Face Mask Detection

By

Group No. 14

(Saishivam Gupta, TE-A, 48.

Suresh Gupta, TE-A, 50

Raj Mazgaonkar, TE-B, 19)

Under the Guidance of

Mrs. Shiwani Gupta
Assistant Professor

for the subject
Machine Learning

In

T.E. COMPUTER ENGINEERING

(Academic Year: 2022-23)

TCET

DEPARTMENT OF COMPUTER ENGINEERING (COMP)

(Accredited by NBA for 3 years, 3rd Cycle Accreditation w.e.f. 1st July 2019)

Choice Based Credit Grading System with Holistic Student Development [CBCGS - H 2019]

Under TCET Autonomy Scheme - 2019

CERTIFICATE

This is to certify that

Saishivam Gupta
Suresh Gupta
Raj Mazgaonkar

Have satisfactorily completed the requirements of the T.E Capstone Project Report

On

Face Mask Detection

Shiwani Gupta Dr. Harshali Patil **Subject In-charge HOD COMP**

Examiners

1. Signature: …………………. 2. Signature: …………………. Name: Name:

Date:

Place: Mumbai

TABLE of CONTENTS

LIST OF FIGURES

LIST OF REFERENCES

1.Introduction

1.1 Motivation

Amid the ongoing COVID-19 pandemic, there are no efficient face mask detection applications which are now in high demand for transportation means, densely populated areas, residential districts, large-scale manufacturers and other enterprises to ensure safety. The absence of large datasets of 'with_mask' images has made this task cumbersome and challenging.

1.2 Application

Our face mask detector doesn't use any morphed masked images dataset and the model is accurate. Owing to the use of MobileNetV2 architecture, it is computationally efficient, thus making it easier to deploy the model to embedded systems (Raspberry Pi, Google Coral, etc.).

This system can therefore be used in real-time applications which require face-mask detection for safety purposes due to the outbreak of Covid-19. This project can be integrated with embedded systems for application in airports, railway stations, offices, schools, and public places to ensure that public safety guidelines are followed.

2.Problem Definition

1.To create our face mask detector, we trained a two-class model of people wearing masks and people not wearing masks.

2.We fine-tuned MobileNetV2 on our masks/no mask dataset and obtained a classifier that is ~99% classifier.

3.We then took this face mask classifier and applied it to images by:

Detecting faces in images

Extracting each individual face

4.Our face mask detector is accurate and since we use the MobileNetV2 architecture,it’s also computationally efficient.

3.Technology used

3.1 Hardware and Software Requirement

Operating Environment

• Operating System: Windows 8

• Processor: Intel I3 or Higher

• Memory: 4GB or more

Programming Language used : Python 3.9.9 (64-Bit)

Product Functions

• Preprocessing

• Training the Images

• Face Mask Detection

• Message Passing

Coronavirus disease 2019 has affected the world seriously. One major protection method for people is to wear masks in public areas. Furthermore, many public service providers require customers to use the service only if they wear masks correctly. However, there are only a few research studies about face mask detection based on image analysis. In this paper, we propose RetinaFaceMask, which is a high-accuracy and efficient face mask detector. The proposed RetinaFaceMask is a one-stage detector, which consists of a feature pyramid network to fuse high-level semantic information with multiple feature maps, and a novel context attention module to focus on detecting face masks. it will help the system to run the overall system to prevent the spreading the Covid 19 and easy to control the mob in a cost effective way. An Iot Component will send a message to the Concerned authority that it will help the entire system to function very smoothly.

3.2 Description of libraries used

Name		**Version**
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**

Tensorflow :- TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

Keras :- Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML.

Imutils :- Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python

Numpy :- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

OpenCV :- OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source Apache 2 License.

Matplotlib :- Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

SciPy :- SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

Chapter 4. Implementation

4.1 Data Description

Step 1: Data Visualization

In the first step, let us visualize the total number of images in our dataset in both categories. We can see that there are 690 images in the 'yes' class and 686 images in the 'no' class.

The number of images with facemask labelled 'yes': 690

The number of images with facemask labelled 'no': 686

Step 2: Data Augmentation

In the next step, we augment our dataset to include more number of images for our training. In this step of data augmentation, we rotate and flip each of the images in our dataset. We see that, after data augmentation, we have a total of 2751 images with 1380 images in the ‘yes’ class and ‘1371’ images in the ‘no’ class.

Number of examples: 2751

Percentage of positive examples: 50.163576881134134%, number of pos examples: 1380

Percentage of negative examples: 49.836423118865866%, number of neg examples: 1371

4.2 Data Preparation

Step 3: Splitting the data

In this step, we split our data into the training set which will contain the images on which the CNN model will be trained and the test set with the images on which our model will be tested. In this, we take split_size =0.8, which means that 80% of the total images will go to the training set and the remaining 20% of the images will go to the test set.

The number of images with facemask in the training set labelled ‘yes’: 1104

The number of images with facemask in the test set labelled ‘yes’: 276

The number of images without facemask in the training set labelled ‘no’: 1096

The number of images without facemask in the test set labelled ‘no’: 275

After splitting, we see that the desired percentage of images has been distributed to both the training set and the test set as mentioned above.

4.3 Choice of Model

Step 4: Building the Model

In the next step, we build our Sequential CNN model with various layers such as Conv2D, MaxPooling2D, Flatten, Dropout and Dense. In the last Dense layer, we use the 'softmax' function to output a vector that gives the probability of each of the two classes.

4.4 Model Training and Validation

Step 5: Pre-Training the CNN model

After building our model, let us create the 'train_generator' and 'validation_generator' to fit them to our model in the next step. We see that there are a total of 2200 images in the training set and 551 images in the test set.

```
Found 2200 images belonging to 2 classes.
Found 551 images belonging to 2 classes.
```

Step 6: Training the CNN model

This step is the main step where we fit our images in the training set and the test set to our Sequential model we built using keras library. I have trained the model for 30 epochs (iterations). However, we can train for more number of epochs to attain higher accuracy lest there occurs over-fitting.

```
history = model.fit_generator(train_generator,
                              epochs=30,
                              validation_data=validation_generator,
                              callbacks=[checkpoint])
>>Epoch 30/30
220/220 [==============================] – 231s 1s/step – loss: 0.0368 – acc: 0.9886 – val_loss: 0.1072 – val_acc: 0.9619
```

We see that after the 30th epoch, our model has an accuracy of 98.86% with the training set and an accuracy of 96.19% with the test set. This implies that it is well trained without any over-fitting.

Step 7: Labeling the Information

After building the model, we label two probabilities for our results. [‘0’ as ‘without_mask’ and ‘1’ as ‘with_mask’]. I am also setting the boundary rectangle color using the RGB values.[‘RED’ for ‘without_mask’ and ‘GREEN’ for ‘with_mask]

```
labels_dict={0:’without_mask’,1:’with_mask’}
color_dict={0:(0,0,255),1:(0,255,0)}
```

Step 8: Importing the Face detection Program

After this, we intend to use it to detect if we are wearing a face mask using our PC’s webcam. For this, first, we need to implement face detection. In this, we are using the Haar Feature-based Cascade Classifiers to detect the facial features.

```
face_clsfr=cv2.CascadeClassifier(‘haarcascade_frontalface_default.xml’)
```

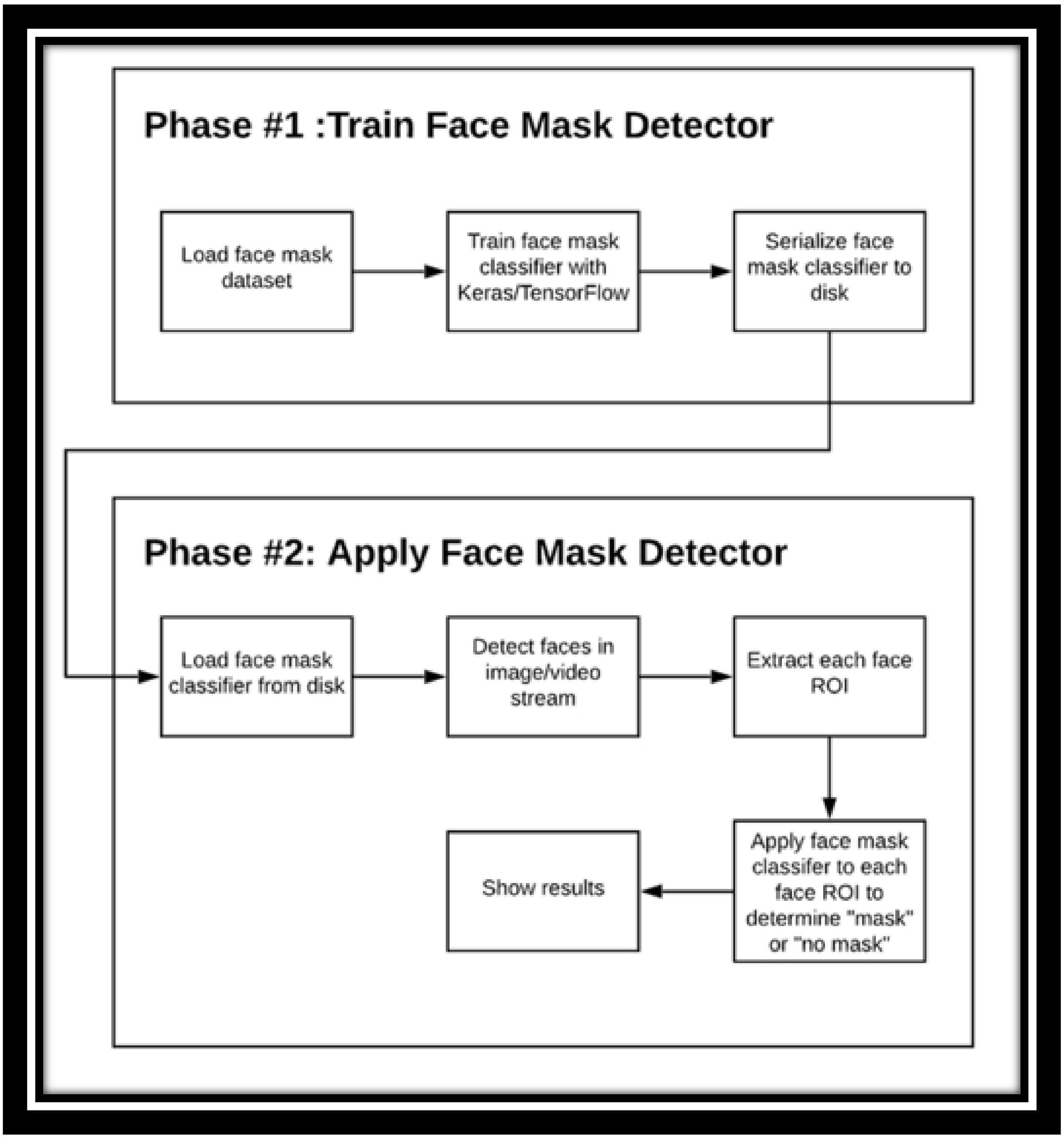
This cascade classifier is designed by OpenCV to detect the frontal face by training thousands of images. The .xml file for the same needs to be downloaded and used in detecting the face. We have uploaded the file to the GitHub repository.

Step 9: Detecting the Faces with and without Masks

In the last step, we use the OpenCV library to run an infinite loop to use our web camera in which we detect the face using the Cascade Classifier. The code webcam = cv2.VideoCapture(0) denotes the usage of webcam.

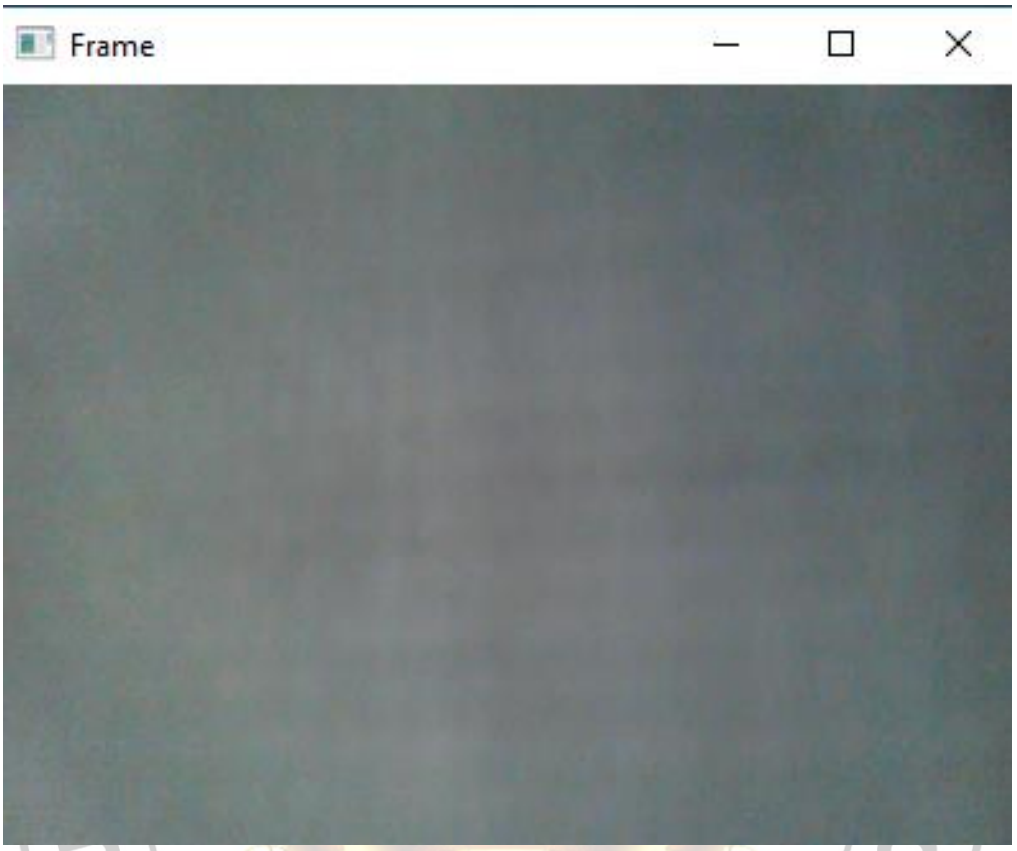
The model will predict the possibility of each of the two classes ([without_mask, with_mask]). Based on the higher probability, the label will be chosen and displayed around our faces.

Flow Chart of FaceMask Detection System


Phase #1 :Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifer to each face ROI to determine "mask" or "no mask"
Show results

Chapter 5. Result and Analysis

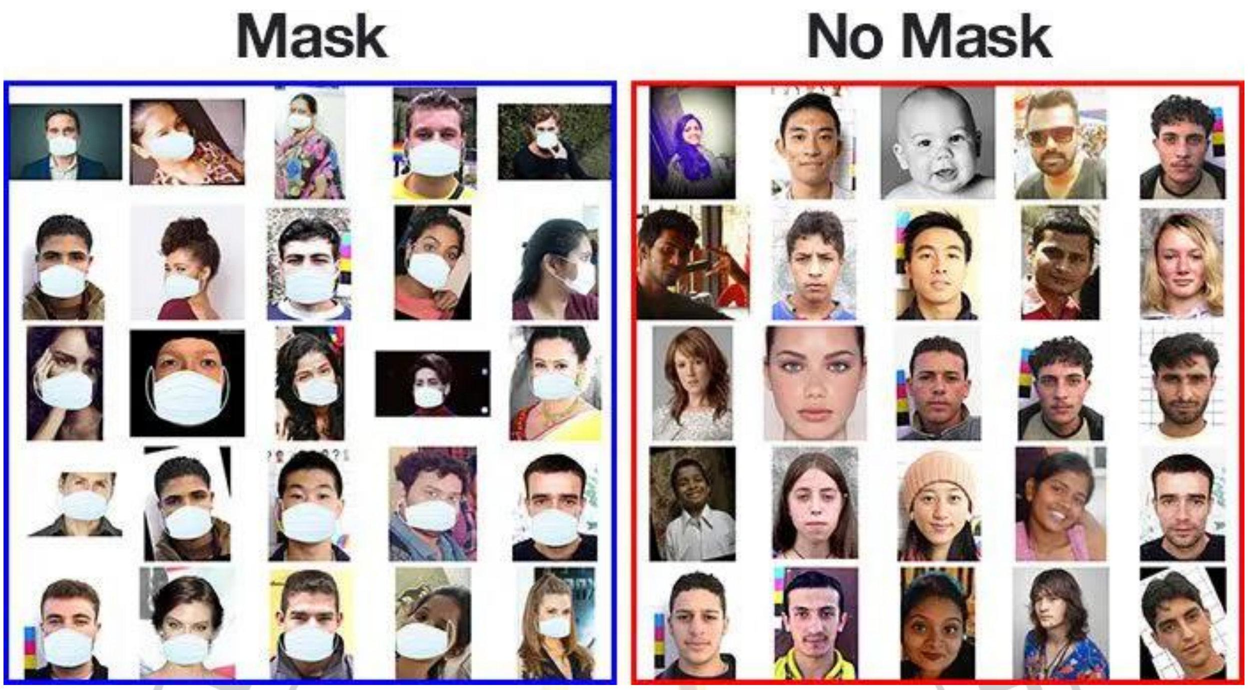
Live webcam :



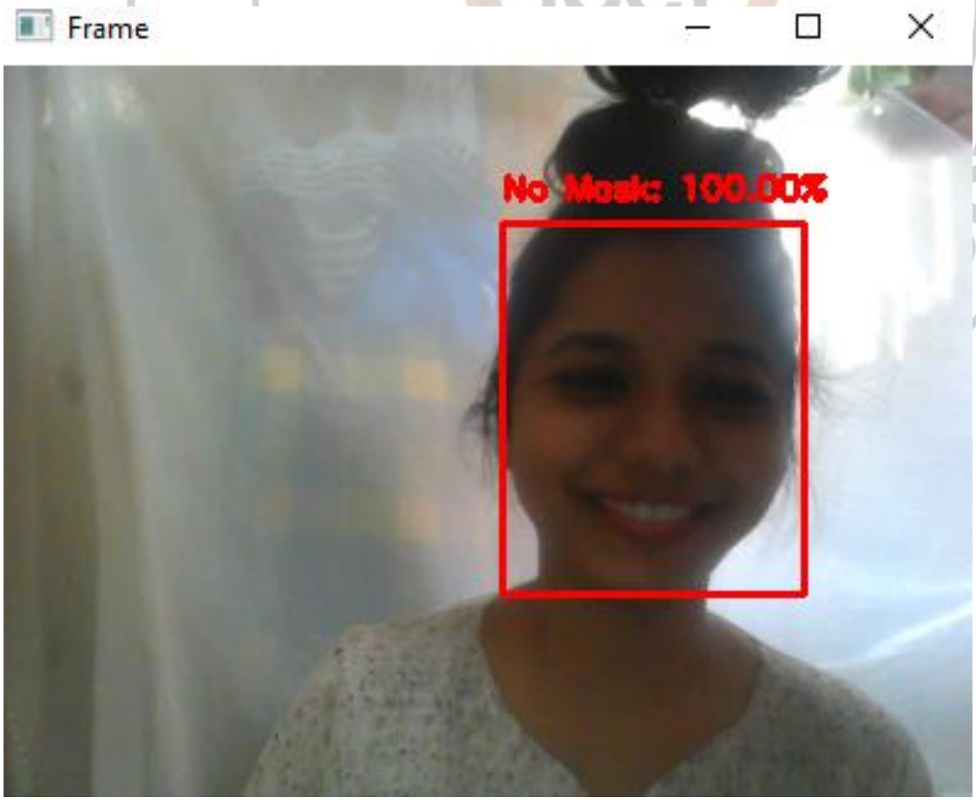
Face Mask Detection in webcam stream:
The flow to identify the person in the webcam wearing the face mask or not. The process is two-fold.
 1. To identify the faces in the webcam
 2.Classify the faces based on the mask.

Identify the Face in the Webcam:
To identify the faces a pre-trained model provided by the OpenCV framework was used. The model was trained using web images. OpenCV provides 2 models for this facedetector

Training Dataset :

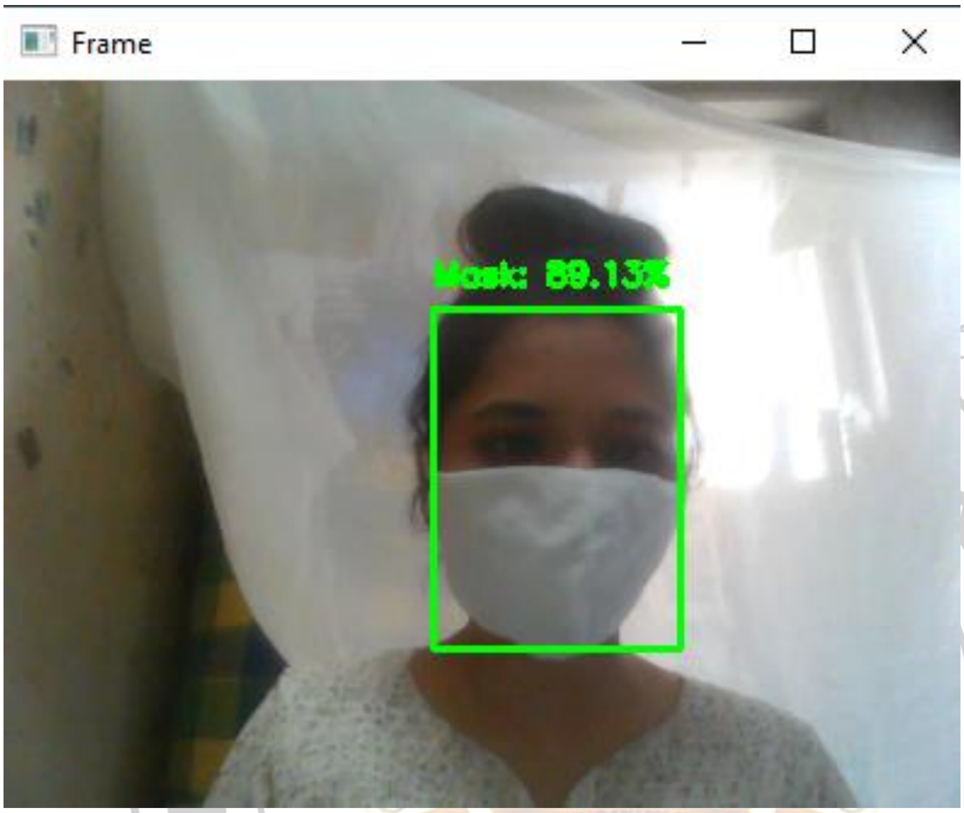


REAL TIME INPUT :

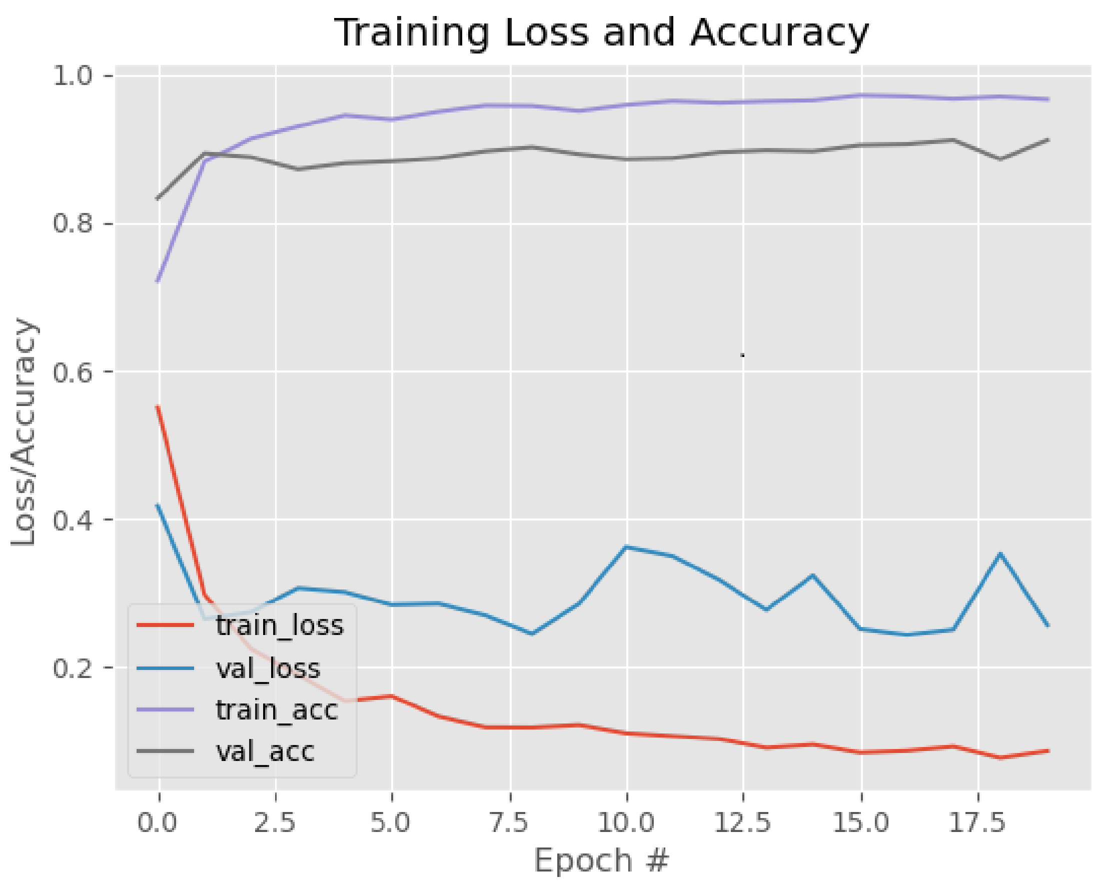


Here we can see that our mode is 100 % sure or accurate that a person is not wearing a mask.

REAL TIME OUTPUT :



Here we can see that our mode is 89.13 % sure or accurate that a person is wearing a mask.



Chapter 6. Conclusion and Future Scope

In this project, we have proposed a novel face mask detector, namely FaceMask Detection System, which can possibly contribute to public healthcare. The architecture of FaceMask Detection System consists of CNN as backbone, FPN as the neck, and context attention modules as the heads. The strong backbone ResNet and light backbone MobileNet can be used for high and low computation scenarios, respectively. In order to extract more robust features, we utilize transfer learning to adopt weights from a similar task face detection, which is trained on a very large dataset. Furthermore, we have proposed a novel context attention head module to focus on the face and mask features, and a novel algorithm object removal cross class, i.e. ORCC, to remove objects with lower confidence and higher IoU. The proposed method achieves state-of-the-art results on a public face mask dataset. An Iot component is added to the System to display the messages in a hardware. the data from the program is collected by the Iot hardware and it will dispalyed as the warning signal to the concerned authority.

Future Scope :

some countries with strong vaccination records may now have lower immunity than others but If suppose in future we come across similar condition like COVID19 and lockdown is again applied on the whole world, our project will be most helpful during such pandemic condition. We will link our python code / software to a hardware like self opening doors like in malls and other public places where the door will not open until a person is wearing a mask. This will help people to be more protected and safe. As far as the efficiency of our project goes, we can enhance the accuracy of this model by more image Training , and also proper lighting and usage of efficient cameras the accuracy of the system can be improved.

Chapter 7. Case Study

7.1 Problem Definition

Google Search Analysis with Python

7.2 Introduction

Google doesn’t give much access to the data about daily search queries, but another application of google known as Google Trends can be used for the task of Google search analysis. Google Trends provides an API that can be used to analyze the daily searches on Google. This API is known as pytrends, you can easily install it in your systems by using the pip command; pip install pytrends.

Now let’s get started with the task of Google search analysis by importing the necessary Python libraries:

```
import
pandas
as pd
        from pytrends.request import TrendReq
        import matplotlib.pyplot as plt
        trends = TrendReq()
```

Here I will be analyzing the Google search trends on the queries based on “Machine Learning”, so let’s create a DataFrame of the top 10 countries which search for “Machine Learning” on Google:

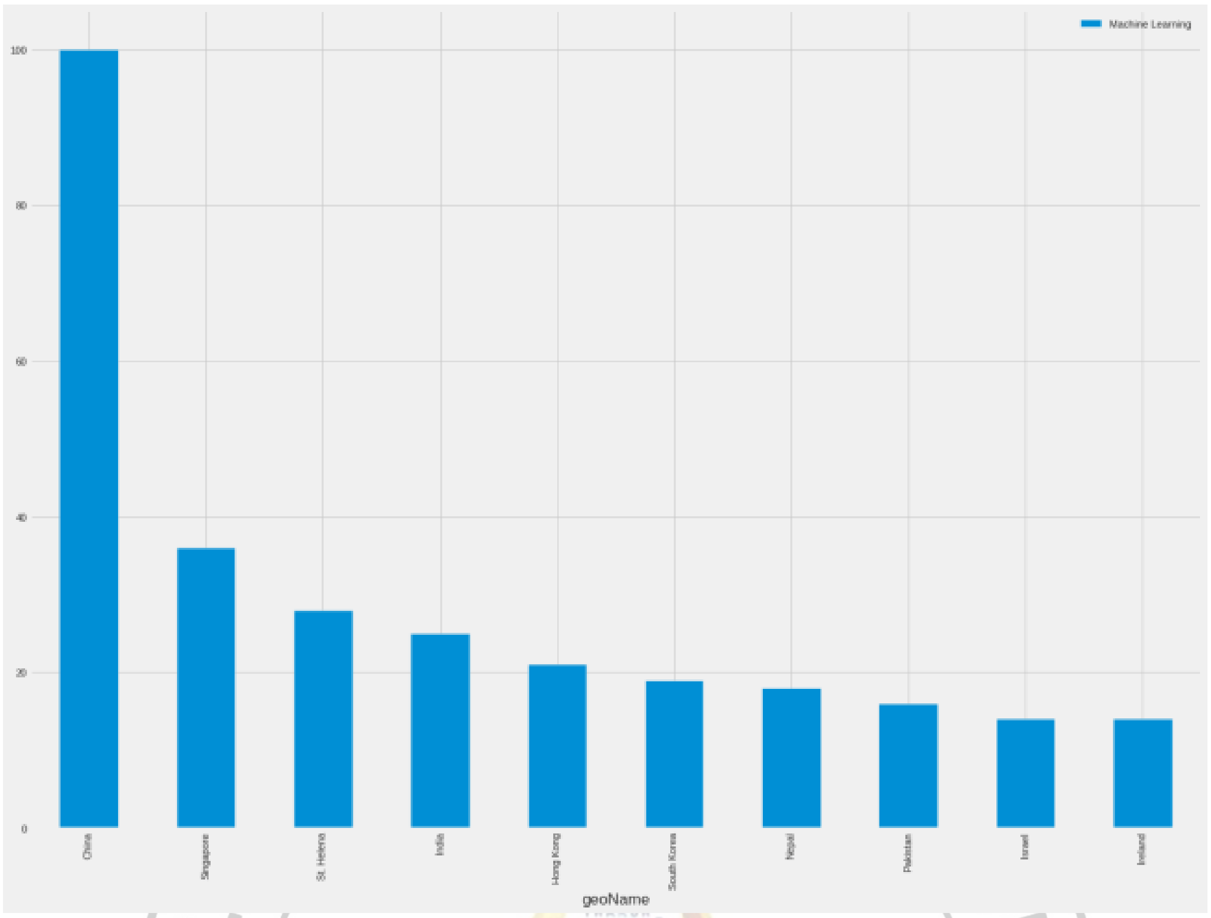
```
trends.build_payload(kw_list=["Machine
Learning"])

                                              data = trends.
                                              interest_by_region()
                                              data = data.sort_values
                                              (by="Machine Learning",
                                              ascending=False)
                                              data = data.head(10)
                                              print(data)
```

```
Machine Learning
geoName
China                  100
```

```
Singapore              36
St. Helena             28
India                 25
Hong Kong              21
South Korea            19
Nepal                 18
Pakistan               16
Israel                14
Ireland               14
```

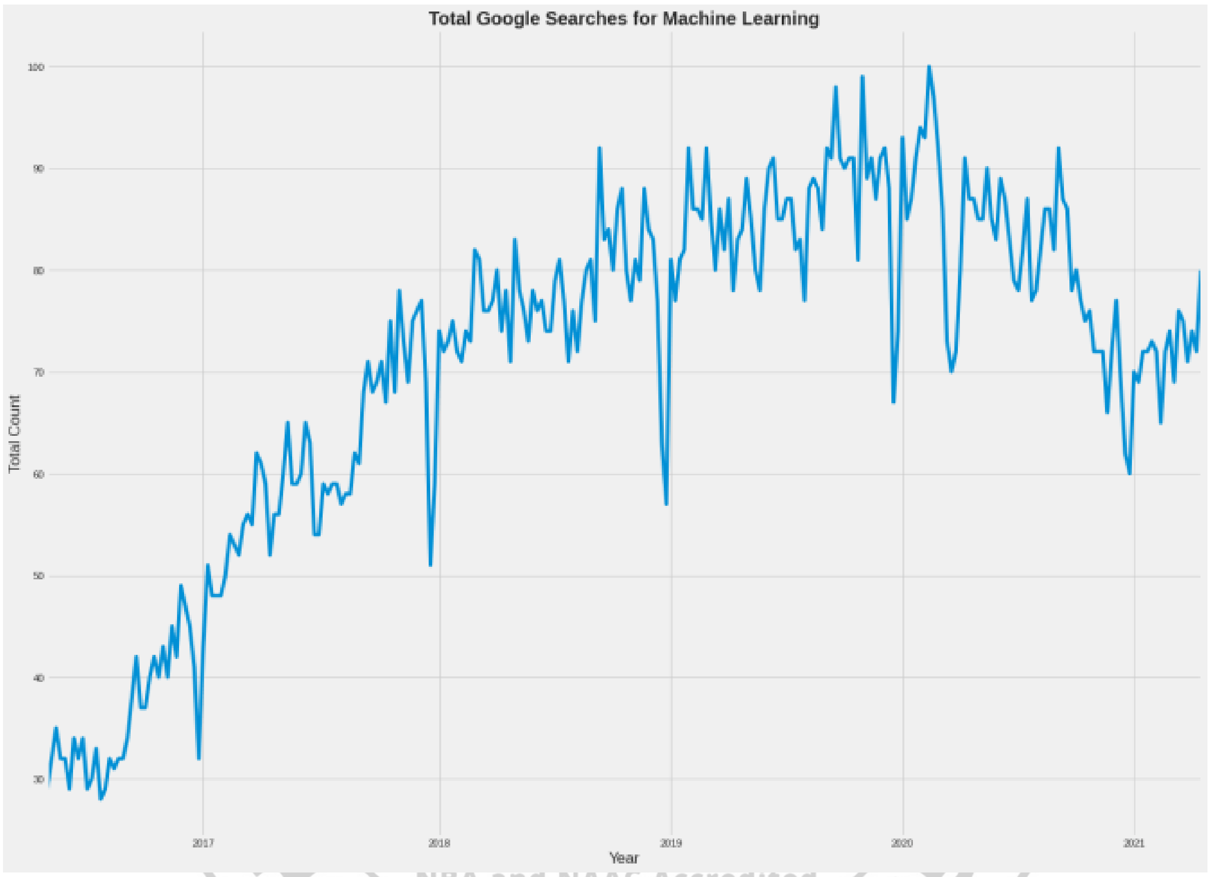
So, according to the above results, the search queries based on "Machine learning" are mostly done in China. We can also visualize this data using a bar chart:

```
data.reset_index().plot(x="geoName",y="Machine
Learning",
                                             figsize=(20,15),
                                              kind="bar")
                                             plt.style.use
                                             ('fivethirtyeight')
                                             plt.show()
```



So as we all know that Machine Learning has been the focus of so many companies and students for the last 3-4 years, so let’s have a look at the trend of searches to see how the total search queries based on “Machine Learning” increased or decreased on Google:

```
data =
TrendReq(hl='en-
US', tz=360)
          data.build_payload(kw_list=['Machine Learning'])
          data = data.interest_over_time()
          fig, ax = plt.subplots(figsize=(20, 15))
          data['Machine Learning'].plot()
          plt.style.use('fivethirtyeight')
          plt.title('Total Google Searches for Machine Learning
          ', fontweight='bold')
          plt.xlabel('Year')
          plt.ylabel('Total Count')
          plt.show()
```



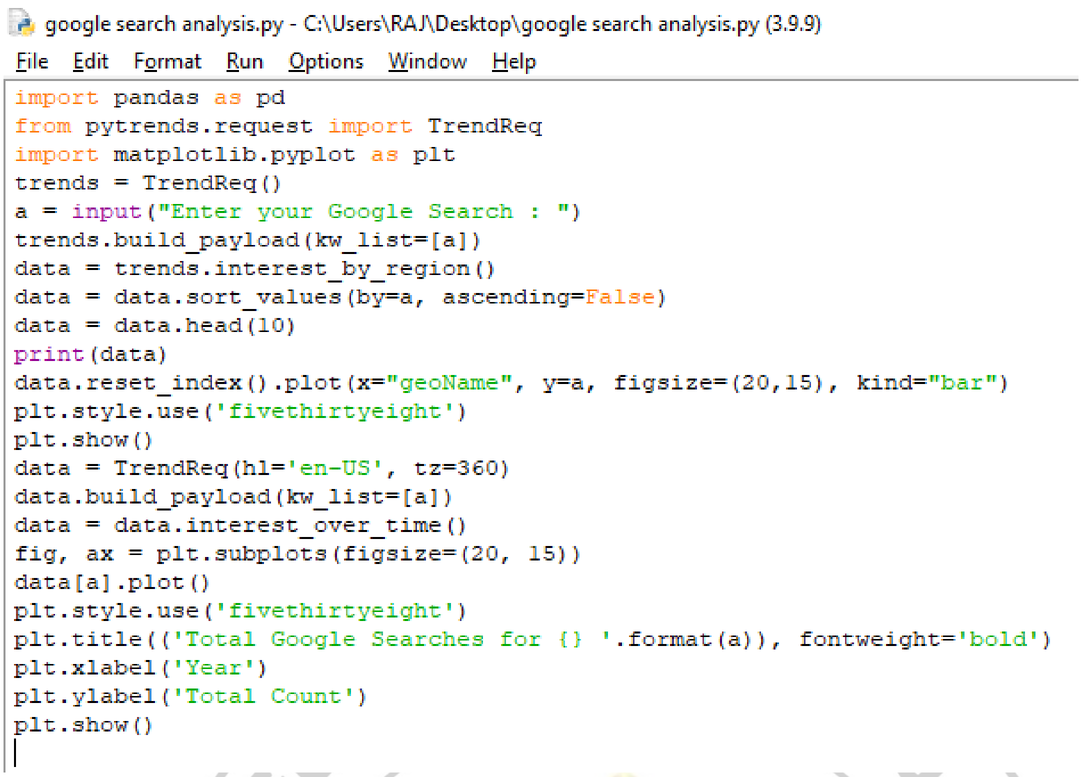
7.3 Implementation :

Full Code :

```
import pandas as pd
```

```
from pytrends.request import TrendReq
import matplotlib.pyplot as plt
trends = TrendReq()
a = input("Enter your Google Search : ")
trends.build_payload(kw_list=[a])
data = trends.interest_by_region()
data = data.sort_values(by=a, ascending=False)
data = data.head(10)
print(data)
data.reset_index().plot(x="geoName", y=a, figsize=(20,15), kind="bar")
plt.style.use('fivethirtyeight')
plt.show()
data = TrendReq(hl='en-US', tz=360)
data.build_payload(kw_list=[a])
data = data.interest_over_time()
fig, ax = plt.subplots(figsize=(20, 15))
data[a].plot()
plt.style.use('fivethirtyeight')
plt.title(('Total Google Searches for {} '.format(a)), fontweight='bold')
plt.xlabel('Year')
plt.ylabel('Total Count')
plt.show()
```

Screenshot :


google search analysis.py - C:\Users\RAJ\Desktop\google search analysis.py (3.9.9)
File Edit Format Run Options Window Help

```
import pandas as pd
from pytrends.request import TrendReq
import matplotlib.pyplot as plt
trends = TrendReq()
a = input("Enter your Google Search : ")
trends.build_payload(kw_list=[a])
data = trends.interest_by_region()
data = data.sort_values(by=a, ascending=False)
data = data.head(10)
print(data)
data.reset_index().plot(x="geoName", y=a, figsize=(20,15), kind="bar")
plt.style.use('fivethirtyeight')
plt.show()
data = TrendReq(hl='en-US', tz=360)
data.build_payload(kw_list=[a])
data = data.interest_over_time()
fig, ax = plt.subplots(figsize=(20, 15))
data[a].plot()
plt.style.use('fivethirtyeight')
plt.title(('Total Google Searches for {} '.format(a)), fontweight='bold')
plt.xlabel('Year')
plt.ylabel('Total Count')
plt.show()
```

7.4 Conclusion :

So we can see that searches based on “machine learning” on Google started to increase in 2017 and the highest searches were done in 2020 till today. This is how we can analyze Google searches based on any keyword. A business can perform Google search analysis to understand what people are looking for on Google at any given time.

List of References :

https://iopscience.iop.org/article/10.1088/1742-6596/1916/1/012084/pdf

http://cse.anits.edu.in/projects/projects2021C4.pdf

https://www.geeksforgeeks.org/facemask-detection-using-tensorflow-in-python/

https://www.semanticscholar.org/paper/Masked-Face-Recognition-Using-Convolutional-Neural-Ejaz-Islam/b6fa4261c2f6b6198b73caf1ddc7cc0b88ad184f

https://dl.acm.org/doi/10.1145/954339.954342

https://www.mendeley.com/catalogue/5877ae99-1c1a-33b7-a28a-b2077e33d910/