COMP 6721

(Winter 2021)

Project : AI Face Mask Detector

Project Phase - II

Submitted By

Team	**Name**	**Id**	**Specialization**
RN_06	Kaustav Mondal	40115265	Data Specialist
	Tanvi Nakhawa	40151711	Training Specialist
	Md Manik Hossain	40108891	Evaluation Specialist

INTRODUCTION

Covid-19 pandemic has made it mandatory for people to wear masks at all the public places. Hence, efficient AI systems are required to ensure people are wearing the masks and keeping the risk of spreading this virus low. We are designing an AI face mask detector that analyses the images and detects whether the person is wearing a face mask or not. In order to gain the accurate output, Deep learning CNN (Convolutional Neural Network) is a very intuitive and efficient library because we can tune the variables of the model.

DATASET

Before training the data, we collected the data for all the classes for our model to feed into. Then we did some preprocessing of the data. For example, some images in the dataset were corrupted and were throwing the error as "UnidentifiedImageError", so we manually deleted those images from the dataset. Then we performed the preprocessing, created and trained the model (i.e., CNN model) and fed the test images and finally evaluated the accuracy, precision, recall, F1- measure, and the confusion matrix.

Our datasets consist of three classes, they are, images of a person with a mask (*with_mask*), images of a person not wearing a mask (*without_mask*), and all kinds of random images. We named them as *not_a_person*. We have created a table below for describing our datasets (i.e., with_mask, without_mask, not_a_person) in detail.

Item	with_mask	without_mask	not_a_person
References	Dataset source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7775036/ SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2 Preeti Nagrath,Rachna Jain,Agam Madan,Rohan Arora,Piyush Kataria,and Jude Hemanthb https://www.google.com/search?q=face&tbm=isch&ved=2ahUKEwj0g-23-snvAhUFBd8KHTWjCEEQ2-cCegQIABAA&oq=face&gs_lcp=CgNpbWcQAzIECCMQJzIHCAA		https://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html https://www.google.com/search?q=face&tbm=isch&ved=2ahUKEwj0g-23-snvAhUFBd8KHTWjCEEQ2-cCegQIABAA&oq=face&gs_lcp=CgNpbWcQAzIECCMQJzIHCAA
Number of Images	1269 in training set 310 in testing set	1368 in training set 318 in testing set	1314 in training set 350 in testing set
Folder Size	45 MB - 1GB 5MB-10MBin test	90MB - 1GB 5MB-10MBin test	85MB - 1GB 5-10MB in test

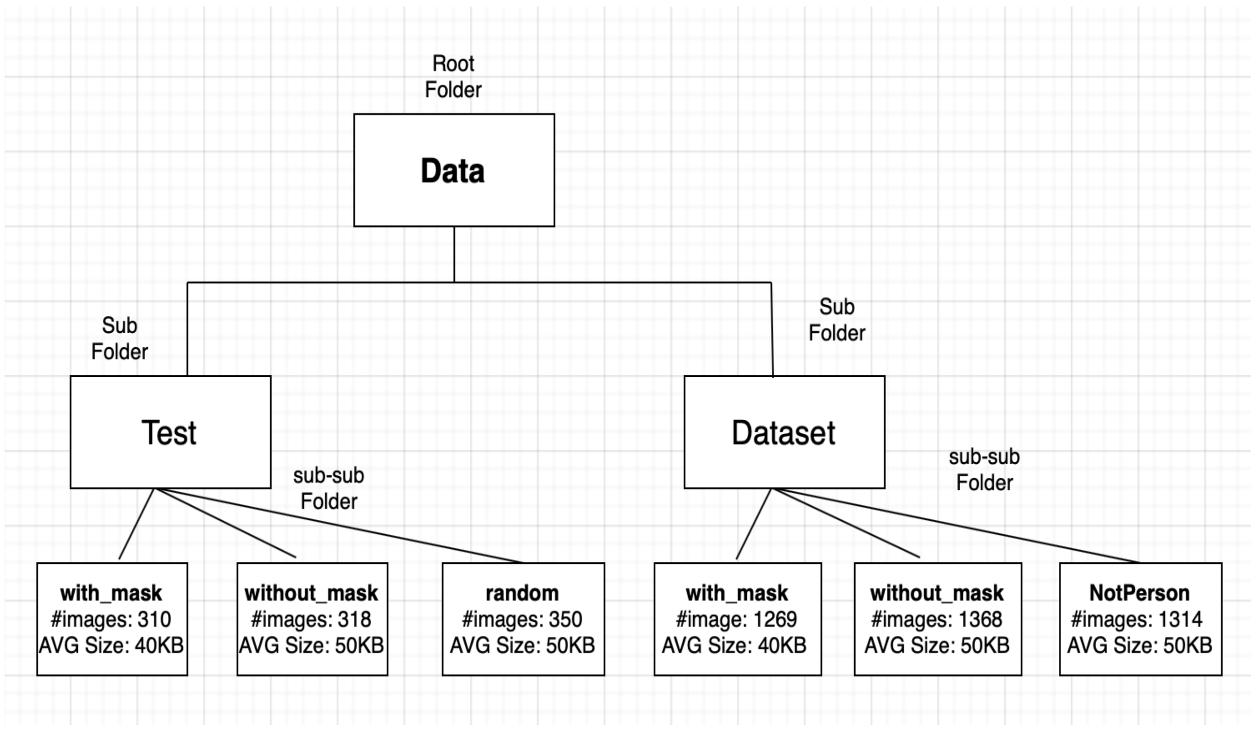

Root Folder
Data
Sub Folder
Sub Folder
Test
Dataset
sub-sub Folder
sub-sub Folder
with_mask #images: 310 AVG Size: 40KB
without_mask #images: 318 AVG Size: 50KB
random #images: 350 AVG Size: 50KB
with_mask #image: 1269 AVG Size: 40KB
without_mask #images: 1368 AVG Size: 50KB
NotPerson #images: 1314 AVG Size: 50KB

Fig-1: Datasets structure and statistics

We organized several steps to make sure that the data size is uniform throughout the all three classes, before working with the dataset for our Convolution Neural Network (CNN).

Data Statistics

Figure-1 depicts the statistics of the datasets and overall structure of the datasets

Data Transformation

Every image in the datasets is resized to 32*32 for having consistent sized images to train and test the model. The PIL images are converted to tensor. The tensor images are then normalised by mean and standard deviation. It does not support a PIL image.

CNN ARCHITECTURE

The architecture of CNN and how the model has been trained is discussed below in detail. In our CNN model, the first two layers are convolutional, followed by the two linear layers, as explained below.

a) Convolutional: The convolutional layer applies a 2D convolution over an input image. A weight matrix extracts certain features of images , the weights are calculated to minimize the loss function.

1. Convolutional Layer I

There are 3 input features per pixel (RGB) and 16 output channels in the first convolutional layer. The kernel size is set to 3 with padding set to 1.With these settings extra empty pixels are added to prevent the output in the corner from being missed.

2. Convolutional Layer II

Kernel Size: 3, Padding: 1 is set like in the convolution layer 1. In this layer we have 16 input channels and 8 output channels. The output of the first convolution layer acts as an input for the second convolution layer.

b) Linear: In this layer, the input images are linearly transformed.

1. Linear Layer I

Linear layer 1 takes 8*8*8 input features and 32 output features. It uses the output of the convolution layer.

2. Linear Layer II

This layer mainly provides the chances which determine which class the image belongs to.

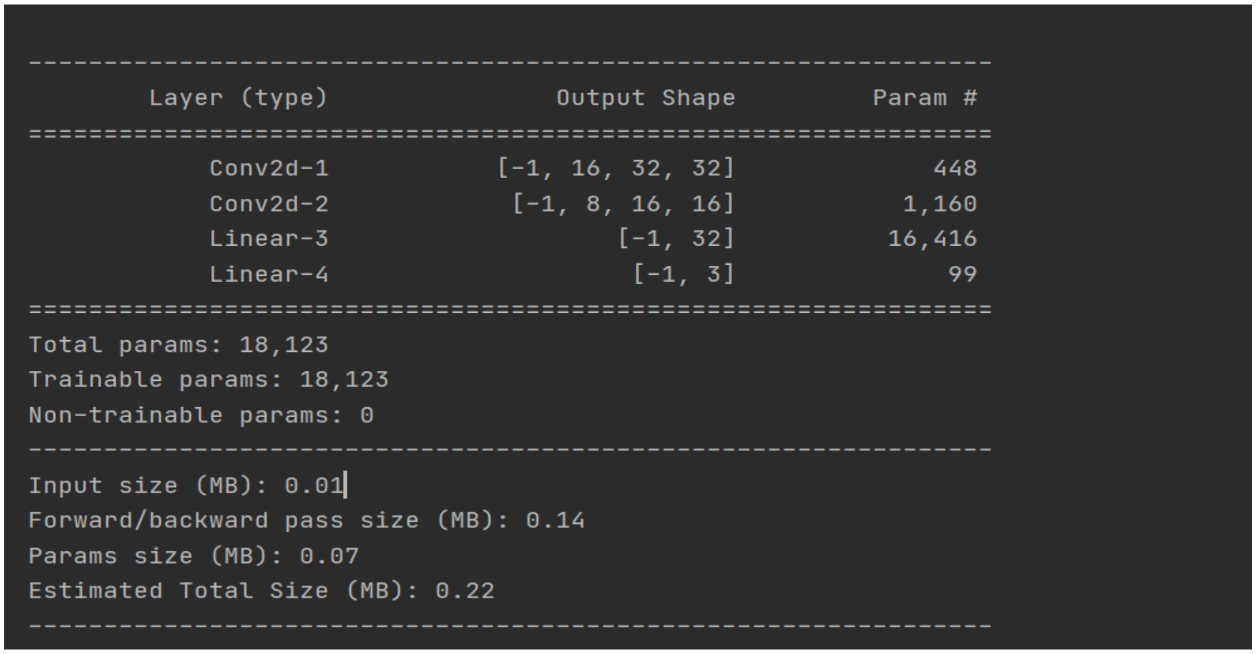
Loss

Cross Entropy is used to calculate the training loss. It gives the loss i.e. the difference between the actual value and predicted value.

Forward Pass
The information moves in the forward direction taking the output of the first layer as input to the next one.

Backward propagation:
Back-propagation propagates the total loss back into the neural network to update the weights in such a way that it minimizes the loss.

Max Pooling
The output after **max-pooling** is a feature map containing the most prominent features of the previous feature map.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 32, 32]             448
            Conv2d-2            [-1, 8, 16, 16]           1,160
            Linear-3                   [-1, 32]          16,416
            Linear-4                    [-1, 3]              99
================================================================
Total params: 18,123
Trainable params: 18,123
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 0.14
Params size (MB): 0.07
Estimated Total Size (MB): 0.22
----------------------------------------------------------------
```

K-fold Cross-validation:

In k fold cross validation, a parameter k is used to split the dataset into k groups. Here our value of k is 10.

The Stratified k-Fold technique is used so that the training and test dataset have data from all the three classes 'masked', 'unmasked' and 'not a person'.

How does cross validation work :

(1) Randomly shuffles the images in the dataset

(2) Split the dataset into k=10 groups

As mentioned earlier, StratifiedKFold takes images from all the classes of the dataset to ensure generality.

EVALUATION

Metrics used to evaluate the model are:

Accuracy : It is the percentage of testset images that the algorithm correctly predicted.

Confusion matrix: It is used to summarise the performance of the classification algorithm.

Precision: It is the ratio of true positives to all the positive outcomes from the algorithm.

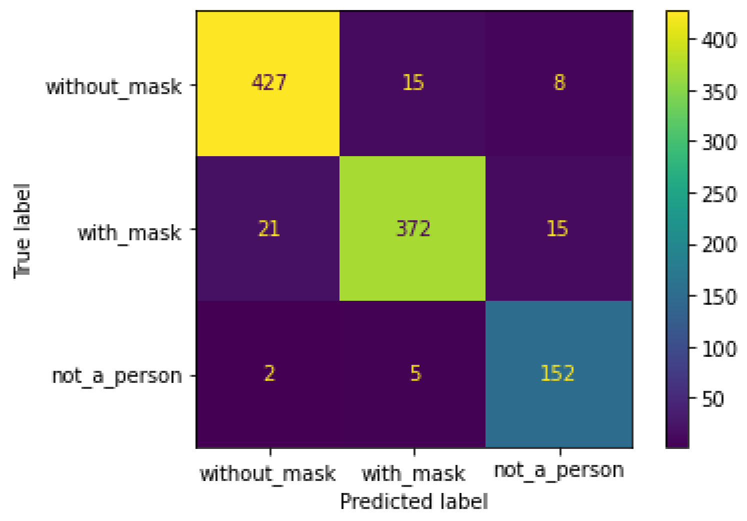
Recall: It is the ratio of true positive to positive and negative ie total observations of the algorithm.

F1: F1 measure is a weighted combination of precision and recall.

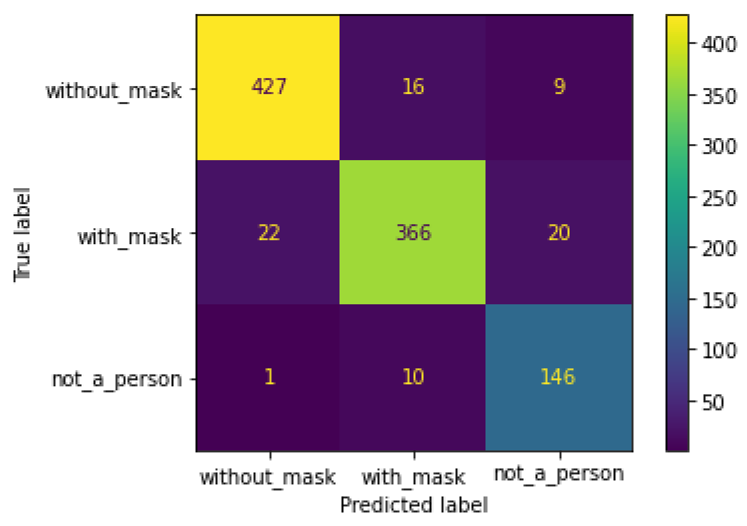
K FOLD VALIDATION AND MODELING

```
Training Loss for 0):  tensor(126.3102, grad_fn=<AddBackward0>)
Training Loss for 1):  tensor(63.8171, grad_fn=<AddBackward0>)


Accuracy =  93.5103244837758
           precision   recall  f1-score   support

        0      0.95      0.95      0.95       450
        1      0.91      0.95      0.93       392
        2      0.96      0.87      0.91       175
```

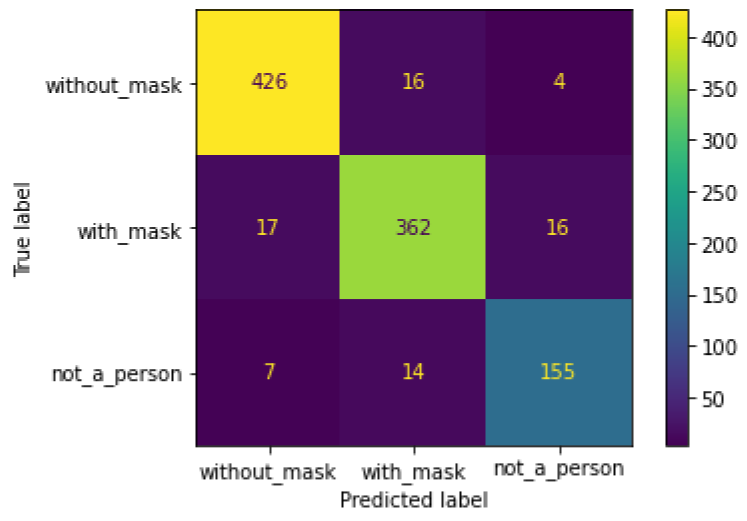
```
   accuracy                          0.94      1017
   macro avg       0.94      0.92      0.93      1017
weighted avg       0.94      0.94      0.93      1017
```



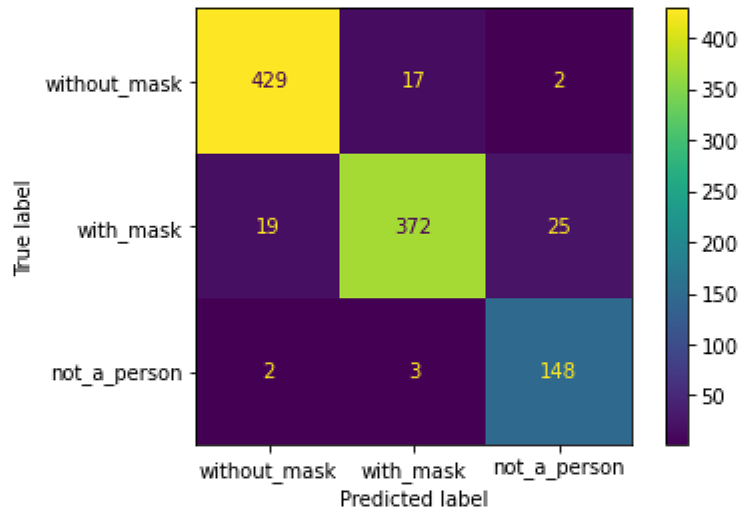
```
Training Loss for 0):  tensor(124.6122, grad_fn=<AddBackward0>)
Training Loss for 1):  tensor(68.1968, grad_fn=<AddBackward0>)


Accuracy =  92.33038348082596
              precision    recall  f1-score   support

           0       0.94      0.95      0.95       450
           1       0.90      0.93      0.92       392
           2       0.93      0.83      0.88       175

    accuracy                           0.92      1017
   macro avg       0.92      0.91      0.91      1017
weighted avg       0.92      0.92      0.92      1017
```



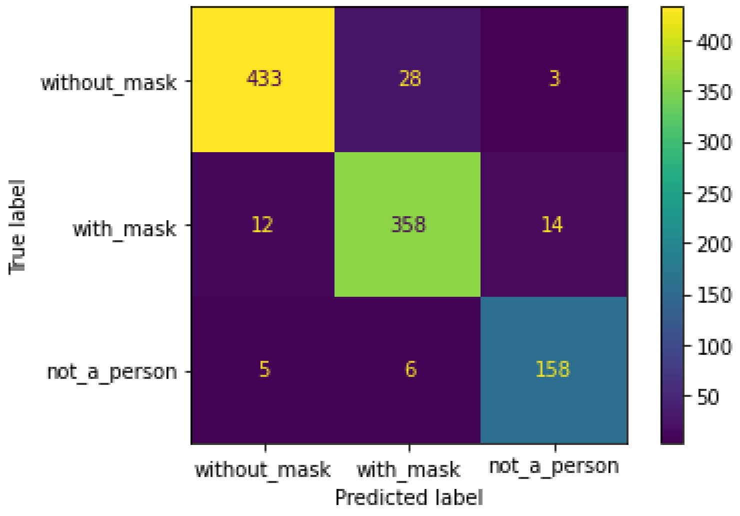
```
Training Loss for 0):  tensor(124.8892, grad_fn=<AddBackward0>)
Training Loss for 1):  tensor(63.5061, grad_fn=<AddBackward0>)


Accuracy =  92.72369714847592
              precision    recall  f1-score   support

           0       0.96      0.95      0.95       450
           1       0.92      0.92      0.92       392
           2       0.88      0.89      0.88       175

    accuracy                           0.93      1017
   macro avg       0.92      0.92      0.92      1017
weighted avg       0.93      0.93      0.93      1017
```



```
Training Loss for 0):  tensor(140.7497, grad_fn=<AddBackward0>)
Training Loss for 1):  tensor(67.5643, grad_fn=<AddBackward0>)


Accuracy =  93.31366764995084
          precision    recall  f1-score   support

       0       0.96      0.95      0.96       450
       1       0.89      0.95      0.92       392
       2       0.97      0.85      0.90       175

   accuracy                           0.93      1017
  macro avg       0.94      0.92      0.93      1017
weighted avg       0.93      0.93      0.93      1017
```

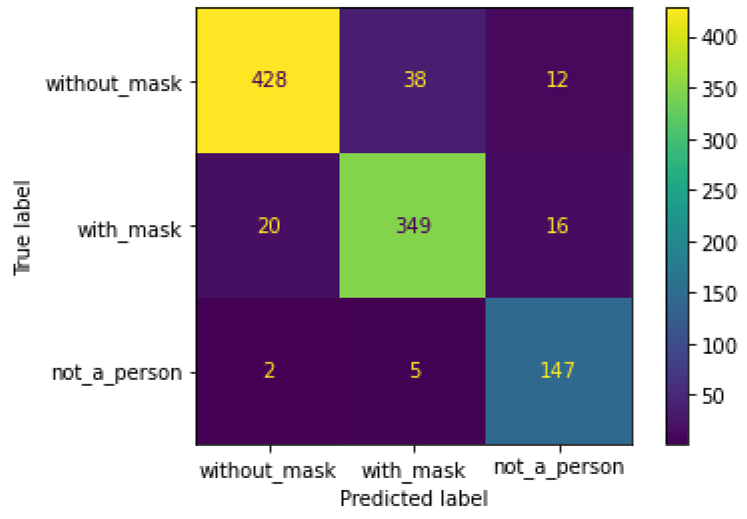


```
Training Loss for 0):  tensor(121.0871, grad_fn=<AddBackward0>)
Training Loss for 1):  tensor(65.2400, grad_fn=<AddBackward0>)


Accuracy =  93.31366764995084
          precision    recall  f1-score   support

       0       0.93      0.96      0.95       450
       1       0.93      0.91      0.92       392
       2       0.93      0.90      0.92       175

   accuracy                           0.93      1017
  macro avg       0.93      0.93      0.93      1017
weighted avg       0.93      0.93      0.93      1017
```

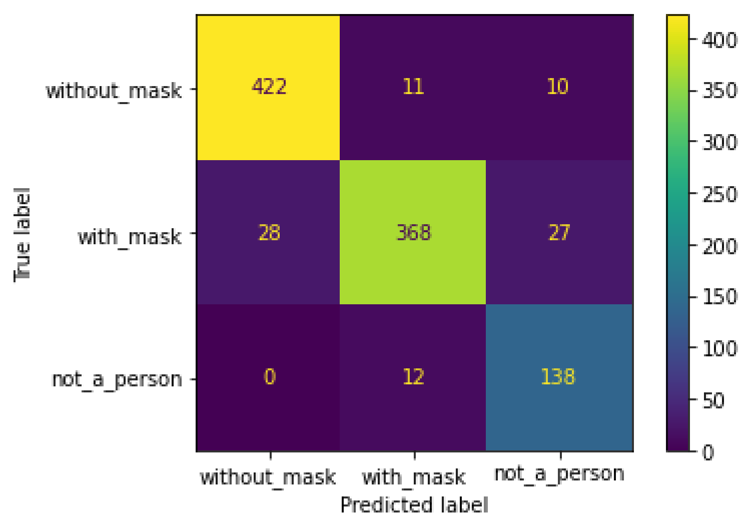


```
Training Loss for 0):  tensor(128.1167, grad_fn=<AddBackward0>)
Training Loss for 1):  tensor(70.4787, grad_fn=<AddBackward0>)


Accuracy =  90.85545722713864
          precision    recall  f1-score   support

       0       0.90      0.95      0.92       450
       1       0.91      0.89      0.90       392
       2       0.95      0.84      0.89       175

   accuracy                           0.91      1017
  macro avg       0.92      0.89      0.90      1017
```

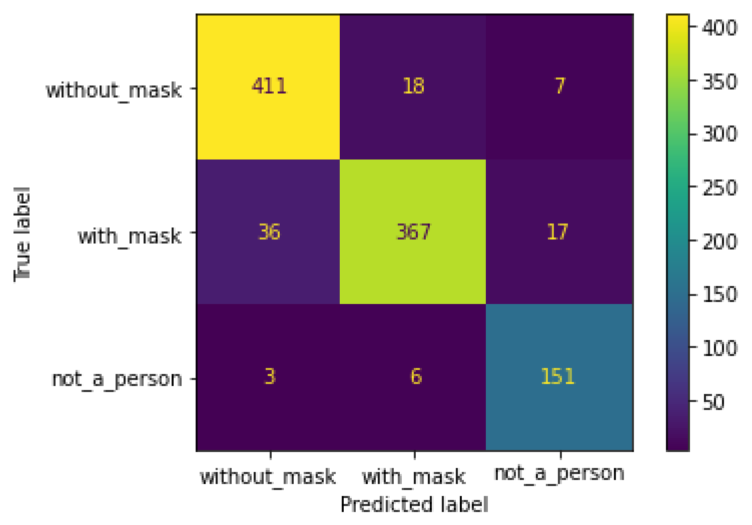
```
weighted avg       0.91       0.91       0.91       1017
```



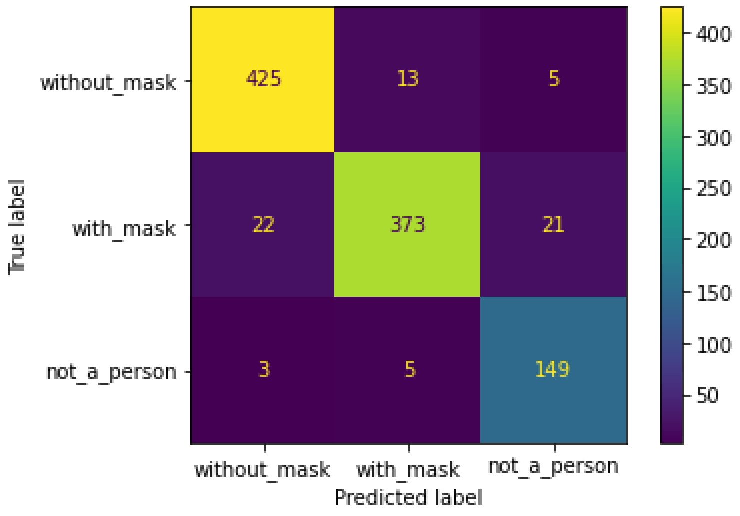
```
Training Loss for 0):  tensor(122.5199, grad_fn=<AddBackward0>)
Training Loss for 1):  tensor(65.8090, grad_fn=<AddBackward0>)


Accuracy =  91.33858267716536
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       450
           1       0.87      0.94      0.90       391
           2       0.92      0.79      0.85       175

    accuracy                           0.91      1016
   macro avg       0.91      0.89      0.90      1016
weighted avg       0.92      0.91      0.91      1016
```



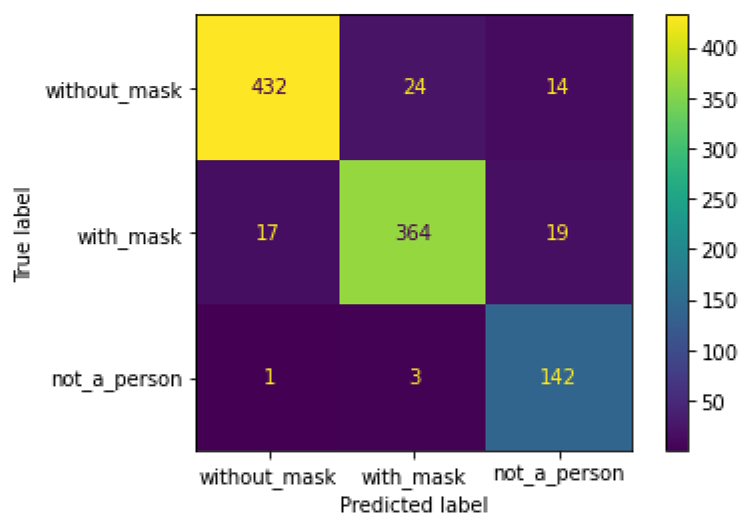
```
Training Loss for 0):  tensor(126.2384, grad_fn=<AddBackward0>)
Training Loss for 1):  tensor(67.4460, grad_fn=<AddBackward0>)


Accuracy =  91.43700787401575
             precision    recall  f1-score   support

         0       0.94      0.91      0.93       450
         1       0.87      0.94      0.91       391
         2       0.94      0.86      0.90       175

    accuracy                           0.91      1016
   macro avg       0.92      0.90      0.91      1016
weighted avg       0.92      0.91      0.91      1016
```



```
Training Loss for 0):  tensor(129.5401, grad_fn=<AddBackward0>)
Training Loss for 1):  tensor(63.4137, grad_fn=<AddBackward0>)


Accuracy =  93.20866141732283
              precision    recall  f1-score   support

           0       0.96      0.94      0.95       450
           1       0.90      0.95      0.92       391
           2       0.95      0.85      0.90       175

    accuracy                           0.93      1016
   macro avg       0.94      0.92      0.92      1016
weighted avg       0.93      0.93      0.93      1016
```

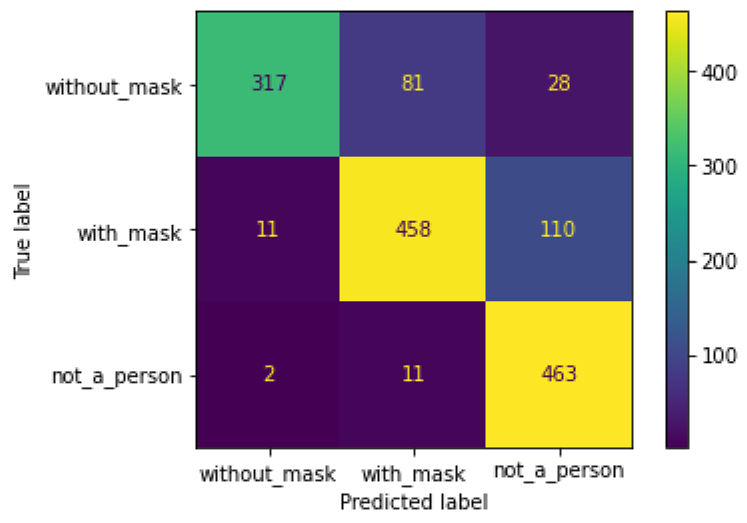


```
Training Loss for 0):  tensor(133.8866, grad_fn=<AddBackward0>)
Training Loss for 1):  tensor(72.0934, grad_fn=<AddBackward0>)


Accuracy =  92.32283464566929
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       450
           1       0.91      0.93      0.92       391
           2       0.97      0.81      0.88       175

    accuracy                           0.92      1016
   macro avg       0.93      0.90      0.91      1016
weighted avg       0.92      0.92      0.92      1016
```



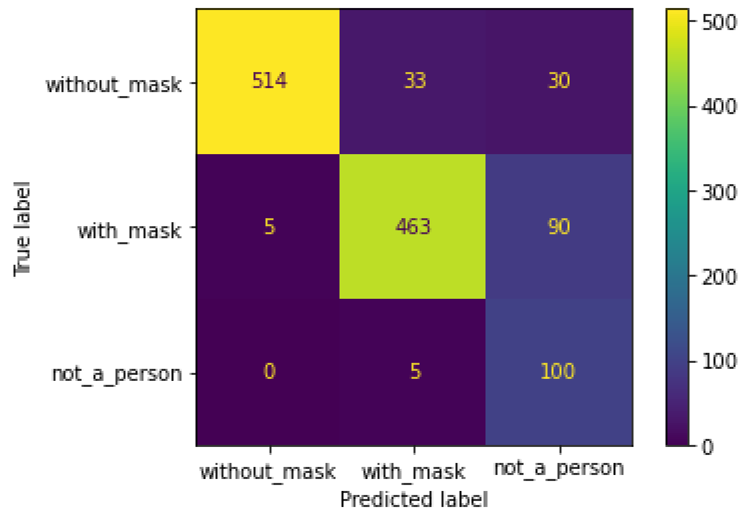
Total testdata metrics

```
Accuracy =  83.59216745442268
             precision    recall  f1-score   support

           0       0.74      0.96      0.84       330
           1       0.79      0.83      0.81       550
           2       0.97      0.77      0.86       601

    accuracy                           0.84      1481
   macro avg       0.84      0.85      0.84      1481
weighted avg       0.85      0.84      0.84      1481
```



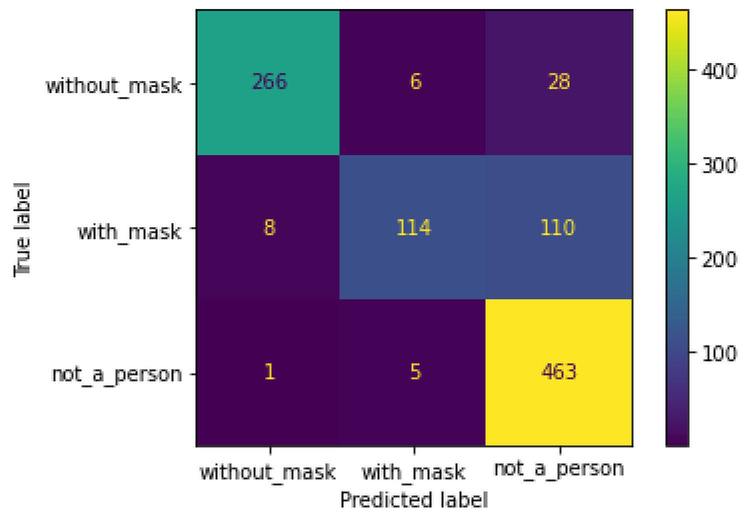
Female testdata metrics

```
Accuracy =  86.85483870967742
              precision    recall  f1-score   support

           0       0.89      0.99      0.94       519
           1       0.83      0.92      0.87       501
           2       0.95      0.45      0.62       220

    accuracy                           0.87      1240
   macro avg       0.89      0.79      0.81      1240
weighted avg       0.88      0.87      0.86      1240
```



Male testdata metrics

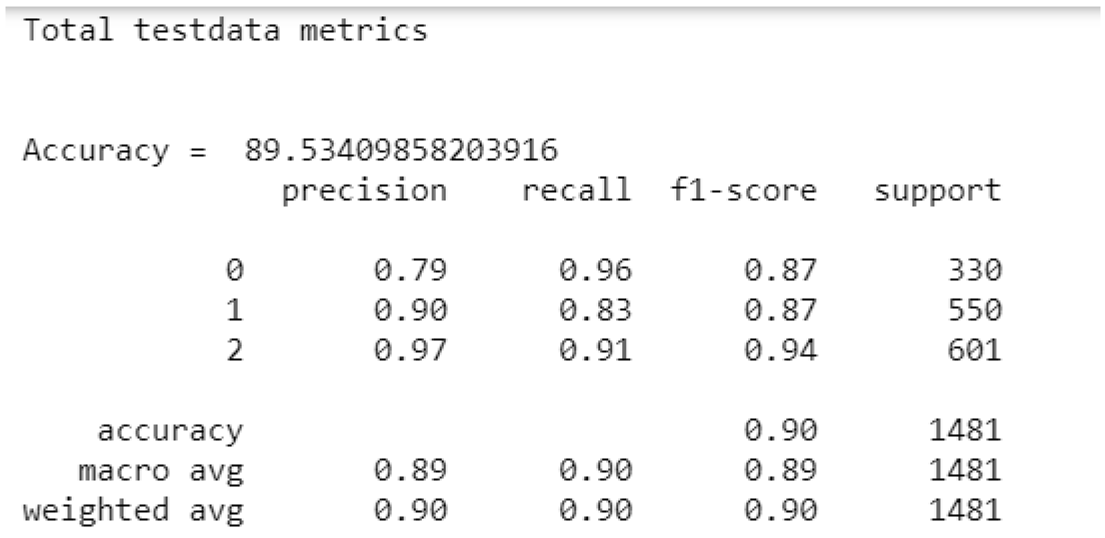
```
Accuracy =  84.21578421578421
              precision    recall  f1-score   support

           0       0.89      0.97      0.93       275
           1       0.49      0.91      0.64       125
           2       0.99      0.77      0.87       601

    accuracy                           0.84      1001
   macro avg       0.79      0.88      0.81      1001
weighted avg       0.90      0.84      0.85      1001
```



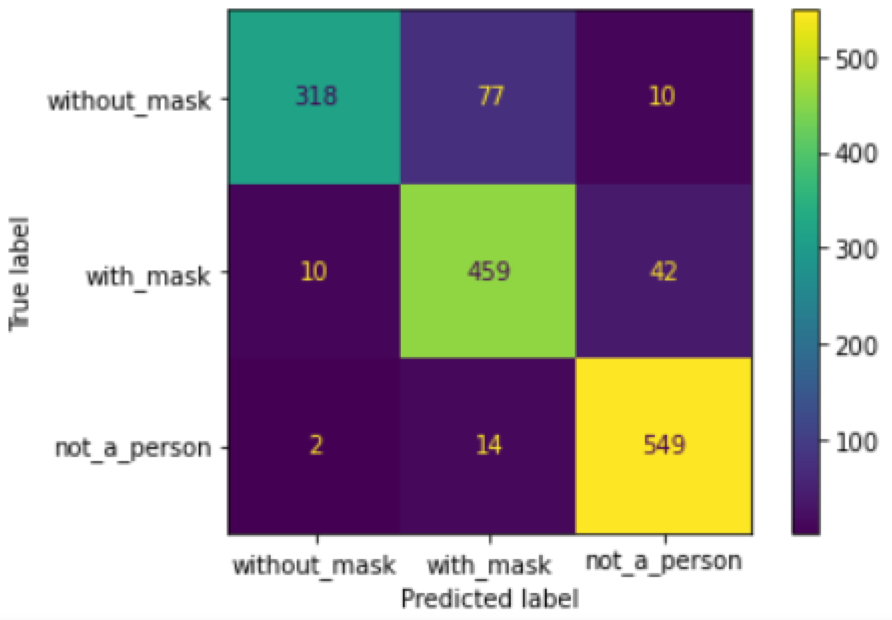
BIAS DETECTION & ELIMINATION

To check the Bias of our model we have followed the below approaches,

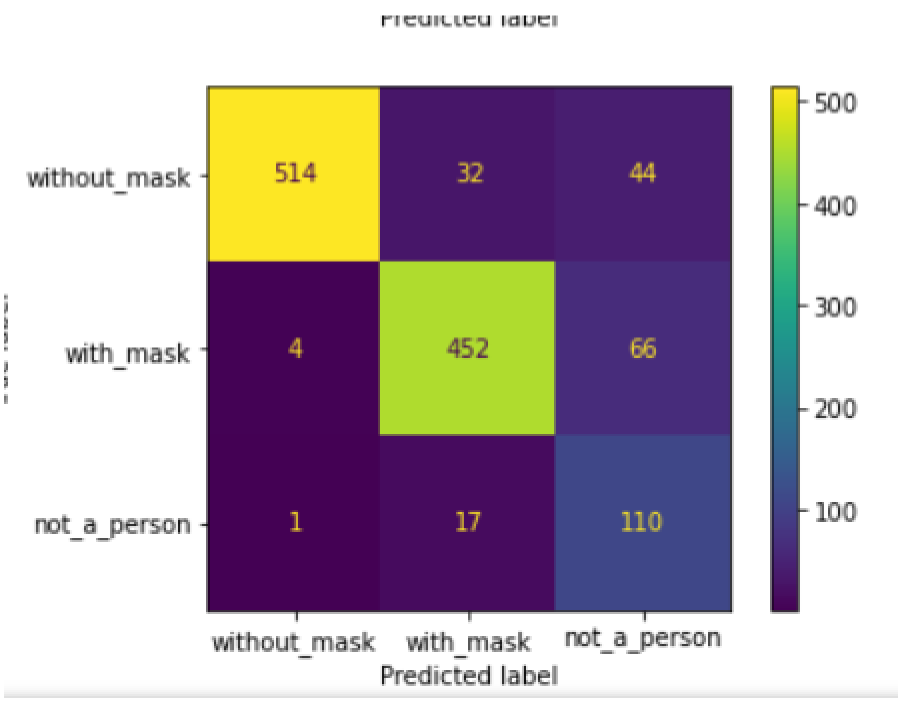
1. We have tested the model separately on Male and Female images and then observed the results. The accuracy was biased.
2. Since the results were not satisfied after the first few iterations of testing, we decided to keep an equal number of male and female images in our dataset. This improved the performance of our model which resulted in better test results while testing the model on both male and female images.



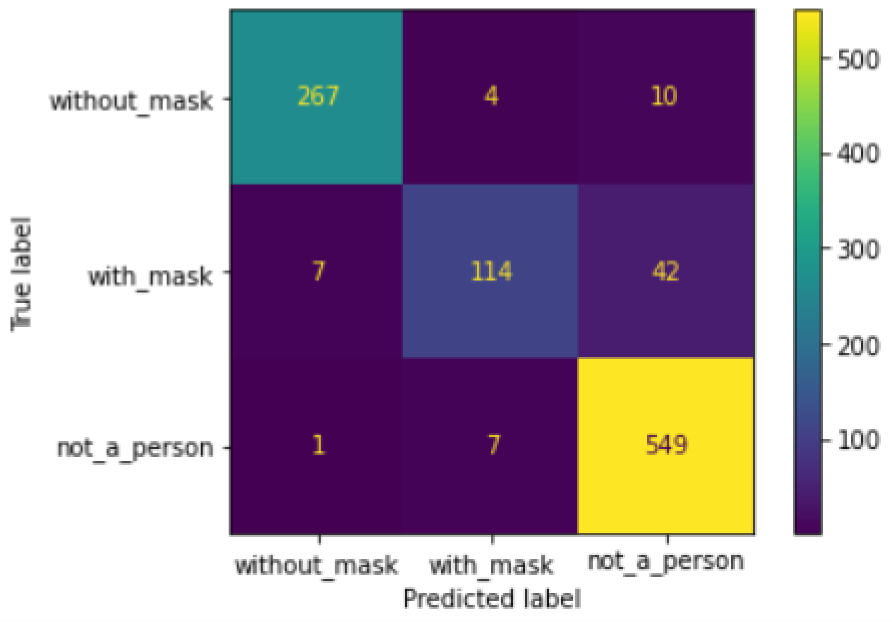
```
Total testdata metrics


Accuracy =  89.53409858203916
              precision    recall  f1-score   support

           0       0.79      0.96      0.87       330
           1       0.90      0.83      0.87       550
           2       0.97      0.91      0.94       601

    accuracy                           0.90      1481
   macro avg       0.89      0.90      0.89      1481
weighted avg       0.90      0.90      0.90      1481
```



```
Female testdata metrics


Accuracy =  86.7741935483871
              precision    recall  f1-score   support

           0       0.87      0.99      0.93       519
           1       0.87      0.90      0.88       501
           2       0.86      0.50      0.63       220

    accuracy                           0.87      1240
   macro avg       0.87      0.80      0.81      1240
weighted avg       0.87      0.87      0.86      1240
```



```
Male testdata metrics


Accuracy =  92.9070929070929
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       275
           1       0.70      0.91      0.79       125
           2       0.99      0.91      0.95       601

    accuracy                           0.93      1001
   macro avg       0.88      0.93      0.90      1001
weighted avg       0.94      0.93      0.93      1001
```



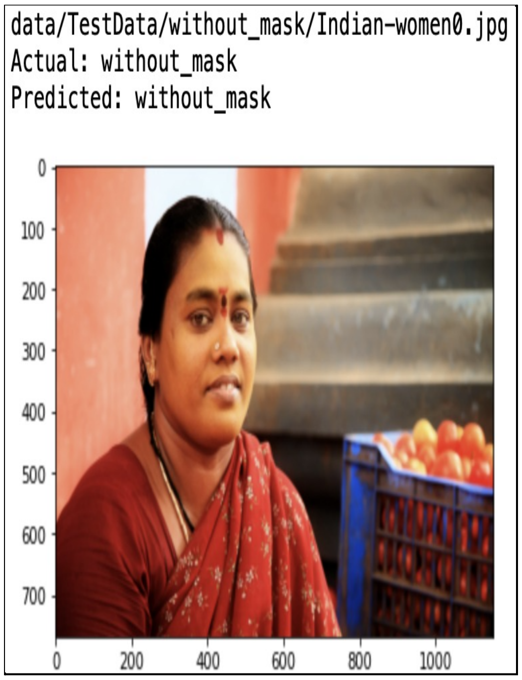
Changes made After Phase I:

In our previous data set the images contained only faces which made our data set very uniform but this time we have added images with full body and our model performed really well to classify on this data set. Besides, we added many images of people which were captured from different angles. There are many images where the face is almost covered by a hat spectacle. So in all together our current dataset contains a variety of images of people wearing masks. Also, there are images of people (both male and female) from different geographical parts of the world.

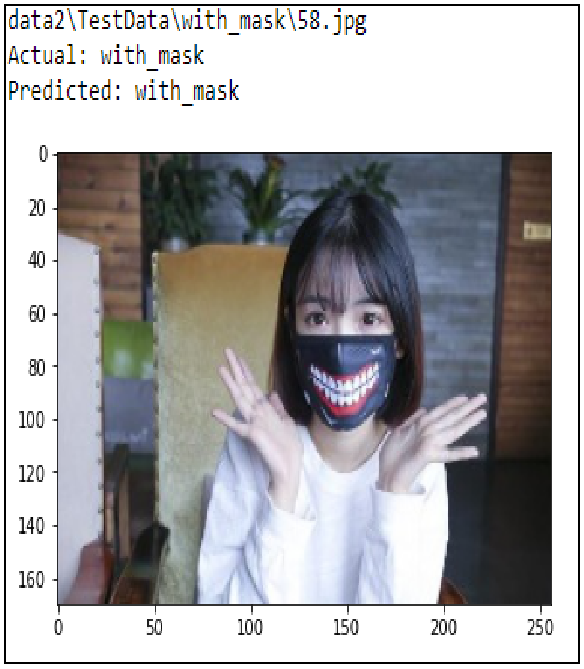
Also, in our previous data set there were only one type of images in ‘notperson’ dataset but this time we have added various types of images like images of animals, houses, books, vegetables etc.

After training our model on this new data set we could achieve very good results when the model was tested on different test data.

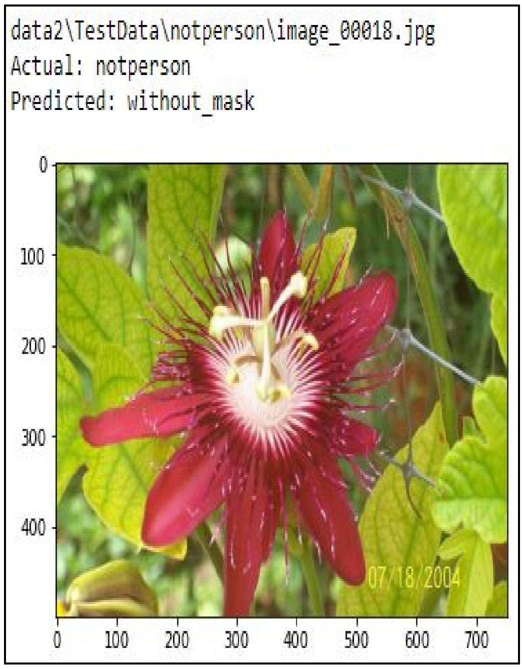
Few Sample Results :



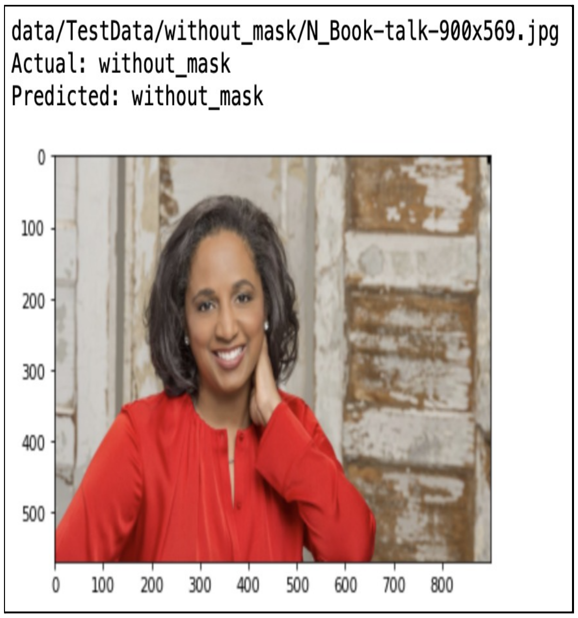
```
data/TestData/without_mask/Indian-women0.jpg
Actual: without_mask
Predicted: without_mask
```



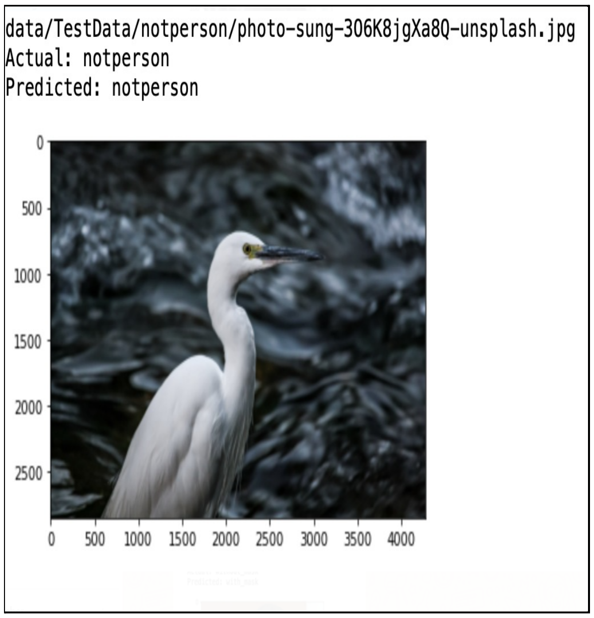
```
data2\TestData\with_mask\58.jpg
Actual: with_mask
Predicted: with_mask
```



```
data2\TestData\notperson\image_00018.jpg
Actual: notperson
Predicted: without_mask
```



```
data/TestData/without_mask/N_Book-talk-900x569.jpg
Actual: without_mask
Predicted: without_mask
```



```
data/TestData/notperson/photo-sung-306K8jgXa8Q-unsplash.jpg
Actual: notperson
Predicted: notperson
```

CONCLUSION

In this phase, we were able to find out that our model was biased towards gender centric images, like our model ran better for male images but not for the female images. We were successful in minimizing this bias and making our model more generalised. Also we were able to successfully implement stratified K FOLD CROSS VALIDATION.

REFERENCES

1. **https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/#:~:text=Precision%20%2D%20Precision%20is%20the%20ratio,the%20total%20predicted%20positive%20observations.&text=F1%20score%20%2D%20F1%20Score%20is,and%20false%20negatives%20into%20account.**
2. **https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/**
3. **https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html**
4. **https://pytorch.org/ignite/metrics.html**
5. **https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html**
6. **https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html**
7. **https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7775036/**
8. **https://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html**
9. **https://pypi.org/project/tqdm/**
10. **https://pytorch.org/docs/stable/optim.html**
11. **https://pytorch.org/vision/stable/transforms.html**

12. ttps://www.educative.io/edpresso/how-to-create-a-confusion-matrix-in-python-using-scikit-learn
13. https://machinelearningmastery.com/k-fold-cross-validation/
14. https://towardsdatascience.com/understanding-and-reducing-bias-in-machine-learning-6565e23900ac