INT-248 PROJECT

Face Mask Detection Using Customized CNN

Name: - Shivmangal Singh Yadav

Section: - KM038

Roll: - A13

Reg. no: - 11709300

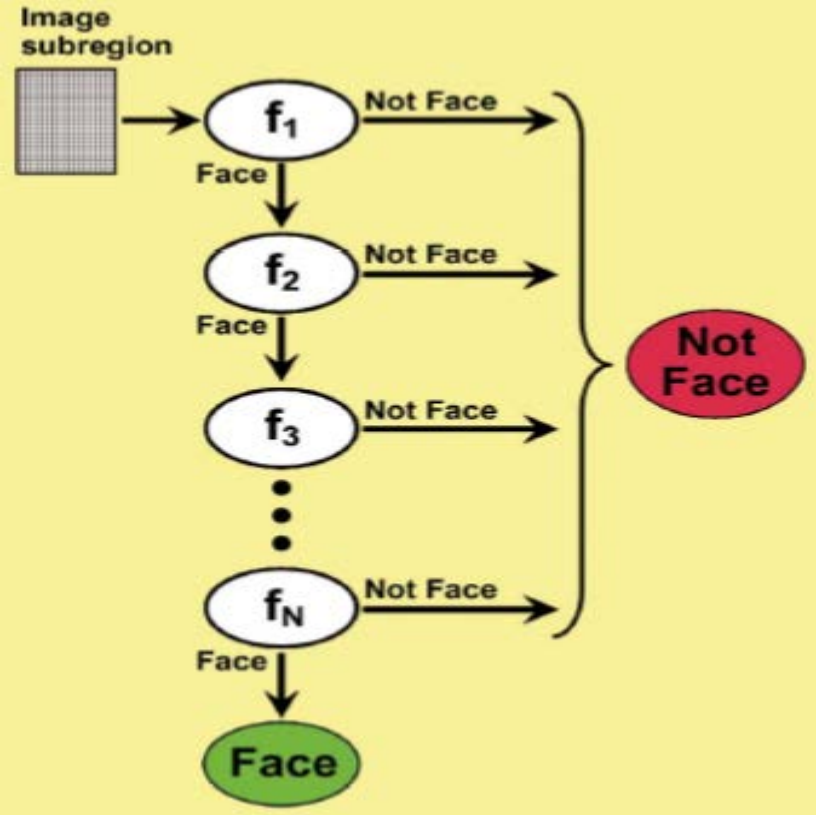
E-mail:-shivmangal.0000ved@gmail.com

Mobile no:- 7740027721

Code Link- Face Mask Detection Using CNN

Abstract: - To avoid COVID-19 spreading people must have to wear mask .Because of daily needs and expanses people can't stay all the time in their homes.This project is to develop a machine learning model using Customized CNN to predict whether a person has wearied the mask or not. With the help of OpenCv (It is cross-platform library which is used in of image analysis, feature detection in includes face detection, eye detection etc. Besicaly it is written in c++

language and it later binded with java and python also. In the project finding the face co-ordinate positions from the image I have been using “haarcascades” library of OpenCv for frontal face. It works in following way –

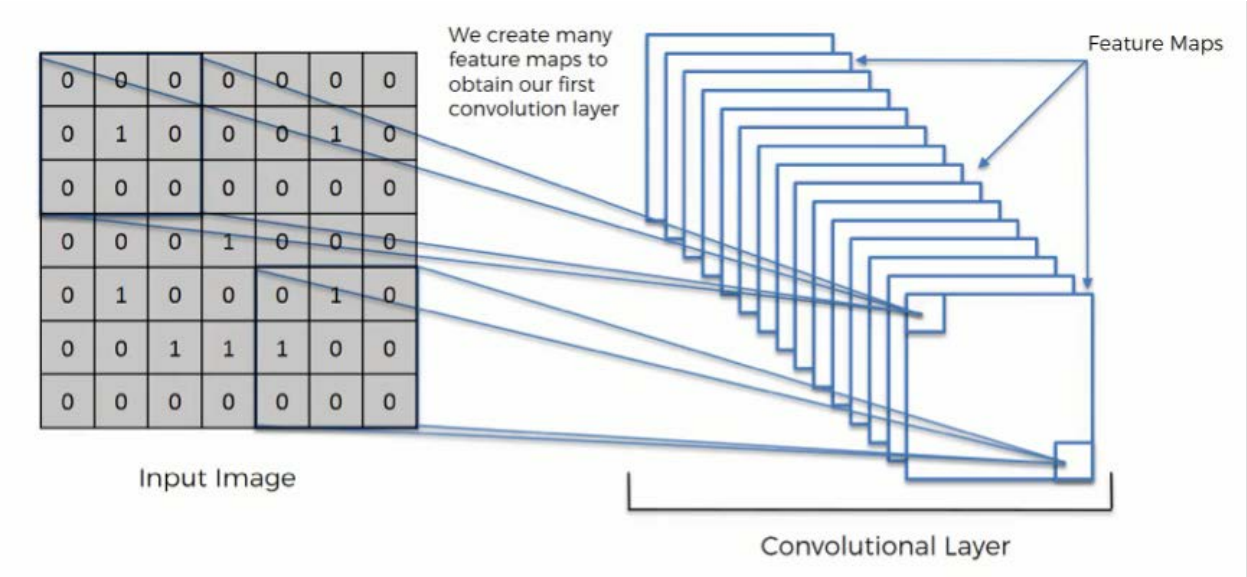


This model has been trained using by 396 images (with mask) and 385 images (without mask).This model is working with validation accuracy of 96.45% with loss of 2.39% in 20 epoch.

Keywords:-CNN, Deep Learning, OpenCV, Tecsorflow , Keras etc.

Introduction:- Convolutional Neural Networks (**CNN**) is one of important part of Neural Networks which is very effective in areas such as image recognition and classification it identifying faces, powering vision robots , objects and and

self-driving cars.



I am using CNN for face mask detection.

Methods

Providing content/data access to google colab from google drive :

-First I have make connected google colab with my google drive by authentication for dataset and some OpenCv library so that generated model will be stored and will be easy to access whenever required. Dataset link- https://github.com/prajnasb/observations/tree/master/experiements/data

Data Preprocessing:-

Initially I was having data set in two category one is 'without mask' and another one is 'with mask' .I made dictionary with value 0 and 1 respectively. After that I pick up all the images one by one and convert them in gray and crop them in 100X100 .And store them in 'data' and 'target list .Now I have save them so that every time no need to pre-process the same data again and again.

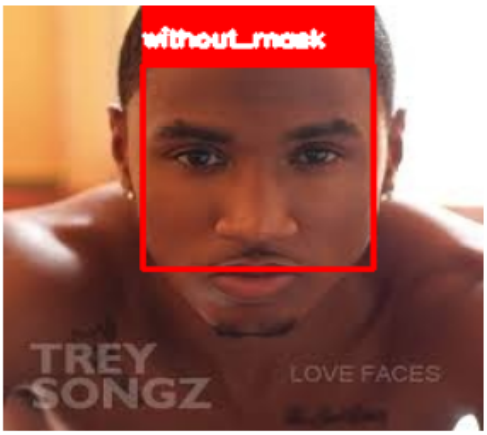
Applying CNN and Training the model :-

First I have loading the 'data' and 'target' list .My first layer is 'Conv2D' followed by 'Relu' and 'MaxPooling' layers .Second layers is 'Flatten' layer followed by 'Dropout' .I have uses Dense layer of 64 neurons. At last I have added layer with two outputs .To compile the model I have uses very famous 'adam' .Summary of every epoch is in following image -

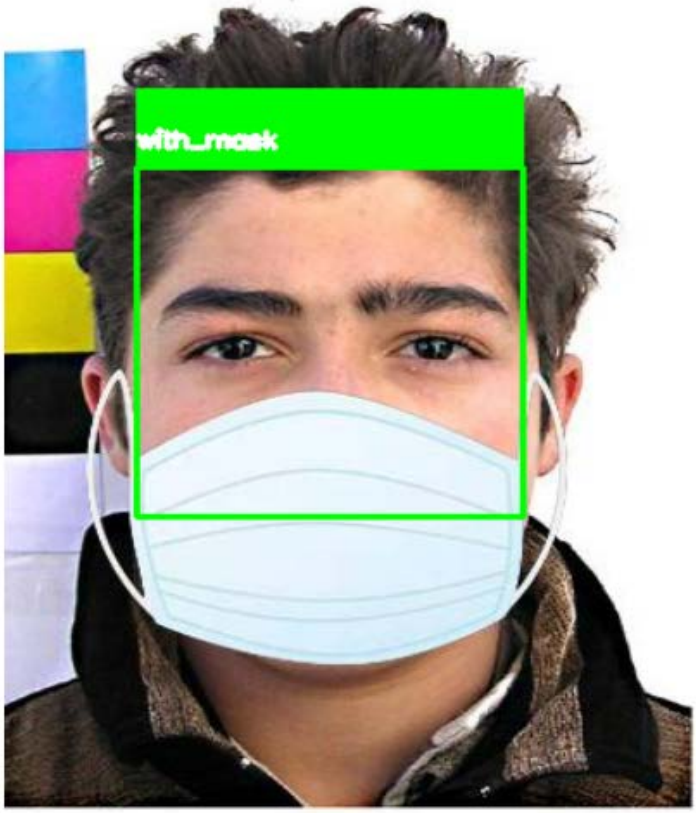
```
INFO:tensorflow:Assets written to: model-001.model/assets
18/18 [==============================] - 42s 2s/step - loss: 0.7127 - accuracy: 0.5258 - val_loss: 0.6908 - val_accuracy: 0.5177
Epoch 2/20
18/18 [==============================] - ETA: 0s - loss: 0.6566 - accuracy: 0.5971INFO:tensorflow:Assets written to: model-002.model/assets
18/18 [==============================] - 48s 3s/step - loss: 0.6566 - accuracy: 0.5971 - val_loss: 0.5528 - val_accuracy: 0.7376
Epoch 3/20
18/18 [==============================] - ETA: 0s - loss: 0.5365 - accuracy: 0.7237INFO:tensorflow:Assets written to: model-003.model/assets
18/18 [==============================] - 42s 2s/step - loss: 0.5365 - accuracy: 0.7237 - val_loss: 0.4648 - val_accuracy: 0.7376
Epoch 4/20
18/18 [==============================] - ETA: 0s - loss: 0.4425 - accuracy: 0.7772INFO:tensorflow:Assets written to: model-004.model/assets
18/18 [==============================] - 42s 2s/step - loss: 0.4425 - accuracy: 0.7772 - val_loss: 0.3284 - val_accuracy: 0.8794
Epoch 5/20
18/18 [==============================] - ETA: 0s - loss: 0.3263 - accuracy: 0.8592INFO:tensorflow:Assets written to: model-005.model/assets
18/18 [==============================] - 42s 2s/step - loss: 0.3263 - accuracy: 0.8592 - val_loss: 0.2144 - val_accuracy: 0.9078
Epoch 6/20
18/18 [==============================] - ETA: 0s - loss: 0.2753 - accuracy: 0.8841INFO:tensorflow:Assets written to: model-006.model/assets
18/18 [==============================] - 42s 2s/step - loss: 0.2753 - accuracy: 0.8841 - val_loss: 0.1773 - val_accuracy: 0.9291
Epoch 7/20
18/18 [==============================] - 41s 2s/step - loss: 0.2132 - accuracy: 0.9251 - val_loss: 0.1820 - val_accuracy: 0.9220
Epoch 8/20
18/18 [==============================] - ETA: 0s - loss: 0.1805 - accuracy: 0.9358INFO:tensorflow:Assets written to: model-008.model/assets
18/18 [==============================] - 42s 2s/step - loss: 0.1805 - accuracy: 0.9358 - val_loss: 0.1426 - val_accuracy: 0.9504
Epoch 9/20
18/18 [==============================] - ETA: 0s - loss: 0.1346 - accuracy: 0.9519INFO:tensorflow:Assets written to: model-009.model/assets
18/18 [==============================] - 42s 2s/step - loss: 0.1346 - accuracy: 0.9519 - val_loss: 0.1280 - val_accuracy: 0.9362
Epoch 10/20
18/18 [==============================] - 41s 2s/step - loss: 0.1337 - accuracy: 0.9608 - val_loss: 0.1477 - val_accuracy: 0.9433
Epoch 11/20
18/18 [==============================] - ETA: 0s - loss: 0.1209 - accuracy: 0.9537INFO:tensorflow:Assets written to: model-011.model/assets
18/18 [==============================] - 42s 2s/step - loss: 0.1209 - accuracy: 0.9537 - val_loss: 0.1169 - val_accuracy: 0.9362
Epoch 12/20
18/18 [==============================] - ETA: 0s - loss: 0.1111 - accuracy: 0.9590INFO:tensorflow:Assets written to: model-012.model/assets
18/18 [==============================] - 42s 2s/step - loss: 0.1111 - accuracy: 0.9590 - val_loss: 0.0980 - val_accuracy: 0.9574
Epoch 13/20
18/18 [==============================] - 41s 2s/step - loss: 0.0726 - accuracy: 0.9804 - val_loss: 0.1352 - val_accuracy: 0.9433
Epoch 14/20
18/18 [==============================] - 41s 2s/step - loss: 0.0576 - accuracy: 0.9875 - val_loss: 0.1018 - val_accuracy: 0.9574
Epoch 15/20
18/18 [==============================] - 41s 2s/step - loss: 0.0729 - accuracy: 0.9715 - val_loss: 0.1080 - val_accuracy: 0.9504
Epoch 16/20
18/18 [==============================] - 41s 2s/step - loss: 0.0586 - accuracy: 0.9768 - val_loss: 0.1154 - val_accuracy: 0.9645
Epoch 17/20
18/18 [==============================] - 41s 2s/step - loss: 0.0414 - accuracy: 0.9857 - val_loss: 0.1138 - val_accuracy: 0.9645
Epoch 18/20
18/18 [==============================] - 41s 2s/step - loss: 0.0300 - accuracy: 0.9911 - val_loss: 0.0997 - val_accuracy: 0.9574
Epoch 19/20
18/18 [==============================] - 41s 2s/step - loss: 0.0252 - accuracy: 0.9947 - val_loss: 0.1035 - val_accuracy: 0.9574
Epoch 20/20
18/18 [==============================] - 41s 2s/step - loss: 0.0239 - accuracy: 0.9911 - val_loss: 0.0981 - val_accuracy: 0.9645
```

Making the model to predict :- Before making the model to predict first I will load the image from my drive or directly from URL of that image .After that I have preprocessed like converting color to gray ,reducing shape into 100X100 .

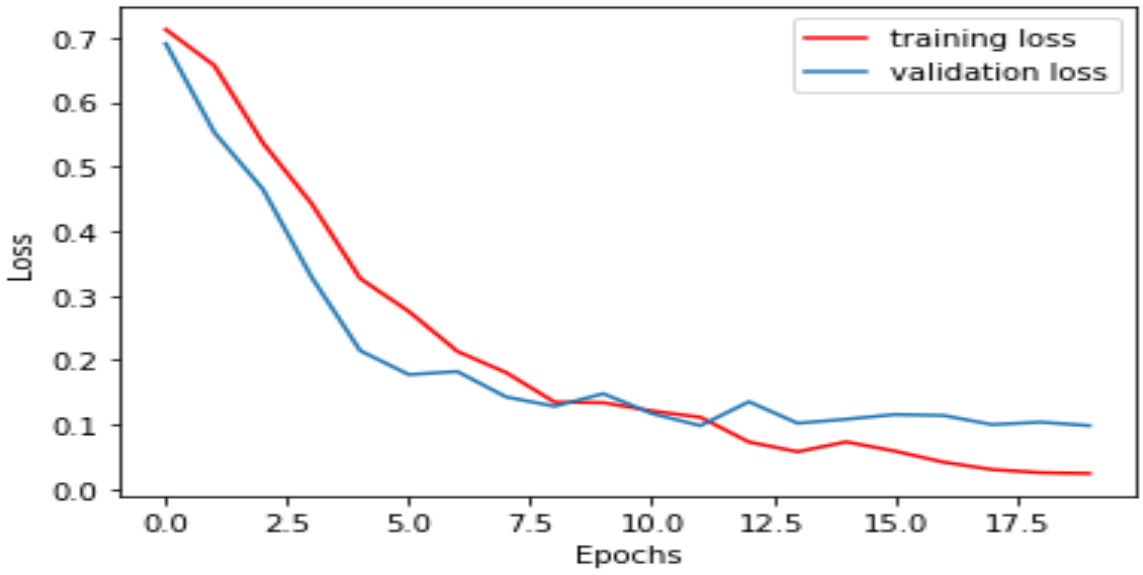
```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 44269  100 44269    0     0   229k      0 --:--:-- --:--:-- --:--:--  229k
```

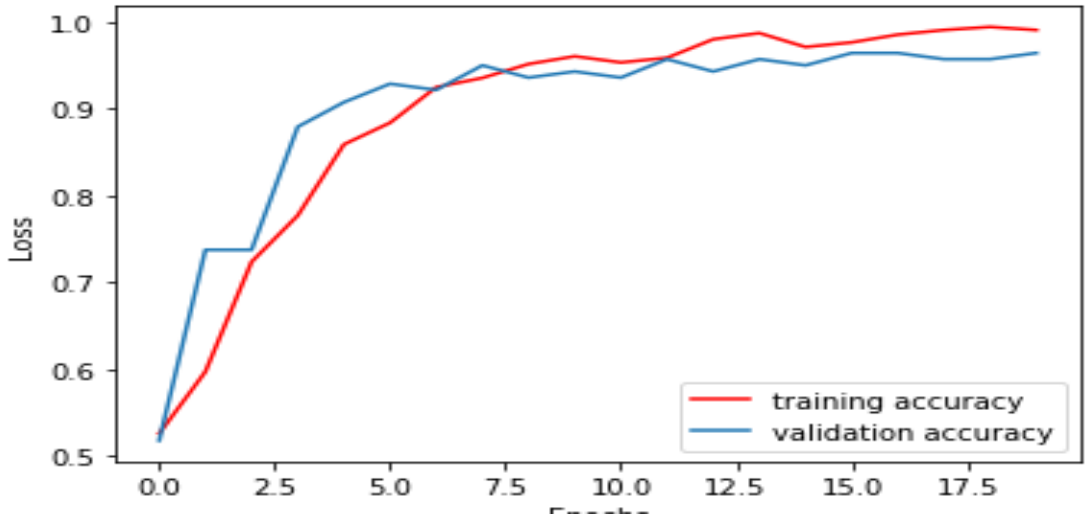


```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 44269  100 44269    0     0   151k      0 --:--:-- --:--:-- --:--:--  151k
```



Results:-Model is working with 3.5% loss and with 89% of validation accuracy. I have trained the model with 20 epochs.





Conclusion:- To make conclude in prevention of covid-19 spread we can further by improving the accuracy of this model we can deploy model on airport,railways stations for surveillances .This model is build with the help of OpenCv, Tesorflow ,Keras important library whether for data preprocessing or image cascading on google colab platform. Google colab provides nearly all kinds of library and features supports which are required for any machine learning model required. It is providing GPU for faster processing.

References

1. Wikipedia
2. Towardsdatascience
3. OpenCv
4. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
5. Stackoverflow
6. https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn