A MAJOR PROJECT

on

IMPLEMENTATION OF FACE MASK DETECTION USING OPENCV

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

By

MADDI NIKITHA	187Y1A0422
PERABATHULA VAMSHI KRISHNA	187Y1A0451

Under the Guidance of
E. SREENIVASULU, Assistant Professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY & MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
NAAC Accredited Institution with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

June, 2022

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY & MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
NAAC Accredited Institution with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Date:

CERTIFICATE

This is to certify that the project work entitled "**IMPLENTATION OF FACE MASK DETECTION USING OPENCV**" work done by **MADDI NIKITHA (187Y1A0422) AND PERABATHULA VAMSHI KRISHNA (187Y1A0451)** students of Department of Electronics and Communication Engineering, is a record of bonafide work carried out by the members during a period from January, 2022 to June, 2022 under the supervision of **E. SREENIVASULU, Assistant Professor**. This project is done as a fulfilment of obtaining Bachelor of Technology Degree to be awarded by Jawaharlal Nehru Technological University Hyderabad, Hyderabad.

The matter embodied in this project report has not been submitted by us to any other university for the award of any other degree.

MADDI NIKITHA **PERABATHULA VAMSHI KRISHNA**

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Date: (**E. SREENIVASULU**)

The Viva-Voce Examination of above students, has been held on……

Head of the Department External Examiner

Principal

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our guide **E. Sreenivasulu**, **Assistant Professor**, Department of Electronics and Communication Engineering, for his excellent guidance and invaluable support, which helped us accomplish the B.Tech (ECE) degree and prepared us to achieve more life goals in the future. His total support of our dissertation and countless contributions to our technical and professional development made for a truly enjoyable and fruitful experience. Special thanks are dedicated for the discussions we had on almost every working day during our project period and for reviewing our dissertation.

We are very much grateful to our Project Coordinator**, Dr. G. Amarnath**, **Associate Professor,** Department of ECE, MLRITM, Dundigal, Hyderabad, who has not only shown utmost patience, but was fertile in suggestions, vigilant in directions of error and has been infinitely helpful.

We are extremely grateful to **Dr. Srinivas Bachu**, **Associate Professor & HOD-ECE**, MLRITM, Dundigal, Hyderabad, for the moral support and encouragement given in completing my project work.

We wish to express deepest gratitude and thanks to **Dr. K. Venkateswara Reddy, Principal,** MLRITM, for his constant support and encouragement in providing all the facilities in the college to do the project work.

We would also like to thank all our faculties, administrative staff and management of MLRITM, who helped us to completing the mini project.

On a more personal note, I thank my **beloved parents and friends** for their moral support during the course of our project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	Machine Learning
AI	Artificial Intelligence
NLP	Natural language processing
CNN	Convolutional neutral networks
SSD	Single shot detection
SVM	Support vector machine
YOLO	You Only Look Once
FPN	Feature Pyramid Network
RMFD	Real World Masked Face Dataset
API	Application programming Interface
IOU	Intersection over Union
GPU	Graphics Processing Unit
TFPU	TensorFlow Processing Unit
CPU	Central Preprocessing Unit
IBM	International Business Machine
MRI	Magnetic resonance imaging

ABSTRACT

COVID-19 pandemic has rapidly increased health crises globally and is affecting our day to-day lifestyle. Many measures are recommended by WHO to control the infection rate and avoid exhausting the limited medical resources. A motive for survival recommendations is to wear a safe facemask, stay protected against the transmission of coronavirus. By wearing a facemask, the most effective preventive care must be taken against COVID-19. Monitoring manually if the individuals are wearing face mask correctly and to notify the victim in public and crowded areas is a difficult task.

This project approaches a simplified way to achieve facemask detection and notifying the individual if not wearing facemask. Our project uses image processing and machine learning techniques. We collect data of images of face with and without masks and then image processing applied to it. We are giving data set of samples containing images with and without mask. So that we train the data using machine learning techniques like convolution neural networks. We use image processing technique viola jones algorithm to take images as input.

The output will be of color bounded box shown as without mask if the detected face is without mask and it sends the information to person and higher authorities too. If the person is wearing mask the bounded box will be of shown as mask. It indicates that it is safe now. The system runs in real- time and detects if an individual face has a facemask, if not then notifies the person-in-charge that the individual has not been equipped with a mask.

CHAPTER 1
INTRODUCTION

1.1 Introduction

As the world's population is promptly increasing, the COVID-19 cases have also increased in a vertical manner. Presently, there are 37 million active COVID-19 cases worldwide. It is predicted that by the end of 2021 this number will cross 50 million and 1 in every 6 people in India will suffer from COVID-19. According to research conducted by MIT, by the start of 2021, India alone will have a total of 10 million cases. That's why the implementation of the proposed project. Project is necessary for the overall benefit of various large institutions and businesses which are trying to avoid spreading the COVID-19, while maintaining their productivity.

1.2 Motivation

Coronavirus disease (COVID-19) is an irresistible infection caused by a newfound Coronavirus. A great many people tainted with the COVID-19 infection will encounter mellow to direct respiratory ailment and recoup without requiring exceptional treatment. More seasoned individuals and people with fundamental clinical issues like cardiovascular infection, diabetes, ongoing respiratory ailment, and malignant growth are sure to create genuine ailment. The most ideal approach to forestall and hinder transmission is to be all around educated about the COVID-19 infection, the illness it causes and the way it spreads. Shield yourself as well as other people from disease by washing your hands or utilizing a liquor-based rub regularly, not contacting your face and wearing a veil. The first three parts need to be governed by ourselves but it can either urge people or motivate them to wear masks, the proposed project implementation has attempted to make people aware that face masks are essential for their own and other's safety.

1.3 Machine Learning

Artificial intelligence is a study like arithmetic or science. It examines approaches to construct shrewd projects and machines that can innovatively tackle issues, which has consistently been viewed as a human right. AI is that the study of getting PCs to act without being expressly customized. In the previous decade, AI has given us self-driving vehicles, reasonable discourse acknowledgment, viable web search, and an endlessly improved comprehension of the human genome. AI has become so unavoidable today that you likely use it many times each day without knowing it. Numerous analysts additionally think it's the foremost ideal approach to realize ground towards human-level AI. Profound learning, or profound neural learning, is a subset of AI, which utilizes the neural organizations to examine various variables with a structure that is like the human neural framework.

1.3.1 Issues in ML

1. Thinking Power: One region where ML has not aced effectively is thinking power, an unmistakably human characteristic.

2. Logical Limitation: If the zone of natural language processing (NLP) is considered, text and discourse data are the way to comprehend dialects by NLP calculations.

3. Adaptability: Although the observed ML executions being conveyed on a critical premise, everything relies upon information just as its versatility.

4. Administrative Restriction for Data in ML: ML generally need significant sums (indeed, gigantic) of information in stages, for example, preparing, cross-approval and so forth.

1.4 The Image Net

The proposed presentation is measured on image Net. The image Net venture is a huge visual information base intended for use in visual article acknowledgment programming research. In excess of 14 million pictures have been hand-clarified by the task to demonstrate what items are imagined and in any event 1,000,000 of the pictures, bouncing boxes are likewise given. Image Net contains in excess of 20,000 classifications with an ordinary classification, for example, "inflatable" or "strawberry", comprising of a few hundred pictures.

1.5 Classification

In ML, the process to identify which set of categories are belongs to one group based on the relevant observations is called as classification method. The classification algorithm classifies the data into each corresponding classes. It is a mode of supervised learning and it puts data in respective categories as per their tags.

CHAPTER 2

LITERATURE SURVEY

Literature Review Object detection is one of the trending topics in the field of image processing and computer vision. Ranging from small scale personal applications to large scale industrial applications, object detection and recognition is employed in a wide range of industries. Some examples include image retrieval, security and intelligence, OCR, medical imaging and agricultural monitoring. In object detection, an image is read and one or more objects in that image are categorized. The location of those objects is also specified by a boundary called the bounding box. Traditionally, researchers used pattern recognition to predict faces based on prior face models. A breakthrough face detection technology then was developed named as Viola Jones detector that was an optimized technique of using Haar digital image features used in object recognition. However, it failed because it did not perform well on faces in dark areas and non-frontal faces. Since then, researchers are eager to develop new algorithms based on deep learning to improve the models. Deep learning allows us to learn features with end-to-end manner and removing the need to use prior knowledge for forming feature extractors. There are various methods of object detection based on deep learning which are divided into two categories: one stage and two object detectors.

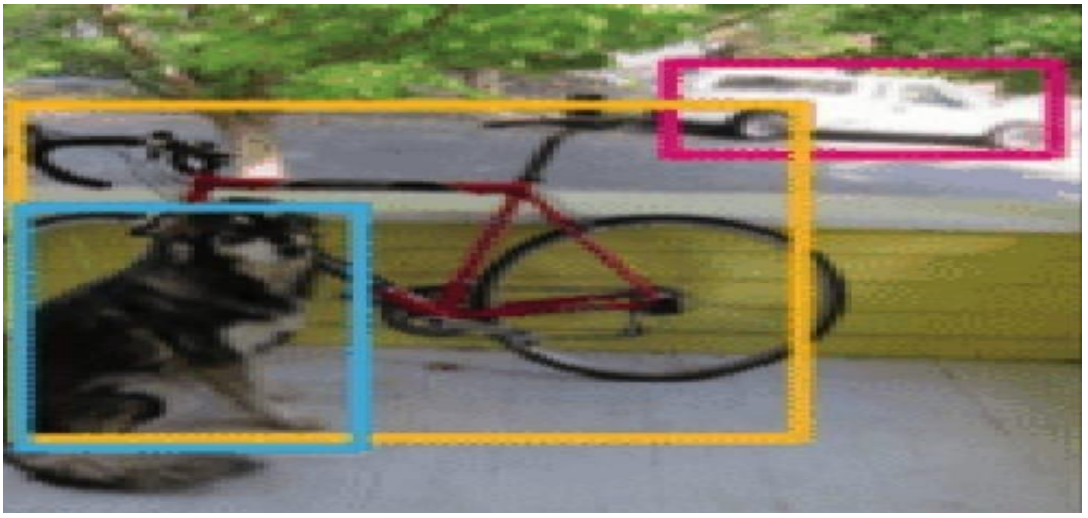

Figure 2.1: Bounding boxes in an image.

Two stage detectors use two neural networks to detect objects, for instance region- based convolutional neural networks (R-CNN) and faster R-CNN. The first neural network is used to generate region proposals and the second one refines these region proposals; performing a coarse-to-fine detection. This strategy results in high detectionperformance compromising on speed. The seminal work R-CNN is proposed by R. Girshick et aI. R-CNN uses selective search to propose some candidate regions which may contain objects. After that, the proposals are fed into a CNN model to extract features, and a support vector machine (SVM) is used to recognize classes of objects. However, the second stage of R-CNN is computationally expensive since the network has to detect proposals on a one-by-one manner and uses a separate SVM for final classification. Fast R-CNN solves this problem by introducing a region of interest (ROI) pooling layer to input all proposal regions at once.

Faster RCNN is the evolution of R-CNN and Fast R-CNN, and as the name implies its training and testing speed is greater than those of its predecessors. While R-CNN and Fast R-CNN use selective search algorithms limiting the detection speed, Faster R-CNN learns the proposed object regions itself using a region proposal network (RPN). On the other hand, a one stage detector utilizes only a single neural network for region proposals and for detection; some primary ones being SSD (Single Shot Detection) and YOLO (You Only Look Once). To achieve this, the bounding boxes should be predefined. YOLO divides the image into several cells and then matches the bounding boxes to objects for each cell. This, however, is not good for small sized objects. Thus, multi scale detection is introduced in SSD which can detect objects of varying sizes in an image. Later, in order to improve detection accuracy, Lin et. Al proposes Retina Network (Retina Net) by combining an SSD and FPN (feature pyramid network) to increase detection accuracy and reduce class imbalance.

One-stage detectors have higher speed but trades off the detection performance but then only are preferred over two-stage detectors. Like object detection, face detection adopts the same architectures as one-stage and two-stage detectors, but in order to improve face detection accuracy, more face-like features are being added. However, there is occasional research focusing on face mask detection. Some already existing face mask detectors have been modeled using OpenCV, Pytorch Lightning, Mobile Net, Retina Net and Support Vector Machines. Here, we will be discussing

two projects. One project used Real World Masked Face Dataset (RMFD) which contains 5,000 masked faces of 525 people and 90,000 normal faces. These images are 250 x 250 in dimensions and cover all races and ethnicities and are unbalanced.

This project took 100 x 100 images as input, and therefore, transformed each sample image when querying it, by resizing it to 100x100. Moreover, this project uses PyTorch then they convert images to Tensors, which is the base data type that PyTorch can work with. RMFD is imbalanced (5,000 masked faces vs 90,000 non-masked faces). Therefore, the ratio of the samples in train/validation while splitting the dataset was kept equal using the train test split function of sklearn. Moreover, to deal with unbalanced data, they passed this information to the loss function to avoid unproportioned step sizes of the optimizer. They did this by assigning a weight to each class, according to its represent ability in the dataset. They assigned more weight to classes with a small number of samples so that the network will be penalized more if it makes mistakes predicting the label of these classes.

While classes with large numbers of samples, they assigned to them a smaller weight. This makes their network training agnostic to the proportion of classes. The weights for each class were chosen using the formula below: Class W eight = 1 − Class Cardinality Cardinalities of all classes to load the data efficiently this project used the data loader. For instance, in this project, they used the PyTorch lighting, and to load them for training and validation they divided data into 32 batches and assigned the works of loading to the 4 number of workers, and this procedure allowed them to perform multi-process data loading.

Like most of the projects, this project also used Adam optimizer. If any Model has a high rate of learning, it learns faster, but it bounces a lot to reach the global minima and may diverge from the global minima. However, a small learning rate may take considerably lower time to train, but it reaches to the global minima. If the loss of the model declines quickly for any learning rate, then that learning rate would be the best learning rate. However, it seems that this project considered the 0.00001 learning rate would be the best for their model so that it could work efficiently. To train the model they defined a model check pointing callback where they wanted to save the best accuracy and the lowest loss. They tried to train the model for 10 epochs and after finding optimal epoch, they saved the model for 8 epochs to test on the real data. To get rid of the problem of occlusions of the face which causes trouble face detectors to

detect masks in the images, they used a built-in OpenCV deep learning face detection model. For instance, the Haar-Cascade model could be used but the problem of the Haar-Cascade model is that the detection frame is a rectangle, not a square. That is why, without capturing the portion of the background, the face frame can fit the entirety of the face, which can interfere with the face mask model predictions. In the second project [9], a dataset was created by Prajna Bhandary using a PyImageSearch reader. This dataset consists of 1,376 images belonging to all races and is balanced. There are 690 images with masks and 686 without masks. Firstly, it took normal images of faces and then created a customized computer vision Python script to add face masks to them. Thereby, it created a real-world applicable artificial dataset. This method used the facial landmarks which allow them to detect the different parts of the faces such as eyes, eyebrows, nose, mouth, jawline etc. To use the facial landmarks, it takes a picture of a person who is not wearing a mask, and, then, it detects the portion of that person's face. After knowing the location of the face in the image, it extracted the face Region of Interest (ROI). After localizing facial landmarks, a picture of a mask is placed into 3 the face. In this project, embedded devices are used for deployment that could reduce the cost of manufacturing.

MobileNetV2 architecture is used as it is a highly efficient architecture to apply on embedded devices with limited computational capacity such as Google Coral, NVIDIA Jetson Nano. This project performed well, however, if a large portion of the face is occluded by the mask, this model could not detect whether a person is wearing a mask or not. The dataset used to train the face detector did not have images of people wearing face masks as a result, if the large portion of faces is occluded, the face detector would probably fail to detect properly. To get rid of this problem, they should gather actual images of people wearing masks rather than artificially generated images.

CHAPTER 3
METHODOLOGY

3.1 Dataset

The dataset which we have used consists of 3835 total images out of which 1916 are of masked faces and 1919 are of unmasked faces. All the images are actual images extracted from Bing Search API, Kaggle datasets and RMFD dataset. From all the three sources, the proportion of the images is equal. The images cover diverse races i.e Asian, Caucasian etc. The proportion of masked to unmasked faces determine that the dataset is balanced. We need to split our dataset into three parts: training dataset, test dataset and validation dataset. The purpose of splitting data is to avoid overfitting which is paying attention to minor details/noise which is not necessary and only optimizes the training dataset accuracy. We need a model that performs well on a dataset that it has never seen (test data), which is called generalization.

The training set is the actual subset of the dataset that we use to train the model. The model observes and learns from this data and then optimizes its parameters. The validation dataset is used to select hyperparameters (learning rate, regularization parameters). When the model is performing well enough on our validation dataset, we can stop learning using a training dataset. The test set is the remaining subset of data used to provide an unbiased evaluation of a final model fit on the training dataset. Data is split as per a split ratio which is highly dependent on the type of model we are building and the dataset itself. If our dataset and model are such that a lot of training is required, then we use a larger chunk of the data just for training which is our case. If the model has a lot of hyperparameters that can be tuned, then we need to take a higher amount of validation dataset. Models with a smaller number of hyperparameters are easy to tune and update, and so we can take a smaller validation dataset. In our approach, we have dedicated 80% of the dataset as the training data and the remaining 20% as the testing data, which makes the split ratio as 0.8:0.2 of train to test set. Out of the training data, we have used 20% as a validation data set. Overall, 64% of the dataset is used for training,16%forvalidationand20%fortesting.

3.2 Computer Vision

In the field of Artificial Intelligence, Computer Vision plays a vital role to train computers to understand the visual world. Using digital images which are generated from various sources such as cameras and videos in which machines identify and classify objects.

Images are broken into pixels which are the elements of the picture or the smallest unit of information which makes pictures. Computer vision not only converts a picture into pixels and also senses what is in the picture through those pixels. You have to know the bigger picture of how to extract the information from those pixels.

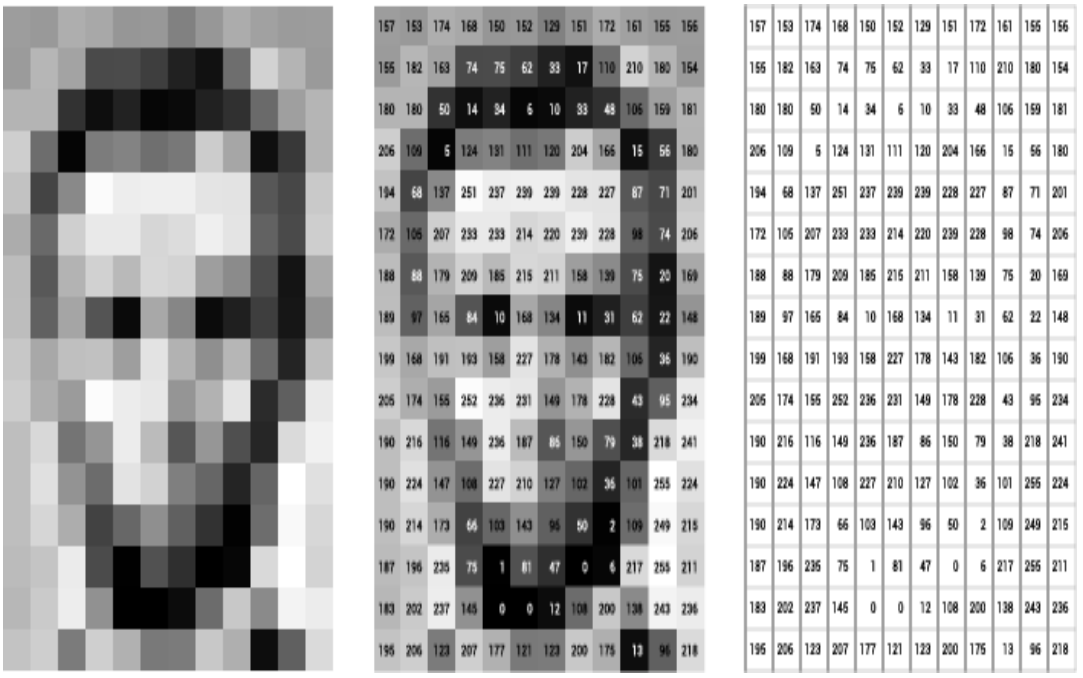

Figure 3.1: OpenCV

The inputs to our model are the live video image frames.

The output is either a beep sound with a red rectangular box around the face if there is no mask on that particular face or a green rectangular box around the face ensuring there is a mask on his/her face. We trained our model using various pre-trained models like the haar-cascade model, Caffe model.

3.3 Requirement Engineering

Requirements engineering is the process of describing, documenting, and supporting requirements. We need a realistic and full set of functional and non-functional requirements for any project to be successful and error-free. A successful project must be practical, clear, traceable, complete, and valid.

3.3.1 Functional Requirement

The functional requirements sections, the functional capabilities of the system are described. It describes the function of a system or its component, where the function is described as a specification of behavior between inputs and outputs. Hence this section will introduce the different parts required. The reader should be able to comprehend the system, and no prior knowledge of the subject is required to understand this documentation.

3.3.2 Data Requirement

The proposed system uses data containing different images of faces with and without masks. The custom dataset will be imported into the proposed system, and it will be trained using labelled images. To detect faces, image processing is performed in the captured image frame. The result can be seen in the same frame with a square box around the faces and a label that says with or without a mask.

3.3.3 Non-Functional Requirement

Non-functional requirements describe the behavior of the system. It covers all the requirements which were not covered in the functional requirements. Any mistakes in non-functional requirements can lead to critical issues and may result in application failure. This defines how accurately the system should function.

- Ease of use.
- Availability
- Reliability
- Maintainability
- Usability
- Environmental
- Recoverability

3.4 Training

At the training time, for each pixel, we compare the default bounding boxes having different sizes and aspect ratios with ground truth boxes and finally use Intersection over Union (IOU) method to select the best matching box. IOU evaluates how much part of our predicted box match with the ground reality. The values range from 0 to 1 and increasing values of IOU determine the accuracies in the prediction; the best value being the highest value of IOU. The equation and pictorial description of IoU is given as follow: IoU (B1, B2) = B1 ∩ B2 B1 ∪ B2 Figure 4: Pictorial representation of IoU.

3.5 Hyper parameters

A hyper parameter is a parameter or a variable we need to set before applying an algorithm into a dataset. These parameters express the "High Level" properties of the model such as its complexity or how fast it should learn. Hyper parameters are fixed before the actual training process begins. They can be divided into two categories: optimizer hyper parameters and model hyper parameters. Optimizer parameters help us to tune or optimize our model before the actual training process starts. Some common optimizer hyper parameters are as follows. Learning rate is a hyper parameter that controls how much we are adjusting the weights of our neural network with respect to the gradient. Mini-batch size is a hyper parameter that influences the resource requirements of the training and impacts training speed and number of iterations. Epochs are the hyperparameters that determine the frequency of running the model. One epoch is when an entire dataset is passed forward and backward through the neural network only once. Model hyperparameters are parameters that are more involved in the architecture or structure of the model. They help us to define our model complexity based on the different layers like the input layer, hidden layer, and output layer of a neural network. Initially, we trained with different values of hyperparameters by changing one and keeping the other constant and noted down the results in each case. We selected the hyperparameters that produced better performance through evaluation metrics.

CHAPTER 4
DESIGN OF SOFTWARE

4.1 Use case Diagram

The most important aspect of designing a diagram is to capture its dynamic behavior. The use case diagram is always changing. The internal agent and external agent are both called “Actors”. The content of the Use case diagram is Actors, Use cases and their relationships. The system or the subsystem of an application are modeled by using a diagram. A particular functionality of a system is captured by a single use case diagram. The actors and the use cases are prepared and identified when the system is analyzed to gather the functionalities.

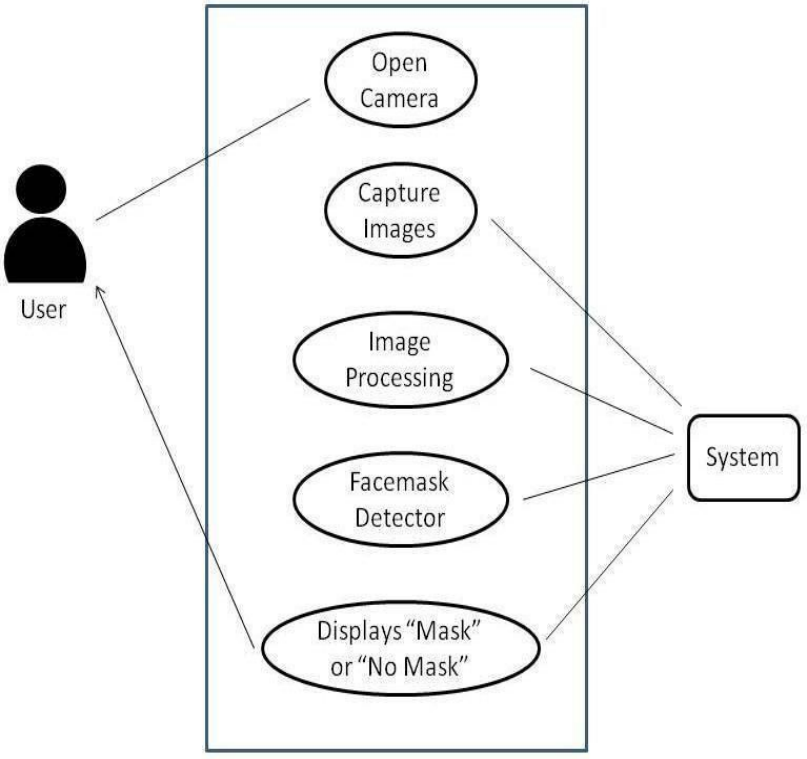

Figure 4.1: Use case diagram

4.1.1 Sequence Diagram

A Sequence diagram is an interaction diagram that shows the interactions between elements in a model. These interactions are part of the dynamic behaviour of the system but from a different angle. A sequence diagram focuses on time and uses the vertical axis of the diagram to visually display the order of interaction to represent when messages are being sent.

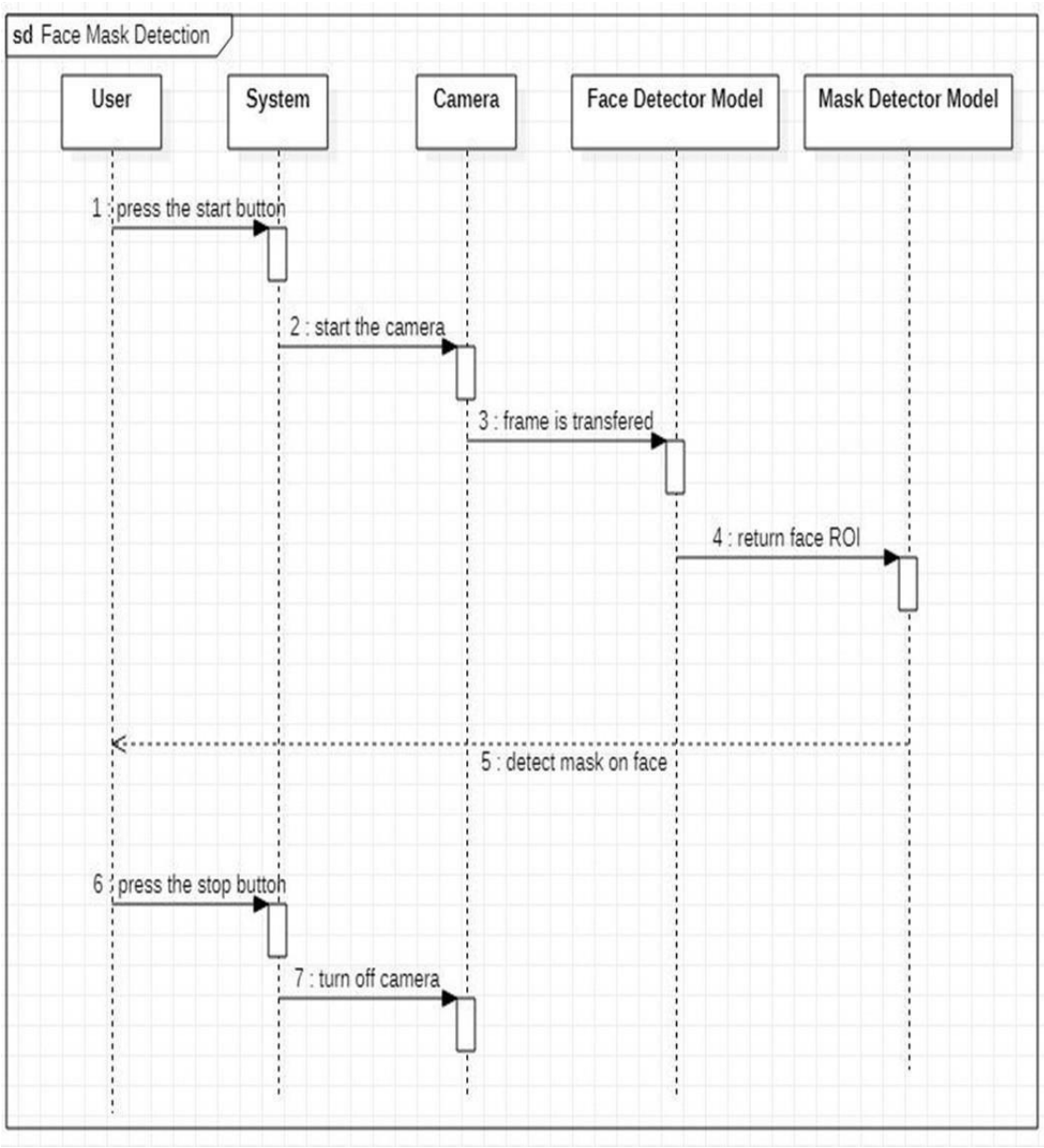

Figure 4.2 : Sequence diagram

4.1.2 Activity Diagram

Activity diagrams show how activities are coordinated to provide services at different levels of abstraction. Events usually need to be achieved with some operations. Especially if the operation is aimed at achieving various things that need to be coordinated, or how the events of a single use case arerelated to each other, especially if the activities overlap and need to be coordinated.

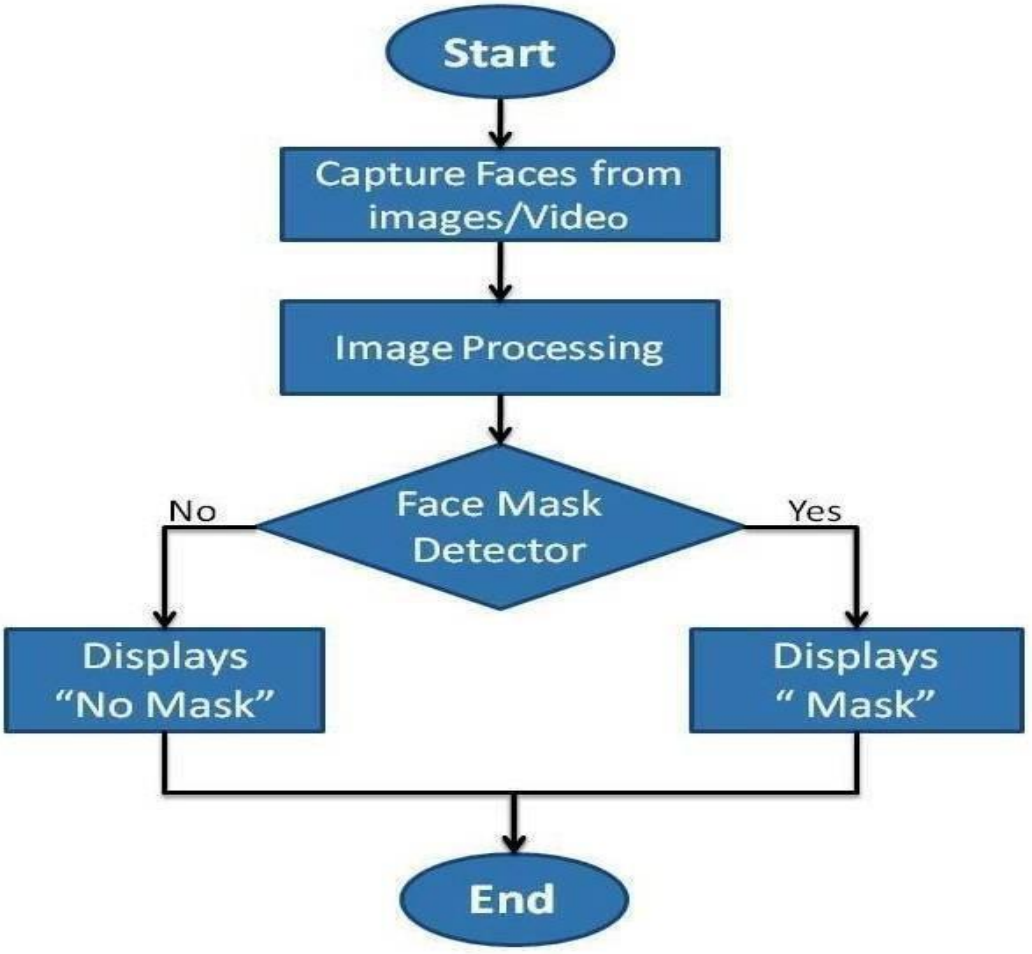

Figure 4.3: Activity diagram

4.1.3 System architecture

A system architecture is a model which defines the structure, behavior, and more perspectives of a system. An architecture interpretation is an authorized description and system representation, organized in such a way that supports reasoning about the structures and system behaviors. A system architecture consists of system components and the sub-systems developed, that will work together to implement the comprehensive system.

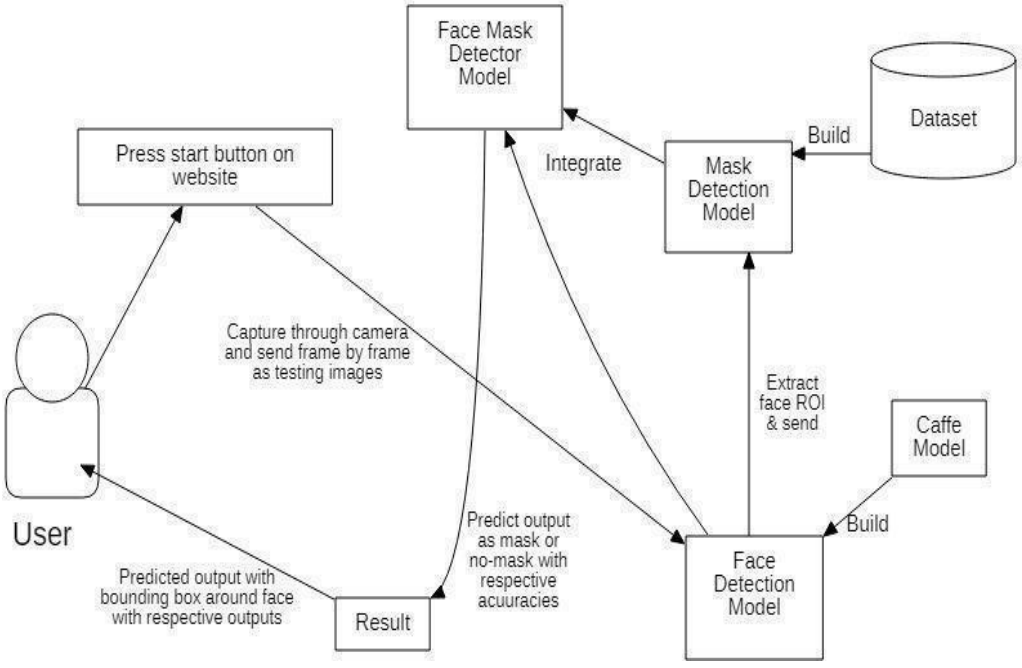

Figure 4.4 : System architecture

Our system architecture starts from the user. The user interacts with the web application to access the web camera. So, the web camera starts capturing frames and sends the images to the face detection model. The face detection model makes use of the caffe model and detects the face Region of Interest (ROI) from the images. Now, the face detection model passes all the face Region of Interest (ROI) to the Mask detection model.

The mask detection model is trained with the dataset using neural networks and ImageNet weights. Now, the face mask detection model predicts either with or without mask present on the face and returns the output on the web application. To stop the web camera and the whole process, the user needs to press the stop button on the web application.

4.2 Webcam

A webcam is a video camera that feeds or streams an image or video in real time to or through a computer network, such as the Internet. Webcams are typically small cameras that sit on a desk, attach to a user's monitor, or are built into the hardware. Webcams can be used during a video chat session involving two or more people, with conversations that include live audio and video.

Webcam software enables users to record a video or stream the video on the Internet. As video streaming over the Internet requires much bandwidth, such streams usually use compressed formats. The maximum resolution of a webcam is also lower than most handheld video cameras, as higher resolutions would be reduced during transmission. The lower resolution enables webcams to be relatively inexpensive compared to most video cameras, but the effect is adequate for video chat sessions.

The term "webcam" (a clipped compound) may also be used in its original sense of a video camera connected to the Web continuously for an indefinite time, rather than for a particular session, generally supplying a view for anyone who visits its web pageover the Internet. Some of them, for example, those used as online traffic cameras, are expensive, rugged professional video cameras.

Figure 4.5: Typical low-cost webcam used with many personal computers (2007)

Figure 4.6: Higher cost webcam (2017) with 1080p resolution and built-in stereo microphones

4.2.1 Technology

Webcams typically include a lens, an image sensor, support electronics, and may also include one or even two microphones for sound.

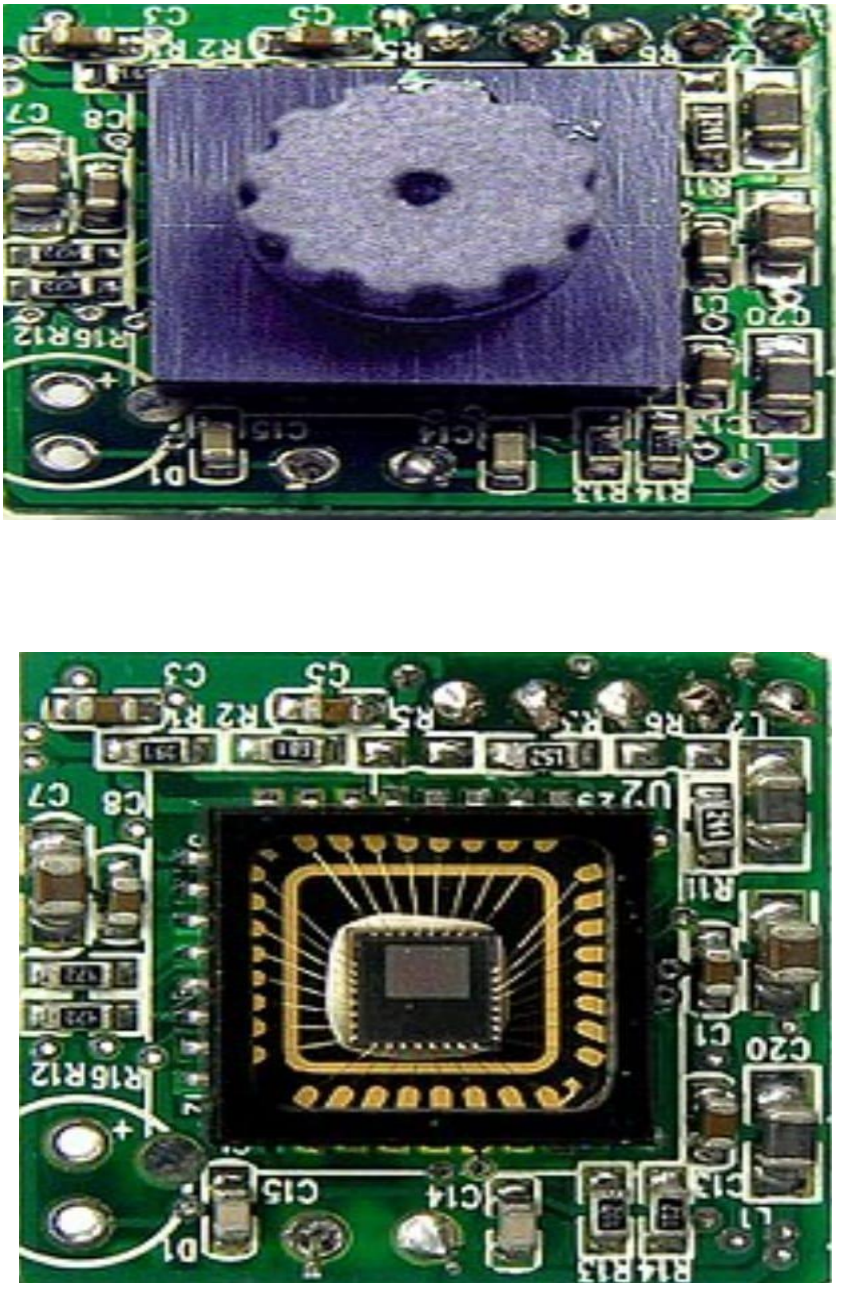

Figure 4.7: Image sensor, lens and supporting circuitry

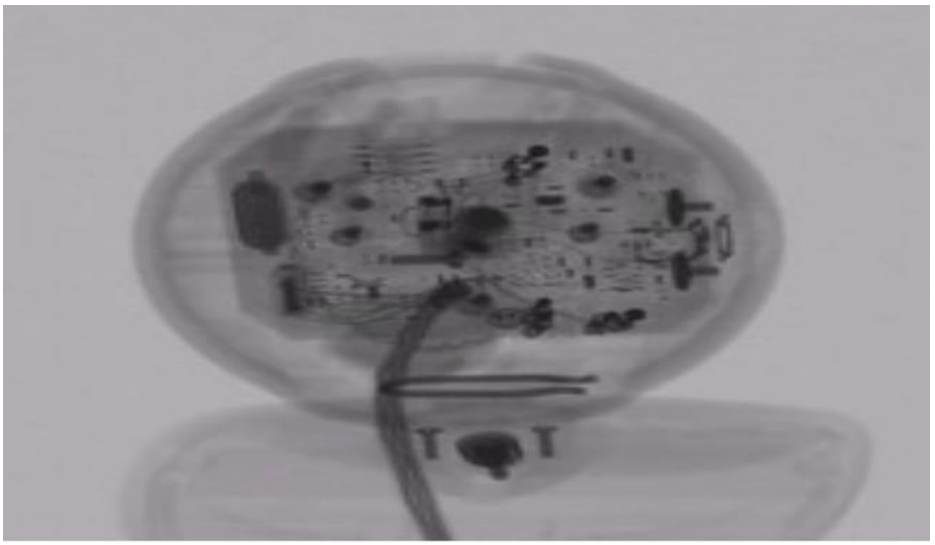

Figure 4.8: Animated set of X-ray images of a webcam. Images acquired using industrial CT scanning

4.2.2 Image sensor

Image sensors can be CMOS or CCD, the former being dominant for low-cost cameras, but CCD cameras do not necessarily outperform CMOS-based cameras in the low-price range. Most consumer webcams are capable of providing VGA-resolution video at a frame rate of 30 frames per second. Many newer devices can produce video in multi-megapixel resolutions, and a few can run at high frame rates such as the PlayStation Eye, which can produce 320×240 video at 120 frames per second. The Wii Remote contains an image sensor with a resolution of 1024×768 pixels. Common resolutions of laptops' built-in webcams are 720p (HD), and in lower-end laptops 480p. The earliest known laptops with 1080p (Full HD) webcams like the Samsung 700G7C were released in the early 2010s. [3]

As the bayer filter is proprietary, any webcam contains some built-in image processing, separate from compression.

4.2.3 Optics

Various lenses are available, the most common in consumer-grade webcams being a plastic lens that can be manually moved in and out to focus the camera. Fixed-focus lenses, which have no provision for adjustment, are also available. As a camera system's depth of field is greater for small image formats and is greater for lenses with a large f-number (small aperture), the systems used in webcams have a sufficiently large depth of field that the use of a fixed-focus lens does not impact image sharpness to a great extent.

Most models use simple, focal-free optics (fixed focus, factory-set for the usual distance from the monitor to which it is fastened to the user) or manual focus.

4.2.4 Uses

The most popular use of webcams is the establishment of video links, permitting computers to act as videophones or videoconference stations. For example, Apple's insight camera, which is built into Apple laptops, iMacs and a majority of iPhones, can be used for video chat sessions, using the Messages instant messaging program. Other popular uses include security surveillance, computer vision, video broadcasting, and for recording social videos.

The video streams provided by webcams can be used for a number of purposes, eachusing appropriate software

4.3 Video monitoring

Webcams may be installed at places such as childcare centres, offices, shops and private areas to monitor security and general activity

4.4 Computer Vision

Figure 4.9: Computer vision Identifying

Computer vision, a type of artificial intelligence, enables computers to interpret and analyze the visual world, simulating the way humans see and understand their environment. It applies machine learning models to identify and classify objects in digital images and videos, then lets computers react to what they see. Different types computer vision includes image segmentation, object detection, facial recognition, edge detection, pattern detection, image classification, and feature matching.

Computer vision enables a wide range of technological innovation. It allows self-driving cars to safely steer through streets and highways; it enables facial recognition tools to match images of people's faces to their identities; and it enables augmented-reality applications to mix virtual objects with real-world images.

Computer vision applications are used across industries to improve the consumer experience, reduce costs, and tighten security. Manufacturers use it to spot defective products on the assembly line and prevent them from shipping to customers. Insurance adjusters use it to assess vehicle damage and reduce fraud in the claims process. Medical professionals use it to scan X-rays, MRIs, and ultrasounds to detect health problems. Banks use it to verify customers' identities before conducting large transactions

4.4.1 Sub Fields

Sub-domains of computer vision include scene reconstruction, object detection, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, visual servoing, 3D scene modeling, and image restoration.

4.4.2 Applications

Computer Vision for Defect detection.

Computer Vision for Metrology.

Computer Vision for Intruder Detection.

Computer Vision for Assembly verification.

Computer Vision for Screen reader.

Computer Vision for Code and character reader (OCR)

Computer Vision + robotics for bin picking.

CHAPTER 5

CONSTRUCTION

5.1 Implementation

The implementation of the project is done using python. To be particular, for the aim of machine learning Anaconda is getting used.

Anaconda is one of several Python distributions. Anaconda is a new distribution of Python. It was formerly known as Continuum Analytics. Anaconda has more than 100 new packages. Anaconda is not only used for machine learning but also used for many other technologies.

On Python technology, we found Anaconda to be easier. Since it is easy and destroysmany problems:

- Installing Python on multiple platforms.
- Separating out different environments.
- Dealing with not having correct privileges.
- Start with specific packages and libraries.

In Anaconda, we mainly used Jupyter notebook, which helped us a lot with packages. There are many inbuilt packages in Jupyter notebook which helped a lot. Image processing is used to detect or extract some useful content from the images. Firstly, the images will be loaded, then the images are converted into an array. The array contains the RGB color values present in an image and it will be helped in detecting the faces in the image. In image processing we use OpenCV.

For image processing, we use OpenCV. OpenCV is an open-source machine learning software library. It has many optimized algorithms. This library has C ++, Python, Java and MATLAB interfaces.

5.1.1 Implementation Details

For the purpose of proper implementation and functioning algorithms and techniques are used. Following are the algorithms used:

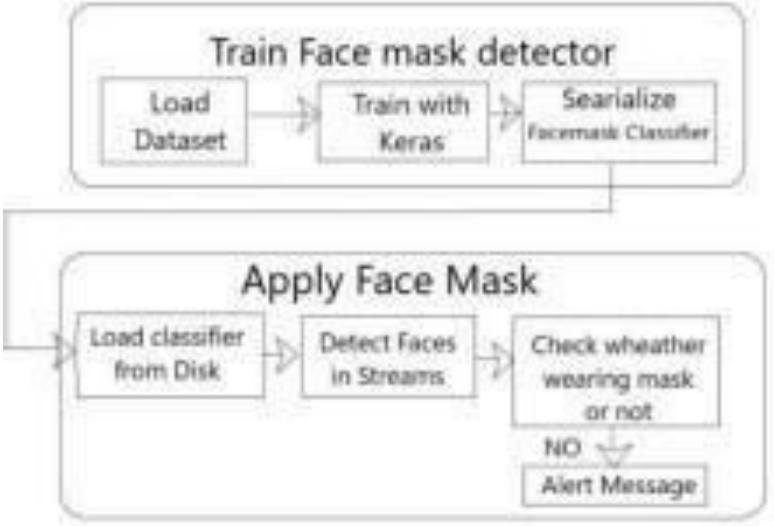

Figure 5.1: Train and applying Face Mask

5.1.2 Image Processing

Firstly, we have to collect the data we need to complete the model. Image processing is used to detect or extract some useful content from the images. We use image processing in this model, there are certain steps to perform Image Processing:

- Load the images using Python or any other Programming language.
- Convert the images into arrays.
- After loading images, we can perform any algorithm on the arrays we have.

In image processing, we can load the images in any language you want. Then the images will be converted into arrays which represent the RGB color values in the arrays according to the indexes.

5.1.3 CNN (Convolutional Neural Network)

We proposed an Optimistic Convolution Network that helps to ensure whether in public the people are wearing masks or not by monitoring them automatically. Here in Fig 1, we have described an architecture that shows how our system functions automatically to prevent the spread of COVID19.

Our system uses the TensorFlow and Keras algorithm to detect whether an individual is wearing a face mask along with the Convolutional Neural network model. Here we first train the system with the Dataset from Kaggle and train it with Keras and TensorFlow, once the training is done then we will load the face mask classifier from the disk, here faces are detected from a real time video stream. This process also involves use of MobileNet in order to train a huge collection of images and classification of high-quality images.

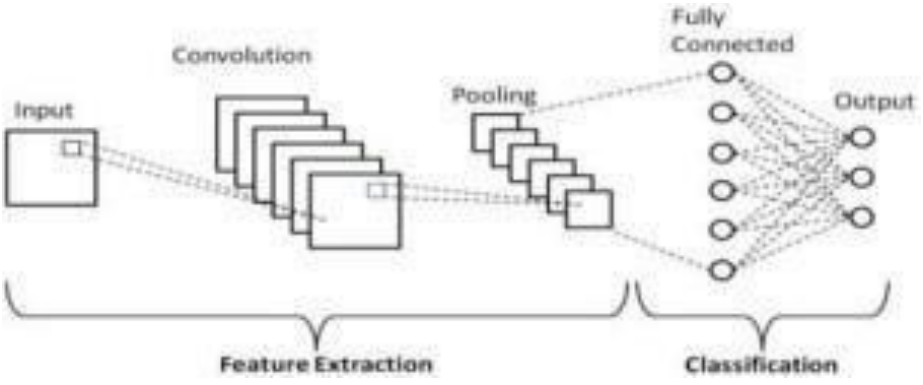

Figure 5.2: Convolutional Neural Network

Here the image dataset is loaded from Keras and then the images are converted into an array, later MobileNet is used to preprocess input images and to append images to the data list. In the proposed system the main contribution includes person face identification and face mask detection. These both are done in Real Time with the help of MobileNet and OpenCV. A square box is displayed on every person's face with the color of red and green where red indicates the personis not wearing a mask and green indicates a person is wearing a mask.

5.2 Software Details

- Anaconda Distribution (v5.1)
- Python (3.6.5)
- Jupyter Notebook

5.3 Hardware Details

- Operating system: Windows 7 or newer, 64-bit macOS 10.9+, orLinux.
- System architecture: 64-bit x86, 32-bit x86 with Windows or Linux.
- CPU: Intel Core 2 Quad CPU Q6600 @ 2.40GHz or greater.
- RAM: 4 GB or greater.

Libraries and Functions

NUMPY:

NumPy came into existence when there was a need to apply faster operations on arrays. When we try applying any operation on arrays it consumes a lot of time and time complexity gets increased. All these problems can be resolved using the python library NumPy that provides multidimensional array objects such asmatrices and masked arrays. NumPy provides an array object which is nearly50x faster than python lists. Ndarray (n-dimensional array), an array object in NumPy provides a lot of functions that help in working without any difficulty.

SKLEARN.MODEL_SELECTION:

This helps in splitting the data frames or matrices or arrays into training and testing samples randomly (without any order).

```
sklearn.model_selection.train_test_split(*arrays, test_size=None, train_size=None, random_state=None, shuffle=True,
stratify=None)                                                                                    [source]
```

Figure 5.3 train_test_split() method

5.4 Testing

Software testing is an examination led to give partners data about the nature of the product item or administration under test. Software testing can likewise give a target, autonomous perspective on the product to permit the business to acknowledge and comprehend the dangers of programming execution. Test strategies incorporate the way toward executing a program or application with the plan of discovering programming bugs (mistakes or different deformities), and checking that the product item is fit for use.

Programming testing includes the execution of a product part or framework segment to assess at least one property of intrigue. When all is said in done, these properties demonstrate the degree to which the segment or framework under test:

- Satisfies all the requirements, that has guided its structure,
- responses more effectively to a wide range of sources of information,
- works more than its capabilities in the limited amount of time,
- it is satisfactorily usable without any issues,
- it can be initialized and execute around in its expected outcomes, and
- The output from the program comes with a good accuracy.

The number of potential tests even for a small application also have limitless test cases. All the testing techniques use some methodology to select the test cases that compete within accessible time and assets. Therefore, testing a program ordinarily helps to run an application or a program with the expectation of creating mistakes or different imperfections which are known as bugs in the project content. The testing process is an iterative process, when it clears one bug another new bug comes into the scenario.

5.5 OPEN CV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. 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, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

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 Open CL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

5.5.1 PYTHON

Python can serve as scripting language for web applications, e.g., via mod wsgi for the Apache web server. With Web Server Gateway Interface, a standard API has evolved to facilitate these applicate. Web frameworks like Django, Pylons, Pyramids, TurboGears, web2py, Tornado, Flask, Bottle and Zope support developers in the design and maintenance of complex applications. Pyjs and Iron Python can be used to develop the client-side of Ajax-based applications. SQLAlchemy can be used as a data mapper to a relational database. Twisted is a framework to program communications between computers, and is used (for example) by Dropbox.

Libraries such as NumPy, SciPy and Matplotlib allow the effective use of Python in scientific computing, with specialized library such Bio python and Astropy providing domain-specific functionality. Sage Math is a computer algebra system with a notebook interface programmable in Python: its library covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus. OpenCV has Python bindings with a rich set of features for computer vision and image processing.

Python is commonly used in artificial intelligence projects and machine learning projects with the help of libraries like TensorFlow, Keras, Pytorch and Scikit-learn. As a scripting language with modular architecture, simple syntax and rich text processing tools, Python is often used for natural language processing. Python can also be used to create games, with libraries such as Pygame, which can make 2D games.

Python has been successfully embedded in many software products as a scripting language, including in finite element method software such as Abaqus, 3D parametric modeler like FreeCAD, 3D animation packages such as 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, Motion Builder, Softimage, the visual effects compositor Nuke, 2D imaging programs like GIMP, Inkscape, Scribus and Paint Shop Pro, and musical notation programs like score writer and capella. GNU Debugger uses Python as a pretty printer to show complex structures such as C++ containers. Esri promotes Python as the best choice for writing scripts in ArcGIS. It has also been used in several video games, and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go.

Many operating systems include Python as a standard component. It ships with most Linux distributions, AmigaOS4 (using Python 2.7), FreeBSD (as a package), NetBSD, OpenBSD (as a package) and macOS and can be used from the command line (terminal). Many Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora use the Anaconda installer. Gentoo Linux uses Python in its package management system, Portage.

Python is used extensively in the information security industry, including in exploit development.

Most of the Sugar software for the One Laptop per Child XO, now developed at Sugar Labs, is written in Python. The Raspberry Pi single-board computer project has adopted Python as its main user-programming language.

LibreOffice includes Python, and intends to replace Java with Python. Its Python Scripting Provider is a core feature since Version 4.0 from 7 February 2013.

Languages influenced by Python

Python's design and philosophy have influenced many other programming languages:

- Cobra uses indentation and a similar syntax, and its Acknowledgements document lists Python first among languages that influenced it.
- ECMAScript/JavaScript borrowed iterators and generators from Python.
- GDScript, a scripting language very similar to Python, bulit-in to the Godot game engine.
- Go is designed for the "speed of working in a dynamic language like Python" and shares the same syntax for slicing arrays.
- Groovy was motivated by the desire to bring the Python design philosophy to Java.
- Julia was designed to be "as usable for general programming as Python".
- Nim uses indentation and similar syntax.
- Ruby's creator, Yukihiro Matsumoto, has said: "I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python. That's why I decided to design my own language.
- "Swift, a programming language developed by Apple, has some Python-inspired syntax.

Python's development practices have also been emulated by other languages. For example, the practice of requiring a document describing the rationale for, and issues surrounding, a change to the language (in Python, a PEP) is also used in Tcl, Erlang and, Swift.

CHAPTER 6

SOURCE CODE

```
import cv2
haar_data = cv2.CascadeClassifier('haar.xml')
capture=cv2.VideoCapture(0)
while True:
    flag,img=capture.read()
    if flag:
        faces=haar_data.detectMultiScale(img)
        for x,y,w,h in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,255),4)
        cv2.imshow('result',img)
        if cv2.waitKey(2) == 27:
            break
capture.release()
cv2.destroyAllWindows()
import numpy as np
capture=cv2.VideoCapture(0)
data=[]
while True:
    flag,img=capture.read()
    if flag:
        faces=haar_data.detectMultiScale(img)
        for x,y,w,h in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,255),4)
            face=img[y:y+h,x:x+w,:]
            face=cv2.resize(face,(50,50))
```

```
            print(len(data))
            if len(data)<200:
                data.append(face)
        cv2.imshow('result',img)
        if cv2.waitKey(2) == 27 or len(data)>=200:
            break
capture.release()
cv2.destroyAllWindows()
np.save('with_mask',data)
np.save('without_mask',data)
import matplotlib.pyplot as plt
plt.imshow(data[0])
import numpy as np
import cv2
with_mask=np.load('with_mask.npy')
without_mask=np.load('without_mask.npy')
with_mask.shape
with_mask=with_mask.reshape(200,50*50*3)
without_mask=without_mask.reshape(200,50*50*3)
x=np.r_[with_mask,without_mask]
labels=np.zeros(x.shape[0])
labels[200:]=1.0
names={0:'Mask',1:'No Mask'}
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,labels,test_size=
    0.25)
```

```
from sklearn.decomposition import PCA
pca=PCA(n_components=3)
x_train=pca.fit_transform(x_train)
svm=SVC()
svm.fit(x_train,y_train)
x_test=pca.transform(x_test)
y_pred=svm.predict(x_test)
accuracy_score(y_test,y_pred)
haar_data = cv2.CascadeClassifier('haar.xml')
capture=cv2.VideoCapture(0)
data=[]
font=cv2.FONT_HERSHEY_COMPLEX
while True:
    flag,img=capture.read()
    if flag:
        faces=haar_data.detectMultiScale(img)
        for x,y,w,h in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,255),4)
            face=img[y:y+h,x:x+w,:]
            face=cv2.resize(face,(50,50))
            face=face.reshape(1,-1)
            face=pca.transform(face)
            pred=svm.predict(face)
            n=names[int(pred)]
            cv2.putText(img,n,(x,y),font,1,(244,250,250),2)
            #print(n)
        cv2.imshow('result',img)
        if cv2.waitKey(2) == 27:
            break
```

CHAPTER 7

RESULT AND ANALYSIS

The face mask detector model is loaded into this script, the detector model is then used to make predictions and calculate the probability of a person wearing or not wearing a mask. These predictions are looped in for every frame and for each iteration the detector model is referenced and the values that were trained in the earlier training script are now used. Imutils was used to call the Video stream function which helps us initializing video streams over the network or with the help of local webcams on our personal computers. OpenCV was used to make frames and display data in the frames like the predictions and the probability.

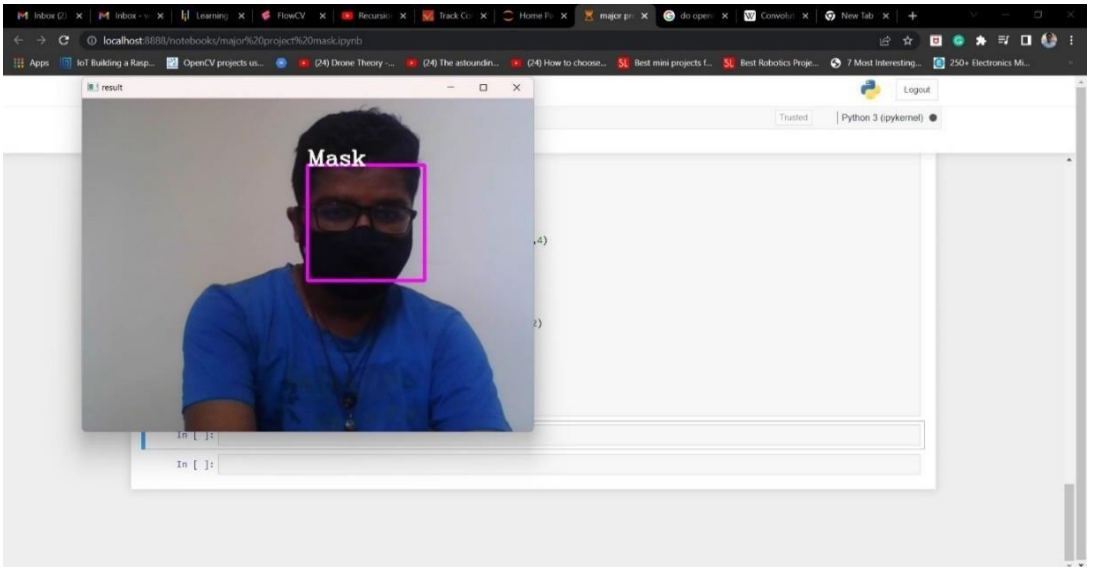

Figure 7.1: With mask

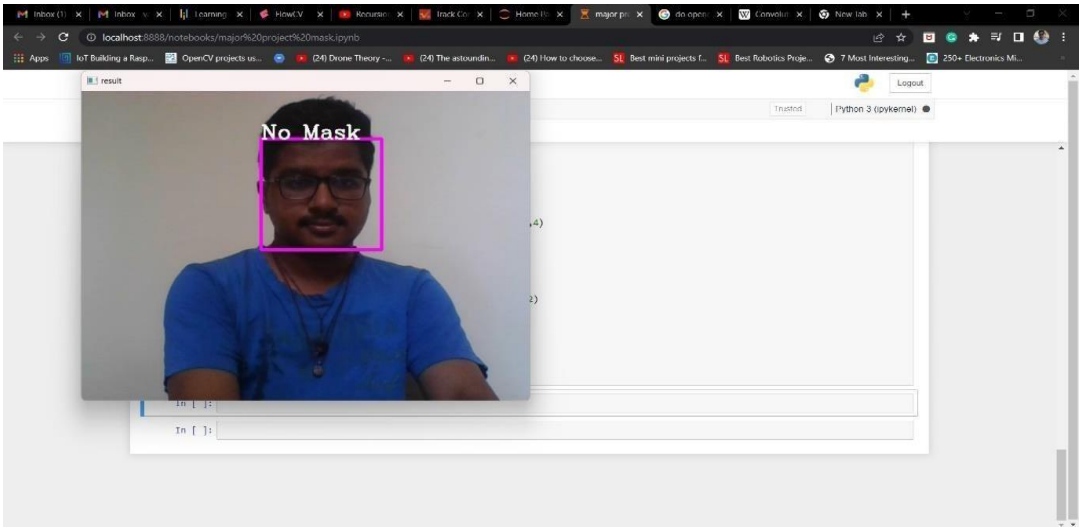

Figure 7.2: Without mask

We implemented our model on images containing one and more faces. We also implemented it on videos and live video streams by removing and wearing masks one by one. Some screenshots of the results are shown below:

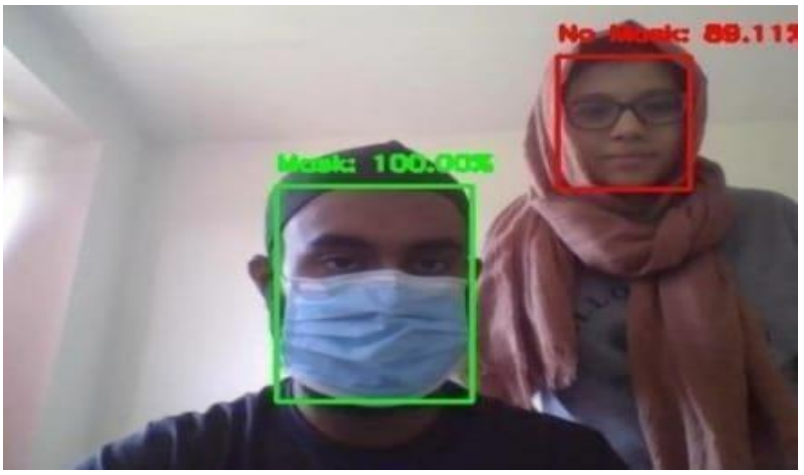

Figure 7.3: One person with mask and one person without mask.

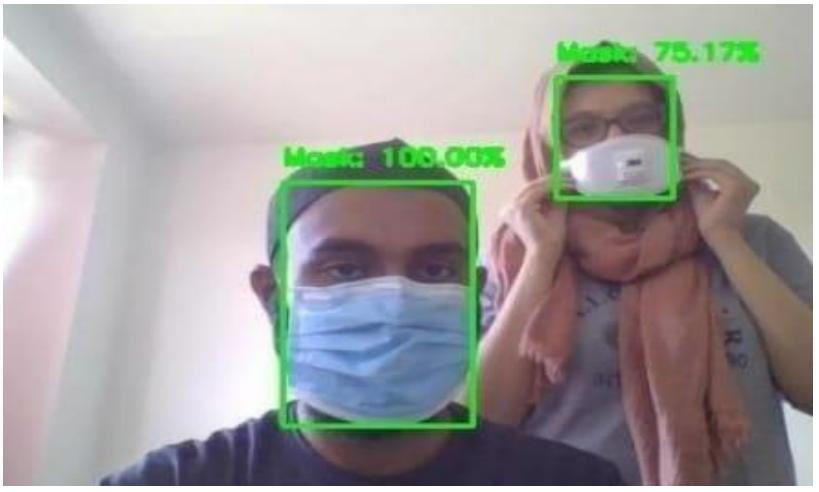

Figure 7.4: Both persons with mask.

Figure 7.5: Both persons without masks.

CHAPTER 8
ADVANTAGES AND APPLICATIONS

8.1 ADVANTAGES

As a key element in facial imaging applications, such as facial recognition and face analysis, face detection creates various advantages for users, including:

1. Improved security. Face detection improves surveillance efforts and helps track down criminals and terrorists. Personal security is also enhanced since there is nothing for hackers to steal or change, such as passwords.

2. Easy to integrate. Face detection and facial recognition technology is easy to integrate, and most solutions are compatible with the majority of security software.

3. Automated identification. In the past, identification was manually performed by a person; this was inefficient and frequently inaccurate. Face detection allows the identification process to be automated, thus saving time and increasing accuracy.

4. Disadvantages of face detection, while face detection provides several large benefits to users, it also holds various disadvantages, including:

5. Massive data storage burden. The ML technology used in face detection requires powerful data storage that may not be available to all users.

6. Detection is vulnerable. While face detection provides more accurate results than manual identification processes, it can also be more easily thrown off by changes in appearance or camera angles.

7. A potential breach of privacy. Face detection's ability to help the government track down criminals creates huge benefits; however, the same surveillance can allow the government to observe private citizens. Strict regulations must be set to ensure the technology is used fairly and in compliance with human privacy rights.

8.2 APPLICATIONS

1. It is majorily used in public places.
2. It is also used in colleges to detect the face masks.
3. It can be used in homes apartments, complexes, factories, hotels etc.
4. Low Cost

CHAPTER 9
CONCLUSION AND FUTURE SCOPE

9.1 CONCLUSION

To mitigate the spread of COVID-19 pandemic, measures must be taken. We have modeled a face mask detector using SSD architecture and transfer learning methods in neural networks. To train, validate and test the model, we used the dataset that consisted of 1916 masked faces images and 1919 unmasked faces images. These images were taken from various resources like Kaggle and RMFD datasets. The model was inferred on images and live video streams. To select a base model, we evaluated the metrics like accuracy, precision and recall and selected MobileNetV2 architecture with the best performance having 100% precision and 99% recall. It is also computationally efficient using MobileNetV2 which makes it easier to install the model to embedded systems. This face mask detector can be deployed in many areas like shopping malls, airports and other heavy traffic places to monitor the public and to avoid the spread of the disease by checking who is following basic rules and who is not.

9.2 FUTURE SCOPE

More than fifty countries around the world have recently initiated wearing face masks compulsory. People have to cover their faces in public, supermarkets, public transports, offices, and stores. Retail companies often use software to count the number of people entering their stores. They may also like to measure impressions on digital displays and promotional screens. We are planning to improve our Face Mask Detection tool and release it as an open-source project. Our software can be equated to any existing USB, IP cameras, and CCTV cameras to detect people without a mask. This detection live video feed can be implemented in web and desktop applications so that the operator can see notice messages. Software operators can also get an image in case someone is not wearing a mask. Furthermore, an alarm system can also be implemented to sound a beep when someone without a mask enters the area. This software can also be connected to the entrance gates and only people wearing face masks can come in.

REFERENCES

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001, vol. 1. IEEE, 2001, pp. I–I.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[3] R. Girshick, "Fast r-cnn," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in Advances in neural information processing systems, 2015, pp. 91–99.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in European conference on computer vision. Springer, 2016, pp. 21–37.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detect tion," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," 2017. ´

[8] Haddad, J., 2020. How I Built A Face Mask Detector For COVID-19 Using Pytorch Lightning. [online] Medium. Available at: https://towardsdatascience.com/how-i-built-a-face-mask-detector-for-covid-19-using pytorch-lightning-67eb3752fd61.

[9] Rosebrock, A., 2020. COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, And Deep Learning - Pyimagesearch. [online] PyImageSearch. Available at: https://www.pyimagesearch.com/2020/05/04/covid 19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/.