BAN 676

Deep Learning Project

Gender & Mask Detection in Real Time Video Stream (Meet Expectations)

By,
Prathamesh Rajiv Bhople (YV5392)
Subramanian Tirunelveli Padmanabhan (LV6503)

Contents

1. Introduction

Since COVID-19 has hit the world, there is a need to take vaccines and for people to wear masks. People occasionally neglect/forget to wear masks leading to spread of disease and unpleasant situations. Using Convolutional Neural Networks, the idea is to build a web application that will first detect gender and then detect Masks usage for multiple people on a live webcam.

This project is divided into three phases. First phase is to develop Convolutional neural networks model and train it for the detection of gender and presence of facemask. Second phase is to run the models on a live web stream. Third phase is to integrate the model and develop a Web Application for the same.

Based on the presence of mask, the results will be displayed on a web application. If no mask is detected, a mail is sent to the administrator informing the same.

2. Data Description

To develop and train the gender detection model, we are referring to the images classified into male and female. Similarly, for facemask detection, we are referring to the images classified into mask and no-mask. Some Images are scraped from the web and some downloaded. Following are some examples of the images:

MALE

FEMALE

WITH MASK

WITHOUT MASK

3. Model Build - Gender Detection

Data Preprocessing

In data preprocessing, we are going to convert the images into arrays. We created two lists' namely 'data' and 'label'. Once the images are fetched, we shuffle images resize the images ,and update weights uniformly to avoid bias and overfitting. While training, this leads us to a better model. In this case, we used (96, 96, 3) where 96 is the height x width of images and 3 represents RGB channels. We append the images in the data list.

In the labels list, we append the label '1' if the image is female and '0' if image is male. Once the data and label are stored in lists, we convert these lists into arrays where feature scaling is diving by 255. Because, in images every pixel value is from 1 to 255, there when uniformly divided by 255, we have array values from 0 to 1.

We then split the data into train and test, where 80% is the training data and 20% is the testing data. After this, we converted the labels into one-hot encoding with 2D labels. Furthermore, we are using ImageDataGenerator for the data augmentation to generate more images with horizontal shifts, rotation etc. to train the model better.

Model Definition

We define the model where input shape of an image are the arguments for the function along with classes. We create a model object using keras.models Sequential API of Keras Library. We are using Conv2D as convolutional layers with padding and activation function as “relu”.

To optimize, we are using Batch Normalization in between layers. MaxPooling2D layer is used to reduce the noise and will focus on the necessary features of the images. Dropout is used to deactivate certain number of neurons to avoid overfitting. Flatten layer is used to covert two dimensions into single dimensions.

Finally, we have a Dense layer as the output with ‘sigmoid’ as the activation function since we have two categories, male and female. Once the model is built, it is compiled and trained with loss function as ‘binary_crossentropy’, optimizer as ‘adam’ and metrics as ‘accuracy’.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 96, 96, 32)        896

activation (Activation)      (None, 96, 96, 32)        0

batch_normalization (BatchN  (None, 96, 96, 32)        128
ormalization)

max_pooling2d (MaxPooling2D  (None, 32, 32, 32)        0
)

dropout (Dropout)            (None, 32, 32, 32)        0

conv2d_1 (Conv2D)            (None, 32, 32, 64)        18496

activation_1 (Activation)    (None, 32, 32, 64)        0

batch_normalization_1 (Batc  (None, 32, 32, 64)        256
hNormalization)

conv2d_2 (Conv2D)            (None, 32, 32, 64)        36928
```

activation_2 (Activation) (None, 32, 32, 64) 0

batch_normalization_2 (Batc (None, 32, 32, 64) 256
hNormalization)

max_pooling2d_1 (MaxPooling (None, 16, 16, 64) 0
2D)

dropout_1 (Dropout) (None, 16, 16, 64) 0

conv2d_3 (Conv2D) (None, 16, 16, 128) 73856

activation_3 (Activation) (None, 16, 16, 128) 0

batch_normalization_3 (Batc (None, 16, 16, 128) 512
hNormalization)

conv2d_4 (Conv2D) (None, 16, 16, 128) 147584

activation_4 (Activation) (None, 16, 16, 128) 0

batch_normalization_4 (Batc (None, 16, 16, 128) 512
hNormalization)

max_pooling2d_2 (MaxPooling (None, 8, 8, 128) 0
2D)

dropout_2 (Dropout) (None, 8, 8, 128) 0

flatten (Flatten) (None, 8192) 0

dense (Dense) (None, 1024) 8389632

activation_5 (Activation) (None, 1024) 0

batch_normalization_5 (Batc (None, 1024) 4096
hNormalization)

dropout_3 (Dropout) (None, 1024) 0

dense_1 (Dense) (None, 2) 2050

activation_6 (Activation) (None, 2) 0

==========

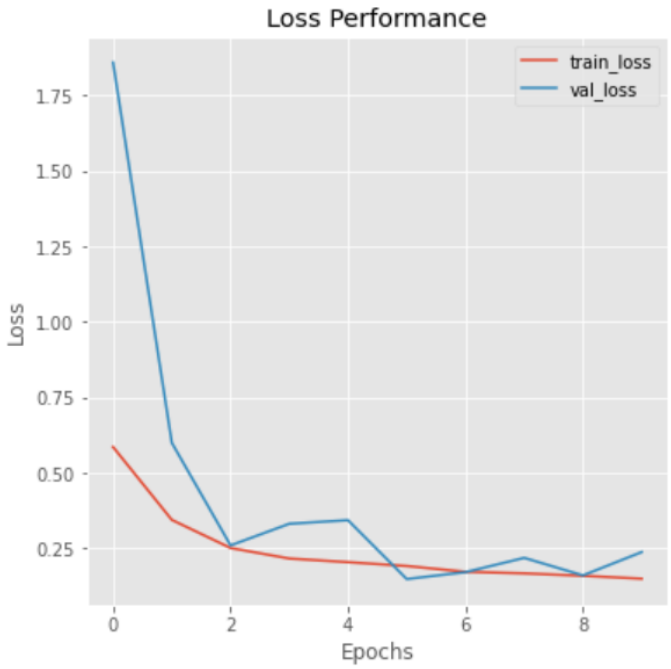
===

Total params: 8,675,202
Trainable params: 8,672,322
Non-trainable params: 2,880

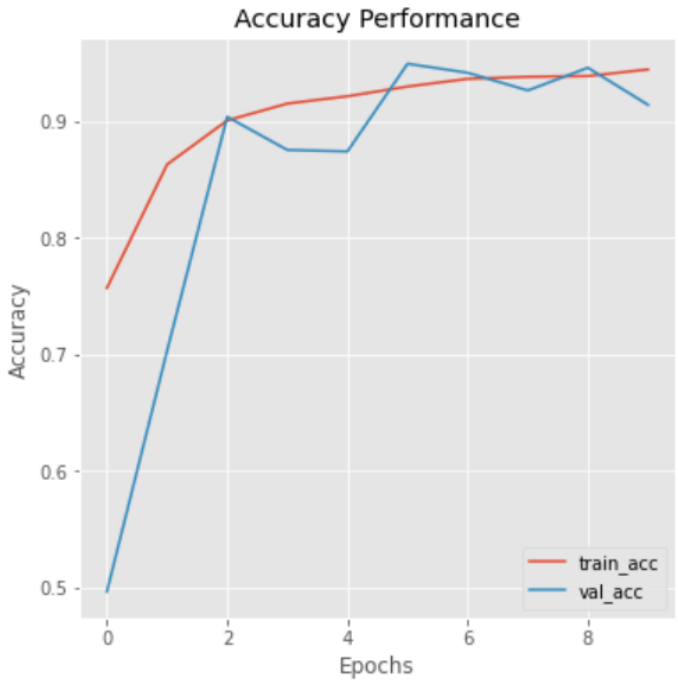
_____

Model Performance

Following plot depicts the training and valid loss performance of the model:



Similarly, below plot depicts the training and valid accuracy performance of the model:



Once entire model trained, it is saved as **gender_detection.model** on the disk.

4. Model Build - Facemask Detection

Data Preprocessing

Here are converting the with mask and without mask images into arrays. We are using the *load_img* function from keras API, which is used to read the images with the specified target size for uniform size of the images. In this case we are setting the target size of the images as 224 x 224.

Once the image is read, we are using the *img_to_array* function from keras preprocessing to convert an image into two-dimensional array and store it in data list, and labels in label list. To change the labels from text to one hot encoding categorical array, we are using the LabelBinarizer method from sklearn preprocessing module.

Once data and labels are stored in the respective lists, we are changing them into numpy arrays to train the model. Now, we are splitting the data into train and test where 80% data is for training and 20% data is for testing. Additionally, we are using ImageDataGenerator for data augmentation to generate more images with horizontal shifts, rotation etc. to better train the model.

Model 1: CNN Model

In this model, we are using the conventional Conv2D, MaxPooling2D, Dropout, Flatten and Dense layers to build a model for training the with and without mask images. Once data array and labels of the images are created, we compile and train the model with loss function as 'sparse_categorical_crossentropy', optimizer as 'adam' and metrics as 'accuracy'. Following is the CNN face mask detection model,

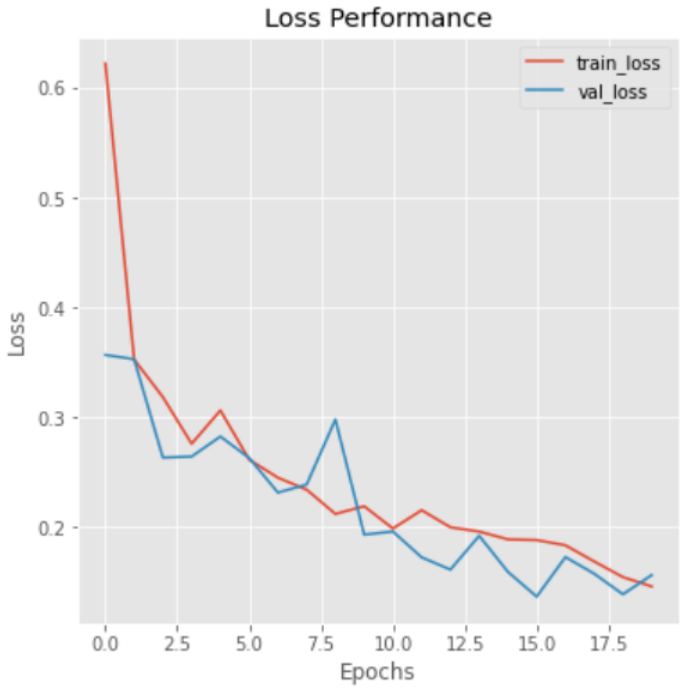
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
=

conv2d_9 (Conv2D)           (None, 222, 222, 32)      896

max_pooling2d_7 (MaxPooling  (None, 111, 111, 32)     0
2D)

conv2d_10 (Conv2D)          (None, 109, 109, 64)      18496

max_pooling2d_8 (MaxPooling  (None, 54, 54, 64)       0
2D)

conv2d_11 (Conv2D)          (None, 52, 52, 128)       73856

max_pooling2d_9 (MaxPooling  (None, 26, 26, 128)      0
2D)

conv2d_12 (Conv2D)          (None, 24, 24, 256)       295168

max_pooling2d_10 (MaxPoolin  (None, 12, 12, 256)      0
g2D)

dropout_8 (Dropout)         (None, 12, 12, 256)       0

flatten_2 (Flatten)         (None, 36864)             0

dense_7 (Dense)             (None, 128)               4718720
```

```
dropout_9 (Dropout)          (None, 128)               0

dense_8 (Dense)              (None, 256)               33024

dropout_10 (Dropout)         (None, 256)               0

dense_9 (Dense)              (None, 2)                 514

=================================================================
Total params: 5,140,674
Trainable params: 5,140,674
Non-trainable params: 0
_________________________________________________________________
```

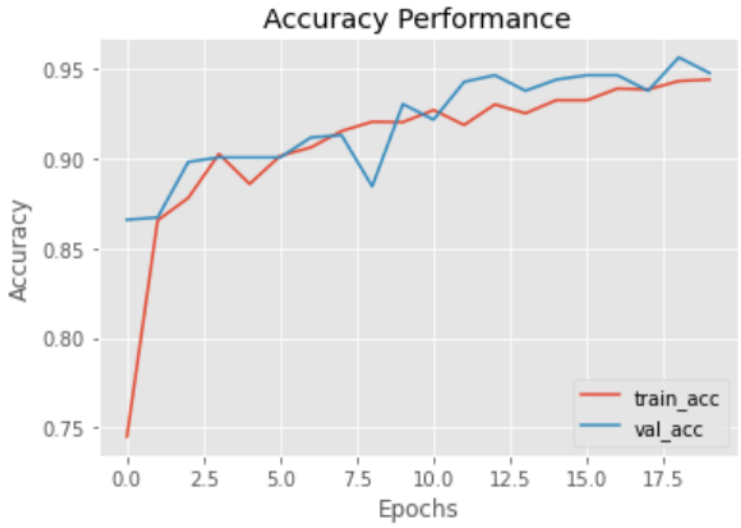
Once the model is trained, it is saved as ***mask_detector_cnn.model*** on the disk.

Model Performance

Following plot depicts the Loss performance of this model:



Following plot depicts the Accuracy performance of this model:



Comparing Model 1 and Model 2, it can be inferred that model with MobileNetV2 model trains much better and faster than conventional CNN model.

5. Alternative Model

MobileNetV2 Transfer Learning.

In this model, we are using the MobileNetV2 model as a Transfer Learning method. MobileNetV2 are very fast in processing as compared to convolutional neural networks. Also, this MobileNetV2 uses lesser parameters. This helps in training the model much faster.

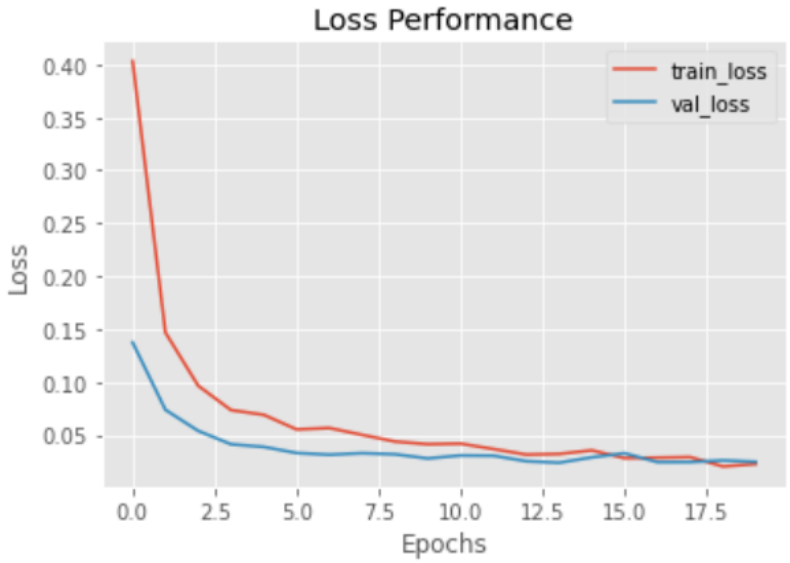
Input data array is passed to the MobileNetV2 model, and the output of this model is passed as an input to the head model as first parameter. Along with this, we are doing average pooling and flattening the layer. Later, we are adding the dropout layer to avoid the overfitting of the model.

Finally, we are adding the output layer with two neurons for categories with and without mask and using ‘softmax’ as activation function. Once the base and head model are designed, we are calling the Model function from keras models’ module to connect base and head model.

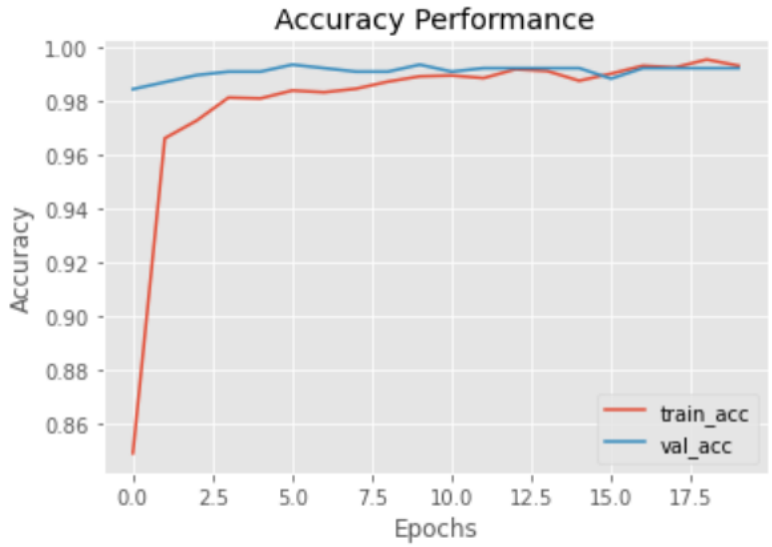
We will be freezing the layers in the MobileNetV2 model so that they will not be updated during first training process and adding some trainable layers on top these layers. This will be the actual training model for facemask detection. Once the model is designed, it is compiled and trained with loss function as ‘binary_crossentropy’, optimizer as ‘adam’ and metrics as ‘accuracy’.

Model Performance

Following plot depicts the training and valid loss performance of this model:



Following plot depicts the training and valid accuracy performance of this model:



Once entire model trained, it is saved as mask_detector.model on the disk.

6. References

1. **Keras Layers API**
   Author: Keras
   URL: https://keras.io/api/layers/

2. **MobileNetV2: Inverted Residuals and Linear Bottlenecks**
   Author: Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen
   URL:
   https://openaccess.thecvf.com/content_cvpr_2018/papers/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.pdf

3. **A Survey on Transfer Learning**
   Author: Sinno Jialin Pan, Qiang Yang
   URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5288526

NOTE: Results of the project are displayed using streamlit web-application. Output results are mentioned in the *BAN676_Group_Project_Report_Exceed Expectation.docx* project report.