2021

UNITED INTERNATIONAL UNIVERSITY

QUEST FOR EXCELLENCE



REPORT ON

Face Mask Detection Using Python



UNITED INTERNATIONAL UNIVERSITY

A Project Report on

Face Mask Detection Using Python

Supervised By

Ahmed Imran Kabir

Adjunct Lecturer

School of Business and Economics

United International University

Prepared By

Sriman Mitra

Id- 111 161 294

Date of Submission

January 18, 2021

Letter of Transmittal

January 18, 2021

Ahmed Imran Kabir

Adjunct Lecturer

School of Business and Economics

United International University

Subject: Submission of a report on “Face Mask Detection Using Python.”

Dear Sir,

It is a pleasure for me to submit a report on Face Mask Detection Using Python. This report contains two python files accordingly for training the model and video streaming. A model was first trained and using this model video streaming file was developed.

I sincerely believe that this project helped me to quest a new experience and will help enrich my adaptability quality in the long run when I will involve myself in practical field. I am grateful for your valuable advices and great cooperation. I tried my best to go deep into the matters and make full use of my capabilities in making this formal report meaningful, though; there may be some mistakes and shortcomings. I will be pleased to answer any kind of query you think necessary.

I am requesting you to approve this report. I hope that this report based on artificial intelligence will satisfy you. For any of your further queries I would be at your disposal at your convenience. Thank you.

Sincerely Yours,

Sriman Mitra-111 161 294



Signature & Date

Certification of Similarity Index

Declaration of the Student

I the undersigned solemnly declare that the project report ‘FACE MASK DETECTION USING PYTHON’ is based on my own work. It was carried out during the period of my study under the supervision of ‘**Ahmed Imran Kabir’** Sir.

I assert that the findings and conclusions drawn on this project are an outcome of my word. I further certify that:

i. The project report was completely done by me under the supervision of my supervisor.
ii. The work has not yet been submitted for any other degree/ diploma/ certificate in this university and any other university.
iii. I have followed the guidelines provided by the University for preparing this report.
iv. Whenever I have used any materials or taken any help from other sources, I have given credit to them and included the details in the references.

NAME: Sriman Mitra

ID: 111 161 294

Acknowledgement

Firstly, I am thankful to Almighty who gave me strength and made me able to complete this report. At the time thanks to all the people who helped me a lot in this pandemic situation. Without these helps, I have probably not finished this report on time, so I would like to express my gratitude to all of you.

Secondly, I would like to express my gratitude to my honorable project supervisor **Ahmed Imran Kabir** for his valuable advice, encouragement, guidance and supervision and for his support in completing this project. I am grateful to him for supporting me and giving me the proper direction.

In addition, I am also thankful to those who were both directly and indirectly related to the report work, provided me with crucial information that helps to complete this report. Heartfelt appreciation is expressed to them for their valuable time and cooperation. I would like to express my profound towards my parents for their co-operation and encouragement, which helped me a lot to complete this project.

Finally, I would like to give a special thanks to the United International University for giving me the opportunity to work on the project by which I learned a lot of thing about research work, data analysis tools and methods and so on. By making this type of report, I was able to enrich my knowledge as well.

Executive Summary

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, I 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, I have used Python to train my model. To train the model, several packages of python was needed. Also, after training the model, I have created a video streaming model to detect faces. These processes will be discussed in depth in this report.

Table of Contents

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*

1.4 Theoretical Framework *2*

1.5 Motivation of the Study *2*

1.6 Limitations of the Study *3*

1.7 Definition of the Key Terms *3*

1.7.1 Python Programming Language *3*

1.7.2 Deep Learning *3*

1.7.3 Convolutional Neural Network *4*

1.7.4 Face Mask Detector *4*

1.8 Organization of the Chapters *5*

Chapter 2: Review of the Literature **6**

Introduction *6*

Python Programming Language *6*

Artificial Intelligence *6*

Machine Learning *6*

Deep Learning *7*

Face Detection *7*

Convolutional Neural Networks *8*

Chapter 3: Research Methods **9**

3.1 Introduction *9*

3.2 Datasets and Data Types *9*

3.3 Research Design *11*

3.4 Data Analysis Plan *11*

Chapter 4: Research Analysis and Findings **13**

4.1 Installing the dependencies *13*

4.2 Data Preprocessing *17*

4.2.1 Mask Detector Model *17*

4.3 Training Model *20*

4.4 Using models in real time camera *26*

Chapter 5: Discussion **32**
5.1 Conclusion ***32***
5.2 Suggestions for future research and recommendations ***32***
References **33**

List of Figures

FIGURE 1: PEOPLE UNWILLING TO WEAR MASK 2
FIGURE 2: CONVOLUTIONAL NEURAL NETWORK 4
FIGURE 3: ORGANIZATION OF THE CHAPTERS 5
FIGURE 4: WITH MASK DATASET 10
FIGURE 5: WITHOUT MASK DATASET 10
FIGURE 6: RESEARCH DESIGN 11
FIGURE 7: FACE MASK DETECTOR (WITH MASK) 12
FIGURE 8: FACE MASK DETECTOR (WITHOUT MASK) 12
FIGURE 9: CREATING AND ACTIVATING VIRTUAL ENVIRONMENT 13
FIGURE 10: INSTALLING PACKAGES (1) 14
FIGURE 12: REQUIREMENTS 15
FIGURE 13: INSTALLING PACKAGES ALL TOGETHER (1) 15
FIGURE 14: INSTALLING PACKAGES ALL TOGETHER (2) 16
FIGURE 15: IMPORTING PACKAGES 17
FIGURE 16: SETTING DIRECTORY AND CATEGORIES AND LOOPING THEM 18
FIGURE 17: ENCODING LABELS 19
FIGURE 18: MOBILENET NEURAL NETWORK 20
FIGURE 19: SETTING INITIAL LEARNING RATE, EPOCHS AND BATCH SIZE 20
FIGURE 20: DATA AUGMENTATION AND CREATING BASE MODEL 21
FIGURE 21: CREATING HEAD MODEL AND LOOPING BASE MODEL 22
FIGURE 22: COMPILING HEAD & BASE MODEL AND FITTING THE MODEL 23
FIGURE 23: MAKING PREDICTION ON THE TESTING SET 24
FIGURE 24: DEVELOPING CODES TO PLOT MODEL 25
FIGURE 25: PLOT (TRAINING ACCURACY AND LOSS) 25
FIGURE 26: FACE DETECTOR FILES 26
FIGURE 27: IMPORTING PACKAGES 26
FIGURE 28: DEFINING FRAME, FACE & MASK DETECTOR 27
FIGURE 28: LOOPING OVER THE DETECTIONS 27
FIGURE 29: ENSURING THE DIMENSION OF THE FRAME TO BOUND PROPERLY 28
FIGURE 30: MAKING PREDICTIONS 28
FIGURE 32: LOOPING OVER THE FRAMES FROM THE VIDEO STREAM 29
FIGURE 33: LOOPING OVER THE DETECTED FACE LOCATIONS AND THEIR CORRESPONDING 30
FIGURE 34: SHOWING THE OUTPUT FRAME & CLEANING UP THE INTERFACE 31

List of Abbreviation

Name	Abbreviation
AI	Artificial Intelligence
Venv	Virtual Environment
Np	Numpy
DNN	Deep Neural Network
CNN	Convolutional Neural Network
ConvNet	Convolutional Neural Network
MNN	Mobile Neural Network
MobileNet	Mobile Neural Network
ANN	Artificial Neural Network
Listdir	List Directory
Plt	Plot
INIT_LR	Initial Learning Rate
BS	Batch Size

Chapter 1: Introduction

The awareness of wearing facemask is increasing day by day due to the Covid pandemic. Wearing mask is so important for people to prevent rapidly rising Covid cases. Before, we saw that people generally used mask to prevent air pollution or allergic infection. But now it is not optional it became essential to wear mask. But in a densely populated country like ours, it is much difficult to aware people to wear masks. To maintain this properly we might take help of AI. Because it makes task easier, decrease costs as well as so effective.

1.1 Background of the Study

This project is basically based on Artificial Intelligence (AI) with taking the help of deep learning method and machine learning. To complete the project, I first needed to teach machine. However, machine cannot understand human languages and emotions. Therefore, I needed to use a programming language. Here, I have used *'Python Programming Language'* to complete my project.

1.2 Statement of the problem

Not wearing mask is a huge problem and the barrier to prevent Covid 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 clash of many countries in the previous year. The world economy is still not stable. In many countries Covid cases is 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 in a regular basis.

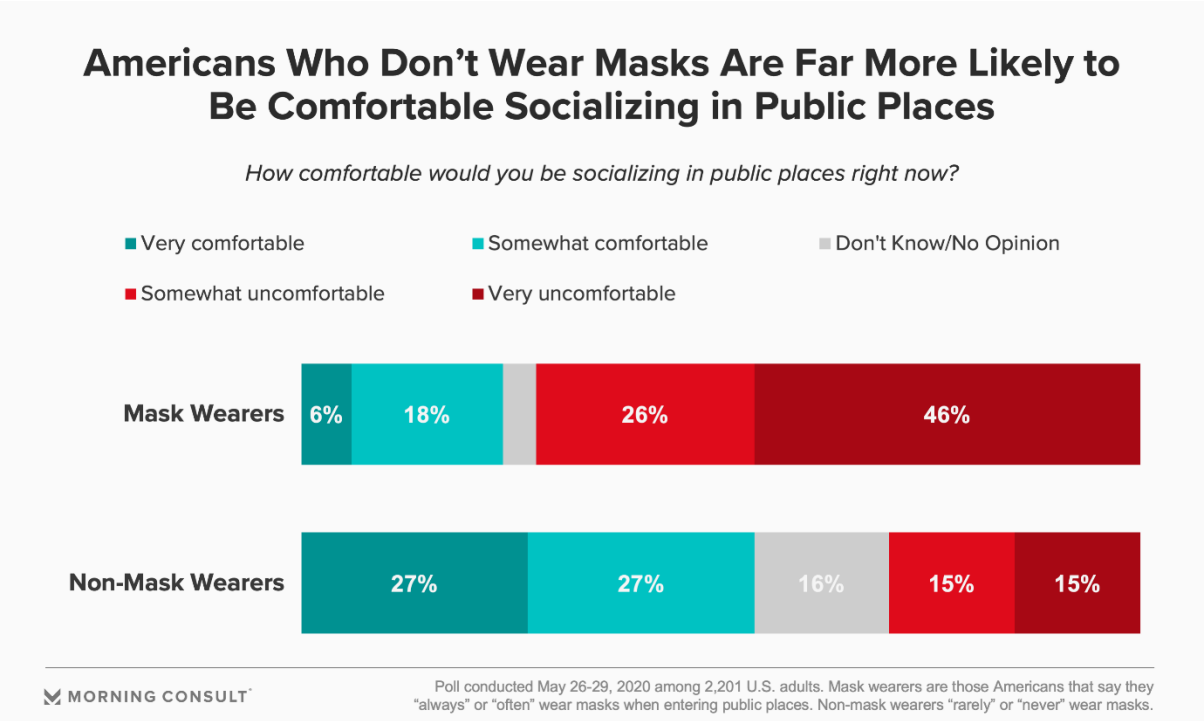
1.3 Objectives of the Study

- 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 Theoretical Framework

I worked on this project to make a modern solution to monitor that people are following the safety precautions. This project is very much related to the current pandemic situation. A research shows that 45% of all the Americans refuse to wear mask. Following is a research that shows how much people are unwilling to wear mask.

Figure 1: People unwilling to wear mask



Similar situation is also applicable for many countries. It is high time to think about a modern solution of this problem. A face mask detection model might be a solution of this problem. By using this government can remotely monitor people. Therefore, it will make task easier for authorities to control the situation.

1.5 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 behavior towards the pandemic.
- Maximize the effectiveness of the process.

1.6 Limitations of the Study

- It is not possible for a single person to work on a huge project.
- 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

1.7 Definition of the Key Terms

1.7.1 Python Programming Language

Python is a high-level programming language. It is one of the most popular programming language because of the following

i. Ease of use/ User friendliness
ii. Fast to develop
iii. Open source language
iv. Has a great community
v. Vast amount of libraries and packages
vi. Probably the best for making prototypes
vii. Good for data analysis and visualization also for AI based deep learning

Instead of having such advantages, it has also some limitations as below

i. Case Sensitive
ii. Slower than other languages
iii. Not so useful in mobile environment

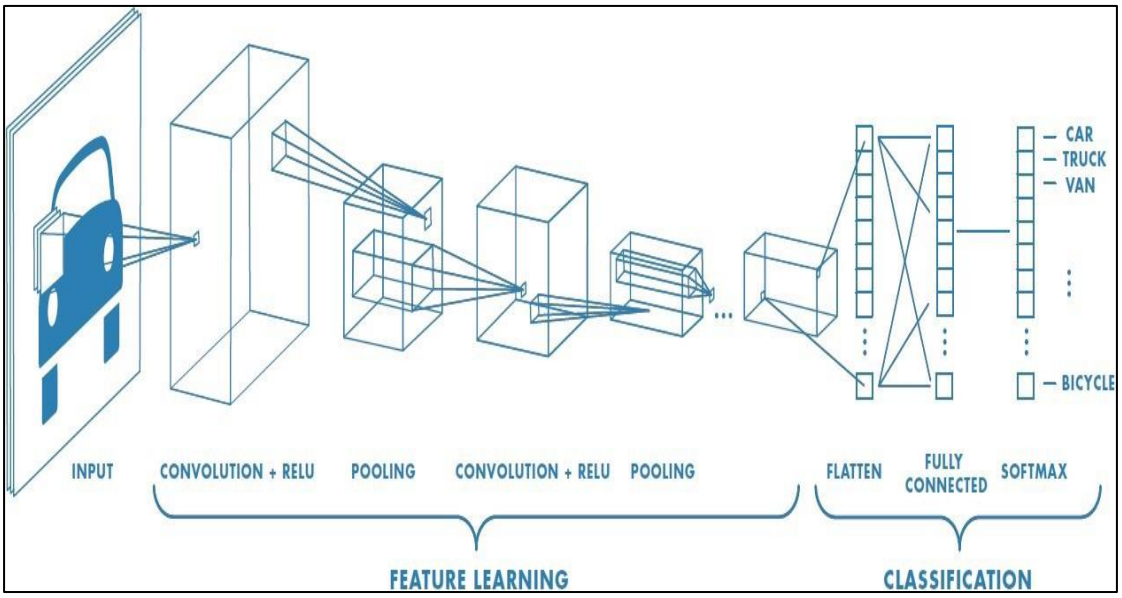
1.7.2 Deep Learning

It is a part of AI. Using deep learning model, we can make machine work for us. Machine can work like a human but more effectively and more errorless. Because machine can work without any rest. A machine goes through many trial and error processes to complete a model. So, there are less probability to have error from machine. By using deep learning, we can make machine imitating human activities.

1.7.3 Convolutional Neural Network

CNN or ConvNet is a class of deep learning or deep neural network. It is used for analyzing visual imagery. It is similar to a traditional ANN. It is self-optimized through learning. It is used for pattern recognition of images. As, I built an face mask detection model, CNN helped me to detect special pattern of mask from the dataset I have used.

Figure 2: Convolutional Neural Network

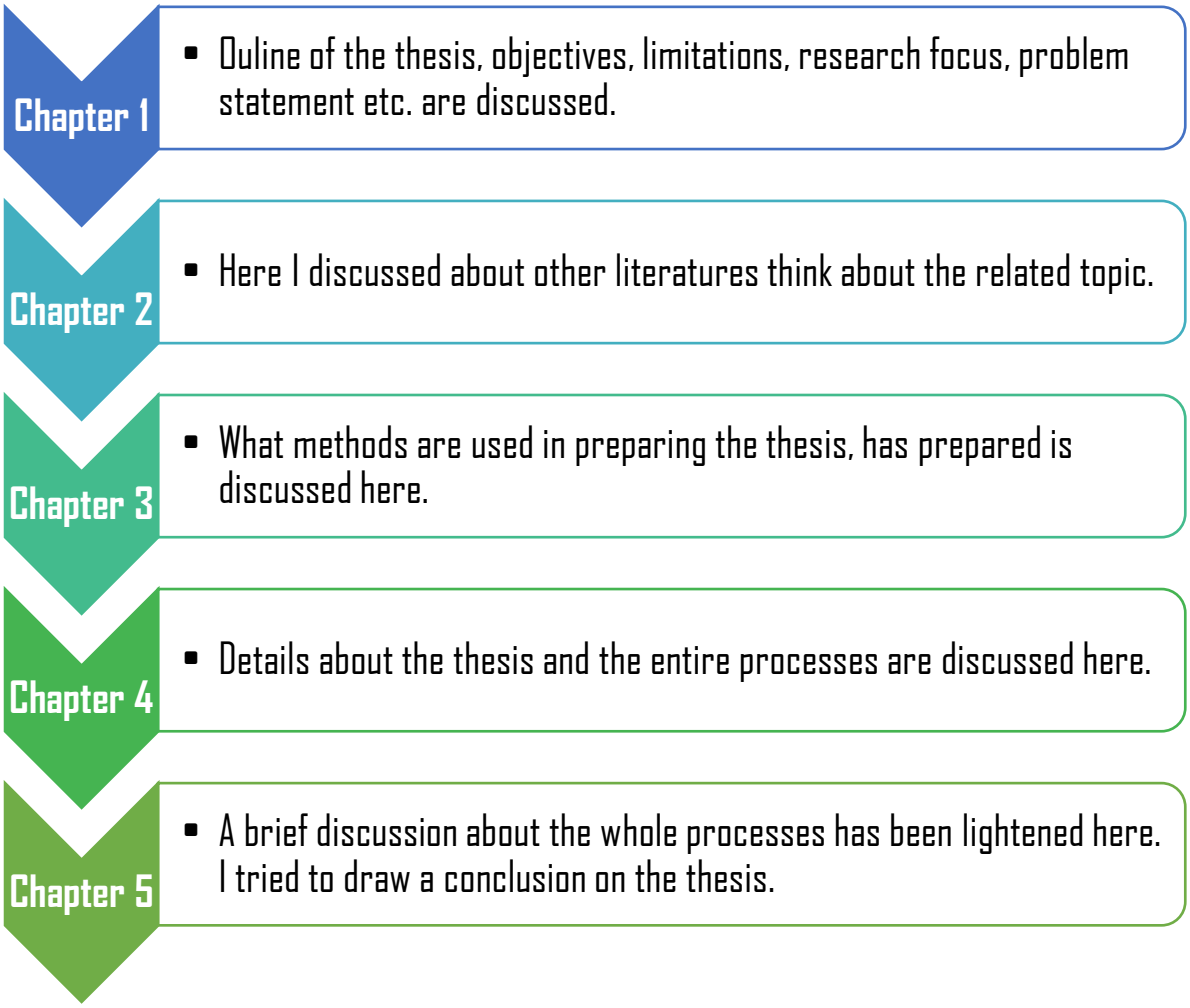


1.7.4 Face Mask Detector

To develop a face mask detector, two developing two individual detector is necessary. Firstly, we need to develop a mask detector model individually and a face detector model individually. Then we need to combine both of them in a separate file, which will finally create a face mask detection prototype. Luckily, a vast amount of face detector model is available online. So, we only need to a mask detector model.

1.8 Organization of the Chapters

Figure 3: Organization of the chapters



Chapter 2: Review of the Literature

Introduction

There are a vast amount of studies available online that discussed about the Artificial Intelligence and related fields. All the studies shows the factors related to AI like deep learning, machine learning, CNN, ANN, MobileNet, programming languages and so on. I will try to define all the factors related to my thesis that other literature discussed about.

Python Programming Language

Michel F. Sanner mentioned that python is an interpreted, object-oriented and interactive as well as simple yet powerful general purpose programming language. He also told that various type of high-level data types are provided by python. (Sanner, 1999)

Numbers, tuples, lists, dictionaries and strings are the main type of objects in python. Though there are some other objects can be found in python also. In addition, python has various kind of statements that are simple in nature. (van & de Boer, 1991)

Artificial Intelligence

Researchers are trying to make machine think by itself with the help of AI. AI is used for robotics. It is one of the sub area of AI. AI can also be used in medical field. *Eugene Charniak* mention that AI is the study of mental faculties using computational models. AI refers to provide intelligence to the machine so that it can act like human, solve problem using its own intelligence. (Charniak, 1985)

Machine Learning

Machine learning is a method of AI to choose for computer vision, controlling robot, and speech and face recognition and so on. Many AI developers consider that train a system by showing it examples is easier than doing it manually. There is a broad range of machine learning in the field of computer science. (Jordan & Mitchell, Machine learning: Trends, perspectives, and prospects, 2015)

Machine learning is powering up the modern society. Many sectors such as e-commerce, web developer, social networking and many more are getting the blessings of machine learning. It helps to identify objects from images, recognizing speech and so on. (LeCun, Bengio, & Hinton, 2015)

Deep Learning

Deep learning algorithms are a subset of machine learning algorithms. It helps to aim discovering multiple levels of distributed representation. It used for solve traditional AI problems. I has been using to learn high-level abstractions in data. Computer vision, transfer learning, natural language processing etc. are the approaches of deep learning method. (Guo, et al., 2016)

The conventional machine learning techniques had limitations. Deep learning methods resolved the problem. It advances in solving high-level problems, discovering intricate structures in high-level data, image and speech recognition and many more. (LeCun, Bengio, & Hinton, 2015)

We can apply deep learning successfully to analyze image and recognize target. For this, we can ow to the non-uniformity of the shape, position and size of welding defects. Before that, it was a complicated task to analyze and evaluate the acquired welding defects images manually. (Pan, Pang, Wang, Wang, & Chen, A new image recognition and classification method combining transfer learning algorithm and mobilenet model for welding defects., 2020)

The term Deep Learning or Deep Neural Network refers to Artificial Neural Networks (ANN) with multi layers. It has become one of the most powerful tools over the last decades. As well as it has become very popular in the literature as it is able to handle a huge amount of data. The interest in having deeper hidden layers has recently begun to surpass classical methods performance in different fields, especially in pattern recognition. (Albawi, Mohammed, & Al-Zawi, 2017, August)

Face Detection

Same spatial configuration, large components of non-rigidity and textural differences among faces make face recognition a difficult task. So, we need to train machine a lot by giving them more and more examples. It has also potential applications in human-computer interfaces and surveillance systems. (Sung & Poggio, 1998)

Convolutional Neural Networks

CNN is one of the attractive field of Artificial Neural Networks. Difficult image-driven pattern recognition tasks are solved using CNN techniques. It helps making architecture precise but simple. It is a simplified way of ANN. CNN analogous to traditional ANN in that they are comprised of neurons that self-optimize through learning. (O'Shea & Nash, 2015)

Convolutional neural network (CNN) is a part of deep neural network (DNN). In fact, it is one of the most popular among the DNNs. The name convolutional came from the mathematical linear operation between matrixes. Convolutional layer, non-linearity layer, pooling layer and fully connected layer are some layers of CNN. There are no parameters in pooling and non-linearity layers. However, parameters are present in convolutional and fully connected layers. (Albawi, Mohammed, & Al-Zawi, 2017, August)

Chapter 3: Research Methods

3.1 Introduction

This project has been completed with the help of machine learning, precisely the deep learning methods. Here I have used convolutional neural network with a slight change. The change is here I have replaced ConvNet with MobileNet as MobileNet is faster to develop and easy to use for developing prototype. Although using CNN might provide me better model, but it would become a huge task to develop such prototype. Also, it needs better devices to perform well.

3.2 Datasets and Data Types

Here, I used two sample as input to train the machine. The first set contains different images of people wearing different types of masks. Moreover, the other set contains different types of without mask images. I have tried to use different types of images to ensure getting a good model.

The data I have collected were categorical data. Later the data were transformed into numerical data by using python codes. As machine cannot process categorical data I had to transform them into numerical. The datasets I have used for the research purpose was mainly secondary data as I have collected them from outside source. The datasets were collected from the GitHub account of *'Balaji Srinivas'* which has been appreciated in the reference (S., 2020).

There are total 1915 images in the folder named 'with_mask'. Moreover, the 'without_mask' folder contains 1918 images. I believe, if the datasets could be better, would have produced much better model. Unfortunately, I had limitations of providing datasets as my device would not work properly and might have problem with a huge project. A glimpse of the datasets are shown below.

Figure 4: With mask dataset

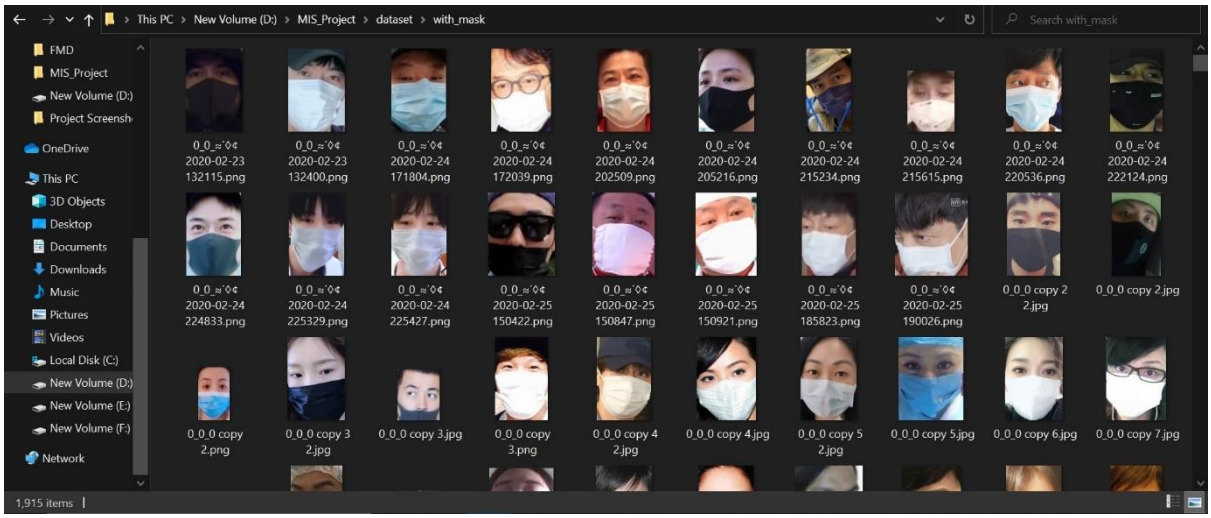
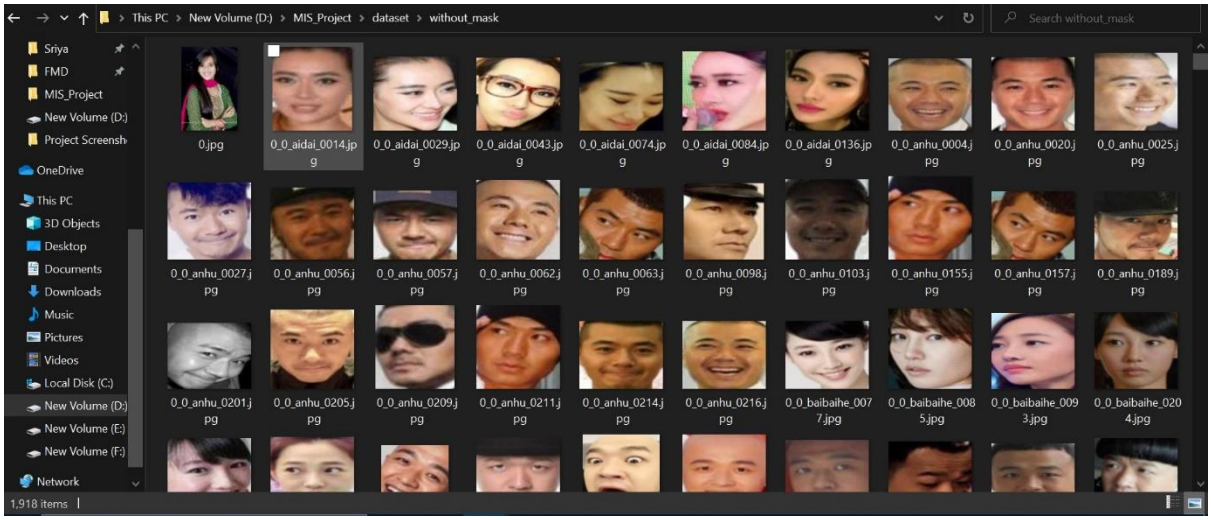


Figure 5: Without mask dataset



3.3 Research Design

This project is actually a quantitative research. Because I had to collect data, analyze 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 shows that my 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. I also can 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 6: Research Design



3.4 Data Analysis Plan

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

Figure 7: Face Mask Detector (With Mask)

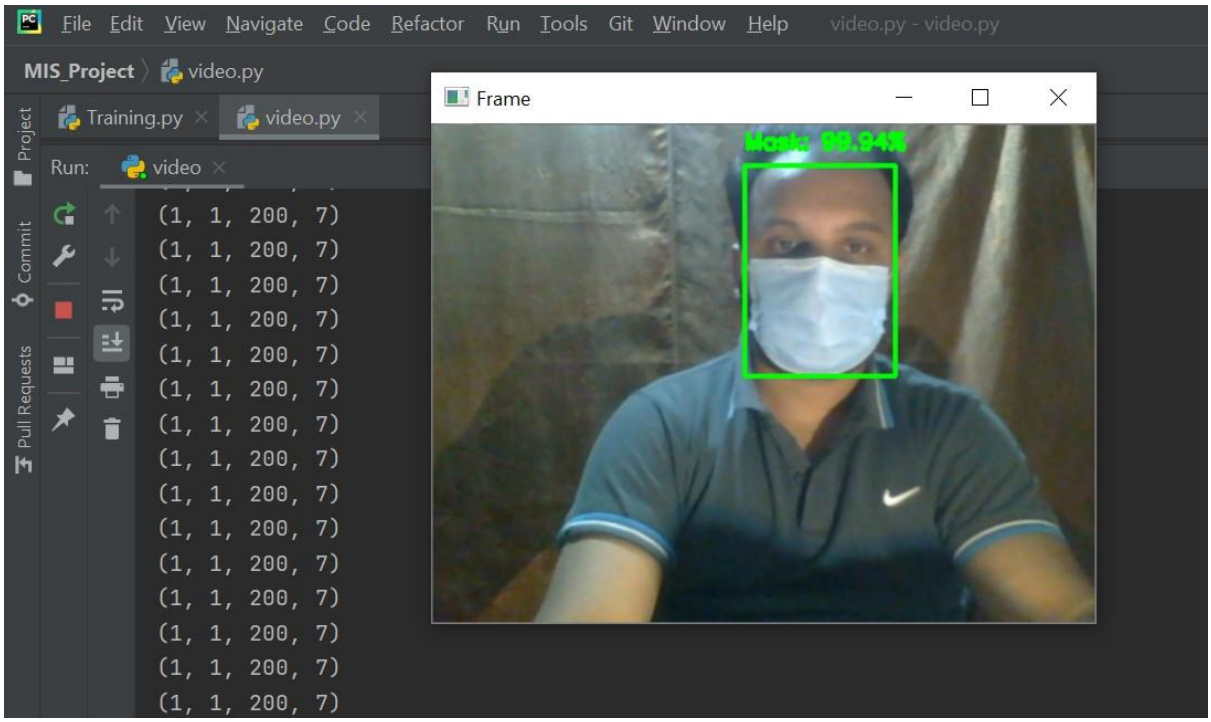
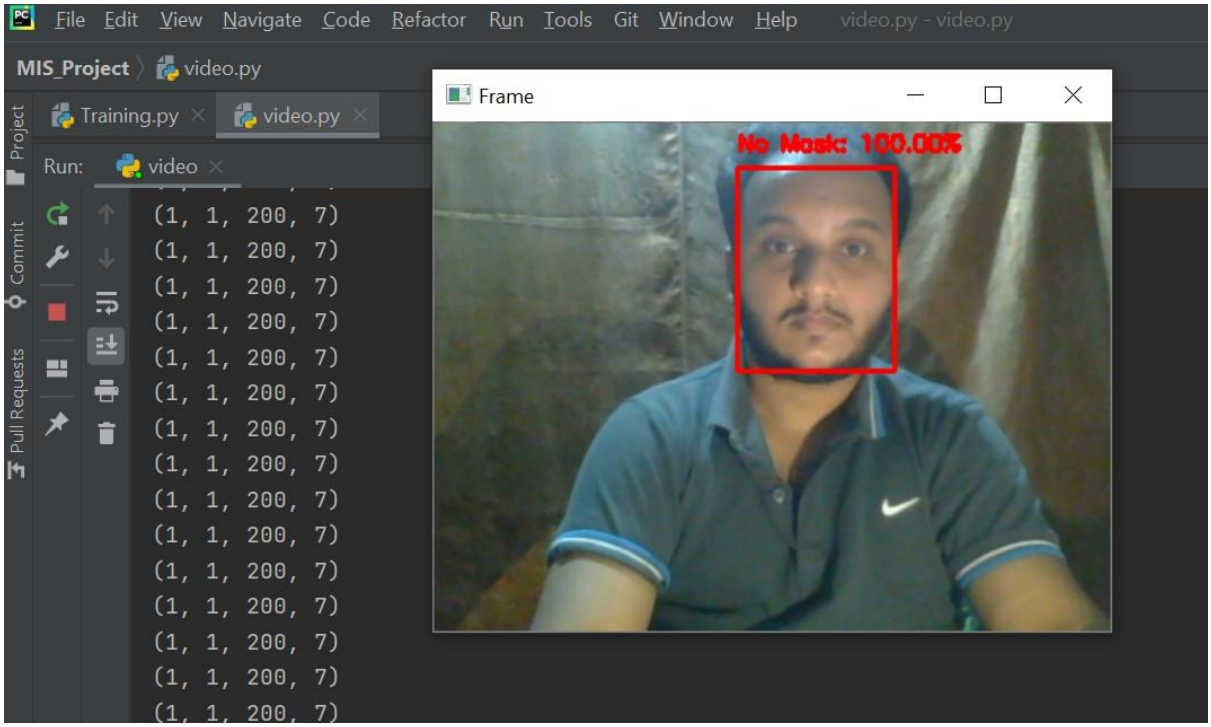


Figure 8: Face Mask Detector (Without Mask)



Chapter 4: Research Analysis and Findings

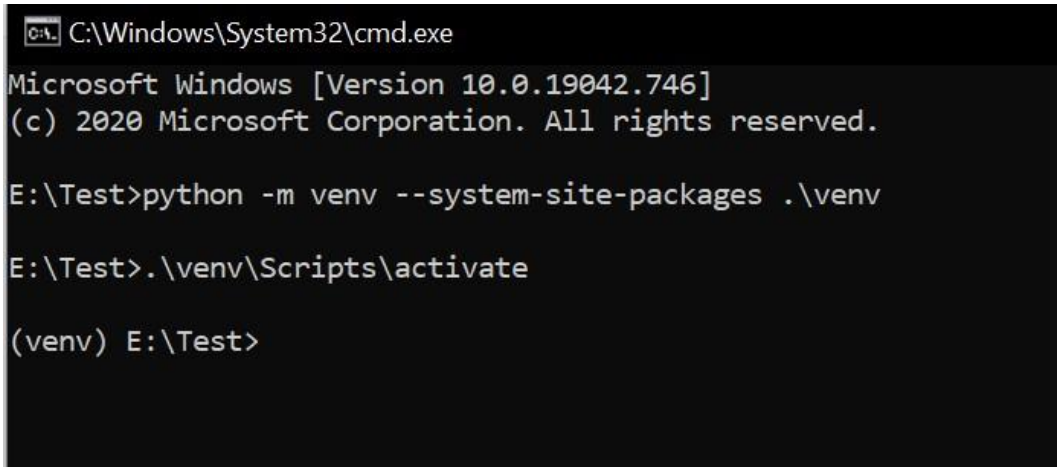
4.1 Installing the dependencies

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 two ways as follow. For installation purpose, we first need to create and activate a virtual environment to the working folder as the following picture.

Figure 9: Creating and Activating Virtual Environment



a. Individually Installation

For the individual installation, we need to run code into the venv as the following picture. The codes are as follow:

- pip install tensorflow==1.15.2
- pip install keras==2.3.1
- pip install imutils==0.5.3
- pip install numpy==1.18.2

- pip install opencv-python==4.2.0.*
- pip install matplotlib==3.2.1
- pip install scipy==1.4.1

Figure 10: Installing Packages (1)



Figure 11: Installing Packages (2)

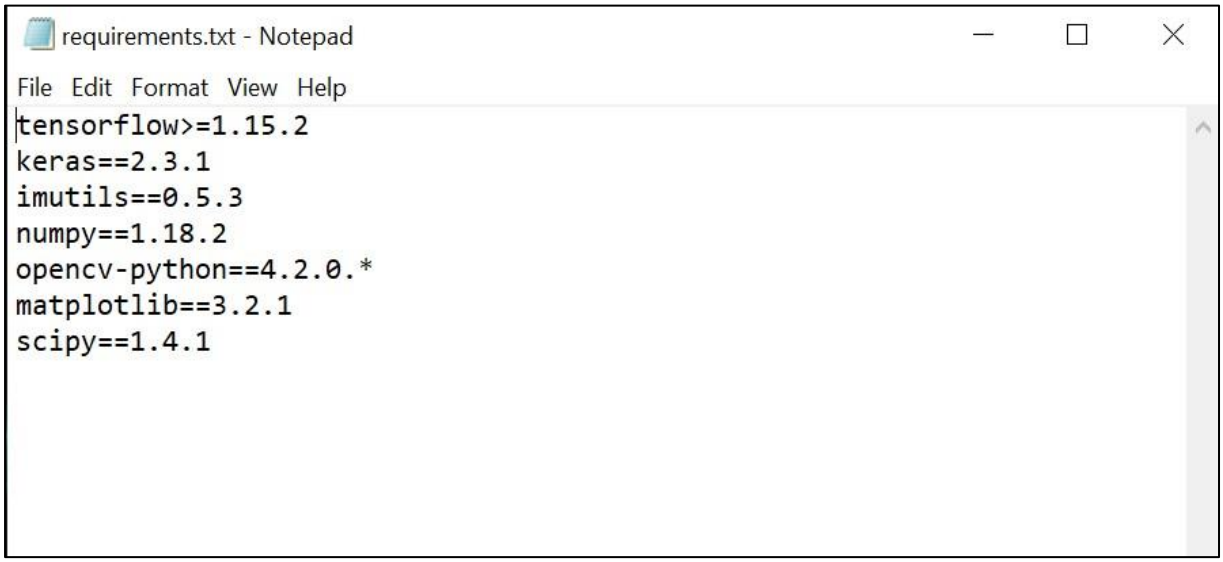


In my computer, all the packages were previously installed that's why it is showing the 'Requirement already satisfied' message. But for the newly installation it will install properly.

b. Installing All Together

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.

Figure 12: Requirements

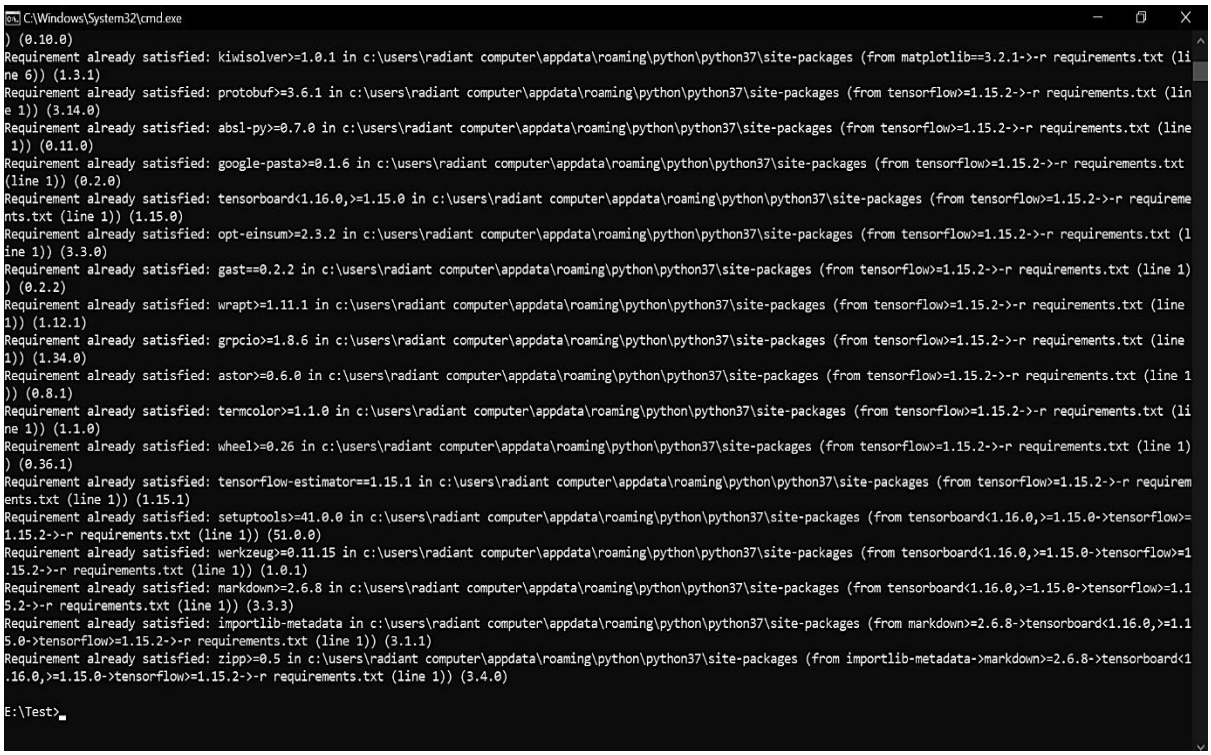


Then we need to deactivate venv and run the following code.

Figure 13: Installing Packages All Together (1)



Figure 14: Installing Packages All Together (2)



Using a txt file to install all the requirements is an easier process. However, sometime it does not work and install all the requirements. That is why we sometime needs to use the individual installation process.

4.2 Data Preprocessing

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

4.2.1 Mask Detector Model

Figure 15: Importing Packages

```
# Importing 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 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

```

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

Figure 16: Setting Directory and Categories and Looping Them

```
DIRECTORY = r"D:\MIS_Project\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)
```

Here we can see my directory is set on my D drive containing MIS_Project folder. Moreover, inside the DIRECTORY, I mentioned where my dataset folder is present. Therefore, machine can understand the dataset it need to use to run the program. Inside CATGORIES, I have mentioned the values named with_mask and without_mask. These are also the folders present in the DIRECTORY. I have already discussed more about the dataset in *'Chapter-3'.*

Here I have created two empty lists called 'data' and 'labels'. Later, I appended the entire images array into this *'data'* list. Inside the *'label'* list I appended the label of those images whether it is with mask or without mask.

After that, I have looped through the CATAGORIES by using for command. By *'os.path.join'* command I tried to first loop through the with mask and then through the without mask. Then I listed down all the images into the particular by using *'listdir'* command. Then I loaded images using the preprocessing function *'load_img'* of the package *'keras'* I have imported previously. I gave the images a target size of (224, 224) which is the height and width of the images I wanted to be. Then I converted all the images into array by using another *'keras'* function called *'img_to_array'.* I have used the *'mobiletnet'* function of the package called *'tensorflow'.* That is why I used *'preprocess_input'* function here.

After preprocessing the images successfully, I appended the images into the previously created *'data'* list and category into the *'labels'* list.

Figure 17: Encoding Labels

```
# 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)
```

Now, the images are in numerical values. Which is good for machine to understand easily. Therefore, I needed to convert the labels into numerical value also. For thet I have used *'LabelBinarizer'* function from the package *'sklearn'*. Then I converted the *'labels'* contain with_mask and without_mask values into categorical. It converted the labels into numerical value.

After converting both the lists into numerical value, I converted both the lists into *'np'* array. Because deep learning models only work in array format.

Then I used *'train_test_split'* to split my training and testing data. Here the test_size is 0.20, which indicates I 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. I have used stratify into labels which is nothing but classify the labels and the random_state indicates the set of train and test split I am getting. It actually does not matter what number is this. It does not affect much on the split. If I provide 'none' in the random_state, it will just choose a number randomly to decide the splitting of train and test indices.

4.3 Training Model

To train the model I used convolutional neural network with a slight change. To understand the change, we first need to understand convolutional neural network (CNN) model.

I have used MobileNet a version of ConvNet because MobileNet is faster than ConvNet as it uses depth-wise separable convolution. So, it uses less parameter. That is why MobileNet is less accurate too. However, in my scenario, using MobileNet did not make a much difference in outcome.

Figure 18: MobileNet Neural Network

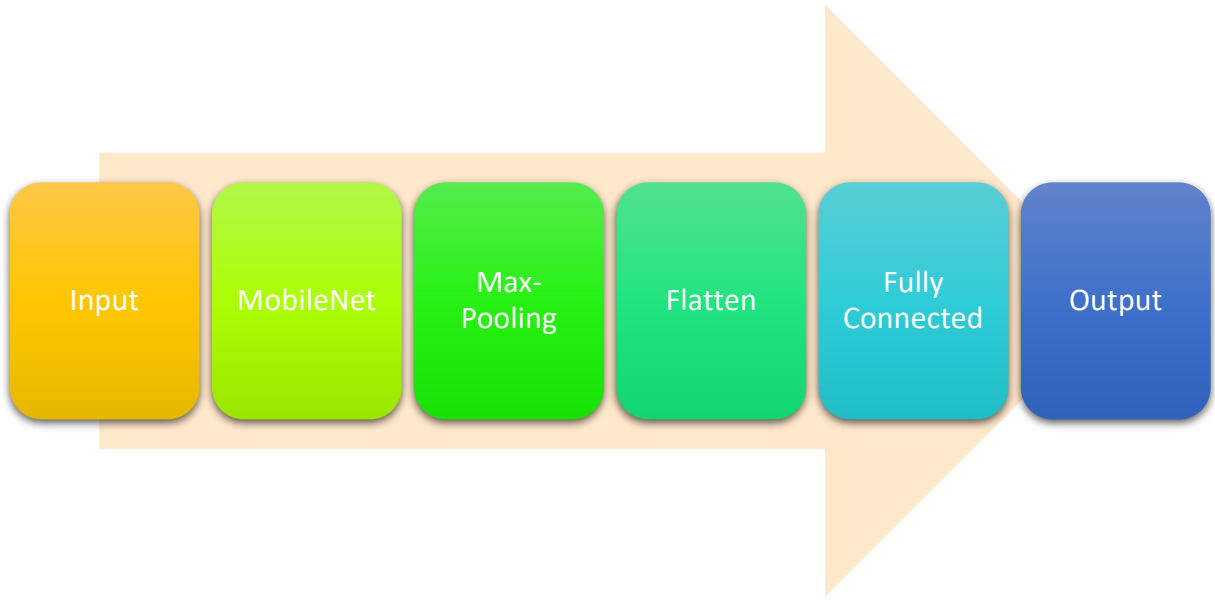


Figure 19: Setting Initial Learning Rate, Epochs and Batch Size

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

Here, I have provided INIT_LR as 1e^(-4). It is the learning rate of my project. Kepping learning rate less helps to calculate the loss properly. By this getting a better accuracy is much easier. I have also provide 20 EPOCHS and my batch size is 32.

I previously mentioned that I have used MobileNet which will generate two models. One is the MobileNet model and the output of this model will create a regular model that I have developed. These models are called as Headmodel and Basemodel in my project.

Figure 20: Data Augmentation and Creating Base Model

```
# 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)))
```

Here, I have used *'ImageDataGenerator'.* It actually create many images from one single image by changing some of the properties. It has many functionalities like shifting image, rotating image, flipping image, zooming image etc. Here I have changed some properties as shown in the figure above.

Also, I have loaded the *'MobileNetV2'* for creating the models as previously mentioned. Here, I mentioned the weights, include_top, input_tensor and shape of the inputs. By this I have created my baseModel.

Figure 21: Creating Head Model and Looping 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
```

After creating my baseModel, I created my headModel. As previously told, headModel will be created using the output of baseModel, I have passed the baseModel output into headModel. Then I have created the pooling by using *'AveragePooling2D'* function. The size of pooling is 7/7 here. Then I have added Flatten to the layer and a a Dense layer using 128 neurons and my activation layer is *'relu'* here. Relu is used for non-linear model like mine. To avoid the overfitting I have used Dropout. Finally, I have created my headModel with two layers (with mask and without mask) and softmax as my activation value as softmax is the best fit for binary models.

Now I called the model function. As previously mention baseModel is the input model and headModel is the output model. I have frozen the baseModel by using for loop. Because I have just used baseModel instead of CNN and to prevent it from running in the training.

Figure 22: Compiling Head & Base Model and Fitting the Model

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

# 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)
```

When both the models are ready, I needed to compile the models by givinhg them my initial learning rate, ADAM optimimizer (a got to optimizer) and tracking the accuracy metrics.

Then I needed to fit my model. For this, I have flown the ImageDataGenerator I have previously used to get more training data to train the images.

Figure 23: Making Prediction on the Testing Set

```
# 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")
```

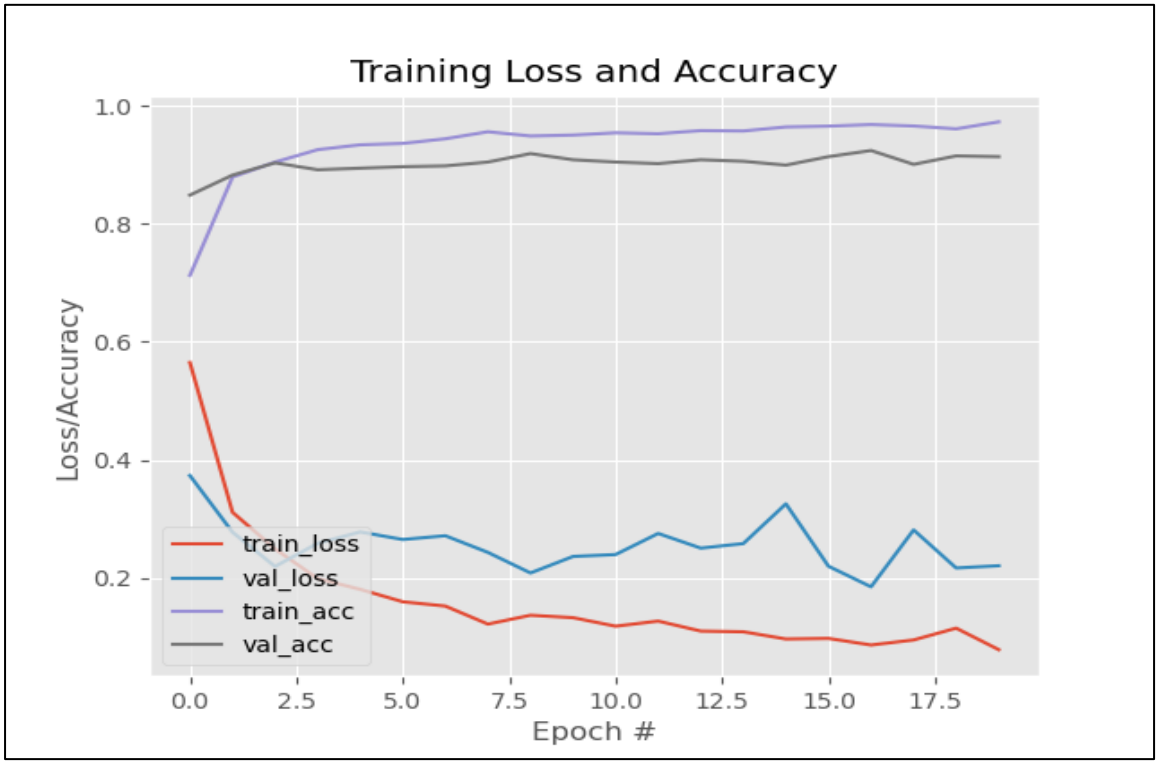
After compiling the model and training the head, I needed to evaluate the network to make prediction to test the set. I have used *'model.predict'* function to evaluate the network. I also needed to find the index for every images. For that, I have used *'np.argmax'* function. Then I created function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

Figure 24: Developing Codes to Plot Model

```
# 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["acc"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_acc"], 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")
```

Now I have created function to plot the accuracy and loss of my model by using MATPLOTLIB. Following is the picture of the plotting the accuracy and loss of the model file.

Figure 25: Plot (Training Accuracy and Loss)

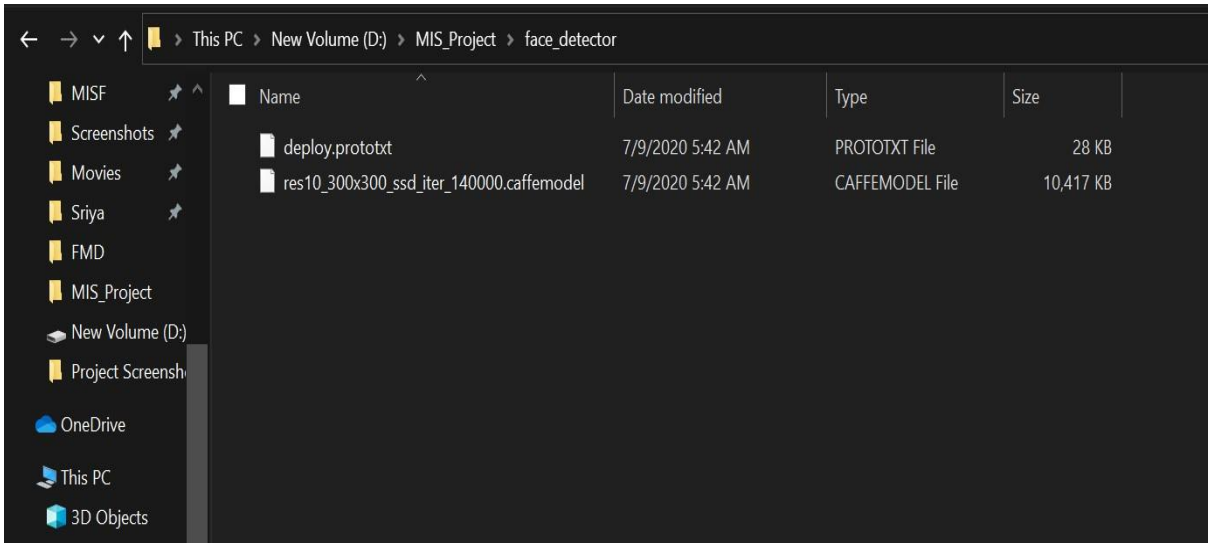


Here, we can see that the accuracy level is quite good and I got a constantly decreasing loss, which is also good for our model. Therefore, we can call it a good model.

4.4 Using models in real time camera

I have already built my mask detector model. Now, I need a face detector to use both of the detectors to use in real time camera. As face detector I have used following files available into the internet.

Figure 26: Face Detector Files



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

Figure 27: Importing Packages

```
# 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

```

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

Figure 28: Defining Frame, Face & Mask Detector

```
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 = []
```

I have created variables named frame for video, faceNet for face detection and maskNet for mask detection. Here, I defined all the varriables into a function named detect_and_predict_mask.

Figure 28: Looping over the detections

```
    # 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")
```

Here, I 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.

Figure 29: Ensuring the Dimension of the Frame to Bound Properly

```
            # 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))
```

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

Figure 30: Making Predictions

```
    # 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)
```

I retuned 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 shoe the percentage of wearing the mask.

Figure 31: Loading the Face and Mask Detector Model

```
# load our serialized face detector model from disk
prototxtPath = r"D:\MIS_Project\face_detector\deploy.prototxt"
weightsPath = r"D:\MIS_Project\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("D:\MIS_Project\mask_detector.model")

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

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

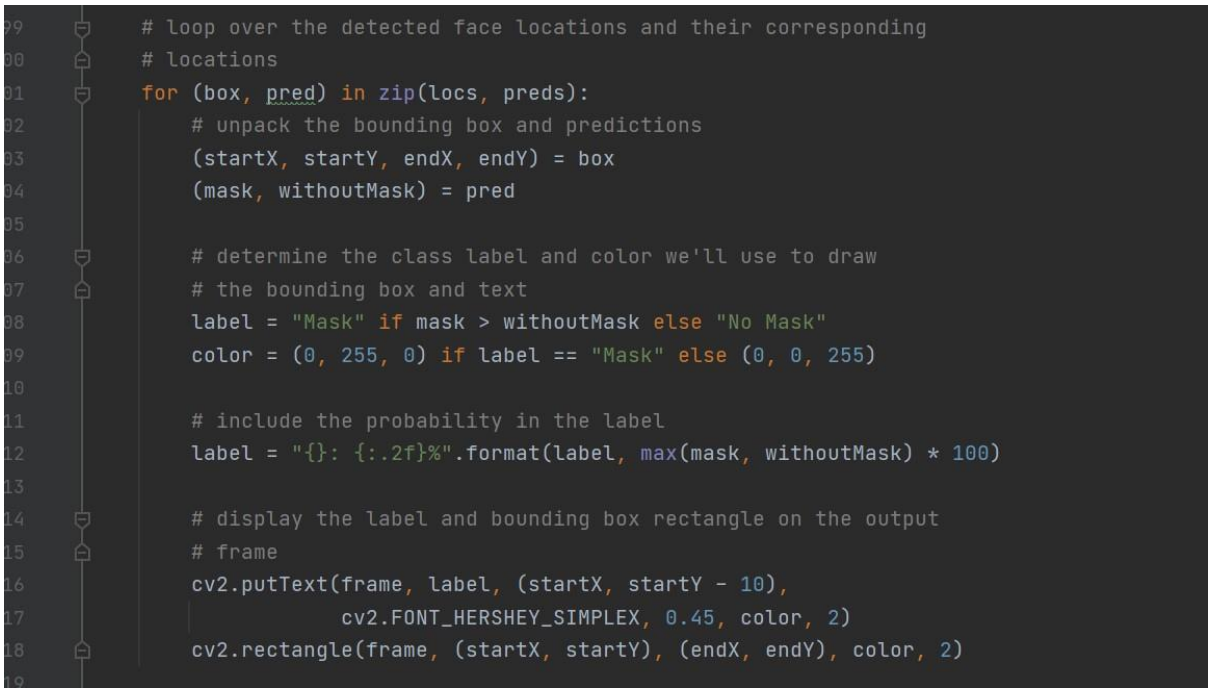
Figure 32: Looping over the Frames from the Video Stream

```
# 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)
```

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

Figure 33: Looping over the Detected Face Locations and Their Corresponding



```
    # 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)
```

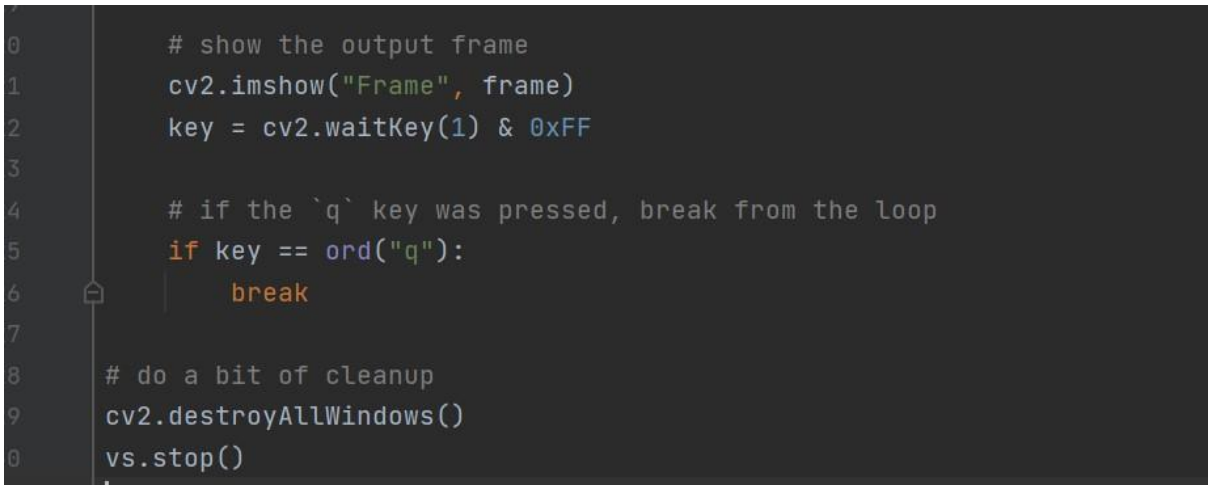
Here I unpacked the tuple to get the coordinates of the X and Y. Here StartX = X1, StartY = Y1, EndX =X2 and EndY = Y2. It will create the rectangular box roaming around the face. As well as the prediction will be mask or withoutMask.

Then I 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, I set the color of the rectangular box for mask as Green (0, 255, 0). For without mask, it will be Red (0, 0, 255). Here to understand the color, we know that machine only understand three basic color, we called as BGR. Here, 0 defines completely negative and 255 defines completely positive. Therefore, for making green, I set Blue (B) as 0, Green (G) as 255 and Red (R) as 0 and it produces completely green color. Same process has been followed for producing red color.

Then I 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 upto 10% of error prediction. For not wearing mask properly, it will show different values also.

Then I have displayed font name, coordinates, size and color as well as the coordinates and color rectangular box.

Figure 34: Showing the Output Frame & Cleaning Up the Interface



```
    # 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()
```

Here, I have shown the output of the frame and finally breaking the loop. I have set q button to break the loop. Finally, I have set code to destroy all the windows and stop all the process regarding video streaming.

Chapter 5: Discussion

5.1 Conclusion

Covid 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 to develop a real life project based on my project. I tried to achieve highest accuracy possible by me 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, I can say that the goal of making this project is fully utilized and achieved.

5.2 Suggestions for future research and recommendations

I faced some limitations completing this project. From these drawbacks, future researchers can learn a lot. In addition, I come up with some recommendations for future researchers from my experience completing this project.

- If possible, researchers could use a larger dataset to get more accurate resullt.
- 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.

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.

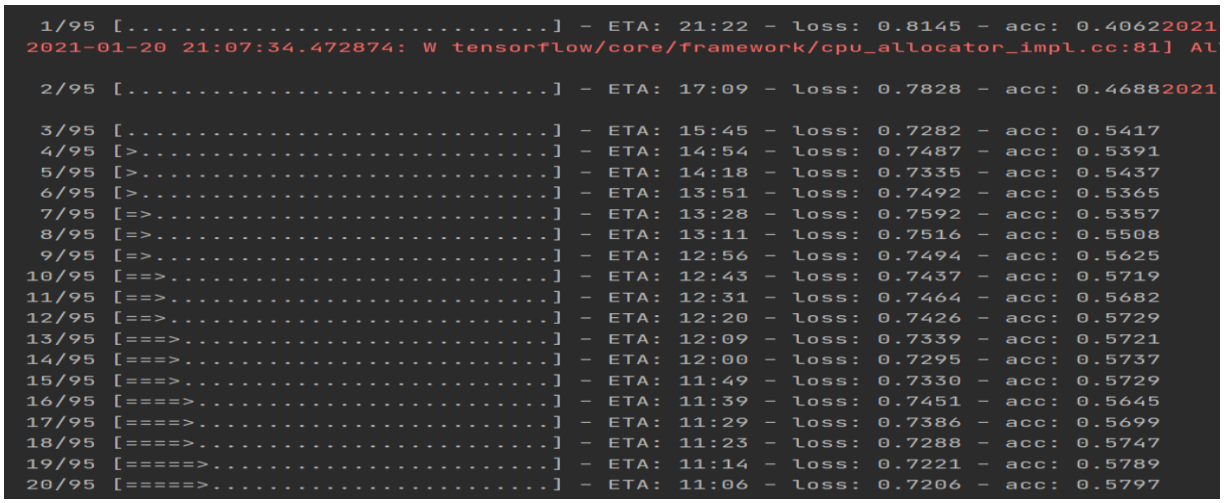
S., B. (2020, July 9). *balajisrinivas*. Retrieved from Github: https://github.com/balajisrinivas/Face-Mask-Detection

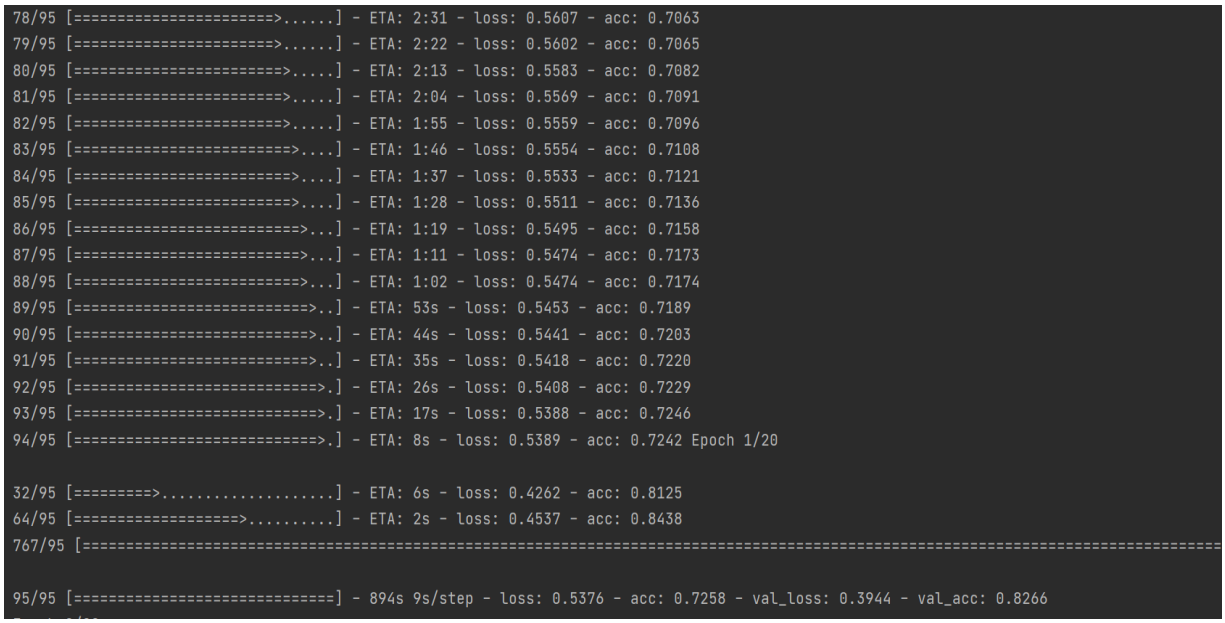
Sanner, M. F. (1999). Python: a programming language for software integration and development. *J Mol Graph Model, 17(1)*, 57-61.

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.

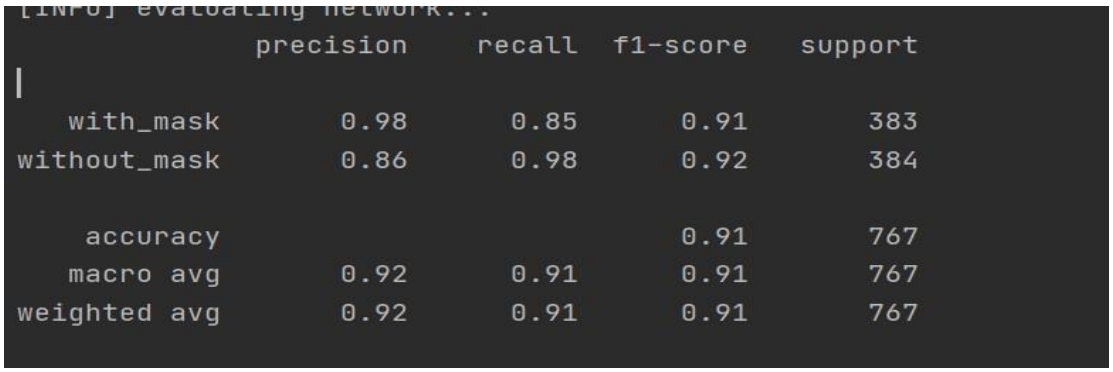
van, R. G., & de Boer, J. (1991). Interactively testing remote servers using the Python programming language. *CWi Quarterly, 4(4)*, 283-303.

Appendix





This process will continue for total 20 times. So, wait until it is completed. It might take so long. So, don't worry left it for some time.



After completing all the EPOCHS something like the above picture will appear.

The codes are following:

Mask Detector Model

```
# Importing 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

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

DIRECTORY = r"D:\MIS_Project\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)

# 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)

# 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)))

# 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

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

# 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)

# 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")

# 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["acc"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_acc"], 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")
```

Face-Mask Detector

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


def detect_and_predict_mask(frame, faceNet, maskNet):
    # 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 = []

    # 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")

            # 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))

    # 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)


# load our serialized face detector model from disk
prototxtPath = r"D:\MIS_Project\face_detector\deploy.prototxt"
weightsPath =
r"D:\MIS_Project\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("D:\MIS_Project\mask_detector.model")

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

# 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)
```

```
    # 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)

    # 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()
```