REAL TIME SCRUTINYOF FACE MASK AT FIRM USING DEEP LEARNING FRAMEWORK

A PROJECT REPORT

Submitted by

NANTHINI E (913117104051)

SAILENDRI G R (913117104113)

VIKASHINI T P (913117104111)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VELAMMAL COLLEGE OF ENGINEERING AND TECHNOLOGY, VIRAGANOOR, MADURAI-625009

ANNA UNIVERSITY: CHENNAI 600 025

MARCH 2021

BONAFIDE CERTIFICATE

Certified that this project report "**REAL TIME SCRUTINY OF FACE MASK AT FIRM USING DEEP LEARNING FRAMEWORK**" is the bonafide work of "**NANTHINI E (913171104051), SAILENDRI GR (913117104111), VIKASHINI TP (913117104111)"** who carried out the project work under mysupervision.

SIGNATURE
Dr.P.ALLI, M.S., Ph.D.
HEAD OF THE DEPARTMENT
CSE DEPARTMENT
VELAMMAL COLLEGE OF ENGINEERING AND TECHNOLOGY MADURAI

SIGNATURE
Dr.G.VINOTH CHAKRAVARTHY M.E.,PhD.
SUPERVISOR
ASSOCIATE PROFESSOR
CSE DEPARTMENT
VELAMMAL COLLEGE OF ENGINEERING AND TECHNOLOGY MADURAI

Submitted for Project Viva Voce Examination held on _____

ABSTRACT

The COVID-19 pandemic caused by novel coronavirus is continuously spreading until now all over the world. Many countries require everyone to wear a mask in public to prevent the spread of coronavirus. 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. In this work, we propose an approach for detecting facial masks from video using deep learning. A deep learning architecture is trained on a dataset that consists of images of people with and without masks collected from various sources. Convolutional Neural Networks (CNN) can be trained to classify people wearing face masks with impressive accuracy. And if a person is found to be without mask we will be monitoring the duration of the person without mask in a database and an alert message indicating that person to wear face mask will be sent by using twilio in python.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background:

This project background is to identify whether the person working in a firm is wearing facial mask. Monitoring manually whether the person is wearing mask or not is a tedious task and nowadays people are not wearing mask while going to the public places. As the usage of smart phones are increasing during this time, we have planned to send a text message to the people who are without mask by monitoring them via a video stream and thereby helpful in reducing the spread of the novel corona virus.

1.2 Motivation:

In today's world COVID-19 has become a pandemic all over the world. People all over the world are facing challenging situations due to this pandemic. Every day a large number of people are being infected and died. Fever, dry cough, tiredness, diarrhea, loss of taste, and smell are the major symptoms of coronavirus which is declared by the World Health Organization (WHO). Many precautionary measures have been taken to fight against coronavirus. Among them cleaning hands, maintaining a safe distance, wearing a mask, refraining from touching eyes, nose, and mouth are the main, where wearing a mask is the simplest one. COVID-19 is a disease that spread from human to human which can be controlled by ensuring proper use of a facial mask. The spread of COVID-19 can be limited if people strictly maintain social distancing and use a facial mask. Very sadly, people are not obeying these rules properly which is speeding the spread of this virus. So our motive to identify the people who are not wearing facial mask inside the firm

and informing or suggesting that person to wear mask in public places, to prevent the spread of coronavirus.

1.3 Objectives:

The main aim is to develop an application to identify whether the people working in a firm are wearing facial mask. The workers will be monitored using the video stream. From that video stream the workers without mask will be identified and the duration of that person without mask will be monitored and stored in a database. The contact number of that person will be stored. By using their twilio account id the contact number of the person without mask will be retrieved and a text message indicating that person to wear facial mask will be sent. In existing systems, a person has to monitor manually or there are systems to classify people who are with masks and without masks. Here, our project explains a method of building a Face Mask Detector using Convolutional Neural Networks(CNN) Python, Keras,Tensorflow and OpenCV.Another important reason that we are sending text message to the person to wear facial mask. In simple words the objectives of the application is, to ensure that every Person working on a firm need to wear facial mask and thereby helpful in preventing the spread of the novel coronavirus.

1.4 Scope of the Project:

- To identify whether person working in a firm is wearing facial mask from the video stream.
- To monitor the duration of that person without mask .
- To obtain the contact number of that person.
- To alert the person to wear mask by sending a text message.

1.5 Limitations:

This application requires images of various person with and without mask to be trained. While training if we use a mask of a particular colour then we are in need to use the same colour mask during the entire process . suppose if we use different colour mask then it fails to classify whether the person is wearing mask or not. And we are sending text messages to the person to wear mask but there is no guarantee that every person will read the text message at that instant.

1.6 Importance of the Project:

It is important that you need to wear mask while going outside. This paper aims at designing a system to find out whether a person is using a mask or not and if a person is found to be without mask we are going to inform the person about the importance of wearing facial mask by a text message. Firstly, web cameras are used to capture real-time video footage of different people in the firm. From that video footage, facial images are extracted and these images are used to identify the mask on the face. The learning algorithm Convolutional Neural

Network (CNN) is used for feature extraction from the images. Whenever the architecture identifies people without face mask this information along with the duration without mask is stored in a database and the phone number of that person is identified from the database and an alert message indicating the person to wear the facial mask will be sent to take necessary precaution.

1.7 Functional Requirements:

A functional requirement defines a function of a system or its component, where a function is described as a specification of behavior between outputs and inputs.

1.7.1 Hardware Requirements:

- Main Processor : Above 2 GHz
- Ram : at least 4 GB
- Hard Disk : 10 GB
- OS – Windows 7, 8 and 10 (32 and 64 bit)
- System and webcamera

1.7.2 Software Requirements:

- Operating System :Windows OS
- Language :Python
- Developing Tool : Anaconda spyder notebook

1.8 Non Functional Requirements:

A non functional requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. The plan for

implementing non-functional requirements is detailed in the system architecture, because they are usually architecturally significant requirements.

Non functional requirements are in the form of “system shall be <requirement>”, an overall property of the system as a whole or of a particular aspect and not a specific function. The system’s overall properties commonly mark the difference between whether the development project has succeeded or failed. Non-functional requirements are often called “Quality Attributes” of a system. Other terms for non functional requirements are “Qualities”, ”Quality Goals”, ”Quality of Service Requirements”, ”Constraints”, ”Non-Behavioural Requirements”, or “Technical Requirements”. Informally these are sometimes called the “Ilities”, from attributes like stability and portability.

Nonfunctional requirements are those requirements which elaborate the performance characteristic of the system and define the constraints on how the system will do.

- Defines the constraints, targets or control mechanisms for the new system.
- Describes how well or to what standard a function should be provided.
- They are sometimes defined in terms of metrics to make them more tangible.
- Identify realistic, measurable target values for each service level.
- These include reliability, performance, service availability, responsiveness, throughput and security.

1.9 Report Organization

Chapter 1: Introduction

Chapter 2: Literature Survey

Chapter 3: Design Considerations

Chapter 4: Architecture of the project

Chapter 5: Testing and Deployment

Chapter 6: User Manual

Chapter 7: Conclusion and Future scope

Chapter 8: Code

Chapter 9 Bibliography

Introduction

This part gives us an introduction about the idea, scope of the project, impact of the project, the challenges we have phases and all the functional requirements and non-functional requirements.

Literature Survey

It consists of introduction or preparatory to a thesis or research report, it will suggest how the review findings will lead to the research the writer proposes to undertake.

Architecture of the project

This section describes the system architecture and the modules we have designed and the detailed description about all the modules.

Testing and Deployment

This section deals with the available testing methods and the method that we have implemented how they are deployed and maintained.

User Manual

This part describes the components and its specifications, and how they should be handled.

Conclusions and future scope

Here we described about our works of the project and the future scope for this project.

Code

This section describes the main coding and its functionalities and screenshots.

Bibliography

This part lists all the papers, books that we referred and get inference from them.

CHAPTER 2

LITERATURE SURVEY

2.1 Literature Survey:

This chapter deals with the survey of various tracking and announcement systems.

[1].Implementation of a Computer Vision Framework for Tracking and Visualizing Face Mask Usage in Urban Environments,2020 IEEE International Smart Cities Conference,2020.

Authors: Gabriel T.S. Draughon; Peng Sun; Jerome P. Lynch

In this paper, the author proposed the practical computer vision sensing framework for person and face mask tracking. The face mask detection module was trained and validated on a challenging low resolution face mask dataset curated from 50+ hours of surveillance camera footage. From this framework we learnt the methodology for monitoring the person continuously for hours under surveillance camera footage and instead of surveillance camera we tried it using webcamera.

Advantages:

- Works well on low resolution image.
- Data visualization can be done in cloud infrastructure.

Disadvantages:

- Space for some application tool is limited.
- Working on large number of dataset with high pixel value is difficult.

[2]. An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network ,published in 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS).

Author: Mohammad Marufur Rahman; Md. Motaleb Hossen Manik; Md. Milon Islam; Saifuddin Mahmud; Jong-Hoon Kim

In this paper, the author proposed the deep learning architecture which 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.

From this paper we have learnt more about the classification of image based on whether the person is wearing face mask or not with higher accuracy using the proposed deep learning framework.

Advantages:

The trained architecture achieved 98.7% accuracy on distinguishing people with and without a facial mask for previously unseen test data.

Disadvantages:

- The developed system faces difficulties in classifying faces covered by hands since it almost looks like the person wearing a mask.
- Since the information about the violator is sent via internet, the system fails when there is a problem in the network.

[3]. Study of Masked Face Detection Approach in Video Analytics, 2016 Conference on Advances in Signal Processing (CASP) Cummins College of Engineering for Women, Pune.

Author: Gayatri Deore, Ramakrishna Bodhula, Dr. Vishwas Udpikar

In this approach author has created a method simpler in complexity thereby making real time implementation feasible. General mask detection algorithms deal with complex algorithms like feature based algorithms and learning based algorithms. Methods based on facial features exploit the information of facial features such as mouth or skin color to decide whether there is mask on face or not. In this project author proposes support vector machines for occluded face recognition. In this approach occluded face detection with gabor wavelets, principle components analysis and support vector machine.

However we are not going to use gabor wavelet method but we learned a lot to detect face mask without exploiting the image which is detected from the video stream.

Advantages:

- Real time detection saves human resource and cost
- Monitoring people becomes easy.

Disadvantages:

- Face detection using gabor wavelets is computationally intensive.
- Needs more learning about the technique.

CHAPTER 3

DESIGN CONSIDERATIONS

3.1 Assumptions and Dependencies:

This system is used to limit the growth of covid-19. Firstly a learning architecture model is trained and tested by a dataset containing of the faces of the random people with and without the mask. The images from the CCTV camera are fed into the trained model and are classified as whether the person in the image is wearing a mask or not. If the person is not wearing the mask, then that individual person is identified and a remainder message is sent to the concerned person's mobile number. At the same time, this information is recorded simultaneously in order to find the frequent defaulters.

3.2 General Constraints:

Hardware Limitations	GPUs and TPUs are always preferable.
Algorithm	Efficient algorithm gives efficient results.
Dataset	It is always very difficult to balance between over fitting and under fitting datasets.

3.3Working Principle:

The working principle is given as below:

- Live stream is broke by then into images and preprocessed.
- Then the image is sent into the trained model.

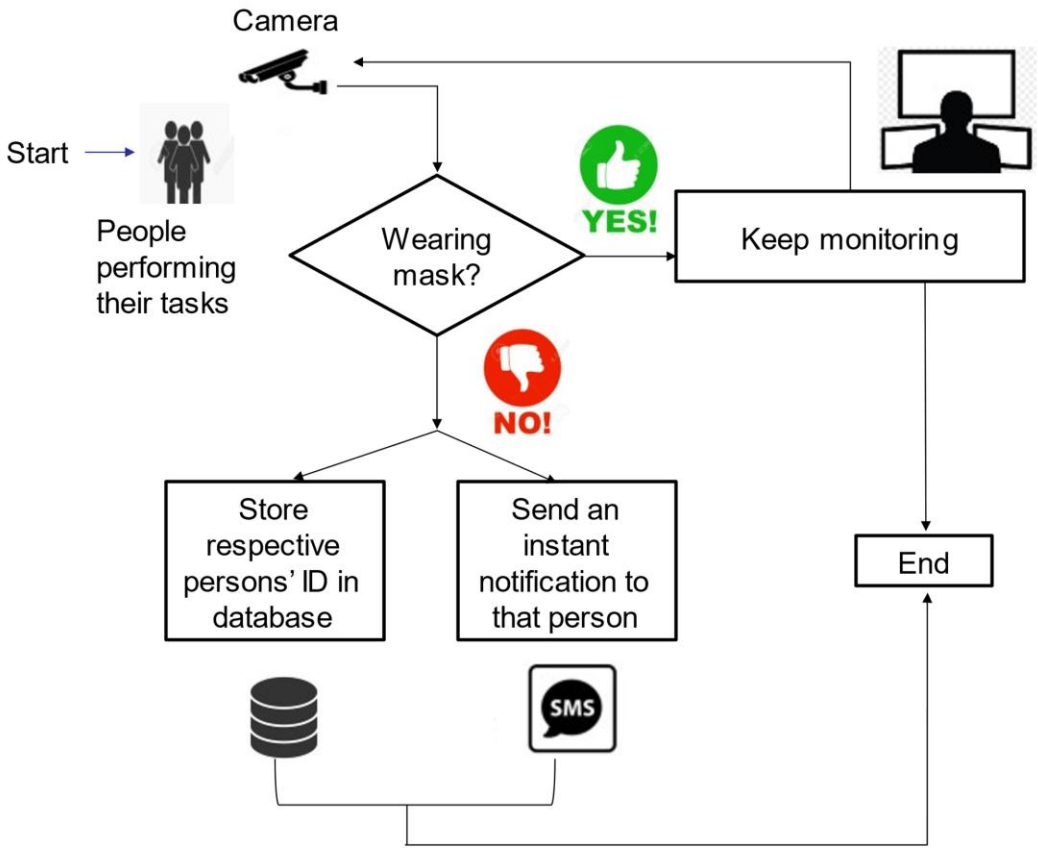
- If the person does not wear the mask, then an SMS alert is sent to him using twilio.
- His name is also sent to the authorities to avoid the repetition of defaulters.

CHAPTER 4

ARCHITECTURE OF THE PROJECT

This series describes the modules and the architecture of the system.

4.1 Architecture:



4.1.1 System architecture diagram

A face mask detection is a technique to find out whether someone is wearing a mask or not. It is similar to detect any object from a scene. Many systems have been introduced for object detection. Deep learning techniques are highly used in medical applications. Recently, deep learning architectures have shown a remarkable role in object detection. These architectures can be incorporated in

detecting the mask on a face. Moreover, nowadays every organisation or institution consists of CCTV cameras for security purpose. So the data collected from them are then used to perform different operations. Recently, the growth of COVID-19 can be reduced by detecting the facial mask in a smart city network.

This system aims at finding out whether a person is using a mask or not and informing the corresponding authority in a smart city network. Firstly, CCTV cameras are used to capture real-time video footage of different public places in the city. From that video footage, facial images are extracted and these images are used to identify the mask on the face. The learning algorithm MobileNetV2 is used for feature extraction from the images then these features are learned by multiple hidden layers. Whenever the architecture identifies people without face mask, a remainder SMS is sent to the concerned person's phone. The proposed system appraised promising output on data collected from different sources. We also represented a system that can ensure proper enforcement of the law on people who are not following basic health guidelines in this pandemic situation.

4.2 Technologies Used:

Deep Learning:

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g. in the forms of decisions. Understanding in this context means the transformation of visual images into descriptions of the world that make sense to thought processes and can elicit appropriate action. This image understanding can be seen as the

disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory.

Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning. A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The network moves through the layers calculating the probability of each output. For example, a DNN is trained to recognize animal will go over the given image and calculate the probability that the animal in the image is a certain breed. The user can review the results and select which probabilities the network should display.

CNN:

In deep learning, a convolutional neural network (CNN or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

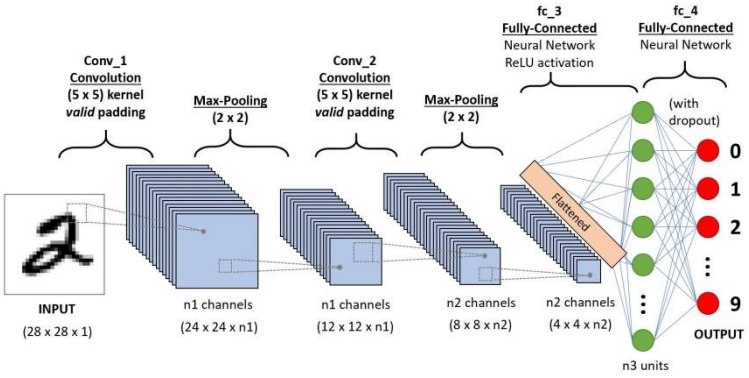


Fig.4.1 CNN ARCHITECTURE

They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series.

Mobilenetv2:

MobileNetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depth wise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.

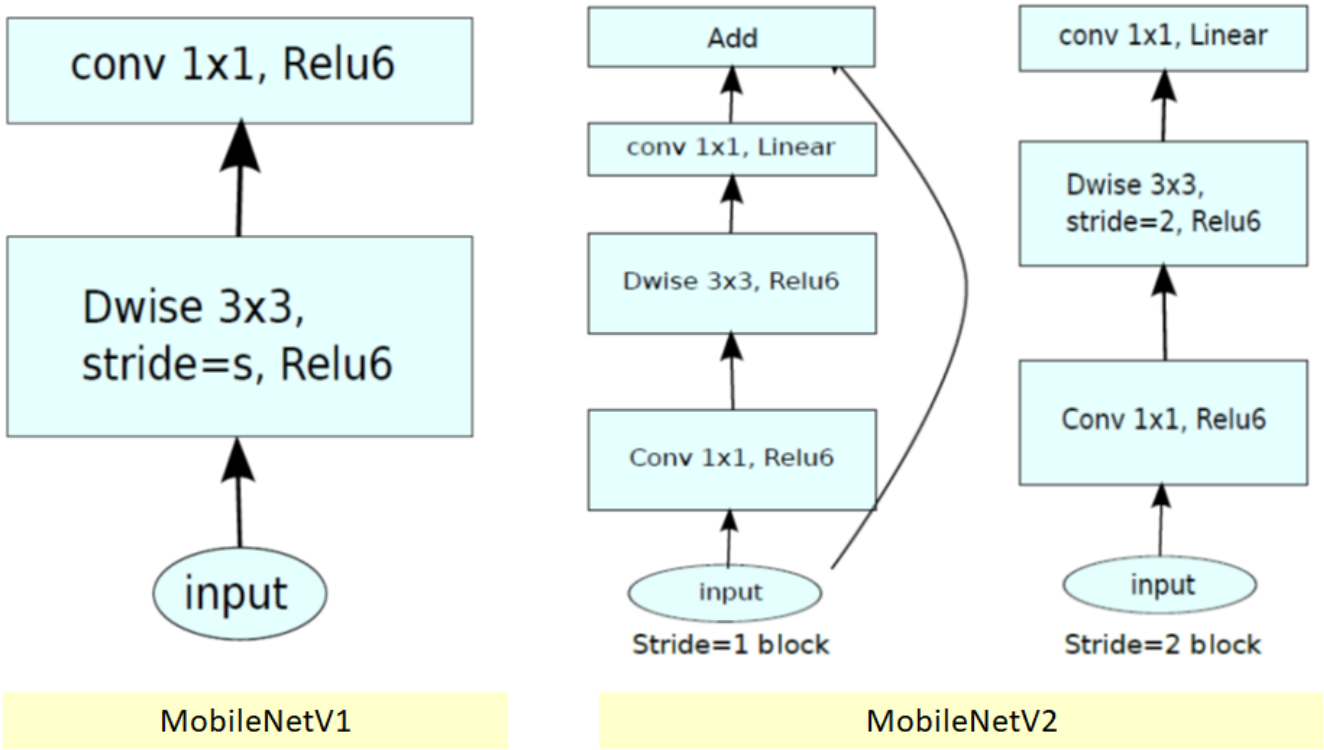
In MobileNetV2, there are two types of blocks. One is residual block with stride of 1. Another one is block with stride of 2 for downsizing. There are 3 layers for both types of blocks. The first layer is 1×1 convolution with ReLU6.The second layer is the depth wise convolution. The third layer is another 1×1 convolution but without any non-linearity. It is claimed that if ReLU is used again, the deep networks only have the power of a linear classifier on the non-zero volume part of the output domain.

Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

And there is an expansion factor t. And t=6 for all main experiments.

If the input got 64 channels, the internal output would get 64×t=64×6=384 channels.

With the recent breakthroughs that have been happening in data science, it is found that for almost all of these sequence prediction problems, Long short Term Memory networks, a.k.a LSTMs have been observed as the most effective solution.



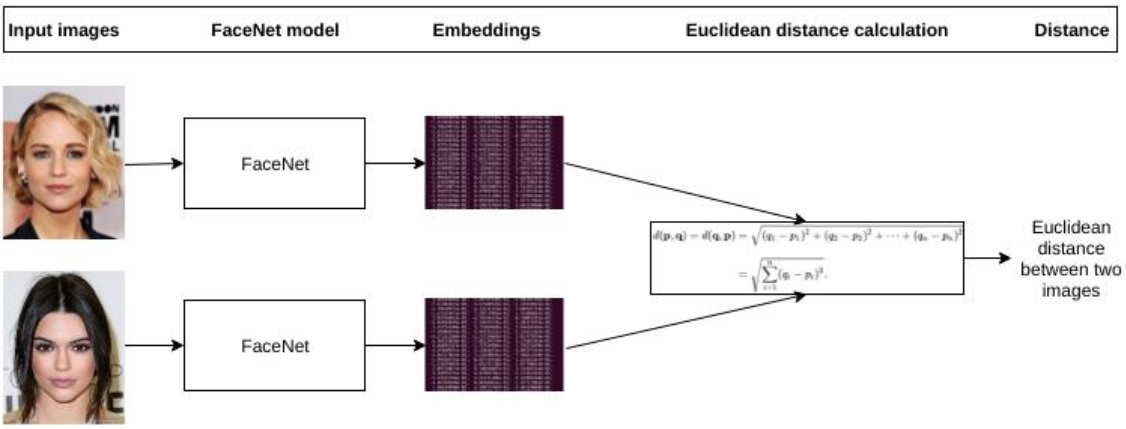
Facenet:

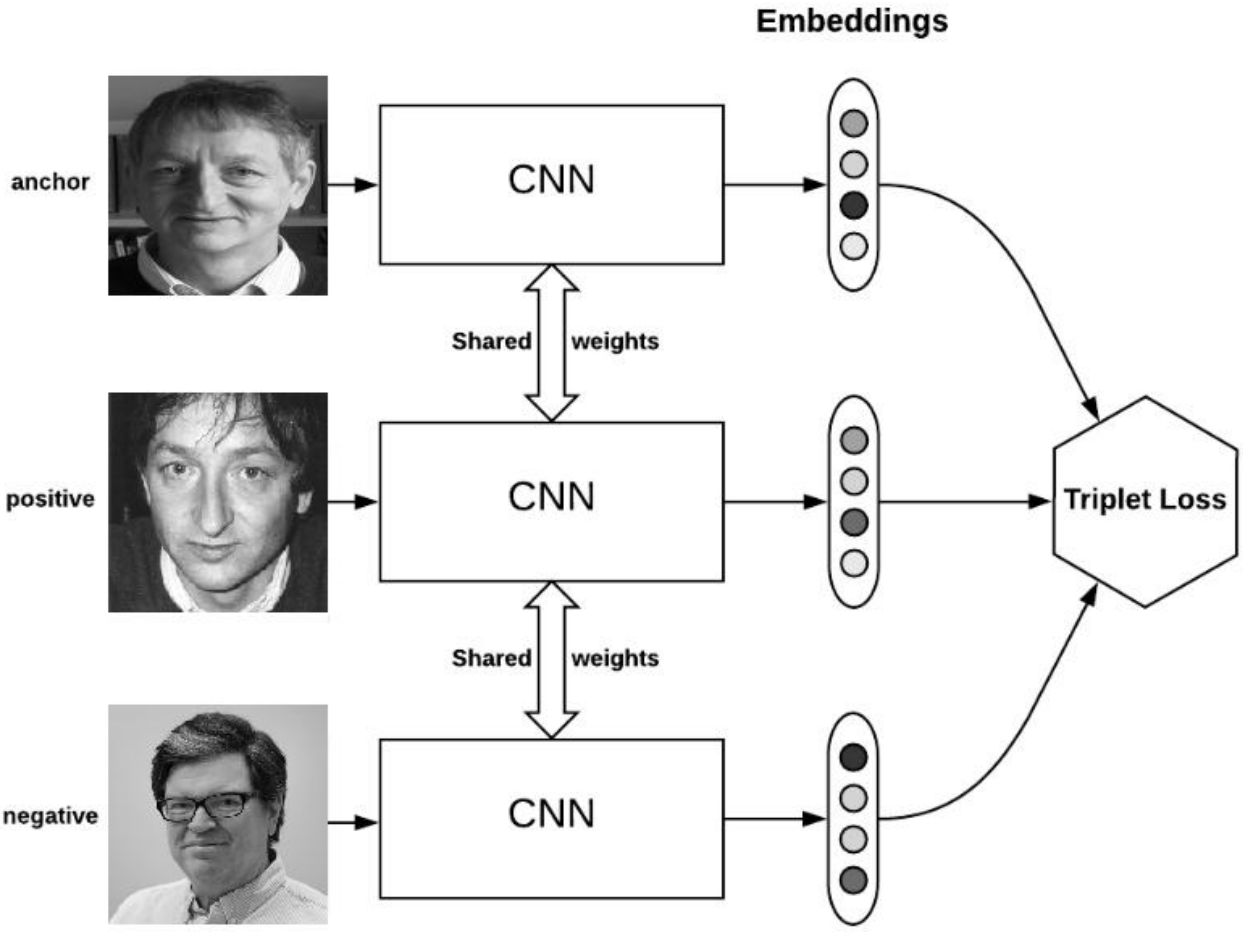
Facial recognition is one of the most exciting applications of deep learning. The rise in the adoption of facial recognition systems has been phenomenal in recent years however, it came under a lot of scrutiny lately. There are various AI aficionados who think that the usage of any kind of facial recognition system should be properly regulated in order to prevent nefarious activities.

The threats like data leakage, privacy violation etc. , originating from careless use of facial recognition systems are pretty real, and hence proper measures should be taken to avoid them, but even after all the recent criticism, you have to admit that it is still a pretty useful application which can be widely used to make people's lives better.

FaceNet provides a unique architecture for performing tasks like face recognition, verification and clustering. It uses deep convolutional networks along with triplet loss to achieve state of the art accuracy.

FaceNet provides a unified embedding for face recognition, verification and clustering tasks. It maps each face image into a euclidean space such that the distances in that space correspond to face similarity, i.e. an image of person A will be placed closer to all the other images of person A as compared to images of any other person present in the dataset.





The main difference between FaceNet and other techniques is that it learns the mapping from the images and creates embeddings rather than using any bottleneck layer for recognition or verification tasks. Once the embeddings are created all the other tasks like verification, recognition etc. can be performed using standard techniques of that particular domain, using these newly generated embeddings as the feature vector. For example we can use k-NN for face recognition by using embeddings as the feature vector and similarly we can use any clustering technique for clustering the faces together and for verification we just need to define a threshold value.

So, the most important thing to note here is that FaceNet doesn't define any new algorithm to carry out the aforementioned tasks, rather it just creates the

embeddings, which can be directly used for face recognition, verification and clustering.

FaceNet uses deep convolutional neural network (CNN). The network is trained such that the squared L2 distance between the embeddings correspond to face similarity. The images used for training are scaled, transformed and are tightly cropped around the face area.

Another important aspect of FaceNet is its loss function . It uses triplet loss function (refer to Fig 1). In order to calculate the triplet loss, we need 3 images namely anchor, positive and negative.

Anaconda:

Anaconda is open-source and Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. There are over 20 million users worldwide. For a solo practitioner, it is the toolkit that equips you to work with thousands of open-source packages and libraries.In cloud-based repository we can find and install over 7,500 data science and machine learning packages. With the conda-install command, we can start using thousands of open-source Conda, R, Python and many other packages. Individual Edition is an open source, flexible solution that provides the utilities to build, distribute, install, update, and manage software in a cross-platform manner. Conda makes it easy to manage multiple data environments that can be maintained and run separately without interference from each other. Anaconda Navigator is a desktop GUI that comes with Anaconda Individual Edition.

It makes it easy to launch applications and manage packages and environments without using command-line commands. These applications are available by default in Navigator Jupyter Lab,Spyder, PyCharm, Glueviz, Orange 3 App, Jupyter Notebook, RStudio, Anaconda Prompt(Windows only).

Spyder:

Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It offers a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package. It is available cross-platform through Anaconda, on Windows, on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian, Fedora, Gentoo Linux, openSUSE and Ubuntu.

Core components

- **Editor**

  Work efficiently in a multi-language editor with a function/class browser, real-time code analysis tools (pyflakes, pylint, and pycodestyle), automatic code completion (jedi and rope), horizontal/vertical splitting, and go-to-definition.

- **Interactive console**

  Harness the power of as many IPython consoles as you like with full workspace and debugging support, all within the flexibility of a full GUI interface. Instantly run your code by line, cell, or file, and render plots right inline with the output or in interactive windows.

- **Documentation viewer**

  Render documentation in real-time with Sphinx for any class or function, whether external or user-created, from either the Editor or a Console.

- **Variable explorer**

  Inspect any variables, functions or objects created during your session. Editing and interaction is supported with many common types, including numeric/strings/bools, Python lists/tuples/dictionaries, dates/timedeltas, Numpy arrays, Pandas index/series/dataframes, PIL/Pillow images, and more.

- **Development tools**

  Examine your code with the static analyzer, trace its execution with the interactive debugger, and unleash its performance with the profiler. Keep things organized with project support and a built-in file explorer, and use find in files to search across entire projects with full regex support.

Opencv:

OpenCV (Open Source Computer Vision Library) is an open source 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. These 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. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many start-ups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

Application areas include:

- 2D and 3D feature toolkits
- Ego motion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics

- Motion understanding
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Motion tracking
- Augmented reality

OpenCV (*Open source computer vision*) is a library of programming functions mainly aimed at real-time computer vision. The library is cross-platform and free for use under the open-source BSD license. OpenCV supports deeplearning frameworks like TensorFlow, Torch, PyTorch and Caffe according to a defined list of supported layers. It promotes OpenVisionCapsules which is a portable format, compatible with all other formats.

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- Boosting
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Support vector machine (SVM)
- Deep neural networks (DNN)

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. Besides that OpenCV also runs on the following desktop operating systems: FreeBSD, NetBSD, OpenBSD. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers. OpenCV runs on the following mobile operating systems: Android, iOS, Maemo, BlackBerry 10. The user can get official releases from SourceForge or take the latest sources from GitHub. OpenCV also uses CMake.

Tensorflow:

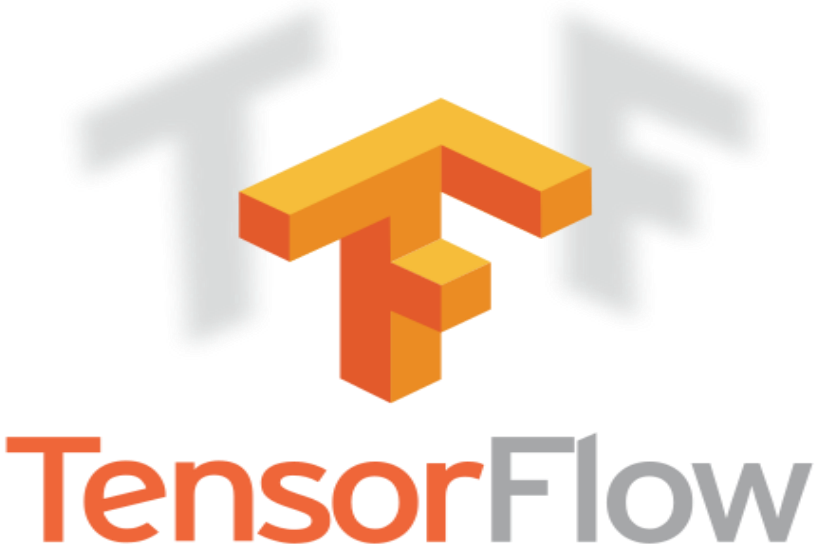


Fig.5.9 TENSORFLOW

TensorFlow is a free and open-source software platform for machine learning and 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 was developed by the Google Brain team for internal Google use.

It was released under the Apache License 2.0 on November 9, 2015. TensorFlow provides stable Python (for version 3.7 across all platforms) and C APIs; and without API backwards compatibility guarantee: C++, Go, Java, JavaScript and Swift (early release). Third-party packages are available for C#, Haskell, Julia, MATLAB, R, Scala, Rust, OCaml and Crystal.

Basically tensorflow is a computing graph processor with tensors as main data elements. "New language support should be built on top of the C API. However, not all functionality is available in C yet." Some more functionality is provided by the Python API.

TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow APIs are arranged hierarchically, with the high-level APIs built on the low-level APIs. Machine learning researchers use the low-level APIs to create and explore new machine learning algorithms. In this class, you will use a high-level API named tf.keras to define and train machine learning models and to make predictions. tf.keras is the TensorFlow variant of the open-source Keras API.

It includes many sequential and parallel operations related to statistical and deep learning. So it is very similar to other machine learning toolkits but it has the support of leading professionals including Caffe's architects which is one of the fastest convolutional network frameweoks. It also has a dynamic visualisation toolkit (tensorboard) and a high performance model server (serving).

Keras:

Keras is a high-level library that's built on top of Theano or TensorFlow. It provides a scikit-learn type API (written in Python) for building Neural Networks. Developers can use Keras to quickly build neural networks without worrying about the mathematical aspects of tensor algebra, numerical techniques, and optimization methods.

The key idea behind the development of Keras is to facilitate experimentations by fast prototyping. The ability to go from an idea to result with the least possible delay is key to good research.

This offers a huge advantage for scientists and beginner developers alike because they can dive right into Deep Learning without getting their hands dirty with low-level computations. The rise in the demand for Deep Learning has resulted in the rise in demand for people skilled in Deep Learning.

Every organization is trying to incorporate Deep Learning in one way or another, and Keras offers a very easy to use as well as intuitive enough to understand API

which essentially helps you test and build Deep Learning applications with least considerable efforts. This is good because Deep Learning research is such a hot topic right now and scientists need a tool to try out their ideas without wasting time on putting together a Neural Network model.

Salient Features of Keras

- Keras is a high-level interface and uses Theano or Tensorflow for its backend.
- It runs smoothly on both CPU and GPU.
- Keras supports almost all the models of a neural network – fully connected, convolutional, pooling, recurrent, embedding, etc. Furthermore, these models can be combined to build more complex models.
- Keras, being modular in nature, is incredibly expressive, flexible, and apt for innovative research.
- Keras is a completely Python-based framework, which makes it easy to debug and explore.

Sklearn:

Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

Please note that sklearn is used to build machine learning models. It should not be used for reading the data, manipulating and summarizing it. There are better libraries for that (e.g. NumPy, Pandas etc.)

Components of scikit-learn:

Scikit-learn comes loaded with a lot of features. Here are a few of them to help you understand the spread:

Supervised learning algorithms: Think of any supervised machine learning algorithm you might have heard about and there is a very high chance that it is part of scikit-learn. Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox. The spread of machine learning algorithms is one of the big reasons for the high usage of scikit-learn. I started using scikit to solve supervised learning problems and would recommend that to people new to scikit / machine learning as well.

Cross-validation: There are various methods to check the accuracy of supervised models on unseen data using sklearn.

Unsupervised learning algorithms: Again there is a large spread of machine learning algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.

Various toy datasets: This came in handy while learning scikit-learn. I had learned SAS using various academic datasets (e.g. IRIS dataset, Boston House prices dataset). Having them handy while learning a new library helped a lot.

Feature extraction: Scikit-learn for extracting features from images and text (e.g. Bag of words)

Community / Organizations using scikit-learn:

One of the main reasons behind using open source tools is the huge community it has. Same is true for sklearn as well. There are about 35 contributors to scikit-learn till date, the most notable being Andreas Mueller (P.S. Andy's machine learning cheat sheet is one of the best visualizations to understand the spectrum of machine learning algorithms).

Twilio:

Twilio is an American cloud communications platform as a service (CPaaS) company based in San Francisco, California. Twilio allows software developers to programmatically make and receive phone calls, send and receive text messages, and perform other communication functions using its web service APIs. Twilio uses Amazon Web Services to host telephony infrastructure and provide connectivity between HTTP and the public switched telephone

network (PSTN) through its APIs.

Twilio follows a set of architectural design principles to protect against unexpected outages, and received praise for staying online during the widespread Amazon Web Services outage in April 2011.

Twilio supports the development of open-source software and regularly makes contributions to the open-source community. In June 2010 Twilio launched OpenVBX, an open-source product that lets business users configure phone numbers to receive and route phone calls. One month later, Twilio engineer Kyle Conroy released Stashboard, an open-source status dashboard written in the Python programming language that any API or software service can use to display whether their service is functioning properly. Twilio also sponsors Localtunnel, created by

now ex-Twilio engineer Jeff Lindsay, which enables software developers to expose their local development environment to the public Internet from behind a NAT.

Twilio lists a number of other open-source projects on their website including, Flask Restful:

Python Flask (web framework) to build REST APIs,

Shadow:

Runs requests through a release candidate with real production traffic, Banker's Box:

Wrapper for storage backend.

CHAPTER 5

IMPLEMENTATION

5.1 Modules:

The various modules of this project have been explained below:

5.1.1 Preprocessing:

The images captured by the CCTV cameras require preprocessing. In the preprocessing step, the image is transformed into a greyscale image because RGB colour image contains too many redundant information which might not be required for mask detection. Then we reshape the images to maintain uniformity of the input images. Then the images are normalized and after the normalization, the value of pixels range from 0 to 1. Normalization helps the learning algorithm to learn faster and capture necessary features from the images.

5.1.2 Deep Learning Architecture:

The deep learning architecture learns the nonlinear features from the given samples. This learned architecture is used to predict the images that were taken by the live camera. To train our deep learning architecture, we collected images from different resources.

Data Collection:

Data from different resources were collected for training and testing the model. We collected a total of images of people with mask and images of people without mask. For training purposes 80% of the samples used and the rest were used for the testing purpose.

Architecture Development:

The learning model is based on mobilenetv2 which is useful for pattern

recognition in the images. The network comprises of one input layer, several hidden layers and one output layer. A flatten layer reshapes the information into a vector to feed into a dense network. The dense layer comprises of a series of neurons which learn the non-linear features. The dropout layer prevents the network from overfitting by dropping out units. Finally a dene layer containing two neurons distinguishes the classes. RsLu and softmax are used as the activation functions. Then Adam optimizer is applied.

5.1.3 Screening and reminding through SMS:

The main purpose of our system is to screen the persons who do not wear the face mask properly and remind them through a SMS. The learning architecture identifies whether any input image is without a face mask. If such a person is detected, then a remainder message is sent to their cell phone by using Twilio by recognizing that person through face net.This information is also recorded and sent to the corresponding authority in order to find the frequent defaulter. If proper actions are taken, people would wear the face mask properly and this would help greatly to limit the growth of covid-19.

5.2 Compilation Guidelines:

5.2.1 Target Environments:

In order to build the target application, a number of different compilation tests have been executed on different platforms. The target platforms requirements theoretically supported are listed in Table 6.1.1: Implementation Compilation Guidelines.

Target Platform	Core Application [Console view]	Core Application [GUI View]	Tests Executed [U denotes untested]
Linux	✓	✓	GCC 4.8.2
Microsoft Windows	✓	✓	Spyder Python 3.7 Version 64 bit/32 bit

Table 5.1.1: Implementation Compilation Guidelines

5.2.2 System Requirements:

The system requirements are extremely light beyond compilation of the core application. If the target device is hardware limited a cross compiler should be used on a different platform and computer.

Hardware	Target Device	Build Device
Memory	<64Mb Available RAM [Console] 512Mb Available RAM [GUI]	Recommended compile environment has a minimum of 2Gb or RAM. If you wish to compile all libraries(namely Qt)from source, the task was only successful with more than 10Gb Swap Space.
Processor	Single Core 1GHz	Dual Core Recommended or more.
Hard Disk	200Mb Free	10Gb recommended
Graphical Processor	None if using core	Console(Spyder)

	application as part of another application [Console] Required[GUI]	Or Microsoft Windows GUI

Table 5.1.2 System Requirements

CHAPTER 6

TESTING AND DEPLOYMENT

Testing is an investigation conducted to provide with the information about the quality of the product or service under test. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding bugs.

Testing can be stated as the process of validating and verifying that a product meets the requirements that guided its design and development, works as expected and can be implemented with the same characteristics.

6.1 Unit Testing:

In the unit testing the analyst tests the program making up a system. Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures ,and operating procedures, are tested to determine whether they are fit for use. The software units in a system are the modules and routines that are assembled and integrated to perform a specific function. In a large system, many modules on different levels are needed.

Unit testing can be performed from the bottom up starting with the smallest and lowest level modules and proceeding one at a time. For each module in a Bottom-up testing, a short program executes the module and provides the needed data.

6.2 Integration Testing:

Integration testing is any of hardware testing that seeks to verify the interfaces between components against a system design. Normally the former is considered a better practice since it allows interfaces issues to be located more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components(modules). Testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Progressively target groups of tested hardware components corresponding to elements of the architectural design are integrated and tested until the Integration testing is a systematic technique for constructing the program structure while conducting test to uncover errors associate with interfacing, Objectives are used to take unit test modules and built program structure that has been directed by design

6.3 System Testing:

System testing, or end to end testing, tests a completely integrated system to verify that it meets requirements. The purpose of this test is to evaluate the system's compliance with the specified requirements. System testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer based system.

In addition, the system testing should ensure that the program, as well as working as expected, does not also destroy or partially corrupt its operating environment or processes within that environment to become inoperative (this includes not corrupting shared memory, not consuming or locking up excessive resources and leaving any parallel processes unharmed by its presence) In this, the developed system is tested for proper working and check whether discrepancies between the system and its original objective

6.4 Deployment and Maintenance:

The Deployment Phase is the final phase of the software development life cycle and puts the product into production. After the project team tests the product and the product passes each testing phase, the product is ready to go live. This means that the product is ready to be used in a real environment by all end users of the product .Deployment starts directly after the system is appropriately tested, approved for release, and also or otherwise distributed into a production environment. This may involve installation, customization (such as by setting parameters to the customer's values), testing and possibly an extended period of evaluation. The phases include deployment preparation and procedures, product deployment, transferring ownership of the product, and closing the deployment phase. Maintaining and enhancing hardware to cope with newly discovered faults or requirements can take substantial time and effort, as missed requirements may force redesign of the system.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

Deep learning has lots of applications in today's world. We try to incorporate technology in all the fields possible. Similarly, to keep people safe from the spread of corona virus, we implemented this project, with SSD Model and MobilenetV2 architecture, which notifies the people not wearing mask instantly in messenger and also there is a database which stores the ID of the people. The system proposed in this study will act as a valuable tool to strictly impose the use of a facial mask in firms for all people. We tested this model with images and also with real time streaming of videos and the accuracy is also achieved. In future, we will try to add location by using GPS to the database so that if a firm has two or more branches, we can categorize them easily. And also we would like to integrate our work with CCTV cameras for mass screening purposes so that not only in firms, we can also use it in public places. As a result, with the help of these technologies,we can prevent the spread of corona virus and lead a happy life.

CHAPTER 8

CODE

8.1 To Create Augmented Dataset:

```
import os
from os import listdir
import sys
import random
import argparse
import numpy as np
from PIL import Image, ImageFile

__version__ = '0.3.0'


IMAGE_DIR = os.getcwd()+"\\images"
MASK_IMAGES=["default-mask.png","black-mask.png","blue-mask.png"]
MASK_IMAGE_PATHS =[os.path.join(IMAGE_DIR, mask_image) for
mask_image in MASK_IMAGES]
FACE_FOLDER_PATH=os.getcwd()+"\\dataset\\without_mask"
AUGMENTED_MASK_PATH=os.getcwd()+"\\dataset\\with_mask"



def create_mask(face_path,mask_path,augmented_mask_path,color):
    show = False
    model = "hog"
    FaceMasker(face_path, mask_path,augmented_mask_path,color=color).mask()



class FaceMasker:
    KEY_FACIAL_FEATURES = ('nose_bridge', 'chin')

    def __init__(self, face_path, mask_path,augmented_mask_path, show=False,
model='hog',color="default"):
        self.face_path = face_path
        self.mask_path = mask_path
        self.show = show
        self.model = model
```

```
        self.augmented_mask_path=augmented_mask_path
        self._face_img: ImageFile = None
        self._mask_img: ImageFile = None
        self.color=color

    def mask(self):
        import face_recognition

        face_image_np = face_recognition.load_image_file(self.face_path)
        face_locations = face_recognition.face_locations(face_image_np,
model=self.model)
        face_landmarks = face_recognition.face_landmarks(face_image_np,
face_locations)
        self._face_img = Image.fromarray(face_image_np)
        self._mask_img = Image.open(self.mask_path).convert("RGBA")

        found_face = False
        for face_landmark in face_landmarks:
            # check whether facial features meet requirement
            skip = False
            for facial_feature in self.KEY_FACIAL_FEATURES:
                if facial_feature not in face_landmark:
                    skip = True
                    break
            if skip:
                continue

            # mask face
            found_face = True
            self._mask_face(face_landmark)

        if found_face:
            if self.show:
                self._face_img.show()

            # save
            self._save()
        else:
            print('Found no face.')

    def _mask_face(self, face_landmark: dict):
        nose_bridge = face_landmark['nose_bridge']
        nose_point = nose_bridge[len(nose_bridge) * 1 // 4]
        nose_v = np.array(nose_point)
```

```
        chin = face_landmark['chin']
        chin_len = len(chin)
        chin_bottom_point = chin[chin_len // 2]
        chin_bottom_v = np.array(chin_bottom_point)
        chin_left_point = chin[chin_len // 8]
        chin_right_point = chin[chin_len * 7 // 8]

        # split mask and resize
        width = self._mask_img.width
        height = self._mask_img.height
        width_ratio = 1.2
        new_height = int(np.linalg.norm(nose_v - chin_bottom_v))

        # left
        mask_left_img = self._mask_img.crop((0, 0, width // 2, height))
        mask_left_width = self.get_distance_from_point_to_line(chin_left_point,
nose_point, chin_bottom_point)
        mask_left_width = int(mask_left_width * width_ratio)
        mask_left_img = mask_left_img.resize((mask_left_width, new_height))

        # right
        mask_right_img = self._mask_img.crop((width // 2, 0, width, height))
        mask_right_width = self.get_distance_from_point_to_line(chin_right_point,
nose_point, chin_bottom_point)
        mask_right_width = int(mask_right_width * width_ratio)
        mask_right_img = mask_right_img.resize((mask_right_width, new_height))

        # merge mask
        size = (mask_left_img.width + mask_right_img.width, new_height)
        mask_img = Image.new('RGBA', size)
        mask_img.paste(mask_left_img, (0, 0), mask_left_img)
        mask_img.paste(mask_right_img, (mask_left_img.width, 0),
mask_right_img)

        # rotate mask
        angle = np.arctan2(chin_bottom_point[1] - nose_point[1],
chin_bottom_point[0] - nose_point[0])
        rotated_mask_img = mask_img.rotate(angle, expand=True)

        # calculate mask location
        center_x = (nose_point[0] + chin_bottom_point[0]) // 2
        center_y = (nose_point[1] + chin_bottom_point[1]) // 2

        offset = mask_img.width // 2 - mask_left_img.width
        radian = angle * np.pi / 180
```

```
        box_x = center_x + int(offset * np.cos(radian)) - rotated_mask_img.width //
2
        box_y = center_y + int(offset * np.sin(radian)) - rotated_mask_img.height //
2

        # add mask
        self._face_img.paste(mask_img, (box_x, box_y), mask_img)

    def _save(self):
        new_face_path=self.augmented_mask_path+"\\with-mask-"+self.color+"-
"+self.face_path.split("\\")[-1]
        self._face_img.save(new_face_path)
        print(f'Save to {new_face_path}')

    @staticmethod
    def get_distance_from_point_to_line(point, line_point1, line_point2):
        distance = np.abs((line_point2[1] - line_point1[1]) * point[0] +
                          (line_point1[0] - line_point2[0]) * point[1] +
                          (line_point2[0] - line_point1[0]) * line_point1[1] +
                          (line_point1[1] - line_point2[1]) * line_point1[0]) / \
                   np.sqrt((line_point2[1] - line_point1[1]) * (line_point2[1] -
line_point1[1]) +
                           (line_point1[0] - line_point2[0]) * (line_point1[0] -
line_point2[0]))
        return int(distance)


if __name__ == '__main__':
    for MASK_IMAGE_PATH in MASK_IMAGE_PATHS:
        COLOR=MASK_IMAGE_PATH.split("\\")[-1].split(".")[0]
        FACE_IMAGE_PATHS=[os.getcwd()+"\\dataset\\without_mask\\"+path for
path in listdir(FACE_FOLDER_PATH)]
        for FACE_IMAGE_PATH in FACE_IMAGE_PATHS:
            print("face image path: ",FACE_IMAGE_PATH)
            create_mask(FACE_IMAGE_PATH,MASK_IMAGE_PATH
,AUGMENTED_MASK_PATH,COLOR)
```

8.2 To Train Augmented Dataset:

```
# USAGE
# python train_mask_detector.py --dataset dataset

# import the necessary packages
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

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

print(os.getcwd())
# construct the argument parser and parse the arguments
dataset_path=os.getcwd()+"//dataset"
model_path=os.getcwd()+"//model//mask_model"
plot_path=os.getcwd()+"//plot"


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

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")
imagePaths = list(paths.list_images(dataset_path))
imagePaths = [imagePath.replace("\\","//",-1) for imagePath in imagePaths]
data = []
labels = []

# loop over the image paths
for imagePath in imagePaths:
        # extract the class label from the filename
```

```
        label = imagePath.split("//")[-2]

        # load the input image (224x224) and preprocess it
        image = load_img(imagePath, target_size=(224, 224))
        image = img_to_array(image)
        image = preprocess_input(image)

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

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

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

# partition the data into training and testing splits using 75% of
# the data for training and the remaining 25% for testing
(trainX, testX, trainY, testY) = train_test_split(data, labels,
        test_size=0.20, stratify=labels, random_state=42)

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

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

# construct the head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
```

```
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

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

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

# compile our model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
	metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model.fit(
	aug.flow(trainX, trainY, batch_size=BS),
	steps_per_epoch=len(trainX) // BS,
	validation_data=(testX, testY),
	validation_steps=len(testX) // BS,
	epochs=EPOCHS)

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
	target_names=lb.classes_))

# serialize the model to disk
print("[INFO] saving mask detector model... path: %s"%(model_path+".h5"))
model.save(model_path+".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")
```

8.3 To Test The Model:

```
# 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
#from pygame import mixer
import numpy as np
import imutils
import time
import cv2
import os
import math

#system libraries
import os
import sys
from threading import Timer
import shutil
import time


import face_recognition
from datetime import datetime


#.....................................

def excel_func():
    path = os.getcwd()+"//dataset//att_without_mask"
```

```
images = []
classNames = []
myList = os.listdir(path)
print(myList)
for cl in myList:
    curImg = cv2.imread(f'{path}/{cl}')
    images.append(curImg)
    classNames.append(os.path.splitext(cl)[0])
    print(classNames)

print("classNames completed")
encodeListKnown = findEncodings(images)
print('Encoding Complete')

cap = cv2.VideoCapture(0)
matches=[]

success, img = cap.read()
#img = captureScreen()
imgS = cv2.resize(img,(0,0),None,0.25,0.25)
imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

facesCurFrame = face_recognition.face_locations(imgS)
encodesCurFrame = face_recognition.face_encodings(imgS,facesCurFrame)


for encodeFace,faceLoc in zip(encodesCurFrame,facesCurFrame):
    matches = face_recognition.compare_faces(encodeListKnown,encodeFace)
    faceDis = face_recognition.face_distance(encodeListKnown,encodeFace)
    print(faceDis)
    matchIndex = np.argmin(faceDis)

if matches[matchIndex]:
    name = classNames[matchIndex].upper()
    print(name+"hi")
    #else:
    #   name = "unknown"
y1,x2,y2,x1 = faceLoc
y1, x2, y2, x1 = y1*4,x2*4,y2*4,x1*4
cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,0),2)
cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,255,0),cv2.FILLED)
cv2.putText(img,name,(x1+6,y2-
6),cv2.FONT_HERSHEY_COMPLEX,1,(255,255,255),2)
pname = markAttendance(name)
print(pname+"hello")
```

```
    cv2.imshow('Webcam',img)
    cv2.waitKey(1)

    # importing the client from the twilio
    from twilio.rest import Client

    detailsdict = {"SAILENDRI": ["ACad9e820f038d6a4dc7aaed3446a9ffe7",
"defe618bc71028525b92e2bfc54a3ed9",
"+13213514012","+917868898423"],"VIKASHINI":
["AC679013caffa4d1ebdac437dcd3664339",
"d339ada8faa030abe1b121b5078e7723",
"4407036522","+919150098698"],"NANDHINI":
["AC874288b8bc33c7fbfb9ef3596aedf02c",
"76757066c3bc7ec48fe950b46130cdcc", "+13852066234","+917598470275"]}
    pname=pname.rstrip("\n")
    #name=name.rstrip("")
    x,y,z,a = detailsdict.get(name)
    print(x)
    print(y)
    print(z)
    print(a)
    # Your Account Sid and Auth Token from twilio account
    account_sid = x
    auth_token = y
    # instantiating the Client
    client = Client(account_sid, auth_token)
    # sending message
    message = client.messages.create(body='!!!Hey there please wear mask!!!',
from_=z
                                     , to=a)
    # printing the sid after success
    print(message.sid)



def findEncodings(images):
  encodeList = []
  for img in images:
      img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
      encode = face_recognition.face_encodings(img)[0]
      encodeList.append(encode)
  return encodeList

def markAttendance(name):
```

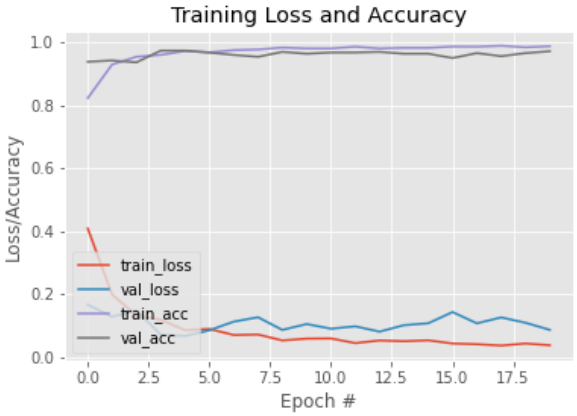
```
  with open('peoplelist.csv','r+') as f:
    myDataList = f.readlines()
    nameList = []
    for line in myDataList:
      entry = line.split(',')
      final_name = entry[0]
      nameList.append(entry[0])
    #if name not in nameList:
      nowtime = datetime.now()
    dtString = nowtime.strftime('%H:%M:%S')
    f.writelines(f'\n{name},{dtString}')
    print(final_name)
  #f = open("peoplelist.csv", "r+")
  return final_name
#### FOR CAPTURING SCREEN RATHER THAN WEBCAM
# def captureScreen(bbox=(300,300,690+300,530+300)):
#     capScr = np.array(ImageGrab.grab(bbox))
#     capScr = cv2.cvtColor(capScr, cv2.COLOR_RGB2BGR)
#     return capScr



while(True):
    detections = None
    def detect_and_predict_mask(frame, faceNet, maskNet,threshold):
        # grab the dimensions of the frame and then construct a blob
        # from it
      global detections
      (h, w) = frame.shape[:2]
      blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),(104.0, 177.0, 123.0))

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

        # initialize our list of faces, their corresponding locations,
        # and the list of predictions from our face mask network
      faces = []
      locs = []
      preds = []
        # loop over the detections
      for i in range(0, detections.shape[2]):
              # extract the confidence (i.e., probability) associated with
        confidence = detections[0, 0, i, 2]
```

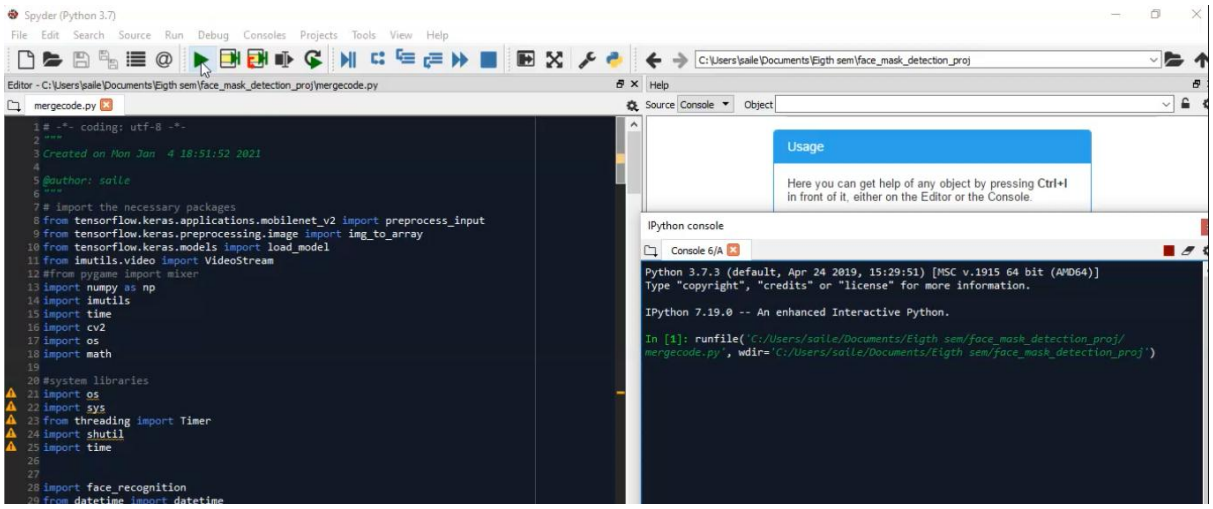
```
            # filter out weak detections by ensuring the confidence is
            # greater than the minimum confidence
        if confidence >threshold:
                # compute the (x, y)-coordinates of the bounding box for
                # the object
            box = detections[0, 0, i, 3:7]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

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

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

                # add the face and bounding boxes to their respective
                # lists
            locs.append((startX, startY, endX, endY))
                #print(maskNet.predict(face)[0].tolist())
            preds.append(maskNet.predict(face)[0].tolist())
    return (locs, preds)

# SETTINGS
   MASK_MODEL_PATH=os.getcwd()+"\\model\\mask_model.h5"
   FACE_MODEL_PATH=os.getcwd()+"\\face_detector"
   SOUND_PATH=os.getcwd()+"\\sounds\\alarm.wav"
   THRESHOLD = 0.5

# Load Sounds
#mixer.init()
#sound = mixer.Sound(SOUND_PATH)

# load our serialized face detector model from disk
   print("[INFO] loading face detector model...")
   prototxtPath = os.path.sep.join([FACE_MODEL_PATH, "deploy.prototxt"])
   weightsPath =
os.path.sep.join([FACE_MODEL_PATH,"res10_300x300_ssd_iter_140000.caffe
model"])
```

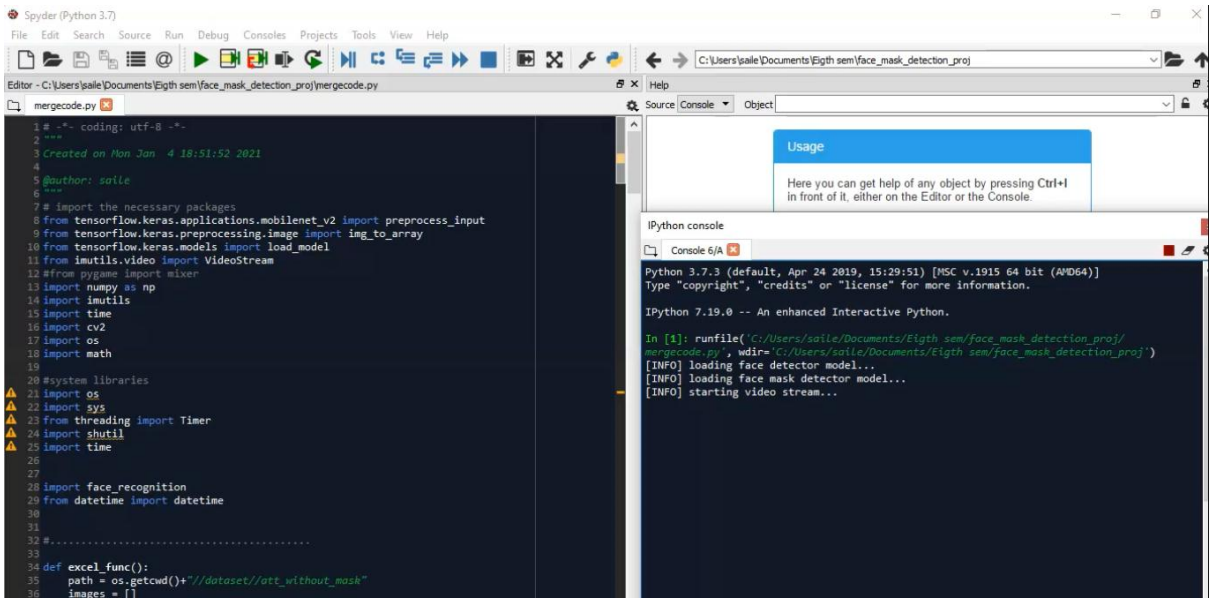
```
    faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

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

# initialize the video stream and allow the camera sensor to warm up
    print("[INFO] starting video stream...")
    vs = VideoStream(0).start()
    time.sleep(2.0)

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

          # 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
              #bounding box and text

            label = "Mask" if mask > withoutMask else "No Mask"

                #if(label=="No Mask") and (mixer.get_busy()==False):
                #    sound.play()


            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
```

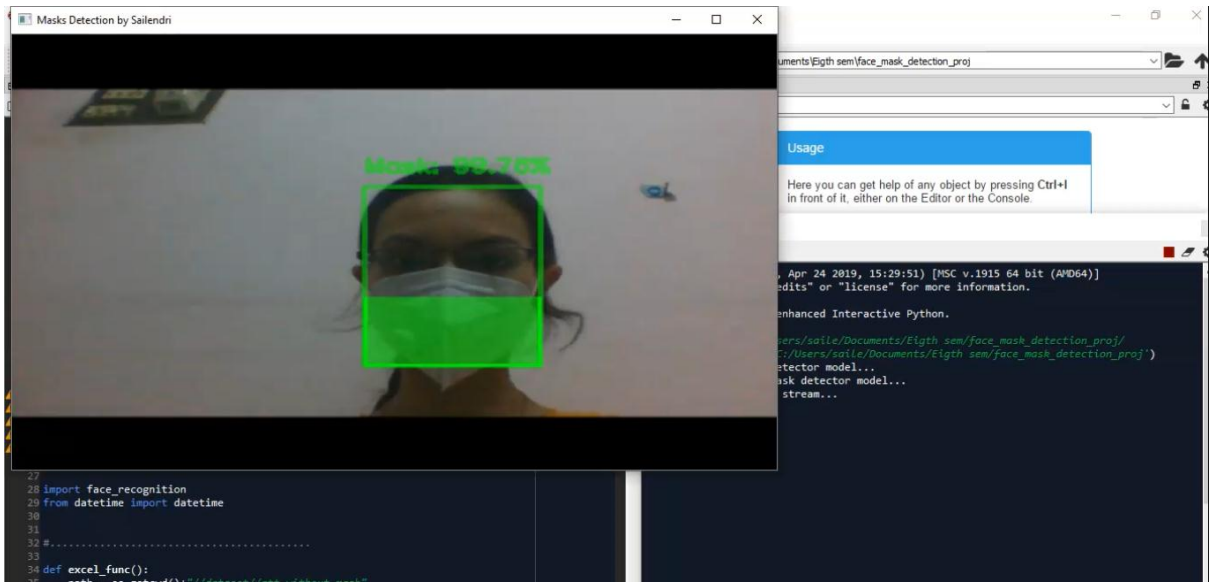
```
                # frame
            cv2.putText(original_frame, label, (startX, startY -
10),cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
            cv2.rectangle(original_frame, (startX, startY), (endX, endY), color, 2)
            cv2.rectangle(frame, (startX, startY+math.floor((endY-startY)/1.6)),
(endX, endY), color, -1)


        cv2.addWeighted(frame, 0.5, original_frame, 0.5 , 0,frame)

          # show the output frame
        frame= cv2.resize(frame,(860,490))
        cv2.imshow("Masks Detection by Sailendri", frame)
        key = cv2.waitKey(1) & 0xFF


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


# do a bit of cleanup
    cv2.destroyAllWindows()
  #  cap.release()
    vs.stop()
    import mergecode
    mergecode.py
```

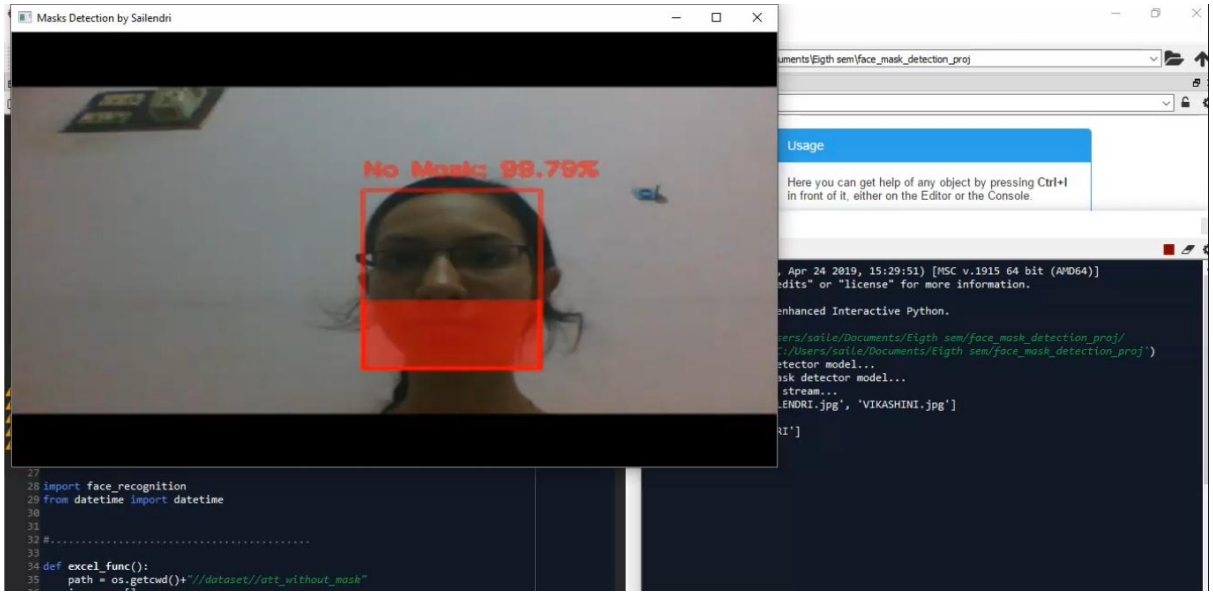
8.4 Screenshots:

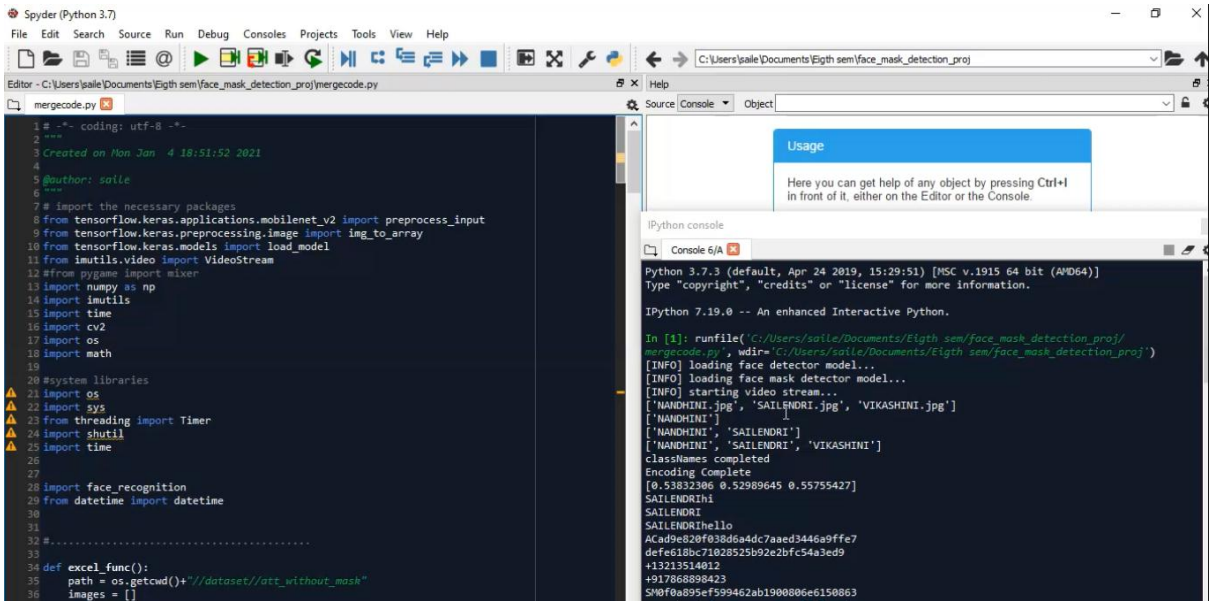




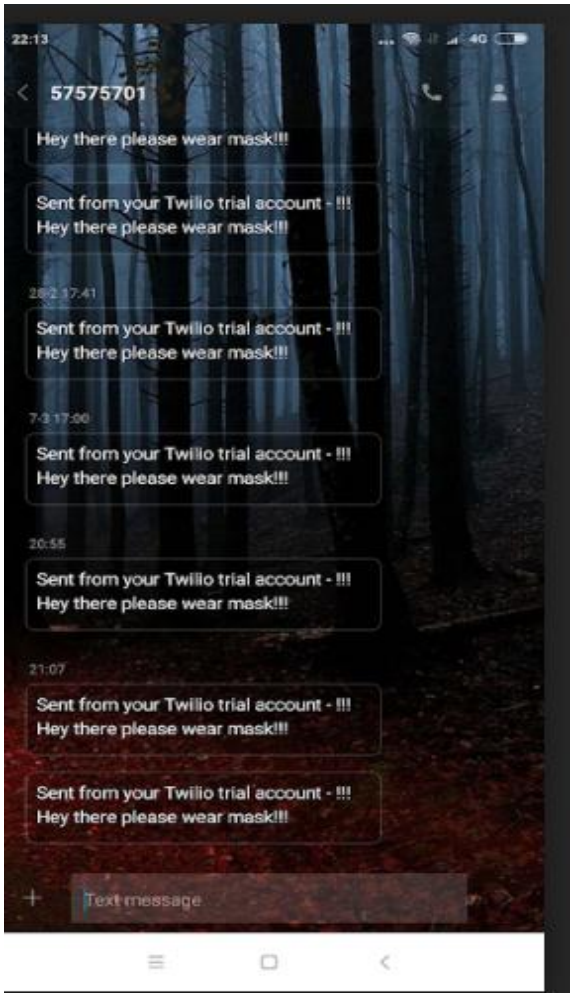




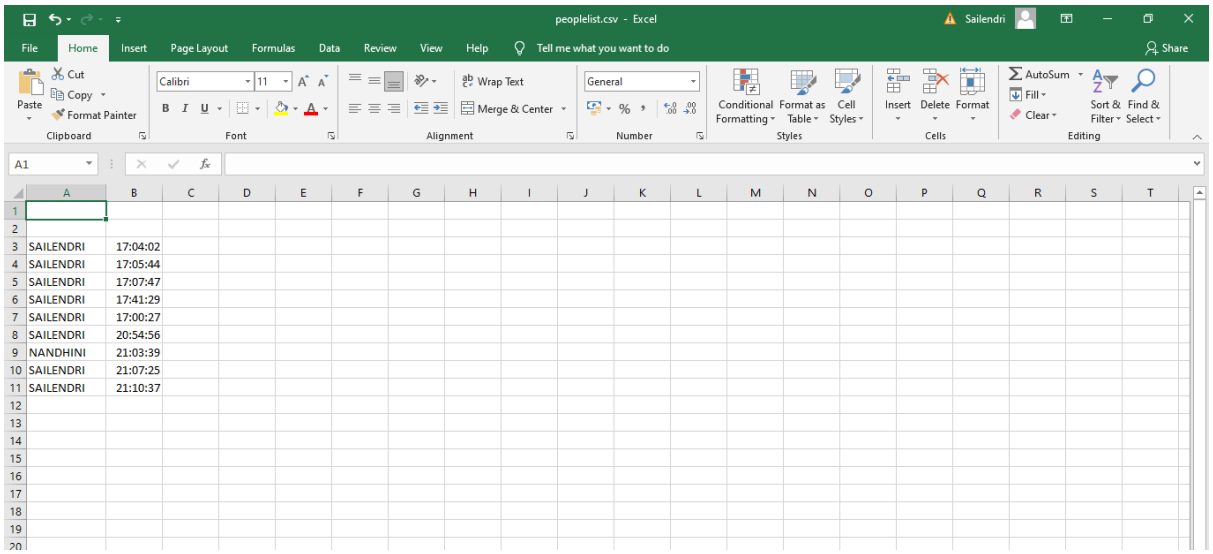




NOTIFICATION IN PHONE:



RECORD:



peoplelist.csv - Excel

	A	B
1		
2		
3	SAILENDRI	17:04:02
4	SAILENDRI	17:05:44
5	SAILENDRI	17:07:47
6	SAILENDRI	17:41:29
7	SAILENDRI	17:00:27
8	SAILENDRI	20:54:56
9	NANDHINI	21:03:39
10	SAILENDRI	21:07:25
11	SAILENDRI	21:10:37

CHAPTER 9

BIBLIOGRAPHY

- 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), Vancouver, BC, Canada, 2020, pp. 1-5, doi: 10.1109/IEMTRONICS51293.2020.9216386.

- A. Lodh, U. Saxena, A. khan, A. Motwani, L. Shakkeera and V. Y. Sharmasth, "Prototype for Integration of Face Mask Detection and Person Identification Model – COVID-19," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2020, pp. 1361-1367, doi: 10.1109/ICECA49313.2020.9297399.

- *Face Mask Detection | Kaggle*, [online] Available: https://www.kaggle.com/andrewmvd/face-mask-detection.

- Study of Masked Face Detection Approach in Video Analytics, Year: 2016,Author: Gayatri Deore, Ramakrishna Bodhula, Dr. Vishwas Udpikar, Prof. Vidya ,Published in: 2016 Conference on Advances in Signal Processing (CASP) Cummins College of Engineering for Women, Pune.