FACE MASK DETECTION FOR ATM SECURITY

ABHISHEK ANAND

DEPARTMENT OF INFORMATION TECHNOLOGY

INSTITUTE OF ENGINEERING AND MANAGEMENT, KOLKATA

FACE MASK DETECTION FOR ATM SECURITY

A REPORT SUBMITTED IN PARTIAL FULFILLMENT FOR

THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

FROM

INSTITUTE OF ENGINEERING AND MANAGEMENT KOLKATA

BY

ABHISHEK ANAND

ENROLLMENT NUMBER: 12019002004061

UNIVERSITY REGISTRATION NUMBER: 027786 of 2019-20

Under the guidance of

PROF. KAJARI SUR

DEPARTMENT OF INFORMATION TECHNOLOGY

INSTITUTE OF ENGINEERING AND MANAGEMENT, KOLKATA

Affiliated to

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, KOLKATA, WEST BENGAL

INSTITUTE OF ENGINEERING & MANAGEMENT
Salt Lake Electronics Complex, Kolkata-700 091, WB, INDIA
Phone : (033) 2357-2969/2059/2995
(033) 2357-8189/8908/5389
Fax : 91-33-2357-8302
E-mail : director@iemcal.com
Website : www.iem.edu.in

CERTIFICATE

I hereby recommend that the thesis entitled " **FACE MASK DETECTION FOR ATM SECURITY**" prepared by **Abhishek Anand** (Enrollment No: 12019002004061, University registration No: 027786 of 2019-20) under my supervision be accepted in partial fulfilment of the requirements for the degree of " Bachelor of Technology" in "**Information Technology**" of **Institute of Engineering and Management, Kolkata**.

……..…….

(Prof. Kajari Sur)

Project Guide

DEPARTMENT OF INFORMATION TECHNOLOGY,

INSTITUTE OF ENGINEERING AND MANAGEMENT KOLKATA

COUNTERSIGNED

…….

(Dr. Arun Kumar Bar)

Principal

IEM Kolkata

……..……

(Prof. (Dr.) Moutushi Singh)

Head of the Department

Dept. of IT, IEM Kolkata

Declaration

I declare that this project report titled "FACE MASK DETECTION FOR ATM SECURITY" submitted in partial fulfillment of the degree of B. Tech in Information Technology is a record of original work carried out by me under the supervision of Prof. Kajari Sur, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

……

Abhishek Anand

Enr. No.- 12019002004061

IEM Kolkata

Place: Kolkata

Date:

Acknowledgement

I am very much indebted to my guide Prof. Kajari Sur for giving me the opportunity to pursue my B. Tech. thesis work under her guidance. I would like to express my sincere gratitude for her valuable guidance and insightful suggestions throughout my thesis work. It is her exceptional encouragement and precious knowledge in the field of machine learning that has given this project a meaningful shape. She has always encouraged me to get my work published in various journals of repute.

It is beyond my literal and material means to express my heartfelt thanks and deep gratitude to my mentor Prof. Kajari Sur for her constant encouragement and advice throughout this project.

I was extremely fortune to have Prof. (Dr.) Moutushi Singh as the Head of the department. I thank her for all the support she has provided during the entire duration of the project.

It is my pleasure to acknowledge all my teachers for their extraordinary support and encouragement.

Finally, I thank my parents who have always supported me with deep love and sacrifice throughout my life. I pray for their happiness and good health and I dedicate my work to them in the most sincere way I can think of.

.

Place: Kolkata **Abhishek Anand**

Date:

Abstract

Global pandemic COVID-19 circumstances emerged as an epidemic of the dangerous diseases all over the world. Wearing a face mask will help prevent the spread of infection and prevent the individual from contracting any airborne infectious germs. Using Face Mask Detection System, one can monitor if people are wearing masks or not. Face Mask Detector can be used in various other use-cases, e.g. Face Mask Detection for ATM. If a user is wearing a face mask in an ATM then the security cameras are not able to capture his/her face. If any mishap occurs then authorities will not be able to track them. The main aim to secure the ATM from cyber threat. If there will be any mishap occurs the alarm will be triggered so that security guard can be alert from that. Hence, it is necessary to take off the mask in an ATM.

The algorithm used in the project to achieve the objective is MobileNet V2. An image of a few people wearing a mask and without wearing a mask is used as an input dataset. There are few processes involved in achieving the objective of the project that include pre-processing, data augmentation, training, testing and image segmentation. After the processes, with the help of the Mask R-CNN algorithm, will get a segmented image of the input dataset of people wearing a mask and people not wearing a mask. Then, the model is implemented using a webcam where get result of people wearing a mask and not wearing a mask along with the accuracy in percentage.

Keywords: COVID-19, ATM, MobileNet V2, dataset, image segmentation, R-CNN

Table of Contents

List of Figures

List of Abbreviations

AI	*Artificial Intelligence*
CV	*Computer Vision*
ML	*Machine learning*
OpenCV	*Open-Source Computer vision Library*
Accuracy	*Percentage of correct classified images*
Matrix	*Multi-dimensional*
Optimizer	*A function used to compute gradient*
Relu	*Rectifiled linear unit*
Scalar	*Single Value*
Softmax	*Activation Function*
Test data	*Data that the network*
Training Data	*Data used for training*
CNN	*Convolution neural network*
OS	*Operating System*
NP	*Numpy*
Flatten	*A Utility Layer*
Validation data	*Data Used for validating algorithm*
LB	*Labels*
Split	*Data dividation*
DNN	*Deep Neural Network*
Locs	*Location*
PREDS	*Prediction*
RGB	*Red Green Blue*

Chapter 1

Introduction

1.1 Motivation of the work

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. Affecting the health and lives of masses, COVID-19 has called for strict measures to be followed in order to prevent the spread of disease. From the very basic hygiene standards to the treatments in the hospitals, people are doing all they can for their own and the society's safety, face masks are one of the personal protective equipment. 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 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. Face mask detection means to identify whether a person is wearing a mask or not. The first step to recognize the presence of a mask on the face is to detect the face, which makes the strategy divided into three parts: to detect body, to detect faces and to detect masks on those faces. Face detection is one of the applications of object detection and can be used in many areas like security, biometrics, law enforcement and more. There are many detector systems developed around the world and being implemented. However, all this science needs optimization; a better, more precise detector, because the world cannot afford any more increase in corona cases or any other viral diseases. In this project, This report is about the development of a face mask detector that is able to distinguish between faces with masks and faces with no masks and also there is a website and an Android application for fetching the information related to the percentage of people wearing mask in a particular area. In this report, In this project, It is proposed a detector which employs Single Shot Detector (SSD) for face detection and a neural network to detect presence of a face mask. The implementation of the algorithm is on images, videos and live video streams.

1.2 The idea of ATM security

In Current time as everyone can see that cameras are used everywhere, such as banks, shopping malls, campus area, residential area, institutes etc. at such places it is more important to maintain security. The main objective for movement recognizes the events and aims of one or more people from the progression of observations on the people's action. Supervised learning and understanding of unusual human being deeds is more complex, difficult and wide task. Human act Recognition and extracting feature from the) database has usual a lot of consideration in the computer-vision, machine learning communities. As information technology continues to evolve quickly, Human-Computer Interface (HCI) is playing a role of growing importance. Even after huge development of input devices, still the interaction with computers is cumbersome and an uncomfortable experience. The adaption of computers to natural means of communication like speech and body language has changed the whole experience of interacting with machines. Thriving realization of these modalities into an edge has the possible of easing the HCI bottleneck that has become noticeable with the advances in computing and communication. The HCI can be briefly explained by the point of communication between the human user and the computer.

After developing the face mask detection, In this project it came with the thought that Covid-19 is already in control. So there is no such need of face mask detection and observation of the number of peoples wearing mask or not in any particular area. So, In this project, Authors came with the innovative idea with the model that had trained for face mask detection that It can be use as the security purposes because now a days there is a big issue of ATM theft and robbery in wide area of our country. So, This trained model will detect the face on the entrance gate of the ATM with the help of cctv camera and If the face will be covered by anything our system will trigger the alarm so that the security get an alert for any theft or robbery in the ATM.

1.3 System Design and Work

First of all, The model is deployed using the dataset obtained from Kaggle and implemented this model on images containing one and more faces. This also implemented on videos and live video streams by removing and wearing masks one by one. The dataset which this project has used consists of 9833 total images out of which 4915 are of masked faces and 4918 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. This needs 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 which is not necessary and only optimizes the training dataset accuracy. This needs 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 is used 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, This model 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 it is building and the dataset itself. If this dataset and model are such that a lot of training is required, then it will 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 this model need to take a higher amount of validation dataset. Models with a smaller number of hyperparameters are easy to tune and update, and so this model can take a smaller validation dataset. In our approach, This model has 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, This model has used 20% as a validation data set. Overall, 64% of the dataset is used for training, 16% for validation and 20% for testing.

After Deploying the model it is implemented using the cctv camera then this model able to detect the total number of peoples wearing mask or not in a frame. The model is tested using different live streams, different videos and picture and we got the accuracy around 85%. After that it is implemented for testing in the ATM security alarm purpose and this model got the desired result from that testing. The design of this project is simple. For implementing it, the project require a cctv camera and when there is some people came in front of the camera this trained model will detect the face using the cctv camera and decide whether the person wearing mask or not. If the person will wear a mask or something so that his/her face not visible clearly then our system will trigger an alarm so that security guards will be active in due time.

Chapter 2

Literature Review

The Literature Survey is used to provide a brief overview and explanation of the reference papers. The literature survey conveys the technical details related to the project in a proper and detailed manner.

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 are employed in a wide range of industries. Some examples include image retrieval, security and intelligence, OCR, medical imaging and agricultural monitoring.

2.1 Different Literature surveys

Xinbei Jiang, Tianhan Gao, Zichen Zhu and Yukang Zhao [1] proposed a system in Real-Time Face Mask Detection using YOLOv3. The Properly Wearing Masked Face Detection Dataset (PWMFD) is used in the paper, which has 9205 images samples wearing masks. The relationships among channels are obtained by integrating the attention mechanism into Darknet53 using the SE block, so that the network can focus on the feature. In order to better describe the spatial difference between predicted and ground truth boxes and to improve the stability of bounding box regression, Glou loss is implemented. The extreme foreground-background class imbalance was solved using focal loss. The final results showed that SE-YOLOv3 is better than YOLOv3 and other state-of the-art detectors on PWMFD. While comparing with YOLOv3, the proposed model achieved 8.6% higher mAP and detection speed.

Samuel Ady Sanjaya and Suryo Adi Rakhmawan [2] developed in Face Mask Detection Using MobileNetV2. In the paper, a machine learning algorithm MobilenetV2 is used for face mask identification. The steps for building the model are collecting the data, pre-processing, splitting the data, testing the model, and implementing the model. The proposed model can achieve an accuracy of 96.85%.

Sunil Singh, Umang Ahuja, Munish Kumar, Krishna Kumar and Monika Sachdeva [3] proposed a system in Face Mask Detection using YOLOv3 and faster R-CNN models. This paper, draws bounding boxes on people on the screen in red or green color whether they are wearing a mask or not and keeps the ratio of people wearing masks on a daily basis.

G. Jignesh Chowdary, Narinder Singh Puny, Sanjay Kumar Sonbhadra and Sonali Agarwal [4] developed a system in Face Mask Detection using Transfer Learning of InceptionV3. In the paper, a transfer learning model is proposed to automate the process of identifying the people who are not wearing masks. The model uses deep learning algorithm Inception V3 to detect face masks. The Simulated Masked Face Dataset is used for training and testing. Due to the limited availability, image augmentation technique is used for better training and testing of the model. The model achieved an accuracy of 99.9% during training and 100% during testing.

Shilpa Sethi, Mamtha Kathuria and Trillok Kaushik [5] implemented in Face mask detection using deep learning. In order to achieve high accuracy and low inference time, the proposed technique uses one-stage and two-stage detectors. The ResNet50 and the concept of transfer learning to fuse high-level semantic information are implemented in this paper. During mask detection, in order to improve localization performance, bounding box transformation is used. Three popular baseline models viz. ResNet50, AlexNet and MobileNet are used for experimenting the model. The proposed along with these models can produce high accuracy in less inference time. The proposed technique achieved an accuracy of 98.2% when implemented with ResNet50. In comparison with the recently published Retina facemask detector, the proposed model achieves 11.07% and 6.44% higher precision and recall in mask detection. The proposed model is best suited for video surveillance devices.

Riya Chiragkumar Shah and Rutva Jignesh Shah [6] in proposed a system of Detection of Face Mask using Convolutional Neural Network. The model proposed here is designed and modeled using python libraries namely tensorflow, keras and opencv. The model used is the MobileNetV2 of convolutional neural networks. In this paper, a model is developed using the above mentioned libraries. The model is tested for different conditions with different hyper parameters. First dataset is fed in the model, run the training program, which trains the model on the given dataset. Then the detection program is run, which turns on the video stream, captures the frames continuously from the video stream with an anchor box using object detection process. The output is then passed through MobileNetV2 layers where it is classified into people wearing a mask surrounded by green boxes and people not wearing a box surrounded by red boxes.

Safa Teboulbi, Seifeddine Messaoud, Mohamed Ali Hajjaji and Abdellatif Mtibaa [7] developed a system in Real-Time Implementation of AI Based Face Mask Detection and Social Distancing Measuring System for COVID-19 Prevention. This research paper focuses on implementing a Face Mask and Social Distancing Detection model as an embedded vision system. The pretrained models such as the MobileNet, ResNet Classifier, and VGG are used. This paper consists of two principal blocks. The first block includes the training and the testing models, whereas the second block consists of the whole framework testing. This reult detects people wearing a mask and not wearing a mask and ensures social distancing.

Xueping Su, Meng Gao, Jie Ren, Yunhong Li, Mian Dong and Xi Liu [8] implemented in Face mask detection and classification through deep transfer learning. This paper describes a new algorithm for face mask detection that integrates transfer learning and Efficient-Yolov3, using EfficientNet as the backbone feature extraction network, and GIou as the loss function to decrease the number of network parameters and improve the accuracy of mask detection. This paper divides the mask into two categories of qualified masks and unqualified masks, creates a mask classification data set, and proposes a new mask classification algorithm then combines transfer learning and MobileNet, improves the generalization of the model and solves the problem of small data size and easy overfitting.

Mohamed Almghraby and Abdelrady Okasha Elnady[9] proposed a system in Face Mask Detection in Real-Time using MobileNetv2. The created model for detecting face masks in this paper uses deep learning, tensorflow, keras and opencv. The MobilenetV2 algorithm is used in this paper to detect face masks. The present model dedicates 80 percent of the training dataset to training and 20% to testing, and splits the training dataset into 80% training and 20% validation, resulting in a final model with 65 percent of the dataset for training, 15 percent for validation, and 20% for testing. Stochastic Gradient Descent (SGD) is used as an optimization approach with learning rate of 0.001 and momentum 0.85.

Chhaya Gupta and Nasib Singh Gill[10] proposed a system of Corona mask: A Face Mask Detector for Real-Time Data. Convolutional Neural Network (CNN) algorithm is used in this project to erect faces. In this paper, a dataset has been created which consists of 1238 images which are divided into two classes "mask" and "no mask". Live streaming videos can also be used as input and people wearing a mask and not wearing a mask can be detected. The convolutional neural network is trained on the dataset and it gives 95% of accuracy.

With the rise of Covid-19 cases all over the world, it is mandatory to follow the guidelines provided by WHO and local health authorities. The above-cited paper was useful in visualization and understanding the flow of the entire project and also designing its architecture, input and its format to be given for each module along with expected output were identified. The above-cited papers are focused at one point in detecting people not wearing a mask. With reference to all these papers, our project is also focused on detecting people not wearing face masks using MobileNetV2.

Chapter 3

Technologies Used

3.1 Tensorflow

TensorFlow is an open-source machine learning software that focuses on deep neural networks from start to finish. TensorFlow is a collection of libraries, tools, and community resources that are diverse and comprehensive. It enables programmers to construct and deploy cutting-edge machine learningbased applications. The Google Brain team first designed the TensorFlow python deep-learning library for internal usage. The open-source platform's application in R&D and manufacturing systems has increased since then. There are a few key principles in TensorFlow. Tensors are TensorFlow's fundamental building blocks. In the TensorFlow python deep-learning framework, a tensor is an array that represents many forms of data. Unlike a one-dimensional vector or array or a two-dimensional matrix, a tensor can have n dimensions. The shape represents dimensionality. A one-dimensional tensor is a vector; a two-dimensional tensor is matrix; and a zero-dimensional tensor is a scalar.

3.2 Keras

Google developed Keras, a high-level deep learning API for building neural networks. It is written in python and helps with neural network development. It is modular, quick, and simple to use. It was created by Google developer Francois Chollet. Low-level computation is not handled by Keras. Instead, it makes use of a library known as the "Backend". Keras provides the ability to swap between different back ends. TensorFlow, Theano, PlaidML, MXNet, and CNTK (Microsoft Cognitive Toolkit) are among the frameworks support by Keras. TensorFlow is the only one of these five frameworks that has accepted Keras as its official high-level API. Keras is a deep learning framework that is built on top of TensorFlow and has built-in modules for all neural network computations. Simultaneously, the TensorFlow core API may be used to build custom computations with tensors, computation graphs, sessions, and so on. It gives users complete control and flexibility over their applications, as well as the ability to quickly implement ideas.

3.3 OpenCV Python

OpenCV is a large open-source library for image processing, machine learning, and computer vision. python, c++, java, and other programming languages are among the languages supported by OpenCV. It can recognize objects, faces, and even human writing by analysing efficient library for numerical operations. One of OpenCV's goals is to provide a simple-to-use computer vision infrastructure that allows individuals to quickly create rather complex vision applications. Over 500 functions in the OpenCV library cover a wide range of vision topics, including factory product inspection, medical imaging, security, user interface, camera calibration, stereo vision, and robotics. Because computer vision and machine learning are frequently used together, OpenCV also includes a comprehensive machine learning library.

3.4 NumPy and SciPy

NumPy is the most important python package for scientific computing. It is a library that includes a multidimensional array object, derived objects (such as masked arrays and matrices), and a variety of routines for performing fast array operations, such as mathematical, logical, shape manipulation, sorting, selecting, basic linear algebra, basic statistical operations, random simulation, and more. Many other popular python packages, like as pandas and matplotlib, are compatible with NumPy. (NumPy 2022.). On the other hand, SciPy is another important python library for scientific and technical computing that is free and open source. It is a set of mathematical algorithms and utility functions based on the python numpy extension. It gives the user a lot of power by providing high-level commands and classes for manipulating and displaying data in an interactive python session. SciPy builds on NumPy, developers do not need to import NumPy if SciPy has already been imported.

3.5 Imutils and Matplotlib

Imutils are a set of convenience functions for OpenCV and python 2.7 and python 3 that make basic image processing functions including translation, rotation, scaling, skeletonization, and presenting matplotlib pictures easier. Matplotlib is an important python visualization library for 2D array plots. Matplotlib is a multi-platform data visualization package based on numpy arrays and intended to operate with scipy stack. One of the most important advantages of visualization is that it provides visual access to large volumes of data in simple images. Matplotlib has a variety of plots such as line, bar, scatter, histogram, and so on. It is a cross-platform package that provides a variety of tools for creating in python from data stored in lists or arrays, and it is one of the most capable libraries available in python.

Chapter 4

Dataset Preparation

4.1 Implementation of the problem

A Face Mask Detection model is the subject of this project. A model has been developed to determine whether people are wearing masks. The primary camera on the computer was used in this system, and the video was sent as input to the model that was deployed. This system is built using OpenCV libraries, as well as images that are supplied into the system during the learning process. The system was identified using the MobileNet algorithm. FIGURE 11 shows the overall system design.

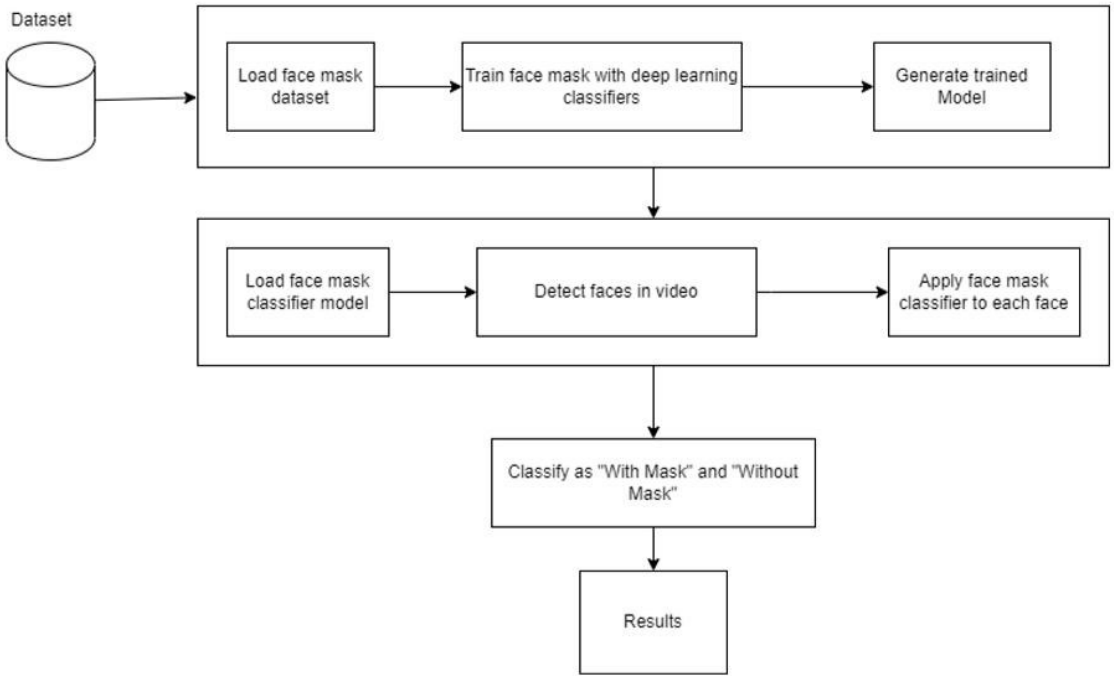

FIGURE 1. System architecture

4.2 Preparing Dataset

The dataset which this project has used consists of 9833 total images out of which 4915 are of masked faces and 4918 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. This needs 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 which is not necessary and only optimizes the training dataset accuracy. This needs 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 is used 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, This model 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 it is building and the dataset itself. If this dataset and model are such that a lot of training is required, then it will 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 this model need to take a higher amount of validation dataset. Models with a smaller number of hyperparameters are easy to tune and update, and so this model can take a smaller validation dataset. In our approach, This model has 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, This model has used 20% as a validation data set. Overall, 64% of the dataset is used for training, 16% for validation and 20% for testing.

Different types of people's facial images are displayed above the image figure. These images were taken from Kaggle's website. The dataset also contained images taken by the author using his computer's default camera and added to the dataset. These images were utilized to achieve the purpose of model training.

The datasets also contain images with masks. These have also been collected from the Kaggle website. For data training, pictures were taken from different viewpoints and by different people. The author included a couple of his own photos in the image datasets, which he took with the computer's default camera.

4.3 Architecture

The working of the Single Shot Detector algorithm relies on an input image with a specified bounding box against the objects. The methodology of predicting an object in an image depends upon a very renowned convolution fashion. For each pixel of a given image, a set of default bounding boxes

(usually 4) with different sizes and aspect ratios are evaluated. Moreover, for all the pixels, a confidence score for all possible objects is calculated with an additional label of 'No Object'. This calculation is repeated for many different feature maps. In order to extract feature maps, This model usually use the predefined trained techniques which are used for high-quality classification problems. This part of the model is called a base model. At the training time, the bounding boxes evaluated are compared with the ground truth boxes and in the backpropagation, the trainable parameters are altered as per requirement. This model got truncate just before the classification layer and add feature layers which keep on decreasing in size.

At each feature space, In this project It is used a kernel to produce outcomes which depict corresponding scores for each pixel whether there exists any object or not and the corresponding dimensions of the resulting bounding box. It is a very dense network having 16 layers of convolution which are useful in extracting features to classify and detect objects. The reason for the selection is because the architecture consists of stacks of convolutions with 3x3 kernel size which thoroughly extracts numerous feature information along with max-pooling and ReLU to pass the information flow in the model and add non-linearity respectively from the given image. For additional nonlinearity, it uses 1x1 convolution blocks which do not change the spatial dimension of the input. Due to the small size of filters striding over the image, there are many weight parameters which end up giving improved performance. The block diagram shows the working functionality of SSDSome additional feature layers are added at the end of the base model to take care of offsets and confidence scores of different bounding boxes. At the end part of the figure, It can be seen that the layers being flattened to make predictions for different bounding boxes. In the end, non-maximum suppression is used whose purpose is to remove duplicatejk.b or quite similar bounding boxes around the same objects. There may be situations where the neighbouring pixel also predicts a bounding box for an object with a bit less confidence which is finally rejected.

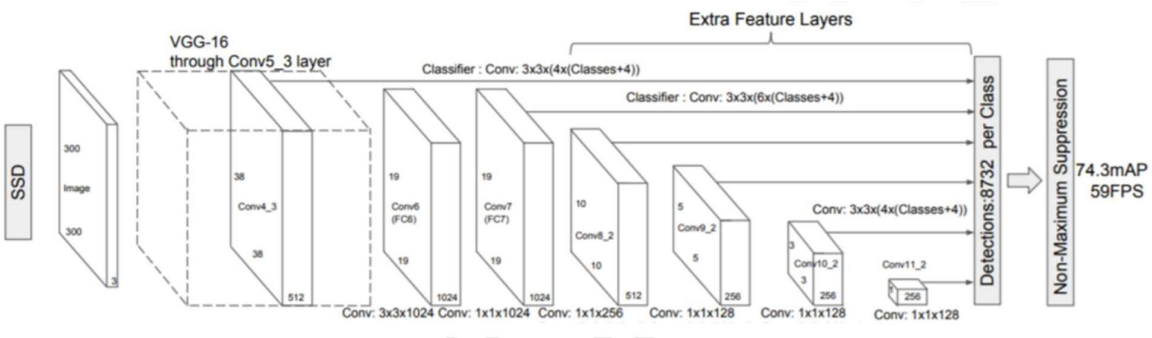

FIGURE 2 : CNN Layers

The problem can be solved in two parts: first detecting the presence of several faces in a given image or stream of video and then in the second part, detecting the presence or absence of a face mask on the face. In order to detect the face, this Project has used the OpenCV library. The latest OpenCV includes a Deep Neural Network (DNN) module, which comes with a pre-trained face detection convolutional neural network (CNN). The new model enhances the face detection performance compared to the traditional models. Whenever a new test image is given, it is first converted into BLOBS (Binary Large Object refers to a group of connected pixels in a binary image) and then sent into the pre-trained model which outputs the number of detected faces. Every face detected comes out with a level of confidence which is then compared with a threshold value to filter out the irrelevant detections. After got the faces, It needs to evaluate the bounding box around it and send it to the second part of the model to check if the face has a mask or not. The second part of the model is trained by us using a dataset consisting of images with masks and without masks. In this project, It is used Keras along with Tensorflow to train the model. The first part of the training includes storing all labels of the images in a Numpy array and the corresponding images are also reshaped (224, 244, 3) for the base model. Image augmentation is a very useful technique because it increases our dataset with images with a whole new perspective. Before inputting, It performed the following image augmentations randomly: rotations up to 20 degrees, zooming in and out up to 15%, width or height shift up to 20%, up to 15 degrees shear angle in the counterclockwise direction, flip inputs horizontally and points outside the boundaries of the inputs are filled from the nearest available pixel of the input. For image classification, it is now a common practice to use transfer learning which means using a model which has been pre-trained on millions of labels before and it has been tested that this method results in a significant increase in accuracy. Obviously, the assumption here is that both problems have sufficient similarities. It uses a well-structured and deep neural network that has been trained on a large amount of data set. Due to the somewhat same nature of the problem, this model can use the same weights which have the capability to extract features and later in the deep layers, convert those features to objects. The base model that has used here is MobileNetV2 with the given 'ImageNet' weights. ImageNet is an image database that has been trained on hundreds of thousands of images hence it helps a lot in Image classification. For the base model, It truncate the head and use a series of our self-defined layers. It is used an average pooling layer, a flattened layer, a dense layer with an output shape (None, 128), and activation ReLU, a 50% dropout layer for optimization, finally another dense layer 5 with an output shape.

Chapter 5

Design

5.1 UML Diagrams:

A UML diagram is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains graphical elements (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts. The kind of the diagram is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines. UML specification does not preclude mixing of different kinds of diagrams, e.g. to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced. At the same time, some UML Tools do restrict set of available graphical elements which could be used when working on specific type of diagram. UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams. Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts. Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

5.1.1 Use Case Diagram

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

(i) Scenarios in which your system or application interacts with people, organizations, or external systems.

(ii) Goals that your system or application helps those entities (known as actors) achieve.

(iii) The scope of your system

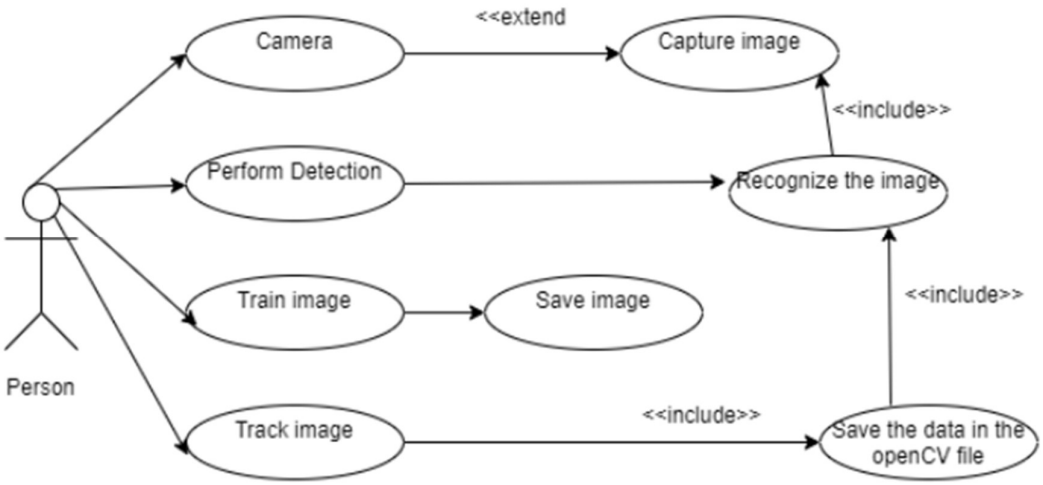

FIGURE 3: Use case diagram

5.1.2 Sequence Diagram:

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios. Sequence diagrams can be useful references for businesses and other organizations.

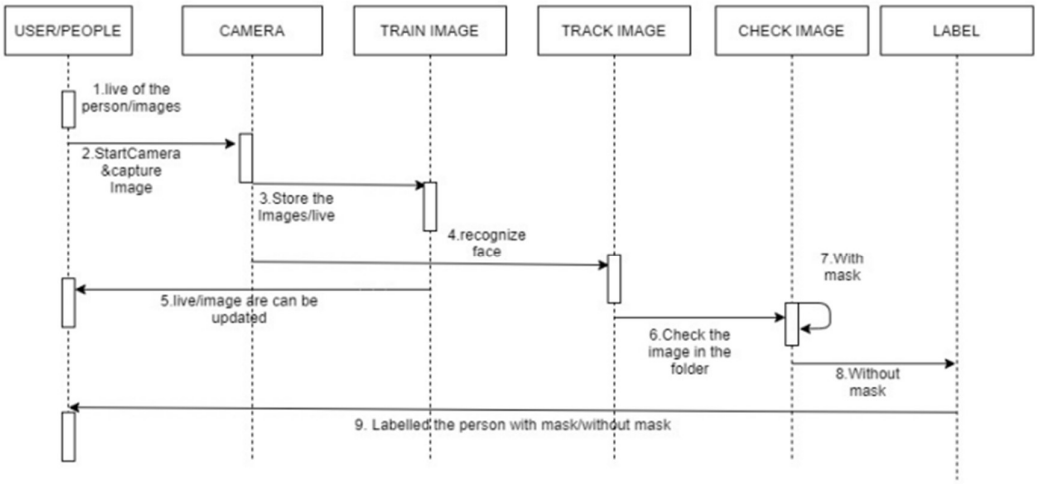

Figure 4: Sequence Diagram

5.1.3 Activity Diagram

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

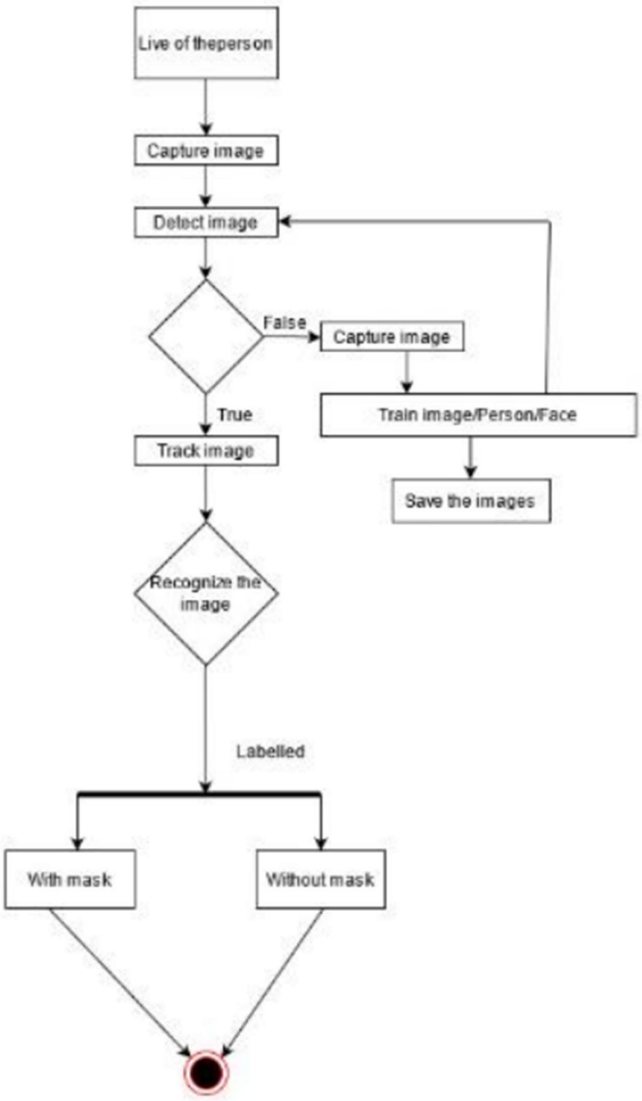

Figure 5: Activity Diagram

5.1.4 Block Diagram

A block diagram is a graphical representation of a system – it provides a functional view of a system. Block diagrams give us a better understanding of a system's functions and help create interconnections within it. They are used to describe hardware and software systems as well as to represent processes.

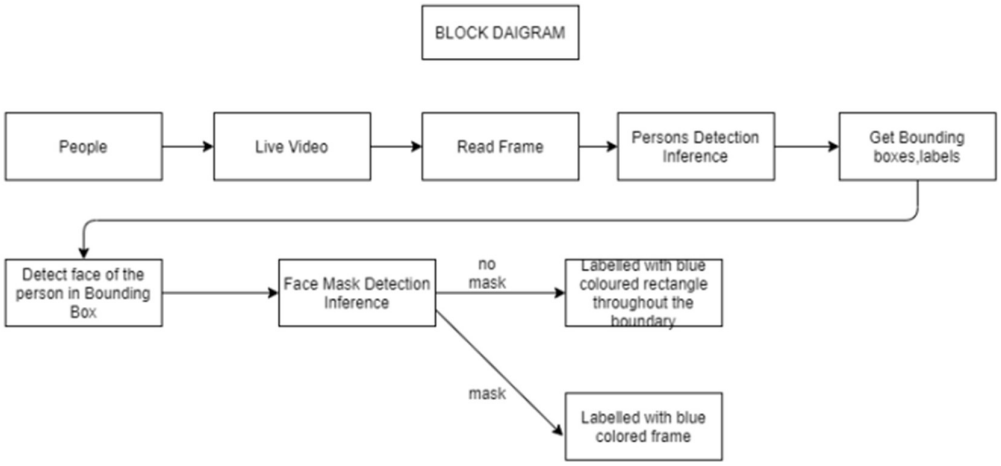

Figure 6: Block Diagram

5.1.5 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

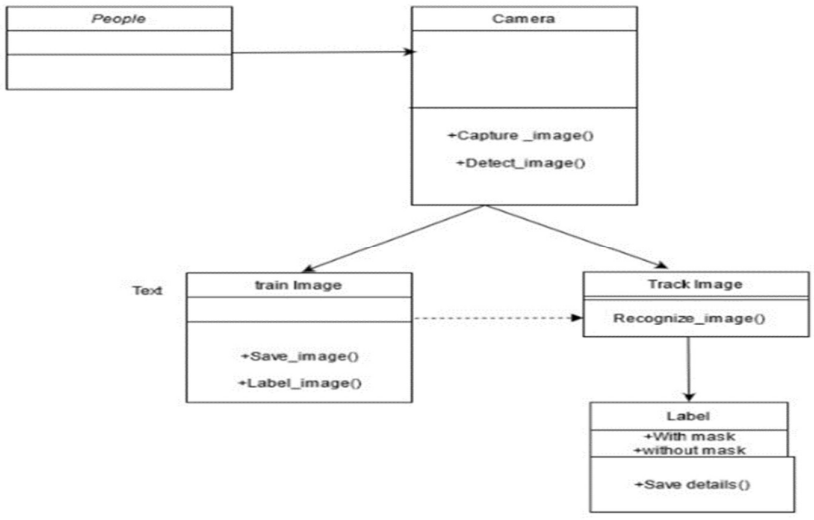

Figure 7: Class Diagram

5.1.6 Data flow Diagram

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored. A graphical tool for defining and analyzing minute data with an active or automated system, including process, data stores, and system delays.

Data Flow Data is a key and basic tool for the architecture of all other objects. Bubble-bubble or data flow graph is another name for DFD.

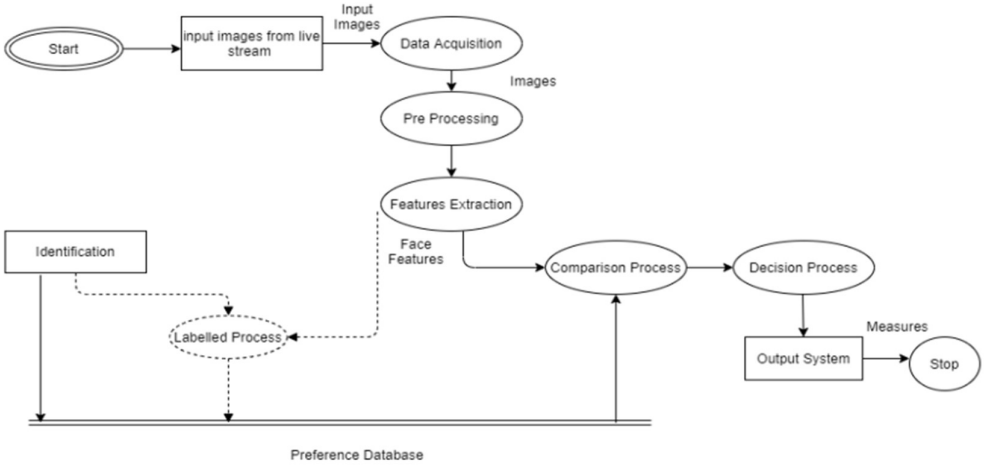

Figure 8: Data flow diagram

5.1.7 Flow Chart Diagram

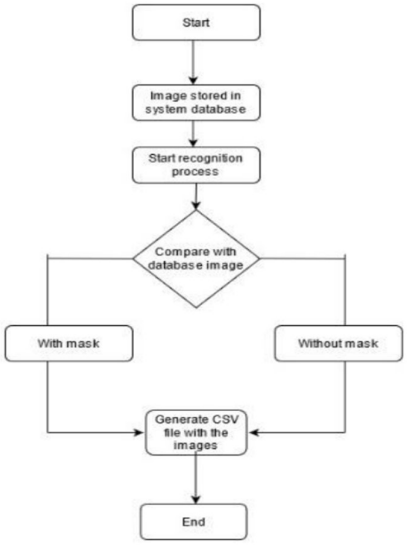

Figure 9: Flow Chart Diagram

Chapter 6

Image Processing Technique

6.1 CNN architecture

CNN architecture occur in a variety of shapes and sizes, but they all have convolutional and pooling (or subsampling) layers that are organised into modules. These modules are followed by one or more fully connected layers, like a normal feedforward neural network. To create a deep model, modules are frequently stacked on top of one another. The network receives an image directly, which is then processed through various rounds of convolution and pooling. The results of these processes are then fed into one or more fully connected layers. The output layer receives the inputs from the layers above it, executes the calculations using its neurons, and then computes the output. Even though this is the most used base architecture in the literature, various architecture modifications have been proposed in recent years with the goal of boosting image classification accuracy or lowering computation costs.

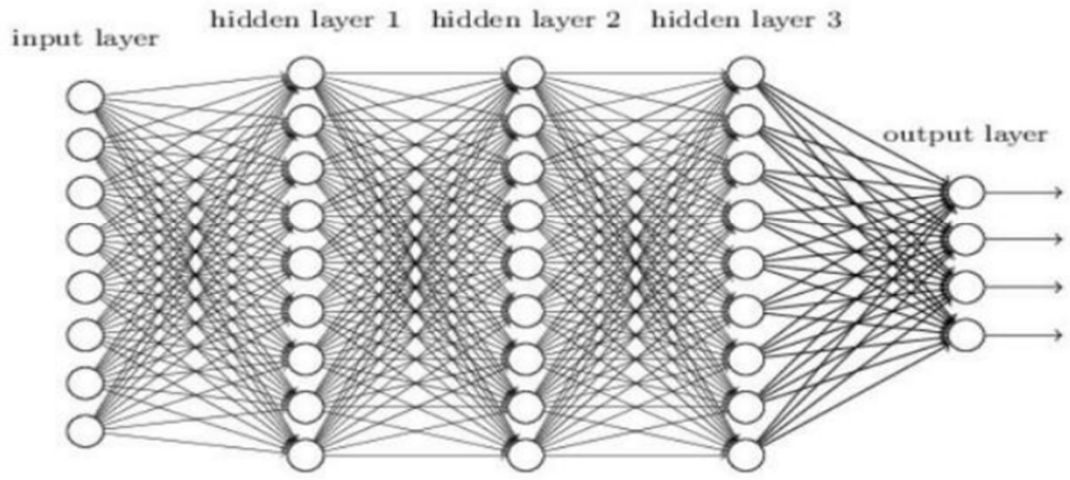

FIGURE 10: CNN Architecture

6.2 Workflow of CNN

The higher performance of convolutional neural networks with picture, speech, or audio signals inputs distinguishes them from other neural networks. Convolutional layer, pooling layer, and fully connected layer are the main three basic types of layers available. A convolutional network's first layer is the

convolutional layer. While further convolutional layers or pooling layers can be added after convolutional layers, the fully connected layer is the last layer. The CNN becomes more complicated with each layer, detecting larger areas of the image. Earlier layers concentrate on basic elements like colors and borders. As the visual data travels through the cnn layers, it begins to distinguish larger elements or features of the item, eventually identifying the target object.

6.2.1 Convolutional layer

The convolutional layer is the most important component of a CNN because it is where most of the computation takes place. It requires input data, a filter, and a feature map, among other things. For example, a color image made up of a 3D matrix of pixels is sent to the input layer. This means the input will have three dimensions: height, width, and depth, which match to the rgb color space of a picture. A feature detector, also known as a kernel or a filter, will traverse across the image's receptive fields, checking for the presence of the feature. Convolution is the term for this procedure. The feature detector is a two-dimensional (2-D) weighted array that represents a portion of the image. The filter size, which can vary in size, is usually a 3x3 matrix, which also affects the size of the receptive field. After that, the filter is applied to a portion of the image, and a dot product between the input pixels and the filter then shifts by a stride, and the procedure is repeated until the kernel has swept across the entire image. A feature map, activation map, or convolved feature is the ultimate output of a series of dot products from the input and the filter.

6.2.2 Pooling layer

Downsampling, often known as pooling layers, is a dimensionality reduction technique that reduces the number of factors in an input. The pooling process, like the convolutional layer, sweeps a filter across the entire input, but this filter has no weights. Instead, the kernel populates the output array by applying an aggregation function to the values in the receptive field. Pooling is divided into two categories: Max pooling and Average pooling. Max Pooling is a convolutional procedure in which the filter or Kernel takes the pixel with the highest value from the input and sends it to the output array. This approach is used more commonly as compared to average pooling. Average pooling, on the other hand, is a pooling technique that employs the average value for patches of a feature map to build a downsampled feature map.

6.2.3 Fully connected layer

The fully connected layer's name is self-explanatory. In a neural network, fully connected layers are those where all the inputs from one layer are connected to each activation unit of the next layer. In partially connected layers, the pixel values of the input image are not directly connected to the output layer, as previously stated. Each node in the output layer is connected directly to a node in the previous layer in the fully connected layer. This layer performs categorization based on the features extracted by the preceding layers and the filters applied to them. While convolutional and pooling layers typically utilize relu functions to classify inputs, fully connected layers typically use a softmax activation function to produce a probability ranging from 0 to 1.

6.3 Architecture used for our project

In this project, It is used mobileNet Architecture for our project because it is very fast in smooth livestreaming.

6.3.1 MobileNets

MobileNets are a CNN design that is both efficient and portable in real world applications. CNNs that can fit on a mobile device to classify photos or detect objects with low latency are known as MobileNets. They are often very tiny CNN architectures, making them simple to execute in real-time on embedded devices such as smartphones and drones. The design is very adaptable, having been tested on CNNs with 100-300 layers and outperforming other architectures such as VGGNet. CNNs embedded into Android phones to run Google's Mobile Vision API, which can automatically recognize labels of popular objects in photos, are in real-world examples of MobileNets CNN architecture. Figure 9 shows MobileNet architecture.

6.3.1.1 MobileNetV2

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

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

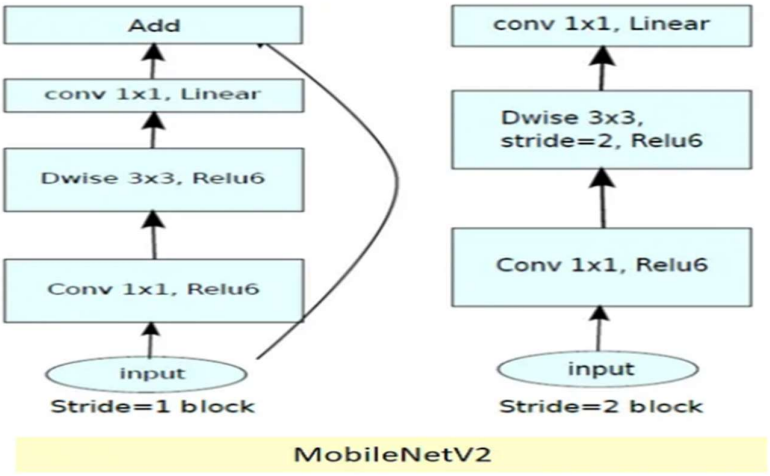

FIGURE 11: MobileNetV2 Convolutional Block

6.3.2 Overall Architecture of MobileNetV2

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

Figure 12: Architecture of MobileNetV2

where t: expansion factor, c: number of output channels, n: repeating number, s: stride. 3×3 kernels are used for spatial convolution.

In typical, the primary network (width multiplier 1, 224×224), has a computational cost of 300 million multiply-adds and uses 3.4 million parameters. (Width multiplier is introduced in MobileNetV1.)

The performance trade offs are further explored, for input resolutions from 96 to 224, and width multipliers of 0.35 to 1.4.

The network computational cost up to 585M MAdds, while the model size vary between 1.7M and 6.9M parameters.

To train the network, 16 GPU is used with batch size of 96.

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

Figure 13: MoblileNet Size distribution

Chapter 7

Methodology

7.1 The Proposed Methodology

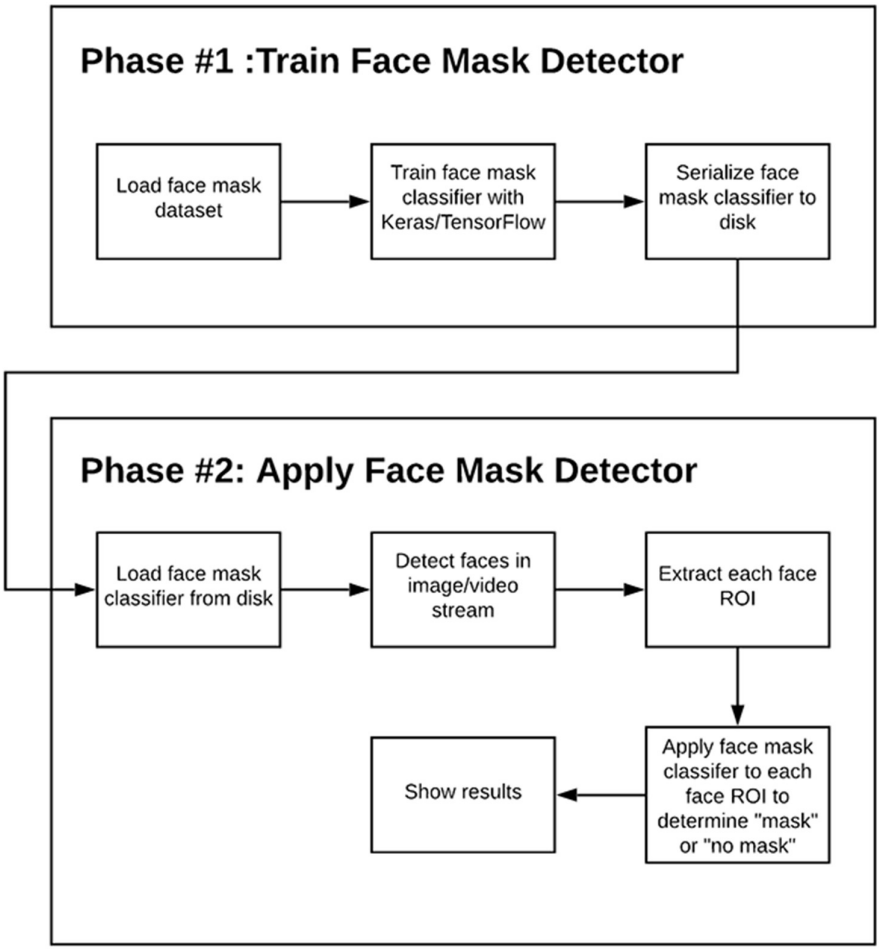

Figure 14: FlowChart of the methodology used

First of all,In this project, It collected the dataset from kaggle and using that dataset it is trained the face mask detection model with the help of tensorflow and keras. When this trained model got an accuracy around 87% This implemented the model with the help of video streaming module named ffmpeg(fast forward moving picture expert group. After proper streaming this model got the results in terms of a person wearing mask or not. Using this result it is added another algorithm which gives alert notification if a person wearing mask or something so that In an ATM machine the suspicious activities get

trapped and alert system can alert the available guards present in the ATM for security purpose.

7.2 Dataset and Training

The dataset which this project has used consists of 9833 total images out of which 4915 are of masked faces and 4918 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. This needs 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 which is not necessary and only optimizes the training dataset accuracy. This needs 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 is used 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, This model 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 it is building and the dataset itself. If this dataset and model are such that a lot of training is required, then it will 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 this model need to take a higher amount of validation dataset. Models with a smaller number of hyperparameters are easy to tune and update, and so this model can take a smaller validation dataset. In our approach, This model has 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, This model has used 20% as a validation data set. Overall, 64% of the dataset is used for training, 16% for validation and 20% for testing.

Different types of people's facial images are displayed above the image figure. These images were taken from Kaggle's website. The dataset also contained images taken by the author using his computer's default camera and added to the dataset. These images were utilized to achieve the purpose of model training.

The datasets also contain images with masks. These have also been collected from the Kaggle website. For data training, pictures were taken from different viewpoints and by different people. The author included a couple of his own photos in the image datasets, which he took with the computer's default camera.

In the proposed work, face mask detection is achieved through deep neural networks because of their better performance than other classification algorithms. But training a deep neural network is expensive because it is a time-consuming task and requires high computational power. To train the network faster and cost-effective, deep-learning-based transfer learning is applied here. Transfer learning allows to transferring of the trained knowledge of the neural network in terms of parametric weights to the new model. It boosts the performance of the new model even when it is trained on a small dataset. There are several pre-trained models like AlexNet, MobileNet, ResNet50 etc. that had been trained with 14 million images from the ImageNet dataset. In the proposed model, ResNet50 is chosen as a pre-trained model for facemask classification. The last layer of ResNet50 is fine-tuned by adding five new layers. The newly added layers include an average pooling layer of pool size equal to 5 × 5, a flattering layer, a dense ReLU layer of 128 neurons, a dropout of 0.5 and a decisive layer with softmax activation function for binary classification as shown in the figure.

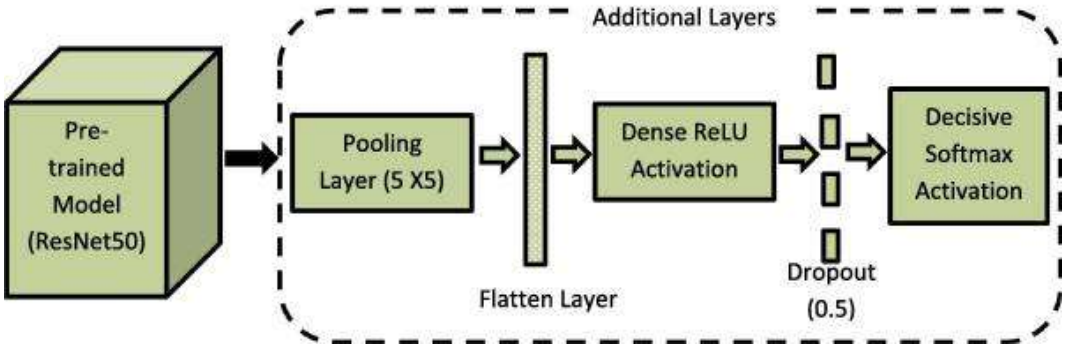

Figure 15: Traing using ResNet

7.3 Testing Dataset

It is observed that dataset, this project consider primarily, contains two major classes that is, mask and non-mask class but the mask class further, contains an inherent variety of occlusions other than surgical/cloth facemask, for example, occlusion of ROI by other objects like a person, hand, hair or some food item as shown in Figure. These occlusions are found to impact the performance of face

and mask detection. Thus, obtaining an optimal trade-off between accuracy and computational time for face detection is not a trivial task. So, an image complexity predictor is being proposed here. Its purpose is to split data into soft versus hard images at the initial level followed by a mask and non-mask classification at a later level through a facemask classifier. The important question that it needs to answer is how to determine whether an image is soft or hard. The answer to this question is given by the "Semi-supervised object classification strategy" proposed by Lonescu et al. The Semi-supervised object classification strategy is suitable for our task because it predicts objects without localizing them. For implementing this strategy, It took three sets of image samples: the first set (L) contains labelled (hard/soft) training images, the second set (U) contains unlabelled training images and the third set (T) contains unlabelled test images. This model further, applied the curriculum learning approach as suggested, which operates iteratively, by training the hard/soft predictor at each iteration on enlarged training set L. The training set L is enlarged by randomly moving k samples from U to L. This model stopped the learning process when L grew three times its original size. Initially, 500 labelled samples are populated in L. The initial labelling of samples in L is done using the three most correlated image properties that make the image complex. These properties are namely object density (including full, truncated and occluded faces), mean area covered by object normalized by image size and image resolution. The object density is evaluated by human annotators. It took 50 trusted annotators and each annotator is shown 10 images.

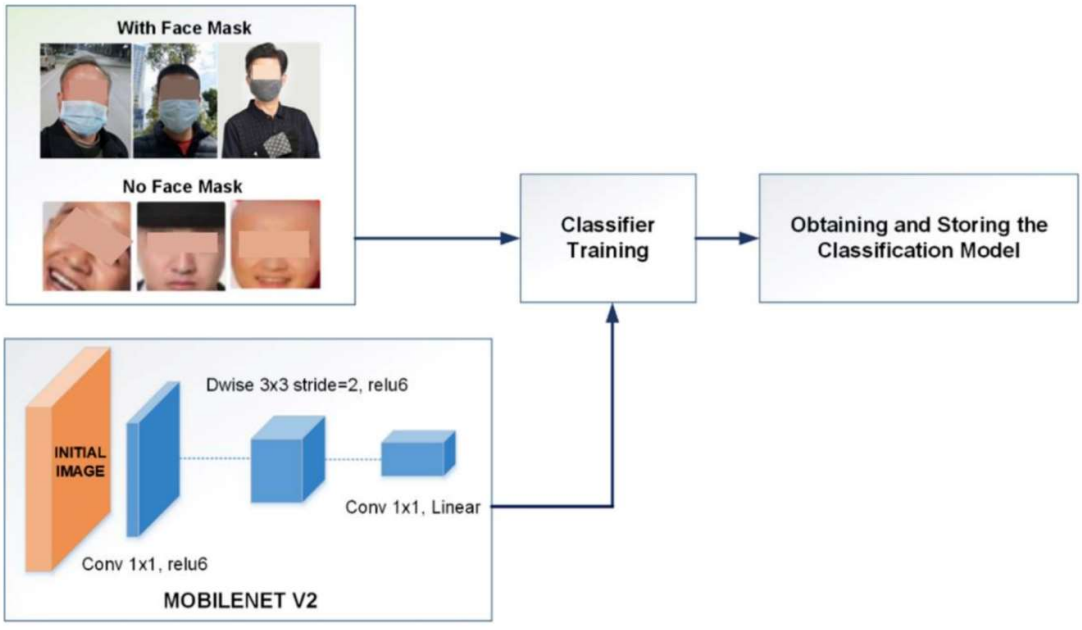

Figure 16: Various Annotators

The project supposed to asked two questions to each annotator. The questions were: "Is there a human being in the image?" and "How many human faces including full, truncated and occluded faces are present in the given image?". It is ensured that the annotation task is not trivial by presenting images in a random order such that if the answer to one image is positive then for another image, it may be negative. In this project, It is recorded the response time of each annotator to answer the questions. It is removed all response time longer than 30 s to avoid bias. Further, each annotator response time is normalized by subtracting it from meantime and dividing by standard deviation. In this project, It is computed the geometric mean of all response times per image and saved the values as object density. This further, observed that image complexity is positively related to object density whereas negatively related to object size and image resolution. Based on these image properties, a ground truth visibility difficulty score is assigned to each image.

7.4 Training loss and accuracy

When the model is deployed for the first time using 4000 number of images then it got the accuracy about 72% but after increasing the number of images of with mask and without mask to 10000 then it got an accuracy of 87%. The final accuracy the model got is shown in the given figure. The figure shows the training loss and accuracy during the model training.

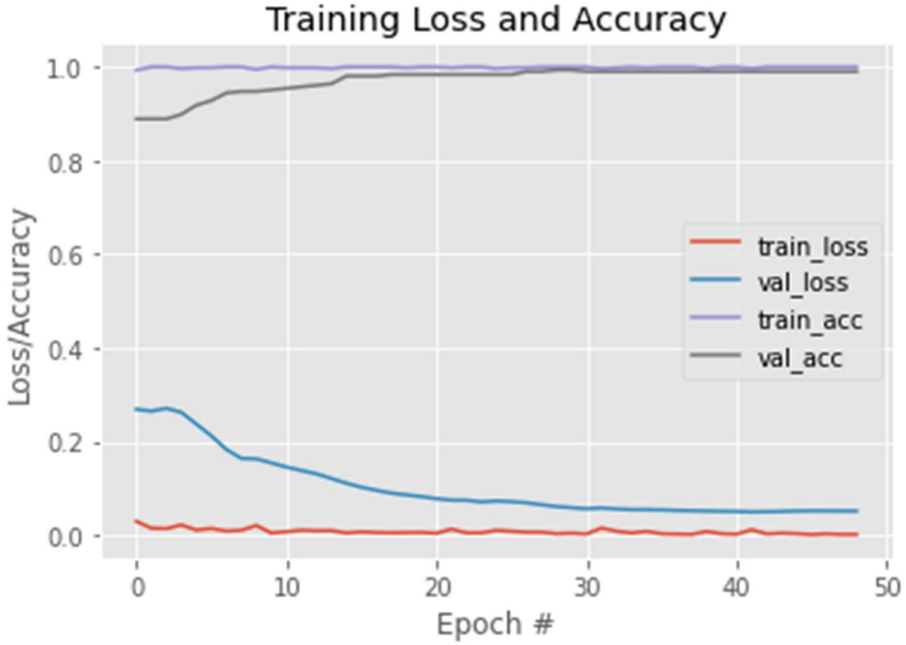

Figure 17: Epoch graph of training loss and accuracy

7.5 Testing the model

In this project, It is tried using three different base models for detecting 'mask' or 'no mask'. The exercise was done to find the best fit model in our scenario. The evaluation process consists of first looking at the classification report which gives us insight towards precision, recall and F1 score.

First of all, The model is tested whether It can properly detect the number of available faces present in a frame and simultaneously whether It can classify the number of peoples wearing mask or not.

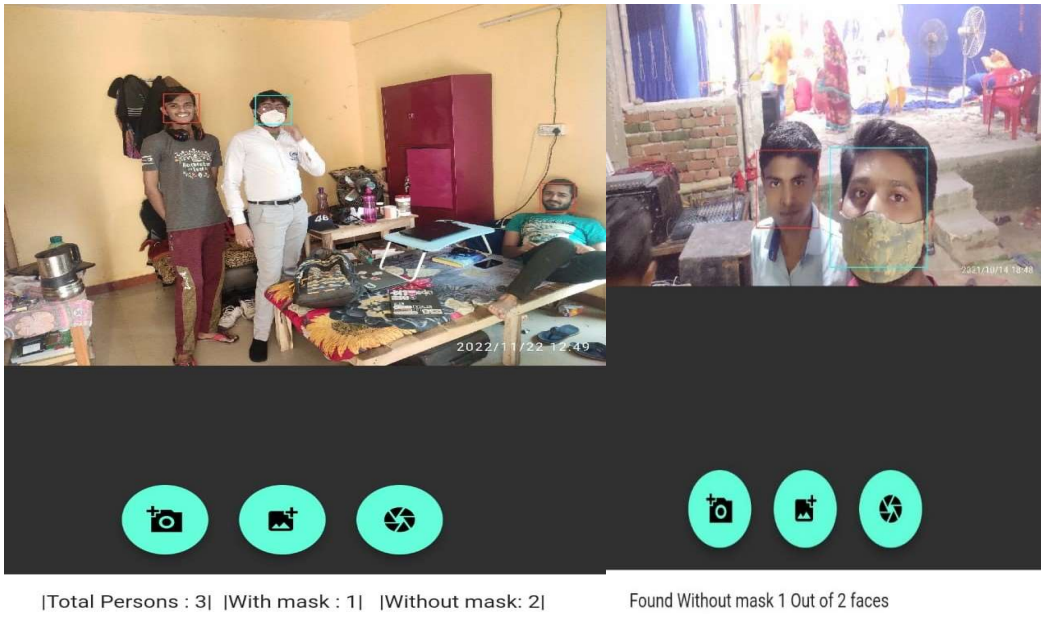

Figure 18: Model Testing images for face mask detection

After the proper testing of classification of with mask and without mask persons in a frame. The algorithm is implemented to give alert while a person wearing a mask to prevent any suspicious activity in the ATM. The result this model got from this testing is shown below.

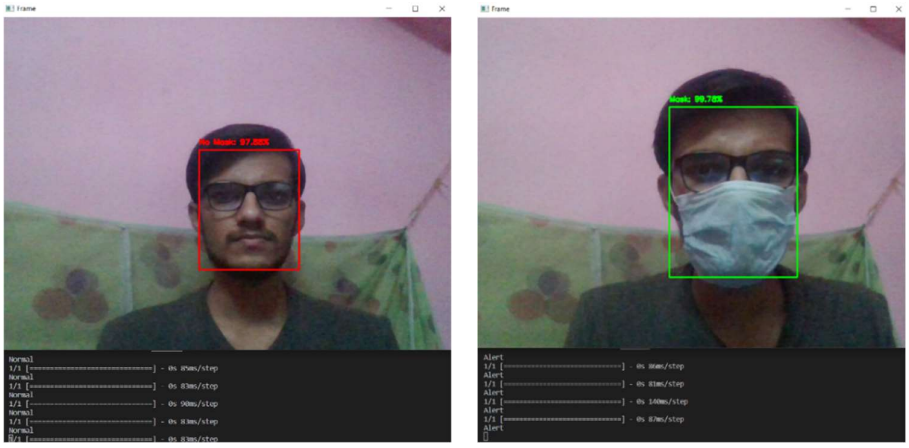

Figure 19: Model testing images for Alert system

Chapter 8

Final Overview

8.1 Result

First of all, The model used the dummy images of different persons wearing mask and not wearing masks together (Dummy images taken from Internet to test whether this model working fine for different faces in same frame or not) to check whether our model functioning correctly or not. In the figure 20, For testing the result this model used a picture which have two persons with mask, 2 persons without mask and two images of mask without face, and This got the result from this test-case that only the persons wearing mask and not wearing mask are detected but the image which doesn't contain the face is not detected. In figure 21, 22 and 23, For testing the result this model used images which contains various faces to test whether it works for all type of pictures or not, so This model got the perfect result from these Test cases as well.

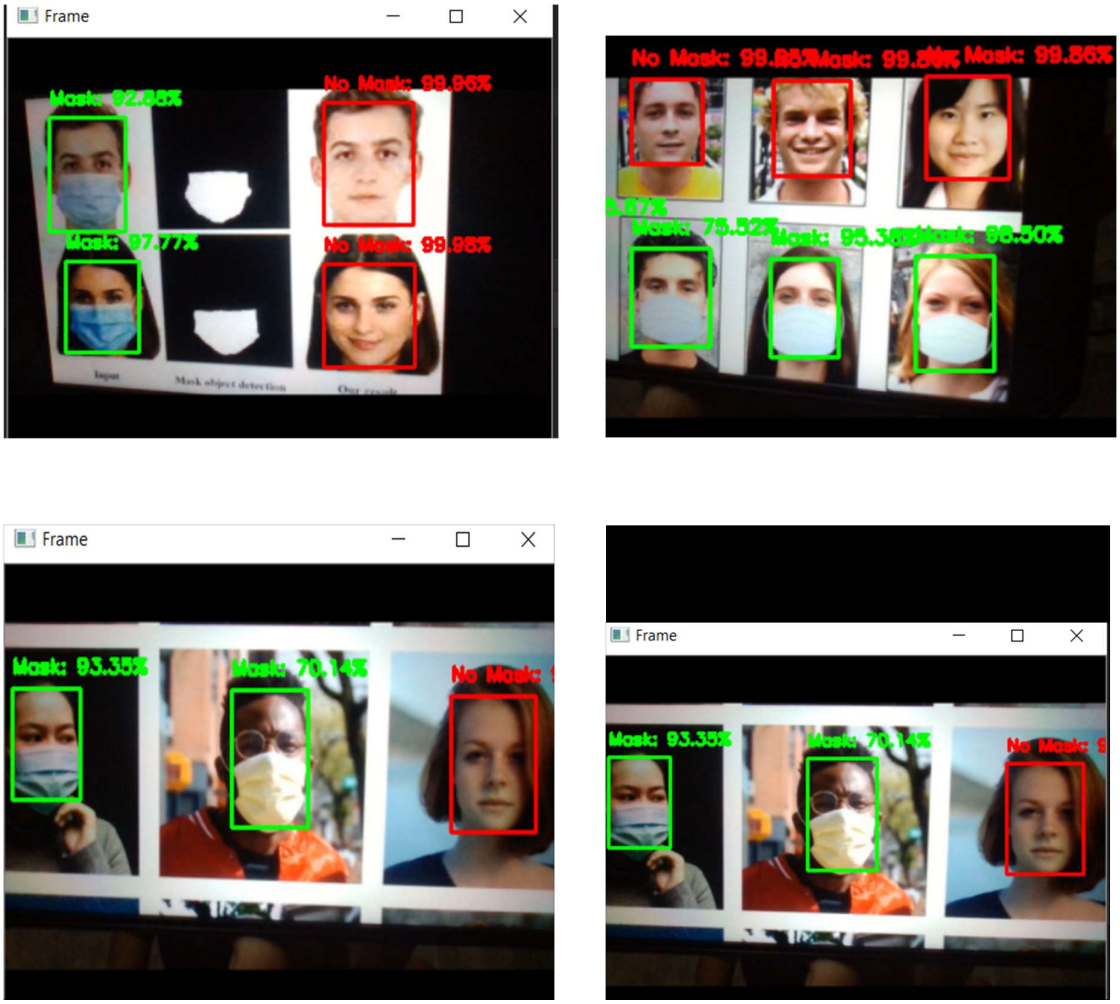

Figure 20,21,22,23: Dummy Images to test the model accuracy

Now This project used the algorithm it developed to give a security alert (whenever the person is wearing mask or covered their face with something) and tested using the livestreaming camera to detect whether it is giving the desired result or not.

Figure 24,25,26,27,28,29: Output of the alert system.

For testing whether alarm system working properly or not, The model is tested using wide range of faces and various face postures with mask and without mask and got the desired result with an accuracy of nearly 87%.

8.2 Conclusion

In this project, It has modeled a face mask detector using SSD architecture and transfer learning methods in neural networks. To train, validate and test the model, It is used the dataset that consisted of 4915 masked faces images and 4918 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, It has to be 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.

After the covid 19 era ends, It came in thought where to use this model so the idea came to implement an alarm system with this so that this trained model can work in an ATM for security purpose. Using face mask detection over there can moniter all the suspicious activity over there and can alert the guards if some unusual things happens inside the ATM. This model will detect whether the person wearing mask or not, If someone is covering his or her face, the alarm will be triggered and security get alert. It will be mandatory to go unmasked inside the ATM for the security purpose.

8.3 Limitation

Our model accuracy is currently 87% which is quite low which is currently planning to increasing it to 95% so that this model can get more accurate result. Also our model needs sufficient amount of light to detect the face properly, It is difficult to detect faces if there is darkness all around. In this model, It is trying to improve it with the help of the advanced algorithms of opencv python.

8.4 Future Prospects

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. In this project, It is 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. It will detect the faces and send the data to the server from a particular area about the percentage of people who are wearing mask or not. This detection live video feed can be implemented in web and desktop applications so that everyone can see the data. Software users can also get safety measures in case someone is not wearing a mask. Furthermore, This software can also be connected to the entrance gates and only people wearing face masks can come in. Also It is the plan to add a database in the ATM so that all the images capured in the frame can be kept in the database for the security purpose so that if any robbery or theft occurred in the ATM, police can identify the culprits immediately.

Chapter 9

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 detection," 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-usingpytorch-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/covid19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/ .

[10] 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.

[11] Francois Chollet "Xception: Deep Learning with Depthwise Separable Convolutions" in Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR), 2017, pp. 1251-1258.

[12] M. Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "MobileNetV2:Inverted Residuals and Linear Bottlenecks," in Proceedings of the IEEE conference on computer vision.

[13] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification- verification," in Advances in neural information processing systems, 2014, pp. 198hy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," 2014.

[14] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on, pp. 138–142, IEEE, 1994.

[15] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," CoRR abs/1411.7923, 2014.

[16] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, "A practical transfer learning algorithm for face verification," in Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 3208–3215, IE8– 1996.

[17] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," ACM computing surveys (CSUR), vol. 35, no. 4, pp. 399–458, 2003.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. KarpatEE, 2013.

[19] P. N. Belhumeur, J. P. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," Pattern Analysis and Machine Intelligence, IEEE Transactions on 19(7), pp. 711– 720, 1997.

[20] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, "A practical transfer learning algorithm for face verification," in Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 3208–3215, IEEE, 2013.

[21] Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. CoRR, abs/1406.4773, 2014. 1, 2, 3.ss

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. Nature, 1986. 2, 4