PROJECT REPORT

"REAL TIME FACE MASK DETECTION BY USING DEEP LEARNING MODELS"

Submitted towards the partial fulfillment of the criteria for award of Post Graduate Program in Data Analytics by Imarticus

Submitted By:

*VENUJAYAKANTH. B (*IL021097*)*

Course and Batch: *PGA-04-MAY-2021*

ABSTRACT

Covid-19 cases in India and around the world shows an increasing trend from 2020. Government have been taking several measures to prevent and overcome this global pandemic, wearing mask had become mandatory for all the people who are all come outside of their home. In real time we need a system which detect mask of a person automatically without human intervention and allowing them inside an organizations/institutions/public places etc. To build such as system Computer Vision and Neural networks (especially CNN – Convolution Neural Network) serves better. To serve this task Images of people wearing mask and not wearing mask were taken. A CNN model was built on this image dataset to predict whether a person is wearing a mask or not, it is combined with the computer vision framework to detect face as images in any live streaming web cam. First a CNN model with high accuracy (> 90 per cent) is built on the image dataset, after that by using opencv library face was detected in live streaming web cam as an image and the model is applied on that detected image to predict whether a person is wearing mask or not. In out-of-sample prediction CNN model performed well with above 60 per cent accuracy.

ACKNOWLEDGEMENTS

We are using this opportunity to express our gratitude to everyone who supported us throughout the course of this group project. We are thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. We are sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

Further, we were fortunate to have great teachers who readily shared their immense knowledge in data analytics and guided us in a manner that the outcome resulted in enhancing our data skills. We wish to thank, all the faculties, as this project utilized knowledge gained from every course that formed the PGA program.

We certify that the work done by us for conceptualizing and completing this project is original and authentic.

Date: May, 2021 Venujayakanth B.

Place: Coimbatore

CERTIFICATE OF COMPLETION

I hereby certify that the project titled ***"REAL TIME FACE MASK DETECTION BY USING DEEP LEARNING MODELS"*** was undertaken and completed under my supervision by Venujayakanth B from the batch of PGA-04 (May 2021)

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1. Title & Objective of the study

The Capstone project I worked upon was **"Real Time Face Mask Detection by using Deep Learning Models".** This project comes under Healthcare Domain. In this project I am going to predict whether a person is wearing a mask or not in live streaming web cam. For doing this I have considered an image dataset consist of 100 mask wearing and 100 mask not-wearing person from ages of all variation. OpenCV framework is applied to detect face as images in live streaming video and CNN model is applied on that detected images to predict whether a person is wearing mask or not. It assists engineers to deploy the model in any organizations/ institutions/public places to check people without human intervention. The objective of this study is as follows

1. To build optimum model to predict mask wearing person by CNN model
2. To detect face in Real time by OpenCV framework
3. To apply the optimum CNN model on OpenCV framework to detect mask wearing person

1.1. Need of the Study

Covid-19 cases in India and around the world shows an increasing trend from 2020. Government have been taking several measures to prevent and overcome this global pandemic, wearing mask had become mandatory for all the people who are all come outside of their home. In real time we need a system which detect mask of a person automatically without human intervention and allowing them inside an organizations/institutions/public places etc. To build such as system Computer Vision and Neural networks (especially CNN – Convolution Neural Network) serves better.

1.2. Data Sources

The Data for above study was taken from Prajna Bhandary (https://www.linkedin.com/feed/update/urn/).For analytical purpose 200 dataset consist of people with mask and without mask of all age group variation were taken.

1.3. Tools & Techniques

- Deep learning models – Convolution Neural Network (RNN).
- Computer Vision – OpenCV framework
- Python programming language was used for building models.
- Jupyter Notebook and Google Colab platform was used for programming.

CHAPTER 2: DATA PREPROCESSING

The Data for model building was taken from the Prajna Bhandary (). It consists of nearly 200 images of person with mask and without mask, a separate dataset folder was created in C: drive having two subfolders namely with mask and without mask.

2.1. Importing Dataset

Since the dataset is image and it is stored in local disk, to import that dataset into python Jupyter notebook 'OS' library is utilized. The below figure describes it

```
In [2]: Folder = ['with mask','without mask']
        Folder

Out[2]: ['with mask', 'without mask']

In [3]: Data_path1 = os.listdir('C:/Users/Venu/Desktop/mask dataset/with mask')
        Data_path1

Out[3]: ['0-with-mask - Copy (2).jpg',
         '0-with-mask - Copy.jpg',
         '0-with-mask.jpg',
         '101-with-mask - Copy (2).jpg',
         '101-with-mask - Copy.jpg',
         '101-with-mask.jpg',
         '104-with-mask - Copy (2).jpg',
         '104-with-mask - Copy.jpg',

In [41]: Data_path2 = os.listdir('C:/Users/Venu/Desktop/mask dataset/without mask')
         Data_path2

Out[41]: ['105 - Copy (2).jpg',
          '105 - Copy.jpg',
          '105.jpg',
          '109 - Copy (2).jpg',
          '109 - Copy.jpg',
          '109.jpg',
          '11 - Copy (2) - Copy.jpg',
          '11 - Copy (2).jpg',
          '11 - Copy (3).jpg',
```

The command **os.listdir** list the number of image file in a particular path in local PC. Since the image is stored in two folders as with mask and without mask so two list Data_path1 and Data_path2 was created and it is merged into single list as follows:

```
In [42]: Image_folder = []
         for x in Data_path1:
             Image_folder.append(x)
         for x in Data_path2:
             Image_folder.append(x)
         Image_folder

Out[42]: ['0-with-mask - Copy (2).jpg',
          '0-with-mask - Copy.jpg',
          '0-with-mask.jpg',
          '101-with-mask - Copy (2).jpg',
          '101-with-mask - Copy.jpg',
          '101-with-mask.jpg',
```

2.2 Dataset Manipulation:

The Dataset that was imported from folder in local disk is jpg image file, to build a model it is necessary that the data should be numeric. Images loaded are RGB image of varying m*n pixel size so it is resized into a standard size of 100*100 and then converted into a three dimensional array of shape (100, 100, 3) where 3 represents three channel - RGB. Below figure describes it:

```
In [167]: Data1 = []
          Target = []
          for x1 in range(0,len(Folder)):
              if x1 == 0:
                  Data_path11 = Data_path1
              else:
                  Data_path11 = Data_path2
              for x2 in range(0,len(Data_path11)):
                  kk = cv2.imread('C:/Users/Venu/Desktop/mask dataset/'+ Folder[x1] + '/' + Data_path11[x2])
                  img_ac = cv2.resize(kk,(100,100))
                  img_array = np.array(img_ac)
                  Data1.append(img_array)
                  Target.append(x1)
```

Thus two new arrays were created one for image dataset i.e. 3D image array and other is target array consist of ‘0’s (with mask) and ‘1’s (without mask).

2.3. Dataset Standardization:

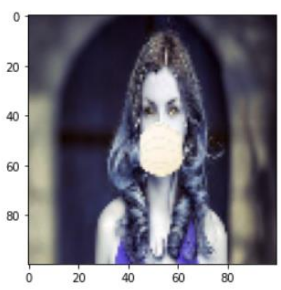
The image array dataset value will be ranging from 0 to 255, to make analysis faster and accurate the range of image array value is converted into 0 to 1 by standardization. The each value in the array is divided by 255 (the maximum pixel value) as follows:

```
In [171]: Data2 = []
          for x in range(0, len(Data1)):
              Data2.append(Data1[x]/255)
          Data3 = np.array(Data2)
```

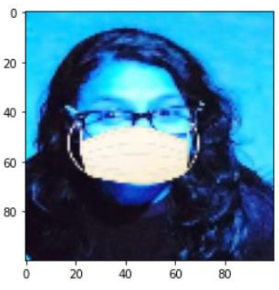
2.4. Dataset Visualization:

To cross check, image array can be visualized by matplot library (plt.imshow) as follows:

```
In [174]: plt.imshow(Data3[100])
Out[174]: <matplotlib.image.AxesImage at 0x23452189070>
```




```
In [175]: plt.imshow(Data1[9])
Out[175]: <matplotlib.image.AxesImage at 0x23455a964f0>
```



CHAPTER 3: MODEL BUILDING AND TESTING

To detect mask in real time we need two deep learning architecture, one is computer vision and other is Convolution Neural Network. The computer vision is used to detect face in live streaming video and CNN is used to predict whether a person is wearing a mask or not.

3.1. Face Recognition by OpenCV:

Before building a model let us see how to detect a face in an image or in a video. OpenCV library in python helps to recognize a face in an image or in a video. The Haar-Cascade Classifier in opencv classifies the region of interest (in our case it is face) from rest of region, to do this it is necessary to load haar-cascade frontal face default xml file which has the code to detect ROI (face) in image or in video as follows:

```
In [13]: import cv2

In [49]: haar_data = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

To detect face in live streaming video such as web cam, first we need to switch on the video, this can be done by **cv2.videocapture(0)**, here the value '0' represents default web camera in PC. We can check whether the video is correctly opened or not by **cv2.isOpened(),** if it is True then it will read the video as image and store it in a variable (klk), the flag variable represents the video correctness i.e. to check whether the video is reading correctly or not. After reading the image we want to detect at which area the face present i.e. ROI. This can be found in values (x, y, w, h) with the help of detectMultiscale option and then we want to draw a rectangle around the face by cv2.rectangle. Once the image is displayed it can be retained in the display by using waitkey option, '0' represents infinite milliseconds, windows are closed until we press an escape key (i.e. ANSI code of Escape key is 27). Finally we need to release the camera and close the windows by capture.release() and capture.destroyAllWindows().

```
In [18]: capture = cv2.VideoCapture(0)
         while (capture.isOpened()):
             flag,klk = capture.read()
             if flag:
                 bb = haar_data.detectMultiScale(klk)
                 for x,y,w,h in bb:
                     cv2.rectangle(klk,(x,y),(x+w,y+h),(255,0,255))
                     cv2.imshow('ttt',klk)
                     if cv2.waitKey(0) == 27:
                         break
             capture.release()
             cv2.destroyAllWindows()
```

3.2. Train-Test-Split:

To build the model it is significant to split the data into trained dataset and testing dataset with the help of sklearn library as follows:

```
In [50]: from sklearn.model_selection import train_test_split

In [222]: xtrain,xtest,ytrain,ytest = train_test_split(Data3,Target, test_size=0.2,random_state=10)

In [223]: xtrain.shape
Out[223]: (141, 100, 100, 3)

In [224]: xtrain = xtrain.reshape(-1,100,100,3)

In [225]: xtrain.shape
Out[225]: (141, 100, 100, 3)
```

CNN takes input in 4D so dataset is converted into four dimension by using xtrain.reshape (-1, 100, 100, 3) which implies images of 100*100 pixel with 3 channels and 141 images.

3.3 Model Building:

Convolution Neural Network was used for mask prediction. This network is built by sequential model by adding convolution and max pooling layers alternatively of desired kernel size. Four layers (Conv2D – Max Pooling – Conv2D – Max Pooling) followed by a flatten layer to make 4D to 1D, at last a dense layers is given to make interconnected network. Here in the convolution layer Rectified linear function is used and in the last dense layer softmax function is utilized to predict the binary category.

```
In [180]: from keras.models import Sequential

In [181]: from keras.layers import Dense, Conv2D, MaxPooling2D

In [182]: from keras.layers import Activation, Flatten

In [255]: model = Sequential()

In [256]: model.add(Conv2D(64,(4,4),activation ='relu',input_shape = ( 100, 100, 3)))

In [257]: model.add(MaxPooling2D(pool_size= (2,2)))

In [258]: model.add(Conv2D(32,(4,4),activation ='relu'))

In [259]: model.add(MaxPooling2D(pool_size= (2,2)))

In [260]: model.add(Flatten())

In [261]: model.add(Dense(50,activation='softmax'))
```

In the first convolution layer 64 node with 4*4 kernel size is utilized and in the second convolution layer 32 node with 4*4 kernel size is utilized . In CNN feature extraction is done to predict the image with higher accuracy so the first layer has 64 nodes and second layer has 32 nodes. The build model summary is as follows:

```
In [262]: model.summary()

Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_17 (Conv2D)           (None, 97, 97, 64)        3136
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 48, 48, 64)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 45, 45, 32)        32800
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 22, 22, 32)        0
_________________________________________________________________
flatten_7 (Flatten)          (None, 15488)             0
_________________________________________________________________
dense_7 (Dense)              (None, 50)                774450
=================================================================
Total params: 810,386
Trainable params: 810,386
Non-trainable params: 0
_________________________________________________________________
```

From the summary it is clear that the total parameters to be trained is 810,386 this was done in 10 epochs by using Adam optimizer, at each epochs the accuracy level of the model is increased and reaches 98 per cent as follows:

```
In [263]: model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])

In [264]: model.fit(xtrain,ytrain,epochs=10)

Epoch 1/10
5/5 [==============================] - 15s 3s/step - loss: 1.5530 - accuracy: 0.4681
Epoch 2/10
5/5 [==============================] - 15s 3s/step - loss: 0.6477 - accuracy: 0.6525
Epoch 3/10
5/5 [==============================] - 15s 3s/step - loss: 0.4625 - accuracy: 0.8156
Epoch 4/10
5/5 [==============================] - 15s 3s/step - loss: 0.3825 - accuracy: 0.8227
Epoch 5/10
5/5 [==============================] - 15s 3s/step - loss: 0.2535 - accuracy: 0.8652
Epoch 6/10
5/5 [==============================] - 17s 3s/step - loss: 0.1732 - accuracy: 0.9504
Epoch 7/10
5/5 [==============================] - 16s 3s/step - loss: 0.1012 - accuracy: 0.9929
Epoch 8/10
5/5 [==============================] - 17s 3s/step - loss: 0.0893 - accuracy: 0.9645
Epoch 9/10
5/5 [==============================] - 15s 3s/step - loss: 0.1066 - accuracy: 0.9574
Epoch 10/10
5/5 [==============================] - 16s 3s/step - loss: 0.0438 - accuracy: 0.9858

Out[264]: <tensorflow.python.keras.callbacks.History at 0x2340161aa00>
```

3.4 Model Testing:

To check out-of-sample performance of the model, prediction is done for the test dataset by creating a function as follows:

```
In [267]: sm1 = 0
          for x in range(0,36):
              sm = np.argmax(model.predict(xtest[x].reshape(-1,100,100,3)))
              if sm ==0:
                  sm1 = sm1 + 1
          print('Accuracy_score = ', sm1/36*100)

          Accuracy_score =  66.66666666666666
```

Predicted value is in the form of arguments i.e. it gives an array of ‘x’ values in our case it is 100 so np.argmax is used to find the maximum argument value in that 100 data points of the predicted value. The Accuracy score of out-of-sample prediction is above 60 per cent. So the model is quite better in predicting mask wearing persons.

CHAPTER 4: CONCLUSION

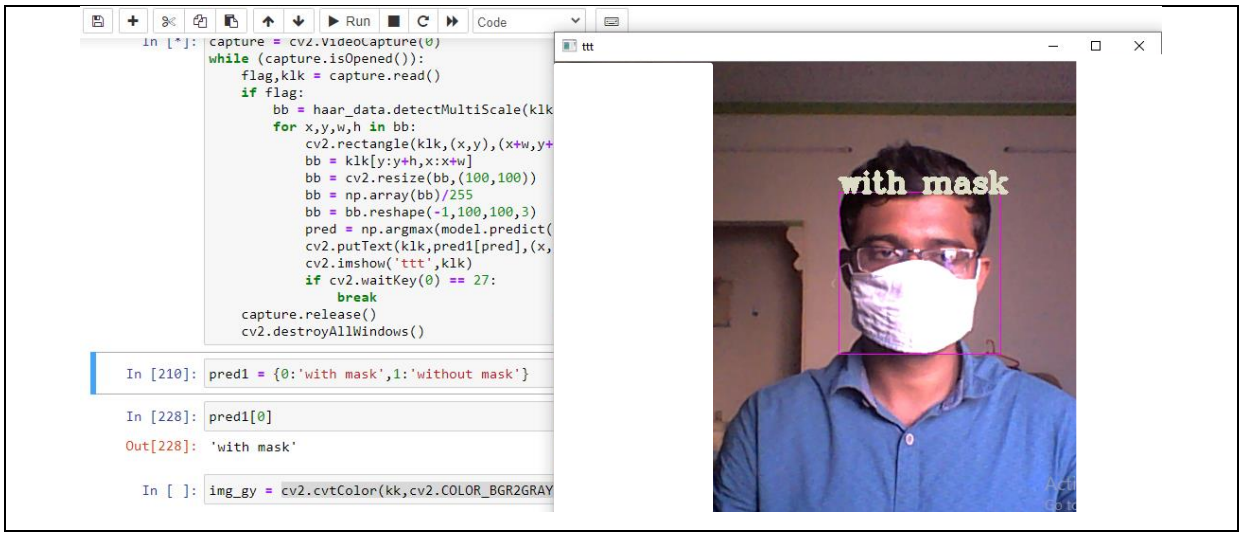
An optimum CNN model was build and this can be implemented on the computer vision framework as follows:

In [275]:
```
capture = cv2.VideoCapture(0)
while (capture.isOpened()):
    flag,klk = capture.read()
    if flag:
        bb = haar_data.detectMultiScale(klk)
        for x,y,w,h in bb:
            cv2.rectangle(klk,(x,y),(x+w,y+h),(255,0,255))
            bb = klk[y:y+h,x:x+w]
            bb = cv2.resize(bb,(100,100))
            bb = np.array(bb)/255
            bb = bb.reshape(-1,100,100,3)
            pred = np.argmax(model.predict(bb))
            cv2.putText(klk,pred1[pred],(x,y),cv2.FONT_HERSHEY_COMPLEX,1,(200,222,222),3)
            cv2.imshow('ttt',klk)
            if cv2.waitKey(0) == 27:
                break
    capture.release()
    cv2.destroyAllWindows()
```

In [273]:
```
pred1 = {0:'with mask',1:'without mask'}
```

- A default camera is opened in PC by cv2.videoCapture(0)
- If the camera is opened correctly i.e. capture.isopened() then read the video as infinite image (klk).
- If camera is working correctly (flag = True), then detect the region of interest (face) as x,y,w,h in the image by haar_data.detectMultiscale(klk)
- Draw a rectangle on the image and read only the face region (klk[y:y+h,x:x+w]) to give as input to the model.
- Resize the face image into 100*100 pixel
- Convert the image into array since the model takes only numeric input.
- Standardize the image array for quicker analysis.
- Apply the standardized image array to the model to predict whether a person is wearing a mask or not.
- Put text on the image by cv2.putText and display the image.
- Retain the image until escape key is pressed (cv2.Waitkey (0) == 27).

REAL TIME FACE MASK DETECTION – WITH MASK



REAL TIME FACE MASK DETECTION – WITHOUT MASK