
- NUMPY
- OPENCV – PYTHON
- MATPLOTLIB
- SCIPY


```

train_mask_detector.py x detect_mask_video.py x
1 # import the necessary packages
2 from tensorflow.keras.preprocessing.image import ImageDataGenerator
3 from tensorflow.keras.applications import MobileNetV2
4 from tensorflow.keras.layers import AveragePooling2D
5 from tensorflow.keras.layers import Dropout
6 from tensorflow.keras.layers import Flatten
7 from tensorflow.keras.layers import Dense
8 from tensorflow.keras.layers import Input
9 from tensorflow.keras.models import Model
10 from tensorflow.keras.optimizers import Adam
11 from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
12 from tensorflow.keras.preprocessing.image import img_to_array
13 from tensorflow.keras.preprocessing.image import load_img
14 from tensorflow.keras.utils import to_categorical
15 from sklearn.preprocessing import LabelBinarizer
16 from sklearn.model_selection import train_test_split
17 from sklearn.metrics import classification_report
18 from imutils import paths
19 import matplotlib.pyplot as plt
20 import numpy as np
21 import os

```

Fig 3.1 importing applications

```

train_mask_detector.py x detect_mask_video.py x
1 # import the necessary packages
2 from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
3 from tensorflow.keras.preprocessing.image import img_to_array
4 from tensorflow.keras.models import load_model
5 from imutils.video import VideoStream
6 import numpy as np
7 import imutils
8 import time
9 import cv2
10 import os

```

Fig 3.2 importing applications

3.6 DATA SET

First we collected a dataset from the internet. We are collecting 2 types of dataset with mask dataset and without mask dataset. These are the main training mods which help to train the program .



Fig 3.3 with mask dataset.



Fig 3.4 without mask dataset.

3.7 TRAINING PROCESS

We are train the collected video frames using Deep learning techniques. Because, it will train to multiple times and push a very fast results. Deep Learning is a one of the best algorithm for making a project and it is also give an accurate output.

3.8 MASK DETECTION

Finally, it will working in real time detection with mask or without mask. The detection will occur with a square box around the person face , if the mask is on the color of the box will be green and shows mask on, if not the color will be red and labiling will be no mask.

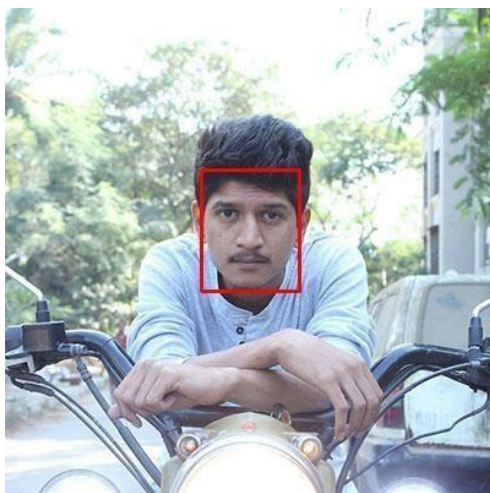


Fig 3.5 face detection

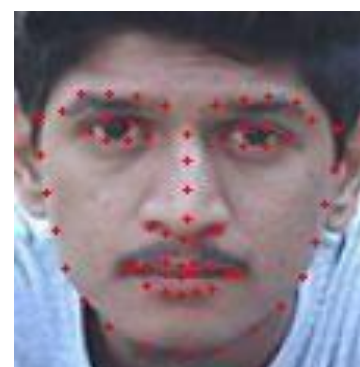


Fig 3.6 face marking



Fig 3.7 with mask detection

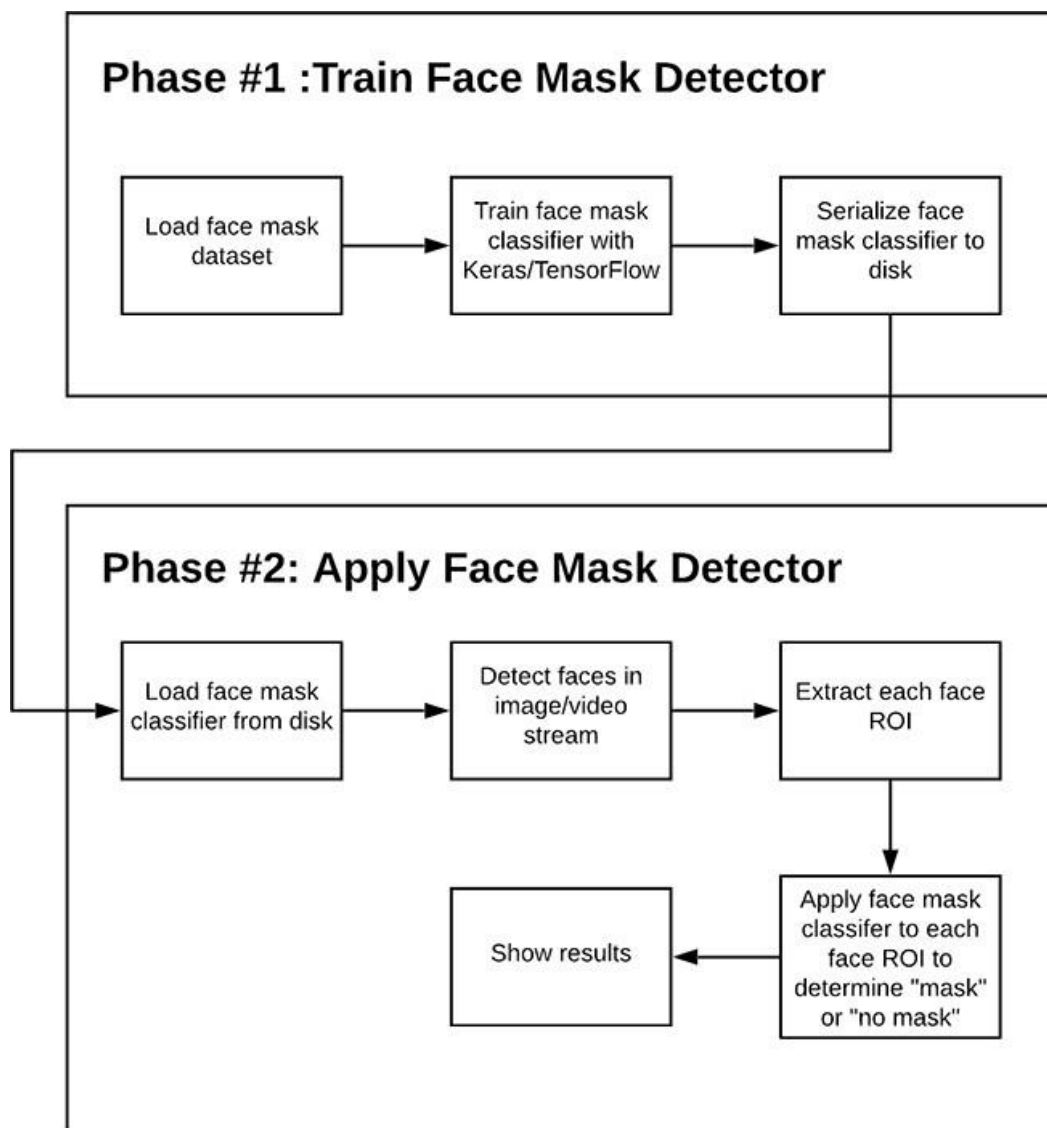


Fig 3.8 Architecture of training

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 RESULTS

The `train_mask_detect.py` code run in the command prompt and takes the data from the dataset. The python trains the computer with the pictures of with mask and without mask

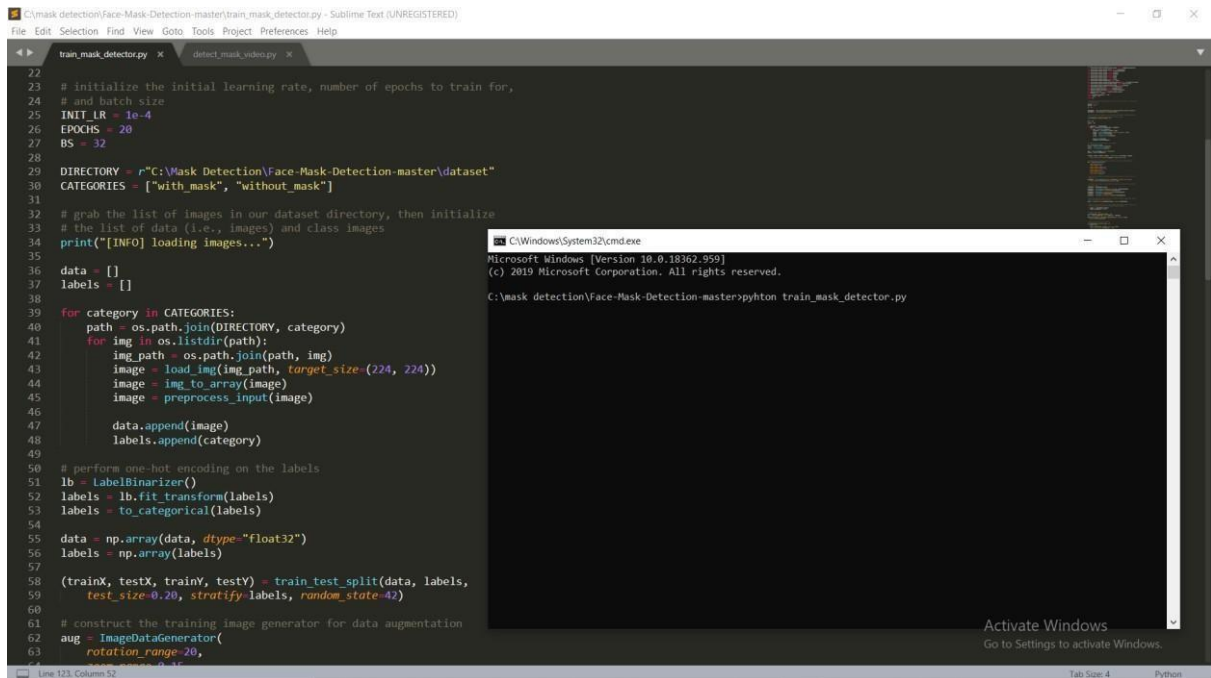


Fig 4.1 Training the computer

Opening the command prompt with the desired path of the code saved in the drive the path should be a compulsory part to run the or train the computer. The code run on the basis of python while training the computer with data from the data sets which have two folders with mask dataset and without mask dataset.

- `train_mask_detector.py`

: Accepts our input dataset and fine-tunes MobileNetV2 upon it to create our

`mask_detector.model`

. A training history

`plot.png`

containing accuracy/loss curves is also produced

•
•

- detect_mask_image.py

: Performs face mask detection in static images

- detect_mask_video.py

: Using your webcam, this script applies face mask detection to every frame in the stream.

```

1 # import the necessary packages
2 from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
3 from tensorflow.keras.preprocessing.image import img_to_array
4 from tensorflow.keras.models import load_model
5 from imutils.video import VideoStream
6 import numpy as np
7 import imutils
8 import time
9 import cv2
10 import os
11
12 def detect_and_predict_mask(frame, faceNet, maskNet):
13     # grab the dimensions of the frame and then construct a blob
14     # from it
15     (h, w) = frame.shape[:2]
16     blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
17                                 (104.0, 177.0, 123.0))
18
19     # pass the blob through the network and obtain the face detect
20     faceNet.setInput(blob)
21     detections = faceNet.forward()
22     print(detections.shape)
23
24     # initialize our list of faces, their corresponding locations,
25     # and the list of predictions from our face mask network
26     faces = []
27     locs = []
28     preds = []
29
30     # loop over the detections
31     for i in range(0, detections.shape[2]):
32         # extract the confidence (i.e., probability) associated with
33         # the detection
34         confidence = detections[0, 0, i, 2]
35
36         # filter out weak detections by ensuring the confidence is
37         # greater than the minimum confidence
38         if confidence > 0.5:
39             # compute the (x, y)-coordinates of the bounding box for
40             # the object
41             box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
42             (startX, startY, endX, endY) = box.astype("int")

```

fig 4.2 detect mask code

```

1 # import the necessary packages
2 from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
3 from tensorflow.keras.preprocessing.image import img_to_array
4 from tensorflow.keras.models import load_model
5 from imutils.video import VideoStream
6 import numpy as np
7 import imutils
8 import time
9 import cv2
10 import os
11
12 def detect_and_predict_mask(frame, faceNet, maskNet):
13     # grab the dimensions of the frame and then construct a blob
14     # from it
15     (h, w) = frame.shape[:2]
16     blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
17                                 (104.0, 177.0, 123.0))
18
19     # pass the blob through the network and obtain the face detect
20     faceNet.setInput(blob)
21     detections = faceNet.forward()
22     print(detections.shape)
23
24     # initialize our list of faces, their corresponding locations,
25     # and the list of predictions from our face mask network
26     faces = []
27     locs = []
28     preds = []
29
30     # loop over the detections
31     for i in range(0, detections.shape[2]):
32         # extract the confidence (i.e., probability) associated with
33         # the detection
34         confidence = detections[0, 0, i, 2]
35
36         # filter out weak detections by ensuring the confidence is
37         # greater than the minimum confidence
38         if confidence > 0.5:
39             # compute the (x, y)-coordinates of the bounding box for
40             # the object
41             box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
42             (startX, startY, endX, endY) = box.astype("int")

```

Fig 4.3 compiling of code

The compiling leads to the allowment of webcam to ensure the working and the feed will be live in order to detect the person the detection method draws a square box around the person face



Fig 4.4 detecting the person for a mask

Computer detect the person face with a detection box around the face, the training from the dataset help to detect the person with a possibility around the box and the red color says that there is no mask on the persons face.



Fig 4.5 detection result

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 CONCLUSION

The current study used OpenCV, Python and CNN to detect whether people were wearing face masks or not. The models were tested with images and real-time video streams. Even though the accuracy of the model is around 80%, the optimization of the model is a continuous process and we are building a highly accurate solution by tuning the hyper parameters. MobileNetV2 was used to build the mobile version of the same. This specific model could be used as a use case for edge analytics.

5.2 FUTURE WORKS

- **Deep learning fundamentals and theory without unnecessary mathematical fluff.** I present the basic equations and back them up with code walkthroughs that you can implement and easily understand. You don't need a degree in advanced mathematics to understand this book
- **How to implement your own custom neural network architectures.** Not only will you learn how to implement state-of-the-art architectures, including ResNet, SqueezeNet, etc., but you'll *also* learn how to create your own custom CNNs
- **How to train CNNs on your own datasets.** Most deep learning tutorials don't teach you how to work with your own custom datasets. Mine do. You'll be training CNNs on your own datasets in no time
- **Object detection (Faster R-CNNs, Single Shot Detectors, and RetinaNet) and instance segmentation (Mask R-CNN).** Use these chapters to create your own custom object detectors and segmentation networks

REFERENCES

- [1] W. H. Organization et al., "Coronavirus disease 2019 (covid-19): situation report, 96," 2020.
- [2] P. A. Rota, M. S. Oberste, S. S. Monroe, W. A. Nix, R. Campagnoli, J. P. Icenogle, S. Penaranda, B. Bankamp, K. Maher, M.-h. Chen et al., "Characterization of a novel coronavirus associated with severe acute respiratory syndrome," *science*, vol. 300, no. 5624, pp. 1394–1399, 2003.
- [3] Z. A. Memish, A. I. Zumla, R. F. Al-Hakeem, A. A. Al-Rabeeah, and G. M. Stephens, "Family cluster of middle east respiratory syndrome coronavirus infections," *New England Journal of Medicine*, vol. 368, no. 26, pp. 2487–2494, 2013.
- [4] Y. Liu, A. A. Gayle, A. Wilder-Smith, and J. Rocklöv, "The reproductive number of covid-19 is higher compared to sars coronavirus," *Journal of travel medicine*, 2020.
- [5] Y. Fang, Y. Nie, and M. Penny, "Transmission dynamics of the covid-19 outbreak and effectiveness of government interventions: A data-driven analysis," *Journal of medical virology*, vol. 92, no. 6, pp. 645–659, 2020.
- [6] N. H. Leung, D. K. Chu, E. Y. Shiu, K.-H. Chan, J. J. McDevitt, B. J. Hau, H.-L. Yen, Y. Li, D. KM, J. Ip et al., "Respiratory virus shedding in exhaled breath and efficacy of face masks."
- [7] S. Feng, C. Shen, N. Xia, W. Song, M. Fan, and B. J. Cowling, "Rational use of face masks in the covid-19 pandemic," *The Lancet Respiratory Medicine*, 2020.

APPENDIX

A. SOURCE CODE:

detect_mask_video.py

```
# 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()
```

train_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:\face mask\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
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")

```

B. SCREENSHOTS

