

**Real-Time Face Mask Detection
Minor Project II**

Submitted by:

Manan Chaudhary (9919103178)

Kartikeya Consul (9919103183)

Megha Jain (9919103188)

Under the supervision of:

Mr. Surendra Kumar



Department of CSE/IT

**Jaypee Institute of Information Technology University,
Noida**

Month 2021

ACKNOWLEDGEMENT

We would like to place on record our deep sense of gratitude to _____, Designation, Jaypee Institute of Information Technology, India for his/her generous guidance, help and useful suggestions.

We express our sincere gratitude to _____, Dept. of _____, India, for his/her stimulating guidance, continuous encouragement and supervision throughout the course of present work.

We also wish to extend our thanks to _____ and other classmates for their insightful comments and constructive suggestions to improve the quality of this project work.

Signature(s) of Students

Name of Students (Enrolment)

Manan Chaudhary (9919103178)

Kartikeya Consul (9919103183)

Megha Jain (9919103188)

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and beliefs, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma from a university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place:

Date:

Name: Kartikeya Consul

Enrolment No.: 9919103183

Name: Megha Jain

Enrolment No.: 9919103188

Name: Manan Chaudhary

Enrolment No.: 9919103178

CERTIFICATE

This is to certify that the work titled “Real-Time Face Mask Detection” submitted by: Manan Chaudhary, Kartikeya Consul and Megha Jain of B.Tech of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of any other degree or diploma.

Digital Signature of Supervisor

Name of Supervisor

Designation

Date

ABSTRACT

Usage of facemask has been increasing day by day due to the COVID-19 pandemic. It is necessary to teach people to wear mask to COVID to rise. However, it is also difficult to control people in over-populated country like us. For this reason, we can use facemask detection model using the security camera in the public places and control people remotely. In this report, we have developed a facemask detector using python codes and artificial intelligence. Nowadays, AI is being used in almost every sector. Smart problem needs smart solutions. AI helps people to reduce human effort and time. AI can be created by training the machine. To train the machine, multiple machine language or programming language can be used because machine cannot understand human language. So that machine needs to be trained using the programming language tools. As previously mentioned, we have used Python to train our model. To train the model, several packages of python were needed. Also, after training the model, we have created a video streaming model to detect faces. These processes will be discussed in depth in this report.

TABLE OF CONTENTS

	Page No.
Abstract	i
List of Figures	iv
Chapter 1: INTRODUCTION.....	1
1.1 Background of the Study	1
1.2 Statement of the Problem	1
1.3 Objectives of the Study	1
1.5 Motivation of the Study	1
1.6 Limitations of the Study	1
Chapter 2: RESEARCH DESIGN	3
2.1 Dataset and Data Types	3
2.2 Research Design	4
2.3 Data Analysis Plan	4
Chapter 3: RESEARCH ANALYSIS & FINDINGS.....	5
3.1 Installing the Dependencies	5
3.2 Data Preprocessing	7
3.2.1 Mask Detector Model	7
3.3 Training Model	9
3.4 Using models in Real Time Camera	14
Chapter 4: DISCUSSION	20
4.1 Conclusion	20
4.2 Suggestion for Future Research and Recommendations	20

LIST OF FIGURES

Figure 1: With Mask Dataset	3
Figure 2: Without Mask Dataset	3
Figure 3: Research Design	4
Figure 4: Face Mask Detector (Without Mask)	4
Figure 5: Face Mask Detector (With Mask)	4
Figure 6: Requirements	5
Figure 7: Installing all the Packages Together	6
Figure 8: Installing all the Packages Together (1)	6
Figure 9: Installing all the Packages Together (2)	7
Figure 10: Importing Packages	8
Figure 11: Setting Directory and Categories and looping them.	8
Figure 12: Encoding Labels.....	9
Figure 13: Mobile-Net Neural Network	13
Figure 14: Setting Initial Learning Rate, Epochs and Batch Size.....	10
Figure 15: Data Augmentation and Creating Base-Model.....	11
Figure 16: Creating Head Model and Looping Base Model.....	11
Figure 17: Compiling Head & Base Model and Fitting the Mode.....	12
Figure 18: Making Prediction on the Testing Set	12
Figure 19: Developing Codes to Plot Model.....	13
Figure 20: Plot (Training Accuracy and Loss)	13
Figure 21: Face Detector Files	14
Figure 22: Importing Packages	14
Figure 23: Defining Frame, Face & Mask Detector.....	15
Figure 24: Looping over the detections.....	15
Figure 25: Ensuring the Dimension of the Frame to Bound Properly.....	16
Figure 26: Making Predictions	16

Figure 27: Loading the Face and Mask Detector Model.....	17
Figure 28: Looping over the Frames from the Video Stream.....	17
Figure 29: Looping over the Detected Face Locations and Their Corresponding	18
Figure 30: Showing the Output Frame & Cleaning Up the Interface.....	18

Chapter 1

1. INTRODUCTION

The world is facing a huge health crisis due to the rapid transmission of coronavirus (COVID-19). Several guidelines were issued by the World Health Organization (WHO) for protection against the spread of coronavirus. According to WHO, the most effective preventive measure against COVID-19 is wearing a mask in public places and crowded areas, but it is very difficult to monitor people manually in these areas. To maintain this, we might take help of AI to make a model to help us monitor people whether they are wearing a mask or not.

1.1. Background of the Study

In this paper, a CNN (Convolutional Neural Network) model is proposed to automate the process of identifying the people who are not wearing mask. The proposed model is built by fine-tuning the pre-trained state-of-the-art deep learning model, MobileNetV2. The proposed model is trained and tested on the Face Mask Detection Dataset. Image augmentation technique is adopted to address the limited availability of data for better training and testing of the model. The model outperformed the other recently proposed approaches by achieving an accuracy of 99.6% during training and 99.8% during testing.

1.2. Statement of the problem

Not wearing mask is a huge problem and the barrier to prevent Covid-19 crisis. It is not an easy task to aware people about this pandemic. This pandemic causes great economic crisis to the world. We saw GDP crash of many countries in the previous year. The world economy is still not stable. In many countries Covid-19 cases are still rising in a rapid way. Before vaccination is widely available, it is so difficult to stop this virus to rise. Only way is prevention. It is said that 'Prevention is better than Cure'. To prevent this virus, we need to be aware, wear mask and sanitize ourselves on a regular basis.

1.3. Objectives of the Study

- To make a face mask detector proto type.
- To monitor people in public places that they are following prevention protocol.
- To check if a person is wearing mask or not
- To make the face mask detection process from manual to automatic.
- To lower costs and human effort.
- To maximize the effectiveness of the process.
- To learn about deep learning methods through the research work.
- To develop my knowledge and experience by working on this project.

1.4. Motivation of the Study

- This reduces costs as well as human effort
- Authorities can monitor more people remotely by using security cameras only
- Increasing public awareness will be much easier
- Researcher can observe human behaviour towards the pandemic
- Maximize the effectiveness of the process

1.5. Limitations of the Study

- The model is smaller here as the dataset is smaller
- Lack of proper knowledge in the related fields.

- The accuracy of the model might not be the best as the data is limited
- Lack of resources
- Lower implementation possibility as it is a sample project

2.1. Dataset and Data Types

2.2. Research Design

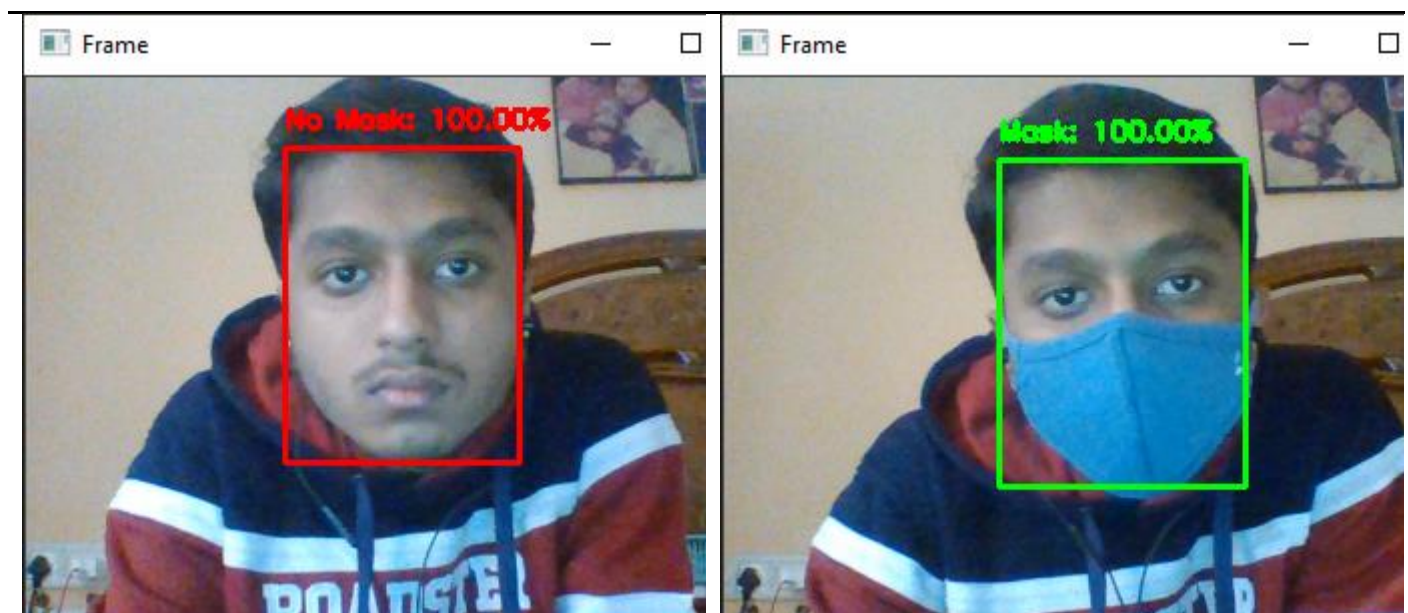
This project is actually a quantitative research. Because we had to collect data, analyse and visualize them with python programming language, train model using the python tools and finally get an output from a raw input, which can detect mask in human face. All the characteristics show that our paper is actually a result of quantitative research. In addition, this project is descriptive research as the aim of the project is to develop model with higher accuracy and precision. We can also call this research as an experimental as well as an action research. Preparation of this project can be described shortly through the following figure.



Figure 3: Research Design

2.3. Data Analysis Plan

As previously mentioned, the datasets contain different types images divided into two sub-folders. We have used Python to analyse data. With the help of the data, firstly we tried to train machine and develop a mask detector model. Then we collected a face detector from online (S., 2020). After that, we combined both the detector to develop face mask detector. After the development is completed, we implemented it into real time using our laptop camera.



Face Mask Detector (Without Mask)

Figure 4

Face Mask Detector (With Mask)

Figure 5

3. RESEARCH ANALYSIS & FINDINGS

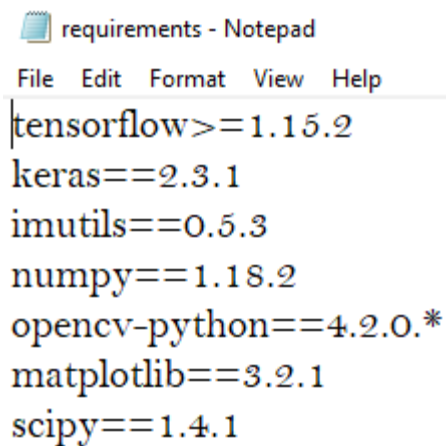
3.1 Installing the dependencies

To develop our model and using the model in the real time camera, we first need to install the following dependencies or python packages.

- i. Tensorflow (version 1.15.2)
- ii. Keras (version 1.15.2)
- iii. Imutils (version 1.15.2)
- iv. Numpy (version 1.15.2)
- v. Opencv (version 1.15.2)
- vi. Matplotlib (version 1.15.2)
- vii. Scipy (version 1.15.2)

To install these packages we can follow the way as follows:

For installing all together, we first need to create a txt file using notepad as the following and save this txt file as requirements to the working folder.



```
requirements - Notepad
File Edit Format View Help
tensorflow>=1.15.2
keras==2.3.1
imutils==0.5.3
numpy==1.18.2
opencv-python==4.2.0.*
matplotlib==3.2.1
scipy==1.4.1
```

Figure 6: Requirements

Then we need to run the following code:


```

Anaconda Prompt (anaconda3) - pip install -r requirements.txt
(base) C:\Users\consu\OneDrive\Desktop\College Folder\Sem-5\Minor Project\Face-Mask-Detection-master\Face-Mask-Detection-master>pip install -r requirements.txt
Collecting keras==2.3.1
  Downloading Keras-2.3.1-py2.py3-none-any.whl (377 kB)
    | 377 kB 3.3 MB/s
Collecting imutils==0.5.3
  Downloading imutils-0.5.3.tar.gz (17 kB)
Collecting numpy==1.18.2
  Downloading numpy-1.18.2-cp38-cp38-win_amd64.whl (12.8 MB)
    | 12.8 MB 3.3 MB/s
Collecting opencv-python==4.2.0.*
  Downloading opencv-python-4.2.0.34-cp38-cp38-win_amd64.whl (33.1 MB)
    | 33.1 MB 3.3 MB/s
Collecting matplotlib==3.2.1
  Downloading matplotlib-3.2.1-cp38-cp38-win_amd64.whl (9.2 MB)
    | 9.2 MB 6.8 MB/s
Collecting scipy==1.4.1
  Downloading scipy-1.4.1-cp38-cp38-win_amd64.whl (31.0 MB)
    | 31.0 MB 3.3 MB/s
Requirement already satisfied: opt-einsum==3.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (3.3.0)
Requirement already satisfied: wrapt==1.12.1 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (1.12.1)
Requirement already satisfied: termcolor==1.1.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (1.1.0)
Requirement already satisfied: astunparse==1.6.3 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (1.6.3)
Requirement already satisfied: tensorflow==1.15.2 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (2.4.0)
Requirement already satisfied: protobuf==3.9.2 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (3.14.0)
Requirement already satisfied: flatbuffers==1.12.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (1.12.0)
Requirement already satisfied: google-pasta==0.2 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (0.2.0)
Requirement already satisfied: keras-preprocessing==1.1.2 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (1.1.2)
Requirement already satisfied: wheel==0.35 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (0.35.1)
Requirement already satisfied: gast==0.3.3 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (0.3.3)
Requirement already satisfied: absl-py==0.10 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (0.11.0)
Requirement already satisfied: tensorflow-estimator==2.5.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (2.4.0)
Requirement already satisfied: grpcio==1.32.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (1.32.0)
Requirement already satisfied: six==1.15.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (1.15.0)
Requirement already satisfied: h5py==2.10.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (2.10.0)
Requirement already satisfied: typing-extensions==3.7.4 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (3.7.4.3)
Collecting keras-applications==1.0.6
  Downloading Keras Applications-1.0.6-py3-none-any.whl (50 kB)
    | 50 kB 1.6 MB/s
Requirement already satisfied: pyyaml in c:\users\consu\anaconda3\lib\site-packages (from keras==2.3.1->-r requirements.txt (line 2)) (5.3.1)
Requirement already satisfied: pyparsing==2.0.4 in c:\users\consu\anaconda3\lib\site-packages (from matplotlib==3.2.1->-r requirements.txt (line 6)) (2.4.7)
Requirement already satisfied: cytoolz==0.10 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (0.10.0)
Requirement already satisfied: kiwisolver==1.0.1 in c:\users\consu\anaconda3\lib\site-packages (from matplotlib==3.2.1->-r requirements.txt (line 6)) (1.3.0)
Requirement already satisfied: python-dateutil==2.1 in c:\users\consu\anaconda3\lib\site-packages (from matplotlib==3.2.1->-r requirements.txt (line 6)) (2.8.1)
Requirement already satisfied: google-auth-oauthlib==0.5 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (0.4.2)
Requirement already satisfied: tensorboard-plugin-wit==1.6.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorboard==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (1.7.0)
Requirement already satisfied: setuptools==41.0.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorboard==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (50.3.1.post20201107)
Requirement already satisfied: werkzeug==0.11.15 in c:\users\consu\anaconda3\lib\site-packages (from tensorboard==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (1.0.1)
Requirement already satisfied: markdown==2.6.8 in c:\users\consu\anaconda3\lib\site-packages (from tensorboard==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (3.3.3)
Requirement already satisfied: requests==2.21.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorboard==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (2.24.0)
Requirement already satisfied: google-auth==2.16.3 in c:\users\consu\anaconda3\lib\site-packages (from tensorboard==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (1.24.0)
Requirement already satisfied: requests-oauthlib==0.7.0 in c:\users\consu\anaconda3\lib\site-packages (from google-auth-oauthlib==0.5->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (1.3.0)
Requirement already satisfied: urllib3==1.25.0 in c:\users\consu\anaconda3\lib\site-packages (from requests==2.21.0->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (1.25.11)
Requirement already satisfied: certifi==2017.4.17 in c:\users\consu\anaconda3\lib\site-packages (from requests==2.21.0->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (2021.5.30)
Requirement already satisfied: idna==2.5 in c:\users\consu\anaconda3\lib\site-packages (from requests==2.21.0->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (2.10)
Requirement already satisfied: chardet==3.0.2 in c:\users\consu\anaconda3\lib\site-packages (from requests==2.21.0->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (3.0.4)
Requirement already satisfied: pyasn1-modules==0.2.1 in c:\users\consu\anaconda3\lib\site-packages (from google-auth==2.16.3->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (0.2.8)
Requirement already satisfied: rsa==3.14 in c:\users\consu\anaconda3\lib\site-packages (from google-auth==2.16.3->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (4.6)
Requirement already satisfied: cachetools==5.0 in c:\users\consu\anaconda3\lib\site-packages (from google-auth==2.16.3->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (4.2.0)
Requirement already satisfied: oauthlib==3.0.0 in c:\users\consu\anaconda3\lib\site-packages (from requests-oauthlib==0.7.0->google-auth-oauthlib==0.5->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (3.1.0)
Requirement already satisfied: pyasn1==0.5.0 in c:\users\consu\anaconda3\lib\site-packages (from pyasn1-modules==0.2.1->google-auth==2.16.3->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (0.4.8)
Building wheels for collected packages: imutils
  Building wheel for imutils (setup.py) ... done

```

Figure 7: Installing all the Packages Together

```

Anaconda Prompt (anaconda3)
Collecting keras-applications==1.0.6
  Downloading Keras Applications-1.0.6-py3-none-any.whl (50 kB)
    | 50 kB 1.6 MB/s
Requirement already satisfied: pyyaml in c:\users\consu\anaconda3\lib\site-packages (from keras==2.3.1->-r requirements.txt (line 2)) (5.3.1)
Requirement already satisfied: pyparsing==2.0.4 in c:\users\consu\anaconda3\lib\site-packages (from matplotlib==3.2.1->-r requirements.txt (line 6)) (2.4.7)
Requirement already satisfied: cytoolz==0.10 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (0.10.0)
Requirement already satisfied: kiwisolver==1.0.1 in c:\users\consu\anaconda3\lib\site-packages (from matplotlib==3.2.1->-r requirements.txt (line 6)) (1.3.0)
Requirement already satisfied: python-dateutil==2.1 in c:\users\consu\anaconda3\lib\site-packages (from matplotlib==3.2.1->-r requirements.txt (line 6)) (2.8.1)
Requirement already satisfied: google-auth-oauthlib==0.5 in c:\users\consu\anaconda3\lib\site-packages (from tensorflow==1.15.2->-r requirements.txt (line 1)) (0.4.2)
Requirement already satisfied: tensorboard-plugin-wit==1.6.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorboard==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (1.7.0)
Requirement already satisfied: setuptools==41.0.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorboard==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (50.3.1.post20201107)
Requirement already satisfied: werkzeug==0.11.15 in c:\users\consu\anaconda3\lib\site-packages (from tensorboard==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (1.0.1)
Requirement already satisfied: markdown==2.6.8 in c:\users\consu\anaconda3\lib\site-packages (from tensorboard==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (3.3.3)
Requirement already satisfied: requests==2.21.0 in c:\users\consu\anaconda3\lib\site-packages (from tensorboard==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (2.24.0)
Requirement already satisfied: google-auth==2.16.3 in c:\users\consu\anaconda3\lib\site-packages (from tensorboard==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (1.24.0)
Requirement already satisfied: requests-oauthlib==0.7.0 in c:\users\consu\anaconda3\lib\site-packages (from google-auth-oauthlib==0.5->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (1.3.0)
Requirement already satisfied: urllib3==1.25.0 in c:\users\consu\anaconda3\lib\site-packages (from requests==2.21.0->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (1.25.11)
Requirement already satisfied: certifi==2017.4.17 in c:\users\consu\anaconda3\lib\site-packages (from requests==2.21.0->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (2021.5.30)
Requirement already satisfied: idna==2.5 in c:\users\consu\anaconda3\lib\site-packages (from requests==2.21.0->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (2.10)
Requirement already satisfied: chardet==3.0.2 in c:\users\consu\anaconda3\lib\site-packages (from requests==2.21.0->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (3.0.4)
Requirement already satisfied: pyasn1-modules==0.2.1 in c:\users\consu\anaconda3\lib\site-packages (from google-auth==2.16.3->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (0.2.8)
Requirement already satisfied: rsa==3.14 in c:\users\consu\anaconda3\lib\site-packages (from google-auth==2.16.3->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (4.6)
Requirement already satisfied: cachetools==5.0 in c:\users\consu\anaconda3\lib\site-packages (from google-auth==2.16.3->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (4.2.0)
Requirement already satisfied: oauthlib==3.0.0 in c:\users\consu\anaconda3\lib\site-packages (from requests-oauthlib==0.7.0->google-auth-oauthlib==0.5->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (3.1.0)
Requirement already satisfied: pyasn1==0.5.0 in c:\users\consu\anaconda3\lib\site-packages (from pyasn1-modules==0.2.1->google-auth==2.16.3->tensorflow==2.4->tensorflow==1.15.2->-r requirements.txt (line 1)) (0.4.8)
Building wheels for collected packages: imutils
  Building wheel for imutils (setup.py) ... done

```

Figure 8: Installing all the Packages Together (1)

```

Anaconda Prompt (anaconda3)
n requirements.txt (line 1)) (4.2.0)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\consu\anaconda3\lib\site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboa
nd==2.4->tensorflow>1.15.2->-r requirements.txt (line 1)) (3.1.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\consu\anaconda3\lib\site-packages (from pyasn1-modules>=0.2.1->google-auth<2,>=1.6.3->tensorboard==2.4->
tensorflow>1.15.2->-r requirements.txt (line 1)) (0.4.8)
Building wheels for collected packages: imutils
  Building wheel for imutils (setup.py) ... done
  Created wheel for imutils: filename=imutils-0.5.3-py3-none-any.whl size=25855 sha256=fce1e219e12600551e2563baa6ba15e0bafba926642e15676b2ba7f7d2af3784
  Stored in directory: c:\users\consu\appdata\local\pip\cache\wheels\c0\d6\0f\b0c3892b70c59f0d202f8619a449f7d14cb839a0af2f943869
Successfully built imutils
Installing collected packages: numpy, keras-applications, scipy, keras, imutils, opencv-python, matplotlib
  Attempting uninstall: numpy
    Found existing installation: numpy 1.19.2
    Uninstalling numpy-1.19.2:
      Successfully uninstalled numpy-1.19.2
  Attempting uninstall: scipy
    Found existing installation: scipy 1.5.2
    Uninstalling scipy-1.5.2:
      Successfully uninstalled scipy-1.5.2
  Attempting uninstall: keras
    Found existing installation: Keras 2.4.3
    Uninstalling Keras-2.4.3:
      Successfully uninstalled Keras-2.4.3
  Attempting uninstall: imutils
    Found existing installation: imutils 0.5.4
    Uninstalling imutils-0.5.4:
      Successfully uninstalled imutils-0.5.4
  Attempting uninstall: opencv-python
    Found existing installation: opencv-python 4.5.1.48
    Uninstalling opencv-python-4.5.1.48:
      Successfully uninstalled opencv-python-4.5.1.48
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.3.2
    Uninstalling matplotlib-3.3.2:
      Successfully uninstalled matplotlib-3.3.2
ERROR: After October 2020 you may experience errors when installing or updating packages. This is because pip will change the way that it resolves dependency conflicts.

We recommend you use --use-feature=2020-resolver to test your packages with the new resolver before it becomes the default.

tensorflow 2.4.0 requires numpy==1.19.2, but you'll have numpy 1.18.2 which is incompatible.
Successfully installed imutils-0.5.3 keras-2.3.1 keras-applications-1.0.8 matplotlib-3.2.1 numpy-1.18.2 opencv-python-4.2.0.34 scipy-1.4.1
(base) C:\Users\consu\OneDrive\Desktop\College Folder\Sem-5\Minor Project\Face-Mask-Detection-master\Face-Mask-Detection-master>

```

Figure 9: Installing all the Packages Together (2)

3.2 Data Pre-processing

After installing all the necessary packages, we can now work on the data preprocessing and training a model for further progress.

3.2.1 Mask Detector Model

```

# import the necessary packages
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

```

Figure 10: Importing Packages

At first we need to import all the necessary packages into the python file. Packages we need are shown above.

```

DIRECTORY = r"C:\Users\consu\Downloads\Face-Mask-Detection-master\Face-Mask-Detection-master\dataset"
CATEGORIES = ["with_mask", "without_mask"]

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")

data = []
labels = []

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

        data.append(image)
        labels.append(category)

```

Figure 11: Setting Directory and Categories and Looping Them

Here we can see our directory is set on C drive containing Face-Mask-Detection-master folder. Moreover, inside this DIRECTORY, we mentioned where our dataset folder is present. Therefore, machine can understand the dataset it needs to use to run the program. Inside CATGORIES, We have mentioned the values named with_mask and without_mask. These are also the folders present in the DIRECTORY. We have already discussed more about the dataset in 'Chapter-2'. Here, we have created two empty lists called 'data' and 'labels'. Then, later we appended the entire images array into this 'data' list and inside the 'label' list; we appended the label of those images whether it is with mask or without mask. After that, we have looped through the CATAGORIES by using for command. By 'os.path.join' command, we tried to first loop through the with_mask and then through the without_mask. Then we listed down all the images into the particular by using 'listdir' command. Then we loaded images using the pre-processing function 'load_img' of the package 'keras' that was imported previously. We gave the images a target size of (224, 224) which is the height and width of the images we wanted to be. Then we converted all the images into array by using another 'keras' function called 'img_to_array'. We have used the 'mobilenet' function of the package called 'tensorflow'. That is why we used 'preprocess_input' function here.

After pre-processing the images successfully, we appended the images into the previously created 'data' list and category into the 'labels' list.

```
# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

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

(trainX, testX, trainY, testY) = train_test_split(data, labels,
                                                    test_size=0.20, stratify=labels, random_state=42)
```

Figure 12: Encoding Labels

Now, the images are in numerical values which are good for machine to understand easily. For that, we have used 'LabelBinarizer' function from the package 'SkLearn'. Then we converted the 'labels' containing with_mask and without_mask values into categorical. It converted the labels into numerical value. After converting both the lists into numerical value, we converted both the lists into 'np' array because deep learning models only work in array format. Then we used 'train_test_split' to split our training and testing data. Here the test_size is 0.20, which indicates that we have given 20% of the images for testing purpose and the rest 80% for the training purpose. It is always good to give more data for training purpose to get a good test result. We have used stratify into labels which is nothing but classify the labels and the random_state indicates the set of training and testing split we are getting. It actually does not matter what number is this. It does not affect much on the split. If we provide 'none' in the random_state, it will just itself choose a number randomly to decide the splitting of train and test indices.

3.3 Training Model

To train the model, we used Convolutional Neural Network with a slight change. To understand the change, we first need to understand convolutional neural network (CNN) model. We have used MobileNet (a version of ConvNet) because MobileNet is faster than ConvNet as it uses depth-wise separable convolution which means it uses less parameters. That is why MobileNet is less accurate too. However, in our scenario, using MobileNet did not make a much difference in the outcome.

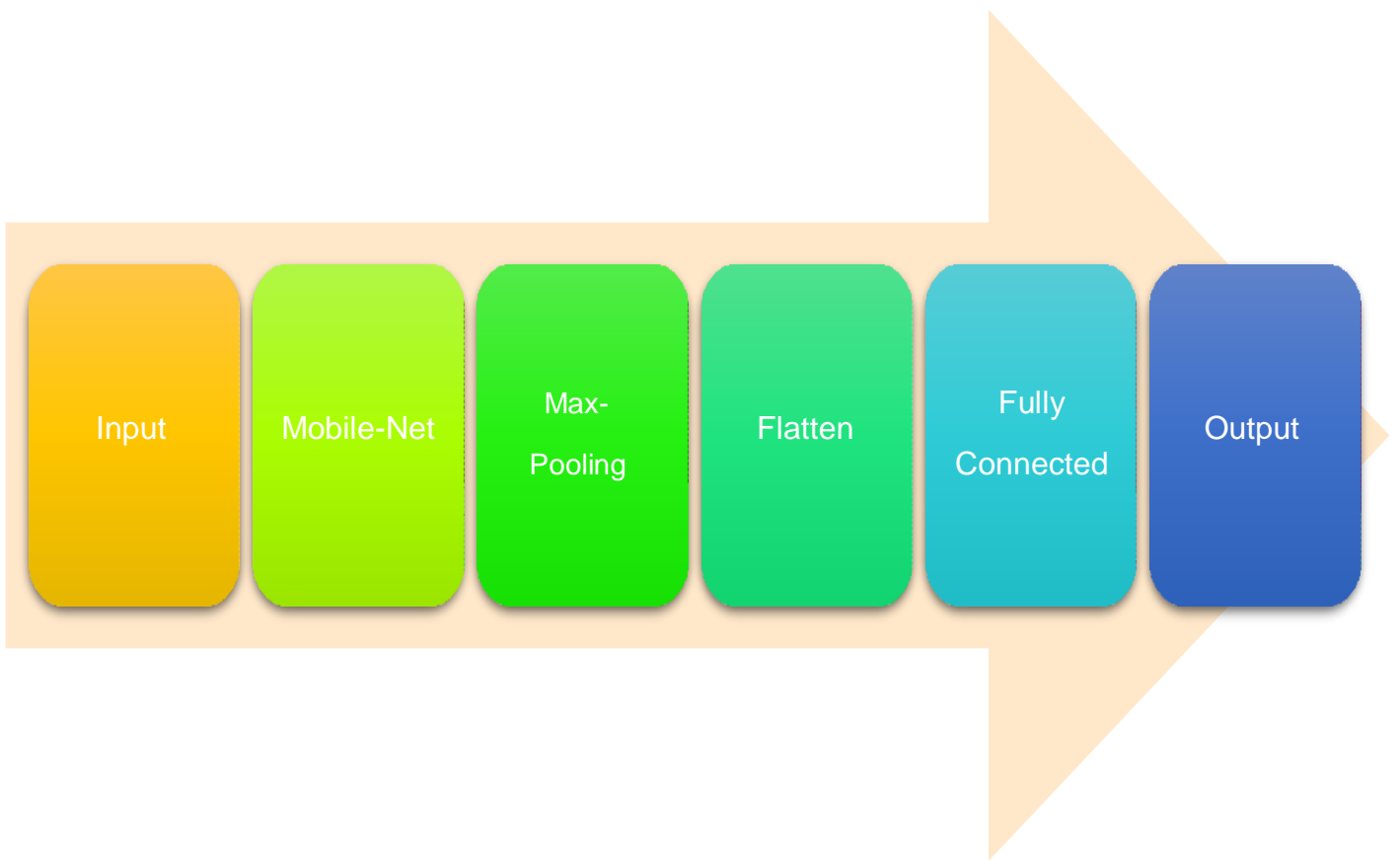


Figure 13: Mobile-Net Neural Network

```
# initialize the initial learning rate, number of epochs to train for,  
# and batch size  
INIT_LR = 1e-4  
EPOCHS = 20  
BS = 32
```

Figure 14: Setting Initial Learning Rate, Epochs and Batch Size

Here, we have initialised our learning rate as $1e^{-4}$. Keeping learning rate less helps to calculate the loss properly. By doing this, we can get a better accuracy easily. We have also provided 20 EPOCHS and our batch size is 32.

We have previously mentioned that we have used Mobile-Net which will generate two models. One is the Mobile-Net model and the output of this model will create a regular model that we have developed. These models are called as Head-model and Base-model in our project.

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

# load the MobileNetV2 network, ensuring the head FC layer sets are
# left off
baseModel = MobileNetV2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))
```

Figure 15: Data Augmentation and Creating Base-Model

Here, we have used 'ImageDataGenerator'. It actually creates many images from one single image by changing some of the properties. It has functionalities like shifting image, rotating image, flipping image, zooming image etc. Here, we have changed some properties as shown in the figure above. Also, we have loaded the 'MobileNetV2' for creating the models as previously mentioned. Here, we mentioned the weights, include_top, input_tensor and shape of the inputs. By using this, we have created our Base-Model.

```
# construct the head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False
```

Figure 16: Creating Head Model and Looping Base Model

After creating our base-Model, we have created our head-Model. As previously told, head-Model will be created using the output of the base-Model; we have passed the output of the base-Model into the head-Model. Then we have created the pooling by using 'AveragePooling2D' function. The size of pooling is 7/7 here. Then we have added Flatten to the layer and a dense layer using 128 neurons and our activation layer is 'relu' here. Relu is used for non-linear model like ours. To avoid the over-fitting, we have used Dropout. Finally, we have created our head-Model with two layers (with mask and without mask) and softmax as our activation value as it is the best fit for binary models. Now we called the model function. We have frozen the base-Model by using a 'for' loop to prevent it from running in the training because we have just used base-Model instead of CNN.

```
# compile our model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
              metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)
```

Figure 17: Compiling Head & Base Model and Fitting the Mode

When both the models are ready, we needed to compile the models by giving them our initial learning rate, ADAM optimizer (a go-to optimizer) and tracking the accuracy metrics. Then we needed to fit our model. For this, we have flown the ImageDataGenerator which we have previously used to get more training data to train the images.

```
# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

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

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
                          target_names=lb.classes_))

# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")
```

Figure 18: Making Prediction on the Testing Set

After compiling the model and training the head, we needed to evaluate the network to make prediction to test the set. We have used 'model.predict' function to evaluate the network. We also needed to find the index for every image. For that, we have used 'np.argmax' function. Then we created a function to get the classification report and to save the model file into the disk.

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

Figure 19: Developing Codes to Plot Model

Now, we have created a function to plot the accuracy and loss of our model by using MATPLOTLIB library. Following is the picture of the plotting the accuracy and loss of the model file:

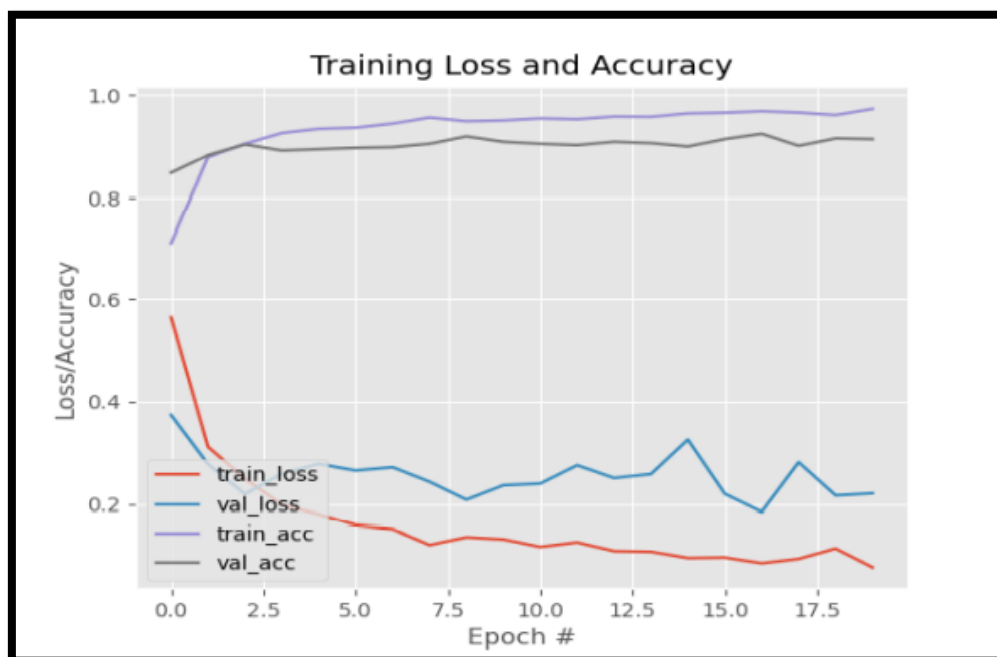


Figure 20: Plot (Training Accuracy and Loss)

Here, we can see that the accuracy level is quite good and we got a constantly decreasing loss, which is also good for our model. Therefore, we can call it a good model.

3.4 Using models in Real-Time Camera

We have already built our mask detector model. Now, we need a face detector to use both of the detectors to use in Real-Time camera. As face detector we have used following files available into the internet.



Name	Status	Date modified	Type	Size
 deploy.prototxt	✓	10-09-2021 12:46	PROTOTXT File	28 KB
 res10_300x300_ssd_iter_140000.caffemodel	✓	10-09-2021 12:46	CAFFEMODEL File	10,417 KB

Figure 21: Face Detector Files

Using mask detector model and the face detector, we needed to create a new python file to use both the detectors in real time camera.

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

Figure 22: Importing Packages

As the previous one, we imported all the necessary packages into the python file.

```
def detect_and_predict_mask(frame, faceNet, maskNet):
    # grab the dimensions of the frame and then construct a blob
    # from it
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
                                  (104.0, 177.0, 123.0))

    # pass the blob through the network and obtain the face detections
    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)

    # initialize our list of faces, their corresponding locations,
    # and the list of predictions from our face mask network
    faces = []
    locs = []
    preds = []
```

Figure 23: Defining Frame, Face & Mask Detector

We have created variables named frame for video, face-Net for face detection and mask-Net for mask detection. Here, we defined all the variables into a function named detect_and_predict_mask.

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

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

Figure 24: Looping over the detections

Here, we created a ‘for’ loop over the detection and extracted the confidence level associated with the detection. It will filter out the weak detection by ensuring minimum confidence level. Now, it will compute the X and Y coordinates, as they are the width and height of the rectangular box roaming around the face.


```

# ensure the bounding boxes fall within the dimensions of
# the frame
(startX, startY) = (max(0, startX), max(0, startY))
(endX, endY) = (min(w - 1, endX), min(h - 1, endY))

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

# add the face and bounding boxes to their respective
# lists
faces.append(face)
locs.append((startX, startY, endX, endY))

```

Figure 25: Ensuring the Dimension of the Frame to Bound Properly

Here, we ensured that the coordinates of the box will be into the frame and detect the face properly. Also, we have extracted the face ROI and convert it from BGR to RGB channel. The box size will be 224*224 pixels. Then we appended the face and the box into their respective lists.

```

# only make a predictions if at least one face was detected
if len(faces) > 0:
    # for faster inference we'll make batch predictions on *all*
    # faces at the same time rather than one-by-one predictions
    # in the above `for` loop
    faces = np.array(faces, dtype="float32")
    preds = maskNet.predict(faces, batch_size=32)

# return a 2-tuple of the face locations and their corresponding
# locations
return (locs, preds)

```

Figure 26: Making Predictions

We returned back the locs and preds from here. Locs is the x,y coordinates of the rectangular box roaming around the face and preds is the prediction of the person in the frame is either with mask or without mask. It will be green and show the percentage of wearing the mask.


```
# load our serialized face detector model from disk
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
maskNet = load_model("mask_detector.model")

# initialize the video stream
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
```

Figure 27: Loading the Face and Mask Detector Model

Here, we have loaded the face detector files and the mask detector model we previously created using python coding. We gave them a path and saved the face detector in 'faceNet' variable and mask detector in 'maskNet' variable. We created the 'faceNet' variable using 'cv2.dnn.readNet' function. And then we loaded the mask detector using 'load_model' function. Then we printed, "[INFO] starting video stream..." to understand what is going on. By using 'VideoStream' function, we loaded our camera. Here, notice that the 'src=0'. It indicates that we are using one camera only. If anyone needs to use more cameras, he/she can just input the number of camera into the src.

```
# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)

    # detect faces in the frame and determine if they are wearing a
    # face mask or not
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
```

Figure 28: Looping over the Frames from the Video Stream

Here using while loop, we made a machine to read the frame. Here, every frame indicates to an image. Multiple frames jointly make a video stream. Here, we also have set the frame width as 400 pixels. Anyone can change it according to his/her choice. Also, we made the locs and preds a tuple and added the value of detect_and_predict_mask function.

```

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

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

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

    # display the label and bounding box rectangle on the output
    # frame
    cv2.putText(frame, label, (startX, startY - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
    cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

```

Figure 29: Looping over the Detected Face Locations and Their Corresponding

Here, we unpacked the tuple to get the coordinates of the X and Y. Here, StartX = X1, StartY = Y1, EndX = X2 and EndY = Y2. It will create the rectangular box roaming around the face as well as the prediction will be with_mask or without_mask. Then, we created a label for the prediction. If the person is wearing mask it will show 'Mask' and if the person is not wearing mask it will show 'No Mask'. Here, we set the colour of the rectangular box for mask as Green (0, 255, 0). For without mask, it will be Red (0, 0, 255). Then, we displayed the label using format string. It will show the maximum possible percentage for mask or without mask. Here maximum prediction for mask and No Mask will be above 90%. It will take up to 10% of error prediction. For not wearing mask properly, it will show different values also. Then, we have displayed font name, coordinates, size and colour as well as the coordinates and coloured rectangular box.

```

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

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

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()

```

Figure 30: Showing the Output Frame & Cleaning Up the Interface

And at last, we have shown the output of the frame and set the 'q' key to quit from the loop. Finally, we have set the code to destroy all the windows and stop all the processes regarding video streaming.

4. Discussion

4.1 Conclusion

Covid-19 pandemic causes so much death and responsible for global health crisis. The world is still struggling to prevent the situation and get out of the global crisis. Every country is trying in its own way. However, this situation is so difficult to control. In this circumstance, prototype like this might be a blessing for many countries. Using traditional machine learning methods, this type of project can be developed in a larger way. The focus of this project was to create a prototype of a Face-Mask Detector that might help us to develop a real life project based on our project. We tried to achieve the highest accuracy possible in the model. By this project, we can understand things related to Machine Learning and Artificial Intelligence. Also, we can learn about the python tools to develop such project. The main goal was to learn deeply about how deep learning works. In addition, this project might help other researchers who might work on related project. In the end, we can say that the goal of making this project is fully utilized and achieved.

4.2 Suggestions for Future Research and Recommendations

We faced some limitations completing this project. From these drawbacks, future researchers can learn a lot. In addition, we came up with some recommendations for future researchers from our experience completing this project.

- If possible, researchers could use a larger dataset to get more accurate result.
- Learn deeply about the python programming language and its usage.
- Learn more and more about deep learning, a sub area of machine learning.
- Increase knowledge about convolutional neural network, will help a lot to get higher accuracy model.
- Use ConvNet instead of MobileNet if possible to achieve better result.

Chapter 5: References

- Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). *Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET). IEEE., 1-6.*
- Charniak, E. (1985). Introduction to artificial intelligence. India: Pearson Education India.
- Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, 187, 27-48.
- Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Sciencemag*, 255-259.
- Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. . *Science*, 349(6245), 255-260.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. . *nature*, 521(7553), 436-444.
- Loey, M., Manogaran, G., Taha, M. H., & Khalifa, N. E. (2020). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. . *Measurement*, 167, 108288.
- O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.
- Pan, H., Pang, Z., Wang, Y., Wang, Y., & Chen, L. (2020). A new image recognition and classification method combining transfer learning algorithm and mobilenet model for welding defects. *IEEE Access*, 8,, 119951-119960.
- Pan, H., Pang, Z., Wang, Y., Wang, Y., & Chen, L. (2020). A new image recognition and classification method combining transfer learning algorithm and mobilenet model for welding defects. *EEE Access*, 8, 119951-119960.
- S., B. (2020, July 9). balajisrinivas. Retrieved from Github: <https://github.com/balajisrinivas/Face-Mask-Detection>
- Sanner, M. F. (1999). Python: a programming language for software integration and development. *J Mol Graph Model*, 17(1), 57-61.
- Sung, K. K., & Poggio, T. (1998). Example-based learning for view-based human face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1), 39- 51.
- van, R. G., & de Boer, J. (1991). Interactively testing remote servers using the Python programming language. *CWi Quarterly*, 4(4), 283-303.