COMP 6721 – APPLIED ARTIFICIAL INTELLIGENCE

PROJECT PHASE – II

Team ID :- NS_10

Presented to :-

Dr. René Witte

AI FACE MASK DETECTOR

Authors : -

Harshkumar Nileshkumar Patel (40165709) – Data Specialist

Chirag Hasmukhbhai Patel (40160656) – Evaluation Specialist

Harshil Jayeshbhai Patel (40163431) – Training Specialist

1. Dataset :-

The dataset for accomplishing the task of developing an AI Face Mask Detector is constructed by collecting images from various resources available online like Kaggle and several other web resources[1][2][3][4][5].The dataset contains over 2000 images which is partitioned as follows into 5 categories (class) - Cloth Mask , N95 Mask, N95 Mask with Valve, Surgical Mask, No Mask , each class contains average of 400 images.

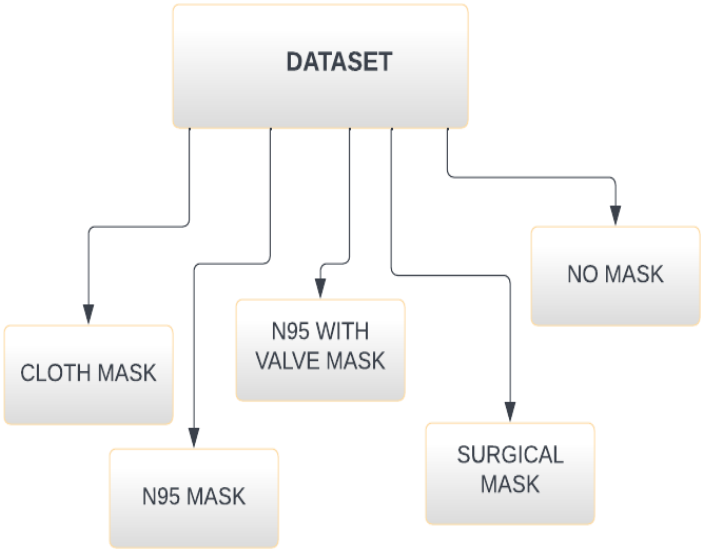

Fig 1.1 Dataset

Now it is necessary to pre-process the dataset before feeding it as input to the CNN and this task is accomplished by following several key points like compressing the images before loading them to train the model, keeping balanced number of images in each class and loading the data using the method load_data , using the raw image data will not provide the desired results therefore it is necessary to scale each image present in the dataset to a specific size and hence tensors of the same dimensions will be obtained which will be fed as input to the model.

Our Dataset contains various images of different dimensions so as a part of pre-processing all the images are scaled to a particular dimension in this case (250 X 250). Also, for uniformness, every image is converted to RGB format and then we apply normalization so that each pixel value falls in similar range.

2. CNN Architecture

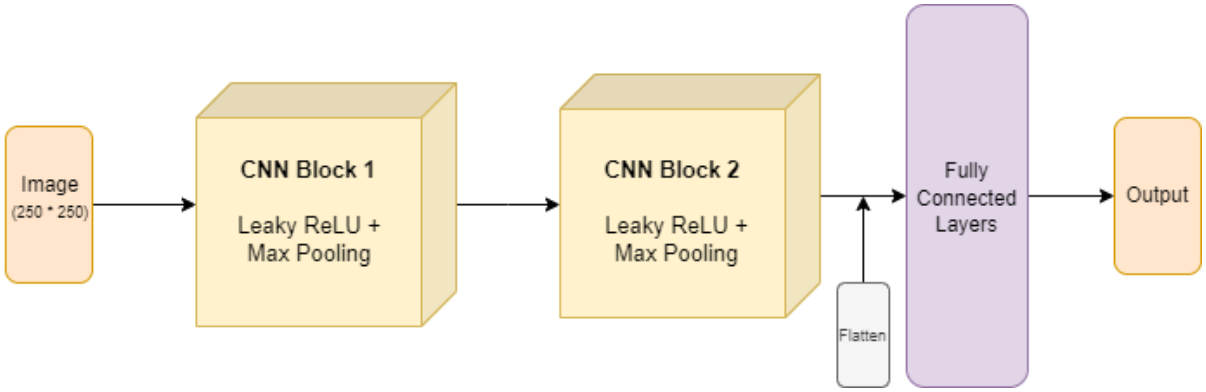

Fig 2.1 CNN Architecture

There are four types of layers for a convolutional neural network: the convolutional layer, the pooling layer, the ReLU correction layer and the fully-connected layer[6].

Convolution Layer :- The purpose of this layer to Its purpose is to detect the presence of a set of features in the images received as input. This is done by convolution filtering: the principle is to "drag" a window representing the feature on the image, and to calculate the convolution product between the feature and each portion of the scanned image. It provides a feature map, which tells us where the features are in the image.

ReLU correction layer :- The ReLU correction layer replaces all negative values received as inputs by zeros. It acts as an activation function.

Pooling Layer :- The pooling operation consists in reducing the size of the images while preserving their important characteristics. The pooling layer reduces the number of parameters and calculations in the network. This improves the efficiency of the network and avoids over-learning.

Fully Connected Layer :- This type of layer receives an input vector and produces a new output vector. To do this, it applies a linear combination and then possibly an activation function to the input values received. A flatten layer is used to flat the tensor which has 3 dimensions. The flatten layer converts the tensor to one-dimensional. Then 3 linear added to reduce the size of the tensor and learn the features[7].

In this project the CNN architecture consists of two CNN blocks and each block contains two convolution layers and one max pooling layer. Here for the purpose of optimization we tried using different optimization algorithm, but ASGD (Average Stochastic gradient descent) gave better performance , it is ordinary stochastic gradient descent that records an average of its parameter vector over time. That is, the update is the same as for ordinary stochastic gradient descent, but the algorithm also keeps track of parameter vector and when optimization is done, this averaged parameter vector takes the place of w

$$\bar{w} = \frac{1}{t}\sum_{i=0}^{t-1} w_i .$$

[8].The pre-set hyperparameter values used are as follows :-

Learning Rate:- 0.01, Epochs :- 30, Batch Size :- 15

3. Bias Detection and Correction

Bias is a prejudice in favour or against a person, group, or a thing that is unfair. A key challenge in modern AI application is the presence of bias in the classifications and predictions of machine learning. It has a great influence on the decision-making capacity of model for certain group. Therefore, it is required to understand and figure out the bias and to eliminate it. By developing a fair Model, the decisions taken by the ML model would be therefore unbiased and would enable much more transparency that would benefit all. [10]

When evaluating the model built in phase-I we found that the model contained bias due to several reasons therefore, to improve the performance of the model we tried to eliminate the biases after detecting them. We used gender and age attributes to detect the bias, for detecting gender bias, the dataset was divided into Male and Female group, and it was found that the model was biased towards Female group and in order to overcome this issue, we took several steps like, we pre-processed the dataset by removing irrelevant images as well as adding new images for all the categories and hence properly balancing the dataset, increased the number of epochs and decreased the learning rate from 0.01 to 0.0001. For detecting age bias, we take into consideration two age groups, Below 30 and Above 30, and it was found that the model did not show too much bias for this attribute, ––but the model was not confused to classify the images between N95 and Cloth Mask, so we removed it by following the steps followed for the gender attribute. The results before and after eliminating the bias are mentioned as below :

3.1 Gender Based Bias

a. Female Bias

Before Eliminating Bias

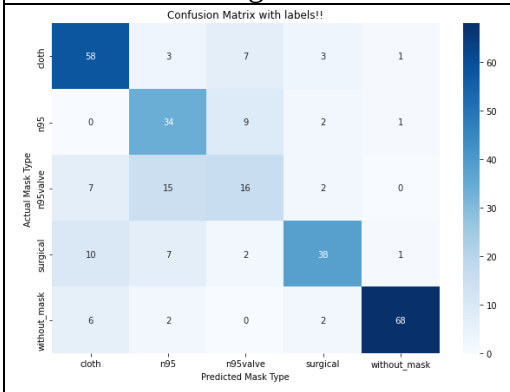

Fig 3.1.1 Confusion Matrix (Gender Bias (Before)) (Female)

After Eliminating Bias

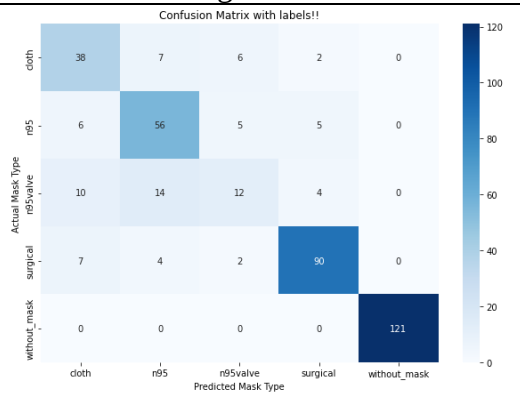

Fig 3.1.2 Confusion Matrix (Gender Bias (After)) (Female)

Test Accuracy :- **72.78 %**

	precision	recall	f1-score	support
0.0	0.72	0.81	0.76	72
1.0	0.56	0.74	0.64	46
2.0	0.47	0.40	0.43	40
3.0	0.81	0.66	0.72	58
4.0	0.96	0.87	0.91	78
accuracy			0.73	294
macro avg	0.70	0.69	0.69	294
weighted avg	0.74	0.73	0.73	294

Fig 3.1.3 Classification Report (Gender Bias (Before)) (Female)

Test Accuracy :- **81.49 %**

	precision	recall	f1-score	support
0.0	0.62	0.72	0.67	53
1.0	0.69	0.78	0.73	72
2.0	0.48	0.30	0.37	40
3.0	0.89	0.87	0.88	103
4.0	1.00	1.00	1.00	121
accuracy			0.81	389
macro avg	0.74	0.73	0.73	389
weighted avg	0.81	0.81	0.81	389

Fig 3.1.4 Classification Report (Gender Bias (Before)) (Female)

Table 3.1.1 Female Bias

b. Male Bias

Before Eliminating Bias	After Eliminating Bias
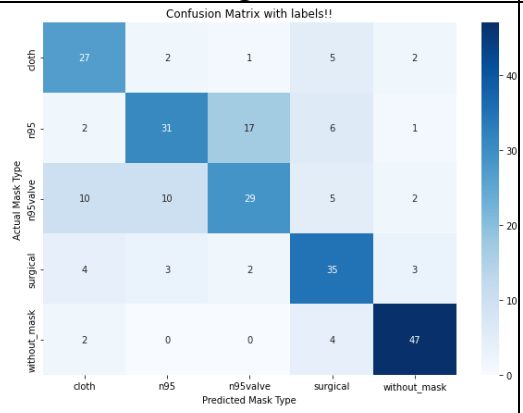	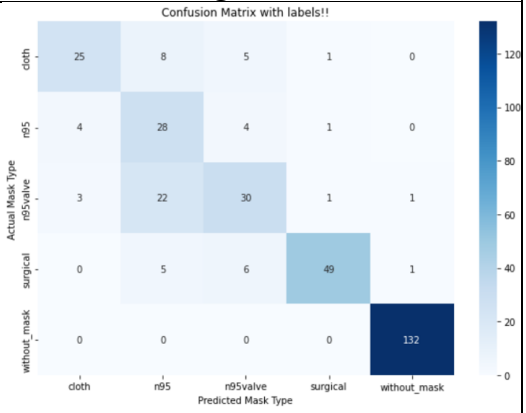
Fig 3.1.5 Confusion Matrix (Gender Bias (Before)) (Male)	Fig 3.1.6 Confusion Matrix (Gender Bias (After)) (Male)
Test Accuracy :- **67.60 %**	Test Accuracy :- **80.90 %**
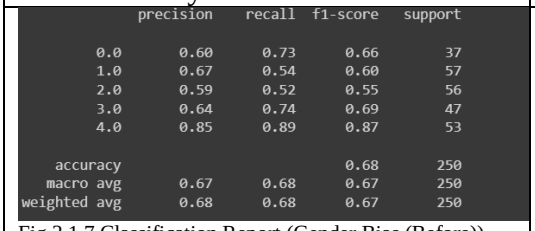	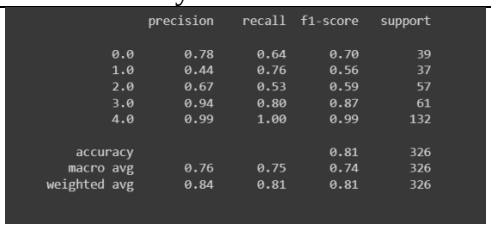
Fig 3.1.7 Classification Report (Gender Bias (Before)) (Male)	Fig 3.1.8 Classification Report (Gender Bias (Before)) (Male)

Table 3.1.2 Male Bias

3.2 Age Based Bias

a. Below 30

Before Eliminating Bias	After Eliminating Bias
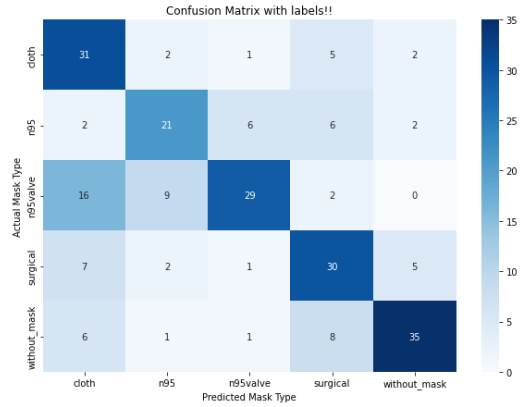	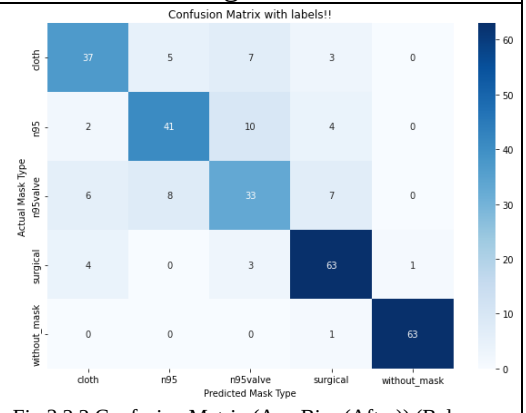
Fig 3.2.1 Confusion Matrix (Age Bias (Before)) (Below 30)	Fig 3.2.2 Confusion Matrix (Age Bias (After)) (Below 30)
Test Accuracy :- **63.47 %**	Test Accuracy :- **79.53 %**

	precision	recall	f1-score	support
0.0	0.50	0.76	0.60	41
1.0	0.60	0.57	0.58	37
2.0	0.76	0.52	0.62	56
3.0	0.59	0.67	0.62	45
4.0	0.80	0.69	0.74	51
accuracy			0.63	230
macro avg	0.65	0.64	0.63	230
weighted avg	0.66	0.63	0.64	230

Fig 3.2.3 Classification Report (Age Bias (Before)) (Below 30)

	precision	recall	f1-score	support
0.0	0.76	0.71	0.73	52
1.0	0.76	0.72	0.74	57
2.0	0.62	0.61	0.62	54
3.0	0.81	0.89	0.85	71
4.0	0.98	0.98	0.98	64
accuracy			0.80	298
macro avg	0.79	0.78	0.78	298
weighted avg	0.79	0.80	0.79	298

Fig 3.2.4 Classification Report (Age Bias (Before)) (Below 30)

Table 3.2.1 Below 30 (Age) group Bias

b. Above 30

Before Eliminating Bias	After Eliminating Bias

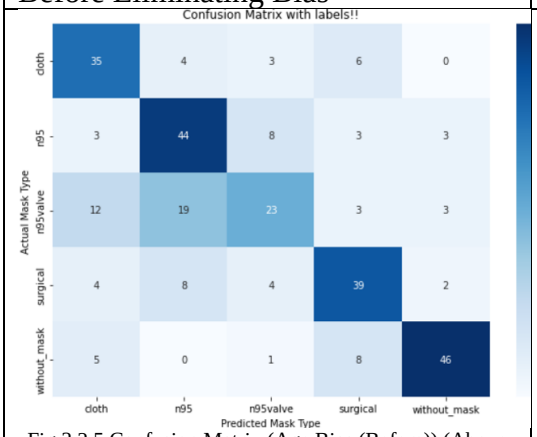

Fig 3.2.5 Confusion Matrix (Age Bias (Before)) (Above 30)

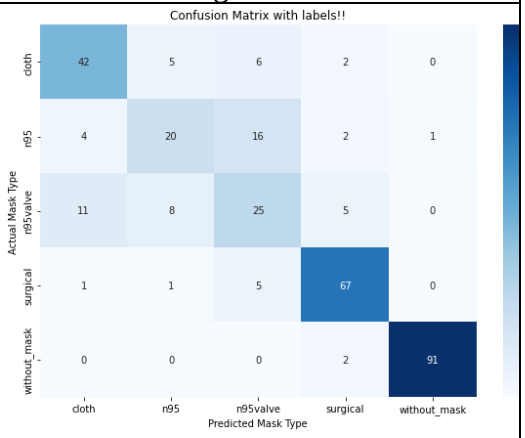

Fig 3.2.6 Confusion Matrix (Age Bias (Before)) (Above 30)

Test Accuracy :- **65.30 %**

	precision	recall	f1-score	support
0.0	0.59	0.73	0.65	48
1.0	0.59	0.72	0.65	61
2.0	0.59	0.38	0.46	60
3.0	0.66	0.68	0.67	57
4.0	0.85	0.77	0.81	60
accuracy			0.65	286
macro avg	0.66	0.66	0.65	286
weighted avg	0.66	0.65	0.65	286

Fig 3.2.7 Classification Report (Age Bias (Before)) (Above 30)

Test Accuracy :- **78.02 %**

	precision	recall	f1-score	support
0.0	0.72	0.76	0.74	55
1.0	0.59	0.47	0.52	43
2.0	0.48	0.51	0.50	49
3.0	0.86	0.91	0.88	74
4.0	0.99	0.98	0.98	93
accuracy			0.78	314
macro avg	0.73	0.72	0.72	314
weighted avg	0.78	0.78	0.78	314

Fig 3.2.8 Classification Report (Age Bias (Before)) (Above 30)

Table 3.2.2 Above 30 (Age) group Bias

So, in this manner the bias was detected for two attributes : Age, Gender in the existing model and they were eliminated respectively by analysing and pre-processing the old dataset by adding the proper images for the respective class which produced bias in the model and hence it can be seen that after performing the required steps, the bias is being removed and performance of the mode after removing the bias is increasing.

4. K-Fold Cross Validation

The most common technique for model evaluation and model selection in machine learning is K-fold cross-validation. In this technique, all the sample will have the chance of being tested. In this, K stands for the number of iterations over the dataset. In each iteration the dataset is splitted intro K parts, where 1 part is used for testing and rest K-1 parts are used for training purpose. The below image shows the process of 5-Fold Cross Validation.

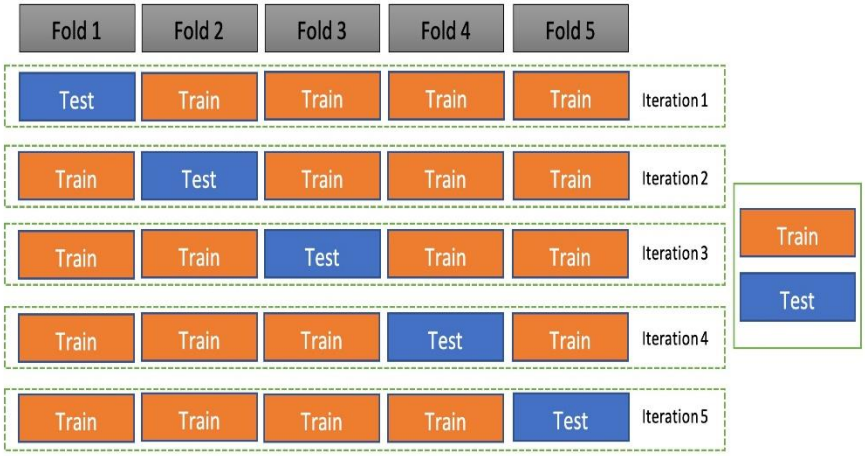

Fig 4.1 K-Fold dataset splitting[8]

In this phase-II of the project, we have performed 10-Fold cross validation in order to train the model. So, the dataset is divided into 10 parts, where the first part will be used for testing purpose and rest of them to train the model and the same will be repeated for rest of folds with different part being selected for testing and training.

The key reason behind using this technique is that it enables each part of the dataset to be trained as well as tested and it ultimately results in increase of accuracy such that the model trained in this manner will provide more accurate results over unknown samples. Whereas, in the phase-I of the project, the training and testing dataset is splitted using normal test/train split where every time the training is performed on the same training dataset which increases the chances of overfitting but with use of K-Fold the risk of overfitting can be significantly reduced.

Results of each individual fold, as well as the aggregate statistics, for precision, recall, F1, and accuracy:-

```
Fold 1
Epoch [1], train_loss: 1.5610, train_accuracy: 35.4590, valid_loss: 1.5006, valid_accuracy: 46.3054
Epoch [2], train_loss: 1.4234, train_accuracy: 54.3705, valid_loss: 1.3550, valid_accuracy: 50.2463
Epoch [3], train_loss: 1.2840, train_accuracy: 59.2084, valid_loss: 1.2636, valid_accuracy: 58.1281
Epoch [4], train_loss: 1.1547, train_accuracy: 64.9258, valid_loss: 1.1299, valid_accuracy: 62.5616
Epoch [5], train_loss: 1.0498, train_accuracy: 67.8395, valid_loss: 1.1026, valid_accuracy: 64.5320
Epoch [6], train_loss: 0.9674, train_accuracy: 69.7086, valid_loss: 1.0079, valid_accuracy: 66.5025
Epoch [7], train_loss: 0.9100, train_accuracy: 71.0830, valid_loss: 0.9238, valid_accuracy: 66.0099
Epoch [8], train_loss: 0.8538, train_accuracy: 73.5569, valid_loss: 0.8595, valid_accuracy: 67.4877
Epoch [9], train_loss: 0.8142, train_accuracy: 73.7768, valid_loss: 0.8861, valid_accuracy: 67.9803
Epoch [10], train_loss: 0.7749, train_accuracy: 75.0412, valid_loss: 0.8256, valid_accuracy: 68.4729

Test Accuracy of the model: 71.98879551820728 %
              precision    recall  f1-score   support

         0.0       0.58      0.64      0.61        59
         1.0       0.65      0.53      0.58        66
         2.0       0.34      0.25      0.29        48
         3.0       0.76      0.89      0.82        96
         4.0       0.97      0.99      0.98        88

    accuracy                           0.72       357
   macro avg       0.66      0.66      0.66       357
weighted avg       0.70      0.72      0.71       357
```

```
Fold 2
Epoch [1], train_loss: 0.7460, train_accuracy: 76.0308, valid_loss: 0.7277, valid_accuracy: 72.4138
Epoch [2], train_loss: 0.7197, train_accuracy: 76.6905, valid_loss: 0.8479, valid_accuracy: 74.8768
Epoch [3], train_loss: 0.6957, train_accuracy: 77.8999, valid_loss: 0.7321, valid_accuracy: 76.8473
Epoch [4], train_loss: 0.6662, train_accuracy: 79.4393, valid_loss: 0.6910, valid_accuracy: 74.8768
Epoch [5], train_loss: 0.6558, train_accuracy: 79.2743, valid_loss: 0.6777, valid_accuracy: 76.3547
Epoch [6], train_loss: 0.6219, train_accuracy: 81.2534, valid_loss: 0.6664, valid_accuracy: 75.3695
Epoch [7], train_loss: 0.6099, train_accuracy: 81.3634, valid_loss: 0.7422, valid_accuracy: 75.3695
Epoch [8], train_loss: 0.5910, train_accuracy: 82.1880, valid_loss: 0.6957, valid_accuracy: 76.8473
Epoch [9], train_loss: 0.5654, train_accuracy: 83.9472, valid_loss: 0.6621, valid_accuracy: 75.8621
Epoch [10], train_loss: 0.5447, train_accuracy: 83.9472, valid_loss: 0.6597, valid_accuracy: 73.8916

Test Accuracy of the model: 75.63025210084034 %
              precision    recall  f1-score   support

         0.0       0.67      0.59      0.63        59
         1.0       0.69      0.73      0.71        66
         2.0       0.42      0.33      0.37        48
         3.0       0.79      0.88      0.83        96
         4.0       0.97      0.99      0.98        88

    accuracy                           0.76       357
   macro avg       0.71      0.70      0.70       357
weighted avg       0.74      0.76      0.75       357
```

```
Fold 3
Epoch [1], train_loss: 0.5551, train_accuracy: 83.4615, valid_loss: 0.4524, valid_accuracy: 90.5941
Epoch [2], train_loss: 0.5259, train_accuracy: 85.1099, valid_loss: 0.4510, valid_accuracy: 89.1089
Epoch [3], train_loss: 0.5196, train_accuracy: 85.2198, valid_loss: 0.4773, valid_accuracy: 90.0990
Epoch [4], train_loss: 0.4961, train_accuracy: 86.7582, valid_loss: 0.4771, valid_accuracy: 90.5941
Epoch [5], train_loss: 0.4878, train_accuracy: 86.4835, valid_loss: 0.4293, valid_accuracy: 89.6040
Epoch [6], train_loss: 0.4746, train_accuracy: 87.9121, valid_loss: 0.4300, valid_accuracy: 90.0990
Epoch [7], train_loss: 0.4635, train_accuracy: 87.1978, valid_loss: 0.4608, valid_accuracy: 90.0990
Epoch [8], train_loss: 0.4432, train_accuracy: 88.4615, valid_loss: 0.4632, valid_accuracy: 89.1089
Epoch [9], train_loss: 0.4300, train_accuracy: 89.6154, valid_loss: 0.4315, valid_accuracy: 89.6040
Epoch [10], train_loss: 0.4175, train_accuracy: 90.1099, valid_loss: 0.4835, valid_accuracy: 90.0990

Test Accuracy of the model: 77.03081232492997 %
              precision    recall  f1-score   support

         0.0       0.69      0.68      0.68        59
         1.0       0.73      0.71      0.72        66
         2.0       0.45      0.35      0.40        48
         3.0       0.79      0.88      0.83        96
         4.0       0.97      0.99      0.98        88

    accuracy                           0.77       357
   macro avg       0.72      0.72      0.72       357
weighted avg       0.76      0.77      0.76       357
```

```
Fold 4
Epoch [1], train_loss: 0.4176, train_accuracy: 89.8901, valid_loss: 0.3527, valid_accuracy: 95.0495
Epoch [2], train_loss: 0.4068, train_accuracy: 90.3846, valid_loss: 0.3697, valid_accuracy: 93.0693
Epoch [3], train_loss: 0.3955, train_accuracy: 90.8791, valid_loss: 0.3190, valid_accuracy: 92.5743
Epoch [4], train_loss: 0.3786, train_accuracy: 91.4286, valid_loss: 0.3300, valid_accuracy: 92.5743
Epoch [5], train_loss: 0.3707, train_accuracy: 91.7033, valid_loss: 0.3309, valid_accuracy: 94.0594
Epoch [6], train_loss: 0.3620, train_accuracy: 91.9231, valid_loss: 0.3169, valid_accuracy: 92.5743
Epoch [7], train_loss: 0.3432, train_accuracy: 92.8571, valid_loss: 0.3281, valid_accuracy: 92.5743
Epoch [8], train_loss: 0.3378, train_accuracy: 93.0769, valid_loss: 0.3482, valid_accuracy: 92.5743
Epoch [9], train_loss: 0.3272, train_accuracy: 93.5714, valid_loss: 0.3033, valid_accuracy: 92.5743
Epoch [10], train_loss: 0.3172, train_accuracy: 93.6813, valid_loss: 0.3896, valid_accuracy: 92.0792

Test Accuracy of the model: 77.87114845938375 %
              precision    recall  f1-score   support

         0.0       0.77      0.61      0.68        59
         1.0       0.73      0.71      0.72        66
         2.0       0.42      0.44      0.43        48
         3.0       0.81      0.91      0.85        96
         4.0       0.99      0.99      0.99        88

    accuracy                           0.78       357
   macro avg       0.74      0.73      0.73       357
weighted avg       0.78      0.78      0.78       357
```

```
Fold 5
Epoch [1], train_loss: 0.3203, train_accuracy: 94.0659, valid_loss: 0.3016, valid_accuracy: 94.0594
Epoch [2], train_loss: 0.3102, train_accuracy: 94.3956, valid_loss: 0.2697, valid_accuracy: 94.0594
Epoch [3], train_loss: 0.2918, train_accuracy: 94.8901, valid_loss: 0.3816, valid_accuracy: 94.0594
Epoch [4], train_loss: 0.2872, train_accuracy: 94.8352, valid_loss: 0.2712, valid_accuracy: 93.5644
Epoch [5], train_loss: 0.2788, train_accuracy: 95.2198, valid_loss: 0.2528, valid_accuracy: 92.5743
Epoch [6], train_loss: 0.2744, train_accuracy: 95.3846, valid_loss: 0.2574, valid_accuracy: 93.0693
Epoch [7], train_loss: 0.2623, train_accuracy: 95.8791, valid_loss: 0.3132, valid_accuracy: 94.0594
Epoch [8], train_loss: 0.2553, train_accuracy: 96.1538, valid_loss: 0.2768, valid_accuracy: 93.5644
Epoch [9], train_loss: 0.2516, train_accuracy: 96.2088, valid_loss: 0.2469, valid_accuracy: 92.0792
Epoch [10], train_loss: 0.2451, train_accuracy: 96.4835, valid_loss: 0.2659, valid_accuracy: 91.5842

Test Accuracy of the model: 77.59103641456582 %
              precision    recall  f1-score   support

         0.0       0.76      0.69      0.73        59
         1.0       0.72      0.64      0.68        66
         2.0       0.39      0.44      0.41        48
         3.0       0.83      0.89      0.85        96
         4.0       1.00      1.00      1.00        88

    accuracy                           0.78       357
   macro avg       0.74      0.73      0.73       357
weighted avg       0.78      0.78      0.78       357
```

```
Fold 6
Epoch [1], train_loss: 0.2446, train_accuracy: 95.6593, valid_loss: 0.1830, valid_accuracy: 98.5149
Epoch [2], train_loss: 0.2352, train_accuracy: 96.4286, valid_loss: 0.1708, valid_accuracy: 98.0198
Epoch [3], train_loss: 0.2267, train_accuracy: 97.0879, valid_loss: 0.1663, valid_accuracy: 98.5149
Epoch [4], train_loss: 0.2199, train_accuracy: 96.3187, valid_loss: 0.1605, valid_accuracy: 98.5149
Epoch [5], train_loss: 0.2161, train_accuracy: 96.9231, valid_loss: 0.1807, valid_accuracy: 98.0198
Epoch [6], train_loss: 0.2118, train_accuracy: 96.9231, valid_loss: 0.1755, valid_accuracy: 98.5149
Epoch [7], train_loss: 0.2023, train_accuracy: 97.7473, valid_loss: 0.2043, valid_accuracy: 98.5149
Epoch [8], train_loss: 0.1983, train_accuracy: 97.3626, valid_loss: 0.1910, valid_accuracy: 98.5149
Epoch [9], train_loss: 0.1953, train_accuracy: 97.8022, valid_loss: 0.2159, valid_accuracy: 98.5149
Epoch [10], train_loss: 0.1871, train_accuracy: 97.9121, valid_loss: 0.1740, valid_accuracy: 98.0198

Test Accuracy of the model: 79.27170868347339 %
              precision    recall  f1-score   support

         0.0       0.77      0.69      0.73        59
         1.0       0.80      0.62      0.70        66
         2.0       0.43      0.58      0.50        48
         3.0       0.85      0.89      0.87        96
         4.0       1.00      1.00      1.00        88

    accuracy                           0.79       357
   macro avg       0.77      0.76      0.76       357
weighted avg       0.81      0.79      0.80       357
```

```
Fold 7
Epoch [1], train_loss: 0.1888, train_accuracy: 97.6923, valid_loss: 0.1272, valid_accuracy: 99.5050
Epoch [2], train_loss: 0.1843, train_accuracy: 98.0769, valid_loss: 0.1289, valid_accuracy: 99.5050
Epoch [3], train_loss: 0.1803, train_accuracy: 98.0769, valid_loss: 0.1146, valid_accuracy: 99.5050
Epoch [4], train_loss: 0.1739, train_accuracy: 98.3516, valid_loss: 0.1479, valid_accuracy: 99.5050
Epoch [5], train_loss: 0.1683, train_accuracy: 98.4066, valid_loss: 0.1201, valid_accuracy: 99.5050
Epoch [6], train_loss: 0.1641, train_accuracy: 98.3516, valid_loss: 0.1358, valid_accuracy: 99.5050
Epoch [7], train_loss: 0.1596, train_accuracy: 98.5714, valid_loss: 0.1214, valid_accuracy: 99.5050
Epoch [8], train_loss: 0.1550, train_accuracy: 99.1209, valid_loss: 0.1298, valid_accuracy: 99.5050
Epoch [9], train_loss: 0.1533, train_accuracy: 98.9011, valid_loss: 0.1354, valid_accuracy: 99.5050
Epoch [10], train_loss: 0.1476, train_accuracy: 98.7912, valid_loss: 0.1457, valid_accuracy: 99.5050

Test Accuracy of the model: 78.71148459383754 %
              precision    recall  f1-score   support

         0.0       0.77      0.69      0.73        59
         1.0       0.71      0.71      0.71        66
         2.0       0.41      0.42      0.41        48
         3.0       0.84      0.89      0.86        96
         4.0       1.00      1.00      1.00        88

    accuracy                           0.79       357
   macro avg       0.75      0.74      0.74       357
weighted avg       0.79      0.79      0.79       357
```

```
Fold 8
Epoch [1], train_loss: 0.1444, train_accuracy: 99.0110, valid_loss: 0.1248, valid_accuracy: 100.0000
Epoch [2], train_loss: 0.1405, train_accuracy: 98.9560, valid_loss: 0.1069, valid_accuracy: 99.5050
Epoch [3], train_loss: 0.1420, train_accuracy: 98.7363, valid_loss: 0.1011, valid_accuracy: 99.0099
Epoch [4], train_loss: 0.1352, train_accuracy: 99.1758, valid_loss: 0.1137, valid_accuracy: 99.5050
Epoch [5], train_loss: 0.1303, train_accuracy: 98.9011, valid_loss: 0.1066, valid_accuracy: 99.0099
Epoch [6], train_loss: 0.1271, train_accuracy: 99.2308, valid_loss: 0.1124, valid_accuracy: 99.5050
Epoch [7], train_loss: 0.1248, train_accuracy: 99.1758, valid_loss: 0.1160, valid_accuracy: 99.5050
Epoch [8], train_loss: 0.1186, train_accuracy: 99.1758, valid_loss: 0.1184, valid_accuracy: 99.0099
Epoch [9], train_loss: 0.1181, train_accuracy: 99.3407, valid_loss: 0.1542, valid_accuracy: 99.0099
Epoch [10], train_loss: 0.1178, train_accuracy: 99.4505, valid_loss: 0.1098, valid_accuracy: 99.0099

Test Accuracy of the model: 78.71148459383754 %
              precision    recall  f1-score   support

         0.0       0.78      0.73      0.75        59
         1.0       0.69      0.68      0.69        66
         2.0       0.43      0.42      0.42        48
         3.0       0.83      0.89      0.86        96
         4.0       1.00      1.00      1.00        88

    accuracy                           0.79       357
   macro avg       0.75      0.74      0.74       357
weighted avg       0.79      0.79      0.79       357
```

```
Fold 9
Epoch [1], train_loss: 0.1134, train_accuracy: 99.2308, valid_loss: 0.0981, valid_accuracy: 99.0099
Epoch [2], train_loss: 0.1099, train_accuracy: 99.3956, valid_loss: 0.1012, valid_accuracy: 99.0099
Epoch [3], train_loss: 0.1083, train_accuracy: 99.3407, valid_loss: 0.1235, valid_accuracy: 99.0099
Epoch [4], train_loss: 0.1058, train_accuracy: 99.3407, valid_loss: 0.0982, valid_accuracy: 99.0099
Epoch [5], train_loss: 0.1011, train_accuracy: 99.6703, valid_loss: 0.1034, valid_accuracy: 99.0099
Epoch [6], train_loss: 0.1017, train_accuracy: 99.5604, valid_loss: 0.1045, valid_accuracy: 99.0099
Epoch [7], train_loss: 0.0983, train_accuracy: 99.5604, valid_loss: 0.1043, valid_accuracy: 99.0099
Epoch [8], train_loss: 0.0961, train_accuracy: 99.4505, valid_loss: 0.0985, valid_accuracy: 99.0099
Epoch [9], train_loss: 0.0939, train_accuracy: 99.5604, valid_loss: 0.1124, valid_accuracy: 99.0099
Epoch [10], train_loss: 0.0933, train_accuracy: 99.5055, valid_loss: 0.0953, valid_accuracy: 99.0099

Test Accuracy of the model: 79.27170868347339 %
              precision    recall  f1-score   support

         0.0       0.78      0.71      0.74        59
         1.0       0.71      0.73      0.72        66
         2.0       0.43      0.42      0.42        48
         3.0       0.85      0.89      0.87        96
         4.0       1.00      1.00      1.00        88

    accuracy                           0.79       357
   macro avg       0.75      0.75      0.75       357
weighted avg       0.79      0.79      0.79       357
```

```
Fold 10
Epoch [1], train_loss: 0.0928, train_accuracy: 99.3407, valid_loss: 0.0893, valid_accuracy: 100.0000
Epoch [2], train_loss: 0.0909, train_accuracy: 99.6703, valid_loss: 0.1195, valid_accuracy: 100.0000
Epoch [3], train_loss: 0.0879, train_accuracy: 99.6154, valid_loss: 0.0812, valid_accuracy: 100.0000
Epoch [4], train_loss: 0.0866, train_accuracy: 99.5604, valid_loss: 0.0829, valid_accuracy: 100.0000
Epoch [5], train_loss: 0.0834, train_accuracy: 99.7253, valid_loss: 0.0849, valid_accuracy: 100.0000
Epoch [6], train_loss: 0.0853, train_accuracy: 99.6154, valid_loss: 0.0767, valid_accuracy: 100.0000
Epoch [7], train_loss: 0.0818, train_accuracy: 99.7253, valid_loss: 0.0906, valid_accuracy: 99.5050
Epoch [8], train_loss: 0.0787, train_accuracy: 99.7253, valid_loss: 0.0777, valid_accuracy: 99.5050
Epoch [9], train_loss: 0.0760, train_accuracy: 99.7253, valid_loss: 0.0748, valid_accuracy: 99.5050
Epoch [10], train_loss: 0.0756, train_accuracy: 99.6154, valid_loss: 0.0802, valid_accuracy: 99.5050

Test Accuracy of the model: 80.11204481792717 %
              precision    recall  f1-score   support

         0.0       0.80      0.73      0.76        59
         1.0       0.75      0.68      0.71        66
         2.0       0.44      0.52      0.48        48
         3.0       0.87      0.89      0.88        96
         4.0       1.00      1.00      1.00        88

    accuracy                           0.80       357
   macro avg       0.77      0.76      0.77       357
weighted avg       0.81      0.80      0.80       357
```

Now it can be observed that the training and testing accuracy increases gradually with increasing number of folds , initially after normal test/train split of phase-I the testing accuracy was "**62.80 %**", whereas after applying the k-fold technique in phase-II , the testing accuracy at the end of fold-10 increased to "**80.11 %**".

5. Evaluation

Phase - I

To train the model using constructed dataset we have trained the model with 50 epochs and then evaluated the performance of the model using metrics like Recall, Precision, Accuracy and F1-Score for the testing phase.

The confusion matrix to describe the performance of the model is as below :-

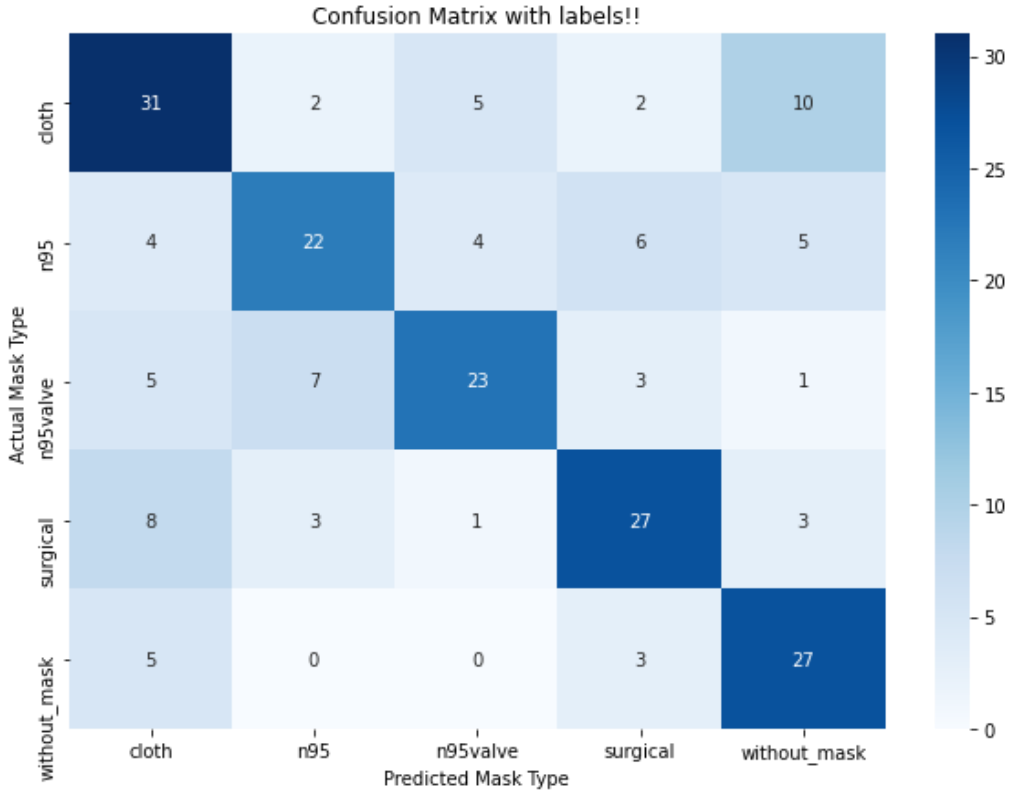

Fig 5.1 Confusion Matrix

As it can be seen the confusion matrix shows some of the images are not classified according to their respective classes, there are several reasons for that like data in some of the classes are imbalanced, model is not efficient enough to accurately classify images of various classes.

The accuracy of our model is **61.85%** for the validation phase and **62.8%** for the testing phase.

	precision	recall	f1-score	support
0.0	0.58	0.62	0.60	50
1.0	0.65	0.54	0.59	41
2.0	0.70	0.59	0.64	39
3.0	0.66	0.64	0.65	42
4.0	0.59	0.77	0.67	35
accuracy			0.63	207
macro avg	0.63	0.63	0.63	207
weighted avg	0.63	0.63	0.63	207

Fig 5.2 Classification Report

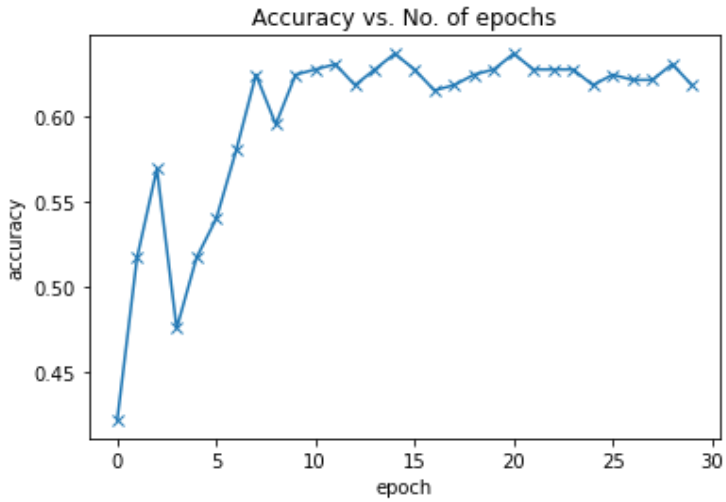

Fig 5.3 Accuracy vs No. of epochs

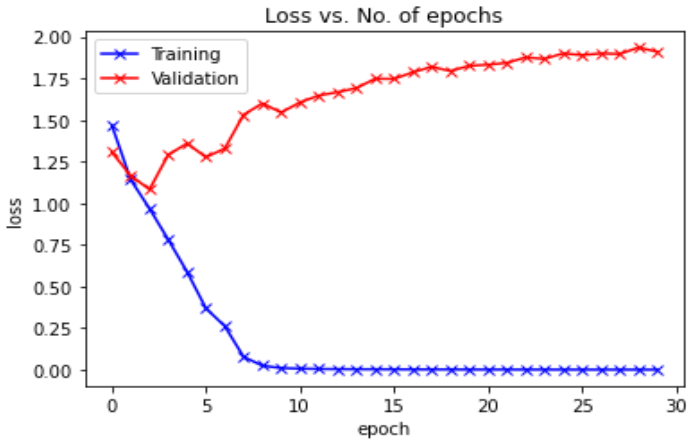

Fig 5.4 Loss vs No. of epochs

From the above two images, the first one shows the plot for Accuracy vs. number of epochs where it is observed that by training the model with a greater number of epochs, it increases the accuracy, and the second image shows the plot for loss vs. number of epochs and as the number of epochs are increasing the loss (training) decreases.

Phase – II

In the second phase of the project, the main aim was to analyse and remove any possible bias present during project phase-I , we accomplished it by following certain measures and the second part was to utilize the K-Fold Cross Validation technique in order to improve the performance of the model. We have used 10-Fold Cross Validation in this project.

```
Fold 10
Epoch [1], train_loss: 0.0928, train_accuracy: 99.3407, valid_loss: 0.0893, valid_accuracy: 100.0000
Epoch [2], train_loss: 0.0909, train_accuracy: 99.6703, valid_loss: 0.1195, valid_accuracy: 100.0000
Epoch [3], train_loss: 0.0879, train_accuracy: 99.6154, valid_loss: 0.0812, valid_accuracy: 100.0000
Epoch [4], train_loss: 0.0866, train_accuracy: 99.5604, valid_loss: 0.0829, valid_accuracy: 100.0000
Epoch [5], train_loss: 0.0834, train_accuracy: 99.7253, valid_loss: 0.0849, valid_accuracy: 100.0000
Epoch [6], train_loss: 0.0853, train_accuracy: 99.6154, valid_loss: 0.0767, valid_accuracy: 100.0000
Epoch [7], train_loss: 0.0818, train_accuracy: 99.7253, valid_loss: 0.0906, valid_accuracy: 99.5050
Epoch [8], train_loss: 0.0787, train_accuracy: 99.7253, valid_loss: 0.0777, valid_accuracy: 99.5050
Epoch [9], train_loss: 0.0760, train_accuracy: 99.7253, valid_loss: 0.0748, valid_accuracy: 99.5050
Epoch [10], train_loss: 0.0756, train_accuracy: 99.6154, valid_loss: 0.0802, valid_accuracy: 99.5050

Test Accuracy of the model: 80.11204481792717 %
              precision    recall  f1-score   support

         0.0       0.80      0.73      0.76        59
         1.0       0.75      0.68      0.71        66
         2.0       0.44      0.52      0.48        48
         3.0       0.87      0.89      0.88        96
         4.0       1.00      1.00      1.00        88

    accuracy                           0.80       357
   macro avg       0.77      0.76      0.77       357
weighted avg       0.81      0.80      0.80       357
```

Fig 5.5 Classification report after 10th fold

As it can be seen from the classification report above that after using 10-Fold Cross the testing accuracy is increased to "**80.11 %**" , while it was "**62.80 %**" in the project phase-I. So, there has been improvement in the performance of the model by the usage of K-fold technique.

The confusion matrix and the classification report below show better numbers in comparison to the previous phase as the dataset for all the classes have been equally balanced and using K-Fold the model is trained in a better manner, hence increasing the accuracy of the model.

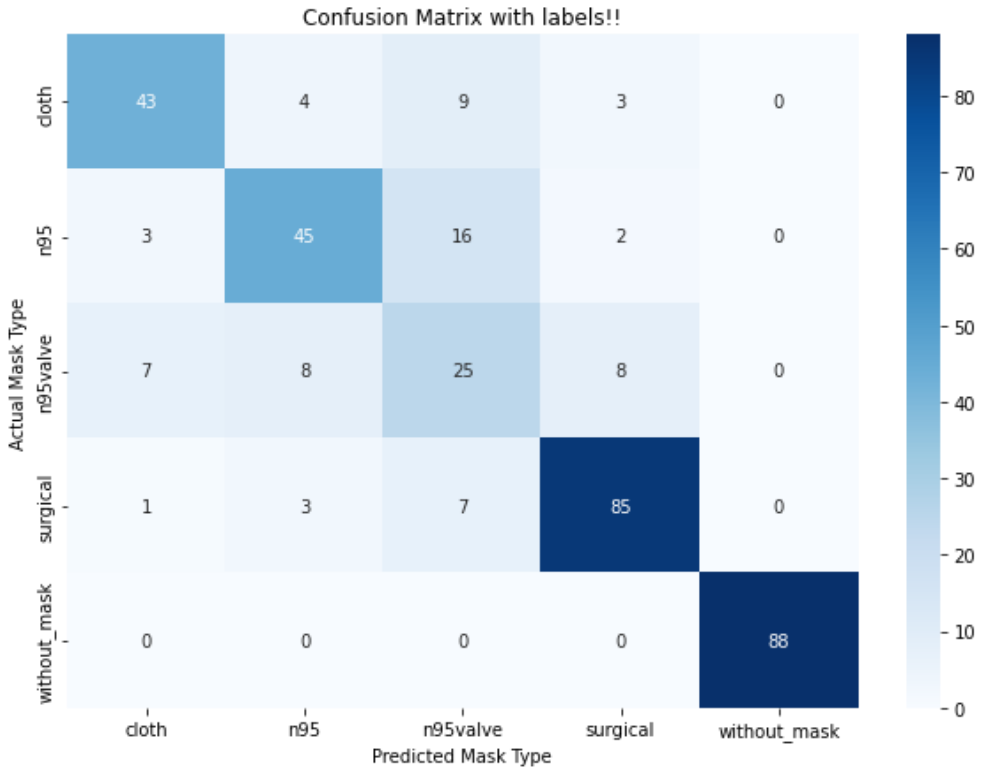

Fig 5.6 Confusion Matrix

	precision	recall	f1-score	support
0.0	0.80	0.73	0.76	59
1.0	0.75	0.68	0.71	66
2.0	0.44	0.52	0.48	48
3.0	0.87	0.89	0.88	96
4.0	1.00	1.00	1.00	88
accuracy			0.80	357
macro avg	0.77	0.76	0.77	357
weighted avg	0.81	0.80	0.80	357

Fig 5.7 Classification Report

6. Reference

1.https://www.kaggle.com/andrewmvd/face-mask-detection

2.https://www.kaggle.com/prasoonkottarathil/face-mask-lite-dataset

3.https://www.kaggle.com/datasets/ashwingupta3012/male-and-female-faces-dataset

4.https://humansintheloop.org/mask-dataset-download/?submissionGuid=90621ead-d768-4016-b823-179ee4908f34

5.https://www.kaggle.com/datasets/anglinab/facemasks

6.https://towardsdatascience.com/understand-the-architecture-of-cnn-90a25e244c7

7.https://medium.com/thecyphy/train-cnn-model-with-pytorch-21dafb918f48

8.https://en.wikipedia.org/wiki/Stochastic_gradient_descent#cite_note-23

9.https://sqlrelease.com/introduction-to-k-fold-cross-validation-in-python

10.https://medium.com/sfu-cspmp/model-transparency-fairness-552a747b444