A Synopsis Report

[Based on B.Tech Final Year BTP]On

Face Mask Detection

Submitted By

Vishal Srivastava

Ashwani Singh

Bachelor of Technology,

IT(2K18)]Under the Guidance of

Er. Prateek Srivastava

Assistant Professor (Department of Information Technology)

University Institute of Engineering and Technology

Department of Information Technology

CANDIDATES’ DECLARATION

I hereby declare that the major project work being presented in this report entitled “**Face Mask Detection**” submitted in the departmentof Information Technology, University Institute of Engineering and Technology, Kanpur is the authentic work carried out by us under the guidance of Er. Prateek Srivastava.

Date: 20/05/2022

Vishal Srivastava (CSJMA18001390175)

Ashwani Singh (CSJMA17001390133)

[Department of Information Technology,UIET, CSJM University, Kanpur]

ACKNOWLEDGEMENT

On the submission of our Synopsis report on “**Face Mask Detection**”

We would like to express our indebted gratitude and special thanks toour Assistant Professor ***Er. Prateek Srivastava*** of ***Department of Information Technology*** who in spite of being extraordinarily busy, spare time for guidance and keep us on the correct path. We truthfullyappreciate and value his admired supervision and support from the start to the end of this project.

We were obliged to him for having helped us shape the trouble and providing insights towards the way out. We would like to offer our special thanks to for providing a solid backdrop for out studies and explore afterward. They have been great sources of motivation to meand I thank them from the core of my heart. Last but not the least I would like to thank each and every person who is involved directly orindirectly to make this project successful.

Date: 20/05/2022

Vishal Srivastava (CSJMA18001390175)

Ashwani Singh (CSJMA17001390133)

[Department of Information Technology,UIET, CSJM University, Kanpur]

Abstract

Changes in the lifestyle of everyone around the world. In those changes wearing a mask has been very vital to every individual. Detection of people who are not wearing masks is a challenge due to Outbreak of the Coronavirus pandemic has created various the large number of populations. This project can be used in schools, hospitals, banks, airports, and etc. as a digitalized scanning tool. The technique of detecting people's faces and segregating them into two classes namely the people with masks and people without masks is done with the help of image processing and deep learning. With the help of this project, a person who is intended to monitor the people can be seated in a remote area and still can monitor efficiently and give instructions accordingly. Various libraries of python such as Open CV, Tensor flow and Keras. In Deep Learning Convolution Neural Networks is a class Deep Neural Networks which is used to train the models used for this project.

The purpose of the project "Face Mask Detection " is to create a tool that identifies the image of a human that can calculate the probability that he/she wearing a mask or not. Due to COVID, starts going through various stages of reopening, face masks have become an important element of our daily lives to stay safe. Wearing face masks will be required in order to socialize or conduct business. So, this application utilizes a camera to detect if a person is wearing a mask or not.

Table of Contents

1. Introduction

1.1. Overall Description:

The trend of sporting face mask publically is rising because of the Covid-19 epidemic everywhere in the world. Because Covid-19 people wont to wear mask to shield their health from air pollution. Whereas other are self-conscious concerning their looks, they hide their emotions from the general public by activity their faces. Somebody treated the wearing face masks works on hindering Covid-19 transmission. Covid-19 is that the last epidemic virus that hit the human health within the last century .In 2020, the fast spreading of Covid-19 has forced the who to declare Covid-19 as international pandemic. Quite 5 million cases were infected by Covid-19 in not up to half dozen month across 188 countries. The virus spreads through shut contact and in packed and overcrowded areas. The corona virus epidemic has given rise to a unprecedented degree of worldwide scientific cooperation.Computer science supported machine learning and deep learning will facilitate to fight Covid-19 in several ways. Machine learning a valuate huge quantities of knowledge to forecast the distribution of Covid-19 to function early warning mechanism for potential pandemics, and classify vulnerable population. Folks are forced by laws to wear face masks publically many countries. These rules and law we have a tendency yore developed as associate degree action to the exponential growth in cases an deaths in several areas.

However, the method observation massive teams of individuals are changing into a lot of difficult. The monitoring process involves the finding of anyone who isn't sporting a face mask. Here we introduce a mask face detection model that's supported machine learning and image process techniques. The planned model may be detect the mask with image and real time detection people wearing mask or not wearing a mask. The model is integration between deep learning and classical machine learning techniques with Oven CV, Tensor Flow and Keras. We have a tendency to introduced a comparison between them to seek out the foremost appropriate algorithm program that achieved the very best accuracy and consumed the smallest mount time within the method of coaching and detection.

1.1.1 CNN (Convolution Neural Network)

CNN Convolutional Neural Network are designed to process data through multiple layers of arrays. This type of neural networks is used in application like image recognition of face recognition. The primary difference between CNN and other ordinary neural network is that CNN takes input as a two dimensional array and operates directly on the images rather than focusing on feature extraction which other neural network focus on.The dominant approach of CNN includes solutions for problems of recognition. Top companies like google and facebook have invested in research and developments towards recognition projects to get activites done with greater speed.

1.1.2 Training of CNN Model

Convolutional Neural Network in this planned method, the mask detection model is constructed victimization the successive API of the Keras library. This permits us to make the new layers for our model step by step. The assorted layers used for our CNN model is represented below.

The **1st layer** is that the Conv2D layer with one hundred filters and therefore the filter size or the kernel size of 3x3. During this first step, the activation operate used is the “ReLu”. This ReLu function stands for the corrected linear measure which is able to output the input directly if is positive, otherwise it’ll output zero. The input size is also initialized as 150X150X3 for all the photographs to be trained and tested victimization this model.

In the **second layer**, the MaxPooling2D is employed with the poll size of 2x2.

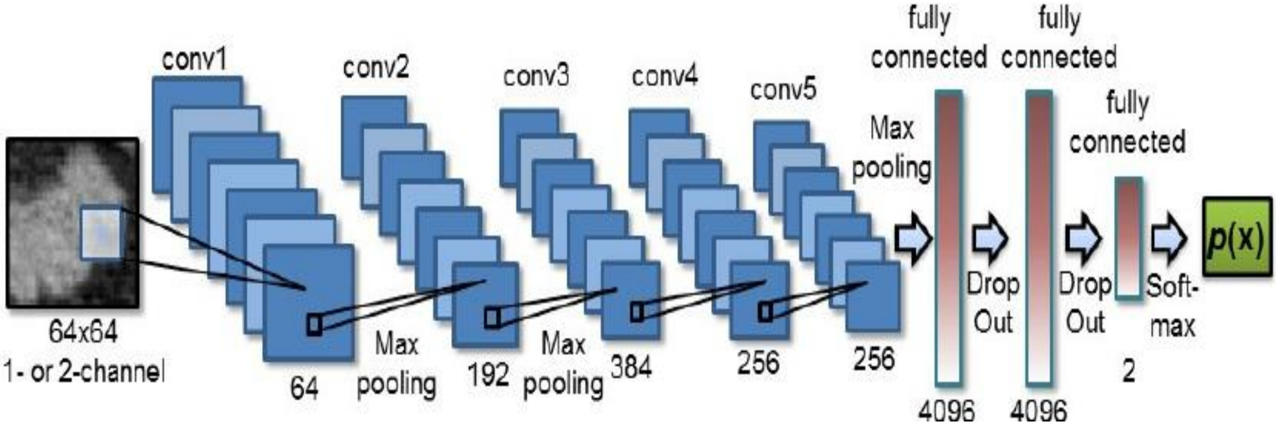
The **next layer** is once more a Conv2D layer with another one hundred filters of constant filter size 3X3 and {also the} activation operate used is that the ‘ReLu’. This Conv2D layer is followed by a MaxPooling3=2D layer with poll size 2x2.

In consecutive step, we have a tendency to use the **flatten () layer** to flatten all the layers into one 1D layer.

After the flatten layer, we use the dropout (0.5) layer to forestall the model form overfitting.

Finally, towards the end, we have a tendency to use the **dense layer** with fifty units and therefore the activation operate as ReLu.

The last layer of our model are going to be another dense layer with solely 2 units and the activation function used will be the 'softmax' function. The softmax function outputs a vector which is able to represent the chance distribution of every of the input units. Here, two inputs units are used. The softmax function will output a vector with two probability distribution values.



1.2. Purpose

In this work, we propose a introduce a mask face detection model that's supported machine learning and image process techniques. The planned model may be detect the mask with image and real time detection people wearing mask or not wearing amask. The model is integration between deep learning and classical machine learning techniques with Oven CV, Tensor Flow and Keras. We have a tendency to introduced a comparison between them to seek out the foremost appropriate algorithm program that achieved the very best accuracy and consumed the smallest mount time within the method of coaching and detection.

2. Proposed Model

2.1. Programming language Requirement

Python is a widely used high-level programming language for general purpose programming, created by Guido van Rossum and first released in1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation todelimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than

possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small andlarge scale.

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

2.2. Libraries required

2.2.1 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. Using NumPy in Python gives functionalitycomparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

2.2.2 MATPLOTLIB

MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB- like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality.

2.2.3 PANDAS

Pandas is an open-source Python Library providing high- performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived fromthe word Panel Data– an Econometrics from Multidimensional data.

2.2.4 Oven CV

Open CV (Open Source Computer Vision Library) is a collection of algorithms for computer vision. it basics focus on real time image processing it is free for commercial and research use under a BSD license.

2.2.5 Tensor Flow

Tensor Flow is a mathematical computation library for training and building your machine learning and deep learning model with a simple to use high level APIs.

2.2.6 Keras

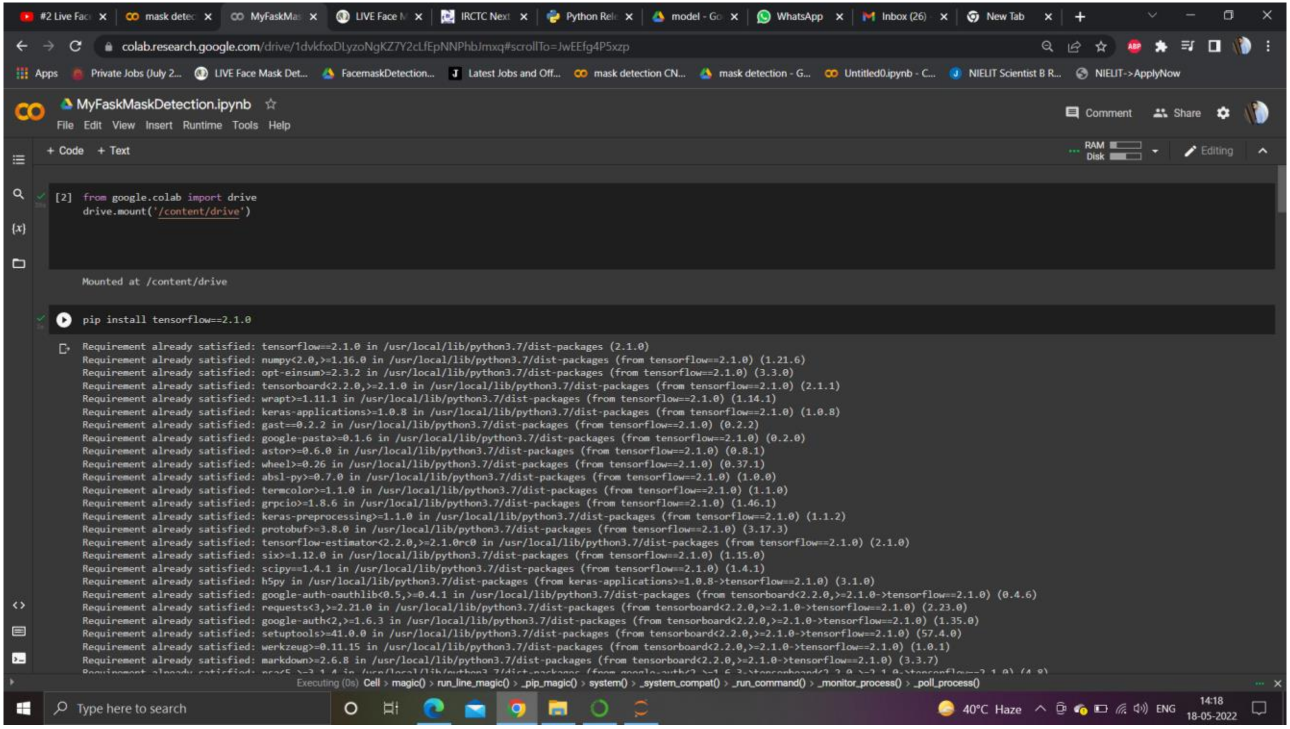
Keras is a neural network API. It is library written specifically in python. In addition, It works with other libraries and packages such as tensorflow which makes deep learning easier. Keras was developed to allow for quick experimentation and for fast prototyping.

Raw Data

In this stage, the historical data is collected from https://data-flair.training and https://indianaiproduction.com this historical data is used for the training of CNN model. We will use python built in library pandas function web-reader for web scrapping.

Implementation:

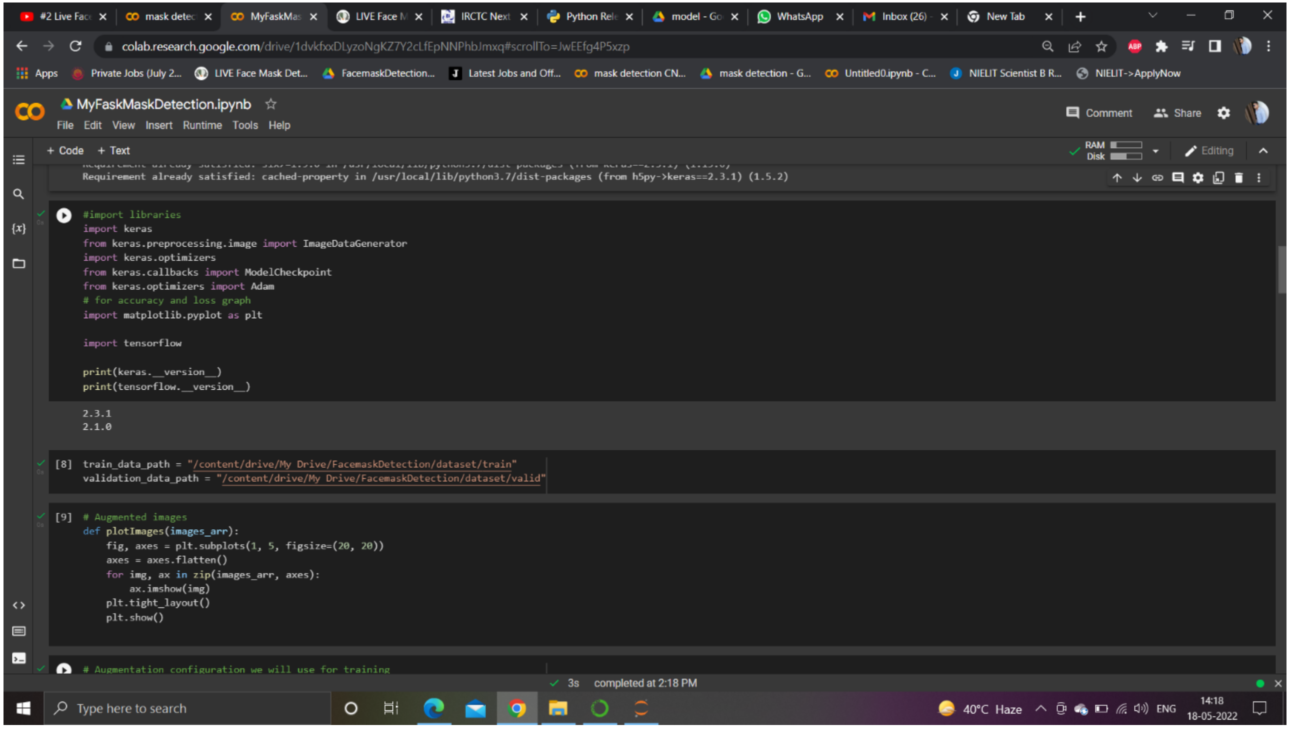
1. Training CNN Model


MyFaskMaskDetection.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
Comment Share Editing

```
[2] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

```
pip install tensorflow==2.1.0

Requirement already satisfied: tensorflow==2.1.0 in /usr/local/lib/python3.7/dist-packages (2.1.0)
Requirement already satisfied: numpy<2.0,>=1.16.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (1.21.6)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (3.3.0)
Requirement already satisfied: tensorboard<2.2.0,>=2.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (2.1.1)
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (1.14.1)
Requirement already satisfied: keras-applications>=1.0.8 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (1.0.8)
Requirement already satisfied: gast==0.2.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (0.2.2)
Requirement already satisfied: google-pasta>=0.1.6 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (0.2.0)
Requirement already satisfied: astor>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (0.8.1)
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (0.37.1)
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (1.0.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (1.1.0)
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (1.46.1)
Requirement already satisfied: keras-preprocessing>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (1.1.2)
Requirement already satisfied: protobuf>=3.8.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (3.17.3)
Requirement already satisfied: tensorflow-estimator<2.2.0,>=2.1.0rc0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (2.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (1.15.0)
Requirement already satisfied: scipy==1.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.1.0) (1.4.1)
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from keras-applications>=1.0.8->tensorflow==2.1.0) (3.1.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (0.4.6)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (2.23.0)
Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (1.35.0)
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (57.4.0)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (1.0.1)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1.0) (3.3.7)
```

Executing (0s) Cell > magic() > run_line_magic() > _pip_magic() > system() > _system_compat() > _run_command() > _monitor_process() > _poll_process()
Type here to search
40°C Haze ENG 14:18 18-05-2022


MyFaskMaskDetection.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
Comment Share Editing

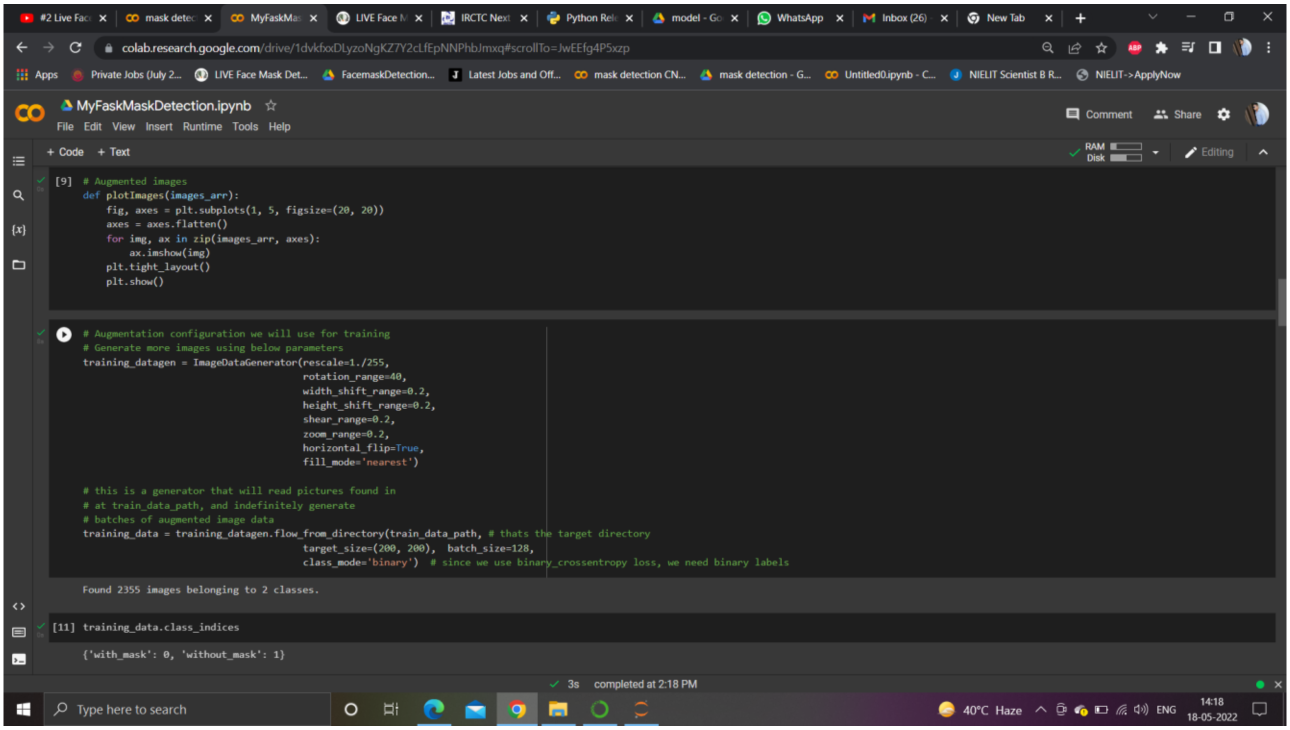
```
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages (from h5py->keras==2.3.1) (1.5.2)
```

```
#import libraries
import keras
from keras.preprocessing.image import ImageDataGenerator
import keras.optimizers
from keras.callbacks import ModelCheckpoint
from keras.optimizers import Adam
# for accuracy and loss graph
import matplotlib.pyplot as plt

import tensorflow

print(keras.__version__)
print(tensorflow.__version__)

2.3.1
2.1.0
```

```
[8] train_data_path = "/content/drive/My Drive/FacemaskDetection/dataset/train"
    validation_data_path = "/content/drive/My Drive/FacemaskDetection/dataset/valid"
```

```
[9] # Augmented images
    def plotImages(images_arr):
        fig, axes = plt.subplots(1, 5, figsize=(20, 20))
        axes = axes.flatten()
        for img, ax in zip(images_arr, axes):
            ax.imshow(img)
        plt.tight_layout()
        plt.show()
```

```
# Augmentation configuration we will use for training
```


MyFaskMaskDetection.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
3s completed at 2:18 PM

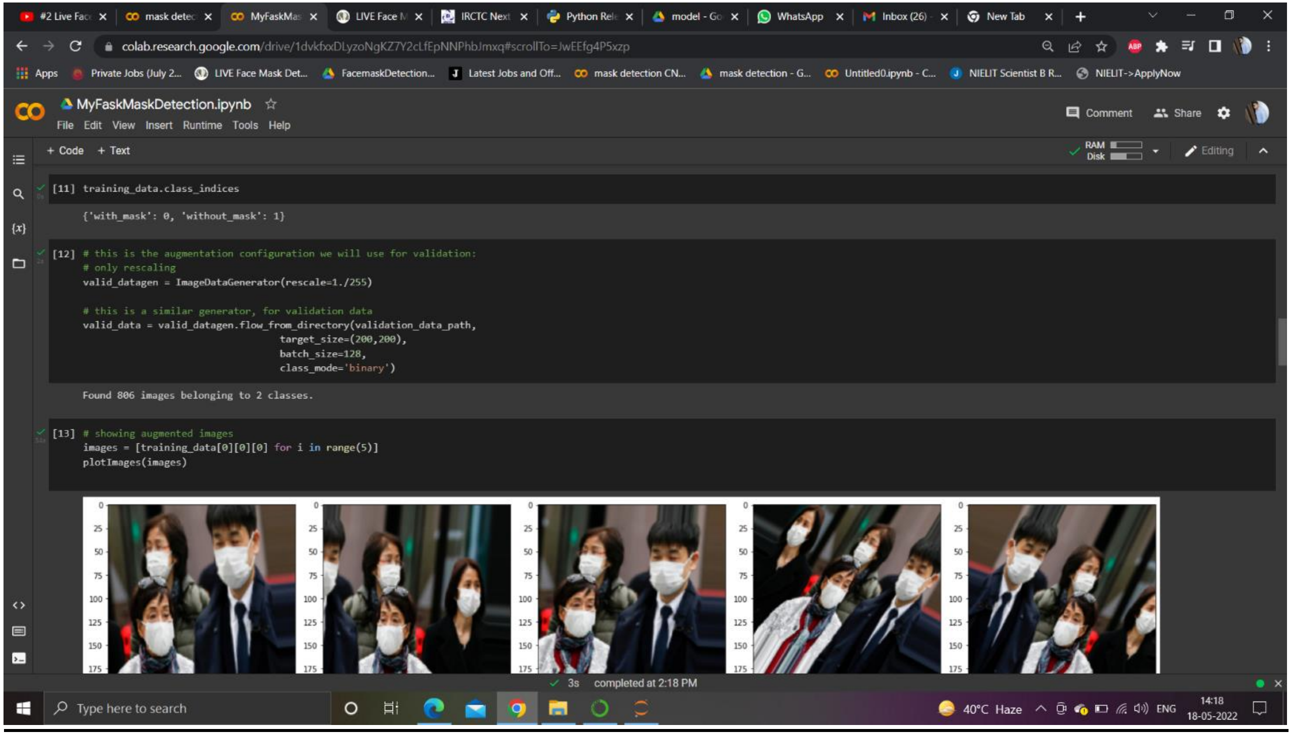
```
[9] # Augmented images
    def plotImages(images_arr):
        fig, axes = plt.subplots(1, 5, figsize=(20, 20))
        axes = axes.flatten()
        for img, ax in zip(images_arr, axes):
            ax.imshow(img)
        plt.tight_layout()
        plt.show()
```

```
# Augmentation configuration we will use for training
# Generate more images using below parameters
training_datagen = ImageDataGenerator(rescale=1./255,
                                      rotation_range=40,
                                      width_shift_range=0.2,
                                      height_shift_range=0.2,
                                      shear_range=0.2,
                                      zoom_range=0.2,
                                      horizontal_flip=True,
                                      fill_mode='nearest')

# this is a generator that will read pictures found in
# at train_data_path, and indefinitely generate
# batches of augmented image data
training_data = training_datagen.flow_from_directory(train_data_path, # thats the target directory
                                      target_size=(200, 200),  batch_size=128,
                                      class_mode='binary')  # since we use binary_crossentropy loss, we need binary labels
```

```
Found 2355 images belonging to 2 classes.
```

```
[11] training_data.class_indices
```

```
{'with_mask': 0, 'without_mask': 1}
```


MyFaskMaskDetection.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
0
25
50
75
100
125
150
175
3s completed at 2:18 PM

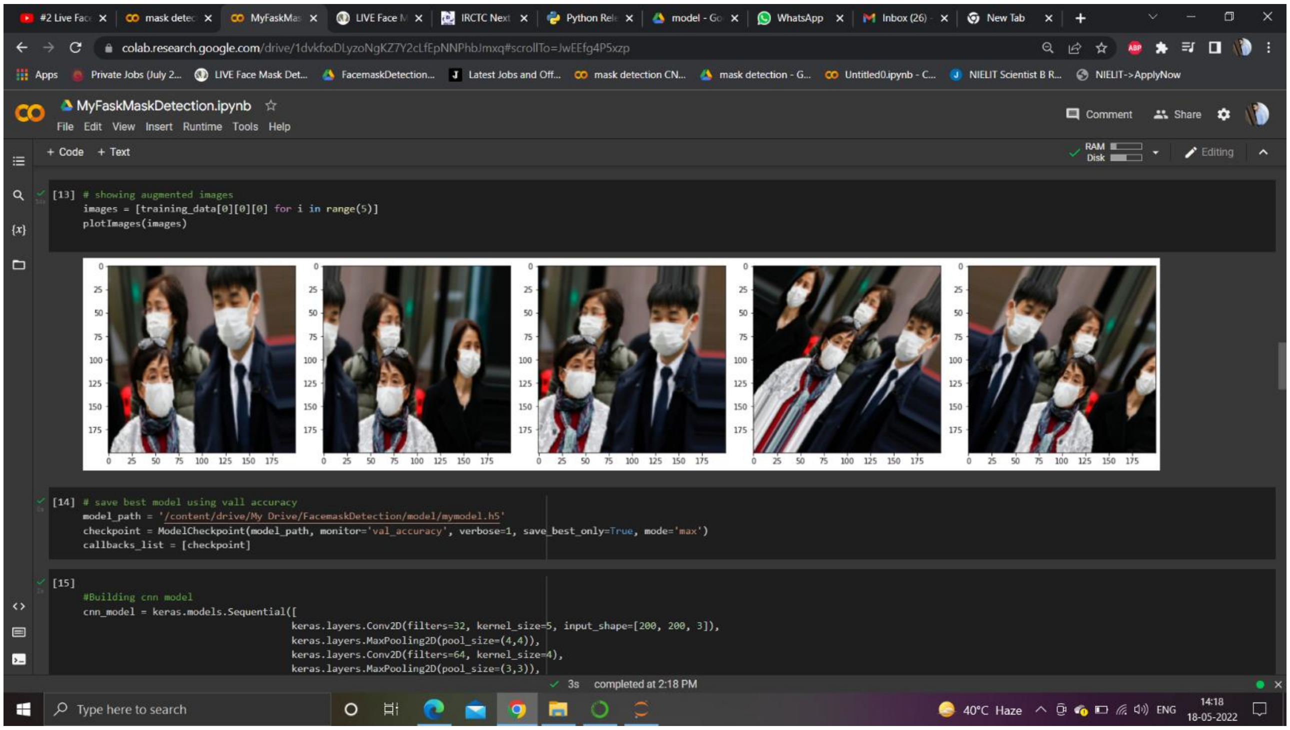
```
[11] training_data.class_indices
```

```
{'with_mask': 0, 'without_mask': 1}
```

```
[12] # this is the augmentation configuration we will use for validation:
     # only rescaling
     valid_datagen = ImageDataGenerator(rescale=1./255)

     # this is a similar generator, for validation data
     valid_data = valid_datagen.flow_from_directory(validation_data_path,
                                   target_size=(200,200),
                                   batch_size=128,
                                   class_mode='binary')
```

```
Found 806 images belonging to 2 classes.
```

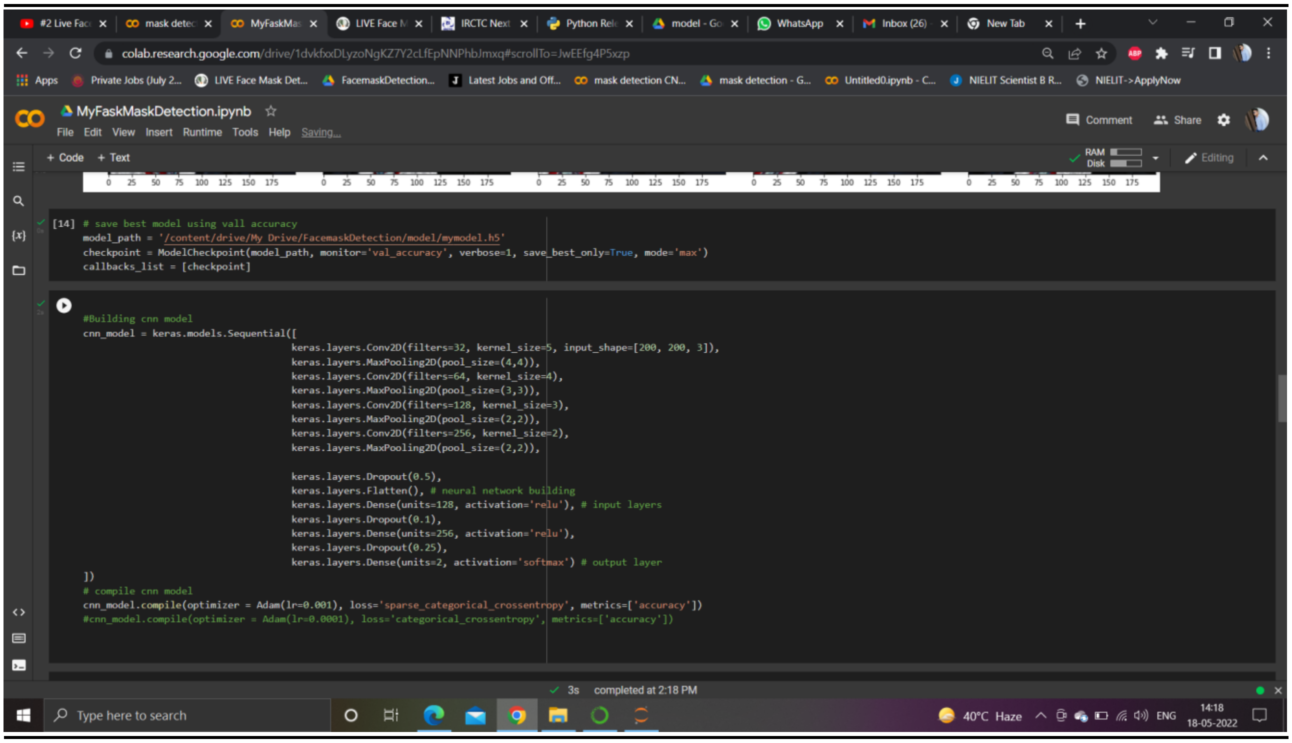
```
[13] # showing augmented images
     images = [training_data[0][0][0] for i in range(5)]
     plotImages(images)
```


MyFaskMaskDetection.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
3s completed at 2:18 PM

```
[13] # showing augmented images
     images = [training_data[0][0][0] for i in range(5)]
     plotImages(images)
```

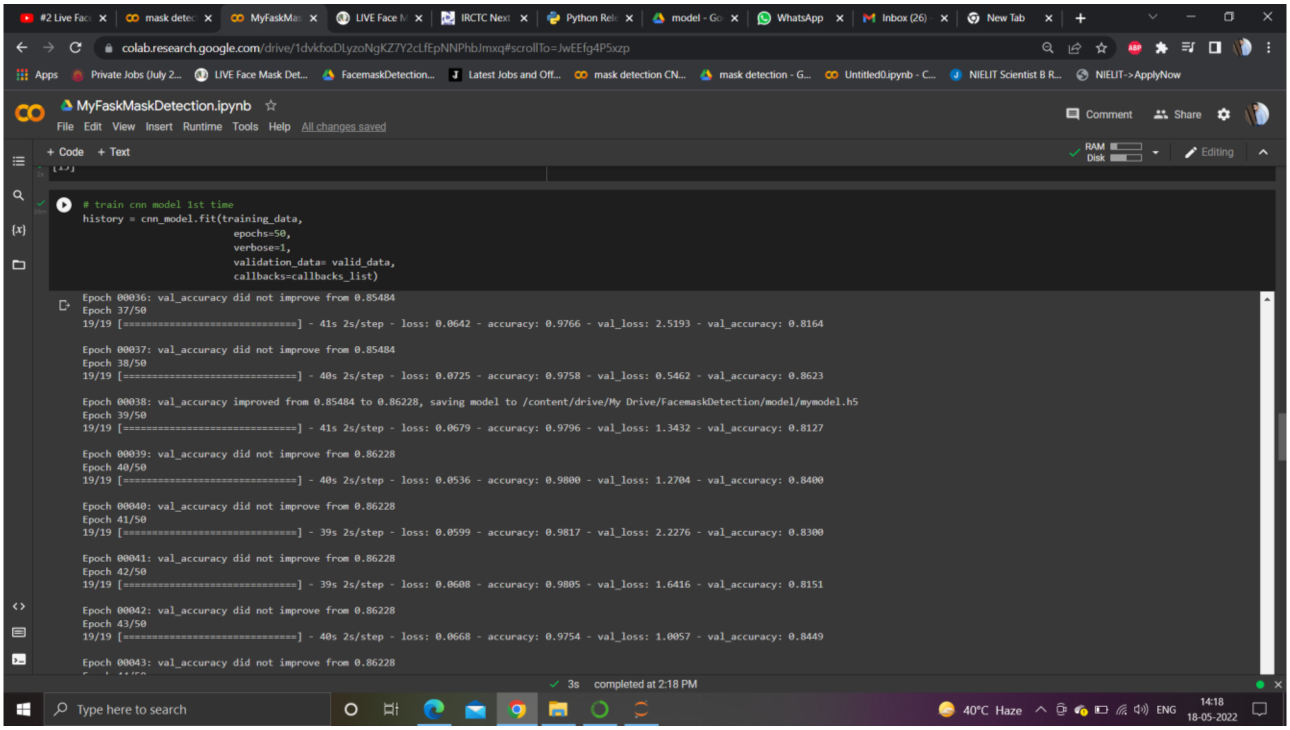
```
[14] # save best model using vall accuracy
     model_path = '/content/drive/My Drive/FacemaskDetection/model/mymodel.h5'
     checkpoint = ModelCheckpoint(model_path, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
     callbacks_list = [checkpoint]
```

```
[15]
     #Building cnn model
     cnn_model = keras.models.Sequential([
                                 keras.layers.Conv2D(filters=32, kernel_size=5, input_shape=[200, 200, 3]),
                                 keras.layers.MaxPooling2D(pool_size=(4,4)),
                                 keras.layers.Conv2D(filters=64, kernel_size=4),
                                 keras.layers.MaxPooling2D(pool_size=(3,3)),
```


MyFaskMaskDetection.ipynb
File Edit View Insert Runtime Tools Help Saving...
+ Code + Text
3s completed at 2:18 PM

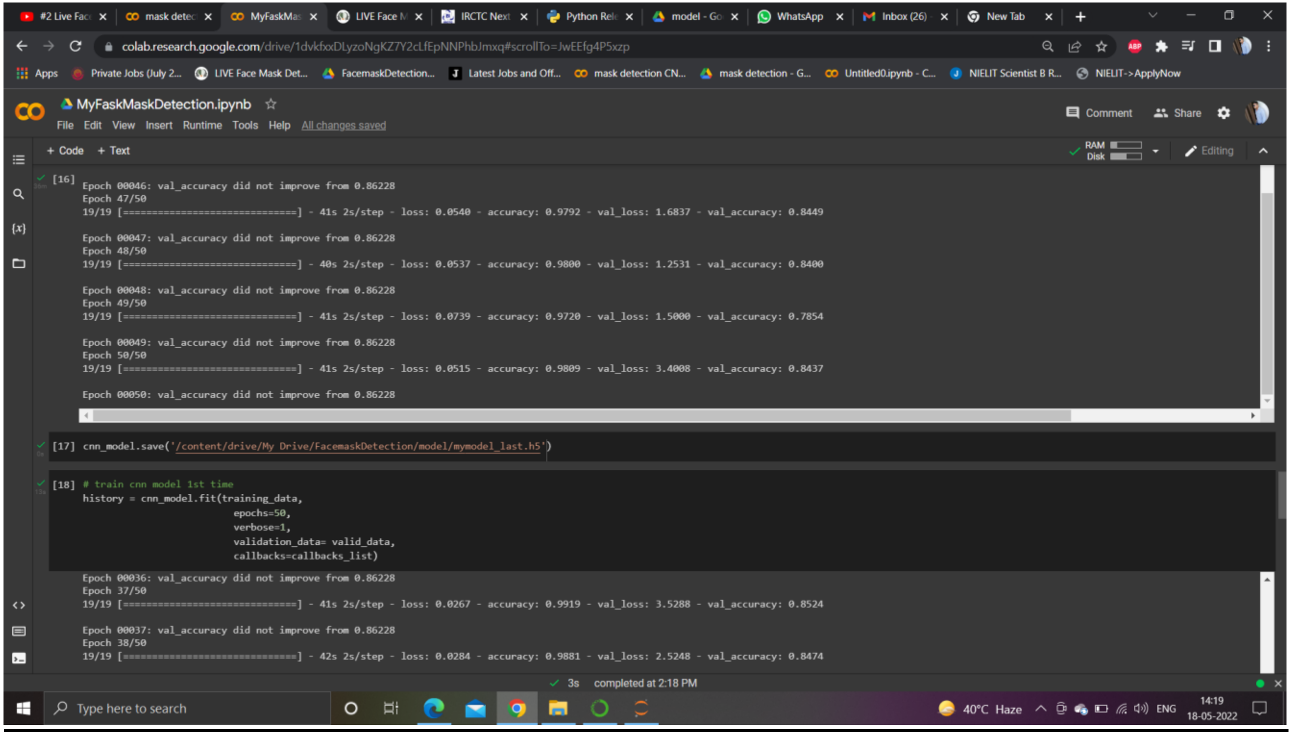
```
[14] # save best model using vall accuracy
     model_path = '/content/drive/My Drive/FacemaskDetection/model/mymodel.h5'
     checkpoint = ModelCheckpoint(model_path, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
     callbacks_list = [checkpoint]
```

```
     #Building cnn model
     cnn_model = keras.models.Sequential([
                                 keras.layers.Conv2D(filters=32, kernel_size=5, input_shape=[200, 200, 3]),
                                 keras.layers.MaxPooling2D(pool_size=(4,4)),
                                 keras.layers.Conv2D(filters=64, kernel_size=4),
                                 keras.layers.MaxPooling2D(pool_size=(3,3)),
                                 keras.layers.Conv2D(filters=128, kernel_size=3),
                                 keras.layers.MaxPooling2D(pool_size=(2,2)),
                                 keras.layers.Conv2D(filters=256, kernel_size=2),
                                 keras.layers.MaxPooling2D(pool_size=(2,2)),

                                 keras.layers.Dropout(0.5),
                                 keras.layers.Flatten(), # neural network building
                                 keras.layers.Dense(units=128, activation='relu'), # input layers
                                 keras.layers.Dropout(0.1),
                                 keras.layers.Dense(units=256, activation='relu'),
                                 keras.layers.Dropout(0.25),
                                 keras.layers.Dense(units=2, activation='softmax') # output layer
     ])
     # compile cnn model
     cnn_model.compile(optimizer = Adam(lr=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
     #cnn_model.compile(optimizer = Adam(lr=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```


MyFaskMaskDetection.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
RAM Disk
Editing
3s completed at 2:18 PM

```
# train cnn model 1st time
history = cnn_model.fit(training_data,
                        epochs=50,
                        verbose=1,
                        validation_data= valid_data,
                        callbacks=callbacks_list)
```

```
Epoch 00036: val_accuracy did not improve from 0.85484
Epoch 37/50
19/19 [==============================] - 41s 2s/step - loss: 0.0642 - accuracy: 0.9766 - val_loss: 2.5193 - val_accuracy: 0.8164

Epoch 00037: val_accuracy did not improve from 0.85484
Epoch 38/50
19/19 [==============================] - 40s 2s/step - loss: 0.0725 - accuracy: 0.9758 - val_loss: 0.5462 - val_accuracy: 0.8623

Epoch 00038: val_accuracy improved from 0.85484 to 0.86228, saving model to /content/drive/My Drive/FacemaskDetection/model/mymodel.h5
Epoch 39/50
19/19 [==============================] - 41s 2s/step - loss: 0.0679 - accuracy: 0.9796 - val_loss: 1.3432 - val_accuracy: 0.8127

Epoch 00039: val_accuracy did not improve from 0.86228
Epoch 40/50
19/19 [==============================] - 40s 2s/step - loss: 0.0536 - accuracy: 0.9800 - val_loss: 1.2704 - val_accuracy: 0.8400

Epoch 00040: val_accuracy did not improve from 0.86228
Epoch 41/50
19/19 [==============================] - 39s 2s/step - loss: 0.0599 - accuracy: 0.9817 - val_loss: 2.2276 - val_accuracy: 0.8300

Epoch 00041: val_accuracy did not improve from 0.86228
Epoch 42/50
19/19 [==============================] - 39s 2s/step - loss: 0.0608 - accuracy: 0.9805 - val_loss: 1.6416 - val_accuracy: 0.8151

Epoch 00042: val_accuracy did not improve from 0.86228
Epoch 43/50
19/19 [==============================] - 40s 2s/step - loss: 0.0668 - accuracy: 0.9754 - val_loss: 1.0057 - val_accuracy: 0.8449

Epoch 00043: val_accuracy did not improve from 0.86228
```

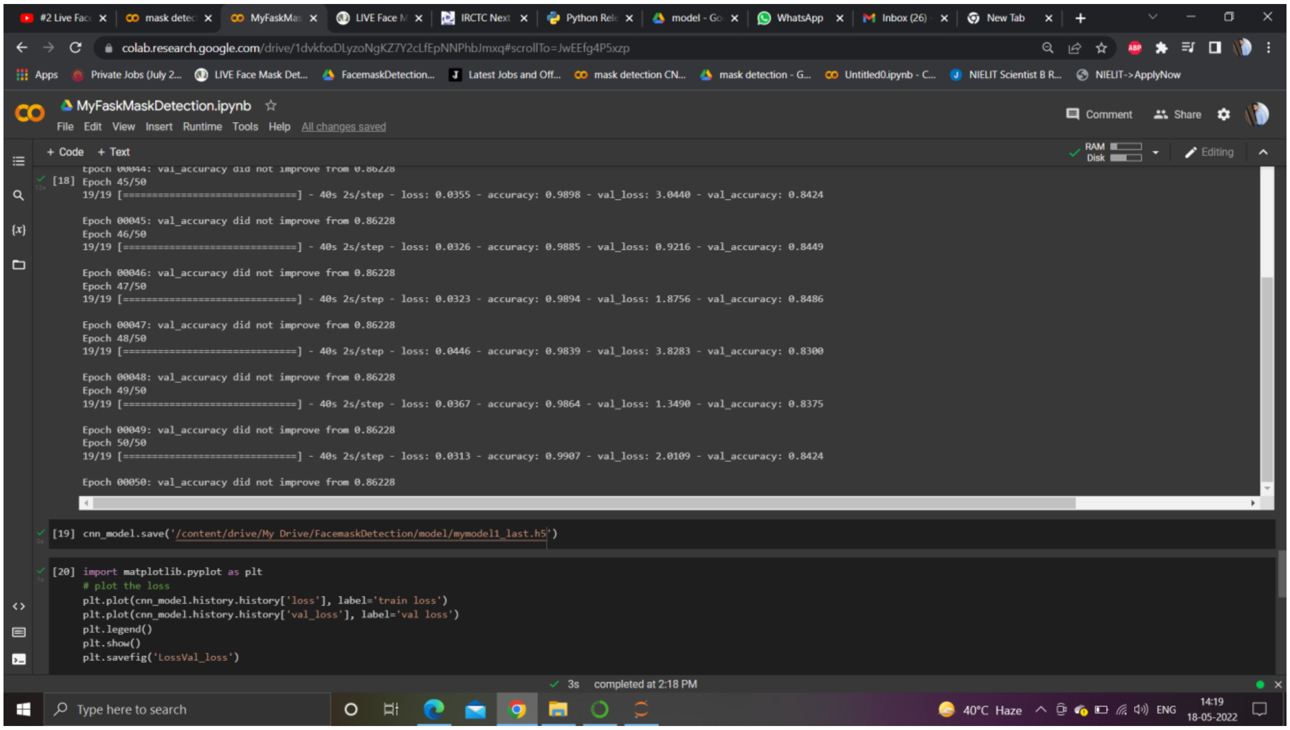

MyFaskMaskDetection.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
RAM Disk
Editing
3s completed at 2:18 PM

```
[16]
Epoch 00046: val_accuracy did not improve from 0.86228
Epoch 47/50
19/19 [==============================] - 41s 2s/step - loss: 0.0540 - accuracy: 0.9792 - val_loss: 1.6837 - val_accuracy: 0.8449

Epoch 00047: val_accuracy did not improve from 0.86228
Epoch 48/50
19/19 [==============================] - 40s 2s/step - loss: 0.0537 - accuracy: 0.9800 - val_loss: 1.2531 - val_accuracy: 0.8400

Epoch 00048: val_accuracy did not improve from 0.86228
Epoch 49/50
19/19 [==============================] - 41s 2s/step - loss: 0.0739 - accuracy: 0.9720 - val_loss: 1.5000 - val_accuracy: 0.7854

Epoch 00049: val_accuracy did not improve from 0.86228
Epoch 50/50
19/19 [==============================] - 41s 2s/step - loss: 0.0515 - accuracy: 0.9809 - val_loss: 3.4008 - val_accuracy: 0.8437

Epoch 00050: val_accuracy did not improve from 0.86228
```

```
[17] cnn_model.save('/content/drive/My Drive/FacemaskDetection/model/mymodel_last.h5')
```

```
[18] # train cnn model 1st time
history = cnn_model.fit(training_data,
                        epochs=50,
                        verbose=1,
                        validation_data= valid_data,
                        callbacks=callbacks_list)
```

```
Epoch 00036: val_accuracy did not improve from 0.86228
Epoch 37/50
19/19 [==============================] - 41s 2s/step - loss: 0.0267 - accuracy: 0.9919 - val_loss: 3.5288 - val_accuracy: 0.8524

Epoch 00037: val_accuracy did not improve from 0.86228
Epoch 38/50
19/19 [==============================] - 42s 2s/step - loss: 0.0284 - accuracy: 0.9881 - val_loss: 2.5248 - val_accuracy: 0.8474
```

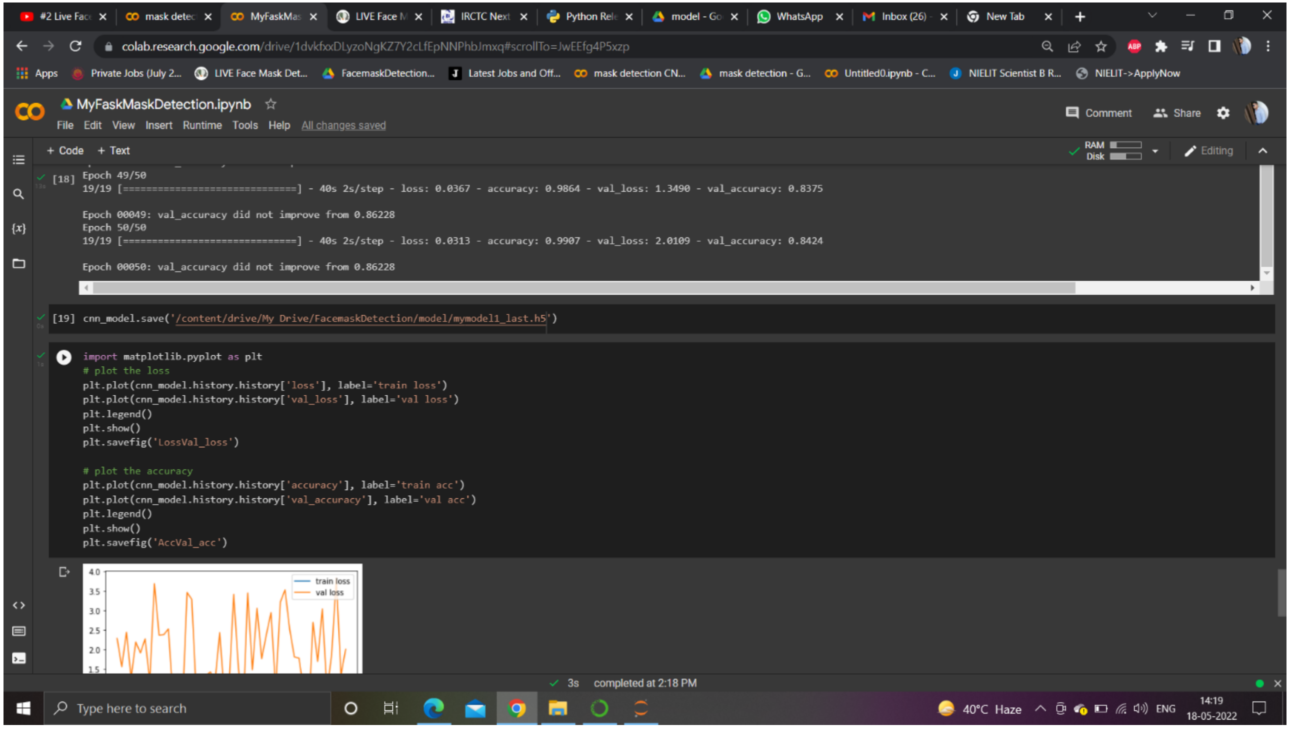


colab.research.google.com/drive/1dvkfxxDLyzoNgKZ7Y2cLfEpNNPhbJmxq#scrollTo=JwEEfg4P5xzp
MyFaskMaskDetection.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
completed at 2:18 PM
Type here to search
40°C Haze
14:19
18-05-2022

```
Epoch 00044: val_accuracy did not improve from 0.86228
[18] Epoch 45/50
19/19 [==============================] - 40s 2s/step - loss: 0.0355 - accuracy: 0.9898 - val_loss: 3.0440 - val_accuracy: 0.8424

Epoch 00045: val_accuracy did not improve from 0.86228
Epoch 46/50
19/19 [==============================] - 40s 2s/step - loss: 0.0326 - accuracy: 0.9885 - val_loss: 0.9216 - val_accuracy: 0.8449

Epoch 00046: val_accuracy did not improve from 0.86228
Epoch 47/50
19/19 [==============================] - 40s 2s/step - loss: 0.0323 - accuracy: 0.9894 - val_loss: 1.8756 - val_accuracy: 0.8486

Epoch 00047: val_accuracy did not improve from 0.86228
Epoch 48/50
19/19 [==============================] - 40s 2s/step - loss: 0.0446 - accuracy: 0.9839 - val_loss: 3.8283 - val_accuracy: 0.8300

Epoch 00048: val_accuracy did not improve from 0.86228
Epoch 49/50
19/19 [==============================] - 40s 2s/step - loss: 0.0367 - accuracy: 0.9864 - val_loss: 1.3490 - val_accuracy: 0.8375

Epoch 00049: val_accuracy did not improve from 0.86228
Epoch 50/50
19/19 [==============================] - 40s 2s/step - loss: 0.0313 - accuracy: 0.9907 - val_loss: 2.0109 - val_accuracy: 0.8424

Epoch 00050: val_accuracy did not improve from 0.86228
```

```
[19] cnn_model.save('/content/drive/My Drive/FacemaskDetection/model/mymodel1_last.h5')
```

```
[20] import matplotlib.pyplot as plt
     # plot the loss
     plt.plot(cnn_model.history.history['loss'], label='train loss')
     plt.plot(cnn_model.history.history['val_loss'], label='val loss')
     plt.legend()
     plt.show()
     plt.savefig('LossVal_loss')
```

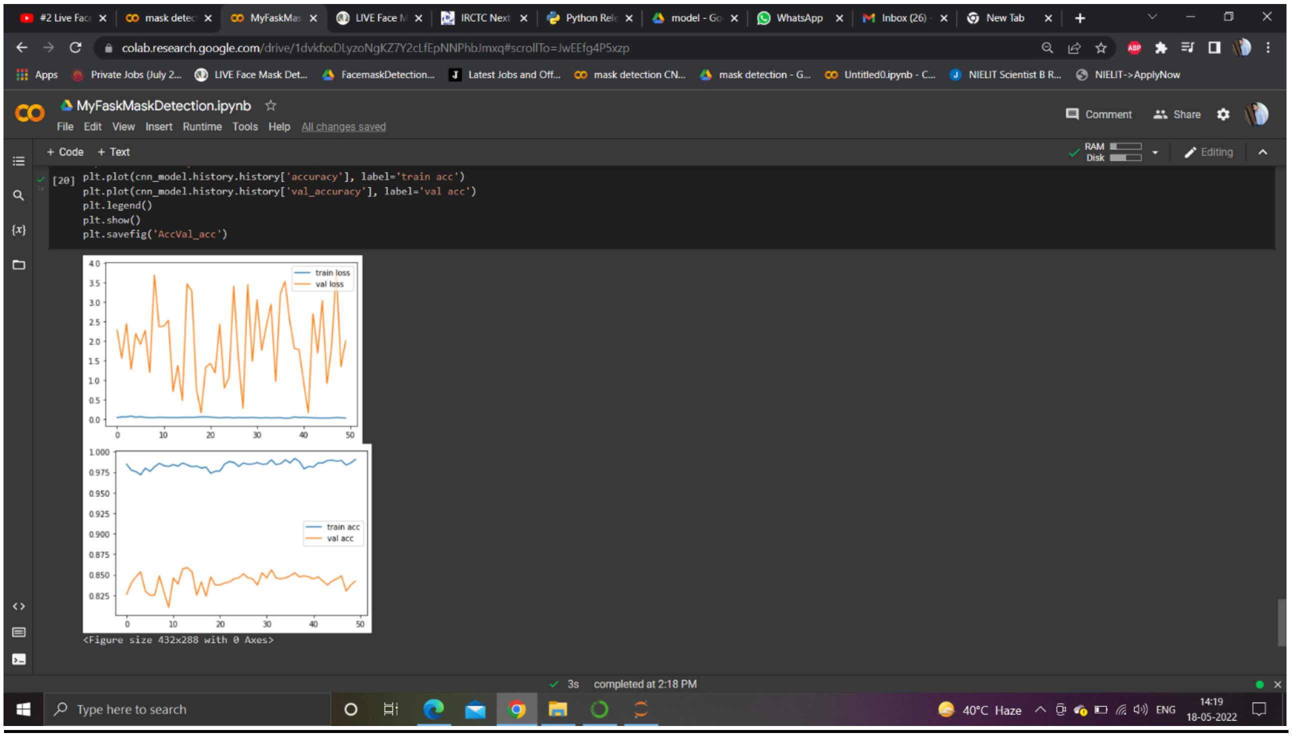


colab.research.google.com/drive/1dvkfxxDLyzoNgKZ7Y2cLfEpNNPhbJmxq#scrollTo=JwEEfg4P5xzp
MyFaskMaskDetection.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
4.0
3.5
3.0
2.5
2.0
1.5
train loss
val loss
completed at 2:18 PM
Type here to search
40°C Haze
14:19
18-05-2022

```
[18] Epoch 49/50
19/19 [==============================] - 40s 2s/step - loss: 0.0367 - accuracy: 0.9864 - val_loss: 1.3490 - val_accuracy: 0.8375

Epoch 00049: val_accuracy did not improve from 0.86228
Epoch 50/50
19/19 [==============================] - 40s 2s/step - loss: 0.0313 - accuracy: 0.9907 - val_loss: 2.0109 - val_accuracy: 0.8424

Epoch 00050: val_accuracy did not improve from 0.86228
```

```
[19] cnn_model.save('/content/drive/My Drive/FacemaskDetection/model/mymodel1_last.h5')
```

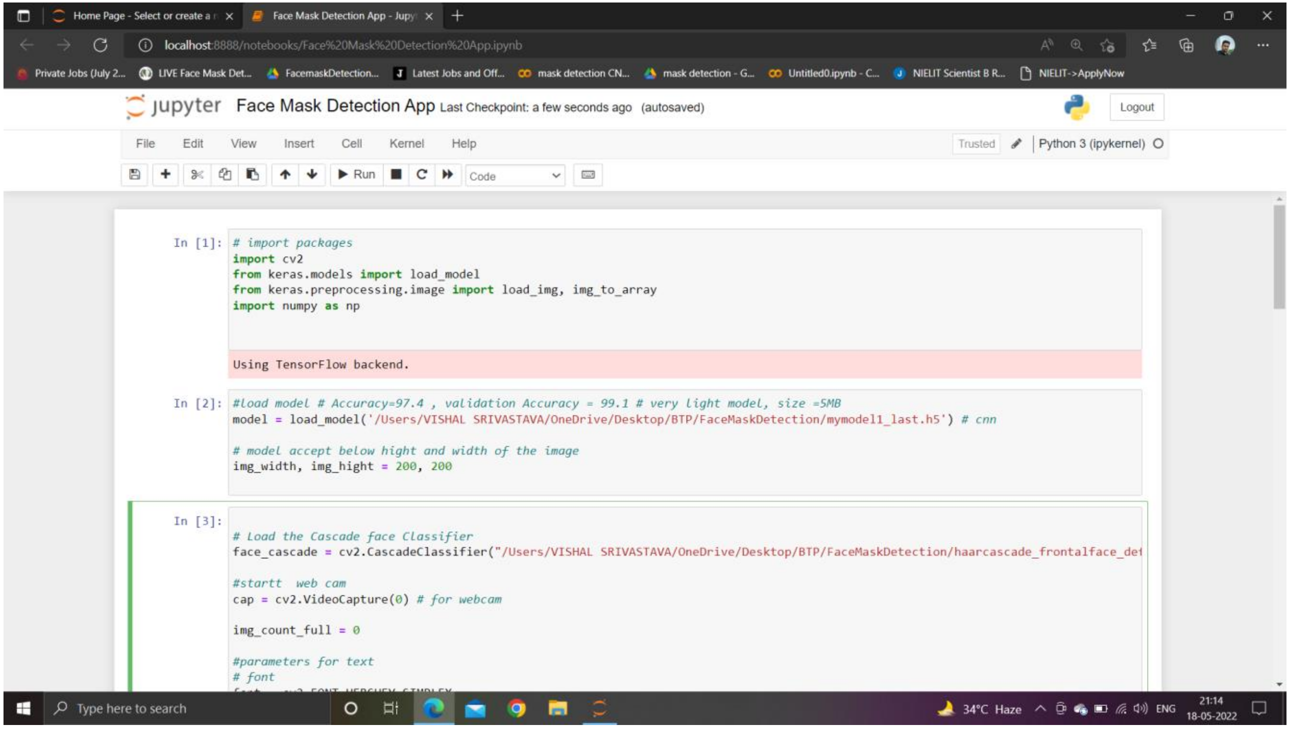
```
import matplotlib.pyplot as plt
# plot the loss
plt.plot(cnn_model.history.history['loss'], label='train loss')
plt.plot(cnn_model.history.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(cnn_model.history.history['accuracy'], label='train acc')
plt.plot(cnn_model.history.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```



```
[20] plt.plot(cnn_model.history.history['accuracy'], label='train acc')
     plt.plot(cnn_model.history.history['val_accuracy'], label='val acc')
     plt.legend()
     plt.show()
     plt.savefig('AccVal_acc')
```

```
<Figure size 432x288 with 0 Axes>
```

2.Face Mask Detection Application

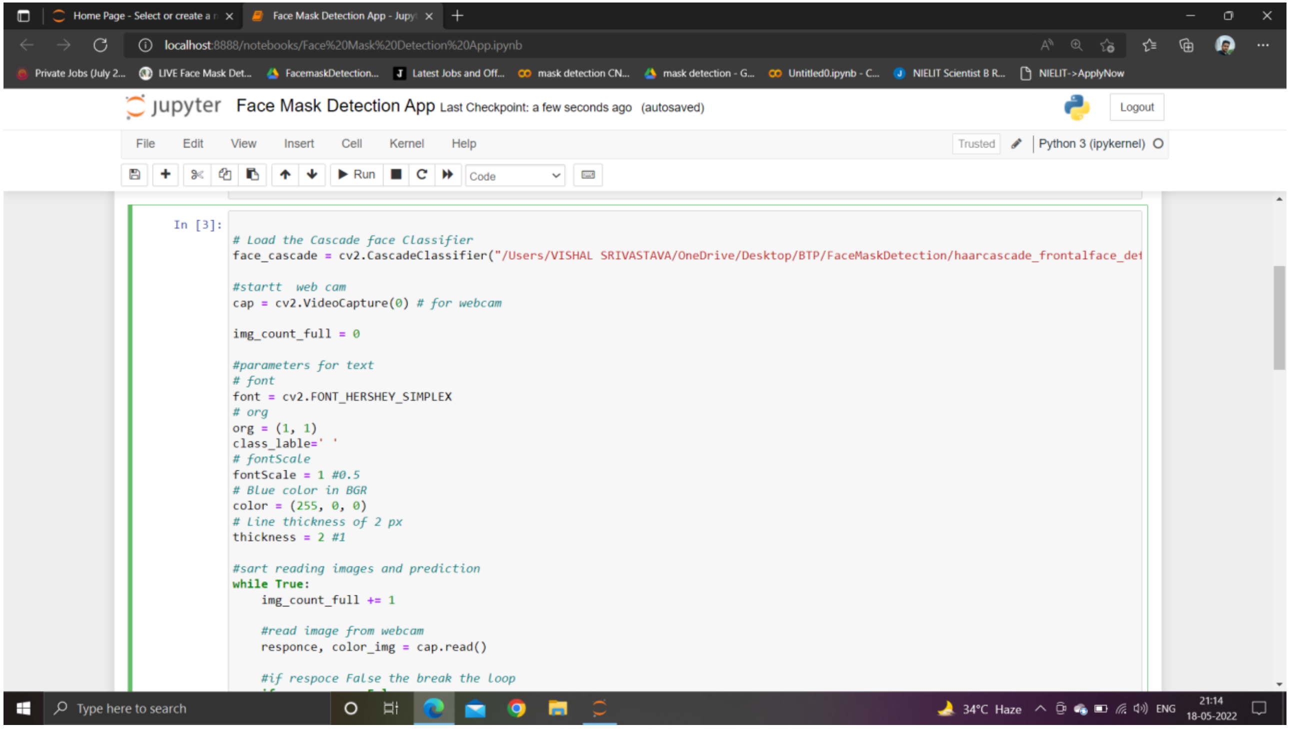

Home Page - Select or create a r
Face Mask Detection App - Jupy
localhost:8888/notebooks/Face%20Mask%20Detection%20App.ipynb
jupyter Face Mask Detection App Last Checkpoint: a few seconds ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3 (ipykernel)
Run
Code
Type here to search
34°C Haze
ENG
21:14
18-05-2022

In [1]:
```
# import packages
import cv2
from keras.models import load_model
from keras.preprocessing.image import load_img, img_to_array
import numpy as np
```

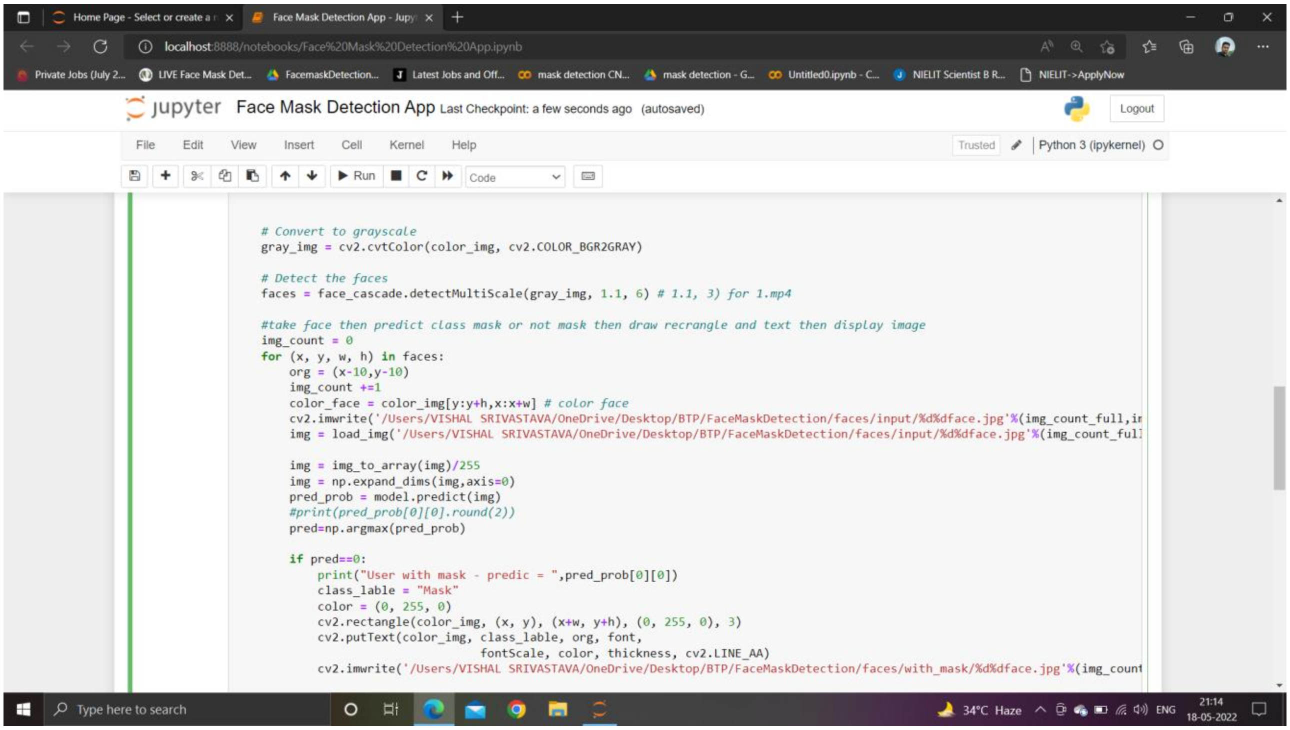
```
Using TensorFlow backend.
```

In [2]:
```
#load model # Accuracy=97.4 , validation Accuracy = 99.1 # very light model, size =5MB
model = load_model('/Users/VISHAL SRIVASTAVA/OneDrive/Desktop/BTP/FaceMaskDetection/mymodel1_last.h5') # cnn

# model accept below hight and width of the image
img_width, img_hight = 200, 200
```

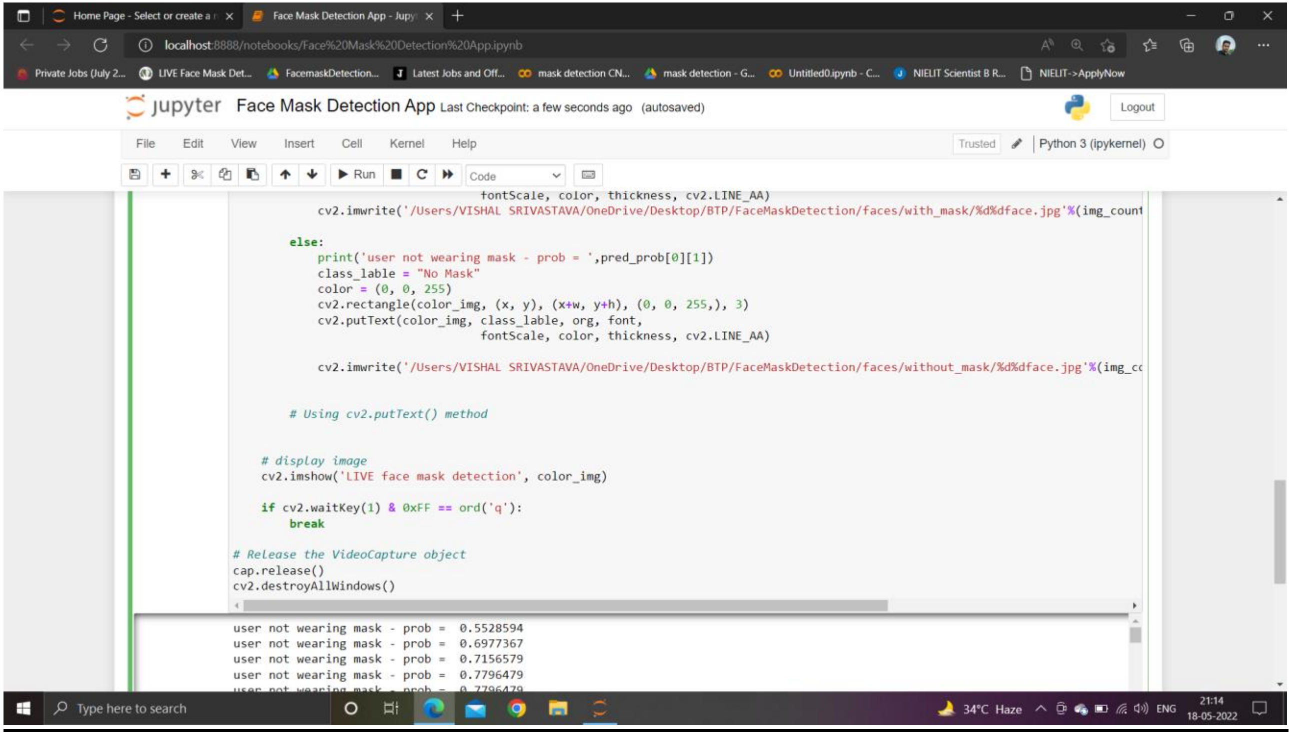
In [3]:
```
# Load the Cascade face Classifier
face_cascade = cv2.CascadeClassifier("/Users/VISHAL SRIVASTAVA/OneDrive/Desktop/BTP/FaceMaskDetection/haarcascade_frontalface_def

#startt  web cam
cap = cv2.VideoCapture(0) # for webcam

img_count_full = 0

#parameters for text
# font
font = cv2.FONT_HERSHEY_SIMPLEX
```


Home Page - Select or create a r
Face Mask Detection App - Jupy
localhost:8888/notebooks/Face%20Mask%20Detection%20App.ipynb
jupyter Face Mask Detection App Last Checkpoint: a few seconds ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3 (ipykernel)
Run
Code
Type here to search
34°C Haze
ENG
21:14
18-05-2022

In [3]:
```
# Load the Cascade face Classifier
face_cascade = cv2.CascadeClassifier("/Users/VISHAL SRIVASTAVA/OneDrive/Desktop/BTP/FaceMaskDetection/haarcascade_frontalface_def

#startt  web cam
cap = cv2.VideoCapture(0) # for webcam

img_count_full = 0

#parameters for text
# font
font = cv2.FONT_HERSHEY_SIMPLEX
# org
org = (1, 1)
class_lable=' '
# fontScale
fontScale = 1 #0.5
# Blue color in BGR
color = (255, 0, 0)
# Line thickness of 2 px
thickness = 2 #1

#sart reading images and prediction
while True:
    img_count_full += 1

    #read image from webcam
    responce, color_img = cap.read()

    #if respoce False the break the loop
```

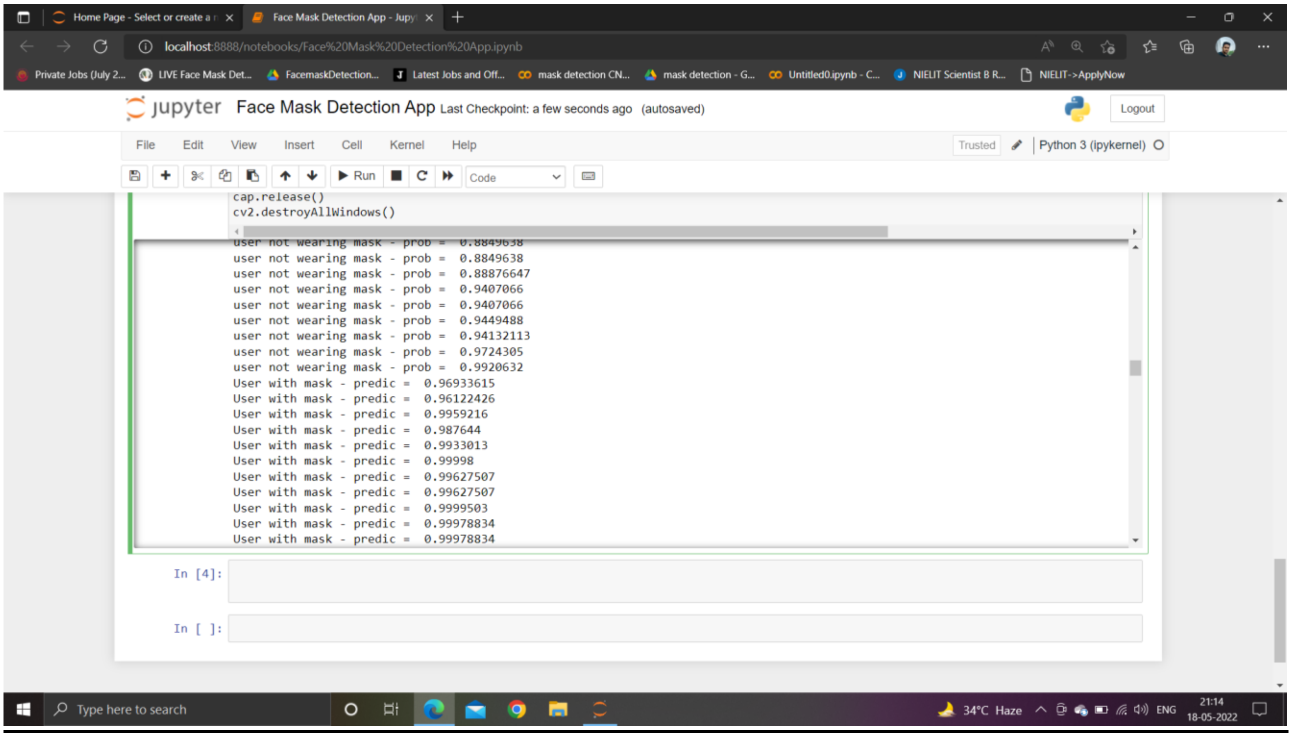


```
    # Convert to grayscale
    gray_img = cv2.cvtColor(color_img, cv2.COLOR_BGR2GRAY)

    # Detect the faces
    faces = face_cascade.detectMultiScale(gray_img, 1.1, 6) # 1.1, 3) for 1.mp4

    #take face then predict class mask or not mask then draw recrangle and text then display image
    img_count = 0
    for (x, y, w, h) in faces:
        org = (x-10,y-10)
        img_count +=1
        color_face = color_img[y:y+h,x:x+w] # color face
        cv2.imwrite('/Users/VISHAL SRIVASTAVA/OneDrive/Desktop/BTP/FaceMaskDetection/faces/input/%d%dface.jpg'%(img_count_full,in
        img = load_img('/Users/VISHAL SRIVASTAVA/OneDrive/Desktop/BTP/FaceMaskDetection/faces/input/%d%dface.jpg'%(img_count_full

        img = img_to_array(img)/255
        img = np.expand_dims(img,axis=0)
        pred_prob = model.predict(img)
        #print(pred_prob[0][0].round(2))
        pred=np.argmax(pred_prob)

        if pred==0:
            print("User with mask - predic = ",pred_prob[0][0])
            class_lable = "Mask"
            color = (0, 255, 0)
            cv2.rectangle(color_img, (x, y), (x+w, y+h), (0, 255, 0), 3)
            cv2.putText(color_img, class_lable, org, font,
                                   fontScale, color, thickness, cv2.LINE_AA)
            cv2.imwrite('/Users/VISHAL SRIVASTAVA/OneDrive/Desktop/BTP/FaceMaskDetection/faces/with_mask/%d%dface.jpg'%(img_count
```


Home Page - Select or create a n
Face Mask Detection App - Jupyt
localhost:8888/notebooks/Face%20Mask%20Detection%20App.ipynb
jupyter
Face Mask Detection App
Last Checkpoint: a few seconds ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Trusted
Python 3 (ipykernel)
Run
Code
Type here to search
34°C Haze
ENG
21:14
18-05-2022

```
                                   fontScale, color, thickness, cv2.LINE_AA)
            cv2.imwrite('/Users/VISHAL SRIVASTAVA/OneDrive/Desktop/BTP/FaceMaskDetection/faces/with_mask/%d%dface.jpg'%(img_count

        else:
            print('user not wearing mask - prob = ',pred_prob[0][1])
            class_lable = "No Mask"
            color = (0, 0, 255)
            cv2.rectangle(color_img, (x, y), (x+w, y+h), (0, 0, 255,), 3)
            cv2.putText(color_img, class_lable, org, font,
                                   fontScale, color, thickness, cv2.LINE_AA)

            cv2.imwrite('/Users/VISHAL SRIVASTAVA/OneDrive/Desktop/BTP/FaceMaskDetection/faces/without_mask/%d%dface.jpg'%(img_cc


        # Using cv2.putText() method


    # display image
    cv2.imshow('LIVE face mask detection', color_img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the VideoCapture object
cap.release()
cv2.destroyAllWindows()
```

```
user not wearing mask - prob =  0.5528594
user not wearing mask - prob =  0.6977367
user not wearing mask - prob =  0.7156579
user not wearing mask - prob =  0.7796479
user not wearing mask - prob =  0.7796479
```

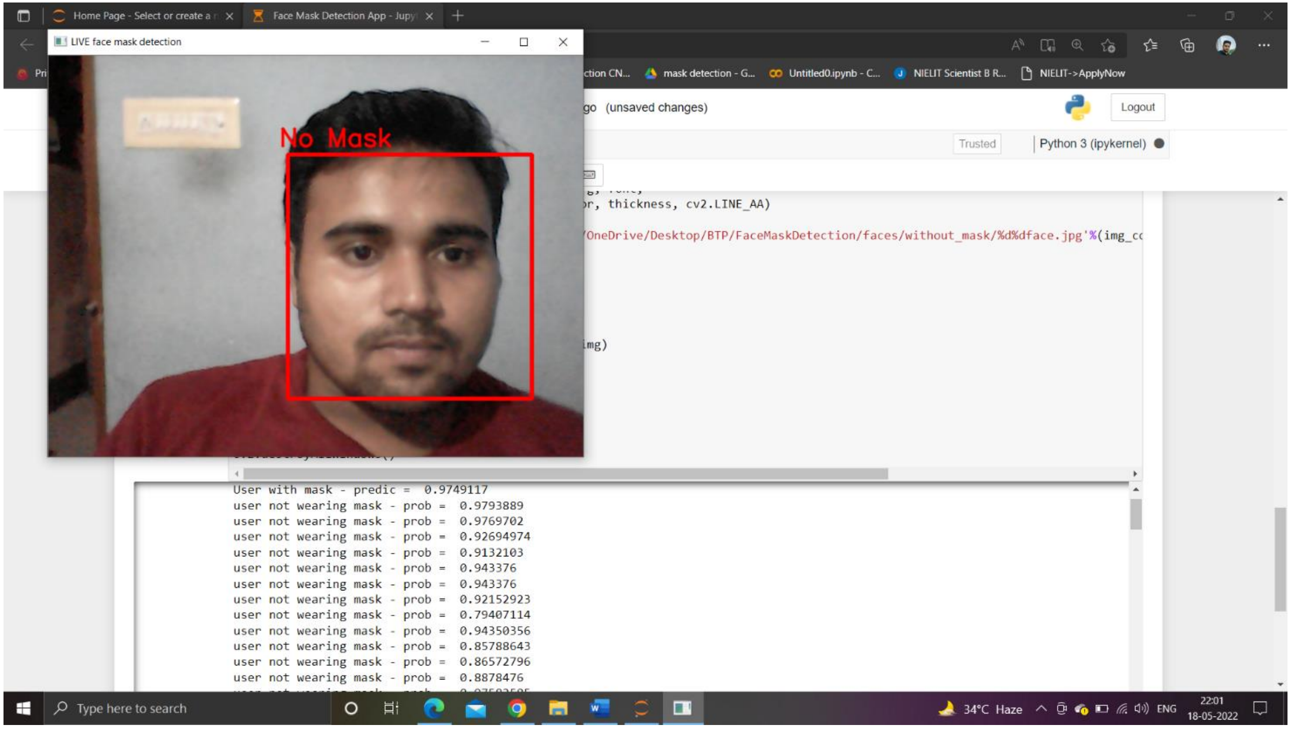


```
cap.release()
cv2.destroyAllWindows()
```

```
user not wearing mask - prob =  0.8849638
user not wearing mask - prob =  0.8849638
user not wearing mask - prob =  0.88876647
user not wearing mask - prob =  0.9407066
user not wearing mask - prob =  0.9407066
user not wearing mask - prob =  0.9449488
user not wearing mask - prob =  0.94132113
user not wearing mask - prob =  0.9724305
user not wearing mask - prob =  0.9920632
User with mask - predic =  0.96933615
User with mask - predic =  0.96122426
User with mask - predic =  0.9959216
User with mask - predic =  0.987644
User with mask - predic =  0.9933013
User with mask - predic =  0.99998
User with mask - predic =  0.99627507
User with mask - predic =  0.99627507
User with mask - predic =  0.9999503
User with mask - predic =  0.99978834
User with mask - predic =  0.99978834
```

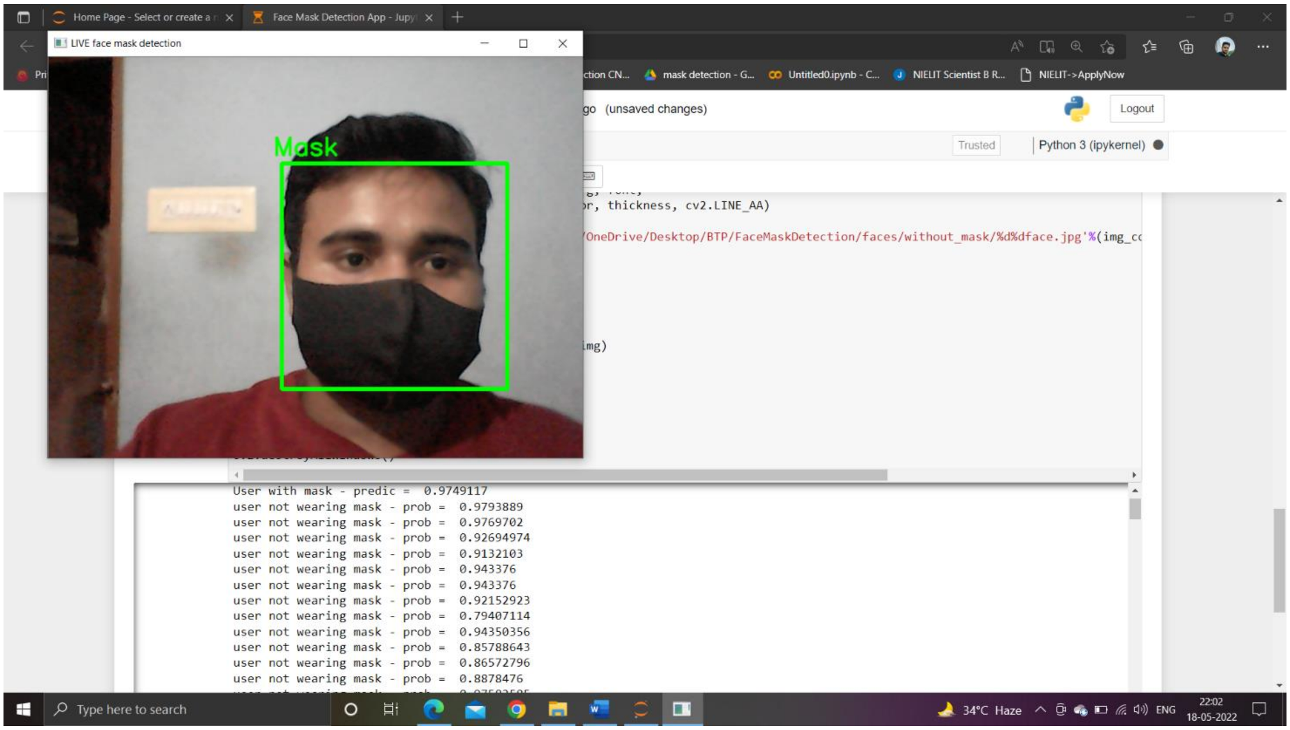
```
In [4]:

In [ ]:
```

3. Result:



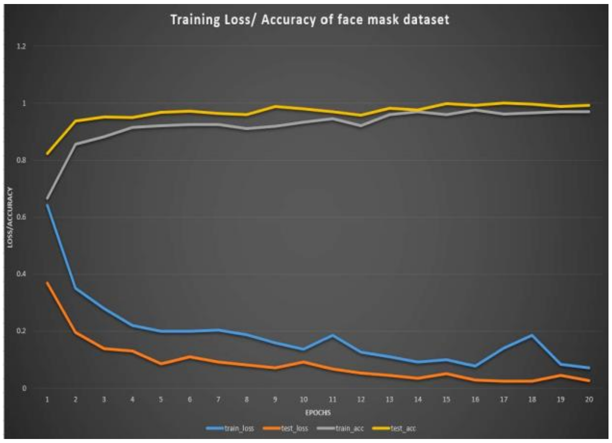
```
r, thickness, cv2.LINE_AA)
/OneDrive/Desktop/BTP/FaceMaskDetection/faces/without_mask/%d%dface.jpg'%(img_c
img)
```

```
User with mask - predic =  0.9749117
user not wearing mask - prob =  0.9793889
user not wearing mask - prob =  0.9769702
user not wearing mask - prob =  0.92694974
user not wearing mask - prob =  0.9132103
user not wearing mask - prob =  0.943376
user not wearing mask - prob =  0.943376
user not wearing mask - prob =  0.92152923
user not wearing mask - prob =  0.79407114
user not wearing mask - prob =  0.94350356
user not wearing mask - prob =  0.85788643
user not wearing mask - prob =  0.86572796
user not wearing mask - prob =  0.8878476
```



```
r, thickness, cv2.LINE_AA)
/OneDrive/Desktop/BTP/FaceMaskDetection/faces/without_mask/%d%dface.jpg'%(img_c
img)
```

```
User with mask - predic =  0.9749117
user not wearing mask - prob =  0.9793889
user not wearing mask - prob =  0.9769702
user not wearing mask - prob =  0.92694974
user not wearing mask - prob =  0.9132103
user not wearing mask - prob =  0.943376
user not wearing mask - prob =  0.943376
user not wearing mask - prob =  0.92152923
user not wearing mask - prob =  0.79407114
user not wearing mask - prob =  0.94350356
user not wearing mask - prob =  0.85788643
user not wearing mask - prob =  0.86572796
user not wearing mask - prob =  0.8878476
```



As we have collected dataset and train our model using Supervised learning algorithm CNN to detect if a person wearing a mask or not.

And using epochs=50 we acquire more and more accuracy. And by keep tracking of values we save our model wherever we got best accuracy and minimum loss.

4. Conclusion:

As the technology are blooming with emerging trends the availability so we have novel face mask detector which can possibly contribute to public health care department. The architecture consists of MobileNetV2 classifier and ADAM optimizer as the backbone it can be used for high and low computation scenarios. The our face mask detection is trained on CNN model and we are used Oven CV, Tensor Flow, Keras and python to detect whether person is wearing a mask or not . The model was tested with image and real- time video stream. The accuracy of model is achieved and, the optimization of the model is continuous process. This specific model could be used as use case of edge analytics.

5. References:

[1] B. QIN and D. Li, Identifying facemask-wearing condition using image super-resolution with classification network to prevent COVID-19, May 2020, doi: 10.21203/rs.3.rs-28668/v1.

[2] M.S. Ejaz, M.R. Islam, M. Sifatullah, A. SarkerImplementation of principal component analysis on masked and non-masked face recognition 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT) (2019), pp. 15, 10.1109/ICASERT.2019.8934543

[3] Jeong-Seon Park, You Hwa Oh, Sang ChulAhn, and Seong- Whan Lee, Glasses removal from facial image using recursive error compensation, IEEE Trans. Pattern Anal. Mach. Intell. 27 (5) (2005) 805–811, doi: 10.1109/TPAMI.2005.103.

[4] C. Li, R. Wang, J. Li, L. Fei, Face detection based on YOLOv3, in:: Recent Trends in Intelligent Computing, Communication and Devices, Singapore, 2020, pp. 277–284, doi: 10.1007/978-981-13- 9406-5_34.

[5] N. Ud Din, K. Javed, S. Bae, J. YiA novel GAN-based network for unmasking of masked face IEEE Access, 8 (2020), pp. 4427644287, 10.1109/ACCESS.2020.2977386

[6] A. Nieto-Rodríguez, M. Mucientes, V.M. BreaSystem for medical mask detection in the operating room through facial attributes Pattern Recogn. Image Anal. Cham (2015), pp. 138- 145, 10.1007/978-3-319-19390-8_16

[7] S. A. Hussain, A.S.A.A. Balushi, A real time face emotion classification and recognition using deep learning model, J. Phys.: Conf. Ser. 1432 (2020) 012087, doi: 10.1088/1742- 6596/1432/1/012087.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.

[9] Xin, M., Wang, Y. Research on image classification model based on deep convolution neural network. J Image Video Proc. 2019, 40 (2019).

[10] Sultana, F., A. Sufian, and P. Dutta. "A review of object detection models

based on convolutional neural network." arXiv preprint arXiv:1905.01614 (2019).

[11] X. Jia, "Image recognition method based on deep learning," 2017 29th Chinese Control And Decision Conference (CCDC), Chongqing, 2017, pp. 4730-4735, doi:10.1109/CCDC.2017.7979332.

[12] MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications - Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam.

[13] Adaptive Subgradient Methods for Online Learning and Stochastic Optimization - John Duchi, EladHazan, Yoram Singer.