MINOR PROJECT

Submitted in partial fulfillment of the
Requirements for the award of the degree
of

Bachelor of Technology

in

Computer Science & Engineering

Department of Computer Science & Engineering
Guru Tegh Bahadur Institute of Technology

Guru Gobind Singh Indraprastha University
Dwarka, New Delhi
Year 2017-2021

Covid-19 Suraksha Kavach

By:

Bhavnoor Singh(18/CSE1/2017)
Guneet Pal Singh(28/CSE1/2017)
Jagmeet Singh(46/CSE1/2017)
Japjot Singh(48/CSE1/2017)

DECLARATION

I hereby declare that all the work presented in this Minor Project for the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **Computer Science & Engineering,** Guru Tegh Bahadur Institute of Technology, affiliated to Guru Gobind Singh Indraprastha University Delhi is an authentic record of our own work carried out by ourselves.

Date: 03/01/2021

Bhavnoor Singh(018/CSE1/2017)
Guneet Pal Singh(028/CSE1/2017)
Jagmeet Singh(046/CSE1/2017)
Japjot Singh(048/CSE1/2017)

ACKNOWLEDGEMENT

I would like to express our great gratitude towards **Mrs. Amanpreet Kaur** who has given us support and suggestions. Without their help we could not have presented this work upto the present standard. We also take this opportunity to give thanks to all others who gave us support for the project or in other aspects of our study at Guru Tegh Bahadur Institute of Technology.

Bhavnoor Singh(018/CSE1/2017) Date : 03/01/2021

Guneet Pal Singh(028/CSE1/2017)

Jagmeet Singh(046/CSE1/2017)

Japjot Singh(048/CSE1/2017)

ABSTRACT

The purpose of this document is to present a detailed description of the Covid-19 Suraksha Kavach. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This application helps the user to maintain the social distancing Measure with wearing a mask so that he can be protected, healthy and uninfected.

Covid-19 Suraksha Kavach will be applicable everywhere where citizens live. It will be more efficient and easier way to have a record on systems through which everyone can easily be protected from Covid-19 stay healthy wherever they go. This system will be helpful in every sector as and air pollution has been spreaded rapidly across the globe and this application can help the private and public bodies to ensure the safety among the citizens of the country.

CONTENTS

INTRODUCTION

1.1 INTRODUCTION TO PYTHON

Python is a general purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido Van Rossum during 1985-1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python is named after a TV show called ‘‘Monty Python’s Flying Circus’’. Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation. Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming]and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.[50]Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. The language's core philosophy is summarized in the document *The Zen of Python* (*PEP 20*), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal. Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks.

An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure.

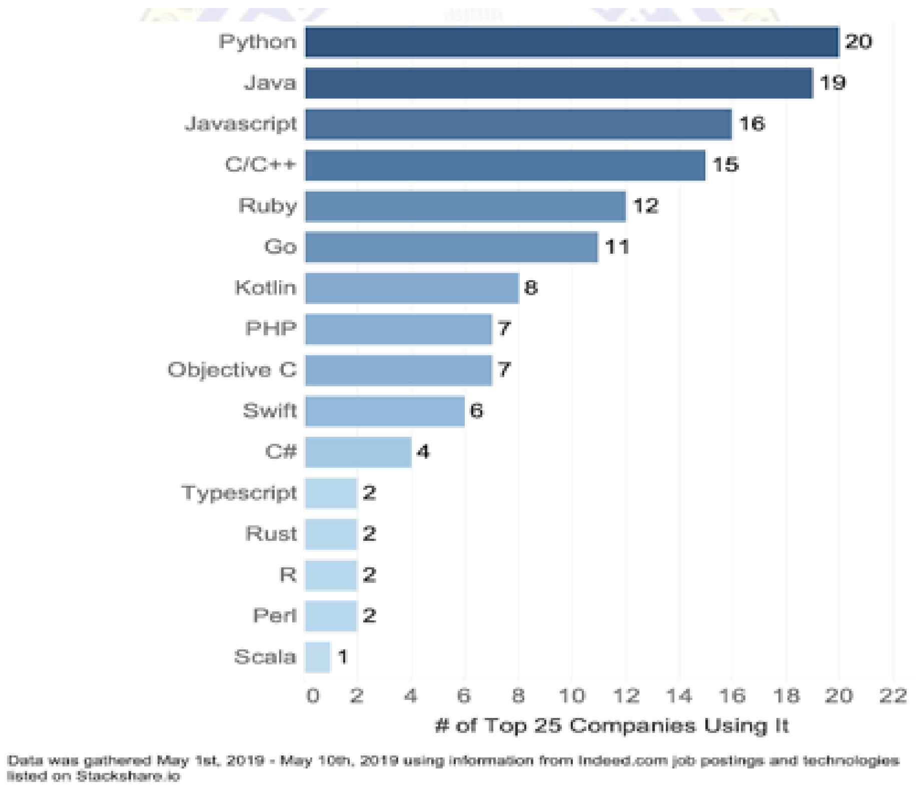


Fig: 1.1 Most-In- Demand Programming Technologies

There are two major Python versions- Python 2 and Python 3.

Python is a dynamic, interpreted (bytecode-compiled) language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and you lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that does not make sense as it runs.

Python 3.6.0 is used in the development of software requirement of our appliation. Numeric handling has been improved in many ways, for both floating-point numbers and for the decimal class. There are some useful additions to the standard library, such as a greatly enhance module, the module for parsing command-line options, convenient and counter classes in the collections module, and many other improvements.

1.2 Introduction to Machine Learning

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that

fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs

Any technology user today has benefitted from machine learning.Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consume

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

In this tutorial, we'll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-

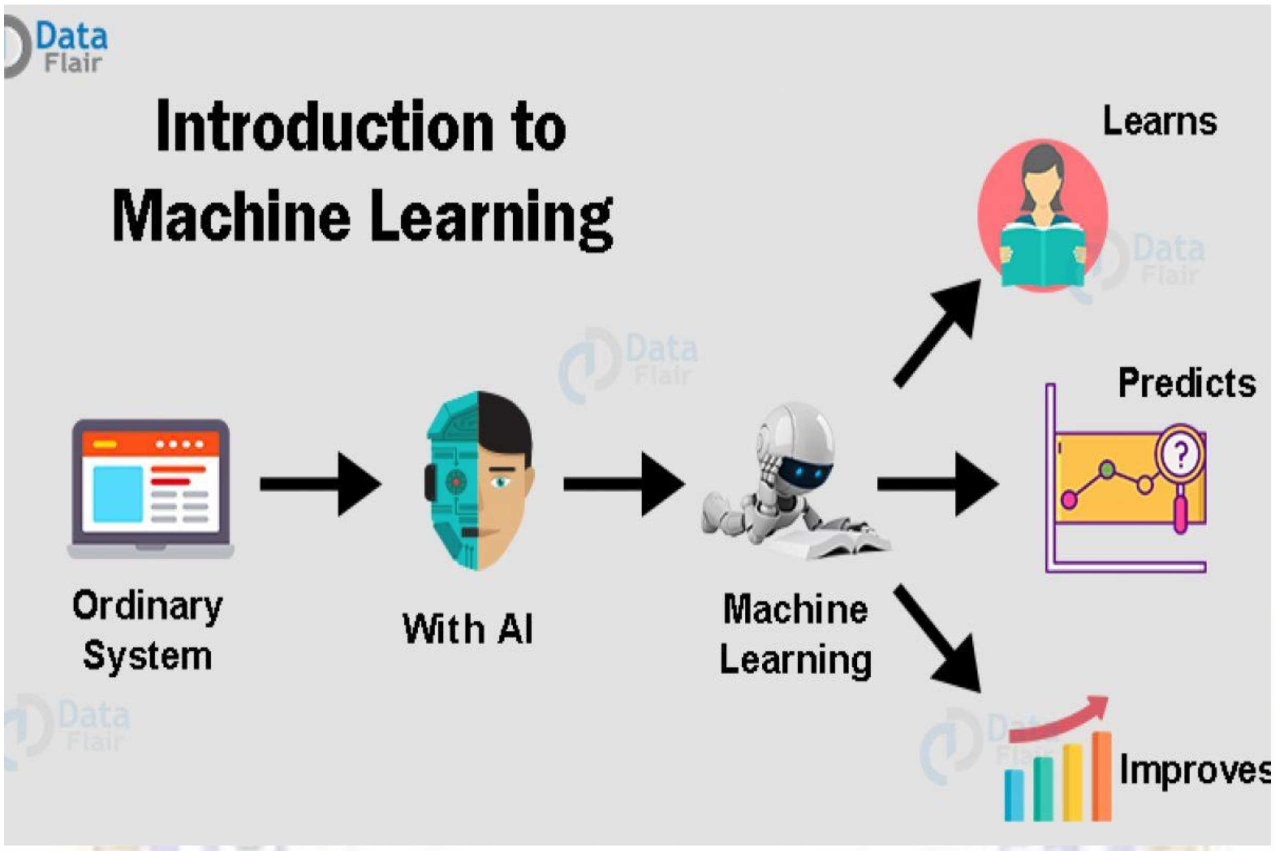


Fig 1.2: Introduction to machine learning

nearest neighbor algorithm, decision tree learning, and deep learning.

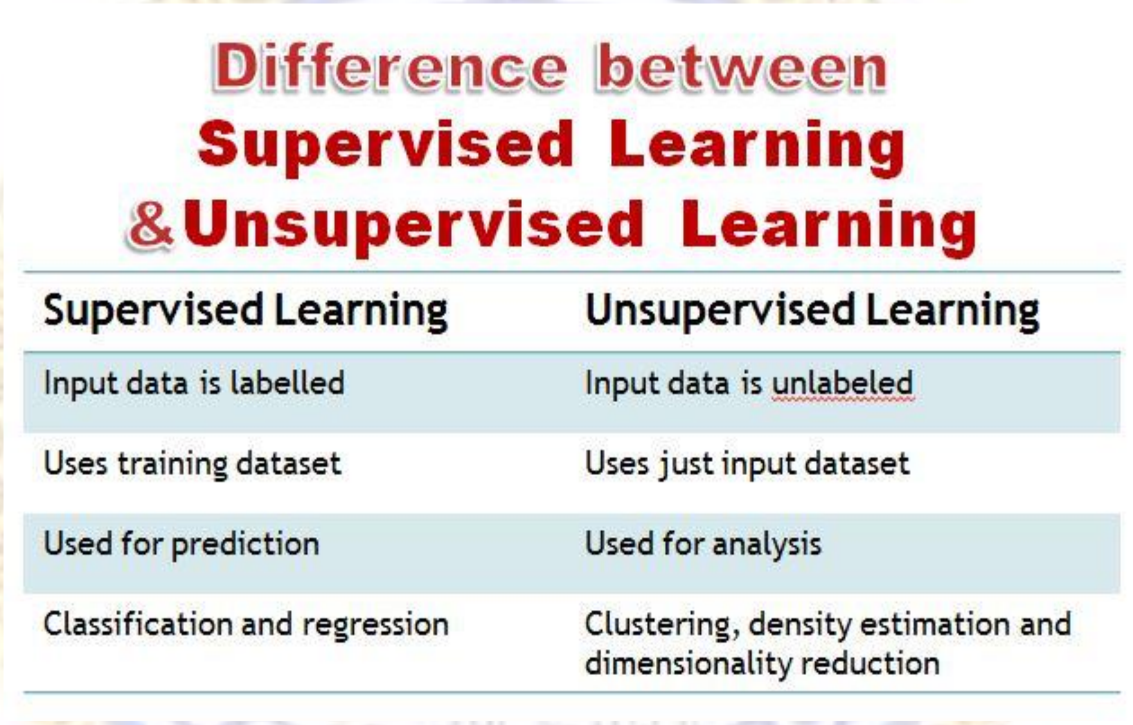
We'll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we'll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed. Two of the most widely adopted machine learning methods are supervised learning which trains algorithms based on example input and output data that is labeled by humans, and unsupervised learning which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let's explore these methods in more detail.

Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to "learn" by comparing its actual output with the "taught" outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data. For example, with supervised learning, an algorithm may be fed data with images of sharks labeled as



Difference between Supervised Learning & Unsupervised Learning

Supervised Learning	Unsupervised Learning
Input data is labelled	Input data is unlabeled
Uses training dataset	Uses just input dataset
Used for prediction	Used for analysis
Classification and regression	Clustering, density estimation and dimensionality reduction

Fig 1.3: Difference between supervised and unsupervised learning

Fish and images of oceans labeled as water. By being trained on this data, the supervised learning algorithm should be able to later identify unlabeled shark images as fish and unlabeled ocean images as water.

A common use case of supervised learning is to use historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails. In supervised learning, tagged photos of dogs can be used as input data to classify untagged photos of dogs.

Unsupervised Learning

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable.
The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases.
With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases.

Without being told a “correct” answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together.

1.3 Libraries used in Application

1.3.1 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.[5] 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 Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

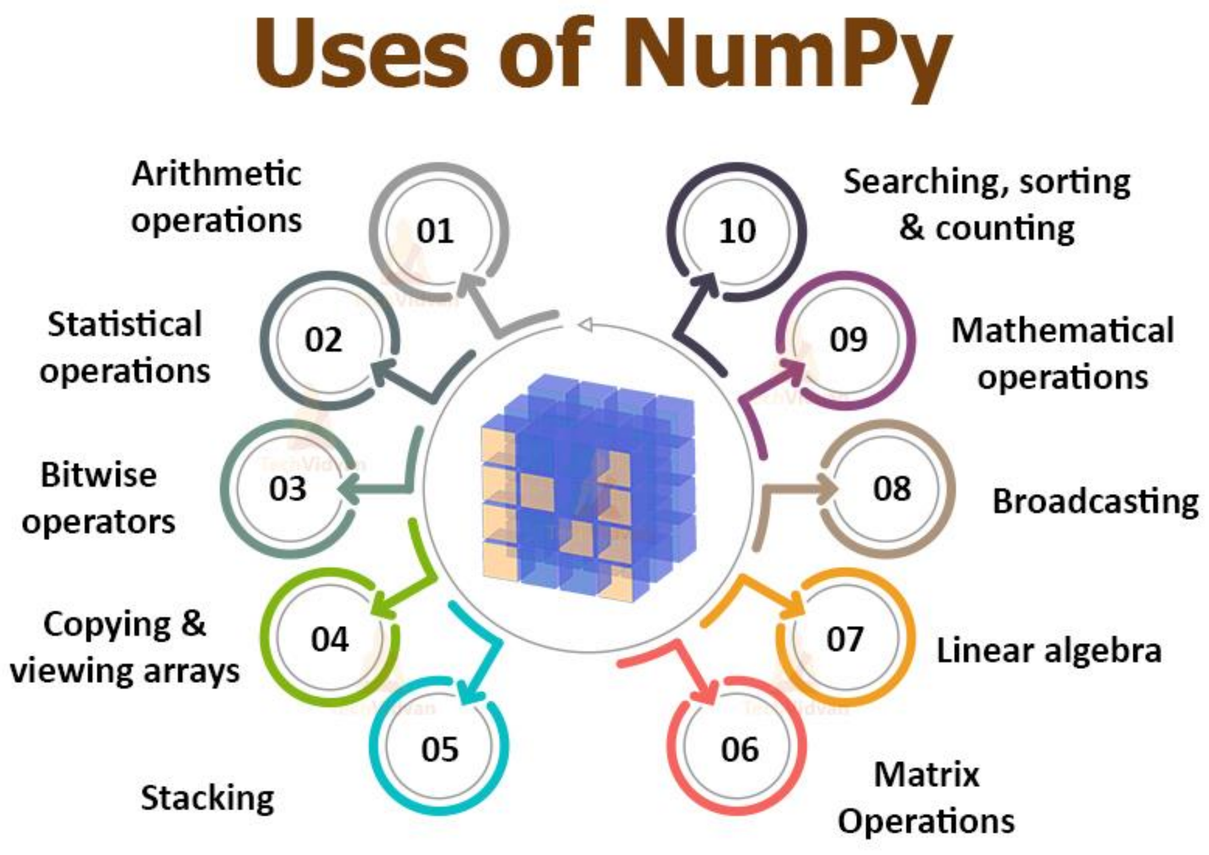


Fig 1.4: Applications of Numpy

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops, using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted,[18] and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available

1.3.2 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.

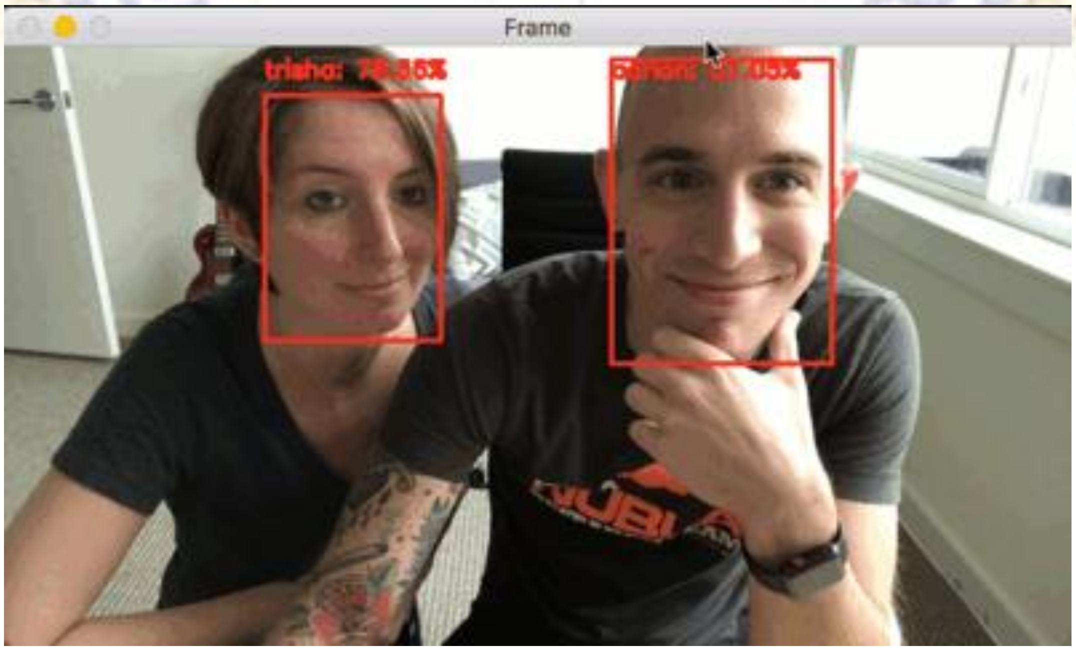


Fig 1.5: Example of OpenCv

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. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

1.3.3 SciPy

SciPy is an open-source Python library which is used to solve scientific and mathematical problems. It is built on the NumPy extension and allows the user to manipulate and visualize data with a wide range of high-level commands. As mentioned earlier, SciPy builds on NumPy and therefore if you import SciPy, there is no need to import NumPy

SciPy provides a number of special functions that are used in mathematical physics such as elliptic, convenience functions, gamma, beta, etc. To look for all the functions, you can make use of help() function as described earlier.

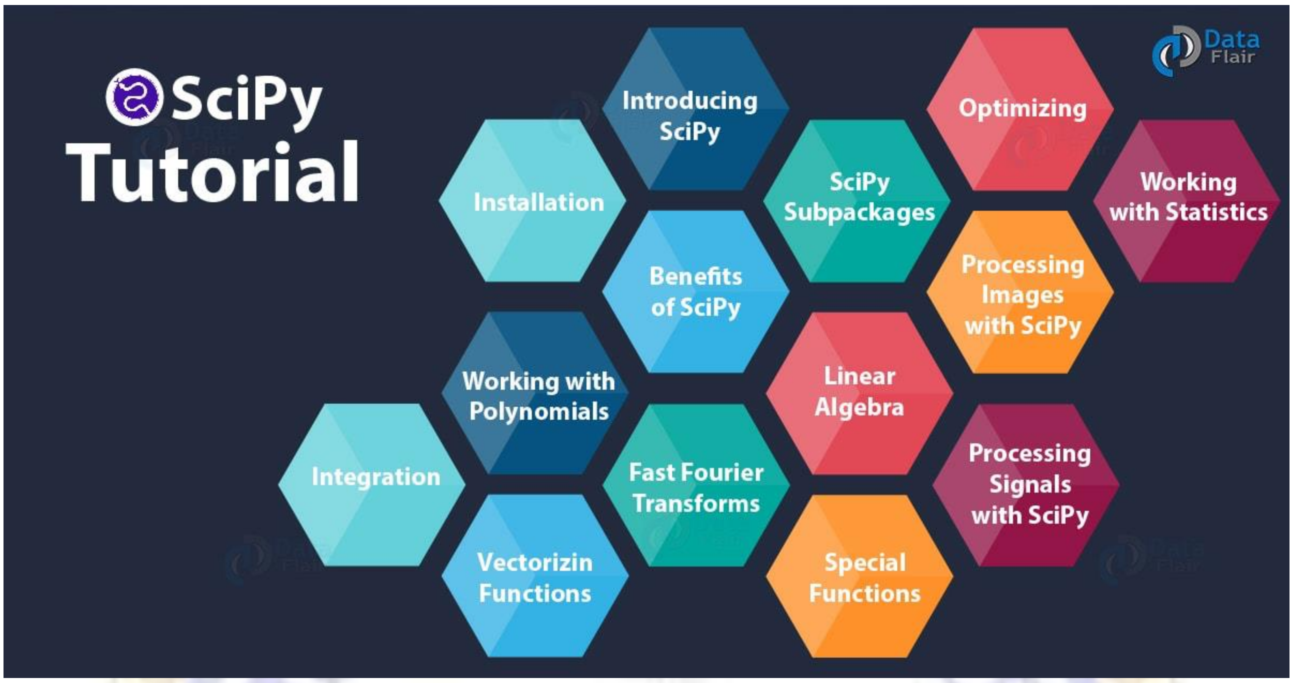


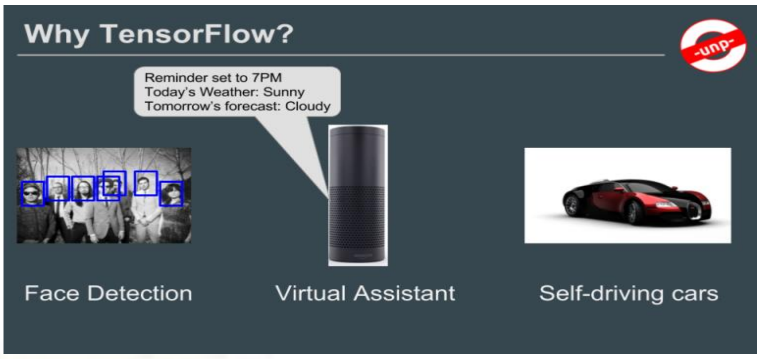
Fig 1.6: Uses of Scipy

1.3.4 Tensorflow

Machine learning is a complex discipline. But implementing machine learning models is far less daunting and difficult than it used to be, thanks to machine learning frameworks—such as Google's TensorFlow—that ease the process of acquiring data, training models, serving predictions, and refining future results.

Created by the Google Brain team, TensorFlow is an open source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor.

It uses Python to provide a convenient front-end API for building applications with the framework, while executing those applications in high-performance C++.



TensorFlow allows developers to create *dataflow graphs*—structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or *tensor*.

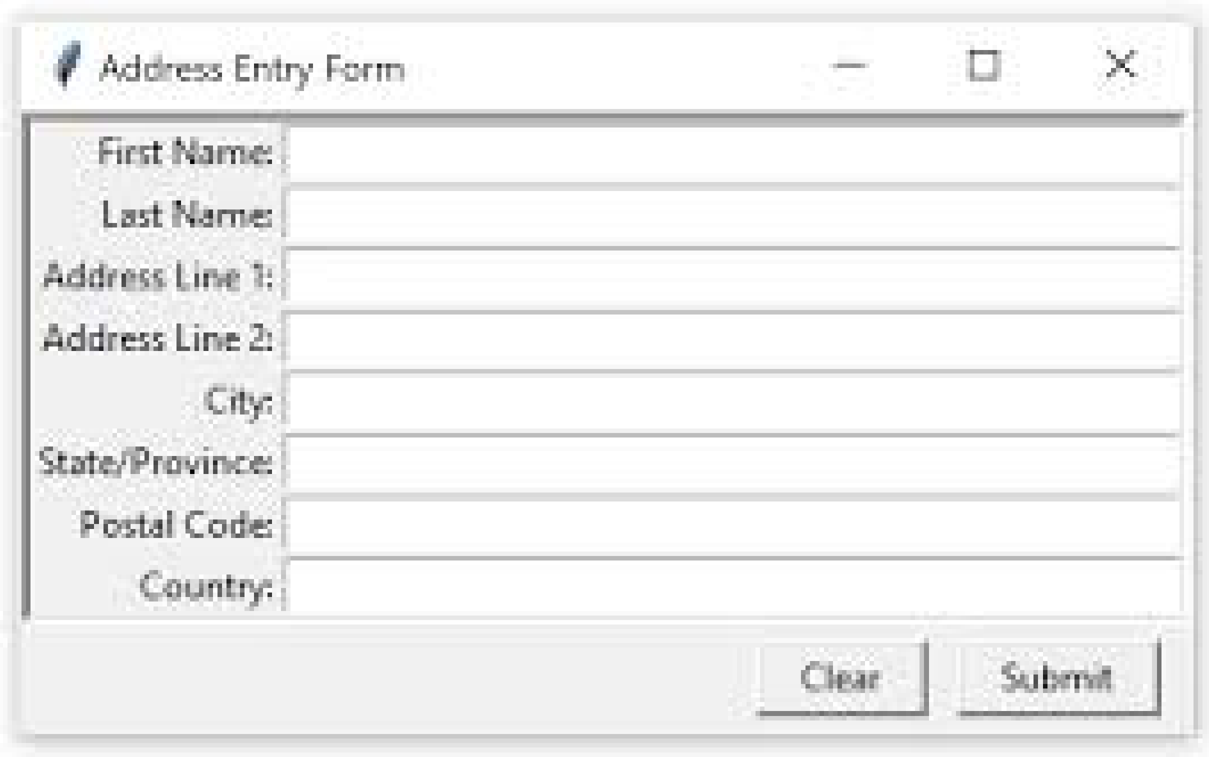
TensorFlow provides all of this for the programmer by way of the Python language. Python is easy to learn and work with, and provides convenient ways to express how high-level abstractions can be coupled together. Nodes and tensors in TensorFlow are Python objects, and TensorFlow applications are themselves Python applications.

1.4 Overview of COVID-19 Suraksha Kavach

Covid-19 Suraksha Kavach is made with the help of Tkinter which is one of the concepts of python.

It uses tkinter module for the GUI. Talking about the application, the user can identify its face with an accuracy to wear the mask and be protected from the covid-19 and air pollution.This has also helped to maintain social distancing norm by staying away at a distance in our application to stay protected and healthy. Tkinter has integrated the python code to work as an application to make it more user friendly and helpful to be protected from Covid-19.

This application can be used to help the people from not being affected by corona virus by recognising the face and checking the mask accuracy with minimum distance required between two people to stay protected and healthy.


Address Entry Form

First Name:	
Last Name:	
Address Line 1:	
Address Line 2:	
City:	
State/Province:	
Postal Code:	
Country:	

Fig 1.7: (For Example) Address entry form based on tkinter

What is Tkinter?

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.

Enter Your Name
Enter

TKINTER PROGRAMMING

- GUI (Graphical User Interface) is a human-computer interface that uses windows, icons and menus instead of plain text and uses a mouse as well as a keyboard as an input device.

- Enter the main event loop to take action against each event triggered by the user.

Fig 1.8: Tkinter Programming

To create using tkinter Application:
1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the python code. Note that the name of the module in Python 2.x is ‘Tkinter’ and in Python 3.x is ‘tkinter’.
import tkinter
There are two main methods used you the user need to remember while creating the Python application with GUI.

1. Tk(screenName=None, baseName=None, className=’Tk’, useTk=1): To create a main window, tkinter offers a method ‘Tk(screenName=None, baseName=None, className=’Tk’, useTk=1)’. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

   m=tkinter.Tk() where m is the name of the main window object

2. mainloop(): There is a method known by the name mainloop() is used when you are ready for the application to run. Mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event till the window is not closed.

1.5 Objective of the project

This project will serve the following objectives:

a. Covid-19 Suraksha Kavach helps us to do following:

1.Face Identification

2. Mask Identifcation

3. Accuracy of mask worn

4. Alert box in Mask Identification

5. Distance Measuring between two faces.

6.Warning to maintain distance.

b. It provides an efficient way to protect people from Covid-19 .

c. This is a GUI based application which is easy to understand and use.

d. User can view the the accuracy of the mask he has worn.

e. This is a secure application that is very user friendly , easy to work and flexible for the user.

Fig 1.9: Person Wearing Face mask

Fig 1.10: Maintaining Social Distance

1.6 Purpose of the project

It uses tkinter module for the GUI. Talking about the application, the user can identify its face with an accuracy to wear the mask and be protected from the covid-19 and air pollution.This has also helped to maintain social distancing norm by staying away at a distance in our application to stay protected and healthy. Tkinter has integrated the python code to work as an application to make it more user friendly and helpful to be protected from Covid-19.
This application can be used to help the people from not being affected by corona virus by recognising the face and checking the mask accuracy with minimum distance required between two people to stay protected and healthy.
The aim of the project Covid-19 Suraksha Kavach is to ensure the safety of the citizens in the country by taking safety norms like the features face mask recognition and to maintain minimum distance between the citizens to stay protected from covid-19 and stay healthy.

We can use these features in offices to be aware whether the mask is worn by every employee and we can use it in traffic signals and public places like Malls,Metro etc to have inspection on the citizens following the social distancing norms or not.

2. Requirement Analysis with SRS

2.1 Introduction:

2.1.1 Purpose: The purpose of this document is to present a detailed description of the Covid-19 Suraksha Kavach. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This application helps the user to maintain the social distancing Measure with wearing a mask so that he can be protected , healthy and uninfected.

2.1.2. Scope: An Covid-19 Suraksha Kavach will be applicable everywhere where citizens live. It will be more efficient and easier way to have a record on systems through which everyone can easily be protected from Covid-19 stay healthy wherever they go.This system will be helpful in every sector as and air pollution has been spreaded rapidly across the globe and this application can help the private and public bodies to ensure the safety among the citizens of the country.

2.1.3. Definitions and Abbreviations:

Following are the definitions for the jargoned words.

Terms Definitions

1.Face Identification:

The web camera of the hardware will identify the face of the user.

2. Mask Identifcation:

The web camera of the hardware will identify the mask worn by the user

3. Accuracy of mask worn:

It will measure the accuracy of the mask work in percentage.

4. Alert box in Mask Identification:

It will change its colour if the mask is not worn or vice versa as an alert.

5. Distance Measuring between two faces:

This feature is a part of Social Distancing feature and it will measure the distance between the faces identified by the web camera of the user.

6.Warning to maintain distance.

It will show a warning alert if the user is not maintaining the social distancing norm.

2.1.4. References: This web application has been prepared on the basis
of discussion with Team members, faculty members and also taken information from
following books &amp; website.
2.1.4.1. Websites: www.google.com , www.wikipedia.org

Page No:
2.1.4.2. Books: Fundamental of Software Engineering By Rajiv Mall. , Software
Engineering : A practitioner's approach Ed. By Pressman, Roger. Software
Engineering Seventh Edition Ian Sommerville.
Software Engineering Ed.2 by Jalota &amp; Pankaj. Schaum's Series, "Software
Engineering". Software Engineering by KK Aggarwal
2.1.5 Overview: The rest of this SRS document describes the various system requirements,
interfaces
and functionalities in detail.

2.2) Overall Description:

It uses tkinter module for the GUI. Talking about the application, the user can identify its face with an accuracy to wear the mask and be protected from the covid-19 and air pollution.This has also helped to maintain social distancing norm by staying away at a distance in our application to stay protected and healthy. Tkinter has integrated the python code to work as an application to make it more user friendly and helpful to be protected from Covid-19.

This application can be used to help the people from not being affected by corona virus by recognising the face and checking the mask accuracy with minimum distance required between two people to stay protected and healthy.

2.2.1. Product Perspective:

The proposed system shall be developed using client/server and be compatible with Microsoft Windows Operating System.

2.2.1.1 System Interfaces

None

2.2.1.2 User Interfaces

The Covid-19 Suraksha Kavach will have following user friendly and menu driven interfaces:

1.Face Identification:

The web camera of the hardware will identify the face of the user.

2. Mask Identifcation:

The web camera of the hardware will identify the mask worn by the user

3. Accuracy of mask worn:

It will measure the accuracy of the mask work in percentage.

4. Alert box in Mask Identification:

It will change its colour if the mask is not worn or vice versa as an alert.

5. Distance Measuring between two faces:

This feature is a part of Social Distancing feature and it will measure the distance between the faces identified by the web camera of the user.

6.Warning to maintain distance.

It will show a warning alert if the user is not maintaining the social distancing norm.

Page No:

2.2.1.3 Hardware Interfaces

a. Support for printer(dot matrix,deskjet,laserjet).

b. Computer systems will be in the networking environment as it is a multi user system.

2.2.1.4 Software Interfaces

a. Any window-based operating system.

b. Platform: Python

c. Environment: Spyder

2.2.1.5 Communication Interfaces

None

2.2.1.6 Memory Constraints

At least 4 GB RAM and 2 GB hard disk will be required for running the application.

2.2.1.7 Operations

None

2.2.2 Product Functions

The system will allow access only to authorised users with specific roles. Depending upon the user's role, he/she will be able to access only specific modules of the system.

Major functions that the software will perform:

1.Face Identification:

The web camera of the hardware will identify the face of the user.

2. Mask Identifcation:

The web camera of the hardware will identify the mask worn by the user

3. Accuracy of mask worn:

It will measure the accuracy of the mask work in percentage.

4. Alert box in Mask Identification:

It will change its colour if the mask is not worn or vice versa as an alert.

5. Distance Measuring between two faces:
This feature is a part of Social Distancing feature and it will measure the distance between the faces identified by the web camera of the user.

6.Warning to maintain distance.
It will show a warning alert if the user is not maintaining the social distancing norm.

2.2.3 User Characteristics

a. Educational level: At least graduate should be comfortable with English language.
b. Experience: The major function such as login can be done by authorised user.

Page No:
c. Technical Expertise: Should be comfortable using general-purpose applications on a computer.
2.2.4 Constraints
Users at Covid-19 Suraksha Kavach will have to implement a security policy to safeguard banking operations such as login information from being modified by unauthorised users.
2.2.5 Assumptions and Dependencies

a. This project is Tkinter based project where user can perform all operations of banking.
b. It will not affect the environment at all.
c. As this project is tkinter based so it will be fast and accurate.

2.3 Specific Requirements

2.3 .1 External Interface Requirements
2.3.1.1 User Interfaces

This will be the first screen that will be displayed. It will allow user to access different screens based on role of the user.

1.Face Identification:
The web camera of the hardware will identify the face of the user.

2. Mask Identifcation:
The web camera of the hardware will identify the mask worn by the user

3. Accuracy of mask worn:
It will measure the accuracy of the mask work in percentage.

4. Alert box in Mask Identification:
It will change its colour if the mask is not worn or vice versa as an alert.

5. Distance Measuring between two faces:
This feature is a part of Social Distancing feature and it will measure the distance between the faces identified by the web camera of the user.

6.Warning to maintain distance.
It will show a warning alert if the user is not maintaining the social distancing norm.

2.3.1.2 Hardware Interface
a. Support for printer(dot matrix,deskjet,laserjet).
b. Computer systems will be in the networking environment as it is a multi user system.

2.3.1.3 Software Interfaces
a. Any window-based operating system.

b. Platform: Python

c. Environment: Spyder

2.3.1.4 Communication Interfaces

None

2.3.2 Functional Requirements

Following are the services which this system will provide. These are the facilities and functions required by user:

1.Face Identification:

The web camera of the hardware will identify the face of the user.

2. Mask Identifcation:

The web camera of the hardware will identify the mask worn by the user

3. Accuracy of mask worn:

It will measure the accuracy of the mask work in percentage.

4. Alert box in Mask Identification:

It will change its colour if the mask is not worn or vice versa as an alert.

5. Distance Measuring between two faces:

This feature is a part of Social Distancing feature and it will measure the distance between the faces identified by the web camera of the user.

6.Warning to maintain distance.

It will show a warning alert if the user is not maintaining the social distancing norm.

3. Flowcharts

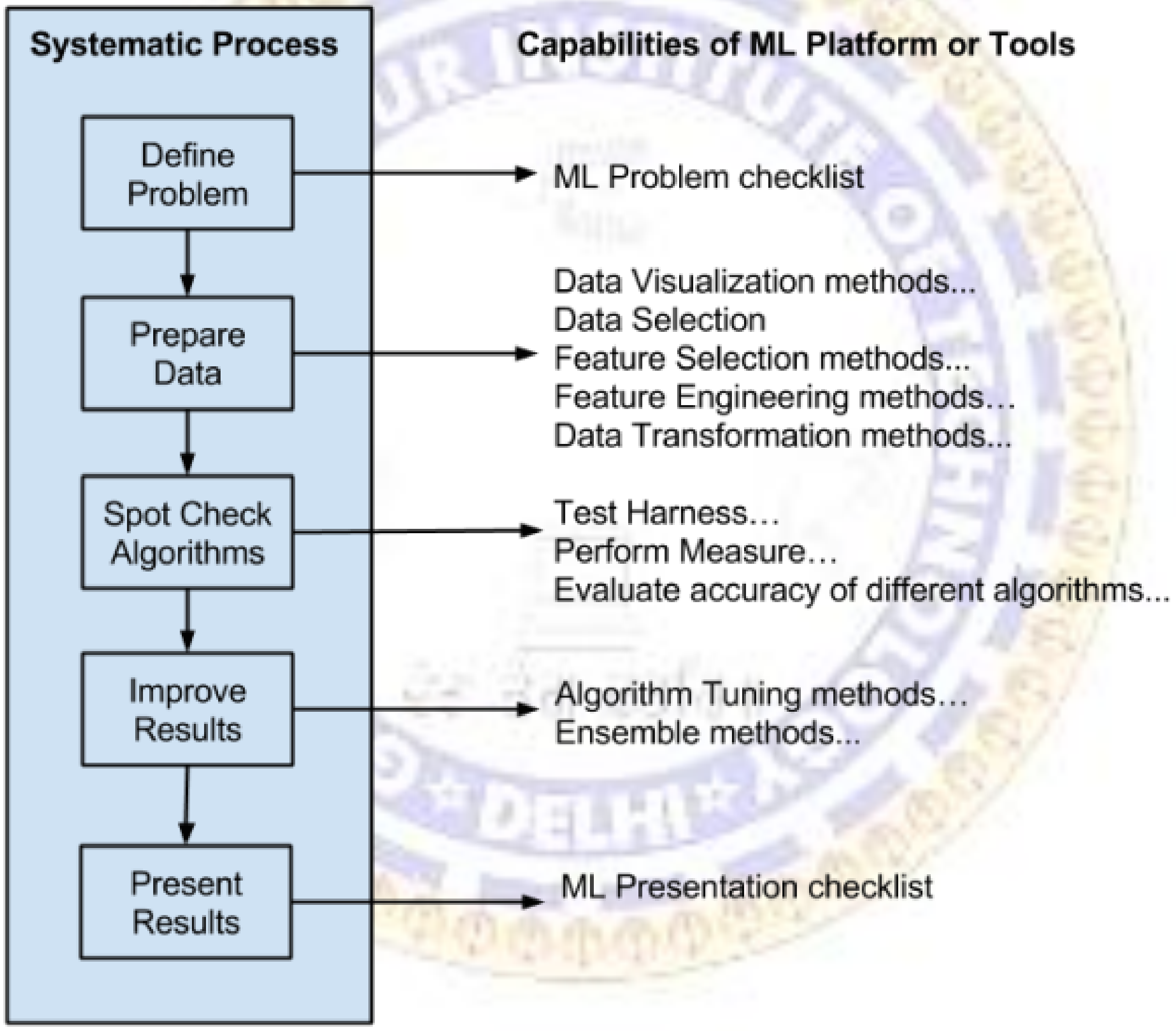

Systematic Process
Capabilities of ML Platform or Tools
Define Problem
Prepare Data
Spot Check Algorithms
Improve Results
Present Results
ML Problem checklist
Data Visualization methods...
Data Selection
Feature Selection methods...
Feature Engineering methods…
Data Transformation methods...
Test Harness…
Perform Measure…
Evaluate accuracy of different algorithms...
Algorithm Tuning methods…
Ensemble methods...
ML Presentation checklist

Fig 3.1: Machine Learning

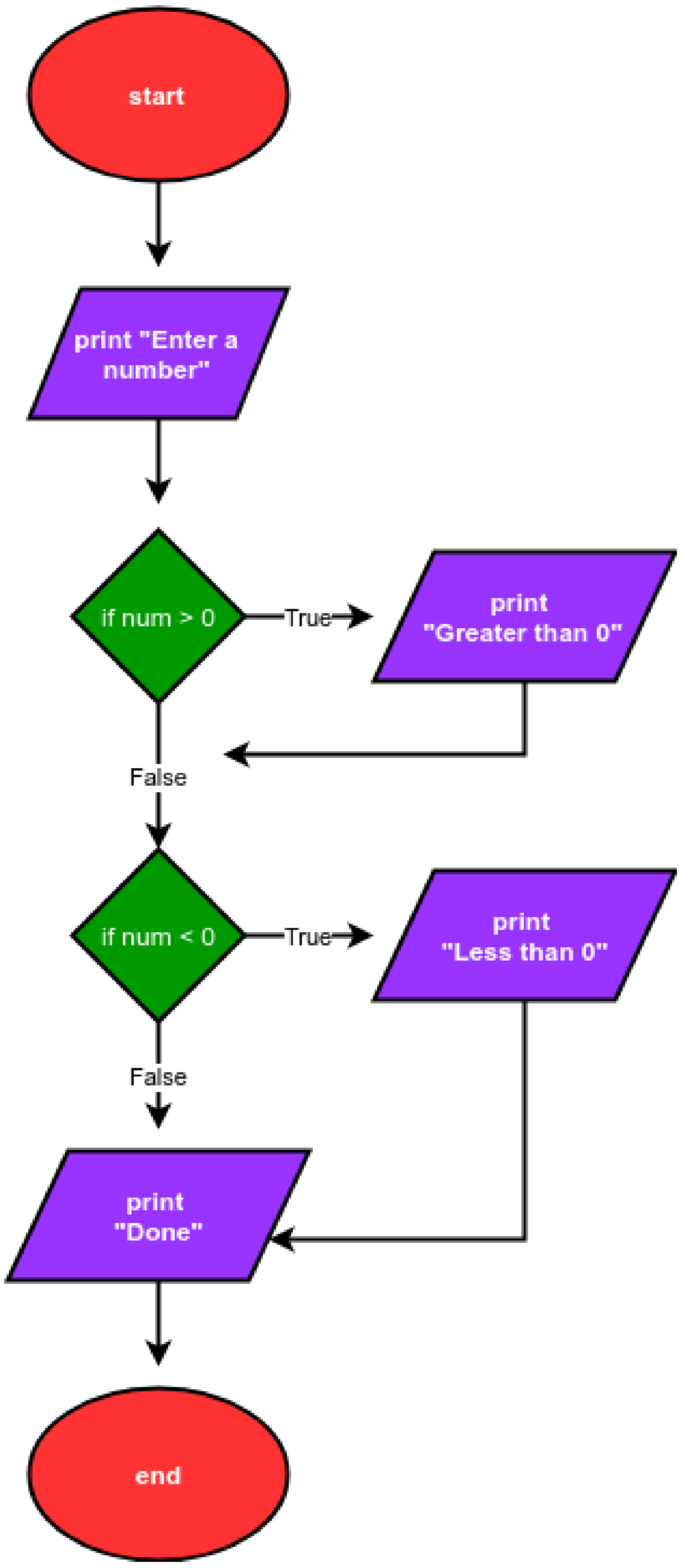

start
print "Enter a number"
if num > 0
True
print "Greater than 0"
False
if num < 0
True
print "Less than 0"
False
print "Done"
end

Fig 3.2. Numpy

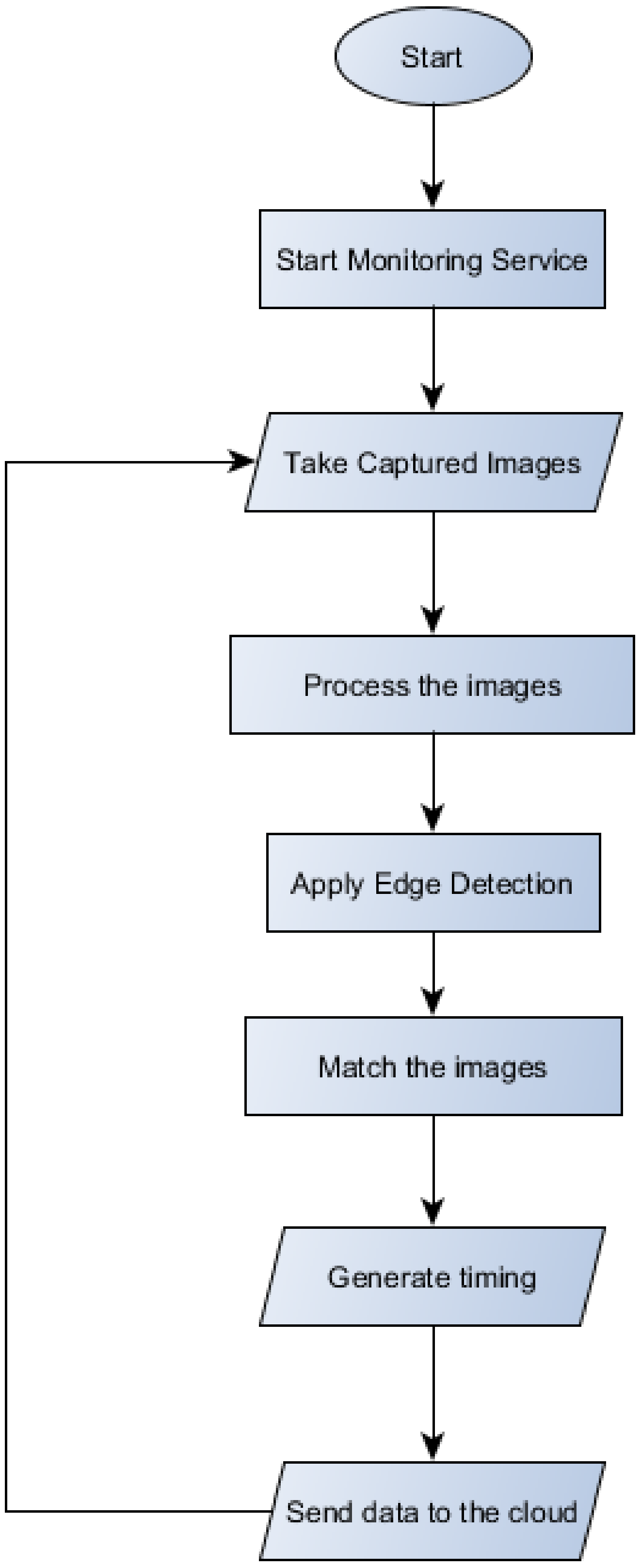


Fig 3.3. OpenCV

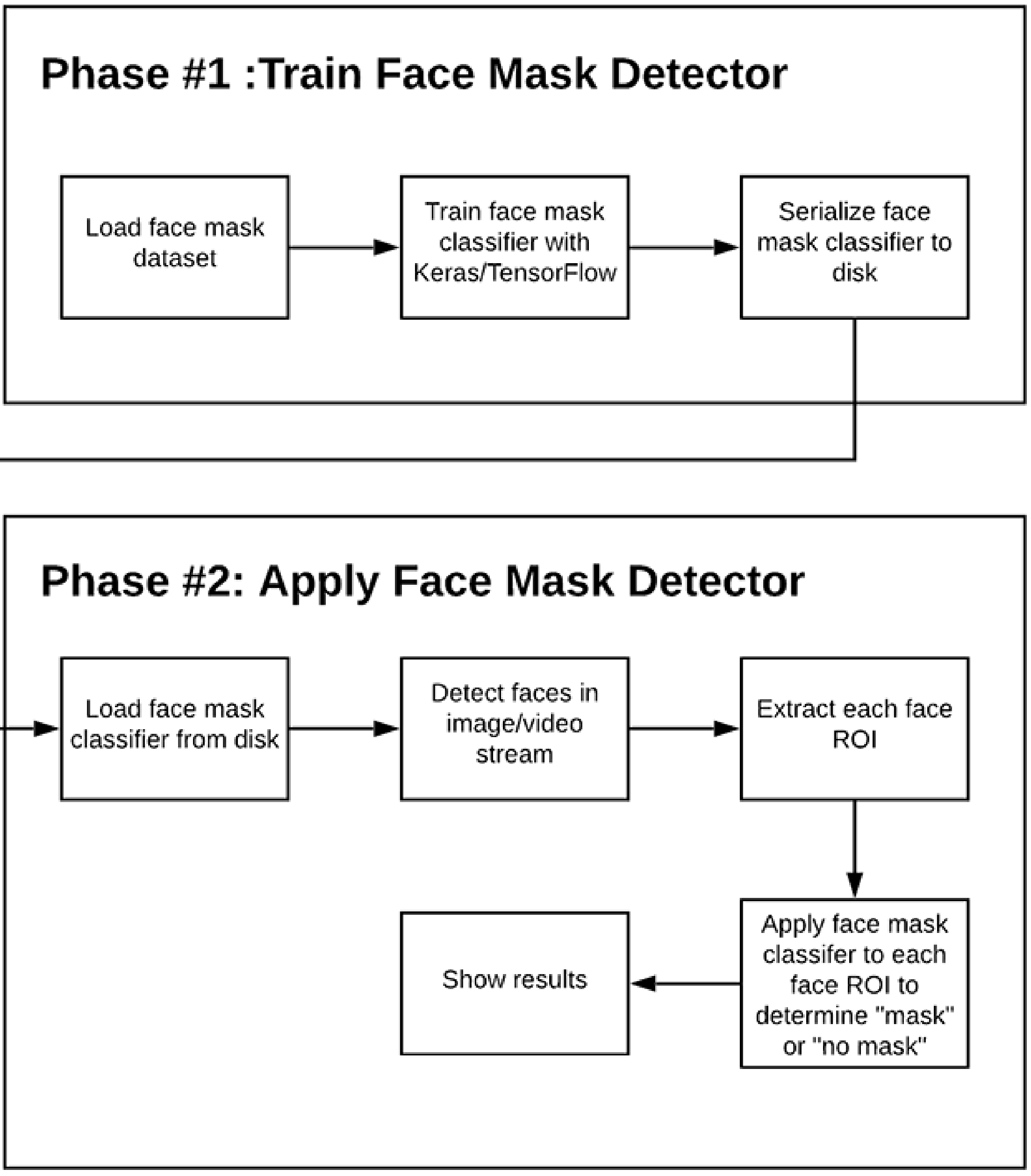

Phase #1 :Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifer to each face ROI to determine "mask" or "no mask"
Show results

Fig 3.4. Two Phase Face Mask Detection

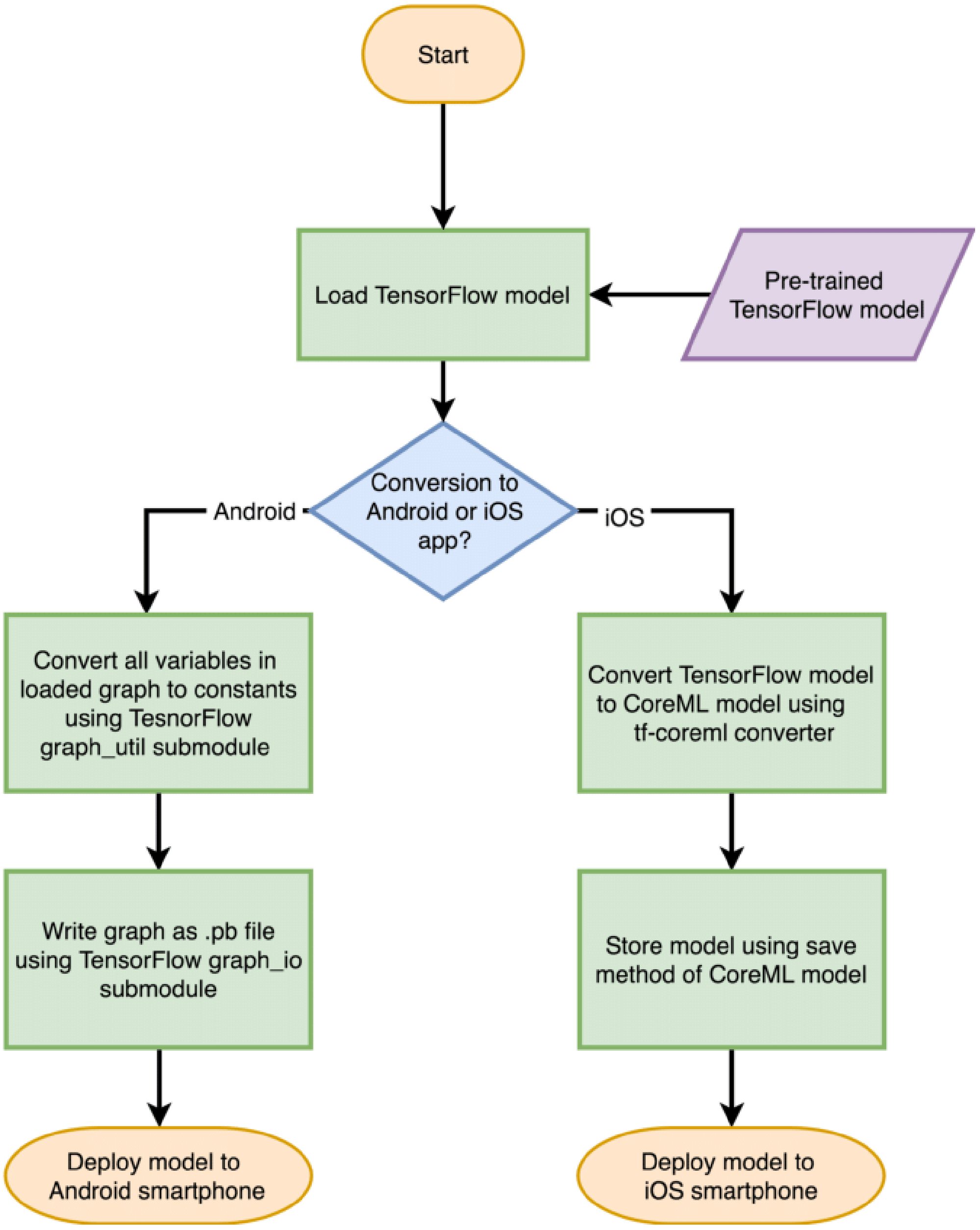

Start
Load TensorFlow model
Pre-trained TensorFlow model
Conversion to Android or iOS app?
Android
iOS
Convert all variables in loaded graph to constants using TesnorFlow graph_util submodule
Convert TensorFlow model to CoreML model using tf-coreml converter
Write graph as .pb file using TensorFlow graph_io submodule
Store model using save method of CoreML model
Deploy model to Android smartphone
Deploy model to iOS smartphone

Fig 3.5: Tensorflow

4.System Analysis

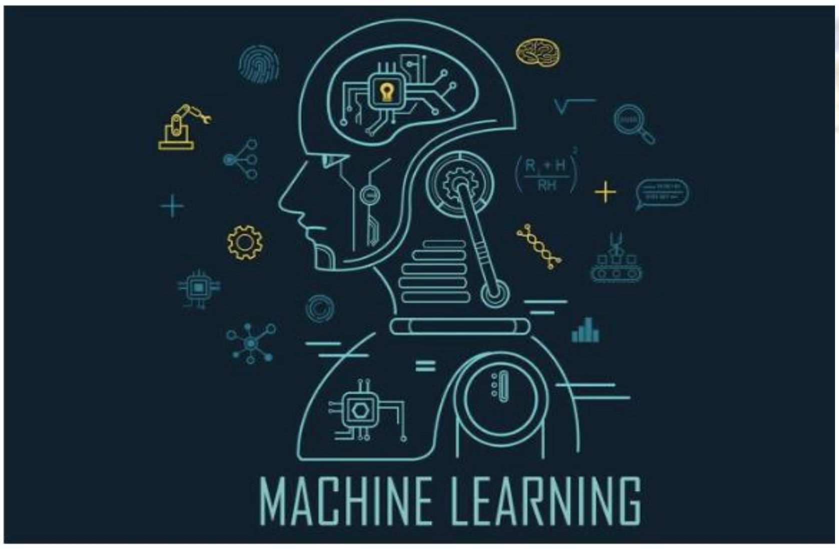
The corona virus COVID-19 pandemic is causing a global health crisis so the effective protection methods is wearing a face mask in public areas according to the World Health Organization (WHO). The COVID-19 pandemic forced governments across the world to impose lockdowns to prevent virus transmissions. Reports indicate that wearing face masks while at work clearly reduces the risk of transmission. An efficient and economic approach of using AI to create a safe environment in a manufacturing setup. A hybrid model using deep and classical machine learning for face mask detection will be presented. A face mask detection dataset consists of with mask and without mask images ,
We will use the dataset to build a COVID-19 face mask detector with computer vision using Python, OpenCV, and Tensor Flow and Keras.
Key Words: Deep Learning, Computer Vision, OpenCV, Tensorflow, Keras.

The trend of wearing face masks in public is rising due to the COVID- 19 corona virus epidemic all over the world. Before Covid-19, People used to wear masks to protect their health from air pollution. While other people are self-conscious about their looks, they hide their emotions from the public by hiding their faces. Scientists proofed that wearing face masks works on impeding COVID-19 transmission. COVID- 19 (known as corona virus) is the latest epidemic virus that hit the human health in the last century. In 2020, the rapid spreading of COVID-19 has forced the World Health Organization to declare COVID- 19 as a global pandemic.
More than five million cases were infected by COVID-19 in less than 6 months across 188 countries. The virus spreads through close contact and in crowded and overcrowded areas.
The corona virus epidemic has given rise to an extraordinary degree of worldwide scientific cooperation. Artificial Intelligence (AI) based on Machine learning and Deep Learning can help to fight Covid-19 in many ways. Machine learning allows researchers and clinicians evaluate vast quantities of data to forecast the distribution of COVID-19, to serve as an early warning mechanism for potential pandemics, and to classify vulnerable populations.

The provision of healthcare needs funding for emerging technology such as artificial intelligence, IoT, big data and machine learning to tackle and predict new diseases. In order to better understand infection rates and to trace and quickly detect infections, the AI’s power is being exploited to address the Covid-19 pandemic. People are forced by laws to wear face masks in public in many countries. These rules and laws were developed as an action to the exponential growth in cases and deaths in many areas. However, the process of monitoring large groups of people is becoming more difficult. The monitoring process involves the detection of anyone who is not wearing a face mask. Here we introduce a mask face detection model that is based on computer vision and deep learning. The proposed model can be integrated with surveillance cameras to impede the COVID-19 transmission by allowing the detection of people who are wearing masks not wearing face masks. The model is integration between deep learning and classical machine learning techniques with opencv, tensor flow and keras. We have used deep transfer leering for feature extractions and combined it with three classical machine learning algorithms. We introduced a comparison between them to find the most suitable algorithm that achieved the highest accuracy and consumed the least time in the process of training and detection.

4.1 MACHINE LEARNING LIBRARIES

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



decisions, Understanding in this context means the transformation of visual images (the input of the retina) 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.

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.
Other approaches have been developed which don't fit neatly into this three-fold categorization, and sometimes more than one is used by the same machine learning system.

The scientific discipline of computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, multi- dimensional data from a 3D scanner or medical scanning device. The technological discipline of computer vision seeks to apply its theories and models to the construction of computer vision systems.

a. COMPUTER VISION

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do, Computer vision tasks include

Fig 4.2: Computer Vision

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 levels, with higher-level learned features defined in terms of lower-level features.

Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.

Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding. As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images.

The image data can take many forms, such as video sequences, views from multiple cameras, or multi- dimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.

b. DEEP LEARNING

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

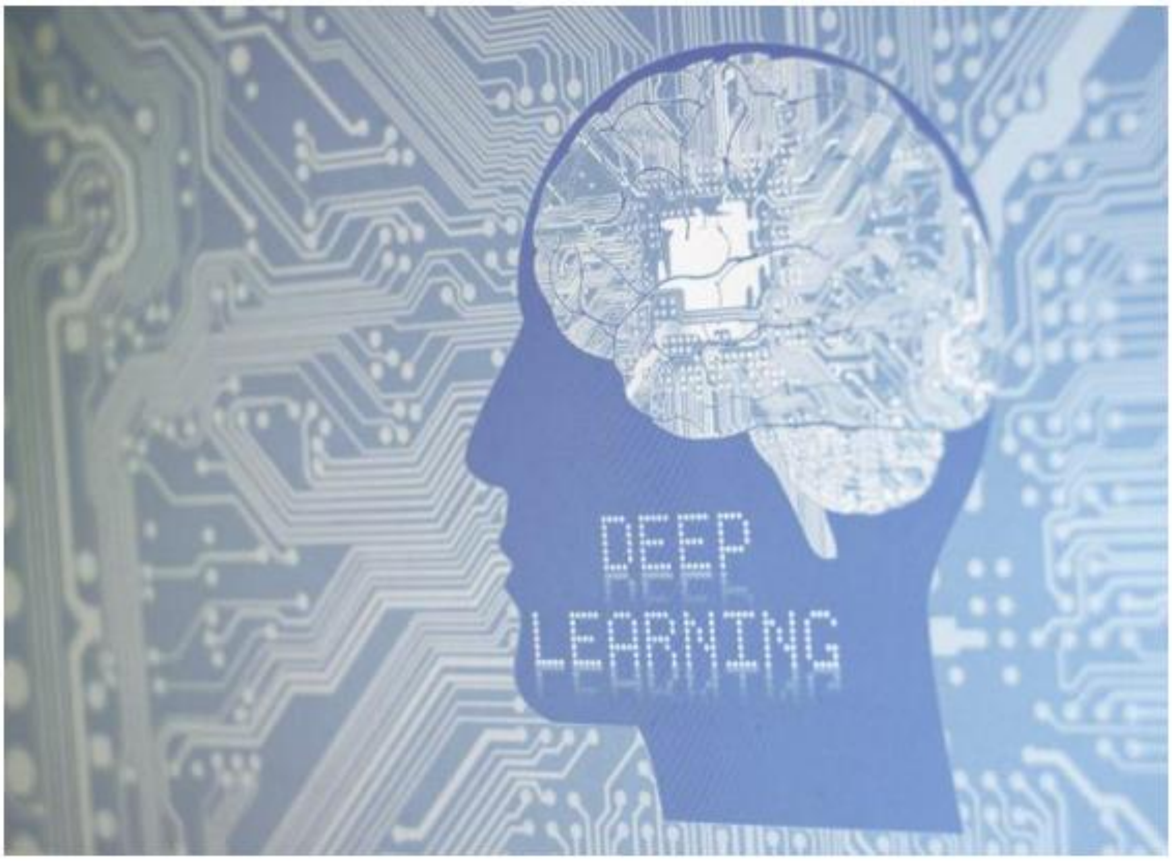


Fig 4.3: Deep learning

The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, we call this approach to AI deep learning. Deep learning excels on problem domains where the inputs (and even output) are analog. Meaning, they are not a few quantities in a tabular format but instead are images of pixel data, documents of text data or files of audio data. Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

c. 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

Fig 4.4: Features of OpenCv

using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. 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 startups such as Applied Minds, Video Surf, 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.
It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. 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 template interface that works seamlessly with STL containers.

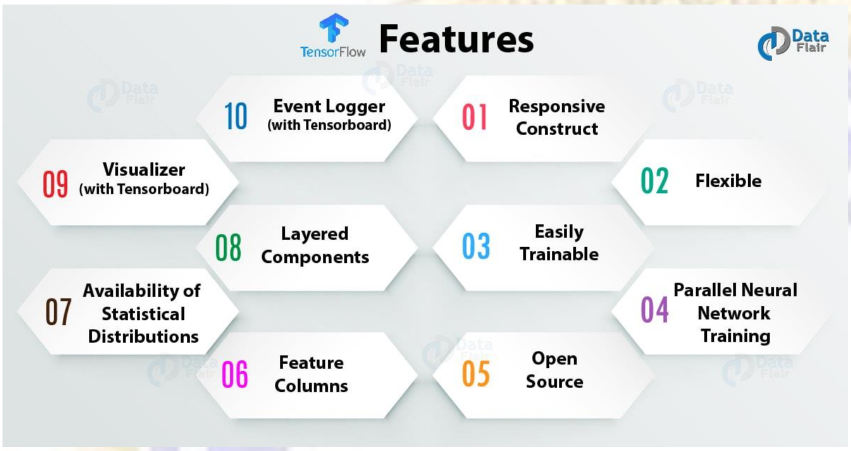
d. TENSORFLOW

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.



TensorFlow
Features
Data Flair
10 Event Logger (with Tensorboard)
01 Responsive Construct
09 Visualizer (with Tensorboard)
02 Flexible
08 Layered Components
03 Easily Trainable
07 Availability of Statistical Distributions
04 Parallel Neural Network Training
06 Feature Columns
05 Open Source

Fig 4.5: Features of Tensorflow

The name Tensor Flow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems, not least RankBrain in Google search and the fun DeepDream project.It can run on single CPU systems, GPUs as well as mobile devices and large scale distributed systems of hundreds of machines.

e. 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.

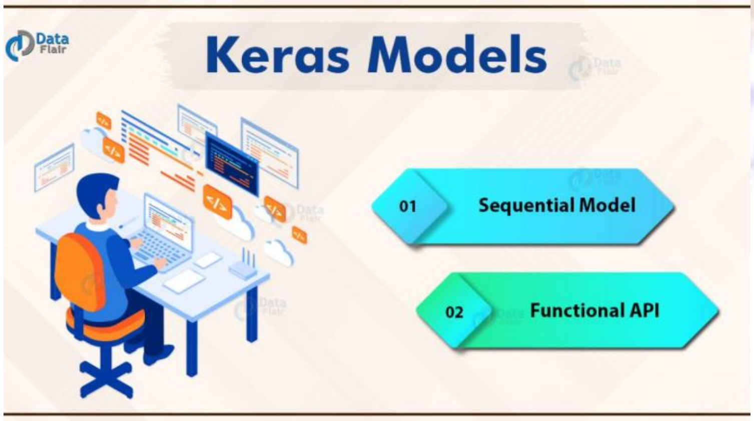


Fig 4.6: Keras Models

It was developed to make implementing deep learning models as fast and easy as possible for research and development. It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license.
Keras was developed and maintained by François Chollet, a Google engineer using four guiding principles:
also has a C++ interface. Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU).

f. PyTorch

The majority of the images were augmented by OpenCV. The
set of images were already labeled “mask” and “no mask”. The images that were present were of different sizes and resolutions, probably extracted from different sources or from machines (cameras) of different resolutions.

Data preprocessing steps as mentioned below was applied to all the raw input images to convert them into clean versions, which could be fed to a neural network machine learning model.

PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Face book’s AI Research lab (FAIR).
It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface. Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU). • Deep neural networks built on a tape-based automatic differentiation system PyTorch defines a class called Tensor (torch.Tensor) to store and operate on homogeneous multidimensional rectangular arrays of numbers. PyTorch Tensors are similar to NumPy Arrays, but can also be operated on a CUDA-capable Nvidia GPU. PyTorch supports various sub-types of Tensors.

4.2 Proposed System

The proposed system focuses on how to identify the person on image/video stream wearing face mask with the help of computer vision and deep learning algorithm by using the OpenCV, Tensor flow, Keras and PyTorch library.

Face Mask Detector using Deep Learning (PyTorch) and Computer Vision (OpenCV)

Overview

The World Health Organization (WHO) reports suggest that the two main routes of transmission of the COVID-19 virus are **respiratory droplets** and **physical contact**.

Respiratory droplets are generated when an infected person coughs or sneezes. Any person in close contact (within 1 m) with someone who has respiratory symptoms (coughing, sneezing) is at risk of being exposed to potentially infective respiratory droplets.

Droplets may also land on surfaces where the virus could remain viable; thus, the immediate environment of an infected individual can serve as a source of transmission (contact transmission).

Wearing a medical mask is one of the prevention measures that can limit the spread of certain respiratory viral diseases, including COVID-19.

In this study, medical masks are defined as surgical or procedure masks that are flat or pleated (some are shaped like cups); they are affixed to the head with straps. They are tested for balanced high filtration, adequate breathability and optionally, fluid penetration resistance.

The study analyses a set of video streams/images to identify people who are compliant with the government rule of wearing medical masks. This could help the government to take appropriate action against people who are non-compliant.

Objective

To identify the person on image/video stream wearing face mask with the help of computer vision and deep learning algorithm by using the PyTorch library.

Approach

1.Train Deep learning model (MobileNetV2)

2.Apply mask detector over images / live video stream

Flowchart

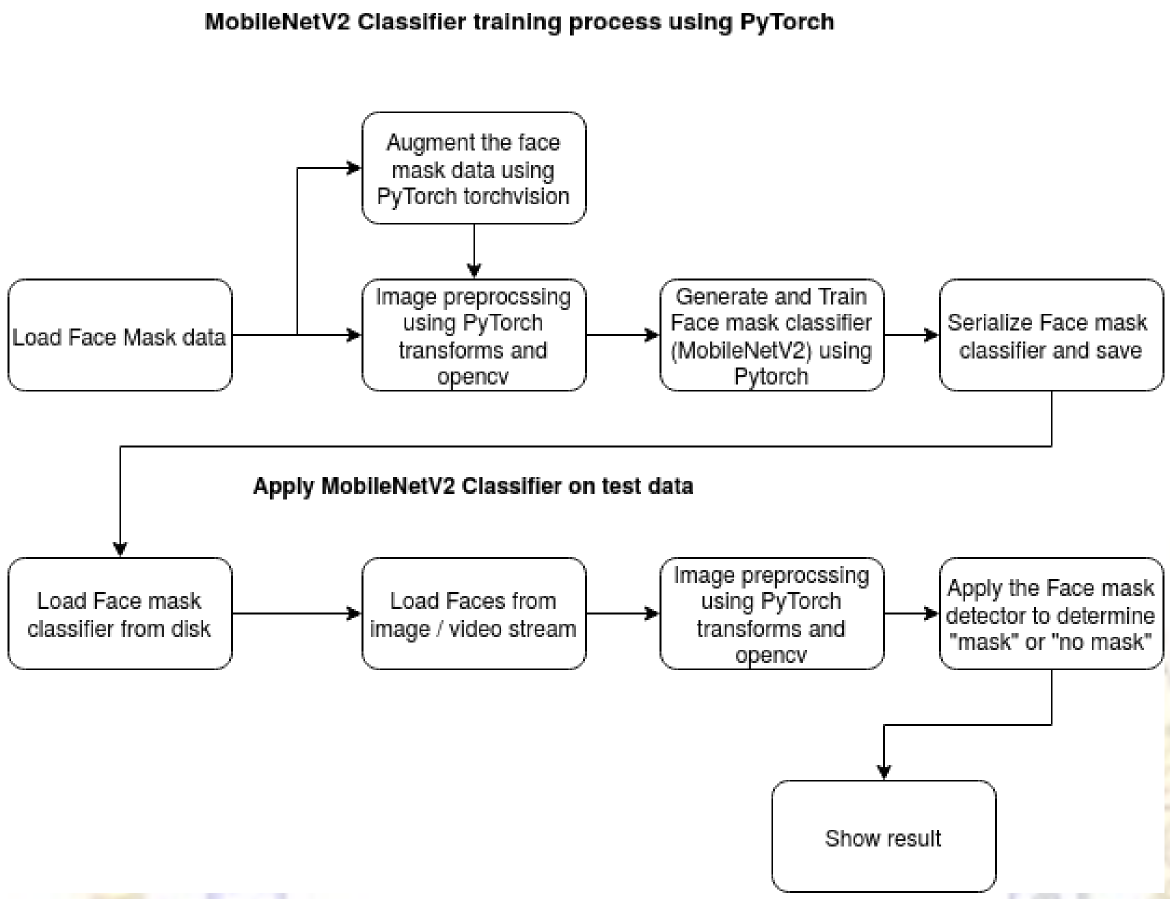

MobileNetV2 Classifier training process using PyTorch
Load Face Mask data
Augment the face mask data using PyTorch torchvision
Image preprocssing using PyTorch transforms and opencv
Generate and Train Face mask classifier (MobileNetV2) using Pytorch
Serialize Face mask classifier and save
Apply MobileNetV2 Classifier on test data
Load Face mask classifier from disk
Load Faces from image / video stream
Image preprocssing using PyTorch transforms and opencv
Apply the Face mask detector to determine "mask" or "no mask"
Show result

Fig 4.7: Flow diagram of MobileNetV2 Classifier

Data At Source

The raw images used for the current study were downloaded from PyImageSearch article and the majority of the images were augmented by OpenCV. The set of images were already labeled “mask” and “no mask”. The images that were present were of different sizes and resolutions, probably extracted from different sources or from machines (cameras) of different resolutions.

Data Preprocessing

Preprocessing steps as mentioned below was applied to all the raw input images to convert them into clean versions, which could be fed to a neural network machine learning model.

- 1.Resizing the input image (256 x 256)

- 2.Applying the color filtering (RGB) over the channels (Our model MobileNetV2 supports 2D 3 channel image)

- 3.Scaling / Normalizing images using the standard mean of PyTorch build in weights

- 4.Center cropping the image with the pixel value of 224x224x3

- 5.Finally Converting them into tensors (Similar to NumPy array)

Deep Learning Frameworks

To implement this deep learning network we have the following options.

- 1.TensorFlow

- 2.Keras

- 3.PyTorch

- 4.Caffee

- 5.MxNet

- 6.Microsoft Cognitive ToolKit

We are using the PyTorch because it runs on Python, which means that anyone with a basic understanding of Python can get started on building their deep learning models.and also it has the following advantage compared with TensorFlow 1. Data Parallelism 2. It looks like a Framework.

In PyTorch, I have used the following module to develop the algorithm,

- **PyTorch DataLoader** – which is used to load the data from the Image Folder

- **PyTorch DataSets ImageFolder** – which is used to locate the image sources and also have a predefined module to label the target variable.

- **Pytorch Transforms** – helped to apply the preprocessing steps over the source image while reading from the source folder.

-

- **PyTorch Device** – identifies the running system capabilities like CPU or GPU power to train the model. It will help us to switch the system usage.

- **Pytorch TorchVision** – it will help us to load the libraries which are created before. Like pre-trained models, image sources and so on. It is one of core in PyTorch

- **PyTorch nn** – it is one of the core modules. This module helps us to build our own Deep Neural Network (DNN) models. It has all the libraries needed to build the model. Like Linear layer, Convolution layer with 1D, conv2d, conv3d, sequence, CrossEntropy Loss (loss function), Softmax, ReLu and so on.

- **PyTorch Optim** – help us to define the model optimizer. it will help the model to learn the data well. For example Adam, SDG and so on.

- **Pytorch PIL** – helps to load the image from the source.

- **PyTorch AutoGrad** – another important module, it provides automatic differentiation for all operations on Tensors. For example, a single line of code .backward() to calculate the gradients automatically. This is very useful while implementing the backpropagation in DNN.

Image Classification Algorithm from PyTorch

4.3 MobileNetV2

CNN (Convolutional Neural Network) has many versions of pre-trained and well-architected networks for example AlexNet, ResNet, Inception, LeNet, MobileNet and so In our case I have chosen the MobileNetV2 due to its lightweight and very efficient mobile-oriented model.

MobileNetV2 builds upon the ideas from MobileNetV1, using depth wise separable convolution as efficient building blocks. However, V2 introduces two new features to the architecture:

1) Linear bottlenecks between the layers, and

2) Shortcut connections between the bottlenecks. The basic structure is shown below

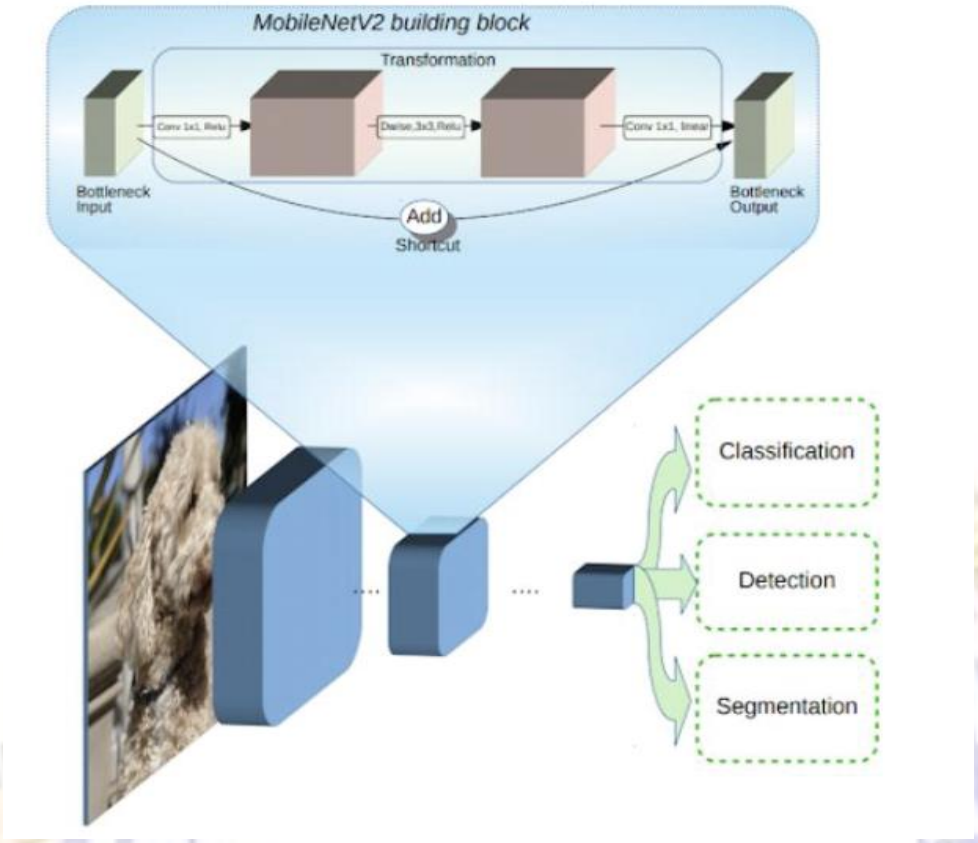


Fig 4.8: MobileNetV2 building blocks

The typical MobilenetV2 architecture has as many layers listed below, In Pytorch we can use the models library in TorchVision to create the MobileNetV2 model instead of defining/building our own model. The weights of each layer in the model are predefined based on the ImageNet dataset. The weights indicate the padding, strides, kernel size, input channels and output

channels.MobileNetV2 was chosen as an algorithm to build a model that could be deployed on a mobile device. A customized fully connected layer which contains four sequential layers on top of the MobileNetV2 model was developed.
The layers are:

1. Average Pooling layer with 7×7 weights
2. Linear layer with ReLu activation function
3. Dropout Layer
4. Linear layer with Softmax activation function with the result of 2 values.

The final layer softmax function gives the result of two
probabilities each one represents the classification of “mask” or “not mask”.

4.4 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

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:

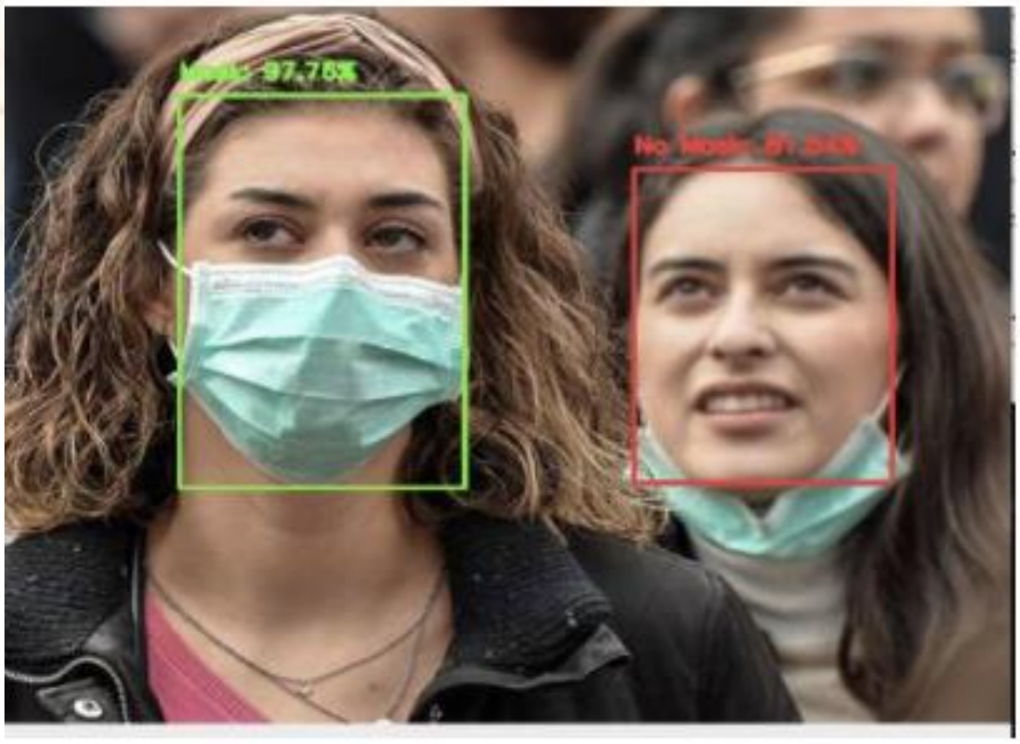


Fig 4.9: Detection of “Mask” and “No Mask”

1. **Floating-point 16 version of the original Caffe implementation.**

2.8 bit quantized version using Tensor flow

The Caffe model in this face mask detector. There has been a lot of discussion around deep learning based approaches for person detection. This encouraged us to come up with our own algorithm to solve this problem. Our work on face mask detection comprises of data collection to tackle the variance in kinds of face masks worn by the workers.

The face mask detection model is a combination of face detection model to identify the existing faces from camera feeds and then running those faces through a mask detection model.

5. MobileNetV2

MobileNetV2 builds upon the ideas from MobileNetV1 [1], using depthwise separable convolution as efficient building blocks. However, V2 introduces two new features to the architecture: 1) linear bottlenecks between the layers, and 2) shortcut connections between the bottlenecks1. The basic structure is shown below.

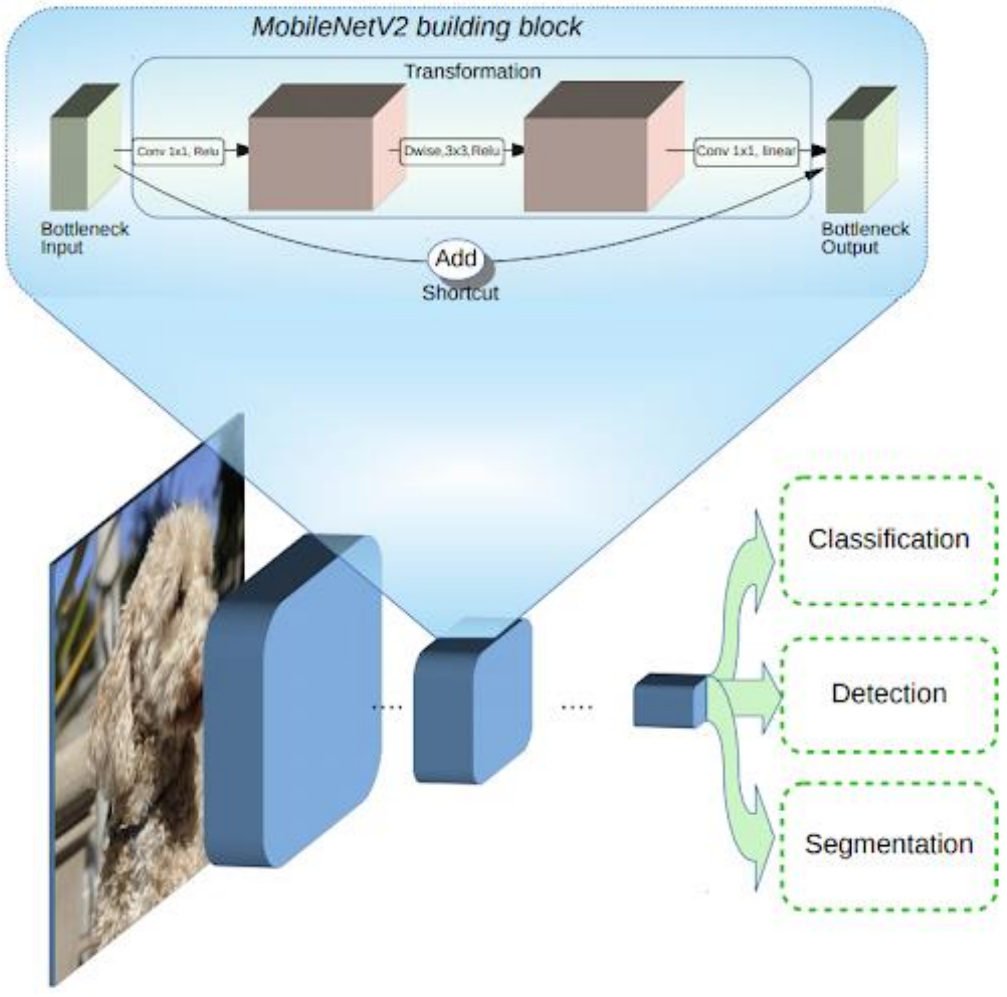
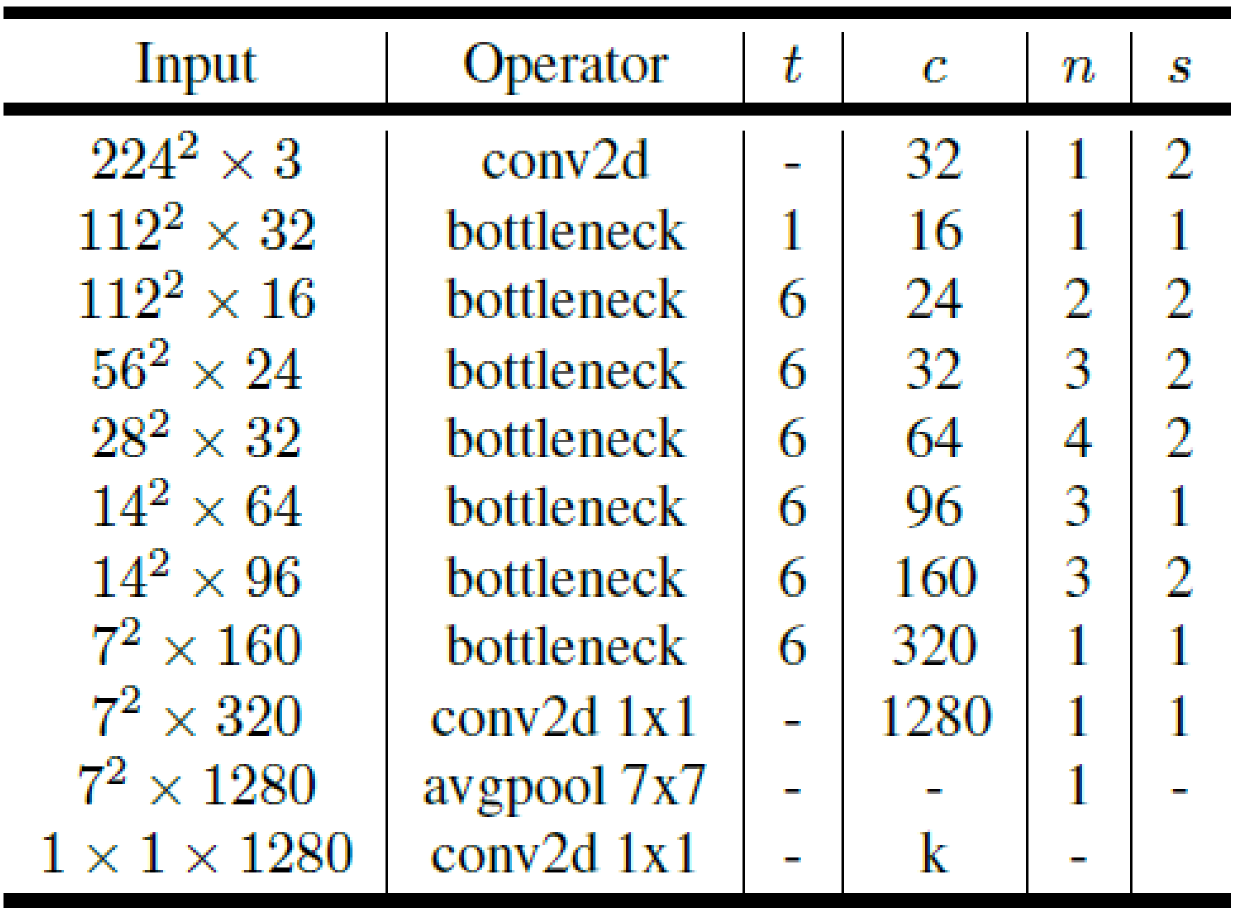


Fig 5.1: Basic Structure of MobileNetV

5.1 MobileNetV2 Architecture

The typical MobilenetV2 architecture has as many layers listed below, In Pytorch we can use the models library in TorchVision to create the MobileNetV2 model instead of defining/building our own model.

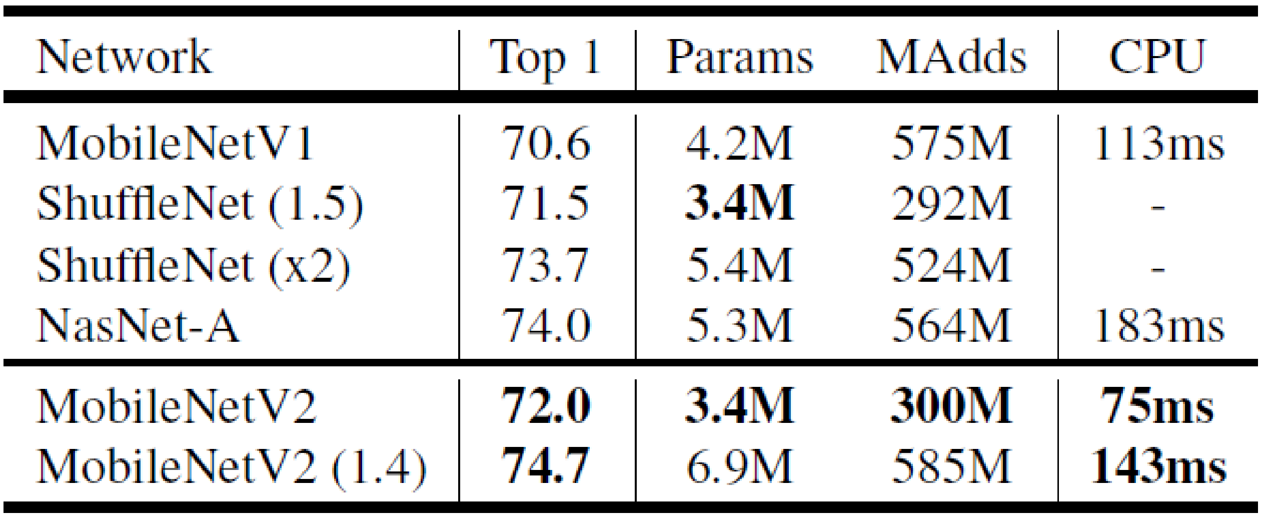
The weights of each layer in the model are predefined based on the ImageNet dataset. The weights indicate the padding, strides, kernel size, input channels and output channels.



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

5.2 Why choose MobileNetV2?

Based on ImageNet dataset MobileNetV2 outperforms MobileNetV1 and ShuffleNet (1.5) with comparable model size and computational cost. And also it will perform well for the smaller dataset.



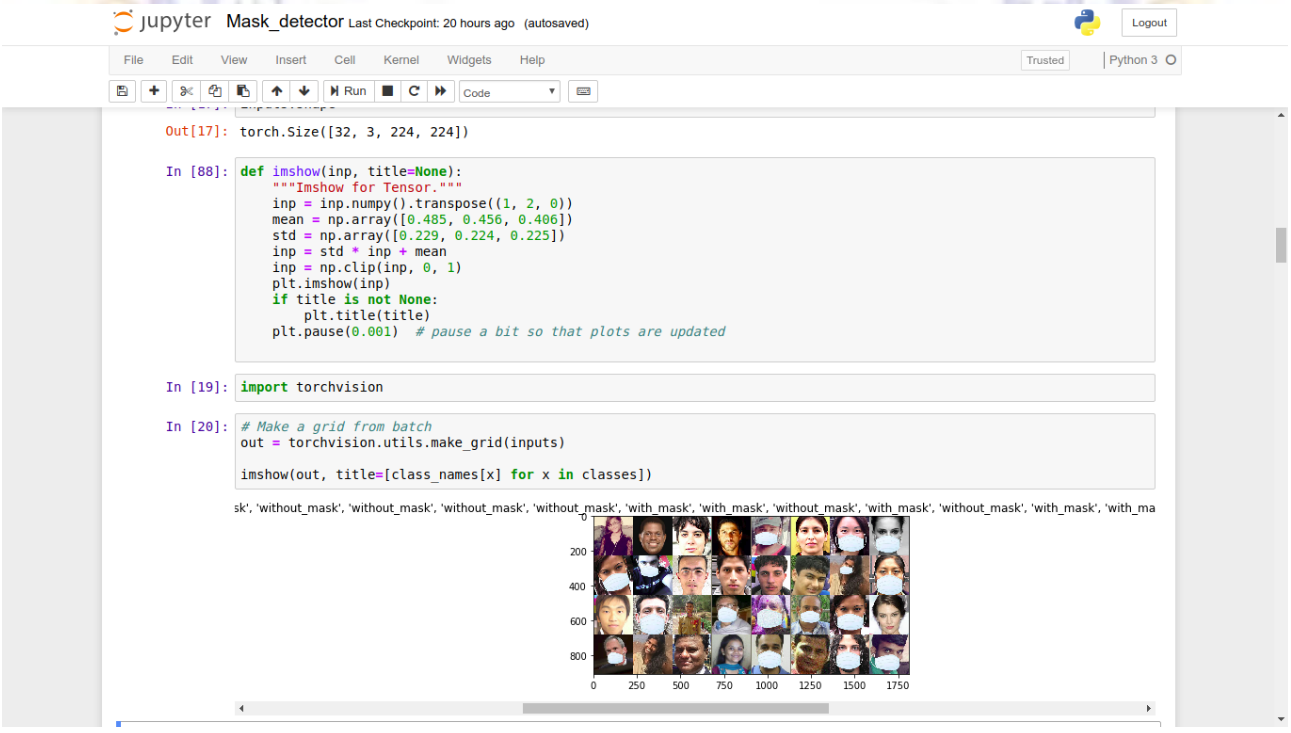
Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Face Mask Detector

Training with Face Mask

As described above I have used the augmented data which is a smaller dataset. I have also applied all the necessary preprocessing requisites applied on the top of source images using the **PyTorch transforms, PIL and DataSets**.

The sample face mask training files look like as below.



jupyter Mask_detector Last Checkpoint: 20 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
Out[17]: torch.Size([32, 3, 224, 224])

In [88]: def imshow(inp, title=None):
             """Imshow for Tensor."""
             inp = inp.numpy().transpose((1, 2, 0))
             mean = np.array([0.485, 0.456, 0.406])
             std = np.array([0.229, 0.224, 0.225])
             inp = std * inp + mean
             inp = np.clip(inp, 0, 1)
             plt.imshow(inp)
             if title is not None:
                 plt.title(title)
             plt.pause(0.001)  # pause a bit so that plots are updated

In [19]: import torchvision

In [20]: # Make a grid from batch
         out = torchvision.utils.make_grid(inputs)

         imshow(out, title=[class_names[x] for x in classes])
```

sk', 'without_mask', 'without_mask', 'without_mask', 'without_mask', 'with_mask', 'with_mask', 'without_mask', 'with_mask', 'without_mask', 'with_mask', 'with_ma

Fig 5.2: Training set example

Model Overwritten

MobileNetV2 was chosen as an algorithm to build a model that could be deployed on a mobile device. A customized fully connected layer which contains four sequential layers on top of the

MobileNetV2 model was developed.
The layers are:
1.Average Pooling layer with 7×7 weights

2.Linear layer with ReLu activation function

3.Dropout Layer

4.Linear layer with Softmax activation function with the result of 2 values.
The final layer softmax function gives the result of two probabilities each one represents the classification of “mask” or “not mask”.

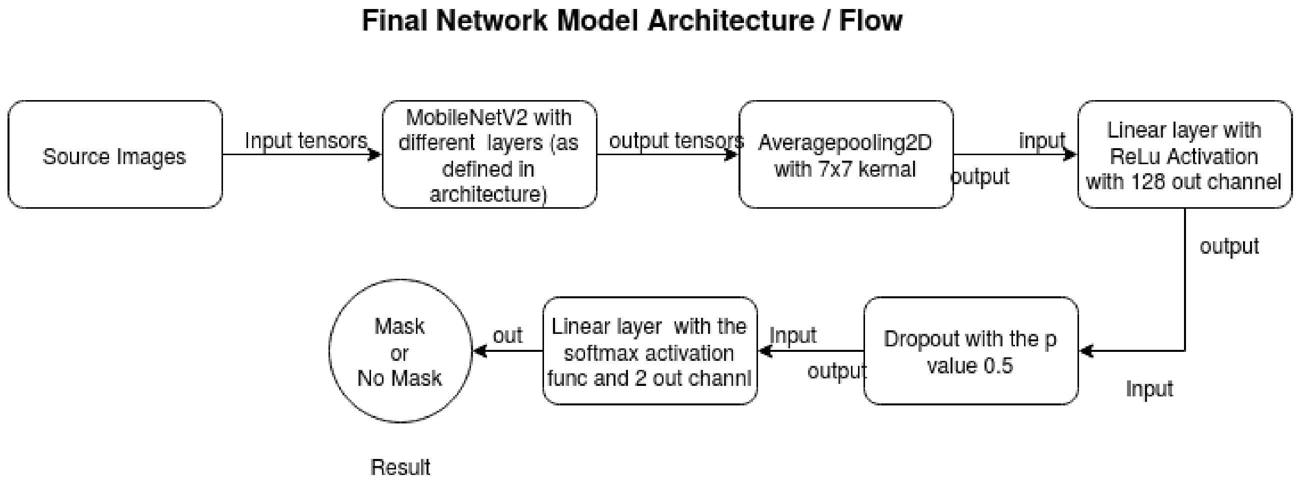

Final Network Model Architecture / Flow
Source Images
Input tensors
MobileNetV2 with different layers (as defined in architecture)
output tensors
Averagepooling2D with 7x7 kernal
output
input
Linear layer with ReLu Activation with 128 out channel
output
Input
Dropout with the p value 0.5
output
Input
Linear layer with the softmax activation func and 2 out channl
out
Mask or No Mask
Result

Fig 5.3: Accuracy Overview

The data set has been divided into two sets, likely a training and validation set. The accuracy of image classifier over the training set vs validation.

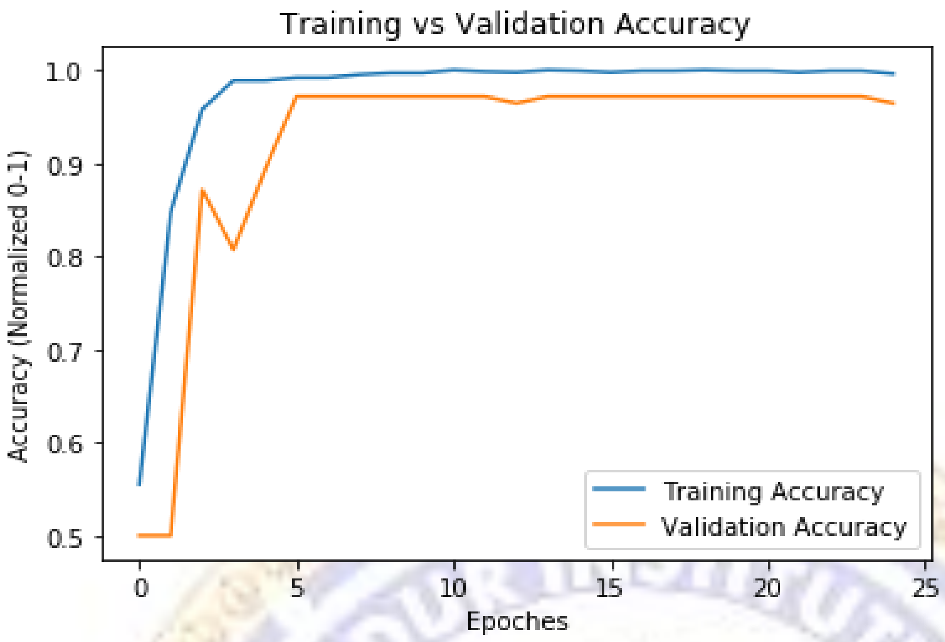
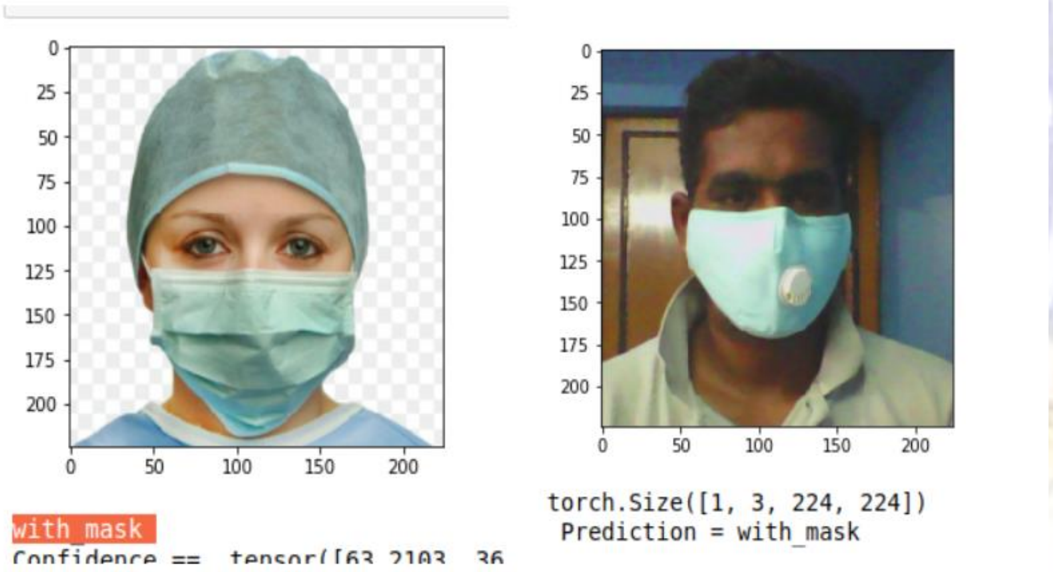


Fig 5.4: Training vs Validation Accuracy



```
with_mask
Confidence ==   tensor([63.2103,  36
```

```
torch.Size([1, 3, 224, 224])
Prediction = with_mask
```

Fig 5.5: 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

IDENTIFY THE FACE IN THE WEBCAM

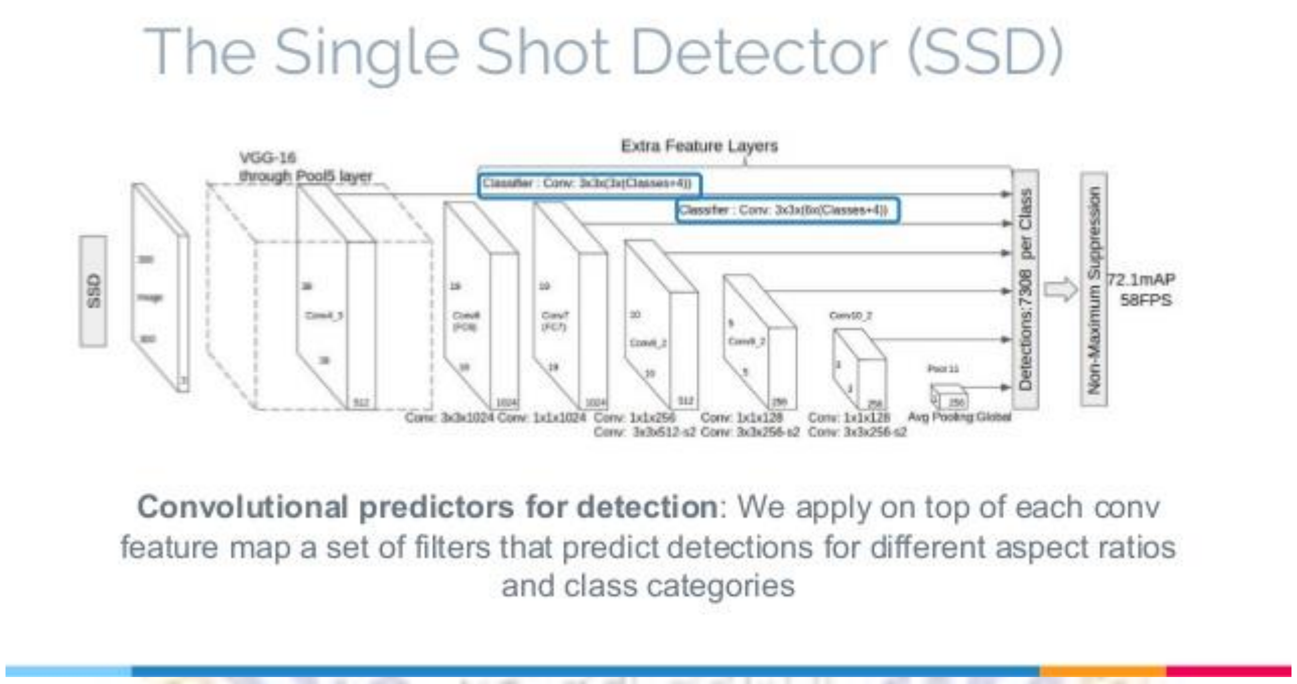
To identify the faces a pre-trained model provided by the OpenCV framework was used. The pre-trained model is based on Single-Shot-Multibox Detector (SSD) and employs a ResNet-10 Architecture as the backbone. The model was trained using web images. OpenCV provides 2 models for this face detector:

1.Floating-point 16 version of the original Caffe implementation.

2.8 bit quantized version using Tensorflow

In this example I have used the Caffe model in this face mask detector.

SSD BASIC ARCHITECTURE


SSD
VGG-16 through Pool5 layer
Extra Feature Layers
Classifier : Conv: 3x3x(3x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
Conv4_3
Conv6 (FC6)
Conv7 (FC7)
Conv8_2
Conv9_2
Conv10_2
Pool 11
Conv: 3x3x1024 Conv: 1x1x1024 Conv: 1x1x256 Conv: 3x3x512-s2 Conv: 1x1x128 Conv: 3x3x256-s2 Conv: 1x1x128 Conv: 3x3x256-s2 Avg Pooling: Global
Detections:7308 per Class
Non-Maximum Suppression
72.1mAP
58FPS

The Single Shot Detector (SSD)

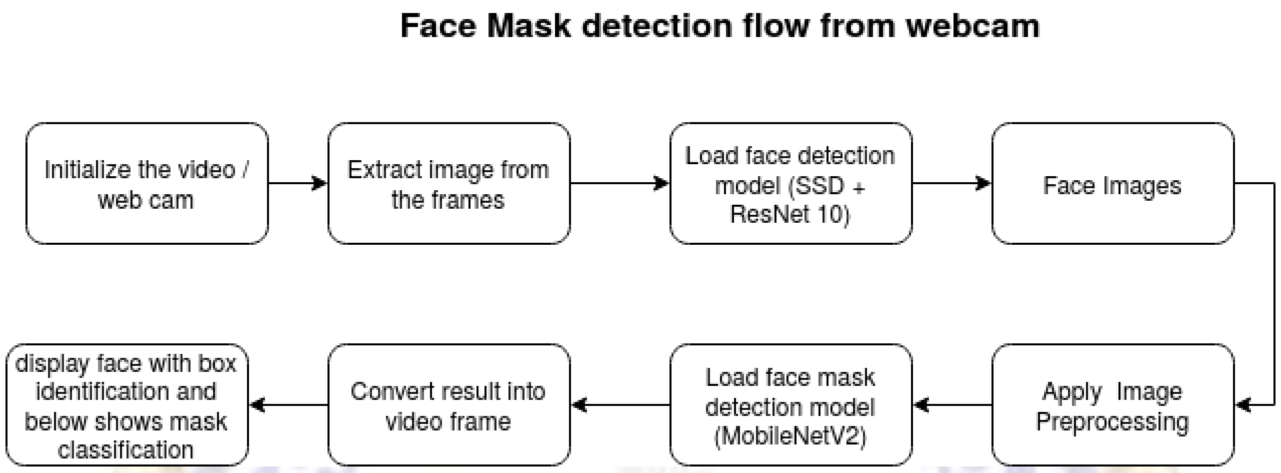
Convolutional predictors for detection: We apply on top of each conv feature map a set of filters that predict detections for different aspect ratios and class categories

Fig 5.6: Single-shot MultiBox Detector

Single-shot MultiBox Detector is a one-stage object detection algorithm. This means that, in contrast to two-stage models, SSDs do not need initial object proposals generation step. This makes it, usually, faster and more efficient. This model will return the array of faces detected on the video frame.

6. FACE MASK CLASSIFIER

Subsequent to face detection, the mask detector model was used to identify the frame face covered by mask or not. The flow of this process described in the image below.


Face Mask detection flow from webcam
Initialize the video / web cam
Extract image from the frames
Load face detection model (SSD + ResNet 10)
Face Images
Apply Image Preprocessing
Load face mask detection model (MobileNetV2)
Convert result into video frame
display face with box identification and below shows mask classification

Face Mask identification on Stream

The testing video has been attached here.

Use Cases

Here are a few use cases where this mask detection technology could be leveraged.

Airports

The Face Mask Detection System could be used at airports to detect travelers without masks. Face data of travelers can be captured in the system at the entrance. If a traveler is found to be without a face mask, their picture is sent to the airport authorities so that they could take quick action.

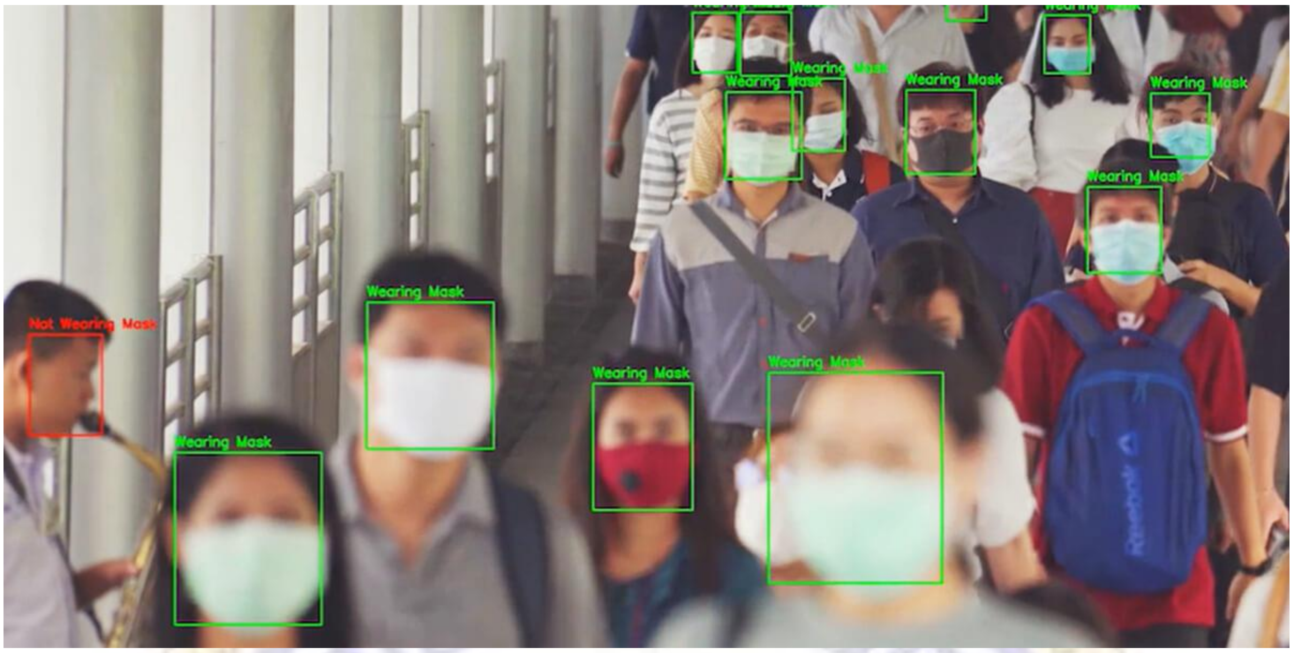


Fig 6.1: The Face Mask Detection System

Hospitals

Using Face Mask Detector System, Hospitals can monitor if quarantined people required to wear a mask are doing so or not. The same holds good for monitoring staff on duty too.

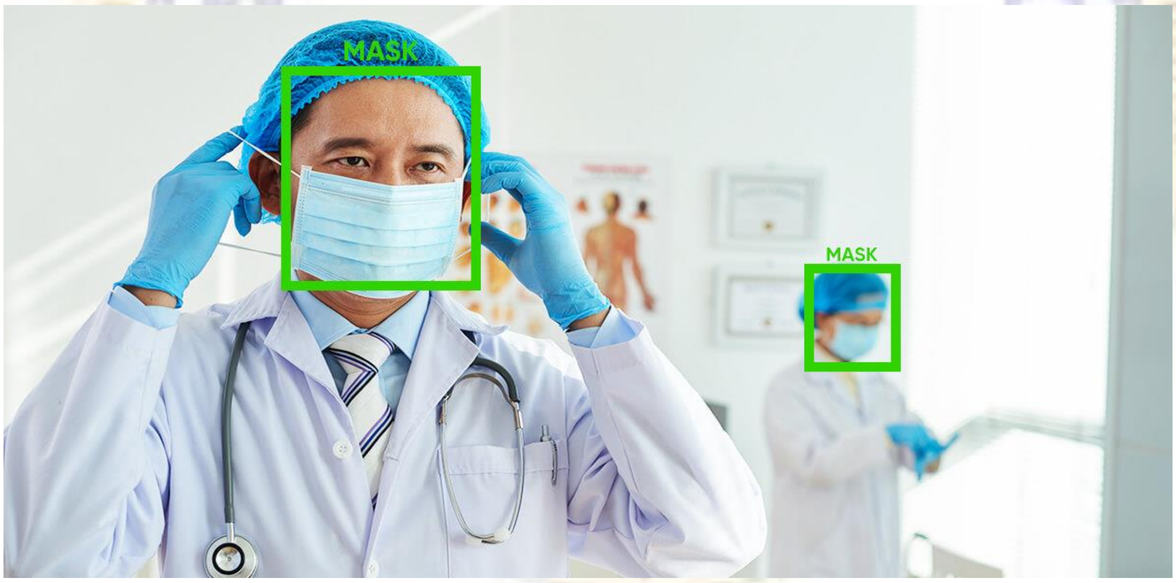


Fig 6.2: Face Mask Detector System in hospitals

Offices & Working Spaces

The Face Mask Detection System can be used at office premises to ascertain if employees are maintaining safety standards at work. It monitors employees without masks and sends them a reminder to wear a mask.

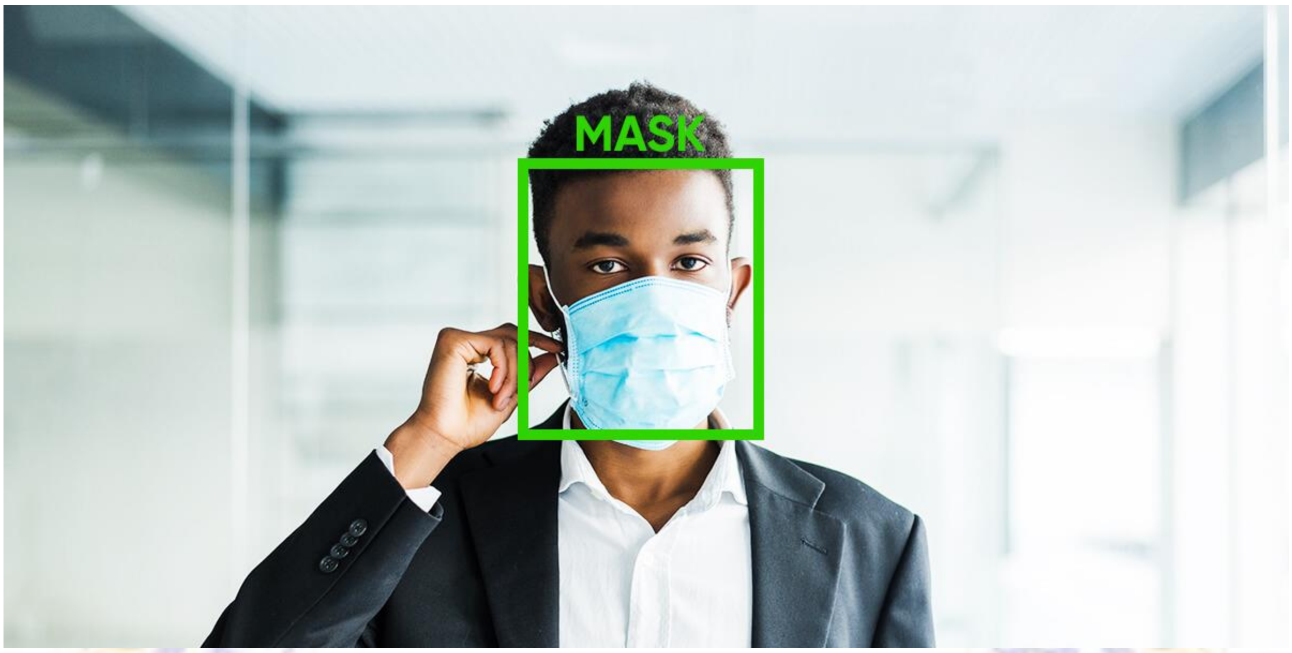


Fig 6.3: Face Mask Detection System in offices

Government

To limit the spread of coronavirus, the police could deploy the face mask detector on its fleet of surveillance cameras to enforce the compulsory wearing of face masks in public places.

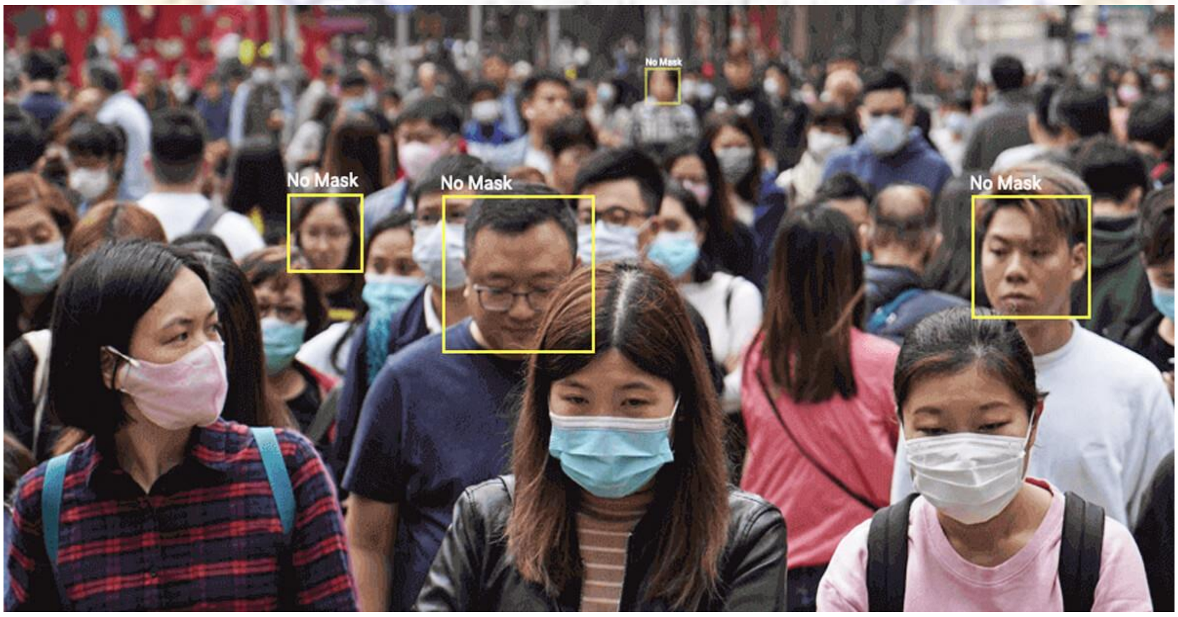


Fig 6.4: Face Mask Detection System in public place

7. SOCIAL DISTANCING USING OPENCV

Social distancing, also called **physical distancing**, is a set of non-pharmaceutical interventions or measures taken to prevent the spread of a contagious disease by maintaining a physical distance between people and reducing the number of times people come into close contact with each other. It typically involves keeping a certain distance from others (the distance specified
may differ from time to time and country to country) and avoiding gathering together in large groups.

Today's unfortunate circumstances due to COVID-19, keeping distance among people is crucial. The goal is to detect people using Haar Cascade Classifier and find the distance between people to check whether a norm social distance of 50 inch is maintained by people.

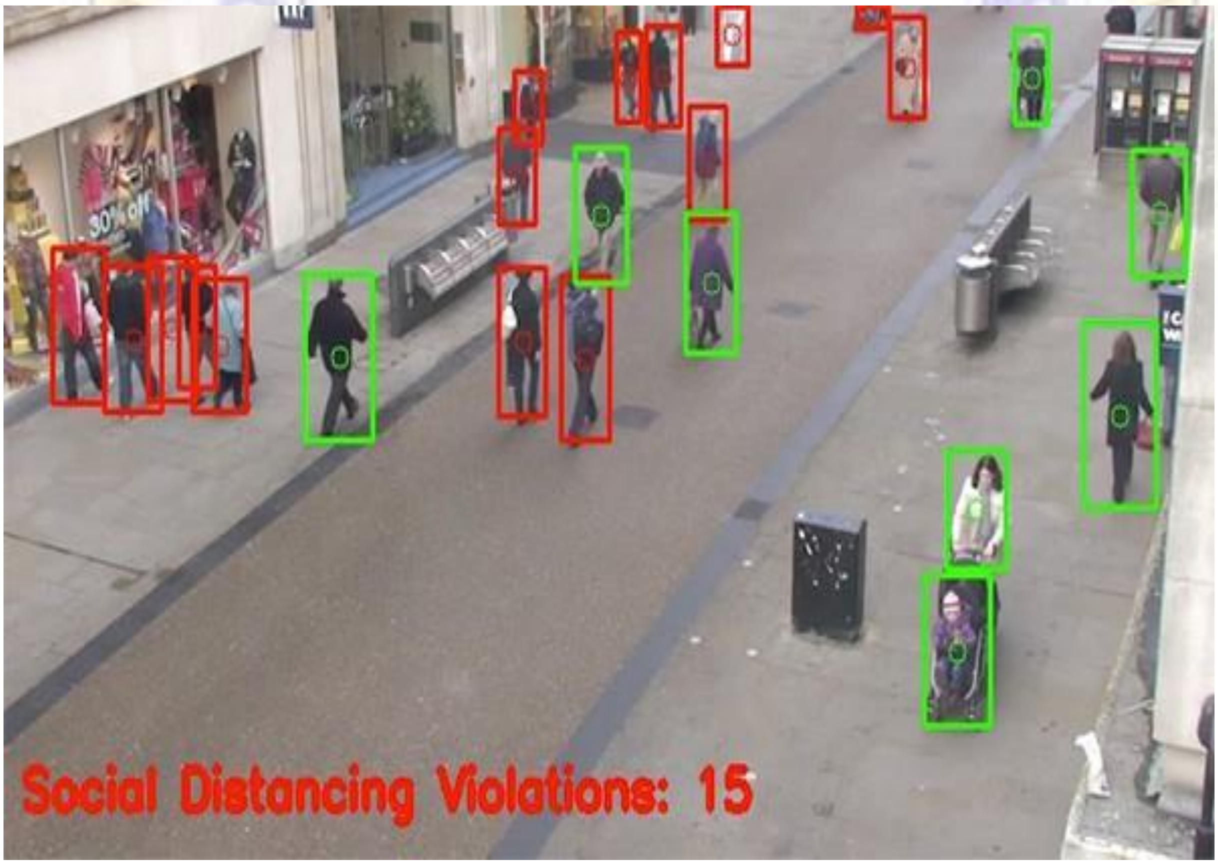


Fig 7.1: Social distancing violations

7.1 TOOLS AND DEPENDENCIES

· Python

· OpenCV

· NumPy

· Math

DESCRIPTION

Step 1: Find the number of people in the frame/Image and creating a bounding box over the people. Full body detection using Haar cascades is a machine learning based approach where a cascade function is trained with a set of input data. OpenCV already contains many pre-trained classifiers for face, eyes, smiles, etc. Today we will be using the full body classifier.
The classifier can be installed using this link-

```
import cv2
import numpy as np
import math
body_cascade = cv2.CascadeClassifier('/home/mini/Downloads/haarcascade_fullbody.xml')
cap=cv2.VideoCapture(0)
while (1):
   ret,img = cap.read()
   img = cv2.flip(img,1)
   gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
   body = body_cascade.detectMultiScale(gray,1.3,5)
   pts = []
   p =[]
   for(x,y,w,h) in body:
      cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
      cv2.circle(img,(int((2*x+w)/2),int((2*y+h)/2)),5,(255,0,0),5)
      pts.append(int(2*x+w)/2)
      p.append(int(2*y+h)/2)
```

A few things to note:

· The detection works only on grayscale images. So, it is important to convert the colour image to grayscale.

· **detectMultiScale** function is used to detect the faces. It takes 3 arguments — the input image, *scaleFactor* and *minNeighbours*. *scaleFactor* specifies how much the image size is reduced with each scale. *minNeighbours* specifies how many neighbours each candidate rectangle should have to retain it.

· *faces* contain a list of coordinates for the rectangular regions where faces were found. We use these coordinates to draw the rectangles in our image.

Step 2: Find the pixel distance between the centre point of one person and the other and then mapping the pixel distance into meter.

Using the list pts and p for saving the x and y co-ordinates of the centre of rectangle for more than one person.

Calculating the Euclidian distance between two points using the formula,

D = [(x2-x1)2 + (y2-y1)2]1/2

Find the PPI (Pixels Per Inch) of your webcam and convert pixel distance into inch.

```
for i in range(1, len(pts)):
if pts[i - 1] is None or pts[i] is None:
continue
cv2.line(img,(int(pts[i-1]),int(p[i-1])),(int(pts[i]),int(p[i])),(0,0,255),3)
a1=int(pts[i-1])
b1=int(p[i-1])
a2=int(pts[i])
b2=int(p[i])
c=math.sqrt((a1-a2)**2+(b1-b2)**2)
d=str(math.trunc(c/ppi))+"inch"
```

Displaying a WARNING message on screen if the distance between two people is less than 50inch. (Taking 50 inch as minimum distance required for social distancing)

```
cv2.putText(img,d,(0,50),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),3,cv2.LINE_AA)
if math.trunc(c/ppi)<20:
cv2.putText(img,"WARNING",(100,50),cv2.FONT_HERSHEY_SIMPLEX
,1,(0,0,255),3,cv2.LINE_AA)
cv2.imshow('img',img)
if cv2.waitKey(1) == ord('q'):
break cap.release() cv2.destroyAllWindows()
```

7.2 MobileNetV2 — Light Weight Model (Image Classification)

Fig 7.2: MobileNetV2 for Mobile Devices

MobileNetV2, by **Google**, is briefly reviewed. In the previous version MobileNetV1, **Depthwise Separable Convolution** is introduced which dramatically reduce the complexity cost and model size of the network, which is suitable to Mobile devices, or any devices with low computational power. In MobileNetV2, a better module is introduced with **inverted residual structure**. **Non-linearities in narrow layers are removed** this time. With MobileNetV2 as backbone for feature extraction, state-of-the-art performances are also achieved for object detection and semantic segmentation. This is a paper in **2018 CVPR** with more than **200 citations**. (Sik-Ho Tsang @ Medium)

Outline

- **MobileNetV2 Convolutional Blocks**
- **Overall Architecture**
- **Ablation Study**
- **Experimental Results**

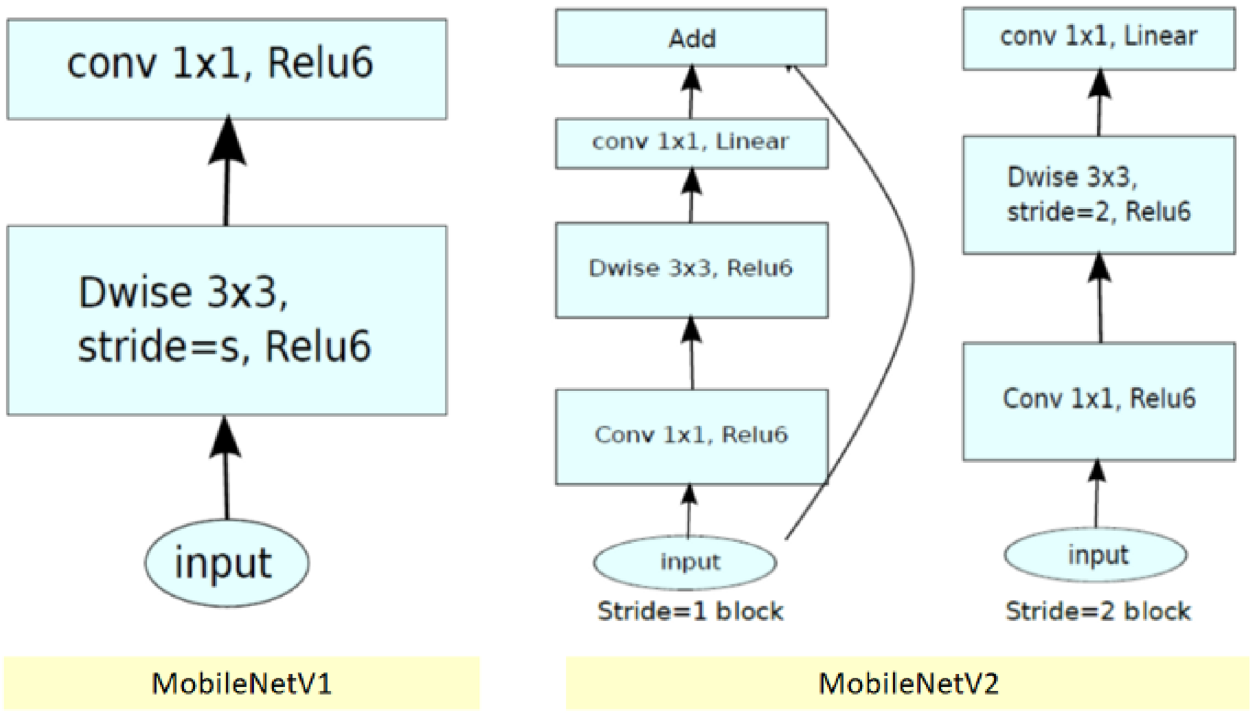
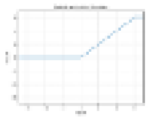


Fig 7.3: MobileNetV2 Convolutional Blocks

- In MobileNetV1, there are 2 layers.
- The **first layer** is called a **depthwise convolution**, it performs lightweight filtering by applying a single convolutional filter per input channel.
- The **second layer** is a **1×1 convolution**, called a **pointwise convolution**, which is responsible for building new features through computing linear combinations of the input channels.

- **ReLU6** is used here for comparison. (Actually, in MobileNetV1 tech report, I cannot find any hints that they use ReLU6… Maybe we need to check the codes in Github…), i.e. **min(max(*x*, 0), 6)** as follows:



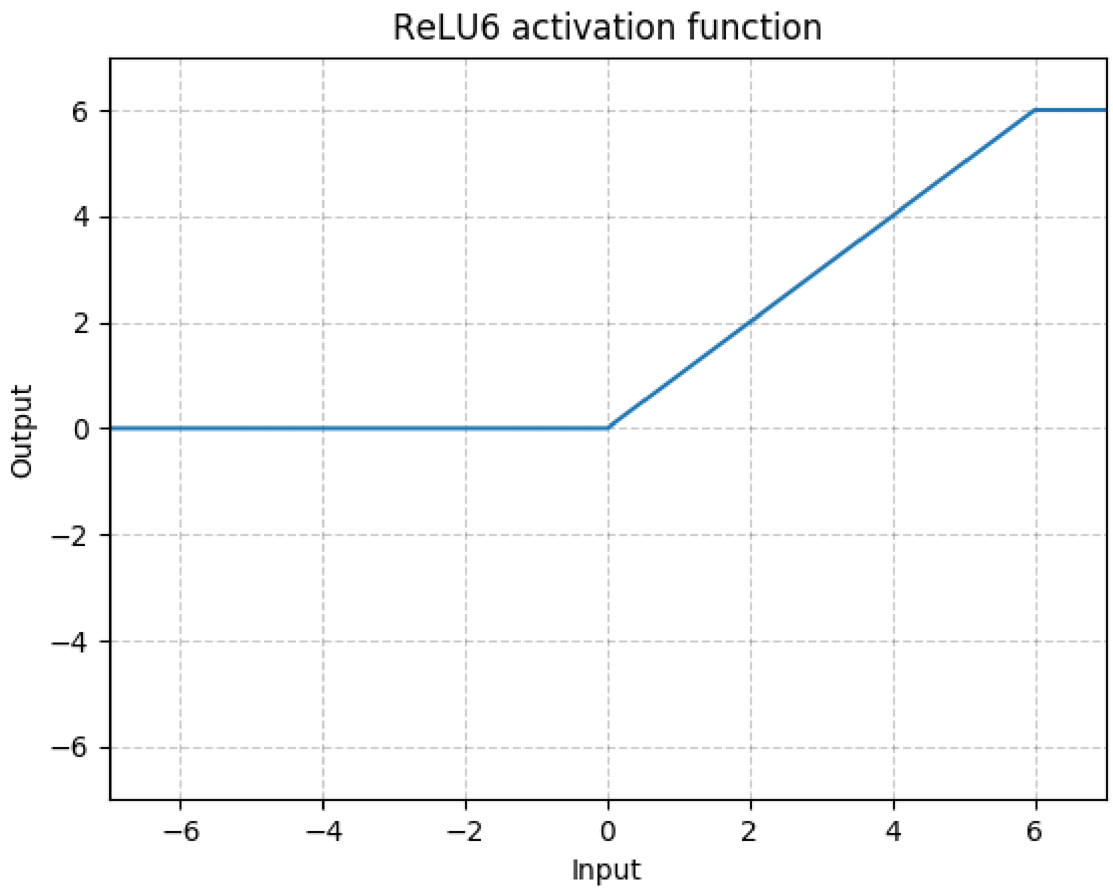


Fig7.4: ReLU6

- ReLU6 is used due to its robustness when used with low-precision computation, based on [27] MobileNetV1.

1.2. MobileNetV2

- 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.
- This time, the **first layer** is **1×1 convolution with ReLU6.**
- The **second layer** is the **depthwise 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.

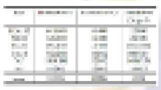
Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

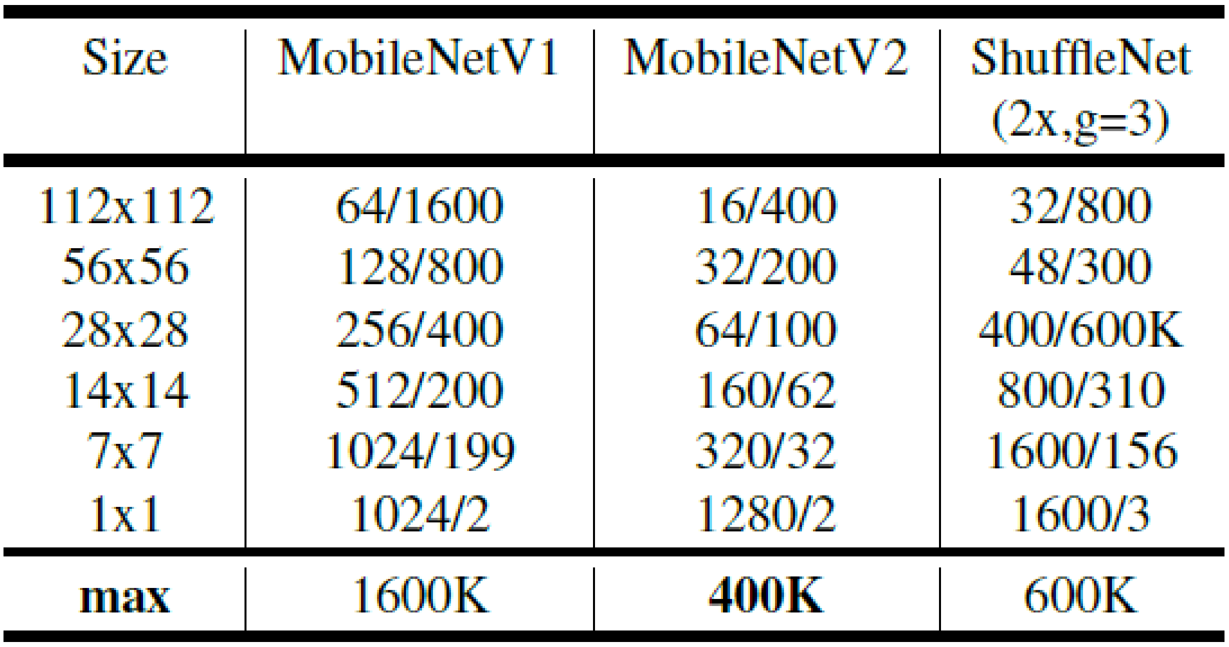
- And there is an expansion factor *t*. And *t*=6 for all main experiments.
- If the input got 64 channels, the internal output would get 64×*t*=64×6=384 channels.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Fig 7.5: MobileNetV2 Overall Architecture

- where *t*: expansion factor, *c*: number of output channels, *n*: repeating number, s: stride. 3×3 kernels are used for spatial convolution.
- In typical, the **primary network** (width multiplier 1, **224×224**), has a computational cost of **300 million multiply-adds** and uses **3.4 million parameters**. (Width multiplier is introduced in MobileNetV1.)
- The performance trade offs are further explored, for **input resolutions from 96 to 224**, and **width multipliers of 0.35 to 1.4**.
- The network computational cost up to 585M MAdds, while the model size vary between 1.7M and 6.9M parameters.
- To train the network, 16 GPU is used with batch size of 96.





Size	MobileNetV1	MobileNetV2	ShuffleNet (2x,g=3)
112x112	64/1600	16/400	32/800
56x56	128/800	32/200	48/300
28x28	256/400	64/100	400/600K
14x14	512/200	160/62	800/310
7x7	1024/199	320/32	1600/156
1x1	1024/2	1280/2	1600/3
max	1600K	**400K**	600K

Fig 7.6: Number of Maximum Channels/Memory in Kb) at Each Spatial Resolution for Different Architecture with 16-bit floats for activation

7.3 Ablation Study

1. Impact of Linear Bottleneck

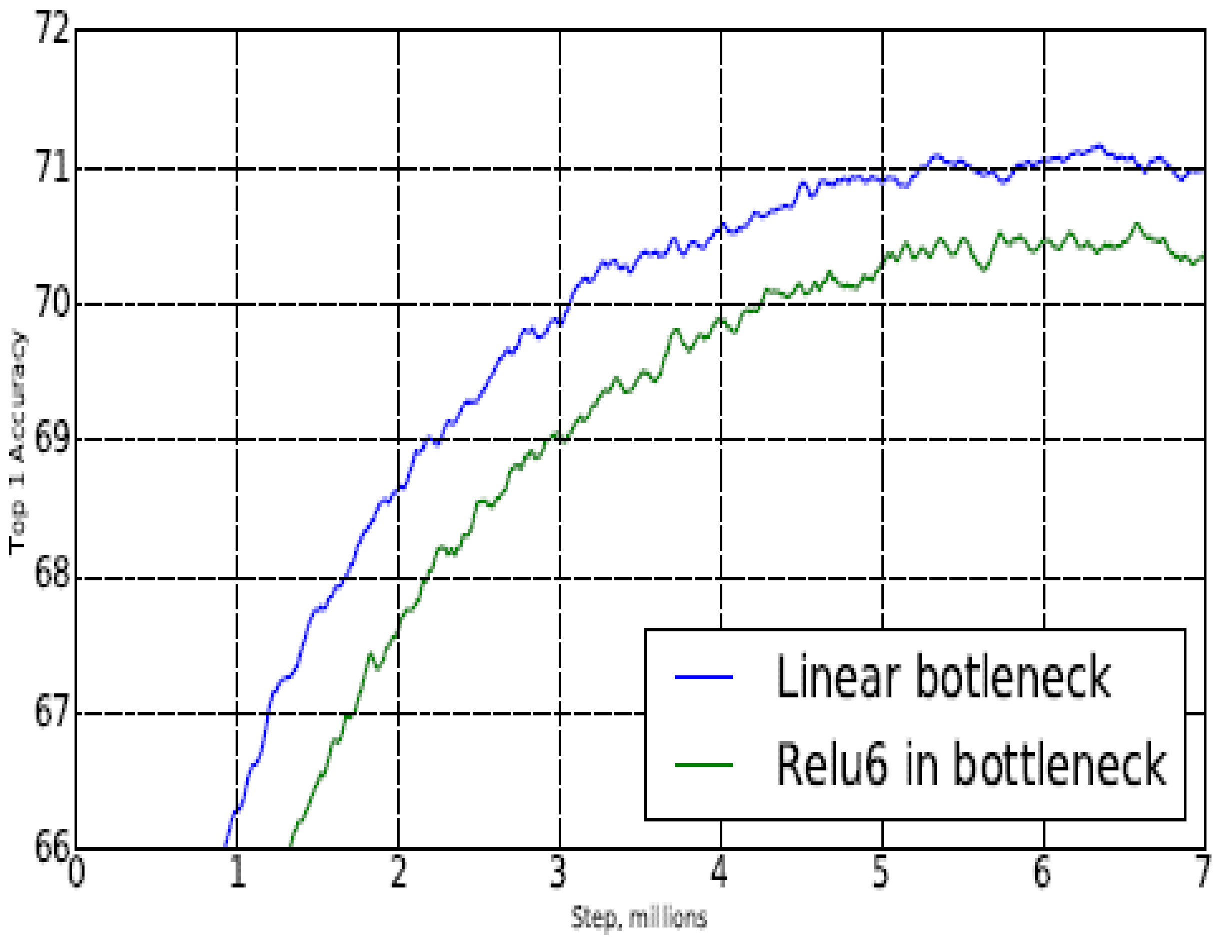


Fig 7.6: Accuracy of ReLU6

- With the removal of ReLU6 at the output of each bottleneck module, accuracy is improved.

2. Impact of Shortcut

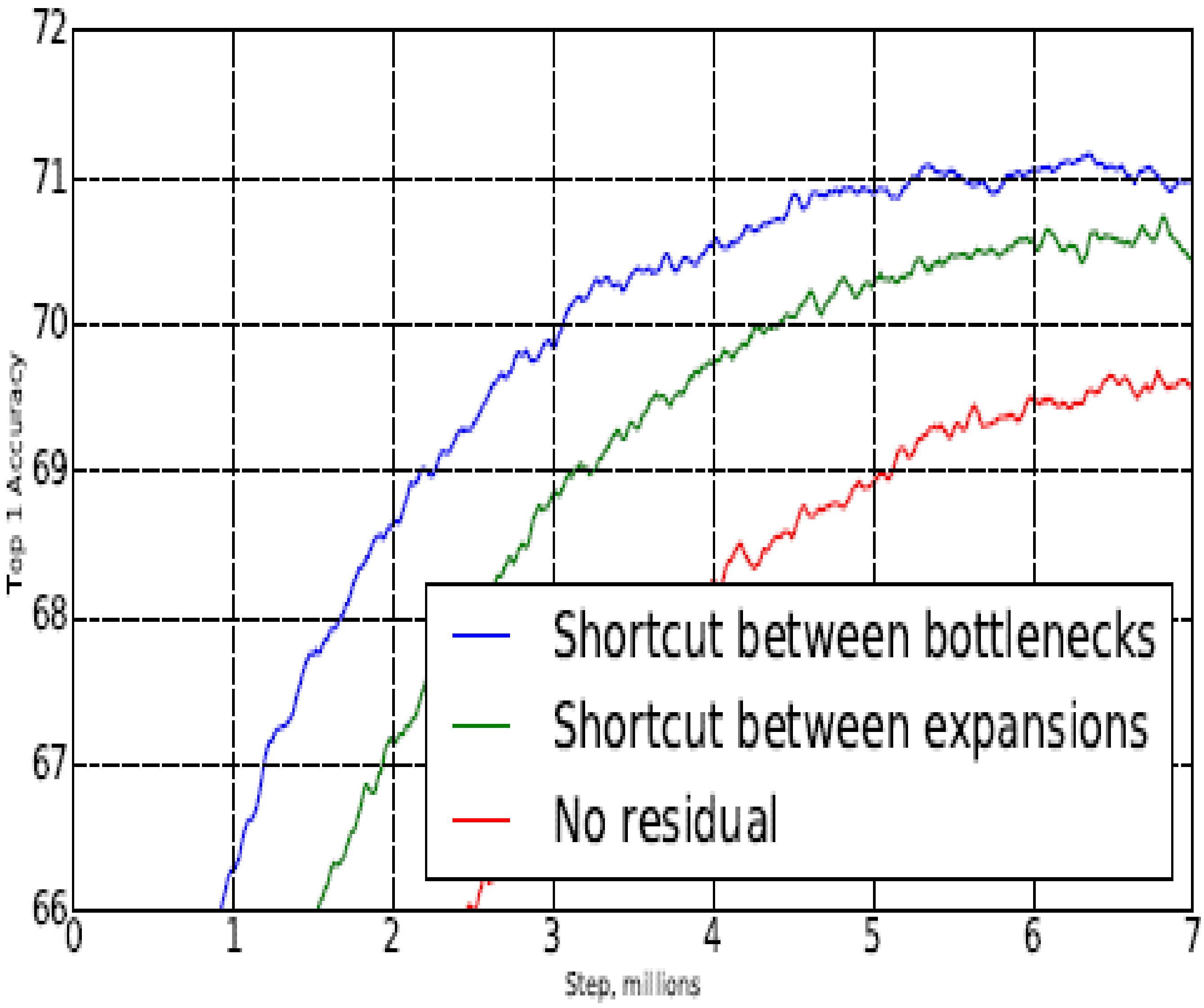


Fig 7.7: Impact of Shortcut

- With shortcut between bottlenecks, it outperforms shortcut between expansions and the one without any residual connections.

3. Experimental Results

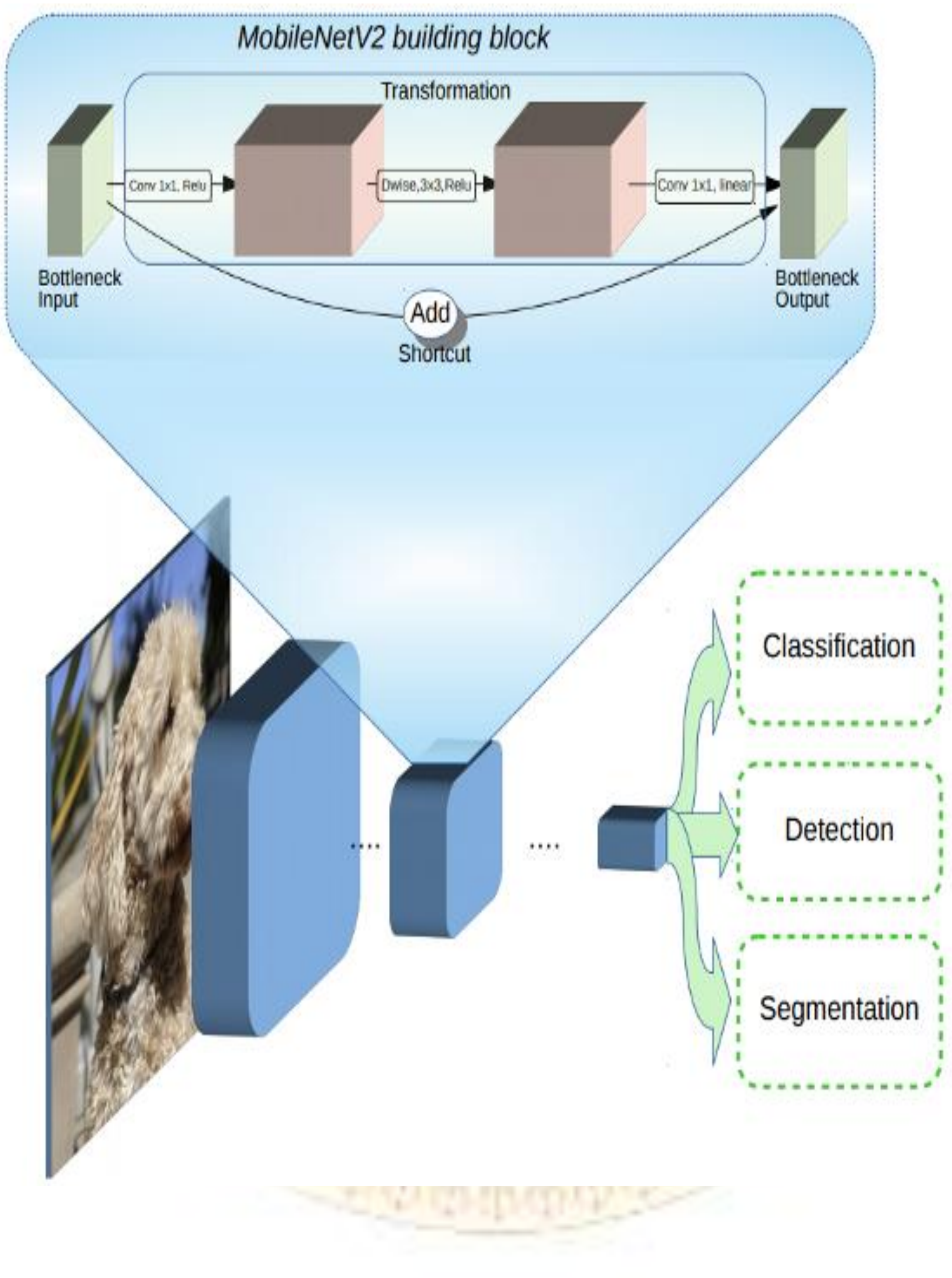
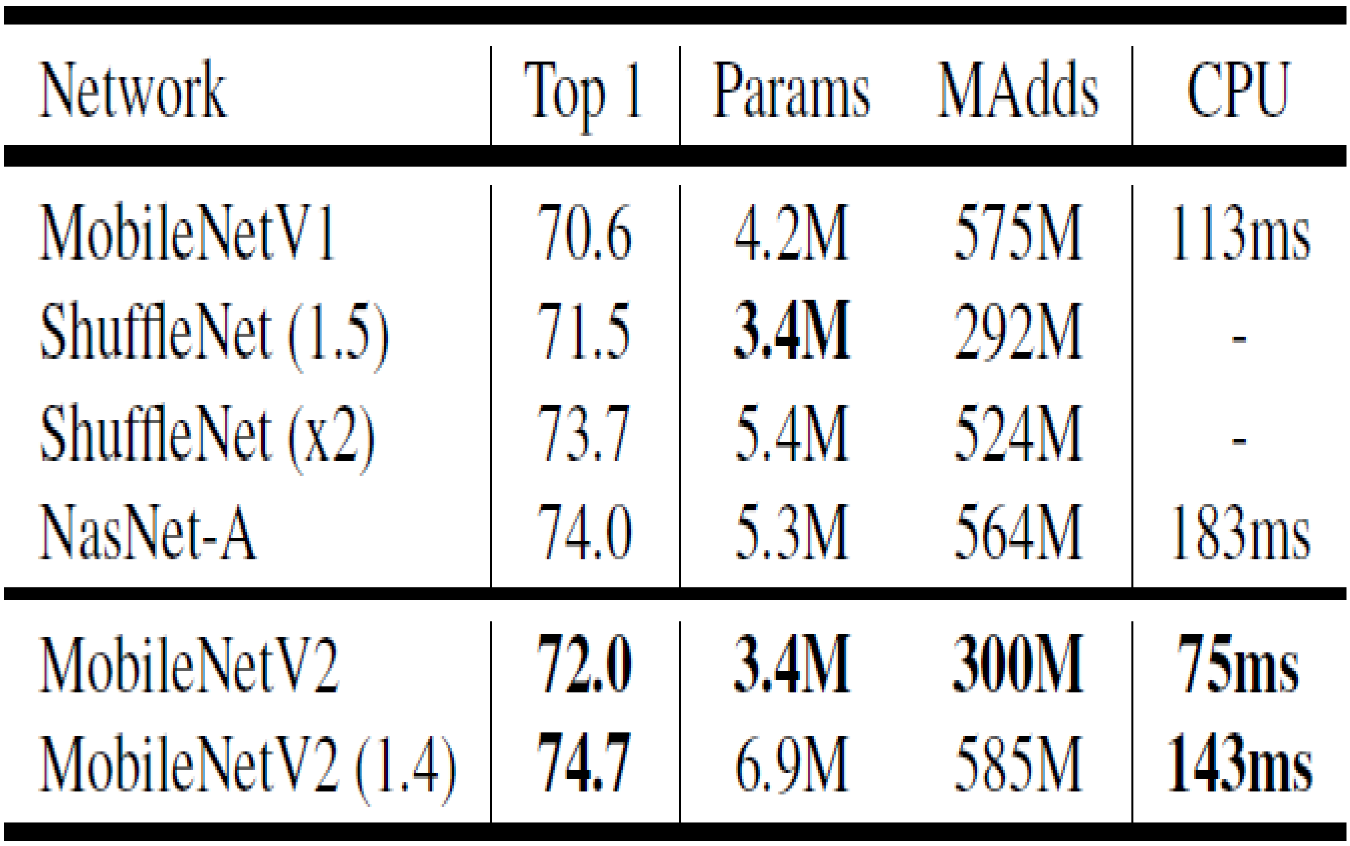


Fig 7.8: MobileNetV2 for Classification, Detection and Segmentation

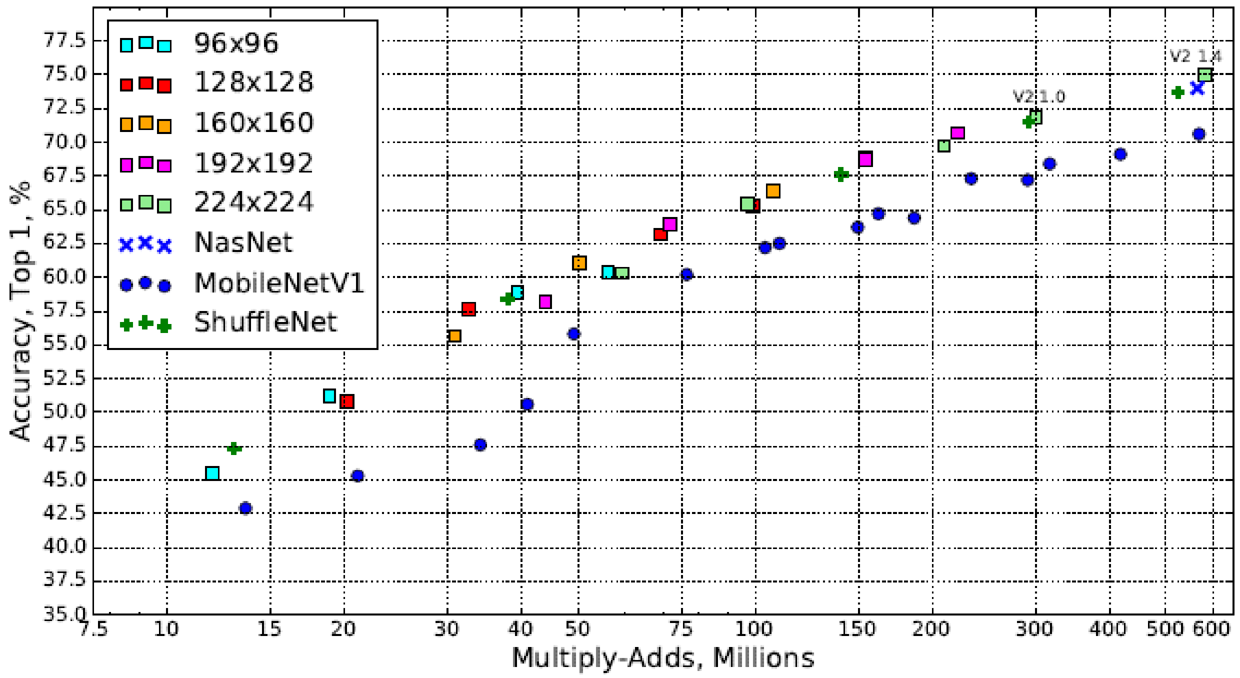
4. ImageNet Classification



Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Fig 7.9: ImageNet Top-1 Accuracy

- MobileNetV2 outperforms MobileNetV1 and ShuffleNet (1.5) with comparable model size and computational cost.
- With width multiplier of 1.4, MobileNetV2 (1.4) outperforms ShuffleNet (×2), and NASNet with faster inference time.



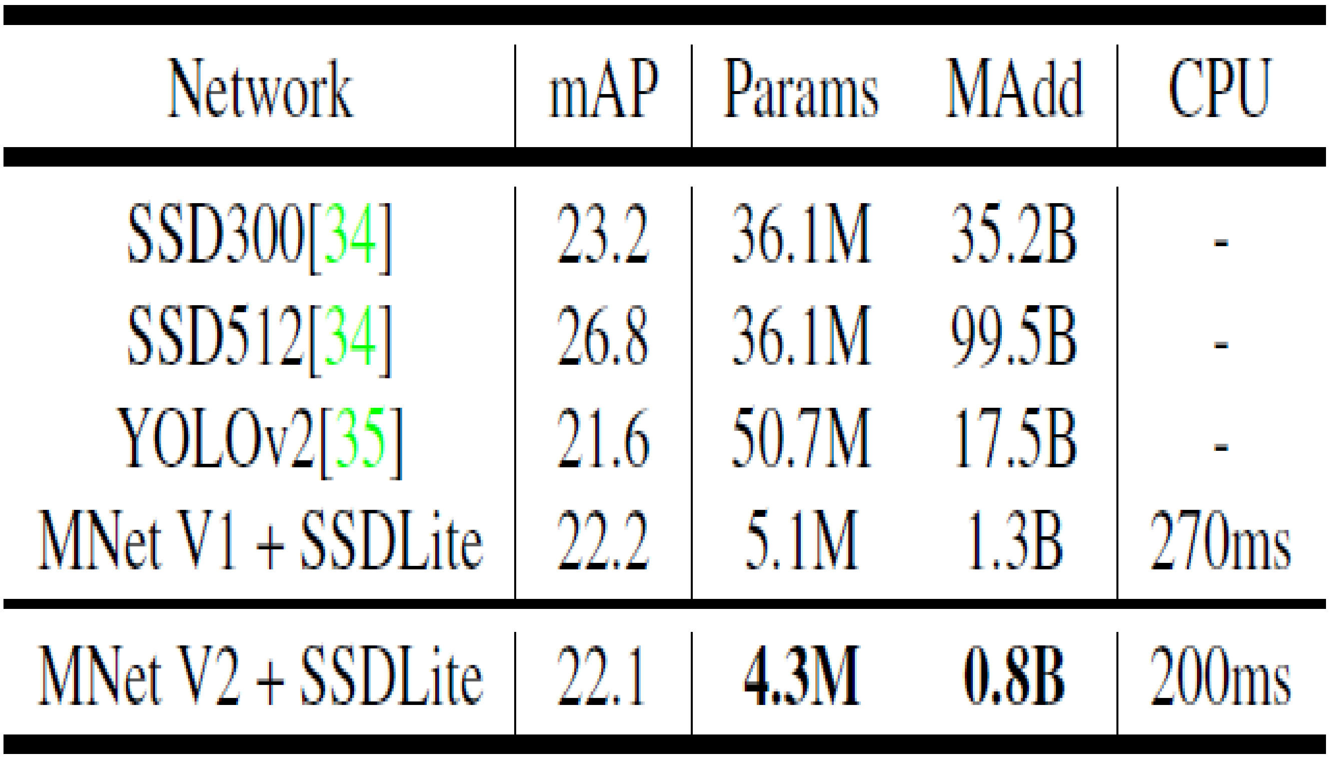
- As shown above, different input resolutions and width multipliers are used. It consistently outperforms MobileNetV1.

5. MS COCO Object Detection

	Params	MAdds
SSD[34]	14.8M	1.25B
SSDLite	**2.1M**	**0.35B**

Fig:7.10: SSDLite

- First, SSDLite is introduced by modifying the regular convolutions in SSD with depthwise separable convolutions (MobileNetV1 one).
- SSDLite dramatically reduces both parameter count and computational cost.



Network	mAP	Params	MAdd	CPU
SSD300[34]	23.2	36.1M	35.2B	-
SSD512[34]	26.8	36.1M	99.5B	-
YOLOv2[35]	21.6	50.7M	17.5B	-
MNet V1 + SSDLite	22.2	5.1M	1.3B	270ms
MNet V2 + SSDLite	22.1	**4.3M**	**0.8B**	200ms

Fig 7.11: MS COCO Object Detection

- MobileNetV2 + SSDLite achieves competitive accuracy with significantly fewer parameters and smaller computational complexity.
- And the inference time is faster than MobileNetV1 one.
- Notably, MobileNetV2 + SSDLite is 20× more efficient and 10× smaller while still outperforms YOLOv2 on COCO dataset.

6. PASCAL VOC 2012 Semantic Segmentation

Network	OS	ASPP	MF	mIOU	Params	MAdds
MNet V1	16	✓		75.29	11.15M	14.25B
	8	✓	✓	78.56	11.15M	941.9B
MNet V2*	16	✓		75.70	4.52M	5.8B
	8	✓	✓	78.42	4.52M	387B
MNet V2*	16			**75.32**	**2.11M**	**2.75B**
	8		✓	77.33	2.11M	152.6B
ResNet-101	16	✓		80.49	58.16M	81.0B
	8	✓	✓	82.70	58.16M	4870.6B

Fig 7.12: PASCAL VOC 2012 Validation Set

- Here, MobileNetV2 is used as feature extractor for DeepLabv3.
- With the disabling of Atrous Spatial Pyramid Pooling (ASPP) as well as Multi-Scale and Flipping (MP), also changing the output stride from 8 to 16, mIOU of 75.32% is obtained, with far low of model size and computational cost.

8. Turning a Raspberry Pi 3B+ into a powerful object recognition edge server with Intel Movidius NCS2

We turn the raspberry PI 3B+ into an object recognition server by deploying a MobileNet-SSD architecture for a fully-working solution using the Intel OpenVINO platform.

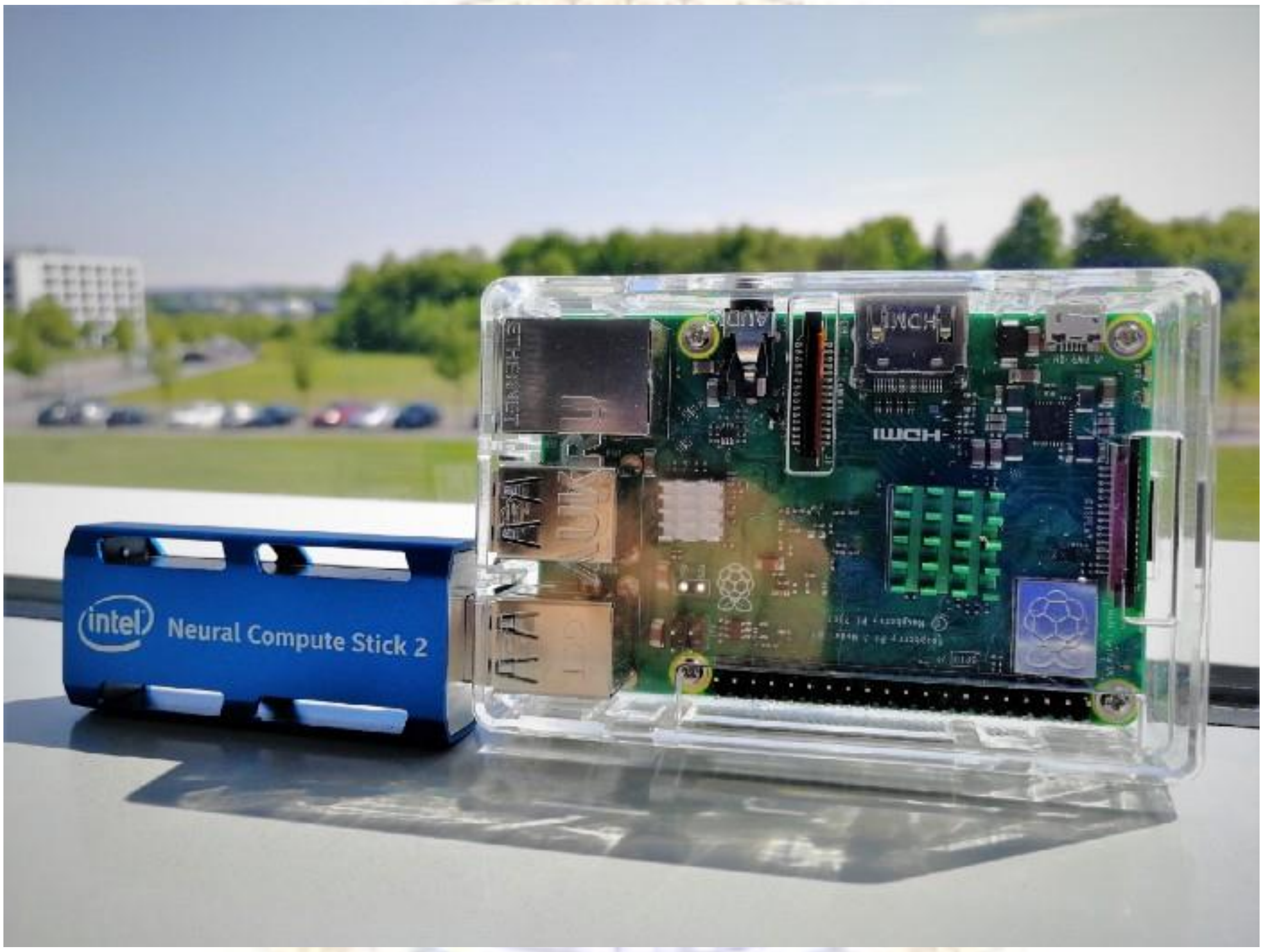


Fig 8.1: MobileNet-SSD architecture

The Intel NCS2 attached to a Raspberry Pi Model 3B+, the hardware used in this tutorial
In this part, we are going to use a readily compiled neural network in the Intel Neural Compute stick in order for it to be able to receive Base64 encoded images and turn them into bounding-box predictions. Additionally, an example for a front-end that sends camera input to the PI will be provided. Please make sure to also check out the amazing write-up on how to deploy and test models by Mattio Varile.

- A pre-trained and compiled model will be provided in an attachment.
- Training and compiling the model for a custom dataset and more details on the front end will be part of another story

8.1 Text-based Graph Convolutional Network

A semi-supervised graph-based approach for text classification and inference

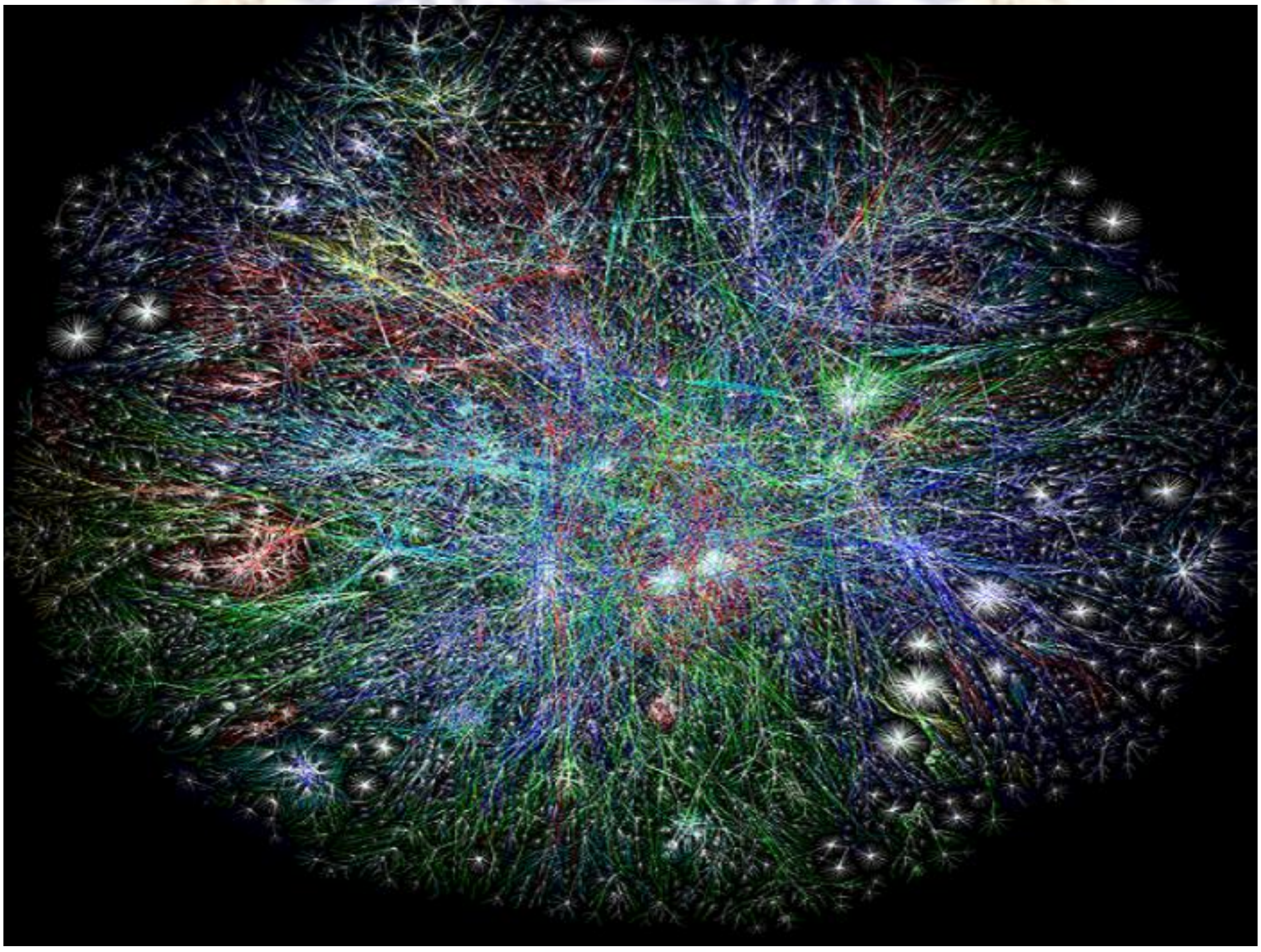


Fig 8.2: Semi-supervised graph-based approach

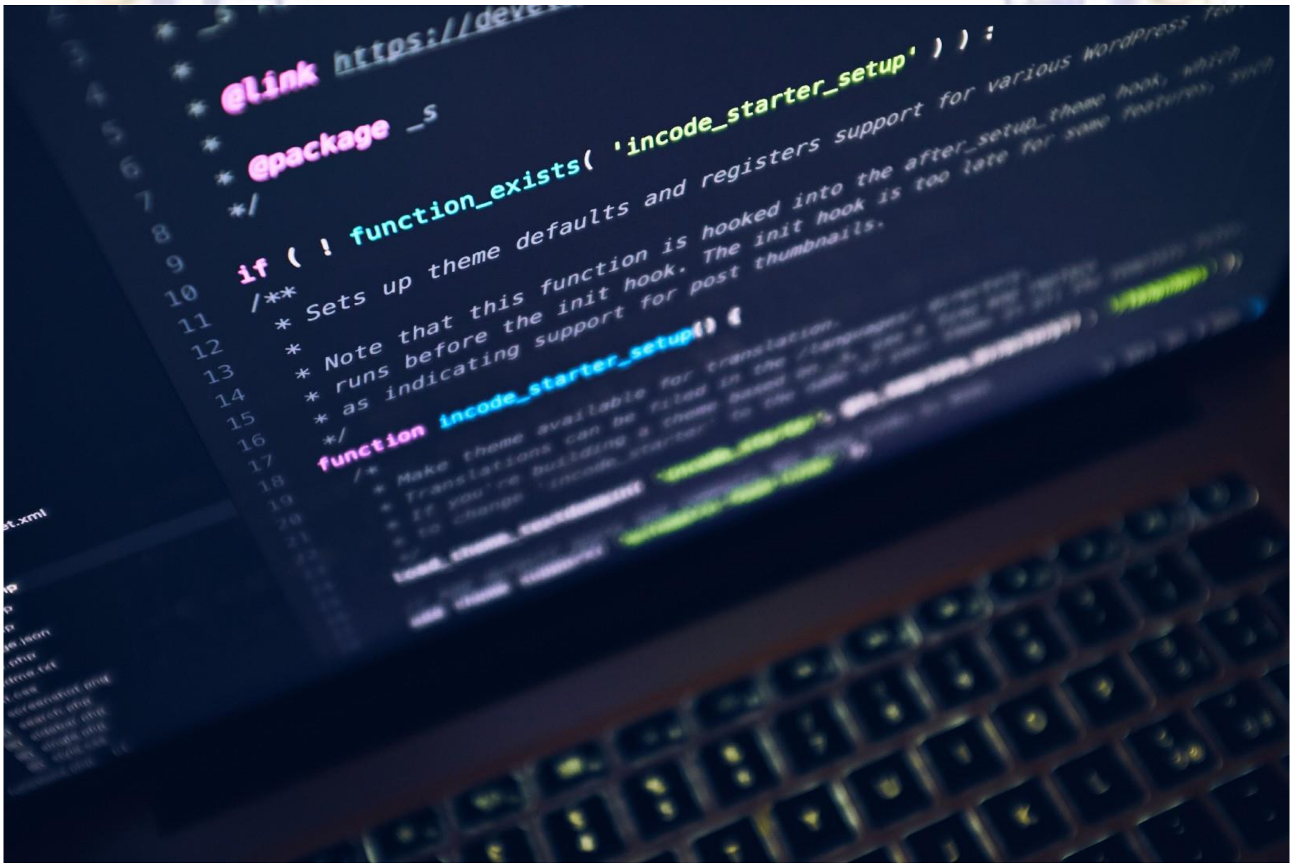
The details of text-based Graph Convolutional Network (GCN) and its implementation using PyTorch and standard libraries. The text-based GCN model is an interesting and novel state-of-the-art *semi-supervised* learning concept that was proposed recently (expanding upon the previous GCN idea by Kipf *et al*. on non-textual data) which is able to very accurately infer the labels of some unknown textual data given related known labeled textual data.

At the highest level, it does so by embedding the entire corpus into a single graph with documents (some labelled and some unlabelled) and words as nodes, with each document-word & word-word edges having some predetermined weights based on their relationships with each other (eg. Tf-idf).

A GCN is then trained on this graph with documents nodes that have known labels, and the trained GCN model is then used to infer the labels of unlabelled documents. …

8.2 Bayes Text Classification in Kotlin for Android without TensorFlow

Explore Bayes Text Classification in pure Kotlin without TensorFlow APIs.



```
 * @link https://devel
 * @package _s
 */
if ( ! function_exists( 'incode_starter_setup' ) ) :
/**
 * Sets up theme defaults and registers support for various WordPress
 *
 * Note that this function is hooked into the after_setup_theme hook, which
 * runs before the init hook. The init hook is too late for some features, such
 * as indicating support for post thumbnails.
 */
function incode_starter_setup() {
```

Fig 8.3: Bayes Text Classification

Text Classification has been an important task in **Natural Language Processing** because of its capabilities and a wide range of uses. We will learn about using this technique in a non-deep learning way, without using TensorFlow and Neural Networks. This classifier will work in an Android application so it's needed to write in Kotlin or Java.
But why Kotlin, why not our TensorFlow or Python?

We aren't using TensorFlow? Because it's written in C++, models are constructed in Python and we need to run it in Kotlin!
TensorFlow and TensorFlow Lite can work efficiently (or sometimes in-mind blowing ways) on Android. A similar algorithm could be created in any programming language like C, C++ or even Swift (native to iOS), if it can be created in Kotlin (native to Android). …

8.3 MACHINE LEARNING APPROACHES FOR TIME SERIES DATA

ML Approaches for Time Series

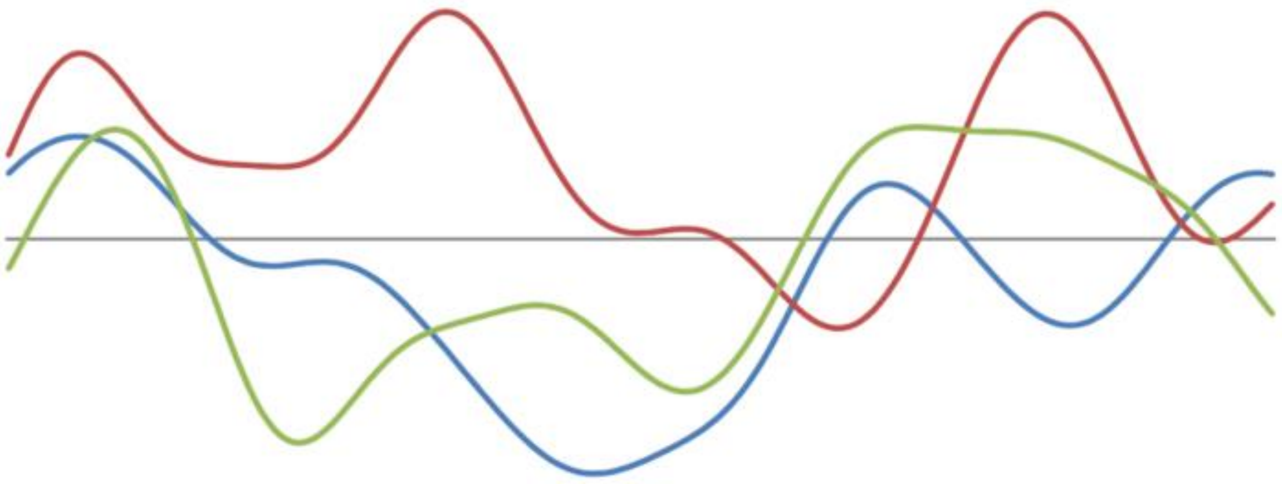


Fig 8.4: Modeling time series with non conventional models

Machine Learning techniques to analyze time series data and explore their potential use in this case of scenarios.

In this first post only the first point of the index is developed. The rest have a separate post which can be accessed from the index.

Data creation, windows and baseline model

Genetic programming: Symbolic Regression

Extreme Learning Machines

Gaussian Processes

Convolutional Neural…

8.4 Getting started with Apache Cassandra and Python

Fig 8.5: Apache Cassandra

Apache Cassandra, its purpose, usage, configuration, and setting up a cluster and in the end, how can you access it in your Python applications. At the end of this post, you should have a basic understanding of Cassandra and how you can use in your Python apps.

Given that COVID-19 is showing no signs of slowing down in many regions of the world, it's more important than ever to maintain "social distancing" aka physical distancing between persons not of the same household in both indoor and outdoor settings. Several national health agencies including the Center for Disease Control (CDC) have recognized that the virus can be spread if "an infected person coughs, sneezes, or talks, and droplets from their mouth or nose are launched into the air and land in the mouths or noses of people nearby."
To minimize the likelihood of coming in contact with these droplets, it is recommended that any two persons should maintain a physical distance of 1.8 meters apart (approximately 6 feet).

Different governing bodies have different rules for safe social distancing and different penalties for breaking them.
We don’t advocate putting in place surveillance to enforce social distancing. Rather, as an interesting exercise, we will use the fact that machine learning and object detection have come a long way in being able to recognize objects in an image. Let’s understand how Python can be used to monitor social distancing.

Fig 8.6: Social Gathering

Python can be used to detect people’s faces in a photo or video loop, and then estimate their distance from each other. While the method we’ll use is not the most accurate, it’s cheap, fast and good enough to learn something about the efficacy of computer vision when it comes to social distancing. Also, this approach is simple enough to be extended to an android compatible solution, so it can be used in the field.
Social Distancing, Python, And Trigonometry

Let’s start by outlining the process to address our use case:

1. Pre-Process images taken by a video camera to detect the contours of the components of the images (using Python)
2. Classify each contour as a face or not (also using Python)
3. Approximate the distance between faces detected by the camera (do you remember your trigonometry?)

To accomplish the Python parts, we’ll use the OpenCV library, which implements most of the state-of-the-art algorithms for computer vision. It’s written in C++ but has bindings for both Python and Java. It also has an Android SDK that could help extend the solution for mobile devices. Such a mobile application might be particularly useful for live, on demand assessments of social distancing using Python.

Steps 1 and 2 in our process are fairly straightforward because OpenCV provides methods to get results in a single line of code. The following code creates a function faces_dist, which takes care of detecting all the faces in an image:

```
def faces_dist(classifier, ref_width, ref_pix):
  ratio_px_cm = ref_width / ref_pix
  cap = cv2.VideoCapture(0)
  while True:
      _, img = cap.read()
      gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
      # detect all the faces on the image
      faces = classifier.detectMultiScale( gray, 1.1, 4)
      annotate_faces( img, faces, ratio_px_cm)

      k = cv2.waitKey(30) & 0xff
      # use ESC to terminate
      if k==27:
         break
  # release the camera
  cap.release()
```

After starting the camera in video mode, we enter into an infinite loop to process each image streamed from the video. To simplify identification, the routine transforms the image to grayscale, and then, using the classifier passed as an argument, we get a list of tuples that contains the X, Y coordinates for each face detected along with the corresponding Width and Height which gives us a rectangle

Next, we call the annotate_faces function, which is in charge of drawing the detected rectangles and calculating the euclidean distance between the objects detected:

```
def annotate_faces( img, faces, ratio_px_cm ):
  points = []
  for (x, y, w, h) in faces:
      center = (x+(int(w/2)), y+(int(h/2)))
      cv2.circle( img, center, 2, (0,255,0),2)
      for p in points:
          ed = euclidean_dist( p, center ) * ratio_px_cm
          color = (0,255,0)
          if ed < MIN_DIST:
              color = (0,0,255)
          # draw a rectangle over each detected face



          cv2.rectangle( img, (x, y), (x+w, y+h), color, 2)
          # put the distance as text over the face's rectangle
          cv2.putText( img, "%scm" % (ed),
              (x, h -10), cv2.FONT_HERSHEY_SIMPLEX,
              0.5, color, 2)
          # draw a line between the faces detected
          cv2.line( img, center, p, color, 5)
      points.append( center )

  cv2.imshow('img', img )
```

9. Object Detection Using Python

Now that we're well on our way to solving the problem, let's step back and review Python's object detection capabilities in general, and human face detection in particular. We're using a classifier to do human face detection. In the simplest sense, a classifier can be thought of as a function that chooses a category for a given object. In our case, the classifier is provided by the OpenCV library (of course, you could write your own classifier, but that's a subject for another time). OpenCV has already been pre-trained against an extensive dataset of images featuring human faces. As a result, it can be very reliable in detecting areas of an image that correspond to human faces. But because real world use cases can differ, OpenCV provides two models for detection:

- **Local Binary Pattern (LBP)** divides the image into small quadrants (3×3 pixels) and checks if the quadrants surrounding the center are darker or not. If they are, it assigns them to 1; if not 0. Given this information, the algorithm checks a feature vector against the pre-trained ones and returns if an area is a face or not. This classifier is fast, but prone to error with higher false positives.

- **Haar Classifier** is based on the features in adjacent rectangular sections of the image whose intensities are computed and compared. The Haar-like classifier is slower than LBP, but typically far more accurate.

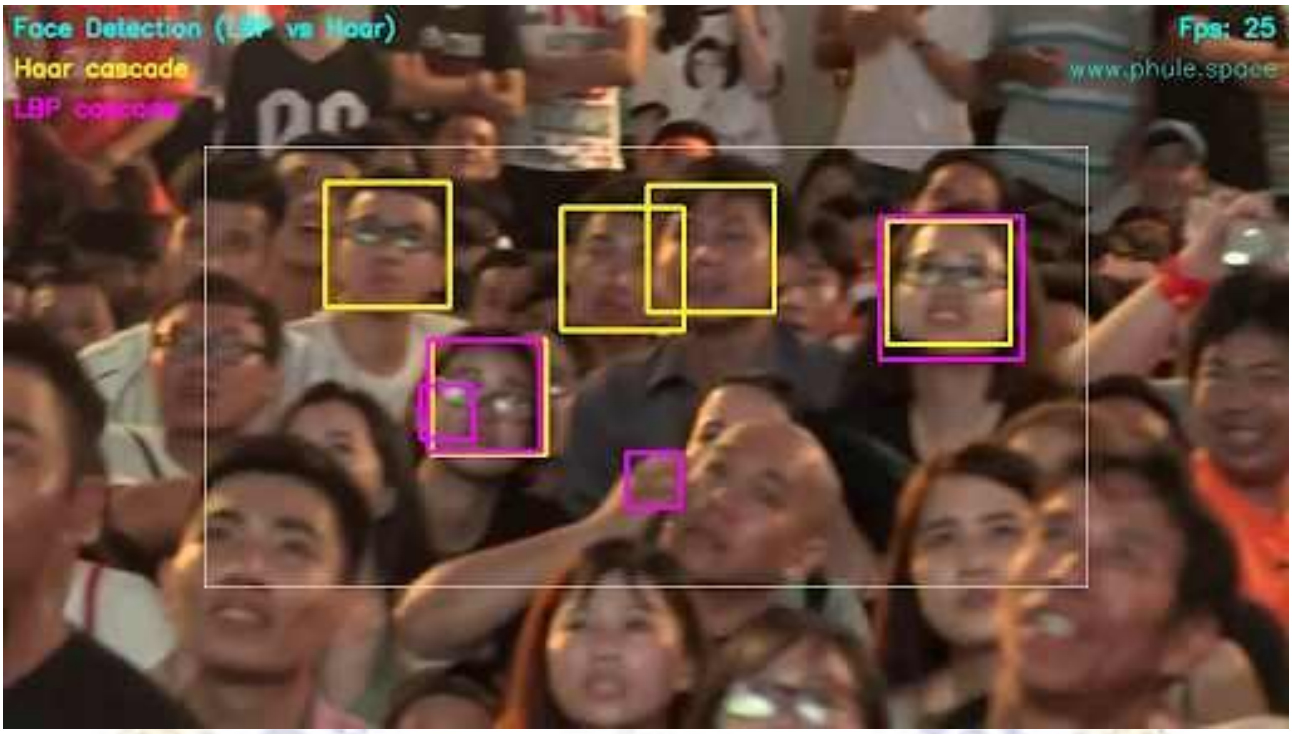


Fig 9.1: Object Detection

Object Detection is quite specific since it pertains to social distancing: we need to detect faces, which both classifiers are suited for. But we also need to build an area that represents the most prominent part of the face. That's because in order to calculate the distance between faces, we first need to approximate the distance the faces are from the camera, which will be based on a comparison of the widths of the facial areas detected.

As a result, we have chosen the Haar Classifier since in our tests the rectangle detected for the face was better than that approximated by LBP.
Approximating Distances

Now that we can reliably detect the faces in each image, it's time to tackle step 3: calculating the distance between the faces. That is, to approximate the distance between the centroid of each rectangle drawn. To accomplish that, we have to calculate the ratio between pixels and cm measured from a known distance for a known object reference. The reference_pixels function calculates this value which is used in the faces_dist function described earlier:

```
def reference_pixels(image_path, ref_distance, ref_width):
  # open reference image
```

```
   image = cv2.imread( image_path )
   edged = get_edged( image )
   # detect all contours over the gray image
   allContours = cv2.findContours( edged.copy(), cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE )
   allContours = imutils.grab_contours( allContours )
   markerContour = get_marker( allContours )
   # use the marker width to calculate the focal length of the camera
   pixels = (cv2.minAreaRect( markerContour ))[1][0]
   return pixels
```

We can use the classic Euclidean distance formula between each centroid, and then transform it to measurement units (cm) using the ratio calculated:

```
D = square_root( (point1_x  + point2_x ) ^2 +  (point1_y  + point2_y ) ^2 )
```

With this simple formula our annotate_faces function adds the approximation to a line drawn from each centroid. If the approximation is lower than the minimal distance required the line will be red.
Object Detection Programmed For Social Distancing

The complete source code for this example is available in my Github repository. There, you'll find the code in which we pass three arguments to our Python script:

- The path of the reference image
- The reference distance in centimeters
- The reference width in centimeters

Given these parameters, our code will calculate the focal length of the camera, and then use detect_faces.py to determine the approximate distances between the faces the camera detects. To stop the infinite cycle, just press the ESC key. The GitHub repo also contains reference images and two datasets for OpenCV's classifiers.

```
classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
#classifier = cv2.CascadeClassifier('lbpcascade_frontalface_improved.xml')
 IMG_SRC = sys.argv[1]
```

10. Conclusion

The current study used OpenCV, tensorflow, keras and CNN to detect whether people were wearing face masks or not. The models were tested with images and real-time video streams. Even though the accuracy of the model is around 90%, the optimization of the model is a continuous process and we are building a highly accurate solution by tuning the hyperparameters. MobileNetV2 was used to build the mobile version of the same. This specific model could be used as a use case for edge analytics.

The scenario would mostly focus on accepting and obeying the precautions and rules that WHO has imposed more precisely as responsibility of one will totally embark on themselves
and not government. Social Distancing would undoubtedly be the most important factor as COVID 19 spreads through close contact with infected ones. Using installed CCTV and drones, authorities can keep a track of human activities and control large crowd to come together and prevent violating the law.

The focus of this project is to prevent the spreading of coronavirus disease by urging people through its Face Mask and Social Distancing detection platform so that the covid-19 norms can be maintained properly.

Thus, implementing this idea can reduce the on-ground efforts of the police and they can entirely focus on supervising conditions exclusively on those areas where conditions are unfavorable and thus, they can utilize time wisely and save energy for equitable situations.

11. REFERENCES

- G.A.F Seber and A.J. Lee, Linear Regression Analysis. John Wiley & Sons Publishers, USA, 2012.
- Baldi, P., Frasconi, P., Smyth, P. (2003). Modeling the Internet and the Web - Probabilistic Methods and Algorithms.
- Lim, D.H., Kim, J.S., Lee, B.G., Wave Information Estimation and Revision Using Linear Regression Model. Journal of Korea Multimedia Society, 19, 8 (2016), 1377-1385.
- Rivest, R.L. (1987). Learning decision lists. Machine Learning,2, 229-246.
- Chan YH. Biostatistics 201: Linear regression analysis. Age (years). Singapore Med J 2004;45:55-61.
- Chakrabarti, S. (2003). Mining the Web, Morgan Kaufmann.
- Cohen, P.R. (1995) Empirical Methods in Artificial Intelligence. Cambridge, MA: MIT Press. This is an excellent reference on experiment design, and hypothesis testing, and related topics that are essential for empirical machine learning research.
- Brunak, S. (2002). Bioinformatics: A Machine Learning Approach. Cambridge, MA: MIT Press.
- Rivest, R.L. (1987). Learning decision lists. Machine Learning,2, 229-246.

- Quinlan, J.R. (1986). Induction of decision trees. Machine Learning, 1, 81-106.
- C. Cortes and V. Vapnik. Support-vector networks. Machine learning, 20(3):273–297, 1995.
- Monitoring Social Distancing for Covid-19 Using OpenCV and Deep Learning by Rucha Visal, Atharva Theurkar, Bhairavi Shukla.
- Ibiyemi T.S., Ogunsakin J., Daramola S.A. 2012.“Bi-Modal Biometric Authentication by Face Recognition and Signature Verification”, International Journal of Computer Applications, vol.42, no. 20, pp 17-21.
- Object Detection and Tracking using Deep Learning and Artificial Intelligence for Video Surveillance Applications by Mohana and HV Ravish Aradhya.
- S. Feng, C. Shen, N. Xia, W. Song, M. Fan, and B. J. Cowling, “Rational use of face masks in the covid19pandemic,”The Lancet Respiratory Medicine, 2020.
- P. A. Rota, M. S. Oberste, S. S. Monroe, W. A. Nix, R. Campagnoli, J. P. Icenogle, S. Penaranda, B. Bankamp,K. Maher, M.-h. Chenet al., “Characterization of a novel coronavirus associated with severe acute respiratorysyndrome,”science, vol. 300, no. 5624, pp. 1394–1399, 2003
- Zhang C., Zhang Z. A Survey of Recent Advances in Face Detection. Microsoft Corporation; Albuquerque, NM, USA: 2010. TechReport, No. MSR-TR-2010-66.
- Monitoring COVID-19 social distancing with person detection by Narinder Singh Punn, Sanjay Kumar Sonbhadra and Sonali Agarwal.
- COVID-19 and Importance of Social Distancing by Fahim Aslam.
- Monitoring Social Distancing for Covid-19 Using OpenCV and Deep Learning Rucha Visal, Atharva Theurkar, Bhairavi Shukla
- Ibiyemi T.S, Akintola A.G. 2012 “Speaker Authentication and Speech Recognition Enabled Telephone Auto-Dial in Yorùbá”, International Journal of Science and Advanced Technology, vol.12, no.4, pp 88-187.
- Ibiyemi T.S., Aliu S.A. 2003. “Automatic Face Recognition by Computer”, Abacus: Mathematics Series, vol 30, no. 2B, September, pp180-188
- Ibiyemi T.S., Aliu S.A. 2002. “On Computation of Optimum Basis Vector for Face Detection and Recognition”, Abacus: Mathematics Series, 29(2), pp 144-149
- .Brian Harding, Cat Jubinski, “A Standalone Face Recognition Access Control System”, ECE4760 Final Project Report,
- Turk M., Pentland A. 1991. “Eigenfaces for Recognition”, Journal of Cognitive Neuroscience, vol. 3, no.1, pp71-86 URL: http://people.ece.edu/land/courses/ece4760

- LipeikaAntanas, Lipeikiene Joana, TelksnysLaimutis. 2002. " Development of Isolated word Speech Recognition", Informatica, vol.13, no.1, 37-46
- E-Hocine Bourouba, et al. 2006."Isolated Words Recognition System Based on Hybrid Approach DTW/GHMM", Informatica 30 373-384
- Satyahad Singh, and Rajan E.G. 2011. "MFCC VQ based Speaker Recognition and its Accuracy Affecting Factors", International Journal of Computer Applications, vol. 21, no.6, pp.1-6
- RashidulHasan, Mustafa Jamil, GolamRabbani, Saifur Rahman 2004. "Speaker Identification using Mel Frequency Cepstral Coefficients", Proc. 3rd International Conference on Electrical and Computer engineering, ICECE 2004, 28-30 December, Dhaka Bangladesh, pp. 565-568
- Srinivasan A.. 2012. "Speaker Identification and Verification using Vector Quantisation and Mel Frequency Cepstral coefficients", Research Journal of Applied Sciences, Engineering and technology", vol.4, no.1, pp. 33-40
- Allam Musa. 2011,."MareText Independent Speaker Identification based on K-Means Algorithm", International Journal on Electrical engineering and Informatics, vol.3, no.1, pp100-108

APPENDIX A
SCREEN SHOTS

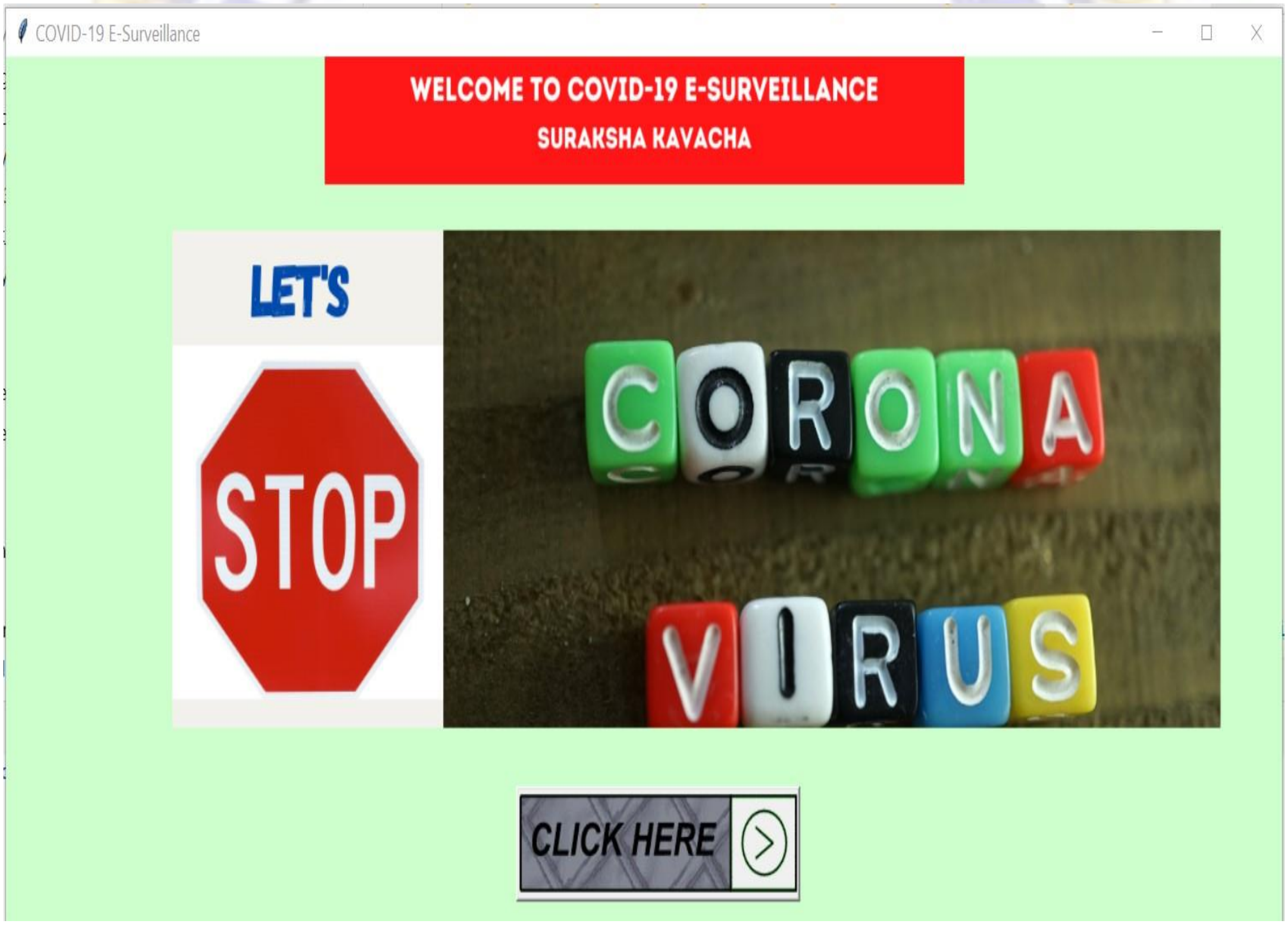


Fig 1: Welcome Page

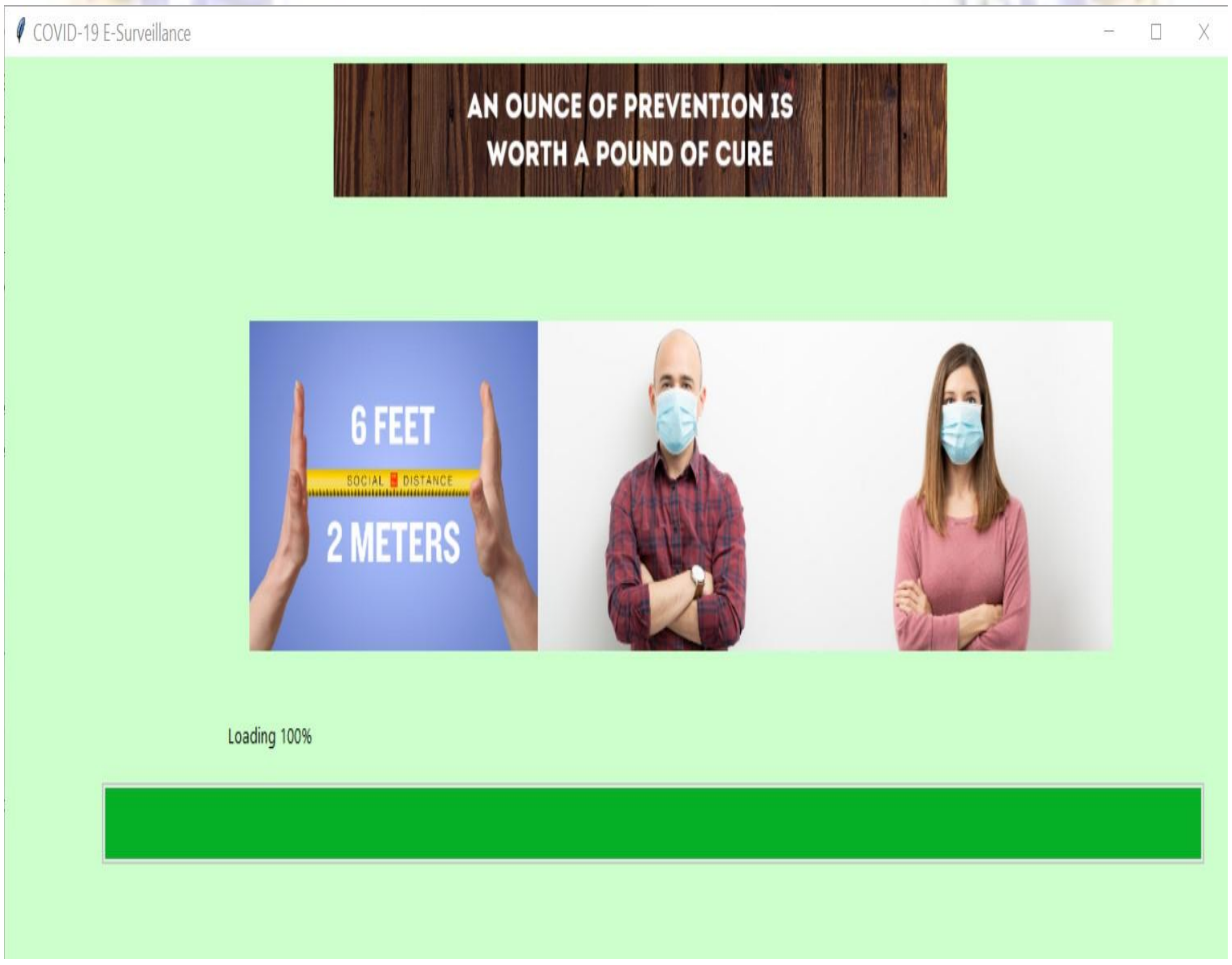

COVID-19 E-Surveillance
AN OUNCE OF PREVENTION IS WORTH A POUND OF CURE
6 FEET
SOCIAL DISTANCE
2 METERS
Loading 100%

Fig 2: Loading Page

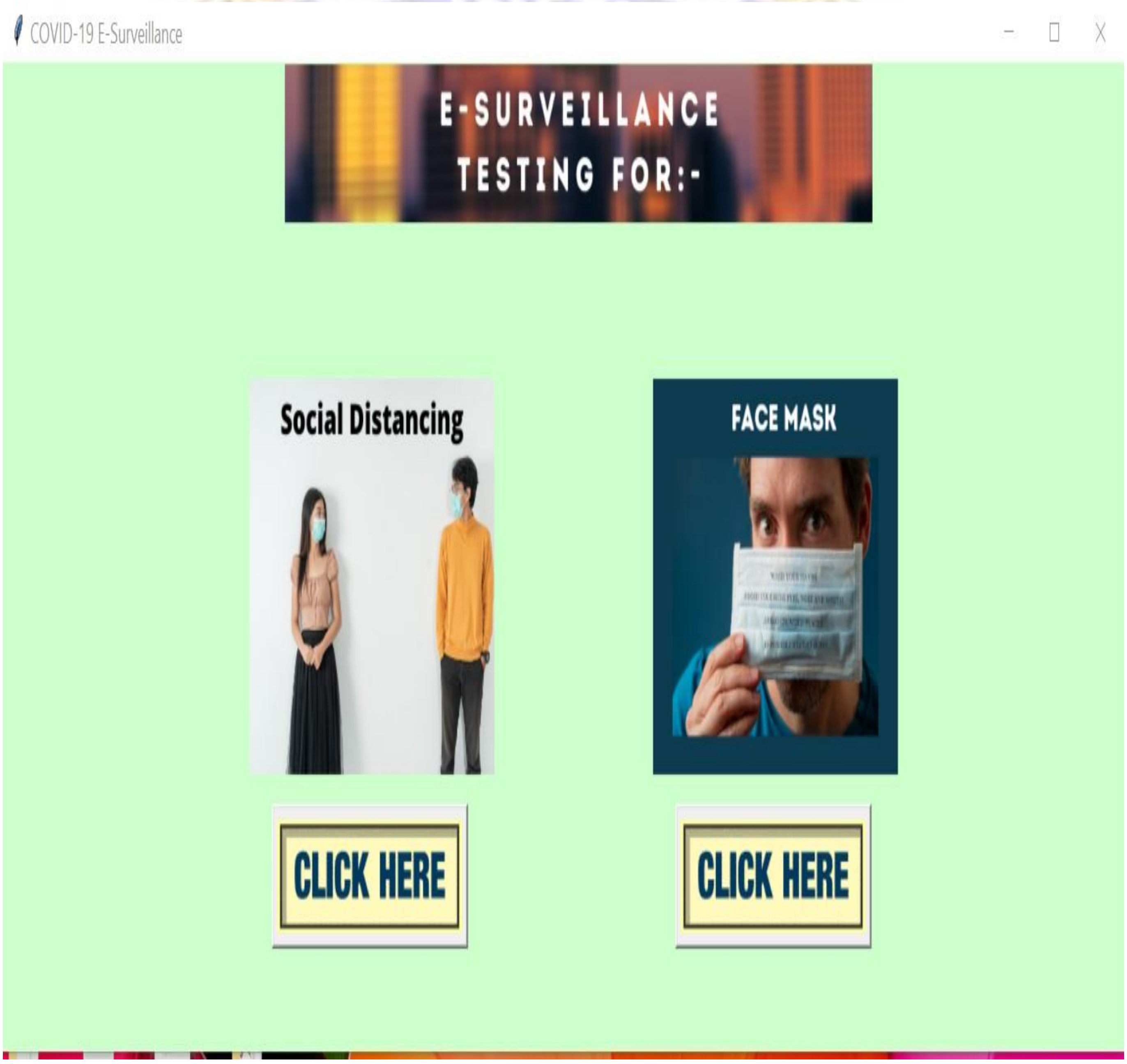
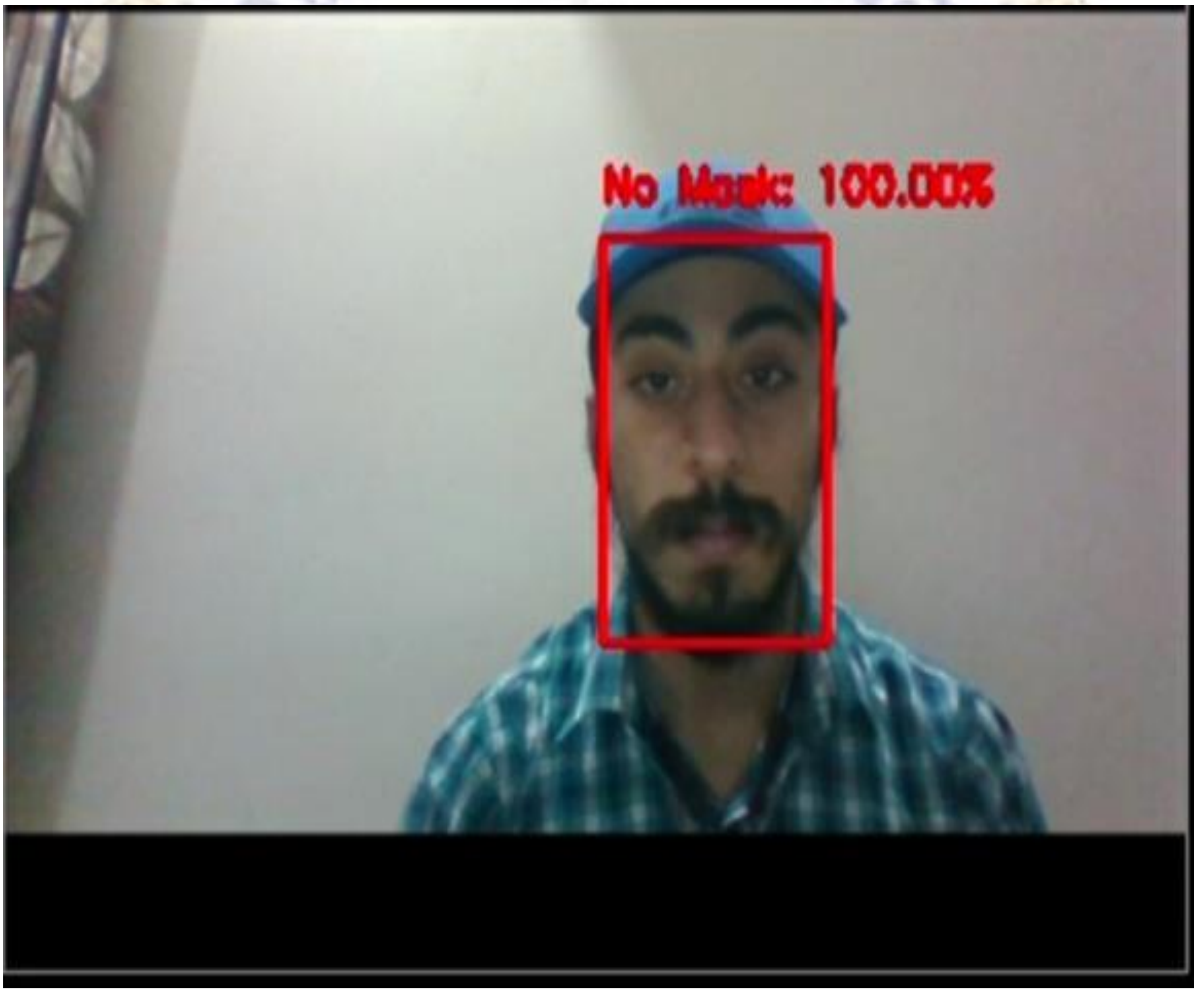
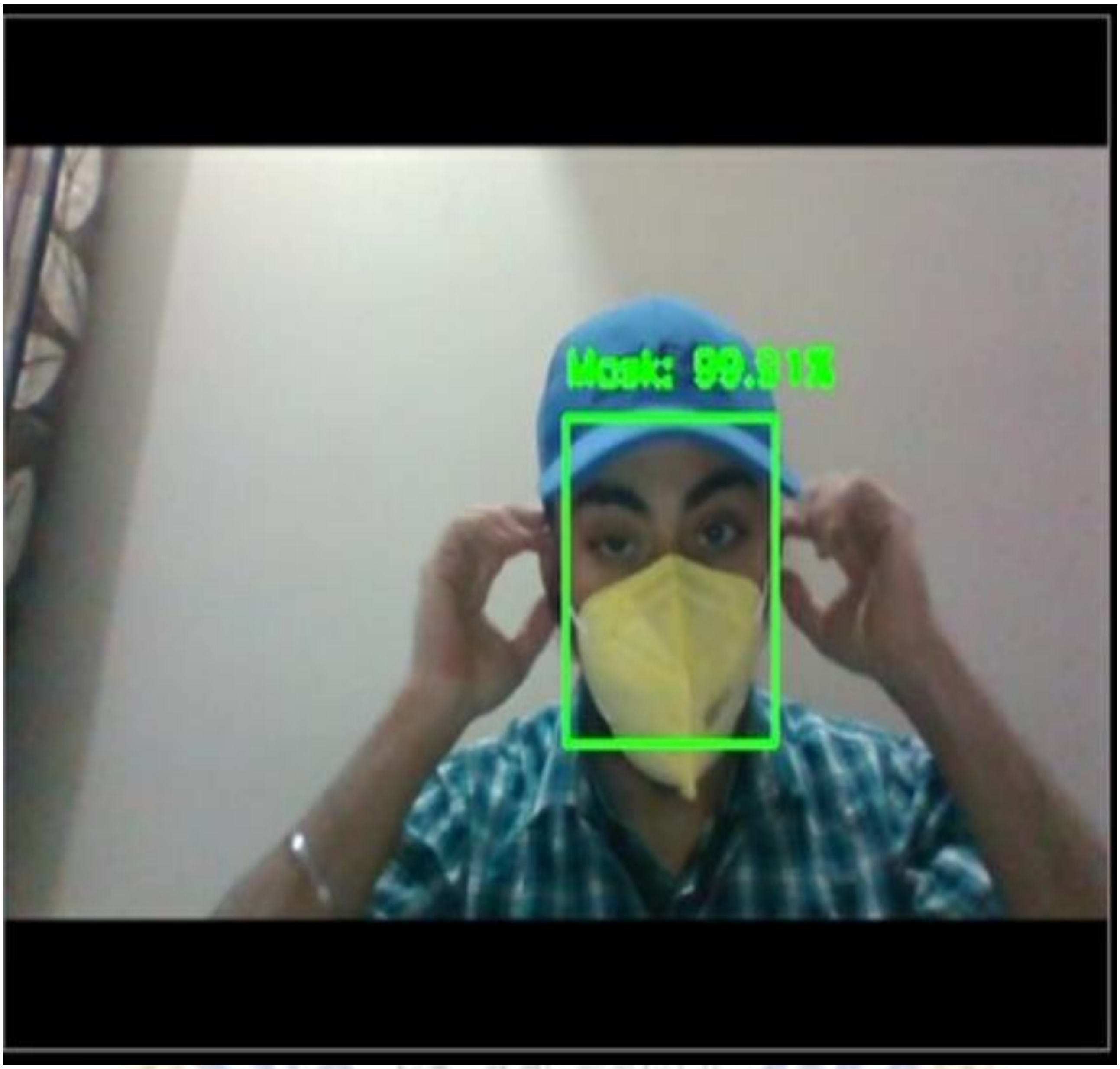
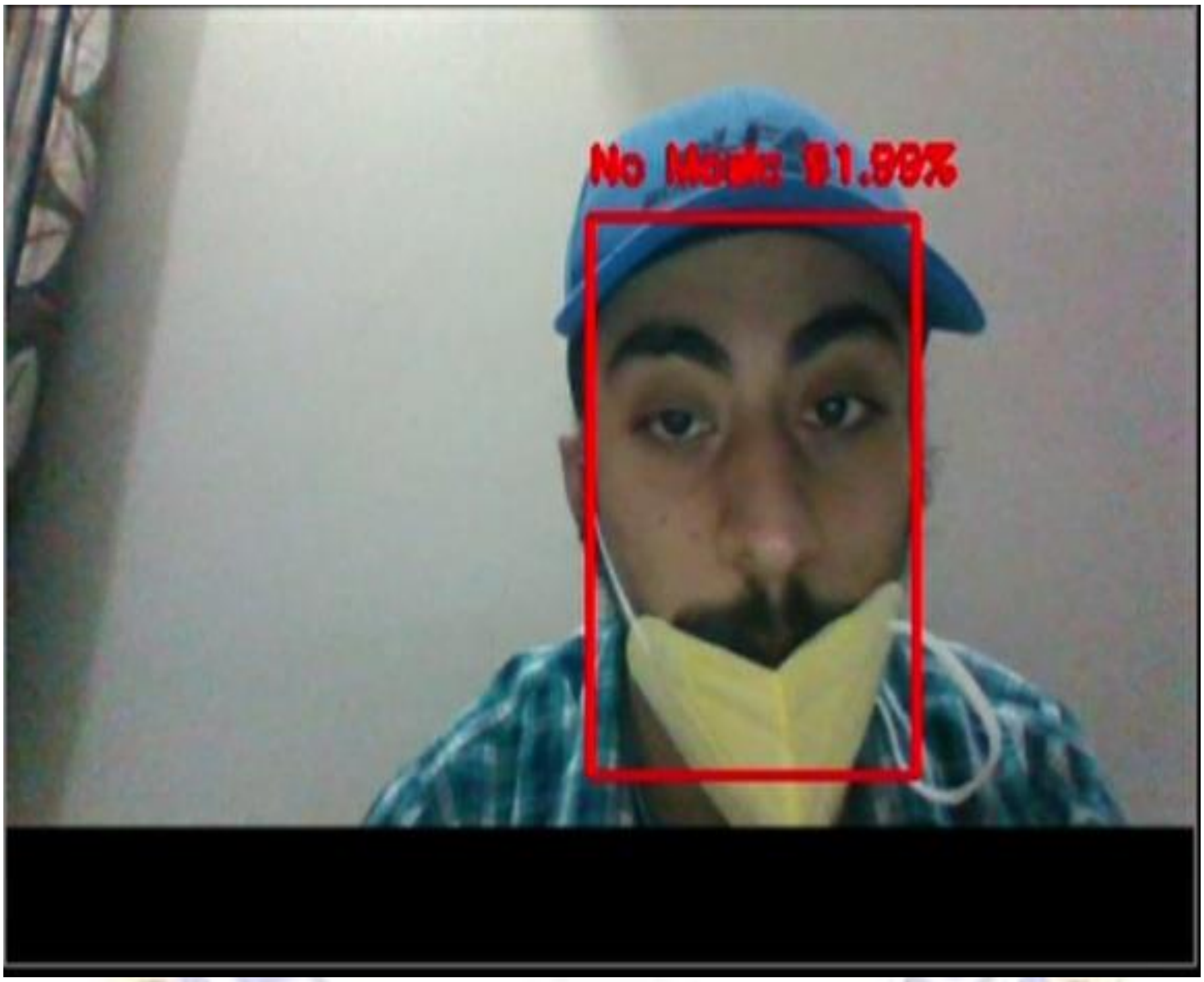
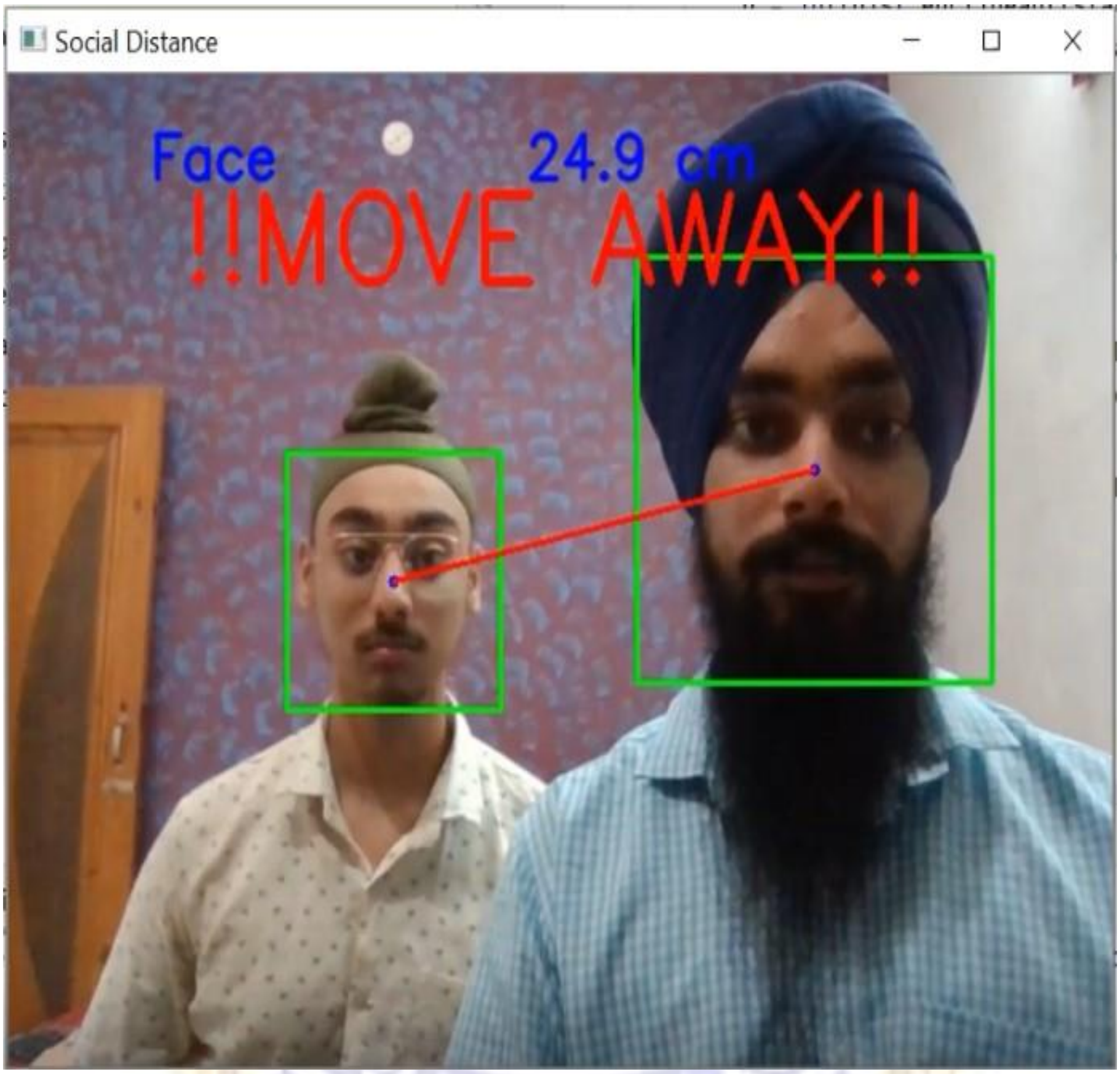


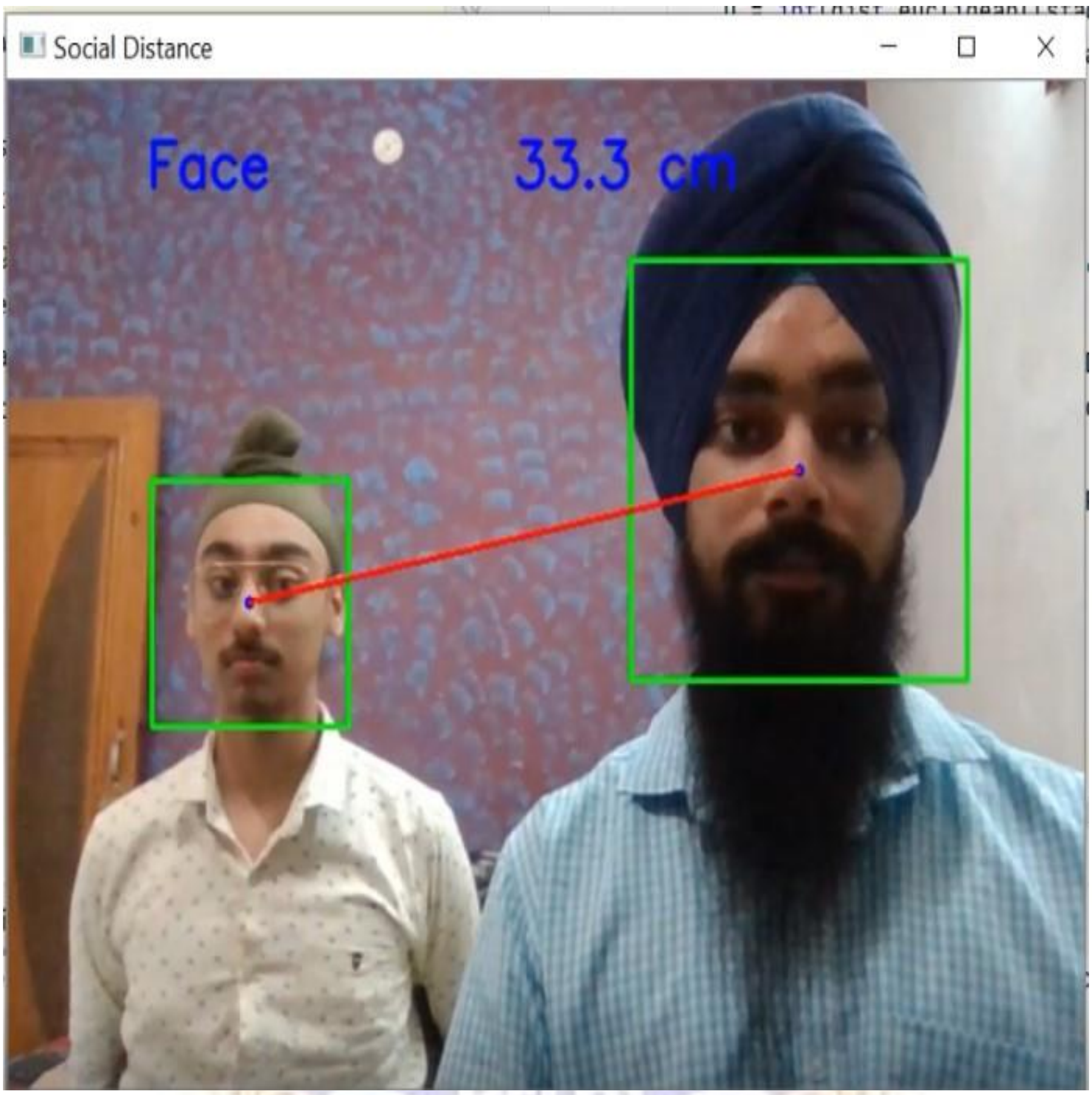
Fig 3: Features Access Page




Mask: 99.91%


No Mask: 91.98%


Social Distance
Face
24.9 cm
!!MOVE AWAY!!


Social Distance
Face
33.3 cm

APPENDIX B

SOURCE CODE

1. Welcome file

```
from tkinter import *
from PIL import Image,ImageTk
import load


class welcomes:
    def __init__(self):
        self.rt=Tk()
        self.rt.title("COVID-19 E-Surveillance")
        self.rt.config(background="#ccffcc")    #FFFFCC       #94b8b8
        self.rt.geometry("1205x550+160+100")

        self.l1=Label(self.rt)
        self.l1.place(relx=0.25, rely=0.0, height=81, width=604)
        img1=ImageTk.PhotoImage(Image.open("bank_images/welCo.png"))
        self.l1.configure(image=img1)
        self.l1.configure(background="#ccffcc")

        self.l2=Label(self.rt)
        self.l2.place(relx=0.13, rely=0.2, height=315, width=990)
        limg=ImageTk.PhotoImage(Image.open("bank_images/Let's.png"))
        self.l2.configure(image=limg)

        self.but=Button(self.rt,command=self.click)
        self.but.place(relx=0.4, rely=0.84, height=73, width=268)
        img2 = ImageTk.PhotoImage(Image.open("bank_images/cli.gif"))
        self.but.configure(image=img2)
        self.rt.mainloop()

    def click(self):
        self.rt.destroy()
        lo=load.load_class()


ob=welcomes()
```

2. Load

```
from tkinter import *
from PIL import Image,ImageTk
from tkinter.ttk import Progressbar
import threading
import buttons
import time

class load_class:
    def __init__(self):
        self.rt=Tk()
        self.rt.title("COVID-19 E-Surveillance")
        self.rt.config(background="#ccffcc")
        self.rt.geometry("1205x550+160+100")
        #self.rt.geometry("920x450+200+180")

        self.l1=Label(self.rt)
        self.l1.place(relx=0.25, rely=-0.10, height=200, width=650)
        self.l1.config(background="#ccffcc")
        img1=ImageTk.PhotoImage(Image.open("bank_images/loads.png"))
        self.l1.configure(image=img1)

        self.label1 = Label(self.rt)
        self.label1.place(relx=0.20, rely=0.29, height=200, width=850)
        self.label1.config(background="#ccffcc")
        imgq = ImageTk.PhotoImage(Image.open("bank_images/sociald.png"))
        self.label1.configure(image=imgq)

        self.v1=IntVar() #has the current value of progressbar
        self.l2=Label(self.rt)
        self.l2.place(relx=0.06, rely=0.71, height=40, width=380)
        self.l2.configure(background="#ccffcc")
        self.l2.configure(width=380)

        self.pb=Progressbar(self.rt,orient=HORIZONTAL)
        self.pb.configure(mode='determinate',maximum=100)
        self.pb.configure(variable=self.v1)

        self.pb.place(relx=0.08, rely=0.80, relwidth=0.90, relheight=0.0, height=49)
        self.pb.configure(length="650")

        tup = (101,)
        self.t1=threading.Thread(target=self.move,args=tup,name='first')
        self.t1.start()
        self.rt.after(5000,self.check)
        self.rt.mainloop()

    def move(self,a):
        for i in range(a):
            self.v1.set(i)
            self.l2.configure(text="Loading "+str(self.v1.get()) +"%")
            time.sleep(0.01)

    def check(self):
        if self.v1.get() != 100:
            self.rt.after(500,self.check)
        else:
            self.rt.destroy()
            object=buttons.button()
```

3. buttons

```
from tkinter import *
from PIL import Image,ImageTk
import social
from detect_mask_video import mask

class button:
    def __init__(self):

        self.rt=Tk()
        self.rt.title("COVID-19 E-Surveillance")
        self.rt.config(background="#ccffcc")    #FFFFCC       #94b8b8
        self.rt.geometry("1145x500+160+100")

        self.l1=Label(self.rt)
        self.l1.place(relx=0.25, rely=0.0, height=81, width=604)
        img1=ImageTk.PhotoImage(Image.open("bank_images/hebutton.png"))
        self.l1.configure(image=img1)
        self.l1.configure(background="#ccffcc")

        self.l2=Label(self.rt)
        self.l2.place(relx=0.22, rely=0.32, height=200, width=250)
        limg=ImageTk.PhotoImage(Image.open("bank_images/2.png"))
        self.l2.configure(image=limg)

        self.l21 = Label(self.rt)
        self.l21.place(relx=0.580, rely=0.32, height=200, width=250)
        limg1 = ImageTk.PhotoImage(Image.open("bank_images/Face Mask.png"))
        self.l21.configure(image=limg1)

        self.but=Button(self.rt,command=self.click)
        self.but.place(relx=0.24, rely=0.75, height=73, width=200)
        img2 = ImageTk.PhotoImage(Image.open("bank_images/wm.gif"))
        self.but.configure(image=img2)

        self.but2 = Button(self.rt, command=mask)
        self.but2.place(relx=0.60, rely=0.75, height=73, width=200)
        img3 = ImageTk.PhotoImage(Image.open("bank_images/wm.gif"))
        self.but2.configure(image=img3)
        self.rt.mainloop()

    def click(self):
        self.rt.destroy()
        lo=social.sociall()

#     def clicked(self):
#         self.rt.destroy()
#         mask()
```

4. Social

```
from tkinter import *
from PIL import Image,ImageTk
import threading
import time
import cv2
import buttons
from scipy.spatial import distance as dist

class sociall:
    def __init__(self):
        self.rt = Tk()
        self.rt.title("COVID-19 E-Surveillance")
        self.rt.config(background="#ccffcc")  # FFFFCC       #94b8b8
        self.rt.geometry("1145x500+160+100")
        cap = cv2.VideoCapture(0)
        face_model = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
        while True:
            status, photo = cap.read()
            face_cor = face_model.detectMultiScale(photo)
            l = len(face_cor)
            # str(len(face_cor))
            photo = cv2.putText(photo, " " + "Face Distance" +"  ",  (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 
0), 2,
                                cv2.LINE_AA)
            stack_x = []
            stack_y = []
            stack_x_print = []
            stack_y_print = []
            global D

            if len(face_cor) == 0:
                pass
            else:
                for i in range(0, len(face_cor)):
                    x1 = face_cor[i][0]
                    y1 = face_cor[i][1]
                    x2 = face_cor[i][0] + face_cor[i][2]
                    y2 = face_cor[i][1] + face_cor[i][3]

                    mid_x = int((x1 + x2) / 2)
                    mid_y = int((y1 + y2) / 2)
                    stack_x.append(mid_x)
                    stack_y.append(mid_y)
                    stack_x_print.append(mid_x)
                    stack_y_print.append(mid_y)

                    photo = cv2.circle(photo, (mid_x, mid_y), 3, [255, 0, 0], -1)
                    photo = cv2.rectangle(photo, (x1, y1), (x2, y2), [0, 255, 0], 2)

                if len(face_cor) == 2:
                    D = int(dist.euclidean((stack_x.pop(), stack_y.pop()), (stack_x.pop(), stack_y.pop())))
                    photo = cv2.line(photo, (stack_x_print.pop(), stack_y_print.pop()),
                                     (stack_x_print.pop(), stack_y_print.pop()), [0, 0, 255], 2)
                else:
                    D = 0

                if D < 250 and D != 0:
                    photo = cv2.putText(photo, "!!MOVE AWAY!!", (100, 100), cv2.FONT_HERSHEY_SIMPLEX, 2, [0, 0,
255], 4)

                photo = cv2.putText(photo, str(D / 10) + " cm", (300, 50), cv2.FONT_HERSHEY_SIMPLEX,
                                    1, (255, 0, 0), 2, cv2.LINE_AA)
                cv2.imshow('Social Distance', photo)
                # if D > 250 :
                #
                #     break
                if cv2.waitKey(100) == 27:
                    cv2.destroyAllWindows()
                    self.click()
                    break
        # self.but = Button(self.rt, command=self.click)
        # self.but.place(relx=0.4, rely=0.84, height=73, width=268)
        # img2 = ImageTk.PhotoImage(Image.open("bank_images/cli.gif"))
        # self.but.configure(image=img2)
        self.rt.mainloop()

    def click(self):
        self.rt.destroy()
        lo=buttons.button()
```

5. Mask

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

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

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

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

	# loop over the detections
	for i in range(0, detections.shape[2]):
		# extract the confidence (i.e., probability) associated with
		# the detection
		confidence = detections[0, 0, i, 2]

		# filter out weak detections by ensuring the confidence is
		# greater than the minimum confidence
		if confidence > 0.5:
			# compute the (x, y)-coordinates of the bounding box for
			# the object

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

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

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

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

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

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

# load our serialized face detector model from disk

def mask():
    prototxtPath = r"face_detector\deploy.prototxt"
    weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
    faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

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

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

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

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

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

            # determine the class label and color we'll use to draw
            # the bounding box and text
            label = "Mask" if mask > withoutMask else "No Mask"
            color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

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

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

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

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

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

6. Training of model

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

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

DIRECTORY = r"C:\Users\hp1\Fake mask\Face-Mask-Detection-master\Face-Mask-Detection-master\dataset"
CATEGORIES = ["with_mask", "without_mask"]

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

data = []
labels = []

for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = load_img(img_path, target_size=(224, 224)) # returns an instance of image
        image = img_to_array(image) # converts the image to numpy array
        image = preprocess_input(image)  #converts

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")

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

7. Requirements

```
tensorflow>=1.15.2
keras==2.3.1
imutils==0.5.3
numpy==1.18.2
opencv-python==4.2.0.*
matplotlib==3.2.1
scipy==1.4.1
```