Medical Face Mask Detection

M.Sc. Final Project

Amir M. Dahan

Supervised by Prof. Margarita Osadchy

Department of Computer Science
University Of Haifa

February 2022

Outline

Project 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.

Expected Achievements

Contents:

1. Take both datasets and use augmentation to improve pictures.

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.

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

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), and run on an unseen test images

The Data

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.

Transformations

- 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

The Face Mask Detection Dataset

Kaggle's face-mask-detection Dataset, Comprised of the following three classes:

- Face with mask (2997)
- Face without mask (2997)
- Mask worn incorrectly (2997)

The Face Mask Detection Dataset

A custom superset of Kaggle's face-mask-detection & natural-images, Comprised of the following classes:

- airplane (727)
- car (968)
- cat (885)
- dog (702)
- flower (843)
- fruit (1000)
- mask_weared_incorrect (1000)
- motorbike (1000)
- with_mask (1000)
- without_mask (1000)

Architecture

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

- CNN, no other choice for that kind of task.
- Started from a known network for image classification.
- Hyper parameters tuning

Architecture (cont.)


```
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)
  )
)
```

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

Architecture - Training

In training phase we train two models, in three steps as followed:

1. Natural Image:
   - A Pre-Trained Resnet18 Model that detects human beings (alongside other)
   - Fine tune model's top layers with training data
   - Isn't being used for final classification
   - Used for Human Model Retraining

Architecture - Training

In training phase we train two models, in three steps as followed:

1. Natural Image:
   - A Pre-Trained Resnet18 Model that detects human beings (alongside other)
   - Fine tune model's top layers with training data
   - Isn't being used for final classification
   - Used for Human Model Retraining
2. Human Model
   - Fine tune Natural Image model's top layer
   - Add three classes (with_mask, without_mask, mask_weared_incorrect)
   - Remove 'person' class in return
   - Used to determine if the object is a person

Architecture - Training

In training phase we train two models, in three steps as followed:

1. Natural Image:
   - A Pre-Trained Resnet18 Model that detects human beings (alongside other)
   - Fine tune model's top layers with training data
   - Isn't being used for final classification
   - Used for Human Model Retraining
2. Human Model
   - Fine tune Natural Image model's top layer
   - 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
   - Used for Human Model Retraining

Architecture - Training

In training phase we train two models, in three steps as followed:

1. Natural Image:
   - A Pre-Trained Resnet18 Model that detects human beings (alongside other)
   - Fine tune model's top layers with training data
   - Isn't being used for final classification
   - Used for Human Model Retraining
2. Human Model
   - Fine tune Natural Image model's top layer
   - 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
   - Used for Human Model Retraining

Implementation Note

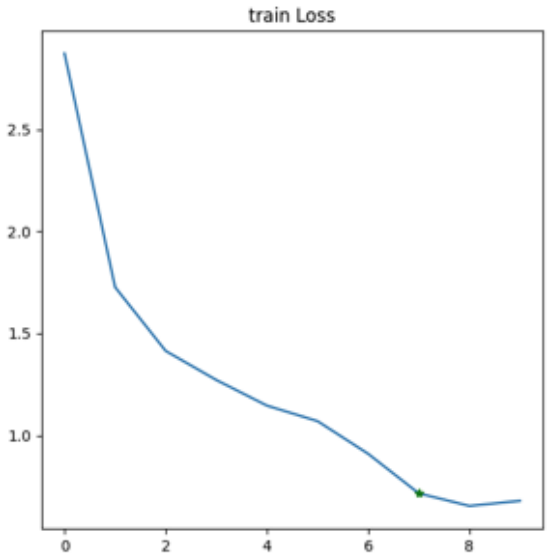
- High RAM and CUDA memory consumption (over 4GB)
- Works perfectly on a google-collab machine
- Debugged on a pre-trained resnet18
- My model is more accurate
- Code knows to automatically detect environment and use model accordingly

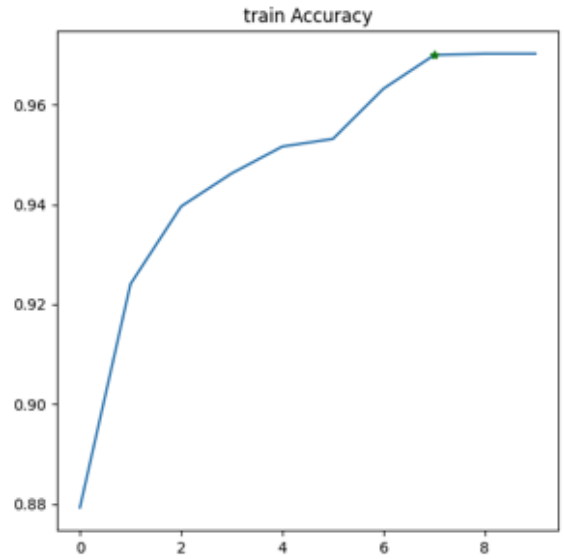
The Testing Pipeline

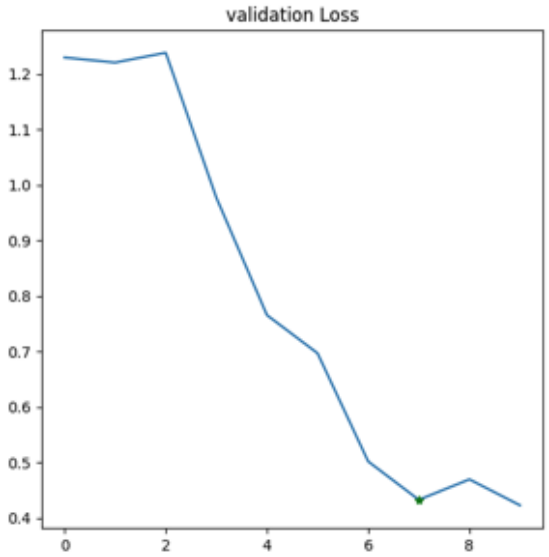
- Take an image
- Determine if the object/s in the picture are human (using Human Detection Model model)
- Crop the object/s one by one (using first model), & determine masked/non-masked/partially-masked (using second model)

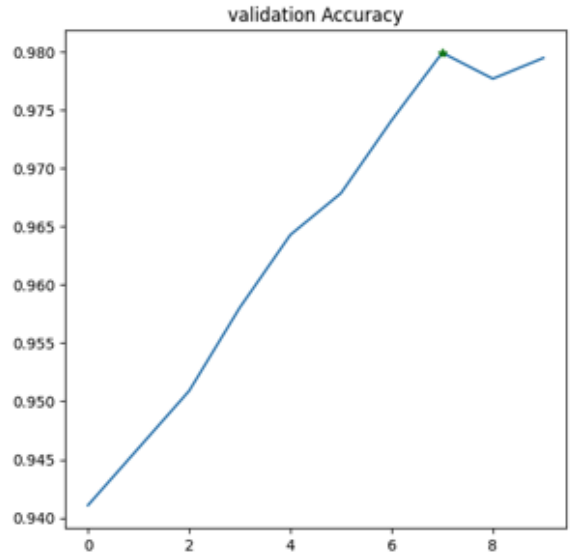
Results

Human Detection - Accuracy: 97.49%, Loss: 0.4224



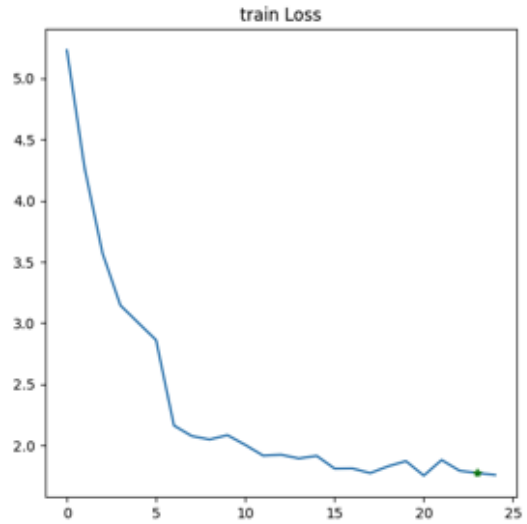


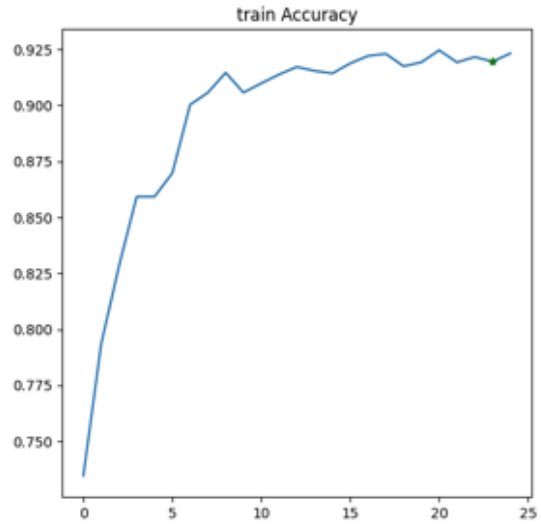


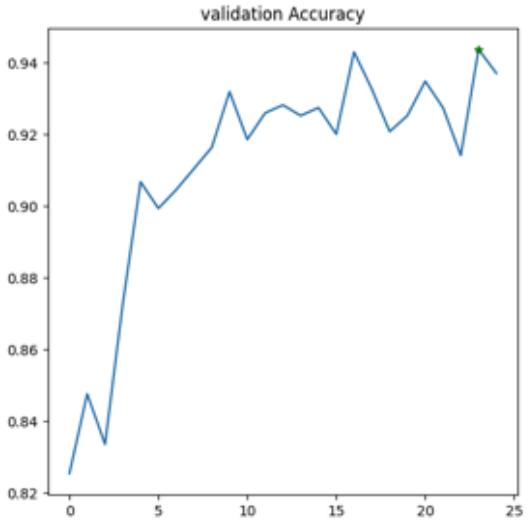


Results - Cont.

Medical Face Mask Detection - Accuracy: 93.10% Loss: 1.6244







GUI

For our convenience, a simple GUI is also implemented

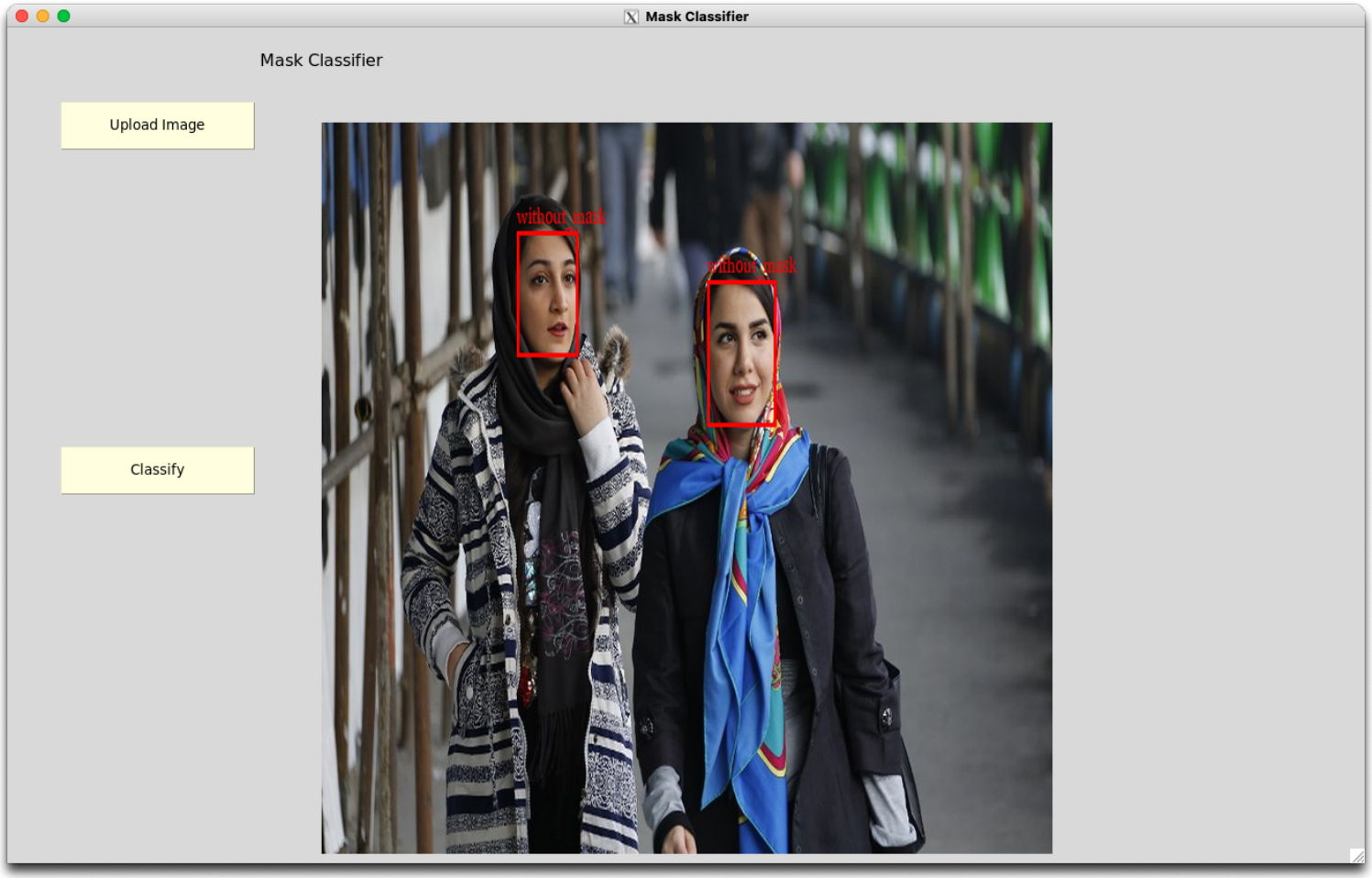


Figure: Non-Masked Sample

Video

A Nice Video tool is also added, allowing having a video as an input, and working as follows:

- Open the video with cv2 API
- Cut it into frames and save it in a list
- Classify and annotate each frame
- Present it to the user with the desired frame rate

Disclaimer

- Since x11 server over network has inherent latency, suspension is redundant here and not applied.
- There's also an option to have frames coming from a USB camera

Conclusions

An itemized thank-you:

- I'd like to thank you for the opportunity to work on this project
- I learned a lot of new things from, which I'll surely take with me throughout my career.
- Also, thanks to you, I got familiar with the area of ML (course taken back in 2017), Deep Learning (taken this year) course & Lab.
- Oh, and I didn't forget to enjoy the ride!