Automatic Face Mask Detection Using Python

Submitted by

Pujari Dhakshith Kumar Reddy- 210420244040

Takellapati Abhiram Choudary- 210420244056

V. Siva Manoj Kumar- 210420244061

in partial fulfilment for award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND BUSINESS SYSTEM

CHENNAI INSTITUTE OF TECHNOLOGY, CHENNAI- 600 069

ANNA UNIVERSITY: CHENNAI 600 025

DECEMBER 2022

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**Automatic Face Mask Detection Using Python**" is the bonafide work of Pujari Dhakshith Kumar Reddy, Takellapati Abhiram Chowdary, V Siva Manoj Kumar who carried out the project work under my supervision.

SIGNATURE

Dr. B.Sundarambal Ph.D

Head of the department

Department of computer science and business systems

Chennai Institute of technology,

Sarathy Nagar, Kundrathur,

Chennai-600069

SIGNATURE

Mrs. M Sindhu M.E

Assistant professor

Department of computer science and business systems

Chennai Institute of technology,

Sarathy nagar, kundrathur,

Chennai-60006

Submitted for viva-voce held on _____ at Chennai Institute of Technology,kundrathur.

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We convey our profound thanks and gratitude to our honorable Chairman Mr. P. Sriram, Chennai Institute of Technology, Chennai for providing an excellent academic climate, which made this endeavor possible.

We also express our gratitude to our beloved Principal Dr. A. Ramesh, who constantly nurtured our standard of education and devoted their precious time to our needs.

We are deeply indebted to pay our sincere thanks to our respectable Head of the Department and Supervisor Dr. B.Sundarambal, Department of Computer Science and Business Systems for showing some exuberant consent for our project and providing us with all the facilities in the department to complete the project.

We thank all our department teaching and non-teaching staff, who have helped directly and indirectly to complete this project on time.

Finally, we express our heartfelt and deep sense of gratitude to all faculty members in our division and to our friends for their helping hands, valuable support and encouragement during the project work.

TABLE OF CONTENTS

Automatic Face Mask Detection Using Python

Abstract:

The corona virus COVID-19 pandemic is causing a global health crisis so the effective protection methods is wearing a face mask in public areas according to the World Health Organization (WHO). The COVID-19 pandemic forced governments across the world to impose lockdowns to prevent virus transmissions. Reports indicate that wearing facemasks while at work clearly reduces the risk of transmission.

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 OpenCV to do real-time face detection from a live stream via our webcam. The dataset is to build a COVID-19 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 with the help of computer vision and deep learning.

Introduction:

The reason for this is that the virus that causes COVID-19 can be spread even before symptoms appear, by such things as coughing, sneezing, or even speaking at close range. Cloth face coverings have been recommended due to their low cost

and ready availability. By using cloth face coverings, it preserves surgical masks and N-95 masks for healthcare workers who may be involved in direct care of patients with COVID-19. The importance of using face coverings in public is illustrated in the graphic seen here. If I wear my face covering to protect you from me, and you wear your face covering to protect me from you, then we can all dramatically decrease our risk of transmission of the virus that causes COVID-19. This, in conjunction with social distancing and frequent hand washing or use of hand sanitizer, will be important in limiting the spread of COVID-19 as we return to our usual activities.

The wearing of the face masks appears as a solution for limiting the spread of COVID-19. In this context, our projects Aims to create a mask detecting system which will enable us with the information using image processing that if a person is wearing mask on real time or not. A face mask detection dataset consists of with mask and without mask images. we are going to use OpenCV 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, 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 with the help of computer vision and deep learning. Here we introduce a mask face detection model that is based on computer vision and deep learning. The proposed model can be integrated with surveillance cameras to impede the COVID-19 transmission by allowing the detection of people who are wearing masks not wearing face masks. The model is integration between deep learning and classical machine learning techniques with OpenCV, tensor flow and Keras. We have used deep transfer learning for feature extractions and combined it with three classical machine learning algorithms .

Chapter 1 - LITERATURE SURVEY

[1] An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City

Network (Base Paper): COVID-19 pandemic caused by novel corona virus is continuously spreading until now all over the world. The impact of COVID-19 has been fallen on almost all sectors of development. The healthcare system is going through a crisis. Many precautionary measures have been taken to reduce the spread of this disease where wearing a mask is one of them.

In this paper, we propose a system that restricts the growth of COVID-19 by finding out people who are not wearing any facial mask in a smart city network where all the public places are monitored with Closed-Circuit Television (CCTV) cameras. While a person without a mask is detected, the corresponding authority is informed through the city network.

A deep learning architecture is trained on a dataset that consists of images of people with and without masks collected from various sources. The trained architecture achieved 98.7% accuracy on distinguishing people with and without a facial mask for previously unseen test data. It is hoped that our study would be a useful tool to reduce the spread of this communicable disease for many countries in the world.

[2] Case cade framework for masked face detection:

Accurately and efficiently detecting masked faces is increasingly meaningful, since it can be applied on tracking and identifying criminals or terrorists. As a unique face detection task, masked face detection is much more difficult because of extreme occlusions which lead to the loss of face details. Besides, there is almost no existing large-scale accurately labeled masked face dataset, which increase the difficulty of masked face detection.

The CNN-based deep learning algorithms have made great breakthroughs in many computer vision areas including face detection. In this paper, we propose a new CNN-based cascade framework, which consists of three carefully designed convolution neural networks to detect masked faces. Besides, because of the shortage of masked face training samples, we propose a new dataset called "MASKED FACE dataset" to fine-tune our CNN models. We evaluate our proposed masked face detection algorithm on the MASKED FACE testing set, and it achieves satisfactory performance.

[3] Face mask detection using Mobile Net and Global Pooling Block:

Corona virus disease is the latest epidemic that forced an international health emergency. It spreads mainly from person to person through airborne transmission. Community transmission has raised the number of cases over the world. Many countries have imposed compulsory face mask policies in public areas as a preventive action. Manual observation of the face mask in crowded places is a tedious task. Thus, researchers have motivated for the automation of face mask detection system. In this paper, we have presented a Mobile Net with a global pooling block for face mask detection. The proposed model employs a global pooling layer to perform a flatten of the feature vector. A fully connected dense

layer associated with the soft max layer has been utilized for classification. Our proposed model outperforms existing models on two publicly available face mask datasets in terms of vital performance metrics.

[4] Study of the Performance of Machine Learning Algorithms for Face Mask Detection:

Nowadays, the situation of the Covid-19 virus still intensifying throughout the world. The number of populations of each country is severely infected and deaths. One solution to prevent is to wearing a masked face. Many businesses and organization need to adapt and protect an infected person by detecting whoever does not wear masked face; however, the number of users or customers is more than staffs result in difficult checking. This paper studies the performance of the three algorithms: KNN, SVM and Mobile Net to find the best algorithm which is suitable for checking who wearing masked face in a real-time situation. The results show that Mobile Net is the best accuracy both from input images and input video from a camera (real-time).

[5] Face recognition based on modular histogram of oriented directional features:

This paper presents an illumination invariant face recognition system that uses local directional pattern descriptor and modular histogram. The proposed Modular Histogram of Oriented Directional Features (MHODF) is an oriented local descriptor that is able to encode various patterns of face images under different lighting conditions. It employs the edge response values in different directions to encode each sub-image texture and produces multi- region histograms for each image.

The edge responses are very important and play the main role for improving the face recognition accuracy. Therefore, we present the effectiveness of using different directional masks for detecting the edge responses on face recognition accuracy, such as Prewitt kernels, Kirsch masks, Sobel kernels, and Gaussian derivative masks. The performance evaluation of the proposed MHODF algorithm is conducted on several publicly available databases and observed promising recognition rates.

[6] Masked Face Recognition Using Convolution Neural Network:

Recognition from faces is a popular and significant technology in recent years. Face alterations and the presence of different masks make it too much challenging. In the real-world, when a person is uncooperative with the systems such as in video surveillance then masking is further common scenarios. For these masks, current face recognition performance degrades. An abundant number of researches work has been performed for recognizing faces under different conditions like changing pose or illumination, degraded images, etc. Still, difficulties created by masks are usually disregarded.

The primary concern to this work is about facial masks, and especially to enhance the recognition accuracy of different masked faces. A feasible approach has been proposed that consists of first detecting the facial regions. The occluded face detection problem has been approached using Multi-Task Cascaded Convolution Neural Network (MTCNN). Then facial features extraction is performed using the Google Face Net embedding model. And finally, the classification task has been performed by Support Vector Machine (SVM). Experiments signify that this mentioned approach gives a remarkable performance on masked face recognition. Besides, its performance has been also evaluated

within excessive facial masks and found attractive outcomes. Finally, a correlative study also made here for a better understanding.

[7] Control The COVID-19 Pandemic:

Face Mask Detection Using Transfer Learning Currently, in the face of the health crisis caused by the Corona virus COVID-19 which has spread throughout the worldwide. The fight against this pandemic has become an unavoidable reality for many countries. It is now a matter involving many areas of research in the use of new information technologies, particularly those related to artificial intelligence. In this paper, we present a novel contribution to help in the fight against this pandemic. It concerns the detection of people wearing masks because they cannot work or move around as usual without protection against COVID-19.

However, there are only a few research studies about face mask detection. In this work, we investigated using different deep Convolution Neural Networks (CNN) to extract deep features from images of faces. The extracted features are further processed using various machine learning classifiers such as Support Vector Machine (SVM) and K-Nearest Neighbors (K-NN). Were used and examined all different metrics such as accuracy and precision, to compare all model performances. The best classification rate was getting is 97.1%, which was achieved by combining SVM and the MobileNetV2 model. Despite the small dataset used (1376 images), we have obtained very satisfactory results for the detection of masks on the faces.

[8] 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 convolution architectures to make the algorithm as accurate as possible. These convolution 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 Convolution 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 Recognition from faces is a popular and significant technology in recent years. Face alterations and the presence of different masks make it too much challenging.

In the real-world, when a person is uncooperative with the systems such as in video surveillance then masking is further common scenarios. For these masks, current face recognition performance degrades. An abundant number of researches work has been performed for recognizing faces under different conditions like

changing pose or illumination, degraded images, etc. Still, difficulties created by masks are usually disregarded.

The primary concern to this work is about facial masks, and especially to enhance the recognition accuracy of different masked faces. A feasible approach has been proposed that consists of first detecting the facial regions. The occluded face detection problem has been approached using Multi-Task Cascaded Convolutional Neural Network (MTCNN). Then facial features extraction is performed using the Google Face Net embedding model. And finally, the classification task has been performed by Support Vector Machine (SVM). Experiments signify that this mentioned approach gives a remarkable performance on masked face recognition. Besides, its performance has been also evaluated within excessive facial masks and found attractive outcomes. Finally, a correlative study also made here for a better understanding.

[9] Deep Learning Based Assistive System to Classify COVID-19 Face Mask for Human Safety with YOLOv3:

Computer vision learning pay a high attention due to global pandemic COVID-19 to enhance public health service. During the fatality, tiny object detection is a more challenging task of computer vision, as it recruits the pair of classification and detection beneath of video illustration. Compared to other object detection deep neural networks demonstrated a helpful object detection with a superior achievement that is Face mask detection. However, accession with YOLOv3 covered by an exclusive topic which through certainly happening natural disease people get advantage. Added with face mask detection performed well by the YOLOv3 where it measures real time performance regarding a powerful GPU. whereas computation power with low memory YOLO darknet command sufficient

for real time manner. Regarding the paper section below we have attained that people who wear face masks or not, its trained by the face mask image and nonface mask image.

Under the experimental conditions, real time video data that finalized over detection, localization and recognition. Experimental results that show average loss is 0.0730 after training 4000 epochs. After training 4000 epochs map score is 0.96. This unique approach of face mask visualization system attained noticeable output which has 96% classification and detection accuracy.

[10] Deep Learning Framework to Detect Face Masks from Video Footage:

The use of facial masks in public spaces has become a social obligation since the wake of the COVID-19 global pandemic and the identification of facial masks can be imperative to ensure public safety. Detection of facial masks in video footages is a challenging task primarily due to the fact that the masks themselves behave as occlusions to face detection algorithms due to the absence of facial landmarks in the masked regions.

In this work, we propose an approach for detecting facial masks in videos using deep learning. The proposed framework capitalizes on the MTCNN face detection model to identify the faces and their corresponding facial landmarks present in the video frame. These facial images and cues are then processed by a neoteric classifier that utilizes. The MobileNetV2 architecture as an object detector for identifying masked regions.

Related Work :

We have collected and analyzed several years IEEE papers to get a refined visualization on face mask detection system using different methods. Previously the system propose a new CNN- based cascade framework, which consists of three carefully designed convolutional neural networks to detect masked faces. Besides, because of the shortage of masked face training samples, we propose a new dataset called "MASKED FACE dataset" to fine-tune our CNN models. The current study used OpenCV, Pytorch and CNN to detect whether people were wearing face masks or not. The models were tested with images and real-time video streams. Even though the accuracy of the model is around 60%, the optimization of the model is a continuous process and we are building a highly accurate solution by tuning the hyperparameters. MobileNetV2 was used to build the mobile version of the same. This specific model could be used as a use case for edge analytics.

Proposed work:

The proposed project will be able to analyze the images through camera and detects the presence of mask and create masked face images from unmasked face images by appending the mask on the face recognized on the image and store it. the project works by integration of python OpenCV tensor flow for implementing the detection of face mask. We are using R CNN which is really efficient compared to the other machine learning model. The data modeling and analysis tools, such as data mining, machine learning, have the potential to generate a knowledge-rich environment which can help to significantly improve the quality of clinical decisions.

Chapter 2 - Methodology

Proposed system

1. This system is capable to train the dataset of both persons wearing masks and without wearing masks.
2. After training the model the system can predicting whether the person is wearing the mask or not .
3. It also can access the webcam and predict the result.

Tensorflow framework:

Tensor flow is an open-source software library. Tensor flow was originally developed by researchers and engineers. It is working on the Google Brain Team within Google's Machine Intelligence research organization the purposes of conducting machine learning and deep neural networks research. It is an opensource framework to run deep learning and other statistical and predictive analytics workloads. It is a python library that supports many classification and regression algorithms and more generally deep learning.

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google, TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). Tensor Flow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. The name Tensor Flow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

OPENCV:

1. It is a cross-platform library using which we can develop real-time computer vision applications.
2. It mainly focuses on image processing, video capture and analysis including feature like face detection and object detection.
3. Currently Open CV supports a wide variety of programming languages like C++, Python, Java etc. and is available on different platforms including Windows, Linux, OS X, Android, iOS etc.
4. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations. Open CV-Python is the Python API of Open CV.
5. It combines the best qualities of Open CV C++ API and Python language.
6. OpenCV (Open-Source Computer Vision Library) is an opensource computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.
7. The library has more than 2500optimized algorithms, which includes a comprehensive set of both classic and state-of -the-art computer vision and machine learning algorithms.

8. Algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

NUMPY:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of highlevel mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Num array into Numeric, with extensive modifications. NumPy is opensource software and has many contributors.

The Python programming language was not initially designed for numerical computing, but attracted the attention of the scientific and engineering community early on, so that a special interest group called matrix-sig was founded in 1995 with the aim of defining an array computing package. Among its members was Python designer and maintainer Guido van Rossum, who implemented extensions to Python's syntax (in particular the indexing syntax) to make array computing easier.

An implementation of a matrix package was completed by Jim Fulton, then generalized by Jim Hugunin to become Numeric also variously called Numerical Python extensions or NumPy Hugunin, a graduate student at Massachusetts

Institute of Technology (MIT) joined the Corporation for National Research Initiatives (CNRI) to work on J Python in 1997 leaving Paul Dubois of Lawrence Livermore National Laboratory (LLNL) to take over as maintainer.

In early 2005, NumPy developer Travis Oliphant wanted to unify the community around a single array package and ported num-array's features to Numeric, releasing the result as NumPy 1.0 in 2006. This new project was part of SciPy. To avoid installing the large SciPy package just to get an array object, this new package was separated and called NumPy.

MATPLOT:

Matplot is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, WX Python, Qt, or GTK+. There is also a procedural "Pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012 and further joined by Thomas Caswell.

IPYTHON

What exactly is Python? You may be wondering about that. You may be referring to this book because you wish to learn editing but are not familiar with editing languages. Alternatively, you may be familiar with programming languages such

as C, C ++, C #, or Java and wish to learn more about Python language and how it compares to these "big word" languages.

PYTHON CONCEPTS

Python was developed into an easy-to-use programming language. It uses English words instead of punctuation, and has fewer syntax than other languages. Python is a highly developed, translated, interactive, and object-oriented language.

Python translated - Interpreter processing Python during launch. Before using your software, you do not need to install it. This is similar to PERL and PHP editing languages.

Python interactive - To write your own applications, you can sit in Python Prompt and communicate directly with the interpreter.

Python Object-Oriented - Python supports the Object-Oriented program style or method, encoding the code within objects.

Python is a language for beginners - Python is an excellent language for beginners, as it allows for the creation of a variety of programs, from simple text applications to web browsers and games.

Python Features

Python features include –

1. Easy-to-learn - Python includes a small number of keywords, precise structure, and well-defined syntax. T This allows the student to learn the language faster
2. Easy to read - Python code is clearly defined and visible to the naked eye.
3. Easy-to-maintain - Python source code is easy to maintain.

4. Standard General Library - Python's bulk library is very portable and shortcut compatible with UNIX, Windows, and Macintosh.
5. Interaction mode - Python supports interaction mode that allows interaction testing and correction of captions errors.
6. Portable - Python works on a variety of computer systems and has the same user interface for all.
7. Extensible - Low-level modules can be added to Python interpreter. These modules allow system developers to improve the efficiency of their tools either by installing or customizing them.
8. Details - All major commercial information is provided by Python ways of meeting.
9. GUI Programming - Python assists with the creation and installation of a user interface for images of various program phones, libraries, and applications, including Windows MFC, Macintosh, and Unix's X Window.
10. Scalable - Major projects benefit from Python building and support, while Shell writing is not Aside from the characteristics stated above,

Python offers a long list of useful features, some of which are described below. –

- It supports OOP as well as functional and structured programming methodologies.
- It can be used as a scripting language or compiled into Byte-code for large-scale application development.
- It allows dynamic type verification and provides very high-level dynamic data types.
- Automatic garbage pickup is supported by IT.

Pandas

Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance & productivity for users.

Pandas is a python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, realworld data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible opensource data analysis/manipulation tool available in any language. It is already well on its way toward this goal.

Explore data analysis with Python. Pandas Data Frames make manipulating your data easy, from selecting or replacing columns and indices to reshaping your data.

Pandas is well suited for many different kinds of data:

Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet Ordered and unordered (not necessarily fixed-frequency) time series data. Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels Any other form of observational / statistical data sets. The data need not be labeled at all to be placed into a Pandas data structure.

KERAS

KERAS is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages.

It also has extensive documentation and developer guides. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. Keras is a minimalist Python library for deep learning that can run on top of Theano or Tensor Flow.

It was developed to make implementing deep learning models as fast and easy as possible for research and development.

FOUR PRINCIPLES:

- Modularity: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.
- Minimalism: The library provides just enough to achieve an outcome, no frills and maximizing readability.
- Extensibility: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.

- Python: No separate model files with custom file formats. Everything is native Python. Keras is designed for minimalism and modularity allowing you to define deep learning models and run them on top of a Theano or TensorFlow backend very quickly

Machine Learning approaches:

- Viola–Jones object detection framework based on HAAR Features
- Scale-invariant feature transform (SIFT)
- Histogram of oriented gradients (HOG) features

Machine learning (ML)is the study of computer algorithms that improve automatically through experience. Itis seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks. Machine learning is closely related to computational statistics, which focuses on making predictions using computers.

The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

DEEP LEARNING:

1. Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions.
2. Deep learning AI is able to learn without human supervision, drawing from data that is both unstructured and unlabeled.
3. In this, face mask detection is built using Deep Learning technique called as Convolution Neural Networks (CNN).

Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower-level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features. Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features.

Chapter 3 – DESIGN:

UML Diagrams:

A UML diagram is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains graphical elements (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts.

The kind of the diagram is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines.

UML specification does not preclude mixing of different kinds of diagrams, e.g. to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced. At the same time, some UML Tools do restrict set of available graphical elements which could be used when working on specific type of diagram. UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams.

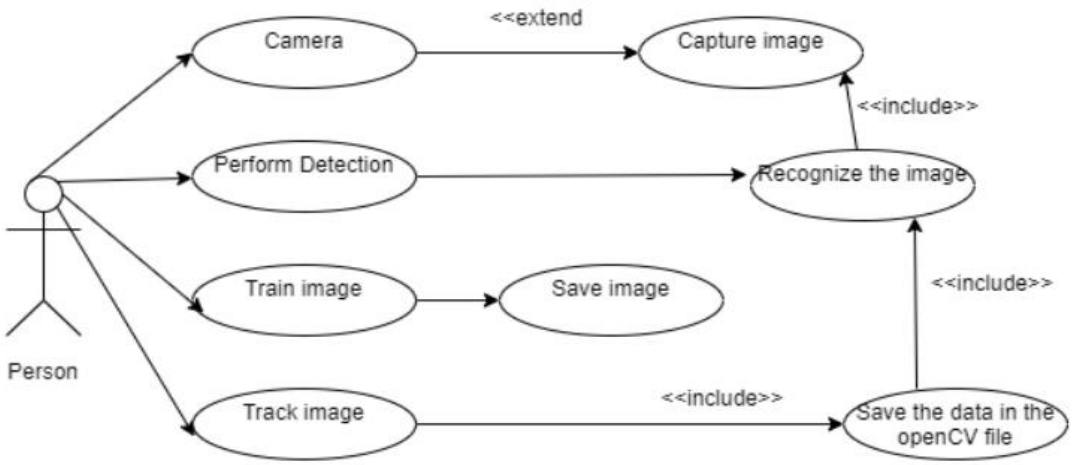
Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

Use Case Diagram

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems.
- Goals that your system or application helps those entities (known as actors) achieve.
- The scope of your system



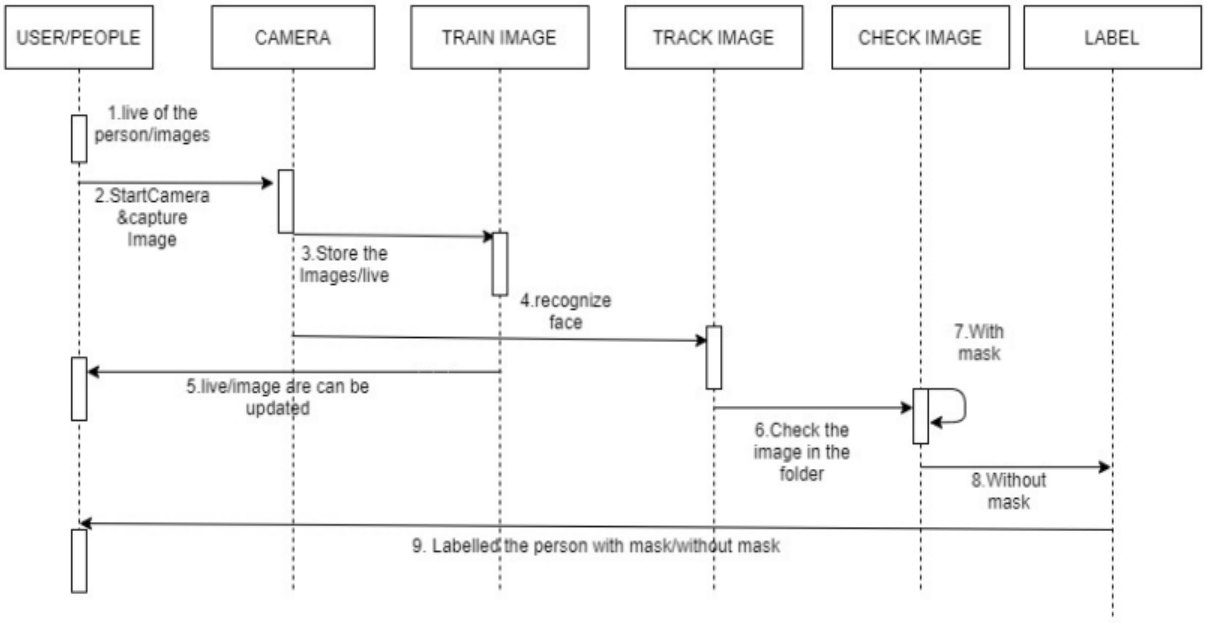
Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used

by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

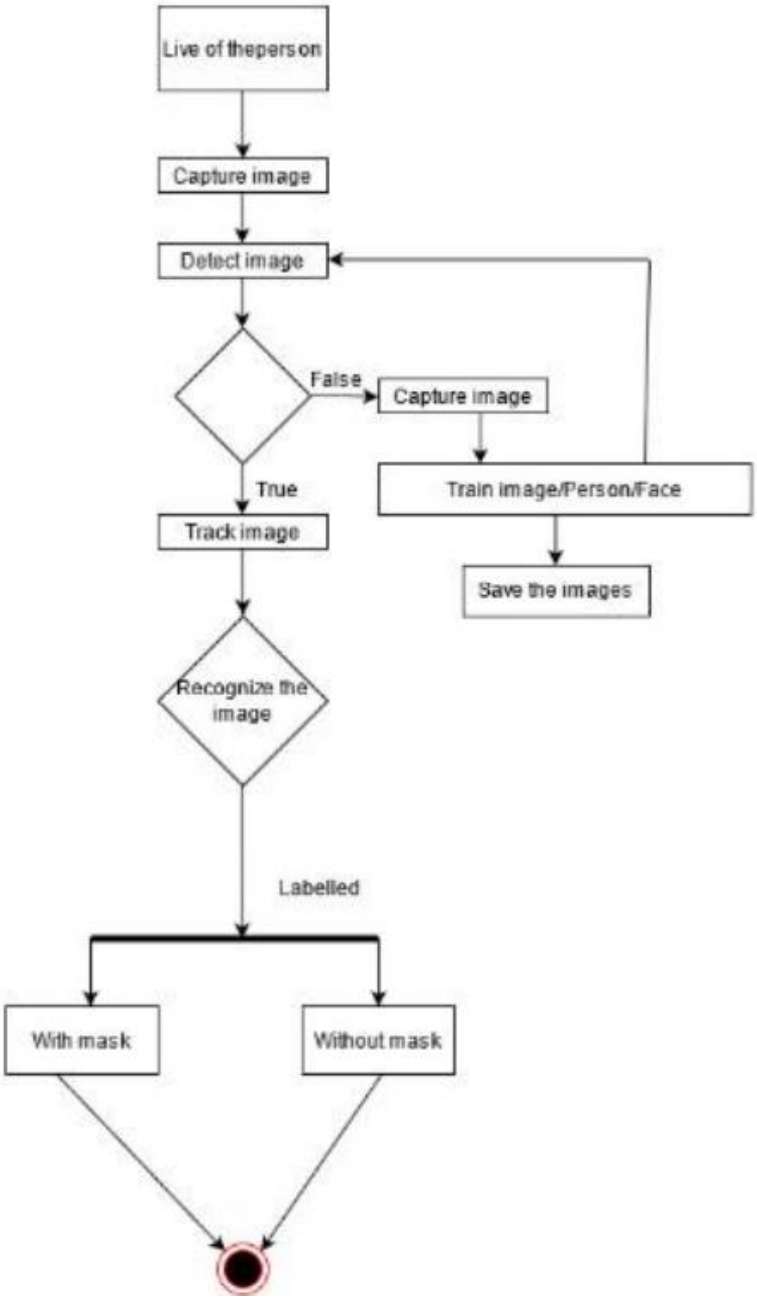
- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario



Activity Diagram

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system.

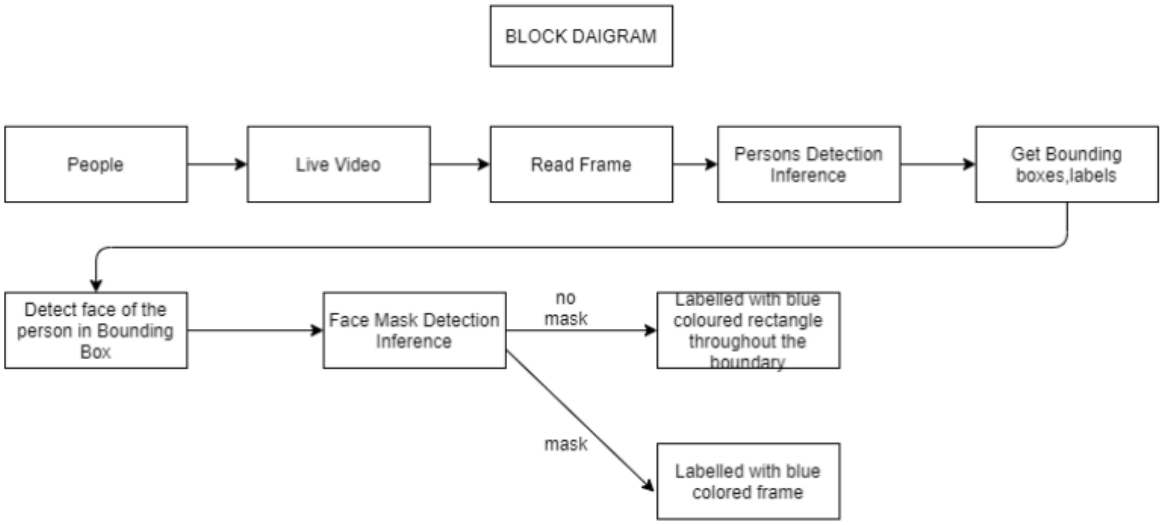
An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.



BLOCK DIAGRAM

A block diagram is a graphical representation of a system – it provides a functional view of a system. Block diagrams give us a better understanding of a system's functions and help create interconnections within it.

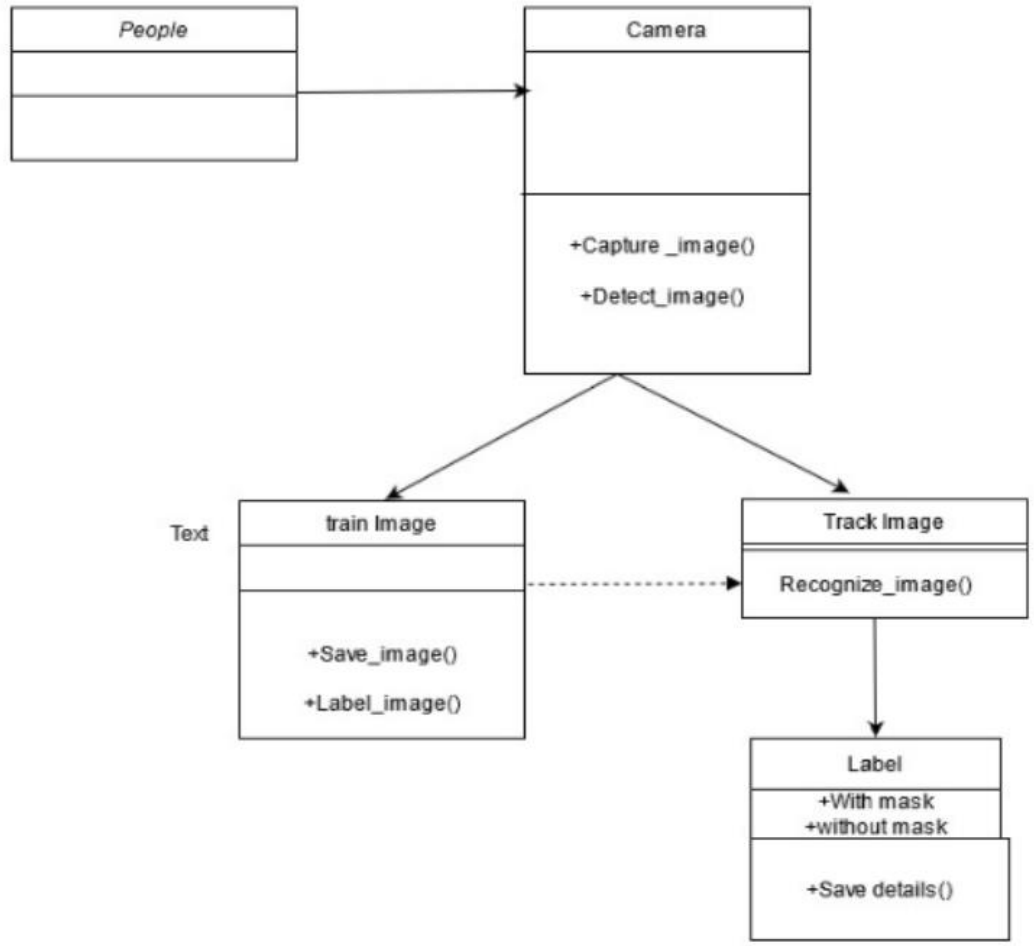
They are used to describe hardware and software systems as well as to represent processes.



CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application.

Class diagrams are the only diagrams which can be directly mapped with objectoriented languages and thus widely used at the time of construction.



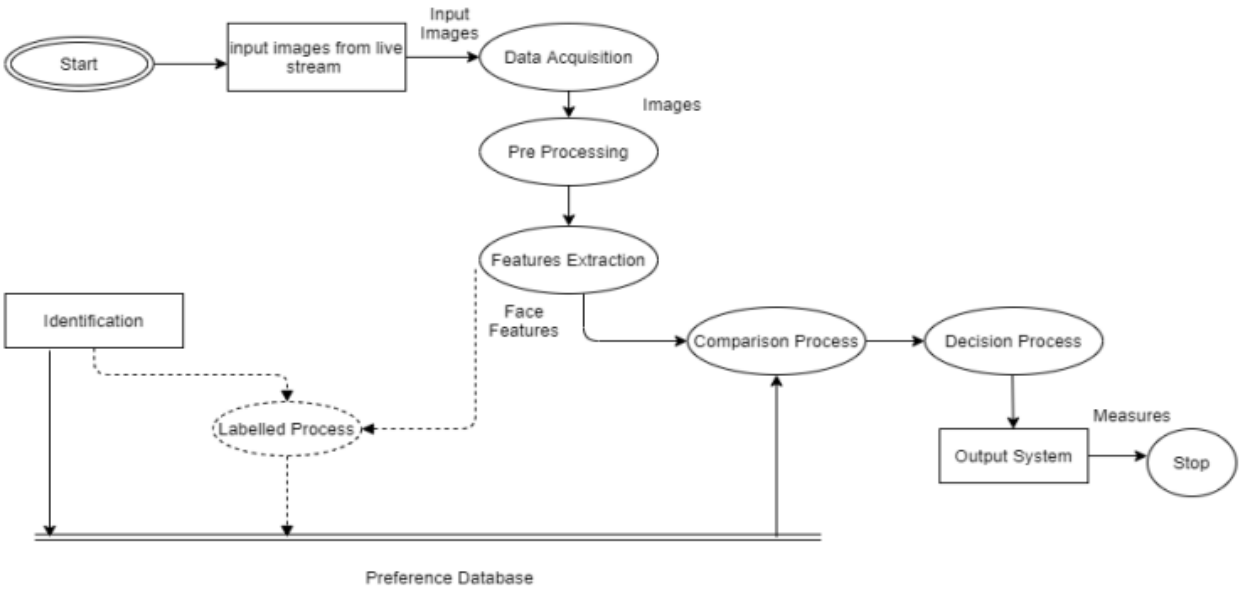
DATA FLOW DIAGRAM

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

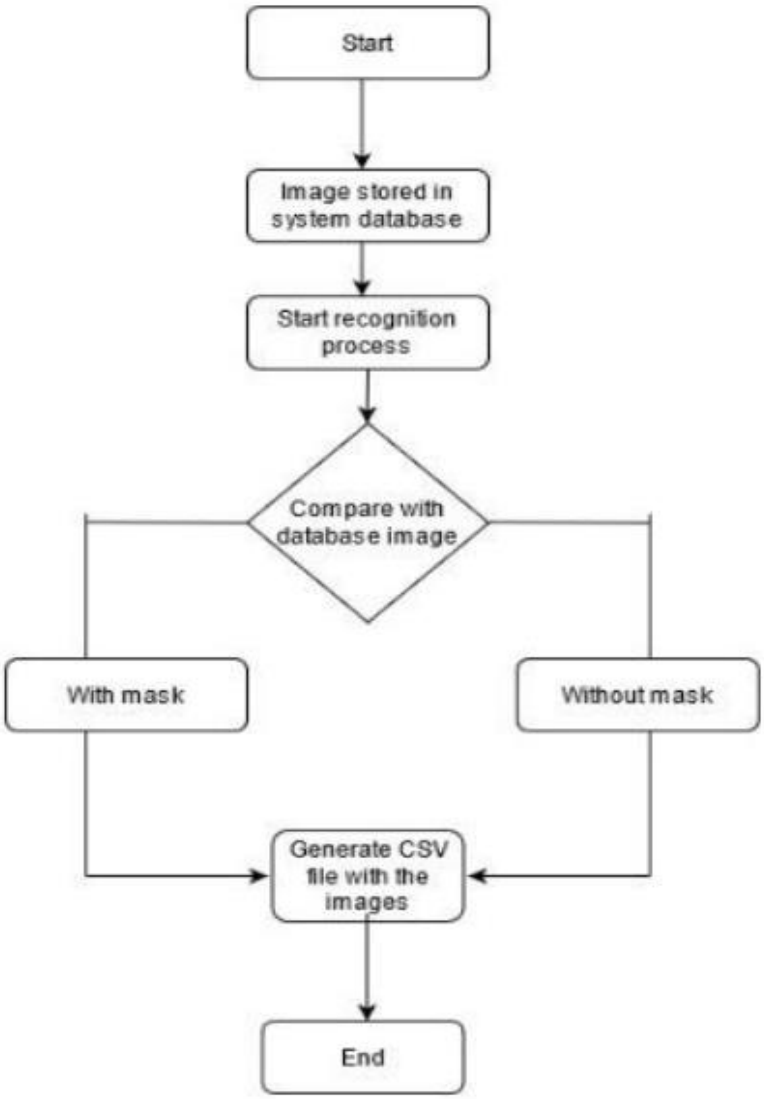
It shows how data enters and leaves the system, what changes the information, and where data is stored.

A graphical tool for defining and analyzing minute data with an active or automated system, including process, data stores, and system delays. Data Flow Data is a key and basic tool for the architecture of all other objects. Bubble-bubble or data flow graph is another name for DFD.

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



FLOW CHART DIAGRAM:



CHAPTER 4 REQUIREMENTS

Functional requirements

The primary purpose of computer results is to deliver processing results to users. They are also employed to maintain a permanent record of the results for future use.

In general, the following are many types of results:

- External results are those that are exported outside the company.
- Internal results, which are the main user and computer display and have a place within the organization.
- Operating results used only by the computer department.
- User-interface results that allow the user to communicate directly with the system.
- Understanding the user's preferences, the level of technology and the needs of his or her business through a friendly questionnaire.

Non-functional requirements

System configuration

This project can run on commodity hardware. We ran entire project on an Intel I5 processor with 8 GB Ram, 2 GB Nvidia Graphic Processor, It also has 2 cores which runs at 1.7 GHz, 2.1 GHz respectively. First part of the is training phase which takes 10-15 mins of time and the second part is testing part which only takes few seconds to make predictions and calculate accuracy.

Hardware requirements

• RAM: 4 GB

• Storage: 500 GB

• CPU: 2 GHz or faster

• Architecture: 32-bit or 64-bit

Software requirements

• Python 3.5 in Google Colab is used for data pre-processing, model training and prediction

• Operating System: windows 7 and above or Linux based OS or MAC OS

• Coding Language : Python.

CHAPTER 5 CODING:

Code To Train Mask Detector:

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

```
# initialize the initial learning rate, number of epochs to train for,
# and batch size
INIT_LR = 1e-4
EPOCHS = 20
BS = 32

DIRECTORY = r"C:\Mask Detection\CODE\Face-Mask-Detection-master\dataset"
CATEGORIES = ["with_mask", "without_mask"]

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")

data = []
labels = []

for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = load_img(img_path, target_size=(224, 224))
        image = img_to_array(image)
        image = preprocess_input(image)
```

```
        data.append(image)
        labels.append(category)

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

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

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

# construct the training image generator for data augmentation
aug = ImageDataGenerator(
        rotation_range=20,
        zoom_range=0.15,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.15,
        horizontal_flip=True,
        fill_mode="nearest")
```

```
# load the MobileNetV2 network, ensuring the head FC layer sets are
# left off
baseModel = MobileNetV2(weights="imagenet", include_top=False,
	input_tensor=Input(shape=(224, 224, 3)))

# construct the head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
	layer.trainable = False

# compile our model
print("[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("mask_detector.model", save_format="h5")

# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")
```

Code To Detect Mask Video:

```
# 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 imutils
import time
import cv2
import os

def detect_and_predict_mask(frame, faceNet, maskNet):
	# grab the dimensions of the frame and then construct a blob
	# from it
	(h, w) = frame.shape[:2]
	blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
		(104.0, 177.0, 123.0))

	# pass the blob through the network and obtain the face detections
	faceNet.setInput(blob)
	detections = faceNet.forward()
	print(detections.shape)

	# initialize our list of faces, their corresponding locations,
	# and the list of predictions from our face mask network
	faces = []
	locs = []
	preds = []

	# loop over the detections
	for i in range(0, detections.shape[2]):
```

```
# extract the confidence (i.e., probability) associated with
# the detection
confidence = detections[0, 0, i, 2]

# filter out weak detections by ensuring the confidence is
# greater than the minimum confidence
if confidence > 0.5:
        # compute the (x, y)-coordinates of the bounding box for
        # the object
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

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

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

```
                # add the face and bounding boxes to their respective
                # lists
                faces.append(face)
                locs.append((startX, startY, endX, endY))

        # only make a predictions if at least one face was detected
        if len(faces) > 0:
                # for faster inference we'll make batch predictions on *all*
                # faces at the same time rather than one-by-one predictions
                # in the above `for` loop
                faces = np.array(faces, dtype="float32")
                preds = maskNet.predict(faces, batch_size=32)

        # return a 2-tuple of the face locations and their corresponding
        # locations
        return (locs, preds)

# load our serialized face detector model from disk
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
maskNet = load_model("mask_detector.model")
```

```
# initialize the video stream
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()

# loop over the frames from the video stream
while True:
	# grab the frame from the threaded video stream and resize it
	# to have a maximum width of 400 pixels
	frame = vs.read()
	frame = imutils.resize(frame, width=400)

	# detect faces in the frame and determine if they are wearing a
	# face mask or not
	(locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

	# loop over the detected face locations and their corresponding
	# locations
	for (box, pred) in zip(locs, preds):
		# unpack the bounding box and predictions
		(startX, startY, endX, endY) = box
		(mask, withoutMask) = pred

		# determine the class label and color we'll use to draw
		# the bounding box and text
		label = "Mask" if mask > withoutMask else "No Mask"
```

```
            color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

            # include the probability in the label
            label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

            # display the label and bounding box rectangle on the output
            # frame
            cv2.putText(frame, label, (startX, startY - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
            cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
            break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

CHAPTER 6 CONCLUSION

As the technology are blooming with emerging trends the availability so we have novel face mask detector which can possibly contribute to public healthcare. The architecture consists of Mobile Net as the backbone it can be used for high and low computation scenarios. In order to extract more robust features, we utilize transfer learning to adopt weights from a similar task face detection, which is trained on a very large dataset. We used OpenCV, tensor flow, and NN to detect whether people were wearing face masks or not. The models were tested with images and real-time video streams. The accuracy of the model is achieved and, the optimization of the model is a continuous process and we are building a highly accurate solution by tuning the hyper parameters. This specific model could be used as a use case for edge analytics. Furthermore, the proposed method achieves state-of-the-art results on a public face mask dataset. By the development of face mask-detection we can detect if the person is wearing a face mask and allow their entry would be of great help to the society

The current study used OpenCV, Pytorch and CNN to detect whether people were wearing face masks or not. The models were tested with images and real-time video streams. Even though the accuracy of the model is around 60%, the optimization of the model is a continuous process and we are building a highly accurate solution by tuning the hyperparameters. MobileNetV2 was used to build the mobile version of the same. This specific model could be used as a use case for edge analytics. As the technology are blooming with emerging trends the availability so we have novel face mask detector which can possibly contribute to public healthcare.

The architecture consists of Mobile Net as the backbone it can be used for high and low computation scenarios. In order to extract more robust features, we utilize transfer learning to adopt weights from a similar task face detection, which is trained on a very large dataset. We used OpenCV, tensor flow, keras , Pytorch and CNN to detect whether people were wearing face masks or not. The models were tested with images and real-time video streams. The accuracy of the model is achieved and, the optimization of the model is a continuous process and we are building a highly accurate solution by tuning the hyper parameters. This specific model could be used as a use case for edge analytics. Furthermore, the proposed method achieves state-of-the-art results on a public face mask dataset. By the development of face mask detection. we can detect if the person is wearing a face mask and allow their entry would be of great help to the society.