FACE MASK DETECTION
USING LIVE VIDEO STREAMING

Submitted in partial fulfillment of the requirements

of the degree of

B. E. Computer Engineering

By

Yogita Likhi	**44**	**182053**
Sherwin Mathias	**51**	**182061**
Ujala Maurya	**52**	**182062**
Nithin Menezes	**56**	**182066**

Supervisor:

Ms. K Priya Karunakaran

Assistant Professor

Department of Computer Engineering

St. Francis Institute of Technology

(Engineering College)

University of Mumbai

2021-2022

CERTIFICATE

This is to certify that the project entitled **"FACE MASK DETECTION USING LIVE VIDEO STREAMING"** is a bonafide work of **"Yogita Likhi(44), Sherwin Mathias(51), Ujala Maurya(52) and Nithin Menezes(56)"** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.E. in Computer Engineering

Ms. K Priya Karunakaran

(Guide)

Dr. Kavita Sonawane

(Head of Department)

Dr. Sincy George

(Principal)

Project Report Approval for B.E.

This project report entitled **FACE MASK DETECTION USING LIVE VIDEO STREAMING** by ***Yogita Likhi, Sherwin Mathias, Ujala Maurya and Nithin Menezes*** is approved for the degree of ***B.E. in Computer Engineering.***

Examiners

1. ----------

2.----------

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Yogita Likhi 44

Sherwin Mathias 51

Ujala Maurya 52

Nithin Menezes 56

Abstract

COVID-19 pandemic has rapidly affected our day-to-day life disrupting the world trade and movements. Wearing a protective face mask has become a new normal. With the advent of second wave of Coronavirus in India things have taken a nasty turn and wearing a mask have now become the necessity. All most all public service providers ask the customers to wear masks correctly to avail of their services. There are even rules on charging a fine from people who are disobeying the rules that are set. Therefore, an automated face mask detection system from live CCTV camera has become a crucial task to help the society.

In this report, we have proposed an automated system which helps authority people to identify the people who is not wearing a mask. The system will also notify to the end user by a sending him alert message. The proposed system in this report is developed with a deep learning algorithm MobileNetV2 for the first step which is face mask detection. For the next step i.e for identifying the face it uses face recognition library in python. We have used Kaggle dataset for face mask detection and for face recognition we have used our own dataset. The steps for building the model are collecting the data, pre-processing, split the data, testing the model, and implement the model. The face mask model can detect people who not wearing a mask and the through face recognition model we can identify the person.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

The year 2020 has shown mankind some mind-boggling series of events amongst which the COVID-19 pandemic is the most life-changing event which has startled the world since the year began. Recently, a study on understanding measures to tackle COVID-19 pandemic carried by the researchers at the University of Edinburgh reveals that wearing a face mask or other covering over the nose and mouth cuts the risk of Coronavirus spread by avoiding forward distance travelled by a person's exhaled breath by more than 90% [3]. Also, carried an exhaustive study to compute the community-wide impact of mask use in general public, a portion of which may be asymptomatically infectious in New York and Washington. The findings reveal that near universal adoption (80%) of even weak masks (20% effective) could prevent 17–45% of projected deaths over two months in New Work and reduces the peak daily death rate by 34–58% [4], [5]. Their results strongly recommend the use of the face masks in general public to curtail the spread of Coronavirus.

Further, with the reopening of countries from COVID-19 lockdown, Government and Public health agencies are recommending face mask as essential measures to keep us safe when venturing into public. People wear face masks once they step out of their homes and authorities strictly ensure that people are wearing face masks while they are in groups and public places. To mandate the use of facemask, it becomes essential to devise some technique that enforce individuals to apply a mask before exposure to public places. To monitor that people are following this basic safety principle, a strategy should be developed. A face mask detector system can be implemented to check this. In this project, we will be developing a system for detecting face mask detector that is able to distinguish between faces with masks and faces with no masks.

In this report, we have proposed an automation system for detecting people who is not wearing a mask then in next step recognizing the face of that person for taking certain against him/her.

1.1. Description

Now, we have understood the necessity of having face mask detection system in order to take appropriate action by recognizing the face of the people using face recognition system. The system in this report aims to develop surveillance system for recognition face of person who have not wear mask so that the authority can take certain action against him. The system takes frame from the live streaming video which is running in some public area for example college campus, building society, etc.

Fig 1.1 shows the working flow of whole system.

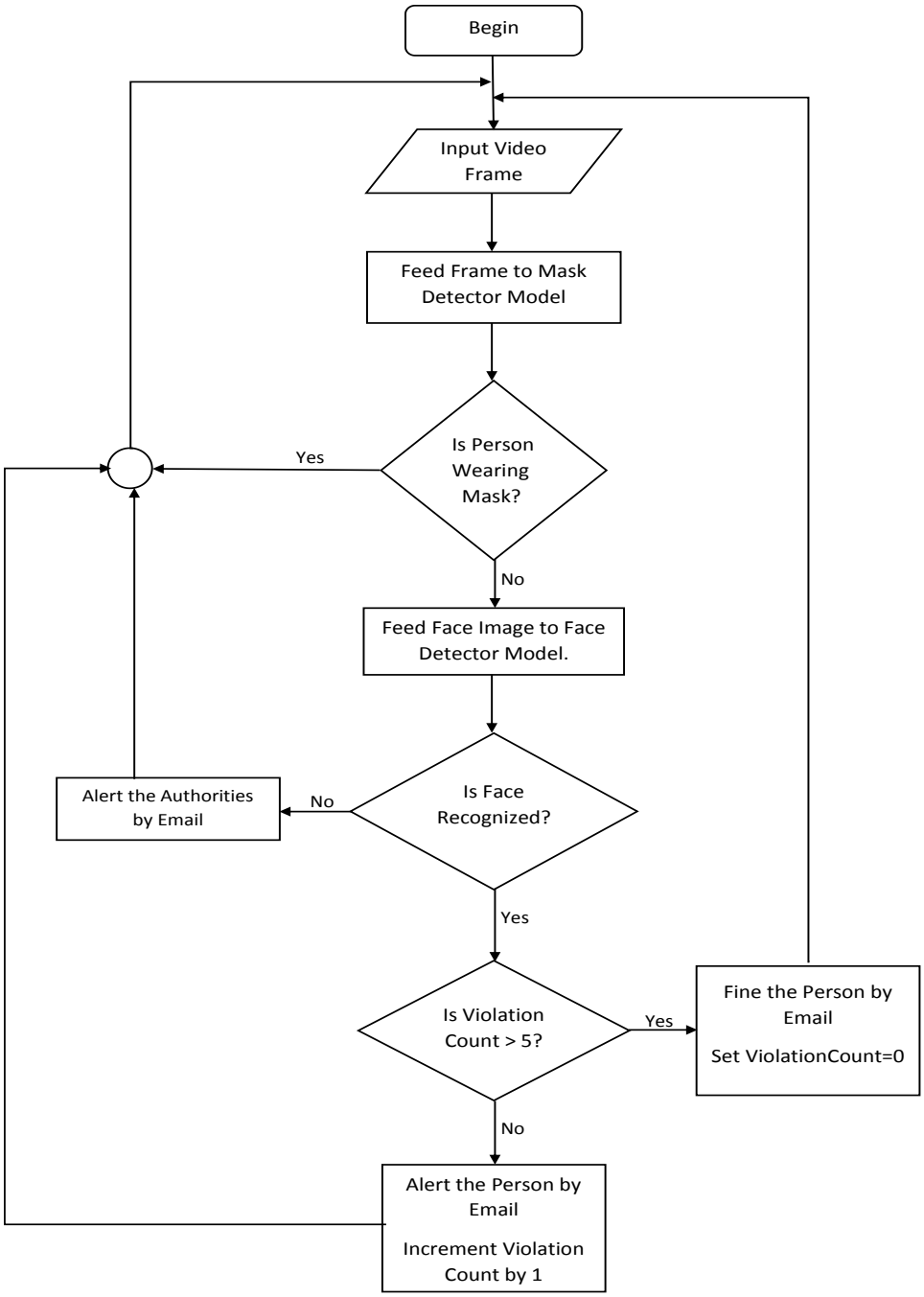

Fig. 1.1 work flow of proposed system

The major functionality of the system given below:

The proposed system can be divided into two parts, first is face mask detection system and second one is face recognition system. Face recognition part will only run if the person detected without mask so that his/her face can be recognize for future work.

At the first, System will take frame(image) as input from the Camera (IP Webcam). Then system will process this image using some pre-processing technique. After the pre-processing step it will send this image to face detection model which will classify each person's face present in the input image as "With Mask" (if mask is found on the face) and "Without Mask" (if mask is not found on the face"). The second step is to recognize the faces of each person which has been classified as "Without Mask". Since, face is properly visible in case of "without mask" it becomes easy to recognize them. Now these images will be sent to the face recognition model which will recognize the face if it is present in our database else it will show "Some Unknow Person Recognizes without mask". Now, from database system will fetch details of the person and accordingly it will send alert message via email system.

1.2. Problem Formulation

It is a necessity to wear a mask whenever we are at public places or places with a high probability of people gathering, to prevent the spread of the virus. A survey shows that 90% of people who participated in the survey are aware of all the guidelines issued by the Government of India as well as other organizations to protect from COVID-19 infection. However, when it came to overall practice, compliance was low especially among the lower social strata. Interestingly, only around 44% are completely compliant in terms of wearing it correctly and in all relevant situations.

Despite constantly spreading awareness on the necessity of wearing masks in public places, this rule is not being taken seriously. Manually it becomes difficult to keep check on who is abiding by the rule and who is not. A system is needed that will keep a check on people, especially in public and crowded places and crowded.

1.3. Motivation

To reduce spread of corona virus and to increase the safety there should be some system that can monitor who is abiding the rule of covid. For this till now the only way of

monitoring the people if they are wearing the mask or not was manual at many places. In crowded places with high probability of people this manual process of checking people becomes very difficult as it is very time consuming.

The motivation behind this project is that if we can take the help of technology like Image processing, Object detection and Deep Learning to detect whether people are wearing or not wearing masks in public places, it would be helpful to increase our safety. Also, if a person is detected as not wearing the mask their face will be recognized and using details from the database, we can send him/her alert via email and take certain action against them.

1.4. Proposed Solution

This system consists of two models, face mask detection model and a face recognition model.

1. **Building The Face Mask Detection Model**
    - The face mask detection model is developed with a Deep learning algorithm through the image classification method MobileNetV2.
    - MobileNetV2 is a convolution neural network which requires very low hardware components and it is faster than other available algorithm which makes this as efficient algorithm [1].
    - Brief explanation of MobileNetV2 is given in first section of literature review chapter.
    - The system uses two datasets, one for training face mask detection model and other for training face recognition model.
    - The first dataset (for mask detection model) is taken from the Kaggle dataset which includes thousands of records. The data which has been collected labelled into two groups; with and without a mask.
    - For accurate result, we have chosen such a dataset in which faces in each images are taken from different angle and from different distance.
    - Before the model creation, first we need to pre-processed the data i.e. resizing the images and then converting images into numpy array for further steps.
    - Then we split into two groups which are training data (80%) and testing data (20%). Each group is containing both of with-mask and without-mask images.

- Next step is to build the face mask detector model with base model MobileNetV2. and the last is saving the model for the future prediction process.
- Now, to make sure the model can predict well, we need to test the model. For this we will apply our trained model on testing dataset and compare the predicted output with actual output.

2. Building The Face Recognition Model

- The face recognition model is developed using FaceNet and MTCNN using one shot learning.
- MTCNN or Multi-Task Cascaded Convolutional Neural Networks is used to perform face detection.
- FaceNet is used for extracting features from an image of a person's face.
- Brief explanation on FaceNet and MTCNN in given in chapter 2 section 2.2.
- In the beginning of the project, we are going for a dataset of only ten students. Once the project runs successfully the size of the dataset can be increased covering all the students.
- Since one shot learning is used the model can be trained even with fewer images.
- The dataset containing images of students will be fed to the face recognition model.
- Faces will be detected in that frame using MTCNN.
- The next step is to perform face embedding using triple loss and Siamese network using the FaceNet algorithm.
- Once the face or faces in the frame has been recognized we can use to find out more details about that person to carry on with further process.

Once, both the model has successfully built with the required accuracy. Now, we will implement this model into our system and we need to connect it with live camera and front end of the system. For live camera, we can either use mobile phone camera using IP webcam or live CCTV camera if we have access of it and can connect it with the system. Video streaming is nothing but continuous frame of images. Our system will read frame after every interval of time, we can keep time interval of 2 seconds between the two frames. The detail working of system is provided in the description section.

For this project there will be two kind of end user one will be the authority who will get alert when any person gets detected without mask so that he/she can take certain action against the person and second will be the normal people of the society or college, etc.

Our project overcomes the drawbacks which is presented in Motivation section. The proposed system is an automated surveillance kind of system for detecting people without mask and storing his record in the database for future analysis. It will automatically detect people and recognizes faces and then sends alert to them via email and to the authority. There is no need of some third person to keep track of all this activity manually which is time consuming and there was a chance of fault.

1.5. Scope

The proposed system will work efficiently in ideal case. If everything which is required to run a system is given properly then system works excellently. It is not feasible for some condition like, it could not correctly classify partially hidden faces. System can be problematic in case of twins as if one of the twins is detected without mask it will recognize face of both the twins and sends alert to both the twins. Since, system requires a camera for capturing video feed, we need to keep camera at certain distance and hight. The camera has to be placed at a height of around 8 to 9 feet height so as to capture the faces properly. If the person is moving nearer to the camera, then face mask detection and face recognition is easy even in low camera resolution, however if field is very wide each person will be relatively small, requires high resolution camera.

System can be implemented to the Mall, Colleges, Restaurant, etc. to automate the face mask detection. We can also add some Beep sound to alert the security in case of without mask using arduino. It can be scale for detecting the social distance between the two-person centroid of each detected person.

Chapter 2

LITERATURE REVIEW

There are some previously available works that have been done related to the face mask detection. Specially in last year because of covid-19 pandemic. For literature review, we studied different approaches and papers that can be use for face mask detection and face recognition. One of this is using ResNet, which was proposed in 2015 by researchers at Microsoft Research a RNN (Residual Neural Network) based model. Since, RNN has less feature compatibility and it has the ability to take arbitrary output/input lengths which can affect the total computational time and efficiency. On the other hand, CNN (Convolution Neural Network) takes fixed input and gives a fixed output which allows it to compute the results at a faster speed. Hence, we found that MobileNetV2 (a CNN based model) is the best model for face mask detection which is developed by google.

Google researchers, in paper [1] have explained the working and architecture of MobileNetV2. MobileNetV2 is a significant improvement over MobileNetV1 and pushes the state of the art for visual recognition including classification, object detection and semantic segmentation. MobileNetV2 is released as part of TensorFlow-Slim Image Classification Library.

Figure [2.1] shows the overview of MobileNetV2 Architecture. MobileNetV2 builds upon the ideas from MobileNetV1 (previous version), using depth wise separable convolution as efficient building blocks. However, V2 introduces two new features to the architecture:
1) linear bottlenecks between the layers, and
2) shortcut connections between the bottlenecks.

Overall, the MobileNetV2 models are faster for the same accuracy across the entire latency spectrum. In particular, the new models use 2x fewer operations, need 30% fewer parameters. Also, MobileNetV2 is a very effective feature extractor for object detection and segmentation.

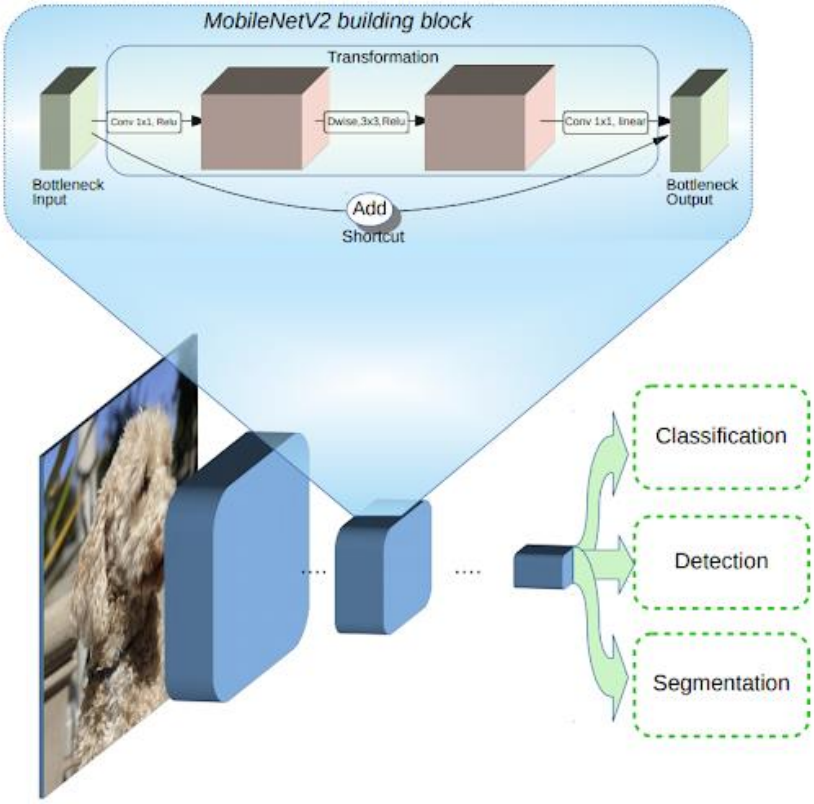

Fig 2.1 Overview of MobileNetV2 Architecture

2.1 Face Mask Detection:

2.1.1 Face Mask detection in image:

Samuel Ady Sanjaya and Suryo Adi Rakhmawan [2] have explained the face mask detection in images in their research paper. In this paper, they have taken image dataset from two sources.

The dataset was collected from the Kaggle dataset and the Real-World Face Masked dataset from public CCTV footage labelled with “with mask” and “without mask”. First, they have resized images, then stored dataset into Numpy array and pre-processed using MobileNetV2. Then, they have divided their dataset into training and (75%) testing data (25%). After this step they have built the model with the base model MobileNetV2 which involves three step adding model parameters, compiling the model, training the model. They also implemented this model in the video.

Fig 2.2 here shows their model building steps.

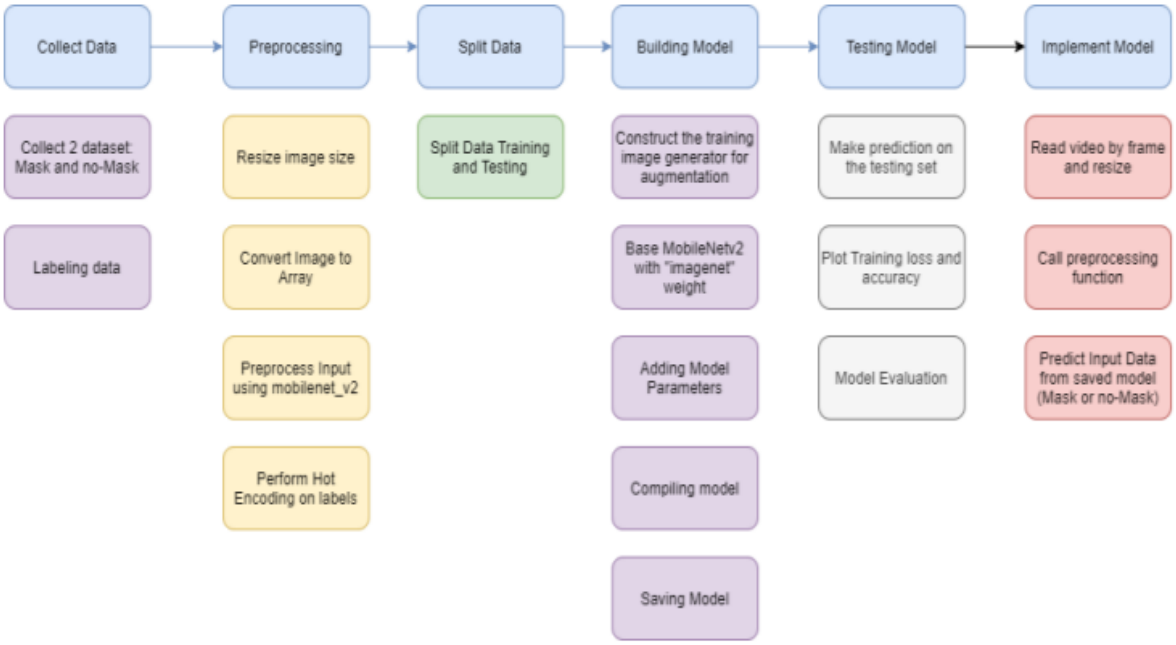

Fig 2.2 Model building steps of Face Mask detection in image

In which system reads frame one by one and process each frame. The built model can detect people who are wearing face mask and not wearing mask with an accuracy of 96.85%. Then they used the model for finding the % of people from 25 cities who were wearing a face mask. The percentage of people wearing face mask in the cities has a strong correlation to the vigilance index of COVID-19 which is 0.62. If the face is detected, it will display on screen using green rectangular shape (OPENCV). In this paper, video streaming had only single person per video with proper face visible. Finally, the model applied to the different set of images obtained from various sources to calculate the percentage of people wearing the mask.

2.1.2 Face Mask detection in video:

Shashi Yadav [3] has proposed the system that ensure the safety of the people at public places by automatically monitoring them whether they maintain a safe social distance, and also by detecting whether or not and individual wears face mask. They have adopted MobileNetV2 with transfer learning technique that can be used on real-time video. They have used custom dataset (3165 images) consisting of different types of face mask which are labelled and divided this dataset into training and testing datasets. Model was developed using 80% of dataset with MobileNetV2. Fig 2.3 here shows their proposed system architecture.

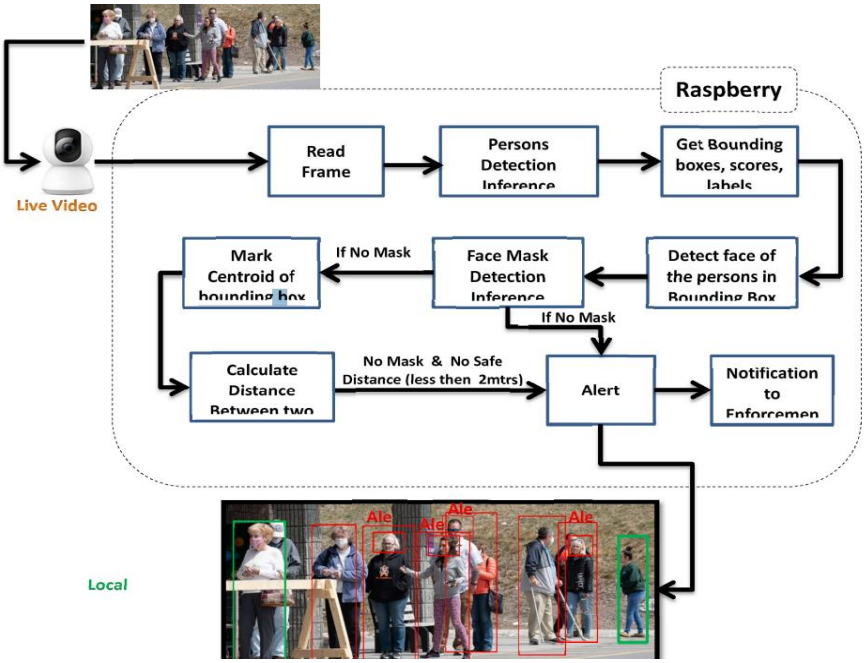

Fig 2.3 Model building steps of Face Mask detection in video

In Fig 2.3 they are reading each frame one by one, then at first step, there model detects person's whole body and draws bounding box around it. Then in each region of bounding box they are detecting the faces. If person detected with mask nothing will happen. But if detected without mask alert notification sent to the authority. Also, they are calculating distances between the two people, from the centroid of the bounding box. If it is less than 2 meters it generates alert notification.

Their proposed model first detects all persons in the range of camera and then detects faces with mask or without mask. If the mask is not visible in the faces, and if the social distance is not preserved, the system generates a warning and send alert to monitoring authorities with face image. The system detects the social distancing and masks with a precision score of 91.7% with confidence score 0.7, precision value 0.91 and the recall value 0.91 with FPS = 28.07.

2.2 Face Detection and Recognition:

Facial recognition is a way of identifying or confirming an individual's identity using their face. It can be used to identify people in photos, videos, or in real-time. Facial recognition systems can be implemented using various algorithms and models that are

available. Out of the many options available, FaceNet and MTCNN are the preferred ones.

[4] suggests the use of a face recognition pipeline which consists of three stages face detection, feature extraction and face classification respectively. It states that MTCNN or Multi-Task Cascaded Convolutional Neural Networks is used to perform face detection. It is a neural network which detects faces and facial landmarks on images. It was published in 2016 by Zhang et al. Further, it states that FaceNet is used for extracting features from an image of a person's face. Florian Schroff, Dmitry Kalenichenko and James Philbin have mentioned in [3] that it generates a high-quality face mapping from the images using deep learning architectures such as ZF-Net and Inception. Then it used a method called triplet loss as a loss function to train this architecture. It directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. Once this space has been produced, tasks such as face recognition, verification and clustering can be easily implemented using standard techniques with FaceNet embeddings as feature vectors.

2.2.1 Performance of FaceNet on input images

Table III in [5] clearly states how well FaceNet performs in case of different facial image datasets. It states how FaceNet is one of the new methods in face recognition technology but it's training process requires complex computing and a long time. To solve this problem, they have integrated TensorFlow learning machine and pre-trained model. This will reduce the training time needed.

This paper takes into consideration many datasets like Yale, JAFFE, AT&T, Georgia Tech, dan Essex etc. In the pre-processing stage the face image detection is carried out on each image of the MTCNN. The original image is cut down to a certain dimension according to the face area detected. Further, for training the model they have used pretrained models like CASIA-WebFace and VGGFace2. The pre-processed images are then trained using the pre-trained models and TensorFlow machine learning assistance as depicted in table II. At last, an accuracy testing of face faces in each face database is carried out. It uses eight different databases and the accuracy for each is calculated and displayed in table III.

Not only this, the accuracy results of FaceNet are compared with different methods used, that is Local Texture Description Framework-based Modified Local Directional Number (LTDF_MLDN) + Nearest Neighborhood Classifier (NNC), Eigenfaces use PCA and KPCA, The string of Grammar Fuzzy K-Nearest Strength (sgFKNN) and PCA + SVM. Table IV states how from the test values, the exactness using FaceNet algorithm is very good where at each dataset the face image produces recognitions that can reach up to 100%.

2.2.2 Using FaceNet for real-time face recognition

The application of face recognition is not limited to identifying people in images in fact, it has more real-world applications in the area of biometrics, Information security, access control, law enforcement, smart cards and surveillance system. For majority of these applications, we need to work with real-time or live input like one taken from the CCTV cameras or mobile, laptop or desktop webcams. Now we know that FaceNet has high performance and accuracy and works really well with images dataset. But to implement face recognition using FaceNet for some real-time and practical applications, we need to be sure that FaceNet and MTCNN works equally well with live video feeds.

Edwin Jose, Greeshma M. and Mithun Haridas have explained in [6] how they have implemented face recognition for a Surveillance System using FaceNet. This paper proposes a system that tracks the subject or the suspect with the camera ID/location together with the timestamp and logs his presence in the database, using multiple camera installation. Their system makes use of FaceNet, MTCNN and one shot learning. It also requires a hardware like CCTV camera which will provide the system with a live stream or video input. The system makes use of one-shot learning so that the system can favor small dataset. In case of one shot learning we can have a little less examples or images for each case.

In short [6] proposes a system that will read frames with OpenCV, detect face using MTCNN, perform face embedding with triplet loss and Siamese Network face Recognition using FaceNet algorithm, store the recognized face for analysis and to monitor intruder alert. The identified face will be exported into an Excel sheet for registering the suspects presence, along with the location ID of the place where the

target person was detected. Location ID will depend on the camera ID in which the intruder is caught.

2.3 To compare and analyze our face mask detection model with already built models.

As discussed in section 2.1 of chapter 2 we will be using MobilenetV2 to build the face mask detection model. There are various face mask detection model build using different algorithms. So, to check if our model is performing sufficiently well like other models, we need to analyze the performance of our model with others.

To do so we need to find face detection models that are already built and tested by someone else. Here, we have found two research papers that have used dataset from Kaggle to build their models. [8] uses Tensor Flow and Keras to build the model. It uses dataset from Kaggle [10] and have an accuracy of 94.58%. [9] uses YoloV3 to build the mask detection model using the same dataset from Kaggle with an accuracy of 87.16%.

Chapter 3

SYSTEM ANALYSIS

3.1 Functional requirements

- The system will allow the user to stream a live video stream.
- We also discussed brief overview and working of MobileNetV2 model which will be used to create face mask detection model.
- Using face mask detection system we can find out the people not wearing masks. These people can then be recognized via face recognition system.
- Once the person is recognized he can be warned using the alert system via a message or mail. If someone is caught doing the same mistake more than five times he/she will be fined.

3.2 Non-Functional requirements

- Performance: The person can be identified from the groups whether they are wearing a mask or not and at a click the authority can be notified so the response time is quite fast.
- Scalability: The system will work for small as well as a medium scale.
- Reliability: The system is reliable but will be dependent on the accuracy of the sentiment analysis of whether a person is wearing or not wearing a mask in a mass group in a live video stream.
- Maintainability: The system will be developed such that it can easily extend to incorporate additional functionalities.
- Availability: The system will be available to the user 24x7.
- Usability: The system will provide a user-friendly interface.

3.3 Specific requirements

Software Requirements:

Operating System: Windows 7

Python 3

Hardware Requirements:

CPU: Intel i3/Ryzen 3 or above

Clock speed: 2.5 GHz or above

GPU: NVIDIA Geforce GTX 690 or above

RAM size: 4GB or above

Hard Disk capacity: 100GB or above

3.4 Use-Case diagram and description

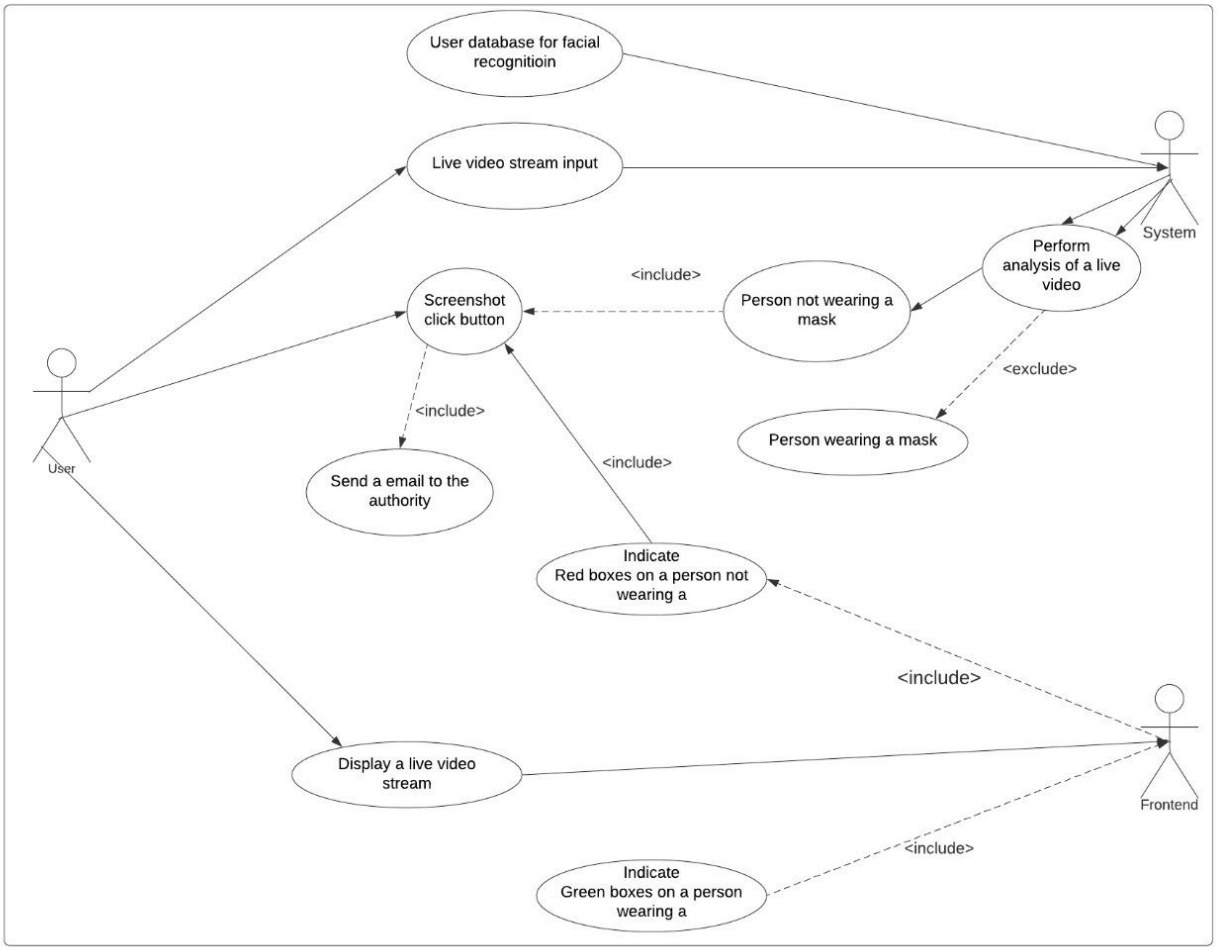

Fig. 3.1 Use-Case diagram

Use Case Diagram:

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and the use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "System" is something being developed and operated, such as a website. The "Actors" are people or entities operating under defined roles within the system. An example of this with reference to this system is shown in Fig. 3.1.

Use Case Specifications:

- **Use case:** Use database for facial recognition.

Brief Description: The dataset used consist of images of people wearing a mask for detection of mask in image as well as video and for face recognition we have used FaceNet and MobieNetV2 is used for proper classification, detection and Segmentation of the images.

Primary Actor: User

Main Flow:

- ➔ The system will access dataset of students or subjects frontal face images.
- ➔ The training of the dataset will be done using FaceNet model.
- ➔ In the real time Face detection and recognition reading frames will be done using OpenCV, face detection using MTNCC, Face embedding using FaceNet and face recognition with Classifier.
- ➔ All this will be stored for analysis of the face.
- ➔ he frontend will provide a screen where he can see the live stream going on through the webcam/CCTV.
- ➔ The user has to stream a live video streaming.

- **Use case:** Live video streaming input.
  Brief Description: This use case will expect the user to stream a live video.

  Primary Actor: User

  Main Flow:

  - ➔ The frontend will provide a screen where he can see the live stream going on through the webcam/CCTV.
  - ➔ The user has to stream a live video streaming.

- **Use case:** Perform sentiment analysis on the live video.
  Brief Description: The system will perform analysis on the live stream using the FaceNet model and detect people whether they are wearing a mask on not.

  Primary Actor: System

  Main Flow:

- ➔ The system will access dataset of students or subjects frontal face images.
- ➔ The training of the dataset will be done using FaceNet model.
- ➔ In the real time Face detection and recognition reading frames will be done using OpenCV, face detection using MTNCC, Face embedding using FaceNet and face recognition with Classifier.
- ➔ All this will be stored for analysis of the face.

Use case: Person wearing a mask.

Brief Description: The system will perform sentiment analysis on the live stream using the SVM algorithm and detect people whether they are wearing a mask on not.

Primary Actor: System

Main Flow:

- ➔ The green box is generated over a person wearing a proper face mask.
- ➔ Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. They were used to create a proper dimension box over the face.

- **Use case:** Person not wearing a mask.

  Brief Description: The system will perform sentiment analysis on the live stream using the SVM algorithm and detect people whether they are wearing a mask on not.

  Primary Actor: System

  Main Flow:

  - ➔ The red box is generated over a person not wearing a proper face mask.
  - ➔ Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. They were used to create a proper dimension box over the face.

- **Use case:** Screenshot click button.

  Brief Description: The system will alert the user whenever they are not wearing the mask by showing a red box, the user can then within a screenshot button give the authority a notification regarding the same.

Primary Actor: User, System, Frontend

Main Flow:

- ➔ The red box is generated over a person not wearing a proper face mask.
- ➔ The user can see the red box on the screen displayed over the live stream the user gave as input, from there the user can get a screenshot can a single click along with the live footage the user can inform the authority or can take hold of that person.

- **Use case:** Email/ notify the authority.

  Brief Description: The system will alert the user whenever they are not wearing the mask by showing a red box, the user can then within a screenshot button give the authority a live proof of a person not abiding the rules along with the system date and time. But the following condition is optional.

  Example: If the system is implemented in our college according to our database of students we can get to know how many times a particular student has not abided the rule. If count is till 5 then inform the authority otherwise not.

  Primary Actor: User, System, Frontend

  Main Flow:

  - ➔ The red box is generated over a person not wearing a proper face mask.
  - ➔ The user can see the red box on the screen displayed over the live stream the user gave as input, from there the user can get a screenshot can a single click along with the live footage the user can inform the authority or can take hold of that person.

- **Use case:** Display a live video stream.

  Brief Description: In order for the system to display the output, the user has to first stram a live video

  Primary Actor: User

  Main Flow:

  - ➔ The red and green boxes will be shown indicating a no mask or mask in the following stream. The live stream is possible through a webcam/ CCTV camera.

- **Use case:** Indicate red box over a person not wearing a mask.

  Brief Description: The system will alert the user whenever they are not wearing the mask by showing a red box, the user can then within a screenshot button give the authority a notification regarding the same.

  Primary Actor: Frontend

  Main Flow:

  ➔ The red box is generated over a person not wearing a proper face mask.

  ➔ The user can see the red box on the screen displayed over the live stream will alert the user.

- **Use case:** Indicate green box over a person wearing a mask.

  Brief Description: The system will alert the user whenever they are wearing the mask by showing green box.

  Primary Actor: Frontend

  Main Flow:

  ➔ The green box will show that the following person is abiding the rule by wearing a proper face mask.

Chapter 4

ANALYSIS MODELING

4.1 Data Modeling

Our data model consists of following entity as displayed in fig. 4.1:

User: User entity is for the person or user who is member of the college or society. It has user_id as it's primary key. User entity consist of user_name, user_email, user_images and counts_of_violation attributes to store the information of the user.

Dataset: This entity has image data in the form of array which represents the pixel value along with label attributes.

Model: Model entity has encoding of each face of the dataset associated with unique_id by which we can fetch details from user database. Model predicts user who has detected without mask.

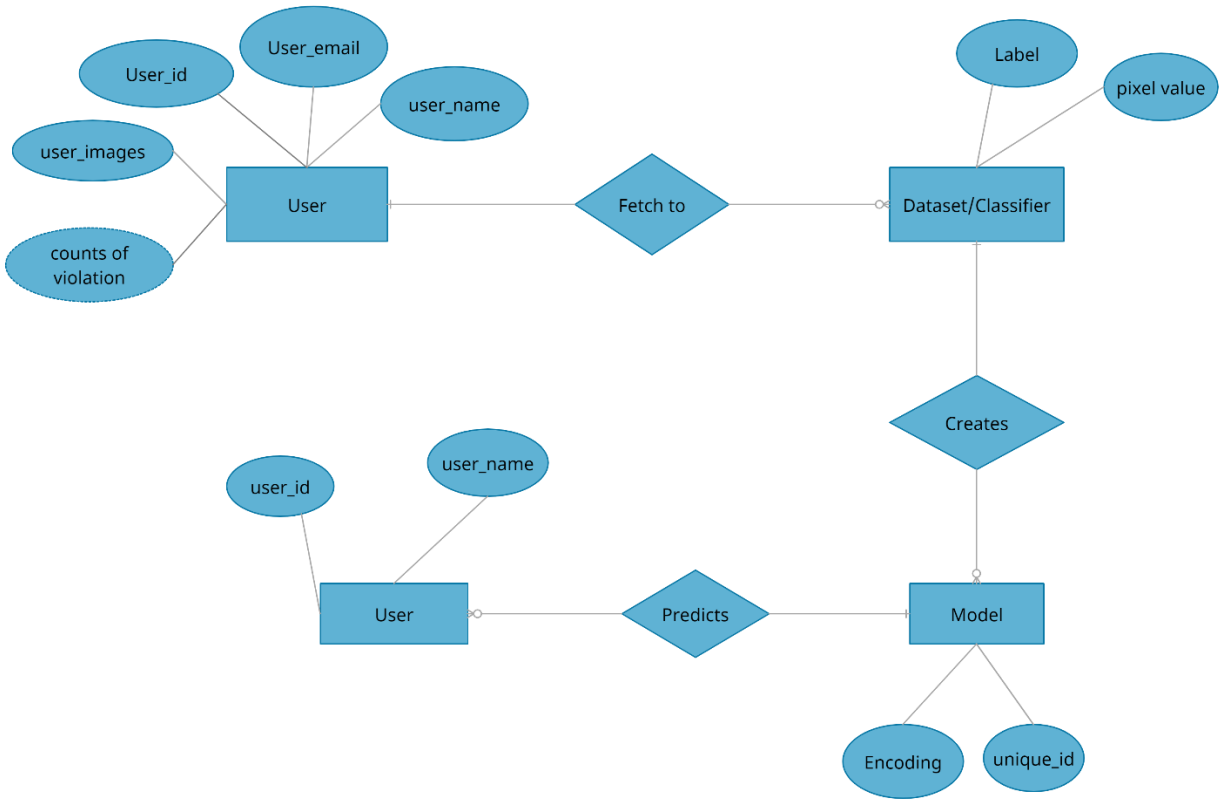

Fig 4.1 E-R model

Table: User/Person

Sr. no.	Attribute Name	Data Type	Field size	Description	NULL Allowed
1	User_id	NUMBER	9999	Keeps unique id for user	NO
2	User_name	STRING	50	stores name for user	NO
3	User_email	STRING	30	stores email of the user	NO
4	User_images	ARRAY	NA	Stores the images of the user	NO
5	Count_of_violation	NUMBER	5	Stores count for the no. of violation	NO

Fig 4.2 Data Dictionary

4.2 Class Diagram

Our project consists of the following classes as displayed in Fig. 4.3:

- **Camera:** Camera class is useful for camera related work. It has attributes like image_width integer type, image_height integer height and address_of_camera string type. For reading camera link it has read_add() method, for live stream video input_video() and for reading frame one by one it has read_frame() method.
- **User/Person:** This class has all user information which consists attributes like user_id(int), user_name(string), user_email(string) adnd coun_of_violation(int). It has two methods get_email() to fetch email of user for sending alert message and get_count() to check the number of violation done by the user.
- **Image Pre-processing:** It has two attribute image_size(tuple<int>) and image (numpy array). Methods for this class includes resize_images(), image_to_array(), pre_process() and apply_encoding_on_label().
- **Face Mask Detection:** It has attributes like images(array), accuracy_value(int) and output_label(string). Methods for this class includes input_images(), detect_mask() for detecting people without mask, test_accuracy() for calculating accuracy of model with given input image and goto_facedetection() methods to identify the face of the person.

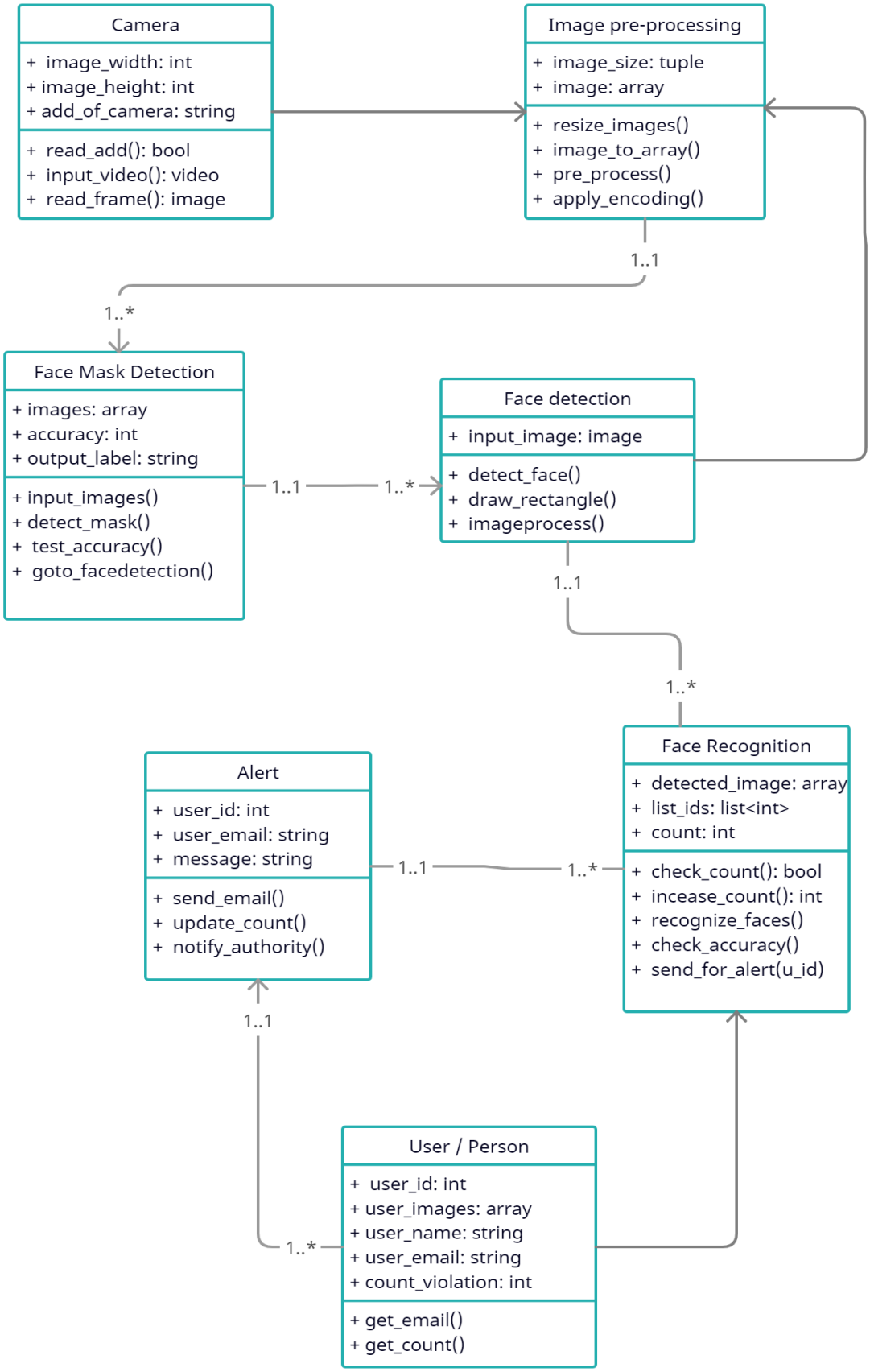

Fig 4.3 Class diagram of proposed system

- **Face Detection:** This class has only one attribute which is input_image for storing image after face mask detection. It has methods like detect_face() for identifying the face of people in images, draw_rectangle() to draw rectangle around detected faces.
- **Face Recognition:** Face recognition class has attributes like list_ids(int) which has list of all the unique id's, count for updating count of violation and images(array) for storing detected image. Methods includes check_count(), increase_count() to update value of count for violation, recognize_faces(), check_accuracy() to calculate accuracy of model with given image and send_for_alert(user_id).
- **Alert:** Alert class has user_id for uniquely identifying user, user_email for storing email of user and message to store the message which has to send. This class includes method like send_email() for sendind email to the user, update_count() to update value of count for violation in user database and notify_authority() to send notification to the frontend user or to the authority person.

4.3 Functional Modeling

4.3.1 DFD Level 0:

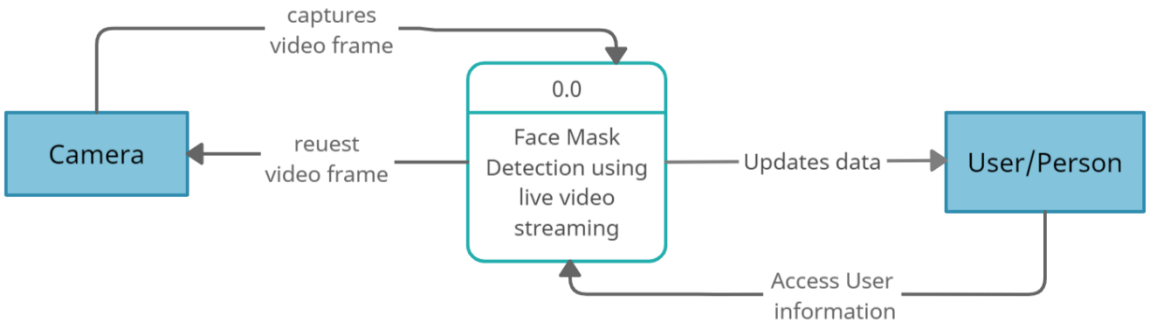

Fig 4.4 Data flow diagram (Level 0)

Fig 4.4 displays the level 0 Data Flow Diagram (DFD) also known as a context diagram of the Face Mask detection system using live video streaming. At level 0, it shows only abstract view of the entire system, showing the system as a single process with its relationship to external entities which is User entity and Camera entity. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

4.3.2 DFD Level 1:

Fig 4.5 displays the level 1 Data Flow Diagram (DFD) of the face Mask detection system using live video streaming. A level 1 DFD notates each of the main sub-processes that together form the complete system. It depicts basic modules in the system and flow of data among various modules. Level 1 DFD also mentions basic processes and sources of information. we can think of a level 1 DFD as an "exploded view" of the context diagram (level 0 diagram).

Level 1 DFD has four main process which is face mask detection (1.1), face recognition (1.2), Increase count of violation (1.3) and Alert message (1.4). It has three main external entity which is Camera entity, User/Person entity and Frontend entity. It includes model for face mask detection and face recognition along with User database. The detail flow of the input and output data and other important module is shown in the diagram.

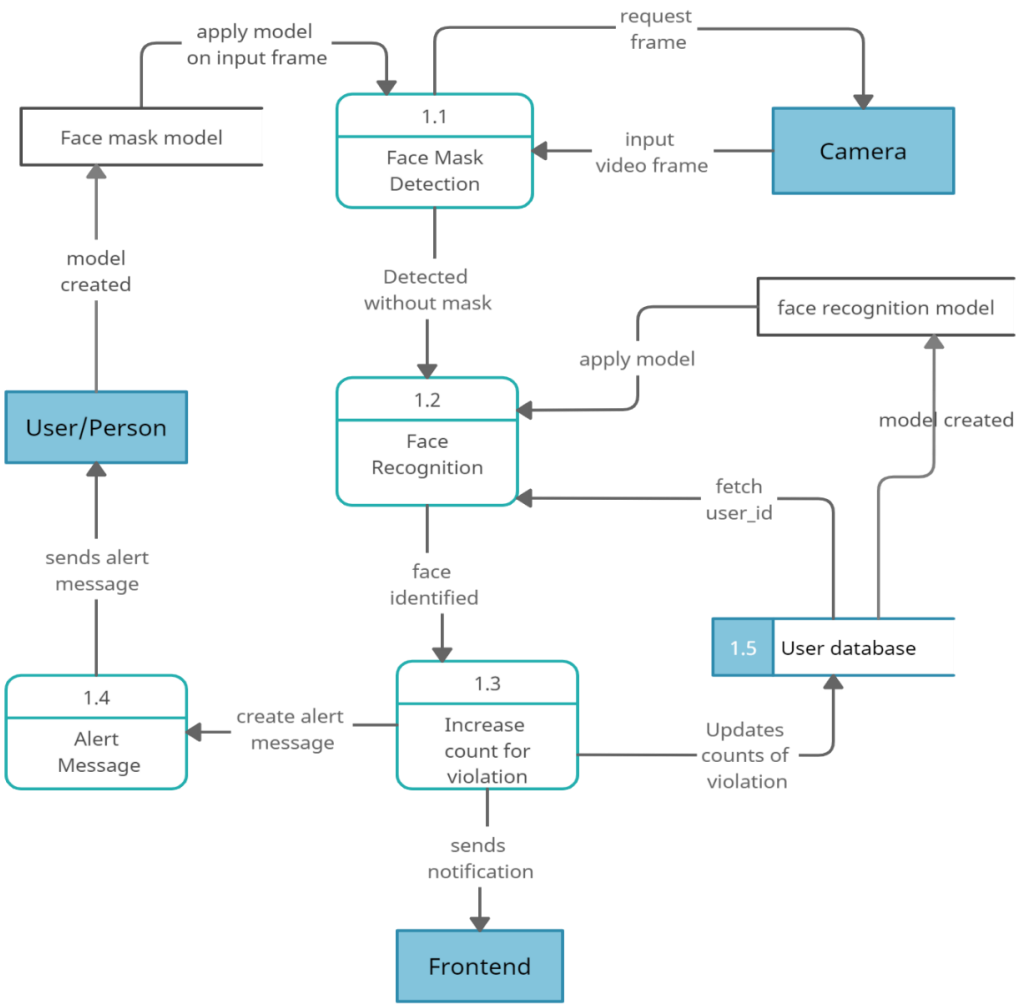

Fig 4.5 Data flow diagram (Level 1)

4.4 TimeLine Chart

Fig 4.6.1 and 4.6.2 displays a timeline chart diagram of the face Mask detection system using live video streaming. A timeline chart makes it easier to conceptualize a process or a sequence of events. It makes easier for us to understand the intricacies of our project or why something seems to take so long to accomplish.

It helps us to manage and keep our complex tasks on time. It allows us to visualize each process or events that took place over a period of time.

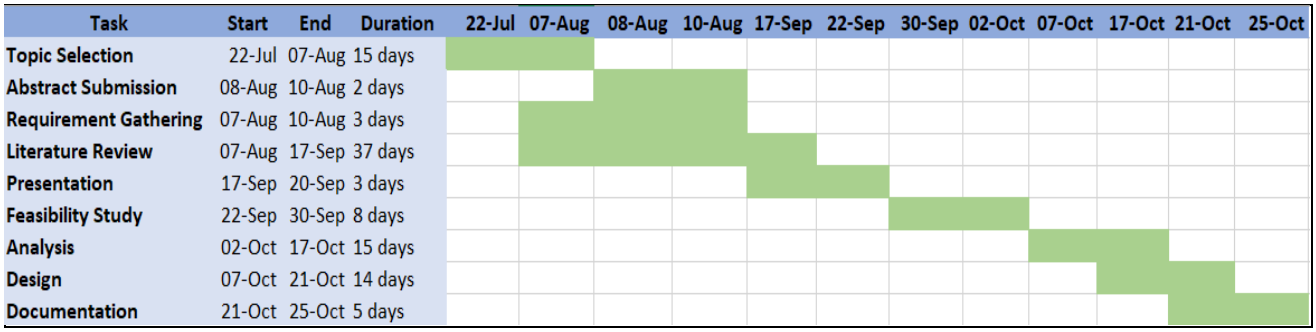

Fig 4.6 Timeline Chart-01 (July-Oct)

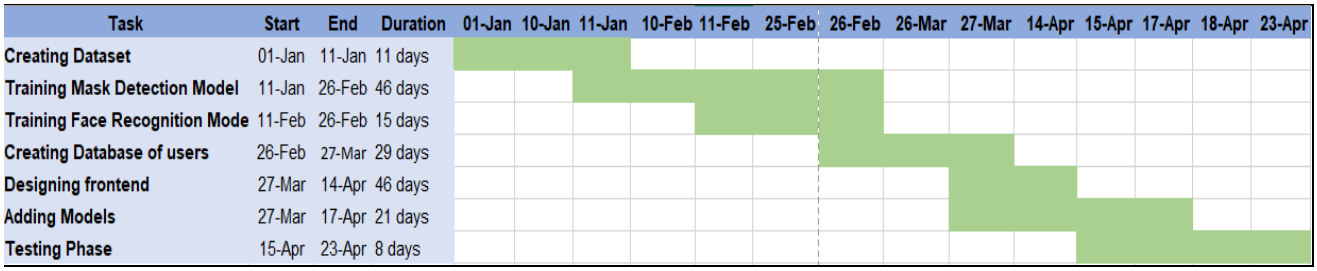

Fig 4.7 Timeline Chart-02 (Jan-Apr)

Chapter 5

DESIGN

5.1 Architectural Design

Fig. 5.1 shows the architectural design of the application. It defines a structured solution to meet all the technical and operational requirements.

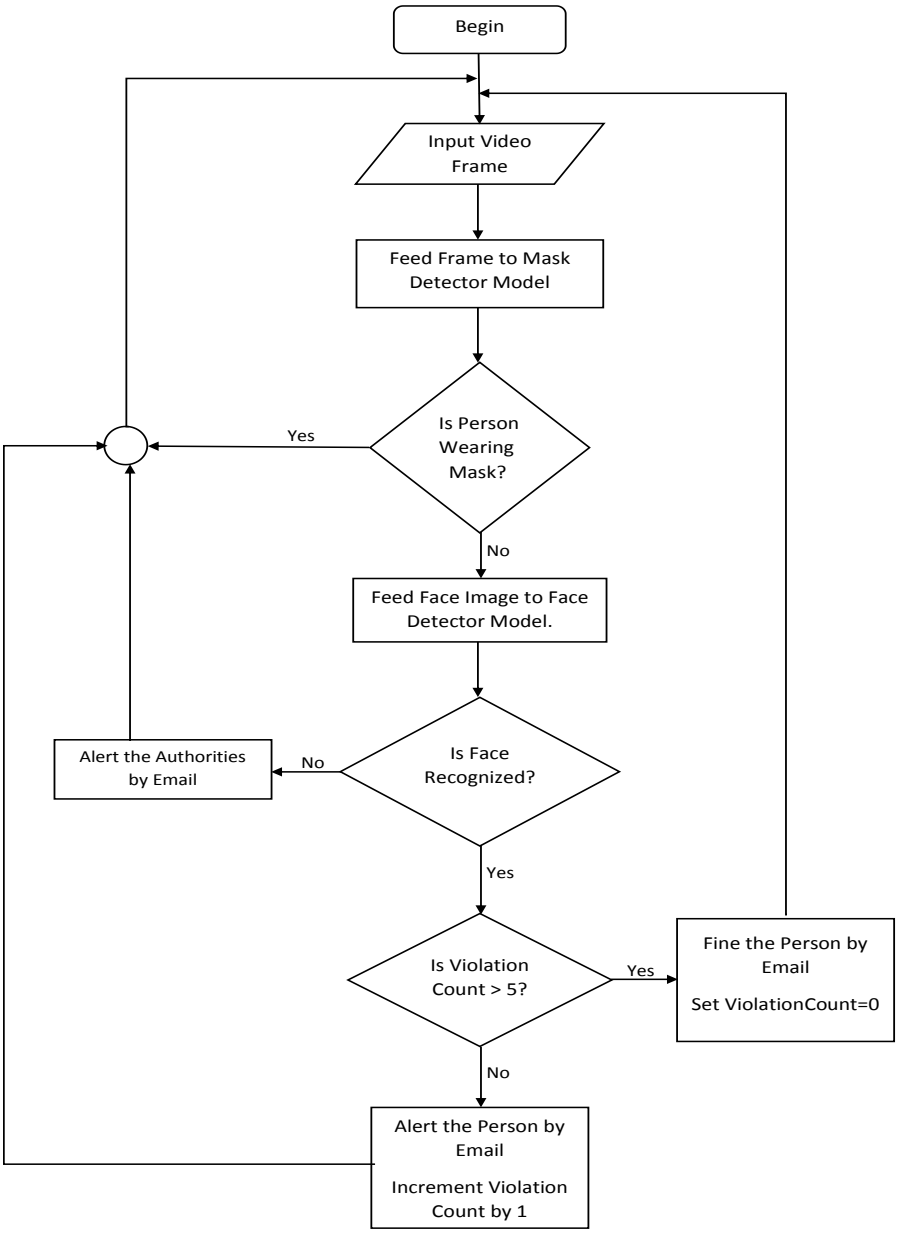

Fig. 5.1 Architectural Diagram

For the training, we will be using an Image dataset on which our model will train. This dataset will have two sections – Image and Category. For each image in the dataset, we will augment the image and generate multiple varying copies of it. The next step is to convert each image and its corresponding category into a numpy array. We will append each resultant numpy array of Image and corresponding category into Input_Data_Array and Category_Array respectively. In this way, we have our training input ready which will be used to train the model.

After this, we will split our dataset into 80:20 ratios. 80% of the input dataset will be used to train the model and 20% of the dataset will be used to test the model. After tuning the hyper parameters for our model, we will begin the training of the model with the help of MobileNet V2 Convolutional Neural Network architecture. Completing the training phase, we will check the accuracy of our model. If the model is having a good and desirable accuracy, we will save the trained Mask Detector Model. Else, we will repeat the step of tuning the hyper parameters and training the model until we get the desirable accuracy.

5.1.1 Training of Mask Detector Model

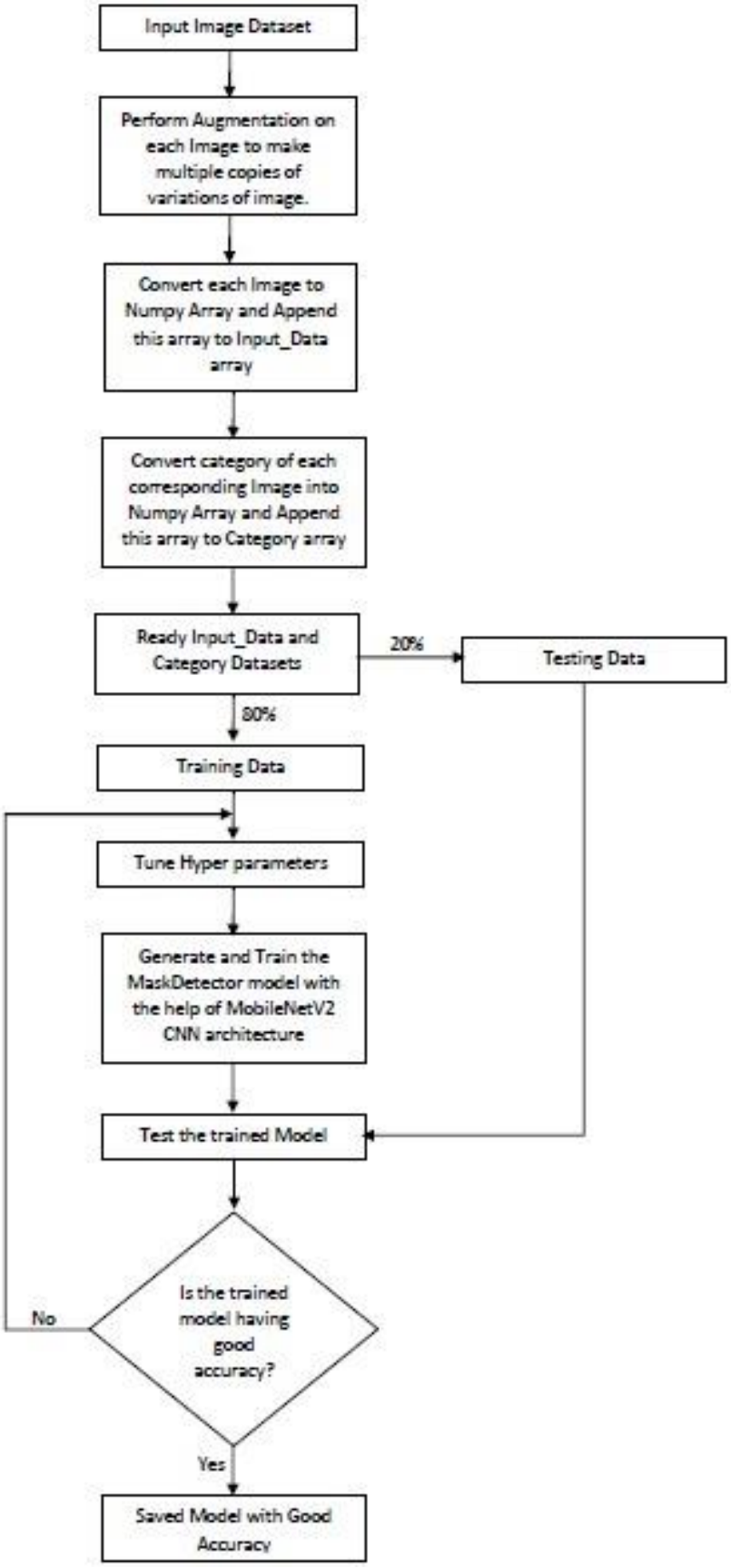

Fig. 5.2 Training of Mask Detector Model

As shown in Fig 5.2 for the training, we will be using an Image dataset on which our model will train. This dataset will have two sections – Image and Category. For each image in the dataset, we will augment the image and generate multiple varying copies of it. The next step is to convert each image and its corresponding category into a numpy array. We will append each resultant numpy array of Image and corresponding category into Input_Data_Array and Category_Array respectively. In this way, we have our training input ready which will be used to train the model.

After this, we will split our dataset into 80:20 ratios. 80% of the input dataset will be used to train the model and 20% of the dataset will be used to test the model. After tuning the hyper parameters for our model, we will begin the training of the model with the help of MobileNet V2 Convolutional Neural Network architecture. Completing the training phase, we will check the accuracy of our model. If the model is having a good and desirable accuracy, we will save the trained Mask Detector Model. Else, we will repeat the step of tuning the hyper parameters and training the model until we get the desirable accuracy.

5.1.2 Workflow of Mask Detector Model

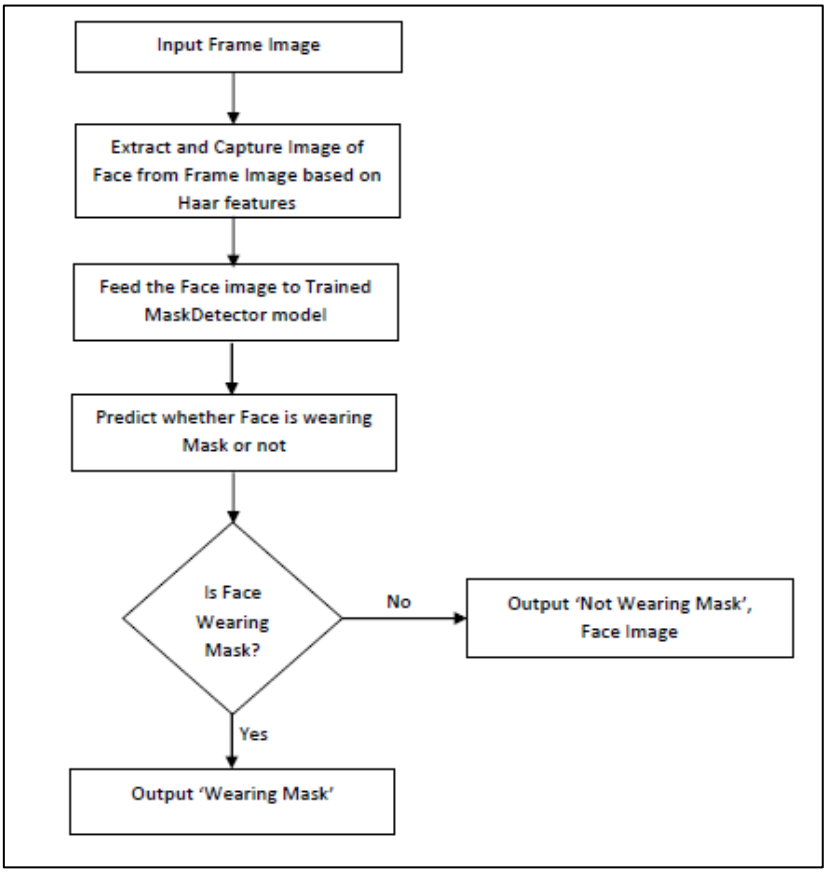

Fig. 5.3 Workflow of Mask Detector Model

In Fig 5.3 the live video will be continuously fed to the Mask Detector Model, one frame at a time. We will be using Haar Cascade Face Detector to capture human face from the video frame. Haar Cascade Face Detector has already pre-trained weights which help in detecting the haar features. Based on alignment of those features, the human face will be detected and captured. This face image will be fed to the Trained Mask Detector Model by us. The Mask Detector Model will come to the conclusion that whether the face is wearing the mask or not. If the face is wearing the mask, the Mask Detector Model will input 'Wearing Mask' else it will output 'Not Wearing Mask'.

5.1.3 Training of Face Detector Model

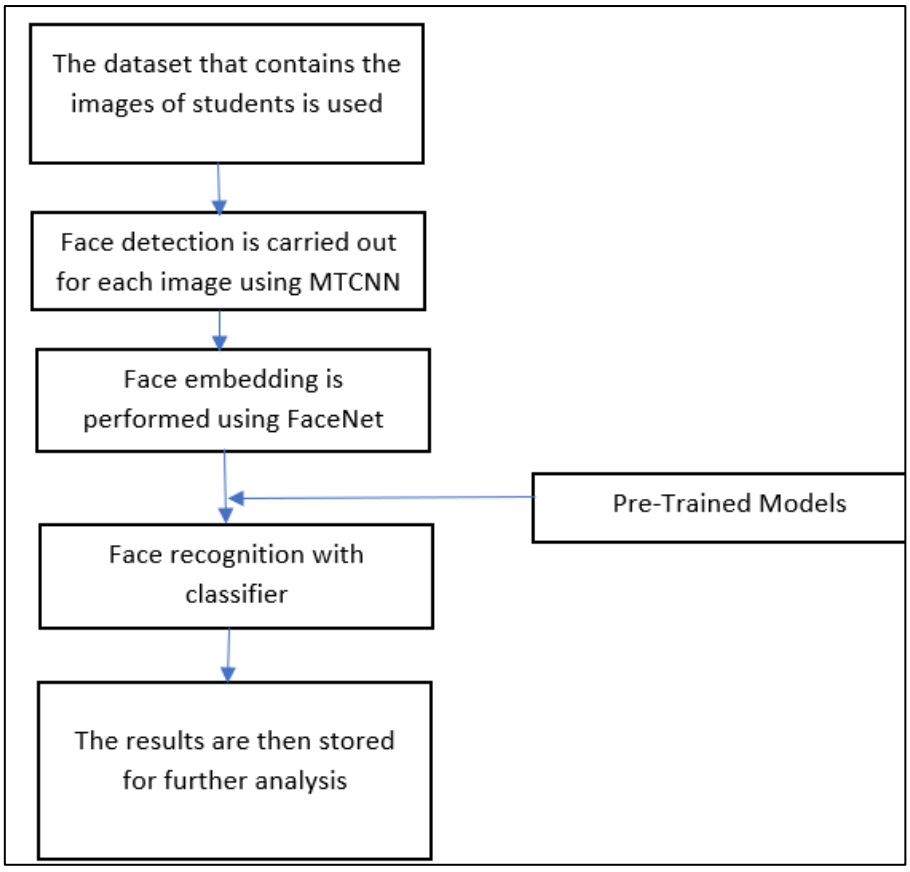

Fig. 5.4 Training of Face Detector Model

As shown in Fig 5.4 the output from the face mask detection model is fed to the face recognition model. Thus, the input to the face recognition model is basically a frame from the live video which has captured people without masks. The task of face recognition model is to identify those people.

So, to build this model we will need a dataset of Face Images of students. At the beginning we are using images for only ten students but this will be increased later. These images than undergo face detection using MTCNN. When face detection is successful, the original image on the dataset with the size of x pixels by y pixels is done by cutting according to the detected face area with a size of 182 pixels by 182 pixels. Next step is to perform face embedding using FaceNet algorithms. These are then passed onto train the model using Siamese network face recognition which is a part of the FaceNet algorithms and some pre-trained models. The results are then stored which we will use for further analysis.

5.1.4 Workflow of Face Detector Model

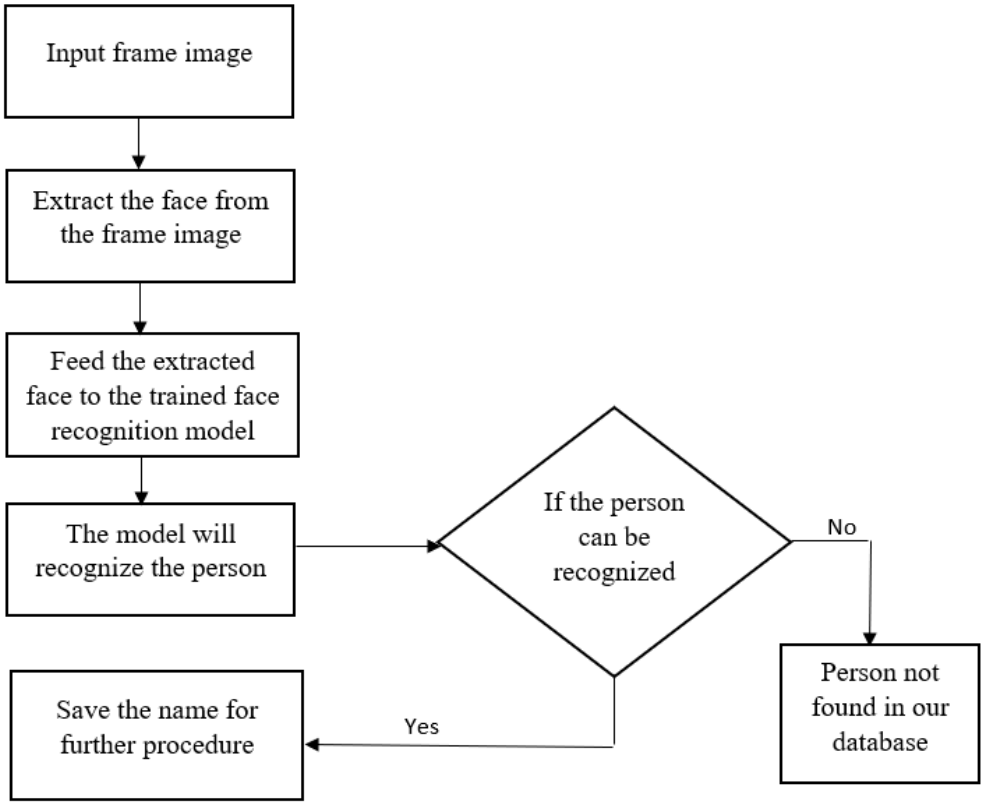

Fig 5.5 Work Flow of face detector

In Fig 5.5 the Face Detector model will input the frame image from the mask detector model and extract and capture the image of the person not wearing the mask. The model will then identify the person in the image. Now, if the model could recognize that person then the name of that person will be saved for taking action against him/her. In case the model could not identify the person it means that the information of that person if not available in our database and the same will be displayed.

5.2 User Interface Design

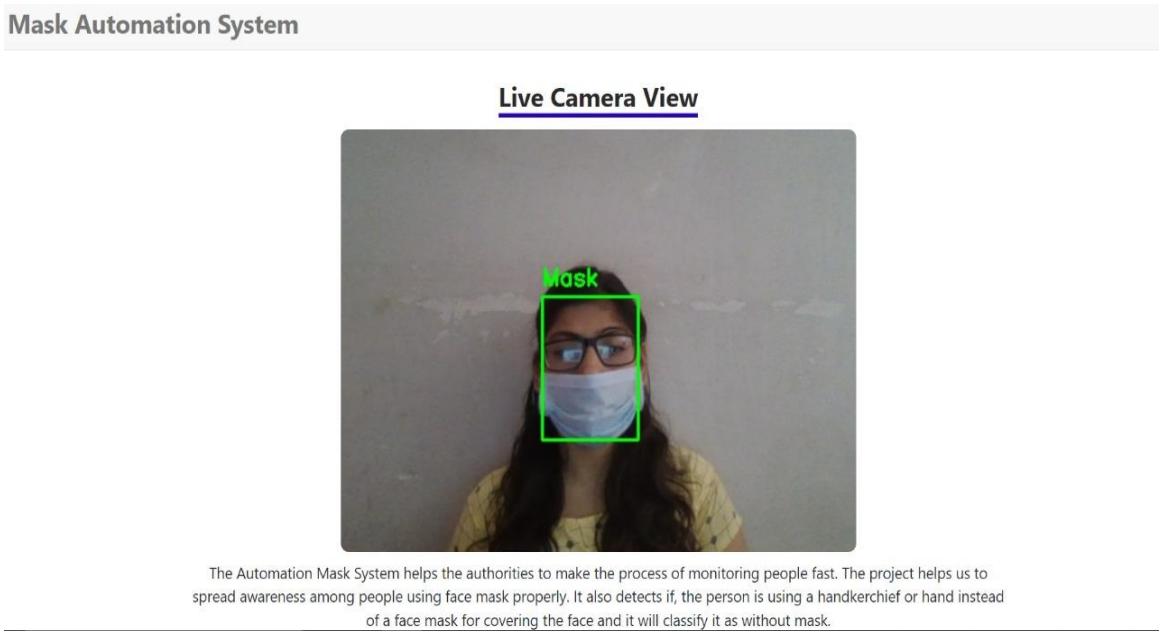

Fig 5.6 Main Page

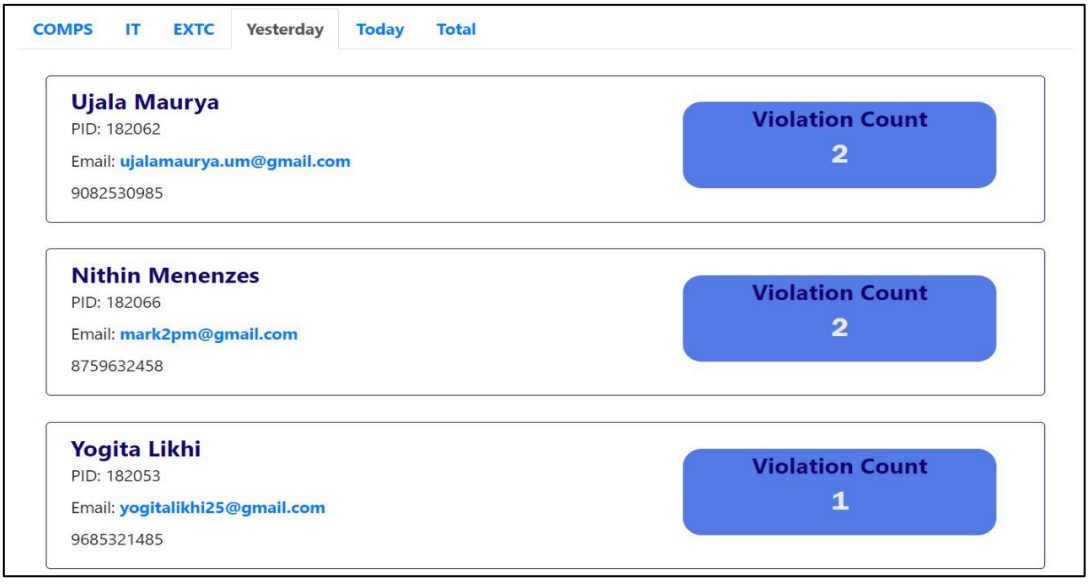

Fig 5.7 Violation Details Page

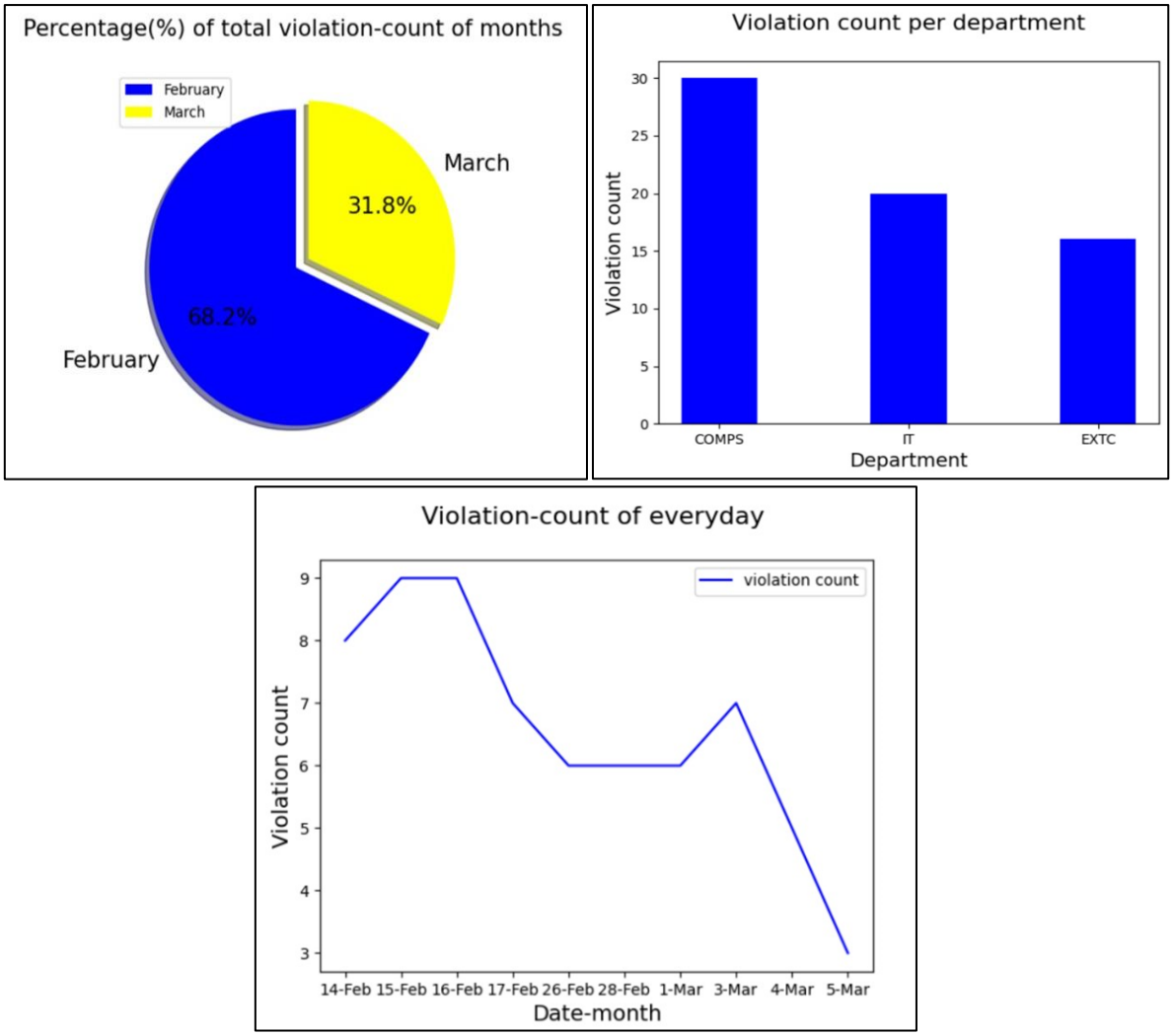

Fig 5.8 Graph Visualization Page

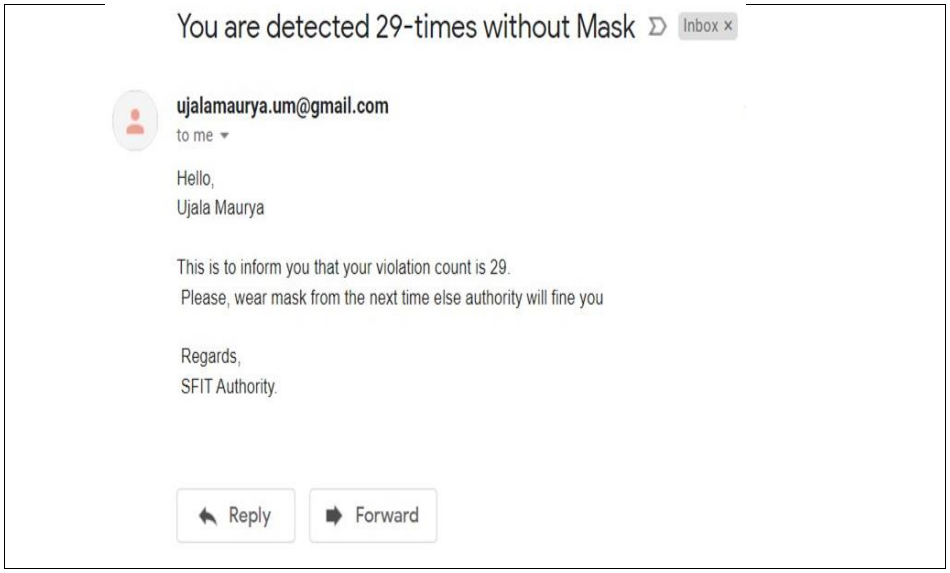

Fig 5.9 Alerting via Gmail

Chapter 6

IMPLEMENTATION

6.1 Algorithms / Method

Mask Detection: In our project we are using CNN based MobileNetV2 Architecture for Mask Detector Model. For live streaming as well as capturing of faces we are using OpenCV along with Haar Face Detector.

Face Recognition: For Face Recognition model, we are using FaceRecognition library to extract encodings of captured faces and FaceNet model to compute the similarity between faces. For live streaming as well as capturing of faces we are using OpenCV along with Haar Face Detector.

6.2 Working of the project

6.2.1 Training the Mask Detector Model

The below code shows the training and saving the mask detector model:

```
# import the necessary packages
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
```

```
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import tensorflow.keras.backend as k
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

#Initialize the parameters(Initial Learning Rate, No of epochs, Batch Size)
INIT_LR = 1e-4
EPOCHS = 20
BS= 32

DIRECTORY = r"C:\Users\SHERWIN MATHIAS\Desktop\project\dataset"
CATEGORIES = ["with_mask","without_mask"]

#Initialize the list of data and class images
data=[]
labels=[]

for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = load_img(img_path, target_size=(224,224))
        image = img_to_array(image)
        image = preprocess_input(image)
        data.append(image)
```

```
        labels.append(category)
#Perform One-Hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

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

#Divide the set into Train Set and Test Set
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.2, stratify=labels,
random_state=42)

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

basemodel = MobileNetV2(weights="imagenet", include_top=False,
                    input_tensor=Input(shape=(224,224,3)))

# Construct the head of the model that will be placed on top of the base model
headmodel = basemodel.output
headmodel = AveragePooling2D(pool_size=(7,7))(headmodel)
headmodel = Flatten(name = "flatten")(headmodel)
headmodel = Dense(128, activation="relu")(headmodel)
headmodel = Dropout(0.5)(headmodel)
```

```
headmodel = Dense(2, activation="softmax")(headmodel)

#Place the headmodel on top of the basemodel
model=Model(inputs=basemodel.input, outputs=headmodel)

#Loop over all layers in basemodel and make them non-trainable so that they will not be updated during first training process
for layer in basemodel.layers:
    layer.trainable = False

#Compile the model
optimizer=Adam(lr=INIT_LR, decay=INIT_LR/EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=optimizer, metrics=["accuracy"])

#Training the headmodel
HEAD = model.fit(aug.flow(trainX, trainY, batch_size=BS),
            steps_per_epoch=len(trainX)//BS,
            validation_data=(testX,testY),
            validation_steps=len(testX)//BS,
            epochs=EPOCHS)

#Make predictions on the testing set
predIdxs = model.predict(testX, batch_size=BS)

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

print(classification_report(testY.argmax(axis=1), predIdxs,
                    target_names=lb.classes_))

model.save("mask_detector.model", save_format="h5")
```

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

6.2.2 Live Mask Detection:

The below code shows the detection of the mask on the face:

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

def detect_and_predict_mask(frame, faceNet, maskNet):
    #grabbing the dimensions of the frame and constructing a blob from it
    (h,w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1, (224,224), (104,177,123))
        #pass the blob throught the network and obtain the face detection
```

```
    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)

    #initialize the list of faces, their corresponding locations and the list of predictions from face_mask_detector model
    faces=[]
    locs=[]
    preds=[]

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

        #filter out mask detections by ensuring the confidence is greated than the minimum confidence
        if confidence>0.5:
            #compute the (x,y)coordinates of the bounding box for the object
            box = detections[0,0,i,3:7] * np.array([w,h,w,h])
            (startX,startY,endX,endY) = box.astype("int")

            #ensuring that the bounding boxes fail within the dimensions of the frame
            (startX,startY) = (max(0,startX)), (max(0,startY))
            (endX,endY) = endX, endY

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

```
            #add the face and bounding boxes to their respective lists
            faces.append(face)
            locs.append((startX,startY,endX,endY))
    #make predictions only if face was detected
    if len(faces)>0:
        faces=np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size = 32)

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

#load serialized face_detector model
prototxtPath = r"C:\Users\SHERWIN MATHIAS\Desktop\project\FaceDetection\face_detector\deploy.prototxt"
weightsPath = r"C:\Users\SHERWIN MATHIAS\Desktop\project\FaceDetection\face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

#load our mask detector model
maskNet = load_model("mask_detector.model")

#initialise the video stream
vs=VideoStream(src=0)
vs.start()

while True:
    #grab the frame from the videostream and resize it to have a max width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width = 400)
        #detect faces in the frame and determine if they are wearing a mask or not
    (locs,preds) = detect_and_predict_mask(frame, faceNet, maskNet)
```

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

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

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

        #display the label and bounding box rectangle on the output frame
        cv2.putText(frame, label, (startX,startY-10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, color, 2)
        cv2.rectangle(frame,(startX,startY),(endX,endY), color, 2)

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

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

#doing the cleanup
cv2.destroyAllWindows()
vs.stop()
```

6.2.3 Accuracy of Mask detector Model:

To compare our model with others, we have trained two models using the same dataset from Kaggle.

The following table shows the analysis done.

Model	Dataset	Accuracy (%)
MobilenetV2	Mask Detection Dataset (from Kaggle)	89
Resnet50	Mask Detection Dataset (from Kaggle)	90

Table 6.1: Comparison between different models

If we compare the results shown in Table 6.1, MobilenetV2 has an accuracy of 89% and Resnet50 has an accuracy of 90%. Deep neural networks like Resnet models come with the tradeoff of memory, data speed and time. Even in real world applications such as an autonomous vehicles or robotic visions, the object detection task must be done on the computationally limited platform.

In these cases, MobilenetV2 models are preferred as they are space and time efficient. Hence, we decided to proceed with the MobilenetV2 model.

6.2.4 Face Recognition model:

```
import face_recognition
import cv2
import os
import glob
import numpy as np
```

- **Encoding faces:**

```
class RecognizePerson:
    def __init__(self):
        self.known_face_encodings = []
```

```
        self.known_face_names = []

        # Resizing frame for a faster speed
        self.frame_resizing = 1

    def encoded_images(self, images_path):
        # Load Images
        images_path = glob.glob(os.path.join(images_path, "*.*"))
        # Store image encoding and names
        for img_path in images_path:
            img = cv2.imread(img_path)
            rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

            # Get the filename only from the initial file path.
            basename = os.path.basename(img_path)
            (filename, ext) = os.path.splitext(basename)

            # Get encoding
            img_encoding = face_recognition.face_encodings(rgb_img)[0]

            # Store file name and file encoding
            self.known_face_encodings.append(img_encoding)
            self.known_face_names.append(filename)
```

- **Detecting faces:**

```
    def detect_known_faces(self, frame):
        small_frame = cv2.resize(frame, (0, 0), fx=self.frame_resizing, fy=self.frame_resizing)
        rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
        face_names = []
```

```
        for face_encoding in face_encodings:
            # Seeing if the face is a match for the known face(s)
         matches=face_recognition.compare_faces(self.known_face_encodings,
face_encoding)
            name = "Unknown"


            # Or instead, use the known face with the smallest distance to the new face
         face_distances=face_recognition.face_distance(self.known_face_encodings,
face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = self.known_face_names[best_match_index]
            face_names.append(name)


        # Convert to numpy array to adjust coordinates with frame resizing quickly
        face_locations = np.array(face_locations)
        face_locations = face_locations / self.frame_resizing
        return face_locations.astype(int), face_names
```

- **Displaying name at frontend**

```
def get_frame(self):
        frame = self.video.read()
        frame = imutils.resize(frame, width=500)


        (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)


        for (box, pred) in zip(locs, preds):
            (startX, startY, endX, endY) = box
            (mask, withoutMask) = pred


            if mask > withoutMask:
                label="Mask"
                color = (0,255,0)
```

```
            cv2.putText(frame, label, (startX,startY-10), font, 0.7, color, 2)
            cv2.rectangle(frame,(startX,startY),(endX,endY), color, 2)

        else:
            color = (0,0,255)
            face=frame[startY:endY,startX:endX]
            (_,pids) = rp.detect_known_faces(face)
            for pid in pids:
                label="No Mask: " + pid
                cv2.putText(frame, label, (startX,startY-10), font, 0.7, color, 2)
                cv2.rectangle(frame, (startX,startY),(endX,endY), color, 2)

                # Addding violation data into MaskDetail Table
                if(pid == 'Unknown'):
                    continue
                m = MaskDetail.objects.filter(user_id = pid)
                if(m):
                    lasttime = m.last().date_time
                    lesstime = timezone.now() - timedelta(minutes=5)
                    if lasttime <= lesstime:
                        md = MaskDetail(mask_status = False, user_id = pid, date_time =
timezone.now())
                        md.save()

playsound('C://Users/ujala/OneDrive/Documents/project/AutomatedMaskSystem/detect_
mask/static/detect_mask/beep-09.mp3')
                else:
                    md = MaskDetail(mask_status = False, user_id = pid)
                    md.save()

        ret, jpeg = cv2.imencode('.jpg', frame)
    return jpeg.tobytes()
```

Chapter 7

TESTING

7.1 Test Cases

MobilenetV2 model validation score

```
Epoch 20/20
95/95 [==============================] - 760s 8s/step - loss: 0.0851 - accuracy: 0.9667 - val_loss: 0.3086 - val_accuracy: 0.8913
```

Fig 7.1 Test case of MobilenetV2

	precision	recall	f1-score	support
with_mask	0.99	0.79	0.88	383
without_mask	0.83	0.99	0.90	384
accuracy			0.89	767
macro avg	0.91	0.89	0.89	767
weighted avg	0.91	0.89	0.89	767

Fig 7.2 Accuracy of MobilenetV2

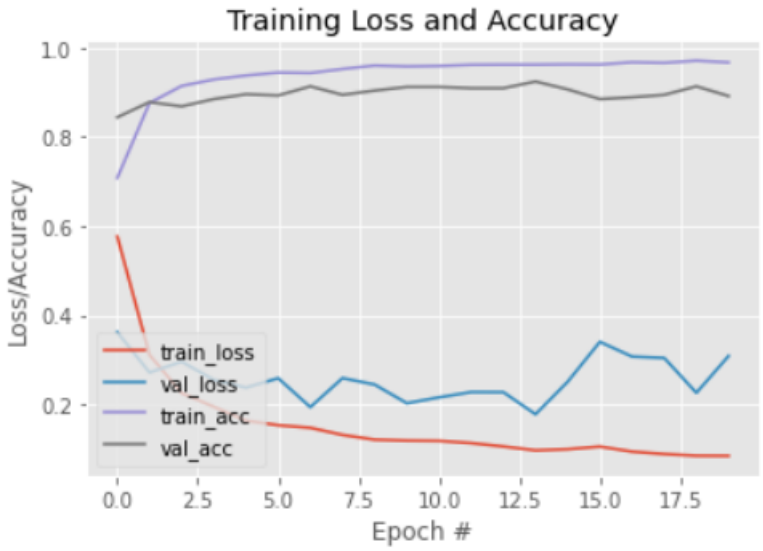

Fig 7.3 Graphical representation of Training loss and Accuracy

There are two main test cases in our project

i. If a person is wearing mask, then the system should detect that person's face with a green box labelled with a tag "mask".

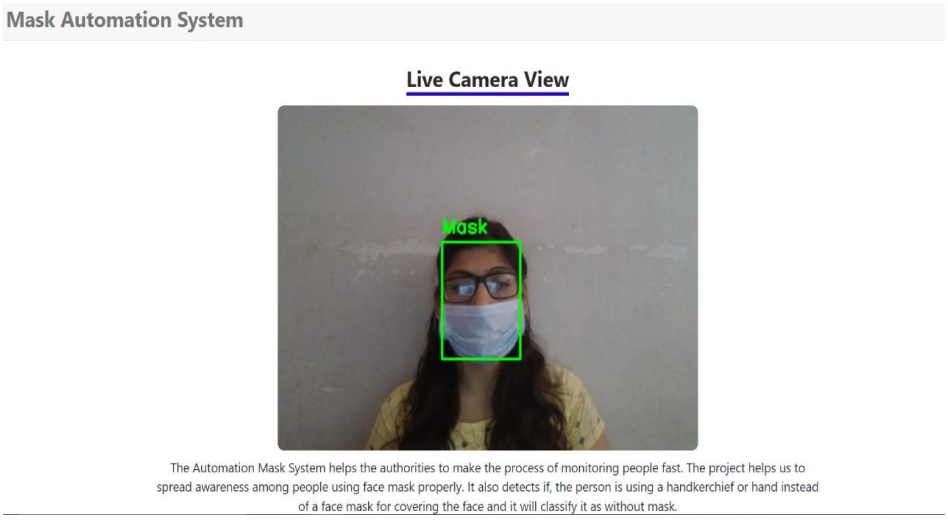

Fig 7.4 Test case (i)

ii. If a person is not wearing a mask, then the system should detect that person's face with a red box labelled with a tag "no mask" and also identify that person and mention his/her PID.

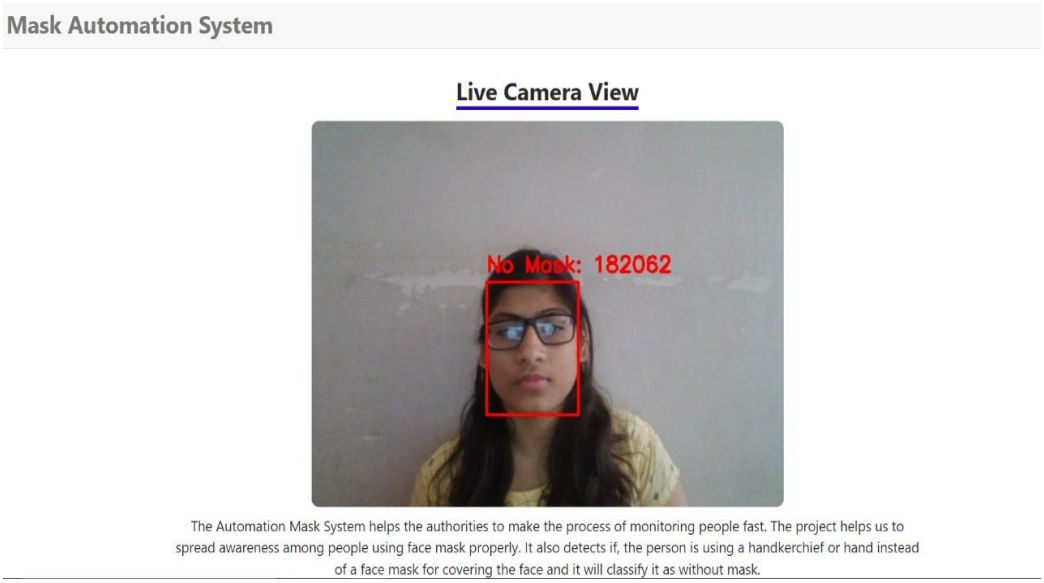

Fig 7.5 Test case (ii)

7.2 Types of Testing used

UI Testing

We have used UI testing as we have developed a website using Django which helps in detecting people who are violating the rule of wearing masks. This includes testing of UI controls like buttons, menus and live video input through webcam, to ensure that the experience flow and features chosen are optimal for the user experience.

Chapter 8

RESULT AND DISCUSSION

Results:

- We trained two models using the same dataset from Kaggle in order to compare and choose the model having better accuracy.
- The following table shows the comparison of the accuracy based on a variety of factors like precision, recall, f1-score and support.

Model	Dataset	Accuracy (%)
MobilenetV2	Mask Detection Dataset (from Kaggle)	89
Resnet50	Mask Detection Dataset (from Kaggle)	90

	precision	recall	f1-score	support
with_mask	0.96	0.84	0.89	383
without_mask	0.85	0.96	0.91	384
accuracy			0.90	767
macro avg	0.91	0.90	0.90	767
weighted avg	0.91	0.90	0.90	767

	precision	recall	f1-score	support
with_mask	0.99	0.79	0.88	383
without_mask	0.83	0.99	0.90	384
accuracy			0.89	767
macro avg	0.91	0.89	0.89	767
weighted avg	0.91	0.89	0.89	767

- From the results shown above, we infer that the MobilenetV2 model has a better accuracy than the Resnet50 model.
- Also, Resnet models come with the tradeoff of memory, data speed and time.
- Even in real world applications such as autonomous vehicles or robotic visions, the object detection task must be done on the computationally limited platform.
- In these cases, MobilenetV2 models are preferred as they are space and time efficient.
- Hence, we decided to proceed with the MobilenetV2 model.
- We also implemented our project live on the college premises.
- Our project correctly predicted without any error, whether the person was wearing the mask or not.
- The project was able to identify the majority of faces of our peers and predicted their PID numbers correctly.

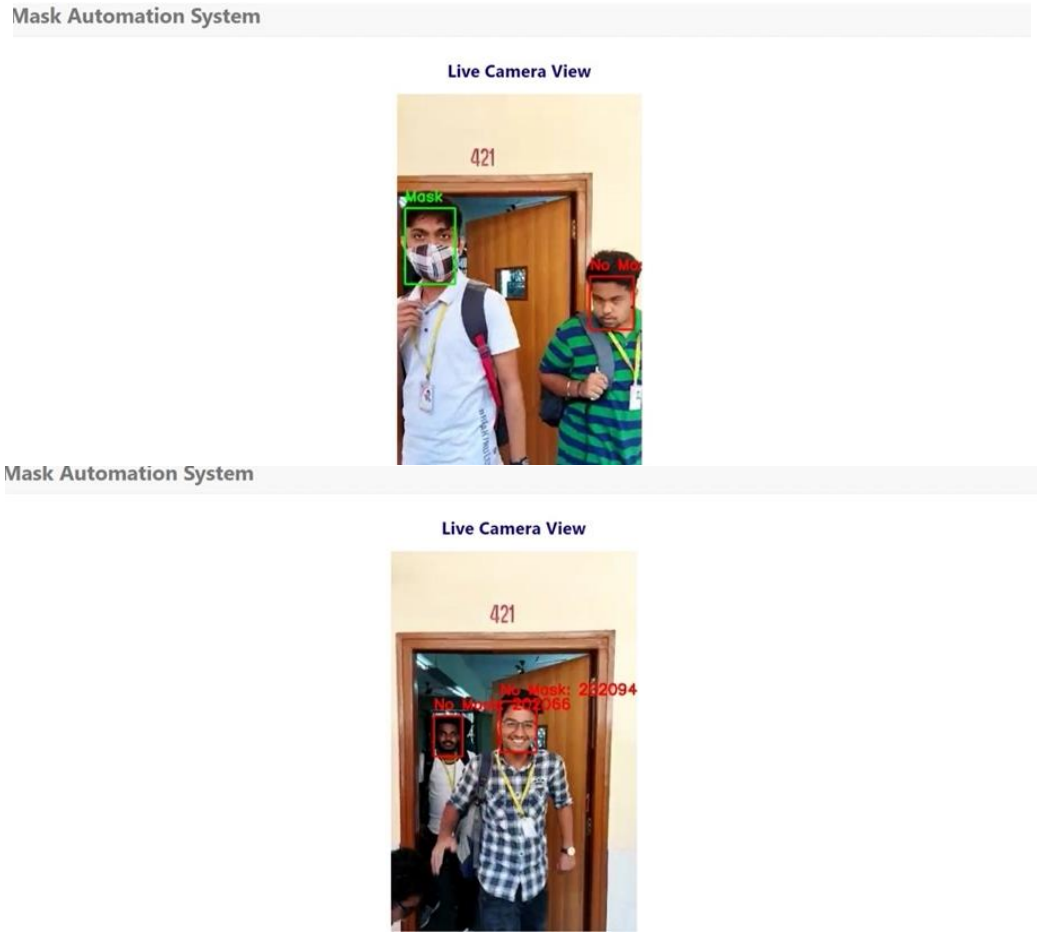

Fig 8.1 Testing model on recorded video

- Based on the results and analysis of our demo, we found out the following range of spatial lengths on which we would position the surveillance cameras to predict the best outcome.
    - Height: 1.6 to 1.8 meters
    - Distance: 1.5 to 2 meters

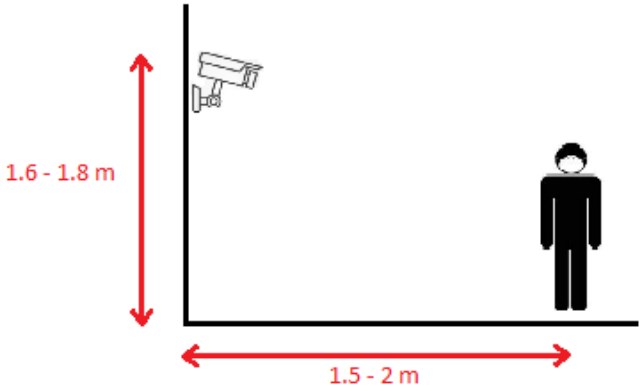

Fig 8.2 Visualizing Height and Distance measures

Discussion:

- Our final model executes and gives effective and desired results.
- It gives correct output for one wearing masks and one who is not.
- The system gives out a beep sound to indicate that there is someone out there without a mask.
- It also recognizes the face of that person and increments the violation count accordingly.
- It also provides graphical and tabular representations along with the relevant criterion-based filtering options (like month, department) for easy and efficient analysis of the happening violations.
- The violators can be alerted by email by just one-click.
- Overall the system works effectively for a real-time input and provides other functions in a very convenient way.

Chapter 9

CONCLUSION AND FUTURE SCOPE

Conclusion:

In conclusion, the report presents the detail information of the proposed system which will help in maintaining a secure environment and to ensure individuals protection by automatically monitoring public places to avoid the spread of the COVID-19 virus. We have given brief about the flow and implementation of our project. We proposed an approach to solve the problem of manual monitoring people not wearing mask. The system uses Computer Vision along with MobileNetv2 and FaceNet deep learning algorithm to solve the problem. We preprocessed the data and trained it using algorithm and then finally tested the model and found the accuracy of around 89%. We have successfully implemented and tested the trained model in live streaming CCTV camera.

Thus, this proposed system will operate in an efficient manner in the current situation when the lockout is eased and helps to track people violating the rule in an automated manner. The solution has the potential to significantly reduce violations by real-time interventions, so the proposed system would improve public safety through saving time and helping to reduce the spread of coronavirus. This solution can be used in places like shopping malls, public society, college campus, etc.

Future Scope:

In near future the threat of virus will end and the rule of wearing masks will be withdrawn but there are still various fields where wearing masks is important, virus or not. Surgeons, people working at pathology labs, research labs, mines and many manufacturing units and industries will still be required to wear masks. So this system can be used at such places to check if people working there are following the norms or not.

REFERENCE

[1] “MobileNetV2: The Next Generation of On-Device Computer Vision Networks”, Mark Sandler and Andrew Howard, Google Research, Tuesday, April 3, 2018.

[2] Samuel Ady Sanjaya, Suryo Adi Rakhmawan, “Face Mask Detection Using MobileNetV2 in The Era of COVID-19 Pandemic”, 20 January 2021.

[3] Shashi Yadav “Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID-19 Safety Guidelines Adherence”, July 2020.

[4] Florian Schroff, Dmitry Kalenichenko and James Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering”, 15 October 2015.

[5] Luka Dulcic, “Face Recognition with FaceNet and MTCNN”, 2018

[6] Ivan William, De Rosal Ignatius Moses Setiadi and Heru agus Santoso, “Face Recognition using FaceNet (Survey, Performance Test, and Comparison)”, October 2019.

[7] Edwin Jose, Greeshma Manikandan, Mithun Haridas and Supriya M H, “Face Recognition based Surveillance System Using FaceNet and MTCNN on Jetson TX2”, March 2019.

Acknowledgements

We express our deep sense of gratitude to our project guide **Ms. K Priya Karunakaran** for encouraging us and guiding us throughout this project. We were able to successfully complete this project with the help of her deep insights into the subject and constant help. We are very much thankful to **Dr. Kavita Sonawane**, HOD of the CMPN Department at St. Francis Institute of Technology and our college principal **Dr. Sincy George** for providing us with the opportunity of undertaking this project which has led to us learning so much in the domain of Deep Neural Network, Machine Learning and Django. Last but not the least we would like to thank all our peers who greatly contributed to the completion of this project with their constant support and help.