Information Security Project Report

"Face Mask Detection"

Jaypee Institute of Information Technology

Submitted by

Ananya Kapoor	*20103104*	*B4*
Dhairya Sachdeva	*20103098*	*B4*

Submitted To

Dr. Purtee Kohli

DECLARATION

We hereby declare that the project titled *“Face Mask Detection”* submitted for the degree of Bachelors in Technology is a bonafide record submitted to Jaypee Institute of Information Technology, under the guidance of Dr. Purtee Kohli, has been carried out by our own efforts and is a record of our original work.

1. Abstract

Face masks help reduce the transmission of SARS-CoV-2 by interfering with the spread of virus-laden droplets ejected from the nose and mouth. Wearing face mask is one of the precautionary steps an individual can take in order to lessen the spread of COVID-19. In this project, a video camera detects if an individual is wearing a face mask or not in real-time.

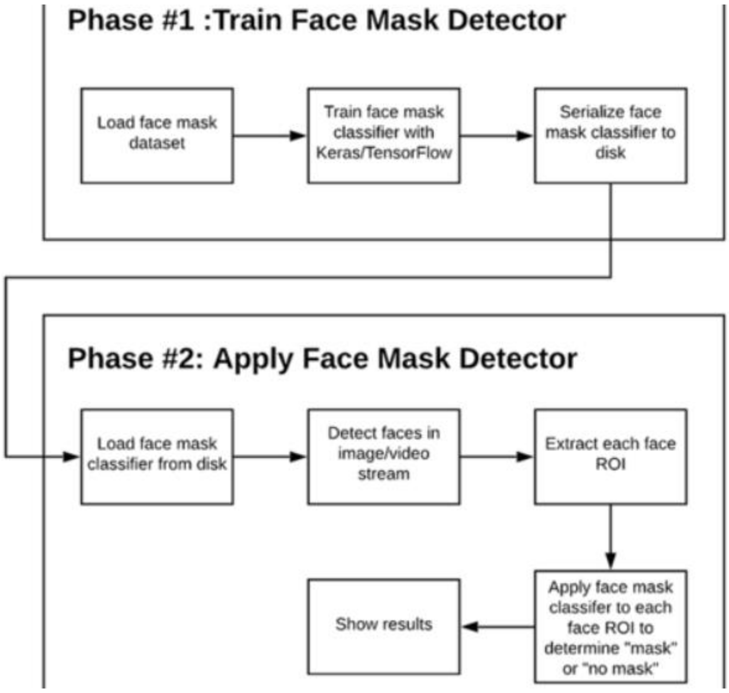
2. Introduction

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. Our project builds a real time model which detects if a person if wearing a face mask or not.

A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision. CNN can run directly on a underdone image and do not need any pre-processing. A convolutional neural network is a feed forward neural network, seldom with up to 20. The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer. CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes. With three or four convolutional layers it is viable to recognize handwritten digits and with 25 layers it is possible to differentiate human faces.

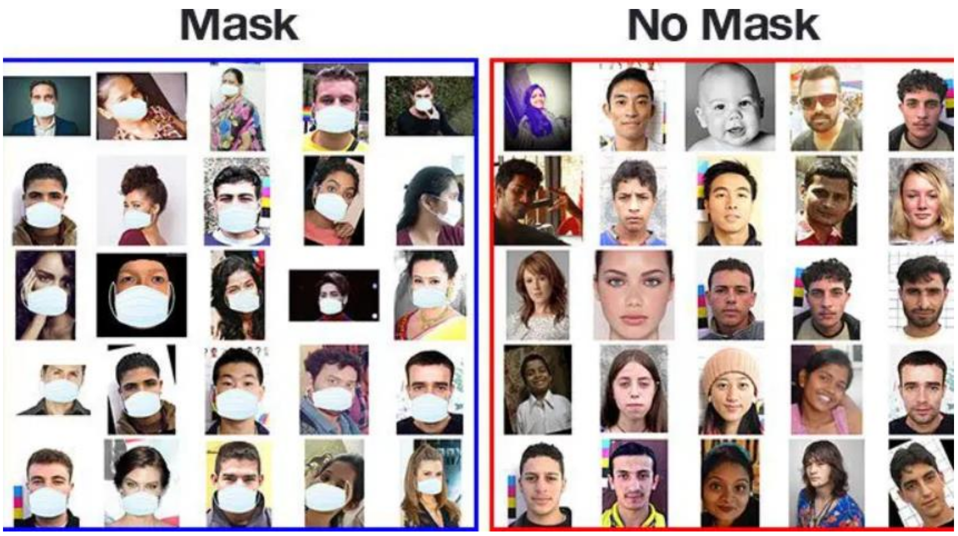
Object Detection is a computer technology related to computer vision, image processing and deep learning that deals with detecting instances of objects in images and videos. We have used Haar Cascade classifiers. Haar cascade is an algorithm that can detect objects in images, irrespective of their scale in image and location. This algorithm is not so complex and can run in real-time. It is a machine learning-based approach where a lot of positive and negative images are used to train the classifier.

Flow Chart of Face Mask Detection System:

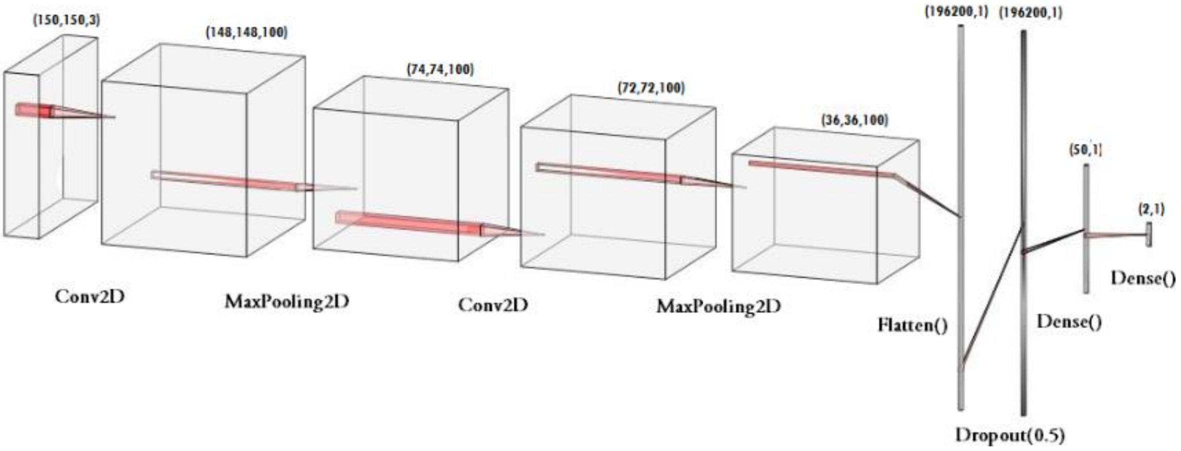

Phase #1 :Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifer to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

In order to train a custom face mask detector, we need to break our project into two distinct phases, each with its own respective sub-steps (as shown by Figure 1 above):

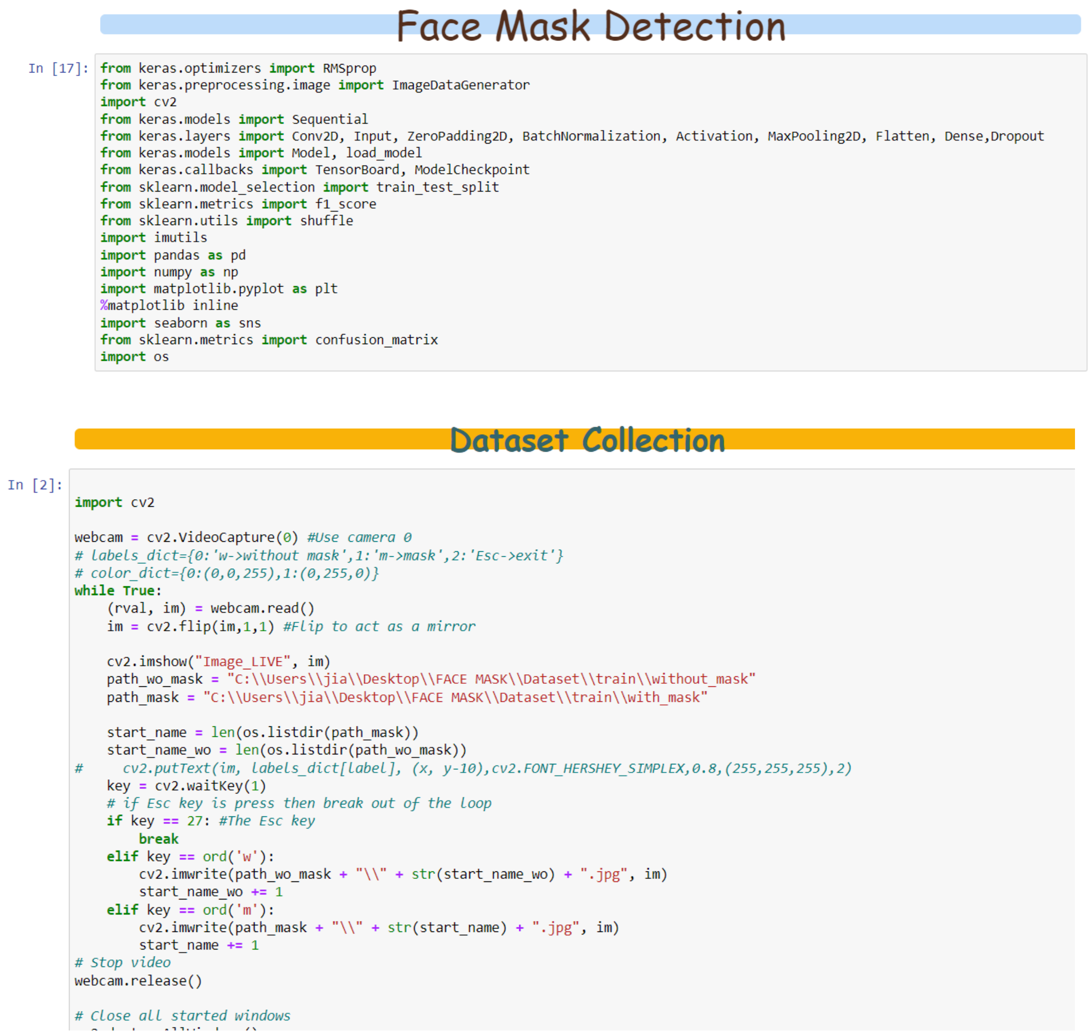
- Training: Here we'll focus on loading our face mask detection dataset from disk, training a model (using Keras/TensorFlow) on this dataset, and then serializing the face mask detector to disk
- Deployment: Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as with_mask or without_mask.



We have used these images to build a CNN model using TensorFlow to detect if you are wearing a face mask by using the webcam of your PC.



3. Code and Result

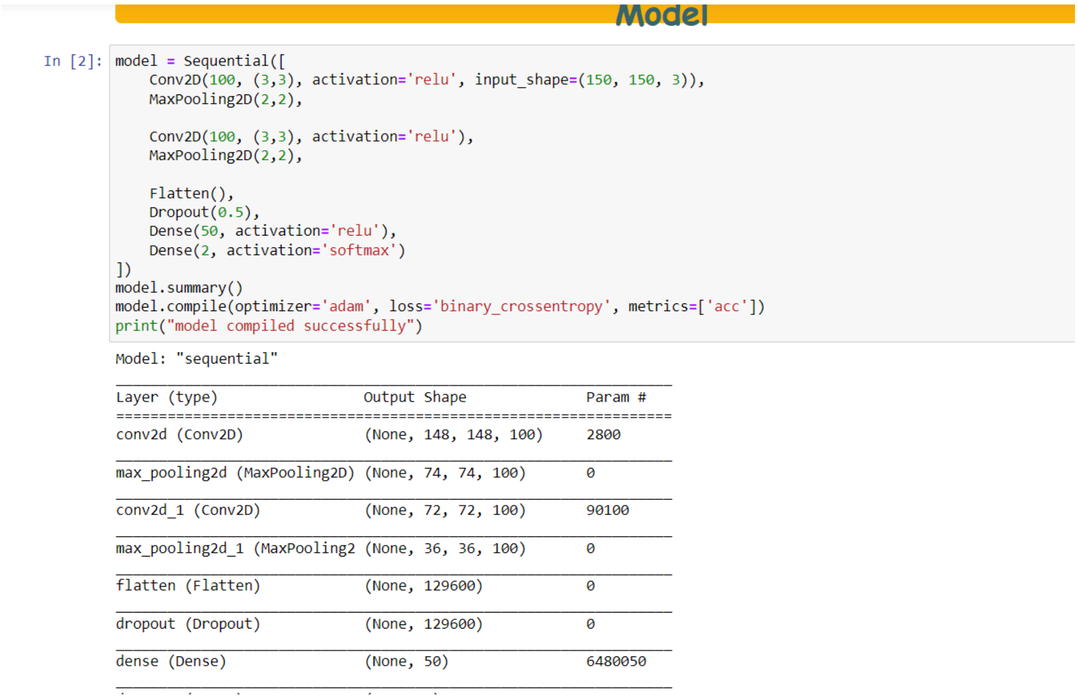


Face Mask Detection

In [17]:
```
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
import cv2
from keras.models import Sequential
from keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense,Dropout
from keras.models import Model, load_model
from keras.callbacks import TensorBoard, ModelCheckpoint
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.utils import shuffle
import imutils
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.metrics import confusion_matrix
import os
```

Dataset Collection

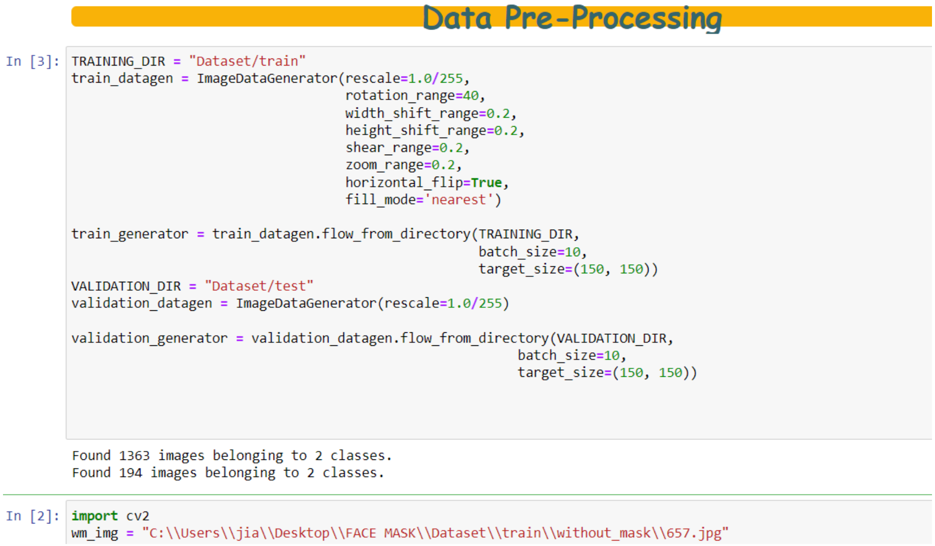
In [2]:
```

import cv2

webcam = cv2.VideoCapture(0) #Use camera 0
# labels_dict={0:'w->without mask',1:'m->mask',2:'Esc->exit'}
# color_dict={0:(0,0,255),1:(0,255,0)}
while True:
    (rval, im) = webcam.read()
    im = cv2.flip(im,1,1) #Flip to act as a mirror

    cv2.imshow("Image_LIVE", im)
    path_wo_mask = "C:\\Users\\jia\\Desktop\\FACE MASK\\Dataset\\train\\without_mask"
    path_mask = "C:\\Users\\jia\\Desktop\\FACE MASK\\Dataset\\train\\with_mask"

    start_name = len(os.listdir(path_mask))
    start_name_wo = len(os.listdir(path_wo_mask))
#     cv2.putText(im, labels_dict[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)
    key = cv2.waitKey(1)
    # if Esc key is press then break out of the loop
    if key == 27: #The Esc key
        break
    elif key == ord('w'):
        cv2.imwrite(path_wo_mask + "\\" + str(start_name_wo) + ".jpg", im)
        start_name_wo += 1
    elif key == ord('m'):
        cv2.imwrite(path_mask + "\\" + str(start_name) + ".jpg", im)
        start_name += 1
# Stop video
webcam.release()

# Close all started windows
```



Model

```
In [2]: model = Sequential([
            Conv2D(100, (3,3), activation='relu', input_shape=(150, 150, 3)),
            MaxPooling2D(2,2),

            Conv2D(100, (3,3), activation='relu'),
            MaxPooling2D(2,2),

            Flatten(),
            Dropout(0.5),
            Dense(50, activation='relu'),
            Dense(2, activation='softmax')
        ])
        model.summary()
        model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
        print("model compiled successfully")
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 148, 148, 100)     2800      
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 100)       0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 100)       90100     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 100)       0         
_________________________________________________________________
flatten (Flatten)            (None, 129600)            0         
_________________________________________________________________
dropout (Dropout)            (None, 129600)            0         
_________________________________________________________________
dense (Dense)                (None, 50)                6480050   
_________________________________________________________________
```

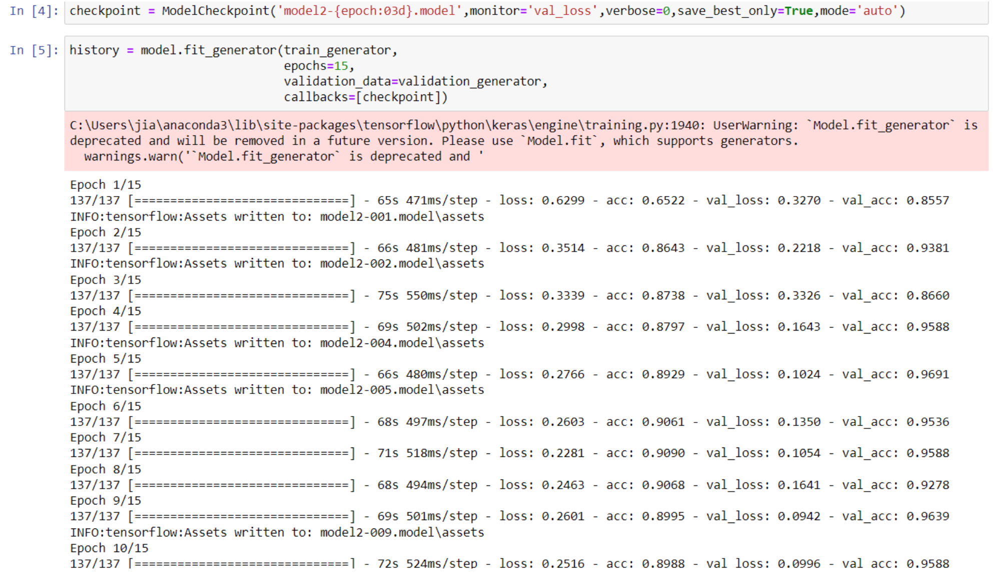


Data Pre-Processing

```
In [3]: TRAINING_DIR = "Dataset/train"
        train_datagen = ImageDataGenerator(rescale=1.0/255,
                                           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')

        train_generator = train_datagen.flow_from_directory(TRAINING_DIR,
                                                            batch_size=10,
                                                            target_size=(150, 150))
        VALIDATION_DIR = "Dataset/test"
        validation_datagen = ImageDataGenerator(rescale=1.0/255)

        validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR,
                                                                 batch_size=10,
                                                                 target_size=(150, 150))
```

```
Found 1363 images belonging to 2 classes.
Found 194 images belonging to 2 classes.
```

```
In [2]: import cv2
        wm_img = "C:\\Users\\jia\\Desktop\\FACE MASK\\Dataset\\train\\without_mask\\657.jpg"
```



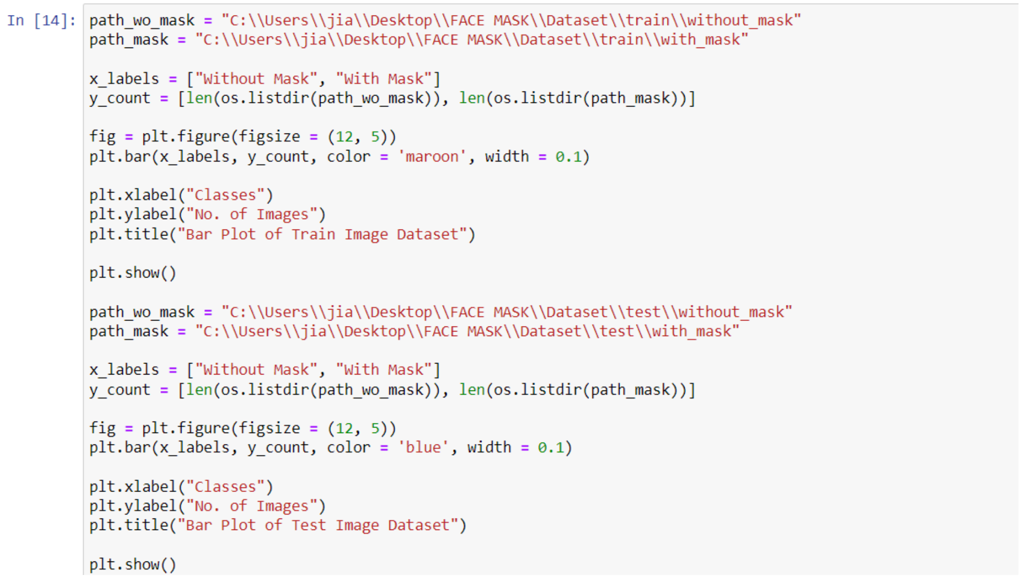
```
In [4]: checkpoint = ModelCheckpoint('model2-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
```

```
In [5]: history = model.fit_generator(train_generator,
                                      epochs=15,
                                      validation_data=validation_generator,
                                      callbacks=[checkpoint])
```

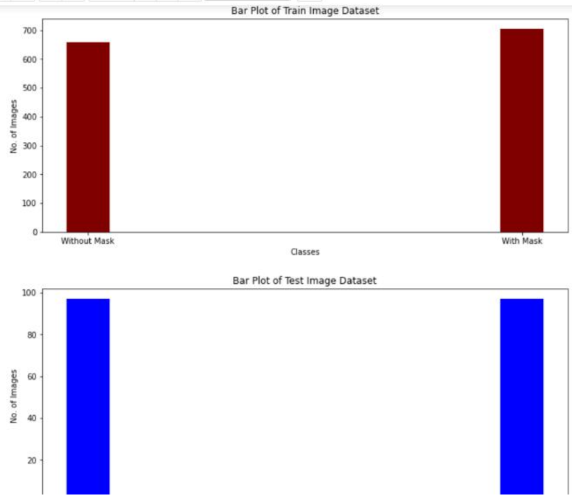
```
C:\Users\jia\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py:1940: UserWarning: `Model.fit_generator` is
deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

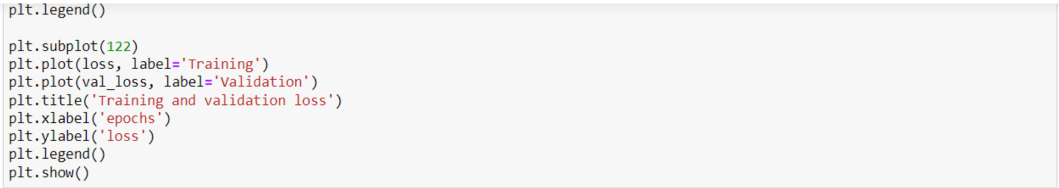
```
Epoch 1/15
137/137 [==============================] - 65s 471ms/step - loss: 0.6299 - acc: 0.6522 - val_loss: 0.3270 - val_acc: 0.8557
INFO:tensorflow:Assets written to: model2-001.model\assets
Epoch 2/15
137/137 [==============================] - 66s 481ms/step - loss: 0.3514 - acc: 0.8643 - val_loss: 0.2218 - val_acc: 0.9381
INFO:tensorflow:Assets written to: model2-002.model\assets
Epoch 3/15
137/137 [==============================] - 75s 550ms/step - loss: 0.3339 - acc: 0.8738 - val_loss: 0.3326 - val_acc: 0.8660
Epoch 4/15
137/137 [==============================] - 69s 502ms/step - loss: 0.2998 - acc: 0.8797 - val_loss: 0.1643 - val_acc: 0.9588
INFO:tensorflow:Assets written to: model2-004.model\assets
Epoch 5/15
137/137 [==============================] - 66s 480ms/step - loss: 0.2766 - acc: 0.8929 - val_loss: 0.1024 - val_acc: 0.9691
INFO:tensorflow:Assets written to: model2-005.model\assets
Epoch 6/15
137/137 [==============================] - 68s 497ms/step - loss: 0.2603 - acc: 0.9061 - val_loss: 0.1350 - val_acc: 0.9536
Epoch 7/15
137/137 [==============================] - 71s 518ms/step - loss: 0.2281 - acc: 0.9090 - val_loss: 0.1054 - val_acc: 0.9588
Epoch 8/15
137/137 [==============================] - 68s 494ms/step - loss: 0.2463 - acc: 0.9068 - val_loss: 0.1641 - val_acc: 0.9278
Epoch 9/15
137/137 [==============================] - 69s 501ms/step - loss: 0.2601 - acc: 0.8995 - val_loss: 0.0942 - val_acc: 0.9639
INFO:tensorflow:Assets written to: model2-009.model\assets
Epoch 10/15
137/137 [==============================] - 72s 524ms/step - loss: 0.2516 - acc: 0.8988 - val_loss: 0.0996 - val_acc: 0.9588
```

Visualizing

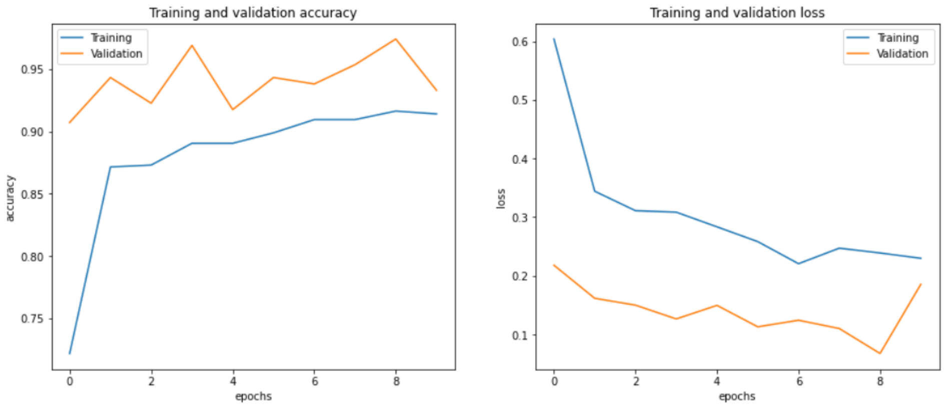


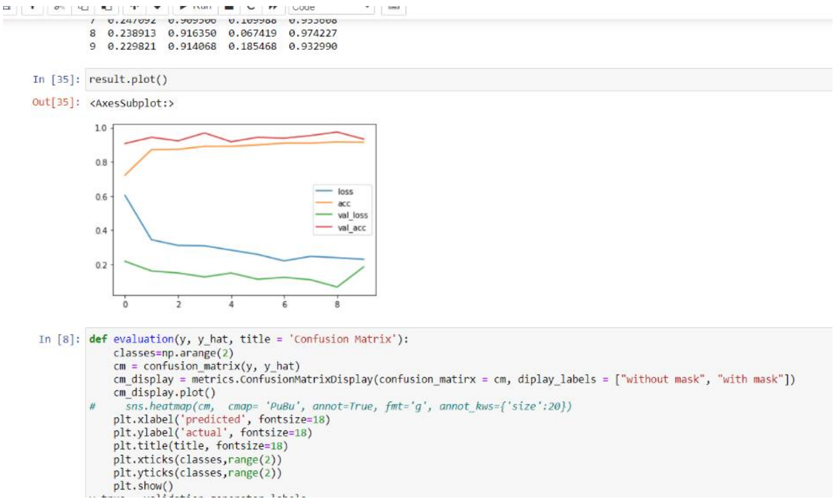
```
In [14]: path_wo_mask = "C:\\Users\\jia\\Desktop\\FACE MASK\\Dataset\\train\\without_mask"
         path_mask = "C:\\Users\\jia\\Desktop\\FACE MASK\\Dataset\\train\\with_mask"

         x_labels = ["Without Mask", "With Mask"]
         y_count = [len(os.listdir(path_wo_mask)), len(os.listdir(path_mask))]

         fig = plt.figure(figsize = (12, 5))
         plt.bar(x_labels, y_count, color = 'maroon', width = 0.1)

         plt.xlabel("Classes")
         plt.ylabel("No. of Images")
         plt.title("Bar Plot of Train Image Dataset")

         plt.show()

         path_wo_mask = "C:\\Users\\jia\\Desktop\\FACE MASK\\Dataset\\test\\without_mask"
         path_mask = "C:\\Users\\jia\\Desktop\\FACE MASK\\Dataset\\test\\with_mask"

         x_labels = ["Without Mask", "With Mask"]
         y_count = [len(os.listdir(path_wo_mask)), len(os.listdir(path_mask))]

         fig = plt.figure(figsize = (12, 5))
         plt.bar(x_labels, y_count, color = 'blue', width = 0.1)

         plt.xlabel("Classes")
         plt.ylabel("No. of Images")
         plt.title("Bar Plot of Test Image Dataset")

         plt.show()
```





```
plt.legend()

plt.subplot(122)
plt.plot(loss, label='Training')
plt.plot(val_loss, label='Validation')
plt.title('Training and validation loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

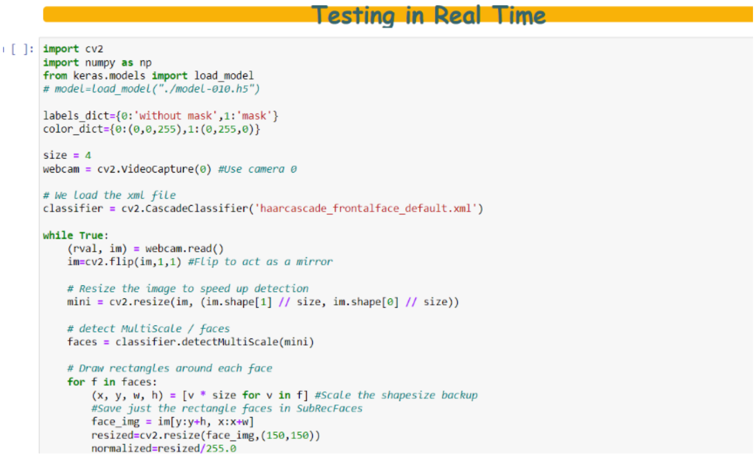




```
7  0.247092  0.909506  0.109988  0.953608
8  0.238913  0.916350  0.067419  0.974227
9  0.229821  0.914068  0.185468  0.932990
```

```
In [35]: result.plot()

Out[35]: <AxesSubplot:>
```

```
In [8]: def evaluation(y, y_hat, title = 'Confusion Matrix'):
            classes=np.arange(2)
            cm = confusion_matrix(y, y_hat)
            cm_display = metrics.ConfusionMatrixDisplay(confusion_matirx = cm, diplay_labels = ["without mask", "with mask"])
            cm_display.plot()
        #     sns.heatmap(cm,  cmap= 'PuBu', annot=True, fmt='g', annot_kws={'size':20})
            plt.xlabel('predicted', fontsize=18)
            plt.ylabel('actual', fontsize=18)
            plt.title(title, fontsize=18)
            plt.xticks(classes,range(2))
            plt.yticks(classes,range(2))
            plt.show()
```



Testing in Real Time

```
[ ]: import cv2
     import numpy as np
     from keras.models import load_model
     # model=load_model("./model-010.h5")

     labels_dict={0:'without mask',1:'mask'}
     color_dict={0:(0,0,255),1:(0,255,0)}

     size = 4
     webcam = cv2.VideoCapture(0) #Use camera 0

     # We load the xml file
     classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

     while True:
         (rval, im) = webcam.read()
         im=cv2.flip(im,1,1) #Flip to act as a mirror

         # Resize the image to speed up detection
         mini = cv2.resize(im, (im.shape[1] // size, im.shape[0] // size))

         # detect MultiScale / faces
         faces = classifier.detectMultiScale(mini)

         # Draw rectangles around each face
         for f in faces:
             (x, y, w, h) = [v * size for v in f] #Scale the shapesize backup
             #Save just the rectangle faces in SubRecFaces
             face_img = im[y:y+h, x:x+w]
             resized=cv2.resize(face_img,(150,150))
             normalized=resized/255.0
```

4. Future Work and Conclusion

This system can 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.

5. References

[1] https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/

[2] https://data-flair.training/blogs/download-face-mask-data/

[3] https://drive.google.com/file/d/1PPO2MCttsmSqyB-vKh5C7SumwFKuhgyj/view