CONCORDIA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

COMP 6721 Applied Artificial Intelligence (Winter 2021)

Project Assignment, Part 1

AI Face Mask Detector

Submitted By:
Piyush Kumar (40119786)

Submitted To:
Dr. René Witte

Dataset

To create our dataset, we have collected images from different sources and divided them in two sets – a training set and a testing set. There are three classes defined in the project – person with face mask, person without face mask and not a person. The training set contains 1100 images for classes and 1150 for class NotPerson thus having a total of 3300 images whereas there are 600 images in the testing set with 200 corresponding to each of the three classes.

[10] , [11] were used to get images for 'Not a Person' class while the images for the other two classes were obtained from [7], [8], [9]. A random chunk of images was picked from different datasets for each class. By doing so, the created dataset has almost the same number of images in all the three classes thus avoiding the problem of data imbalance. This composition of the three classes is explained in detail in the attached file.

Before being given as input to our model, some transformations were applied to the images which are described below:

- **Resizing:** The images did not have the same resolution as they were collected from different sources. So, we resized all of them to 64×64 pixels.
- **Converting image to torch tensor:** The RGB images are converted to torch tensors so that the value of each channel is now in the range [0, 1] instead of [0, 255] as for the original images[13].
- **Normalizing:** The tensor images are normalized using mean and standard deviation. For each channel, the mean value is subtracted from the channel value and then divided by the standard deviation[13].

After these transformations, the data can be given as input to the trained model.

Architecture of CNN

For the implementation of CNN, the core idea is to extract features out of the images using the pixel intensity present in the images. This data is quite huge to store for each feature and each image, hence filter/kernel is used to retain the crucial pixel information. Described below are the types of layers we have included in our architecture-

- **Convolutional layer** [1]: Applies a 2D convolution over an input signal composed of several input planes. A small weight matrix is used to extract features from images by acting as a filter. It is done such that the loss is minimum. The network learns the filters when it detects certain shapes which can be used in identification of an object.
- **Pooling Layer**[2]: The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. We have used MaxPooling in our model.
- **Linear Layer**[3]: Applies a linear transformation to the incoming data.

ITERATION 1-

In our CNN model, we have used two Convolutional layer, two max pooling layers and two linear layers which are explained below:

- **Convolutional Layer 1:** *nn.Conv2d(in_channels=3, out_channels=64, kernel_size=5, padding=1)*
  In the first layer, the input channel is 3 as the images have 3 features (RGB format) and output channels used are 64 to detect more features. The filter is specified to be 5, which in PyTorch means a 5x5 kernel. With padding=1, we are making sure to minimize the loss. Stride is 1 as it is not mentioned.
  Input size of image=64*64
  Output size= (size+2*padding-filter)/stride+1= (64+2*1-5) +1=62

- **MaxPooling Layer 1:** *nn.MaxPool2d(kernel_size=2, stride=2)*
  Input size of image=62*62
  Output size= 62/2=31*31

- **Convolutional Layer 2:** *nn.Conv2d(in_channels=64,out_channels=128, kernel_size=3, padding=1,stride=2)*
  Input channels: 64
  Output channels: 128, Kernel Size: 3, Padding: 1, Stride=2
  The output features of the max pooling layer are the input features to the second convolutional layer, hence input channels are 64. Here, we again increase the channels and have smaller kernel, padding and also stride.

Input size of image=31*31
Output size= (size+2*padding-filter)/stride+1= (31+2*1-3)/2+1=16

- **MaxPooling Layer 2:** *nn.MaxPool2d(kernel_size=2, stride=2)*
  Input size of image=16*16
  Output size= 16/2=8*8
- **Linear Layer 1:** The input is 8*8 features each for 128 channels (128*8*8) which are mapped to output of size 1000.
- **Linear Layer 2:** This is the final layer which maps input size 1000 to 3 classes

Loss : Cross Entropy [4] is more suitable here for finding losses as it works effectively for a classification problem.

Optimizer: Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. It was chosen for its rapid loss convergence during training[5].

We have trained the model in 6 epochs, using 1100 images each for Mask and NonMask class and 1150images for class NotPerson. The images were collected from 3-4 different datasets; hence their dimensions were different. For testing, we used 200 images for each class.

Another important hyperparameter that has been set is the learning rate, which for the purpose of the evaluation metrics tabulated, is set to 0.0001.

ITERATION 2-

In order to improve accuracy of the model, we decided to add more layers to our architecture so that we capture more features and loose less crucial information. We added two convolutional layers and a linear layer.So, the model now has four Convolutional layer, two max pooling layers and three linear layers which are explained below:

- **Convolutional Layer 1:** *nn.Conv2d(in_channels=3, out_channels=64, kernel_size=5, padding=1)*
  Input size of image=64*64
  Output size= (size+2*padding-filter)/stride+1= (64+2*1-5) +1=62
- **Convolutional Layer 2:** *nn.Conv2d(in_channels=64, out_channels=64, kernel_size=5, padding=1)*
  Input size of image=62*62
  Output size= (size+2*padding-filter)/stride+1= (62+2*1-5) +1=60

- **MaxPooling Layer 1:** *nn.MaxPool2d(kernel_size=2, stride=2)*

Input size of image=60*60
Output size= 60/2=30*30

- **Convolutional Layer 3:** *nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, padding=1)*
  Input size of image=30*30
  Output size= (size+2*padding-filter)/stride+1= (30+2*1-3) +1=30

- **Convolutional Layer 4:** *nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1,stride=2)*
  Input size of image=30*30
  Output size= (size+2*padding-filter)/stride+1= (30+2*1-3)/2 +1=15

- **MaxPooling Layer 2:** *nn.MaxPool2d(kernel_size=2, stride=2)*
  Input size of image=15*15
  Output size= 15/2=7*7

Training Process

Iteration 1

In the first iteration of the project, the training set consisted of 3350 images and testing set of 600 images This training data had images in roughly equal proportions for each of the three required classes. The evaluation metrics for the model as tabulated in the subsequent section correspond to using 5 epochs of training. Another important hyperparameter that has been set is the learning rate, which for the purpose of the evaluation metrics tabulated, is set to 0.0001.

Iteration 2:

After changing the architecture of the model, the accuracy of the model increased from 90.67% to 92% which can be seen in the evaluation section.

K-fold Cross-validation

For the second iteration, we are improving the evaluation using the 10-fold cross-validation strategy.

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.[18]

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

Steps to cross-validate a model are listed below, [18]

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    1. Take one group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores. We have taken the average of precision, recall and f1-score values of all the folds which can been seen in the evaluation section.

Advantages of k-Fold cross evaluation-

1. It helps to avoid overfitting of the model as in each fold a new training and testing dataset is provided.
2. It helps to predict the behavior of the model on unknown dataset.
3. It reduces bias as we are using most of the data for fitting, and also significantly reduces variance as most of the data is also being used in validation set.

The accuracy of model of previous model with 10-fold cross validation was found to be % and % for the new model.

Bias Analysis-

We analysed our model for possible bias on Gender and Race.

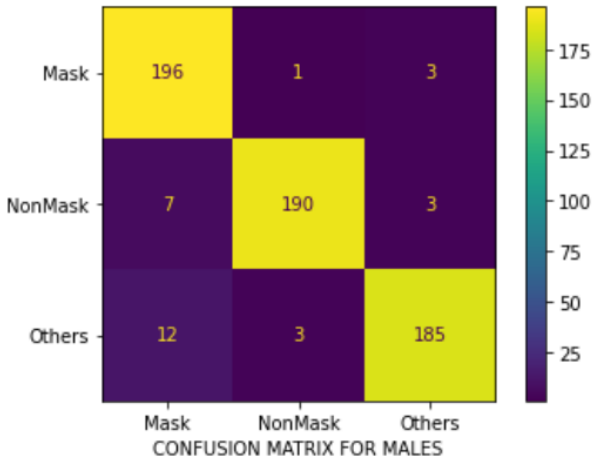
Bias Analysis on Gender

For analysis on gender, we made separate datasets of males and females having 200 images of each category and did evaluation of the model.

Results on male dataset-

Confusion Matrix-

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	196	1	3
NonMask	7	190	3
NotPerson	12	3	185



We decided to analyze bias on the basis of the weighted f1-score.

Classification Report-

Class	Precision	Recall	F1-score
Mask	0.91	0.98	0.94
NonMask	0.98	0.95	0.96
NotPerson	0.97	0.93	0.95

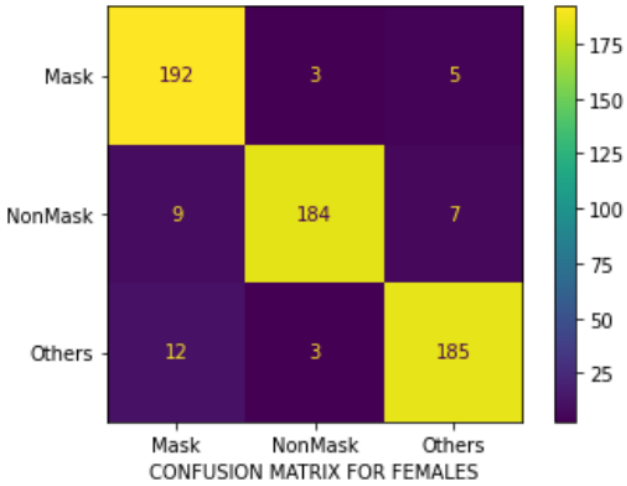
Accuracy=0.9516

F1-score=0.9518

Results on Female dataset-

Confusion Matrix-

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	192	3	5
NonMask	9	184	7
NotPerson	12	3	185



Classification Report-

Class	Precision	Recall	F1-score
Mask	0.90	0.96	0.93
NonMask	0.97	0.92	0.94
NotPerson	0.94	0.93	0.93

Accuracy=0.935

F1-score=0.9351

Bias Analysis Summary on Gender

Dataset	F1-Score
Male	0.9518
Female	0.9351

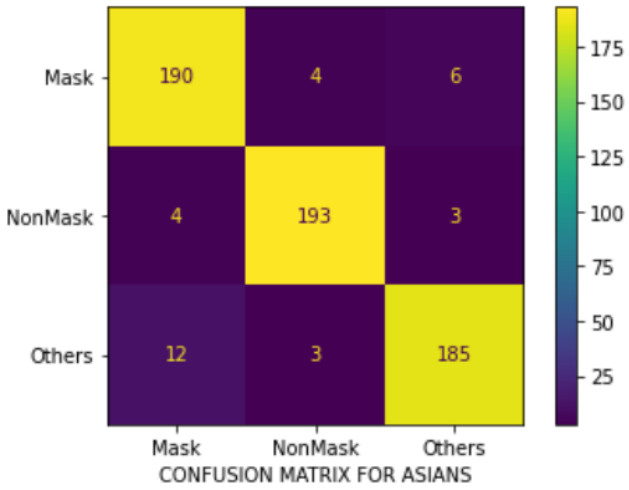
As it is evident that the f1-scores are quite high and are approximately equal for both the datasets, we can say that the model is not gender biased.
We got high f1-scores because of good precision and recall values. This happened because our training data was quite balanced for males and females. As the model got ample images for each gender in all the categories, it didn't learn based on gender.

Bias Analysis on Race
For analysis on Race, we made separate datasets of Asians, Blacks and Whites having 200 images of each category and did evaluation of the model.

Results on Asian dataset

Confusion Matrix-

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	190	4	6
NonMask	4	193	3
NotPerson	12	3	185



Classification Report-

Class	Precision	Recall	F1-score
Mask	0.92	0.95	0.94
NonMask	0.96	0.96	0.96
NotPerson	0.95	0.93	0.94

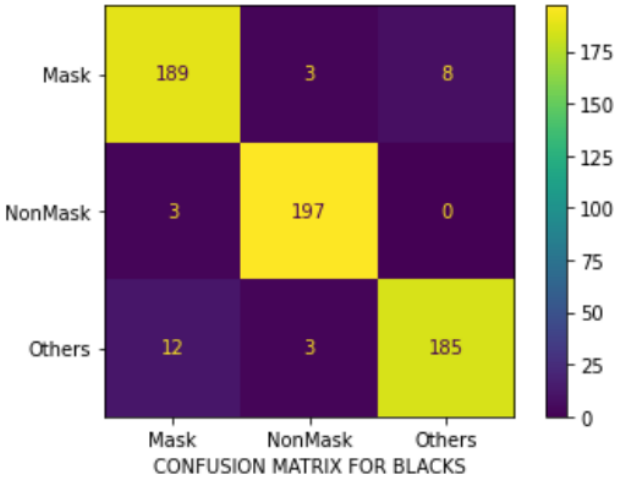
Accuracy=0.9466

F1-Score= 0.9467

Results on Black dataset

Confusion Matrix-

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	189	3	8
NonMask	3	197	0
NotPerson	12	3	185



Classification Report-

Class	**Precision**	**Recall**	**F1-score**
Mask	0.93	0.94	0.94
NonMask	0.97	0.98	0.98
NotPerson	0.96	0.93	0.94

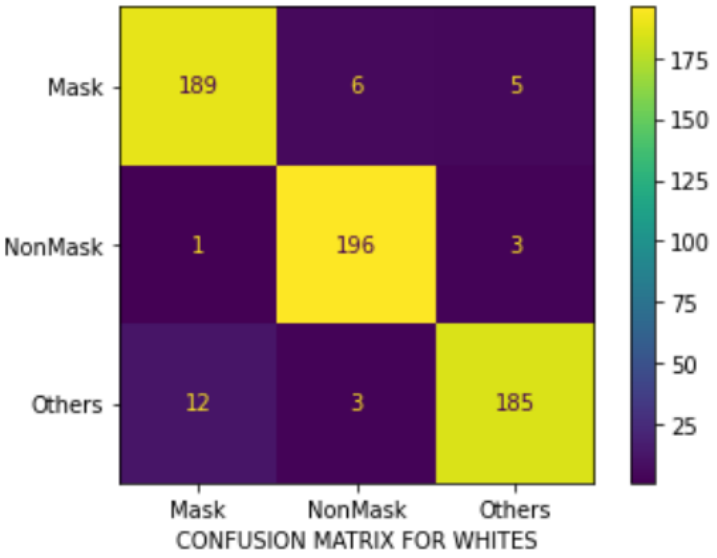
Accuracy=0.9516

F1-Score= 0.9516

Results on White dataset

Confusion Matrix-

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	189	6	5
NonMask	1	196	3
NotPerson	12	3	185



Classification Report-

Class	Precision	Recall	F1-score
Mask	0.94	0.94	0.94
NonMask	0.96	0.98	0.97
NotPerson	0.96	0.93	0.94

Accuracy=0.95

F1-Score= 0.9499

Bias Analysis Summary on Race

Race	F1-Score
Asian	0.9467
Black	0.9516
White	0.9499

As it is evident that the f1-scores are quite high and are approximately equal for all the datasets, we can say that the model is not race biased.

We got high f1-scores because of good precision and recall values. This happened because our training data was quite balanced for all the races. As the model got ample images for each race in all the categories, it didn't learn based on race.

Evaluation

After training the model, the evaluation of results obtained from testing the model on test data provides us insight into the performance of the prepared model. We have tested our model on test data consisting of about 600 images equally divided into each of the three classes. To evaluate the performance of our CNN model, the measures used are – Accuracy, Precision, Recall, F1-Score, and the Confusion Matrix.

Accuracy: [14]Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:
Accuracy = Number of Correct Predictions / Total Number of Predictions

Precision: [15]Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive.

Recall: [16]It is a measure of the classifier's completeness; the ability of a classifier to correctly find all positive instances. For each class, it is defined as the ratio of true positives to the sum of true positives and false negatives.

F1-Score: [16]The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0.

Confusion Matrix: [17]A confusion matrix sorts all cases from the model into categories, by determining whether the predicted value matched the actual value.

ITERATION 1-

Classification Report:
For epochs = 5, we were able to obtain the **accuracy = 0.906**
Total time for training and testing = **607 seconds**

Class	Precision	Recall	F1-score
Mask	0.93	0.85	0.89
NonMask	0.85	0.98	0.91
NotPerson	0.95	0.89	0.91

Confusion Matrix:

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	171	21	8
NonMask	2	196	2
NotPerson	10	13	177

Conclusions:

From the Classification Report and Confusion Matrix obtained after the first evaluation, we have concluded that our model's accuracy is pretty high for the first build and the number of epochs. Class Mask gives the lowest Recall value which shows that our model is the worst at predicting the Mask class. Also, the lowest Precision and Confusion Matrix shows that our model is failing at accurately predicting NonMask class. It is predicting roughly 10% (21) of NonMask images as Mask images, which is not the required trait for our model.

ITERATION 2-

1.Statistics after changing the architecture of the model-

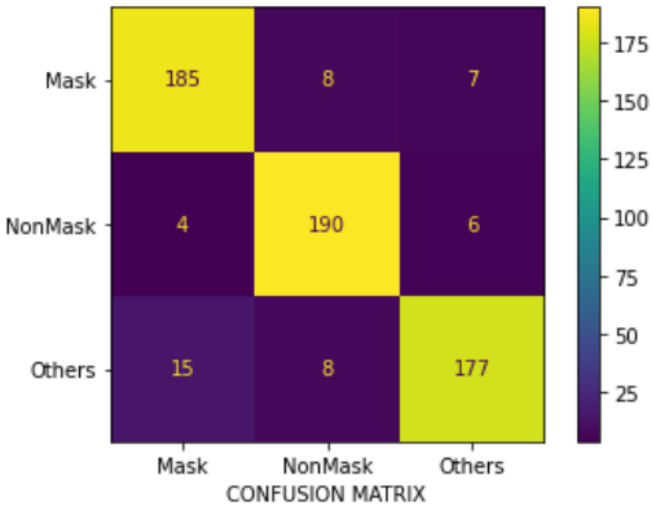
Classification Report:

We were able to obtain the **accuracy = 0.92**

Class	**Precision**	**Recall**	**F1-score**
Mask	0.93	0.93	0.93
NonMask	0.94	0.98	0.96
NotPerson	0.98	0.94	0.96

Confusion Matrix

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	181	8	7
NonMask	4	190	6
NotPerson	15	8	177



2.After implementation of 10-folds on the model of Iteration-1

We were able to obtain the **accuracy = 0.9116**

Fold 1-
Accuracy-0.8626

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	84	10	19
NonMask	1	87	9
NotPerson	1	6	118

Fold 2-
Accuracy-0.9731

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	92	1	3
NonMask	1	110	1
NotPerson	2	1	124

Fold 3-
Accuracy-0.9820

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	108	0	2
NonMask	1	109	0
NotPerson	0	3	112

Fold 4-
Accuracy-0.100

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	117	0	0
NonMask	0	103	0
NotPerson	0	0	115

Fold 5-
Accuracy-0.9731

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	104	1	0
NonMask	1	115	0
NotPerson	2	5	107

Fold 6-
Accuracy-0.9940

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	115	1	0
NonMask	0	105	0
NotPerson	1	0	113

Fold 7-
Accuracy-0.100

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	102	0	0
NonMask	0	125	0
NotPerson	0	0	108

Fold 8-
Accuracy-0.9910

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	108	1	0
NonMask	0	116	0
NotPerson	0	2	108

Fold 9-
Accuracy-0.9820

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	114	0	0
NonMask	1	98	5
NotPerson	0	0	117

Fold 10-
Accuracy-0.100

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	118	0	0
NonMask	0	112	0
NotPerson	0	0	105

******Average Statistics for 10-folds********

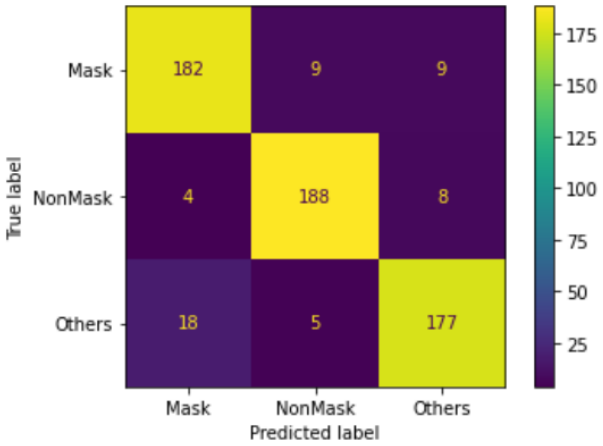
Precision : 0.9773017134262947

Recall : 0.9758

f1-score : 0.9757

Accuracy of the model on test data- 0.9116

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	182	9	9
NonMask	4	188	8
NotPerson	18	5	177



Class	Precision	Recall	F1-score
Mask	0.89	0.91	0.90
NonMask	0.93	0.94	0.94
NotPerson	0.91	0.89	0.90

After using k-fold cross validation, the model's performance increased from iteration-1(90.67%) to 91.16% in terms of the accuracy. The accuracy kept on increasing in each fold.

3.After implementation of 10-folds on the new model-

We were able to obtain the **accuracy = 0.9116**

Fold 1-
Accuracy-0.9074

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	94	7	7
NonMask	6	95	5
NotPerson	2	4	115

Fold 2-
Accuracy-0.9313

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	106	1	2
NonMask	13	111	0
NotPerson	5	2	95

Fold 3-
Accuracy-0.9641

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	123	1	0
NonMask	6	93	0
NotPerson	4	1	107

Fold 4-
Accuracy-0.9940

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	107	0	1
NonMask	0	106	1
NotPerson	0	0	120

Fold 5-
Accuracy-0.9910

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	112	0	0
NonMask	0	111	0
NotPerson	2	1	109

Fold 6-
Accuracy-0.9880

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	117	0	1
NonMask	2	106	1
NotPerson	0	0	108

Fold 7-
Accuracy-0.9761

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	106	3	1
NonMask	2	101	0
NotPerson	2	0	120

Fold 8-
Accuracy-0.9910

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	87	0	1
NonMask	2	116	0
NotPerson	0	0	129

Fold 9-
Accuracy-0.9910

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	120	1	2
NonMask	0	104	0
NotPerson	0	0	108

Fold 10-
Accuracy-0.100

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	100	0	0
NonMask	0	119	0
NotPerson	0	0	116

******Average Statistics for 10-folds********

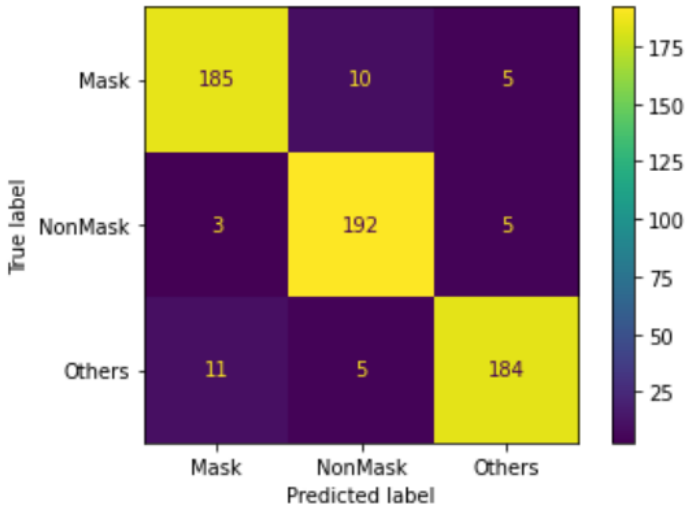
Precision : 0.9742394254464137

Recall : 0.9734

f1-score : 0.9735

Accuracy of the model on test data- 0.9433

Actual \ Predicted	***Mask***	***NonMask***	***NotPerson***
Mask	190	1	9
NonMask	4	191	5
NotPerson	11	4	185



Class	**Precision**	**Recall**	**F1-score**
Mask	0.93	0.95	0.94
NonMask	0.97	0.95	0.96
NotPerson	0.93	0.93	0.93

After using k-fold cross validation, the model's performance increased from iteration-1(90.67%) to 94.33% in terms of the accuracy. The accuracy kept on increasing in each fold.

References

[1] PyTorch Convolutional Layer.
https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html

[2]PyTorch Pooling Layer-
https://pytorch.org/docs/stable/nn.html#pooling-layers

[3] PyTorch Convolutional Layer-
https://pytorch.org/docs/stable/nn.html#linear-layers

[4] PyTorch Cross Entropy-
https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

[5] PyTorch Optimizer-
https://pytorch.org/docs/stable/optim.html

[6] Kaggle: Your Machine Learning and Data Science Community
https://www.kaggle.com/

[7] Face Mask Detection Dataset by *Omkar Gurav*
https://www.kaggle.com/omkargurav/face-mask-dataset

[8] COVID Face Mask Detection Dataset by *Prithwiraj Mitra*
https://www.kaggle.com/prithwirajmitra/covid-face-mask-detection-dataset

[9] Face Mask ~12k Images Dataset by *Ashish Jangra*
https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset

[10] Home Objects dataset by *Caltech*
http://www.vision.caltech.edu/pmoreels/Datasets/Home_Objects_06/

[11] Natural Images by *Prasun Roy*
https://www.kaggle.com/prasunroy/natural-images

[12] A Beginner's Tutorial on Building an AI Image Classifier using PyTorch
https://towardsdatascience.com/a-beginners-tutorial-on-building-an-ai-image-classifier-using-pytorch-6f85cb69cba7

[13] PyTorch Transforms
https://pytorch.org/vision/stable/transforms.html

[14] Accuracy
https://developers.google.com/machine-learning/crash-course/classification/accuracy

[15] Precision
https://medium.com/@kohlishivam5522/understanding-a-classification-report-for-your-machine-learning-model-88815e2ce397

[16] Recall and F1-Score
https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html

[17] Confusion Matrix
https://docs.microsoft.com/en-us/analysis-services/data-mining/classification-matrix-analysis-services-data-mining?view=asallproducts-allversions

[18] k-fold cross validation
https://machinelearningmastery.com/k-fold-cross-validation