Dahan, Amir Michael
Supervised by Prof. Margarita Osadchy

Feb. 2022

M.Sc. Final Project Report - *Medical Face Mask Detection*

Contents

1 Goals

In this project, the goal is to label all faces in the given image as mask/no mask
We'll need to determine which of these women is wearing a medical mask.

FIGURE 1: Original Image

FIGURE 2: Annotated Image

1.1 Expected Achievements

Contents:

1. Take both datasets and use augmentation to improve pictures.
2. Pick two models best suits (CNN) our problem & search for the best hyper parameters.
3. Train two models on datasets and save it for later use
   - Fine-tune the human detector model's upper layer
   - Train mask detector from scratch
4. Test it using a GUI or a great integration script (see appendixes section 7), and run on an unseen test images

2 Data

Data used for this project:

2.1 Datasets

2.1.1 Face Mask Detection Dataset [1]

Comprised of the following three classes:

- Face with mask
- Face without mask
- Mask worn incorrectly

Classes Distribution: (Equal number of images per class, to eliminate bias)

- mask_weared_incorrect: 2997
- with_mask: 2997
- without_mask: 2997

2.1.2 Face Mask Dataset [1] & Natural Images Dataset [2]

A custom superset of both [1] & [2], Comprised of the following classes:

- airplane
- car
- cat
- dog
- flower
- fruit
- mask_weared_incorrect
- motorbike

- with_mask
- without_mask

Classes Distribution:

- airplane: 727
- car: 968
- cat: 885
- dog: 702
- flower: 843
- fruit: 1000
- mask_weared_incorrect: 2997 –> 1000
- motorbike: 788
- with_mask: 2997 –> 1000
- without_mask: 2997 –> 1000

2.2 Transformations used for data

- Random Horizontal Flip: Horizontally flip the given image randomly with a given probability
- Random Resized Crop: Crop the given image to random size and aspect ratio

FIGURE 3: Cropping Sample

3 Architecture

In this section I'll describe the model and the modifications I've made during second phase.

3.1 The Model

I chose a CNN, no other choice for that kind of task. I started from a known network that I previously used for a similar task, combined with an intuition from a known model for image classification. I played a little with the number of filters for each layer, this way I can capture more features. The problem with this method was that it take a lot of RAM (over 4GB), which made me debug on a pre-trained resnet18.

```
CNN(
  (loss_func): CrossEntropyLoss()
  (feature_extractor): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU()
    (8): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=200704, out_features=1024, bias=True)
    (2): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): ReLU()
    (4): Linear(in_features=1024, out_features=512, bias=True)
    (5): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

FIGURE 4: architecture

```
SGD (
Parameter Group 0
    dampening: 0
    lr: 0.001
    momentum: 0.9
    nesterov: False
    weight_decay: 0
)
```

FIGURE 5: optimizer

3.2 Training

The training phase includes three models:

1. Natural Image:
    - A Pre-Trained Model that detects human beings (alongside other)
    - Isn't being used for final classification
    - Used for Human Model Retraining
2. Human Model
    - Fine tune 1 model's top layer
    - Classes: elaborated in 2.1.2
    - Add three classes (with_mask, without_mask, mask_weared_incorrect
    - Remove 'person' class in return
    - Used to determine if the object is a person
3. Face Mask Detection
    - A model we train from scratch
    - Classes: elaborated in 2.1.1
    - Used for Human Model Retraining

3.3 Testing Pipeline

- Take an image
- Determine if the object/s in the picture are human (using Human Detection Model model)

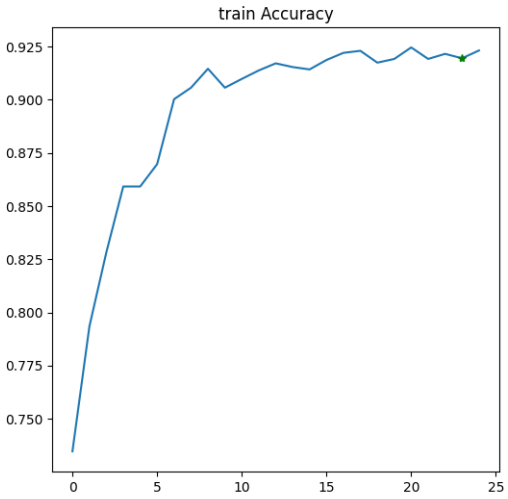
- Crop the object/s one by one (using first model), & determine masked/non-masked/partially-masked (using second model)

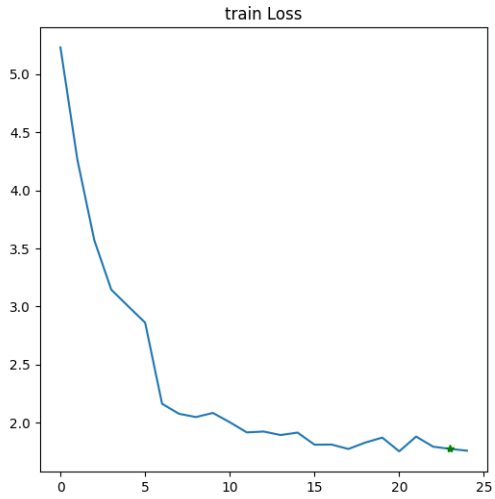
4 Results

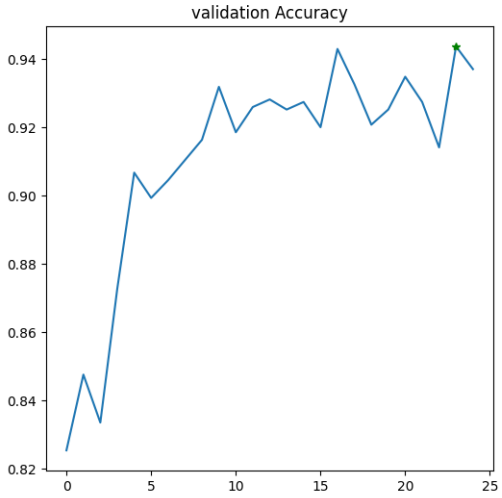
Results will be elaborated for both models:

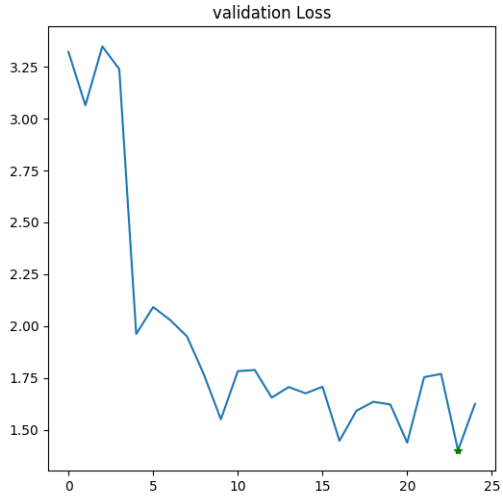
- Medical Face Mask Detection - Accuracy: 93.10% Loss: 1.6244
- Human Detection - Accuracy: 97.49%, Loss: 0.4224

1. Mask Detection

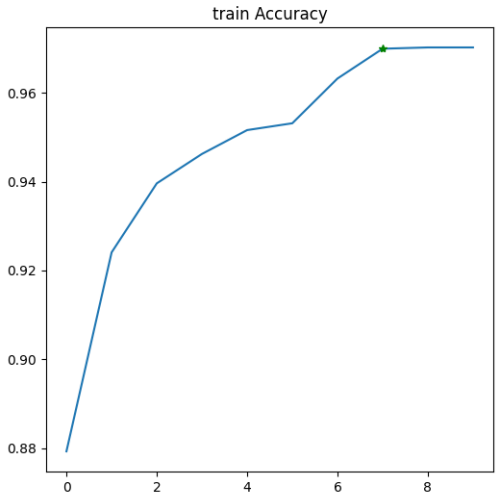


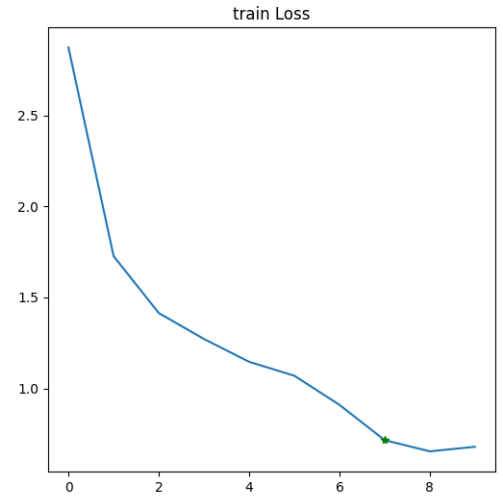


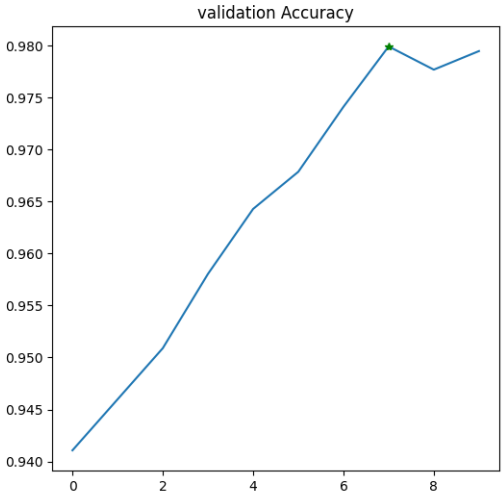


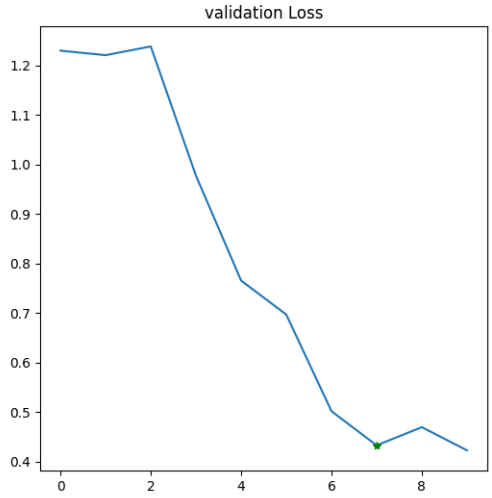


2. Human Detection









5 Gui

In this section we'll describe how our gui looks like and some screenshots

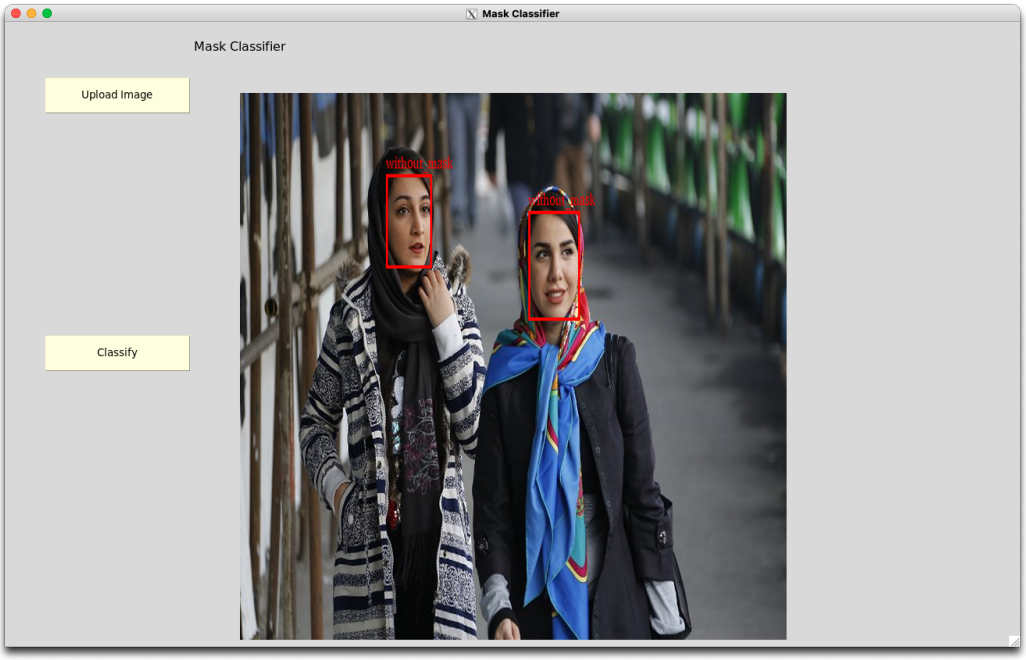


FIGURE 6: Non-Masked Sample

6 Bibliography

References

[1] Face Mask Detection, Building a face mask classifier *https://www.kaggle.com/vijaykumar1799/face-mask-detection*

[2] A compiled dataset of 6899 images from 8 distinct classes. *https://www.kaggle.com/prasunroy/natural-images*

[3] FaceNet: A Unified Embedding for Face Recognition and Clustering *https://arxiv.org/abs/1503.03832*

7 Appendixes

7.1 The Code

My project is consisted of the following components:

doc	Documentation, source and pdf
out	Saved Checkpoint - model & state dict.
samples	Test images for Demo
src	Project's sources
support	Dockerfile to reproduce the required env.

Table 1

Camera.py	Capture frames from Camera or a Video
FaceMaskClassificationUtils.py	Shared Helper and constants
FaceMaskDetection.py	FaceMask classifier model
FinalProject.py	train and classify without a GUI
Gui.py	A useful GUI assisting upload and classify images
HumanDetection.py	Human Detection Model
ObjectCrop.py	External Lib For Objects Detection

Table 2

7.2 How To Run The Project

Code knows to distinguish between google colaboratory environment and other environments, which leads to picking the appropriate model.

- Train the model and test it, export a checkpoint file for the model and weights

```
$$ python src/FinalProject.py --help
optional arguments:
  -h, --help            show this help message and exit
  -H HUMAN_MODEL_PATH, --human_model_path HUMAN_MODEL_PATH
                        Human Model Path
```

```
  -a HUMAN_DATA_PATH, --human_data_path HUMAN_DATA_PATH
                        Human Data Path
  -M MASK_MODEL_PATH, --mask_model_path MASK_MODEL_PATH
                        Face Mask Model path
  -b MASK_DATA_PATH, --mask_data_path MASK_DATA_PATH
                        Face Mask Data path
  -F IMAGE_PATH, --image_path IMAGE_PATH
                        image to classify
  --train
```

- Run GUI to classify images selected by the user:

```
$$ python src/Gui.py --help
optional arguments:
  -h, --help            show this help message and exit
  -H HUMAN_MODEL_PATH, --human_model_path HUMAN_MODEL_PATH
                        Human Model Path
  -M MASK_MODEL_PATH, --mask_model_path MASK_MODEL_PATH
                        Face Mask Model path
```

- Run Video/Camera capture and classify frames

```
$$ python src/Camera.py
usage: Camera.py [-h] -H HUMAN_MODEL_PATH -M MASK_MODEL_PATH -F VIDEO_PATH
                 [--skip_capturing]
```

Prerequisites:

- Pre trained model checkpoint (Model & weights)
- Project's Python source code

7.3 Demo Agena

7.3.1 Textual

```
$ cd ~/Project/ && time python src/FinalProject.py \
```

```
                -H out/CombinedModel.pth \
                -M out/MaskModel.pth \
                -a data/human/ -b data/face-mask/ \
                -N out/NaturalNodel.pth \
                -c data/natural_images \
                -F samples/nre/Partial.jpg
```

```
(venv36) adahan@kodkoda4:~/Project$ cd ~/Project/ && time python src/FinalProject.py -H out_models/CombinedModel2.pth -M out_mo
dels/MaskModel2.pth -a data/human/ -b data/face-mask/ -N out_models/NaturalNodel2.pth -c data/natural_images -F samples/nre/Par
tial.jpg
Namespace(human_data_path='data/human/', human_model_path='out_models/CombinedModel2.pth', image_path='samples/nre/Partial.jpg'
, mask_data_path='data/face-mask/', mask_model_path='out_models/MaskModel2.pth', natural_data_path='data/natural_images', natur
al_model_path='out_models/NaturalNodel2.pth', train=False)
FinalProject: shouldn't train
human model successfully loaded
mask model successfully loaded
FinalProject: loading models took 0:00:02.425118
trying to classify image samples/nre/Partial.jpg
classify if it is human
cpu
classify_is_human: this is a with_mask.
classify_is_human: it is human
cropping image
single_image_classify got 1 faces to classify
classify_mask_usage: this is a mask_weared_incorrect.
FinalProject: testing image took 0:00:00.381641

real    0m6.211s
user    0m5.545s
sys     0m2.538s
```

FIGURE 7: Cropping Sample

7.3.2 Gui

```
$ cd ~/Project/ && python src/Gui.py \
                    -H out/CombinedModel.pth \
                    -M out/MaskModel.pth
```

I'll attach here samples of my self Gui tests:

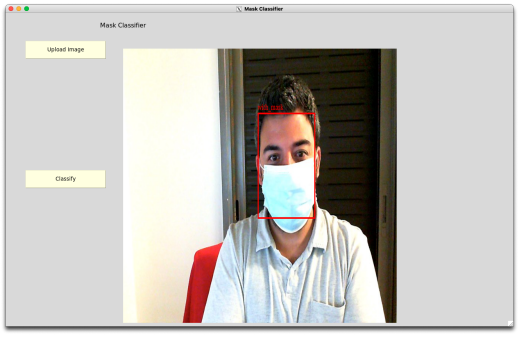


FIGURE 8: Amir is with his mask ON

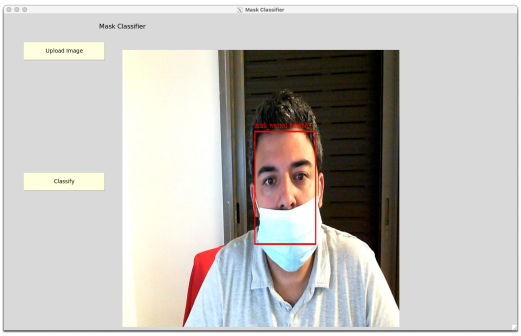


FIGURE 9: Amir is with his mask Partially ON

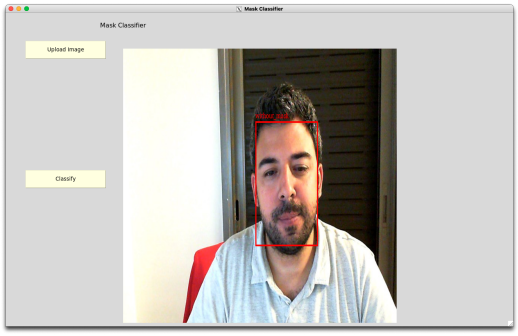


FIGURE 10: Amir is with his mask OFF

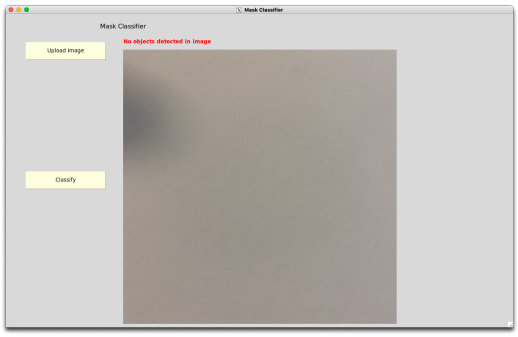


FIGURE 11: A blank wall

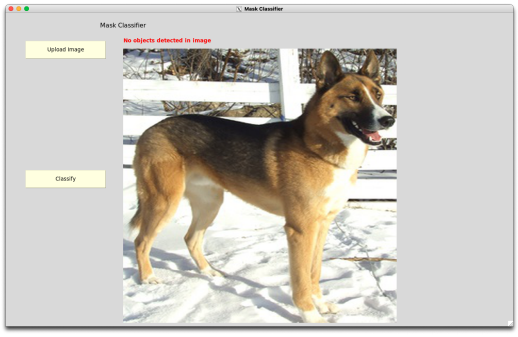


FIGURE 12: A Dog

7.3.3 Video

```
$ cd ~/Project/ && python src/Camera.py \
                   -H out/CombinedModel.pth \
                   -M out/MaskModel.pth \
                   -F samples/nre/Movie.mov --skip
```