Project Assignment, Part II

AI Face Mask Detector

Submitted to:

Dr. René Witte

Due date:
Wednesday, June 22th

By

Team FS-03
Team Members :
Sherif Ghaffar (40058600 - GhaffarSherif - Data Specialist)
Arshia Hamidi (40068250 - ashthedaddy - Training Specialist)
Joseph Goldberger (21958283 - j-gold77 - Evaluation Specialist)
Ali Turkman (40111059 - yellowsub12 - Compliance & Data Specialist)

Github Repo
https://github.com/yellowsub12/Artifical-Intelligence-Project

Submitted on 22/06/2022

Gina Cody School of Engineering and Computer Science

Concordia University

We certify that this submission is our original work and meets the faculty's expectations of originality.

Table of Contents

Dataset

Our original dataset held 4316 images which were divided 75-25 between the training and testing dataset. However, our initial dataset was flawed for several reasons. First of all, it held lots of

duplicates that weren't noticed, and secondly the number of images in each category was uneven, with considerably more images in the N95 and NoMask categories compared to the Surgical and Cloth categories. This was due to the fact that we found an entire Kaggle dataset with thousands of N95 and NoMask images while the others had to be manually sorted. Therefore, to resolve these issues, we first used a special software to detect duplicates and erase them, which removed 225 duplicate images from our original dataset. And then we also trimmed the original dataset. Since the Cloth category was the smallest, holding 668 images, we trimmed down all other categories to 668 images each as well to make everything equal. The result is a total of 2485 images for the dataset, and since the training-testing dataset divide is 75-25, we therefore used 1865 images for training and 620 for testing.

The next goal was to produce the dataset to test the level of bias in the CNN algorithm, although this part will be explained more in detail in the bias section. The two categories chosen were Age and Gender, and thus we decided to create 4 new categories, which are: Child, Adult, Male and Female. Each mask type inside of each category holds around 100 images, all of them taken from the dataset, producing a total of 1603 images spread over 16 sub-categories (4 per each of the Bias category). As for the preprocessing resizing of the dataset images, it was once again done in the Python program, and not externally with another software to do that. PyTorch already has Transformation modules just for this purpose, with images being transformed to 32 x 32 size.

Bias

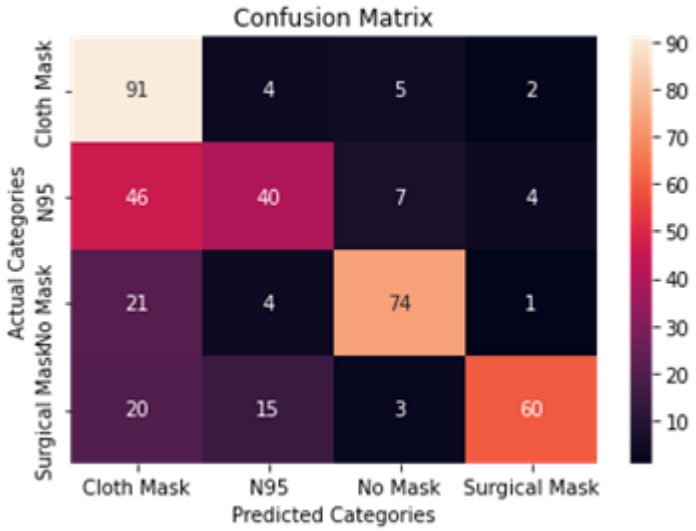
As mentioned in the Dataset section, the 2 chosen categories are Gender and Age. Age is subdivided into Child and Adult, and Gender is subdivided into Male and Female. The purpose of this Phase of the project is to account for disparities in accuracy when it comes to different people, as our Face Mask Detector may only be good for say, adult males, while severely discriminating

against women and children. With that being said, here are the results of the dataset with the noted bias categories.

We have observed some discrepancies between the different dataset bias categories.

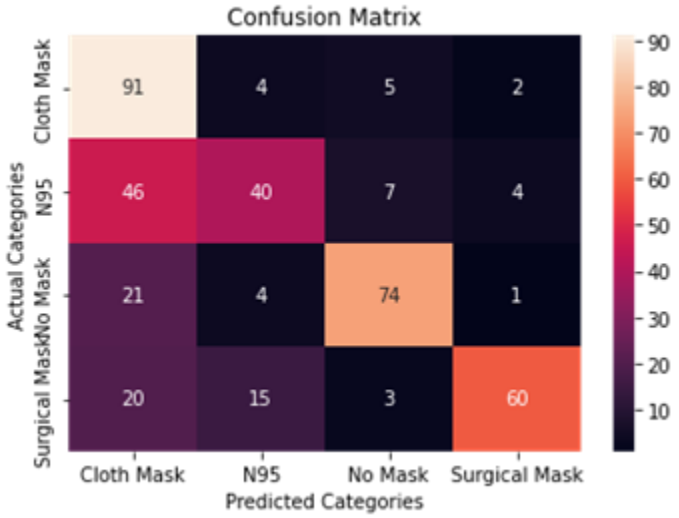
The following confusion Matrices results allow us to find these biases by comparing each of the four categories (Male, Female, Child, Adult):

Adult:



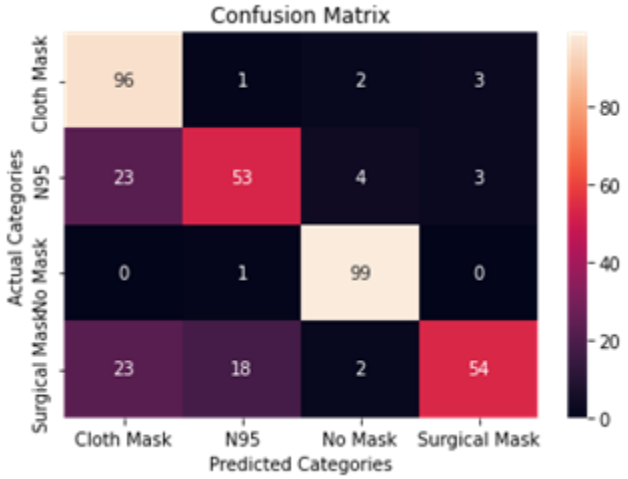
	precision	recall	f1-score	support
0	0.89	0.51	0.65	178
1	0.41	0.63	0.50	63
2	0.74	0.83	0.78	89
3	0.61	0.90	0.73	67
accuracy			0.67	397
macro avg	0.66	0.72	0.67	397
weighted avg	0.73	0.67	0.67	397

Child:



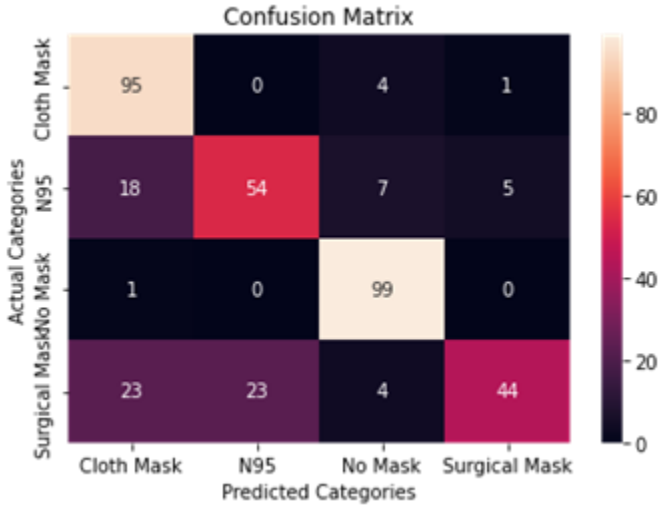
	precision	recall	f1-score	support
0	0.89	0.51	0.65	178
1	0.41	0.63	0.50	63
2	0.74	0.83	0.78	89
3	0.61	0.90	0.73	67
accuracy			0.67	397
macro avg	0.66	0.72	0.67	397
weighted avg	0.73	0.67	0.67	397

Male:



	precision	recall	f1-score	support
0	0.94	0.68	0.79	142
1	0.64	0.73	0.68	73
2	0.99	0.93	0.96	107
3	0.56	0.90	0.69	60
accuracy			0.79	382
macro avg	0.78	0.81	0.78	382
weighted avg	0.84	0.79	0.80	382

Female:



	precision	recall	f1-score	support
0	0.95	0.69	0.80	137
1	0.64	0.70	0.67	77
2	0.99	0.87	0.93	114
3	0.47	0.88	0.61	50
accuracy			0.77	378
macro avg	0.76	0.79	0.75	378
weighted avg	0.84	0.77	0.79	378

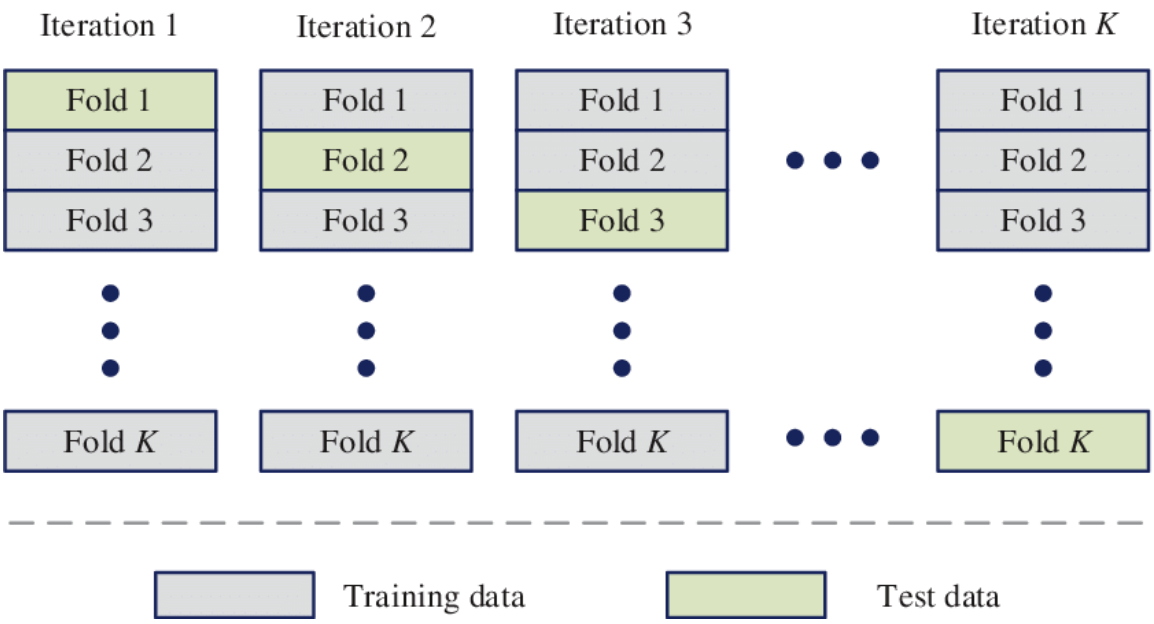
Interestingly there was no difference in bias between the age groups: Child and Adult. The Accuracy percentage were both at 67%. There was a slight difference between Male and Female on the other hand, with the accuracies of 79% and 77% respectively.

However, these results were not significant enough where the bias might be a large enough of a concern.

When Comparing between the Age and Gender categories, there seems to be more discrepancies between them with a difference of around 10% accuracy. This could indicate that the factor that the model relies on more heavily is in fact the gender rather than the age.

K-fold Cross-Validation

The K-fold Cross-Validation is a method used to estimate the performance of a machine learning algorithm. The way that it works is that it runs k (specified by user) experiments, where each time the algorithm tests on 1/k of the data and trains on the rest, with the final result being the average of prior results. For our purposes, the k-fold cross-validation will be used to compare its performance to the prior model, as the k-fold cross-validation method generally has a lower bias than other methods. Our dataset with the bias classifications has 1603 images , thus with 10 folds, the training dataset will receive 160 images from the dataset in each fold, and the testing dataset will receive the remaining 1403 images. Every fold will randomly place one picture, thus one that was in the training dataset in fold 1 may end up in the testing dataset in a different fold.



Prior to k-fold cross validation, our model's accuracy was about 77%. Accuracy has now increased to 80% after using k-fold cross validation. In essence, there will be 10 folds of iteration altogether, during which the training dataset will receive new photos to train on, and the testing dataset will receive different images to test on. Thus, To increase the model's accuracy, use of training dataset has been maximised. As for the code, we used the KFold function from lab 6 and code developed ourselves, on the 1603 bias dataset, and with 10 epochs per each of the 10 folds.

```
from sklearn import datasets
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import KFold


kfold = KFold(n_splits=10, shuffle=True, random_state=None)
fold_value = 1
for training_id, testing_id in kfold.split(dataset):
```

Result After applying k-fold cross validation

The outcome of k-fold cross validation, with k=10, is shown here. We have 10 epochs in each fold. Additionally, we utilise the "classification method" to print accuracy, precision, recall, and f1-score at the conclusion of each fold.

Fold Number: 1

Epoch [1/10], Step [23/23], Loss: 0.7041, Accuracy: 69.44%

…

Epoch [10/10], Step [23/23], Loss: 0.4567, Accuracy: 80.56%

	precision	recall	f1-score	support
0	0.63	0.74	0.68	86
1	0.86	0.45	0.59	158
2	0.90	0.96	0.93	94
3	0.39	0.86	0.54	44
accuracy			0.69	382
macro avg	0.69	0.75	0.68	382
weighted avg	0.76	0.69	0.69	382

Fold Number: 2

Epoch [1/10], Step [23/23], Loss: 1.0866, Accuracy: 50.00%

…

Epoch [10/10], Step [23/23], Loss: 0.5811, Accuracy: 77.78%

	precision	recall	f1-score	support
0	0.84	0.73	0.78	118
1	0.78	0.62	0.69	105

2	0.94	0.95	0.94	99
3	0.54	0.87	0.66	60
accuracy			0.78	382
macro avg	0.78	0.79	0.77	382
weighted avg	0.80	0.78	0.78	382

Fold Number: 3

Epoch [1/10], Step [23/23], Loss: 1.0474, Accuracy: 41.67%

…

Epoch [10/10], Step [23/23], Loss: 0.4367, Accuracy: 83.33%

	precision	recall	f1-score	support
0	0.87	0.68	0.76	131
1	0.67	0.74	0.70	76
2	0.90	0.96	0.93	94
3	0.66	0.79	0.72	81
accuracy			0.78	382
macro avg	0.78	0.79	0.78	382
weighted avg	0.79	0.78	0.78	382

Fold Number: 4

Epoch [1/10], Step [23/23], Loss: 1.0466, Accuracy: 55.56%

…

Epoch [10/10], Step [23/23], Loss: 0.5819, Accuracy: 72.22%

	precision	recall	f1-score	support
0	0.86	0.68	0.76	129

1	0.71	0.65	0.68	91
2	0.95	0.92	0.94	103
3	0.51	0.83	0.63	59
accuracy			0.76	382
macro avg	0.76	0.77	0.75	382
weighted avg	0.79	0.76	0.77	382

Fold Number: 5

Epoch [1/10], Step [23/23], Loss: 0.9789, Accuracy: 61.11%

…

Epoch [10/10], Step [23/23], Loss: 0.2657, Accuracy: 88.89%

	precision	recall	f1-score	support
0	0.76	0.78	0.77	100
1	0.47	0.75	0.58	52
2	0.95	0.97	0.96	98
3	0.85	0.62	0.72	132
accuracy			0.77	382
macro avg	0.76	0.78	0.76	382
weighted avg	0.80	0.77	0.77	382

Fold Number: 6

Epoch [1/10], Step [23/23], Loss: 1.1512, Accuracy: 56.76%

…

Epoch [10/10], Step [23/23], Loss: 0.4071, Accuracy: 83.78%

	precision	recall	f1-score	support

0	0.88	0.73	0.80	123
1	0.76	0.64	0.70	98
2	0.93	0.95	0.94	98
3	0.59	0.90	0.71	63
accuracy			0.79	382
macro avg	0.79	0.81	0.79	382
weighted avg	0.81	0.79	0.79	382

Fold Number: 7

Epoch [1/10], Step [23/23], Loss: 0.8717, Accuracy: 64.86%

…

Epoch [10/10], Step [23/23], Loss: 0.5174, Accuracy: 83.78%

	precision	recall	f1-score	support
0	0.88	0.77	0.82	117
1	0.63	0.68	0.65	76
2	0.97	0.93	0.95	104
3	0.60	0.68	0.64	85
accuracy			0.78	382
macro avg	0.77	0.77	0.77	382
weighted avg	0.79	0.78	0.78	382

Fold Number: 8

Epoch [1/10], Step [23/23], Loss: 1.0058, Accuracy: 62.16%

…

Epoch [10/10], Step [23/23], Loss: 0.3003, Accuracy: 89.19%

	precision	recall	f1-score	support

0	0.74	0.79	0.76	95
1	0.82	0.54	0.65	127
2	0.95	0.93	0.94	102
3	0.47	0.79	0.59	58
accuracy			0.74	382
macro avg	0.74	0.76	0.74	382
weighted avg	0.78	0.74	0.75	382

Fold Number: 9

Epoch [1/10], Step [23/23], Loss: 0.8073, Accuracy: 64.86%

…

Epoch [10/10], Step [23/23], Loss: 0.4724, Accuracy: 83.78%

	precision	recall	f1-score	support
0	0.78	0.73	0.75	110
1	0.80	0.59	0.68	111
2	0.92	0.96	0.94	96
3	0.55	0.82	0.65	65
accuracy			0.76	382
macro avg	0.76	0.77	0.76	382
weighted avg	0.78	0.76	0.76	382

Fold Number: 10

Epoch [1/10], Step [23/23], Loss: 0.9444, Accuracy: 67.57%

…

Epoch [10/10], Step [23/23], Loss: 0.2337, Accuracy: 94.59%

	precision	recall	f1-score	support
0	0.80	0.77	0.78	107
1	0.67	0.73	0.70	77
2	0.94	0.94	0.94	100
3	0.75	0.74	0.75	98
accuracy			0.80	382
macro avg	0.79	0.79	0.79	382
weighted avg	0.80	0.80	0.80	382

K-Fold conclusion : Unfortunately, there was no drastic increase in the accuracy of the training model. We noticed an increase of 5% (from 75% to 80%) in testing accuracy which is definitely good, but we expected a higher increase.

CNN Architecture

NOTE : CNN Architecture has not changed since Phase 1, but some parameters (like # of epochs) have. The changes are noted over the original report.

Our CNN's architecture was inspired by the lab #07. The first part of our CNN has 4 convolutional layers, all of which are followed by batch normalisation, a leaky ReLU activation function and a 2D max pooling layer. This first part outputs a vector, which is then processed by the second part of our CNN, which consists of 3 fully connected layers with ReLU activations. This second part of the network also includes dropout layers to reduce overfitting. It ends with a softmax activation

function. The output of this second layer should be a number representing how likely the CNN thinks each class is for the input image given to it. As the TA instructed, we also saved our trained model separately. We couldn't figure out how to label a confusion matrix with Scorch, so we just used PyTorch to evaluate data then used seaborn to build the confusion matrices with labels. As for the Convolutional Neural Network (CNN), ours is based off the one that was implemented in lab #07, with much of the same functions and techniques, but instead of the CIFAR-10 dataset, we used our own dataset with different categories. The dataset is initially loaded with a DataLoader PyTorch object.

Our training parameters are:

- 30 epochs
- Batch size of 100
- Adam as the optimization algorithm
    - Learning rate of 0.001
    - Momentum of 0.9

The reason why we picked 30 epochs is because we found out that the model would reach an accuracy of 97-100% around 30 epochs and a bit before. However, this accuracy is suspiciously high, which could mean that our neural network is overfitting and we might lower the number of epochs for Phase II of the project.

Phase II : As mentioned above, the number of epochs in phase 1 was 30, which caused overfitting, we now reduced it to 20 to combat that.

Evaluation

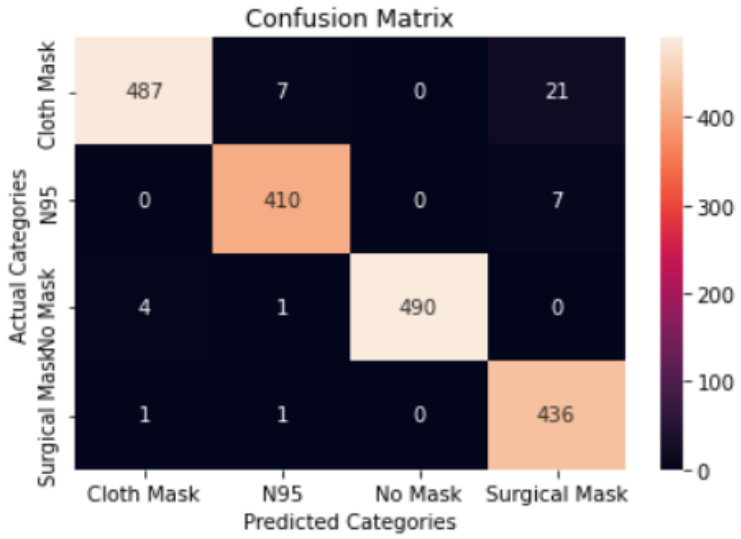
NOTE : Contrary to the Architecture section, the Evaluation has however greatly changed in Phase II due to the changes outlined above and most importantly in the dataset. The changes are noted below.

Using the *convolutional neural network (CNN),* after training the model on the training set, our model's training accuracy was measured by splitting the 1865 images into 19 batches. During our 20 epochs on the training data, our accuracy peaked at 100% accuracy at the 16th epoch, and then ended at 96% accuracy at the 20th epoch.

After saving the model, we were ready to calculate the accuracy, recall, f1 score and plot the confusion matrix on our testing data.

```
Epoch [1/20], Step [19/19], Loss: 1.1703, Accuracy: 49.23%
Epoch [2/20], Step [19/19], Loss: 0.8130, Accuracy: 58.46%
Epoch [3/20], Step [19/19], Loss: 0.5389, Accuracy: 81.54%
Epoch [4/20], Step [19/19], Loss: 0.7012, Accuracy: 73.85%
Epoch [5/20], Step [19/19], Loss: 0.7183, Accuracy: 61.54%
Epoch [6/20], Step [19/19], Loss: 0.6770, Accuracy: 80.00%
Epoch [7/20], Step [19/19], Loss: 0.3479, Accuracy: 81.54%
Epoch [8/20], Step [19/19], Loss: 0.5054, Accuracy: 81.54%
Epoch [9/20], Step [19/19], Loss: 0.4387, Accuracy: 90.77%
Epoch [10/20], Step [19/19], Loss: 0.4162, Accuracy: 81.54%
Epoch [11/20], Step [19/19], Loss: 0.2296, Accuracy: 89.23%
Epoch [12/20], Step [19/19], Loss: 0.3051, Accuracy: 86.15%
Epoch [13/20], Step [19/19], Loss: 0.3616, Accuracy: 83.08%
Epoch [14/20], Step [19/19], Loss: 0.2571, Accuracy: 87.69%
Epoch [15/20], Step [19/19], Loss: 0.1645, Accuracy: 92.31%
Epoch [16/20], Step [19/19], Loss: 0.0713, Accuracy: 100.00%
Epoch [17/20], Step [19/19], Loss: 0.3315, Accuracy: 90.77%
Epoch [18/20], Step [19/19], Loss: 0.0696, Accuracy: 98.46%
Epoch [19/20], Step [19/19], Loss: 0.0440, Accuracy: 98.46%
Epoch [20/20], Step [19/19], Loss: 0.0820, Accuracy: 96.92%
```

```
Time to evaluate our data.
First will be the training data.
```

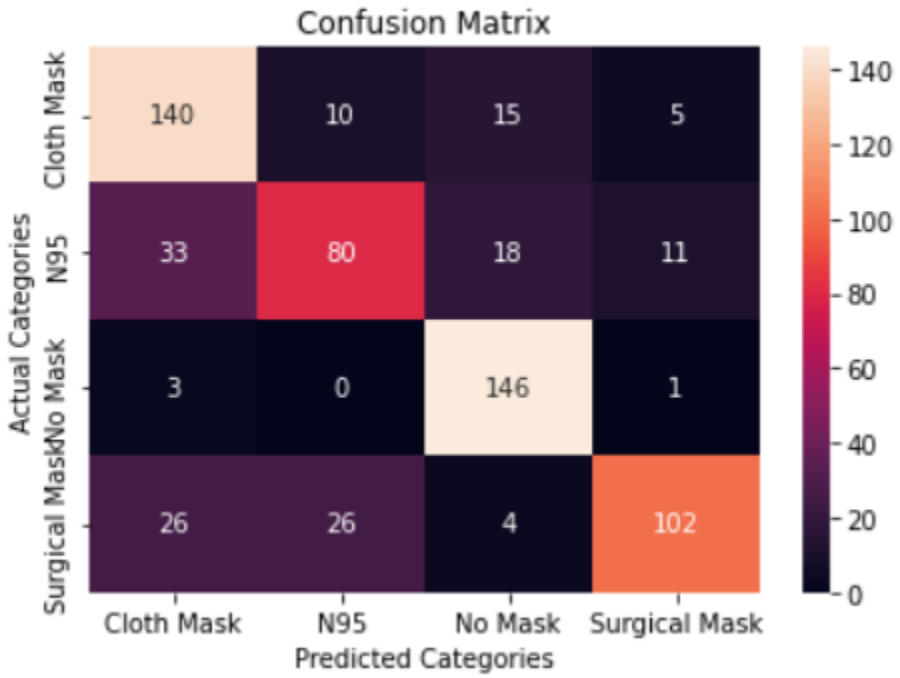


	precision	recall	f1-score	support
0	0.95	0.99	0.97	492
1	0.98	0.98	0.98	419
2	0.99	1.00	0.99	490
3	1.00	0.94	0.97	464
accuracy			0.98	1865
macro avg	0.98	0.98	0.98	1865
weighted avg	0.98	0.98	0.98	1865

Therefore we were able to achieve 98% accuracy for training. As for testing, it was done on 620 images and ended with 75% accuracy on the proper dataset.

```
Test Accuracy of the model on the 620 test images: 75.0 %
Test Accuracy of the model on the 620 test images: 74.0 %
Test Accuracy of the model on the 620 test images: 75.66666666666667 %
Test Accuracy of the model on the 620 test images: 76.25 %
Test Accuracy of the model on the 620 test images: 75.8 %
Test Accuracy of the model on the 620 test images: 75.5 %
Test Accuracy of the model on the 620 test images: 75.48387096774194 %
```

Now to evaluate the testing data.



	precision	recall	f1-score	support
0	0.82	0.69	0.75	202
1	0.56	0.69	0.62	116
2	0.97	0.80	0.88	183
3	0.65	0.86	0.74	119
accuracy			0.75	620
macro avg	0.75	0.76	0.75	620
weighted avg	0.78	0.75	0.76	620

Phase 1 : Our NN performance was great on our training data. Our best precision, recall and accuracy using f1 score were in the no mask category. Our neural network was 94% precise when labelling images with no mask. In this category it got confused most with cloth masks. It also had 90% recall and 92% f1 score in this category, so it performed above average. N95 and surgical masks were next in the rankings, scoring 80% and 77% respectively in precision. The NN on

N95s got confused with both surgical and cloth masks equally whereas the surgical masks were believed to be N95s mostly. Lastly we had cloth masks.

On the test dataset, the NN performance metrics were not quite as good. The overall accuracy was around 78%. Our precision, recall and f1-score were the best for the no mask class. The class that had the worst metrics was the cloth mask class.

We believe there is definitely room for improvement, especially on the cloth masks. The first thing we must do is balance our datasets better. All 4 of our classes should be representing a closer to equal amount of images. The cloth masks class has the lowest amount of images, which showed in the performance metrics for that class. Second, we believe that we overtrained the model, because by the 19th epoch on the training set we were getting 100% accuracy scores, but we continued for 11 more epochs, 33% of the total amount. This overtraining may have led to overfitting and thus our NN had trouble actually distinguishing differences on the testing data.

References

Datasets :

https://www.kaggle.com/datasets/coffee124/facemaskn95

https://www.kaggle.com/prithwirajmitra/covid-face-mask-detection-dataset

https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset

https://www.kaggle.com/datasets/dhruvmak/face-mask-detection

Lab #07 :

https://moodle.concordia.ca/moodle/pluginfile.php/5430653/mod_resource/content/5/lab07-q.pdf