COMP 472 - Project 2 Report

Group NS_07
Jason Dinh (40129138) - Data Specialist
Axel Dzeukou (40089940) - Training Specialist
Vyacheslav Medvedenko (40134207) - Evaluation Specialist
Dante Di Domenico (40125704) - Compliance Specialist

Github Repository: https://github.com/jasondinh99/COMP472_Project_P2

IMPORTANT: Trained model
We cannot include our trained model in the Moodle submission and on Git because the size is too large for both (350+ MB). To use our trained model, please download the model from our Google Drive and save it in the root folder (the same folder as the 'main.py' file).
Part 1 Model: https://drive.google.com/file/d/1RteV9Hwqbqr8MRca4xAm5OW3FatnXcCT/view
Part 2 Model: https://drive.google.com/file/d/16sPB5_CYc3kLfUfO_001vAhASq3fKzpb/view

CONTENTS

- PART 1 -

Dataset

Developing an AI can be difficult at times, but what is the most crucial within the process is choosing the right data set when needing to learn how to decipher images. In this scenario, this AI is being taught to analyze images of people's faces, the purpose is to determine not only if the individual is wearing a mask or not, but what specific mask they are wearing as well, whether it be a cloth mask, procedural mask, or any of the FFP2/N95/KN95 masks.

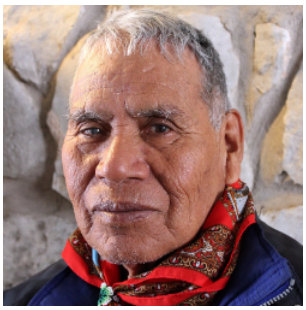
For this project, numerous images of individuals are needed to be collected by the AI and deciphered into their respective categories, which from top left to bottom right [see Figure 1] reads cloth-material mask, particulate respirator mask such as the N95, no mask, and surgical mask,. For the AI to have sufficient material to train with, over 1600 photos have been collected from Kaggle.

For each class, the specific number of photos are as shown:

- 420 no mask images. Sources: [1], [3]
- 402 cloth mask images. Sources: [1], [2], [3]
- 400 surgical mask images. Sources: [1], [2], [3]
- 410 N95 mask images. Sources: [1], [2], [3] [4]

Cloth mask (Class #0)

N95 mask (Class #1)



No mask (Class #2)

Surgical mask (Class #3)

Figure 1: Categories of images

Just so that the dataset would have some distinction to its photos, the angles of which they were taken vary from close-ups, side profiles and portraits of individual's faces. Each image contains only one fully visible face and photos of groups of people were not taken into account as they were not necessary. To avoid any complications when implementing the photos into the program, size formatting was needed with almost all the photos, as well as manually sorting them into each of their designated categories. As it would be easier to maintain a consistent size for the photos, many were cropped to a 1:1 ratio.

As for the process, not all the photos were used in both the training and testing. Around 1200 were used within the training segment while approximately 400 were used for the testing segment, with an equal number from each category.

CNN Architecture

In the beginning, the datasets of images are all reshaped into size 150 x 150, transformed into tensors, and then converted into dataloaders. For the process of testing and training to pass quicker, shuffling was set to true such that the training data is reshuffled at every epoch while using 4 workers.

From here, the 'fit' function is run, which in turn trains the model, prints the validation accuracy for every epoch, and saves the model after 10 epochs.

To extend the functionality of "torch.nn.Module", a base class is created (the base class used to develop all neural networks). We then add various functionalities to the base class to train and validate the model, as well as get the result for each epoch.

Our model is inspired by reference [6], which will be initiated from the "FaceMaskClassifier" class which inherits from the base class. Within this model, there are 3 CNN blocks, each consisting of 2 convolution layers and 1 max-pooling layer. While stride and padding are set to 1, the Relu activation function is used so that negative values are removed from the feature map as pixel values cannot be within the negative range of integers. In order to get more accurate results and reduce bias we changed (20): Linear(in_features=512, out_features=6, bias=True) from part 1 to (20): Linear(in_features=512, out_features=4, bias=True) in part 2 to match the number of classes.

Once convolution has been applied and features from the image have been extracted, a flatten layer is used to flat the tensor. The tensor contains three dimensions to begin with, but after the flatten layer has undergone its job, it leaves the converted tensor to be one-dimensional. Once complete, three linears are added to reduce the size of the tensor and learn its features.

At the end of the training process, the saved model is then reloaded so that it may provide us with the predicted labels of our testing set. The predicted labels are classified as either “0”, “1”, “2”, or “3” which stand for “cloth_mask”, ”n95_mask”, ”no_mask”, ”surgical_mask” respectively.

In part 1 we would save the model that gives us a validation accuracy of at least 70% which was not very ideal. So instead we reduced the number of epochs from 30 to 10 and after 10 epochs used the resulting model to test the validation set for each fold. The final result from running our trained model on the test data yielded an accuracy of 0.6094 or 60.94%. What would have helped us out a bit more would be to gain a deeper understanding of how the functions in our CNN architecture work.

Below is what our CNN architecture segment looks like in part 2:

```
FaceMaskClassification {
  (network): Sequential {
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU()
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU()
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU()
    (14): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (15): Flatten()
    (16): Linear(in_features=82944, out_features=1024, bias=True)
    (17): ReLU()
    (18): Linear(in_features=1024, out_features=512, bias=True)
    (19): ReLU()
    (20): Linear(in_features=512, out_features=4, bias=True)
  }
}
```

Evaluation

For the evaluation, we currently use a set of 400 images. First, we store the labels of each image in a list, and do the same thing for the neural network's predictions. This will yield two lists of integers (numbering 0 - 3 to represent each class), both of length 400. Cloth mask, N-95 mask, No Mask, and Surgical Mask being 0, 1, 2, and 3 respectively.

These lists are then used in Scikit-learn's “confusion_matrix()” function to create a multi-class confusion matrix. The highest values in the confusion matrix are visible along the diagonal, which you can somewhat tell what the accuracy of the neural network will be.

The appropriate formulae as well as Scikit's functions are used to calculate the accuracy, recall, precision and f1 measure for each of the four classes, as well as for the combination of all the classes.

So for the calculations of these metrics, the theoretical formulae from the lecture slides of manipulating confusion matrix values were used. However, it was also interesting to see what the results were for the total of the four classes. For those, sk-learn’s functions were used and gave the same values for all four metrics, which makes sense, given that False Positive and False Negative values were the same.

Multi-Class Confusion matrix:

```
****************************************
        Confusion Matrix
****************************************
Columns are predictions, rows are labels

[[60 14  7 15]
 [21 52  1 26]
 [15  2 85  4]
 [26 23  4 45]]
```

Metrics:

```
CLASS # 0
Accuracy:  0.76
Precision:  0.49
Recall:  0.62
F1_score:  0.55

CLASS # 1
Accuracy:  0.78
Precision:  0.57
Recall:  0.52
F1_score:  0.54

CLASS # 2
Accuracy:  0.92
Precision:  0.88
Recall:  0.80
F1_score:  0.84

CLASS # 3
Accuracy:  0.76
Precision:  0.50
Recall:  0.46
F1_score:  0.48


Total (all classes together):
Accuracy:  0.605
Precision:  0.605
Recall:  0.605
F1_score:  0.605
```

Cloth Mask	Predicted: Yes	Predicted: No
Actual: Yes	60	36
Actual: No	62	242

N-95 Mask	Predicted: Yes	Predicted: No
Actual: Yes	52	48
Actual: No	39	261

No Mask	Predicted: Yes	Predicted: No
Actual: Yes	85	21
Actual: No	12	282

Surgical Mask	Predicted: Yes	Predicted: No
Actual: Yes	45	53
Actual: No	45	257

- PART 2 -

Bias in AI

When it comes to developing any AI based program, any possible bias must be taken into consideration in order to obtain the best and most accurate result no matter the input data. Otherwise, if it is not taken care of, it will produce discriminatory results towards different groups of people whether the category be age, sex or race.

In the second part of the project, to conclude our theories about potential Discriminatory Bias within our AI, we want to study it by going through an analysis of two bias categories, namely gender and race. For both of these categories, we have split our dataset into two groups. For gender, we have male and female, while race is split between pale skin and dark skin. It is pretty conclusive why gender was chosen to have only two subsets, but for race, a third, middle ground option wasn't chosen, for example, tanned skin, because it seemed best to have the program have only the extremes of the spectrum classified, as a sort of "yes or no" or "true or false" and not "maybe".

The same dataset structure is kept between all of the new groups, with each being sorted by the type of mask that they are wearing, if any. For the model used, it is the same CNN one we have used in the first part of the project and it is run on each group separately.

From the results below, you can see that there is no clear bias for when we tested the gender groups. They show very similar evaluation metrics for both the male and female datasets [Figure 2]. On the other hand, there is a possible bias amongst our AI for the race category.

It seems that most of the evaluation metrics have similar results between the pale skin and dark skin category, but it is not the same when comparing them within the no mask class and surgical mask. When comparing the 'non-masked dark skin' images to the 'non-masked pale skin' images, the recall evaluation metric while identifying is lower by an approximate 12%, leaving the results to be 0.82 and 0.94, respectively [Figure 3]. The conclusive result of this means that the AI will misidentify dark skin people to be wearing a mask more often than pale skin people even though one is not present in the image, this overall leads to a higher number of False Negative predictions. The reason for this is because many masks have similar colors to the skin tone of dark skin people in the images, making it harder for the AI to distinguish between masked and unmasked. Similarly, for 'surgical masks', 'dark skin' images have a high False Positive prediction, and a 20% lower precision value compared to 'pale skin' images, with results being 0.61 and 0.81, respectively. However, this is because we do not have as many

images of ‘dark skin’ people with surgical masks which will then not make the AI have the proper training to assess many of the ‘dark skin’ images.

```
**************************************
            Confusion Matrix
**************************************
Columns are predictions, rows are labels

[[144   8   3  10]
 [ 16 153   1  29]
 [ 20   2 170   2]
 [ 21  12   4 110]]

CLASS # 0
Accuracy:  0.89
Precision:  0.72
Recall:  0.87
F1_score:  0.79

CLASS # 1
Accuracy:  0.90
Precision:  0.87
Recall:  0.77
F1_score:  0.82

CLASS # 2
Accuracy:  0.95
Precision:  0.96
Recall:  0.88
F1_score:  0.91

CLASS # 3
Accuracy:  0.89
Precision:  0.73
Recall:  0.75
F1_score:  0.74


Total (all classes together):
Accuracy:  0.8184397163120567
Precision:  0.8184397163120567
Recall:  0.8184397163120567
F1_score:  0.8184397163120567
```

```
**************************************
            Confusion Matrix
**************************************
Columns are predictions, rows are labels

[[196  17   6  17]
 [ 18 166   0  26]
 [  9   1 214   2]
 [ 17  21   5 210]]

CLASS # 0
Accuracy:  0.91
Precision:  0.82
Recall:  0.83
F1_score:  0.82

CLASS # 1
Accuracy:  0.91
Precision:  0.81
Recall:  0.79
F1_score:  0.80

CLASS # 2
Accuracy:  0.98
Precision:  0.95
Recall:  0.95
F1_score:  0.95

CLASS # 3
Accuracy:  0.90
Precision:  0.82
Recall:  0.83
F1_score:  0.83


Total (all classes together):
Accuracy:  0.8497297297297297
Precision:  0.8497297297297297
Recall:  0.8497297297297297
F1_score:  0.8497297297297297
```

Male (Left) VS Female (Right) Dataset [Figure 2]

```
**************************************  **************************************
               Confusion Matrix                        Confusion Matrix
**************************************  **************************************
Columns are predictions, rows are labels  Columns are predictions, rows are labels

[[46  2  3  1]                          [[294  23   6  26]
 [ 2 40  1 10]                           [ 32 280   0  44]
 [14  1 77  2]                           [ 15   2 307   2]
 [ 3  2  0 20]]                          [ 35  31   9 300]]

CLASS # 0                               CLASS # 0
Accuracy:  0.89                         Accuracy:  0.90
Precision:  0.71                        Precision:  0.78
Recall:  0.88                           Recall:  0.84
F1_score:  0.79                         F1_score:  0.81

CLASS # 1                               CLASS # 1
Accuracy:  0.92                         Accuracy:  0.91
Precision:  0.89                        Precision:  0.83
Recall:  0.75                           Recall:  0.79
F1_score:  0.82                         F1_score:  0.81

CLASS # 2                               CLASS # 2
Accuracy:  0.91                         Accuracy:  0.98
Precision:  0.95                        Precision:  0.95
Recall:  0.82                           Recall:  0.94
F1_score:  0.88                         F1_score:  0.95

CLASS # 3                               CLASS # 3
Accuracy:  0.92                         Accuracy:  0.90
Precision:  0.61                        Precision:  0.81
Recall:  0.80                           Recall:  0.80
F1_score:  0.69                         F1_score:  0.80


Total (all classes together):           Total (all classes together):
Accuracy:  0.8169642857142857           Accuracy:  0.8399715504978663
Precision:  0.8169642857142857          Precision:  0.8399715504978663
Recall:  0.8169642857142857             Recall:  0.8399715504978663
F1_score:  0.8169642857142857           F1_score:  0.8399715504978663
```

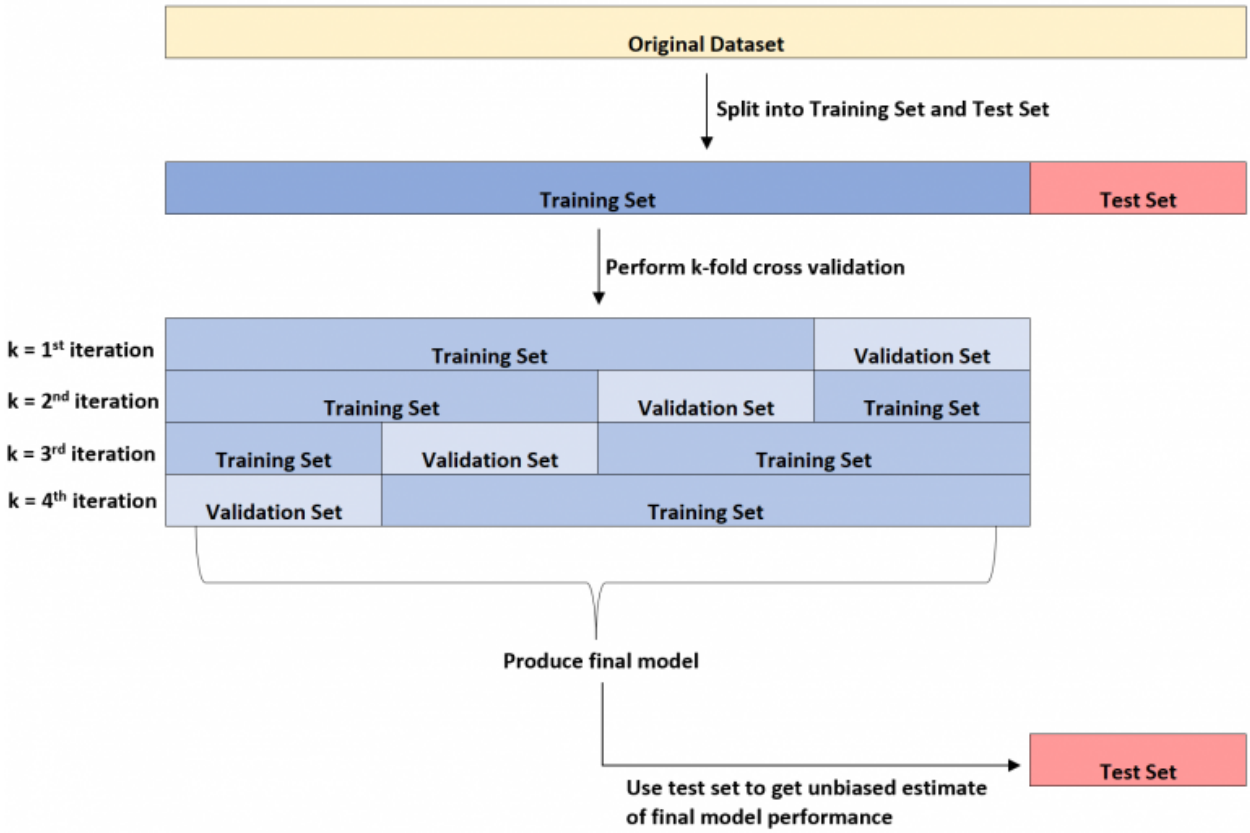
Dark skin Dataset (Left) VS Light skin Dataset (Right) [Figure 3]

Seeing as bias is only exhibited in the race category of the no mask class, it did not take much effort to go ahead and attempt to correct it. Our CNN architecture model was updated during the second part of the project and was implemented. What's expected at the end of this, is for our AI to have similar metric evaluations between both groups of each bias category such that the bias itself will no longer exist.

In the end of using the newer model, we had achieved what we had set out for and with it we were able to overcome the apparent bias we had with the dark skin people having no masks on. Another viable solution to eliminating bias within our AI would be to train the model against more images or the accused group so that it may be more accustomed to those pictures and then provide us with better and more accurate results.

K-fold validation

In the second part of the project, we applied a 10 fold validation which first splits the dataset into training and test sets, then splits the training set further into training and validation sets. We then run the 10-fold validations with the training and validation sets to optimize our parameters on our model. Once appropriate results are received and satisfy us, it is run once more with the original training and test sets so that it may return our trained model, to which we save as “TrainedPart2_model.sav”.



Visual explanation of k-fold validation [Figure 4] [7]

Analysis between test/train split and k-fold

In the first part of the project, we used ‘test_train_split’ on the dataset so that it splits to 80% and 20% for training and test, respectively. We then split the training set to 80% and 20% for training and validation, respectively, this results in a 1-fold validation. In the second part of the project, a 10 fold validation is used with the same percentage split on the datasets. With each fold, after 10 epochs the resulting model is used to test the validation set.

Result after applying train_test_split
The result can be viewed on page 5 of this report
Result after applying k-fold cross validation
The result can be viewed in this link:
https://drive.google.com/file/d/1_Kc2FXgF8wfKe3BDScEr3s-4c6O6qGuA/view
This file is also included in our Moodle submission and Git repository.

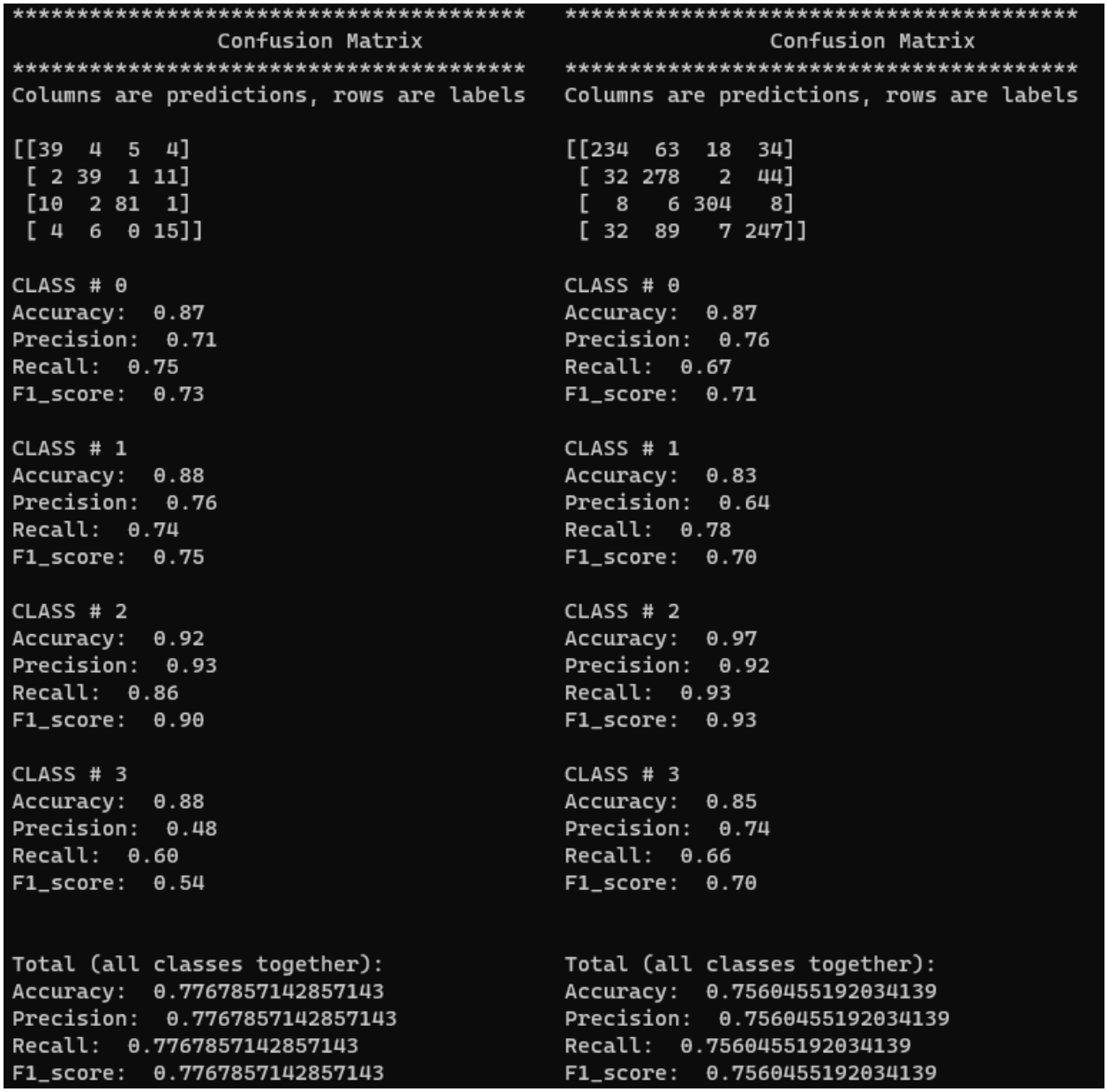
As we can notice by using k-fold cross validation in our project, we get better results. Initially the accuracy of our model was about 60% but now because of using k-fold cross validation we get 65%.

Results of categories after K-fold

We can see below that after applying k-fold validation, all the accuracy results between different categories of genders and races now have differences of under 10%. Which means that this has reduced our bias and gives us a more accurate result.

```
**************************************      **************************************
               Confusion Matrix                        Confusion Matrix
**************************************      **************************************
Columns are predictions, rows are labels    Columns are predictions, rows are labels

[[117  19   6  23]                         [[156  48  17  15]
 [ 15 156   2  26]                          [ 19 161   1  29]
 [ 11   6 174   3]                          [  7   2 211   6]
 [ 17  38   3  89]]                         [ 19  57   4 173]]

CLASS # 0                                   CLASS # 0
Accuracy:  0.87                             Accuracy:  0.86
Precision:  0.73                            Precision:  0.78
Recall:  0.71                               Recall:  0.66
F1_score:  0.72                             F1_score:  0.71

CLASS # 1                                   CLASS # 1
Accuracy:  0.85                             Accuracy:  0.83
Precision:  0.71                            Precision:  0.60
Recall:  0.78                               Recall:  0.77
F1_score:  0.75                             F1_score:  0.67

CLASS # 2                                   CLASS # 2
Accuracy:  0.96                             Accuracy:  0.96
Precision:  0.94                            Precision:  0.91
Recall:  0.90                               Recall:  0.93
F1_score:  0.92                             F1_score:  0.92

CLASS # 3                                   CLASS # 3
Accuracy:  0.84                             Accuracy:  0.86
Precision:  0.63                            Precision:  0.78
Recall:  0.61                               Recall:  0.68
F1_score:  0.62                             F1_score:  0.73


Total (all classes together):               Total (all classes together):
Accuracy:  0.7602836879432624               Accuracy:  0.7578378378378379
Precision:  0.7602836879432624              Precision:  0.7578378378378379
Recall:  0.7602836879432624                 Recall:  0.7578378378378379
F1_score:  0.7602836879432624               F1_score:  0.7578378378378379
```

Male (Left) VS Female (Right) Dataset after k-fold [Figure 5]



Dark skin Dataset (Left) VS Light skin Dataset (Right) after k-fold [Figure 3]

References

[1] A. Jangra, “Face mask detection ~12K images dataset,” *Kaggle*, 26-May-2020. [Online]. Available: https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset. [Accessed: 04-Jun-2022].

[2] D. Makwana, “Face mask classification,” *Kaggle*, 25-Jul-2020. [Online]. Available: https://www.kaggle.com/datasets/dhruvmak/face-mask-detection. [Accessed: 04-Jun-2022].

[3] W. Intelligence, “Face mask detection dataset,” *Kaggle*, 14-Jun-2020. [Online]. Available: https://www.kaggle.com/datasets/wobotintelligence/face-mask-detection-dataset. [Accessed: 07-Jun-2022].

[4] coffee124, “N95 face mask,” *Kaggle*, 05-Jun-2022. [Online]. Available: https://www.kaggle.com/datasets/coffee124/facemaskn95. [Accessed: 07-Jun-2022].

[5] P. Sharma, “Convolutional Neural Network Pytorch: CNN using pytorch,” *Analytics Vidhya*, 10-May-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/10/building-image-classification-models-cnn-pytorch/. [Accessed: 04-Jun-2022].

[6] P. Soni, “Train CNN model with pytorch,” *Medium*, 25-Apr-2022. [Online]. Available: https://medium.com/thecyphy/train-cnn-model-with-pytorch-21dafb918f48. [Accessed: 04-Jun-2022].

[7] Zach, “Validation set vs. test set: What's the difference?,” *Statology*, 20-Sep-2021. [Online]. Available: https://www.statology.org/validation-set-vs-test-set/. [Accessed: 22-Jun-2022].