
TABLE OF CONTENTS

CHAPTER: 1.....07-08

INTRODUCTION

1.1	Motivation of work.....	07
1.2	Abstract	07
1.3	Project Overview.....	08
1.4	Aim.....	08

CHAPTER: 2.....09-10

LITERATURE SURVEY

2.1	An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network	09
2.2	Masked Face Recognition Using Convolutional Neural Network	09
2.3	Existing System.....	10

CHAPTER: 3.....11-33

METHODOLOGY

3.1	Proposed System.....	11
3.2	TENSORFLOW FRAMEWORK	11
3.3	OPENCV.....	12
3.4	NUMPY.....	12
3.5	MATPLOT.....	13
3.6	IPYTHON.....	14
3.7	Python Concepts.....	14
3.8	PANDAS.....	18
3.9	KERAS.....	19
3.10	Machine Learning Approaches.....	20
3.11	HAAR Feature-Based Cascade Classifiers.....	22
3.12	DEEP Learning.....	22

3.13 Neural networks vs Conventional Computers.....	23
3.14 Architecture of Neural Networks.....	24
3.15 Convolution Neural Network.....	25
3.16 Convolutional Layer.....	27
3.17 The NON-Linear Layer.....	28
3.18 The Pooling Layer.....	28
3.19 Fully Connected Layer.....	29
3.20 CNN Model.....	29
3.21 System Architecture.....	31
 CHAPTER: 4.....	34-40
SYSTEM DESIGN	
4.1 UML Diagram.....	34
 CHAPTER: 5.....	41-42
EXPERIMENT ANALYSIS	
5.1 Modules.....	41
5.2 Dataset Link.....	41
5.3 Functional Requirements.....	41
5.4 System Configuration.....	42
5.5 Hardware Requirements.....	42
5.6 Software Requirements.....	42
 CHAPTER: 6.....	43-46
TESTING	
6.1 Introduction.....	43
6.2 Types OF Tests.....	43
6.3 Test Results.....	46

CHAPTER: 7.....47-53

SNAPSHOTS

7.1 Real Time Output.....51

CHAPTER: 8.....54

CONCLUSION

CHAPTER: 9.....55

FUTURE ENHANCEMENT

CHAPTER: 10.....56-57

BIBILOGRAPHY

LIST OF FIGURES

Figure 3.14.1	Layer in Neural Network	24
Figure 3.15.1	CNN Structure	26
Figure 3.15.2	CNN Layers	27
Figure 3.16.1	Convolution with a filter example	28
Figure 3.16.2	Output of Convolution layer	28
Figure 3.18.1	Max Pooling Layer	28
Figure 3.18.2	Overall structure of CNN	29
Figure 3.21.1	System architecture	31
Figure 3.21.2	Layers in CNN Model	32
Figure 4.1.1	Use case Diagram	35
Figure 4.1.2	Sequence Diagram	36
Figure 4.1.3	Activity Diagram	37
Figure 4.1.4	Block Diagram	38
Figure 4.1.5	Class Diagram	38
Figure 4.1.6	Data Flow Diagram	40
Figure 4.1.7	Flowchart Diagram	40

Abstract

The science around the use of masks by the general public to impede COVID-19 transmission is advancing rapidly. Policymakers need guidance on how masks should be used by the general population to combat the COVID-19 pandemic. Here, we synthesize the relevant literature to inform multiple areas: 1) transmission characteristics of COVID-19, 2) filtering characteristics and efficacy of masks, 3) estimated population impacts of widespread community mask use, and 4) sociological considerations for policies concerning mask-wearing. A primary route of transmission of COVID-19 is likely via respiratory droplets, and is known to be transmissible from presymptomatic and asymptomatic individuals. Reducing disease spread requires two things: first, limit contacts of infected individuals via physical distancing and other measures, and second, reduce the transmission probability per contact. The preponderance of evidence indicates that mask wearing reduces the transmissibility per contact by reducing transmission of infected droplets in both laboratory and clinical contexts. Public mask wearing is most effective at reducing spread of the virus when compliance is high. The decreased transmissibility could substantially reduce the death toll and economic impact while the cost of the intervention is low. Given the shortages of medical masks for now we recommend the adoption of public cloth mask wearing, as an effective form of source control, in conjunction with existing hygiene, distancing, and contact tracing strategies. We recommend that public officials and governments strongly encourage the use of widespread face masks in public, including the use of appropriate regulation.

FACE MASK DETECTION USING **OPENCV** **AND IMAGE PROCESSING**

CHAPTER: 1

INTRODUCTION

1.1 MOTIVATION OF WORK:

The world has not yet fully Recover from this pandemic and the vaccine that can effectively treat Covid-19 is yet to be discovered. However, to reduce the impact of the pandemic on the country's economy, several governments have allowed a limited number of economic activities to be resumed once the number of new cases of Covid- 19 has dropped below a certain level. As these countries cautiously restarting their economic activities, concerns have emerged regarding workplace safety in the new post-Covid-19 environment.

To reduce the possibility of infection, it is advised that people should wear masks and maintain a distance of at least 1 meter from each other. Deep learning has gained more attention in object detection and was used for human detection purposes and develop a face mask detection tool that can detect whether the individual is wearing mask or not. This can be done by evaluation of the classification results by analyzing real-time streaming from the Camera. In deep learning projects, we need a training data set. It is the actual dataset used to train the model for performing various actions.

1.2 ABSTRACT:

The Global pandemic COVID-19 circumstances emerged in an epidemic of dangerous disease in all over the world. 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 HAAR-CASACADE algorithm is used for image detection. 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. HAAR feature-based cascade classifier system utilizes only 200 features out of 6000 features to yield a recognition rate of 85-95%.

According to this motivation we demand mask detection as a unique and public health service system during the global pandemic COVID-19 epidemic. The model is trained by face mask image and non-face mask image.

Keywords: COVID-19 epidemic, HAAR-CASACADE algorithm, mask detection, face mask image, non-face mask image

1.3 PROJECT OVERVIEW:

In our proposed system we implemented the supervised learning system approach to detect the face mask. We have applied machine learning approaches like CNN, detecting face and image processing.

1.4 AIM:

The main objective of the face detection model is to detect the face of individuals and conclude whether they are wearing masks or not at that particular moment when they are captured in the image.

CHAPTER: 2

LITERATURE SURVEY

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

COVID-19 pandemic caused by novel coronavirus 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 restrict 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.

2.2 Masked Face Recognition Using Convolutional Neural Network [2]:

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.

2.3 Existing System:

COVID-19 has affected the world badly. Studies have proved that wearing a face mask is one of the precautions to reduce the risk of viral transmission. And many public places as well as public service providers require customers to use the service and place only if they wear mask correctly. So, it is not possible to manually track the customer, whether they have the mask or not. That's why this technology holds the key here.

Disadvantages

- No system to detect face mask in public area
- Sensors were used
- More costly

CHAPTER: 3

METHODOLOGY

3.1 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.

3.2 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.

3.3 OPENCV

It is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including feature like face detection and object detection.

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.

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. It combines the best qualities of Open CV C++ API and Python language.

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. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of -the-art computer vision and machine learning algorithms.

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.

3.4 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 high- level 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 open- source 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.

3.5 MATPLOTTING:

Mat plot 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.

3.6 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.

3.7 PYTHON CONCEPTS:

You can skip to the next chapter if you are not interested in how and why Python. In this chapter, I will try to explain why I think Python is one of the best programming languages available and why it is such a great place to start.

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.

3.7.1 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.

ADVANTAGES/BENEFITS OF PYTHON:

The diverse application of the Python language is a result of the combination of features which give this language an edge over others. Some of the benefits of programming in Python include:

1.Presence of Third-Party Modules:

The Python Package Index (PyPI) contains numerous third-party modules that make Python capable of interacting with most of the other languages and platforms.

2.Extensive Support Libraries:

Python provides a large standard library which includes areas like internet protocols, string operations, web services tools and operating system interfaces. Many high use programming tasks have already been scripted into the standard library which reduces length of code to be written significantly.

3.Open Source and Community Development:

Python language is developed under an OSI-approved opensource license, which makes it free to use and distribute, including for commercial purposes. Further, its development is driven by the community which collaborates for its code through hosting conferences and mailing lists, and provides for its numerous modules.

4.Learning Ease and Support Available:

Python offers excellent readability and uncluttered simple-to-learn syntax which helps beginners to utilize this programming language. The code style guidelines, PEP 8, provide a set of rules to facilitate the formatting of code. Additionally, the wide base of users and active developers has resulted in a rich internet resource bank to encourage development and the continued adoption of the language.

5.User-friendly Data Structures:

Python has built-in list and dictionary data structures which can be used to construct fast runtime data structures. Further, Python also provides the option of dynamic high- level data typing which reduces the length of support code that is needed.

6.Productivity and Speed:

Python has Clean object-oriented design, provides enhanced process control capabilities, and possesses strong integration and text processing capabilities and its own unit testing

framework, all of which contribute to the increase in its speed and productivity. Python is considered a viable option for building complex multi-protocol network applications.

As can be seen from the above-mentioned points, Python offers a number of advantages for software development. As upgrading of the language continues, its loyalist base could grow as well.

Python has five standard data types –

- Numbers
- String
- List
- Tuple
- Dictionary

3.7.2 Python Numbers

Numeric values are stored in number data types. When you give a number a value, it becomes a number object.

3.7.3 Python Strings

In this python uses a string is defined as a collection set of characters enclosed in quotation marks. Python allows you to use any number of quotes in pairs. The slice operator ([] and [:]) can be used to extract subsets of strings, with indexes starting at 0 at the start of the string and working their way to -1 at the end.

3.7.4 Python Lists

The most diverse types of Python data are lists. Items are separated by commas and placed in square brackets in the list ([]). Lists are similar to C-order in some ways. Listings can be for many types of data, which is one difference between them.

The slide operator ([] and [:]) can be used to retrieve values stored in the list, the indicators start with 0 at the beginning of the list and work their way to the end of the list. The concatenation operator of the list is a plus sign (+), while the repeater is an asterisk (*).

3.7.5 Python Tuples

A tuple is a type of data type similar to a sequence of items. A tuple is a set of values separated by commas. The tuples, unlike the list, are surrounded by parentheses.

Lists are placed in parentheses ([]), and the elements and sizes can be changed, but the tuples are wrapped in brackets (()) and cannot be sorted. Tuples are the same as reading lists only.

3.7.6 Python Dictionary

Python dictionaries in Python are a way of a hash table. They are similar to Perl's combination schemes or hashes, and are made up of two key numbers.

The dictionary key can be any type of Python, but numbers and strings are very common. Values, on the other hand, can be anything you choose Python.

Curly braces {} surround dictionaries, and square brackets [] are used to assign and access values.

Different modes in python

Python Normal and interactive are the two basic Python modes.

The scripted and completed.py files are executed in the Python interpreter in the regular manner.

Interactive mode is a command line shell that provides instant response for each statement while simultaneously running previously provided statements in active memory.

The feed programme is assessed in part and whole as fresh lines are fed into the interpreter.

3.8 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, real- world 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.

3.9 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 very quickly define deep learning models and run them on top of a Theano or TensorFlow backend.

3.10 Machine Learning Approaches:

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

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is 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.

3.10.1 Viola-Jones Object detection framework based in HAAR features:

The Viola-Jones algorithm is one of the most popular algorithms for objects recognition in an image. This research paper deals with the possibilities of parametric optimization of the Viola-Jones algorithm to achieve maximum efficiency of the algorithm in specific environmental conditions. It is shown that with the use of additional modifications it is possible to increase the speed of the algorithm in a particular image by 2-5 times with the loss of accuracy and completeness of the work by not more than the 3-5%.

3.10.2 Scale-invariant feature transform (SIFT):

The scale-invariant feature transform (SIFT) is a feature detection algorithm in computer vision to detect and describe local features in images. SIFT key points of objects are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of key points that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches. The determination of consistent clusters is performed rapidly by using an efficient hash table implementation of the generalized Hough transform. Each cluster of 3 or more features that agree on an object and its pose is then subject to further detailed model verification and subsequently outliers are discarded. Finally the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable false matches. Object matches that pass all these tests can be identified as correct with high confidence.

3.10.3 Histogram of oriented gradients (HOG):

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object-detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a

An algorithm is involved in this proposed system

3.11 HAAR Feature-Based Cascade Classifiers:

It is an Object Detection Algorithm used to identify faces in an image or a real time video.

Dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

It is an effective way for object detection.

In this approach, lot of positive and negative images are used to train the classifier.

In this, a model is pre-trained with frontal features is developed and used in this experiment to detect the faces in real-time.

HAAR Cascade is a machine learning-based approach where a lot of positive and negative images are use to train the classifier.

Positive Images: These images contain the images which we want our classifier to identify.

Negative Images: Images of everything else, which do not contain the object we want to detect.

WHY HAAR FEATURE BASED CASCADE CLASSIFIERS IS PREFFERD?

The key advantage of a HAAR-like feature over most other features is its calculation speed.

A HAAR-like feature of any size can be calculated in constant time (approximately 60 microprocessor instructions).

3.12 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.

3.13 NEURAL NETWORKS VERSUS CONVENTIONAL COMPUTERS:

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem-solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don't exactly know how to do. Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem.

Neural networks learn by example. They cannot be programmed to is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable. On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high- level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

3.14 ARCHITECTURE OF NEURAL NETWORKS:

3.14.1 FEED-FORWARD NETWORKS:

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down. to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

3.14.2 FEEDBACK NETWORKS:

Feedback networks can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organization.

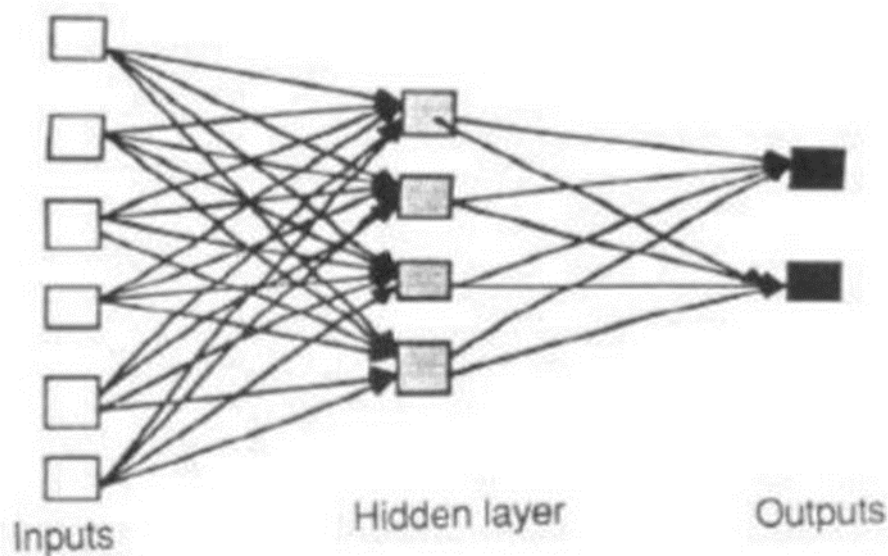


Fig 3.14.1 Layer in Neural Network

3.14.3 NETWORK LAYERS:

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of input units is connected to a layer of hidden units, which is connected to a layer of output units.

The activity of the input units represents the raw information that is fed into the network.

The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.

The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

Also distinguish single-layer and multi-layer architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organizations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

3.15 Convolution Neural Network

A convolution neural network is a special architecture of artificial neural network proposed by Yann LeCun in 1988. One of the most popular uses of the architecture is image classification. CNNs have wide applications in image and video recognition, recommender systems and natural language processing. In this article, the example that this project will take is related to Computer Vision. However, the basic concept remains the same and can be applied to any other use-case!

CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on CNNs. In more detail the image

is passed through a series of convolution, nonlinear, pooling layers and fully connected layers, then generates the output.

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the visual cortex. CNNs use relatively little pre-processing compared to other image classification algorithms. CNN is a special kind of multi-layer NNs applied to 2-d arrays (usually images), based on spatially localized neural input. CNN Generate 'patterns of patterns' for pattern recognition.

Each layer combines patches from previous layers. Convolutional Networks are trainable multistage architectures composed of multiple stages Input and output of each stage are sets of arrays called feature maps. At output, each feature map represents a particular feature extracted at all locations on input. Each stage is composed of: a filter bank layer, a non-linearity layer, and a feature pooling layer. A ConvNet is composed of 1, 2 or 3 such 3-layer stages, followed by a classification module.

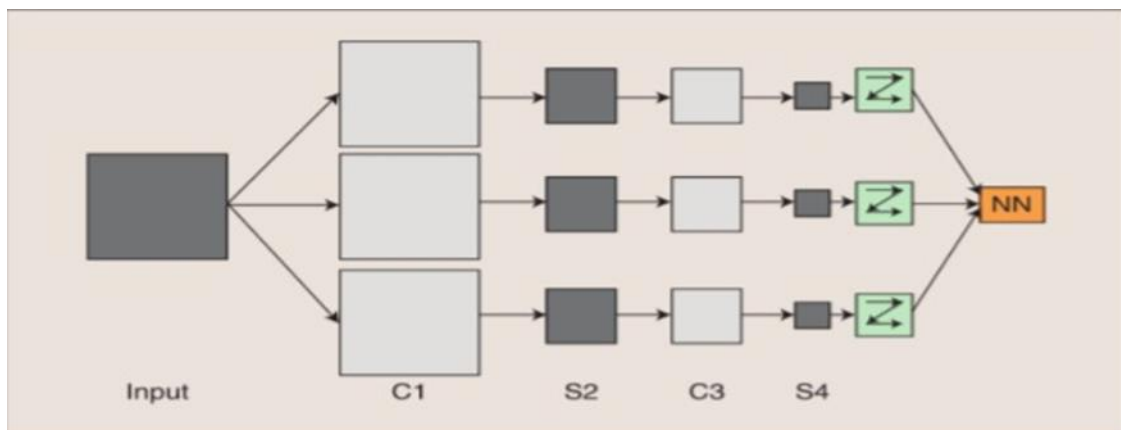


Fig 3.15.1 CNN Structure

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

Filter: A trainable filter (kernel) in filter bank connects input feature map to output feature map. Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli.

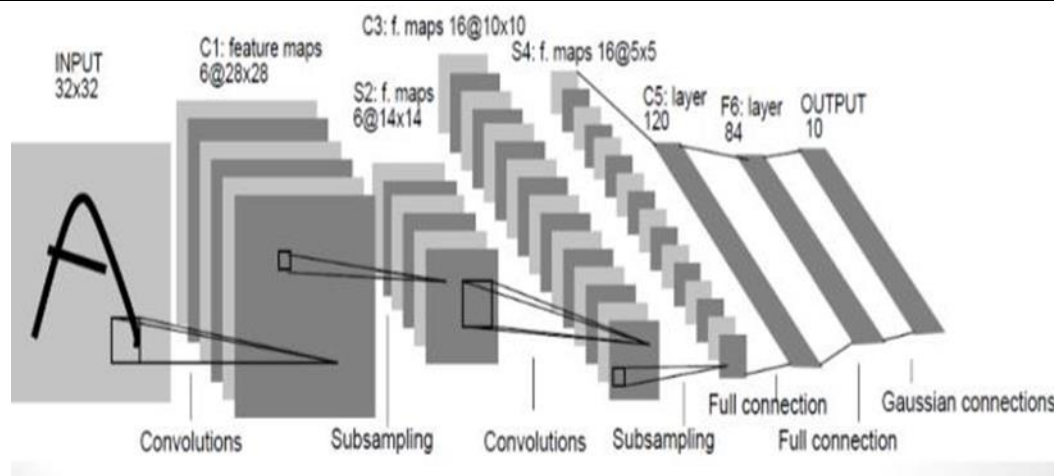


Fig 3.15.2 CNN Layers

3.16 CONVOLUTIONAL LAYER

It is always first. The image (matrix with pixel values) is entered into it. Image that the reacting of the input matrix begins at the top left of image. Next the software selects the smaller matrix there, which is called a filter. Then the filter produces convolution that is moves along the input image. The filter task is to multiple its value by the original pixel values. All these multiplications are summed up and one number is obtained at the end. Since the filter has read the image only in the upper left corner it moves further by one unit right performing a similar operation. After passing the filter across all positions, a matrix is obtained, but smaller than a input matrix.

This operation, from a human perspective is analogous to identifying boundaries and simple Colors on the image. But in order to recognize the fish whole network is needed. The network will be -consists of several convolution layers mixed with nonlinear and pooling layers. Convolution is the first layer to extract features from an input image. Convolution features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

- An image matrix of dimension $(h \times w \times d)$
- A filter $(f_h \times f_w \times d)$
- Outputs a volume dimension $(h-f_h+1) \times (w-f_w+1) \times 1$.

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below



Fig 3.16.1 Convolution with a filter example

Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called “Feature Map” as output shown in below

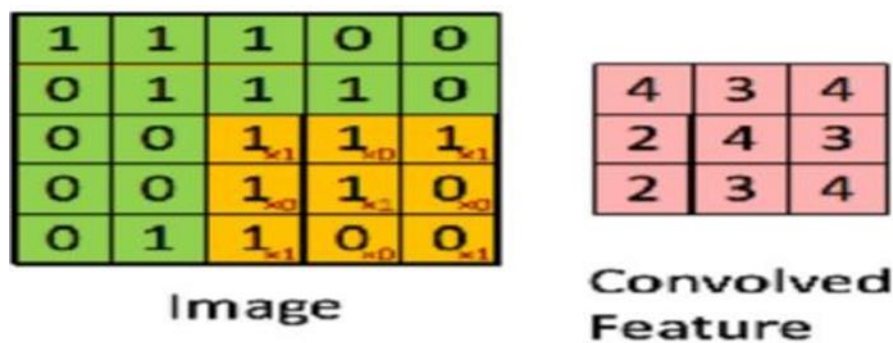


Fig 3.16.2 Output of Convolution layer

3.17 THE NON-LINEAR LAYER:

It is added after each convolution operation. It has the activation function, which brings nonlinear property, without this property a network would not be, sufficiently intense and will not be able to model the response variable.

3.18 THE POOLING LAYER:

It follows the nonlinear layer. It works with width and height of the image and performs a down sampling operation on them. As a result image volume is reduced. This means that if some

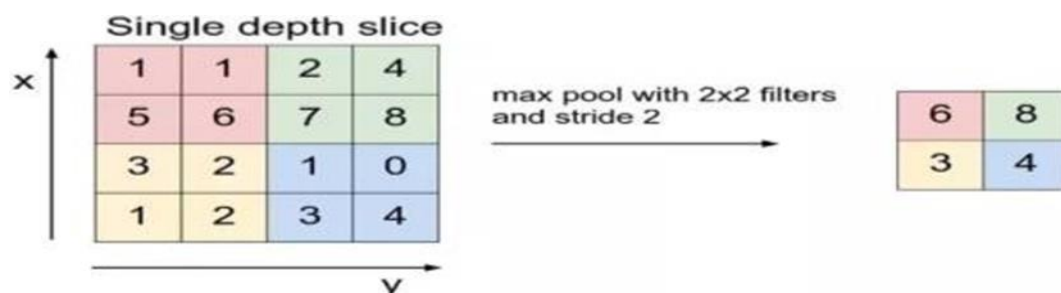


Fig 3.18.1 Max Pooling Layer

features already been identified in the previous convolution operation, then a detailed image is no longer needed for further processing and is compressed to less detailed pictures.

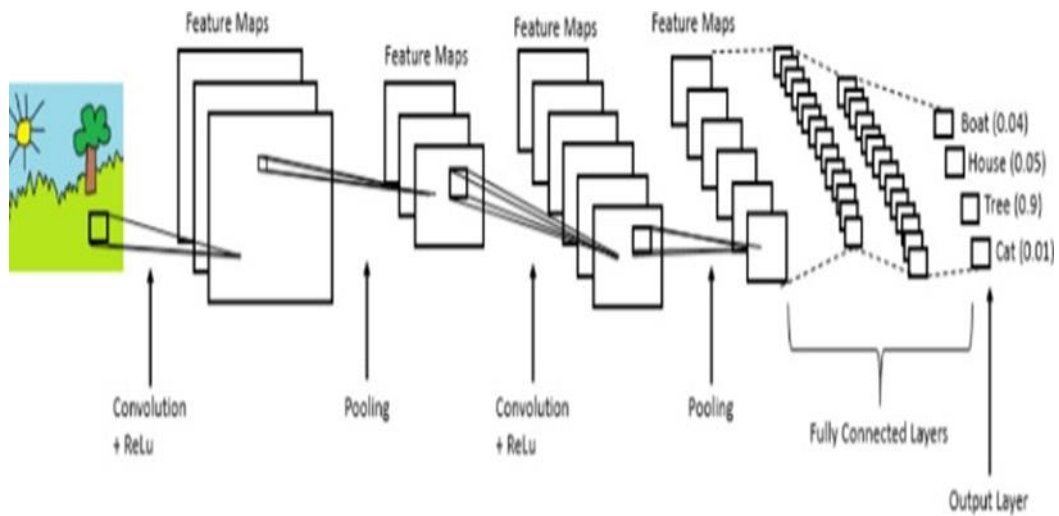


Fig 3.18.2 Overall structure of CNN

3.19 FULLY CONNECTED LAYER:

After completion of series of convolution, non-linear and pooling layer, it's necessary to attach a fully connected layer. This layer takes the output information from convolution network. Attaching a fully connected layer to the end of the network results in N dimensional vector, where N is the amount of classes from which the model selects the desired class.

3.20 CNN MODEL

1. This CNN model is built using the TensorFlow framework and the OpenCV library which is highly used for real-time applications.
2. This model can also be used to develop a full-fledged software to scan every person before they can enter the public gathering.

3.20.1 LAYERS IN CNN MODEL

1. Conv2D Layer:

It has 100 filters and the activation function used is the 'ReLU'. The ReLU function stands for Rectified Linear Unit which will output the input directly if it is positive, otherwise it will output zero.

2. MaxPooling2D:

It is used with pool size or filter size of 2*2.

3. Flatten () Layer:

It is used to flatten all the layers into a single 1D layer.

4. Dropout Layer:

It is used to prevent the model from overfitting.

5. Dense Layer:

The activation function here is soft max which will output a vector with two probability distribution values.

3.21 SYSTEM ARCHITECTURE



Fig 3.21.1 System Architecture

Data Visualization

In the first step, let us visualize the total number of images in our dataset in both categories. We can see that there are 690 images in the 'yes' class and 686 images in the 'no' class.

Data Augmentation

In the next step, we augment our dataset to include a greater number of images for our training. In this step of data augmentation, we rotate and flip each of the images in our dataset.

Splitting the data

In this step, we split our data into the training set which will contain the images on which the CNN model will be trained and the test set with the images on which our model will be tested.

Building the Model

In the next step, we build our Sequential CNN model with various layers such as Conv2D, MaxPooling2D, Flatten, Dropout and Dense.

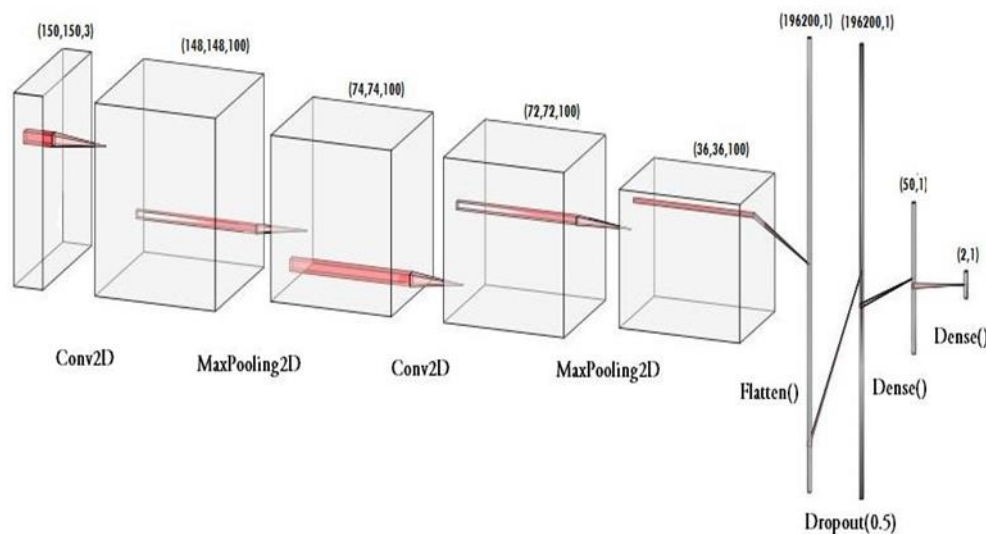


Fig 3.21.1 Layers in CNN Model

Pre-Training the CNN model

After building our model, let us create the 'train_generator' and 'validation_generator' to fit them to our model in the next step.

Training the CNN model

This step is the main step where we fit our images in the training set and the test set to our Sequential model we built using keras library. I have trained the model for 30 epochs (iterations). However, we can train for a greater number of epochs to attain higher accuracy lest there occurs over-fitting.

Labeling the Information

After building the model, we label two probabilities for our results. ['0' as 'without_mask' and '1' as 'with_mask']. I am also setting the boundary rectangle color using the RGB values.

Importing the Face detection Program

After this, we intend to use it to detect if we are wearing a face mask using our PC's webcam. For this, first, we need to implement face detection. In this, I am using the Haar Feature-based Cascade Classifiers for detecting the features of the face.

Detecting the Faces with and without Masks

In the last step, we use the OpenCV library to run an infinite loop to use our web camera in which we detect the face using the Cascade Classifier.

CHAPTER: 4

SYSTEM DESIGN

4.1 UML DIAGRAM:

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.

4.1.1 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

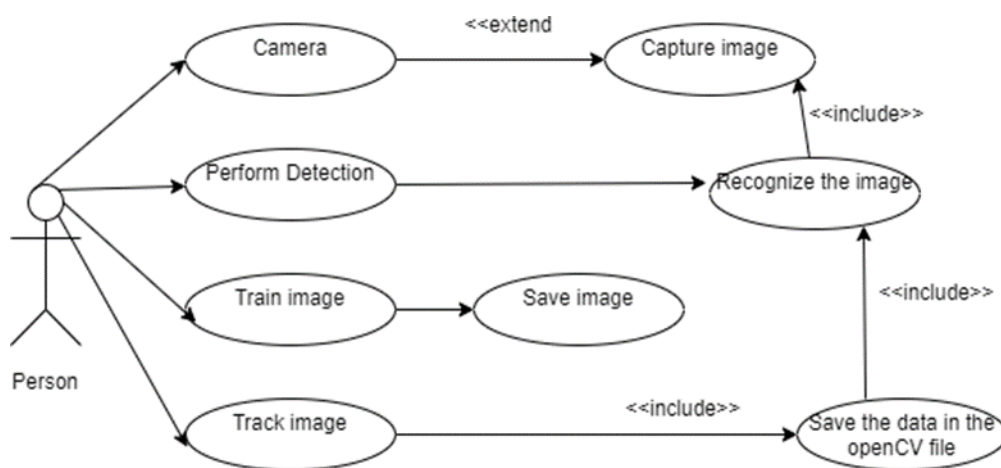


Fig 4.1.1 Use case Diagram

4.1.2 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.

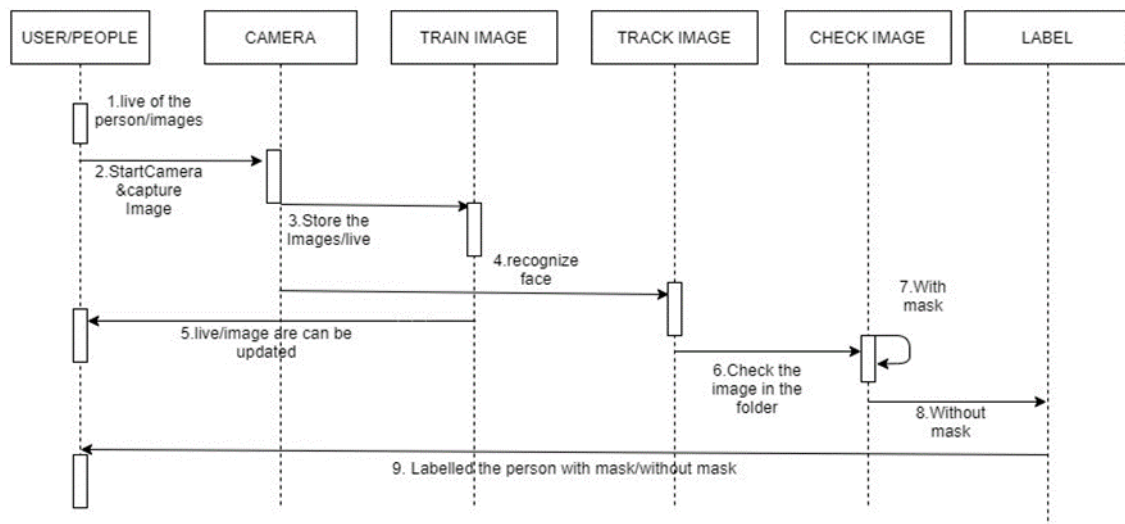


Fig 4.1.2 Sequence Diagram

4.1.3 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.

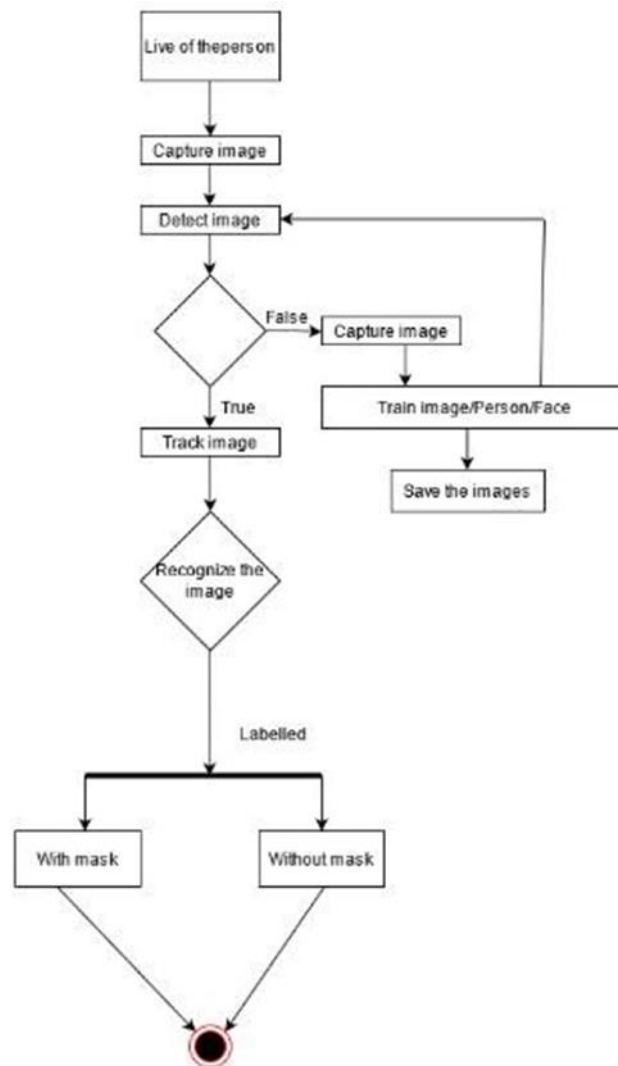


Fig 4.1.3 Activity Diagram

4.1.4 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.

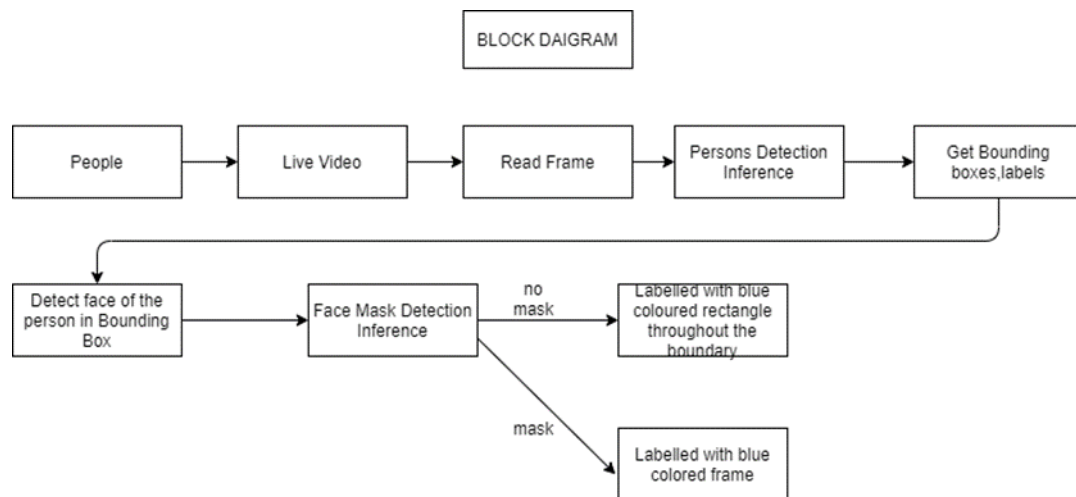


Fig 4.1.4 Block Diagram

4.1.5 CLASS DIAGRAM

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

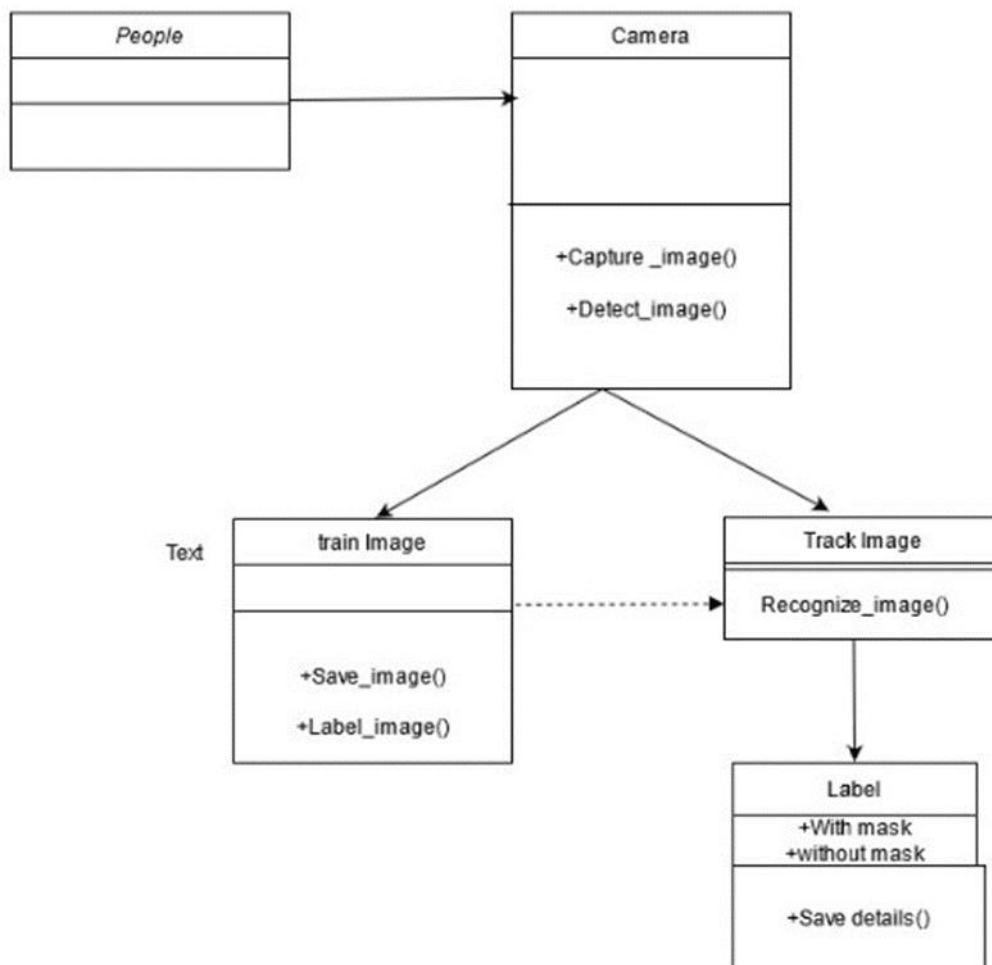


Fig 4.1.5 Class Diagram

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

4.1.6 DATA FLOW DIAGRAM

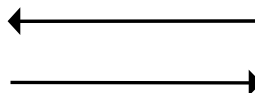
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

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

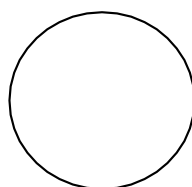
A graphical tool for defining and analyzing minute data with an active or automated system, including process, data stores, and system delays. Data Flow Data is a key and basic tool for the architecture of all other objects. Bubble-bubble or data flow graph is another name for DFD.

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. The following is the Basic Notation for building DFDs:

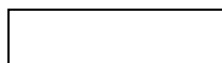
1. **Dataflow:** Data flows in a certain direction from the source to the destination.



2. **Process:** People, processes, or technologies that use or produce (Transforming) Information No information about a body part.



3. **Source:** People, programs, organizations, and other things can be external data sources or locations.



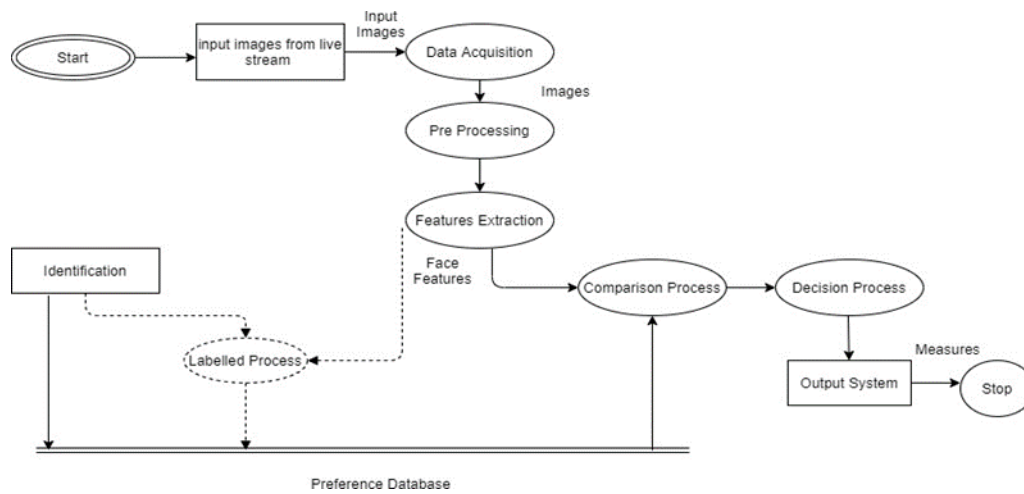


Fig 4.1.6 Data Flow Diagram

4.1.7 FLOWCHART DIAGRAM

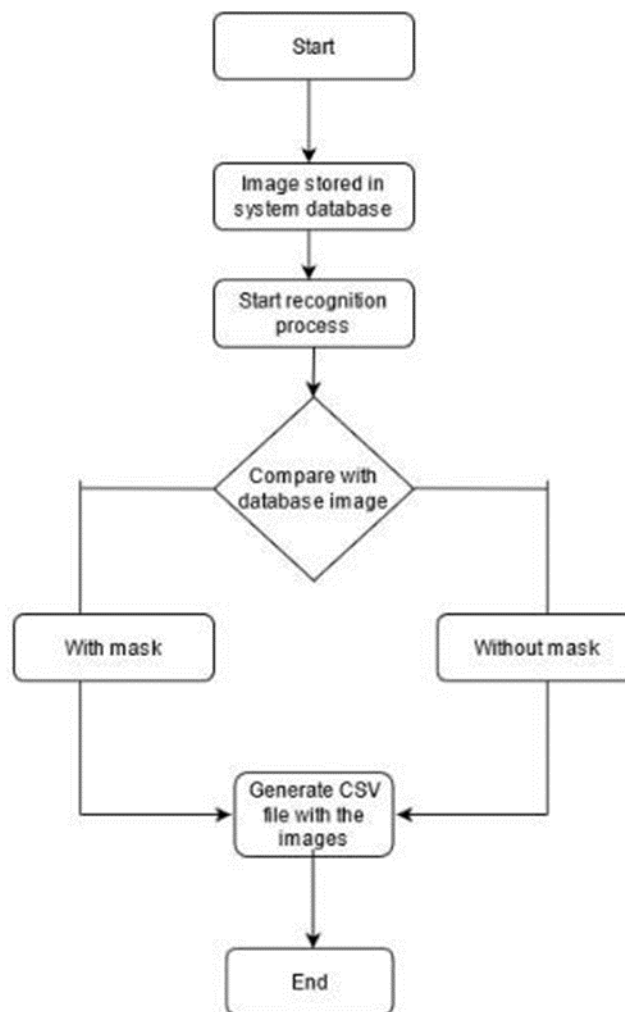


Fig 4.1.7 Flowchart Diagram

CHAPTER: 5

EXPERIMENT ANALYSIS

5.1 MODULES

1. Creating image datasets a data-loaders for train and test using the experiments folder split
 - Training Dataset: A dataset that we feed into our algorithm to train our model.
 - Testing Dataset: A dataset that we use to validate the accuracy of our model but is not used to train the model. It may be called the validation dataset.
2. Training the model
3. Visualizing images

Identify the Face in the Webcam:

To identify the faces a pre-trained model provided by the OpenCV framework was used.

The model was trained using web images. OpenCV provides 2 models for this face detector.

Face Mask Detection in webcam stream:

The flow to identify the person in the webcam wearing the face mask or not. The process is two-fold.

1. To identify the faces in the webcam
2. Classify the faces based on the mask.

5.2 DATASET LINK

<https://github.com/prajnasb/observations/tree/master/experiements/data>

5.3 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.

5.4 SYSTEM CONFIGURATION

This project can run on commodity hardware. We ran entire project on an Intel I5 process with 8GB 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.

5.5 HARDWARE REQUIREMENTS

- RAM: 4 GB
- Storage: 500 GB
- CPU: 2 GHz or faster
- Architecture: 32-bit or 64-bit

5.6 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: 6

TESTING

6.1 INTRODUCTION:

Testing defines the status of the working functionalities of any particular system. Through testing particular software, one cannot identify the defects in it but can analyse the performance of software and its working behavior. By testing the software, we can find the limitations that become the conditions on which the performance is measured on that particular level. In order to start the testing process the primary thing is requirements of software development cycle. Using this phase, the testing phase will be easier for testers. The capacity of the software can be calculated by executing the code and inspecting the code in different conditions such as testing the software by subjecting it to different sources as input and examining the results with respect to the inputs.

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.[28]

6.2 TYPES OF TESTS:

6.2.1 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.

6.2.2 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.

6.2.3 Functional test:

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.

6.2.4 System Test:

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.

6.2.5 White Box Testing:

White Box Testing is a testing 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.

6.2.6 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.

6.2.7 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.

6.2.8 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.

6.2.9 Acceptance Testing:

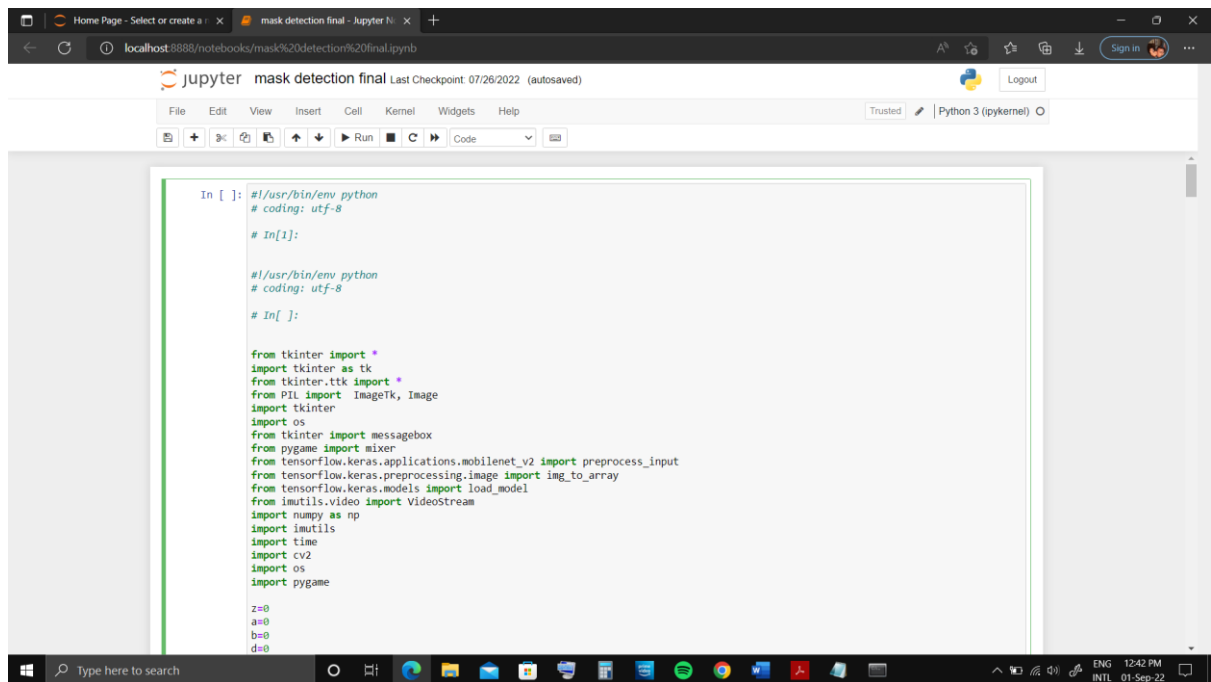
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

6.3 Test Results: All test cases mentioned above passed successfully. No defects encountered

	Test case name	Test case description	Test steps				Test status P/F
			Step	I/p given	Expected o/p	Actual o/p	
TC01	Admin login	To check that the admin has entered all the required details	Login	Required details for login	Login is Successful	Login is Successful	Pass
	Admin login	To check that the admin has entered all the required details	Login	Any Required details for login Is missing	Login is Successful	Login is Un Successful	Fail
TC02	Screenshot	To capture screenshot automatically	Capture screenshot	Enable	Screenshot saved in database	Screenshot successfully saved	Pass
	Screenshot	To capture screenshot automatically	Capture screenshot	Enable	Screenshot saved in database	Error in saving screenshot	Fail
TC03	Mask Detect	To detect mask	Detect mask from live camera feed	Live camera feed	Mask detected with accuracy	Detection Successful	Pass
	Mask Detect	To detect mask	Detect mask from live camera feed	Live camera feed	Mask detected with accuracy	Detection Un-Successful	fail

CHAPTER:7

SNAPSHOTS



```

In [ ]: #!/usr/bin/env python
# coding: utf-8

# In[1]:

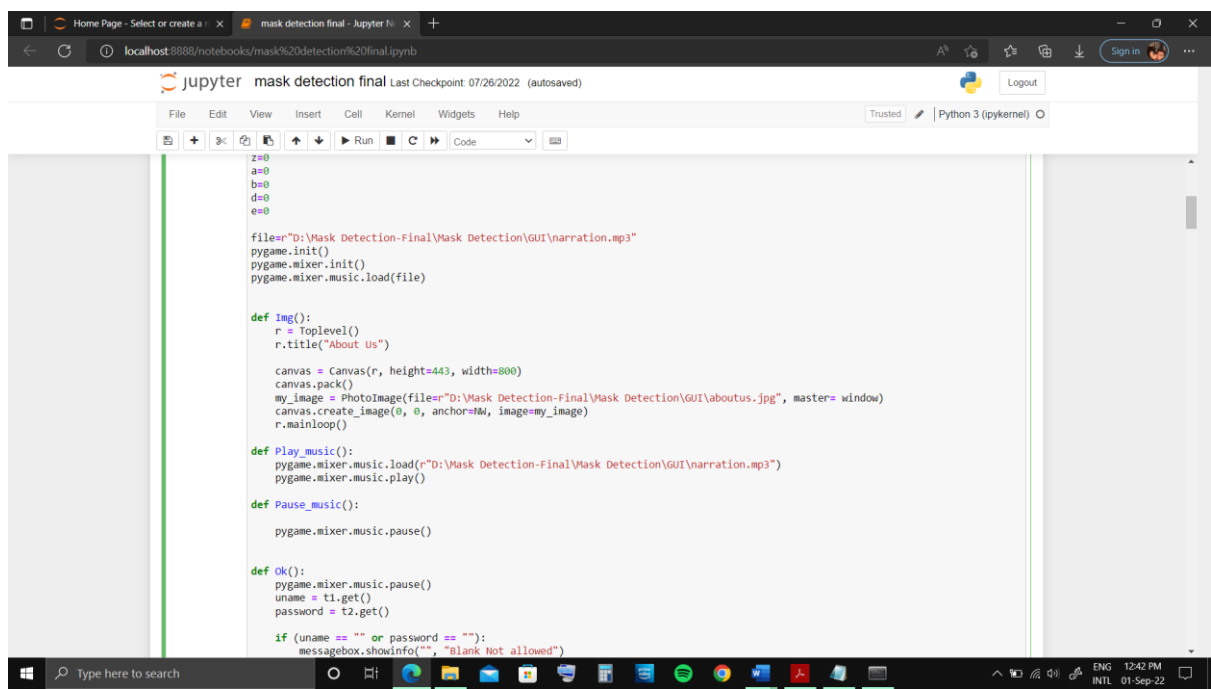
#!/usr/bin/env python
# coding: utf-8

# In[ ]:

from tkinter import *
import tkinter as tk
from tkinter.ttk import *
from PIL import ImageTk, Image
import tkinter
import os
from tkinter import messagebox
from pygame import mixer
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
import pygame

z=0
a=0
b=0
d=0

```



```

z=0
a=0
b=0
d=0
e=0

file=r"D:\Mask Detection-Final\Mask Detection\GUI\narration.mp3"
pygame.init()
pygame.mixer.init()
pygame.mixer.music.load(file)

def img():
    r = Toplevel()
    r.title("About Us")

    canvas = Canvas(r, height=443, width=800)
    canvas.pack()
    my_image = PhotoImage(file=r"D:\Mask Detection-Final\Mask Detection\GUI\aboutus.jpg", master= window)
    canvas.create_image(0, 0, anchor=NW, image=my_image)
    r.mainloop()

def Play_music():
    pygame.mixer.music.load(r"D:\Mask Detection-Final\Mask Detection\GUI\narration.mp3")
    pygame.mixer.music.play()

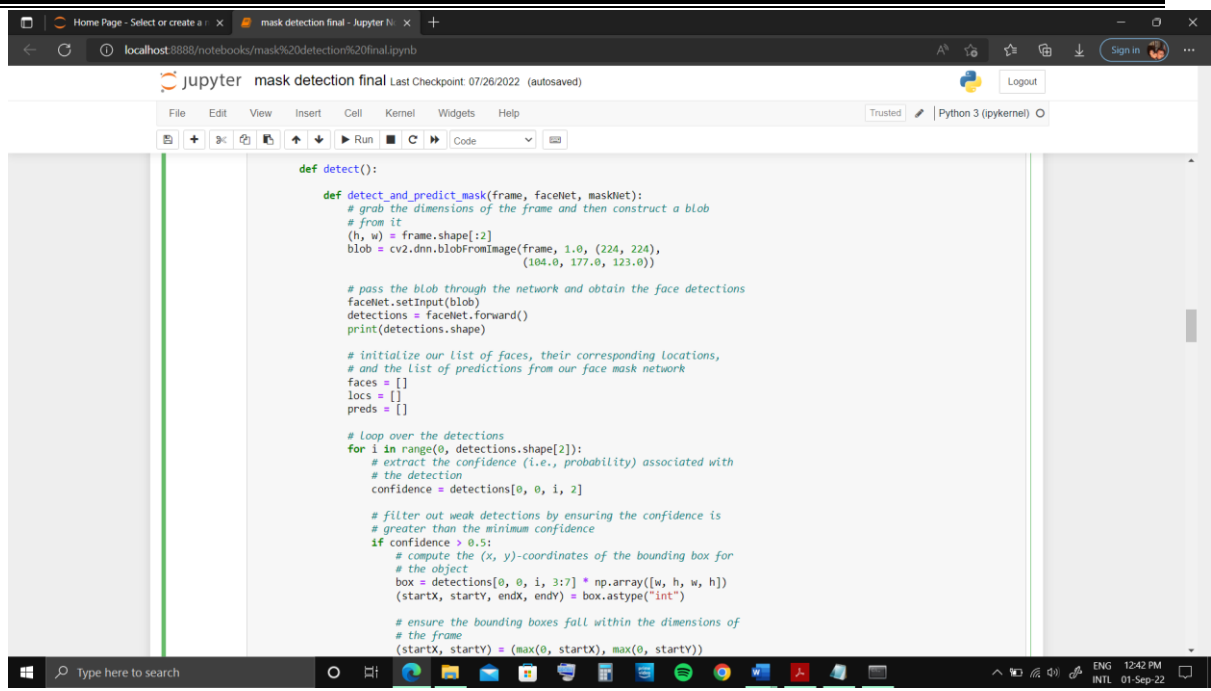
def Pause_music():
    pygame.mixer.music.pause()

def Ok():
    pygame.mixer.music.pause()
    uname = t1.get()
    password = t2.get()

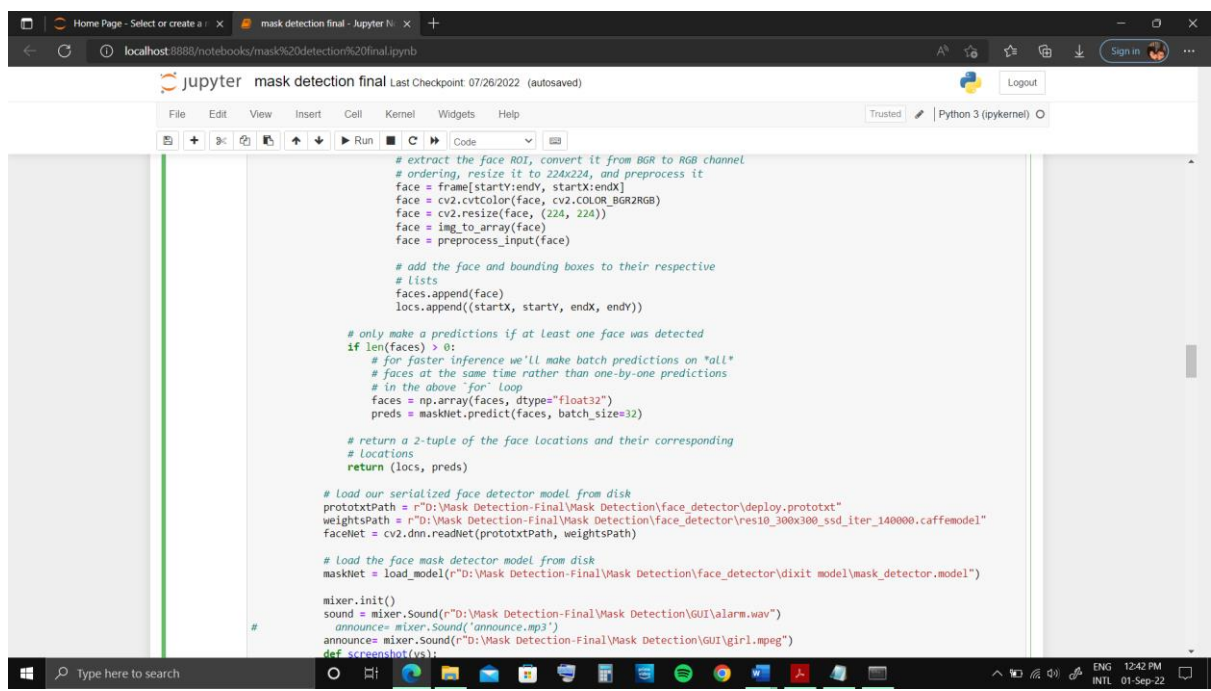
    if (uname == "" or password == ""):
        messagebox.showinfo("", "Blank Not allowed")

```

Face Mask Detection



```
def detect():  
    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))
```



```
                # 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"D:\Mask Detection-Final\Mask Detection\face_detector\deploy.prototxt"  
weightsPath = r"D:\Mask Detection-Final\Mask Detection\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(r"D:\Mask Detection-Final\Mask Detection\face_detector\dixit model\mask_detector.model")  
  
mixer.init()  
sound = mixer.Sound(r"D:\Mask Detection-Final\Mask Detection\GUI\alarm.wav")  
announce = mixer.Sound(r"D:\Mask Detection-Final\Mask Detection\GUI\girl.mpeg")  
def screenshot(y):
```


Face Mask Detection

```
mask detection final - Jupyter Notebook
localhost:8888/notebooks/mask%20detection%20final.ipynb

jupyter mask detection final Last Checkpoint: 07/26/2022 (autosaved)

File Edit View Insert Cell Kernel Widgets Help
Python 3 (ipykernel)

canvas1.place(x=100,y=200)

l1=tk.Label(canvas1, text="Username", font=("calibiri",12),bg="orange")
l1.place(x=50,y=35)
t1=tk.Entry(canvas1, width=25,bd=2)
t1.place(x=150,y=35)

l2=tk.Label(canvas1, text="Password", font=("calibiri",12),bg="orange")
l2.place(x=50,y=65)
t2=tk.Entry(canvas1, width=25,bd=2)
t2.place(x=150,y=65)
t2.config(show="*")

b1=tk.Button(canvas1,text="Login", font=("calibiri",12,"bold"),bg="red",fg='white', command=ok )
b1.place(x=180,y=105)

b8=tk.Button(window,text="About Us", font=("calibiri",12),bg="green",command=Img)
b8.place(x=480,y=365)

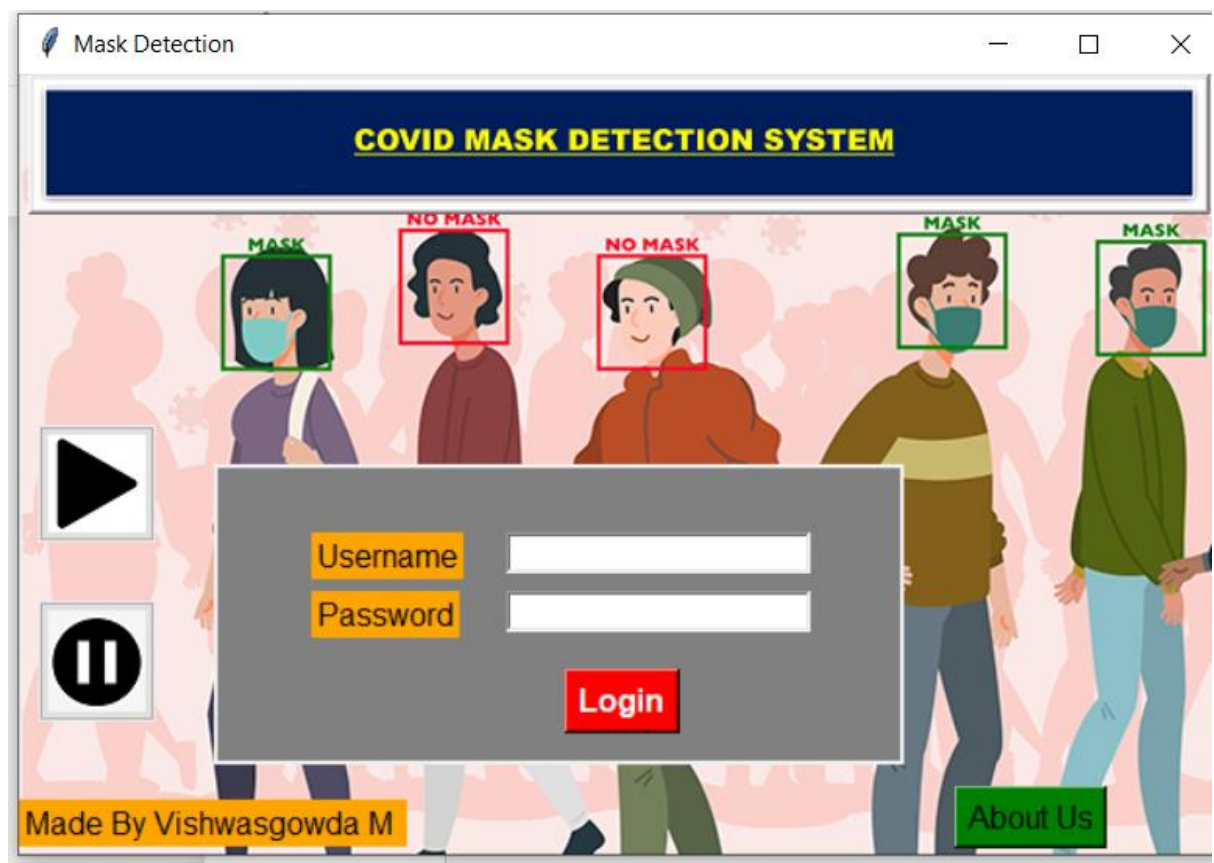
lel=tk.Label(window, text="Made By Vishwasgowda M ", font=("calibiri",12),bg="orange")
lel.place(x=0,y=372)

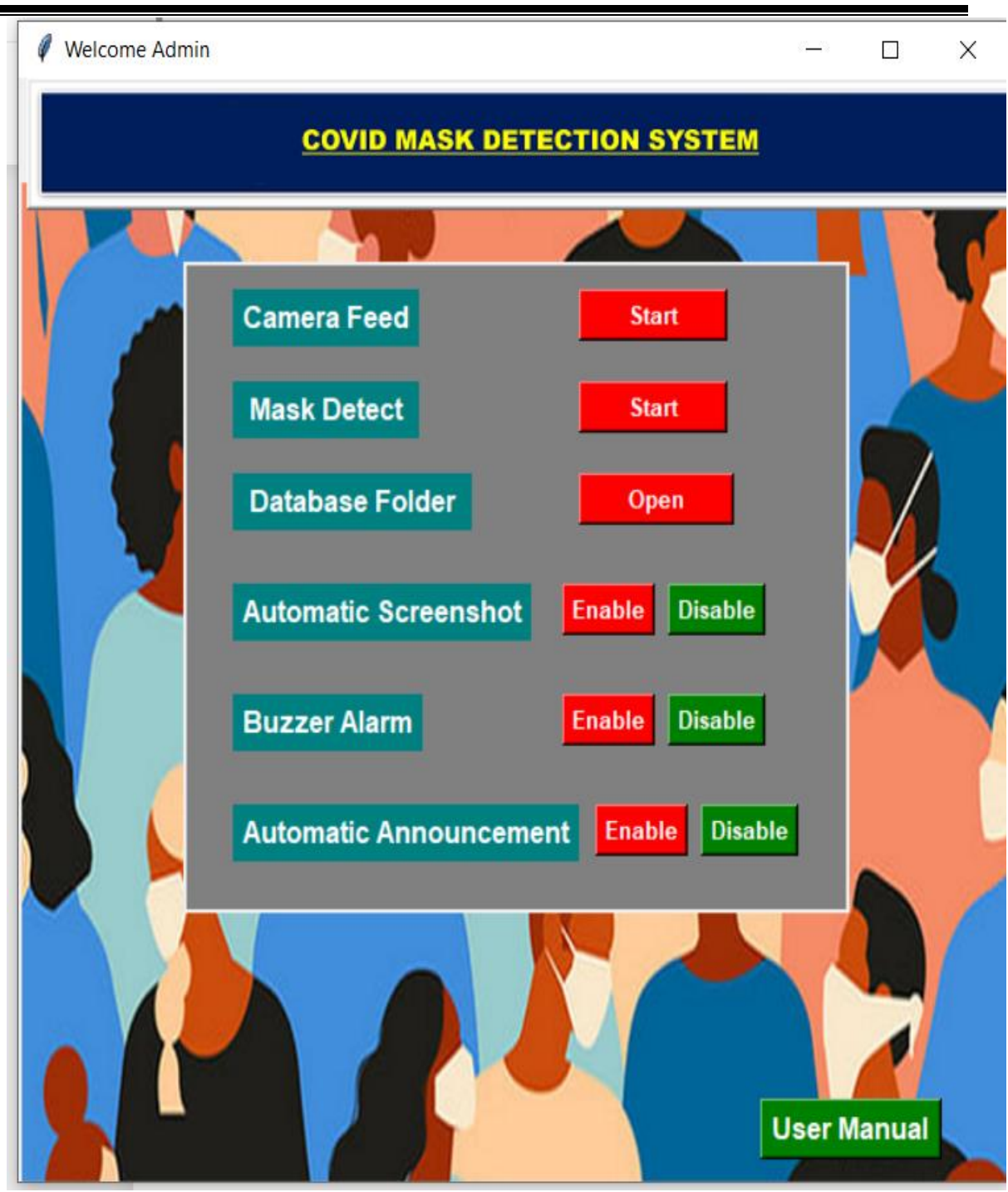
btn11=Button(window,text='Play', width=14,image=play, command=Play_music)
btn11.place(x=10,y=180)

btn12=Button(window, width=14,text='Pause', image=pause, command=Pause_music)
btn12.place(x=10,y=270)

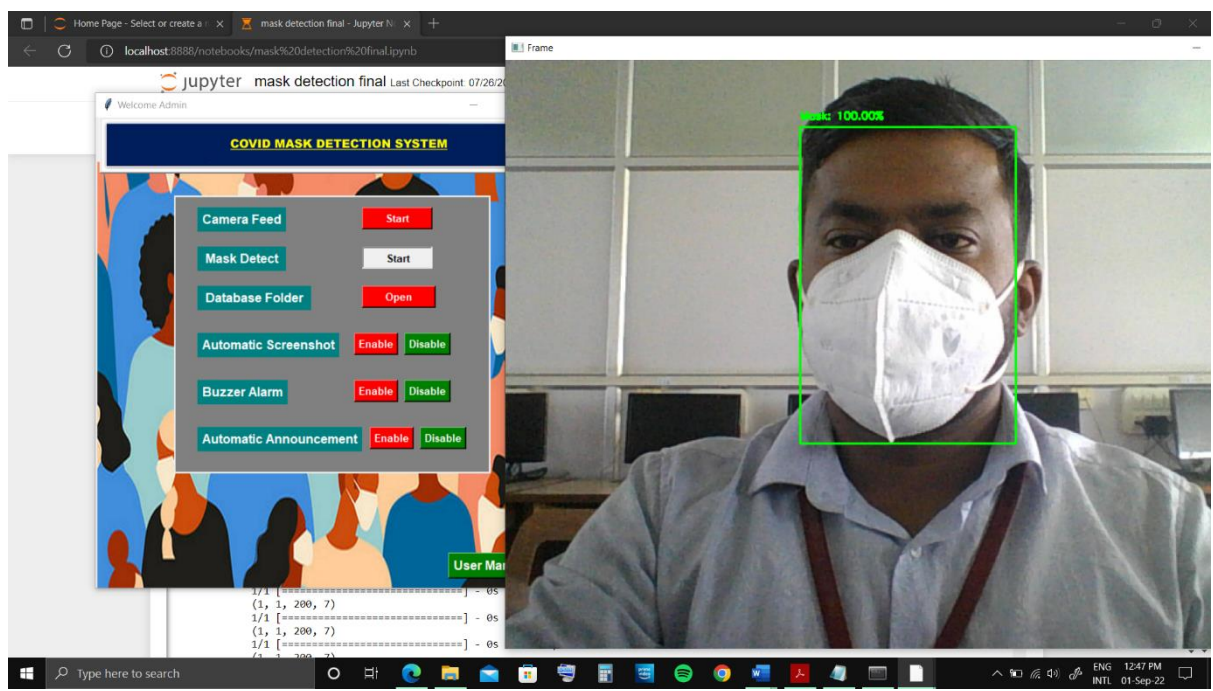
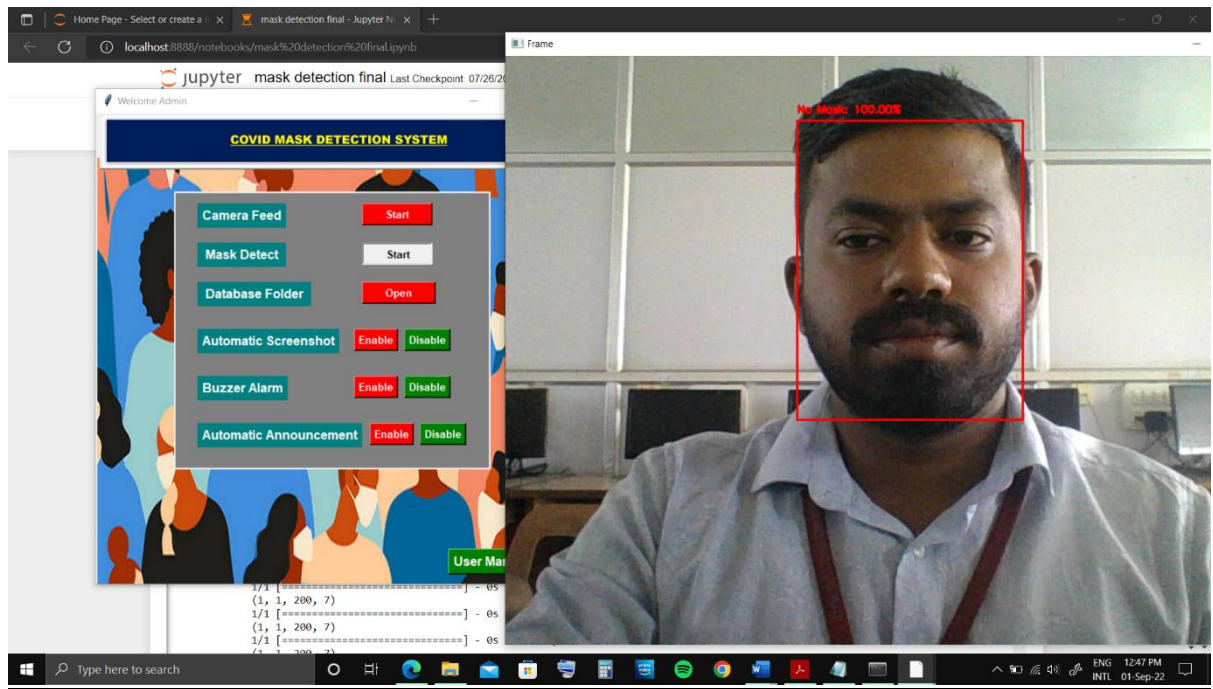
window.geometry("620x400")
window.mainloop()

# In[ ]:
```

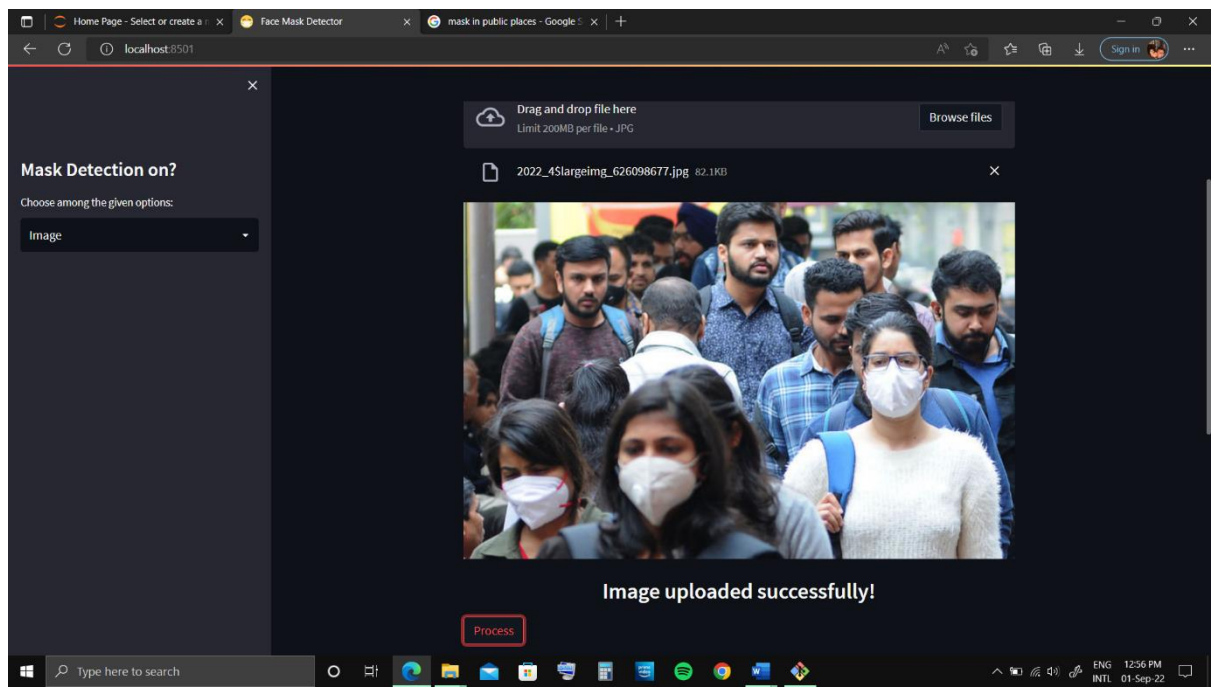
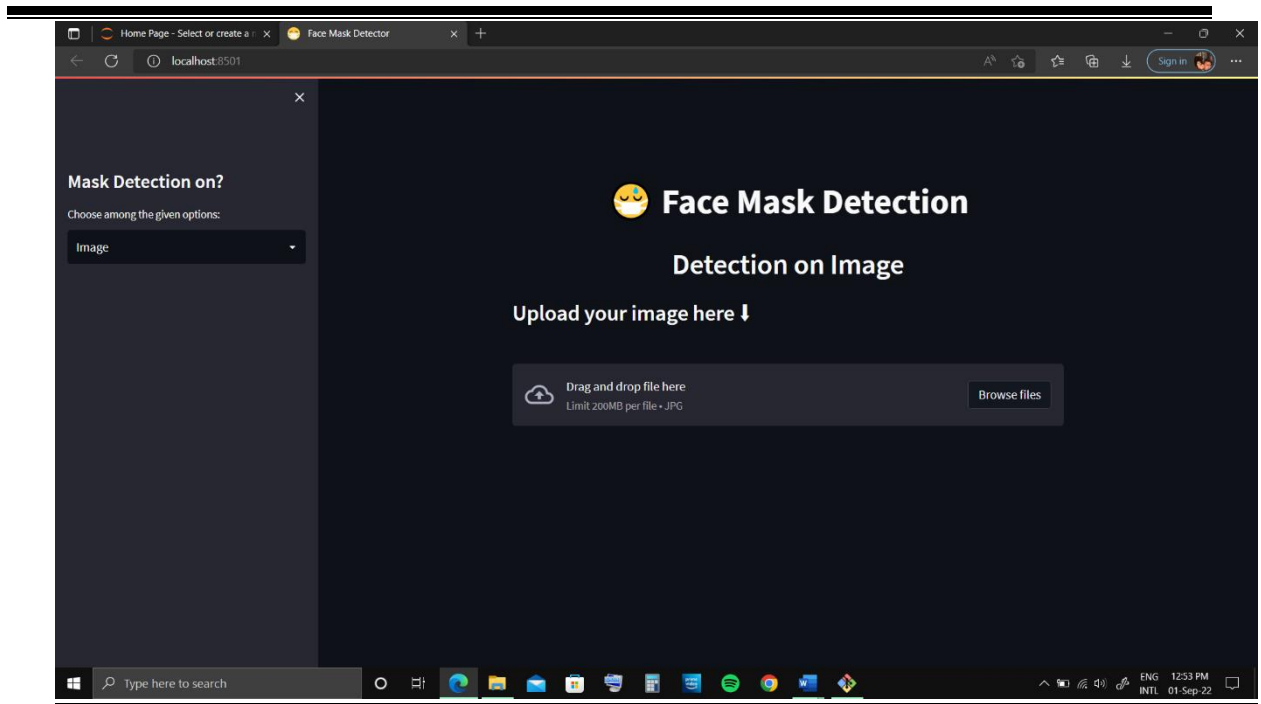




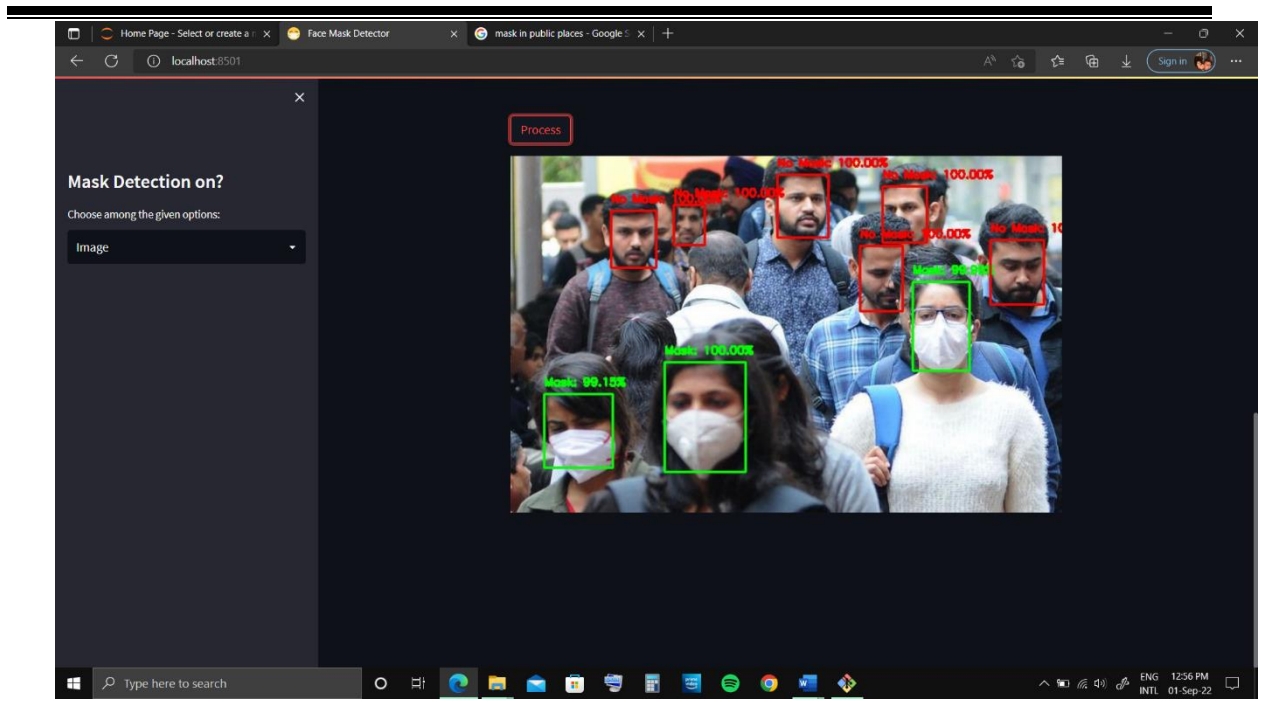
7.1 Real Time Output



Face Mask Detection



Face Mask Detection



CHAPER: 8

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

CHAPTER:9

FUTURE ENHANCEMENT

The possible outcome of the proposed project is to do a web user interface of application. The web cam will be open and it will first identify the face and it will detect whether person is wearing face mask or not. In the future identify the person and send mail to higher authorities who are not wearing mask and we can use other training model and more datasets to for more dataset's accuracy. We can collect the individual person face datasets and can identify. Jingdong's recognition accuracy is stronger than 99 percent. We created the MFDD, RMFRD, and SMFRD datasets, as well as a cutting-edge algorithm based on them. The algorithm will provide contactless face authentication in settings such as community access, campus governance, and enterprise resumption. Our research has given the world more scientific and technological strength.

CHAPTER: 10

BIBLIOGRAPHY

- [1] M. S. Ejaz and M. R. Islam, "Masked Face Recognition Using Convolutional Neural Network," 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), 2019, pp. 1-6, doi: 10.1109/STI47673.2019.9068044.
- [2] M. R. Bhuiyan, S. A. Khushbu and M. S. Islam, "A Deep Learning Based Assistive System to Classify COVID-19 Face Mask for Human Safety with YOLOv3," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)
- [1] M. M. Rahman, M. M. H. Manik, M. M. Islam, S. Mahmud and J. -H. Kim, "An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network," 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), 2020
- [1] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in Advances in neural information processing systems, 2014, pp. 198hy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," 2014.
- [4] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on, pp. 138–142, IEEE, 1994.
- [5] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," CoRR abs/1411.7923, 2014.
- [6] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, "A practical transfer learning algorithm for face verification," in Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 3208–3215, IE8–1996.
- [2] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," ACM computing surveys (CSUR), vol. 35, no. 4, pp. 399–458, 2003.
- [3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. KarpatEE, 2013.
- [7] P. N. Belhumeur, J. P. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," Pattern Analysis and Machine Intelligence, IEEE Transactions on 19(7), pp. 711– 720, 1997.
- [8] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, "A practical transfer learning algorithm for face verification," in Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 3208–3215, IEEE, 2013.

-
- [9] Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. CoRR, abs/1406.4773, 2014. 1, 2, 3
- [10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. Nature, 1986. 2, 4