AI Face Mask Detector Phase 1 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 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.

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

Dataset

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

	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()

Convolutional Neural Network Architecture

The primary model we used was the Base_CNN, shown Figure 1. 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

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.

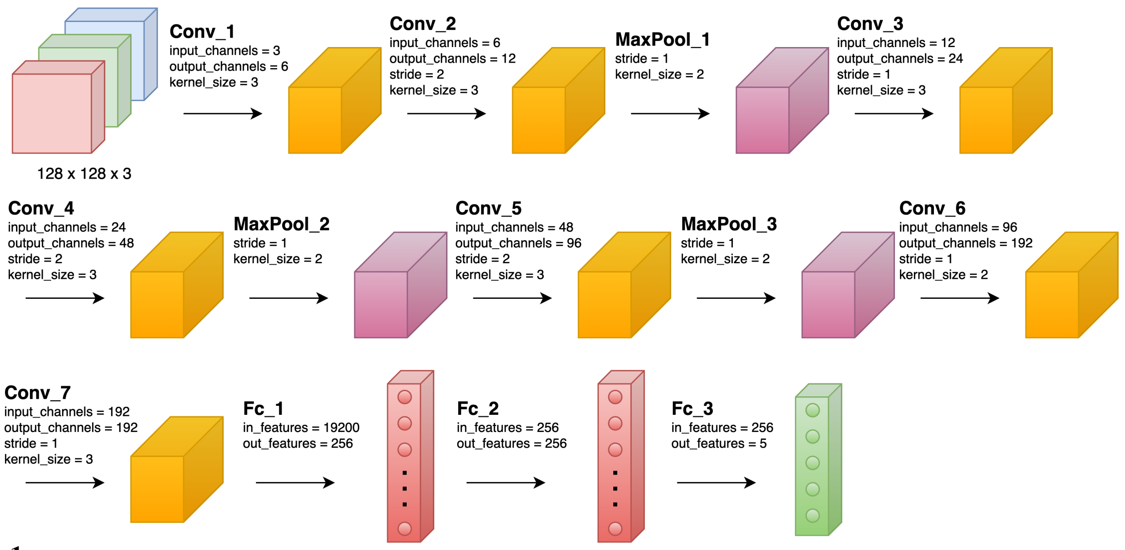

Figure 1

Base_CNN Model Architecture

Evaluation

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 phase, 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 2
Base CNN. 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 3
Less pooling layers. 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 4
Less convolutional layers. 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 %

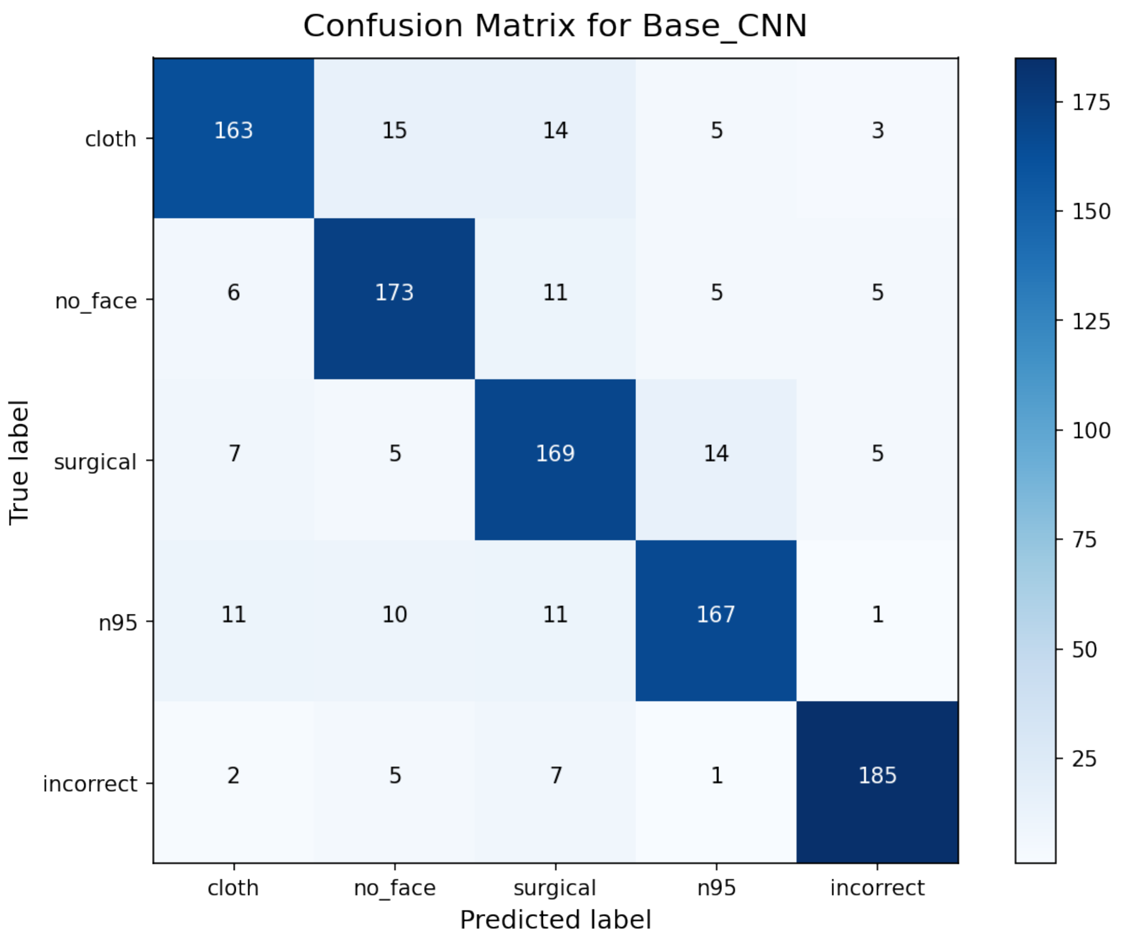

Figure 2
Confusion Matrix of Base CNN

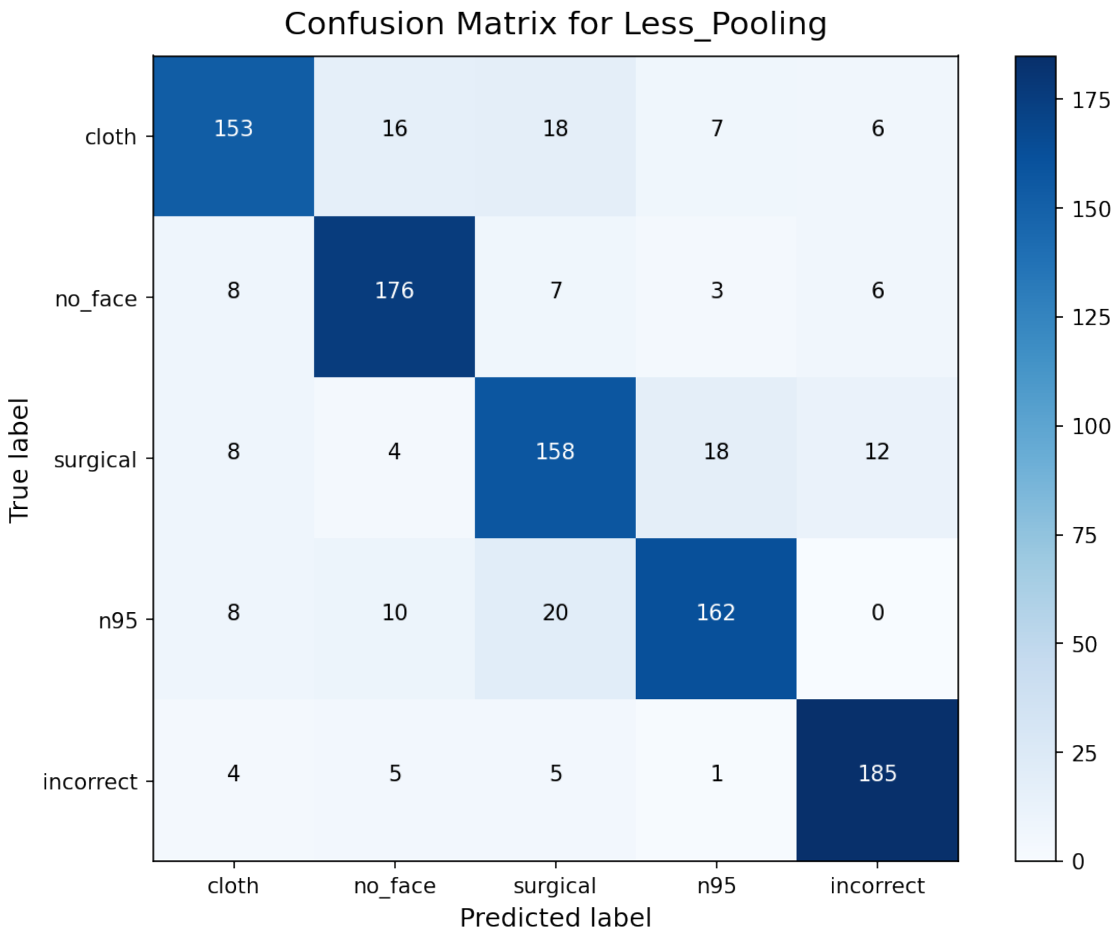

Figure 3
Confusion Matrix of Less Pooling layers

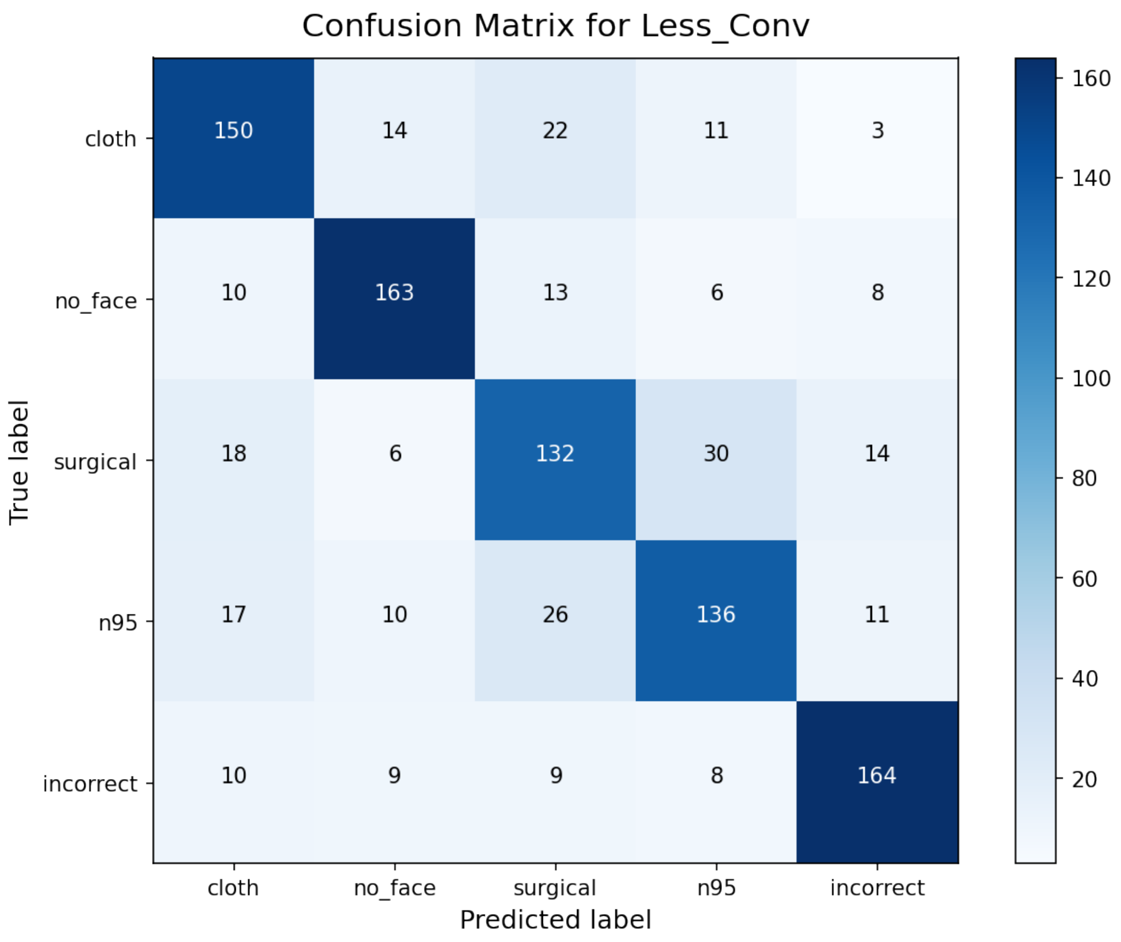

Figure 4
Confusion Matrix of Less convolutional layers

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