

FACEMASK DETECTION USING CONVOLUTIONAL NEURAL NETWORKS

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

By

Karri Satyanarayana Reddy (Reg. No. 38110237)
Gangula Dinesh (Reg. No. 38110133)



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

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600

119

MARCH - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **KARRI SATYANARAYANA REDDY (38110237)** and **GANGULA DINESH (38110133)** who carried out the project entitled "**FACEMASK DETECTION USING CONVOLUTIONAL NEURAL NETWORKS**" under my supervision from **October 2021 to March 2022**.

Internal Guide

Dr K.MOHANA PRASAD , M.Tech , Ph.D.

Head of the Department

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

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, **KARRI SATYANARAYANA REDDY** (38110237) and **GANGULA DINESH** hereby declare that the Project Report entitled **“FACEMASK DETECTION USING CONVOLUTIONAL NEURAL NETWORKS ”** done by me under the guidance of **Mr.K.MOHANA PRASAD**(Internal) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering / Technology degree in **Computer Science and Engineering**.

DATE:

PLACE:

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 **Mr K.MOHANA PRASAD, M.Tech, Ph.d** for his 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

Changes in the lifestyle of everyone around the world. In those changes wearing a mask has been very vital to every individual. Detection of people who are not wearing masks is a challenge due to outbreak of the coronavirus pandemic. This project can be used in schools, hospitals, banks, airports, and where there is no human surveillance places. The technique of detecting people's faces and segregating them into two classes namely the people with masks and the people without masks. It is done with the help of image processing technique and deep learning i.e using convolutional neural networks algorithm. Using CNN model the weight of the data while processing will reduce. With the help of this project we can detect a person who is under the control of remote area wheather he/she was wearing a mask or not by taking data from the live video stream .

CONTENTS

Page No	
ABSTRACT	v
LIST OF FIGURES	x
LIST OF TABLES	ix
CHAPTER 1: INTRODUCTION	10-16
1.1 Preamble	10
1.2 Deep Learning	10
1.3 OpenCV	11
1.4 Keras	12
1.5 Problem Statement	13
1.6 Aim of the project	14
1.7 Significance of the project	15
1.8 Conclusion	16
CHAPTER 2: LITERATURE SURVEY	17 - 41
2.1 Preamble	17
2.2 Literature Review	17
2.3 Conclusion	41
CHAPTER 3: METHODOLOGY	42 -56
3.1 Preamble	42
3.2 Tools Used	42
3.3 Hardware Requirements	44
3.4 Communication Interfaces	45
3.5 Software requirements/Hardware Requirements	45
3.6 System Architecture diagram	46

47	3.7Architecture of CNN model	
	3.8Sequence Diagram	49
	3.9 Activity Diagram	50
	3.10 DataFlow Diagram	51
	3.11 Proposed system	51
	3.12 Proposed Methodology	52
	3.13 Proposed Methodology Modules	52-53
	3.13.1 Data collection	
	3.13.2 preprocessing	
	3.13.3 Splitting the data	
	3.13.4 Building the model	
	3.13.5 Testing the model	
	3.13.6 Implementing the model	
	3.14 Testing	54
	3.15 Types of Testing	54 - 55
	3.15.1 Black box testing	
	3.15.2 White box testing	
	3.16 Test cases	56
	CHAPTER 4: RESULTS AND DISCUSSION	57 - 58
	4.1 Results	57
	4.2Results Comparison	58
	4.3Application	58
	CHAPRER 5 : CONCLUSION AND FUTURE ENHANCEMENT	59
	5.1 Conclusion	59
	5.2 Future Enhancement	59
	REFERENCES	60 - 61

APPENDIX – A	SOURCE CODE	62 - 75
APPENDIX – B	SCREENSHOTS	76 - 81

LIST OF FIGURES

FIG NO	FIG NAME	PAGENO
Fig 3.1	Interactive mode	44
Fig 3.2	Python GUI (IDLE)	44
Fig 3.3	System architecture	46
Fig 3.4	Architecture of CNN model	47
Fig 3.5	Sequence Diagram	49
Fig 3.6	Activity Diagram	50
Fig 3.7	Data flow diagram	51
Fig B1	Python-3.7.4 setup	76
Fig B2	Installation of keras	77
Fig B3	Installation of opencv	78
Fig B4	Installation of Numpy	78
Fig B5	Output Screenshots	79 - 82

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
Table 2.1	Comparison of Literature Review	36
Table 3.1	Test case table	55

CHAPTER 1

INTRODUCTION

1.1 PREAMBLE

In this chapter we are going to introduce our project domain, its architecture, problem statement, aim, methodology we are going to include in our project and what we are going to do in our project.

1.2 Deep Learning

Deep learning is a unique confine of machine learning. A new take on learning representations from data that puts an emphasis on learning successive layers of increasingly meaningful representations. The deep in deep learning isn't a authority to any kind of deeper understanding concluded by the approach rather, it stands for this idea of successive layers of representations. How many layers promote to a pattern of the data is called the depth of the model. Other appropriate names for the field could have been layered representations learning and hierarchical representations learnings. Modern deep learning often require tens or even hundreds of successive layers of rendition and they're all learned significantly from exposure to training data. Meanwhile other approaches to machine learning tend to thesis on learning only one or two layers of representations of the data; they are sometimes called shallow learning. In deep learning these layered description are almost always learned via models called neural networks is a citation to neurobiology, but although some of the central concepts in deep learning were expanded in part by drawing innovation from our understanding of the brain, deep learning models are not models of the brain. Deep learning techniques aim at learning feature ladder with features from higher levels of the hierarchy exhibit by the composition of lower level features. Automatically learning features at multiple levels of extraction allow a system to learn complex functions mapping the input to the output directly from data, without depending entirely on human-crafted features. Deep learning algorithms seek to enterprise the unknown

structure in the input dispatch in order to discover good representations, often at multiple levels, with higher-level learned analysis defined in terms of lower-level features.

The hierarchy of concepts permit the computer to learn tricky concepts by building them out of simpler ones. If we draw a graph showing how these concepts are raised on top of each other, the graph is deep, with many layers. For this reason, we call this approach to AI deep learning. Deep learning excels on problem domains where the inputs and even outputs are analog. Meaning they are not a few plentiful in a tabular format but instead are images of pixel data, documents of text data or files of audio data. Deep learning permits computational models that are contrived of multiple processing layers to learn representations of data with multiple levels of abstraction.

1.3 OpenCV :

OpenCV(open source computer vision library) is an open provenance computer perception and machine learning software library. OpenCV was built to transmit a common configuration for computer vision applications and to hurry the use of machine perception in the commercial products. Being a BSD-licensed outcome, OpenCV makes it easy for businesses to exploit and improve the code.

The library has more than 2500 transform algorithms, which contains a complete set of both typical and state-of-the-art computer discernment and machine learning algorithms. These algorithms can be used to determine and perceive faces identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to compose a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and display markers to overlay it with augmented actuality, etc. OpenCV has more than 47 thousand people of user organizations and evaluate number of downloads unusual 18 million. The library is used greatly in organizations, exploration sector and by governmental bodies. Along

with standard organization like Google, Yahoo, Microsoft Intel, IBM, Sony, Honda, Toyota that absorb the library, there are countless startups such as Applied Minds, Video surf, and Zeitera, that make immense use of OpenCV. It has C++, Python, Java and MATLAB interfaces and reinforce Windows, Linux, Android, and Mac OS. OpenCV leans primarily towards real-time vision solicitation and takes advantage of MMX and SSE command when available. A full featured CUDA and OpenCL interfaces are being actively transpire right now. There are over 500 algorithms and about 10 times as many basis that compose or support those algorithms. OpenCV is written natively in C++ and has a template attachment that works seamlessly with STL containers.

1.4 KERAS

Keras is an API examined for human beings, not machines. Keras obey best practices for minimize consequent load it offers consistent & simple APIs, it minimizes the number of user actions required for familiar use cases, and it produce clear & applicable blunder messages. It also has considerable corroboration and immense guides. Keras restrain numerous implementation of frequently used neural network construction blocks such as layers, objectives, operate justification, optimizers, also a host of tools to make functioning with image and text data trouble-free to clarify the coding mandatory for writing deep neural networks code. It was developed to make contrive deep learning models as fast and easy as feasible for research and expansion. It runs on python 2.7 or 3.5 and can seamlessly achieve on GPUs and CPUs given the elemental frameworks. It is rescue under the liberal MIT license.

Keras was expand and conserve by Francois Chollet, a Google engineer using four guiding

principles:

Modularity: A imitation can be understood as a course or a graph alone. All the perturb of a deep learning model are discrete components that can be integrate in inconsistent ways.

Minimalism: The library contribute just enough to achieve an outcome, no frills and spiral readability.

Extensibility: New constituent are purposely easy to add and use within the framework, conscious for researchers to trail and explore new ideas.

Python: No unrelated model files with practice file formats. Everything is indigenous Python Keras is delineate for minimalism and segmental allowing you to very rapidly define deep learning imitation and run them on brink of a Theano or TensorFlow backend.

1.5 PyTorch

PyTorch is an open origin machine learning library form on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Face book's AI Research lab[FAIR].

It is free and open-source software rescue under the reform BSD authorize. whilst the python extension is more shining and the central focus of development, PyTorch also has a C++ interface. Tensor enumerate like NumPy with strong enumerate via graphics processing units(GPU).

Deep neural networks construct on a tape-based automatic differentiation system PyTorch defines a class called Tensor to stock and manage on homogeneous multidimensional rectangular arrays of numbers. PyTorch Tensors are homogeneous to NumPy Arrays, but can also be manage on a CUDA-capable Nvidia GPU. PyTorch supports various sub-types of Tensors.(48%)

1.5 PROBLEM STATEMENTS

Changes in the lifestyle of public around the world. In those replace wearing a mask has been very vital to every isolated. Detection of people who are not wearing masks is a provocation due to outbreak of the coronavirus pandemic has created various the huge number of populations. This project can be utilize in schools, hospitals, banks, airports, etc. as a digitized scanning tool. The technique of detecting people's faces and sort out them into two classes namely the people with masks and people without masks is finished with the assuage of image processing and deep learning. With the help of this project, a person who is considered to monitor the people can be seated in a remote area and still can monitor logical and give instructions accordingly. Various libraries of python such as OpenCV, Tensor flow and Keras. In deep learning convolution neural networks is a class deep neural networks which is used to train the pattern used for this project.

COVID-19 has affected the world badly. Studies have proved that wearing a face mask is one of the precautions to reduce the risk of vital transmission. And many public places as well as public service providers require customers and their staff to use the service and place only if they wear mask correctly. So, it is not possible manually to track the customer whether they have the mask or not. That's why this technology grasp the key here. In this technique we propose face mask detection using image processing which is one of the high-accuracy and efficient face mask detector. Our system is capable of detecting masked and unmasked faces and can be integrated with webcamera.

1.6 AIM OF THE PROJECT

The whole world today, is covering the COVID-19 pandemic. People are using various measures to control the spread of Corona virus. There are so many vital measures

which are needed to find against COVID-19 and one of such most important is face mask. Lots of research and studies on COVID-19 are still going on studies have also proved that wearing a face mask remarkably reduces the problem of viral transmission. Also a person wearing mask recognizes a sense of protection. However, it is not feasible manually to check whether a person is wearing a mask or not. Technology comes into picture here. Deep learning and artificial intelligence consists of various technologies which provide effective solutions to complex problems in different areas. In order to prevent the spread of Corona virus, we have made an attempt to develop a face mask recognition system using Deep learning. The present work deals with the study of many face detection approaches and the proposed method of detecting face mask using video analytics in deep learning. The main objective of our project is to detect whether the person is wearing or not. And the person who is not wearing a mask the system will send a remainder message to email or phone of the person. To execute this idea we are using webcam to take snapshot in a time and faces in the image are analyzed by image processing techniques. After processing the system will match the each and every person in the image with the registered data base in the organization. If the face of the person is matched upto 30%-40% then the system will consider the person as masked. If the face of the person is matched upto 80%-90% then it will consider as unmasked person. Then it send a warning or fine, remainder message to the E-mail or Phone number that they registered.

1.7 SIGNIFICANCE OF THE WORK

In recent years, face mask plays important role. This is because corona virus COVID-19 pandemic is causing a worldwide health crisis therefore the effective

protection methods is wearing a mask. Reports indicate that wearing face masks while at work or any where outside clearly reduces the danger of transmission. A key weapon against the spread of COVID-19 has been the use of facemasks. This has been mandated by governments of different countries, based on the guidelines by the World Health Organization[WHO].

According to the WHO face mask can be used for control of source or for the protection of healthy people. Our proposed work can detect the unmasked persons effectively. Detection of unmasked persons will be helpful to stop the viral transmission.

1.6CONCLUSION

In this chapter, we have discussed the introduction, problem statement, aim of the project, significance of the work and conclusion. Due to COVID-19 pandemic the facemask detection plays an important role in society. Manually it is not possible to detect the unmasked persons. So, we have proposed this technique to detect face mask of a person. It sends a remainder message to the person who is detected as unmasked.

CHAPTER-2

LITERATURE REVIEW

2.1 PREAMBLE

The previous chapter gives a brief introduction of the project and scope of the work. We have discussed problem statement, aim of the project, methodology and also significance of the work carried out in the project. We have also provided the organization of the whole report in chapter 1. This chapter provides a brief introduction about the literature review done related to the topic of our work. This chapter provides a brief review of various existing approaches discussed in the earlier research works. Different research works speak about deep learning techniques for detection of face mask and other issues and challenges in their perspectives. We have discussed the various research papers and journals with their merits, demerits and future direction in this chapter.

2.2 Literature review related to face mask detection

Most of the research work on facemask detection can be done in two methods either in machine learning or deep learning techniques. In this section, we will briefly discuss all the papers one by one that we have surveyed related to our project.

1. Face Mask Detection using Machine Learning and Deep Learning

Saiyam Jain, Mayank Goya, Deepak Singh, Abhishek Aswal, Upasana Joshi are the authors of this paper. In this paper[1], they proposed a model which focusses on identifying face mask of a person through an image or video stream with the help of Deep Learning and Machine Learning using Keras, Tensor flow, OpenCV, and

the Scikit-learn library. They have designed the model in two phases Training and deployment phases. The training phase of the model on the dataset using Tensor flow &Keras. The deployment phase of the model loading the trained model and applying detector over images/live video stream. The result of this paper is creating a face mask detector using deep learning. They trained the system to distinguish between people wearing mask and people not wearing a mask. They have used MobileNetV2 classifier with the ADAM optimizer for the best results. This model is tested with photos and real-time video streams. It detects the face from the images/videos and extracts each individual's face and apply the face mask classifier to it.[1].

2. Automatic Facemask Detection using MobileNetV2

Vasanthakumari, Mohd.UmarPeerzade, Subhralina Nayak, Bhagya, Padmanabhan are the authors of this paper. In this paper[2], they developed a novel solution for detecting whether a given image of a person has a facemask or not. They had proposed a model that uses MobileNetV2 for image processing. It is a CNN that is 53 layers deep. It can classify images into 1000 categories. It uses depth-wise separable convolutions which constitute as the building blocks. It is the state-of-art network for mobile visual recognition which includes classification, object detection and semantic segmentation. It is faster when compared with MobileNetV1 because it uses two times fewer operations and it needs less number of parameters. This approach consists of following phases data pre processing, Training and testing of the MobileNetV2 model using the dataset of images and classifying new unseen images means classifying images at real-time when an image or video is streamed. Finally as a result this paper executes a model that focusses on detecting people in images as wearing a mask or not, using Tensor flow, **keras**, and mobilenetv2.It uses a deep learning algorithm in

which it takes input as an image, uses weights and biases and outputs the class of the images.[2] (3%)

3.Face Mask Detection using Convolutional Neural Networks

Yash Maheshwari, Nikhil Gugliya, Atharva chennur, Manisha Tandale are the authors of this paper. In this paper[3], Here they proposed a hybrid model using deep and classical machine learning for mask detection. The dataset that is used in this approach consists of with mask and without mask images to detect an unmasked person they had used OpenCV to try for real time face mask detection from a live stream via webcam. They had used the dataset to create a COVID-19 mask detector with computer vision. The proposed system mainly focuses on the way to identify the person on image or video stream wearing mask with the help of computer vision and deep learning algorithm by using the OpenCV, Tensor Flow, keras, and Pytorch library. Firstly it will take the video stream or image that is grabbed from the webcam which is under the control of remote area it will take it as input and after it enter into image processing module. In this module the image or video extract after it will match with the database. After that it will generate a pattern by using pattern matching module. The completion of this phases it generates an output which is separation of masked and unmasked persons. The accuracy of this model is achieved. Finally the result is detection of masked and unmasked person by using above mentioned libraries.[3]

4. Masked Face Recognition Dataset and Application

Zhongyuan Wang, Guangcheg Wang, Baojin Huang, ZhangyangXiong, Qi Hong, Hao Wu, Peng Yi, Kui Jiang, Nanxi Wang are the authors of this paper. Most current advanced face recognition approaches are designed based on deep learning, which depends on a large number of face samples. However at present there are no publicly available masked face recognition datasets. The work proposes three category of masked face datasets including Masked Face Detection Dataset (MFDD), Real-world Masked Face Recognition Dataset (RMFRD), and Simulated Masked Face Recognition Dataset (SMFDD). Among them RMFDD is presently the world's largest real-world masked face dataset. These datasets are freely available to the developers who develops the face mask detectors. A small introduction to these three datasets also given firstly MFDD it mainly includes two parts a) Some of the samples are from related researchers b) The other part of MFDD is crawled from the Internet. The next methodology is RMFRD it is a python crawler tool used to crawl the front-face images of public figures and their respective masked face images from massive internet resources. SMFRD is an dataset methodology which used to expand the volume and diversity of the masked face recognition dataset, To improve data manipulation efficiency, They have developed a mask wearing software based on Dlib library to perform mask wearing automatically. This software is mainly used to wear mask on face images in the popular face recognition datasets. Finally they built the above mentioned three datasets which are MFDD, RMFDD, SMFDD and also developed an algorithm called state-of-art algorithm based on these datasets.

5 A Facemask Detector using Machine Learning and Image Processing Techniques.

Amrit Kumar Bhadani, Anurag Sinha are the authors of this paper. In this paper the authors proposed a technique which is used for identification of facemask with the help of image processing and deep learning. With this project a person who is under remote area are monitored efficiently and give instructions accordingly. This instructions phase is purely manual. Here they used Convolution neural networks as

a class Deep neural networks which is used to train the models used for the project. In this project they proposed a Fast Neural Network for the face mask detection. This approach is used to reduce the computational time for locating human faces with or without mask.

Here in this technique each image is divided into small sub images and each and every sub divided image is tested separately using fast ANN which Artificial neural networks. The result of this experiment achieved high speed with Fast Neural Networks when compared to Convolutional neural networks.

In this paper they performed 9 steps to identify masked and unmasked persons. The first step is Data visualization which is used for visualizing the images into two categories masked and unmasked. Second step is Data Augmentation in this step they expand the data to include large number of images. Splitting the data they split the data into training set and test that data. Building the model they use Conv2D, Maxpooling2D, Flatten, Dropout, and Dense to build a sequential CNN model. Step-5 which is pre-training the CNN model here they create a train-generator and validation-generator which are fit for this model. In the next step the training of the CNN model has to be done. In step-7 they label the information with two probabilities which are with mask and without mask. In the next step they import the face detection program. Last and final step is to detecting the faces with and without mask.

6 COVID-19 Facemask Detection with Deep Learning and Computer Vision.

Vinitha, Velantina are the authors of this paper. In this paper the authors introduced a masked face detection model that is based on computer vision and deep learning. This model can be integrated with surveillance cameras. The proposed method mainly focuses on how to identify the person on image or video stream wearing face mask with the help of computer vision and deep learning algorithms. Firstly it will load the dataset which contains masked and unmasked images of a persons after

loading it will augment the face mask data using pytorch torch vision. After augmenting the image processing using the pytorch has been done and then it will generate and train face mask classifier by using MobileNetV2 and pytorch now it will serialize the face mask classifier and save the data. Now it will load the data from the disk or memory. After grabbing the data from the disk or memory it will load the particular image or video stream to apply the process. Again the 3rd step has to be performed which is image processing using pytorch now finally it will apply the face mask detector to determine whether the person is wearing a mask or not. Finally it shows the results as image boundary around the image with green or red boundary. In this approach they use a web camera to take images or video streams for processing the technique. This method can also be used in both high & low computational devices. This model can also be used in public places to reduce the viral transmission.

7 A Novel GAN-Based Network for Unmasking of Masked Face

Nizam UD Din, Kamaranjaved, Seho and Junehoyi are the authors of this paper. In this paper they proposed they different technique to unmask a person who is wearing mask. This idea is helpful to detect the faces of a person who are hiding them. Due to this pandemic each every on are using facemasks to prevent the viral transmission WHO has declared that facemask is the weapon to reduce the spreading of virus. This method is mainly used to identify a particular person. Recently deep learning image identifying and editing methods give an accurate results but failed to generate results after removing large objects like facial masks in complex nature. In this paper they proposed the technique for removing of large objects also. This model can be explained in three modules Map Modules, Object detection, and Editing Module this editing module contains editing generator which edits the image before unmasking or removing a large object from the image. Here

they use two discriminators called D-whole region and D-mask region. The main role of these two discriminators is to separate the dissimilar part of the image and force the editing generator to produce visually consistent images. There are some exceptions or limitations in this project. This project is trained accordingly to the limited facemasks only. It means only 15 types of facemasks are used in this project. None of the other facemasks were used in the entire project. And another drawback in this project is if the missing region of the person's image is very large then the edge generator cannot generate reasonable edges and the final outcome fails.

8 Face Mask Detection

Ms.A.p. Nithya Priya, V. Mavithasri, S. Monisha, P. Prathibha, K. Rekha are the authors of this paper. Here they proposed a system which is based on deep learning algorithms. They had taken the input as the video stream and perform a segmentation process for isolating the object from rest of the real video. Here the system is divided into two phases face segmentation and detection using semantic segmentation on RGB images. The authors used CNN algorithm with set of trained images which are divided into 2 parts positive and negative images and then classifying the images into masked and unmasked images. Firstly they take a video frame as input and perform segmentation process as mentioned above. The segmentation process has to be in three phases Hue saturation value, Gray level conversion, Descriptor's. After all the 3 phases completed the output from the above phases is combined and it is used as input for the next phase. In the next step two phases are included which are Database management which consists of datasets of two labels with mask and without mask, and the other phase in this step is CNN-Convolutional neural network algorithm is used in this phase. The output of the above two phases is combined and it is considered as input to the next step. And

the last but one step is Console window which checks for the result. Finally it displays the results in two patterns which are masked and masked. Finally the result of this paper is to detect whether the person is wearing a mask or not by using artificial intelligence.

9 Deep Learning Framework to Detect Face Masks from Video Footage

Aniruddha Srinivas Joshi, Shreyas Srinivas Joshi, Goutham Kanahasabai, RudrakshKapil are the authors of this paper. In this paper they had proposed an approach for detecting facial masks in videos using deep learning. The work is based on the MTCNN which is used for identifying the human faces and their corresponding landmarks those are present in the video frame. In this technique the images are processed by a neoteric classifier that uses the MobileNetV2 architecture as an object detector. This MTCNN model mainly consists of three phases which are P-Net, R-Net, O-Net after completion of this MTCNN process it enter into the Facial regions which analyse the faces in the video frame after that the facial mask prediction has been done. The above mentioned process is the proposed work flow of the authors. Firstly they will detect the face by using face detection process it means it will recognise the face first and the starts its processing. Stage-1 which is Proposal Network(P-Net) used to obtain the potential window in the input image or video and draw the boundaries of the image. Stage-2 consists of Refined network (R-Net) which takes all the candidate windows obtained in the previous stage. It rejects the false predictions of the candidate window and utilises the bounding box regression. And finally the stage-3 will comprises of a CNN called O-Net it is responsible for output's facial landmark locations, namely eyes, nose, and mouth regions of the face.

10 Face Mask Recognition using Machine Learning

TejalNerpagar, Sakshi Junnare, Jahnavi Raut, Aarti shah, Prof.P.C. Patil are the authors of this paper. In this paper they propose a technique which is used to detect facemask using image processing. The proposed technique mainly consists of three stages 1. Image pre processing is a first technique which is used for processing the images or videos that are considered as inputs. The second stage is Face detection and crop which will edit the images after detecting the faces. The last and final stage of face detection approach is face mask classifier. The system which the authors had proposed is capable of detecting masked and unmasked faces which are integrated with webcam cameras. This face mask classifier can quickly identify the person with a mask, using the cameras and analytics. Depending on the requirements of the system it is also inserted easily into the latest technology and tools. This technique is a simple and easy technique to detect the facial masks. After detecting the masked and unmasked person through the cameras it will draw a boundary with green and red colorground the image. And also it gives accuracy percentage of masked and unmasked faces on the top of the boundary.

11 Face Mask Assistant: Detection of Face Mask Service Stage Based on Mobile Phone.

Yuzhen chen, MenghanHu, Chunjun Hua, GuangtaoZhai, Jian Zhang, Qingli Li are the authors of this paper. Here they proposed a technique which is used to detect the service life of a mask using microscope of a mobile. This mini microscope will be easily attached to the mobiles at the place of camera lens. They propose the detection system based on the mobile phone. The main contribution of this project to the public service is to detect the life of a mask. First they will combine the texture

features of the face mask's by taking micro photos and their respective service stages based on the method of texture analysis. By using GLCM the texture features are easily extracted from micro photos which are taken by attaching a mini micro-scope to lens of the mobile. Subsequently, this detection method is based on KNN algorithm. While working with this technique we need a mobile and when taking images of a mask the flash of the mobile meet the condition that is it must in the "on" condition to provide enough light. Firstly it will detect or capture the image of a mask and then it will upload an image and the detection result is displayed on the screen of the mobile. The detection stage consists of two phases one is normal use and second is not use phase. To aquire the validity of the collected data the authors photographed the mask in three sided view inside its locations of the face mask each time. The first location is left location which is one third of the face mask and close to the left. And the second location is middle location about two-thirds of the mask. The third location of the mask is right side which is about one-third of the mask and close to the right. The final result of this project is to detect the service life of a mask.

11.Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 abode

Sunil Singh, Umang Ahuja,Munish Kumar, Krishan Kuma, Monika Sachdeva are the authors of this paper The article proposed an efficient real-time deep learning-based technique to automate the process of detecting masked faces, where each masked face is recognize in real-time with the help of bounding boxes. The boundless trials were categorize with popular models, namely, Faster RCNN and YOLO v3. F-RCNN has better correctness, but for applying this in real-world supervision cameras, it would be choose to use the pattern with YOLO algorithm as it apply single-shot detection and has a much unrestricted frame rate than Faster-

RCNN or any other state-of-the-art object detection algorithms. If we look at the speed/reliability tradeoff on the mAP at .5 IOU metric, one can tell YOLOv3 is stronger than faster R-CNN. Which model to use, also hang on on the resources obtainable. If high-end GPUs are accessible on the deployed devices, faster R-CNN must be worn. YOLOv3 can be arrange on mobile phones also. Since this approach is highly sensitive to the spatial location of the camera, the equivalent approach can be fine-tuned to better adjust with the corresponding field of view. These models can be used along with surveillance cameras in offices, metros, railway stations and crowded public areas to check if people are upcoming rules and wearing marks.

12.Real-time Facemask Identification using Facemask Net Deep Learning.

Madhura Inamdar, NinadMehendale are the authors of this paper. In this paper they proposed a facemask net model which aims at detecting whether a person is wearing a mask or not. Along with this improperly worn masks are also detected by using this project. If an image or video stream is given as an input to the facemask detector then the given input is divided into three phases as with a mask, improperly worn mask, and without a mask. It consists of two phases training phase and detection of face mask phase. In the first phase it load the dataset and train the dataset according to the loaded dataset after that it will automatically update the model and enter into the next phase. In the second phase it will load the trained model or dataset into this phase and detect the faces in the images and video streams. These images and video streams are the inputs taken for the detection model. Here in this phase it will extract the region of interest it means where the detector wants to recognize. And finally it will apply our facemask detector which is now shows the results in three categories which are with mask, improperly worn mask and without face mask. Finally this project distinguishes face masks from images and live video streams. On training the model using Facemask net, they got an accuracy of 98.6 %. This classifier was then implemented to images and live

video streams. This face mask identifier is least complicated in structure and gives instantaneous results and hence can be used in CCTV footages to identify whether a person is wearing a mask correctly so that he does not pose any hazard to others.

13 Identifying Facemask-Wearing Condition Using Image Super-Resolution with Classification Network to Prevent COVID-19

Bosheng Qin and Dongxiao Li Bosheng Qin and Dongxiao Li are the authors of this paper. In this study, the authors develop a new facemask-wearing condition recognition method by merging image super-resolution and classifying networks (SRCNet), which quantifies a three-category classification problem based on incontinent 2D facial images. The proposed algorithm restrain four main steps: Image pre-processing, facial detection and cropping, image super-resolution, and facemask-wearing situation identification. The pattern was directed and evaluated on the public dataset Medical Masks Dataset accommodate 3835 images with 671 images of no facemask-wearing, 134 images of incorrect facemask-wearing, and 3030 images of correct facemask-wearing. Finally, the proposed SRCNet achieved 98.70% accuracy. This paper describes the technology beyond the SRCNet and facemask-wearing condition finding, including the proposed algorithm, image pre-processing, facial detection and detach, SR network, facemask-wearing situation spotting network, datasets, and training details. Facemask-wearing condition identification is a kind of three-category categorization problem, counting no facemask-wearing (NFW), incorrect facemask-wearing (IFW), and correct facemask-wearing (CFW). The main goal is to form a facemask-wearing condition identification function, $FWI(x)$, which inputs an unrefined image and outputs the conditions of wearing facemasks for all faces in the image.

14 SSDMNv2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2.

PreetiNagrath, Rachna Jain, Agam Madan, Rohan Arora, Piyush Kataria, Jude Hemanth are the authors of this paper. The main goal of this project is to approximate whether a person has worn a mask correctly, the initial stage would be to train the model utilize a proper dataset. After training the classifier, an accurate face detection model is essential to detect faces, so that the SSDMNv2 model can classify whether the person is garments a mask or not. The task in this paper is to elevate the accuracy of mask detection without being too resource-heavy. For doing this task, the DNN module was used from OpenCV, which constrain a 'Single Shot Multibox Detector' object detection model with ResNet-10 as its backbone architecture. This perspective helps in determine faces in real-time, even on embedded tools like Raspberry Pi. The admirers classifier uses a pre-trained model MobileNetV2 to predict whether the person is wearing a mask or not. In these recommend face mask detection model named SSDMNv2, both the training and development of the image dataset, which was divided into categories of people having masks and people not having masks have been done successfully. The technique of OpenCV deep neural networks used in this pattern generate fruitful results. categorization of images was done accurately using the MobilenetV2 image classifier, which is one of the singleness of the proposed perspective. The problem of several wrong prognosis have been palmy separated from the model as the dataset used was collected from several other sources and images used in the dataset was cleaned manually to increase the accuracy of the results. This paper describe the proceeding of pre-processing the data and then training on data. initial, the authors interpret a function name called sort out alphanumerically to sort the list in lexicographical order. A function pre-processing is defined, which takes the folder to the dataset as input, then loads all the files from the folder and resizes the images as per to the SSDMNv2 model. Then the list is sorted using sorted non-alphanumeric, and then the images are converted into tensors. Then the list is converted to NumPy array for faster enumeration. After this, the process of data accretion is applied to extension the exactness after teaching the model.

15 Improving the Capability of Real-Time Face Masked Recognition using Cosine Distance.

DeviraAnggi Maharani, CarmadiMachbub, PranotoHidayaRusmin, LenniYulianti are the authors of this papers. This paper proposed a method to increase the accuracy of the face mask identification process and detection with the allocation of a person's ID number, using the cosine distance technique. In the detection process, Haar-cascade and MobileNet were implemented to obtain the face bounding box. Subsequently, the VGG16 transfer learning method and the Triplet loss FaceNet with multi-threading technique were compared in the face recognition process. When the training and testing process was done, the model would be stored for real-time recognition. In this study, two face detection procedure had been compared. Before the recollecting the person, the proposed system should proceed with the face detector first. This research venture to add a centroid tracker by comparing the current and previous bb face position with cosine distance. This paper proposed a method to increase the accuracy of the face mask recognition process and detection. To obtain the face bounding box, Haar-cascade and Mobile Net were implemented in the detection process. Subsequently, in the face recognition process, the VGG16 transfer learning method and the Triplet loss Face Net with multi-threading techniques were compared. The evaluation of various combination was done, resulting in Haar-cascade and Triplet loss Face Net as the most suitable for real-time implementation. To assign the ID number of a person, the cosine distance technique was used. FIFO technique was also utilized to increase the FPS result of the image stream with multi-threading. The proposed system successfully generates the name and id number even if the person is wearing a mask.

16. Multistage CNN Architecture for Face Mask Detection

Amit chavda, Jason Dsouza, Sumeet Badgujar, Ankit damani are the authors of this paper. The authors originate a deep learning establish system that can be detect instance where face masks are used appropriately. our system consist of a dual period convolution neural networks(CNN).An unmasked face can be integrate with preinstalled CCTV cameras. In this paper we recommend 2 stages CNN Architecture where the first stage detects human faces, while the second stage uses a lightweight image classifier to classify the faces descry in the first stage as either 'Mask' or 'No Mask' faces and draws bounding boxes around them. Here the retina face can be detected side view faces with good accuracy as well. while compared to MTCNN, Retina face significantly decreases the failure rate from 26.31% to 9.37%. In future this algorithm is extended to videos as well. Training a highly accurate face detector need a lot of labeled date, time, and compute resources for this reason we selected a pretrained model trained on large data sets easily. In the final result we get a highly accurate and robust Face Mask Detection System. Retina Face was selected in Stage 1, while the NASNet Mobile based model was selected as our Face Mask Classifier in Stage 2. The resultant system exhibits high performance and has the capability to detect multiple faces over a wide range of angles.

17 Face recognition in the scene of wearing a mask

Chao Qia , Lei Yang are the authors of this paper. The proposed work of this paper is the process of mask detection is added to MTCNN in this paper, the network structure in MTCNN and the connection relationship between each network need to be redesigned. The P-Net network is responsible for preliminary extraction of face targets; R-Net is responsible for screening face targets from P-Net while filtering out most of the face false detections; Impro-R-Net performs initial extraction of mask targeting among the remaining face targets; The Impro-O-Net network is

responsible for the final determination of face targets and mask targets. Research on occlusion problems in facial recognition is less than gestures, lighting, aging and expression. They propose a new robust face recognition algorithm for partial occlusion. They had used the improved MTCNN to detect face and mask occlusion areas. After detecting the mask occlusion area, we extract the LBP features of the non occlusion area, and input the extracted features into the support vector machine to recognize the face. They recognize the face based on the non-occluded part of the face image. The performance of any facial recognition system depends on the classifier used to recognize the face. In the scene of wearing a mask, the recognition accuracy is better than other networks. This shows that eliminating the interference of the occlusion area can enhance the extraction of the features of the non-occlusion area and improve the accuracy of its recognition. The recognition speed is a bit slower than the original due to the increased detection of the occlusion area. Compared with other algorithms, this method achieves the best recognition effect in the occlusion scene caused by wearing a mask.

18 Masked Face Recognition with Identification Association

Qi Hong, Zhongyuan Wang, Zheng He, Nanxi Wang, Xin Tian, Tao Lu are the writer of this paper. this paper proposes a masked face identification method based on person analyze association, which converts the masked face realization problem into an association uncovering problem linking the masked face and the appearing faces of the same person. Based on the characteristics that person establish approach does not rely solely on facial details, it first takes advantages of re-identification to establish the association between face masked pedestrians and face-unveiled pedestrians. It further supply an productive face image quality assessment to select the most recognizable faces for subsequent recognition n from a variety of appearing candidate faces. Finally, the selected high-quality recognizable faces are used to restore masked faces for identification. The

comparison investigation with the existing disguise face recognition technique show its superiority in terms of accuracy. Person re-identification refers to the approach of recognizing whether a pedestrian appears in a camera to reappear in other cameras, which can be used to establish correlation between the same pedestrian from different cameras. Person re-identification is established on the comprehensive features of the human body's aspect rather than depend merely on the human face, and thus pedestrian correlation can still be recognized through person re-identification when the face is not accessible. Therefore, this paper adopts a face image establishment method based on person re-identification. This paper has recommended a method based on person re-identification association for masked face identification. In addition, for the verification purpose, we have also pioneered a specific face-masked pedestrian dataset using real-world supervision images. Experiments results show that our method can faithfully realize the identity of face-masked pedestrian with far higher accuracy than the existing disguised face recognition method. Practically, the advocated method provides a feasible solution to the problem of identity recognition of fully masked faces.

18. Face mask detection using MobileNet and Global Pooling Block

Coronavirus disease (COVID-19) is the latest epidemic caused by the newly discovered coronavirus. COVID-19 is an infectious disease that affects the respiratory system caused by the SARS-CoV-2 virus. Our objective is face mask detection from the natural images and hence we have devised the transfer learning with a global pooling block as an output layer. Depthwise separable convolution is made up of two layers as follows.

- Depthwise convolution: It is used to apply a single filter to each input channel and acts as layer filtering.
- Pointwise convolution: It is used to create a linear combination of the output with the help of 1×1 convolution

We also have selected three popular pre-trained deep networks namely ResNet50, MobileNet, GoogleNet for evaluation of proposed model. Our proposed model has simulated using Python and Tensorflow. Each model has trained for 50 epochs. The proposed MNGP model outperforms its competing methods with 99% accuracy. However, MobileNet with global max pooling records best performance of 99.56%

Comparison table of Literature Survey

S.No	Year and Title	Domain used for proposed technique	Merits	Demerits/Future implications
1.	A Novel GAN-Based Network for Unmasking of Masked face. March 2 nd 2020	Deep Learning	By using this model the user can guess one's identity	If the facemask color is similar to the neck color of the person then this model fails to generate the unmasked facial image.
2.	A face mask detector using machine learning and image processing techniques, March 2020	Machine Learning	It works even in mobile devices also and it also works on low computational devices.	The system doesn't provide any alarm or alerting system after detecting the unmasked person to wear a mask

3.	COVID-19 face mask detection with deep learning and computer vision Aug 2020	Computer vision, MobileNetV2	The raw input images are converted into clean version.	It doesn't works when the person's image is in the side view(turns left or right view).
4.	Masked face recognition using Data and Application. 23 rd march 2020	Deep Learning	Multiple number of datasets with and without mask are developed. Which are used for improving the face recognition technology	The developed algorithm in this paper gives only 85% accuracy(mentioned in the paper).
5.	Face mask Assistant: Detection of facemask service stage based on mobile phone 1 st may 2021	Machine Learning	It have positive influence on protecting uninfected people and stops the possibly infected patients from	It can detect only disposable masks. It can't detect various types of masks.

			spreading virus.	
6	Masked Face Recognition using Convolutional Neural networks Dec 2019	CNN, MTCNN, FaceNet.	This approach can detect a person's face in any direction	It is used only for masks that are taken in the dataset. It doesn't recognize if the person wear a different mask which is not in the dataset.
7.	Deep Learning Framework to detect facemask from video footage dec 18 th 2020	Deep learning	It can detect the presence of facial mask on subjects not directly facing camera(side view of the person).	If the face detection model fails to detects the face or incorrectly identifies the object as face the performance of this approach gets affected.
8.	Face Mask detection using Machine learning and deep learning. Jan 2021	Machine learning	This model can detect facial images with both photos and video streams.	In this project it gives results only when the image file is saved as JPG and PNG files.
9.	Automatic facemask detection	CNN, MobileNetV2, Deep learning	It will accept all types of images and it	It can detect only one person at a time.

	using MobileNetV2. Feb 2021		can detect real time images.	
10.	Face mask recognition using Machine Learning. Dec 2020	Machine learning	It has high detection accuracy.	This project fails when the direction of the person is not towards the camera.
11.	Face recognition with facial mask application and neural network.	CNN, Deep learning	Reduce the dimensionality of problem and granting faster and better performance.	By this method if the limits are not eliminated it can significantly reduces the efficiency
12.	Real-time face mask identification using facemask net deep learning network	Deep learning	It gives high accuracy and efficient facemask detector.	The camera can't give exact classified partially hidden faces. If the camera height is greater than 10 feet.
13.	Masked face recognition with identification association	CNN, Deep learning	Retina face can detect side view faces with good accuracy when	Training a high accurate face detector need a lot of labelled data, time and compute resources.

			compared to MTCNN.	
14.	Masked face recognition with identification association	MTCNN (Multi-Task Cascaded CNN)		
15.	Face recognition in the scene of wearing a mask	MTCNN(Multi-Task Cascaded CNN)	The technique CNN used in this idea with fully connected layers is used.	It doesn't provide any solutions to posture, lighting, aging, and facial expression problems
16.	SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2	Deep learning, tensor flow, keras and OpenCV	The system is very lightweight and can even be used in embedded devices	This model without data augmentation showed significant decrease in its growth.
17.	Improving the capability of real-time face masked recognition using cosine	CNN, Tensor flow, and OpenCV	It provide tracking id for each and every person who is identified in	If the Cosine distance is 2m then the face recognition falsely classified.

	distance		the camera to avoid duplication.	
18.	Identifying facemask-wearing condition using image super-resolution with classification network to prevent COVID-19	SRCNet,Deep learning	It avoids the noisy data	If video is given as input the performance of the system decreases gradually.
19.	Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment	Deep learning	This technique can be used in smart city innovation and it boost up the development process in many developing countries	These detectors with region proposals require massive computation and high run-time memory.
20.	Face mask detection using MobileNet	MobileNetV2	Train data has been used for both training and validation	The pre-trained MobileNet has been intended for 1000 class

	and Global Pooling Block		of the model while the test data have been used for performance computation depicts training accuracy and loss of the proposed model on DS1.	classification. Thus, we have replaced the output layer with the global pooling (GP) block as our objective is a binary classification.
--	--------------------------	--	--	---

CONCLUSION

In this chapter, we have briefly discussed all the papers that we have surveyed. And gained the knowledge about our ongoing project. We keenly observed the limitations of the existing systems and we will try to overcome those limitations by applying efficient algorithms using different techniques.

CHAPTER -3

METHODOLOGY

3.1 PREAMBLE:

The previous chapter describes the literature review related to the face make detection approach in video analytic using deep learning. Here we use various approaches and techniques. we have discussed the various research papers and journals with their merits and demerits and also the future direction in chapter 2. Now coming to chapter 3 we describe the different kinds of tools used and different requirements to develop the proposed system. And the diagrams related to the project and its modules in detail. It also consists of the hardware requirements like processor, software requirements like python and frontend and backend tools, processor. Here deals with the design of the proposed system. And it also includes the system architecture, data flow diagram, and various UML diagrams to represent the design of the system.

3.2 TOOLS USED:

Python:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created during 1985-1990 by

Guido van Rossum Like Perl, Python source code is also available under the GNU general public license (GPL). Python is designed to be highly readable as it uses English keywords frequently while other languages use punctuations, and it helps to reduce syntactical errors.

- **Python is interpreted** – python is processed at runtime by the interpreter. There is no need for complications before executing your program. And this is similar to PERL AND PHP.

- **Python is interactive** – you can sit at a python prompt and interact with the interpreter directly to write your program.
- **Python is object-oriented** – python supports object-oriented style or a technique of programming that encapsulates code within objects.
- **Python is a beginner's language** – python is a great language for the beginner-level programmer and supports the development of a wide range of applications from simple text processing to www browsers to games.

Characteristics of python:

- Python language is freely available on the official website and you can download it from [the www.python.org](http://www.python.org) website.
- While comparing to C, C++, JavaScript, Java, Corba. python can be easily learned
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, CORBA , ActiveX, COM and Java

Basic Syntax:

Python is an interpreted programming language; here you need to write python files in a text editor and the files must be saved using the .py extension, then put those files into the python interpreter to be executed. The syntax of the Python language is the set of rules which helps beginners to write python programs. The indentation in python refers to the spaces and tabs that are used at the beginning of a statement. Interactive mode and script mode are two modes of python interpreter.

Interactive mode:

Python code can be executed directly in the python command prompt without passing the python script file to the interpreter

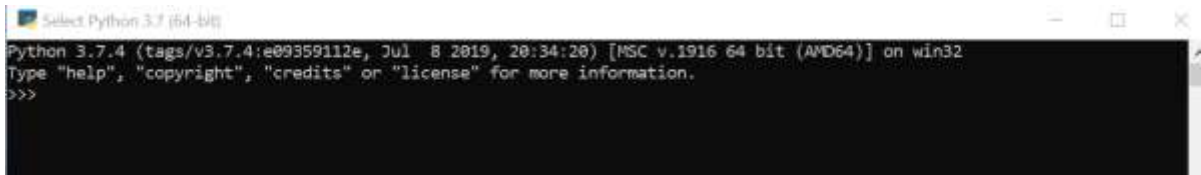


Figure 3.1 interactive mode

IDLE:

It is a Python IDE that Integrated Development Language, which has been bundled with the default implementation of the language. It is completely written in Python and the Tkinter GUI toolkit. IDLE is Python's Integrated Development and Learning Environment. It is cross-platform works mostly the same on Windows, UNIX, and macOS. It consists of a Python shell window (interactive interpreter) with colorizing of code input, output, and error messages.

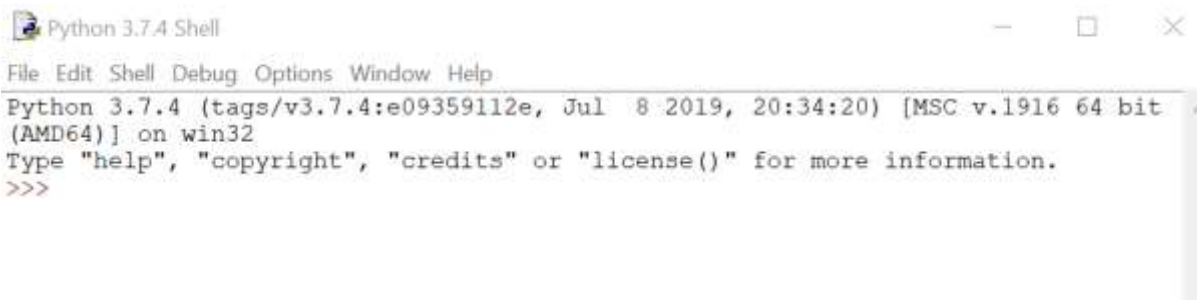


Figure 3.2:python GUL(IDEL)

3.3 HARDWARE REQUIREMENTS

The hardware requirements are used to develop the project are discussed here. We have applied our experiment on a system with processor intel Dual Core or advance with the memory or RAM with at least 8GB. the system type is 64 bits operating system(OS), we have used windows 10 as our operating system and CCTV camera at 1080 p HD. We have used these hardware requirements to develop the project in the python programming language. We have given the summary of software and hardware requirements in the coming section of the chapter.

3.4 COMMUNICATION INTERFACES

Anaconda acts as distribution for programming language

like python and R which include different packages and environments of data science that suitable for windows, mac, and Linux operating systems. Packages management system anaconda is used for managing the anaconda package version. Mini anaconda is the booster version of the anaconda which includes a small number of packages, python, and anaconda. Conda install command is used to install the open-source packages and the anaconda build command is used to install the custom commands.

3.5 SOFTWARE REQUIREMENTS/HARDWARE REQUIREMENTS

- Python IDEL
- Memory (RAM) : 8 GB
- OS : windows 10
- Camera : 1080 p HD
- System type :64-bits Operating System

3.6 SYSTEM ARCHITECHTURE DIAGRAM

This system architecture diagram helps developers and system designers to outline the software and to check whether the software meets their client's needs. Architecture design also knows as the blueprint of the software which contains the components and their relationship. A system architecture is a formal description and representation of a system, organized in a way that supports reasoning about the structures about behaviours of the system.

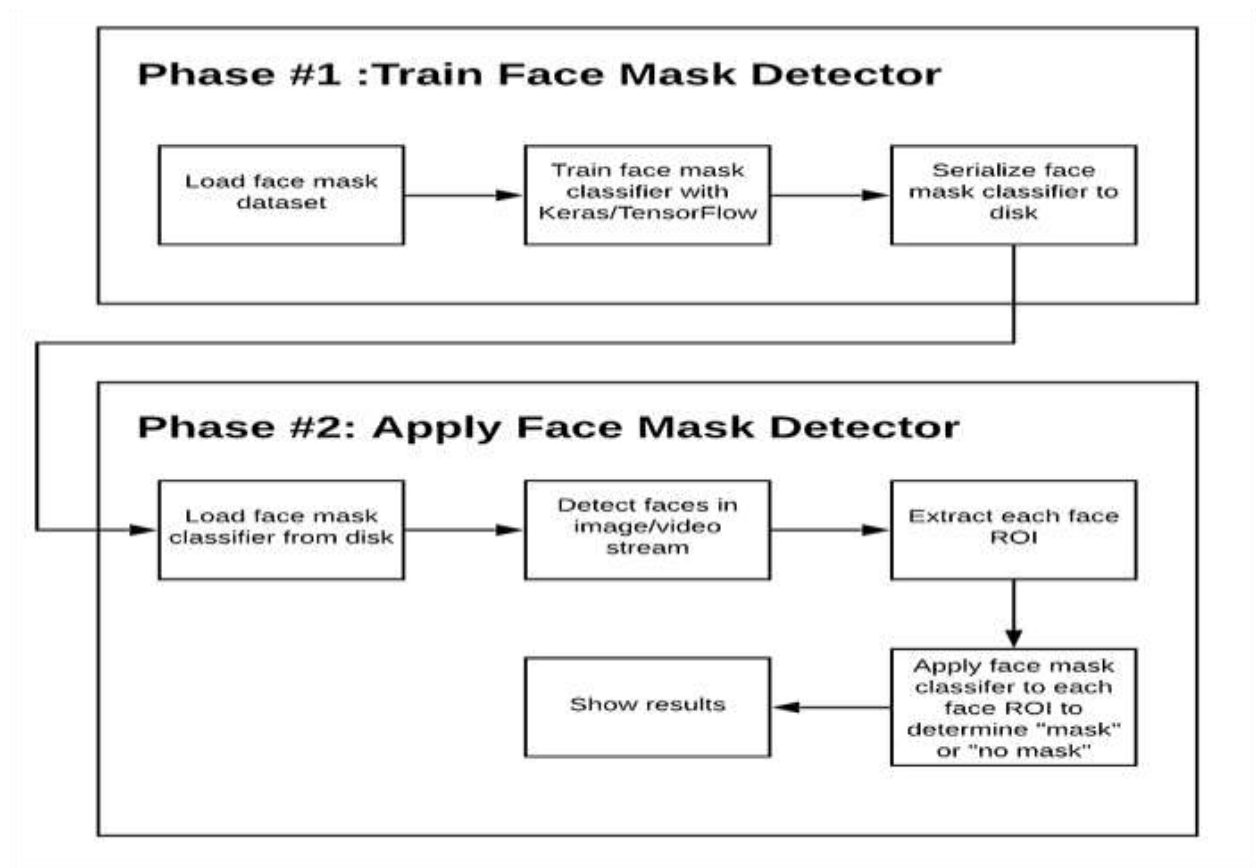


Figure:4.1 System Architecture

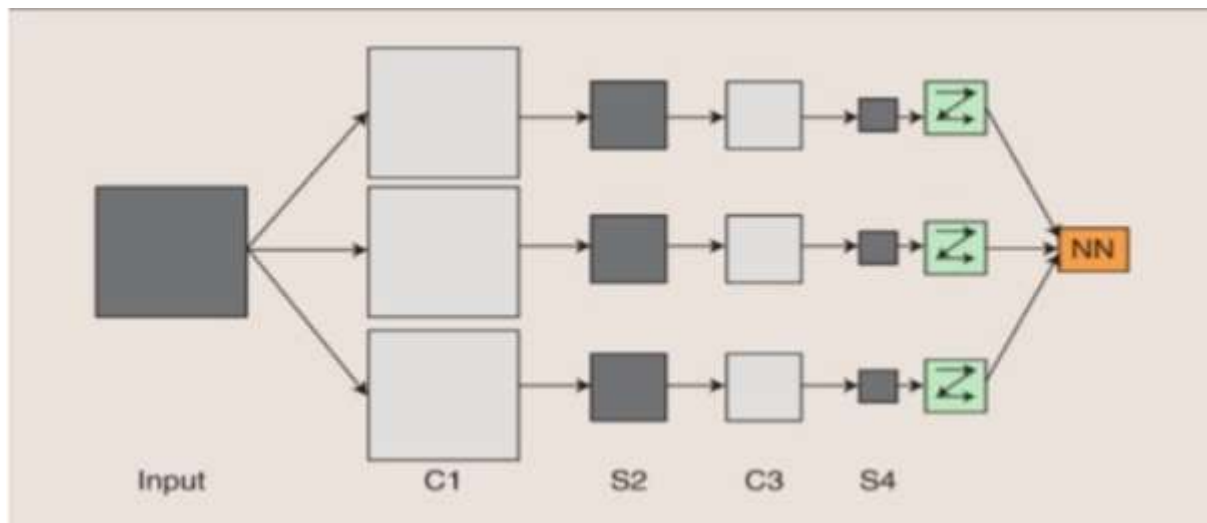
The above diagram represents the system architecture for the face mask detection approach in video analytic using deep learning. The proposed architecture of the face mask detection, firstly we need to perform the user gestures in front of the webcam so that the webcam will capture the images and the system processes those images are converted into greyscale and the denoised images are converted into black and white. The feature extraction can be done on the processed image. After feature extraction classification will be performed using a pre-trained CNN model. Finally, we will get the final out of a masked and unmasked person.

3.7 ARCHITECTURE OF CNN MODEL

In this paper, we present the architecture of the convolution neural networks (CNN) model. The role of the CNN is to reduce the images into a form that is easier to process, without losing the features which are critical for getting a good prediction. The preprocessing required in CNN is much lower than compared to other classification algorithms. CNN is a special kind

of multi-layer NNs applied to 2-d array(usually images), based on spatially localized neural input. CNN generate ‘pattern of patterns’ for pattern recognition. Each layer combines patches from previous layers.

Each stage in the CNN model is composed of :a filter bank layer, a non-linearity layer, feature pooling layer. A convet is composed of 1,2,3,or such 3-layer stages, followed by a classification module.



Basic structure of CNN, where C1,C3 are convolution layers and S2,S4 are pooled/sampled layers.

The CNN model is built using the Tensor-flow framework and the Opencv library which is highly used for real-time applications. This model can be used to develop a full-fledged software to scan person before they can enter the public gathering.

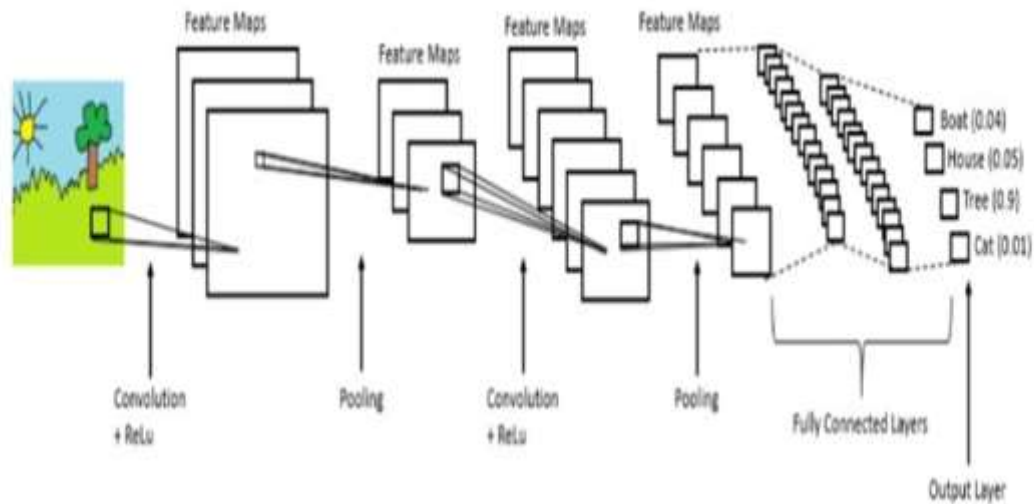


Figure 4.3 overall structure of CNN

3.8 PROPOSED SYSTEM

The below diagram represents the proposed system for face mask detection approach in video analytic using deep learning. In the proposed system of face mask detection, firstly we need to perform the presence of person in front of the webcam so that ‘it can capture the images and the system processes those images . after that feature extraction is done on processed images .classification is preformed using pretrained CNN model. Finally we will get the final output of with mask and without mask.

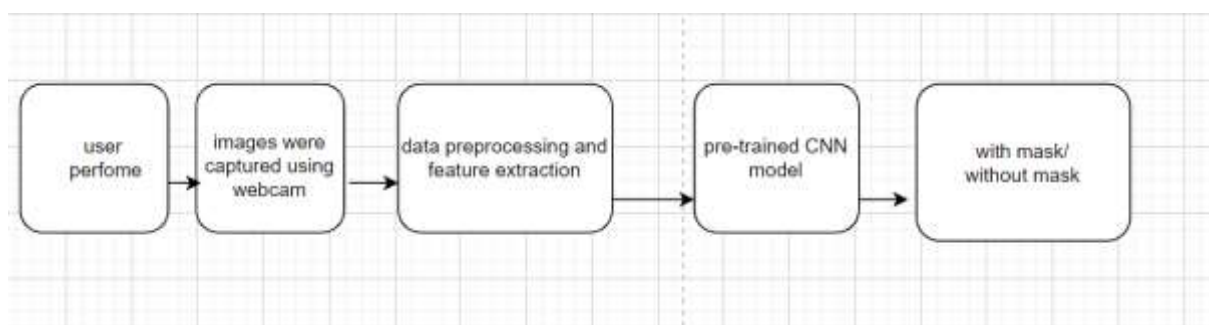


Figure: 4.4 proposed system for face mask detection

3.9 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects work together. this diagram is used by software developers and business professionals to understand requirements for a new system or a existing process.

- Represents the details of UML use case.
- Model the logic of a sophisticated procedure, function or operations.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existinf or future scenario.

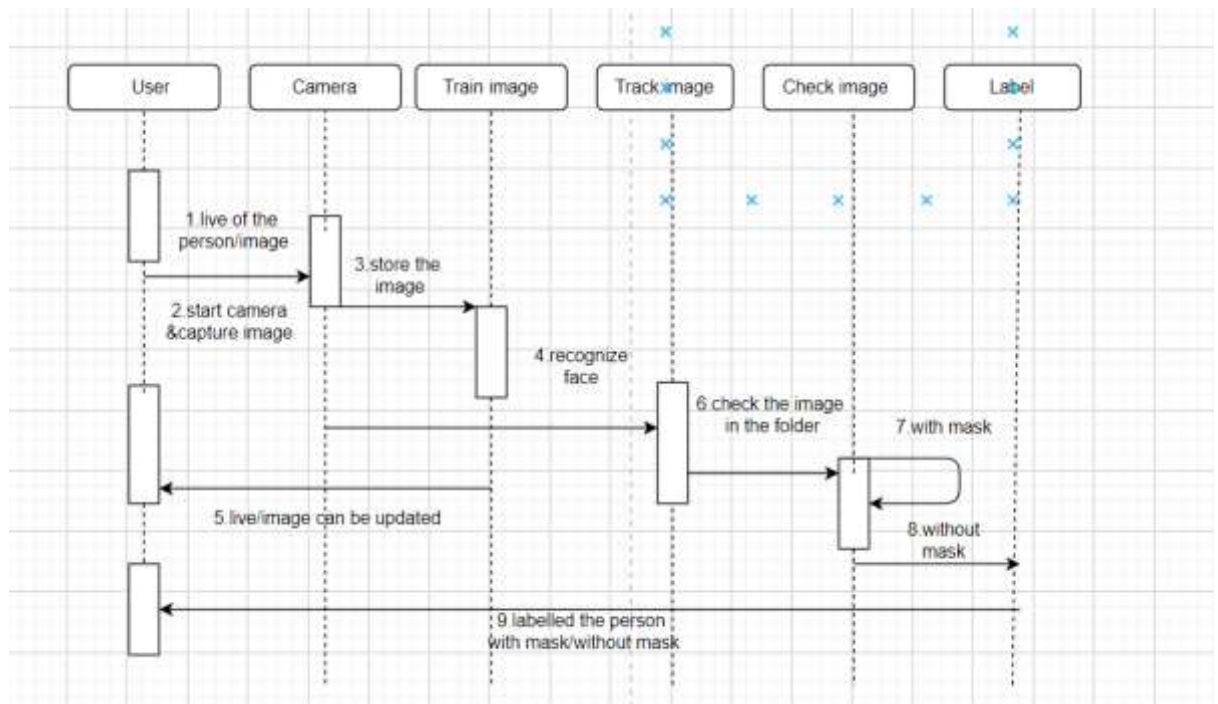


Figure 4.5 Sequence Diagram

3.10 ACTIVITY DIAGRAM

- An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system.
- An activity diagram portrays the control flow from a start point to a finish point, showing the various decision paths that exist while the activity is being executed.

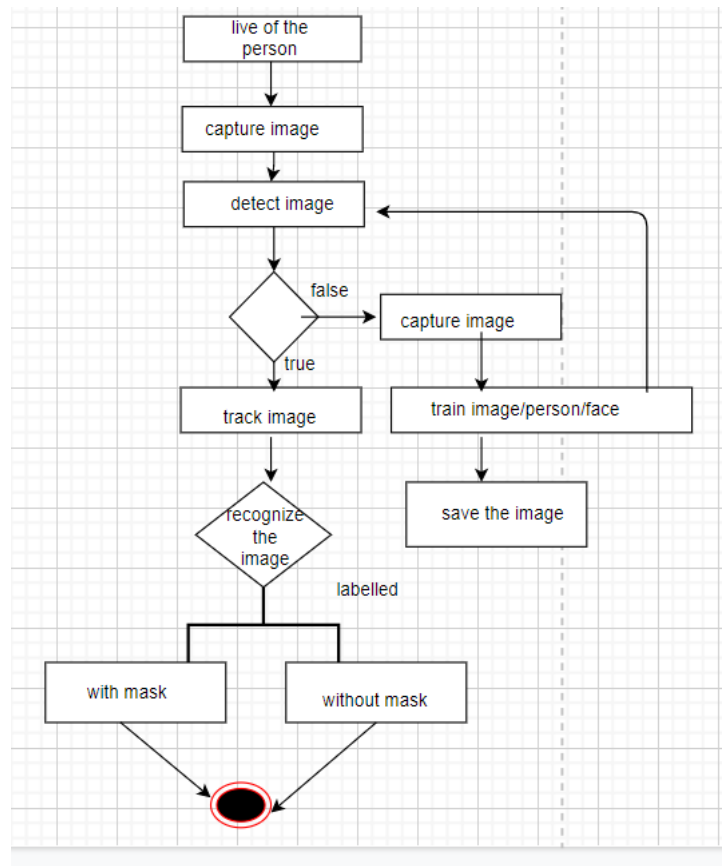


Figure 4.6 Activity Diagram

3.11 DATAFLOW DIAGRAM

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

It shows how data enters and leaves the system, what changes the information, and where data is stored. A graphical tool for defining and analyzing minute data with an active or automated system, including process, data stores, and system delays. Data Flow Data is a key and basic tool for the architecture of

DFDs are a model of the proposed system. They should indicate the requirements on which the new system should be built in a clear and direct manner. This is used as a basis for building chart structure plans over time during the design process.

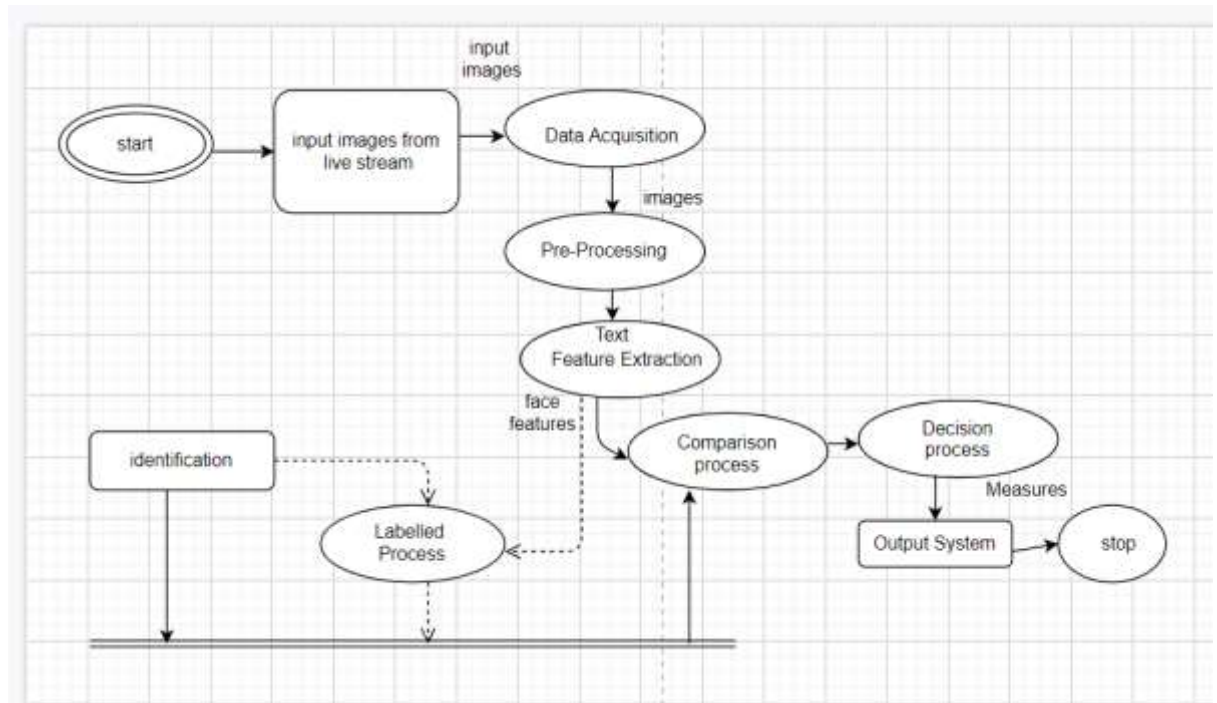


Figure 4.7 Dataflow Diagram

3.12 PROPOSED METHODOLOGY

The proposed methodology detects the person whether he/she is wearing a mask or not. And this model can send a message to the unmasked person's mail or phone. Here we are using CNNV2 algorithm to implement this project. This model is only used for the registered database which contains the details of the working employee of a particular organization. Firstly it will capture the images of a person from a video or it can also take an image this will be considered as input to the model. After capturing the image the proposed method will detect whether the person is wearing a mask or not.

3.13 PROPOSED METHODOLOGY MODULES

- Data collecting
- Pre-processing

- Splitting the data
- Building the model
- Testing the model
- Implementing the model

3.13.1 Data Collecting:

In this module we will collect the data from the CCTV camera. Here the data means the images captured by the camera. And here we are collecting another data which is called the database of the particular organization where this model is going to be used. We use this database for comparing the captured images to detect whether the person is wearing a mask or not.

3.13.2 Pre-Processing :

After completion of the above module we will come to this pre-processing technique. It is an operation on images which are taken as input. This technique is also called as data cleansing technique which includes outlier detection, removes the unwanted or noisy data. This process doesn't increase the image information content but it resolves the undesired distortions.

3.13.3 Splitting the data

Here the data we are considering is splitted into two categories data with masked face and the data with unmasked face. This module is used to compare with database that is registered in the organization. We perform only the operations like sending the messages only the unmasked dataset.

3.13.4 Building the model :

We will build our model by using CNN algorithm. This algorithm works by extracting features from images. It is convolutional neural network algorithm. In this step we build our sequential CNN model with various layers such as conv2D, Maxpooling2D, Flatten, Dropout and Dense.

3.13.5 Training the model :

After building our model, let us create the ‘train generator’ and ‘validation generator’ This step is the main step where we fit our images in the training set and the test set to our sequential model we built using keras library.

3.13.6 Implementing the model :

After building the model, we label two probabilities for our results. ‘0’ as ‘without_mask’ and ‘1’ as ‘with_mask’. We are also setting the boundary rectangle coursing the RGB values.

3.14 TESTING

Software Testing is the process used to attain the quality of computer software. This includes, but is not limited to, the process of executing a program or application with the intent of finding software bugs. Software testing is analysis of the software towards requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of validation and verification.

The usage of the word testing in relation to deep learning models is primarily used for testing the model performance in terms of accuracy/precision of the model. There are many approaches to software testing. Techniques such as black box and white box testing would thus, apply to deep learning models as well for performing quality control checks on deep learning models. Although testing can be done manually, by a user clicking buttons or typing the input, it should be performed automatically by designing software programs that test the application under test.

Verification

Verification is the process of confirming if the software is meeting the business requirements, and is developed the proper specifications and methodologies. Have we built the software right.

Validation

Validation is the process of examining whether or not the software satisfies the user requirements. It can be done at the end of the SDLC. If the software matches requirements for which it is made it is validated. Have we built the right software.

3.15 TYPES OF TESTING

Software testing methods are clasified into black box testing and white box testing. These two testing methods are used to describe the point of view that a test engineer takes when designing test cases.

3.15.1Black box testing

Black box testing is a software testing technique in which functionality of the software under test is tested without looking at the internal code structure, implementation details and knowledge of internal parts of the software and is usually done by testers team. In this testing, they mainlyconcentrate on inputs and outputs of the software system without worrying about internal structure of the software program.

3.15.2 White box testing

White box testing however is when the tester has access to the internal data structures, code and algorithms. These testing method includes creating tests to cause all statements in the program to be executed at least once. Other examples of white box testing are mutation testing and fault injection methods.

3.16 Test cases

Test cases for this project are tabulated and are shown in the below table. A test case is a set of actions executed to verify a particular feature or functionality of your software application. A test case contains test steps, test data, precondition, postcondition, developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using

which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

Test cases	Description	Result
1.	Face detected with mask	Accept
2.	Face detected without mask	Accept

3.17 Conclusion

In this chapter the testing is performed to check whether the accuracy of the system is increased or not. From the above table one can conclude that our system will have two test cases one is with mask and without mask. After testing these test cases finally we conclude that there is an increase in the accuracy.

CHAPTER-4

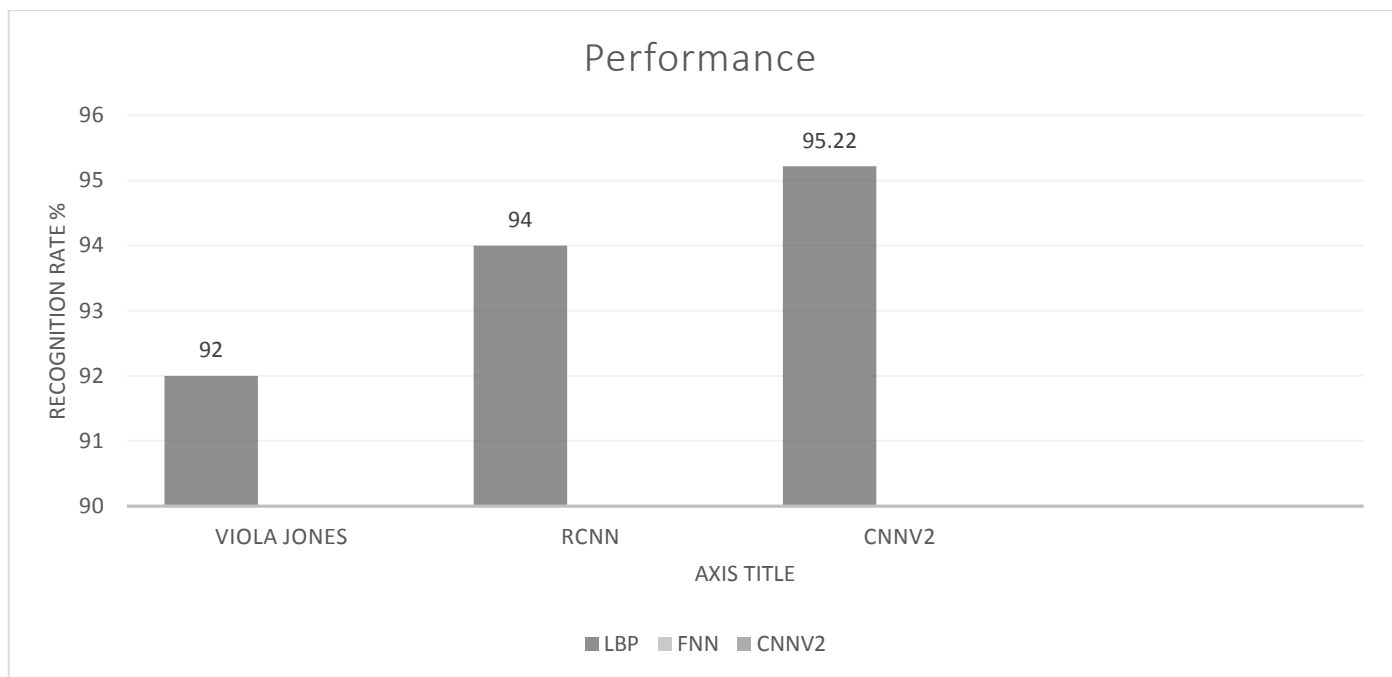
RESULTS AND DISCUSSION

4.1 RESULTS

For detecting the masked and unmasked person we are using CNNV2 and Haar-based Cascade networking algorithms. By using this convolutional neural networks we classify our data that is retrived from the camera is classified into two categories which are '0' as without mask and '1' as with mask. We have taken the dataset and extract the features manually which contribute for the prediction of the output. After training the dataset and applying convolutional neural networks. We are also providing the boundary rectangle around the image that is detected.

4.2 RESULTS COMPARISION :

The proposed work gives the recognition rate upto 95% which is greater than the previous works. In the year 2016 by using viola jones algorithm they got recognition score of 92%. But here we are using CNNV2 which is convolutional neural networks along with Haar-based cascade networking algorithm we got an recognition rate upto 95.22%. The accuracy also increased when compared to the previous inventions on these project. The accuracy rate of our project is shown on the above boundry box which is in red and green colured.



4.3APPLICATION

This proposed project is applicable to identify or detect the unmasked person in over crowded areas also. These project can be used in schools, offices, airports, and where the registered database along with the details of each and every person there our project is applicable. Due to this pandemic wearing a mask is one of the responsibility of every individual to protect there lives as well as the others also. So, detecting of unmasked person is a big challenge in the present situation. By, using our model we detect the person who is not wearing the mask and send them a warning message to alert.

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1 CONCLUSION

Global pandemic COVID-19 circumstances has emerged a dangerous disease in all over the world. Wearing a face mask will help to prevent the spread of infection and prevent the individual from contacting with any infectious germs. Face mask is a challenging task in today's situation to detect whether a person is wearing a mask or not. Here we used Haar-Cascade algorithm along with CNNV2 collating with this classifier it produces a high recognition rate even with varying expressions, efficient feature selection, and low assortment of false positive features. Haar feature based cascade CNNV2 classifier system utilizes only 200 features out of 6000 features to yield a recognition rate of 98.5%. According to this motivation we demand mask detection as a unique and public health service system during the pandemic.

5.2 FUTURE ENHANCEMENT

Here we created the MFDD, RMFRD, and SMFRD datasets to implement our project. This project will send a mail to person who is detected as a unmasked. We are using only the mail id to send a message in future work may use mobile phone number to send a remainder or warning message to the unmasked person.

REFERENCES

- [1] Zhongyuanwang, Nanxi Wang, Heling Chen, Guangcheng Wang. "Masked Face Recognition Dataset and Application". National Engineering Research Center for Multimedia Software, School of computer science, 23rd March 2020.
- [2] Vinitha, Velantina. COVID-19 Facemask detection with deep learning and computer vision". International Research Journal of Engineering and Technology. Aug 2020.
- [3] Amrit Kumar Bhadani, Anurag Sinha "A Face Mask Detector Using Machine Learning and Image processing Techniques". Engineering Science and Technology ,an International Journal. Mar 2020.
- [4] Nizam uddin, Kamran Javed, Seho Bae, and Juneho Yi. "A Novel GAN-Based Network for Unmasking of Masked face". College of Information and Communication Engineering. Published by IEEE, March 2nd 2020.
- [5] Yuzhen Chen, Menghan Hu, Jian Zhang, Qingli Li Senior Member IEEE. "Face Mask Assistant: Detection of Face Mask Service Stage Based on Mobile Phone". IEEE Sensors Journal VOL 21 NO 9. 1ST May, 2021.
- [6] Md. Sabbir Ejaz, and Md. Rabiul Islam. "Masked Face Recognition Using Convolutional Neural Network". International Conference on Sustainable Technologies for Industry. IEEE December 2019.
- [7] Aniruddha Srinivas Joshi, Shreyas Srinivas Joshi, Goutham Kanahasabai, Rudraksh Kapil, Savvasachi Gupta. "Deep Learning Framework to Detect Face Masks from Video Footage". 12th International Conference on Computational Intelligence and Communication Networks. December 18, 2020.
- [8] Vasantha Kumari, Mohd. Umar Peerzade, Subhralina Nayak, Bhagya, Padmanabhan. "Automatic Facemask Detection using MobileNetV2". International Research Journal of Engineering and Technology (IRJET). Feb 2021.

- [9]TejalNerpagar,SakshiJunnare,JahnaviRaut,Aartishah,Prof.P.C.Patil.”Face Mask Recognition using Machine Learning”.International Journal of Innovative Research in Technology.December 2020.
- [10] SaiyamJain,MayankGoyal,DeepakSingh,AbhishekAswal,UpasanaJoshi.”Face Mask Detection using Machine Learning and Deep Learning”.InternationalReasearch Journal of Engineering and Technology(IRJET).Jan 2021.
- [11]Madhura Inamdar · NinadMehendale. “Real-time face mask identification using Facemasknet deep learning network”.
- [12]Bosheng Qin and Dongxiao Li” Identifying Facemask-Wearing Condition Using Image Super-Resolution with Classification Network to Prevent COVID-19” 14 September 2020.
- [13]PreetiNagrath , Rachna Jain, Agam Madan, Rohan Arora, Piyush Kataria, Jude Hemanth” SSDMNv2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2”.2021.
- [14] DeviraAnggi Maharani, CarmadiMachbub, PranotoHidayaRusmin, LenniYulianti “Improving the capability of real-time face masked recognition using cosine distance”.2020.
- [15] Chao Qia, Lei Yang “Face recognition in the scene of wearing a mask”.2020.
- [16] Qi Hong¹ ,Zhongyuan Wang^{1,2}, Zheng He¹ , Nanxi Wang¹ , Xin Tian¹ , Tao Lu³ “Masked Face Recognition with Identification Association”.2020.
- [17] Sunil Singh, Umang Ahuja,Munish Kumar, Krishan Kuma, Monika Sachdeva“Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment”
- [18] Amit chavda, Jason Dsouza, Sumeet Badgujar, Ankit damani Masked face recognition with identification association.
- [19] Chao Qia, Lei Yang Face recognition with facial mask application and neural network.2020
- [20]]PreetiNagrath, Madhura InamdarFace mask detection using MobileNet and GlobalPooling Block.20

APPENDIX A

SOURCE CODE

Importing libraries required for implementation

```
# 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 argparse

import os


# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()

ap.add_argument("-d", "--dataset", required=True,
                help="path to input dataset")

ap.add_argument("-p", "--plot", type=str,
                default="plot.png", help="path to output loss/accuracy
                plot")

ap.add_argument("-m", "--model",
                type=str, default="mask_detector.model",
                help="path to output face mask detector model")

args = vars(ap.parse_args())

```

```

# initialize the initial learning rate, number of epochs
to train for,

# and batch size

INIT_LR = 1e-4

EPOCHS = 20

BS = 32


# 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...")

imagePaths = list(paths.list_images(args["dataset"]))

data = []

labels = []


# loop over the image paths
for imagePath in imagePaths:

# extract the class label from the filename

label = imagePath.split(os.path.sep)[-2]

# load the input image (224x224) and preprocess it

image = load_img(imagePath, target_size=(224, 224))

image = img_to_array(image)

```

```

image = preprocess_input(image)

# update the data and labels lists, respectively
data.append(image)

    labels.append(label)

# convert the data and labels to NumPy arrays
data = np.array(data, dtype="float32")
labels = np.array(labels)

# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

# partition the data into training and testing splits
using 75% of

# the data for training and the remaining 25% for testing
(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 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(args["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(args["plot"])
```

IMPLEMENTATION CODE

```
# 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 argparse
```

```
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, (300, 300),
```

```
(104.0, 177.0, 123.0))
```

```
# pass the blob through the network and obtain the face
detections
```

```
faceNet.setInput(blob)
```

```
detections = faceNet.forward()
```

```
# 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 > args["confidence"]:

# compute the (x, y)-coordinates of the bounding box
forthe 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]

if face.any():

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)

# construct the argument parser and parse the arguments

ap = argparse.ArgumentParser()

ap.add_argument("-f", "--face", type=str,
default="face_detector",
help="path to face detector model directory")

ap.add_argument("-m", "--model", type=str,
default="mask_detector.model",
help="path to trained face mask detector model")

ap.add_argument("-c", "--confidence", type=float,
default=0.5,
help="minimum probability to filter weak detections")

```

```

args = vars(ap.parse_args())

# load our serialized face detector model from disk
print("[INFO] loading face detector model...")

prototxtPath = os.path.sep.join([args["face"],
    "deploy.prototxt"])

weightsPath = os.path.sep.join([args["face"],
    "res10_300x300_ssd_iter_140000.caffemodel"])

faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
print("[INFO] loading face mask detector model...")

maskNet = load_model(args["model"])

# initialize the video stream and allow the camera sensor
to warm up

print("[INFO] starting video stream...")

vs = VideoStream(src=0).start()

time.sleep(2.0)

# 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 aface 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()
```

APPENDIX B

SCREENSHOTS

B1 Python 3.7.4 setup

We have chosen python to perform our implementations work. For this purpose, we have installed python version i.e. python 3.7.4. To

download and use python, we can download it from the website www.python.org. Next we have to run the setup steps and install it. We also have to setup the path in environment variables

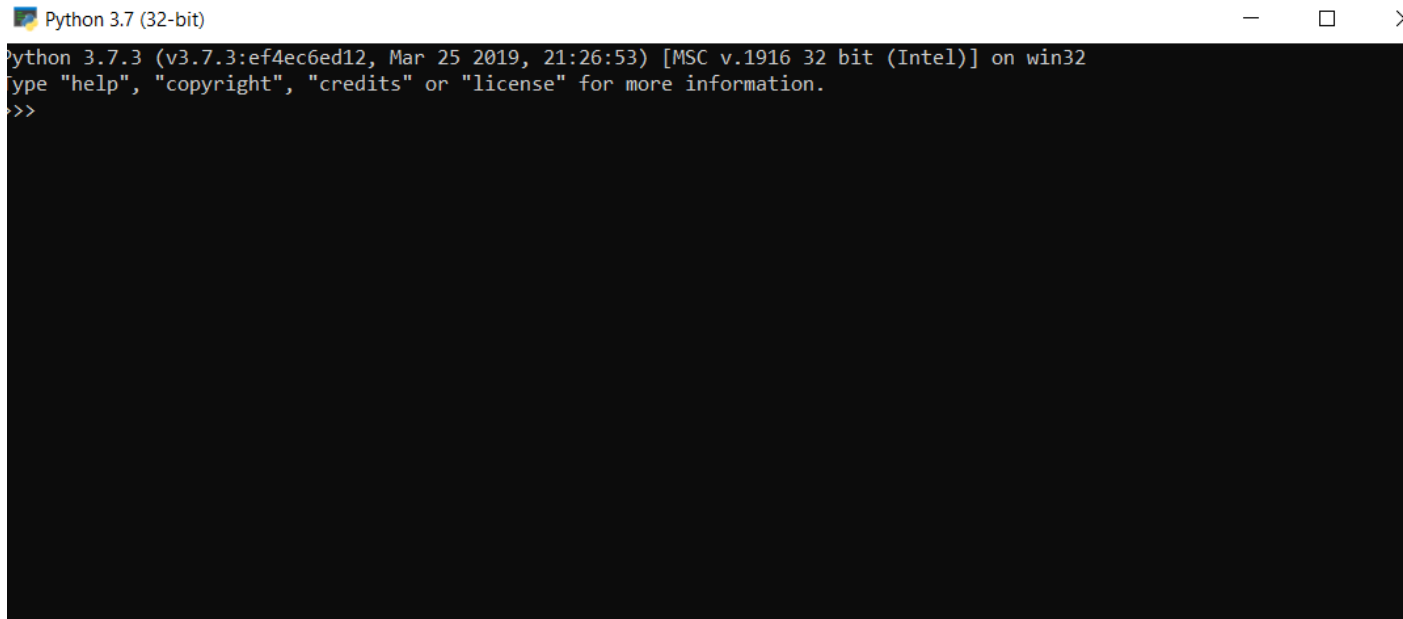


Fig B1 python 3.7.4 Setup

B2 Installation of keras

Tensorflow library is installed using conda install tensorflow command.

```
(base) C:\Users\91789>conda --version
conda 4.10.1

(base) C:\Users\91789>python --version
Python 3.8.8

(base) C:\Users\91789>conda install keras
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: failed with repodata from current_repodata.json, will retry with next
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: C:\Users\91789\anaconda3

added / updated specs:
- keras

The following packages will be downloaded:
```

package	build	
_tfflow_select-2.3.0	eigen	4 KB
absl-py-0.13.0	py38haa95532_0	175 KB
astor-0.8.1	py38haa95532_0	47 KB
astunparse-1.6.3	py_0	17 KB
blinker-1.4	py38haa95532_0	23 KB
cachetools-4.2.2	pyhd3eb1b0_0	13 KB
conda-4.10.3	py38haa95532_0	2.9 MB
coverage-5.5	py38h2bbff1b_2	272 KB
gast-0.4.0	py_0	15 KB
google-auth-1.21.3	py_0	57 KB
google-auth-oauthlib-0.4.4	pyhd3eb1b0_0	18 KB
google-pasta-0.2.0	py_0	46 KB
grpcio-1.36.1	py38hc60d5dd_1	1.7 MB
keras-2.4.3	0	12 KB
keras-applications-1.0.8	py_1	29 KB
keras-base-2.4.3	py_0	37 KB
keras-preprocessing-1.1.2	pyhd3eb1b0_0	35 KB
libprotobuf-3.14.0	h23ce68f_0	1.9 MB
markdown-3.3.4	py38haa95532_0	144 KB
oauthlib-3.1.0	py_0	91 KB
opt_einsum-3.3.0	pyhd3eb1b0_1	57 KB
protobuf-3.14.0	py38hd77b12b_1	242 KB
pyasn1-0.4.8	py_0	57 KB
pyasn1-modules-0.2.8	py_0	72 KB
pyjwt-1.7.1	py38_0	48 KB

B3 Installation of opencv

Opencv module can be installed using conda install opencv command

```
(base) C:\Users\VL788>conda install opencv
Collecting package metadata (current_repodata.json): done
Solving environment: done

# Package Plan #

environment location: C:\Users\VL788\anaconda3

added / updated specs:
- opencv

The following packages will be downloaded:



| package         | size           |
|-----------------|----------------|
| libopencv-4.9.1 | 35.5 MB        |
| opencv-4.9.1    | 22 MB          |
| py-opencv-4.9.1 | 1.5 MB         |
| <b>Total:</b>   | <b>39.1 MB</b> |



The following 0.0 packages will be downloaded:

libopencv      pkg/main/win-64::libopencv-4.9.1-00000110_0
opencv         pkg/main/win-64::opencv-4.9.1-00000110_0
py-opencv      pkg/main/win-64::py-opencv-4.9.1-00000110_0

Proceed [y/n]? y

Downloading and Extracting Packages:
libopencv-4.9.1      35.5 MB | ##### 100%
py-opencv-4.9.1      1.5 MB | ##### 100%
opencv-4.9.1         22 MB | ##### 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(base) C:\Users\VL788>
```

B4 Installation of Numpy

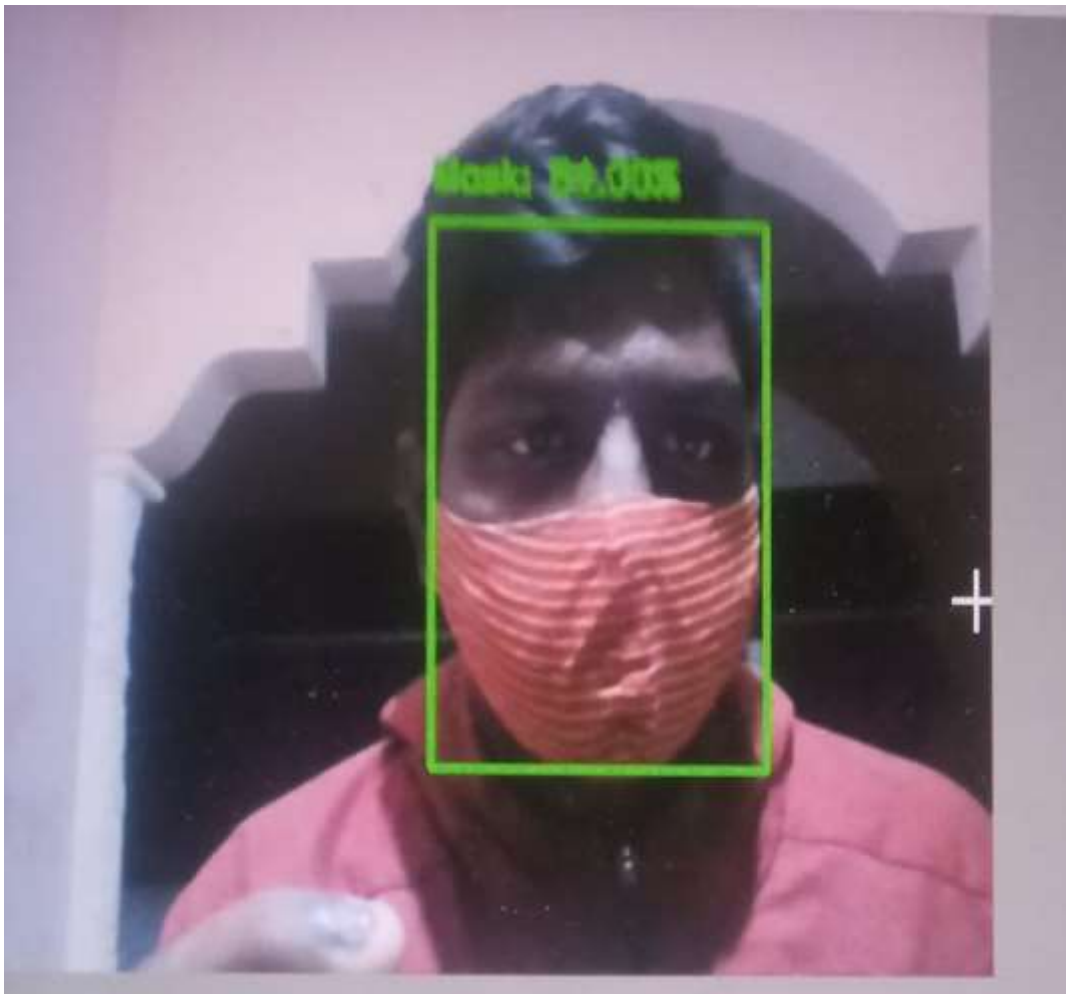
```
(base) C:\Users\VL788>conda install numpy
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.
```

B6 Output Screen shots



Without Mask



With Mask



Without Mask

 Frame



With Mask

