COVID-19 MASK DETECTION USING DEEP LEARNING

A PROJECT REPORT

Submitted by

DINAHAR M [REGISTER NO:211417104053]

ANIRUDH SUBRAMANYAM [REGISTER NO:211417104068]

GOKELNATH GG [REGISTER NO :211417104018]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123. ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2021

BONAFIDE CERTIFICATE

Certified that this project report "**COVID-19 FACE MASK DETECTION .**" is the bonafide work of "**DINAHAR. M(2017PECS383), GOKELNATH GG(2017PECCS393), ANIRUDH S(2017PECCS362)**" who carried out the project work under my supervision.

SIGNATURE

Dr.S.MURUGAVALLI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
COLLEGE, NAZARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Mr. M.MOHAN,
ASSISTANT PROFESSOR,
DEPARTMENT OF CSE,
PANIMALAR ENGINEERING
NAZARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna

University Project Viva-Voce Examination held on xx.04.2021

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Thiru.C.SAKTHIKUMAR,M.E.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide Mr.N.Sathish** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

DINAHAR. M,

GOKELNATH GG,

ANIRUDH S

ABSTRACT

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

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Overview

Face recognition techniques, the most important means of identification, have nearly failed, which has brought huge dilemmas to authentication applications that rely on face recognition, such as community entry and exit, face access control, face attendance, face gates at train stations, face authentication based mobile payment, face recognition based social security investigation, etc. In particular, in the public security check like railway stations, the gates based on traditional face recognition systems can not effectively recognize the masked faces, but removing masks for passing authentication will increase the risk of virus infection. Because the COVID-19 virus can be spread through contact, the unlocking systems based on passwords or fingerprints are unsafe. It is much safer through face recognition without touching, but the existing face recognition solutions are no longer reliable when wearing a mask. To solve above -mentioned difficulties, it is necessary to improve the existing face recognition approaches that heavily rely on all facial feature points, so that identity verification can still be performed reliably in the case of incompletely exposed faces.

1.2 Problem Definition

Regarding the current popular face masks, there are two closely related and different applications, namely,facial mask detection task and masked face recognition task. Face mask detection tasks need to identify whether a person wears a mask as required. Masked face recognition tasks need to identify the specific identity of a person with a mask. Each task has different requirements for the dataset. The former only needs masked face image samples, but the latter requires a dataset which contains multiple face images of the same

subject with and without a mask. Relatively, datasets used for the face recognition task are more difficult to construct.

In order to handle masked face recognition task, this paper proposes three types of masked face datasets, including Masked Face Detection Dataset (MFDD), Real-world Masked Face Recognition Dataset (RMFRD) and Simulated Masked Face Recognition Dataset (SMFRD).

MFDD: The source of MFDD mainly includes two parts: (a) Some of the samples are from related research. The other part of MFDD is crawled from the Internet. We further label the crawled face images, performing annotations such as whether the face wears a mask and the position coordinates of the masked faces. This built dataset contains 24,771 masked face images. MFDD dataset can be used to train an accurate masked face detection model, which serves for the subsequent masked face recognition task. Additionally, it can also be used to determine whether a person is wearing a mask, as it is illegal without wearing a mask during the coronavirus epidemic.

2. LITERATURE SURVEY

In December 2019, adults in Wuhan, capital city of Hubei province and a major transportation hub of China started presenting to local hospitals with severe pneumonia of unknown cause. Many of the initial cases had a common exposure to the Huanan wholesale seafood market that also traded live animals. The surveillance system (put into place after the SARS outbreak) was activated and respiratory samples of patients were sent to reference labs for etiologic investigations.

On December 31st 2019, China notified the outbreak to the World

Health Organization and on 1st January the Huanan seafood market was closed. On 7th January the virus was identified as a coronavirus that had >95% homology with the bat coronavirus and > 70% similarity with the SARS CoV. Environmental samples from the Huanan seafood market also tested positive, signifying that the virus originated from there . The number of cases started increasing exponentially, some of which did not have exposure to the live animal market, suggestive of the fact that human-to-human transmission was occurring.

The first fatal case was reported on 11th Jan 2020. The massive migration of Chinese during the Chinese New Year fuelled the epidemic. Cases in other provinces of China, other countries (Thailand, Japan and South Korea in quick succession) were reported in people who were returning from Wuhan. Transmission to healthcare workers caring for patients was described on 20th Jan, 2020.

By 23rd January, the 11 million population of Wuhan was placed under lock down with restrictions of entry and exit from the region.Soon this lockdown was extended to other cities of Hubei province. Cases of COVID-19 in countries outside China were reported in those with no history of travel to China suggesting that local human-to-human transmission was occurring in these countries. Airports in different countries including India put in screening mechanisms to detect symptomatic people returning from China and placed them in isolation and tested them for COVID-19. Soon it was apparent that the infection could be transmitted from asymptomatic people and also before onset of symptoms.Therefore, countries including India who evacuated their citizens from Wuhan through special flights or had travellers returning from China, placed all people symptomatic or otherwise in isolation for 14 days and tested them for the virus. Cases continued to increase exponentially and modeling studies reported an epidemic doubling time of 1.8 days. In fact on the 12th of February, China changed its definition of confirmed cases to include patients with negative/ pending molecular tests but with clinical, radiologic and epidemiologic features of

COVID-19 leading to an increase in cases by 15,000 in a single day.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

• Support Vector Machine

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).

• Discrete Wavelet Transform

Wavelet filters have been implemented as algorithms due to their simplicity, suitability and regularity for face recognition using a multiresolution approach. But they are outdated as they lack efficiency in resource and time management

3.2 PROPOSED SYSTEM

•Convolution Neural Network

A Convolutional Neural Network, also known as CNN or ConvNet, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be.

- **Caffe Models**

Deep networks are compositional models that are naturally represented as a collection of interconnected layers that work on chunks of data. Caffe defines a net layer-by-layer in its own model schema. The network defines the entire model bottom-to-top from input data to loss. Caffe is being used in academic research projects, startup prototypes, and even large-scale industrial applications in vision, speech, and multimedia. Yahoo! has also integrated caffe with Apache Spark to create CaffeOnSpark, a distributed deep learning framework.

3.3 REQUIREMENT ANALYSIS AND SPECIFICATION

The requirement engineering process of feasibility study, requirements elicitation and analysis, requirement specification, requirements validation and requirement management. Requirement elicitation and analysis is an iterative process that can be represented as a spiral of activities, namely requirements discovery, requirements classification and organization, requirement negotiation and requirements documentation.

1. Working Computer
2. Intel i7 11th gen and above
3. Arduino Chipset and equipments
4. 4gb+ DDR3 RAM
5. Camera with a 5mp sensor and above

6. Monitor
7. Adequate Storage space
8. Functional USB ports for additional sensor connections
9. Ethernet connection

3.3.1 INPUT REQUIREMENTS

1. Camera Module

   A USB Video Class (UVC) compliant video camera designed for teleconferencing applications on desktop or laptop. It's a reliable digital video device to transfer video data with HD quality through a high-speed USB interface in Full HD mode

2. Temperature Sensor module

   The TMP36 temperature sensor is an easy way to measure temperature using an Arduino! The sensor can measure a fairly wide range of temperature (-50°C to 125°C), is fairly precise (0.1°C resolution)

3.3.2 OUTPUT REQUIREMENTS

1. Screen
2. Physical motor modules if necessary

3.4 FEASIBILITY STUDY

A feasibility study is carried out to select the best system that meets performance requirements. The main aim of the feasibility study activity is to determine that it would be financially and technically feasible to develop the product.

3.4.1 TECHNICAL FEASIBILITY

This is concerned with specifying the software will successfully satisfy the user requirement. Open source and business-friendly and it is truly cross platform, easily deployed and highly extensible.

3.4.2 ECONOMIC FEASIBILITY

Economic analysis is the most frequently used technique for evaluating the
effectiveness of a proposed system. The enhancement of the existing system doesn't incur any kind of increase in the expenses. The Programming language chosen is python and the environment chosen is Kali linux or any linux environment as both are open source products it is economically feasible.

3.4.3 TECHNICAL FEASIBILITY

This is concerned with specifying the software will successfully satisfy the user requirement. Open source and business-friendly and it is truly cross platform, easily deployed and highly extensible.

3.4.4 ECONOMIC FEASIBILITY

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. The enhancement of the existing system doesn’t incur any kind of increase in the expenses. The Programming language chosen is python and the environment chosen is Kali linux or any linux environment as both are open source products it is economically feasible.

3.5 TECHNOLOGY STACK

- PYTHON

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included"

language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980's, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting and was discontinued with version 2.7.18 in 2020. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3.

Python consistently ranks as one of the most popular programming languages.

- ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository. It is available for Windows, macOS, and Linux.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and

use multiple environments to separate these different versions.

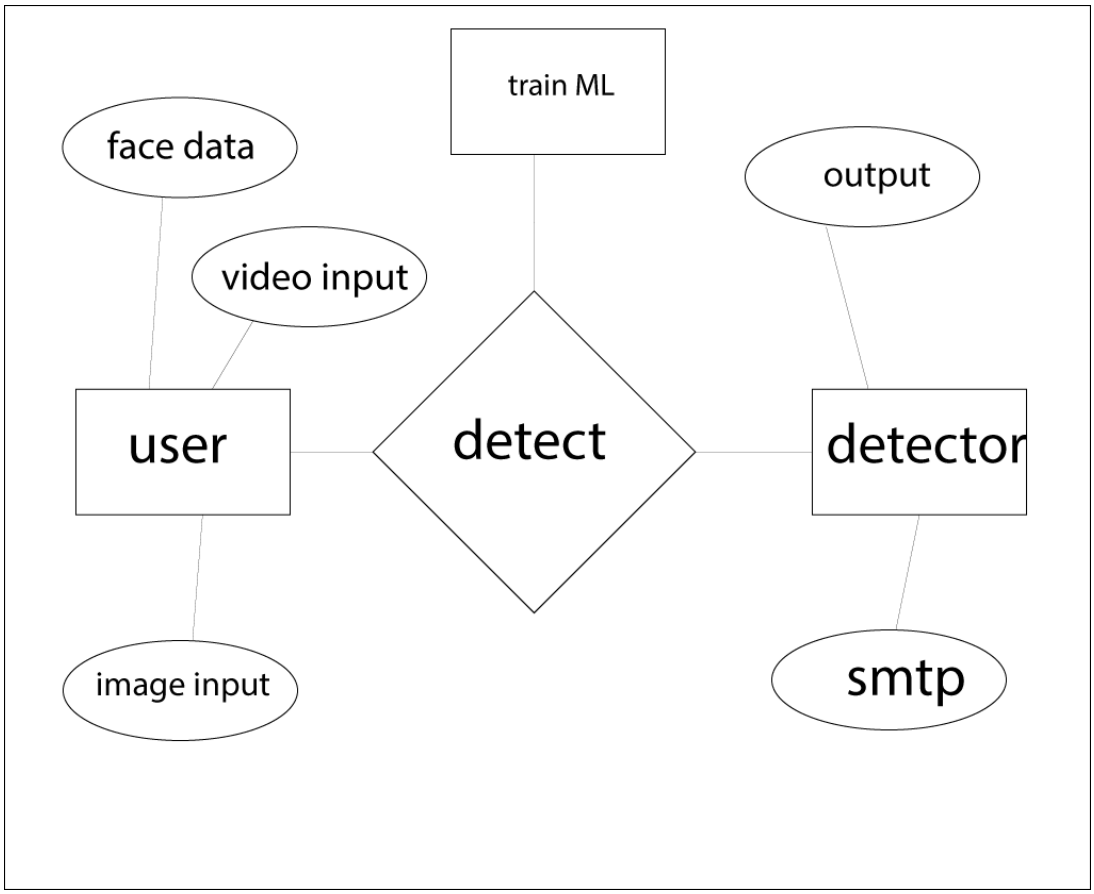
The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

4. SYSTEM DESIGN

4.1. ER diagram

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in a database, so by showing relationships among tables and their attributes, ER diagrams show the complete logical structure of a database.



4.2 Data dictionary

1. Masked Face Detection Dataset (MFDD)
2. Real-world Masked Face Recognition Dataset (RMFRD)
3. Simulated Masked Face Recognition Dataset (SMFRD)

4.3UML Diagrams

UML stands for Unified Modeling Language. It's a rich language to model software solutions, application structures, system behavior and business processes. There are 14 UML diagram types to help you model these behaviors. Unified Modeling Language™ (UML®) is a standard visual modeling language intended to be used for

- modeling business and similar processes,
- analysis, design, and implementation of software-based systems

UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.

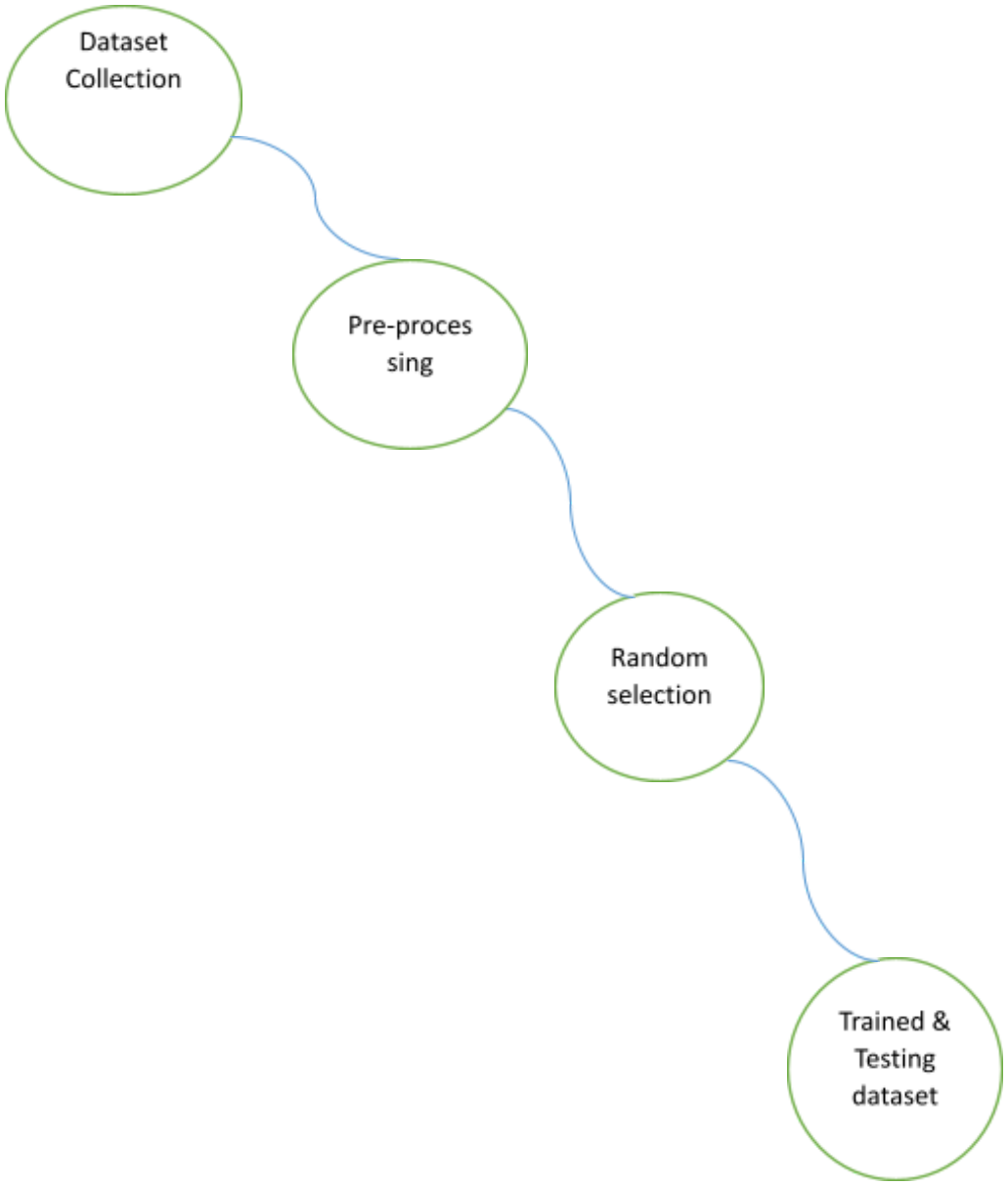
Specification explained that process:

- provides guidance as to the order of a team's activities,
- specifies what artifacts should be developed,
- directs the tasks of individual developers and the team as a whole, and
- offers criteria for monitoring and measuring a project's products and activities.

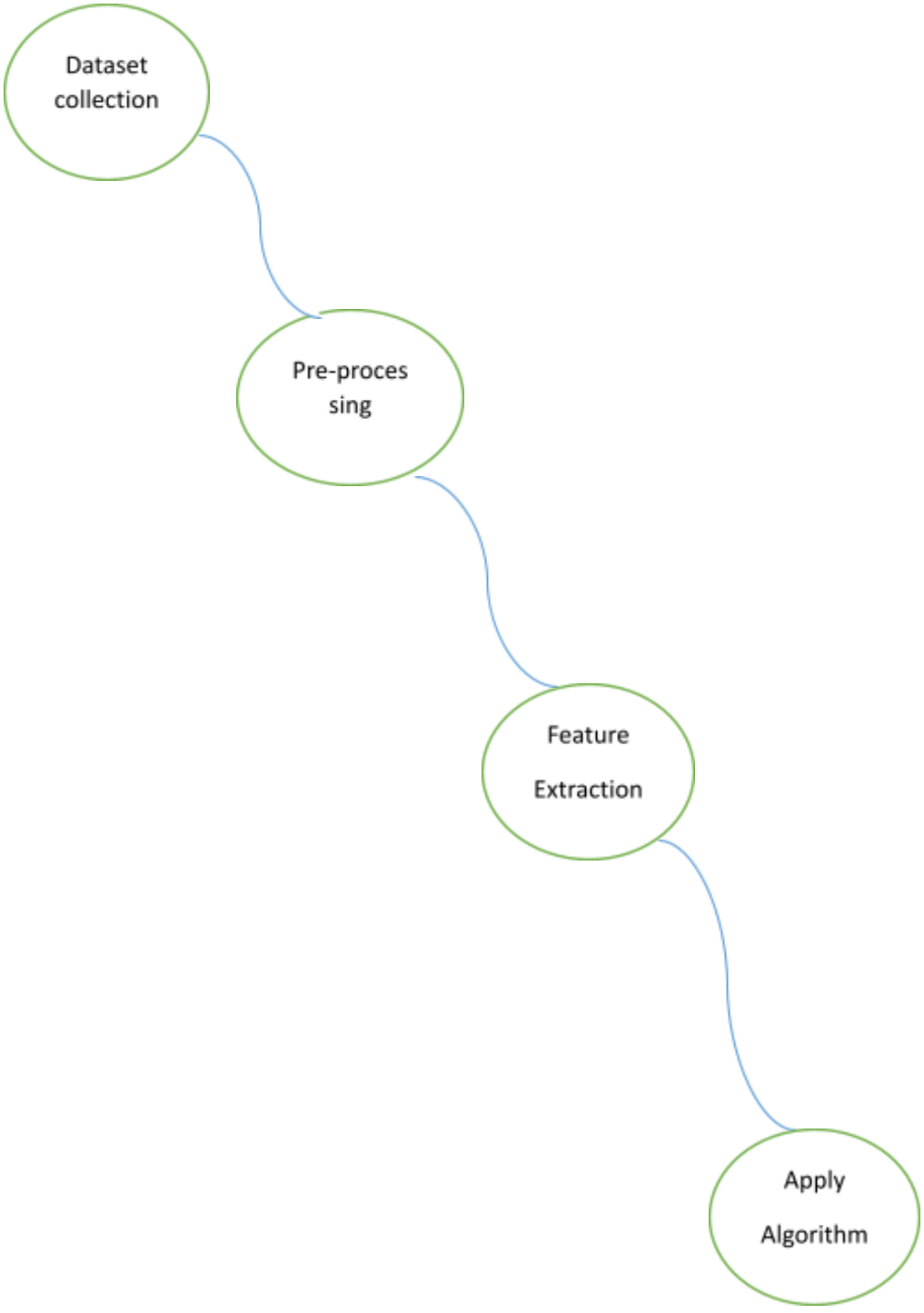
UML is intentionally process independent and could be applied in the context of different processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such a process is Rational Unified Process (RUP).UML is not complete, and it is not completely visual. Given some UML diagrams, we can't be sure to understand the depicted part or behavior of the system from the diagram alone. Some information could be intentionally omitted from the diagram, some information represented on the diagram could have different interpretations, and some concepts of UML have no graphical notation at all, so there is no way to depict those on diagrams. For example, semantics of multiplicity of actors and multiplicity of use cases on use case diagrams is not defined precisely in the UML specification and could mean either concurrent or successive usage of use cases.

4.4 DATA FLOW DIAGRAM

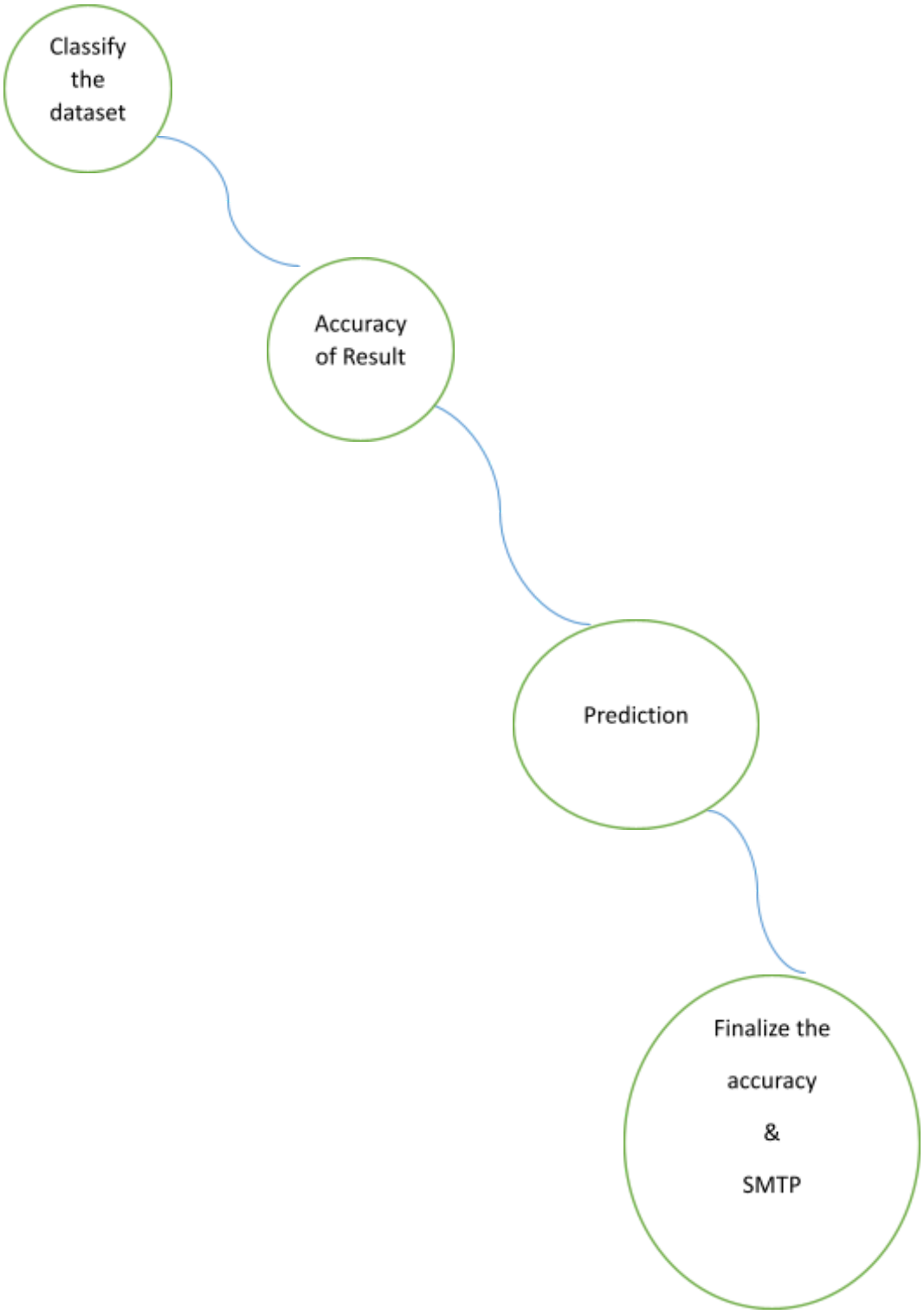
LEVEL 0



LEVEL 1

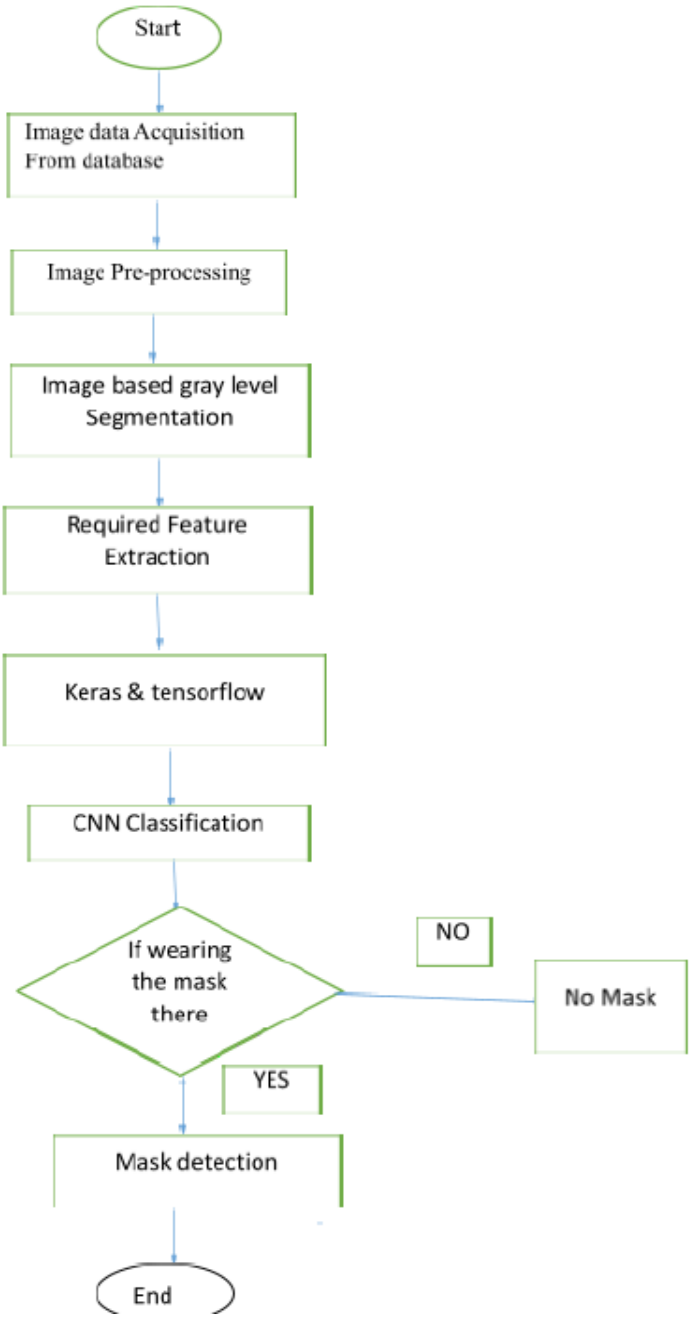

Dataset collection
Pre-proces sing
Feature Extraction
Apply Algorithm

LEVEL 2


Classify the dataset
Accuracy of Result
Prediction
Finalize the accuracy & SMTP

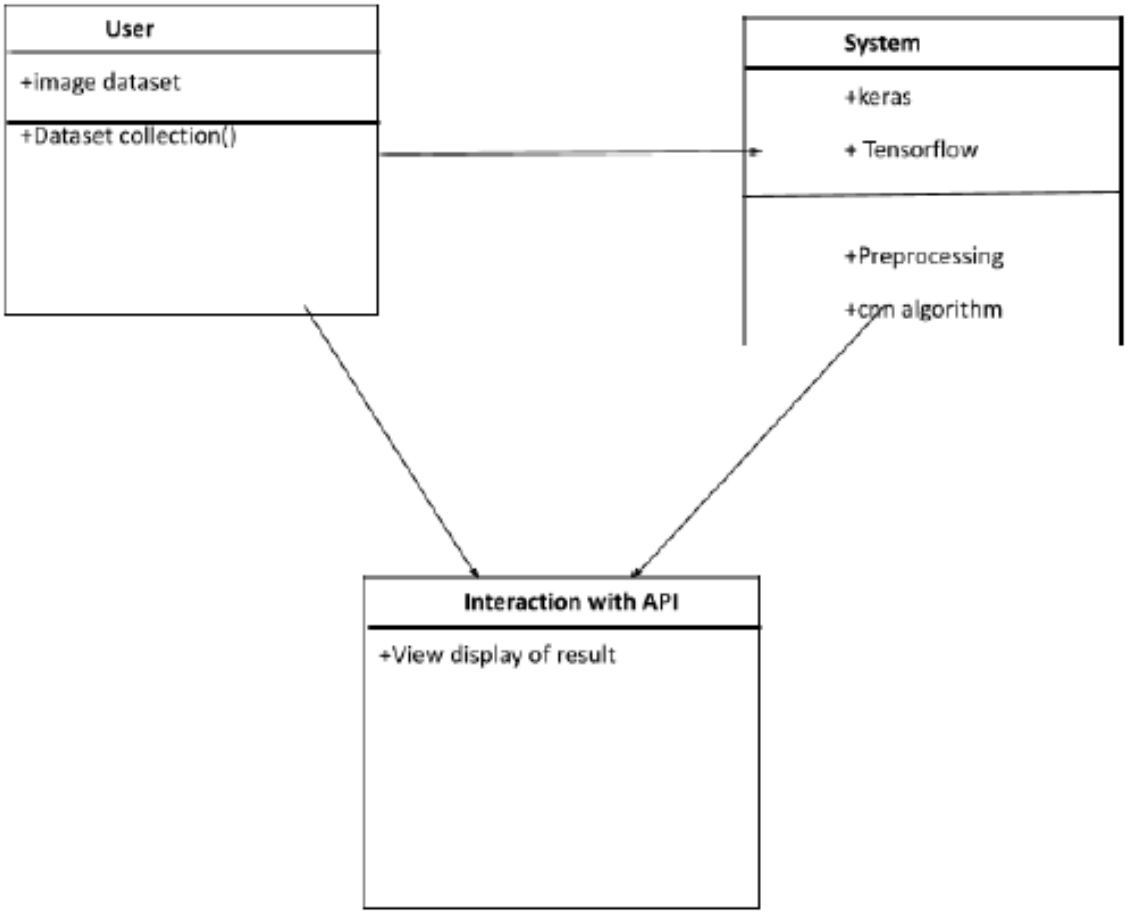
4.5FLOW CHART DIAGRAM

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams. Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence. They can range from simple, hand-drawn charts to comprehensive computer-drawn diagrams depicting multiple steps and routes. If we consider all the various forms of flowcharts, they are one of the most common diagrams on the planet, used by both technical and non-technical people in numerous fields. Flowcharts are sometimes called by more specialized names such as Process Flowchart, Process Map, Functional Flowchart, Business Process Mapping, Business Process Modeling and Notation (BPMN), or Process Flow Diagram (PFD). They are related to other popular diagrams, such as Data Flow Diagrams (DFDs) and Unified Modeling Language (UML) Activity Diagrams.


Start
Image data Acquisition From database
Image Pre-processing
Image based gray level Segmentation
Required Feature Extraction
Keras & tensorflow
CNN Classification
If wearing the mask there
NO
No Mask
YES
Mask detection
End

4.6CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram. The purpose of the class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

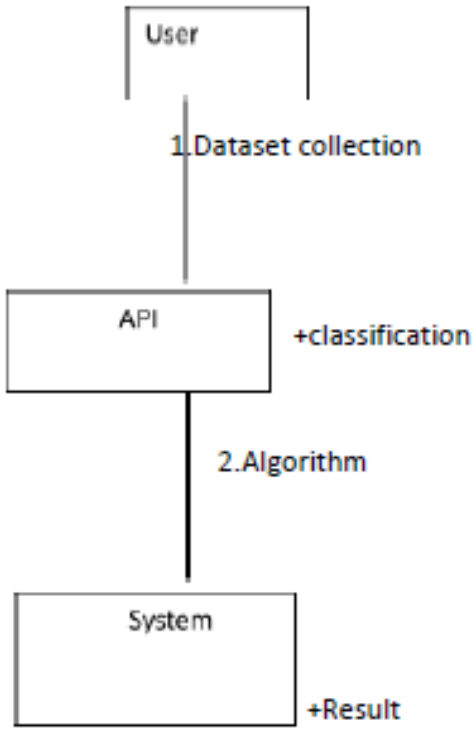


4.7COLLABORATION DIAGRAM

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray

the dynamic behavior of a particular use case and define the role of each object. Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction. A model is then built using the relationships between those elements. Several vendors offer software for creating and editing collaboration diagrams.

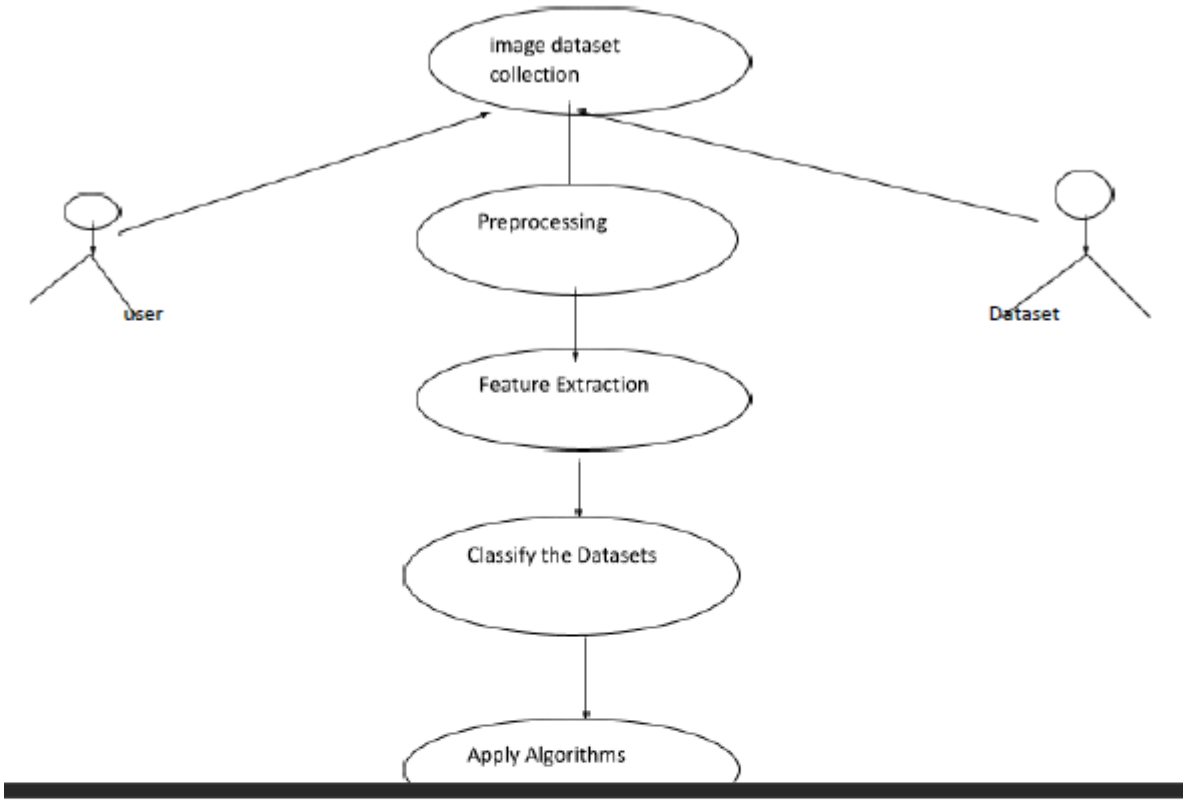
Collaboration Diagram:-

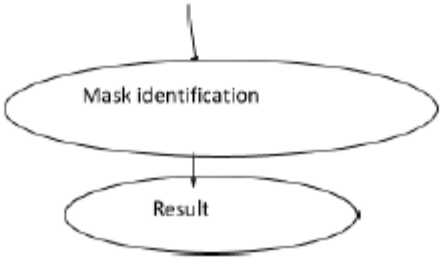


4.8USE CASE DIAGRAM

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

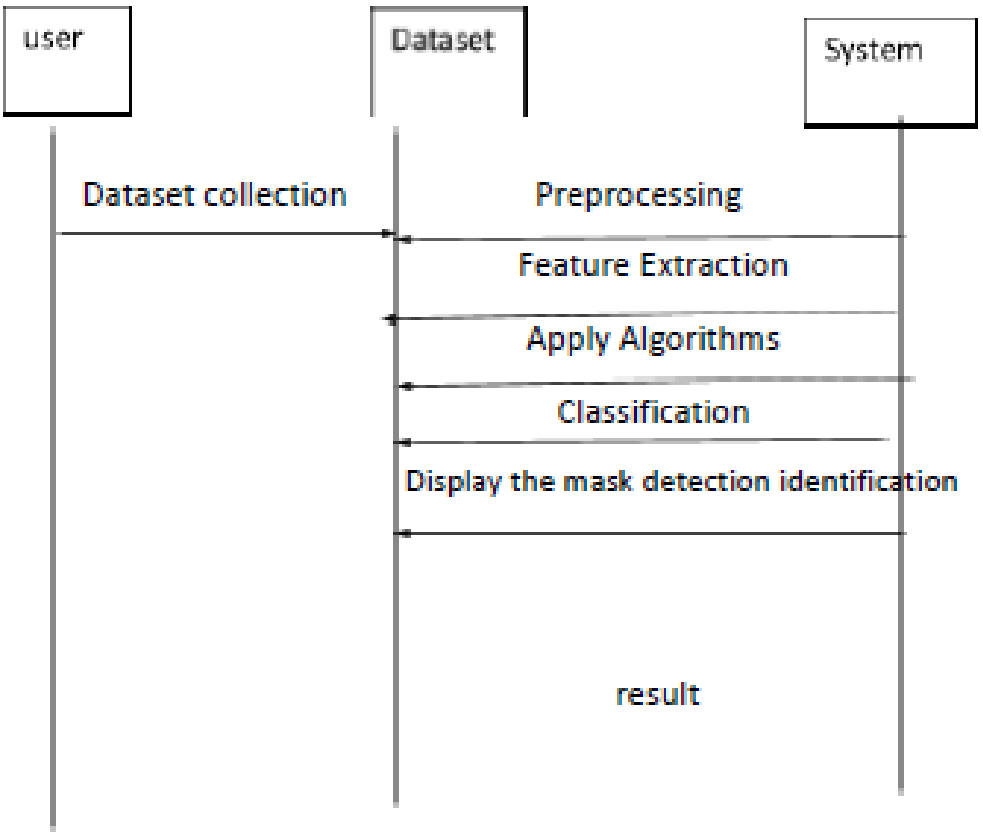
Use case Diagram:-



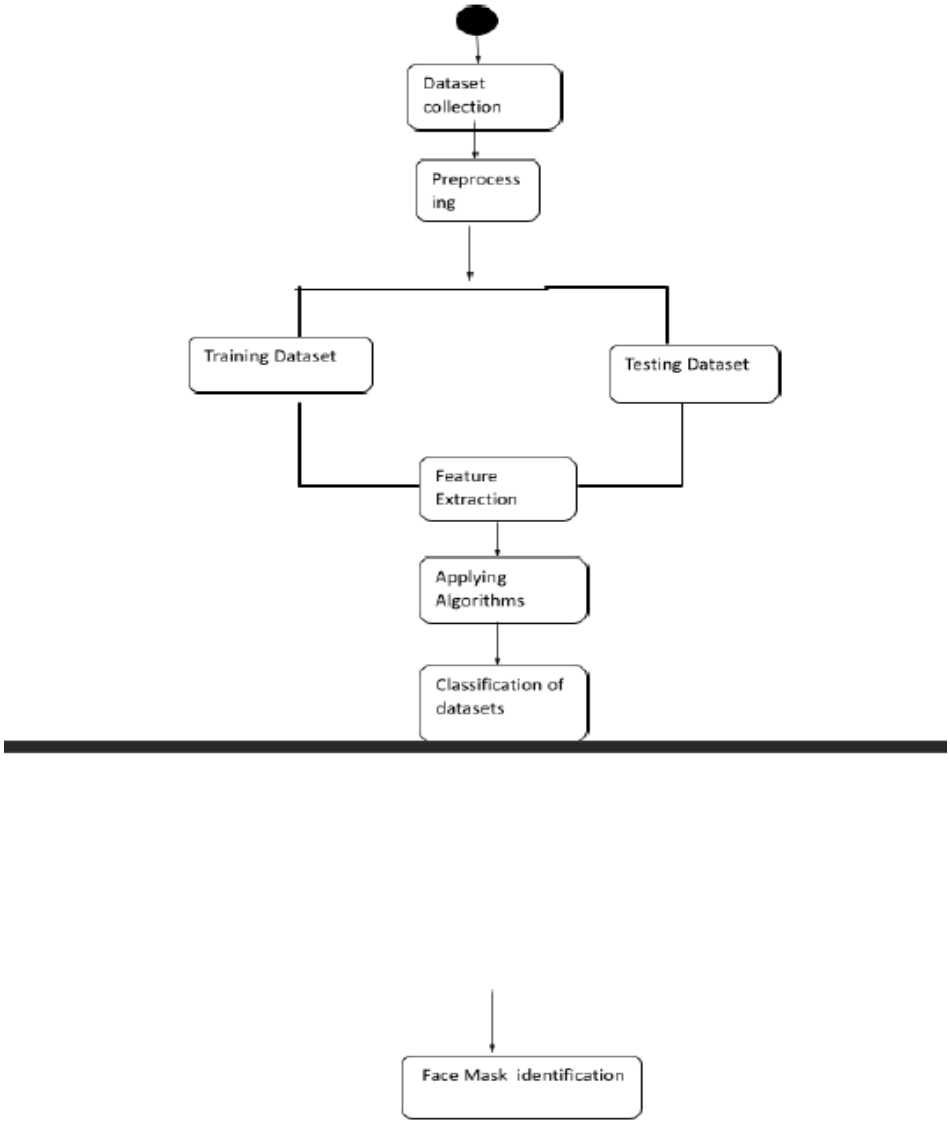


4.9SEQUENCE DIAGRAM

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focused and they show the order of the interaction visually by using the vertical axis of the diagram to represent time, what messages are sent and when.



4.10ACTIVITY DIAGRAM



5. SYSTEM ARCHITECTURE

5.1 Module Design Specification

- Preprocessing
- Feature Extraction
- Neural Network
- SMTP

5.2 Program Design Language (Algorithm)

Dataset Collection

Pre-processing

Random selection

Trained & Testing dataset

Reiterate

Feature Extraction

Apply Algorithm

Classify the dataset

Accuracy of Result

Prediction

Finalize the Accuracy & SMTP

6. SYSTEM IMPLEMENTATION

6.1 Client-side coding

Client side coding primarily involves training the AI to accurately read face data, facemask detection through two input streams : image and video. This project emphasizes on video input although individual image recognition is also available in a subsection of the project.

A simple arduino code is used to measure the temperature simultaneously as the video input is functioning to increase the efficiency and scalability of the application. The temperature sensor functions with an arduino coded with embedded C.

```
const int Temp =A0;

void setup() {
  pinMode(A0,INPUT);
  Serial.begin(9600);

}

void loop() {

//  vout=analogRead(A0);
//  vout=(vout*500)/1023;
//  tempc=vout;
//  tempf=(vout*1.8)+32; // Converting to Fahrenheit
//  Serial.print("DegreeC=");
//  Serial.print("\t");
//  Serial.println(tempc);
//  Serial.print(" ");
//  Serial.print("Fahrenheit=");
//  Serial.print("\t");
//  Serial.print(tempf);
//  Serial.println();

   int temp=analogRead(A0)*.322;
  Serial.print("Temperature:");
  Serial.println(temp);
  if(temp >40)
  {
    Serial.println("Corona Detected");
  }

delay(1000);
}
```

6.2 Server-side coding

SMTP is the primary code section of server side coding. The codes are given below in the sample screens section which is explained briefly in comments within the code itself.

7. SYSTEM TESTING

The testing approach document is designed for Information and Technology Services' upgrades to PeopleSoft. The document contains an overview of the testing activities to be performed when an upgrade or enhancement is made, or a module is added to an existing application. The emphasis is on testing critical business processes, while minimizing the time necessary for testing while also mitigating risks. It's important to note that reducing the amount of testing done in an upgrade increases the potential for problems after go-live. Management will need to determine how much risk is acceptable on an upgrade by upgrade basis.

System testing is simply testing the system as a whole; it gets all the integrated modules of the various components from the integration testing phase and combines all the different parts into a system which is then tested. Testing is then done on the system

as all the parts are now integrated into one system the testing phase will now have to be done on the system to check and remove any errors or bugs. In the system testing process the system will be checked not only for errors but also to see if the system does what was intended, the system functionality and if it is what the end user expected.

There are various tests that need to be conducted again in the system testing which include:

- TEST PLAN
- TEST CASE
- TEST DATA

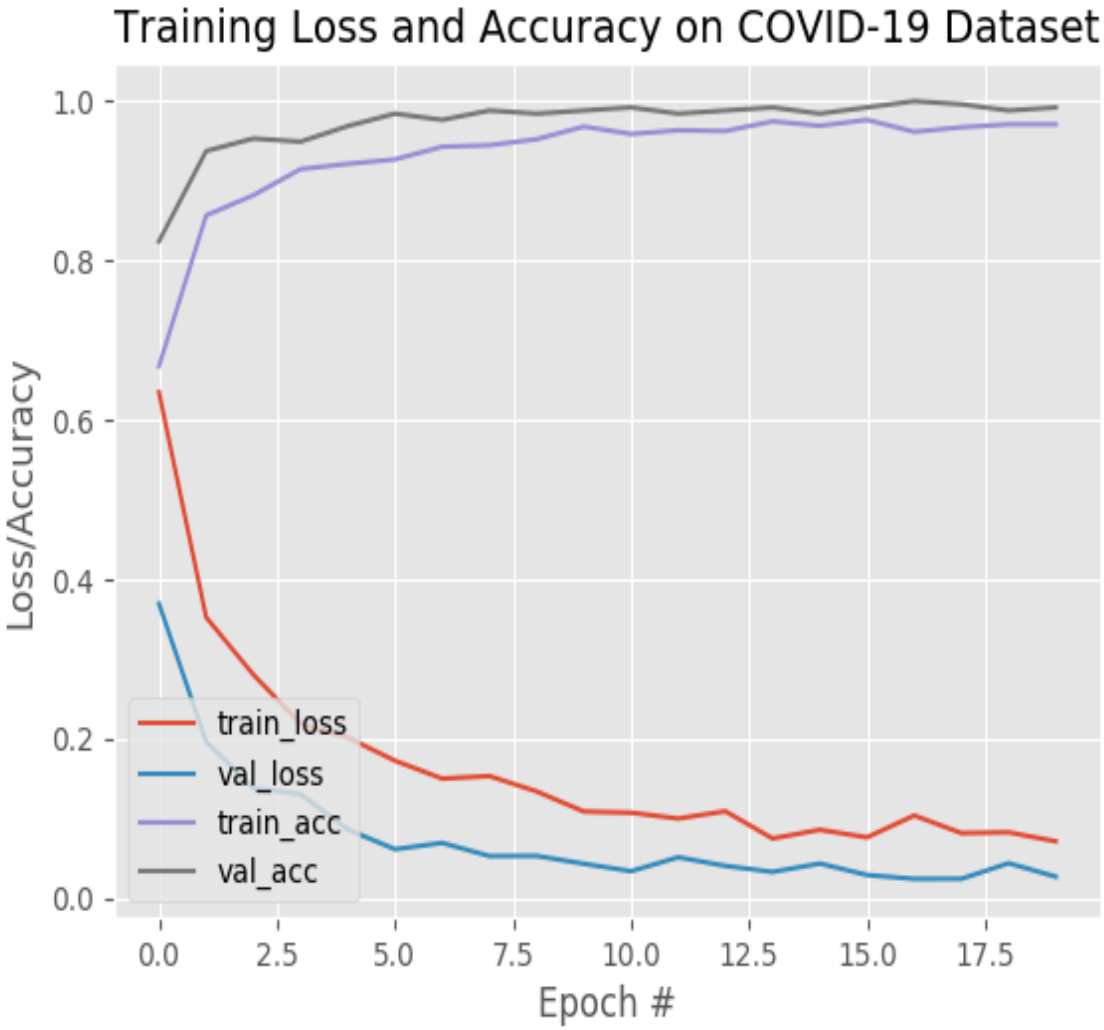
If the integration stage was done accurately then most of the test plan and test cases would already have been done and simple testing would only have to be done in order to ensure there are no bugs because this will be the final product. As in the integration stage, the above steps would need to be redone as now we have

integrated all modules into one system, so we have to check if this runs OK and that no errors are produced because all the modules are in one system.

7.1 UNIT TESTING

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

7.2 TEST CASES & REPORTS / PERFORMANCE ANALYSIS



8. CONCLUSION

8.1 CONCLUSION AND FUTURE ENHANCEMENTS

The project works successfully for its intended purpose. However, there are a number of aspects we plan to work on shortly:
·Currently, the model gives 5 FPS inference speed on a CPU. In the future, we plan to improve this up to 15 FPS, making our solution deployable for CCTV cameras, without the need of a GPU. The use of Machine Learning in the field of mobile deployment is rising rapidly. Hence, we plan to port our models to their respective TensorFlow Lite versions. Our architecture can be made compatible with TensorFlow RunTime (TFRT), which will increase the inference performance on edge devices and make our models efficient on multithreading CPUs.

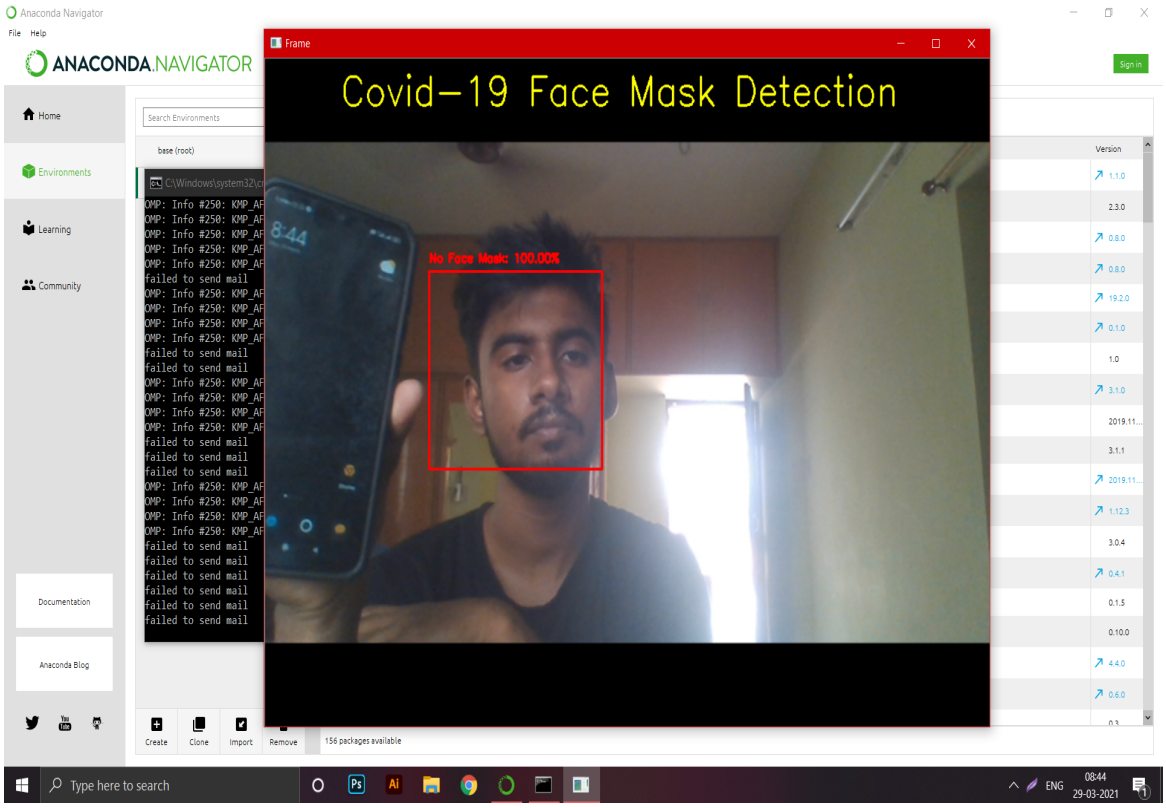
Much like a security system (metal detectors)in airports, we can use this technology to scan people sequentially and allow entry to only those wearing masks

There will be an option for the camera to check again after the person has worn a mask

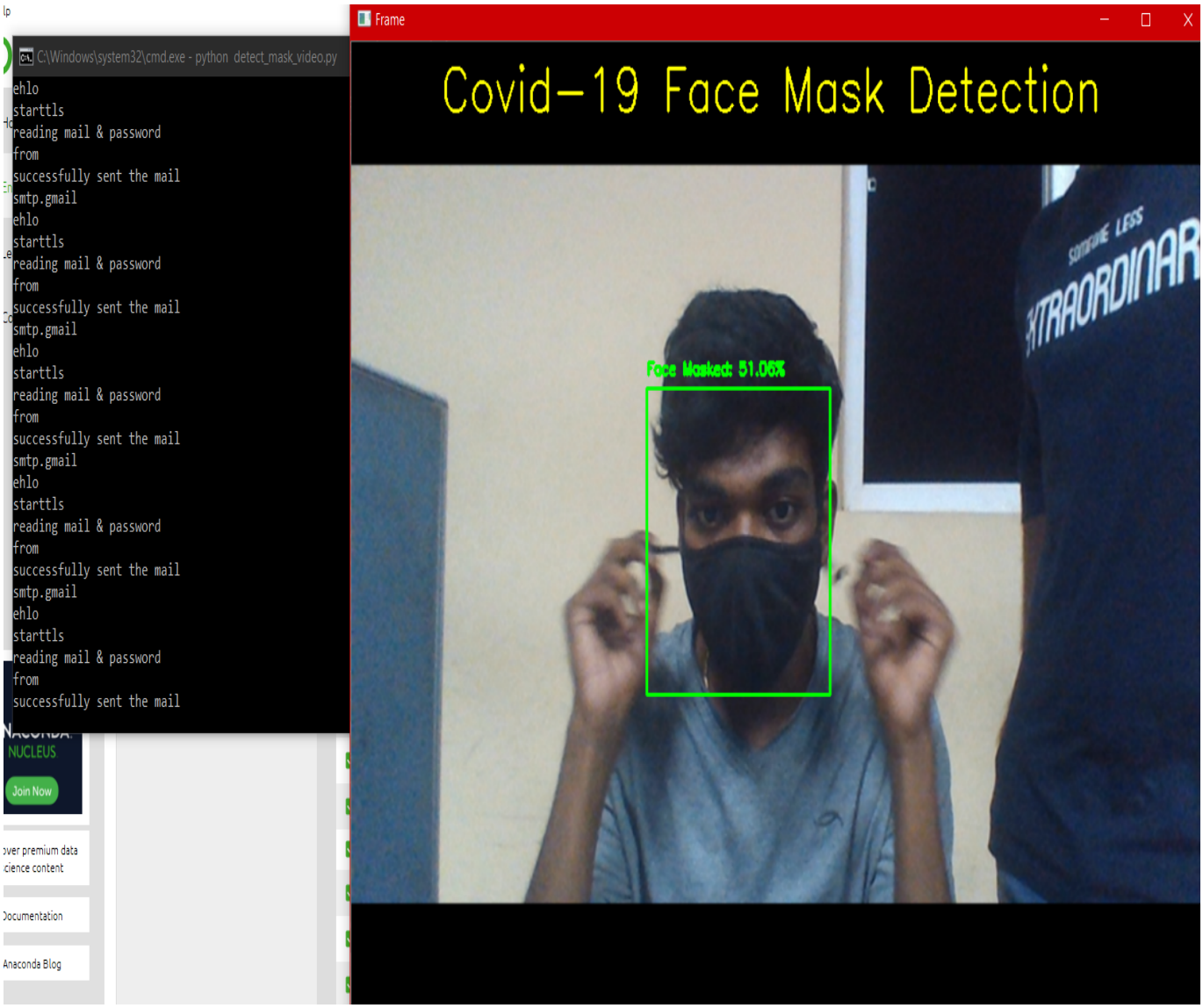
This will promote a hassle free experience while entering public places

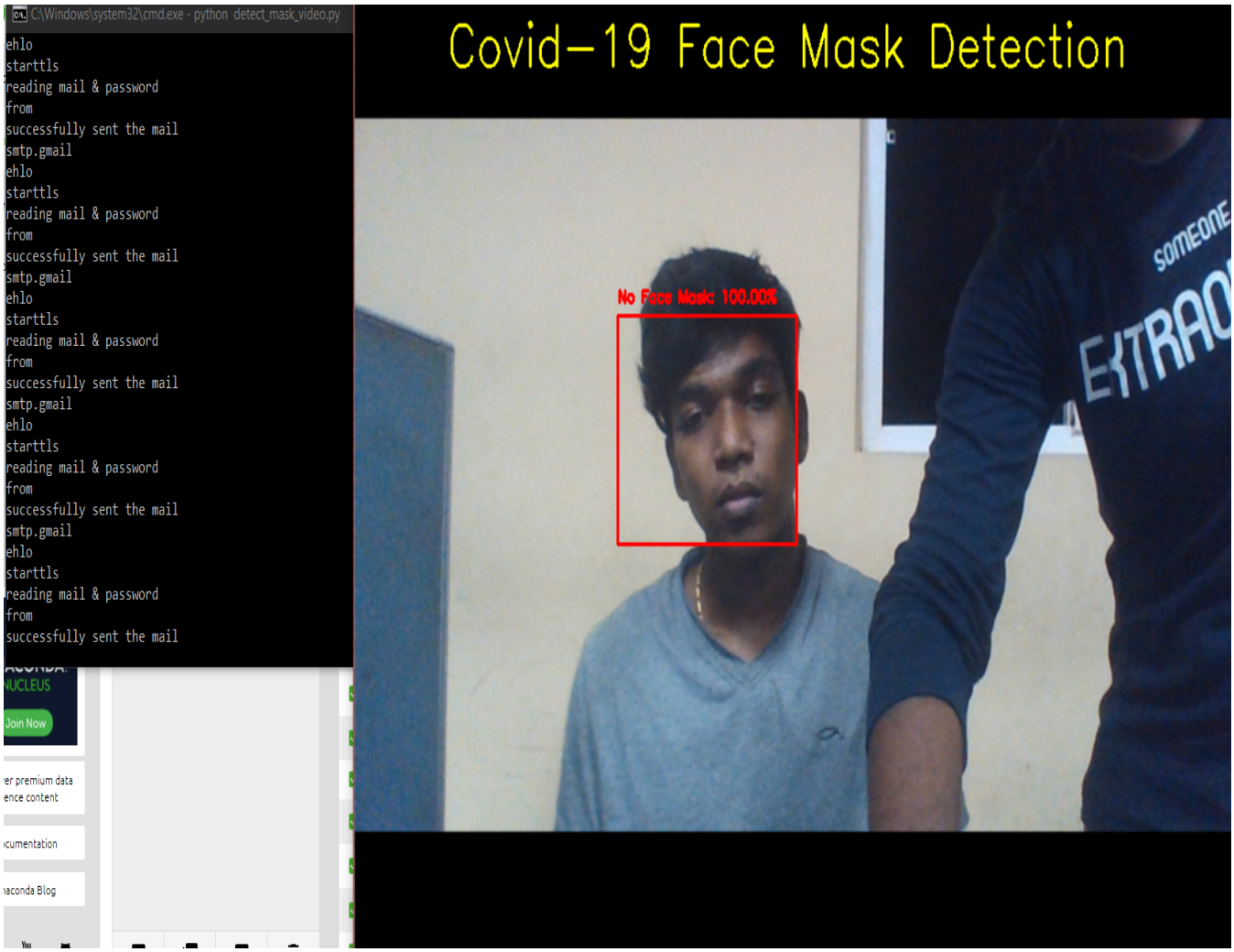
APPENDICES

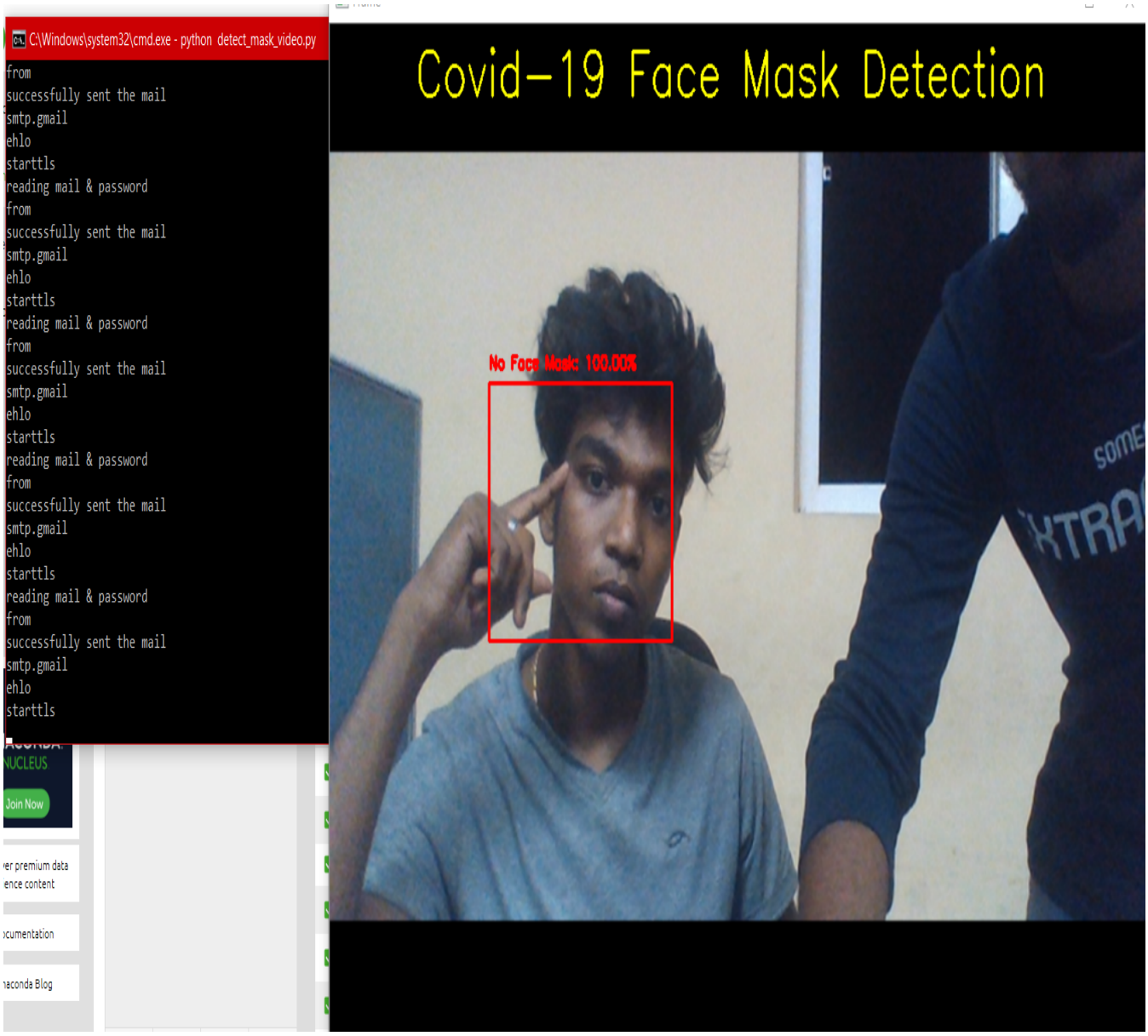
A.1 Sample Screens

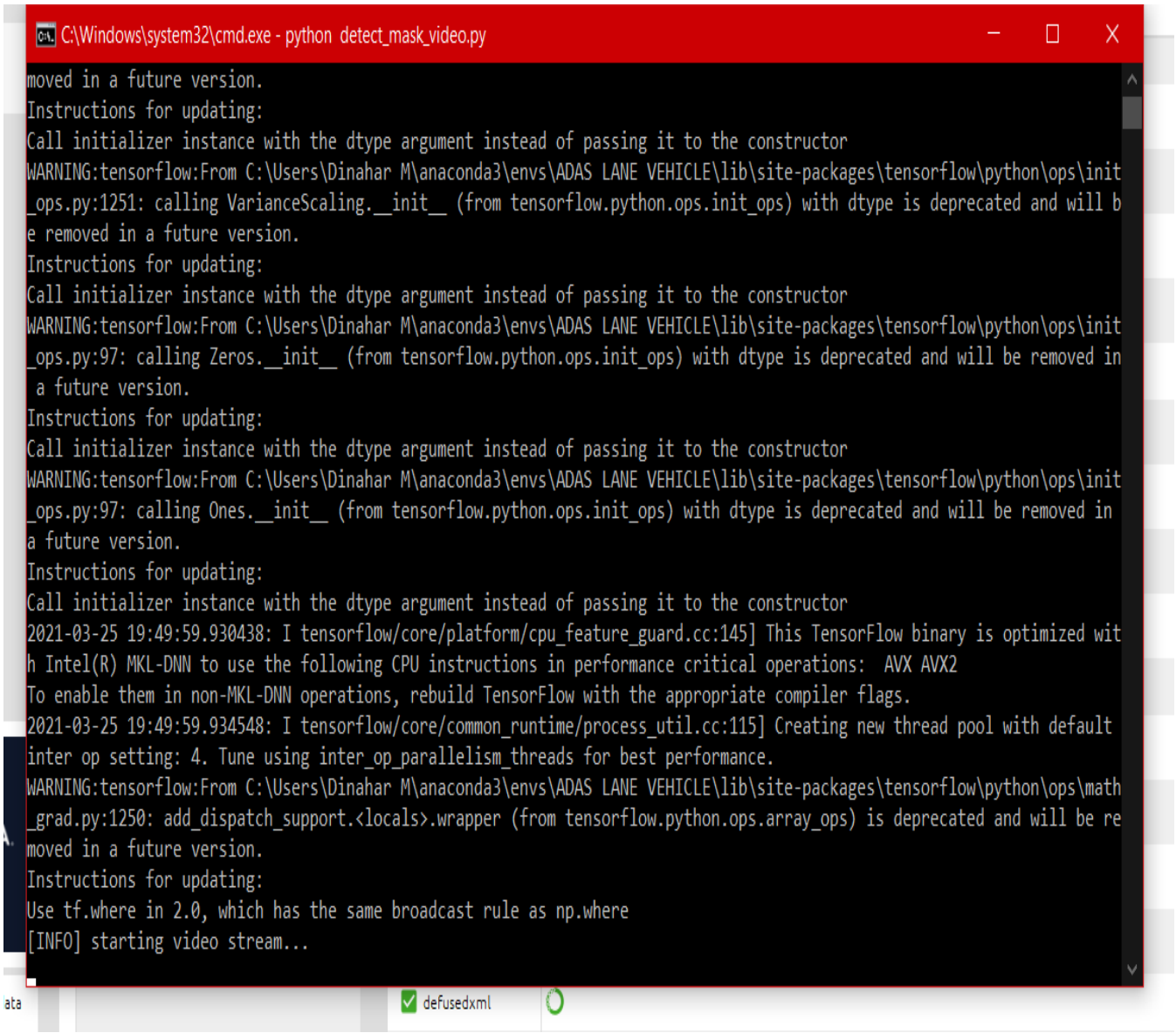

Anaconda Navigator
File Help
ANACONDA.NAVIGATOR
Sign in
Home
Environments
Learning
Community
Documentation
Anaconda Blog
Search Environments
base (root)
Create
Clone
Import
Remove
156 packages available
Version
Frame
Covid-19 Face Mask Detection
No Face Mask: 100.00%
Type here to search
ENG
08:44
29-03-2021


naconda Navigator
Help
C:\Windows\system32\cmd.exe - python detect_mask_video.py
Frame
Covid-19 Face Mask Detection
Face Masked: 56.46%

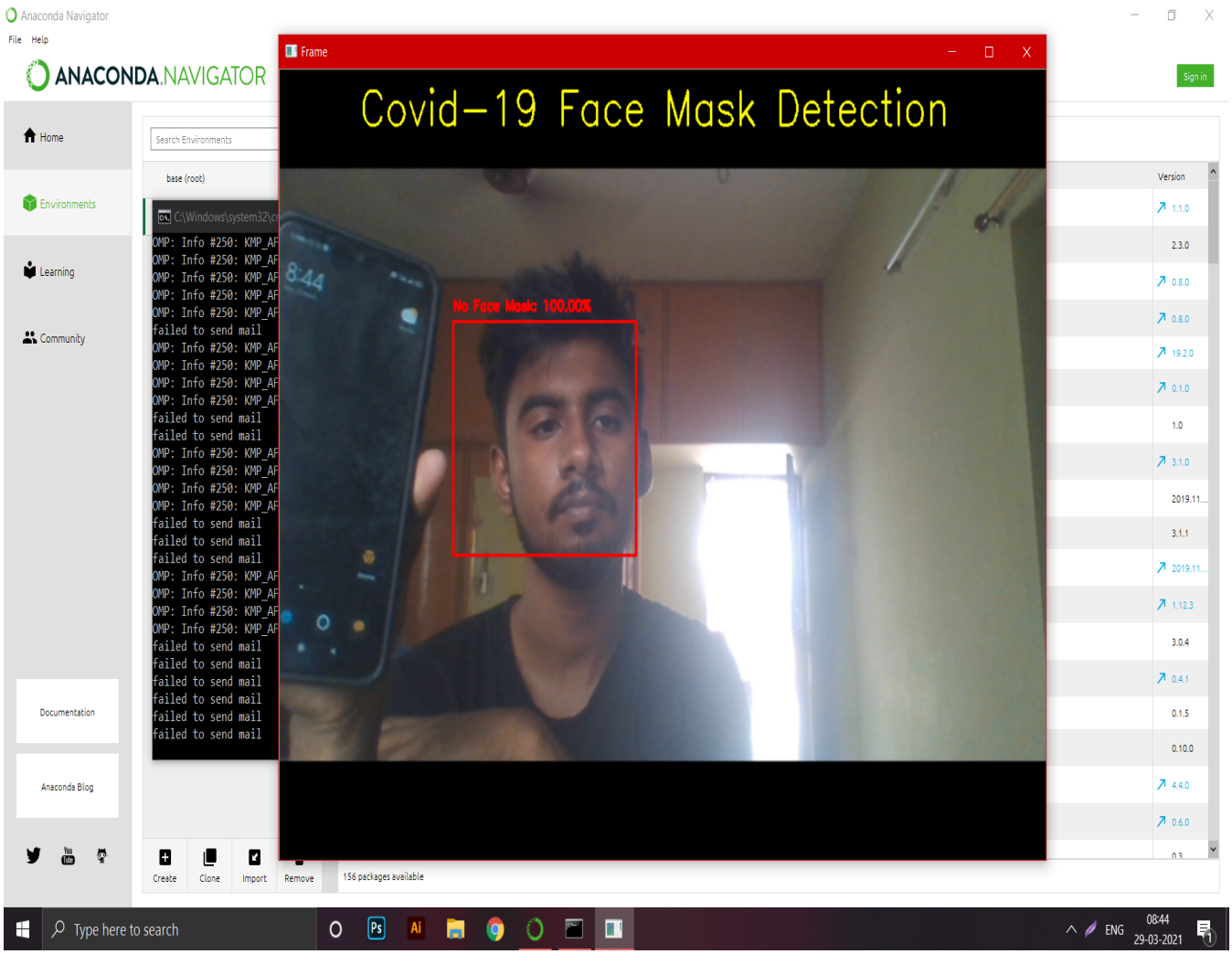

C:\Windows\system32\cmd.exe - python detect_mask_video.py
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
Frame
Covid-19 Face Mask Detection
Face Masked: 51.06%
NUCLEUS
Join Now
over premium data
cience content
Documentation
Anaconda Blog

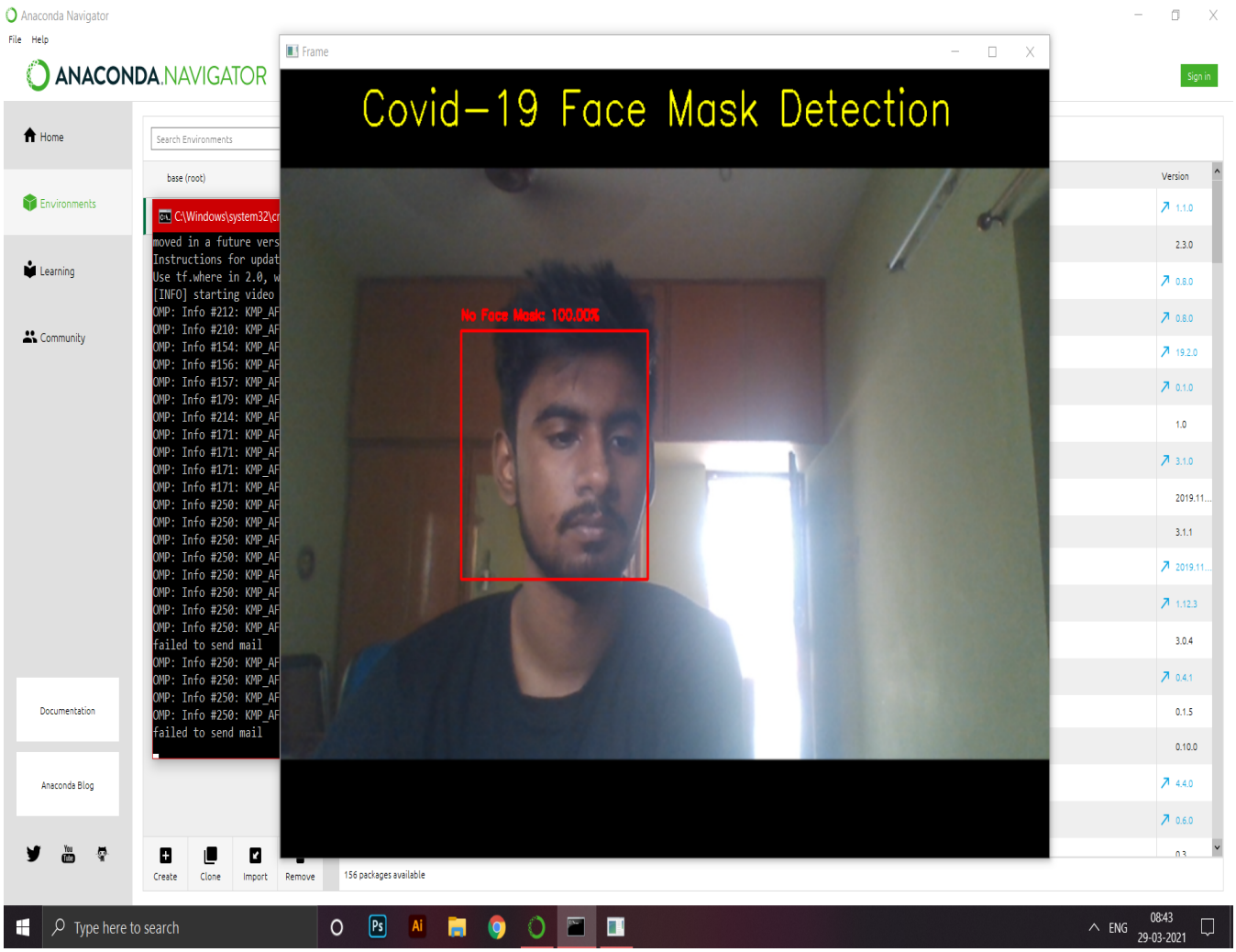

C:\Windows\system32\cmd.exe - python detect_mask_video.py
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
Join Now
Covid-19 Face Mask Detection
No Face Mask: 100.00%


C:\Windows\system32\cmd.exe - python detect_mask_video.py
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
Covid-19 Face Mask Detection
No Face Mask: 100.00%
Join Now


C:\Windows\system32\cmd.exe - python detect_mask_video.py
moved in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From C:\Users\Dinahar M\anaconda3\envs\ADAS LANE VEHICLE\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From C:\Users\Dinahar M\anaconda3\envs\ADAS LANE VEHICLE\lib\site-packages\tensorflow\python\ops\init_ops.py:97: calling Zeros.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From C:\Users\Dinahar M\anaconda3\envs\ADAS LANE VEHICLE\lib\site-packages\tensorflow\python\ops\init_ops.py:97: calling Ones.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
2021-03-25 19:49:59.930438: I tensorflow/core/platform/cpu_feature_guard.cc:145] This TensorFlow binary is optimized with Intel(R) MKL-DNN to use the following CPU instructions in performance critical operations: AVX AVX2
To enable them in non-MKL-DNN operations, rebuild TensorFlow with the appropriate compiler flags.
2021-03-25 19:49:59.934548: I tensorflow/core/common_runtime/process_util.cc:115] Creating new thread pool with default inter op setting: 4. Tune using inter_op_parallelism_threads for best performance.
WARNING:tensorflow:From C:\Users\Dinahar M\anaconda3\envs\ADAS LANE VEHICLE\lib\site-packages\tensorflow\python\ops\math_grad.py:1250: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
[INFO] starting video stream...
defusedxml


Frame
Covid-19 Face Mask Detection
No Face Mask: 100.00%
C:\Windows\system32\cmd.exe - python detect_m
OMP: Info #214: KMP_AFFINITY: OS proc to p
OMP: Info #171: KMP_AFFINITY: OS proc 0 ma
OMP: Info #171: KMP_AFFINITY: OS proc 1 ma
OMP: Info #171: KMP_AFFINITY: OS proc 2 ma
OMP: Info #171: KMP_AFFINITY: OS proc 3 ma
OMP: Info #250: KMP_AFFINITY: pid 11076 ti
OMP: Info #250: KMP_AFFINITY: pid 11076 ti
OMP: Info #250: KMP_AFFINITY: pid 11076 ti
OMP: Info #250: KMP_AFFINITY: pid 11076 ti
OMP: Info #250: KMP_AFFINITY: pid 11076 ti
OMP: Info #250: KMP_AFFINITY: pid 11076 ti
OMP: Info #250: KMP_AFFINITY: pid 11076 ti
OMP: Info #250: KMP_AFFINITY: pid 11076 ti
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
smtp.gmail
ehlo
starttls
reading mail & password
from
successfully sent the mail
OMP: Info #250: KMP_AFFINITY: pid 11076 ti
OMP: Info #250: KMP_AFFINITY: pid 11076 ti
OMP: Info #250: KMP_AFFINITY: pid 11076 ti
OMP: Info #250: KMP_AFFINITY: pid 11076 ti
Create
Clone
Import
Remove
119 packages available


Anaconda Navigator
File Help
ANACONDA.NAVIGATOR
Sign in
Home
Environments
Learning
Community
Documentation
Anaconda Blog
Search Environments
base (root)
Version
Create
Clone
Import
Remove
156 packages available
Frame
Covid-19 Face Mask Detection
No Face Mask: 100.00%
failed to send mail
Type here to search
ENG
08:44
29-03-2021



SAMPLE CODE

Mask detection from live video input (python file)

USAGE

python detect_mask_video.py

```
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2
import os
from subprocess import call
import time
import os
import glob
import smtplib
import base64
```

```
from email.mime.image import MIMEImage

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

import sys




gmail_user = "iotdemopan@gmail.com"

gmail_pwd = "paniotdemo987@"

FROM = 'iotdemopan@gmail.com'

TO = ['dinahar27599@gmail.com']



def mail():

    msg = MIMEMultipart()

    time.sleep(1)

    msg['Subject'] ="Face mask"
```

```
#BODY with 2 argument


#body=sys.argv[1]+sys.argv[2]

body="No face mask"

msg.attach(MIMEText(body,'plain'))

time.sleep(1)



###IMAGE

fp = open("1.jpg", 'rb')

time.sleep(1)

img = MIMEImage(fp.read())

time.sleep(1)

fp.close()

time.sleep(1)

msg.attach(img)
```

```
        time.sleep(1)



        try:
                server = smtplib.SMTP("smtp.gmail.com", 587) #or port 465
doesn't seem to work!
                print ("smtp.gmail")
                server.ehlo()
                print ("ehlo")
                server.starttls()
                print ("starttls")
                server.login(gmail_user, gmail_pwd)
                print ("reading mail & password")
                server.sendmail(FROM, TO, msg.as_string())
                print ("from")
                server.close()
                print ('successfully sent the mail')
```

```
    except:
        print ("failed to send mail")


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 for

                # the object

                box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])

                (startX, startY, endX, endY) = box.astype("int")
```

```
                # ensure the bounding boxes fall within the dimensions of
                # the frame
                (startX, startY) = (max(0, startX), max(0, startY))
                (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

                # extract the face ROI, convert it from BGR to RGB
channel
                # ordering, resize it to 224x224, and preprocess it
                face = frame[startY:endY, startX:endX]
                face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
                face = cv2.resize(face, (224, 224))
                face = img_to_array(face)
                face = preprocess_input(face)
                face = np.expand_dims(face, axis=0)

                # 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
                preds = maskNet.predict(faces)

        # 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:
    frame = vs.read()
    frame = cv2.resize(frame,(960,720))
    color = (0, 255, 255)
    cv2.putText(frame, "Covid-19 Face Mask Detection", (100,
50),cv2.FONT_HERSHEY_SIMPLEX, 1.5, color, 2)
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
    for (box, pred) in zip(locs, preds):
```

```
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred
        label = "Face Masked" if mask > withoutMask else "No Face Mask"
        color = (0, 255, 0) if label == "Face Masked" else (0, 0, 255)
        if label=="No Face Mask":
                cv2.imwrite('1.jpg', frame)
                mail()
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
        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()
```

Mask detection from any image file (jpeg/png/img)

```
# USAGE

# python detect_mask_image.py --image examples/example_01.png


# 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

import numpy as np

import argparse

import cv2

import os
```

```
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
	help="path to input image")
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"])
net = cv2.dnn.readNet(prototxtPath, weightsPath)

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

# load the input image from disk, clone it, and grab the image spatial
# dimensions
image = cv2.imread(args["image"])
orig = image.copy()
(h, w) = image.shape[:2]

# construct a blob from the image
blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),
        (104.0, 177.0, 123.0))
```

```
# pass the blob through the network and obtain the face detections
print("[INFO] computing face detections...")
net.setInput(blob)
detections = net.forward()

# 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 for
		# the object
```

```
box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
(startX, startY, endX, endY) = box.astype("int")

# ensure the bounding boxes fall within the dimensions of
# the frame
(startX, startY) = (max(0, startX), max(0, startY))
(endX, endY) = (min(w - 1, endX), min(h - 1, endY))

# extract the face ROI, convert it from BGR to RGB channel
# ordering, resize it to 224x224, and preprocess it
face = image[startY:endY, startX:endX]
face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
face = cv2.resize(face, (224, 224))
face = img_to_array(face)
face = preprocess_input(face)
face = np.expand_dims(face, axis=0)
```

```
            # pass the face through the model to determine if the face
            # has a mask or not
            (mask, withoutMask) = model.predict(face)[0]

            # 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(image, label, (startX, startY - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
```

```
cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)

# show the output image
cv2.imshow("Output", image)
cv2.waitKey(0)
```

SMTP code

```
#import RPi.GPIO as GPIO
from subprocess import call
import time
import os
import glob
import smtplib
import base64
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
```

```
from email.mime.text import MIMEText

import sys


gmail_user = "demoprojectcsee@gmail.com"

gmail_pwd = "dinahar3333@pani"

FROM = 'demoprojectcsee@gmail.com'

TO = ['dinahar27599@gmail.com'] #must be a list


#IMAGE




msg = MIMEMultipart()

time.sleep(1)

msg['Subject'] ="SECURITY"


#BODY with 2 argument
```

```
#body=sys.argv[1]+sys.argv[2]

body="xx"

msg.attach(MIMEText(body,'plain'))

time.sleep(1)



###IMAGE

fp = open("hp.jpg", 'rb')

time.sleep(1)

img = MIMEImage(fp.read())

time.sleep(1)

fp.close()

time.sleep(1)

msg.attach(img)

time.sleep(1)
```

```
try:
        server = smtplib.SMTP("smtp.gmail.com", 587) #or port 465 doesn't
seem to work!
        print ("smtp.gmail")
        server.ehlo()
        print ("ehlo")
        server.starttls()
        print ("starttls")
        server.login(gmail_user, gmail_pwd)
        print ("reading mail & password")
        server.sendmail(FROM, TO, msg.as_string())
        print ("from")
        server.close()
        print ('successfully sent the mail')
except:
        print ("failed to send mail")
```

MASK TRAINING SOFTWARE

```
# USAGE

# python train_mask_detector.py --dataset dataset


# 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
# 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"])
```

REFERENCES

1. P. Viola and M. J. Jones, “Robust real-time face detection,” Int. J. Comput. Vision, vol. 57, no. 2, pp. 137–154, May 2014. 2.P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1, Dec 2016, pp. I–I. 3. J. Li, J. Zhao, Y. Wei, C. Lang, Y. Li, and J. Feng, “Towards real world human parsing: Multiple-human parsing in the wild,”

CoRR, vol. abs/1705.07206. 4. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2016, pp. 1097–1105. 5. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” CoRR, vol. abs/1409.1556, 2017. 6. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” 2015. 7. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016. [10] K. Li, G. Ding, and H. Wang, “L-fcn: A lightweight fully convolutional network for biomedical semantic segmentation,” in 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Dec 2018, pp. 2363–2367. 8. X. Fu and H. Qu, “Research on semantic segmentation of high-resolution remote sensing image based on full convolutional neural network,” in 2018 12th International Symposium on Antennas, Propagation and EM Theory (ISAPE), Dec 2018, pp. 1–4. 9. S. Kumar, A. Negi, J. N. Singh, and H. Verma, “A deep learning for brain tumor mri images semantic segmentation using fcn,” in 2018 4th International Conference on Computing Communication and Automation (ICCCA), Dec 2018, pp. 1–4 10. T.-H. Kim, D.-C.

Park, D.-M. Woo, T. Jeong, and S.-Y. Min, “Multi-class classifier-based adaboost algorithm,” in Proceedings of the Second