COMP 6721 Project

AI Face Mask Detector - CNN

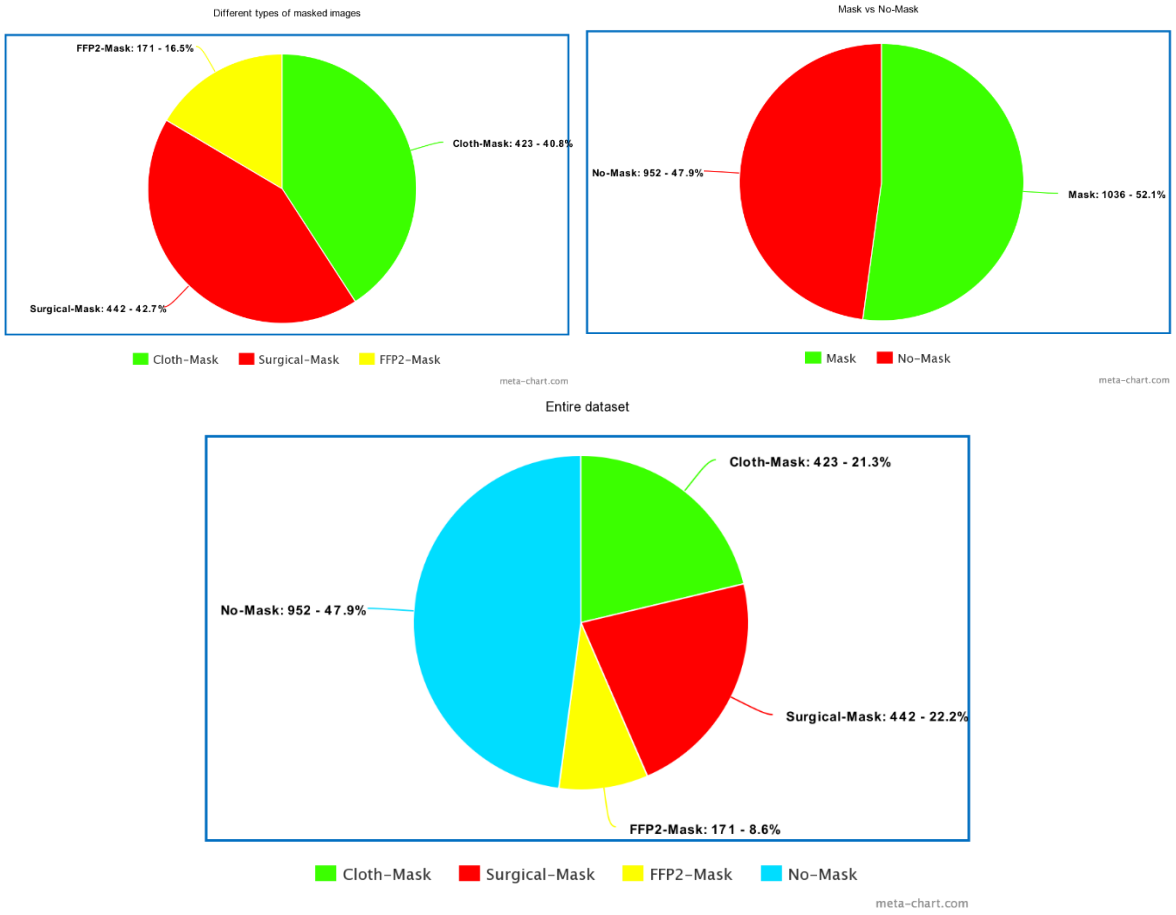
TEAM NAME: **NS_10**

Name	ID	Work
Smit Desai	40120178	Data collection & preprocessing. Model building and training
Aayush Khandelwal	40156633	Model building and training

1. PHASE 1

1.1 COLLECTING DATA

As outlined in the project description, the model had to classify images into 4 different classes: Cloth-Mask, Surgical-Mask, FFP2-Mask, and No-Mask. The dataset has a total of 1988 images. Here are some statistics about the dataset:



Source:

1) Cloth-Mask, Surgical-Mask images were collected from Google images.
2) FFP2-Mask images were collected from Shutterstock.
3) No-Mask images were collected from 2 different Kaggle datasets
    a. https://www.kaggle.com/vinaykudari/facemask by Vinay Kudari
    b. https://www.kaggle.com/spandanpatnaik09/face-mask-detectormask-not-mask-incorrect-mask by Spandanpatnaik

(Please see the attached reference files to view every image source)

1.2 PREPROCESSING

The following preprocessing steps were taken:

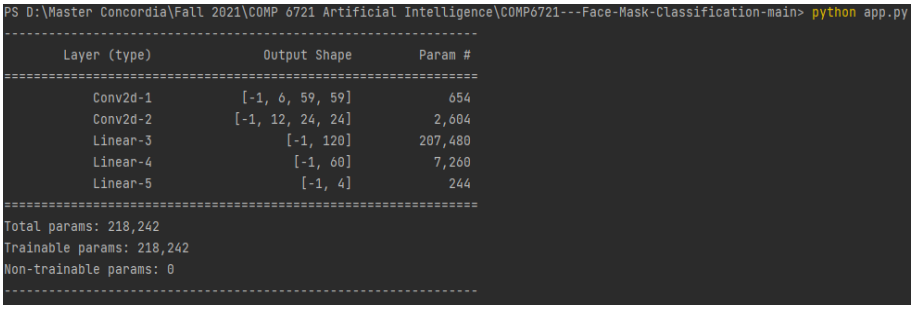
1) **Resize:** In order to feed the images to the CNN network, they were resized (64 px, 64 px)
2) **Horizontal-flips:** To increase randomness, images were flipped horizontally with a probability of 0.5
3) **Conversation to tensor:** Converts the images into an array of numbers, called torch tensor. Each pixel of the input RGB image is divided into three different pixels- red, blue, and green. This creates three different images. For each generated image, the pixel value is divided by 255 to range the pixel range from [0 255] to [0 1]
4) **Train-Test split:** The data set is split into 2 categories during runtime. The training dataset has 1690 images (85%) whereas the test dataset has 298 images (15%)
5) **Batch-loading:** Finally, images from both categories are loaded into a batch of size 32 and are randomly shuffled.

1.3. CNN ARCHITECTURE

Taking account of CNN architecture we have created a Convolution Neural Network which primarily consists of 3 non-linear convolution layers and 2 linear fully connected layers. The architecture initiates with producing a feature map when the input tensor of shape [32, 3,64,64] is passed through this layer. The stack normalization is then applied to all the feature maps. This helps reduce a wider weight range by normalizing with the standard deviation.

Here, PyTorch describes the beta and gamma hyperparameters of batch normalization. For this architecture, we have chosen Relu as our activation function because it sets the negative value to 0. After that pass the output tensor through the dropout layer which will deactivate randomly selected activations according to probability. This gives the model considerable robustness and reduces the possibility of overfitting. At Last, performing MaxPooling with a kernel size of 2X2 to reduce network complexity. This model used a 3x3 size kernel to extract features. To further optimize the memory, we implemented the Relu feature directly on the output sensor without allocating additional memory. For reshaping the formula we have used is (width_of_input –kernel_size + 1)/stride so by placing the value in the above formula gives us the output 32 after applying operation.

Looping the above-mentioned process for the next 2 convolution layers produces an output tensor of shape which then passes it to the Fully connected linear layer, followed by flatting the output tensor first, producing the linear output tensor. The output is shown in Figure 3 given below:



Here, all layers are classes in PyTorch's nn module, and model classes are inherited from nn. Module classes that handle the entire complexity of initializing model parameters (weighting), manipulating them, and storing them in memory.

1.4. CNN Model Training

Here, we have selected five model training epochs and used batch input processing to overcome the customization of large amounts of data in memory. (batch size for input during training is 32).

CNN model training has two phases: forward pass and backward pass. Repeated at each epoch, the input batch goes through a series of layers defined by the forward method of the CNN model class. After performing this forward pass, the loss value is calculated for the specified prediction (at this point model.train() allows gradient manipulation so you can update the model weights). The loss value or gradient is typically the following error calculation: Each output node and inner layer node. This is done using the CrossEntropyLoss function of the Nn module. This function finally uses the SoftMax activation function to narrow the output probability between 0 and 1 and calculate the loss value. Next, loss.backward () performs the backpropagation phase. In this phase, the difference in total weighting is calculated and optimized and optimized. Step () updates the model weights. Here, Adam is used as the optimizer and a learning rate of 0.01 is selected for backward execution. Here, the output prediction gets the probabilities of all four output classes. Among them, the class with the highest probability value was selected as the class predicted by the model.

1.5 EVALUATION

The initial step in evaluating our model involved fine tuning the hyperparameters of the CNN model i.e.the number of epochs in our case. This was determined by applying our generated model to the validation dataset and examining the output. The validation dataset consists of 1988 images split evenly among the four classes namely “Cloth-Mask”, “Surgical-Mask” , “FFP2-Mask” , and “No-Mask”. The output was observed and it indicated that after 5 epochs the improvement in accuracy and loss was unnoticeable

Results of each epoch are as follows.

Epoch 1: Correct predictions: 7
Accuracy: 0.46094674556213017
Loss: 8.298946174855768

Epoch 2: Correct predictions: 801/1690
Accuracy: 0.47396449704142013
Loss: 0.038740938302327894

Epoch 3: Correct predictions: 801/1690
Accuracy: 0.47396449704142013
Loss: 0.038791874174535626

Epoch 4: Correct predictions: 801/1690
Accuracy: 0.47396449704142013
Loss: 0.03877918755514382
Epoch 5: Correct predictions: 801/1690
Accuracy: 0.47396449704142013
Loss: 0.03880976057616917

When model was applied for testing, it predicted 151 images as correct class out of 298. **Accuracy comes out to be 0.5067114093959731 and loss was 0.04058477662553723.**

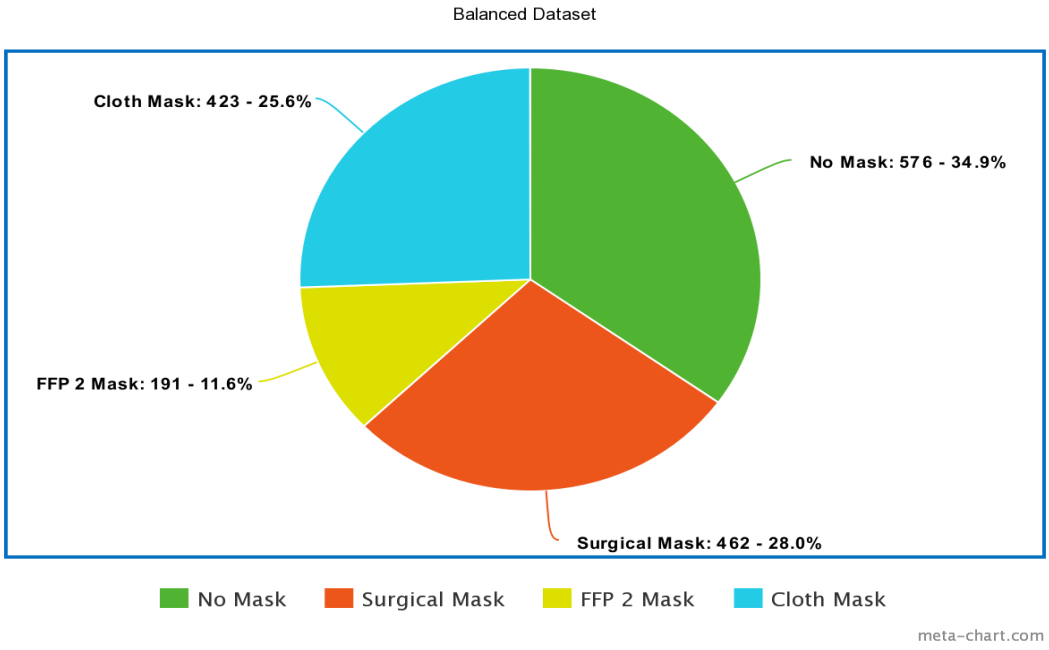
2. PHASE II

2.1 Improvement of Phase I

We have improved the model from Phase 1 taking account of CNN architecture we have updated our Convolution Neural Network by adding more non-linear convolution layers as well as linear fully connected layers. The accuracy and the evaluation matrix which consists of a precision, recall, and F1-score shows the improper result when trained with the previous model used in phase 1 so now the updated model shows promising results i.e. evaluation matrix. In the model, we updated the following things mentioned below:

1. Decreased the learning rate from 0.07 to 0.000001 and
2. Balanced the dataset as earlier No-mask images constituted 50% of the dataset.
3. Updated our Convolution Neural Network by adding more non-linear convolution layers as well as linear fully connected layers. See Figure 1 for CNN Model updated

So now after training with a bigger model and decreased learning rate on the balanced dataset provides a proper output. The new balanced dataset is shown below:



```
(virtual) D:\Workspace\COMP6721---Face-Mask-Classification>python app.py
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 128, 128]             896
       BatchNorm2d-2         [-1, 32, 128, 128]              64
         LeakyReLU-3         [-1, 32, 128, 128]               0
            Conv2d-4         [-1, 32, 128, 128]           9,248
       BatchNorm2d-5         [-1, 32, 128, 128]              64
         LeakyReLU-6         [-1, 32, 128, 128]               0
         MaxPool2d-7           [-1, 32, 64, 64]               0
            Conv2d-8           [-1, 64, 64, 64]          18,496
       BatchNorm2d-9           [-1, 64, 64, 64]             128
        LeakyReLU-10           [-1, 64, 64, 64]               0
           Conv2d-11           [-1, 64, 64, 64]          36,928
      BatchNorm2d-12           [-1, 64, 64, 64]             128
        LeakyReLU-13           [-1, 64, 64, 64]               0
        MaxPool2d-14           [-1, 64, 32, 32]               0
          Dropout-15                 [-1, 65536]               0
           Linear-16                  [-1, 1000]      65,537,000
             ReLU-17                  [-1, 1000]               0
           Linear-18                   [-1, 512]         512,512
             ReLU-19                   [-1, 512]               0
          Dropout-20                   [-1, 512]               0
           Linear-21                     [-1, 4]           2,052
             ReLU-22                     [-1, 4]               0
================================================================
Total params: 66,117,516
Trainable params: 66,117,516
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 38.03
Params size (MB): 252.22
Estimated Total Size (MB): 290.43
----------------------------------------------------------------
```

Figure 1 CNN Model Updated

2.2 BIAS DETECTION AND ELIMINATION

Our goal was to analyze our model for any two of the following biases: age, gender, or race. The bias for which we analyzed our model was “gender bias” and “age bias”. Before analyzing the bias regarding the two measures we updated the model and make the dataset balanced. The process which we followed to check for any possible gender bias was as follows. The first step involved separating our testing data into separate “male” and “female” sets. Initially, the dataset consisted of 1680 images wherein 423 images belonged to the “Cloth Mask” class, 576 images belonged to the “No Mask” class, 462 images belonged to the “Surgical Mask” class and the remaining 191 images were instances of “FFP2 Mask” class. So for the purpose of gender bias detection, we separated the test data set into a male dataset which again divided into 4 classes present on the both male and female set. For the next step, we evaluated the performance of our model for the male dataset first, and similarly, we checked our model’s performance against the female dataset. For age bias, we follow the same strategy and divided our training and test set into 3 categories which were 0-18, 18-55, and 55-above, and classify the total 460 test images into these 3 different folders/categories.

Evaluation Metrics for the gender subclass –MALE and Female:

--PRE-BIAS

Precision, Recall, F1- score Accuracy:

MALE

	precision	recall	f1-score	support
Cloth-Mask	0.94	0.76	0.84	59
FFP2-Mask	1.00	1.00	1.00	10
No-Mask	1.00	0.97	0.98	96

```
Surgical-Mask       0.69      0.95      0.80        37
     accuracy                            0.91       202
    macro avg       0.91      0.92      0.91       202
 weighted avg       0.92      0.91      0.91       202
Confusion matrix, without normalization
[[45  0  0 14]
 [ 0 10  0  0]
 [ 1  0 93  2]
 [ 2  0  0 35]]
```

FEMALE

```
               precision    recall  f1-score   support
   Cloth-Mask       1.00      0.92      0.96        25
    FFP2-Mask       1.00      1.00      1.00        24
      No-Mask       1.00      1.00      1.00        39
Surgical-Mask       0.96      1.00      0.98        51
     accuracy                           0.99       139
    macro avg       0.99      0.98      0.98       139
 weighted avg       0.99      0.99      0.99       139
Confusion matrix, without normalization
[[23  0  0  2]
 [ 0 24  0  0]
 [ 0  0 39  0]
 [ 0  0  0 51]]
---------------------------
```

From the above observation, we found that the “Male cloth mask” class and “Male Surgical mask” classes were performing poorly/inefficient as compared to the all-female counterparts category.

--POSTBIAS Result:

Precision, Recall, F1- score Accuracy:

Male

```
Testing:
Correct prediction: 225/249 and accuracy: 0.9036144578313253 and loss: 0.009048174483230314
```

```
               precision    recall  f1-score   support

   Cloth-Mask       0.98      0.82      0.90        79
    FFP2-Mask       0.77      1.00      0.87        10
      No-Mask       1.00      0.94      0.97        96
Surgical-Mask       0.80      1.00      0.89        64

     accuracy                           0.92       249
    macro avg       0.89      0.94      0.91       249
 weighted avg       0.93      0.92      0.92       249
```

```
Confusion matrix, without normalization
[[65  2  0 12]
 [ 0 10  0  0]
 [ 1  1 90  4]
 [ 0  0  0 64]]
---------------------------
```

FEMALE

```
Testing:
Correct prediction: 184/187 and accuracy: 0.983957219251337 and loss: 0.0026716614729739765
```

	precision	recall	f1-score	support
Cloth-Mask	0.98	0.96	0.97	50
FFP2-Mask	1.00	1.00	1.00	36
No-Mask	1.00	0.95	0.97	39
Surgical-Mask	0.95	1.00	0.98	62
accuracy			0.98	187
macro avg	0.98	0.98	0.98	187
weighted avg	0.98	0.98	0.98	187

```
Confusion matrix, without normalization
[[48  0  0  2]
 [ 0 36  0  0]
 [ 1  0 37  1]
 [ 0  0  0 62]]
---------------------------
```

Evaluation Metrics for the age subclass –teen,men,old:

--PRE-BIAS

Precision, Recall, F1- score Accuracy:

TEEN

	precision	recall	f1-score	support
Cloth-Mask	1.00	0.80	0.89	5
FFP2-Mask	1.00	1.00	1.00	2
No-Mask	1.00	1.00	1.00	21
Surgical-Mask	0.86	1.00	0.92	6
accuracy			0.97	34
macro avg	0.96	0.95	0.95	34
weighted avg	0.97	0.97	0.97	34

```
Confusion matrix, without normalization
[[ 4  0  0  1]
 [ 0  2  0  0]
 [ 0  0 21  0]
 [ 0  0  0  6]]
---------------------------
```

MEN

	precision	recall	f1-score	support

```
          Cloth-Mask       0.98      0.86      0.92        72
           FFP2-Mask       1.00      1.00      1.00        26
             No-Mask       1.00      0.98      0.99        93
       Surgical-Mask       0.87      1.00      0.93        73
            accuracy                           0.95       264
           macro avg       0.96      0.96      0.96       264
        weighted avg       0.96      0.95      0.95       264
       Confusion matrix, without normalization
       [[62  0  0 10]
        [ 0 26  0  0]
        [ 1  0 91  1]
        [ 0  0  0 73]]
       --------------------------
       OLD
                      precision    recall  f1-score   support
          Cloth-Mask       1.00      0.14      0.25         7
           FFP2-Mask       1.00      1.00      1.00         6
             No-Mask       1.00      1.00      1.00        21
       Surgical-Mask       0.60      1.00      0.75         9
            accuracy                           0.86        43
           macro avg       0.90      0.79      0.75        43
        weighted avg       0.92      0.86      0.83        43
       Confusion matrix, without normalization
       [[ 1  0  0  6]
        [ 0  6  0  0]
        [ 0  0 21  0]
        [ 0  0  0  9]]
       --------------------------
```

From the above observation, we found that the “old cloth mask” class and “old surgical mask” class were performing poorly/inefficiently from all the other counterparts.

--POSTBIAS:

TEEN

```
Testing:
Correct prediction: 56/60 and accuracy: 0.9333333333333333 and loss: 0.006236163030068079

              precision    recall  f1-score   support

   Cloth-Mask       1.00      0.87      0.93        30
    FFP2-Mask       1.00      1.00      1.00         3
      No-Mask       1.00      0.95      0.98        21
Surgical-Mask       0.55      1.00      0.71         6

     accuracy                           0.92        60
    macro avg       0.89      0.95      0.90        60
```

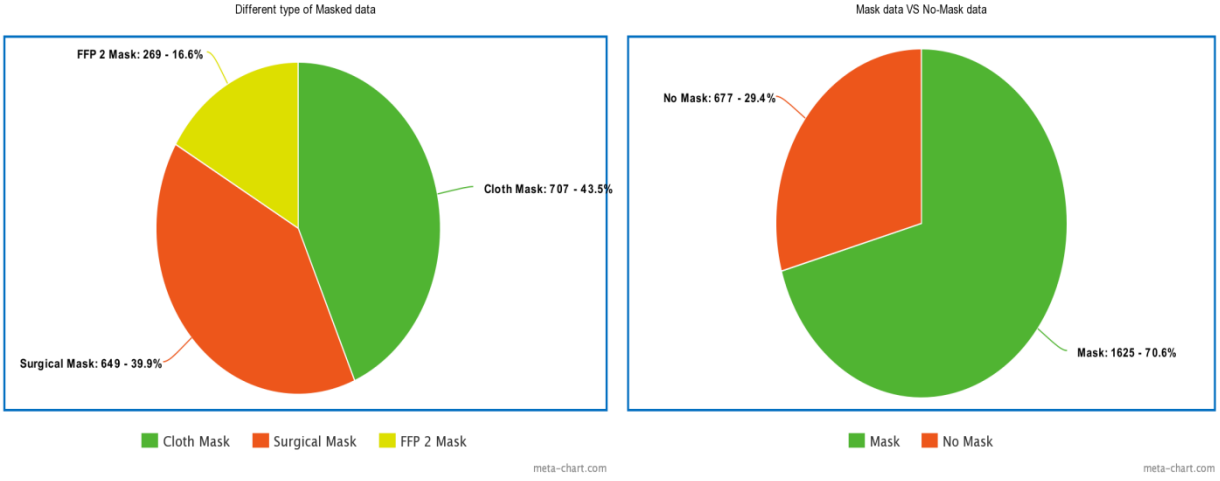
```
 weighted avg       0.95      0.92      0.93        60

Confusion matrix, without normalization
[[26  0  0  4]
 [ 0  3  0  0]
 [ 0  0 20  1]
 [ 0  0  0  6]]
---------------------------
MEN

Testing:
Correct prediction: 307/319 and accuracy: 0.9623824451410659 and loss:
0.0033590598469804447

               precision    recall  f1-score   support

   Cloth-Mask       0.99      0.95      0.97        93
    FFP2-Mask       0.97      1.00      0.99        37
      No-Mask       1.00      0.95      0.97        93
Surgical-Mask       0.92      1.00      0.96        96

     accuracy                           0.97       319
    macro avg       0.97      0.97      0.97       319
 weighted avg       0.97      0.97      0.97       319

Confusion matrix, without normalization
[[88  0  0  5]
 [ 0 37  0  0]
 [ 1  1 88  3]
 [ 0  0  0 96]]
---------------------------
OLD

Testing:
Correct prediction: 60/70 and accuracy: 0.8571428571428571 and loss: 0.013833112588950565

               precision    recall  f1-score   support

   Cloth-Mask       1.00      0.43      0.60        14
    FFP2-Mask       0.86      1.00      0.92         6
      No-Mask       1.00      0.90      0.95        21
Surgical-Mask       0.76      1.00      0.87        29

     accuracy                           0.86        70
    macro avg       0.91      0.83      0.83        70
 weighted avg       0.89      0.86      0.84        70
```

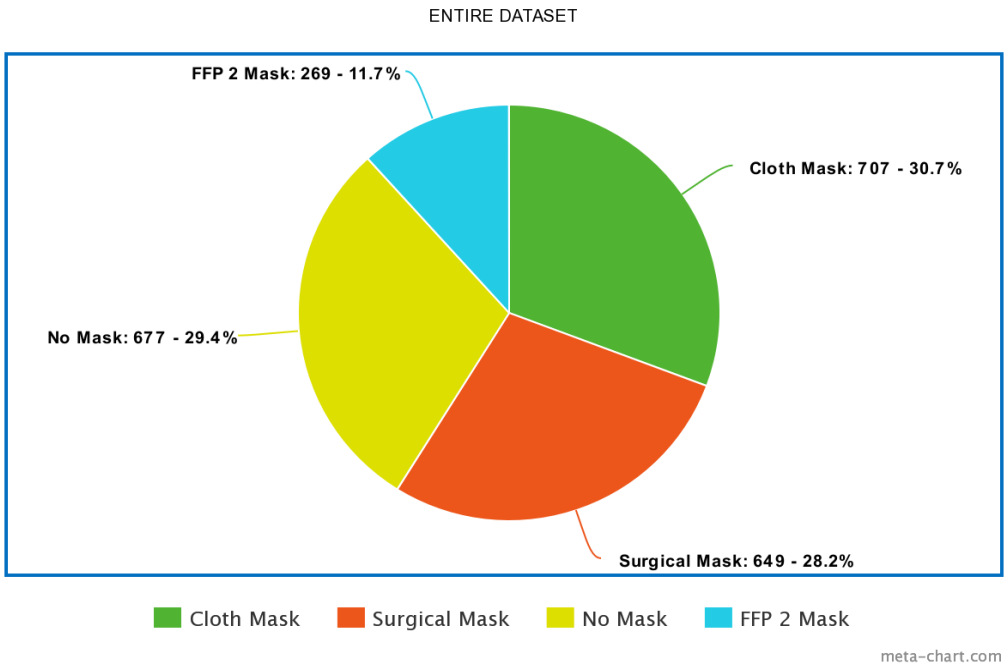
```
Confusion matrix, without normalization
[[ 6  1  0  7]
 [ 0  6  0  0]
 [ 0  0 19  2]
 [ 0  0  0 29]]
---------------------------
```

We observe that previously we have less number of testing data due to which there was some fluctuation in the evaluation matrix i.e. accuracy, precision, recall, and f1-score. The testing data was less in size and poor in quality due to which accuracy was less promising but after adding the images and balancing the dataset among 4 different classes we eliminate bias regarding cloth mask but we still found bias in the “old men surgical mask” class and “teen surgical mask” class.

2.3 INCREASED DATASET FOR BIAS ELIMINATION

As per the project Phase I feedback we have increased the size of the dataset. So now the entire dataset has a total of 2302 images which is further classified into 4 major classes. The four different classes are: Class 1-Cloth Mask which has 707 images, Class 2-FFP2 Mask which consists of 269 images, Class 3 Surgical Mask which has 649 images and Class 4- No Mask which consists of 677 images. The above distribution of the dataset is quite balanced. We have divided the data into train and test set with a distribution of 80% and 20% respectively so now the train set has 1842 images and the test set has 460 images.





2.4 K-FOLD CROSS-VALIDATION

K-fold cross-validation is a technique to evaluate a machine learning model on different data samples using a simple data resampling procedure. In this method, we divided out the whole dataset into 10 groups(folds), out of which, 1 was taken as a testing dataset and the remaining 9 were taken as a training dataset. In this manner, we repeated this process 10 times with random shuffling of the dataset and calculated accuracy in each fold. Then, we calculated the average accuracy from the accuracy of all folds.

Fold	**Pre-Bias**				**Post-Bias**			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
1	0.86	0.86	0.86	0.87	0.90	0.90	0.90	0.90
2	0.91	0.91	0.91	0.91	0.92	0.92	0.92	0.93
3	0.88	0.88	0.88	0.88	0.91	0.91	0.91	0.91
4	0.89	0.89	0.89	0.89	0.91	0.90	0.90	0.91
5	0.90	0.90	0.90	0.90	0.92	0.91	0.91	0.92
6	0.91	0.91	0.91	0.91	0.92	0.92	0.92	0.92
7	0.91	0.91	0.91	0.91	0.89	0.89	0.89	0.90
8	0.81	0.84	0.81	0.77	0.94	0.93	0.93	0.94
9	0.88	0.88	0.88	0.88	0.92	0.91	0.91	0.92
10	0.91	0.91	0.91	0.91	0.95	0.95	0.95	0.95

As observed, there is some deviation from the standard test/train method and k-fold cross-validation for our model. Hence, we can state that our model is efficient to provide output with 92 to 93% accuracy for the given testing data.

The accuracy of standard evaluation (fixed train and test split) was around 95%. The average accuracy when using pre-bias data was around 88.6% but when used post-bias data the accuracy was around 92.1% which means the accuracy of the model is increased after retraining data and biases.

We observed that the K-fold evaluation was efficient in testing the model as it provides consistent results with different training and testing datasets. After adding the post bias data we observe a slight increase in the K-fold results. This was mainly due to balancing the dataset and adding more testing images.

REFERENCES

[1] Joseph Nelson (JAN 26, 2020): Why Image Preprocessing and Augmentation Matter, URL:https://blog.roboflow.com/why-preprocess-augment/

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

[3] https://machinelearningmastery.com/confusion-matrix-machine-learning/

[4] https://deeplizard.com/

[5] https://deeplizard.com/learn/video/0LhiS6yu2qQ

[6]Deeplizard CNN with PyTorch series https://www.youtube.com/channel/UC4UJ26WkceqONNF5S26OiVw

[7] PyTorch_tutorials - https://github.com/gaurav67890/Pytorch_Tutorials/blob/master/cnn-scratch-training.ipynb