FACE MASK DETECTION USING CONVOLUTION NEURAL NETWORK

A PROJECT REPORT
Submitted by

PRADUMNA	2019504043
ENESH NAREN	2019504515
YUVAN RAJU	2019504608

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS & COMMUNICATION ENGINEERING

MADRAS INSTITUTE OF TECHNOLOGY
ANNA UNIVERSITY, CHENNAI 600 044

DECEMBER 2021

ACKNOWLEDGEMENT

We consider it as our privilege and our primary duty to express our gratitude and respect to all those who guided and inspired us in the successful completion of the project.

We owe solemn gratitude to Dr. T. THYAGARAJAN, Dean, Madras Institute of Technology, for having given consent to carry out the project work at MIT Campus, Anna University.

We wish to express our sincere appreciation and gratitude to Dr. M. GANESH MADHAN, Professor and Head of the Department of Electronics Engineering, who has encouraged and motivated us in our endeavors.

We are extremely grateful to our project guides Ms. P. PAVITHRA,
Dr. S. EZHILARASI for their timely and thoughtful guidance and encouragement for the completion of the project.

PRADUMNA 2019504043

ENESH NAREN 2019504515

YUVAN RAJU 2019504608

INDEX

List of Figures

Contents **Page No.**

List of Abbreviations

CNN - Convolutional Neural Network
OpenCV - Open Source Computer Vision

ABSTRACT

The spread of COVID-19 has affected everyone in the world. This virus can be affected from human to human through the droplets and airborne. According to the instructions from WHO, to reduce the spread of COVID-19, every person needs to wear a face mask, do social distancing, evade the crowd area and also always maintain the immune system. Since there is no specific treatment, wearing masks has become an effective method to prevent the transmission of COVID-19 and is required in most public areas, which has also led to a growing demand for automatic real-time mask detection services to replace manual reminding.

However, few studies on face mask detection are being conducted. It is urgent to improve the performance of mask detectors and implement a real-time mask detection.
Computer vision is an interdisciplinary scientific field that involves how computers gain advanced understanding from digital images or videos.
Traditional computer vision tasks include image processing, image classification, object detection, and image recognition.
Object detection can detect instances of visual objects of a certain class in the images, which is a proper solution for the problem mentioned above.
Consequently, mask detection has become a vital computer vision task to help the global society.

To overcome this situation, a robust face mask detection needs to be developed.
In order to detect a face mask, the object detection algorithm can be implemented.
Deploying our face mask detector to embedded devices could reduce the cost of manufacturing such face mask detection systems, hence this architecture was chosen. In this proposed work, our aim is to train a custom deep learning model to detect whether a person *is* or *is not* wearing a mask.

CHAPTER 1

1.1 INTRODUCTION

Since the end of 2019, infectious coronavirus disease (COVID-19) has been reported for the first time in Wuhan, and it has become a public damage fitness issue in China and even worldwide. This pandemic has devastating effects on societies and economies around the world causing a global health crisis. It is an emerging respiratory infectious disease caused by Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2). All over the world, especially in the third wave, COVID-19 has been a significant healthcare challenge. Many shutdowns in different industries have been caused by this pandemic. In addition, many sectors such as maintenance projects and infrastructure construction have not been suspended owing to their significant effect on people's routine life.
Before coronavirus, some people put masks to protect themselves from air pollution, while other people put face masks to hide their faces and their emotions from others. Protection against coronavirus is a mandatory counter measure, according to the WHO. Indeed, wearing a mask is an effective method of blocking 80 of all respiratory infections. Also, the WHO recommends

practicing physical distancing to mitigate the spread of the virus. All over the world, governments are struggling against this type of virus. Many organizations enforce face mask rules for personal protection. Checking manually if individuals entering an organization are wearing masks is cumbersome and possibly conflicting. In this context, we proposed a deep learning-based CNN model, to prevent human-to-human transmissions of the SARS-CoV-2 and detect faces with or without masks. The datasets we used are of the 12k images of people wearing masks and not wearing a mask. The following model achieved a testing accuracy of 93% and validation accuracy of 99%.
We train a COVID-19 face mask detector with OpenCV, Keras/TensorFlow, and Deep Learning.

1.2 OBJECTIVE

- ➢ In this proposed work, our aim is to develop a Convolution Neural Network model to find out whether a person is wearing a Face mask or not, and implement in real-time using Web Cam/Live Streaming using application of computer vision, detecting face masks with OpenCV and Keras/TensorFlow.

- ➢ We deployed a practical and easy to implement model, the COVID-19 mask detector that could potentially be used to help ensure the safety of people.

- ➢ We developed a two-phase COVID-19 face mask detector, detailing how our computer vision/deep learning pipeline will be implemented. We use the 12k dataset to train our custom face mask detector.

- ➢ Implementation of a Python script to train a face mask detector on the dataset using Keras and TensorFlow is shown in this report.

- ➢ We'll use this Python script to train a face mask detector and reviewed the results. Given the trained COVID-19 face mask

detector, we'll proceed to implement Python scripts used to:

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

➔ In order to train a custom face mask detector, we divided the model into two distinct phases, each with its own respective

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

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.

➢ We reviewed each of these phases and associated subsets in detail Our goal is to train a custom deep learning model to detect whether a person is or is not wearing a mask.

➢ Deploying our face mask detector to embedded devices could reduce the cost of manufacturing such face mask detection systems, hence why we choose to use this architecture

The technologies used in this project:-

Artificial Intelligence

Deep Learning

Python

Tensorflow

Keras

CNN

CHAPTER 2

LITERATURE REVIEW AND THEORETICAL BACKGROUND

Before the time of writing this paper, many have proposed and implemented various models to implement the face mask detection. We did a literature review so that we can get some ideas about the existing models and their advantages and disadvantages. Some ideas and papers we came across were:- Real-Time Face Mask Detection Method Based on YOLOv3 - Xinbei Jiang, Tianhan Gao, Zichen Zhu and Yukang Zhao.
They proposed the Properly Wearing Masked Face Detection Dataset (PWMFD), which included 9205 images of mask wearing samples with three categories. Moreover, we proposed Squeeze and Excitation (SE)-YOLOv3, a mask detector with relatively balanced effectiveness and efficiency. Another paper was A Novel Approach to Detect Face Mask using CNN- Md. Shahriar Islam; Eimdadul Haque Moon; Md. Ashikujjaman Shaikat; Mohammad Jahangir Alam. This research paper has proposed a very fast image pre-processing with the mask in the center over the faces. For this system, features extraction and Convolutional Neural Network are used for classification and detection of a masked person. Another one was MaskHunter: real-time object detection of face masks during the COVID-19 pandemic-Zhihao Cao,Mingfeng Shao,Li Xu,Shaomin Mu,Hongchun Qu. A novel object detector namely MaskHunter is proposed in this study for the real-time mask detection. Specifically, the authors propose novel effective structures of backbone, neck and prediction head based on YOLOv4 series, which achieves the state-of-the-art performance and a novel improved Mosaic data augmentation method.

CHAPTER 3

SOFTWARE REVIEW :

- We used Anaconda platform's Spyder in order to train the model and running the python scripts.

3.1 BLOCK DIAGRAM:
Two-phase COVID-19 face mask detector

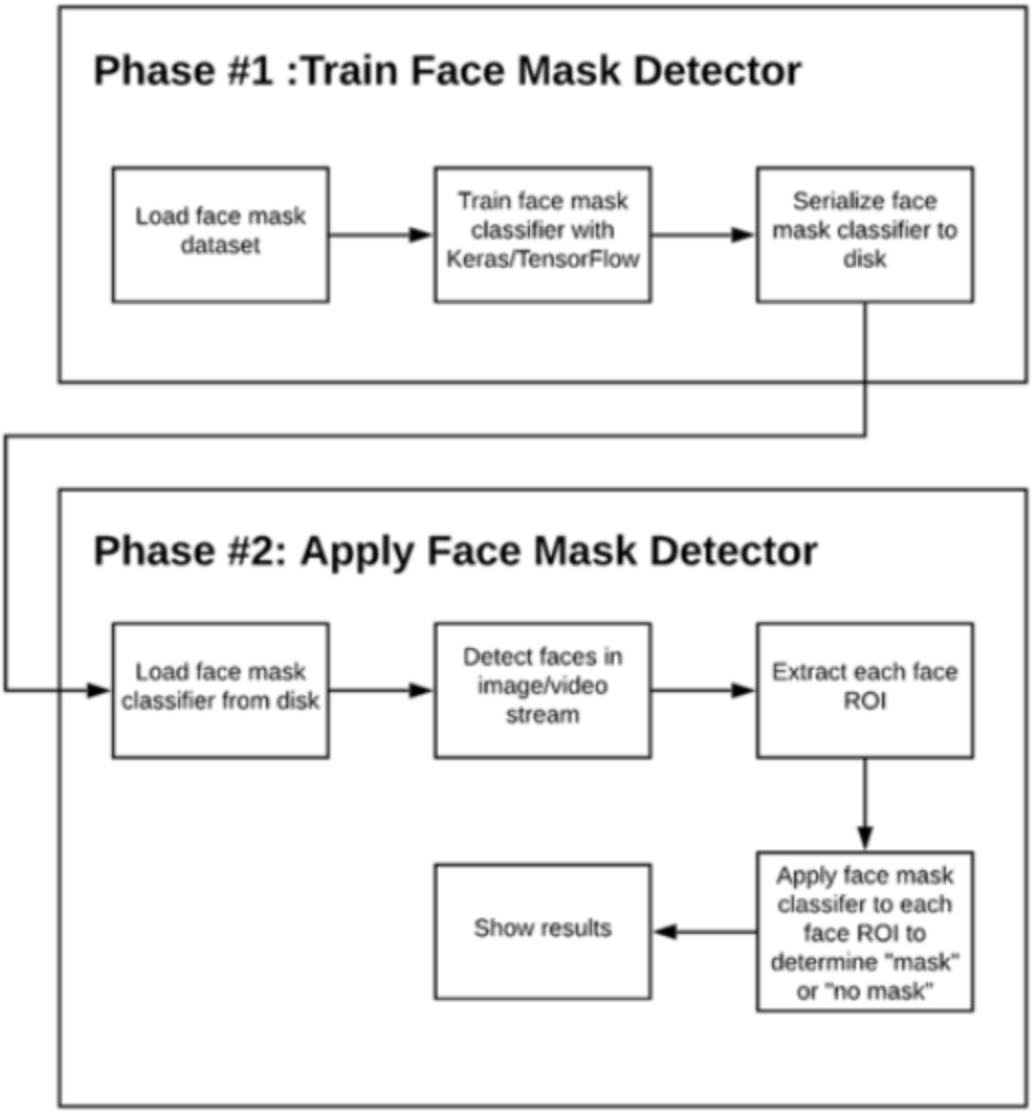

Fig. 1. Block Diagram of two- phase Covid-19 face mask detector

1. **Phase 1**: 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. **Phase 2**: 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.

3.2 The Dataset

The dataset consisted of 12000 images, 6000 face images with masks and 6000 without masks. The original dataset is prepared by CelebFace dataset created by Jessica Li (https://www.kaggle.com/jessicali9530)

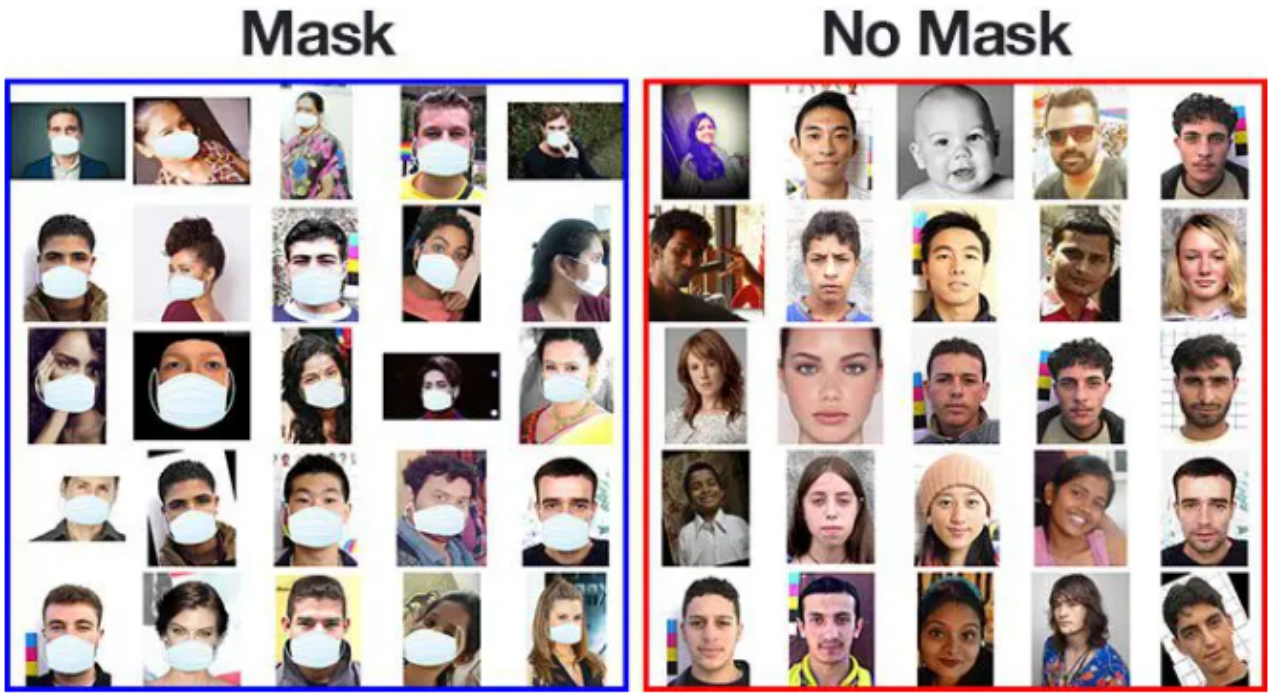

Fig. 2. Dataset for face images with mask and without mask

2.3 Data Preprocessing

tensorflow.keras imports allow for:

- Building a new fully-connected (FC) head
- Pre-processing
- Loading image data

Scikit-Learn (sklearn) is used for binarizing class labels, segmenting our dataset, and printing a classification report.

Imutils paths implementation will help us to find and list images in our dataset.
Matplotlib program is used for plotting the training curves.

```
import cv2,os

data_path='E:/MIT Project/Face Mask Dataset/Train'
categories=os.listdir(data_path)
labels=[i for i in range(len(categories))]

label_dict=dict(zip(categories,labels)) #empty dictionary

print(label_dict)
print(categories)
print(labels)
```

- --**dataset**: The path to the input dataset of faces and and faces with masks

```
img_size=100
data=[]
target=[]


for category in categories:
    folder_path=os.path.join(data_path,category)
    img_names=os.listdir(folder_path)

    for img_name in img_names:
        img_path=os.path.join(folder_path,img_name)
        img=cv2.imread(img_path)

        try:
            gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            #Coverting the image into gray scale
            resized=cv2.resize(gray,(img_size,img_size))
            #resizing the gray scale into 50x50, since we need a fixed common size for all the images in the dataset
            data.append(resized)
            target.append(label_dict[category])
            #appending the image and the label(categorized) into the list (dataset)

        except Exception as e:
            print('Exception:',e)
            #if any exception rasied, the exception will be printed here. And pass to the next image
```

In this block, the following tasks are performed :

Grabbing all of the imagePaths in the dataset & Initializing data and labels lists.

Looping over the imagePaths and loading + pre-processing images

Pre-processing steps include resizing to *100×100* pixels, conversion to array format, and scaling the pixel intensities in the input image to the range *[-1, 1]* (via the preprocess_input convenience function)

- Appending the pre-processed image and associated label to the data and labels lists, respectively
- Ensuring our training data is in NumPy array format

```
import numpy as np

data=np.array(data)/255.0
data=np.reshape(data,(data.shape[0],img_size,img_size,1))
target=np.array(target)

from keras.utils import np_utils

new_target=np_utils.to_categorical(target)
```

The next step is to encode our labels, partition our dataset, and prepare for data augmentation:

CHAPTER 4

ARCHITECTURE IMPLEMENTATION

4.1 Convolutional Neural Network Architexture

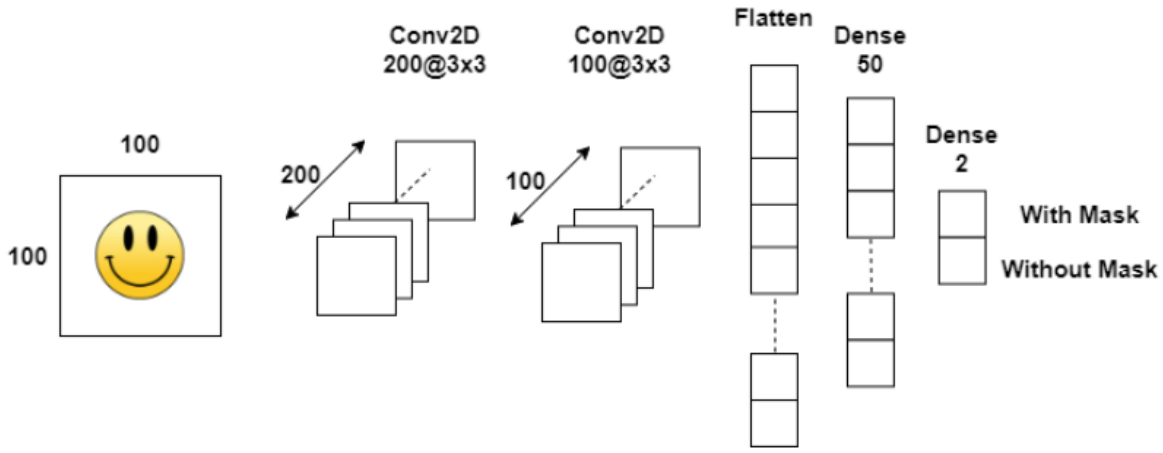

Fig. 3. Convolution Neural Network Architecture

```
import numpy as np

data=np.load('data.npy')
target=np.load('target.npy')

#loading the save numpy arrays in the previous code
```

```
from keras.models import Sequential
from keras.layers import Dense,Activation,Flatten,Dropout
from keras.layers import Conv2D,MaxPooling2D
from keras.callbacks import ModelCheckpoint

model=Sequential()

model.add(Conv2D(200,(3,3),input_shape=data.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
#The first CNN layer followed by Relu and MaxPooling layers

model.add(Conv2D(100,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
#The second convolution layer followed by Relu and MaxPooling layers

model.add(Flatten())
model.add(Dropout(0.5))
#Flatten layer to stack the output convolutions from second convolution layer
model.add(Dense(50,activation='relu'))
#Dense layer of 64 neurons
model.add(Dense(2,activation='softmax'))
#The Final layer with two outputs for two categories

model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

The model is compiled with the Adam optimizer, a learning rate decay schedule, and binary cross-entropy. If the training script is built with > 2 classes, it must be ensured to use categorical cross-entropy

```
from sklearn.model_selection import train_test_split

train_data,test_data,train_target,test_target=train_test_split(data,target,test_size=0.1)
```

```
checkpoint = ModelCheckpoint('model-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
history=model.fit(train_data,train_target,epochs=20,callbacks=[checkpoint],validation_split=0.2)
```

serializes our face mask classification model to disk.

```
Epoch 1/20
673/673 [==============================] - ETA: 0s - loss: 0.1740 - accuracy: 0.9253
```

```
673/673 [==============================] - 407s 604ms/step - loss: 0.1740 - accuracy: 0.9253 - val_loss: 0.1055 - val_accuracy: 0.9623
Epoch 2/20
673/673 [==============================] - ETA: 0s - loss: 0.0623 - accuracy: 0.9786  INFO:tensorflow:Assets written to: model-002.model\assets
673/673 [==============================] - 381s 566ms/step - loss: 0.0623 - accuracy: 0.9786 - val_loss: 0.0400 - val_accuracy: 0.9863
Epoch 3/20
673/673 [==============================] - ETA: 0s - loss: 0.0450 - accuracy: 0.9838  INFO:tensorflow:Assets written to: model-003.model\assets
673/673 [==============================] - 378s 561ms/step - loss: 0.0450 - accuracy: 0.9838 - val_loss: 0.0282 - val_accuracy: 0.9902
Epoch 4/20
673/673 [==============================] - ETA: 0s - loss: 0.0363 - accuracy: 0.9872  INFO:tensorflow:Assets written to: model-004.model\assets
673/673 [==============================] - 373s 555ms/step - loss: 0.0363 - accuracy: 0.9872 - val_loss: 0.0275 - val_accuracy: 0.9903
Epoch 5/20
673/673 [==============================] - ETA: 0s - loss: 0.0287 - accuracy: 0.9903  INFO:tensorflow:Assets written to: model-005.model\assets
673/673 [==============================] - 369s 548ms/step - loss: 0.0287 - accuracy: 0.9903 - val_loss: 0.0189 - val_accuracy: 0.9937
Epoch 6/20
673/673 [==============================] - 369s 548ms/step - loss: 0.0193 - accuracy: 0.9928 - val_loss: 0.0191 - val_accuracy: 0.9922
Epoch 7/20
673/673 [==============================] - ETA: 0s - loss: 0.0159 - accuracy: 0.9944  INFO:tensorflow:Assets written to: model-007.model\assets
673/673 [==============================] - 369s 548ms/step - loss: 0.0159 - accuracy: 0.9944 - val_loss: 0.0069 - val_accuracy: 0.9978
Epoch 8/20
673/673 [==============================] - 371s 551ms/step - loss: 0.0110 - accuracy: 0.9959 - val_loss: 0.0074 - val_accuracy: 0.9974
Epoch 9/20
673/673 [==============================] - ETA: 0s - loss: 0.0123 - accuracy: 0.9957  INFO:tensorflow:Assets written to: model-009.model\assets
673/673 [==============================] - 370s 550ms/step - loss: 0.0123 - accuracy: 0.9957 - val_loss: 0.0041 - val_accuracy: 0.9985
Epoch 10/20
673/673 [==============================] - ETA: 0s - loss: 0.0092 - accuracy: 0.9971  INFO:tensorflow:Assets written to: model-010.model\assets
673/673 [==============================] - 372s 552ms/step - loss: 0.0092 - accuracy: 0.9971 - val_loss: 0.0029 - val_accuracy: 0.9989
Epoch 11/20
673/673 [==============================] - ETA: 0s - loss: 0.0047 - accuracy: 0.9986      INFO:tensorflow:Assets written to: model-011.model\assets
673/673 [==============================] - 370s 550ms/step - loss: 0.0047 - accuracy: 0.9986 - val_loss: 0.0014 - val_accuracy: 0.9998
Epoch 12/20
673/673 [==============================] - 369s 548ms/step - loss: 0.0074 - accuracy: 0.9978 - val_loss: 0.0036 - val_accuracy: 0.9989
Epoch 13/20
673/673 [==============================] - 394s 585ms/step - loss: 0.0093 - accuracy: 0.9972 - val_loss: 0.0114 - val_accuracy: 0.9961
```

```
Epoch 14/20
673/673 [==============================] - 405s 602ms/step - loss: 0.0045 - accuracy: 0.9986 - val_loss: 0.0028 - val_accuracy: 0.9991
Epoch 15/20
673/673 [==============================] - 396s 588ms/step - loss: 0.0057 - accuracy: 0.9984 - val_loss: 0.0092 - val_accuracy: 0.9955
Epoch 16/20
673/673 [==============================] - 393s 584ms/step - loss: 0.0062 - accuracy: 0.9979 - val_loss: 0.0024 - val_accuracy: 0.9991
Epoch 17/20
673/673 [==============================] - 405s 601ms/step - loss: 0.0036 - accuracy: 0.9990 - val_loss: 0.0022 - val_accuracy: 0.9991
Epoch 18/20
673/673 [==============================] - ETA: 0s - loss: 0.0052 - accuracy: 0.9983      INFO:tensorflow:Assets written to: model-018.model\assets
673/673 [==============================] - 403s 599ms/step - loss: 0.0052 - accuracy: 0.9983 - val_loss: 0.0011 - val_accuracy: 0.9998
Epoch 19/20
673/673 [==============================] - ETA: 0s - loss: 0.0058 - accuracy: 0.9982      INFO:tensorflow:Assets written to: model-019.model\assets
673/673 [==============================] - 399s 593ms/step - loss: 0.0058 - accuracy: 0.9982 - val_loss: 0.0011 - val_accuracy: 0.9998
Epoch 20/20
673/673 [==============================] - 398s 591ms/step - loss: 0.0035 - accuracy: 0.9989 - val_loss: 0.0016 - val_accuracy: 0.9998
```

```
from matplotlib import pyplot as plt

plt.plot(history.history['loss'],'r',label='training loss')
plt.plot(history.history['val_loss'],label='validation loss')
plt.xlabel('# epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

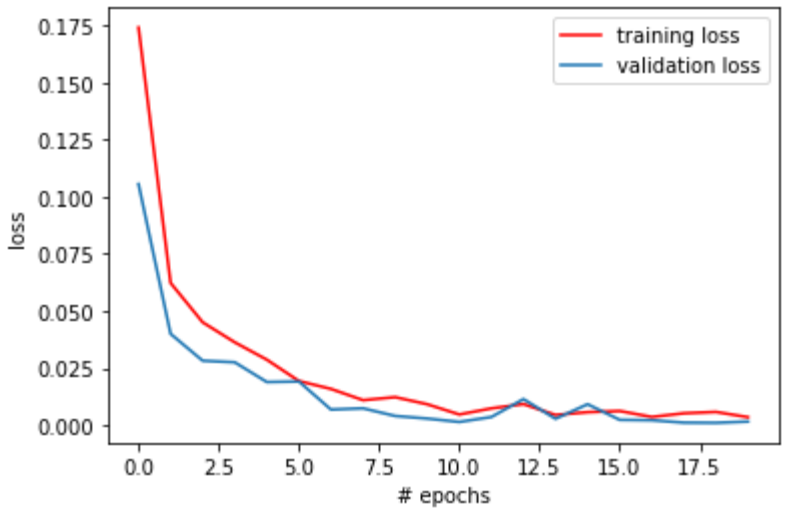

Fig. 4. Graph plot of Training loss and Validation loss against Epochs

```
plt.plot(history.history['accuracy'],'r',label='training accuracy')
plt.plot(history.history['val_accuracy'],label='validation accuracy')
plt.xlabel('# epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

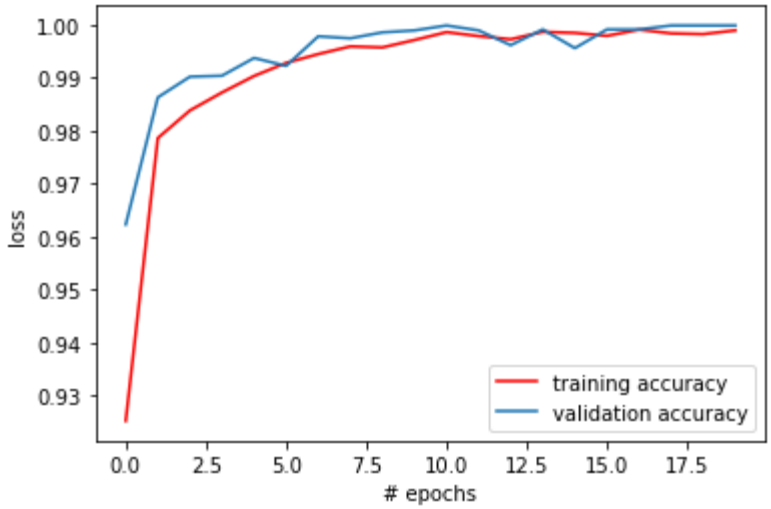

Fig. 5. Graph plot of Training accuracy and Validation accuracy against Epochs

```
In [30]: print(model.evaluate(test_data,test_target))
94/94 [==============================] - 15s 160ms/step - loss: 0.0034 - accuracy: 0.9993
[0.0033934623934328556, 0.999331533908844]
```

Once training is complete, the resulting model is evaluated on the test set.

4.2 Detecting COVID-19 face masks with OpenCV in real-time

After face mask detector is trained, the mask detector program is loaded, performing face detection, and then classifying each face as *with_mask* or *without_mask*

The following step is to launch the mask detector in real-time video streams using the following command:

```
model = load_model('Predator_mask_detection_model.h5')
face_clsfr=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

source=cv2.VideoCapture(0)

labels_dict={0:'with_mask',1:'without_mask'}
color_dict={0:(0,255,0),1:(0,0,255)}
```

```
while(True):

    ret,img=source.read()
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces=face_clsfr.detectMultiScale(gray,1.3,5)

    for (x,y,w,h) in faces:

        face_img=gray[y:y+w,x:x+w]
        resized=cv2.resize(face_img,(100,100))
        normalized=resized/255.0
        reshaped=np.reshape(normalized,(1,100,100,1))
        result=model.predict(reshaped)

        label=np.argmax(result,axis=1)[0]

        cv2.rectangle(img,(x,y),(x+w,y+h),color_dict[label],2)
        cv2.rectangle(img,(x,y-40),(x+w,y),color_dict[label],-1)
        cv2.putText(img, labels_dict[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)


    cv2.imshow('LIVE',img)
    key=cv2.waitKey(1)

    if(key==27):
        break

cv2.destroyAllWindows()
source.release()
```

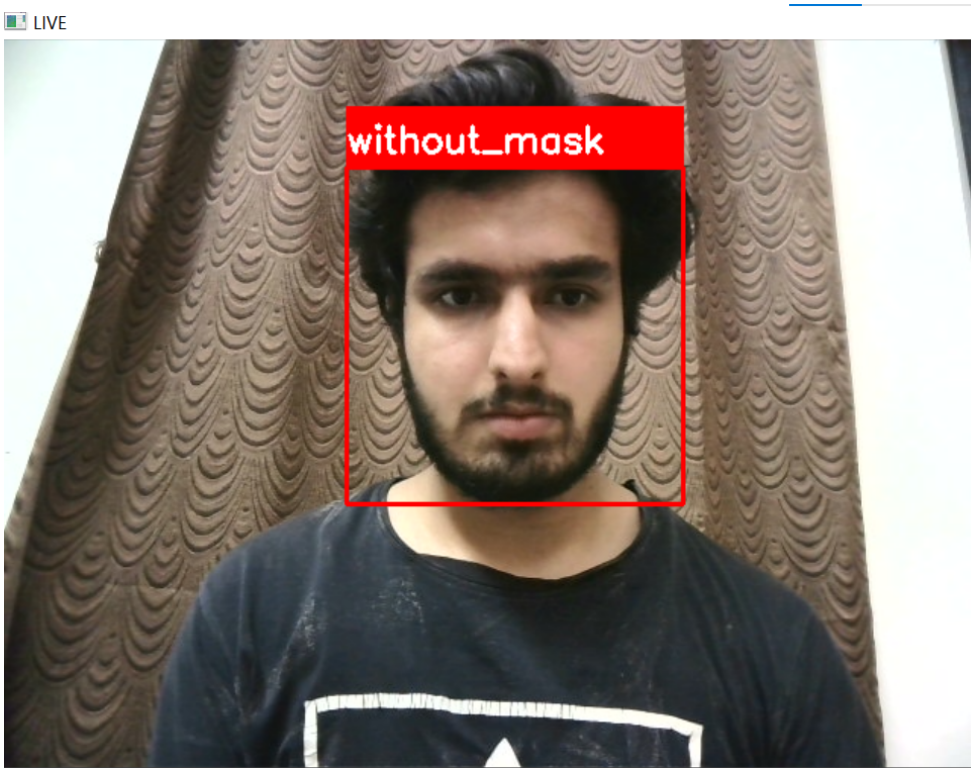

Fig. 7. Final Result of working model, shows without_mask when person is not wearing mask

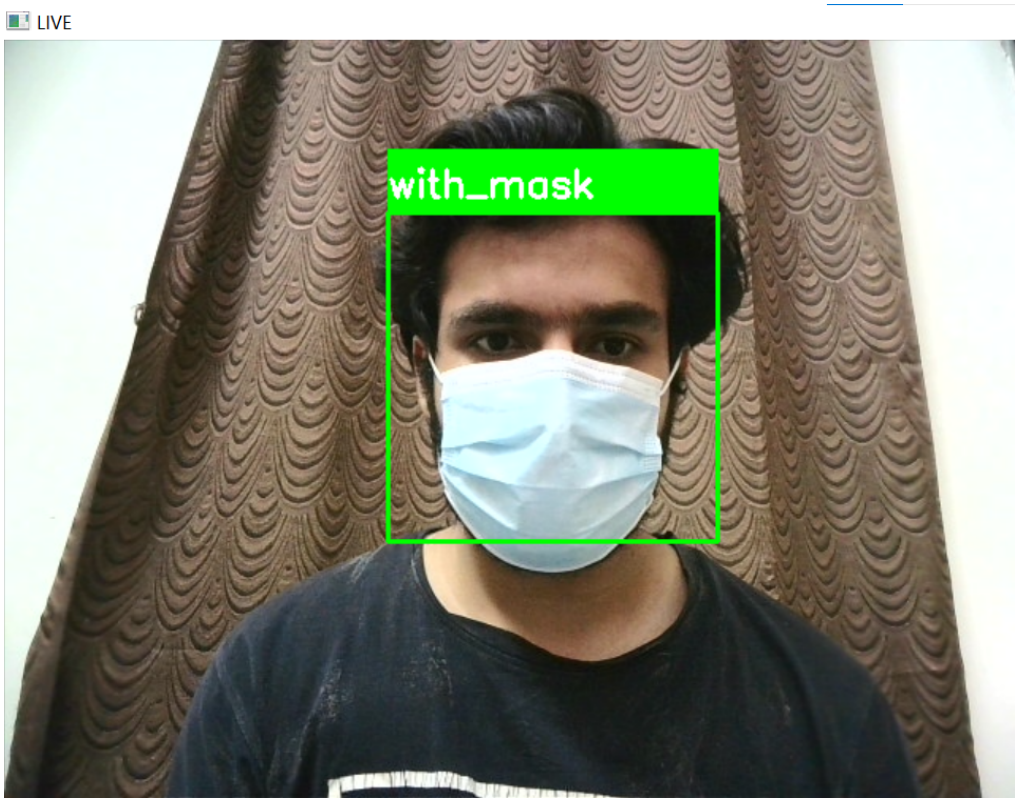

Fig. 8. Final Result of working model, shows with_mask when person is wearing mask

CHAPTER 5

RESULTS AND DISCUSSION

A two-phase COVID-19 face mask detector is developed.

The dataset is reviewed and pre-processed .

A Python script is implemented to train a face mask detector on our dataset using Keras and TensorFlow.

We have implemented following two Python scripts to:

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

The face mask detector model is trained and achieved Accuracy of 99.93%

CHAPTER 6

CONCLUSION AND FUTURE WORK

5.1 Conclusion

COVID-19 is currently a global pandemic, and many countries are applying different mechanisms and technologies to fight this virus. Among these mechanisms, a face mask is one of the protection mechanisms. Our research work's main intuition was focused on the detection of COVID-19 face mask to detect (classify) whether the person is wearing a mask or not wearing a mask with help of the CNN deep learning classification algorithm OpenCV.

5.2 Scope

COVID-19 is currently a global pandemic, and many coun-tries are applying different mechanisms and technologies to fight this virus. Among these mechanisms, a face mask is one of the protection mechanisms. Our research work's main intuition was focused on the detection of COVID-19 face mask to detect (classify) whether the person is wearing a mask or not wearing a mask with help of the CNN deep learning classification algorithm OpenCV.

This face mask detector can be sent in numerous regions like shopping centers, air terminals and other substantial traffic places to screen people in general and

to dodge the spread of the infection by checking who is following essential rules and who isn't.

5.3 Future Work

The present model proposed gives great accuracy for a single face with and without mask. For multiple faces also it gives quite good accuracy. It works easily on any mobile device just by switching on the video stream, with no external hardware requirement. Further we will work for improving the accuracy of multiple face mask detection, to classify the faces into three categories that is, With mask, without mask, Improper Mask instead of just the two with and without mask class by adding datasets with images of people wearing mask not covering their noses properly and also to detect the masked face using the Convolutional Neural Network so as to further improve our model and add marking attendance feature in it by detecting the face even when the mask is on. The proposed model can also be enhanced by means of including various parameters like peoples count, social distance and temperature measurement.

REFERENCES

[1] Jiang, Xinbei & Gao, Tianhan & Zhu, Zichen & Zhao, Yukang. (2021). Real-Time Face Mask Detection Method Based on YOLOv3. Electronics. 10. 837. 10.3390/electronics10070837.

[2] Islam, Md. Shahriar & Moon, Eimdadul & Alam, Mohammmad. (2020). A Novel Approach to Detect Face Mask using CNN. 800-806. 10.1109/ICISS49785.2020.9315927.

[3] Zhihao Cao, Mingfeng Shao, Li Xu, Shaomin Mu, Hongchun Qu. (2021). MaskHunter: real-time object detection of face masks during the COVID-19 pandemic. 10.1049/iet-ipr.2020.1119

[4] A. Das, M. Wasif Ansari and R. Basak, "Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV," 2020 IEEE 17th India Council International Conference (INDICON), 2020, pp. 1-5, doi: 10.1109/INDICON49873.2020.9342585.

[5] H. Adusumalli, D. Kalyani, R. K. Sri, M. Pratapteja and P. V. R. D. P. Rao, "Face Mask Detection Using OpenCV," 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 2021, pp. 1304-1309, doi: 10.1109/ICICV50876.2021.9388375.

[6] S. Sakshi, A. K. Gupta, S. Singh Yadav and U. Kumar, "Face Mask Detection System using CNN," 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2021, pp. 212-216, doi: 10.1109/ICACITE51222.2021.9404731.

[7] S. Susanto, F. A. Putra, R. Analia and I. K. L. N. Suciningtyas, "The Face Mask Detection For Preventing the Spread of COVID-19 at Politeknik Negeri Batam," 2020 3rd International Conference on Applied Engineering (ICAE), 2020, pp. 1-5, doi: 10.1109/ICAE50557.2020.9350556.