A

Mini Project

On

FACE MASK DETECTION USING DEEP LEARNING AND OPENCV

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

B.HEMANTHNAIDU(197R1A0505)

PALLAVI GANJI(197R1A0512)

P.VISHALA REDDY (197R1A0543)

Under the Guidance of

DR.K.SRUJAN RAJU

(Professor & Head of Department)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

2019-2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"FACE MASK SDETECTION USING DEEP LEARNING AND OPENCV"** being submitted by **B. HEMANTH NAIDU(197R1A0505), PALLAVI GANJI(197R1A0512) & P. VISHALA(197R1A0543)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2022-23.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dr. K. Srujan Raju
(Professor)
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Dr.K. Srujan Raju,** Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by his shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee **(PRC) Dr. Punyaban Patel, Ms. Shilpa, Dr. M . Subha Mastan Rao & J. Narasimharao** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to Dr. A. Raji Reddy, Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. Ch. Gopal Reddy, Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project. The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

B.HEMANTH NAIDU (197R1A0505)

PALLAVI GANJI (197R1A0512)

P.VISHALA REDDY (197R1A0543)

ABSTRACT

Wearing a face mask will help prevent the spread of infection and prevent the individual from contracting any airborne infectious germs. Using Face Mask Detection System, one can monitor if the people are wearing masks or not. Here featured classifier is used for image detection combination of CNN. Collating with other existing algorithms, this classifier produces a high recognition rate even with varying expressions, efficient feature selection and low assortment of false positive features. Feature-based cascade classifier system utilizes only 200 features out of 6000 features to yield a recognition rate of 85-95. Studies have proved that wearing a face mask significantly reduces the risk of viral transmission as well as provides a sense of protection. However, it is not feasible to manually track the implementation of this policy. Technology holds the key here. We introduce a Deep Learning based system that can detect instances where face masks are not used properly. Our system consists of a dual stage Convolutional Neural Network (CNN) architecture capable of detecting masked and unmasked faces and can be integrated with pre-installed CCTV cameras. This will help track safety violations, promote the use of face masks, and ensure a safe working environment.

LIST OF FIGURES/TABLES

FIGURE NO	FIGURE NAME	PAGE NO
Figure 4.1	Project Architecture for Face Mask Detection Using Deep Learning and open cv	9
Figure 4.2.1	Anaconda download page	10
Figure 4.2.2	Choosing anaconda prompt	11
Figure 4.2.3	Installation of numpy	11
Figure 4.4.1	The convolutional Layer	13
Figure 4.4.2	Convolutional Neural Networks Scan Images	14

Figure 4.6.1	Use case diagram for Face Mask Detection Using Deep Learning and open cv	17
Figure 4.6.2	Class diagram for Face Mask Detection Using Deep Learning and open cv	18
Figure 4.6.3	Sequence diagram for Face Mask Detection Using Deep Learning and Opencv	19
Figure 4.6.4	Activity diagram for Face Mask Detection Using Deep Learning and open cv	20

LIST OF SCREENSHOTS

SCREENSHOT NO	SCREENSHOT NAME	PAGE NO.
Screenshot 6.1	Requirements	45
Screenshot 6.2	Installing the Packages	45
Screenshot 6.3	Uploading dataset	46
Screenshot 6.4	Training the dataset	46
Screenshot 6.5	Use model in realtime webcam	47
Screenshot 6.6	Output with mask	48
Screenshot 6.7	Output with no mask	48

TABLE OF CONTENTS

ABSTRACT i
LIST OF FIGURES/TABLES ii
LIST OF SCREENSHOTS iii

1. INTRODUCTION

1.1 PROJECT SCOPE 1
1.2 PROJECT PURPOSE 1
1.3 PROJECT FEATURES 1

2. LITERATURE SURVEY

2.1 FACEMASK DETECTION USING SEMANTIC SEGMENTATION 2
2.2 FACE RECOGNITION WITH FACE MASK APPLICATIONS AND NEURAL NETWORK 2
2.3 FACE DETECTION AND SEGMENTATION BASED ON IMPROVED MASKED CNN 2
2.4 REAL-TIME FACE MASK IDENTIFICATION USING FACEMASK NET DEEP LEARNING NETWORK 3
2.5 A HYBRID DEEP TRANSFER MODEL WITH MACHINE LEARNING METHODS FOR FACEMASK DETECTION IN THE ERA OF THE COVID-19 PANDEMIC 4
2.6 COVID 19 FACEMASK DETECTION WITH DEEP LEARNING AND COMPUTER VISION 4

3. SYSTEM ANALYSIS

3.1 PROBLEM DEFINATION 5
3.2 EXISTING SYSTEM 5
3.2.1 DISADVANTAGES OF EXISTING SYSTEM 5
3.3 PROPOSED SYSTEM 6

3.3.1 ADVANTAGES OF PROPOSED SYSTEM 6
3.3.2 APPLICATIONS 6
3.4 FEASIBILITY STUDY 7
3.4.1 ECONOMICAL FEASIBILITY 7
3.4.2 TECHNICAL FEASIBILITY 7
3.4.3 SOCIAL FEASIBILITY 7
3.5 HARDWARE & SOFTWARE REQUIREMENTS
3.5.1 HARDWARE REQUIREMENTS 8
3.5.2 SOFTWARE REQUIREMENTS 8

4. ARCHITECTURE

4.1 PROJECT ARCHITECTURE 9
4.2 SOFTWARE INSTALLATION FOR MACHINE LEARNING 10
4.3 MODULES 12
4.4 ALGORITHM 12
4.5 STEPS FOR EXECUTING THE PROJECT 15
4.6 UML DIAGRAMS 16
4.6.1 USE CASE DIAGRAM 17
4.6.2 CLASS DIAGRAM 18
4.6.3 SEQUENCE DIAGRAM 19
4.6.4 ACTIVITY DIAGRAM 20
4.7 INTRODUCTION TO PYTHON 21

5. IMPLEMENTATION

5.1 DESCRIPTION 36
5.2 SOURCE CODE 37

6. SCREENSHOTS 45

7 .TESTING

7.1 INTRODUCTION TO TESTING 49

7.1.1 UNIT TESTING 49

7.1.2 INTEGRATION TESTING 49

7.1.3 FUNCTIONAL TESTING 49

7.1.4 SYSTEM TESTING 50

7.1.5 WHITE BOX TESTING 50

7.1.6 BLACK BOX TESTING 50

7.1.7 ACCEPTANCE TESTING 51

7.2 TEST CASES 52

8. CONCLUSION & FUTURE SCOPE

8.1 PROJECT CONCLUSION 53

8.2 FUTURE SCOPE 53

9.REFERENCES 54

9.1 GITHUB LINK 56

1. INTRODUCTION

1.1 PROJECT SCOPE

To mandate the use of facemasks, it becomes essential to devise some techniques that enforce individuals to apply a mask before exposure to public places. This application can be very useful in public areas such as airports, railway stations, crowded markets, malls, etc. The proposed method used here is carried out in two steps. The first step is to train the face mask detector using transfer learning. The second step is to use this trained face mask detector on images or videos of people to identify if they are wearing a mask.

1.2 PURPOSE OF THE SYSTEM

Multiple studies have shown that the use of face masks reduces the risk of viral transmission as well as provides a sense of protection (Howard et al., 2020; Verma et al., 2020). However, it is infeasible to manually enforce such a policy on large premises and track any violations. Computer Vision provides a better alternative to this. Using a combination of image classification, object detection, object tracking, and video analysis, we developed a robust system that can detect the presence and absence of face masks in images as well as videos. In this paper, we propose a two-stage CNN architecture, where the first stage detects human faces, while the second stage uses a classifier to classify the faces detected in the first stage as either ‘Mask’ or ‘No Mask’ faces and draws bounding boxes around them along with the detected class name and gives voice alert if there is no mask. This algorithm was further extended to videos as well.

1.3 PROJECT FEATURES

The main feature of this project is to to identify whether the person on video stream is wearing a face mask or not with the help of computer vision and deep learning. There are different deep learning approaches .CNN mostly used algorithms in image and face recognition and also classifier is also involved which is used for extracting facial features.

2. LITERATURE SURVEY

2.1 FACIAL MASK DETECTION USING SEMANTIC SEGMENTATION

Face Detection has evolved as a very popular problem in Image processing and Computer Vision. Many new algorithms are being devised using convolutional architectures to make the algorithm as accurate as possible. These convolutional architectures have made it possible to extract even the pixel details. We aim to design a binary face classifier which can detect any face present in the frame irrespective of its alignment. We present a method to generate accurate face segmentation masks from any arbitrary size input image. Beginning from the RGB image of any size, the method uses Predefined Training Weights of VGG – 16 Architecture for feature extraction. Training is performed through Fully Convolutional Networks to semantically segment out the faces present in that image. Gradient Descent is used for training while Binomial Cross Entropy is used as a loss function. Further the output image from the FCN is processed to remove the unwanted noise and avoid the false predictions if any and make bounding box around the faces. Furthermore, proposed model has also shown great results in recognizing non-frontal faces. Along with this it is also able to detect multiple facial masks in a single frame. Experiments were performed on Multi Parsing Human Dataset obtaining mean pixel level accuracy of 93.884 % for the segmented face masks.

2.2 FACE RECOGNIZATION WITH FACIAL MASK APPLICATION AND NEURAL NETWORKS

Face recognition represents one of the most interesting modalities of biometric. Due to his low intrusiveness and to the constant decrease in image acquisition cost, it’s particularly suitable for a wide number of real time applications. In this paper we propose a very fast image pre-processing by the introduction of a linearly shaded elliptical mask cantered over the faces. Used in association with DCT, for features extraction, and MPL and RBF Neural Networks, for classification, it allows an improvement of system performances without modifying the global computation weight and also a learning time reduction for MLP neural networks.

2.3 FACE DETECTION AND SEGMENTATION BASED ON IMPROVED MASK R-CNN

Deep convolutional neural networks have been successfully applied to face detection recently. Despite making remarkable progress, most of the existing detection methods only localize each face using a bounding box, which cannot segment each face from the background image simultaneously. To overcome this drawback, we present a face detection and segmentation method based on improved Mask R-CNN, named G-Mask, which incorporates face detection and segmentation into one framework aiming to obtain more fine-grained information of face. Specifically, in this proposed method, ResNet-101 is utilized to extract features, RPN is used to generate Roils, and Roiling faithfully preserves the exact spatial locations to generate binary mask through Fully Convolution Network (FCN). Furthermore, Generalized Intersection over Union (Giroux) is used as the bounding box loss function to improve the detection accuracy. Compared with Faster R-CNN, Mask R-CNN, and Multitask Cascade CNN, the proposed G-Mask method has achieved promising results on FDDB, AFW, and WIDER FACE benchmarks.

2.4 REAL-TIME FACE MASK IDENTIFICATION USING FACEMASK LEARNING NETWORK

The pandemic is devastating mankind irrespective of caste, creed, gender, and religion. Until a vaccine is discovered, we should do our bit to constrain the expanse of the corona-virus. Using a face mask can undoubtedly help in managing the spread of the virus. Face mask detector uses or owns Facemask net, deep learning techniques to successfully test whether a person is with wearing a face mask or not. The manuscript presents three-class classification namely person is wearing a mask, or improperly worn masks or no mask detected. Using our deep learning method called Facemask net, we got an accuracy of 98.6 %. The Facemask net can work with still images and also works with a live video stream. Cases in which the mask is improperly worn are when the nose and mouth are partially covered. Our face mask identifier is least complex in structure and gives quick results and hence can be used in CCTV footage to detect whether a person is wearing a mask perfectly so that he does not pose any danger to others. Mass screening is possible and hence can be used in crowded places like railway stations, bus stops, markets, streets, mall entrances, schools, colleges, etc. By monitoring the

placement of the face mask on the face, we can make sure that an individual wears it the right way and helps to curb the scope of the virus.

2.5 A HYBRID DEEP TRANSFER LEARNING MODEL WITH MACHINE LEARNING METHODS FOR FACE MASK DETECTION IN THE ERA OF THE COVID-19 PANDEMIC

The coronavirus pandemic is causing a global health crisis. One of the effective protection methods is wearing a face mask in public areas according to the World Health Organization (WHO). In this paper, a hybrid model using deep and classical machine learning for face mask detection will be presented. The proposed model consists of two components. The first component is designed for feature extraction using Resnet50. While the second component is designed for the classification process of face masks using decision trees, Support Vector Machine (SVM), and ensemble algorithm. Three face masked datasets have been selected for investigation. The Three datasets are the Real-World Masked Face Dataset (RMFD), the Simulated Masked Face Dataset (SMFD), and the Labelled Faces in the Wild (LFW). The SVM classifier achieved 99.64% testing accuracy in RMFD. In SMFD, it achieved 99.49%, while in LFW, it achieved 100% testing accuracy.

2.6 FACEMASK DETECTION WITH DEEP LEARNING AND COMPUTER VISION

An efficient and economic approach of using AI to create a safe environment in a manufacturing setup. A hybrid model using deep and classical machine learning for face mask detection will be presented. A face mask detection dataset consists of with mask and without mask images, we are going to use Opens to do real-time face detection from a live stream via our webcam. We will use the dataset to build a COVID-19 face mask detector with computer vision using Python, Opens, and Tensor Flow and Keas. Our goal is to identify whether the person on image/video stream is wearing a face mask or not with the help of computer vision and deep learning.

3. SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, "what must be done to solve the problem?" The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

3.1 PROBLEM DEFINATION

This presents a study on real-time face mask detection using OpenCV and deep learning techniques and based on the results obtained, it is a noteworthy method for easy detection of facemask.An organization can monitor employee and their safety. And Medical officials can monitor the society to make health policies.

3.2 EXISTING SYSTEM

Face Mask Detection Platform uses Artificial Network to recognize if a user is wearing or not wearing a mask. Face Mask Detection System by using existing cameras combined with Trident Computer Vision platform to detect people without masks. This system consists of a ETL YOLO V4(You Only Look Once) architecture capable of detecting masked and unmasked faces and can be integrated with CCTV cameras.ETL YOLO V4 is producing high precision and detection accuracy.

3.2.1 DISADVANTAGES OF EXISTING SYSTEM

1. Low Accuracy.
2. Losing rate high.
3. Time consuming.

3.3 PROPOSED SYSTEM

World Health Organisation(WHO) has made wearing masks compulsory to protect against this deadly virus, so our project will notify if someone is wearing mask or not. In this Project we will develop a deep learning. We will use the dataset to build a face mask detector with computer vision using Python, Opencv, and Tensor Flow and Keras. Our goal is to identify whether the person on image/video stream is wearing a face mask or not and gives voice alert if person doesn't wear a mask with the help of computer vision and deep learning.

3.3.1 ADVANTAGES OF PROPOSED SYSTEM

- High Accuracy.
- Losing rate reduces.
- Time minimising.

3.3.2 APPLICATIONS

1. Research centers.
2. Medical health organizations.
3. Hospitals.
4. Malls.
5. Theaters.
6. Airports.
7. Educational Sectors.
8. Most organizations involving not working from home.

3.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

3.4.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.4.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system

and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.5 HARDWARE & SOFTWARE REQUIREMENTS

3.5.1 HARDWARE REQUIREMENTS

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

- Processor - I3/Intel Processor
- Hard Disk -160GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA
- RAM - 8Gb

3.5.2 SOFTWARE REQUIREMENTS

- Operating System : Windows 7/8/10
- Server side Script : Python, Anaconda
- IDE : PyCharm
- Libraries Used ;opencv,sklearn,tensosflow numpy,matplotlib,pandas.
- Dataset : Face_mask_detection
- Accessories : Webcam
- Technology : Python 3.6+

4.ARCHITECTURE

4.1 PROJECT ARCHITECTURE

The System architecture represents the design process that identifies the subsystems which constitute the framework and system for control and communication. Its intention is to establish the overall structure and interactions of a software system.

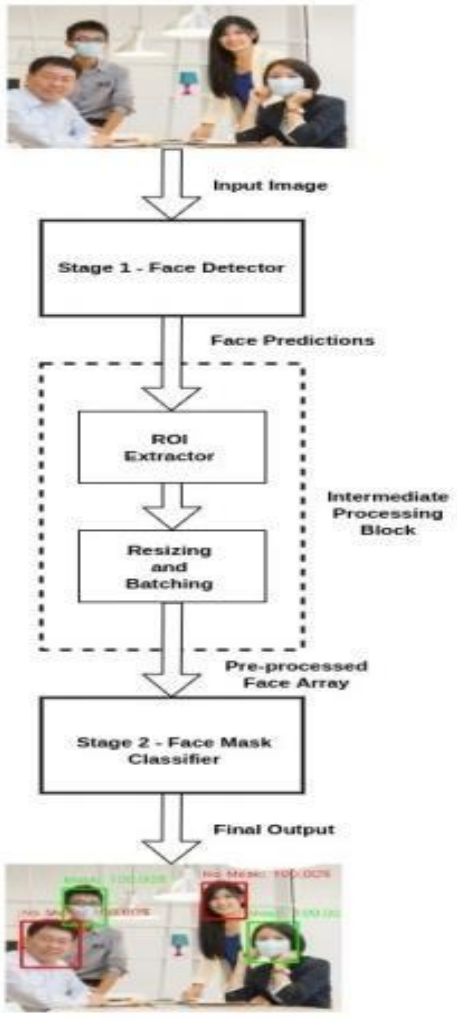


Figure 4.1: Project Architecture of Face Mak detection using deep learning and opencv.

4.2 SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS:

1. Visit the Anaconda downloads page.

Go to the following link: Anaconda.com/downloads

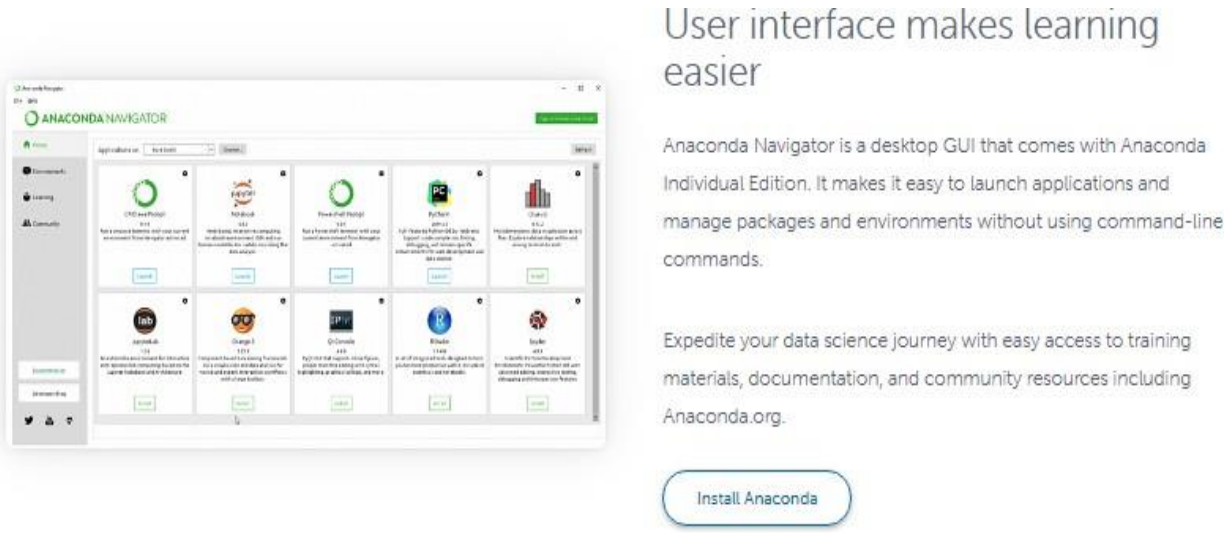


Figure 4.2.1:Anaconda download page

2. Select Windows.

Select Windows where the three operating systems are listed.

3. Download.

Choose Python 3.6 version, 64 bit graphical installer.

4. Let it download in an .exe format.

5. Open and run the installer.

Once the download completes, open and run the .exe installer.

6. Click on next, I agree, install.

7. Completion of the installation, open your windows start menu and select the Anaconda navigator.

8. Click on Jupiter launch button, your Jupiter notebook will get start.

9. You need to install some packages to execute your project in a proper way.

10. Select windows start menu, right click on anaconda prompt, choose run as administrator.

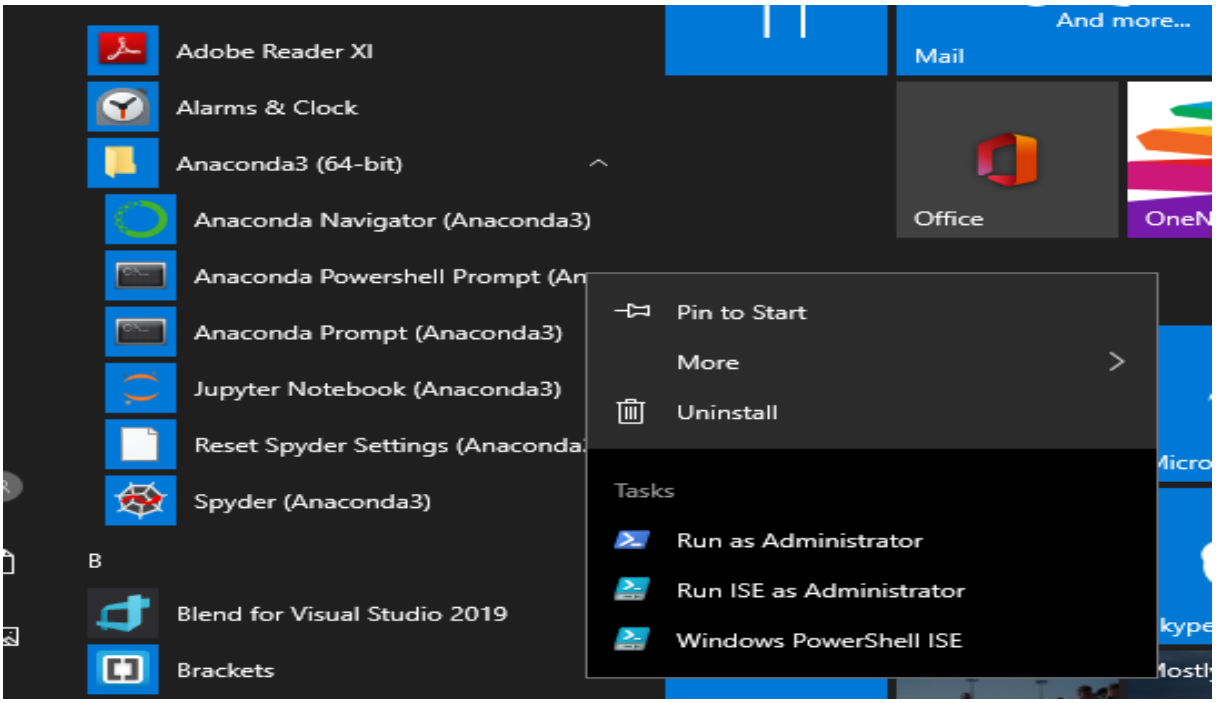


Figure 4.2.2:Choosing anaconda prompt

11. Anaconda prompt will get open, with specified path, type “Pip install package name” which you want to install (like Jumpy, Pandas, seaborne, sickie learn, matplotlib, pilot)

Ex: pip install numpy

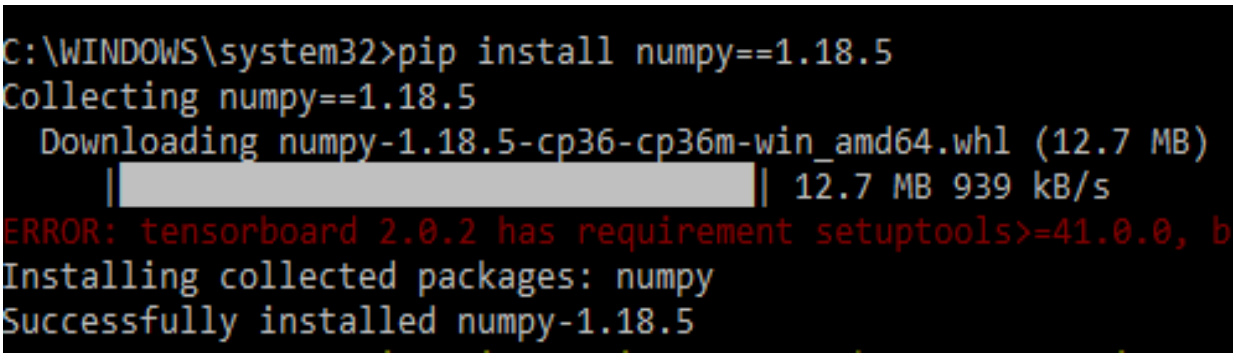


Figure 4.2.3: Installation of numpy

12. You can also install required packages in Jupiter notebook directly by using the syntax as “! Pip install package name.

Ex: pip install Jumpy

4.3 MODULES

TAKE DATASET:

Upload the (face_mask_detection) dataset.

VIEW:

The facial images of a people with or without wearing mask is displayed.

TRAINING:

After upload the dataset to the system to train the model by applying the CNN algorithm and generate the model which will learn from the values of the trained dataset. Display the graphical report how much we can get the accuracy and loss of a data.

TESTING:

Upload a live video in which the model detects if a person is wearing a mask or not by using a Web Cam.

4.4 ALGORITHM

4.4.1 Convolutional Neural Network

Step1: convolutional operation

The first building block in our plan of attack is convolution operation. In this step, we will touch on feature detectors, which basically serve as the neural network's filters. We will also discuss feature maps, learning the parameters of such maps, how patterns are detected, the layers of detection, and how the findings are mapped out.

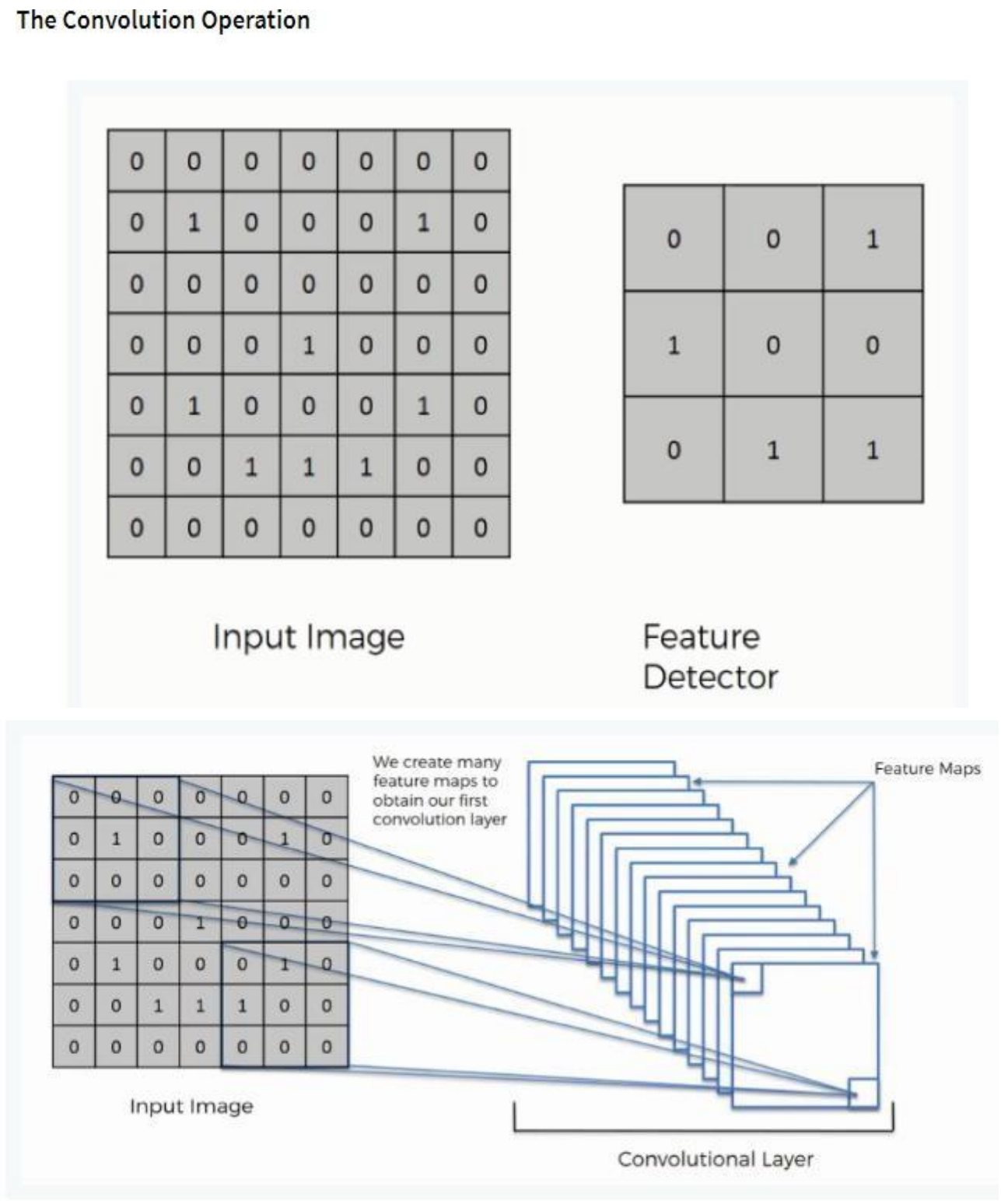


Figure 4.4.1:The convolutional Layer

Step (1b): Relu Layer

The second part of this step will involve the Rectified Linear Unit or Relook. We will cover Relook layers and explore how linearity functions in the context of Convolutional Neural

Networks.

Not necessary for understanding CNN's, but there's no harm in a quick lesson to improve your skills.

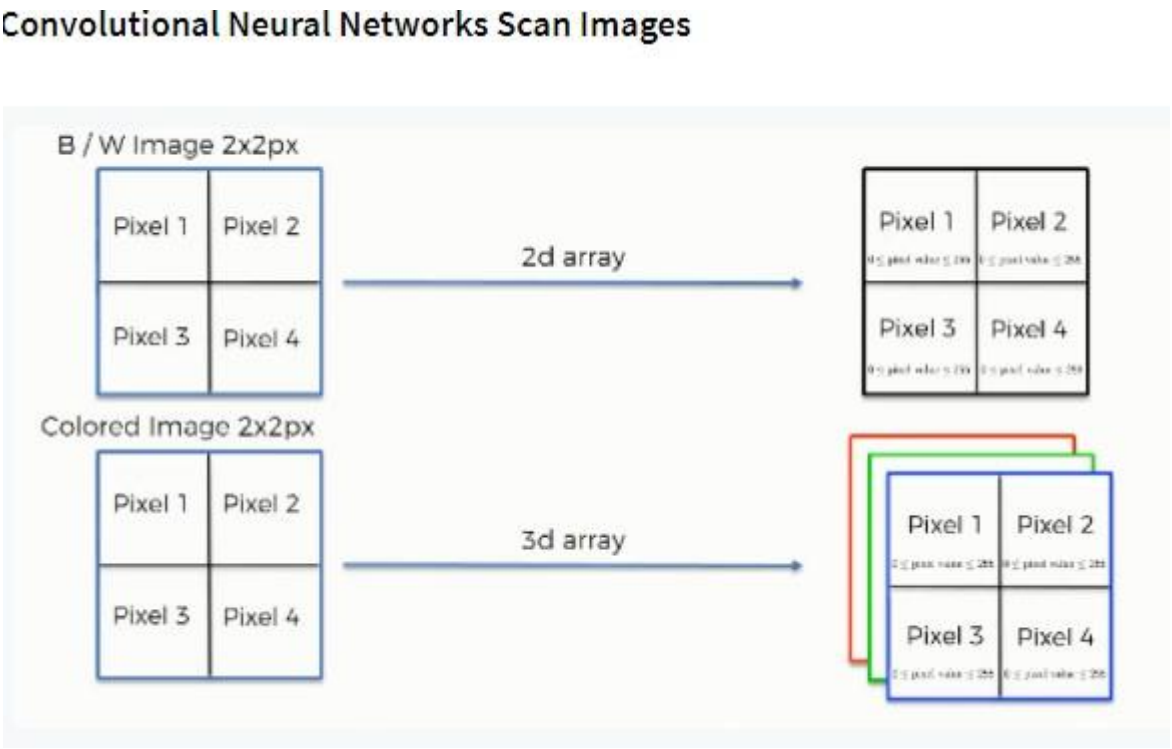


Figure 4.4.2:Convolutional Neural Networks Scan Images

Step 2: Pooling Layer

In this part, we'll cover pooling and will get to understand exactly how it generally works. Our nexus here, however, will be a specific type of pooling; max pooling. We'll cover various approaches, though, including mean (or sum) pooling. This part will end with a demonstration made using a visual interactive tool that will definitely sort the whole concept out for you.

Step 3: Flattening

This will be a brief breakdown of the flattening process and how we move from pooled to flattened layers when working with Convolutional Neural Networks.

Step 4: Full Connection

In this part, everything that we covered throughout the section will be merged together. By learning this, you'll get to envision a fuller picture of how Convolutional Neural Networks operate and how the "neurons" that are finally produced learn the classification of images.

Summary

In the end, we'll wrap everything up and give a quick recap of the concept covered in the section. If you feel like it will do you any benefit (and it probably will), you should check out the extra tutorial in which Soft ax and Cross-Entropy are covered. It's not mandatory for the course, but you will likely come across these concepts when working with Convolutional Neural Networks and it will do you a lot of good to be familiar with them.

4.5 STEPS FOR EXECUTING THE PROJECTS

1. Import the Libraries/packages.
2. Load the dataset...
3. Fit the dataset for training purpose in the specified algorithm.
4. Apply accuracy score algorithm and get calculate accuracy and loss of the classification algorithm.
5. Predict the result.
6. Display the classification report by using classification_report algorithm which takes actual and predicted values as parameters.
7. By classification report we are getting val_loss, train_loss, val_accuracy and train_accuracy.
8. Create a Data Frame with all the DL Algorithms along with their accuracies and display them.
9. Represent the all Algorithms with their accuracy in a graphical manner.
10. Finally detect the masks either mask wearing or not.

4.6 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.

4.6.1 USE CASE DIAGRAM

In the use case diagram, we have basically one actor who is the user in the trained model. A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

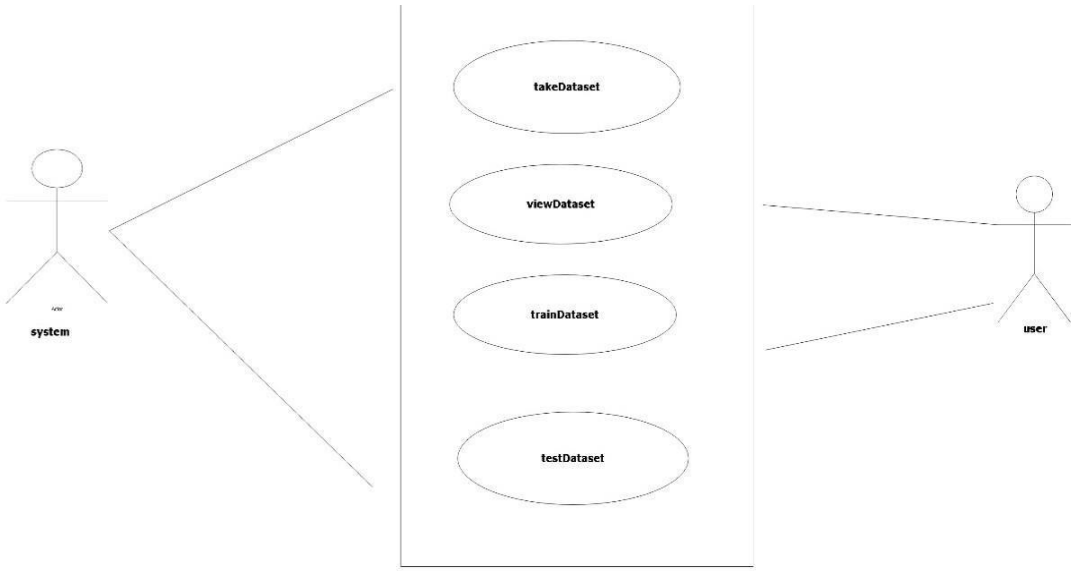


Figure 4.6.1: Use Case Diagram for Face Mask Detection Using Deep Learning and open cv

4.6.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

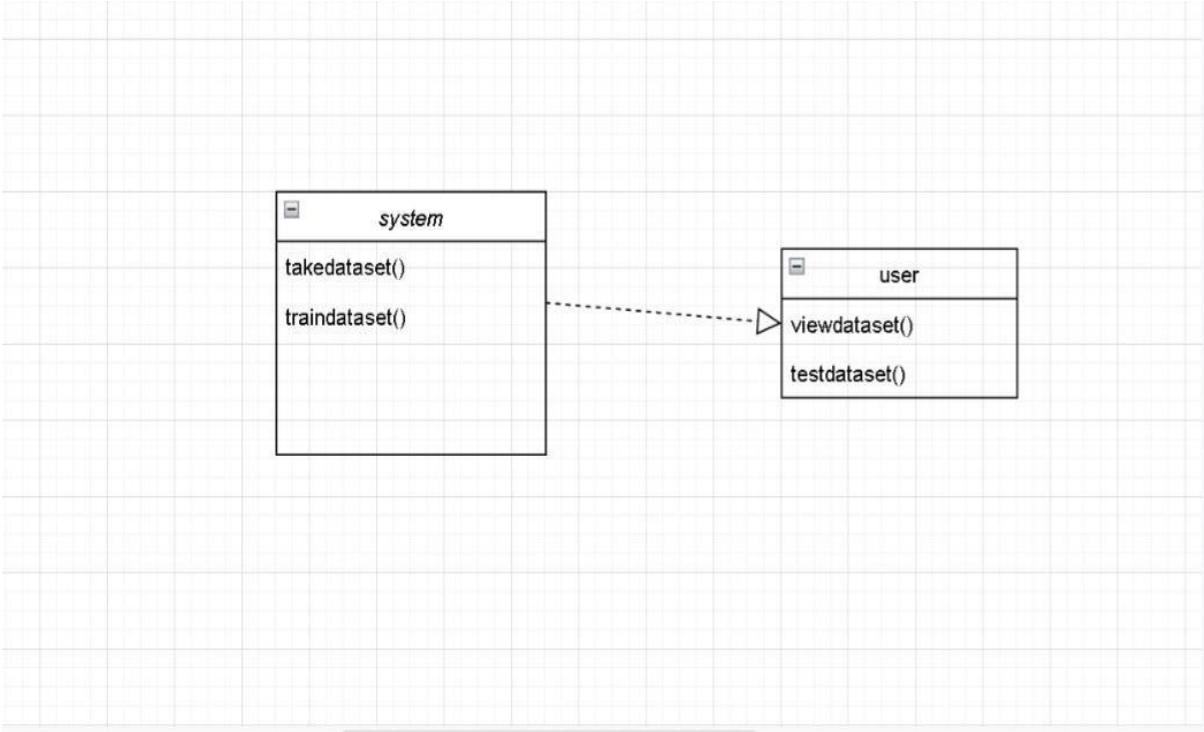


Figure 4.6.2: Class Diagram for Face Mask Detection Using Deep Learning and open cv

4.6.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

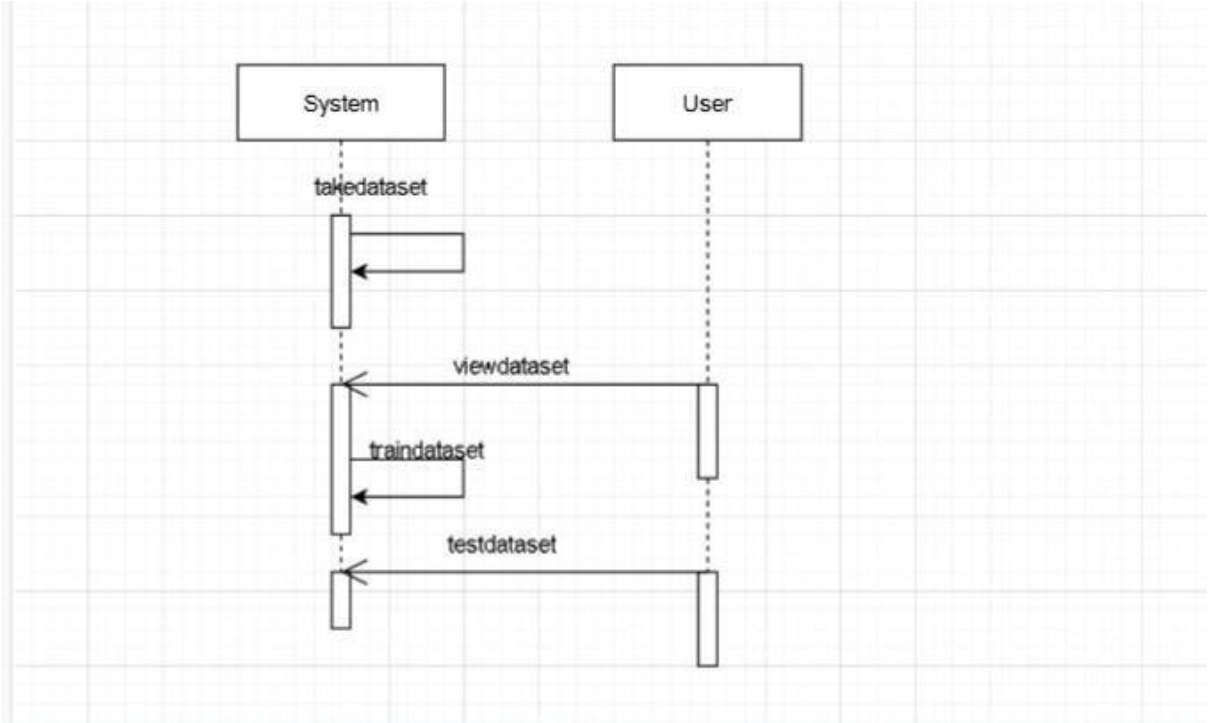


Figure 4.6.3: Sequence Diagram for Face Mask Detection Using Deep Learning and open cv

4.6.4 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

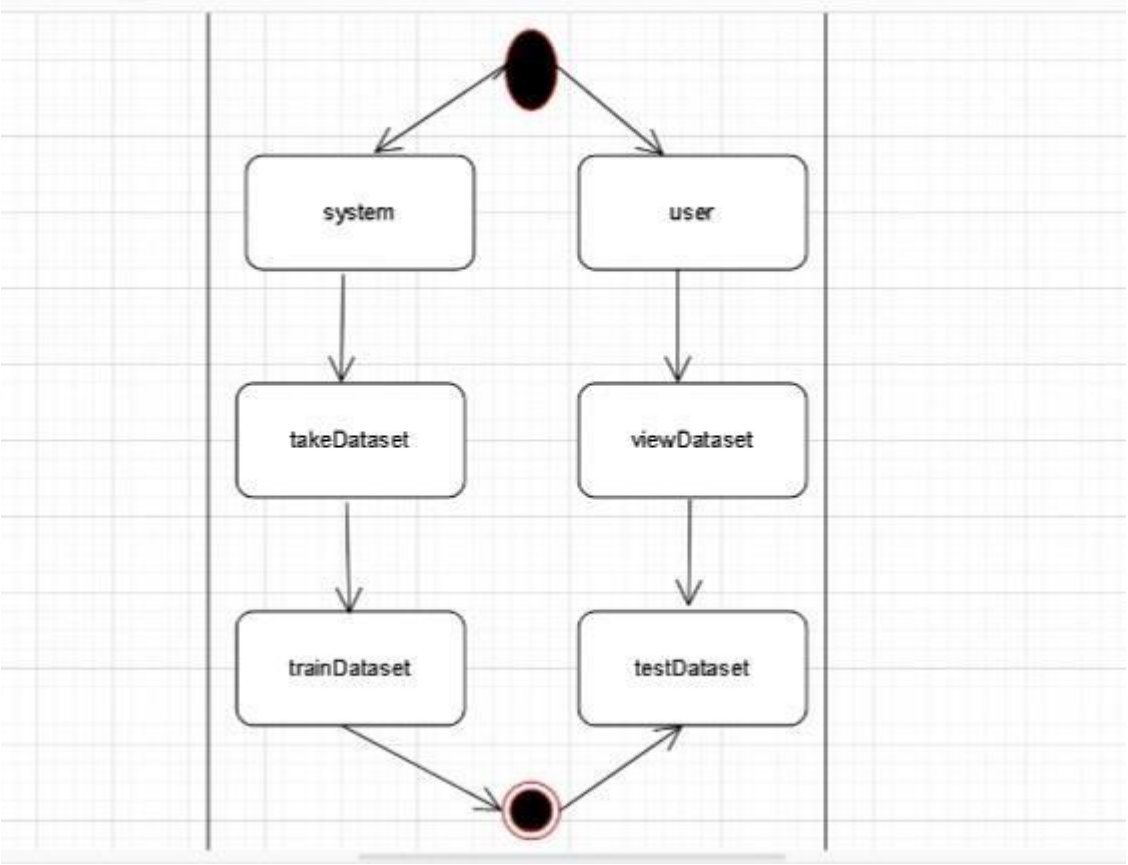


Figure 4.6.4: Activity Diagram for Face Mask Detection Using Deep Learning and open cv

4.7 INTRODUCTION TO PYTHON

Script

What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode.`

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad*, *Microsoft WordPad*, *Microsoft Word*, or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled).

Python

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it for you.

Python concepts

If you're not interested in the the hows and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it's a great one to start programming with.

- Open source general-purpose language.
- Object Oriented, Procedural, Functional.
- Easy to interface with C/ObjC/Java/Fortran.
- Easy-is to interface with C++ (via SWIG).
- Great interactive environment.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Possum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and UNIX shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Possum still holds a vital role in directing its progress.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.

- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs. Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of "thing" each data value is:

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a "float" type. This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn't require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your

program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn't matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it's more accurate to keep track of decimals so it automatically calculates the result as a floating point number.

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them.

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as **read-only** lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive. The normal mode is the mode where the scripted and finished .pie files are run in the Python interpreter. Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole.

Python libraries

1. Requests. The most famous http library written by Kenneth remits. It's a must have for every python developer.

2. Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won't use any other.

3. Python. A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.

4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

5. SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.

6. Beautiful Soup. I know it's slow but this xml and html parsing library is very useful for beginners.

7. Twisted. The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.

8. Numbly. How can we leave this very important library? It provides some advance math functionalities to python.

9. Skippy. When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

10. Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

11. Pygmy. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.

12. Piglet. A 3d animation and game creation engine. This is the engine in which the famous python port of mine craft was made

13. Pit. A GUI toolkit for python. It is my second choice after python for developing GUI's for my python scripts.

14. Pit. Another python GUI library. It is the same library in which the famous Bit torrent client is created.

15. Scaly. A packet sniffer and analyzer for python made in python.

16. Pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. Notch. Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

18. Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. Simply. Simply can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. I Python. I just can't stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

Numpy

Numpy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In numbly dimensions are called *axes*. The number of axes is *rank*.

- Offers Matlab-ish capabilities within Python
- Fast array operations
- 2D arrays, multi-D arrays, linear algebra etc.

Matplotlib

- High quality plotting library.

Python class and objects

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently "the big thing" in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects. Here’s a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it. To elaborate, by sub classing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components. As we’ve seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses. The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python's object model and the object interfaces are more consistent to Python's built-in objects, hence overloading is easier to learn and use.User-made classes can override nearly all of Python's built-in operation methods.

Exceptions

I've talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program's flow, either intentionally or due to errors. They are special events that can occur due to an error, e.g. trying to open a file that doesn't exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don't occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

- Accessing a non−existent dictionary key will raise a Key Error exception.
- Searching a list for a non−existent value will raise a Value Error exception.
- Calling a non−existent method will raise an Attribute Error exception.
- Referencing a non−existent variable will raise a Name Error exception.
- Mixing data types without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files. This is the most common way to use exceptions. When programming with the Python command line interpreter, you don't need to worry about catching exceptions. Your program is usually short enough to not be hurt too much

if an exception occurs. Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don't put any overhead on the code processing. Because exceptions aren't supposed to happen very often, they aren't processed until they occur.

Exceptions can be thought of as a special form of the if/elf statements. You can realistically do the same thing with if blocks as you can with exceptions. However, as already mentioned, exceptions aren't processed until they occur; if blocks are processed all the time.Proper use of exceptions can help the performance of your program. The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing. Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won't spend too much time talking about this, but Python does allow for a programmer to create his own exceptions. You probably won't have to do this very often but it's nice to have the option when necessary. However, before making your own exceptions, make sure there isn't one of the built-in exceptions that will work for you.They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free. Making your own exceptions involves object-oriented programming, which will be covered in the next chapter.To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.To make a custom exception, simply inherit the base exception and define what it will do.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the

very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a *module*; definitions from a module can be *imported* into other modules or into the *main* module.

Testing code

- As indicated above, code is usually developed in a file using an editor.
- To test the code, import it into a Python session and try to run it.
- Usually there is an error, so you go back to the file, make a correction, and test again.
- This process is repeated until you are satisfied that the code works.
- This entire process is known as the development cycle.
- There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.
- This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task. To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values.There are three types of functions in python:

Help (), min (), print ().

Python Namespace

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a name spacing system from daily life, i.e. the naming of people in first name and family name (surname).

An example is a network: each network device (workstation, server, printer,) needs a unique name and address. Yet another example is the directory structure of file systems. The same file name can be used in different directories, the files can be uniquely accessed via the pathnames. Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs. Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- **global names** of a module
- **local names** in a function or method invocation
- **built-in names**: this namespace contains built-in functions (e.g. abs(), camp(), ...) and built-in exception names

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML?

The Extensible Markup Language XML is a markup language much like HTML or SGML. This is recommended by the World Wide Web Consortium and available as an open standard. XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone. XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.The two

most basic and broadly used APIs to XML data are the SAX and DOM interfaces. Simple API for XML SAX: Here, you register callbacks for events of interest and then let the parser proceed through the document. This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory. Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python Web Frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format tinplating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is

amenable to extensions provided by other packages.

Comparing web frameworks

There is also a repository called compare-python-web-frameworks where the same web application is being coded with varying Python web frameworks, tinplating engines and object.

Web framework resources

- When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
- Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
- What is a web framework? Is an in-depth explanation of what web frameworks are and their relation to web servers?
- Jingo vs. Flash vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
- This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
- Python's web frameworks benchmarks is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
- What web frameworks do you use and why are they awesome? Is a language agnostic Reedit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
- This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?" The votes aren't as important as the list of the many frameworks that are available to Python developers.

5.IMPLEMENTATION

5.1 DESCRIPTION

1) UPLOAD MASK DETECTION DATASET:

Upload the (face_mask_detection) dataset.

2) VIEW:

The facial images of a people with or without wearing mask is displayed.

3) TRAINING THE MODEL:

After upload the dataset to the system to train the model by applying the CNN algorithm and generate the model which will learn from the values of the trained dataset. Display the graphical report how much we can get the accuracy and loss of a data.

4) ACCURACY AND LOSS GRAPH:

In graph X-axis represents EPOCH and y-axis represents accuracy and loss values.

5) TESTING:

Upload a live video in which the model detects if a person is wearing a mask or not by using a Web Cam.

5.2 SOURCE CODE

File name: train_ mask_detector

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers  import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os
```

```
INIT_LR = 1e-4

EPOCHS = 20

BS = 32

DIRECTORY = "dataset"

CATEGORIES = ["with_mask", "without_mask"]

print("[INFO] loading images...")

data = []

labels = []

for category in CATEGORIES:

path = os.path.join(DIRECTORY, category)

for img in os.listdir(path):

img_path = os.path.join(path, img)

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

image = img_to_array(image)

image = preprocess_input(image)

data.append(image)

labels.append(category)


lb = LabelBinarizer()

labels = lb.fit_transform(labels)

labels = to_categorical(labels)

data = np.array(data, dtype="float32")

labels = np.array(labels)

(trainX, testX, trainY, testY) = train_test_split(data, labels,test_size=0.20, stratify=labels,
random_state=42)
```

```
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")
baseModel = MobileNetV2(weights="imagenet", include_top=False,
input_tensor=Input(shape=(224, 224, 3)))
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)
model = Model(inputs=baseModel.input, outputs=headModel)


for layer in baseModel.layers:
layer.trainable = False
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
metrics=["accuracy"])
```

```
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)

print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)
predIdxs = np.argmax(predIdxs, axis=1)
print(classification_report(testY.argmax(axis=1), pqqredIdxs,
target_names=lb.classes_))

print("[INFO] saving mask detector model...")
model.save("model/mask_detector.model", save_format="h5")

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("model/plot.png")
```

File name: detect_mask_video.py

```
# import the necessary packages
import random
import pygame as pygame
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os
from playsound import playsound
def detect_and_predict_mask(frame, faceNet, maskNet):
	(h, w) = frame.shape[:2]
	blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),(104.0, 177.0, 123.0))
	faceNet.setInput(blob)
	detections = faceNet.forward()
	print(detections.shape)
	faces = []
	locs = []
	preds = []
	for i in range(0, detections.shape[2]):
		confidence = detections[0, 0, i, 2]
		if confidence > 0.5:
			box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
			(startX, startY, endX, endY) = box.astype("int")
			(startX, startY) = (max(0, startX), max(0, startY))
			(endX, endY) = (min(w - 1, endX), min(h - 1, endY))
			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)
                faces.append(face)
                locs.append((startX, startY, endX, endY))

    if len(faces) > 0:
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)
    return (locs, preds)
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
maskNet = load_model(r"mask_detector.model")
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
flag=0;
det=0
while True:
    frame = vs.read()
    flag +=1
    frame = imutils.resize(frame, width=400)
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
    for (box, pred) in zip(locs, preds):
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred
        print("pred value==========",pred)
        # label=['Mask','No Mask']
        # if label=='No Mask':
        if pred[0] < pred[1]:
            det +=1
            # if flag==10 and det>=2:
            #     flag=0
            #     det=0
            path = r"siren/alert"
```

```
                    file = os.path.join(path, random.choice(os.listdir(path)))
                    pygame.mixer.init()
                    pygame.mixer.music.load(file)
                    pygame.mixer.music.play()
                    # playsound(r'preview (mp3cut.net).mp3', block=False)
                label = "Mask" if mask > withoutMask else "No Mask"
                color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
                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)
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
            break
cv2.destroyAllWindows()
vs.stop()
```

6. SCREENSHOTS

6.1 INSTALL THE DEPENDENCIES

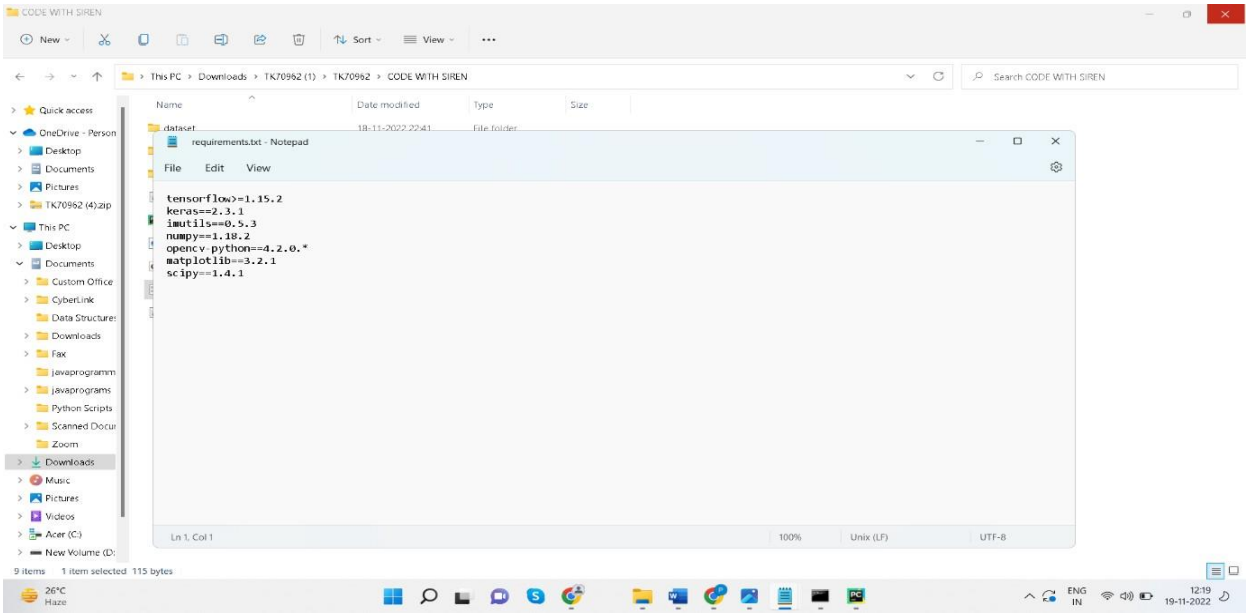


Figure 6.1 Requirements of face mask detection using deeplearning and opencv

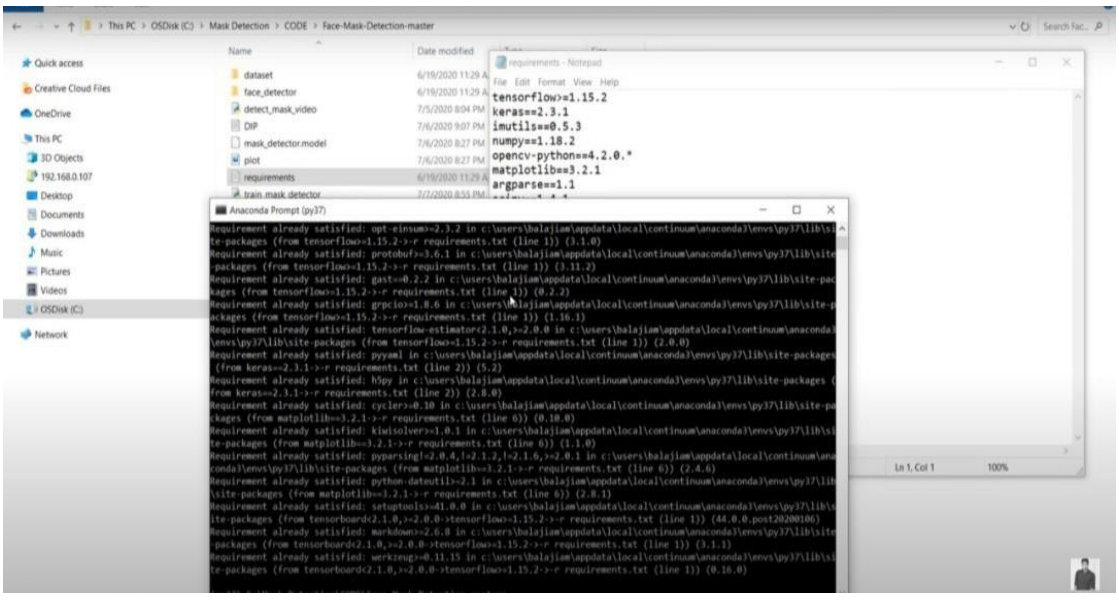


Figure 6.1.1 Installing the packages

6.2 UPLOAD DATASET

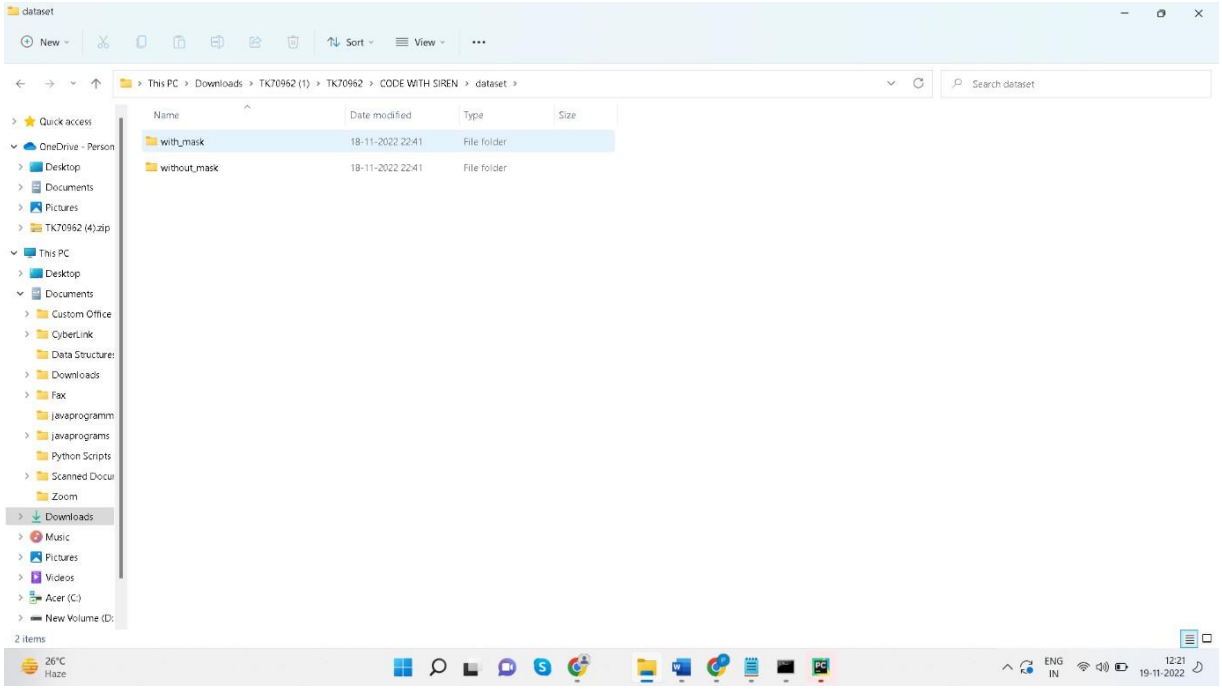


Figure 6.2 Uploading dataset

6.3 DATA PREPROCESSING

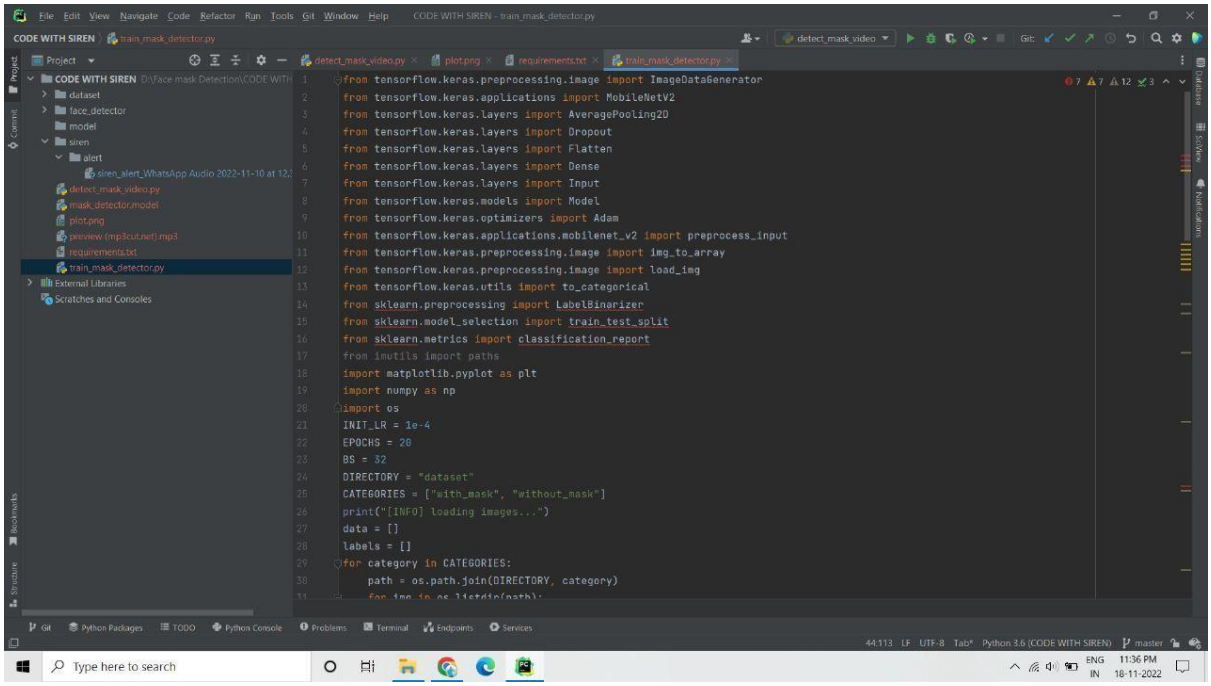


Figure 6.3 Training the dataset

6.4 TRAINING LOSS AND ACCURACY GRAPH:

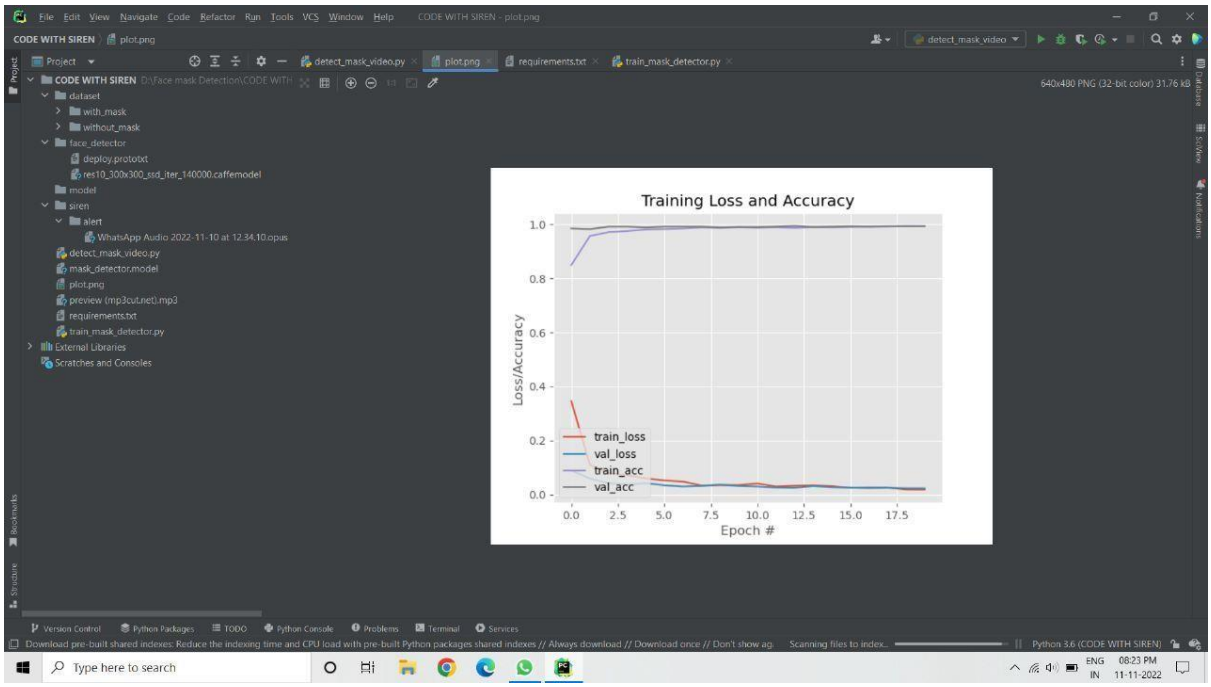
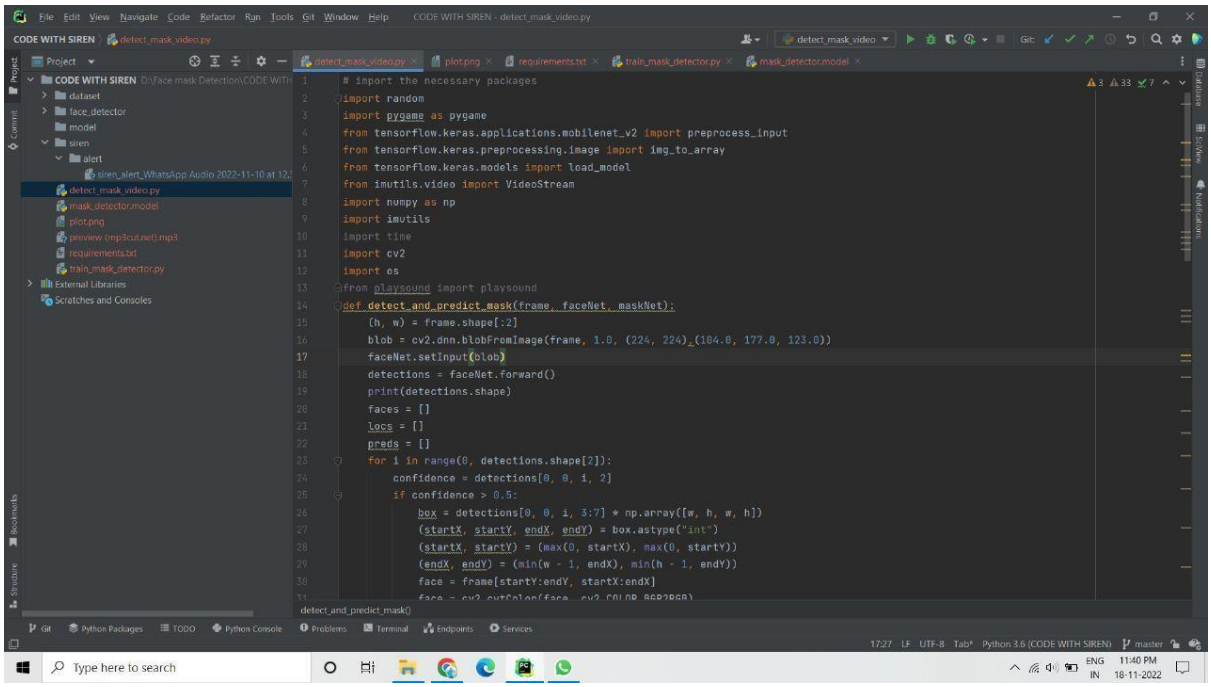


Figure 6.4 Loss and accuracy graph

6.5 USE MODEL IN REAL TIME WEBCAM



```
# import the necessary packages
import random
import pygame as pygame
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os
from playsound import playsound
def detect_and_predict_mask(frame, faceNet, maskNet):
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224), (104.0, 177.0, 123.0))
    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)
    faces = []
    locs = []
    preds = []
    for i in range(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > 0.5:
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
            face = frame[startY:endY, startX:endX]
```

Fig 6.5 detect_mask_video.py code

6.6 OUTPUT

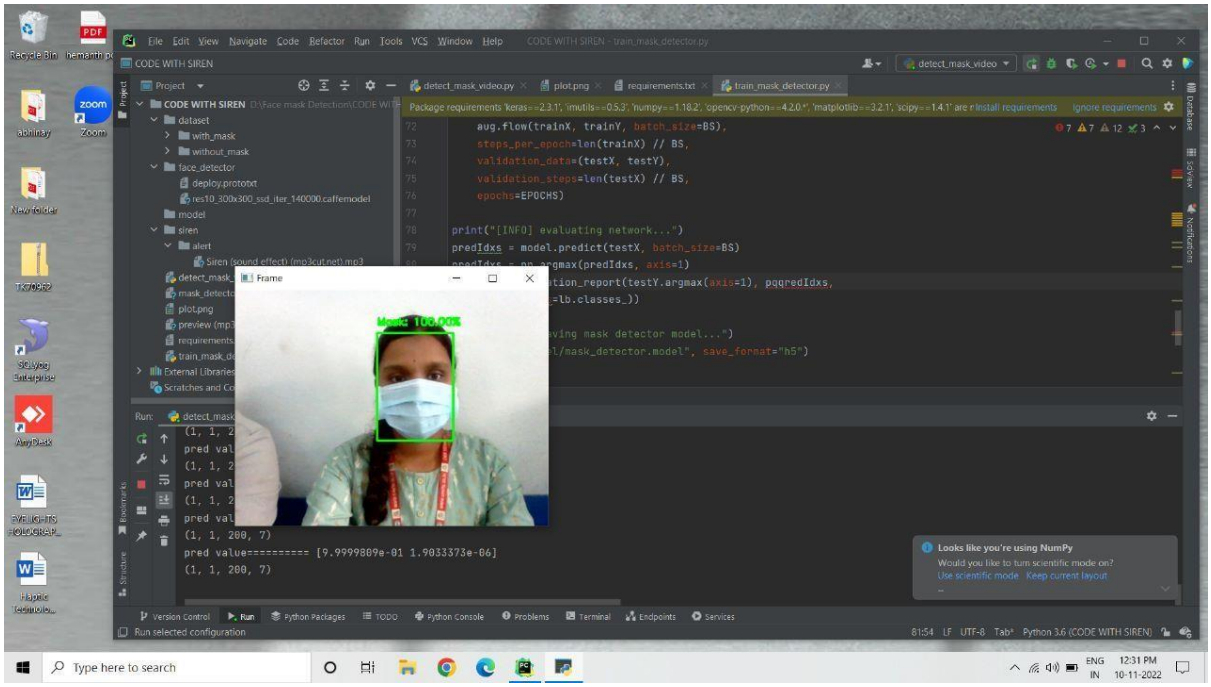


Figure 6.6.1 Output with mask

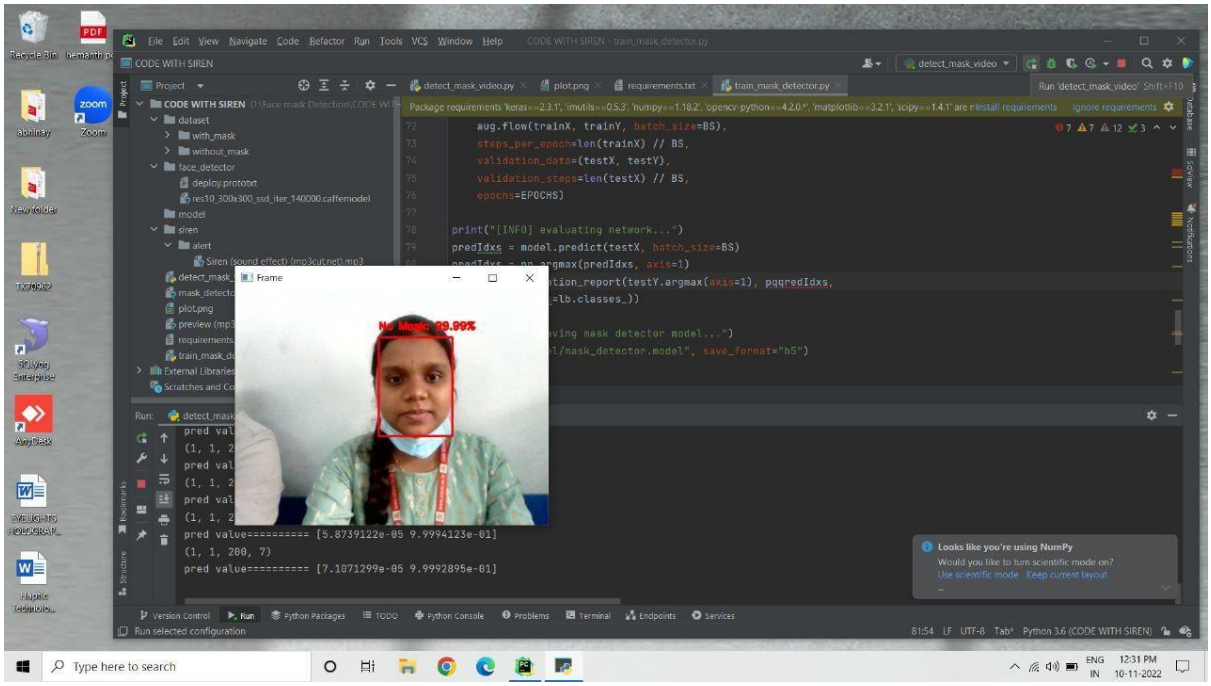


Figure 6.6.2 Output with no mask and gives voice alert.

7. TESTING

7.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.1.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.1.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.1.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user

manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.1.4 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7.1.5 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

7.1.6 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

UNIT TESTING:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

7.1.7 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

7.2 TEST CASES

Input	Output	Result
Input image	Model recognize the face mask successfully	Success

Test cases Model building:

S.NO	Test cases	I/O	Expected O/T	Actual O/T	P/F
1	Read the dataset.	Dataset path.	Dataset need to read successfully.	Dataset fetched successfully.	P
2	Performing pre-processing on the dataset	Pre-processing part takes place	Pre-processing should be performed on dataset	Pre-processing successfully completed.	P
3	Model Building	Model Building for the clean data	Need to create model using required algorithms(CNN-MobileNet)	Model Created Successfully.	P
6	Face Mask detection	Video streaming	Face mask detected successfully	Model recognize the face mask successfully	P

8. CONCLUSION & FUTURE SCOPE

8.1 PROJECT CONCLUSION:

In this application, a two-stage Face Mask Detector was presented. The first stage uses a pretrained caffe model for robust face detection, after comparing its performance with Dib and CNN. An unbiased dataset of masked and unmasked faces was created. The second stage involved training three different lightweight Face Mask Classifier models on the created dataset and based on performance, the deploy prototxt based model was selected for classifying faces as masked or non-masked. Furthermore, Centroid Tracking was added to our algorithm, which helped improve its performance on video streams. In times of the covid-19 pandemic, with the world looking to return to normalcy and people resuming in-person work, this system can be easily deployed for automated monitoring of the use of face masks at workplaces, which will help make them safer.

8.2 FUTURE SCOPE:

The current system is evaluated with different classifiers. After performing the initial analysis, the system classifies every person as “wearing a mask” or flags as “not wearing a mask” and sends an instant alert, so you can take further action — dispatch a public audio announcement, send a custom message to a digital screen, or a personalized message to the person’s phone.

9. REFERENCES

[1] Dalai, N., and Trigs, B., Histograms of oriented gradients for human detection, CVPR '05, 2005.

[2]Deng, J., Dong, W., Soccer, R., Li, L.-J., Li, K., and Fei-Fei, L., ImageNet: A Large-Scale Hierarchical Image Database, IEEE Computer Vision and Pattern Recognition (CVPR), 2009.

[3] Deng, J., Guo, J., Ververas, E., Kotsia, I., and Zafeiriou, S., Retina Face: Single-Shot Multi-Level Face Localisation in the Wild, 2020,

[4] IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 5202-5211, doi: 10.1109/CVPR42600.2020.00525.

[5] Dong, C., Loy, C.C., Tang, X., Accelerating the Super-Resolution Convolutional Neural Network, in Proceedings of European Conference on Computer Vision (ECCV), 2016.

[6] Ejaz, M.S., Islam, M.R., Sifatullah, M., Sarker, A., Implementation of principal component analysis on masked and non-masked face recognition, 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), 2019,

[7] pp. 1–5. Girshick, R., Donahue, J., Darrell, T., and Malik, J., Rich feature hierarchies for accurate object detection and semantic segmentation, Proceedings of the IEEE Conference on Computer Vision and pattern recognition, 2014,

[8] Girshick, R., Fast R-CNN, Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.

[9] Glorot, X., Bengio, Y., Understanding the difficulty of training deep feedforward neural networks, Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS), 2010.

[10] He, K., Zhang, X., Ren, S., and Sun, J., Deep residual learning for image recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770– 778.

[11] Howard, J., Huang, A., Li, Z., Tufekci, Z., Zdimal, V., van der Westhuizen, H., von Delft, A., Price, A., Fridman, L., Tang, L., Tang, V., Watson, G.L., Bax, C.E., Shaikh, R., Questier, F., Hernandez, D., Chu, L.F., Ramirez, C.M., Rimoin, A.W., Face Masks Against COVID-19: An Evidence Review. Preprints 2020, 2020040203, (doi: 10.20944/preprints202004.0203.v1).

[12] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q., Densely Connected Convolutional Networks, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 2261-2269, [13] T., Laine, S., and Aila, T., A StyleBased

Generator Architecture for Generative Adversarial Networks, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019.

[14] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998.

[15] Lin, T., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S.J., Feature pyramid networks for object detection, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, no. 2, 2017a, p. 4.

[16] Lin, T., Goyal, P., Girshick, R., He, K., and Dollár, P., Focal Loss for Dense Object Detection, 2017 IEEE International Conference on Computer Vision (ICCV), Venice, 2017b, pp. 2999-3007, doi: 10.1109/ICCV.2017.324.

[17] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A.C., SSD: Single shot multibox detector, European Conference on Computer Vision, Springer, 2016, pp. 21–37.

[18] Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021) (in press), A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic, Measurement: Journal of the International Measurement Confederation, 167.

9.1.GITHUB LINK

https://github.com/HemanthNaidu1237/FacemaskDetection