AI Mask Detector:
Project Assignment - Part One

Concordia University
June 8th 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

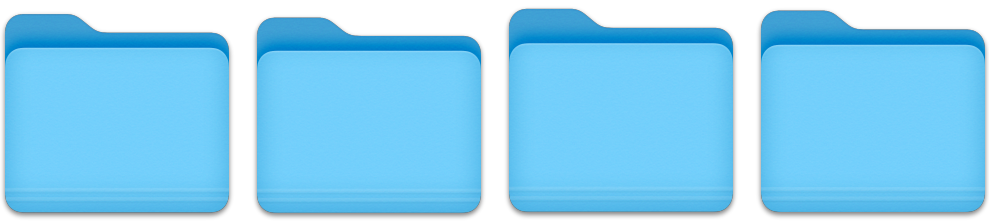
- responsible for analyzing, evaluating, and applying the generated model;

(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

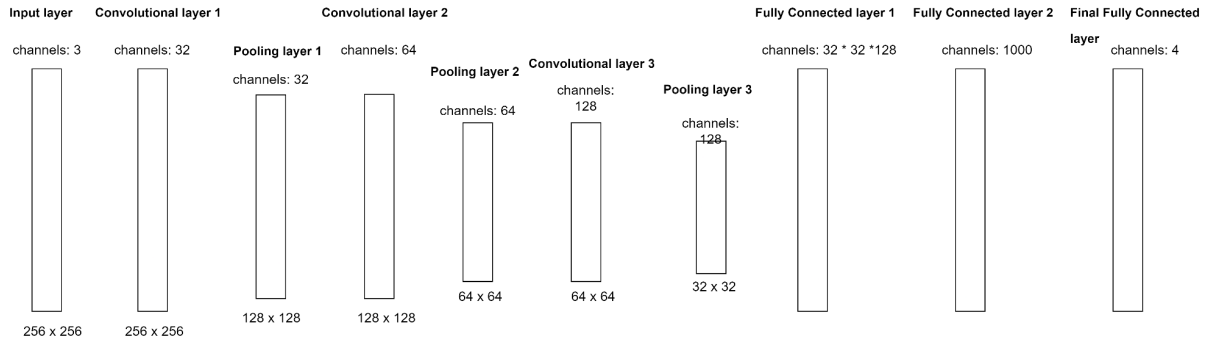


Fig.2 Design schema of the CNN architecture

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

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.

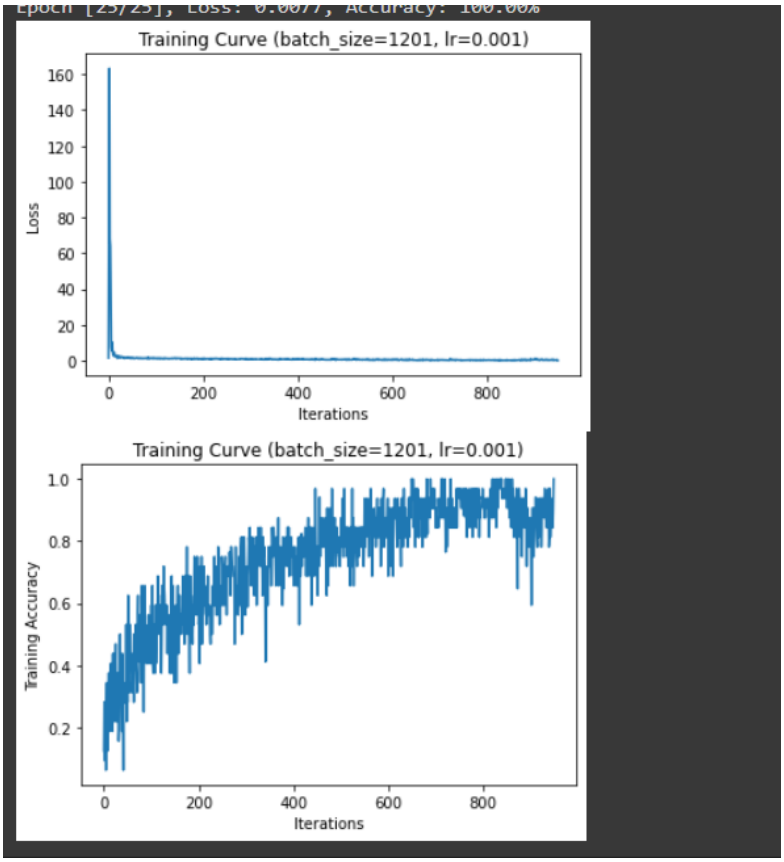


Fig.3 Accuracy graph depicting training curve

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

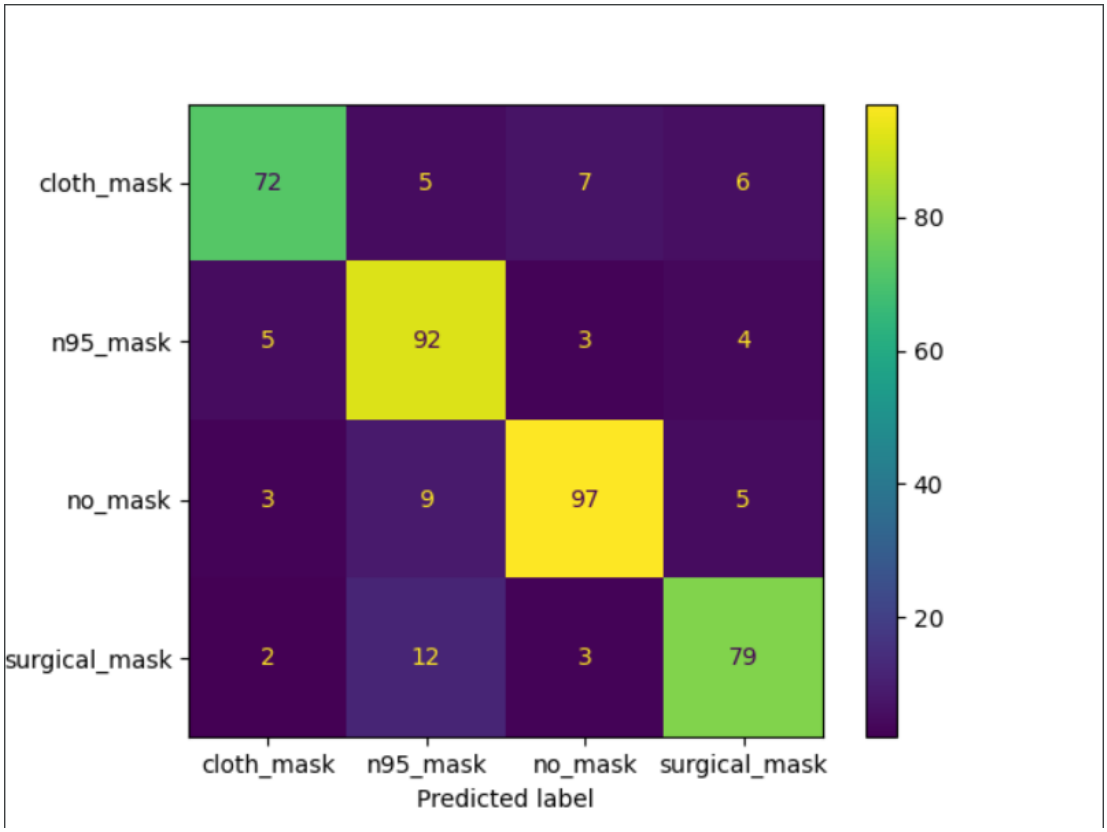


Figure 5. Confusion matrix with specified labels

Results

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%.

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].