Detection of Masked and Non-Masked faces

Aditi Goyal (B19EE003) , Darshit Jain (B19EE024)

Abstract

Corona Virus Disease (COVID-19) pandemic is causing a health crisis. One of the effective methods against the virus is wearing a face mask. It becomes increasingly necessary to check if other people are wearing mask or not. In this project, we have introduced face mask detection that can be used by the authorities to make mitigation, evaluation, prevention, and action planning against COVID-19. The face mask recognition in this study is developed with machine learning algorithms through the image classification method: MobileNetV2, Convolutional Neural Network and ResNet152V2. The steps involved in building these models are pre-processing (on the given data), splitting the data, and testing the model.

Index Terms

MobileNetV2, Convolution Neural Network, ResNet152V2, Tensorflow, Keras, ImageGenerator, ModelCheckpoint,VisualKeras

I. Introduction

IN order to protect ourselves from the COVID-19 Pandemic, almost every one of us tend to wear a face mask. It becomes increasingly necessary to check if the people in the crowd wear face masks in public gatherings such as Malls, Theatres, Parks. The development of an AI solution to detect if the person is wearing a face mask and allow their entry would be of great help to the society.

II. Data Extraction and Preprocessing

The dataset used to build the project is - self built mask detection. The dataset contains about 90,000+ images separated into 2 categories - masked faces and unmasked faces. We picked a smaller set of data from it to train and test. The target values were then converted into categorical data using one hot encoding.

III. Model 1 - Using MobileNetV2

CNN Architecture

In this proposed method, the Face Mask detection model is built using 2 separately defined layers - Base Model and a layer placed on top of it - Head Model. The base layer is made using **MobileNetV2**. MobileNetV2 is a convolutional neural network architecture that is a very effective feature extractor for object detection and segmentation. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.

ModelCheckpoint callback is used in conjunction with training using model.fit() to save a model or weights (in a checkpoint file) at some interval, so the model or weights can be loaded later to continue the training from the state saved. We use it to save the best performing model.

ImageGenerator helps in image augmentation, thus increasing dataset by transformations while the model is being trained.

The other layer added is AveragePooling2D with pool size as (7,7). This means down-sampling the input along its spatial dimensions (height and width) by taking the average value over an input window (of size defined by pool size).

The next layer is Flatten() is followed by a Dense() of 128 perceptrons with ReLu activation function.

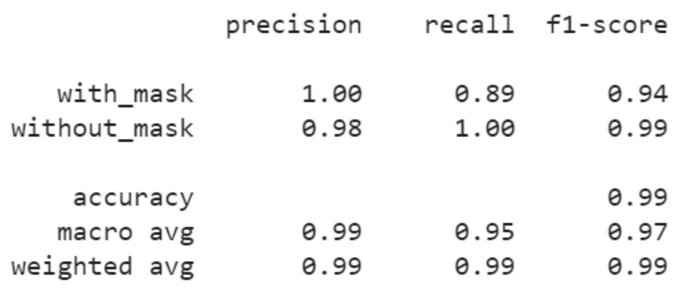
Lastly we have Dropout of 0.5 to prevent over-fitting because the outputs of a layer under dropout are randomly sub-sampled, it has the effect of reducing the capacity or thinning the network during training.

CNN Model Design

Fig. 1. Model Design Plot

Loss and Accuracy Statistics

The model accuracy and the F1-Score is very high implying that the MobileNetV2 model performed well at classifying the images and the cross validation accuracy stopped changing after the 15th epoch.



```
              precision    recall  f1-score

   with_mask       1.00      0.89      0.94
without_mask       0.98      1.00      0.99

    accuracy                           0.99
   macro avg       0.99      0.95      0.97
weighted avg       0.99      0.99      0.99
```

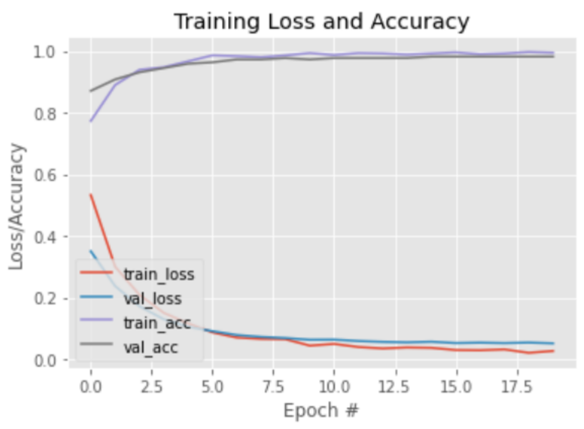


Fig. 2. Loss and Accuracy Values along with plot

IV. Model 2 - Using CNN

CNN Architecture

In this proposed method, the Face Mask detection model is built using the Sequential API of the keras library. This allows us to create the new layers for our model step by step. The various layers in the CNN model is described below.

The first layer is the Conv2D layer with 100 filters of kernel size of 3X3. The activation function used is 'ReLu'-Rectified Linear Unit, for which output is proportional to the input directly if is positive, otherwise, it will be zero. The input size is also initialized as 150X150X3 for all the images to be trained and tested using this model

In the second layer, the MaxPooling2D is used with the pool size of 2X2.

The next layer is again a Conv2D layer with another 100 filters of the same filter size 3X3 and the activation function used is the 'ReLu'. This Conv2D layer is followed by a MaxPooling3=2D layer with pool size 2X2.

In the next step, we use the Flatten() layer to flatten all the layers into a single 1D layer.

After the Flatten layer, we use the Dropout (0.5) layer to prevent the model from overfitting.

Finally, towards the end, we use the Dense layer with 50 units and the activation function as 'ReLu'.

The last layer will be another Dense Layer, with only two units and the activation function used is 'Softmax' function. The softmax function outputs a vector which will represent the probability distributions of each of the input units. Here, two input units are used. The softmax function will output a vector with two probability distribution values.

CNN Model Design

Visualkeras is a Python package to help visualize Keras neural network architectures. Visualkeras computes the size of each layer by the output shape. Values are transformed into pixels. Then, scaling is applied. Each layer is depicted and the width of the plot shows the depth of the neural network.

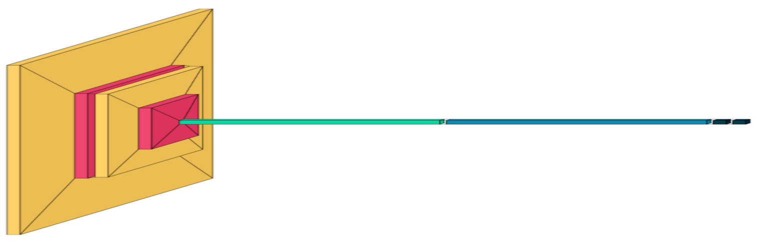
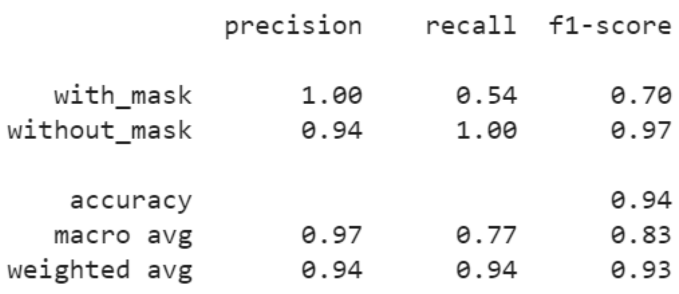


Fig. 3. Model Design Plot

Loss and Accuracy Statistics

The F1-Score for the class *with_mask* is significantly lower than the F1-Score for the class *without_mask*.



```
              precision    recall  f1-score

   with_mask       1.00      0.54      0.70
without_mask       0.94      1.00      0.97

    accuracy                           0.94
   macro avg       0.97      0.77      0.83
weighted avg       0.94      0.94      0.93
```

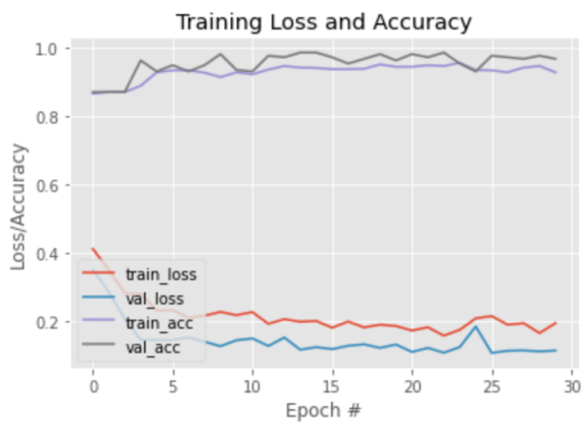


Fig. 4. Loss and Accuracy Values along with plot

V. Model 3 - CNN using Different Filter Values

CNN Architecture

In the model we have experimented with increasing filter sizes in each layer. Filters detect spatial patterns such as edges in an image by detecting the changes in intensity values of the image. Number of filters is chosen based complexity of task. More complex tasks require more filters. And usually number of filters grows after every layer (eg 128 -¿ 256 -¿ 512). First layers (with lower number of filters) catch few of some simple features of images (edges, color tone, etc) and next layers are trying to obtain more complex features based on simple ones. Hence we have emulated the same approach with 32,64,128,256 filters respectively in each succeeding layer to catch all the nuances of the image and get the best training and accuracy.

The best of the layers are similar to before - Flatten(), Dense() with 128 and 256 perceptrons and Dropout finally.

CNN Model Design

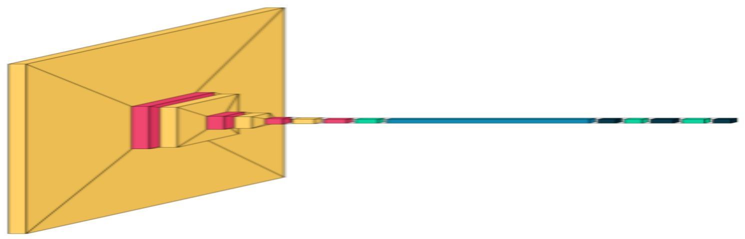
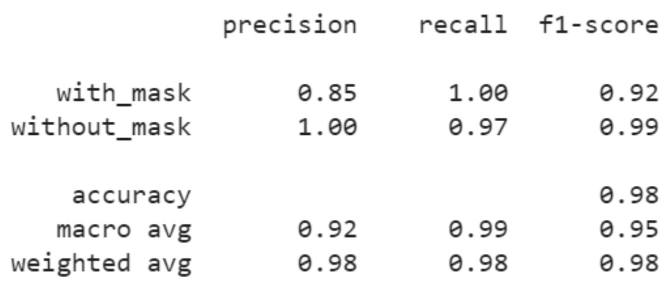


Fig. 5. Model Design Plot

Loss and Accuracy Statistics

The accuracy obtained is 98% and the F-1 score is very high, showing that the model gave good results. Since the classwise distribution is not equal, it is better to look at F1-Score than arriving at conclusions based on accuracy value. The training loss started decreasing at a high rate after 20 epochs, increasing the accuracy of the model.



```
              precision    recall  f1-score

   with_mask       0.85      1.00      0.92
without_mask       1.00      0.97      0.99

    accuracy                           0.98
   macro avg       0.92      0.99      0.95
weighted avg       0.98      0.98      0.98
```

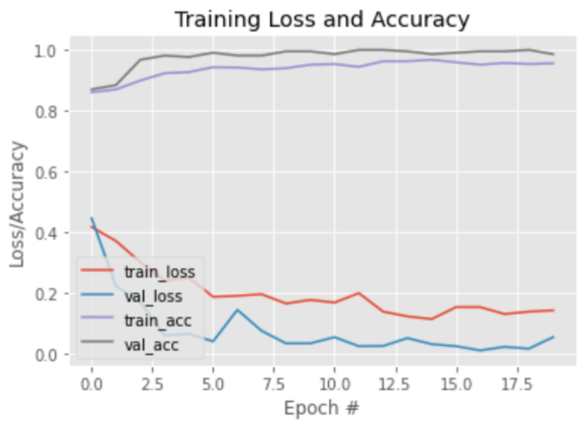


Fig. 6. Loss and Accuracy Values along with plot

VI. Model 4 - ResNet CNN

CNN Architecture

ResNet152V2 is used primarily for Image Classification purposes. The weight used is '**imagenet**' and pooling is defined as 'None'. Resnet introduces a structure called residual learning unit to alleviate the degradation of deep neural networks. This unit's structure is a feedforward network with a shortcut connection which adds new inputs into the network and generates new outputs. The main merit of this unit is that it produces better classification accuracy without increasing the complexity of the model. We select Resnet152 as it achieves the best accuracy among Resnet family members.

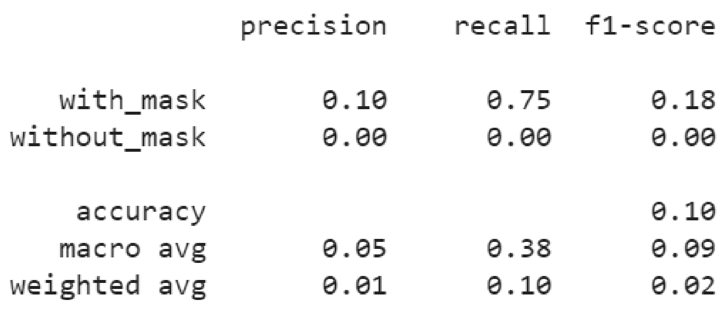
We freeze the weights of initial layers that is make them non-trainable.

CNN Model Design

Fig. 7. Model Design Plot

Loss and Accuracy Statistics

The ResNet152V2 model gave high cross validation accuracy (a maximum value of 95.8%, but it can be seen that the testing accuracy and the F1-Score is dropped drastically. This shows that our model probably overfitted on the data and hence, is not a reliable model as there is also variation in the loss for epochs.



```
              precision    recall  f1-score

   with_mask       0.10      0.75      0.18
without_mask       0.00      0.00      0.00

    accuracy                           0.10
   macro avg       0.05      0.38      0.09
weighted avg       0.01      0.10      0.02
```

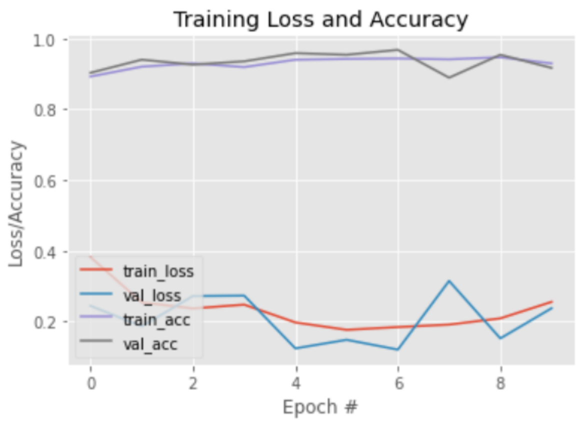


Fig. 8. Loss and Accuracy Values along with plot

VII. Conclusion

Several variations of neural networks such as MobileNetV2, using a less deep mode with max pooling, adding an array of filters to detect patters and finally a ResNet CNN model too. Since the dataset had very clear masked and unmasked images, most of the models gave very high accuracy of over 97% on fine tuning the parameters of the layer. Only ResNet underperformed from expections, since the subset of the actual dataset was small and ResNet being very deep must have overfit to the training data.

Overall we learnt the vast implementations and power of Tensorflow, Keras and Machine Learning in general to solve real world problems and benefit humanity.

Contribution

- Aditi Goyal (B19EE003) : She worked extensively on Model 3 coming up with the best variation of filters and the Model 4 with implementing ResNet CNN efficiently.
- Darshit Jain (B19EE024) : He worked on developing Model 1 and Model 2. He researched about the MobileNet layer and the most appropriate layer parameters for both the CNN models.