FACE MASK DETECTION

A PROJECT REPORT

Submitted by

RUTUJA RAMDAS UGALE (FS19IF011)

TRUPTI ASHOK KADAM (FS19IF026)

JAGRUTI ANANT LIGAM (FS19IF028)

ALPESH ANANT GAYKER (FS19IF029)

In partial fulfillment for the award of

Diploma

In

INFORMATION TECNOLOGY

DEPARTMENT OF INFORMATON TECHNOLGY

GOVERNMENT POLYTECHIC MUMBAI

GOVERNMENT POLYTECHNIC MUMBAI

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled **“FACE MASK DETECTION" is the bonafide work of “RUTUJA UGALE (FS19IF011), TRUPTI KADAM (FS19IF026), JAGRUTI LIGAM (FS19IF028), ALPESH GAYKER (FS19IF029)"** , submitted in partial fulfillment of the of Government Polytechnic Mumbai.

Dr. Seema yardi	Dr. Seema yardi
HEAD OF THE DEPARTMENT	GUIDE NAME
Mrs. Swati. D. Deshpande	
PRINCIPAL	EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project entitled "FACE MASK DETECTION" being submitted by us towards the partial fulfillment of the requirements for the award of Diploma in information Technology is a project work carried by us under the supervision of Dr. Seema yardi and have not been submitted anywhere else.

We will be solely responsible if any kind plagiarism is found.

Date:-

Name of the Students	Enrollment No	Signature
1. RUTUJA UGALE	FS19IF011	
2. TRUPTI KADAM	FS19IF026	
3. JAGRUTI LIGAM	FS19IF028	
4. ALPESH GAYKER	FS19IF029	

ACKNOWLEDGEMENT

The project is a huge team effort. My team and I extend our deepest gratitude and thanks to the following people to have helped us to achieve our work.

I would like to thank Dr. Seema yardi for guiding us and helping in time of need.

My team and I extend thanks to other faculties of our college whom we have approached for the academic help with regards to our project. We also thank our Head of Department Dr. Seema yardi for their support and guidance.

Thanks to all our teachers in the past who have inculcated in us values and work habit, that have allowed us to create the level of success that we have achieved today in our team work.

RUTUJA UGALE (FS19IF011)

TRUPTI KADAM (FS19IF026)

JAGRUTI LIGAM (FS19IF028)

ALPESH GAYKER (FS19IF029)

ABSTARACT

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. 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 basic Machine Learning packages like TensorFlow, Keras, OpenCV and Scikit-Learn. 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.

Table of Contents

CHAPTER 1

INTRODUCTION

1.Introduction

1.1 Basic idea:

During pandemic COVID-19, WHO has made wearing masks compulsory to protect against this deadly virus. In this project we will develop a machine learning project – Real-time Face Mask Detector with Python. The main aim of this project to overcome the rapidly spread this virus in world. And detecting whether person is wearing a proper mask or not.

We will build a real-time system to detect whether the person on the webcam is wearing a mask or not. We will train the face mask detector model using Keras and OpenCV.

we need to break our project into two distinct phases :

1. Training: Here we'll focus on loading our face mask detection dataset from disk, training a model (using Keras/TensorFlow) on this dataset, and then serializing the face mask detector to disk

2. Deployment: Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as

   with_mask

   or

   without_mask

A face mask detection dataset consists of “with mask” and “without mask” images. We will use the dataset to build a COVID-19 face mask detector with computer vision and deep learning using Python, OpenCV, and TensorFlow/Keras.

This dataset consists of 2000 images belonging to two classes:

with_mask

: 1000 images

without_mask

: 1000 images

1.2 Methodology and Future Scope:

To methodology implemented For problem solving based on artificial intelligent using deep learning and machine learning system model

Although numerous researchers have committed efforts in designing efficient algorithms for face detection and recognition but there exists an essential difference between ‘detection of the face under mask’ and ‘detection of mask over face’. As per available literature, very little body of research is attempted to detect mask over face. Thus, our work aims to a develop technique that can accurately detect mask over the face in public areas (such as airports. railway stations, crowded markets, bus stops, etc.) to curtail the spread of Coronavirus and thereby contributing to public healthcare.

We believe we can make this system more advanced in futures .Advances future and user interface will be updated in future.Our system is already user friendly by we will try to make this system more user friendly in future.

1.3 Motivation:

Our goal is to train a custom deep learning model to detect whether a person is or is not wearing a mask.

The main motive of this work to provide machine leaning and deep learning system model for various industry,airports,railway station,crowded markets,bus stops etc.public places to peoples can wearing mask strictly.

1.4 Scope of work:

In this work, a deep learning-based approach for detecting masks over faces in public places to curtail the community spread of Coronavirus is presented. The proposed technique efficiently handles occlusions in dense situations by making use of an ensemble of single and two-stage detectors at the pre-processing level. The ensemble approach not only helps in achieving high accuracy but also improves detection speed considerably. Furthermore, the application of transfer learning on pre-trained models with extensive experimentation over an unbiased dataset resulted in a highly robust and low-cost system. The identity detection of faces, violating the mask norms further, increases the utility of the system for public benefits.

1.5 Our Contribution:

This system having different module.

Two datasets have been used for experimenting the current method. Dataset 1 consists of 2000 images in which 1000 images with people wearing face masks and the rest 1000 images with people who do not wear face masks.

1.6 Outline of the project:

The methodology implemented is based on MobileNetV2,deep learning model .

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 system is user friendly.

CHAPTER 2

LITERATURE REVIEW

2. Literature Review

The COVID-19 is an unparalleled crisis leading to a huge number of casualties and security problems. To reduce the spread of coronavirus, people often wear masks to protect themselves. This makes face recognition a very difficult task since certain parts of the face are hidden. A primary focus of the researchers during the ongoing coronavirus pandemic is to come up with suggestions to handle this problem through rapid and efficient solutions. This project aims to present a review of various methods and algorithms used for human recognition with a face mask. Different approaches i.e. Haar cascade, Adaboost, VGG-16 CNN Model, etc. are described in this paper. A comparative analysis is made on these methods to conclude which approach is feasible. With the advancement of technology and time more reliable methods for human recognition with a face mask can be implemented in the future. Finally, it includes some of the applications of face detection. This system has various applications at public places, schools, etc. where people need to be detected with the presence of a face mask and recognize them and help society.

ABOUT CORE TECHNOLOGIES

A] Convolutional Neural Networks Convolutional Neural Networks (CNNs) are a form of deep, feed-forward artificial neural network used to analyze visual imagery. These networks' architecture was loosely influenced by biological neurons that interact with one another and produce outputs based on inputs. Although work on CNNs began in the early 1980s, they have only recently gained popularity as a result of recent technological advances and computational capabilities that allow the processing of large quantities of data and the training of sophisticated algorithms in a reasonable amount of time.

B] MobileNetV2

MobileNetV2 is a major advancement over MobileNetV1 in terms of classification, object identification, and semantic segmentation for mobile visual recognition. MobileNetV2 is available as part of the TensorFlow-Slim Image Classification Library, or one can use Colaboratory to get started with it right away [2]. It can also download the notebook and use Jupyter to explore it locally. MobileNetV2 is also available as TF-Hub modules, with pre-trained checkpoints available on github. MobileNetV2 expands on the concepts of MobileNetV1 [1], using depthwise separable convolution as a cost-effective building block. However, V2 incorporates two new architectural features: 1) linear bottlenecks between layers, and 2) shortcut connections between bottlenecks.

C] YOLOv2 model:

YOLO is one of the best models in object recognition, able to recognize objects and process frames at the rate up to 150 FPS for small networks. However, In terms of accuracy mAP, YOLO was not the state of the art model but has fairly good Mean average Precision (mAP) of 63% when trained on PASCAL VOC2007 and PASCAL VOC 2012. However, Fast R-CNN which was the state of the art at that time has an mAP of 71%.

D]ResNet50 model:

ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database . The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully.

CHAPTER 3

PLANNING PAHSE

3. Planning phase

3.1 Feasibility study

Objective:

- To enforce the mandate for wearing masks in public places following the Covid-19 pandemic
- To effectively provide a working model for accurate mask detection
- To utilize Image processing approach to identify the presence of masks on face
- To develop a efficient computer vision based system focused on real time automated monitoring of people to detect face mask in public places

Future scope:

- Can be implemented as mobile applications
- Can be develop as API

Advantage:

Public places like Bus stand , Air ports and railway stations

Offices and Education institutes

- Benefits

  Cost effective

  Life saving

  Curb Covid-19 pandemic

3.2 project planning

overall project requirements consists of :

A. TensorFlow:

TensorFlow, an interface for expressing machine learning algorithms, is utilized for implementing ML systems into fabrication over a bunch of areas of computer science, including sentiment analysis, voice recognition, geographic information extraction, computer vision, text summarization, information retrieval, computational drug discovery and flaw detection to pursue research [18]. In the proposed model, the whole Sequential CNN architecture (consists of several layers) uses TensorFlow at backend. It is also used to reshape the data (image) in the data processing.

B. Keras:

Keras gives fundamental reflections and building units for creation and transportation of ML arrangements with high iteration velocity. It takes full advantage of the scalability and cross-platform capabilities of TensorFlow. The core data structures of Keras are layers and models [19]. All the layers used in the CNN model are implemented using Keras. Along with the conversion of the class vector to the binary class matrix in data processing, it helps to compile the overall model.

C. OpenCV:

OpenCV (Open Source Computer Vision Library), an open-source computer vision and ML software library, is utilized to differentiate and recognize faces, recognize objects, group movements in recordings, trace progressive modules, follow eye gesture, track camera actions, expel red eyes from pictures taken utilizing flash, find comparative pictures from an image database, perceive landscape and set up markers to overlay it with increased reality and so forth [20]. The proposed method makes use of these features of OpenCV in resizing and color conversion of data images.

D. 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.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for Numerical Python. In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

E. Imutils:

A series of convenience functions to make basic image processing operations such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and Python.

F. Matplotlib :

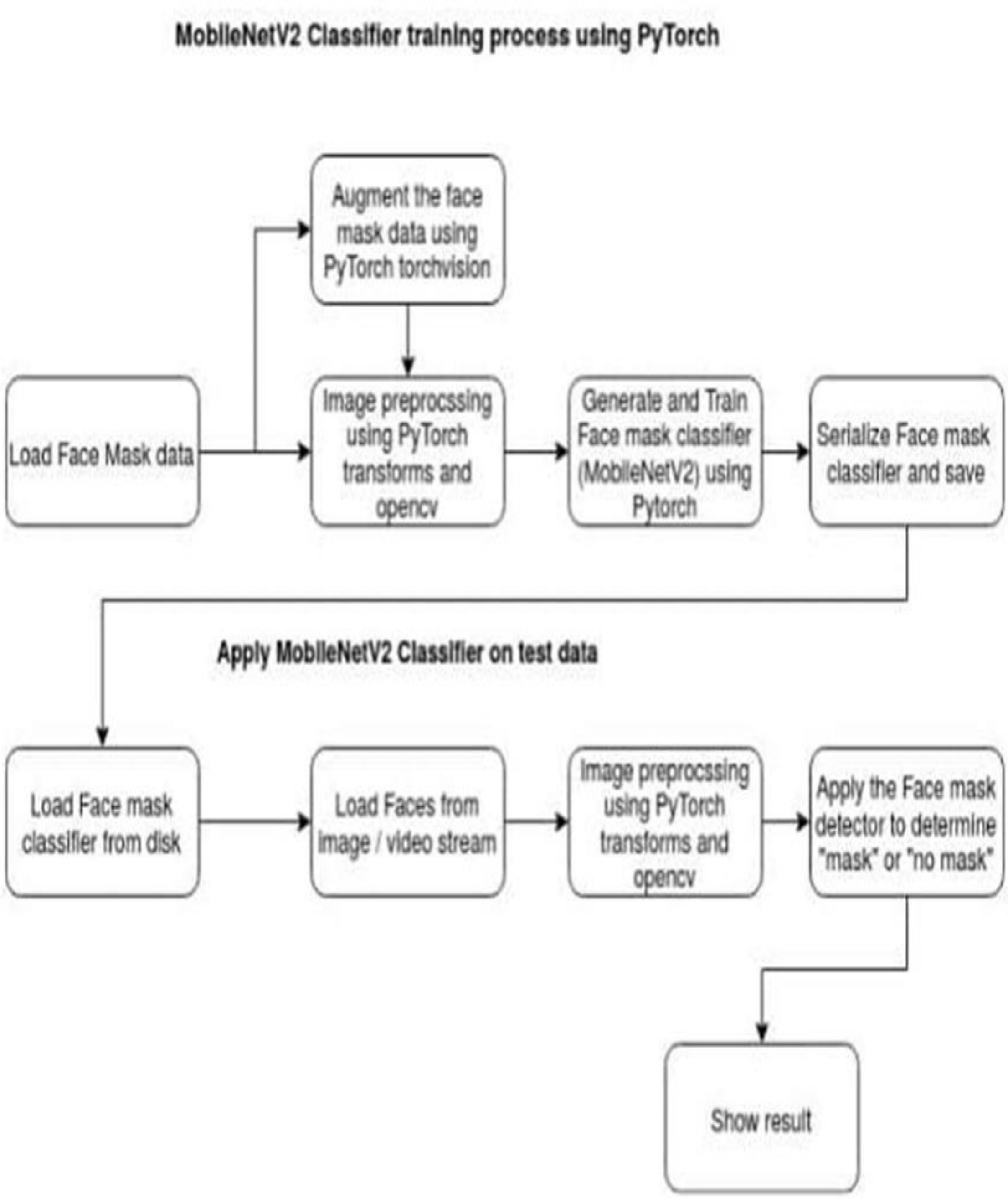
Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

G.Scripy:

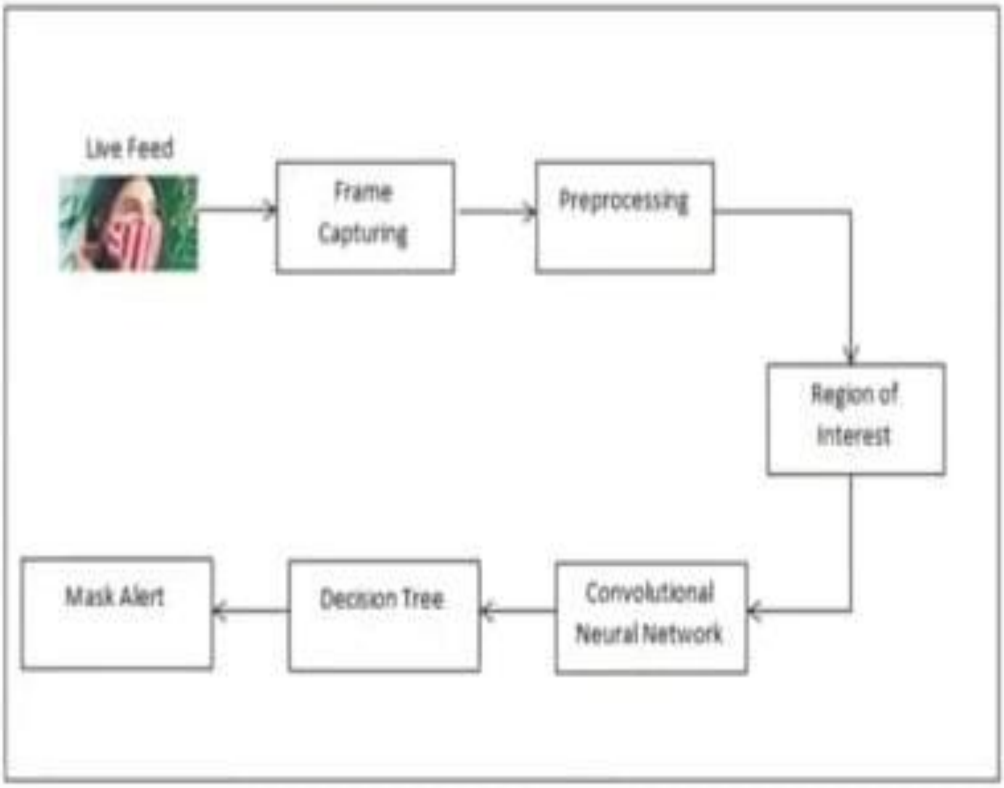
In the script mode, the Python program is written in a file. Python interpreter reads the file and then executes it and provides the desired result. The program is compiled in the command prompt, The interactive mode is more suitable for writing very short programs.

3.3 Flow of the system

3.3.1 flowchart



System Architecture



3.4 System Design

3.4.1 Software Requirements

The Software used far the development of the project:

Operating system: Windows

Programming Language: Python

IDE: spyder python software

Emulators; AV'D

3.4.2 Hardware Requirements

The Hardware used for the development of the project;

Processor: Intel i3 s° Gen

RAM:8GB, minimum 4 GB

Monitor:15 inches color

Keyboard: Optical

Mouse: Optical

3.4.3 Technologies used in this Project

- Artificial Intelligence
- Machine Learning
- Deep Learning
- OpenCV
- Python

3.4.4 Required Skillset to Build the Project:

One must be capable of writing programs in Python and work with microprocessors and sensors. They should be well-versed in areas such as Artificial Intelligence, Machine Learning, Deep Learning, and OpenCV.

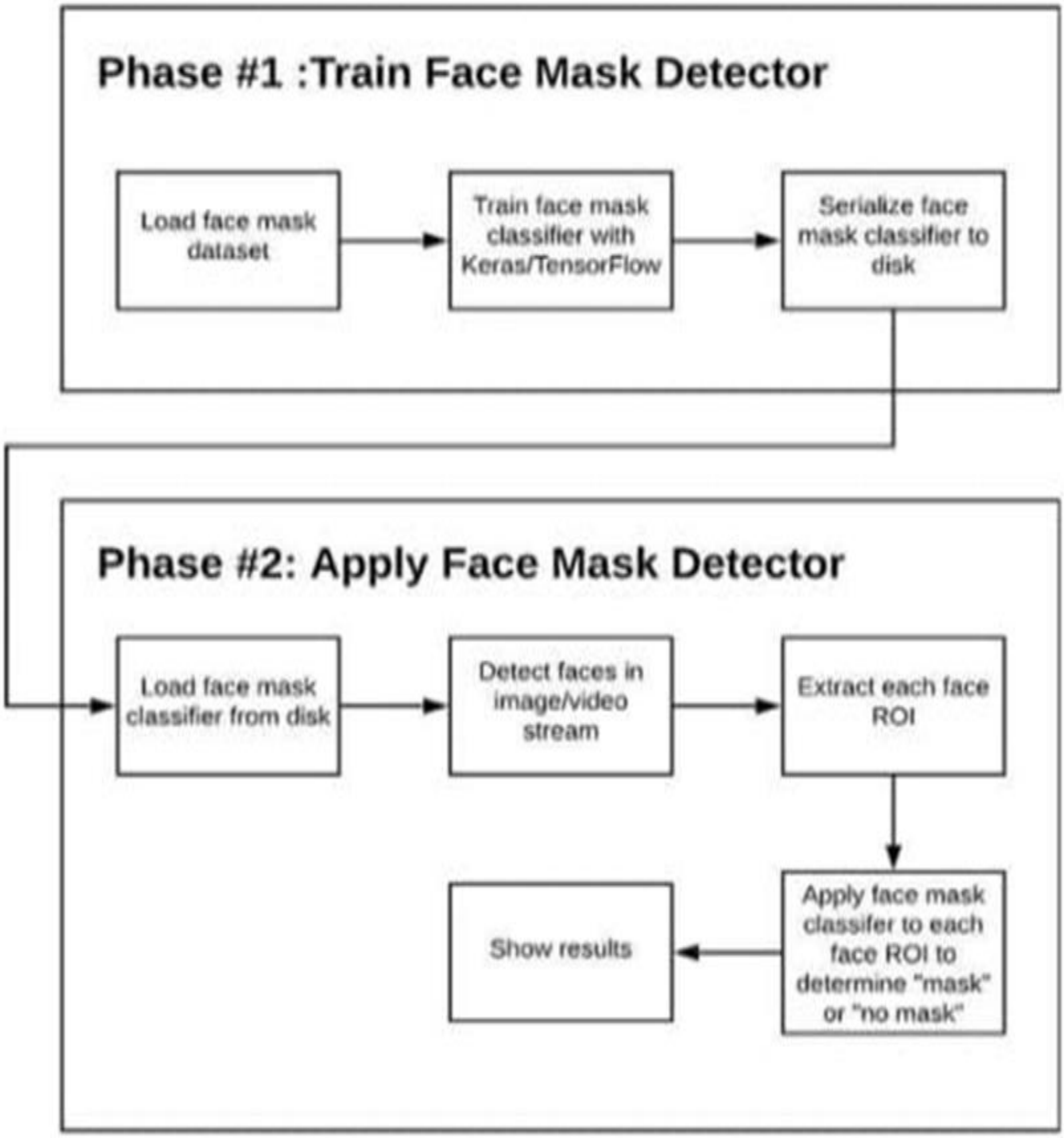
3.5 Database structure

Step-by-Step Implementation

The Face Mask Detection model is created in four step

- Specifying the model : (layer node, the activation function is applied to those nodes)
- Compile : (loss function, Optimizer)
- Fit : (make model learn)
- Predict : (use the model to predict)

To train a customized face mask detector, we must divide our project into two unique stages, each with its own set of sub-steps (as seen in Figure below):



Two Phases COVID-19 Face Detector

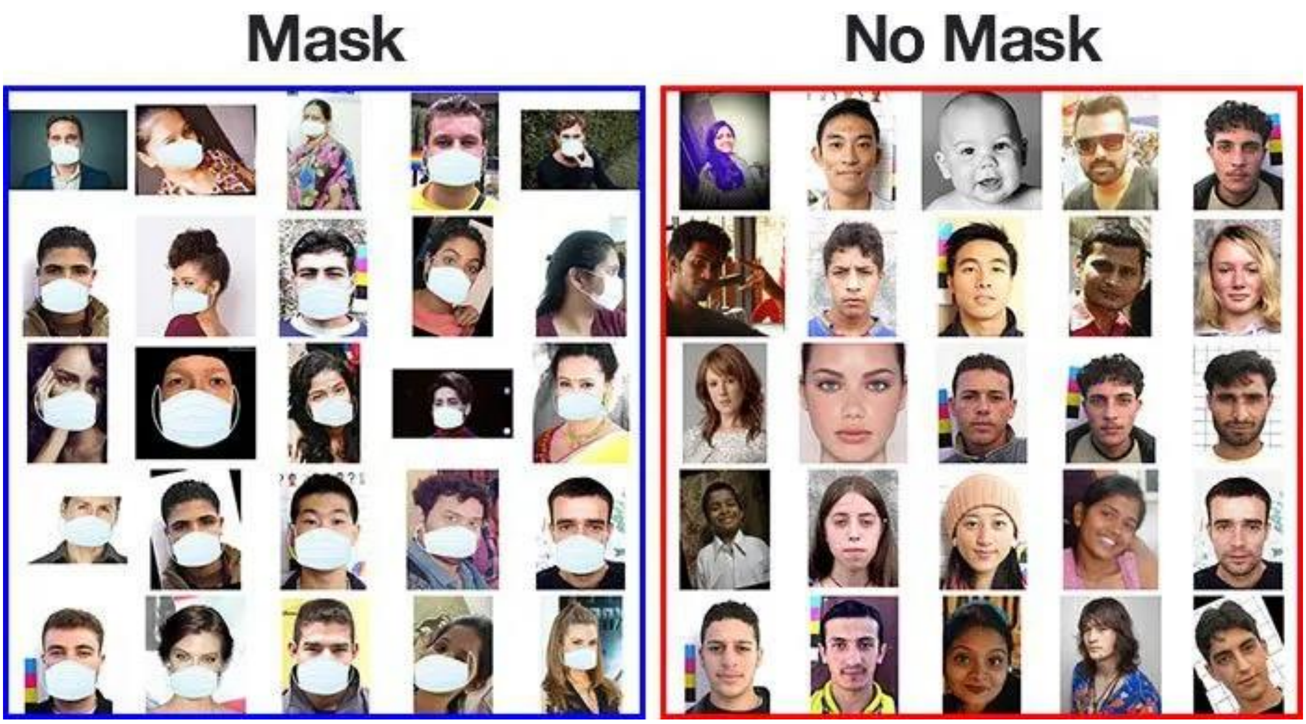
- Training: Here we'll focus on loading our face mask detection dataset from disk, training a model (using Keras/Tensor Flow) on this dataset, and then serializing the face mask detector to disk.

- Deployment: Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as with mask or without mask.

Our Database

With Mask
https://drive.google.com/folderview?id=1WLBSCMDDp7uLrDVQ20Y7IDyb25JcaTne

Without Mask
https://drive.google.com/folderview?id=10baXLReTrVIxRrXT2rHsVUz3ve6ygKL9



3.6 Model Description:

Face Recognition

Face Mask Detection

1) Face Recognition:

Face detection is a sort of computer vision technology that can recognize people's faces in digital photographs.

- Facial recognition entails recognizing the face in a picture as belonging to person X rather than person Y. It is frequently used for biometric applications, like unlocking a smartphone.
- Facial analysis attempts to learn something about people based on their facial features, such as their age, gender, or the emotion they are displaying.
- Facial tracking technique is commonly used in video analysis and attempts to follow a face and its features (eyes, nose, and lips) from frame to frame.

2) Face Mask Detection:

Data At Source: OpenCV was used to increase the size of the images. At the time, the images were titled "cover" and "no veil." The images available were of various sizes and goals and were most likely extracted from various sources or from machines (cameras) of various goals.

Data Processing: Ventures, as indicated below, were applied to all the raw data images to convert them into clean forms that could be handled by a neural organization AI model.

- Resizing the information picture (256 x 256).
- Applying the shading sifting (RGB) over the channels (Our model MobileNetV2 underpins 2D 3 channel picture).
- Scaling/Normalizing pictures utilizing the standard mean of PyTorch work in loads.
- Center trimming the picture with the pixel estimation of 224x224x3.
- Finally Converting them into tensors (Similar to Numpy exhibit).
- Training and,
- Deployment.

CHAPTER 4

TESTING

4. Testing

4.1 Test cases for trained dataset

sr no	test case ID	features	test case description	test data input	excepted result	actual result	status
1	tc 001	camera features	verify that camera is open properly	_	camera should be enable	camera is enable successfully	pass
2	tc002	camera functionality	verify camera is able to detect images	any images	camera should be detect image	camera is detect image successfully	pass
3	tc003	image detect functionality	verify camera is detect without mask image	without face mask image	camera should be detect without mask image	camera is able to detect without mask image successfully	pass
4	tc004	image feature	verify camera should detect face mask	with face mask image	camera should be detect with mask image	camera is able to detect with mask image successfully	pass
5	tc005	image feature	verify if the camera should detect without face mask	face mask image	camera should not be able to detect without face mask image	camera is unable to detect without face mask image	pass
6	tc006	image feature	verify if the camera should detect with face mask image	no face mask image	camera should not able to detect with face mask image	camera is unable to detect with face mask image	pass
7	tc007	camera functionality	verify that the recognized with face mask	with mask image	mask should recognized	mask is message display on screen	pass
8	tc008	camera feature	verify that the recognized without face mask	without face mask	no mask should recongnize	no mask is message display on screen	pass
9	tc009	dataset fetures	verify that the dataset images are loaded	1000 with mask and 1000 without mask images	trained dataset images should be loaded	trained dataset images loaded successfully	pass
10	Tc010	dataset time loading feature	verify that the dataset images are loaded within 5 minutes	1000 with mask and 1000 without mask images	trained dataset images should be loaded.	trained dataset images loaded successfully within 5 minutes	pass

4.2 Test cases for real time camera object:

sir no	Test case ID	Features	test case description	Input data	Expected Result	Actual Results	Status
1	TC001	camera feature	Verify that the camera is enable or disable	-----	camera should be enable	camera is able to detect object	pass
2	TC001	cameras functionality	verity the camera is able to detect object	any object	cameras should be detect the object	cameras able to detect object successfully	pass
3	TC003	cameras tures	verify that enable the front end and back end cameras	any object	cameras should be detect front object and also back end	cameras able to detect front end object and also back end	pass
4	TC004	object feature -re	verify that camera should detect with mask object	without wearing mask object	cameras shouldn't detect with mask object	cameras unable to detect with mask object	pass
5	TC005	object featu -re	verity that cameras should detect without mask object	with wearing mask object	cameras shouldn't detect without wearing object	cameras unable to detect without wearing object	pass
6	TC006	object detect	verify that program detect red square in without mask or not	without wearing object	cameras should detect the red square in object	cameras successfully detect red square in object	pass
7	TC007	object detect	verify that program detect green square in with mask object or not	with mask object	cameras should detect the green square in object	cameras successfully detect green square in object	pass
8	TC008	object detect	verify that program shown squ detector or not	any object	cameras should be able to shown square detector	cameras able to shown square detector	pass
9	TC009	object detector	verify that the limit of square detector	two object	cameras should be able to detect only two object	cameras able to detect two objects	pass
10	TC010	camera features	verify that the accuracy of the detector is more than 75%	any real time object	cameras should able to accura- cy of detector is more than75%	camera able to accuracy of detector is more than 75%	pass
11	TC011	detector feature	verify that camera detect proper object with message	with mask object	cameras should able to detect and display" with mask " message on screen	cameras detect object and display"with mask" message on screen	pass
12	TC012	detector feature	verify that camera detect prope object with message	without mask wearing object	cameras should able to detect and display "without mask" message on screen	cameras detect object and display "without mask" message on screen.	pass
13	TC013	object detector feature	verify that the detector detect the proper object or not	wearing mask object	detector should able to detect with mask object	detector enable to detect with mask object	pass
14	TC014	object detector feature	verify that the detector detect the proper object or not	wearing without mask obje	detector should able to detect without mask object	detector enable to detect without mask object	pass
15	TC015	cameras features	verify that the detector detect the clear images or not	any image	detector should be detect clear image	detector able to detect clear image	pass
16	TC016	cameras features	verify that the cameras detect multiple object or not	multiple objects	cameras should be able to detect multiple objects	Detector able to detect multiple object successfully	pass
17	TC017	cameras features	verify that the length of object	any object	cameras should be able to detect 5cm length of object.	cameras able to detect 5 cm length of object	pass
18	TC018	detector features	verify that the flash settings is enable or not	....	detector should not have flash settings	detector unable a flash settings	pass
19	TC019	detector icon features	verify that the square icon is enable or not	.................,.	detector should have square icon on screen	detector able to display square icon on screen.	pass
20	TC020	detector feature	verify that the accuracy % display on screen	with mask object	detector should detected99.9% detect with mask object	detector able to detect 99.9% with mask object.	pass
21	TC021	detector feature	verify that the accuracy % display on screen	without wearing mask object	detector should be detected 99.9%without mask object	detector able to detect 99.9% without mask object	pass
22	TC022	camera feature	verify that the camera is stopped within time.	....	camera is stopped after loop is exit	camera is unable after loop is exit	pass

Conclusion

In this paper, we briefly explained the motivation of the work at first. Then, we illustrated the learning and performance task of the model. Using basic ML tools and simplified techniques the method has achieved reasonably high accuracy. It can be used for a variety of applications. Wearing a mask may be obligatory in the near future, considering the Covid-19 crisis. Many public service providers will ask the customers to wear masks correctly to avail of their services. The deployed model will contribute immensely to the public health care system. In future it can be extended to detect if a person is wearing the mask properly or not. The model can be further improved to detect if the mask is virus prone or not i.e. the type of the mask is surgical, N95 or not.

References

1.Liang, L., Ai, H.: Summary on Face Detection Researches. Journal of Computers 25(5), 449–459 (2004)

Google Scholar

2.Liu, M.: Real-time Face Tracking Method of Color Images. Journal of Computers 21(6), 527–532 (2003)

Google Scholar

3.https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/

4.Zhao, M.: Study on Coding Algorithm of Wavelet-based Color Face Images. Master Thesis. Computer Academy of Sciences of Sichuan Normal University, Sichuan (2005)

Google Scholar

5.Liang, L., Ai, H.: Face Detections Based on Multi-Template Matching. Journal of China's Image Graphics 44(10), 623–630 (2004)

Google Scholar

6.https://github.com/atuldev19/Face_Mask_Project

7.Chen, M.: Studies on the Face Image Detection and Classification System. PhD Thesis. Computer Science and Engineering Department of Shanghai Jiaotong University, Shanghai (2003)

Google Scholar

8.Lu, C.: Study on Several Problems of Automatic Face Recognition and the System Implementation. PhD Thesis. Computer Academy of Sciences of Tsinghua University, Beijing (1998)

Google Scholar

9.Su, G.: Summary on Face Recognition Technology. Journal of China's Image Graphics 5(11), 220–238 (2000)

Google Scholar

10.https://youtu.be/LfGDm0HgDp8

11.Li, J.: Research Progress of New Face Recognition Technology. Journal of Computers 31(10), 293–295 (2004)

Google Scholar

12.Zhang, J., He, T.: Face Recognition Method Based on Extractions of Facial Geometric Feature Points. Infrared and Laser Engineering 28(4), 40–43 (1999)

Google Scholar

Application code:

Detect mask

```
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from os.path import dirname, join
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")
```

```python
			# 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"deploy.protext"
weightsPath = r"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("Starting the CAMERA...")
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

```
# 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_Project-master\Face_Mask_Project-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)
```

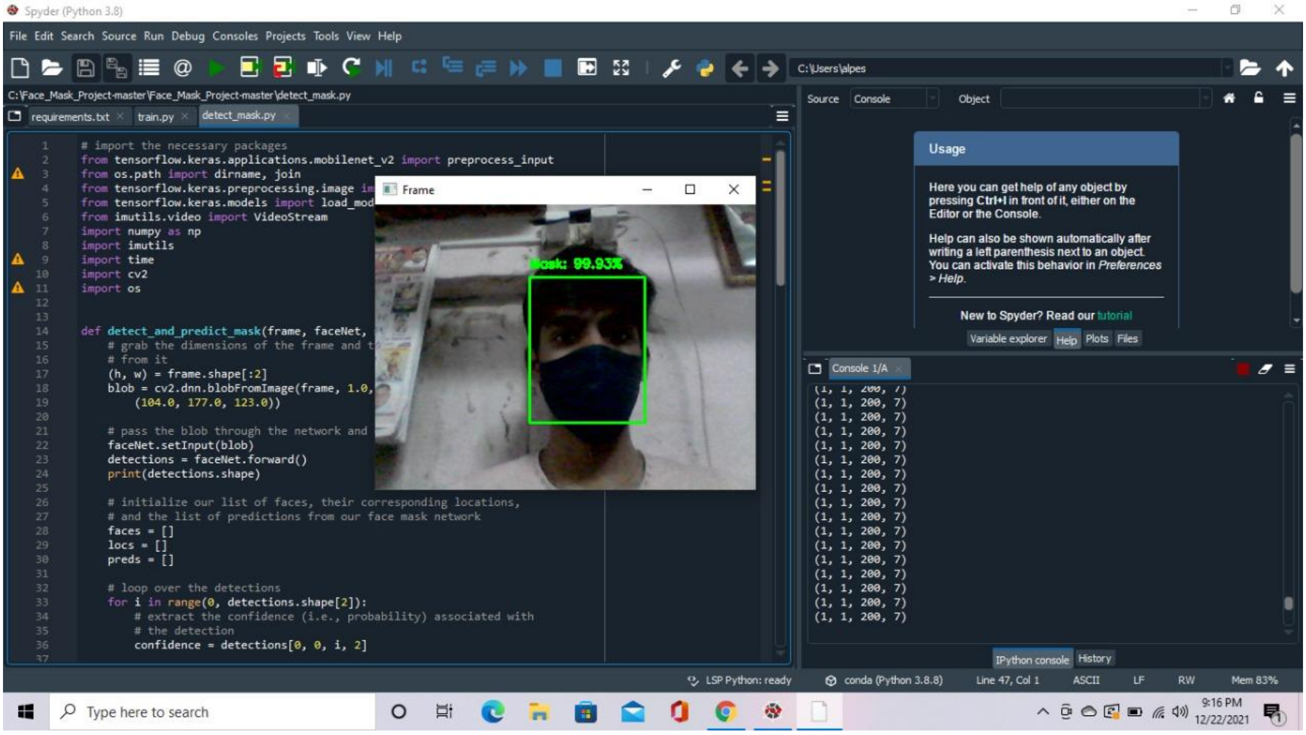
```
		labels.append(category)

# 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
```

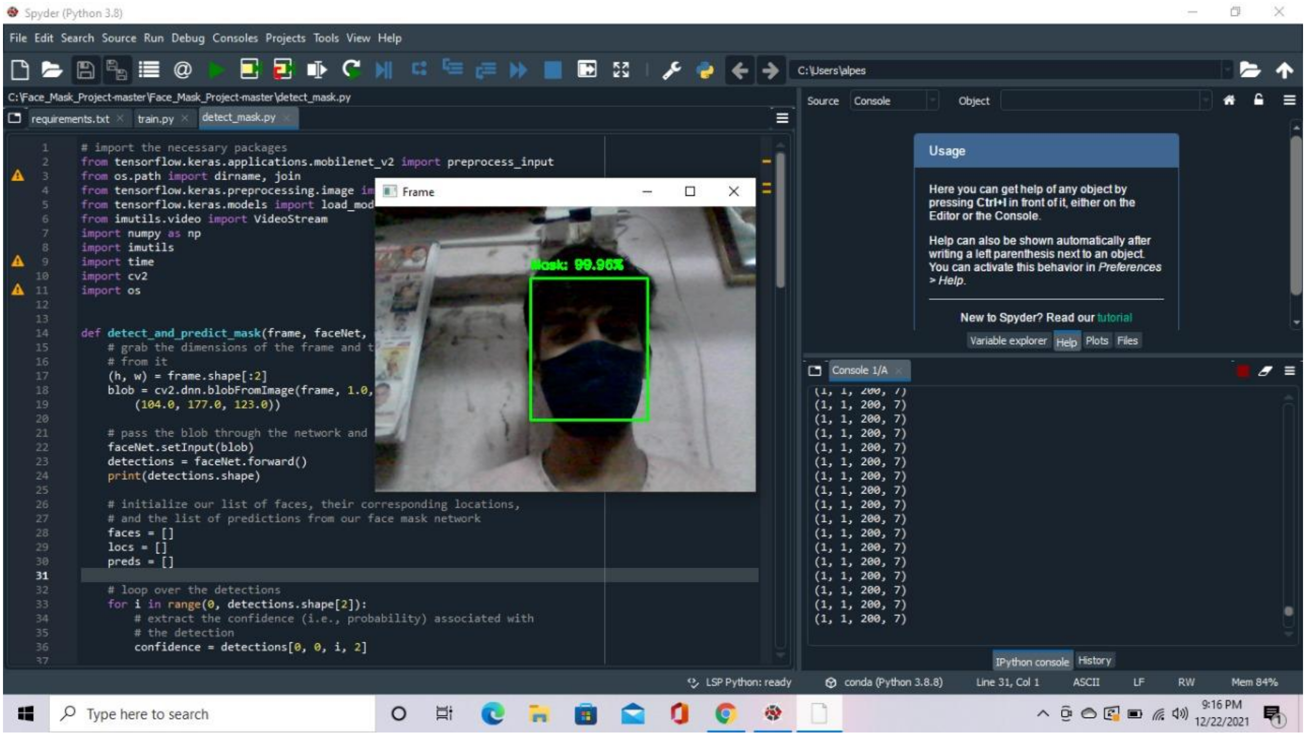
```
baseModel = MobileNetV2(weights="imagenet", include_top=False,
	input_tensor=Input(shape=(224, 224, 3)))

# 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("Compilation of the MODEL is going on...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
```

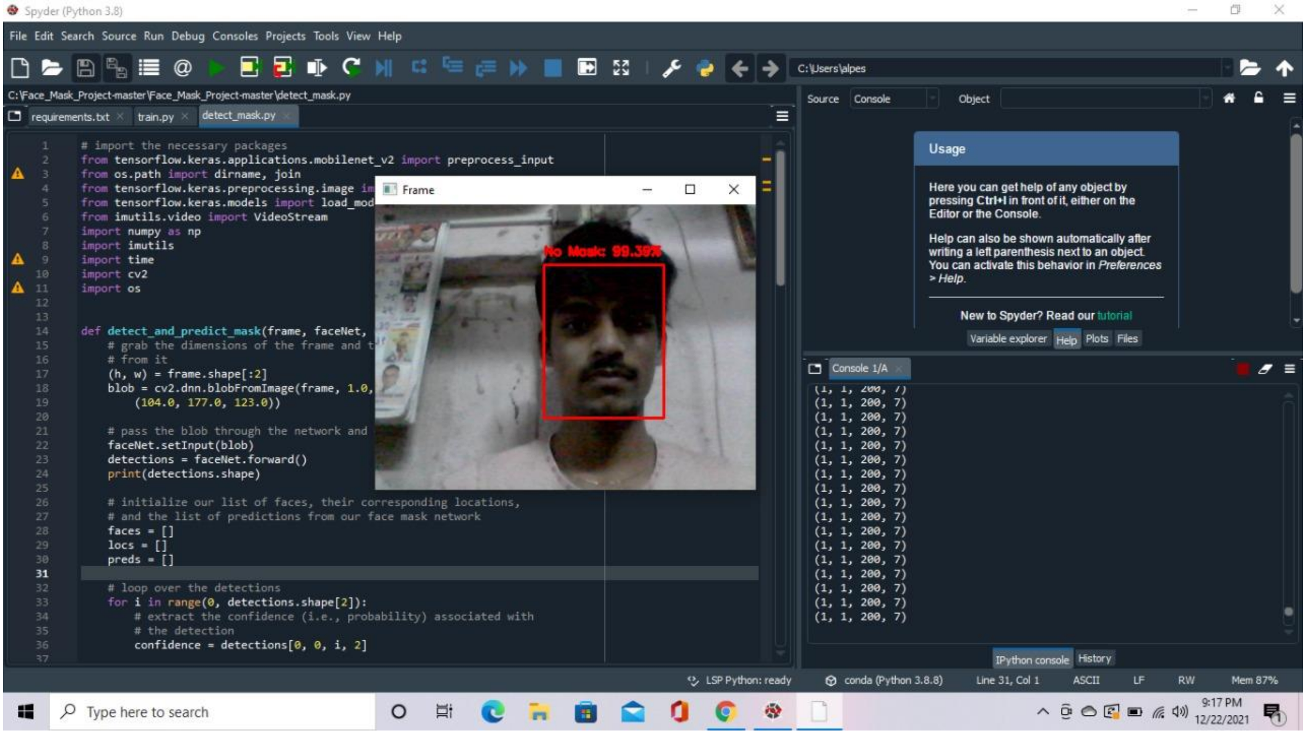
```
        metrics=["accuracy"])

# train the head of the network
print("Training Head Started...")
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("Network evaluation...")
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("saving mask model...")
```

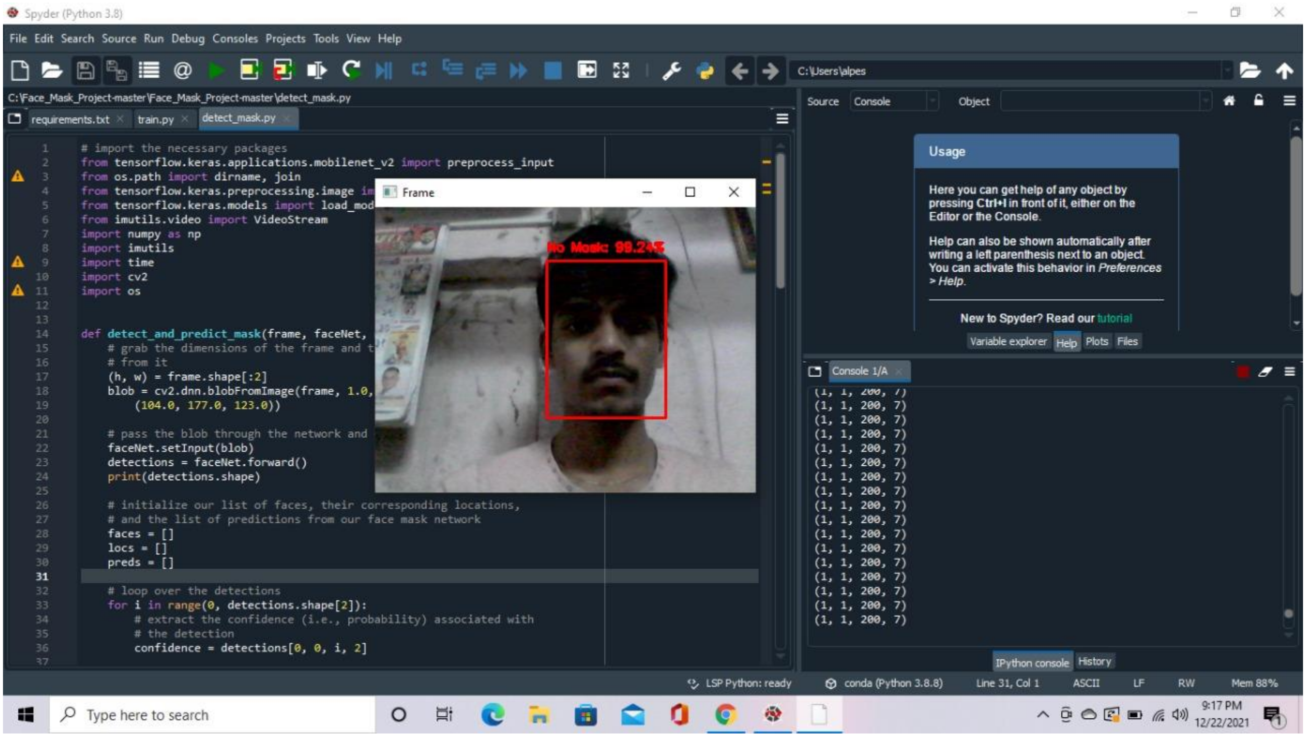
```
model.save("mask_detector.model", save_format="h5")


# 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")
```

OUTPUTS






Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Face_Mask_Project-master\Face_Mask_Project-master\detect_mask.py
requirements.txt
train.py
detect_mask.py
Frame
No Mask: 99.39%
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Console 1/A
(1, 1, 200, 7)
IPython console
History
LSP Python: ready
conda (Python 3.8.8)
Line 31, Col 1
ASCII
LF
RW
Mem 87%
Type here to search
9:17 PM
12/22/2021


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Face_Mask_Project-master\Face_Mask_Project-master\detect_mask.py
requirements.txt
train.py
detect_mask.py
Frame
No Mask: 99.24%
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Console 1/A
(1, 1, 200, 7)
IPython console
History
LSP Python: ready
conda (Python 3.8.8)
Line 31, Col 1
ASCII
LF
RW
Mem 88%
Type here to search
9:17 PM
12/22/2021