AI Mask Detector:
Project Assignment - Part Two

Concordia University
June 22rd 2022

Prof. Rene Witte

Bozidar Leshev (40105294)
Mila Roisin (2957554)
Deniz Dinchdonmez (40128249)

ROLES ASSIGNMENT

(I) Data Specialist: Deniz + Bozhidar

- responsible for creating, pre-processing, loading & analyzing the datasets;

(II) Training Specialist: Deniz + Bozhidar

- responsible for setting up and training the CNN;

(III) Evaluation Specialist: Deniz + Bozhidar + Mila

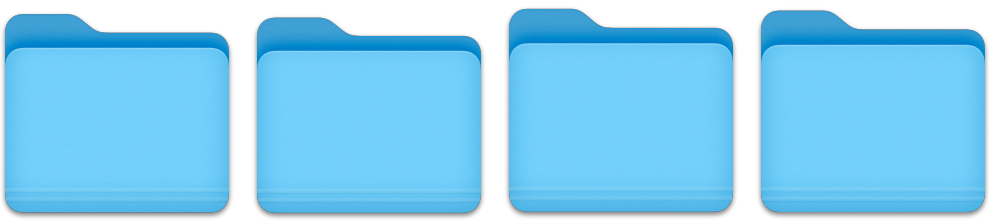
- responsible for analyzing, evaluating, and applying the generated model;

(IV) Compliance Specialist: Bozihar + Mila

- responsible for the overall planning, scope verification, and compliance (in part 2 of the project, this member will also be responsible for the new tasks mentioned below.)

Dataset

Describe how you built your dataset and where you collected images (provide details on each image's source in a file). Provide statistics on the size and structure of your dataset, including, how many images you have in each class and what the resolution for each class is. Describe your pre-processing strategy (e.g., size normalization).



The initial dataset of 853 images were obtained from Kaggle and were initially separated into three distinctions: with mask, without mask and lastly, mask worn incorrectly [1]. The second dataset containing 6024 images focused on two classes: with mask and without mask. [2]

After compiling the two datasets together, we categorized our images into four different file categories based on physical features on the type of mask worn: without mask, with cloth masks, surgical masks and n95 masks [3], this initial dataset was small but it was used to create a model that can be trained. Later on, our final dataset was constructed only from the second dataset by manually placing images into correct categories described before. Majority of the images are in color and in JPG format [3]. Before training the dataset, preprocessing the data via standardization is a necessary step [2]. We resized the dataset and normalized all images to have all values between 0 and 1.

For reference, the final dataset ('classified') can be obtained here.

General Info:

	no_mask	cloth_mask	surgical_mask	n95	Average
Size (MB)	224	179	208	179	197.5
Items	400	400	400	400	400
File Format	JPEG	JPEG, PNG	JPEG, PNG	JPEG, PNG	N/A
Avg Resolution	1280x800	1024×576	1331×790	1024×576	N/A

Sample Images

Fig.1 Sample Images from Dataset (counter clockwise)
(a) no_mask, (b) cloth_mask, (c) surgical_mask, (d) n95

CNN Architecture

The training of the convolutional neural network was done with a batch size of 32 and was iterated over 25 epochs. The original images were converted to a 256x256 resolution to alleviate the RAM limitation. Due to the large amount of data and large tensors used, the training was done through the CUDA cores of an NVIDIA Graphical Processing Unit (GPU). This

would be reducing the training time enormously. Please see the model architecture schematic below for a visual representation of our design.

The model architecture

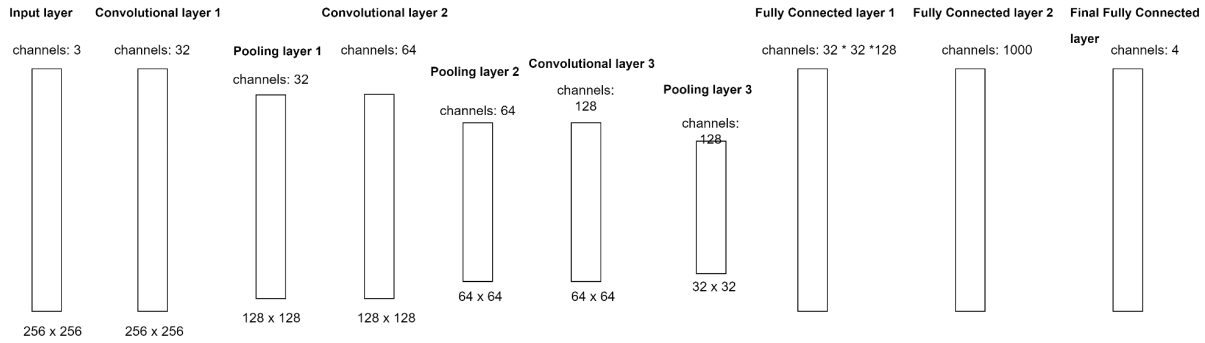


Fig.2 Design schema of the CNN architecture

The basic idea is to decrease the dimension of the input image after each layer and increase the number of channels (features) [5]. Finally, the encoded representation is reshaped into a vector and passed through three fully connected layers, the final layer being the output layer [6].

The parameters convolution layers are set to keep the shape of the output image nearly equal to that of the input [5][6]. The max-pooling layers decrease the size to roughly half the original shape [5]. The activation function ReLU was used [6]. Batch normalization and dropout of 0.1 were used to make the model robust.

The detailed pipeline is as follows:

Convolution 2d(in_channels=3, out_channels=32, kernel_size=3)

Batch Normalization 2d

LeakyReLU

Max Pooling 2d(kernel_size = 2, stride = 2)

Dropout

Convolution 2d(in_channels=32, out_channels=64, kernel_size=3)

Batch Normalization 2d

LeakyReLU

Max Pooling 2d(kernel_size = 2, stride = 2)

Dropout

Convolution 2d(in_channels=64, out_channels=128, kernel_size=3)

Batch Normalization 2d

LeakyReLU

Max Pooling 2d(kernel_size = 2, stride = 2)

Dropout

Flatten

Fully Connected Layer
ReLU
Dropout

Fully Connected Layer
ReLU
Dropout

Final Fully Connected Layer

Part One: Evaluation

As seen in this figure, the accuracy graph indicates that our model is noisy. To correct that, we could increase the batch_size for the training data from 32 to 64 in the future submission of part two.

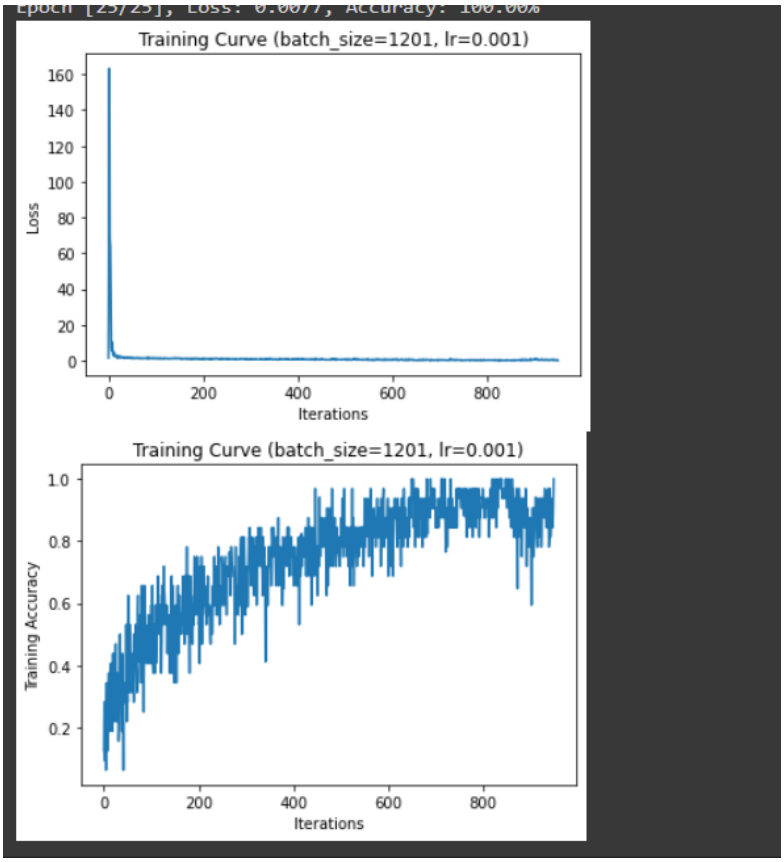


Fig.3 Accuracy graph depicting training curve

1. Precision, recall, F1-measure, accuracy on your testing data (across all four classes)

The weighted Precision, recall, F1-measure & accuracy scores across all classes is as follows:

Precision	Recall	f1-score	accuracy
0.4697	0.4705	0.4460	0.4705

For clarity, the Precision, recall, F1-measure, support for each respective class is added below.

Class	Precision	Recall	f1-score	support
Cloth	0.88	0.80	0.84	90
N95	0.78	0.88	0.83	104
Surgical	0.88	0.85	0.87	114
No Mask	0.84	0.82	0.83	96
Accuracy			0.84	404

	precision	recall	f1-score	support
0	0.88	0.80	0.84	90
1	0.78	0.88	0.83	104
2	0.88	0.85	0.87	114
3	0.84	0.82	0.83	96
accuracy			0.84	404
macro avg	0.84	0.84	0.84	404
weighted avg	0.84	0.84	0.84	404

```
Accuracy of the network on the 404 train images: 84.15841584158416 %
```

Fig. 4 (a) Screen capture of precision, recall, f1 score and support,
(b) Accuracy calculator depicting training images

2. Confusion matrix for the four classes on testing data

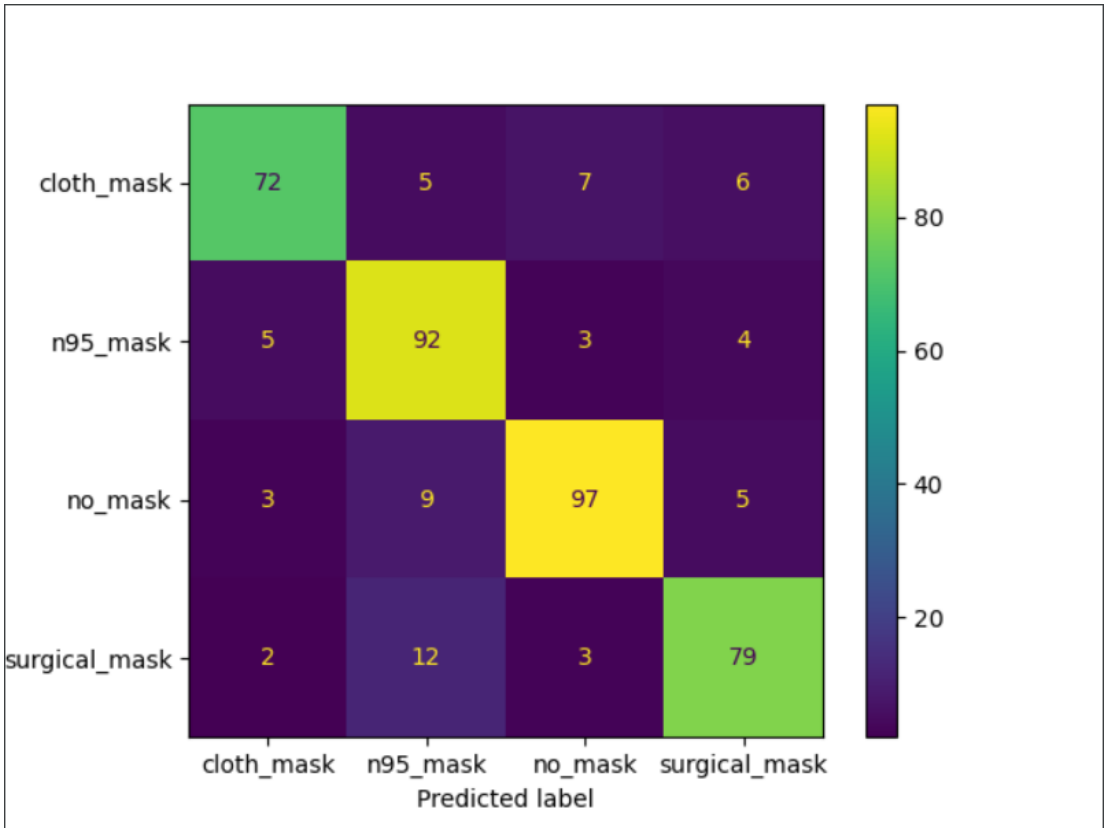


Figure 5. Confusion matrix (Original Model V 2.2) with specified labels

Part Two: Evaluation

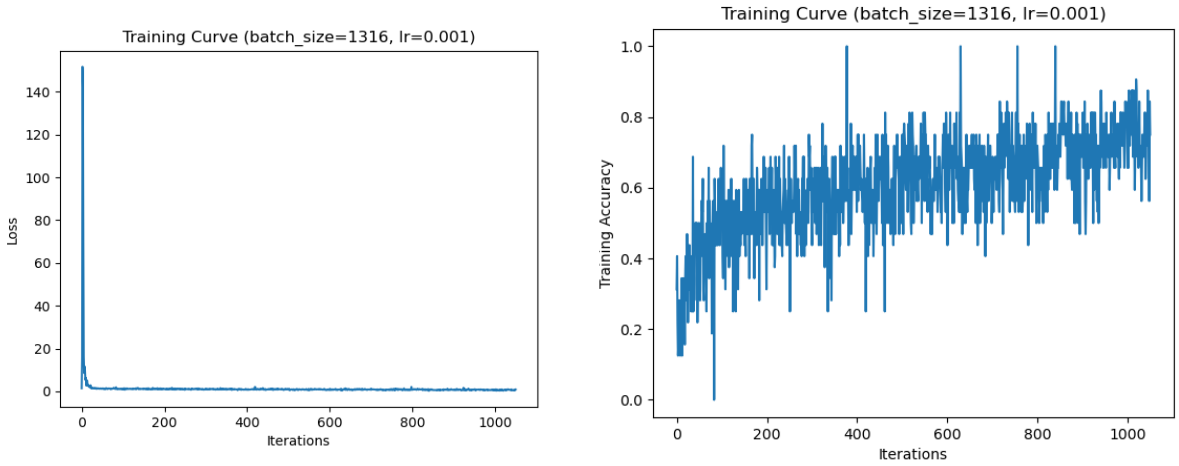


Figure 6. (a) Training Curve depicting loss graph,
(b) Training Curve depicting accuracy graph

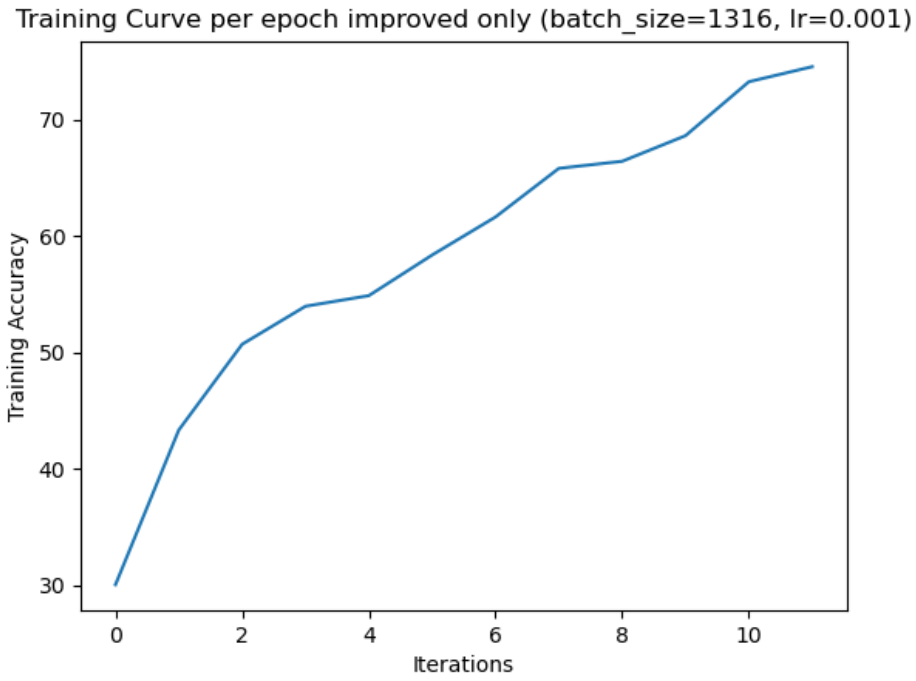


Figure 7 Graph depicting Training Accuracy improvement per epoch

```
--------------------------------
FOLD 0
--------------------------------
Starting K-fold Cross validation of the loaded model
{'0': {'precision': 0.9761904761904762, 'recall': 0.8367346938775511, 'f1-score': 0.9010989010989
0.8461538461538461, 'support': 36}, '2': {'precision': 0.8292682926829268, 'recall': 0.8947368421
0.8421052631578947, 'f1-score': 0.8648648648648649, 'support': 38}, 'accuracy': 0.869565217391304
0.8682192764471246, 'support': 161}, 'weighted avg': {'precision': 0.8783168976321529, 'recall': (
Accuracy :  0.8695652173913043
--------------------------------
FOLD 1
--------------------------------
Starting K-fold Cross validation of the loaded model
{'0': {'precision': 0.8421052631578947, 'recall': 0.7619047619047619, 'f1-score': 0.8, 'support':
0.7555555555555556, 'support': 42}, '2': {'precision': 0.8, 'recall': 0.6829268292682927, 'f1-scor
score': 0.7631578947368423, 'support': 36}, 'accuracy': 0.7639751552795031, 'macro avg': {'precis
'weighted avg': {'precision': 0.7703007518796992, 'recall': 0.7639751552795031, 'f1-score': 0.764
Accuracy :  0.7639751552795031
--------------------------------
FOLD 2
--------------------------------
Starting K-fold Cross validation of the loaded model
{'0': {'precision': 0.8695652173913043, 'recall': 0.851063829787234, 'f1-score': 0.860215053763440
0.8048780487804879, 'support': 37}, '2': {'precision': 0.96875, 'recall': 0.7948717948717948, 'f1-
0.8421052631578947, 'f1-score': 0.8421052631578947, 'support': 38}, 'accuracy': 0.84472049689441,
0.8451094505803853, 'support': 161}, 'weighted avg': {'precision': 0.8558021649113332, 'recall': (
Accuracy :  0.84472049689441
--------------------------------
FOLD 3
--------------------------------
Starting K-fold Cross validation of the loaded model
{'0': {'precision': 1.0, 'recall': 0.8048780487804879, 'f1-score': 0.8918918918918919, 'support':
0.8333333333333334, 'support': 42}, '2': {'precision': 0.96875, 'recall': 0.7948717948717948, 'f1-
0.8974358974358975, 'f1-score': 0.8641975308641975, 'support': 39}, 'accuracy': 0.863354037267080
0.8656655481772851, 'support': 161}, 'weighted avg': {'precision': 0.8844246031746031, 'recall': (
Accuracy :  0.8633540372670807
--------------------------------
FOLD 4
--------------------------------
Starting K-fold Cross validation of the loaded model
{'0': {'precision': 0.8333333333333334, 'recall': 0.8333333333333334, 'f1-score': 0.8333333333333
0.851063829787234, 'support': 45}, '2': {'precision': 0.9210526315789473, 'recall': 0.89743589743
0.8085106382978723, 'f1-score': 0.8351648351648351, 'support': 47}, 'accuracy': 0.857142857142857
0.8571632268440779, 'support': 161}, 'weighted avg': {'precision': 0.8586748795033481, 'recall': (
Accuracy :  0.8571428571428571
--------------------------------
FOLD 5
--------------------------------
```

Figure 8. Sample Image of K-Fold Cross Validation Output on Original Model

*K-Fold Cross Validation Results on **Original Model (10 Folds)***

Fold 0

	precision	recall	f1-score	support
Cloth	0.98	0.84	0.90	49
N95	0.79	0.92	0.85	36
Surgical	0.83	0.89	0.86	38
Mask	0.89	0.84	0.86	38
Accuracy			0.87	404

Fold 1

	precision	recall	f1-score	support
Cloth	0.98	0.84	0.90	49
N95	0.79	0.92	0.85	36
Surgical	0.83	0.89	0.86	38
Mask	0.89	0.84	0.86	38
Accuracy			0.87	404

Fold 2

	precision	recall	f1-score	support
Cloth	0.87	0.85	0.86	47
N95	0.73	0.89	0.80	37
Surgical	0.97	0.79	0.87	39
Mask	0.84	0.84	0.84	38

Accuracy			0.84	404

Fold 3

	precision	recall	f1-score	support
Cloth	1.0	0.80	0.89	41
N95	0.74	0.95	0.83	42
Surgical	0.97	0.79	0.87	39
Mask	0.89	0.87	0.86	39
Accuracy			0.86	404

Fold 4

	precision	recall	f1-score	support
Cloth	0.83	0.83	0.83	30
N95	0.82	0.88	0.85	45
Surgical	0.92	0.90	0.90	39
Mask	0.86	0.81	0.84	47
Accuracy			0.86	404

Fold 5

	precision	recall	f1-score	support
Cloth	0.91	0.85	0.88	34
N95	0.71	0.95	0.81	38
Surgical	0.93	0.79	0.85	47
Mask	0.84	0.75	0.79	41

Accuracy			0.83	404

Fold 6

	precision	recall	f1-score	support
Cloth	0.92	0.81	0.86	42
N95	0.78	0.90	0.84	42
Surgical	0.95	0.844	0.89	45
Mask	0.82	0.90	0.86	31
Accuracy			0.863	404

Fold 7

	precision	recall	f1-score	support
Cloth	0.84	0.82	0.83	34
N95	0.71	0.90	0.8	44
Surgical	0.82	0.69	0.75	39
Mask	0.81	0.74	0.79	43
Accuracy			0.793	404

Fold 8

	precision	recall	f1-score	support
Cloth	0.91	0.91	0.91	34
N95	0.77	0.94	0.85	36
Surgical	0.86	0.72	0.78	43
Mask	0.91	0.89	0.90	47

Accuracy			0.863	404

Fold 9

	precision	recall	f1-score	support
Cloth	0.85	0.74	0.79	46
N95	0.75	0.89	0.81	37
Surgical	0.86	0.79	0.82	38
Mask	0.76	0.79	0.77	39
Accuracy			0.8	404

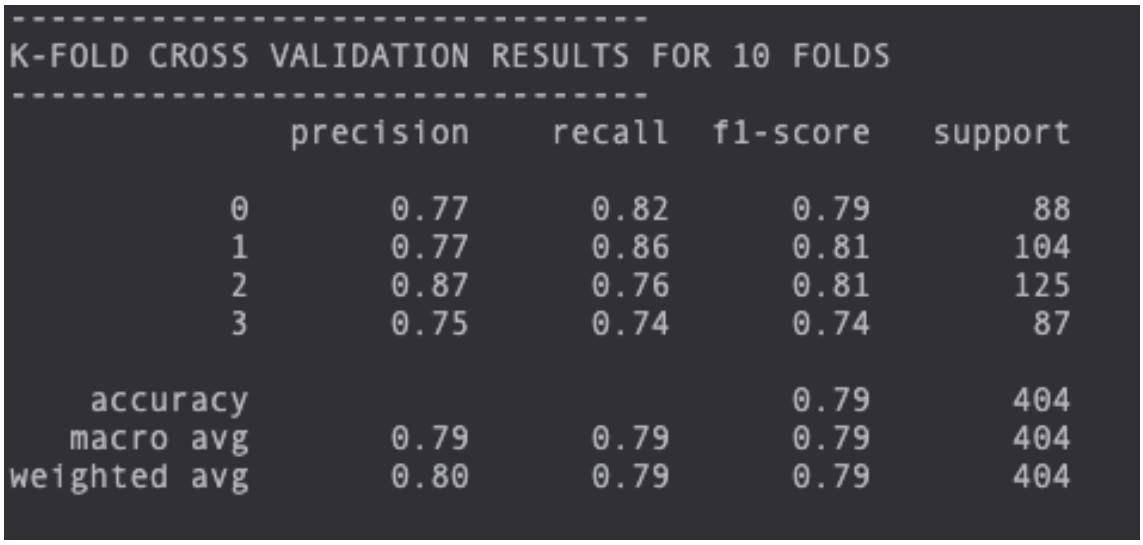


Figure 9. Screen capture of aggregate precision, recall, f1 score and support on Original Model (V2.2)

*K-Fold Cross Validation Results on **New Model (10 Folds)***

Fold 0

	precision	recall	f1-score	support
Cloth	0.84	0.59	0.7	47
N95	0.57	0.91	0.70	34
Surgical	0.73	0.74	0.74	47
Mask	0.70	0.60	0.65	48
Accuracy			0.69	404

Fold 1

	precision	recall	f1-score	support
Cloth	0.82	0.86	0.84	36
N95	0.74	0.84	0.79	51
Surgical	0.95	0.85	0.90	48
Mask	0.78	0.71	0.74	41
Accuracy			0.82	404

Fold 2

	precision	recall	f1-score	support
Cloth	0.78	0.77	0.77	48
N95	0.52	0.87	0.65	31
Surgical	0.85	0.76	0.80	43
Mask	0.84	0.59	0.69	54

Accuracy			0.73	404

Fold 3

	precision	recall	f1-score	support
Cloth	0.77	0.77	0.77	36
N95	0.77	0.81	0.79	52
Surgical	0.78	0.76	0.77	41
Mask	0.85	0.83	0.84	47
Accuracy			0.79	404

Fold 4

	precision	recall	f1-score	support
Cloth	0.87	0.8	0.83	50
N95	0.84	0.79	0.82	53
Surgical	0.78	0.84	0.81	38
Mask	0.71	0.8	0.75	35
Accuracy			0.81	404

Fold 5

	precision	recall	f1-score	support
Cloth	0.80	0.80	0.80	41
N95	0.5	0.71	0.59	34
Surgical	0.84	0.69	0.76	53
Mask	0.72	0.65	0.68	48

Accuracy			0.71	404

Fold 6

	precision	recall	f1-score	support
Cloth	0.77	0.79	0.78	43
N95	0.69	0.81	0.74	46
Surgical	0.81	0.77	0.79	39
Mask	0.83	0.72	0.77	47
Accuracy			0.77	404

Fold 7

	precision	recall	f1-score	support
Cloth	0.87	0.83	0.85	48
N95	0.55	0.91	0.81	32
Surgical	0.93	0.81	0.86	47
Mask	0.86	0.65	0.74	48
Accuracy			0.78	404

Fold 8

	precision	recall	f1-score	support
Cloth	0.925	0.75	0.83	49
N95	0.72	0.78	0.83	55
Surgical	0.81	0.83	0.82	36
Mask	0.61	0.66	0.63	35

Accuracy			0.76	404

Fold 9

	precision	recall	f1-score	support
Cloth	0.8	0.88	0.84	41
N95	0.68	0.86	0.76	51
Surgical	0.89	0.74	0.81	47
Mask	0.81	0.58	0.67	36
Accuracy			0.78	404

	precision	recall	f1-score	support
0	0.73	0.77	0.75	111
1	0.72	0.79	0.75	119
2	0.80	0.78	0.79	105
3	0.75	0.65	0.69	105
accuracy			0.75	440
macro avg	0.75	0.75	0.75	440
weighted avg	0.75	0.75	0.75	440

```
Accuracy of the network on the 440 test images: 74.77272727272727 %
```

Figure 10. (a) Screen capture of aggregate precision, recall, f1 score and support on New Model (V2.3),
(b) Accuracy calculator depicting training images

Confusion Matrix

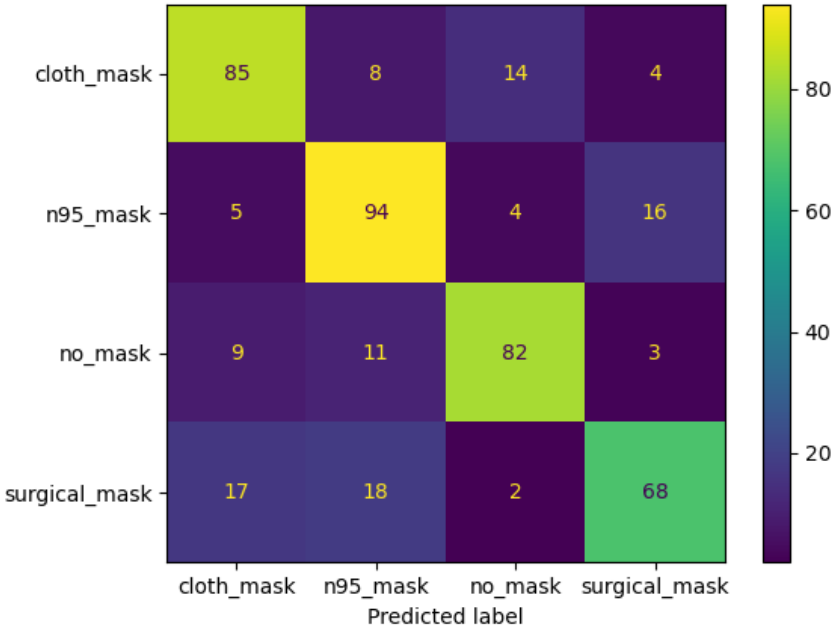


Figure 11. Confusion Matrix of New Model (V 2.3) with specified labels

Bias Analysis

For the bias analysis we decided to look into 2 categories - gender and age. By selecting few images that were not used in neither training nor testing to define the accuracy [7][8].

Before Modification

Gender

Male:

Accuracy of the network on the 40 test images: 37.5 %

	precision	recall	f1-score	support
0	0.57	0.80	0.67	10
1	0.18	0.20	0.19	10
2	0.57	0.40	0.47	10
3	0.12	0.10	0.11	10
accuracy			0.38	40
macro avg	0.36	0.38	0.36	40
weighted avg	0.36	0.38	0.36	40

With the following confusion matrix:

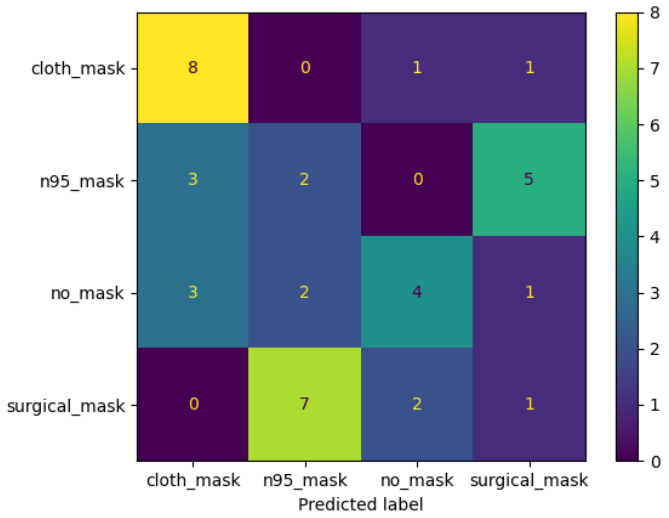


Figure 12. Confusion Matrix of Male on old Model

Female

Accuracy of the network on the 40 test images: 42.5 %

	precision	recall	f1-score	support
0	0.42	0.50	0.45	10
1	0.43	0.30	0.35	10
2	0.42	0.50	0.45	10
3	0.44	0.40	0.42	10
accuracy			0.42	40
macro avg	0.43	0.43	0.42	40
weighted avg	0.43	0.42	0.42	40

With the following confusion matrix:

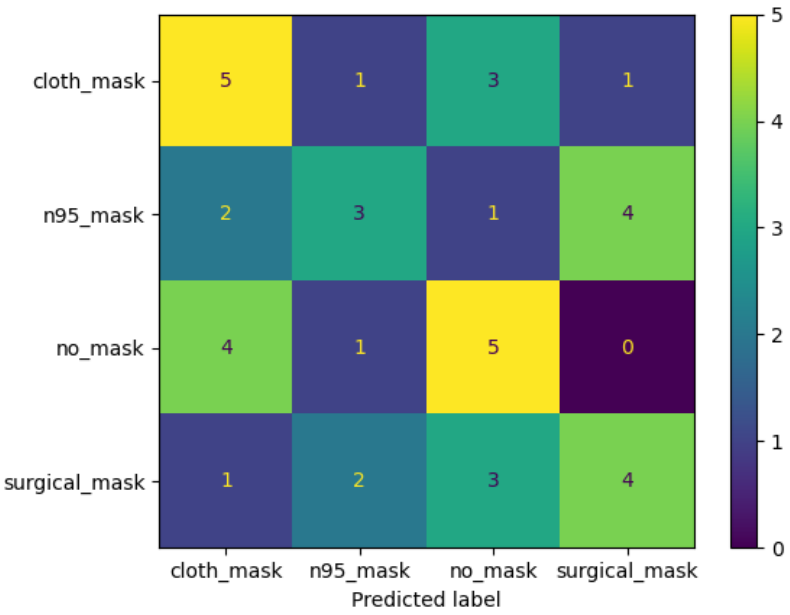


Figure 13. Confusion Matrix of Female on old Model

It can be seen that the model better evaluates females than males

Age

Old

Accuracy of the network on the 40 test images: 62.5 %

	precision	recall	f1-score	support
0	0.55	0.60	0.57	10
1	0.71	0.50	0.59	10
2	0.67	0.60	0.63	10
3	0.62	0.80	0.70	10
accuracy			0.62	40
macro avg	0.64	0.62	0.62	40
weighted avg	0.64	0.62	0.62	40

With the following confusion matrix:

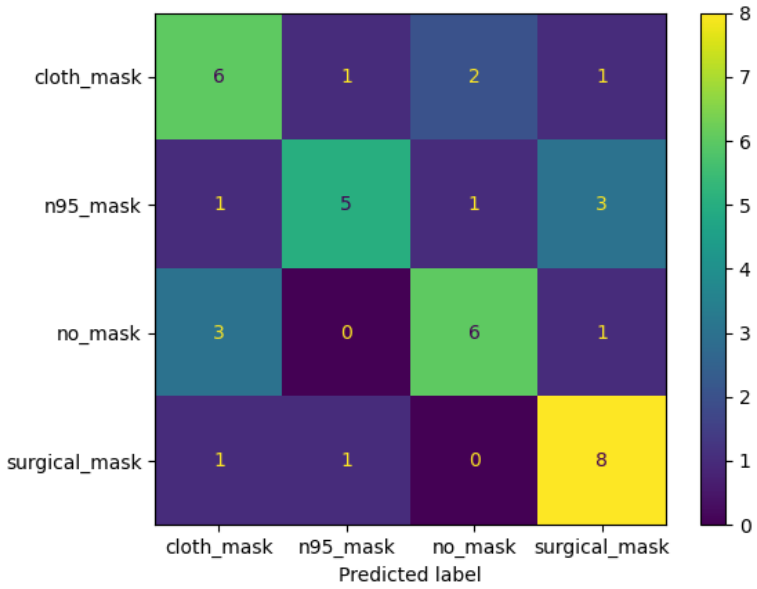


Figure 14. Confusion Matrix of Old(age) on old Model

Young

Accuracy of the network on the 40 test images: 52.5 %

	precision	recall	f1-score	support
0	0.44	0.70	0.54	10
1	0.33	0.10	0.15	10
2	0.70	0.70	0.70	10
3	0.55	0.60	0.57	10
accuracy			0.53	40
macro avg	0.50	0.53	0.49	40
weighted avg	0.50	0.53	0.49	40

With the following confusion matrix:

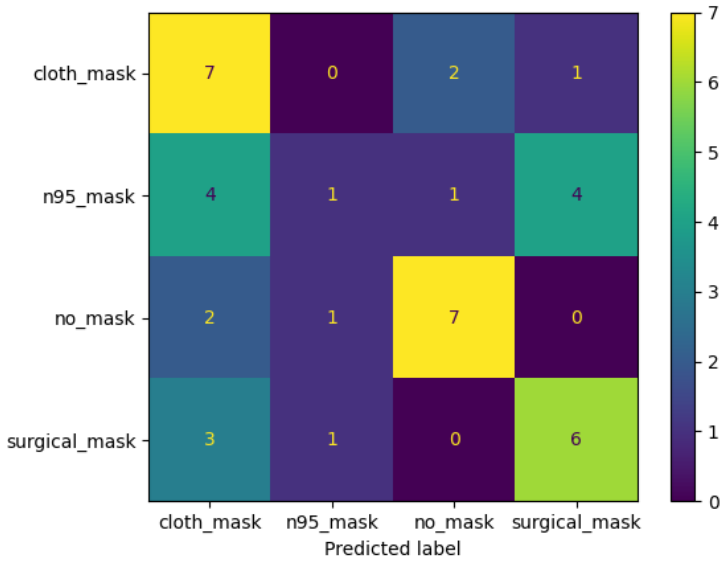


Figure 15. Confusion Matrix of Young on old Model

It can be seen that the model better evaluates old than young

After Modification

After some analysis, we discovered that there were more female images and more old people images used to train our model. Some extra images were added according to the category which was identified as the worst to eliminate bias reasoning. Here is the final dataset that was used (link) [1][7][8]

Gender

Male

Accuracy of the network on the 40 test images: 57.49999999999999 %

	precision	recall	f1-score	support
0	0.55	0.60	0.57	10
1	0.57	0.40	0.47	10
2	0.57	0.80	0.67	10
3	0.62	0.50	0.56	10
accuracy			0.57	40
macro avg	0.58	0.57	0.57	40
weighted avg	0.58	0.57	0.57	40

With the following confusion matrix:

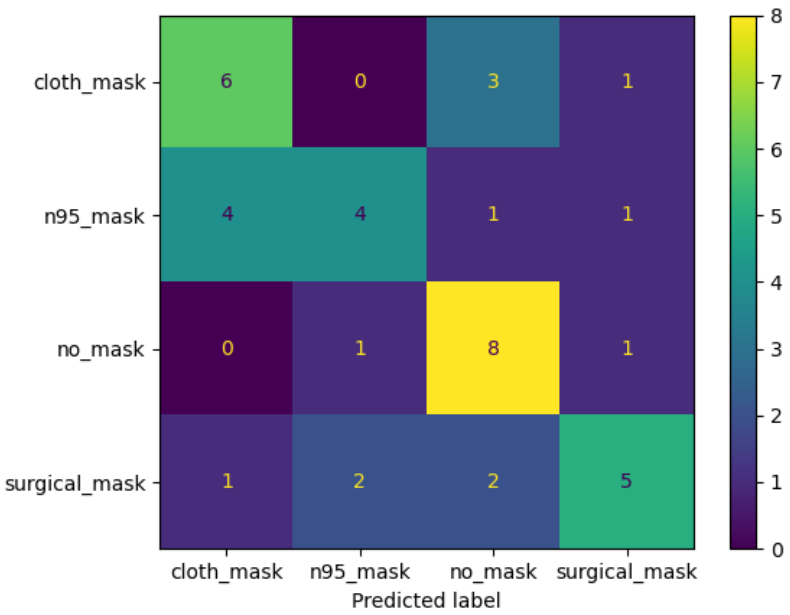


Figure 16. Confusion Matrix of Male on new Model

Female

Accuracy of the network on the 40 test images: 55.00000000000001 %

	precision	recall	f1-score	support
0	0.38	0.50	0.43	10
1	0.50	0.40	0.44	10
2	0.67	0.80	0.73	10
3	0.71	0.50	0.59	10
accuracy			0.55	40
macro avg	0.57	0.55	0.55	40
weighted avg	0.57	0.55	0.55	40

With the following confusion matrix:

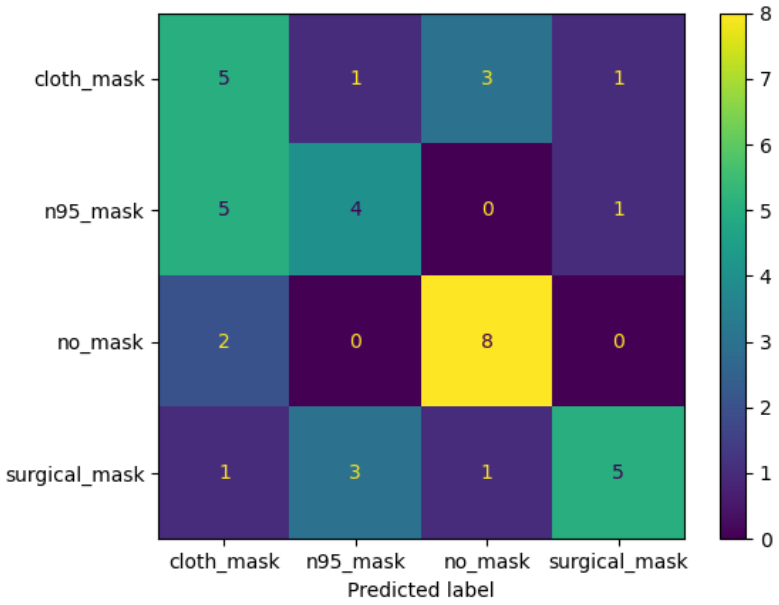


Figure 17. Confusion Matrix of Female on new Model

Age

Old

Accuracy of the network on the 40 test images: 55.00000000000001 %

	precision	recall	f1-score	support
0	0.50	0.60	0.55	10
1	0.62	0.50	0.56	10
2	0.64	0.90	0.75	10
3	0.33	0.20	0.25	10
accuracy			0.55	40
macro avg	0.53	0.55	0.53	40
weighted avg	0.53	0.55	0.53	40

With the following confusion matrix:

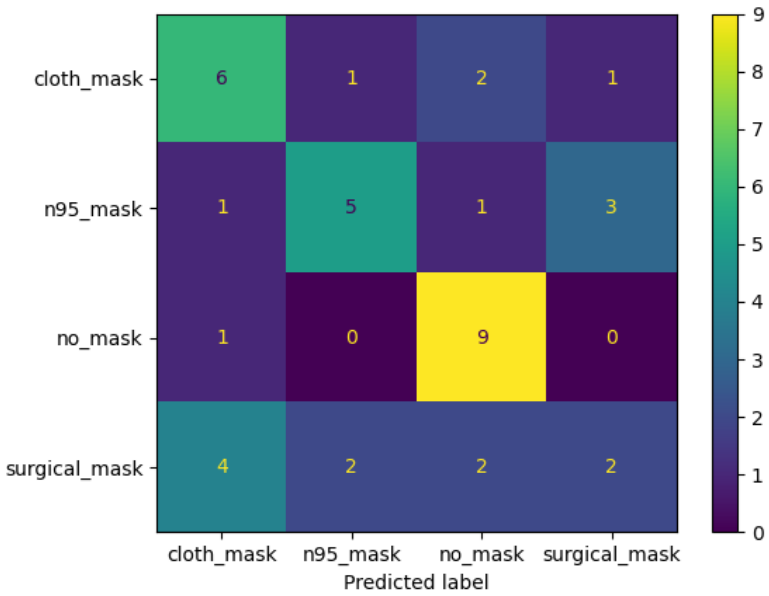


Figure 18. Confusion Matrix of Old(age) on new Model

Young

Accuracy of the network on the 40 test images: 55.00000000000001 %

	precision	recall	f1-score	support
0	0.50	0.50	0.50	10
1	0.50	0.30	0.37	10
2	0.60	0.90	0.72	10
3	0.56	0.50	0.53	10
accuracy			0.55	40
macro avg	0.54	0.55	0.53	40
weighted avg	0.54	0.55	0.53	40

With the following confusion matrix:

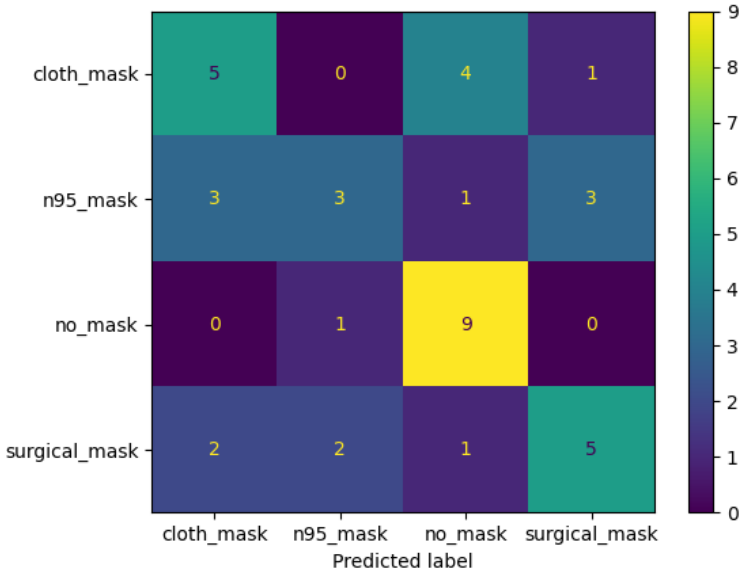


Figure 19. Confusion Matrix of Young on new Model

As can be seen, adding the missing images led to less bias evaluation, for gender it went from 5% difference to 2.5%, and for age it went from 10% to 0%. Note that the goal was not to get a higher evaluation percentage, it was to reduce the gap between evaluations for two categories - gender and age. The main goal was achieved successfully.

Results

Part One

Our results showed that training the dataset over 25 epochs showed accurate improvement through time. There were some small discrepancies in our training success rate for the first 10 epochs, but it adjusted accordingly as more epochs were being conducted over our dataset with our training model. Note that there is an accuracy level of 84.15% within our network --- this will be crucial in evaluating our improving performance level. For the second iteration of this training model, our aim would be to increase the batch size to decrease noise level and increase the accuracy level of the network to above 85%.

Part Two

We focused on two elements: evaluating the trained model against a real image to show that the classifier is working and we implemented the saving of the model on better accuracy and rejecting the epochs that turned out to have worse accuracy. Unfortunately we are not able to train on a better computer with the given resources we have. As well we used a better balanced database, based on our analysis of the two prominent biases as mentioned in earlier parts.

The implementation of the saving on better accuracy did have a negative effect on our model as it reduced the accuracy of our predictions (75%):

	precision	recall	f1-score	support
0	0.73	0.77	0.75	111
1	0.72	0.79	0.75	119
2	0.80	0.78	0.79	105
3	0.75	0.65	0.69	105
accuracy			0.75	440
macro avg	0.75	0.75	0.75	440
weighted avg	0.75	0.75	0.75	440

Figure 20:. Screen capture of aggregate precision, recall, f1 score and support of New Model (V2.5).

With the limited resources that we have and out of curiosity, we trained over our old model for another 25 epochs ,with save on better accuracy, and we noticed an increasing in accuracy average accuracy of 87% :

	precision	recall	f1-score	support
0	0.96	0.76	0.85	104
1	0.92	0.86	0.89	92
2	0.82	0.92	0.87	106
3	0.82	0.93	0.87	102
accuracy			0.87	404
macro avg	0.88	0.87	0.87	404
weighted avg	0.88	0.87	0.87	404

Figure 21. Screen capture of aggregate precision, recall, f1 score and support on Old Model Retraining (V2.2 base model + V2.5 retaining).

This represents an increase of 3% from our old model (84%).

Furthermore if we look at the training curve:

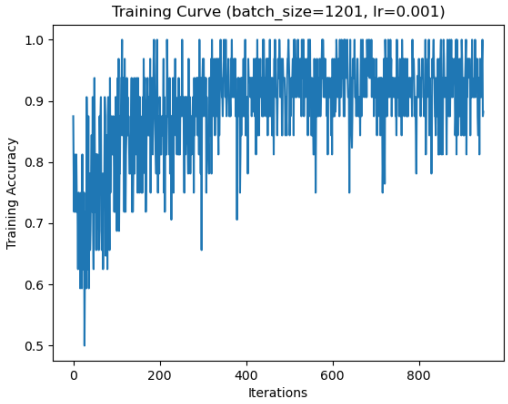


Figure 22. Training Curve depicting accuracy graph of the extra 25 epochs of training on top of the old model, using save-on-better-accuracy

This offers the perfect example of an architecture reaching its maximum potential. The only thing that could potentially improve it is; increasing the batch size.

We also increased the batch size from 32 to 64, and that did not have an effect for the 25 epochs of training. Due to the lack of better hardware, we were unable to retrain for 50 epochs and a batch of 64 combined, to validate if both combinations will have an effect. However there is some reduction in training noise as seen in the figure below:

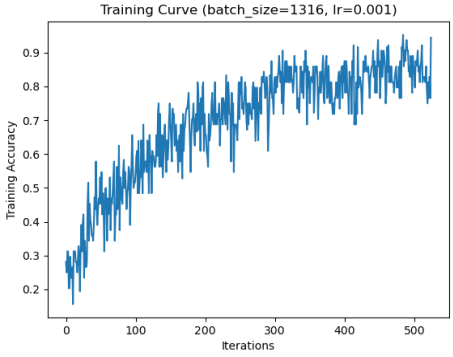


Figure 23. Training Curve depicting accuracy graph of the latest model for 25 epochs and an increase of batch size (64 vs 32). Notice the noise reduction.

Reference Section

[1] A. L. Maranhao, “Face mask detection,” *Face Mask Detection*, 22-May-2020. [Online]. Available: https://www.kaggle.com/datasets/andrewmvd/face-mask-detection/metadata. [Accessed: 07-Jun-2022].

[2] P. Caplan, “What is a JPEG? the invisible object you see every day,” 24-Sep-2013. [Online]. Available: https://www.theatlantic.com/technology/archive/2013/09/what-is-a-jpeg-the-invisible-object-you-see-every-day/279954/. [Accessed: 07-Jun-2022].

[3] A. Mishra, “Face mask detection,” *Kaggle*, 16-Jun-2020. [Online]. Available: https://www.kaggle.com/code/ayushimishra2809/face-mask-detection/data?select=Medical%2Bmask. [Accessed: 08-Jun-2022].

[4] Y. Wang, “Which mask are you wearing? face mask type detection with tensorflow and Raspberry Pi,” *Medium*, 01-Jun-2020. [Online]. Available: https://towardsdatascience.com/which-mask-are-you-wearing-face-mask-type-detection-with-tensorflow-and-raspberry-pi-1c7004641f1. [Accessed: 07-Jun-2022].

[5] Torch Contributors, “MaxPool2d,” MaxPool2d - PyTorch 1.11.0 documentation. [Online].Available:https://pytorch.org/docs/stable/generated/torch.nn.MaxPool2d.html. [Accessed: 08-Jun-2022].

[6] Torch Contributors, “Conv2,” *Conv2d - PyTorch 1.11.0 documentation*. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html. [Accessed: 08-Jun-2022].

[7] arbazkhan971, “Face mask detection using CNN (98% accuracy),” *Face Mask Detection using CNN (98% Accuracy)*, 05-Sep-2021. [Online]. Available: https://www.kaggle.com/arbazkhan971/face-mask-detection-using-cnn-98-accuracy. [Accessed: 18-Jun-2022].

[8] A. Aman10kr Kumar, “Face mask detection using SSD,” *Face Mask Detection using SSD*, 23-Dec-2020. [Online]. Available: https://www.kaggle.com/code/aman10kr/face-mask-detection-using-ssd/data?select=Medical%2Bmask. [Accessed: 19-Jun-2022].