AI Face Mask Detector Part 2 Report

Alexander Fulleringer (40005290) *Training Specialist*,
Cheng Chen (40222770) *Evaluation Specialist*, and
Jun Huang (40168167) *Data Specialist*
Team: OB_13

COMP 6721 Applied Artificial Intelligence

Dr. René Witte

June 8th, 2022

Abstract

This is part 2 of the report of AI Face Mask Detector project completed as part of Concordia University's COMP 6721 Applied AI course. Alexander Fulleringer works as the Training Specialist, Chen Cheng works as the Evaluation Specialist, and Jun Huang works as the Data Specialist. We use a balanced dataset of 5,000 pictures to train, validate and test our convolutional neural network. We look at the accuracy, precision, recall and f-measure of the model's performance on a withheld test set to properly gauge its effectiveness. We furthermore divide the images based on race and gender to ensure that our model has minimum bias with regards to its results.

Our full sources files are in our github repo:youyinnn/ai_face_mask_detector.

Dataset for Part One

For the dataset of our model, we collect 2,390 images across five different *classes* of images of people with or without masks. Since we need a balanced dataset, we apply five image transformation strategies for data augmentation as part of our pre-processing of the images. Table 1 shows the overall information of our dataset.

Table 1
Size and Source of Each Type in the Dataset for Part One

	Type	Size before Augmentation	Size after Augmentation	Source
0	Cloth Mask	418	1,000	Intelligence (2020)
1	No Face Mask	1,006	1,000	Intelligence (2020)
2	Surgical Mask	411	1,000	Intelligence (2020)
3	N95 Mask	387	1,000	COFFEE124 (2022)
4	Mask Worn Incorrectly	168	1,000	COFFEE124 (2022)
		2,390	5,000	**Total**

Data Collection

For data type “Cloth Mask”, “No Face Mask” and “Surgical Mask”, data are collected from the public dataset (Intelligence, 2020, Face Mask Detection). For data type “N95 Mask” and “Mask Worn Incorrectly”, are collected from the public dataset (COFFEE124, 2022, Face Mask Detection). If we consider 400 images for each type of the data, then the previous four types have balanced size, but the data of “Mask Worn Incorrectly” has half size of the others. For that, we consider not using any techniques to generate synthetic data as we think it might mislead the model performance. We randomly apply multiple augmentation strategies to obtain a balanced dataset.

Data Preprocessing

To input the data right away to the net for training and testing, we crop the data into a 1:1 resolution ratio and make the human face locate at the center of the image as much as possible. And then we resize all the data into 256×256 resolution. Note that the these were then further resized to 128×128 to reduce computational load. After we obtained the proper size of images’ data, we augment the data with the API provided by ***PyTorch***. The selected augmentation APIs and their parameters are listed below:

- ColorJitter(hue=.02, saturation=.02)
- RandomHorizontalFlip()
- RandomRotation(20, resample=PIL.Image.BILINEAR)
- RandomAdjustSharpness(sharpness_factor=2)
- RandomAutocontrast()

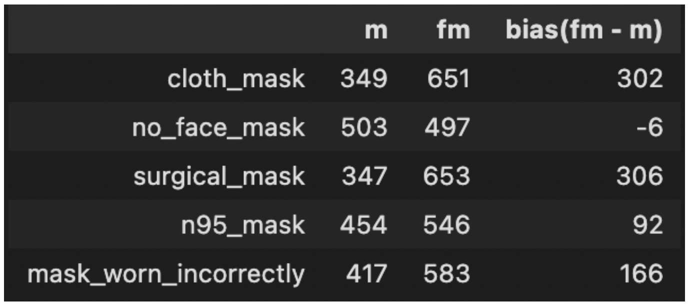
Dataset for Part Two

After evaluating our project's Part one, we observe that our model prediction on the type of *Mask Worn Incorrectly* has higher performance than the other types. We consider this might be due to the unbalanced data size of this type before the augmentation. Hence it leads to more duplication when we applied the augmentation to balance it to the size of 1,000. To address this issue, we collect more data for this particular type to make it has a size around 400. Table 2 shows the overall information of our new dataset in project Part two.

Table 2
Size and Source of Each Type in the Dataset for Part Two

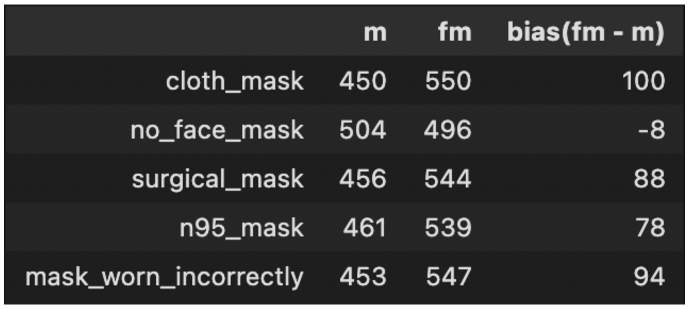
	Type	Size before Augmentation	Size after Augmentation	Source
0	Cloth Mask	418	1,000	Intelligence (2020)
1	No Face Mask	830	1,000	Intelligence (2020)
2	Surgical Mask	410	1,000	Intelligence (2020)
3	N95 Mask	404	1,000	COFFEE124 (2022)
4	Mask Worn Incorrectly	429	1,000	COFFEE124 (2022)
		2,491	5,000	**Total**

As we need to conduct the bias analysis on two sub types: gender, and race, We need to relabel our data set from five mask types into 20 mask-gender-race types. For gender type, we only consider two labels: (1) male; (2) female. For race type, we only consider two labels: (1) Caucasian and Asian; (2) African and Arab. Hence we have a total of 20 ($5 \times 2 \times 2$) labels. As will be introduce in the follow section, even though there are unbalance for two sub types, we still rebalance the data according to the size of each types. Their statistic before and after the rebalance are shown in Figure 1,2,3,4.



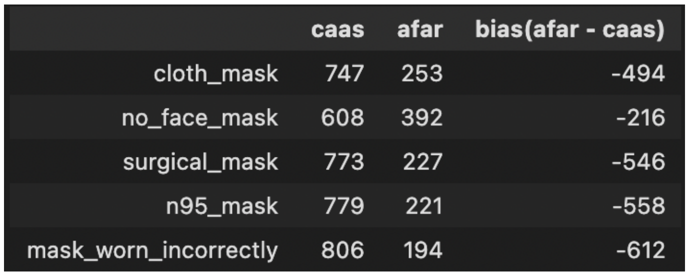
	m	fm	bias(fm - m)
cloth_mask	349	651	302
no_face_mask	503	497	-6
surgical_mask	347	653	306
n95_mask	454	546	92
mask_worn_incorrectly	417	583	166

Figure 1
Dataset Mask-Gender Statistic before Rebalance



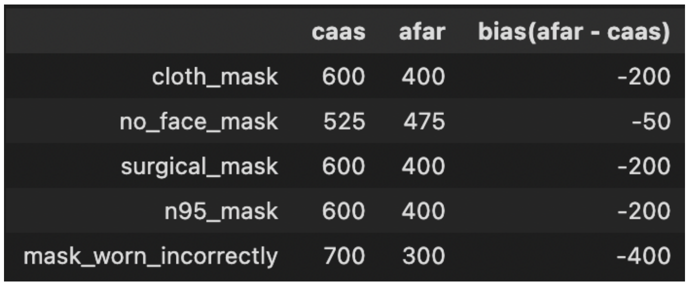
	m	fm	bias(fm - m)
cloth_mask	450	550	100
no_face_mask	504	496	-8
surgical_mask	456	544	88
n95_mask	461	539	78
mask_worn_incorrectly	453	547	94

Figure 2
Dataset Mask-Gender Statistic after Rebalance



	caas	afar	bias(afar - caas)
cloth_mask	747	253	-494
no_face_mask	608	392	-216
surgical_mask	773	227	-546
n95_mask	779	221	-558
mask_worn_incorrectly	806	194	-612

Figure 3
Dataset Mask-Race Statistic before Rebalance



	caas	afar	bias(afar - caas)
cloth_mask	600	400	-200
no_face_mask	525	475	-50
surgical_mask	600	400	-200
n95_mask	600	400	-200
mask_worn_incorrectly	700	300	-400

Figure 4
Dataset Mask-Gender Statistic after Rebalance

Convolutional Neural Network Architecture

Architecture, Part One

The primary model we used was the Base_CNN, shown Figure 5. It combines max pooling and convolutional layers before finishing with 3 linear ones. For our non-linearity we use the ReLU function as it is computationally efficient and effective Ye (2020). When designing the network we started with several alternating pooling and convolution layers with only one linear to perform the final classification, but found that more linear layers significantly improved performance with an allowable increase in training time.

For the convolution and pooling layers we sought inspiration from the famously successful AlexNet model. We noted that it generally increased the number of channels while reducing the dimensions each layer Nayak (2021). Reducing the dimensions of the data was important as early designs that retained dimensions resulted in our running out of memory .

The other variants, both of which performed worse than our base model, involved removing either pooling or convolution layers respectively. While dropping convolution layers does make the model smaller and slightly faster to train, the drop in performance is particularly severe. Dropping pooling layers results in a smaller, but still noticeable, performance loss. This is slightly counterintuitive as MaxPool layers do not actually extract new features like convolution or linear layers do. They can however be considered to prioritize certain activations so they are still clearly valuable.

Training Procedure, Part One

Our training process was completed in two parts. First we tuned hyperparameters and the architecture, then we trained a final model based on the tuning results. Before doing any validation we withhold 20% of the data for final testing so that none of our tuning was performed on our test set, keeping 80% as a training and validation set.

To tune the hyperparameters and architecture we used a stratified k-fold cross validation strategy. We used SciKit Learn's stratified k-fold function to split the remaining 80% into five balanced folds. The balancing is important as imbalanced datasets may lead to artificially good or bad results depending on how the split favours certain classes. We would try a certain model configuration by training it on four folds and validating it on the last, then repeating this so each fold was the test fold once. This allowed us to estimate our final test time performance with a given configuration without tuning our model on the test data.

We used a random search to find the best values for certain hyperparameters as described by Bergstra and Bengio Bergstra and Bengio (2012). As argued in their paper, this is more efficient than grid search and allowed us to tune the learning rate and training epochs with ease. We found that a learning rate of 1e-4 worked best with larger ones having accuracy varying between epochs even after extended training. We started off training for 25 epochs to save on time while we were establishing the general approach, but during our hyperparameter tuning we found that 50 epochs or so was sufficient to get good performance.

Finally, for the final model we took the structure and hyperparameters we'd found worked best on our training set and trained a model on the entire training and validation set. We then evaluated this model on the withheld data and report our metrics below. As this data was never seen at any point during the tuning and design process, the results should be representative of what the network can do in a real application.

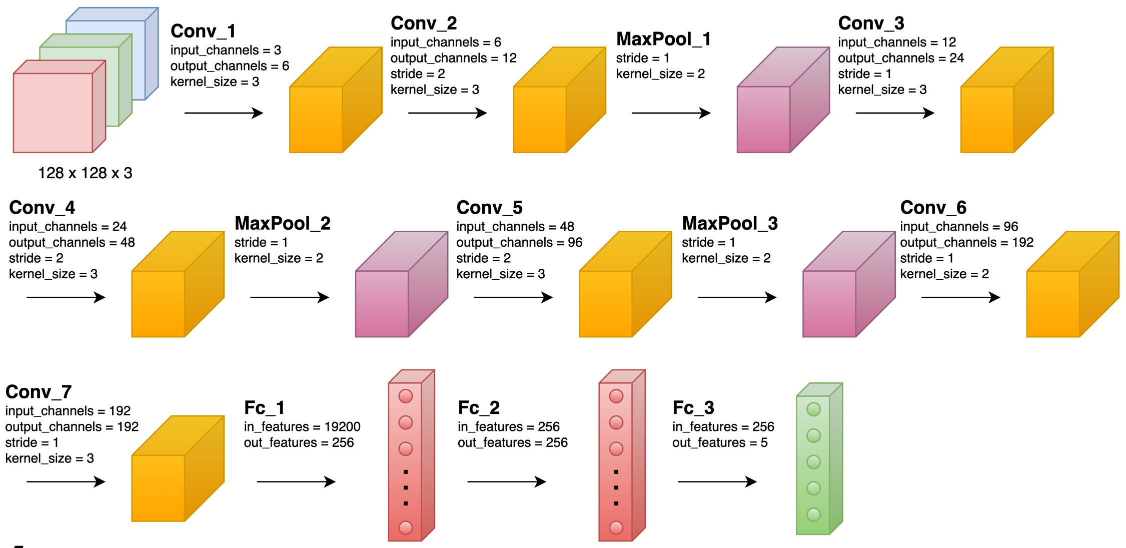


Figure 5
Base_CNN Model Architecture for Part One

Architecture, Part Two

In part 2 we use the Base_CNN_Part2 architecture as is shown in Figure 6, an updated version of the Base_CNN from part 1. We attempted several changes, including altering the strides, padding, and kernel sizes of our convolutional layers, as well as including additional dropout layers to prevent overfitting and improve robustness. Additionally, we fixed some minor issues with our previous network, such as not including non-linearity between our linear layers at the output.

Generally our model from part 1 was not too biased, even when including the new subdivisions of our data. Generally our precision and recall were within 5% of our accuracy, and this trend continued. Going into part 2 we aimed to improve overall performance, particularly for problem classes without compromising what was working well.

We found that there were many minor improvements to be made with regards to our convolutional layers. Previously we had not used different kernel sizes and did not include padding, we have done now. The dropout layers unfortunately were not very effective in improving test time performance. Our validation accuracy dropped significantly when they were included and as such our final model does not make use of them. To guard against overfitting we tracked our training and testing accuracies and ensured we did not continue training beyond the peak test-time accuracy.

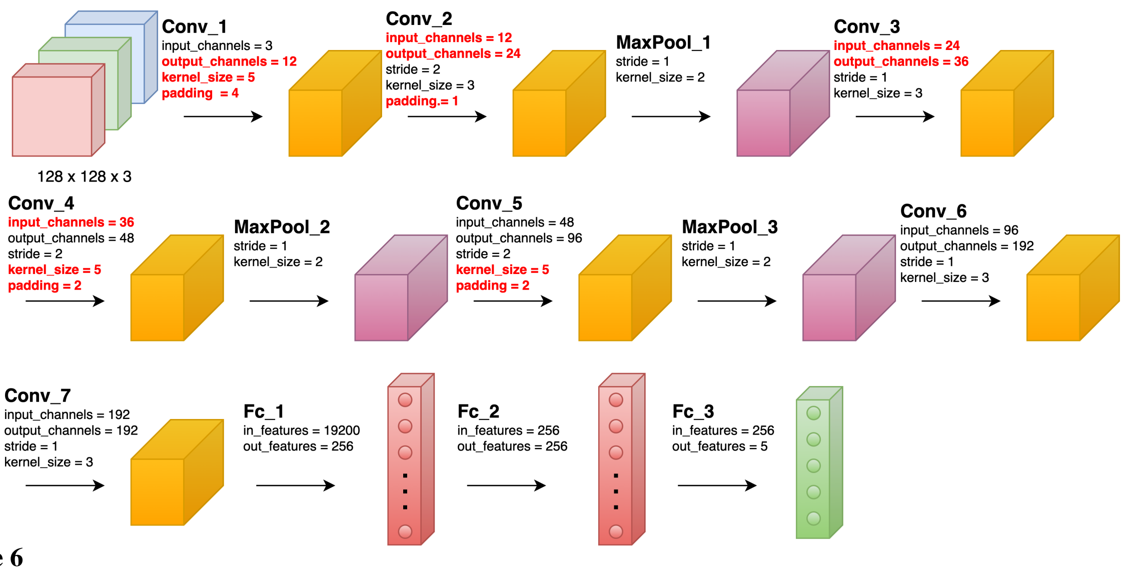


Figure 6
Base_CNN Model Architecture for Part Two

Training Procedure, Part Two

A key goal of part 2 of this project was the implementation of 10-fold cross validation during training and model construction. During part 1 of the project we had already implemented 5-fold cross validation and as such there has not been a huge change in our training procedure. We reduced the number of epochs to 30 from 50 as testing showed that gave similar results at a much faster pace. Otherwise, our procedure remains as described in our previous report (found below). To summarize, we split the data into a training/validation set and a test set. We used the training/validation to perform the cross validation, then trained our final model on the entire training/validation set before testing on the test set. The benefits of this should include improved prediction of test time performance, as well as better generalization to data outside the initial dataset.

We did include monitoring of model accuracy on the training and test datasets across training epochs, shown below. The goal here is to monitor for overfitting, which is a phenomenon where the model effectively memorizes training data and fails to generalize. This can occur if training is continued for too long and the model has sufficient parameters to perform the memorization. As such we make sure to stop the training once the accuracy stop improving as this both reduces training time, and helps prevent overfitting.

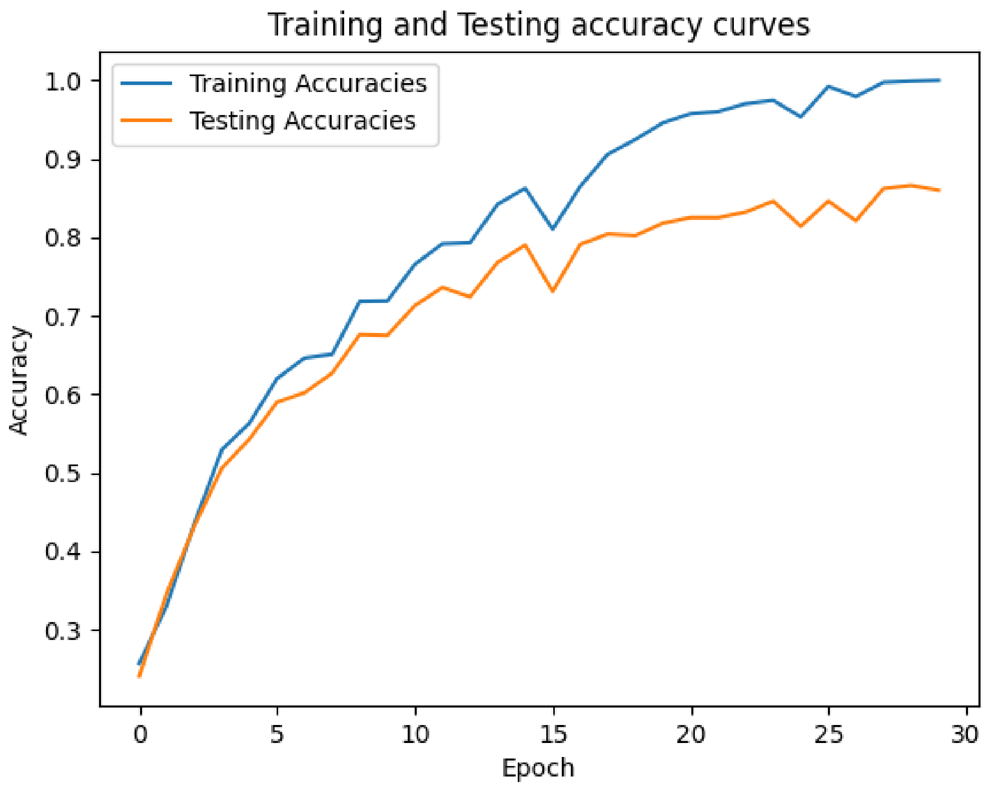


Figure 7
Training Progress across 30 Epochs for our Final Model in Part Two

Evaluation for Part One

Discussion

Given our results, tabulated below, we can conclude the following:

1. In all 3 versions of our models. Base_CNN and Less_Pooling have similar accuracy(from 83% to 85%), while Less_Conv has the least accuracy: 74.5 %, which means that the numbers of convolutional layers have the biggest impact on our models' performance. While accuracy isn't the only metric, these trends are reflected in the precision and recall as well. This follows as pooling layers do not add extract new features in the way convolutional layers can. Pooling layers can be considered to instead report on the maximum activation for a given feature in an area.

2. Generally our precision and recall don't stray too far from the overall accuracy, but it is clear that our model struggled more with certain classes than others. Surgical masks in particular suffer from low precision, meaning our model is likely to predict this class even when it's not correct.

3. The Less_Pooling model has better performance than the Less_Conv model on the cloth_mask class, with the precision of 84.5 %. And Less_Pooling model's performance (83.4 %) on the no_face_mask class is even better than Base_CNN model, which has 83.2 % as its precision on the no_face_mask class.

4. Our models have the best performance on the mask_worn_incorrectly class with the precision ranging from 82 % to 92 %. This may be due to a larger proportion of this class' data being comprised of augmented images.

Conclusion

In the second Part, in addition to generally improving our metrics, we will particularly aim to bring our surgical mask analysis up to par with the other classes. This may involve adding more layers or tuning the model's hyperparameters more, or perhaps increasing the number of sample images for this dataset. It is worth noting that our best performing categories has the smallest number of unaugmented images. This may be responsible for the higher success with this class as the images are more similar to each other than they should be. This would be a good issue to address in part 2 if we are capable. Otherwise we are generally pleased with the performance, while there is some variance between classes no class' performance is significantly worse than the others.

Table 3
Base CNN for Part One. Accuracy: 85.7 %

	Type	Precision	Recall	F-measure
0	Cloth Mask	86.2 %	81.5 %	83.8 %
1	No Face Mask	83.2 %	86.5 %	84.8 %
2	Surgical Mask	79.7 %	84.5 %	82.0 %
3	N95 Mask	87.0 %	83.5 %	85.2 %
4	Mask Worn Incorrectly	93.0 %	92.5 %	92.7 %

Table 4
Less pooling layers for Part One. Accuracy:83.4 %

	Type	Precision	Recall	F-measure
0	Cloth Mask	84.5 %	76.5 %	80.3 %
1	No Face Mask	83.4 %	88.0 %	85.6 %
2	Surgical Mask	76.0 %	79.0 %	77.5 %
3	N95 Mask	84.8 %	81.0 %	82.9 %
4	Mask Worn Incorrectly	88.5 %	92.5 %	90.5 %

Table 5
Less convolutional layers for Part One. Accuracy:74.5 %

	Type	Precision	Recall	F-measure
0	Cloth Mask	73.2 %	75.0 %	74.1 %
1	No Face Mask	80.7 %	81.5 %	81.1 %
2	Surgical Mask	65.3 %	66.0 %	65.7 %
3	N95 Mask	71.2 %	68.0 %	69.6 %
4	Mask Worn Incorrectly	82.0 %	82.0 %	82.0 %

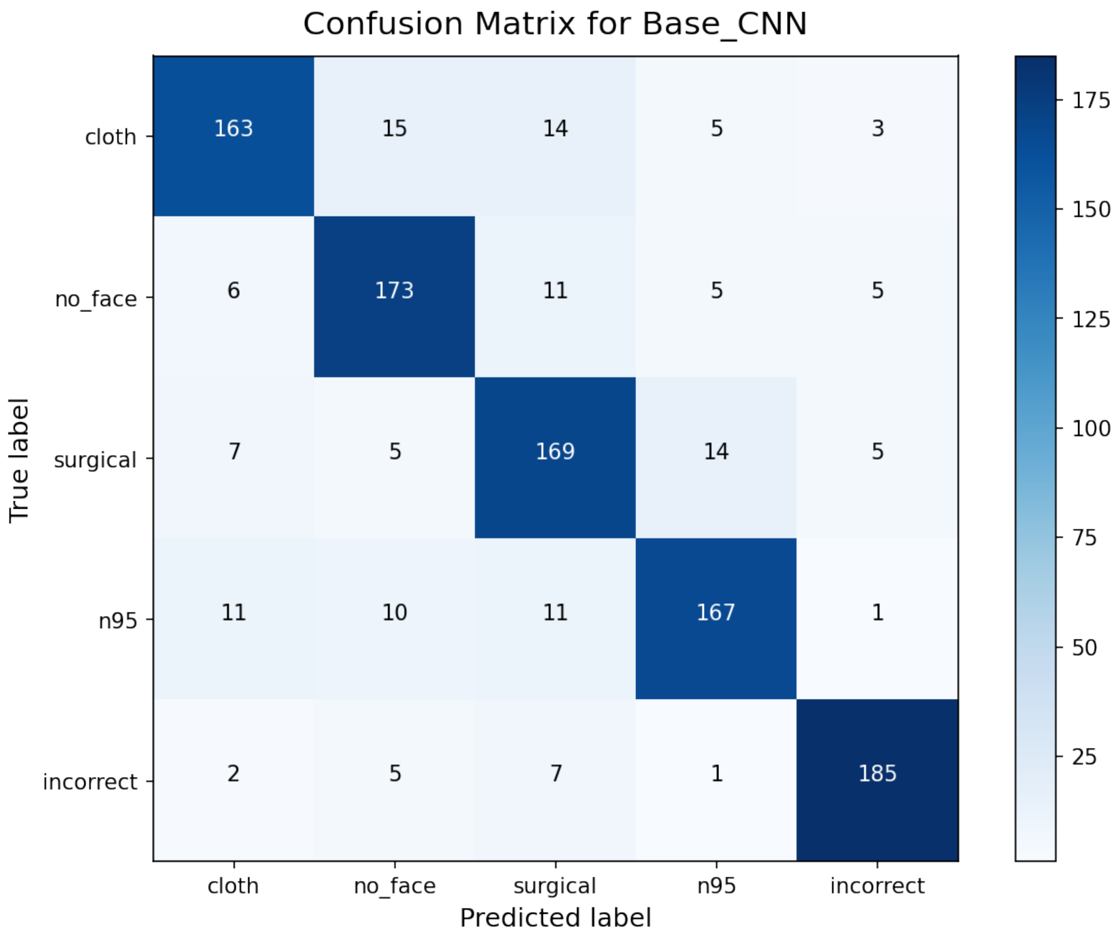


Figure 8
Confusion Matrix of Base CNN for Part One

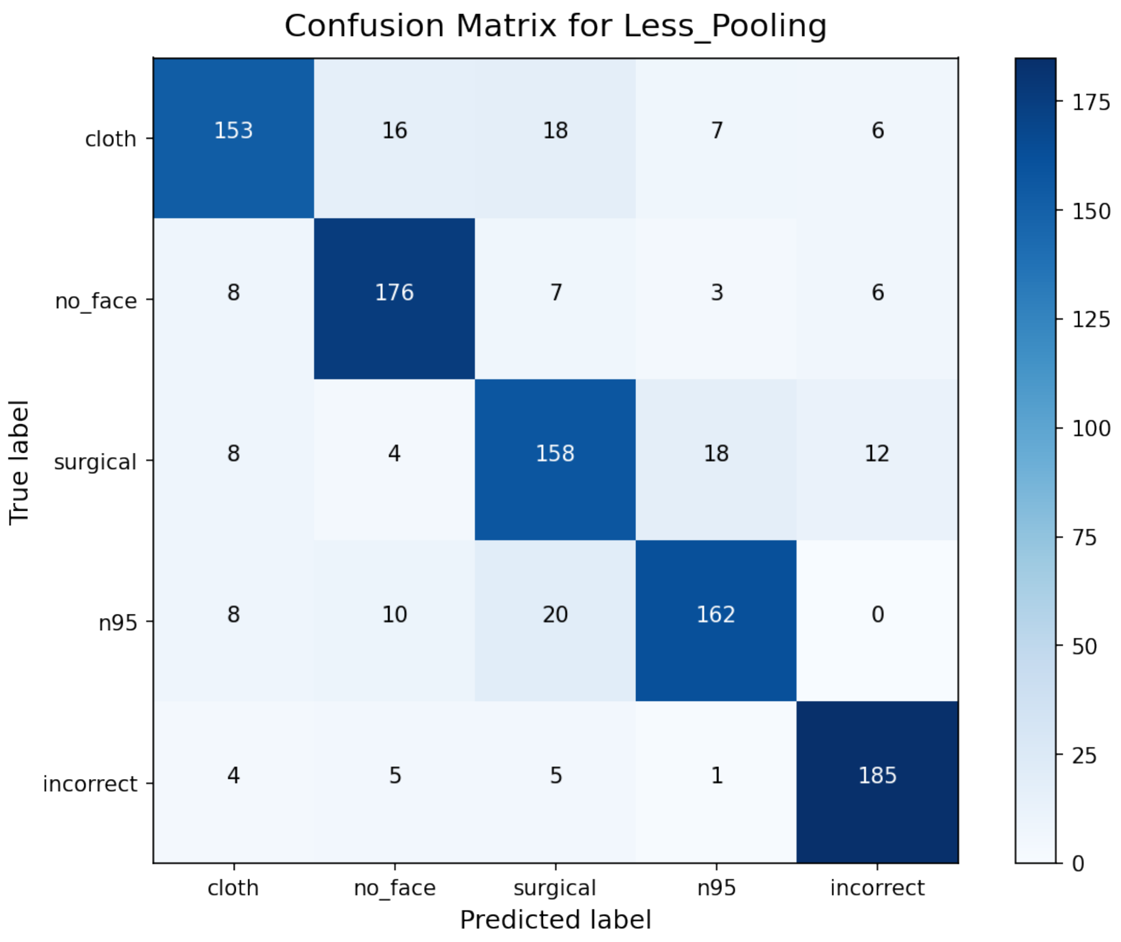


Figure 9
Confusion Matrix of Less Pooling layers for Part One

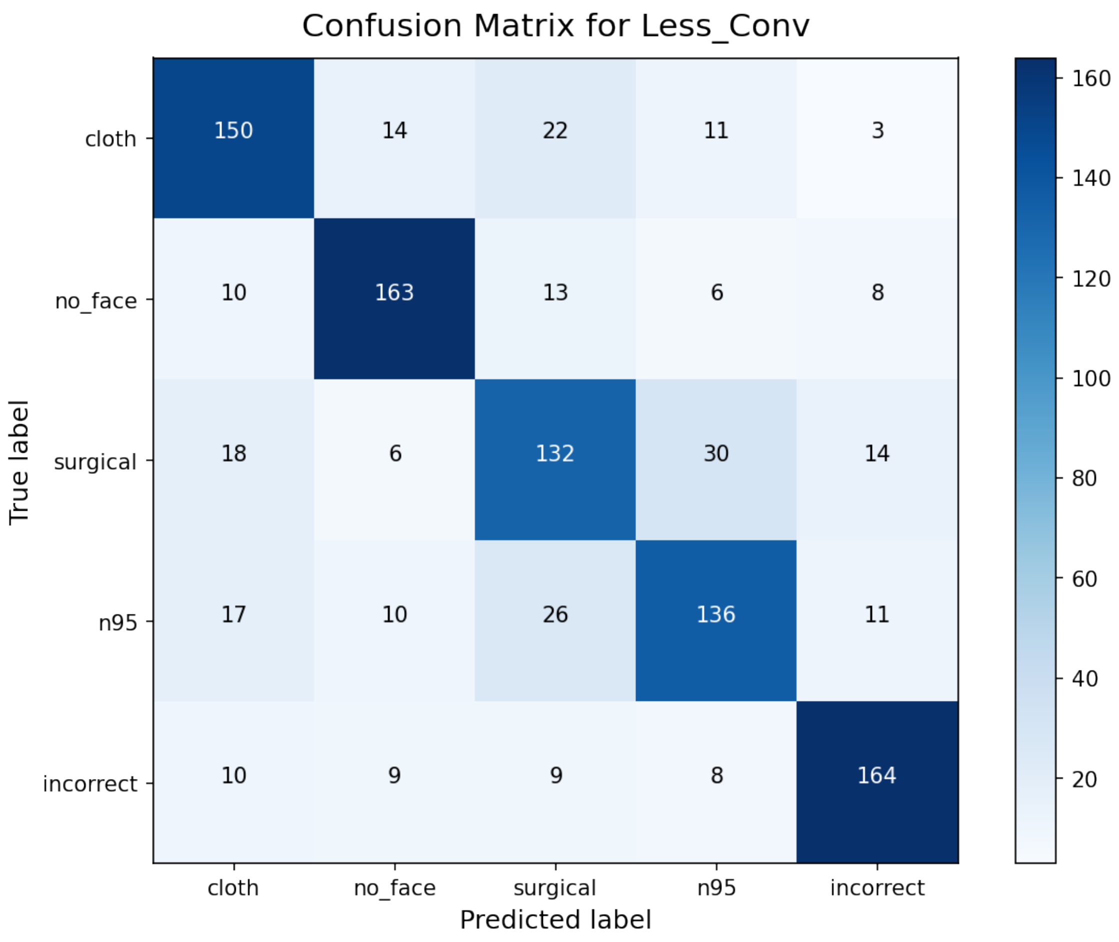


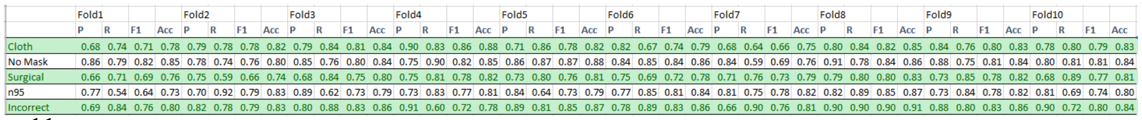
Figure 10
Confusion Matrix of Less Convolutional layers for Part One

K-fold Cross-Validation for Part Two

We have implemented the 10-fold cross-validation on our dataset. Below are the results for the models used in Parts 1 and 2. We discuss the benefits of K-fold cross validation in our training section and so will focus mainly on our observations here.

Conclusion

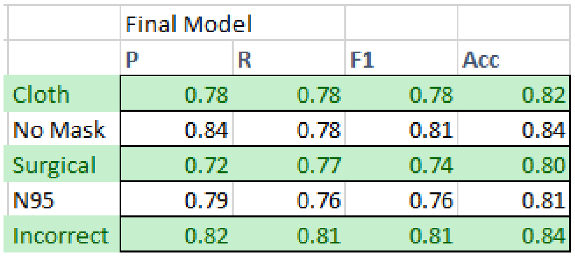
The results during cross validation were slightly worse than those obtained from the part 1 net. This is concerning as our net in part 2 performs better during testing overall than the net in part 1. Likely this indicates some sort of issue with our process, potentially related to how we augmented our data differently for part 2. Overall we see a relatively large improvement between the cross validation and the final results, which is a potential cause for concern. Given additional time we would investigate and attempt to correct what may be an indication of improper data augmentation or other mistakes we may have made. The part 1 CV results are much more indicative of the network's final performance than part 2's. This is a better showcase of the value of cross validation as we get a more meaningful predictor of model performance than what we see in part 2. Further discussion of the value of k-fold validation is present in the training section.



	Fold1				Fold2				Fold3				Fold4				Fold5			
	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc
Cloth	0.68	0.74	0.71	0.78	0.79	0.78	0.78	0.82	0.79	0.84	0.81	0.84	0.90	0.83	0.86	0.88	0.71	0.86	0.78	0.82
No Mask	0.86	0.79	0.82	0.85	0.78	0.74	0.76	0.80	0.85	0.76	0.80	0.84	0.75	0.90	0.82	0.85	0.86	0.87	0.87	0.88
Surgical	0.66	0.71	0.69	0.76	0.75	0.59	0.66	0.74	0.68	0.84	0.75	0.80	0.75	0.81	0.78	0.82	0.73	0.80	0.76	0.81
n95	0.77	0.54	0.64	0.73	0.70	0.92	0.79	0.83	0.89	0.62	0.73	0.79	0.73	0.83	0.77	0.81	0.84	0.64	0.73	0.79
Incorrect	0.69	0.84	0.76	0.80	0.82	0.78	0.79	0.83	0.80	0.88	0.83	0.86	0.91	0.60	0.72	0.78	0.89	0.81	0.85	0.87

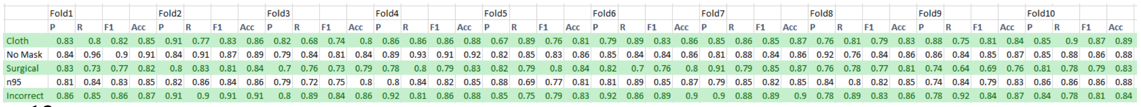
	Fold6				Fold7				Fold8				Fold9				Fold10			
	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc
Cloth	0.82	0.67	0.74	0.79	0.68	0.64	0.66	0.75	0.80	0.84	0.82	0.85	0.84	0.76	0.80	0.83	0.78	0.80	0.79	0.83
No Mask	0.84	0.85	0.84	0.86	0.84	0.59	0.69	0.76	0.91	0.78	0.84	0.86	0.88	0.75	0.81	0.84	0.80	0.81	0.81	0.84
Surgical	0.75	0.69	0.72	0.78	0.71	0.76	0.73	0.79	0.79	0.80	0.80	0.83	0.73	0.85	0.78	0.82	0.68	0.89	0.77	0.81
n95	0.77	0.85	0.81	0.84	0.81	0.75	0.78	0.82	0.82	0.89	0.85	0.87	0.73	0.84	0.78	0.82	0.81	0.69	0.74	0.80
Incorrect	0.78	0.89	0.83	0.86	0.66	0.90	0.76	0.81	0.90	0.90	0.90	0.91	0.88	0.80	0.83	0.86	0.90	0.72	0.80	0.84

Figure 11
Analysis of Each Fold in 10-Fold CV for Part 2



	Final Model			
	P	R	F1	Acc
Cloth	0.78	0.78	0.78	0.82
No Mask	0.84	0.78	0.81	0.84
Surgical	0.72	0.77	0.74	0.80
N95	0.79	0.76	0.76	0.81
Incorrect	0.82	0.81	0.81	0.84

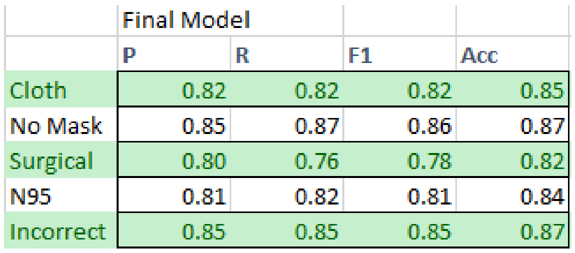
Figure 12
Average of All Part 2 Folds



	Fold1				Fold2				Fold3				Fold4				Fold5			
	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc
Cloth	0.83	0.8	0.82	0.85	0.91	0.77	0.83	0.86	0.82	0.68	0.74	0.8	0.86	0.86	0.86	0.88	0.67	0.89	0.76	0.81
No Mask	0.84	0.96	0.9	0.91	0.84	0.91	0.87	0.89	0.79	0.84	0.81	0.84	0.89	0.93	0.91	0.92	0.82	0.85	0.83	0.86
Surgical	0.83	0.73	0.77	0.82	0.8	0.83	0.81	0.84	0.7	0.76	0.73	0.79	0.78	0.8	0.79	0.83	0.82	0.79	0.8	0.84
n95	0.81	0.84	0.83	0.85	0.82	0.86	0.84	0.86	0.79	0.72	0.75	0.8	0.8	0.84	0.82	0.85	0.88	0.69	0.77	0.81
Incorrect	0.86	0.85	0.86	0.87	0.91	0.9	0.91	0.91	0.8	0.89	0.84	0.86	0.92	0.81	0.86	0.88	0.85	0.75	0.79	0.83

	Fold6				Fold7				Fold8				Fold9				Fold10			
	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc
Cloth	0.79	0.89	0.83	0.86	0.85	0.86	0.85	0.87	0.76	0.81	0.79	0.83	0.88	0.75	0.81	0.84	0.85	0.9	0.87	0.89
No Mask	0.85	0.84	0.84	0.86	0.81	0.88	0.84	0.86	0.92	0.76	0.84	0.86	0.86	0.84	0.85	0.87	0.85	0.88	0.86	0.88
Surgical	0.82	0.7	0.76	0.8	0.91	0.79	0.85	0.87	0.76	0.78	0.77	0.81	0.74	0.64	0.69	0.76	0.81	0.78	0.79	0.83
n95	0.81	0.89	0.85	0.87	0.79	0.85	0.82	0.85	0.84	0.8	0.82	0.85	0.74	0.84	0.79	0.83	0.86	0.86	0.86	0.88
Incorrect	0.92	0.86	0.89	0.9	0.9	0.88	0.89	0.9	0.78	0.89	0.83	0.86	0.78	0.92	0.84	0.87	0.84	0.78	0.81	0.84

Figure 13
Analysis of Each Fold in 10-Fold CV for Part 1

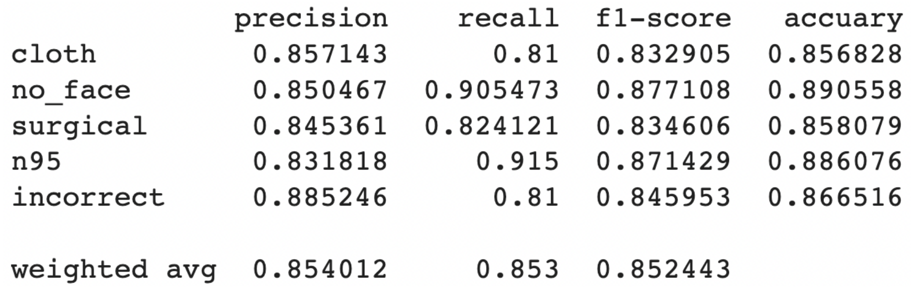


	Final Model			
	P	R	F1	Acc
Cloth	0.82	0.82	0.82	0.85
No Mask	0.85	0.87	0.86	0.87
Surgical	0.80	0.76	0.78	0.82
N95	0.81	0.82	0.81	0.84
Incorrect	0.85	0.85	0.85	0.87

Figure 14
Average of All Folds for Part 1

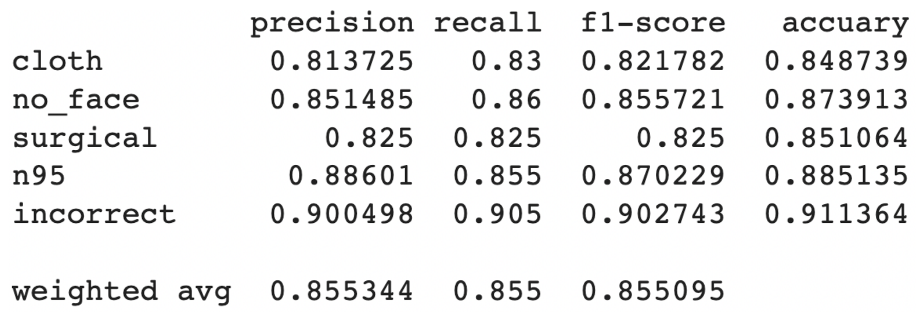
Bias for Part Two

In our project's Part 2, we have decided to analyze the race and the gender as our dataset's attributes. For the race, we have chosen 4 races(Caucasian, Black, Asian and Arab) and divided them into 2 subclasses (Caucasian/Asian, African/Arab) primarily for ease of use. For the gender, we divided our dataset into man and woman. Below are our performance evaluation metrics and confusion matrices for each of our subclasses, and one confusion matrix for our whole dataset.



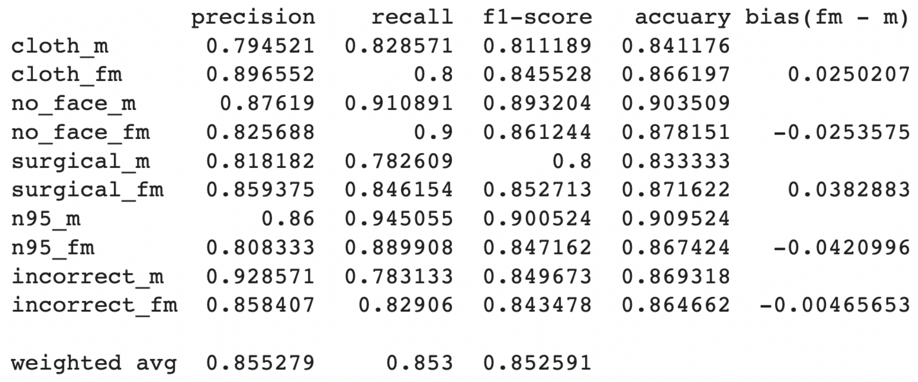
```
              precision    recall  f1-score   accuary
cloth          0.857143      0.81  0.832905  0.856828
no_face        0.850467  0.905473  0.877108  0.890558
surgical       0.845361  0.824121  0.834606  0.858079
n95            0.831818     0.915  0.871429  0.886076
incorrect      0.885246      0.81  0.845953  0.866516

weighted avg   0.854012     0.853  0.852443
```

Figure 15
Mask Types Performance Scores before Dataset Rebalance for Bias



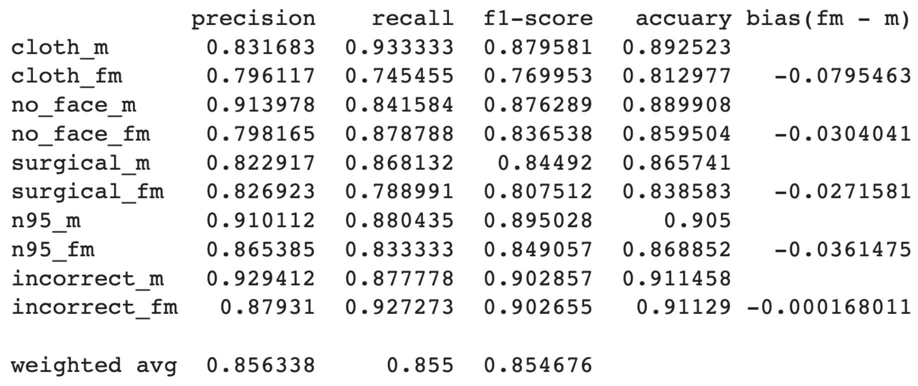
	precision	recall	f1-score	accuary
cloth	0.813725	0.83	0.821782	0.848739
no_face	0.851485	0.86	0.855721	0.873913
surgical	0.825	0.825	0.825	0.851064
n95	0.88601	0.855	0.870229	0.885135
incorrect	0.900498	0.905	0.902743	0.911364
weighted avg	0.855344	0.855	0.855095	

Figure 16
Mask Types Performance Scores after Dataset Rebalance for Bias



	precision	recall	f1-score	accuary	bias(fm - m)
cloth_m	0.794521	0.828571	0.811189	0.841176	
cloth_fm	0.896552	0.8	0.845528	0.866197	0.0250207
no_face_m	0.87619	0.910891	0.893204	0.903509	
no_face_fm	0.825688	0.9	0.861244	0.878151	-0.0253575
surgical_m	0.818182	0.782609	0.8	0.833333	
surgical_fm	0.859375	0.846154	0.852713	0.871622	0.0382883
n95_m	0.86	0.945055	0.900524	0.909524	
n95_fm	0.808333	0.889908	0.847162	0.867424	-0.0420996
incorrect_m	0.928571	0.783133	0.849673	0.869318	
incorrect_fm	0.858407	0.82906	0.843478	0.864662	-0.00465653
weighted avg	0.855279	0.853	0.852591		

Figure 17
Mask-Gender Types Performance Scores before Dataset Rebalance for Bias



	precision	recall	f1-score	accuary	bias(fm - m)
cloth_m	0.831683	0.933333	0.879581	0.892523	
cloth_fm	0.796117	0.745455	0.769953	0.812977	-0.0795463
no_face_m	0.913978	0.841584	0.876289	0.889908	
no_face_fm	0.798165	0.878788	0.836538	0.859504	-0.0304041
surgical_m	0.822917	0.868132	0.84492	0.865741	
surgical_fm	0.826923	0.788991	0.807512	0.838583	-0.0271581
n95_m	0.910112	0.880435	0.895028	0.905	
n95_fm	0.865385	0.833333	0.849057	0.868852	-0.0361475
incorrect_m	0.929412	0.877778	0.902857	0.911458	
incorrect_fm	0.87931	0.927273	0.902655	0.91129	-0.000168011
weighted avg	0.856338	0.855	0.854676		

Figure 18
Mask-Gender Types Performance Scores after Dataset Rebalance for Bias

```
               precision    recall  f1-score   accuary bias(afar - caas)
cloth_caas      0.869565  0.805369  0.836237  0.859281
cloth_afar      0.823529  0.823529  0.823529      0.85       -0.00928144
no_face_caas    0.840909  0.909836  0.874016  0.888112
no_face_afar    0.865854  0.898734  0.881988  0.894444        0.00633256
surgical_caas   0.834437  0.818182   0.82623  0.851955
surgical_afar   0.883721  0.844444  0.863636      0.88         0.0280447
n95_caas         0.82659  0.916667  0.869301  0.884409
n95_afar        0.851064  0.909091  0.879121  0.892157        0.00774826
incorrect_caas  0.898649  0.826087  0.860841  0.877841
incorrect_afar  0.828571   0.74359  0.783784  0.822222        -0.0556187

weighted avg    0.854221     0.853  0.852465
```

Figure 19

Mask-Race Types Performance Scores before Dataset Rebalance for Bias

```
               precision    recall  f1-score   accuary bias(afar - caas)
cloth_caas         0.775     0.775     0.775  0.816327
cloth_afar      0.869048    0.9125  0.890244  0.901099         0.0847724
no_face_caas    0.781513  0.885714  0.830357  0.854962
no_face_afar    0.951807  0.831579   0.88764   0.89899         0.0440281
surgical_caas       0.75      0.75      0.75       0.8
surgical_afar     0.9375    0.9375    0.9375  0.941176          0.141176
n95_caas        0.846847  0.783333  0.813853  0.843066
n95_afar        0.939024    0.9625  0.950617  0.952941          0.109875
incorrect_caas  0.911111  0.878571  0.894545  0.904605
incorrect_afar  0.878788  0.966667  0.920635  0.926471         0.0218653

weighted avg    0.857031     0.855  0.854919
```

Figure 20

Mask-Race Types Performance Scores after Dataset Rebalance for Bias

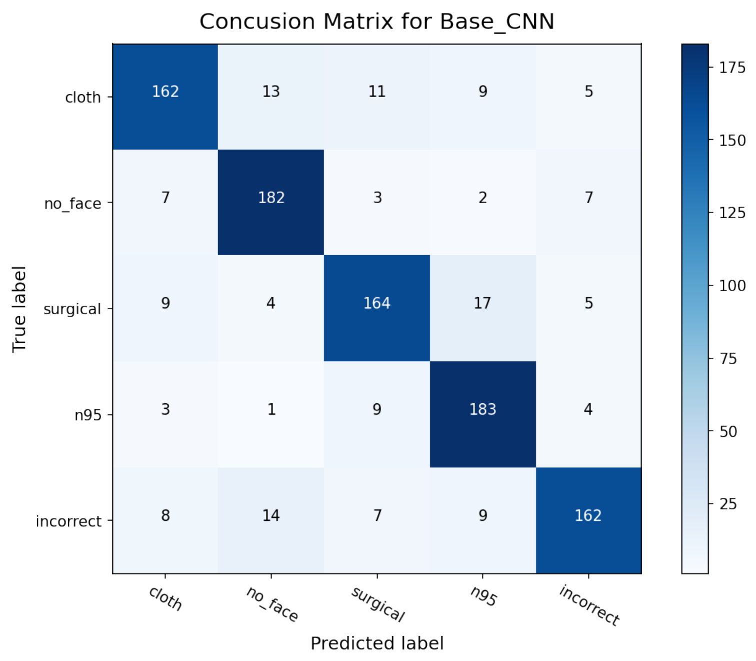


Figure 21

Confusion Matrix of Base CNN for Part One

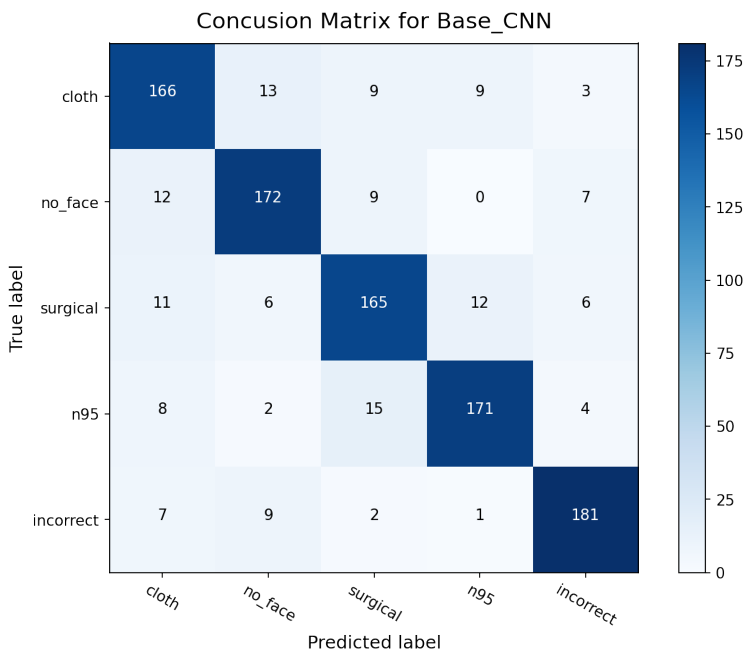


Figure 22
Confusion Matrix of Base CNN for Part Two

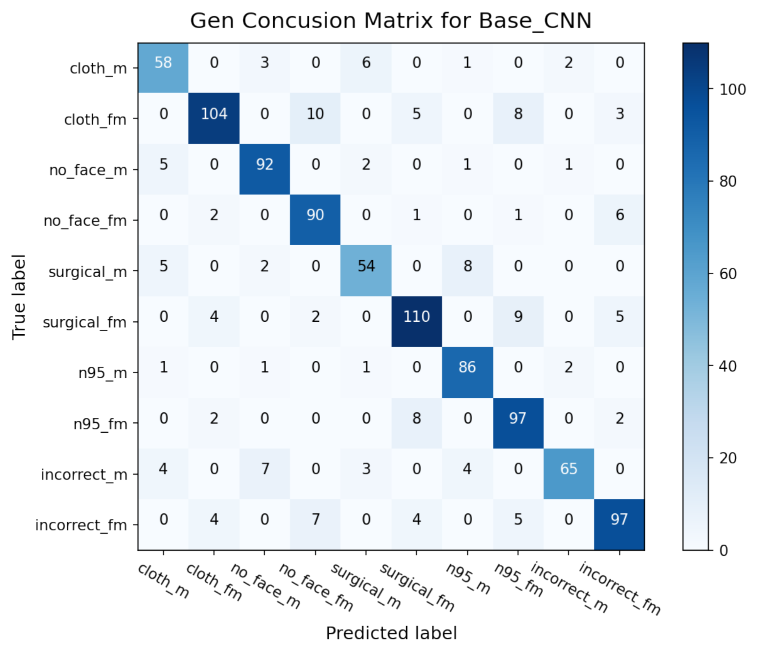


Figure 23
Gender Confusion Matrix of Base CNN for Part One

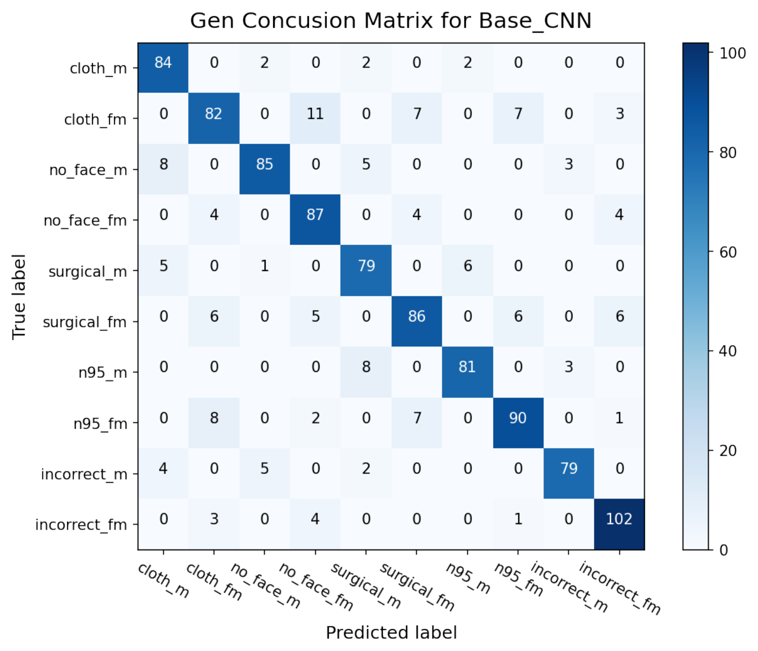


Figure 24

Gender Confusion Matrix of Base CNN for Part Two

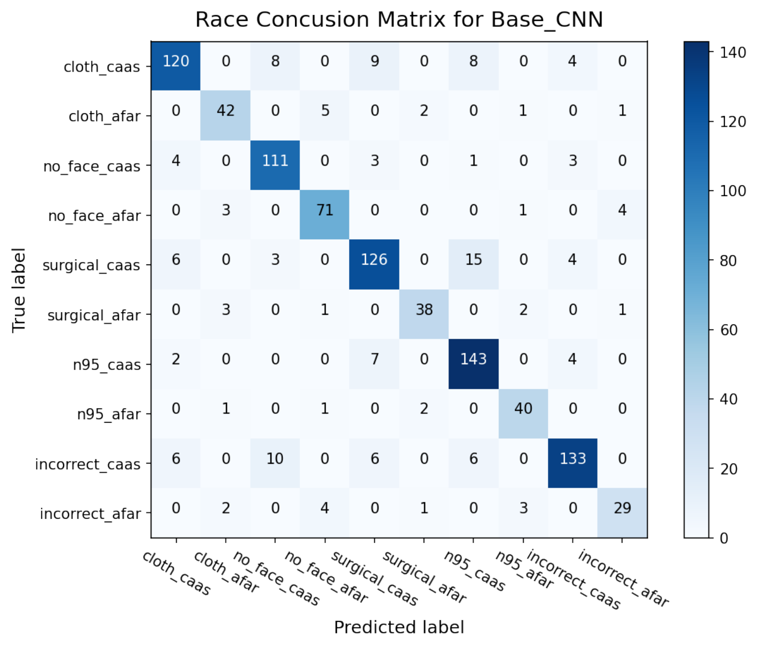


Figure 25

Race Confusion Matrix of Base CNN for Part One

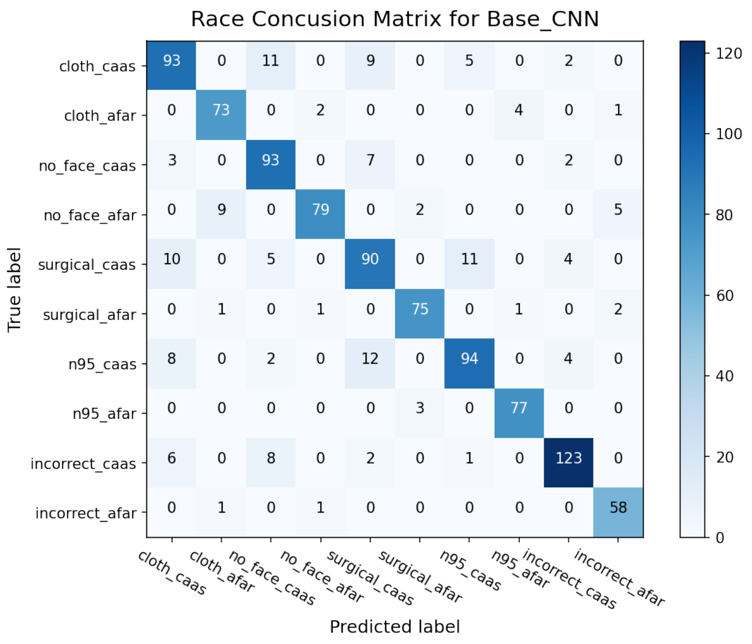


Figure 26

Race Confusion Matrix of Base CNN for Part Two

Conclusion

One major potential source of bias in a model comes from the data used for training. We found that our model has different training effects on different subclasses (like gender and race) of dataset. One major way to address bias is to change the underlying data the model is trained on as without sufficient examples of certain subclasses, the model will never be able to predict them.

First, we divided every subclass of our dataset according to the race and the gender. Then we used our model to test our divided subclasses and recorded the result. After analysing we found that we had biases in the results of our model in Part 1. The bias percentage in subclass gender and race is small, but we still want to eliminate them as can be observed in Figure 15 to Figure 17. Hence we rebalance our dataset according to the data size of the sub class. But the bias are higher than before as can be observed in Figure 18 to Figure 20.

In comparison with our model in Part 1 and Part 2, we can see that our new model in Part 2 has slightly better performance on some classes, but generally performs similarly to our part 1 network. Although our first model has done very well on the testing set, our model in Part 2 has made some improvement successfully. For example, our model in Part 2 made the improvement of 5% in the masks worn incorrectly subclass in total, of 0.39% in the masks worn incorrectly subclass divided by the gender and of 3% in the masks worn incorrectly subclass divided by the race.

References

Bergstra, J., & Bengio, Y. (2012, Feb). *Random search for hyper-parameter optimization.* Journal of Machine Learning Research. Retrieved from https://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

COFFEE124. (2022, 06). *Face mask.* Retrieved 2022-06-04, from https://www.kaggle.com/datasets/coffee124/facemaskn95

Intelligence, W. (2020). *Face mask detection dataset.* Retrieved 2022-06-01, from https://www.kaggle.com/datasets/wobotintelligence/face-mask-detection-dataset

Nayak, S. (2021, May). *Understanding alexnet.* Retrieved from https://learnopencv.com/understanding-alexnet/

Ye, A. (2020, Sep). *If rectified linear units are linear, how do they add nonlinearity?* Towards Data Science. Retrieved from https://towardsdatascience.com/if-rectified-linear-units-are-linear-how-do-they-add-nonlinearity-40247d3e4792