`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 my datasets and save it for later use.
4. Test it using a GUI or a great integration script, 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, comvbined 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

3.3 Testing Pipeline

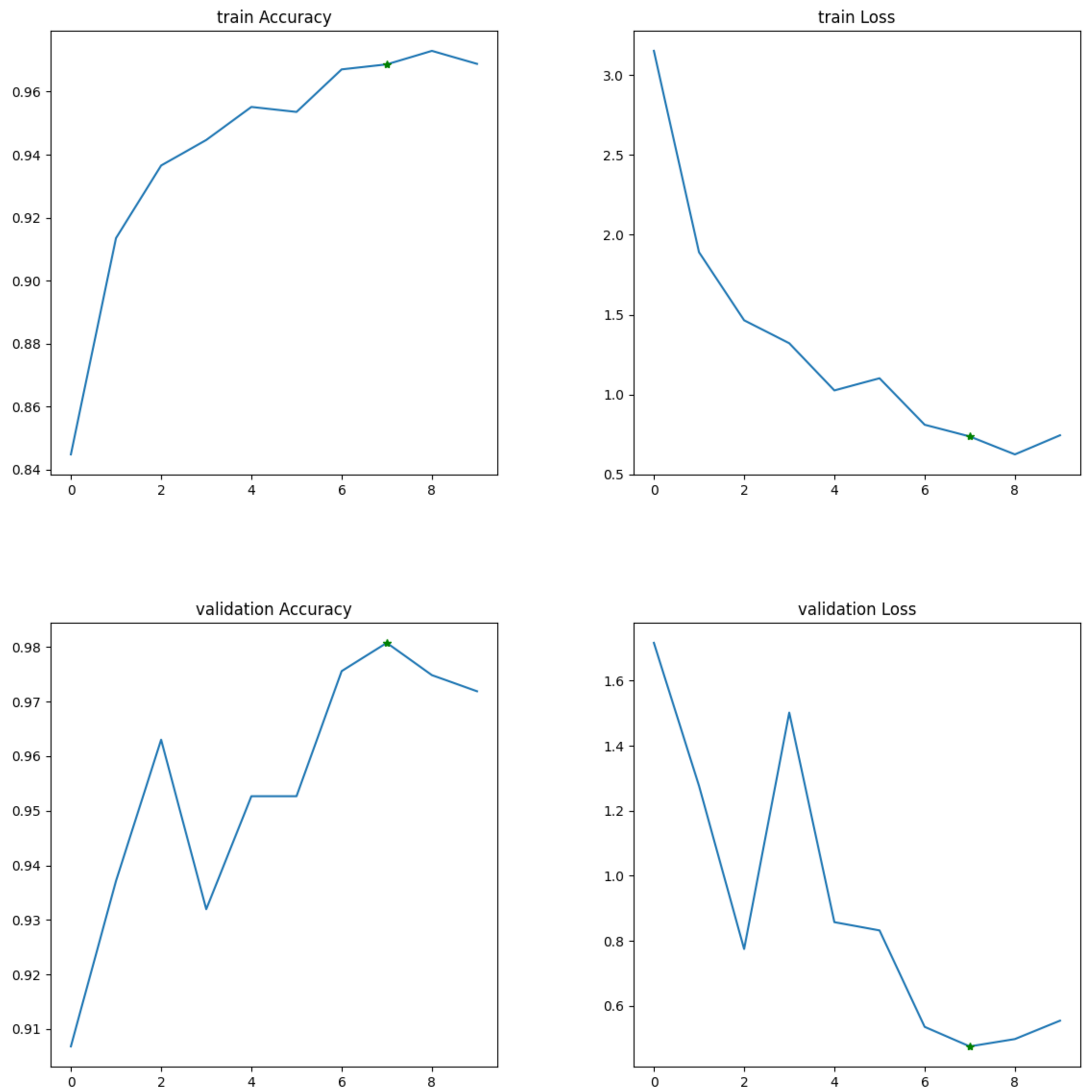
- Take an image
- Determine if the object/s in the picture are human (using Human Detection Model model)
- Crop the object/s one by one (using first model), & determine masked/non-masked/partially-masked (using second model)

4 Results

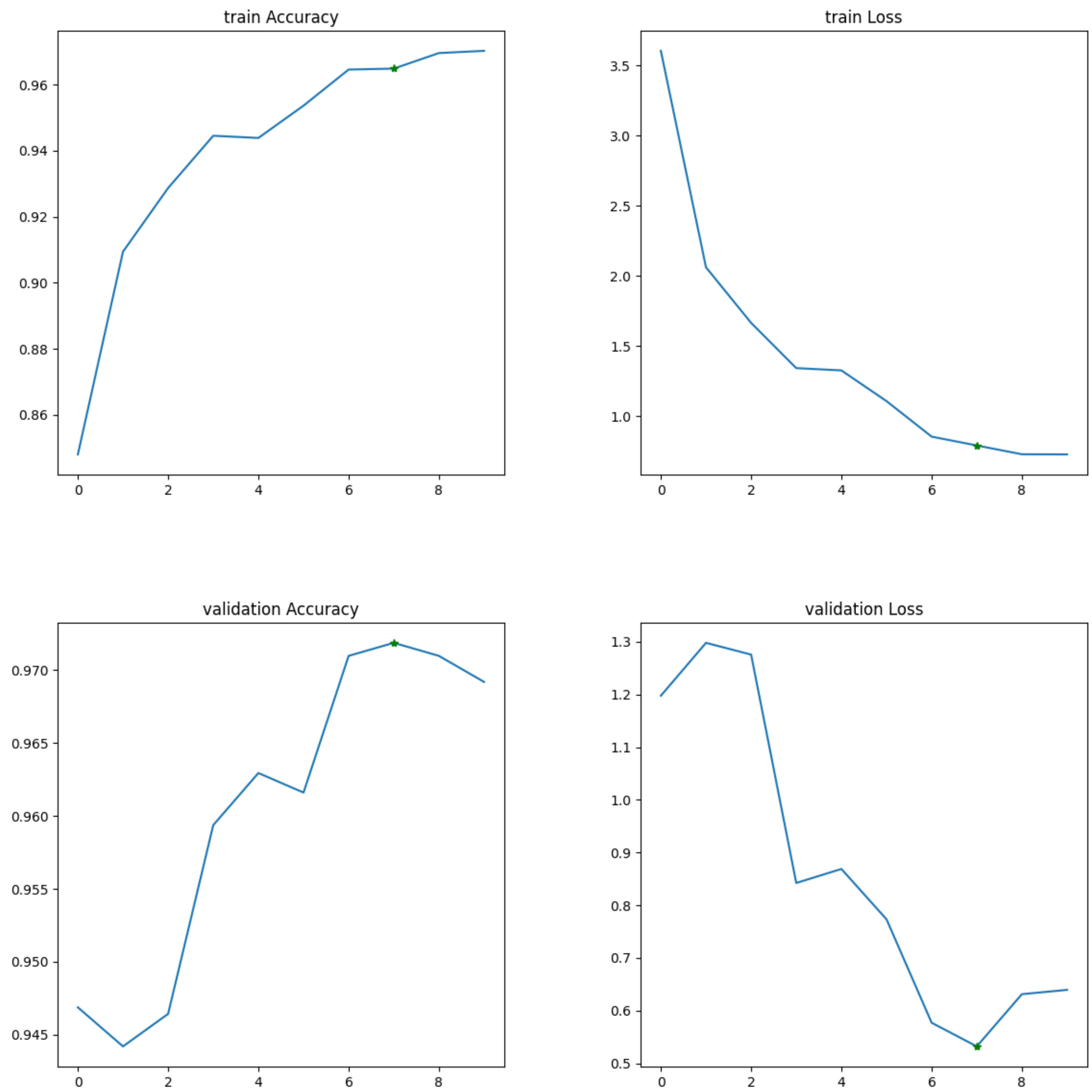
Results will be elaborated for both models: Accuracy:

- Human Detection: 97.62
- Medical Face Mask Detection: 97.62

1. Mask Detection



2. Human Detection



5 Gui

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

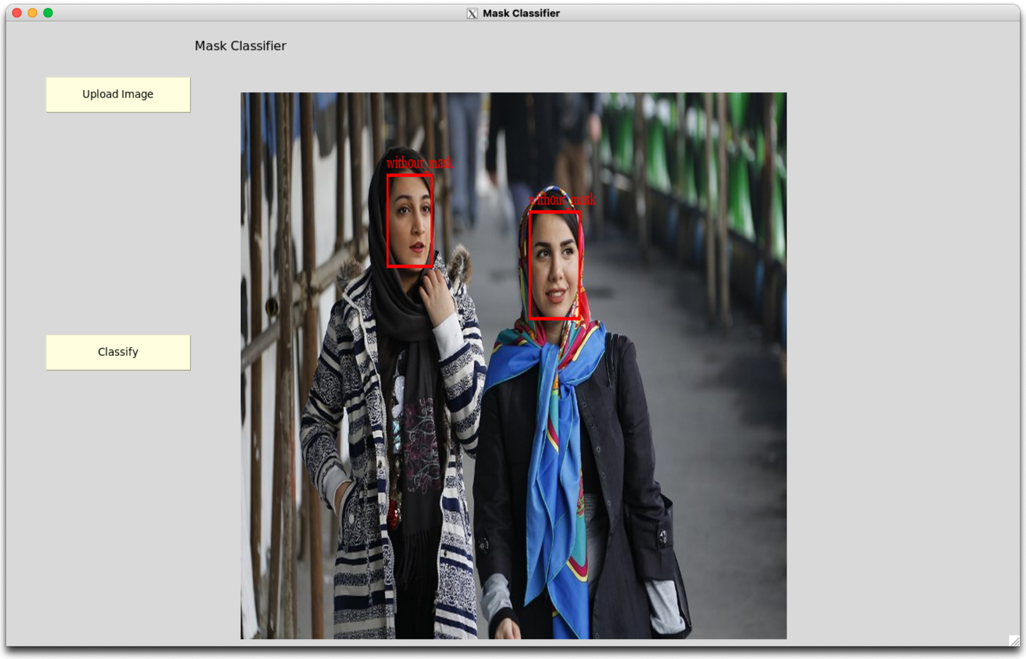


FIGURE 6: Non-Masked Sample

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 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

- 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
```

Prerequisites:

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