FACE MASK DETECTION

A PROJECT REPORT

Submitted by -
Harsh Kakaiya (20MIP10007)
Soumyadev Kundu (20MIP10011)
Vivek Kumar (20MIP10016)
Shreyash Jaiswal (20MIP10046)
Sambhav Jain (20MIP10050)

in partial fulfillment for the award of the degree of

INTEGRATED MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTHRIKALAN, SEHORE
MADHYA PRADESH – 466114

APRIL 2022

VIT BHOPAL UNIVERSITY, KOTHRIKALAN, SEHORE MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

Certified that this project report titled **"FACE MASK DETECTION"** is the bonafide work of **"HARSH KAKAIYA (20MIP10007), SOUMYADEV KUNDU (20MIP10011), VIVEK KUMAR (20MIP10016), SHREYASH JAISWAL (20MIP10046), SAMBHAV JAIN (20MIP10050)"** who carried out the project work under my supervision.Certified further that to the best of my knowledge the work reported at this time doesnot form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR
Dr. Pon Harshavardhanan
Program Chair - Integrated M.Tech. - CDS
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

PROJECT GUIDE
Dr. Pon Harshavardhanan,
Associate Professor Grade 2
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

The Project Exhibition-II Final Review was held on 23 April 2022.

ACKNOWLEDGEMENT

First and foremost, we would like to thank the Lord Almighty for his presence and immense blessings throughout the project work.

We wish to express my heartfelt gratitude to Dr. S. Sountharrajan, Head of the Department, School of Computer Science and Engineering for much of his valuable support and encouragement in carrying out this work.

We would like to thank my internal guide Dr. Pon Harshavardhanan, for continually guiding and actively participating in our project, and giving valuable suggestions to complete the project work.

We would like to thank all the technical and teaching staff of the School of Computer Science andEngineering, who extended directly or indirectly all support.

Last but not least, we are deeply indebted to our parents who have been the greatest support while we worked day and night on the project to make it a success.

LIST OF FIGURES AND GRAPHS

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CNN	Convolutional Neural Network
DL	Deep Learning
FC	Fully-Connected
IoT	Internet of Things
ML	Machine Learning
PCA	Principal Component Analysis
ROI	Region of Interest
SRCNet	Super-Resolution and Classification Network
WHO	World Health Organization

ABSTRACT

The end of 2019 saw the outbreak of the deadly virus known as COVID-19. The World Health Organization, along with several other health bodies, advised the usage of face masks as an effective preventive measure. Sadly, not everyone is following this, and this prompted us to make a face mask detection system that can detect people not wearing a face mask. To do this, we have fine-tuned the MobileNetV2 architecture along OpenCV's face detector to build a model that can detect people not wearing a mask in images and videos. The dataset that we used consisted of 1376 images, 690 images of people wearing a mask, and 686 images of people not wearing a mask. Our model achieved an accuracy of 99.35% in determining whether a person is wearing a mask or not.

CONTENTS

Chapter 1

PROJECT DESCRIPTION AND OUTLINE

1.1 Introduction

In view of the transmission of COVID-19, it was advised by the World Health Organization (WHO) to various countries to ensure that their citizens are wearing masks in public places. Prior to COVID-19, only a few people used to wear masks for the protection of their health from air pollution, and health professionals used to wear masks while they were practicing at hospitals. With the rapid transmission of COVID-19, the WHO has declared it as a global pandemic.

Artificial Intelligence (AI) techniques like Machine Learning (ML) and Deep Learning (DL) can be used in many ways for preventing the transmission of COVID-19. Machine learning and deep learning techniques allow to forecast the spread of COVID-19 and helpful to design an early prediction system that can aid in monitoring the further spread of disease. For the early prediction and diagnosis of complex diseases, emerging technologies like the Internet of Things (IoT), AI, big data, DL and ML are being used for the faster diagnosis of COVID-19.

The main aim of this work is to develop a deep learning model for the detection of persons who are not wearing a face mask. Image augmentation techniques are used to increase the diversity of the training data for enhancing the performance of the proposed model.

1.2 Motivation

Since December 2019, our world is suffering from a deadly virus known as COVID-19. In view of the transmission of COVID-19, it was advised by the World Health Organization to ensure that you are wearing masks in public places and gatherings. Earlier many people started to wear masks in fear of the virus. But, now many people are ignoring the guidelines, and refusing to wear masks in public places and gatherings. This motivated us to build a model that can detect people who are not wearing masks and help stop the spread of COVID-19.

1.3 Problem Statement

The problem that we are faced with is that of face mask detection. Face mask detection refers to the classification of an image containing a face on the basis of whether the face is covered by a face mask or not. This problem can be considered as an application of face detection, but the we are detecting a face covered by a face mask.

1.4 Objective of the Work

Our objective for building this project is to detect those people who are not wearing mask on public gathering places in this time of pandemic. This model can detect masks through both images and through web cam in real time.

1.5 Organization of the Project

This project report has been split up into seven chapters, this being the first. Chapter 2 gives an overview of the already existing work related to this project. In Chapter 3, we talk about the requirements in order to implement this project.

In Chapters 4 and 5 we talk about the methodology and implementation of the project. Any machine learning project can be broken down into two distinct steps – training and deployment. In the training step, our main focus will be on training a machine learning model (using Keras/TensorFlow) on the dataset, and saving the model to our disk. In the deployment step, we'll load the trained model from disk, and use it perform face detection on image and video, and classify the detected faces as with_mask or without_mask. In order to detect masks in images, we follow a two phase process – detect and find the location of the face(s) in the image, and then apply the trained mask classifier on that image. We will also be presenting an evaluation of our model. We'll also be reviewing how the dataset that we used was generated.

In Chapters 6 and 7 we discuss about the outcomes and applicability of the project, and conclude by discussing some limitations and future enhancements that can be done.

1.6 Summary

The rapid spread of the COVID-19 virus has forced us to change our social habits and adopt new norms advised by the WHO and other health bodies. Wearing of masks is the least one can do, but still many people do not follow this simple habit that can stop the spread of this virus. This motivated us to build a machine learning model that can detect such people. In this project, we presented a face mask classifier that can not only make predictions on static images, but also in video streams.

Chapter 2

RELATED WORK INVESTIGATION

2.1 Core Area of the Project

The core area of the project involves face mask detection, which can be considered an application of face detection. Face detection (also called facial detection) is an artificial intelligence-based technology used to find and identify human faces in images.

Most face detection techniques use algorithms and machine learning to find human faces within larger images. Face detection algorithms typically start by searching for human eyes, as they are one of the easiest features to detect. Then the algorithm might attempt to detect eyebrows, the mouth, nose, etc. Face detection is a key area in the field of computer vision and pattern recognition. Face *mask* detection refers to detecting whether a person is wearing a mask or not.

2.2 Existing Approaches

Qin and Li [1] designed a face mask identification method using the SRCNet classification network. They developed a new facemask-wearing condition identification method by combining image Super-Resolution and Classification Network (SRCNet). Their solution contained four main steps – image pre-processing, face detection and crop, image super resolution, and face mask wearing conditions identification. They achieved an accuracy of 98.70%.

Ejaz et al. [2] implemented the Principal Component Analysis (PCA) algorithm for masked and unmasked facial recognition. They concluded that PCA is efficient in recognising faces without a mask with an accuracy of 95% on average. However, the accuracy falls down to 68.75% in identifying faces with a mask.

Chavda et al. [3] used a Deep Learning based system that can detect improper wearing of face masks. Their system consisted of a dual-stage Convolutional Neural Network (CNN) architecture, and they were able to detect masked and unmasked faces.

2.3 Observations from the Investigation

From the literature reviewed, it can be observed that there are many techniques that one can use to perform face mask detection. It can also be seen that Principal Component Analysis does not give good results when compared to algorithms using a deep learning framework with a convolutional neural network architecture.

2.4 Summary

In this chapter, we discussed about the core area of the project which is face mask detection. We also presented various approaches and existing work related to this project, and we concluded that a deep learning framework with a convolutional neural network architecture gives the best results.

Chapter 3

REQUIREMENT ARTIFACTS

3.1 Introduction

We used the Python programming language for this project. Python is an interpreted high-level general-purpose programming language. We used Python due to its simple and readable syntax and also because of the vast number of libraries and packages available to perform machine learning. We have developed this project on a machine running Windows 10/11 Operating System.

Note: You should be able to *train* the model on Google Colab – a free Jupyter notebook environment that runs entirely in the cloud.

3.2 Hardware and Software Requirements

The hardware and software requirements are very basic and anyone working with any project related to machine learning might already have met all of these requirements.

3.2.1 Hardware Requirements

Table 3.1: Hardware Requirements

	Minimum	**Recommended**	**Maximum**
CPU	Inter Core i5 8th gen AMD Ryzen 5 1600x	Intel Core i5 9th gen AMD Ryzen 5 2600x	Intel Core i7 9th gen AMD Ryzen 7 4800H
RAM	8 GB	12 GB	16 GB
GPU	Intel integrated graphics Ryzen integrated graphics	Nvidia GTX 1650 AMD Radeon RX 570	-
Storage	5 GB HDD (SSD recommended)	7GB SSD	13 GB SSD
Webcam	Integrated webcam (3MP/720p/30fps)	External webcam (5MP/1080p/30fps)	-

3.2.2 Software Requirements

You need a Windows 10/11 machine, with the programming language Python 3.9.7 installed. The Python packages and libraries you need to install are:

1. TensorFlow (`tensorflow`): It is a free and open-source software library for machine learning and artificial intelligence. We have used it for pre-processing our data and building our model.
2. Scikit-learn (`sklearn`): It is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms. We have used this library for binarising class labels, segmenting our dataset, and printing a classification report.
3. OpenCV 2 (`cv2`): It is a library of programming functions mainly aimed at real-time computer vision.
4. Imutils (`imutils`): It includes series of convenience functions to make basic image processing functions easy.
5. NumPy (`numpy`): It is a module which adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
6. OS (`os`): This provides functions for interacting with the operating system.
7. Argparse (`argparse`): A module used for command-line parsing.

3.3 Summary

In this chapter the requirements needed to implement this project are stated. The programming language used to build this project is Python 3.9.7. We have used several Python libraries and modules to accomplish our objective. The libraries include TensorFlow, Scikit-learn, OpenCV2, Imutils, NumPy, OS, Argparse. The hardware requirements are very basic and anyone working in the domain of machine learning might already meet those requirements. There are no special requirements in order for one to implement this project.

Chapter 4

DESIGN METHODOLOGY AND ITS NOVELTY

4.1 Methodology and Goal

The proposed system focuses on how to identify the person on image/video stream wearing face mask with the help of computer vision and deep learning algorithm by using OpenCV, TensorFlow/Keras and Deep Learning. For the system to work properly, we have to process the input images in the following manner:

1. Detect the dimension and locations of the face or faces in the image.
2. If there is a face detected, classify that image as `with_mask` or `without_mask` using a face mask classifier.

The dataset that we used consists of 1376 images belonging to two classes:

- `with_mask`: 690 images
- `without_mask`: 686 images.

This dataset was created by Prajna Bhandary [4]. The dataset was created by taking normal images of faces, and adding artificial masks to them, creating an artificial dataset. Figure 4.1 shows a sample of the dataset used.

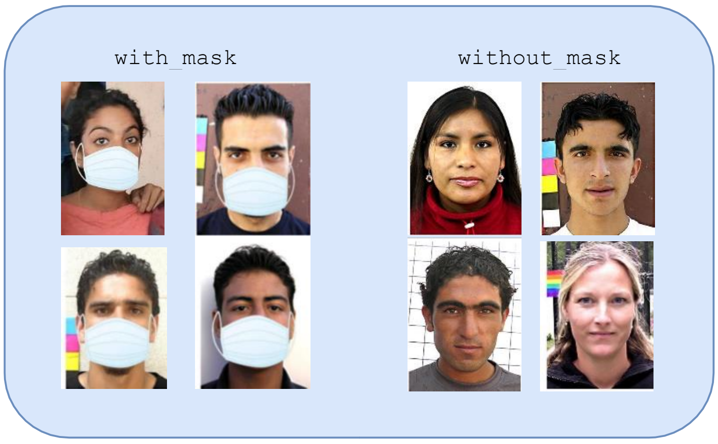


Figure 4.1: Dataset used

4.2 Functional Modules Design and Analysis

As stated earlier, there are two phases to this project. In the first phase, we detect faces in the image, and in the second phase we use our face mask classifier to classify that image into either of two categories – `with_mask` or `without_mask`.

4.2.1 First Phase

This phase focuses on finding the location and dimension of the face(s) present in an image, regardless of whether or not they wear a mask. For this, OpenCV's Deep Learning based face detection model is used with pre-trained weights based on the Caffe deep learning framework. Caffe is a deep learning framework made with expression, speed, and modularity in mind [5]. It is developed by Berkeley AI Research and by community contributors. As a result, the Region of Interest (ROI) is extracted, which contains the location, the width, and the height of the face.

4.2.2 Second Phase

The second phase involves detecting a mask. For this, we trained a machine learning model based on the MobileNetV2 architecture. For the training of the model, the input data of the neural network comes from a scaling of the colour images to a size of 224×224 pixels. The architecture used comprises of an average-pooling layer (7×7), a flatten layer; a hidden layer of 128 neurons with a "ReLu" activation function, a Dropout of 0.5, an output layer with two neurons, and a "Softmax" activation function. The settings used during the training of the model are as follows:

- Learning rate: 10^{-4}
- Epochs: 15
- Batch size: 32
- Optimiser: Adam
- Loss function: Binary Cross Entropy

Adam optimisation is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments [6].

The learning rate mentioned is an initial learning rate. We have applied a learning rate decay schedule, which refers to updating the learning rate during the training. After each epoch, the learning rate is divided by the number of epochs.

Once the training of the model is complete, we can then proceed to make predictions on the processed images which come through from our face detector.

4.3 Software Architectural Designs

As stated earlier, there are two phases to this project – detecting a face and extracting the ROI, and then pre-processing the ROI and passing it through our face mask classifier to classify the image.

Figure 4.2 shows the diagrammatic representation of this two phase process.

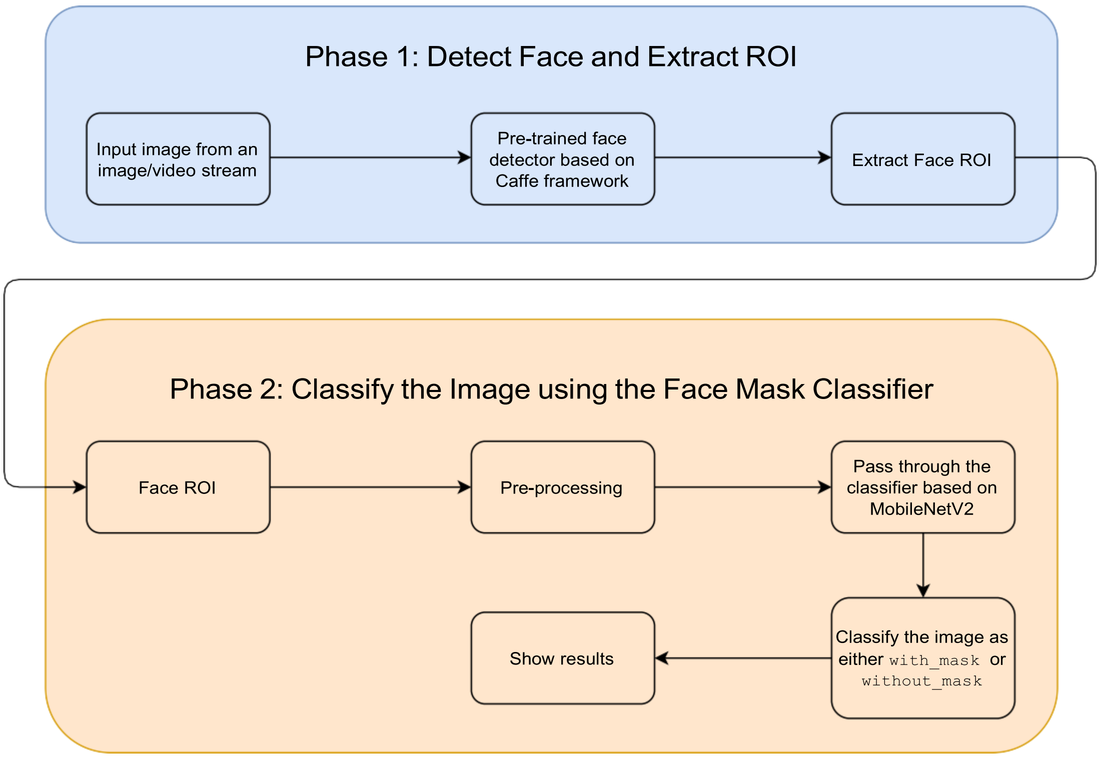

Phase 1: Detect Face and Extract ROI
Input image from an image/video stream
Pre-trained face detector based on Caffe framework
Extract Face ROI
Phase 2: Classify the Image using the Face Mask Classifier
Face ROI
Pre-processing
Pass through the classifier based on MobileNetV2
Classify the image as either with_mask or without_mask
Show results

Figure 4.2: Classifying an image as either `with_mask` or `without_mask`

4.4 Summary

This chapter talks about the methodology used in this project. The main objective is to classify people in images or video stream based on whether they are wearing a mask or not. In order to do this, we process the input images in two phases – first we detect the dimension and locations of the face(s) in the image, and then we classify that image using the face mask classifier which we have trained on the dataset. The dataset consists of 1376 images, belonging to two classes: `with_mask` and `without_mask`.

To find the location and dimensions of the face(s) in the image, we use OpenCV's Deep Learning based face detection model with pre-trained weights. To detect a mask in the image, we trained a model based on the MobileNetV2 architecture. The input image is scaled to 224×224 pixels. We trained this model for 15 epochs using the Adam optimiser and learning rate decay schedule.

Chapter 5

TECHNICAL IMPLEMENTATION AND ANALYSIS

5.1 Outline

In this chapter, the technical implementation of this project is explained. The project is split up into two phases – detecting a face and then classifying whether the person is wearing a mask or not. In order to perform the classification, we need to train a classification model. We provide an explanation of how we trained the model.

Once the training is done, we used the trained classifier in images and videos. The procedure for the same is also discussed. This chapter also discusses the performance of the model.

5.2 Technical Coding and Code Solutions

We wrote three Python scripts which are used to detect whether a person is or is not wearing a mask. The scripts are as follows:

- `train_mask_detector.py`: Accepts our input dataset and fine-tunes MobileNetV2 upon it to create our `mask_detector.model`. A training history plot containing accuracy/loss curves is also produced.
- `detect_mask_image.py`: Performs face mask classification in static images.
- `detect_mask_video.py`: Using your webcam, this script applies face mask classification to every frame in the stream.

The upcoming sections give a brief explanation of each script.

5.2.1 Training our Face Mask Classifier Model (`train_mask_detector.py`)

In order to train the model, we need to import the necessary libraries first. We import `tensorflow.keras` libraries which allow for:

- Data augmentation
- We load the MobileNetV2 classifier which we will be fine-tuning using pre-trained ImageNet weights
- Building a new Fully-Connected (FC) head
- Pre-processing
- Loading image data

Next, we used `sklearn` for binarising class labels, segmenting our dataset, and printing a classification report.

In order to pass the location of the dataset and our pre-trained face detection model, we use `argparse` to parse these command-line arguments. Next, we defined the deep learning hyper-parameters. Next, we pre-process our data.

In order to pre-process the data, we

- Grab all of the image paths in the dataset.
- Initialise the data and labels lists.
- Loop over the image paths in order to load and pre-process the images. Pre-processing steps include resizing to 224×224 pixels, conversion to array format, and scaling the pixel intensities in the input image to the range $[-1, 1]$.
- Append the pre-processed image and associated label to the data and labels lists, respectively.
- Ensure that our training data is in NumPy array format.

Next, we encoded our data labels, partitioned our dataset, and prepared it for data augmentation. Data augmentation is a technique to artificially create new training data from existing training data.

We used one-hot encoding to encode our class labels. One-hot encoding is a process of converting categorical data variables so they can be provided to machine learning algorithms to improve predictions. We segmented our data in to 80% for training and 20% for testing using `train_test_split` function of scikit-learn.

Next, we prepared the MobileNetV2 architecture for fine-tuning. Fine-tuning is a process that takes a model that has already been trained for one given task and then tunes it to make it perform a second similar task.

The fine-tuning steps involve:

1. Loading the MobileNetV2 with pre-trained ImageNet weights, leaving off the head of network.
2. Constructing a new fully-connected head, and appending it to the base in place of the old head.
3. Freezing the base layers of the network. The weights of these base layers will not be updated during the process of back-propagation, whereas the head layer weights will be tuned.

We then compiled and trained our model. We use the `Adam` optimiser, a learning rate decay schedule, and binary cross-entropy loss function.

After our model is trained, we saved our face mask classification model to our disk in order to apply it on images and video streams. Figure 5.1 shows the schematic representation of the training process.

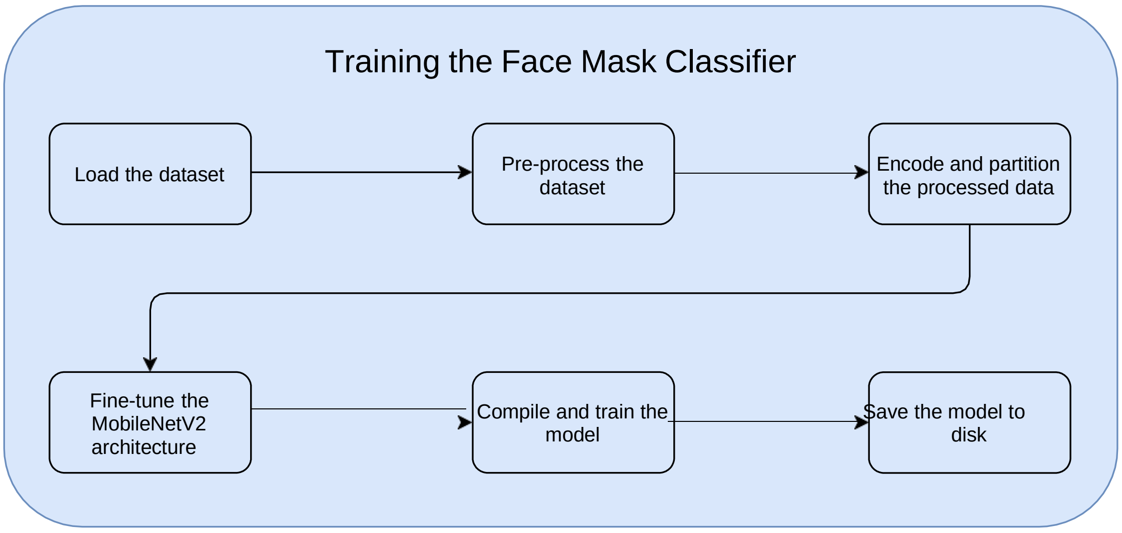

Training the Face Mask Classifier
Load the dataset
Pre-process the dataset
Encode and partition the processed data
Fine-tune the MobileNetV2 architecture
Compile and train the model
Save the model to disk

Figure 5.1: Training the Face Mask Classifier

5.2.2 Face Mask Detector for Images (`detect_mask_image.py`)

Once our face mask detector is trained, we can then on an image to classify the face as either `with_mask` or `without_mask`. This is a tree step process:

1. Load the face mask classifier model from disk.
2. Detect faces in the loaded image using pre-trained face detector.
3. Apply our face mask classifier on the image.

We use `tensorflow` to (i) load our face mask classifier model, and (ii) pre-process the input image. OpenCV is also imported to display the image and perform image manipulations. We parse some command-line arguments which include the locations of the image, our face mask classifier, and the face detector model. It also includes a confidence value – a probability threshold that is used to filter weak face detections.

Next, we loaded our face detector and face mask classifier models. Then, we pre-processed our input image. Pre-processing is handled by OpenCV's `blobFromImage` function. We resize the image to 300 × 300 pixels and perform mean subtraction. We then used our face detector to find where in the image all faces are. We need to ensure that our face detection meets our confidence threshold before we extract the ROI.

We then pre-process the face ROI in the same way we as we did during the training process. We then pass the processed image through our face mask classifier predict whether the image falls under `with_mask` or `without_mask`. We annotated the images accordingly and displayed the result.

5.2.3 Face Mask Detector for Video Streams (`detect_mask_video.py`)

We use the same processes as we did for detecting masks in images, but we do it for every frame of the webcam stream. In order to access the webcam stream, we imported the `VideoStream` class from `imutils.video`.

We wrapped our face detection and mask prediction logic into a single function so we use it inside our frame processing loop. We parse the same command-line arguments as we did for images. Next, we made the following initialisations:

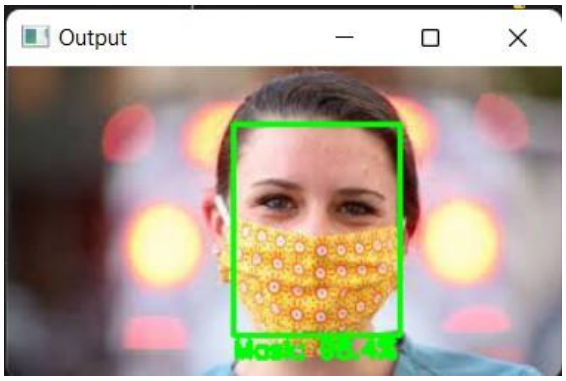
- Face detector
- Face mask classifier
- Webcam video stream

To initialise the video stream, we use the `VideoStream` class that we imported. Next, we begin looping over the frames in the stream. Inside the loop, we grabbed a frame from the stream, and passed it through the wrapper function that we defined earlier to make predictions on whether the people in the image are wearing a mask or not. Next, we annotated that frame and displayed it on the video stream based on the prediction made.

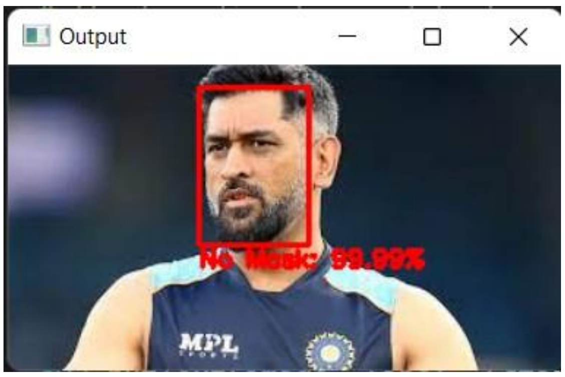
Figure 4.2 shows the schematic representation of the classification of an image or a frame of a video stream.

5.3 Test and Validation

In Figure 5.2, it can be seen that our model correctly finds the location of the face in the images and correctly classifies the two images.



(a) Correctly identified mask



(b) Correctly identified absence of mask

Figure 5.2: Testing on images

5.4 Performance Analysis

With the values obtained during the training, the precision, recall, and F1-score are calculated using Equations (5.2), (5.3), and (5.4), shown in Table 5.1. In the equations, TP refers to number of true positive predictions, TN refers to number of true negative predictions, FT refers to number of false positive predictions and FN refers to number of false negative predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.3}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.4}$$

The values obtained confirm the good performance of the classifier to detect when there is or is not a mask on the face. It can be noted that the F1-score for both classes has a value close to one that indicates the good performance of the model.

Table 5.1: Classification Report

	Precision	**Recall**	**F1-Score**
`with_mask`	0.99	0.99	0.99
`without_mask`	0.99	0.99	0.99
Accuracy			0.99
Macro avg	0.99	0.99	0.99
Weighted avg	0.99	0.99	0.99

As seen in Figure 5.3, convergence is reached at approximately 12 epochs, with an accuracy of 99.35%. The accuracy was calculated using Equation (5.1).

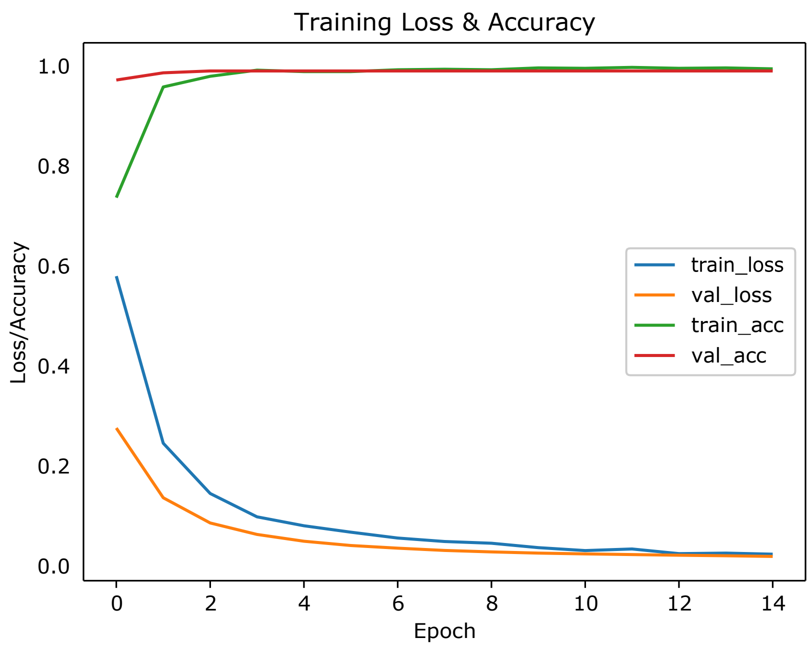


Figure 5.3: Training Loss and Accuracy

5.5 Summary

This chapter gave a brief explanation of the implementation of this project. The first step is to train the model. In order to train the model, we had to pre-process our input images which includes resizing them to 224×224 pixels, converting them into arrays, and scaling their pixel intensities to the range of $[-1, 1]$. We used one-hot encoding to encode our data labels. We used data augmentation to create artificial data from existing data. The dataset was partitioned into 80% training data and 20% testing data.

Next we fine-tuned the MobileNetV2 architecture by replacing its head with a new fully-connected head and freezing the base layers. We loaded this model with pre-trained ImageNet weights. We then compiled our model with the Adam optimiser, a learning rate decay schedule, and binary-cross entropy loss function. Once the model was trained, we saved in to the disk.

To detect masks in images and videos, we first loaded the trained model from disk. Then we detected faces in the loaded images using the pre-trained face detector, and then we applied our face mask classifier on those images. The input images were processed the same way as we did during the training process. In the end, we obtained a model that was 99.35% accurate.

Chapter 6

PROJECT OUTCOME AND APPLICABILITY

6.1 Outline

In this chapter we discuss about the various project outcomes and the applicability of this project. This project is very useful in various applications to identify people with or without masks. It can be used to prevent the spread of COVID-19 by ensuring that people are wearing masks in public places and places of work.

6.2 Project Outcomes

This project can be used to help stop the spread of COVID-19. This project is very useful to various camera teams to identify those people who are not wearing masks in public places. It can accept input and make predictions in both ways – detecting masks in images and photos, and detecting masks using live camera feed. It can help to immediately identify those people who are not wearing a mask. This project can also be used in higher platforms where a large number of people gather. Such places have a higher risk of becoming a hot-spot for the spread of the virus. So this project is very useful for the world in these times.

6.3 Project Applicability

The system can be used in the following places to identify people with or without masks:

Airports: This system can be used at airports to detect travellers without masks. Face data of travellers can be captured in the system at the entrance. If a traveller is found to be without a face mask, their picture can be sent to the airport authorities so that they can take quick action.

Hospitals: Hospitals can monitor if their staff is wearing masks during their shift or not. If any health worker is found without a mask, they will receive a notification with a reminder to wear a mask.

Offices: This system can be used at office premises to detect if employees are maintaining safety standards and wearing a face mask at work. If coupled with a facial recognition system, it can detect employees without masks and send them a reminder to wear a mask.

In Public Places: The Police could use this system on the video stream received via the surveillance cameras installed in public places, and enforce the wearing of face masks in public places.

There are numerous such applications of this system.

Chapter 7

CONCLUSION AND RECOMMENDATIONS

7.1 Outline

We used OpenCV, Keras/TensorFlow, and Deep Learning to train a face mask classifier. We fine-tuned the MobileNetV2 architecture on our dataset and obtained a classifier that is 99.35% accurate. We then took this face mask classifier and applied it to both images and real-time video streams by detecting faces in images/video, extracting each individual face, and applying our face mask classifier.

7.2 Limitations

The main limitation of this system is that it cannot keep a record for live video streams.

7.3 Future Enhancement

There can be some additions made to the dataset to further improve the applicability of our face mask classification model.

- Instead of artificially generated images, one can use actual images of people wearing masks. While the artificially generated dataset worked well, a real dataset is always preferred.
- One could also add images of faces that may confuse the classifier into thinking that the person is wearing a mask when they are not. Such images may include shirts wrapped around faces, a scarf covering the mouth, etc.

Another improvement could be using a single-phased approach. So instead of first detecting a face in the image and then applying our face mask classifier to each face, one could train a model that does both of these tasks together.

REFERENCES

[1] B. Qin and D. Li, "Identifying facemask-wearing condition using image super-resolution with classification network to prevent covid-19," *Sensors*, vol. 20, no. 18, p. 5236, 2020.

[2] M. S. Ejaz, M. R. Islam, M. Sifatullah and A. Sarker, "Implementation of principal component analysis on masked and non-masked face recognition," in *2019 1st international conference on advances in science, engineering and robotics technology (ICASERT)*, IEEE, 2019, pp. 1–5.

[3] A. Chavda, J. Dsouza, S. Badgujar and A. Damani, "Multi-stage cnn architecture for face mask detection," in *2021 6th International Conference for Convergence in Technology (I2CT)*, IEEE, 2021, pp. 1–8.

[4] P. Bhandary. (2020), [Online]. Available: `https://github.com/prajnasb/observations/tree/master/experiements/data` (visited on 12/10/2021).

[5] Y. Jia, E. Shelhamer, J. Donahue *et al.*, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.