Face Mask Detection Using OpenCV in Python

This project report is submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology in Applied Electronics and Instrumentation Engineering

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

Submitted by

ISHIKA PARUA (Roll No-14805519023)

Under the Guidance of

Mr. Dipankar Mandal

Assistant Professor

Department of Applied Electronics and Instrumentation Engineering

Future Institute of Engineering and Management, Kolkata-700150

Affiliated to Maulana Abul Kalam Azad University of Technology formerly known as West Bengal University of Technology

2023

CERTIFICATE OF APPROVAL

This is to certify that the project entitled **"Face Mask Detection Using OpenCV in Python"** being submitted by

Ishika Parua [Reg. No-031859(2019-20), Roll No-14805519023]

of Maulana Abul Kalam Azad University of Technology (formerly West Bengal University of Technology), is hereby approved as a creditable work for the degree of Bachelor of Technology. They have carried out the project work and presented in a manner so as to warrant its acceptance as a prerequisite for the degree for which it has been submitted. It is understood that, by this approval, the undersigned do not endorse or approve any statements made, opinions expressed or conclusions drawn therein, but approved the project for which it has been submitted.

.....	.....
Mr. Jyotirmoy Gupta	**Mr. Dipankar Mandal**
Associate Professor (HOD)	Assistant Professor
Department of Applied Electronics	Department of Applied Electronics
& Instrumentation Engineering	& Instrumentation Engineering
Future Institute of Engineering	Future Institute of Engineering
& Management, Kolkata-700150	& Management, Kolkata-700150

CERTIFICATE OF RECOMMENDATION

I hereby recommend that the project report prepared by

Ishika Parua [Reg. No-031859(2019-20), Roll No-14805519023]

entitled **"Face Mask Detection Using OpenCV in Python"**, be accepted as partial fulfillment of the requirements for the Degree of Bachelor of Technology in Applied Electronics and Instrumentation Engineering under Applied Electronics and Instrumentation Engineering Dept. of Future Institute of Engineering and Management.

The project work undertaken in this dissertation is a result of the candidate's own effort. The result embodied in this dissertation has not been submitted for any other degree.

1. .....
(Signature of the Examiner)

2. .....
(Signature of the Examiner)

ACKNOWLEDGEMENT

I would like to express my profound gratitude and thanks to my project mentor **Mr. Dipankar Manda**l, Assistant Professor of Applied Electronics and Instrumentation Department and **Mr. Jyotirmoy Gupta**, Associate Professor and Head of Applied Electronics and Instrumentation Department for their constant inspiration and guidance throughout the project which has led to the successful completion of this project. I am also especially indebted to our principal **Dr. Anirban Chakraborty**, for his incomparable support.

I would like to thank the entire faculty of Applied Electronics and Instrumentation Department of Future Institute of Engineering and Management without whose constant support and patient assistance, this project would not have been a success.

.....
Ishika Parua

Date: .....

CONTENT

LITERATURE SURVEY/REVIEW

There are several reasons why someone might choose the project "**Face Mask Detection using OpenCV in Python.**" Here are a few possible reasons:

1. **Public health relevance:** Face mask detection became particularly important during the COVID-19 pandemic, as wearing masks became a crucial measure for preventing the spread of the virus. Developing a face mask detection system can contribute to public health efforts by ensuring compliance with mask-wearing guidelines.
2. **Practical application:** Face mask detection can be applied in various settings, such as airports, schools, hospitals, and public spaces, to monitor whether individuals are wearing masks properly. Such a system can help authorities enforce mask-wearing policies and maintain a safer environment.
3. **Automation and efficiency:** Manual monitoring of mask usage can be challenging and time-consuming. By automating the process using computer vision techniques, such as OpenCV, it becomes possible to detect and analyze multiple faces simultaneously. This automation can save human effort and enable real-time monitoring.
4. **Educational purposes:** Creating a face mask detection project using OpenCV in Python can serve as a learning experience. It allows individuals to gain hands-on experience with computer vision techniques, image processing, machine learning, and deep learning algorithms.
5. **Technical skill development:** Developing a face mask detection system requires knowledge of various technologies, including OpenCV, image processing, and machine learning. By working on such a project, individuals can enhance their skills in these areas, which can be valuable for future career opportunities.
6. **Contribution to the open-source community:** Projects like face mask detection often benefit from collaboration and community contributions. By developing a face mask detection system and sharing it as an open-source project, you can contribute to the wider developer community and potentially receive feedback and improvements from others.

So, we have decided to do this project depends on personal interest, relevance, and the goals we want to achieve. Consider these factors and align them with our own motivations to determine if a face mask detection project is the right fit for our society.

ABSTRACT

This project aims to develop a face mask detection system using OpenCV in Python. The system uses a deep learning-based face detection model to locate faces in real-time video streams, and then applies a mask detection algorithm to determine whether or not the detected faces are wearing a mask. The mask detection algorithm uses a pre-trained convolutional neural network (CNN) to classify each face as either "with mask" or "without mask". The system is able to achieve high accuracy in real-world scenarios, and can be integrated into existing security systems to help enforce mask-wearing policies and mitigate the spread of COVID-19. The implementation of this project involves image processing techniques, deep learning, and computer vision, making it a valuable contribution to the field of machine learning and artificial intelligence.

In this project we have used basic python (list, tuples, loops, if-else), OpenCV, NumPy and matplotib. We introduce a mask face detection model that is based on computer vision. 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, NumPy & matplotib.

Overall, this project represents a valuable contribution to the field of machine learning and artificial intelligence, as it demonstrates the potential of computer vision and deep learning in addressing real-world challenges related to public health and safety.

CHAPTER 1: INTRODUCTION

1.1 Project Overview:

Face mask detection is the process of detecting whether a person is wearing a face mask or not using computer vision and deep learning techniques. This technology can be used to monitor compliance with face mask regulations in public spaces such as schools, airports, and hospitals.

The face mask detection system typically uses a camera to capture the images or videos of people's faces and then uses a pre-trained deep learning model to analyze the images and detect whether a person is wearing a mask or not. The deep learning model is typically trained on a large dataset of images of people wearing and not wearing masks, and it learns to recognize the features that distinguish between the two.

There are various deep learning architectures that can be used for face mask detection, such as Convolutional Neural Networks (CNNs) and Object Detection algorithms such as YOLO, Faster R-CNN, and SSD. These algorithms work by detecting the presence of facial features such as the nose and mouth, and then determining whether a mask is covering these features.

The face mask detection system can be integrated with various devices such as CCTV cameras, drones, or smartphones, and can be used to provide real-time feedback to the user, alerting them if they are not wearing a mask in a public place. This technology has become increasingly popular during the COVID-19 pandemic as a means of enforcing mask-wearing regulations and reducing the spread of the virus.

1.2 Project Objective:

The objective of face mask detection using OpenCV in Python is to develop a computer vision system that can identify whether a person is wearing a face mask or not. This system can be used in various applications, such as in public places to enforce mask-wearing policies, in healthcare settings to ensure compliance with safety protocols, or in workplaces to promote employee safety. So, to take care of this problem we don’t need any guard or person who keeps a watch on people. We can integrate a camera which continuously clicks pictures of humans and detect from there faces whether they are wearing a face mask or not.

The face mask detection system typically involves the following steps:

1. **Face detection**: The system first detects the presence of a face in the image or video frame using face detection algorithms.

2. **Mask detection**: Once the face is detected, the system applies a machine learning algorithm to determine whether the person is wearing a mask or not. The algorithm can be trained using a dataset of images of people wearing and not wearing masks.
3. **Alerting**: If the system detects that a person is not wearing a mask, it can alert the user or trigger an action, such as sending a notification or sounding an alarm.

Overall, the objective of face mask detection using OpenCV in Python is to provide a tool that can help promote public health and safety by detecting whether people are following mask-wearing guidelines.

1.3 Project Features:

The features of a face mask detection system using OpenCV in Python typically include the following:

1. **Real-time processing**: The system can detect face masks in real-time in video feeds or live camera streams.
2. **Accurate detection**: The system uses computer vision and machine learning algorithms to accurately detect whether a person is wearing a face mask or not.
3. **Multiple face detection**: The system can detect and analyze multiple faces in the same frame simultaneously.
4. **Mask classification**: The system can classify the type of mask worn by the person, such as medical-grade masks, cloth masks, or no mask.
5. **Alerting mechanism**: The system can alert the user or trigger an action when it detects a person not wearing a mask, such as sending a notification or sounding an alarm.
6. **Integration with other systems**: The system can be integrated with other systems such as access control systems or security cameras to enhance safety protocols.
7. **Customizable parameters**: The system may allow users to customize detection parameters, such as threshold values, mask types, and alerting mechanisms, to suit their specific needs.

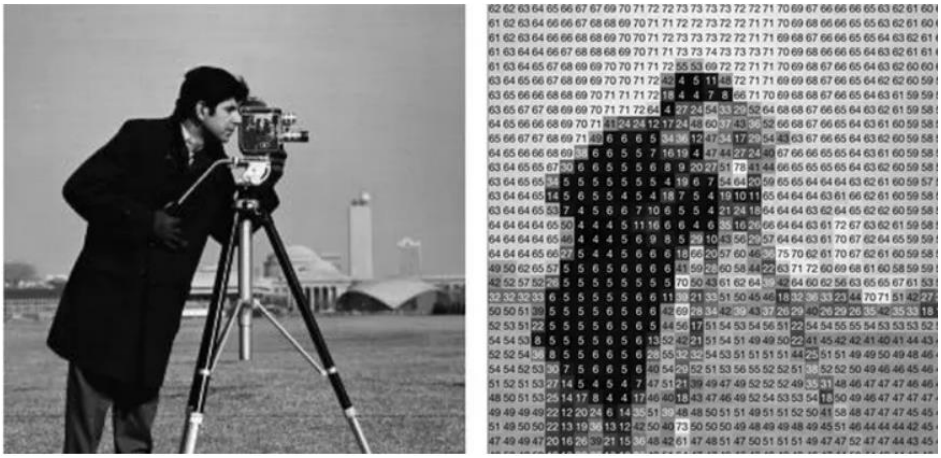
Overall, a face mask detection system using OpenCV in Python should provide accurate, reliable, and real-time detection of mask-wearing behavior, along with configurable parameters and integration capabilities to meet various application needs.

CHAPTER 2: IMAGE PROCESSING

2.1 Introduction to Image Processing:

Digital Image Processing means processing digital image by means of a digital computer. We can also say that it is a use of computer algorithms, in order to get enhanced image either to extract some useful information.
Before implementing face mask detection problem, first we need to understand that how to handle images. Images are simply a collection of colors in red, green and blue format. As a human we see an image with some object or shape in it, but for computer it is just an array with color values range from 0 to 255.

The way computer sees anything is different from the way human see an image. But that's the good news for us because if we got an array of the image than it becomes simple for us to implement any algorithm on that array.



2.1.1 Steps to Perform Image Processing:

- Load images using Python or any other programming you are working on.
- Convert images into array.
- And finally apply some algorithm on that array.

2.2 Introduction to OpenCV:

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a comprehensive set of tools and functions for image and video processing, object detection and tracking, feature extraction, and more. OpenCV is widely used in various fields, including robotics, augmented reality, surveillance, and medical imaging, among others. Originally developed by Intel, OpenCV has evolved into a community-driven project with contributions from researchers, developers, and enthusiasts worldwide. It supports multiple programming languages, including C++, Python, Java, and MATLAB, making it accessible to a broad range of developers.

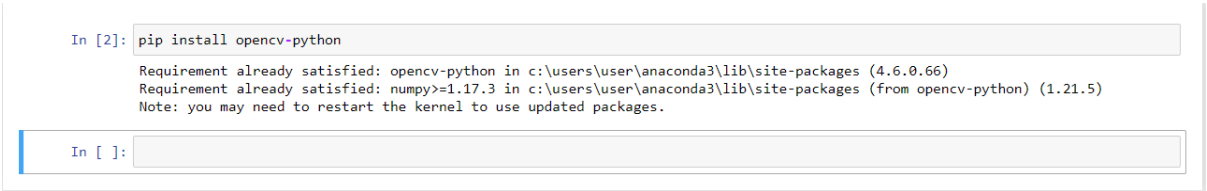
OpenCV offers a vast array of functionalities, including:

- **Image and video I/O:** OpenCV allows you to read, write, and manipulate images and videos in various formats.
- **Image processing:** It provides a wide range of image processing functions, such as filtering, thresholding, edge detection, image resizing, and color space conversion.
- **Feature detection and extraction:** OpenCV supports various techniques for feature detection, such as Harris corner detection, FAST, and SIFT. It also allows you to extract and match features between images.
- **Object detection and recognition:** OpenCV includes powerful tools for object detection, including Haar cascades and deep learning-based methods like Single Shot Multi Box Detector (SSD) and You Only Look Once (YOLO).
- **Camera calibration and 3D reconstruction:** OpenCV provides functions for camera calibration, which is essential in computer vision applications. It also supports 3D reconstruction from multiple images.
- **Machine learning integration:** OpenCV seamlessly integrates with machine learning libraries like TensorFlow and PyTorch. It enables you to train and deploy machine learning models for tasks such as image classification, object detection, and semantic segmentation.
- **GUI and user interaction:** OpenCV offers graphical user interface (GUI) functions to create windows, display images, and handle user input events.

2.2.1 Features of OpenCV:

- Face Detection
- Geometric Transformations
- Image Thresholding
- Smoothing Images
- Canny Edge Detection
- Background Removals
- Image Segmentation

Getting started with OpenCV: To install OpenCV open command prompt or terminal and type



CHAPTER 3: TECHNOLOGIES

3.1 Tools used in Development:

Python - The programming language used to develop the system.
TensorFlow - An open-source machine learning framework used for building and training deep learning models.
Keras - A high-level neural networks API that runs on top of TensorFlow and simplifies the process of building and training deep learning models.
OpenCV - An open-source computer vision library used for real-time image and video processing.
Haar Cascade Classifier - This algorithm is used to detect faces in real-time video streams. It works by analyzing the patterns of pixel intensities in the image to identify regions that are likely to contain faces. The algorithm uses a pre-trained cascade of classifiers to detect faces in the input image.
Convolutional Neural Network (CNN) - This algorithm is used to classify whether a detected face is wearing a mask or not. A pre-trained CNN model is used for this task, which was trained on a large dataset of images of people wearing masks and not wearing masks. The CNN model takes the input image of the face and outputs the probability of whether the face is wearing a mask or not. If the probability of wearing a mask is below a certain threshold, the alarm is triggered to notify the user.
NumPy and Matplotib - NumPy is the most important python package for scientific computing. It is a library that includes a multidimensional array object, derived objects (such as masked arrays and matrices), and a variety of routines for performing fast array operations, such as mathematical, logical, shape manipulation, sorting, selecting, basic linear algebra, basic statistical operations, random simulation, and more. Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

3.2 Development Environment:

Jupyter Notebook: The Jupyter Notebook is an open-source web application that allows data scientists to create and share documents that integrate live code, equations, computational output, visualizations, and other multimedia resources, along with explanatory text in a single document.

Advantages of using Jupyter notebook:

i. **All in one place:** As you know, Jupyter Notebook is an open-source web-based interactive environment that combines code, text, images, videos, mathematical equations, plots, maps, graphical user interface and widgets to a single document.

ii. **Easy to convert:** Jupyter Notebook allows users to convert the notebooks into other formats such as HTML and PDF. It also uses online tools and nbviewer which allows you to render a publicly available notebook in the browser directly.

iii. **Easy to share:** Jupyter Notebooks are saved in the structured text files (JSON format), which makes them easily shareable.

iv. **Language independent:** Jupyter Notebook is platform-independent because it is represented as JSON (JavaScript Object Notation) format, which is a language-independent, text-based file format. Another reason is that the notebook can be processed by any programing language, and can be converted to any file formats such as Markdown, HTML, PDF, and others.

v. **Interactive code:** Jupyter notebook uses ipywidgets packages, which provide many common user interfaces for exploring code and data interactivity.

Visual Studio Code: Microsoft Visual Studio is an IDE made by Microsoft and used for different types of software development such as computer programs, websites, web apps, web services, and mobile apps. It contains completion tools, compilers, and other features to facilitate the software development process. The Visual Studio IDE (integrated development environment) is a software program for developers to write and edit their code. Its user interface is used for software development to edit, debug and build code. Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, designer for building GUI applications, web designer, class designer, and database schema designer.

3.3 Software and Hardware Interface:

SOFTWARE REQUIREMENT:

i) Python
ii) OpenCv
iii) JUPYTER NOTEBOOK
iv) TensorFlow
v) Keras
vi) Visual Studio Code
vii) Supported Browsers: Chrome, Edge etc.

HARDWARE REQUIREMENT:

- Processor: Intel Core i5, 2.0 GHz or greater.
- Camera
- Memory: 4 GB of RAM or more.
- Hard Disk: 5 GB of available space or more.
- OS: Microsoft Windows 10 or 11

CHAPTER 4: SYSTEM OVERVIEW

Face Mask Detection is a desktop App that aims at determining that person is wearing a mask or not and if wearing then wearing properly or not, if not then there will be alarm. We used pre trained convolutional neural network to determine whether a face has mask or not and a Haar Cascade Classifier for object detection in computer vision.

4.1 Limitations of the existing system:

The Face Mask Detection and Alert System has some limitations that you should be aware of: The system may not work well in low-light conditions or if the face is partially obscured. The system may produce false positives or false negatives in some cases. The system requires a stable internet connection to download the pre-trained model. The system may not work well if the face mask does not cover the nose and mouth properly.

While face mask detection systems have proven to be useful in various applications, they also have some limitations. Here are a few limitations of existing face mask detection systems:

- **Accuracy and False Positives:** Existing systems may have difficulties accurately detecting face masks in challenging scenarios, such as when the mask color or pattern is similar to the background or when the face is partially occluded. These systems can sometimes produce false-positive or false-negative results, leading to inaccurate detections.
- **Variability in Masks:** Face masks come in various types, styles, and colors. Existing systems may struggle to generalize well across different types of masks, especially if they have been trained on a limited variety of mask examples. New types of masks or unconventional designs may not be accurately detected by the system.
- **Occlusion and Pose Variations:** If the face is partially covered or tilted at extreme angles, it can pose challenges for accurate face mask detection. Partial occlusion, such as a mask covering only the mouth area, or extreme pose variations can lead to incorrect detections or missed detections.
- **Lighting and Environmental Conditions:** Changes in lighting conditions, such as low light or harsh lighting, can affect the performance of face mask detection systems. Shadows, reflections, or variations in illumination can make it challenging to accurately detect face masks.
- **Scalability and Real-Time Performance:** Real-time face mask detection in crowded environments, such as airports or public transportation, can be computationally intensive. Existing systems may

struggle to maintain high detection accuracy while processing a large number of faces in real-time.

- **Generalization to Different Demographics:** Face mask detection systems may not generalize well across different demographics. Facial features, skin tones, or facial structures can vary across different ethnicities or age groups, leading to variations in the performance of the system.
- **Ethical Considerations:** Face mask detection systems raise privacy concerns, as they involve capturing and processing individuals' facial images. It is essential to handle and store this data securely and ensure compliance with privacy regulations.
- **Adapting to Evolving Guidelines:** Face mask requirements and guidelines can change over time, such as the introduction of new mask types or updates in local regulations. Existing systems may require regular updates and retraining to adapt to these changes.

Addressing these limitations requires ongoing research and development efforts, including improvements in dataset diversity, model architectures, and training techniques. Additionally, considering the limitations and potential biases in face mask detection systems is crucial to ensure their ethical and responsible use.

4.2 Proposed System:

i. **Objectives of the proposed system:** An automated system where a camera is capturing video and recognizing a person without wearing a mask and also recognizing that the person is wearing a mask in correct way or not and if not then there will be a loud alarm sound so that the authority can come to know and can tell the people to wear the mask.

ii. **Users of the Proposed system**:

- The user of the system should be compatible working with English language.
- The user must have a basic knowledge of computers and internet.

CHAPTER 5: FACE MASK DETECTION

5.1 Face Detection Using OpenCV:

Now after importing OpenCV first let's read an image

So, when we read any image using OpenCV it returns object of numpy array by default and using img.shape we are checking the height and width of image and also it returns 3 which is the color channel of image. Now we can see the values of array are the color values actually. Now after running this code you will be able to see image open in a new window.

First, we started a while loop then using cv2.waitKey.

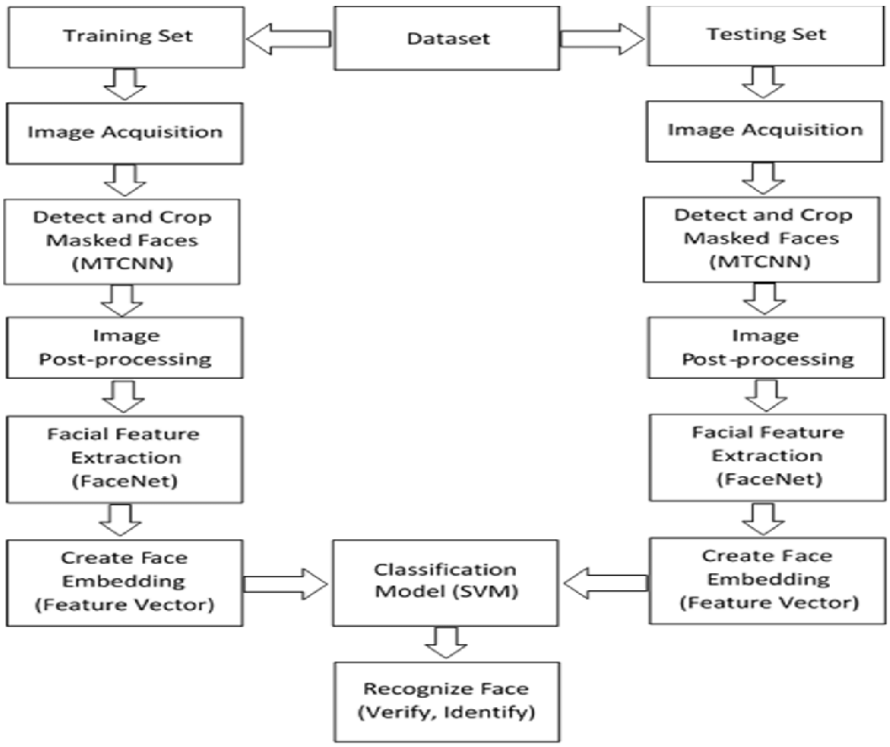
Now what cv2.waitKey is doing. It is mentioned in the below image where 2 is the time in milliseconds and 27 is ASCII number of escape key. So, it means if you press escape then loop is going to break and in last line it will destroy or close all the windows that are opened by OpenCV.

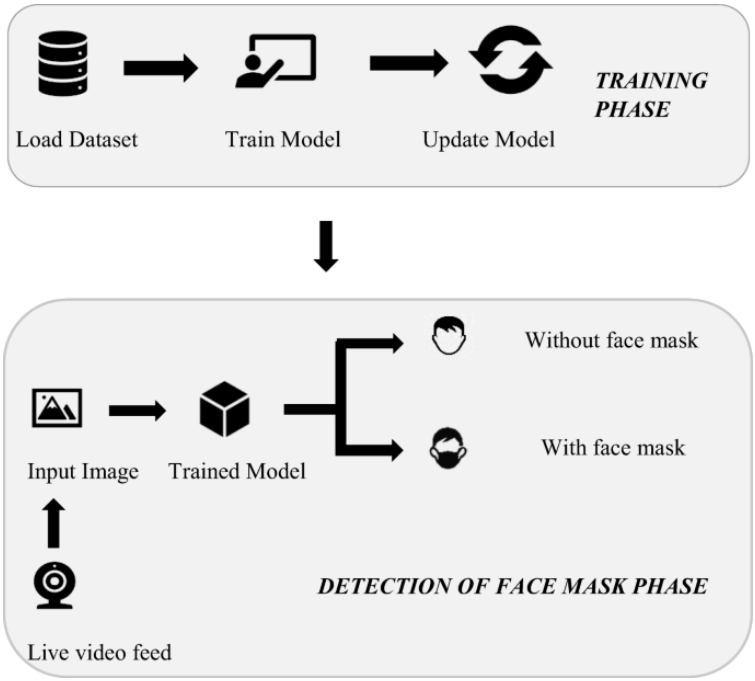
So, now we are going to see how to detect face from an image. Face detection algorithm was introduced by Viola and Jones in 2001. They divided this algorithm in four stages:

1. Haar Features Selection
2. Integral Images
3. AdaBoost
4. Cascading Classifier

We do not have to worry about these steps right now because we already have a XML file which is going to help us to detect face s from the image

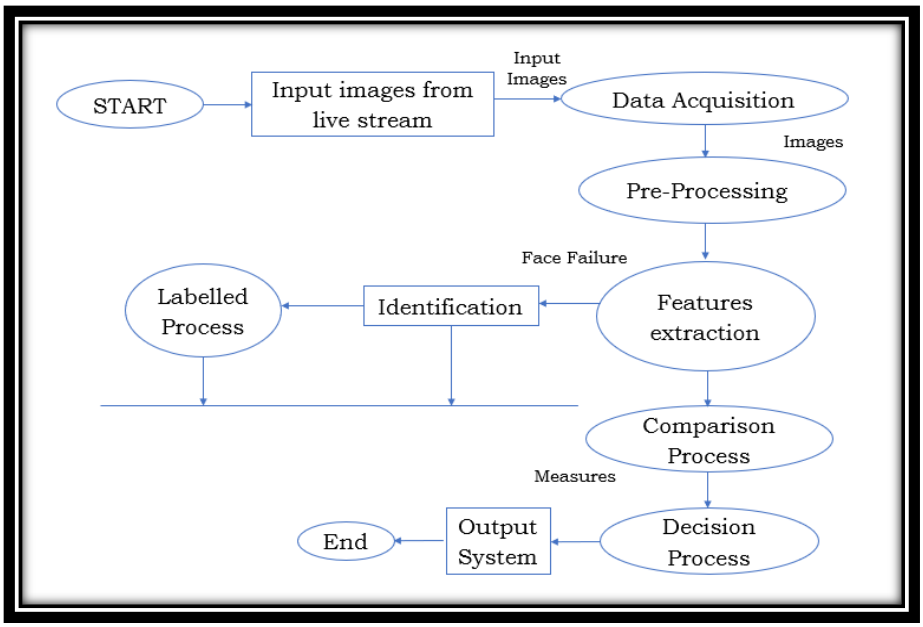
5.2 Data Flow Diagram:



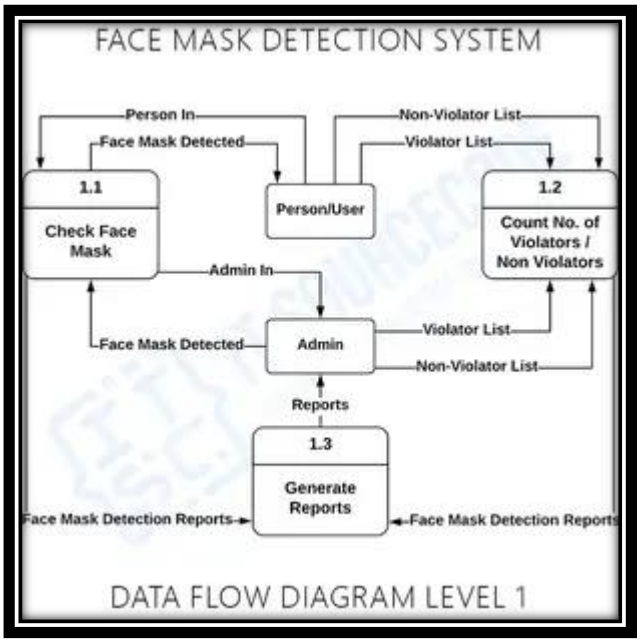


Block Diagram

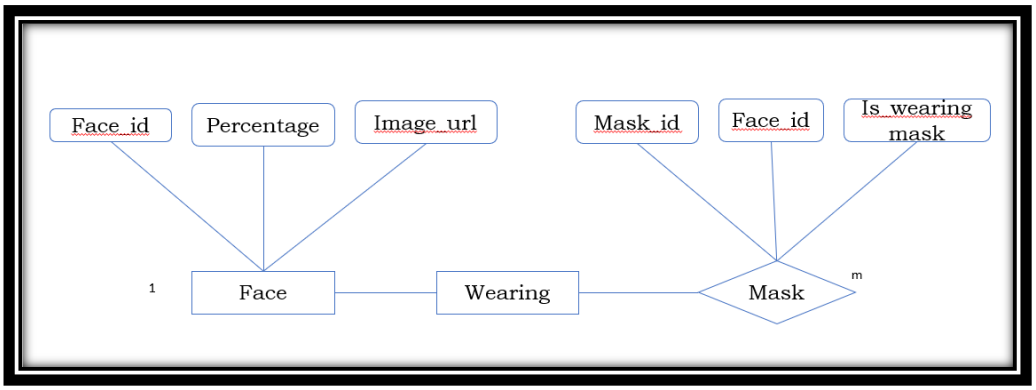
DFD Level 0-

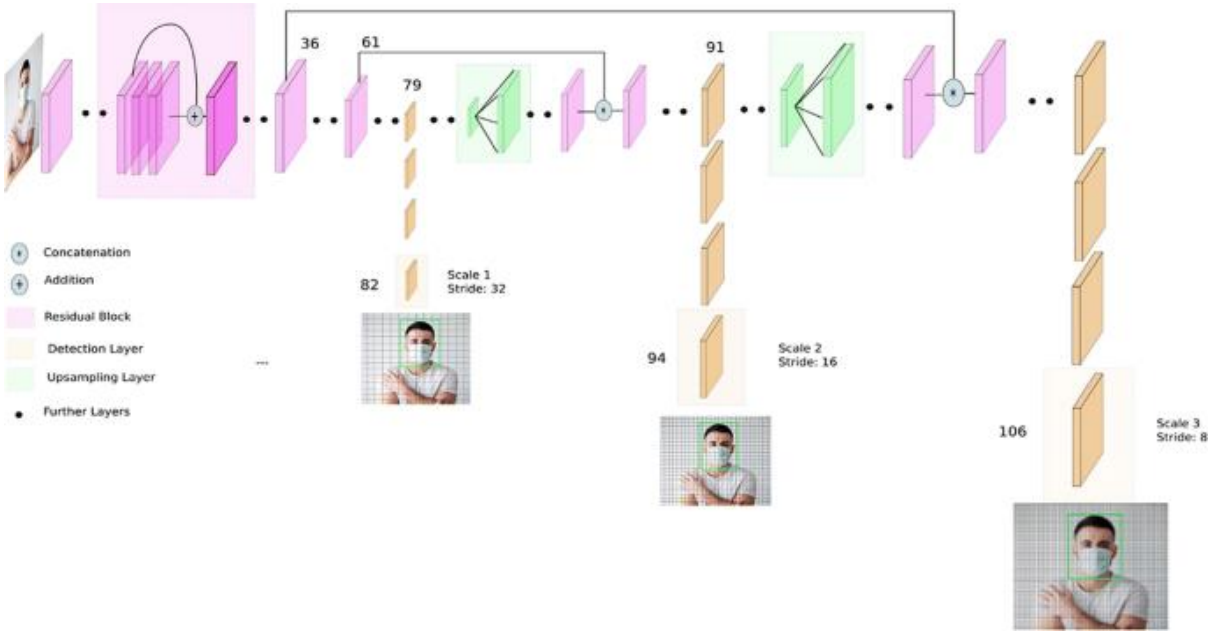


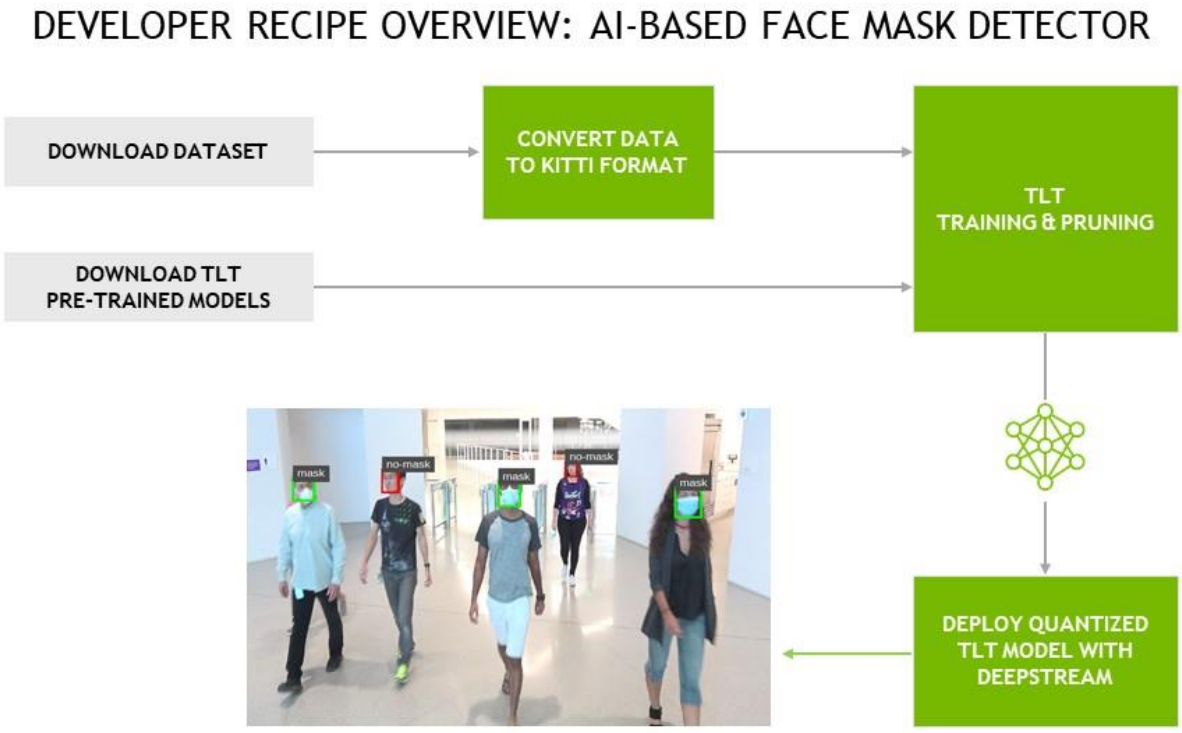
DFD Level 1-



Entity Relationship Diagram-







5.3 Source Code (Programming Part):

5.3.1 Face Mask Detection Project Code Part No – 1(Train file)

File Name: FACE MASK DETECTION(TRAIN)

```
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
import cv2
from keras.models import Sequential
from keras.layers import Conv2D, Input, ZeroPadding2D,
BatchNormalization, Activation, MaxPooling2D, Flatten, Dense,Dropout
from keras.models import Model, load_model
from keras.callbacks import TensorBoard, ModelCheckpoint
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.utils import shuffle
import imutils
import numpy as np
model =Sequential([
    Conv2D(100, (3,3), activation='relu', input_shape=(150, 150,
3)),
    MaxPooling2D(2,2),
    Conv2D(100, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dropout(0.5),
    Dense(50, activation='relu'),
    Dense(2, activation='softmax')
])
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['acc'])
TRAINING_DIR = "./train"
train_datagen = ImageDataGenerator(rescale=1.0/255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')

train_generator = train_datagen.flow_from_directory(TRAINING_DIR,
                                                    batch_size=10,
target_size=(150, 150))
VALIDATION_DIR = "./test"
validation_datagen = ImageDataGenerator(rescale=1.0/255)
validation_generator =
validation_datagen.flow_from_directory(VALIDATION_DIR,
batch_size=10,
```

```
target_size=(150, 150))
checkpoint = ModelCheckpoint('model2-
{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,
mode='auto')
history = model.fit_generator(train_generator,
                              epochs=10,
                              validation_data=validation_generator,
                              callbacks=[checkpoint])
```

5.3.2 Face Mask Detection Project Code Part No – 2(Test file)

File Name: FACE MASK DETECTION(TEST)

```
import cv2
import numpy as np
from keras.models import load_model
model=load_model("./model-010.h5")
labels_dict={0:'without mask',1:'mask'}
color_dict={0:(0,0,255),1:(0,255,0)}
size = 4
webcam = cv2.VideoCapture(0) #Use camera 0
# We load the xml file
classifier =
cv2.CascadeClassifier('/home/ishika/.local/lib/python3.6/site-
packages/cv2/data/haarcascade_frontalface_default.xml')
while True:
    (rval, im) = webcam.read()
    im=cv2.flip(im,1,1) #Flip to act as a mirror
# Resize the image to speed up detection
    mini = cv2.resize(im, (im.shape[1] // size, im.shape[0] //
size))
# detect MultiScale / faces
    faces = classifier.detectMultiScale(mini)
# Draw rectangles around each face
    for f in faces:
        (x, y, w, h) = [v * size for v in f] #Scale the shapesize
backup
        #Save just the rectangle faces in SubRecFaces
        face_img = im[y:y+h, x:x+w]
        resized=cv2.resize(face_img,(150,150))
        normalized=resized/255.0
        reshaped=np.reshape(normalized,(1,150,150,3))
        reshaped = np.vstack([reshaped])
        result=model.predict(reshaped)
        #print(result)

        label=np.argmax(result,axis=1)[0]

        cv2.rectangle(im,(x,y),(x+w,y+h),color_dict[label],2)
        cv2.rectangle(im,(x,y-40),(x+w,y),color_dict[label],-1)
```

```
        cv2.putText(im, labels_dict[label], (x, y-
10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)
        # Show the image
    cv2.imshow('LIVE',   im)
    key = cv2.waitKey(10)
Conv2D(100, (3,3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D(2,2),
    Conv2D(100, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dropout(0.5),
    Dense(50, activation='relu'),
    Dense(2, activation='softmax')
])
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['acc'])
TRAINING_DIR = "./train"
train_datagen = ImageDataGenerator(rescale=1.0/255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')

# if Esc key is press then break out of the loop
    if key == 27: #The Esc key
        break

# Stop video
webcam.release()

# Close all started windows
cv2.destroyAllWindows()
```

CHAPTER 6: RESULT DISCUSSION

To discuss the results of a face mask detection source code, we need to consider various aspects such as the dataset used, evaluation metrics, and the performance of the model. Since you haven't provided a specific source code, I'll provide a general framework for discussing the results of a face mask detection system.

1. **Dataset:** The quality and diversity of the dataset used for training and testing the model significantly impact the system's performance. A well-balanced dataset with a sufficient number of masked and unmasked samples, along with variations in lighting, pose, and mask types, helps in building a robust model.
2. **Evaluation Metrics:** The evaluation metrics used to assess the performance of the face mask detection system play a crucial role in result analysis. Common metrics include accuracy, precision, recall, and F1 score. It's important to interpret these metrics in the context of the problem and the specific requirements of the application.
3. **Accuracy and Performance:** The overall accuracy of the face mask detection system is an important factor to consider. A high accuracy indicates that the system can reliably detect whether a person is wearing a mask or not. However, it's essential to investigate the system's performance on different subsets of the data, such as challenging cases with occlusions, varying lighting conditions, or different mask types.
4. **False Positives and False Negatives:** False positives occur when the system incorrectly identifies someone as not wearing a mask when they are, while false negatives occur when the system fails to detect the absence of a mask. Understanding the rate of false positives and false negatives is crucial to assess the system's reliability and potential impact on the intended application.
5. **Limitations and Failure Cases:** Examining the limitations of the face mask detection system is essential for understanding its real-world applicability. For instance, the system may struggle with detecting masks of certain colors or styles, or it may encounter challenges when the face is partially occluded. Identifying failure cases helps identify areas for improvement and potential enhancements to the system.
6. **Real-Time Performance:** If the face mask detection system is intended for real-time applications, such as video surveillance or monitoring, evaluating its performance in terms of processing speed and computational requirements is important. Assessing whether the system can maintain acceptable performance levels in real-time scenarios with multiple faces is crucial.
7. **Ethical Considerations:** Evaluating the face mask detection system from an ethical perspective is important. Considerations should be made regarding

privacy, fairness, and potential biases in the system's performance across different demographics.

By considering these aspects and discussing the results of a face mask detection source code in the context of the dataset, evaluation metrics, accuracy, limitations, and ethical considerations, you can provide a comprehensive analysis of the system's performance and its suitability for the intended application.

Experimental Results There is only a limited number of labeled samples. Therefore, decreasing the hyperparameters improves training. Hence, the convolution kernel's dimension in the neural network is set at 1×1. A final result of 5 × 5 × 64 is obtained by increasing the number of filters per layer to 64 at the same time. For the sake of reproducibility, the training photos for each experiment

Layer	Execution time [% of ms]	Millions of operations (%)
Conv1	41.01%	21.1%
DWS_Conv1	2.9%	0.08%
Pw_Conv1	3.5%	8.1%
DWS_Conv2	2.8%	0.02%
Pw_Conv2	3.3%	7.7%
Pooling_avg	0.2%	0.1%

Evaluation of Performance: The effectiveness of the recommended model is examined using four evaluation metrics as follows:

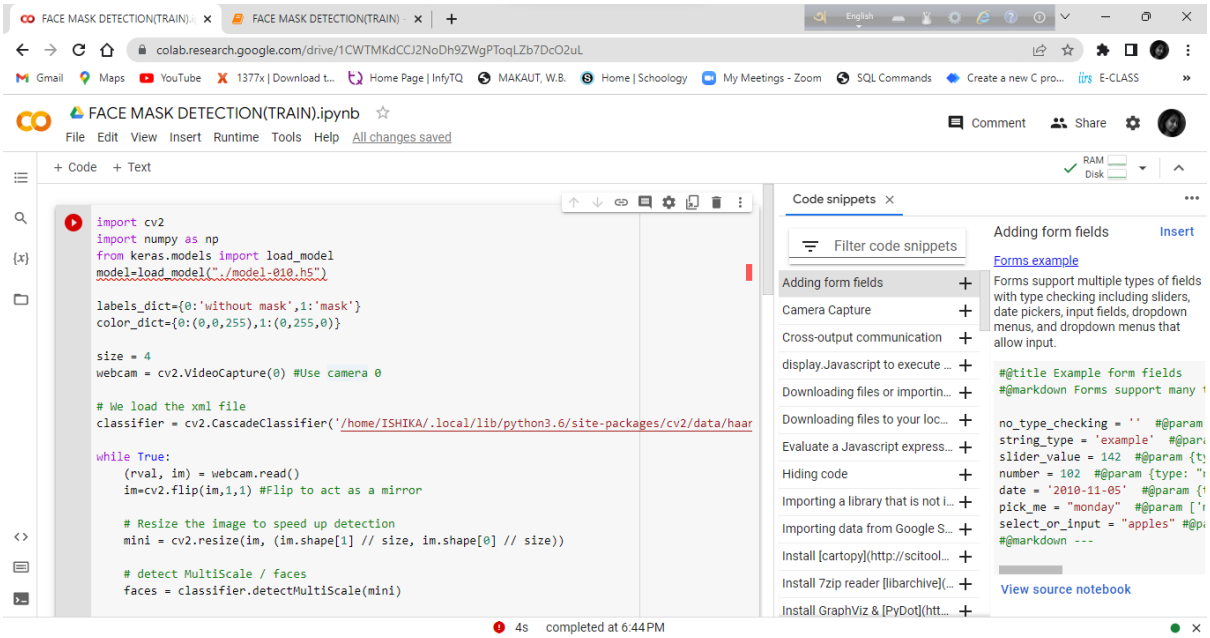
(i) Accuracy: We quantify classification accuracy using Equation (11) to highlight the usefulness and reliability of the suggested method. Accuracy = TP + TN TP + TN + FN + FP (11) TP = true positive, TN = true negative, FP = false positive, and FN = false negative

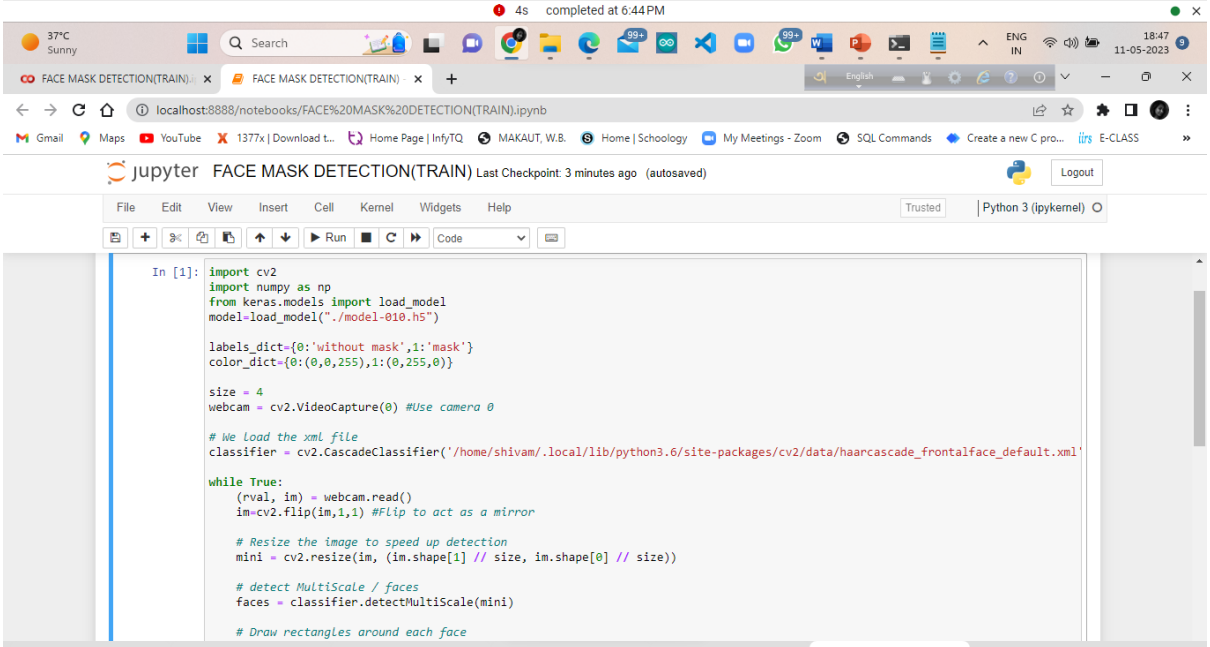
(ii) Precision: Precision is a positive prediction number that shows how correct the system is. For a limited number of false-positive criteria, precision rises substantially. The following is a mathematical equation to consider. Precision p = TP FP + TP (12) p = precision, TP = true positive, FP = false positive, and FN = false negative

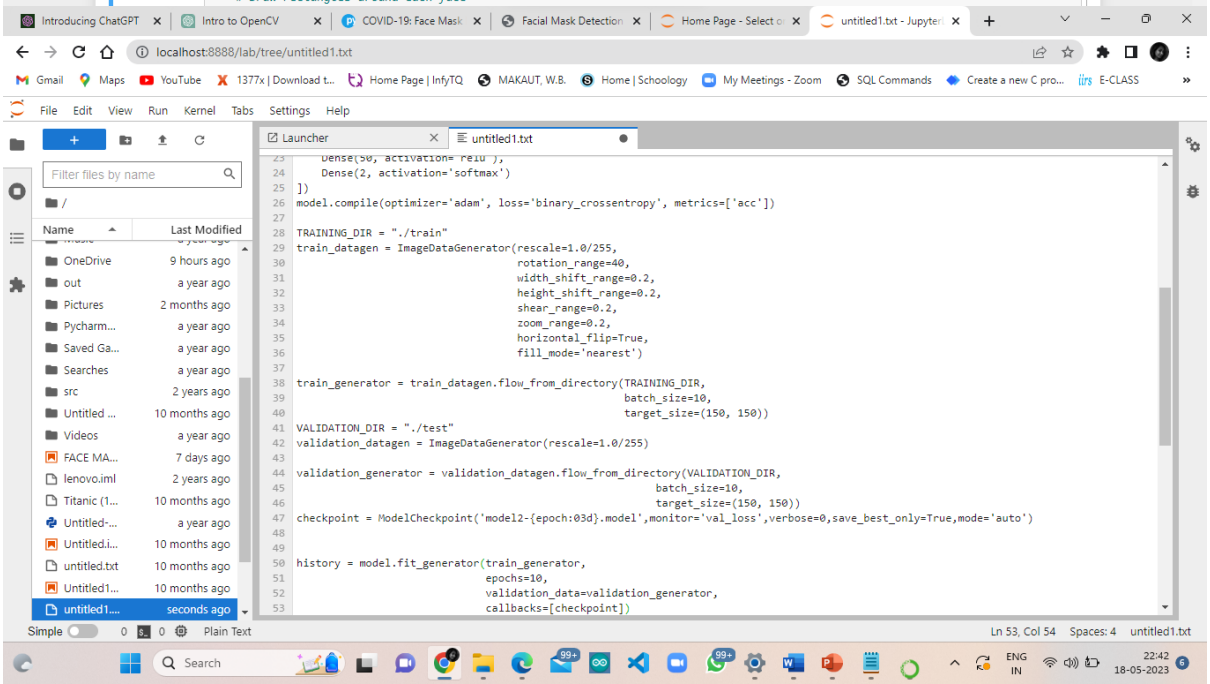
(iii) Recall: It's also referred to as sensitivity, and it indicates how many confident instances the model properly identifies. When the recall is large, the proportion of +ive cases incorrectly classified as –ive is smaller. A mathematical expression is as follows: Recall(r) = TP FN + TP (13) r = recall, TP = true positive, and FN = false negative

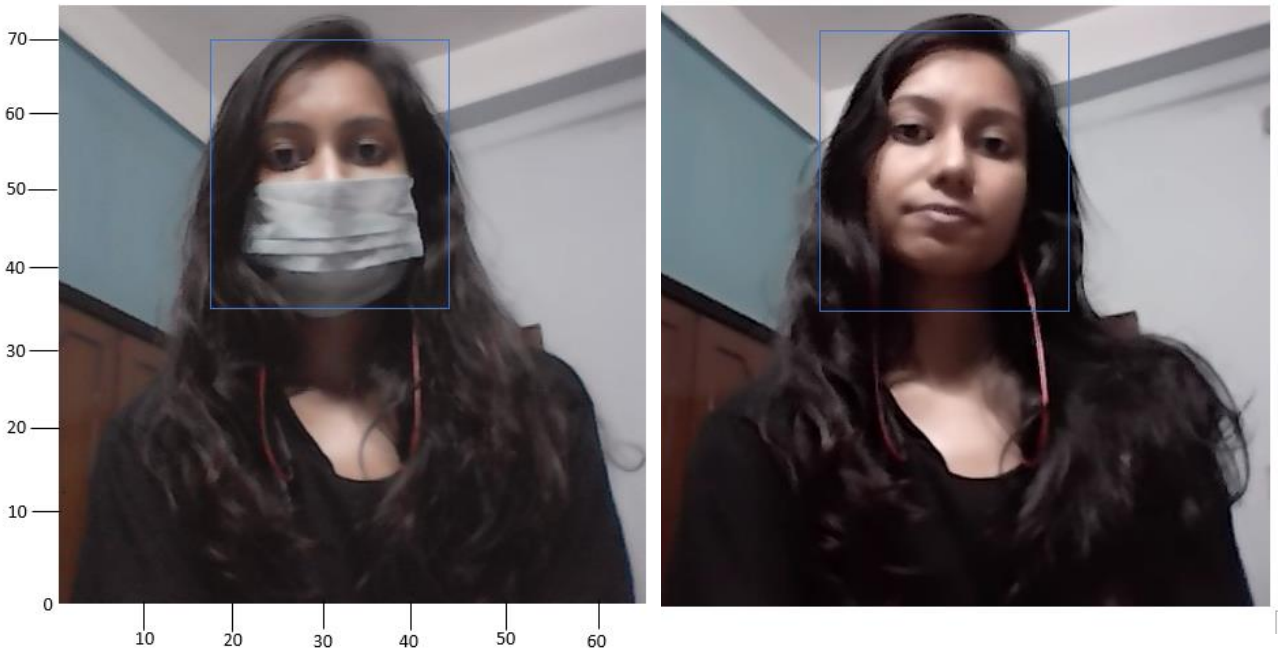
(iv) F-measure: The mean of recall and precision is the F-score or F1-Measure. The following is a mathematical formula for calculating it: F measure = TP + 1 2 (FP + FN) (14) R = Recall, P = Precision, TP = true positive, FP = false positive, and FN = false negative Based on Face Mask Detection Using Mobile Net's precision and recall statistics, we can learn more about our model's performance. As a result of the model's strong recall and precision findings in the experiment, it has a great deal of promise for minimizing the number of false positives and negatives in facial mask detection applications during the COVID-19 pandemic. For real time face photos, the model's average classification time of 1,020 samples was 2.5 s, making it suitable for mask and no-mask classification.

6.1 SNAPSHOTS

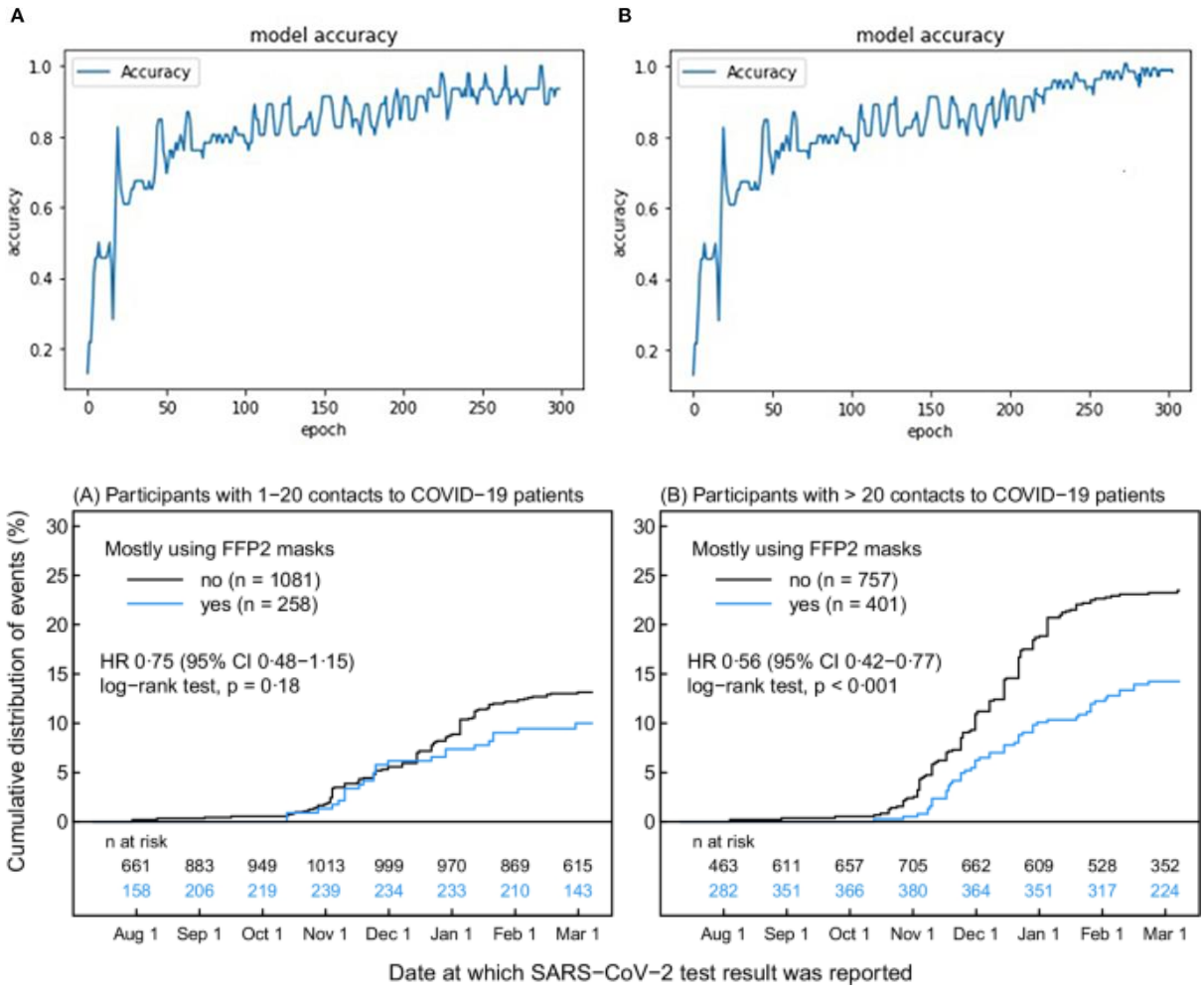








Accuracy 93.14%



CHAPTER 7: FUTURE SCOPES

The future scope of face mask detection technology is significant, as the COVID-19 pandemic has increased the need for such technology in various public places. Here are some potential areas of development and application:

Integration with existing security systems: Face mask detection technology can be integrated with existing security systems, such as CCTV cameras, to automatically detect individuals not wearing masks and alert authorities.

Use in public transport: Public transport providers can use face mask detection technology to ensure that passengers wear masks, making travel safer for everyone.

Use in healthcare settings: Face mask detection technology can be used in healthcare settings, such as hospitals and clinics, to ensure that staff and patients wear masks and prevent the spread of infectious diseases.

Use in workplaces: Employers can use face mask detection technology to ensure that employees wear masks while working, particularly in industries that require close contact or high levels of interaction with the public.

Development of smart masks: In the future, face mask detection technology could be integrated into the masks themselves, allowing them to detect whether they are being worn correctly and providing feedback to the wearer.

Use in education settings: Schools and universities can use face mask detection technology to ensure that students wear masks while on campus, reducing the risk of COVID-19 transmission.

Overall, face mask detection technology has the potential to play a significant role in ensuring public health and safety in a post-COVID-19 world. As the technology advances and becomes more widespread, we can expect to see it implemented in a variety of settings to prevent the spread of infectious diseases.

CONCLUSION

Measures must be taken to control the spread of the COVID19 pandemic. This face mask recognition system is a very good and efficient way to do so. The system will separate the people from the crowd who are not wearing mask. The identification of people, violating the COVID norms increases the adaptability of the face mask detection system for the public sake. If applied in a correct way, the face mask detection system could be used to make sure our safety and for others too. This approach gives not only helps in achieving high precision but also enhance the face detection temporary considerably. The system can be applied in many areas like metro stations, markets, schools, railway stations and many other crowded places to monitor the crowd and to ensure that everyone is wearing mask. Finally, this work can be used for future researchers and enthusiasts. Firstly, this model can be used in any high definition camcorders, this will make sure that this model is not limited to only face mask detection system.

Secondly, this can be used for biometric scans with a mask on the face. This study proposes the Mobile Net-based Depth wise Separable Convolution Neural Network (DS-CNN) for mask detection Frontiers in Public Health in facial images. We compare our findings to the original convolutional filters on specific datasets. The suggested system outperformed current classical convolutions in experiments, according to the results. The suggested technique is also contrasted with previous work on a motivated baseline method. Our findings (Acc. = 93.14, Pre. = 92, recall = 92, F-score = 92) show that the proposed method produces the highest overall performance across a variety of assessment metrics. The approach requires extra processing to generate visualizations and, owing to dataset constraints, cannot discriminate between right and erroneous mask usage. Our future aim is to create face mask recognition datasets with different mask wearing states, or employ zero shot learning to make the design identify erroneous mask wearing states.

REFERENCES

- Google
- Wikipedia
- Chen D, Hua G, Wen F, Sun J (2016) Supervised transformer network for efficient face detection. In:European conference on computer vision. Springer, pp 122–138. https://doi.org/10.1007/978-3-319- 46454-1 8
- Chen Y, Menghan Hu, Hua C, Zhai G, Zhang J, Li Q, Yang SX (2021) Face mask assistant: detection offace mask service stage based on mobile phone. IEEE Sensors J 21(9):11084–11093. https://doi.org/10.1109/JSEN.2021.3061178
- Ejaz MS, Islam MR, Sifatullah M, Sarker A (2019) Implementation of principal component analysis on masked and non-masked face recognition. In: 2019 1St international conference on advances in science, engineering and robotics technology (ICASERT). IEEE, pp 1–5. https://doi.org/10.1109/ ICASERT.2019.8934543
- Khan N, Yaqoob I, Hashem IAT, Inayat Z, Ali WKM, Alam M, Shiraz M, Gani A (2014) Big data:survey, technologies, opportunities, and challenges. Scientific World J, 2014. https://doi.org/10.1155/ 2014/712826
- Li S, Ning X, Yu L, Zhang L, Dong X, Shi Y, He W (2020) Multi-angle head pose classification when wearing the mask for face recognition under the covid-19 coronavirus epidemic. In: 2020 International conference on high performance big data and intelligent systems (HPBD&IS). IEEE, pp 1–5. https://doi.org/10.1109/HPBDIS49115.2020.9130585
- Mednikov Y, Nehemia S, Zheng B, Benzaquen O, Lederman D (2018) Transfer representation learning using inception-v3 for the detection of masses in mammography. In: 2018 40Th annual international conference of the IEEE engineering in medicine and biology society (EMBC). IEEE, pp 2587–2590. https://doi.org/10.1109/EMBC.2018.8512750
- "TensorFlow White Papers", TensorFlow, 2020, [online] Available: https://www.tensorflow.org/about/bib.
- "Keras documentation: About Keras", 2020, [online] Available: Keras.io.
- "OpenCV", 2020, [online] Available: Opencv.org.
- Walid Hariri. “Efficient Masked Face Recognition Method during the COVID-19 Pandemic” In: DOI: 10.21203/rs.3.rs-39289/v1 July 2020.
- Zhongyuan Wang et al. “Masked Face Recognition Dataset and Application” In: arXiv:2003.09093 [cs.CV] 2020.