COVID-19 Face Mask Recognition

Shahar Rotem - 206485898
Afik Bar - 311121289

Data

The dataset contains `24,345` facial image, splitted to train and test sets.

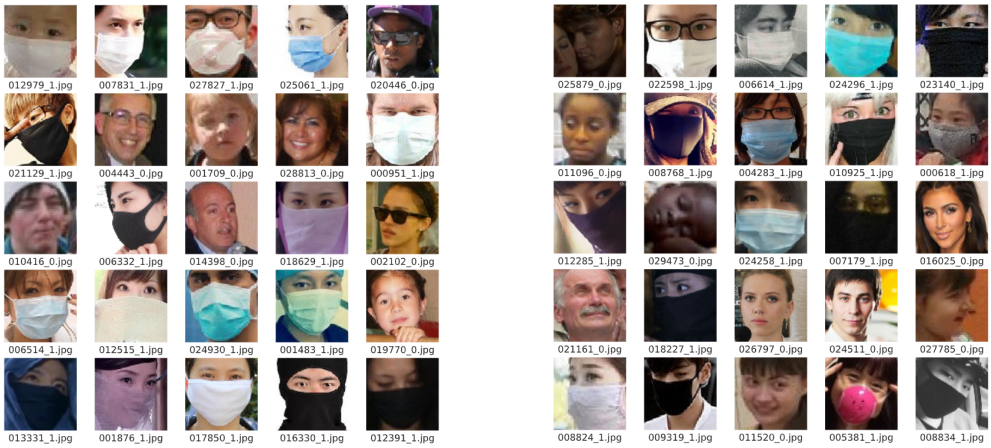

Figure 1. Train and Test Images samples, Respectively.

As we can see from this random samples, Images seems to cover a variety of ages (infants and seniors), and a multiple races.
Furthermore, we can notice that the masks "type" are not fixed to certain color or shape.
Both of the sets are balanced, as we can see from the following plots:

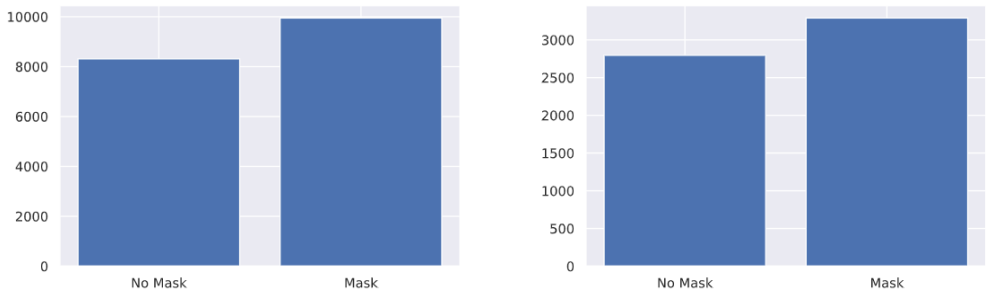

Figure 2. Train and Test Images Mask Ratios, Respectively.

Experiments

7 Layers CNN

Preprocessing

To obtain better generalization, we've used several transformations:

- Random Cropping
- Random Horizontal Flipping
- Normalizing each color channel.

Model Architecture

Our neural network consists of 7 Convolutional layers with batch normalization & ReLU as activation function, 1 average pool, and 1 fully connected layers.

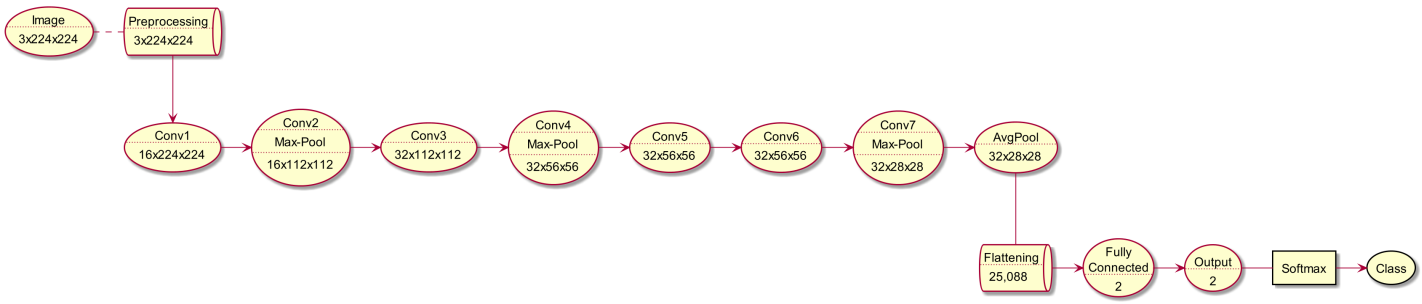

Figure 3. CNN Architecture

Loss Function

We've used Cross Entropy loss function.

Optimizer

We've tried several different optimizers, with different configurations, and found out that Adam-Weighted performed slightly better, with default decay & learning rate.

Regularization

We found out that Dropout consistently preformed worse (Possibly due to Batch normalization).
Our optimizer utilizes Weight decay, which limits the learned change in weights for each batch, we've used the default value of `0.01`.

Convergence

We were able to reach a F1-Score of `96.1` on the test-set.

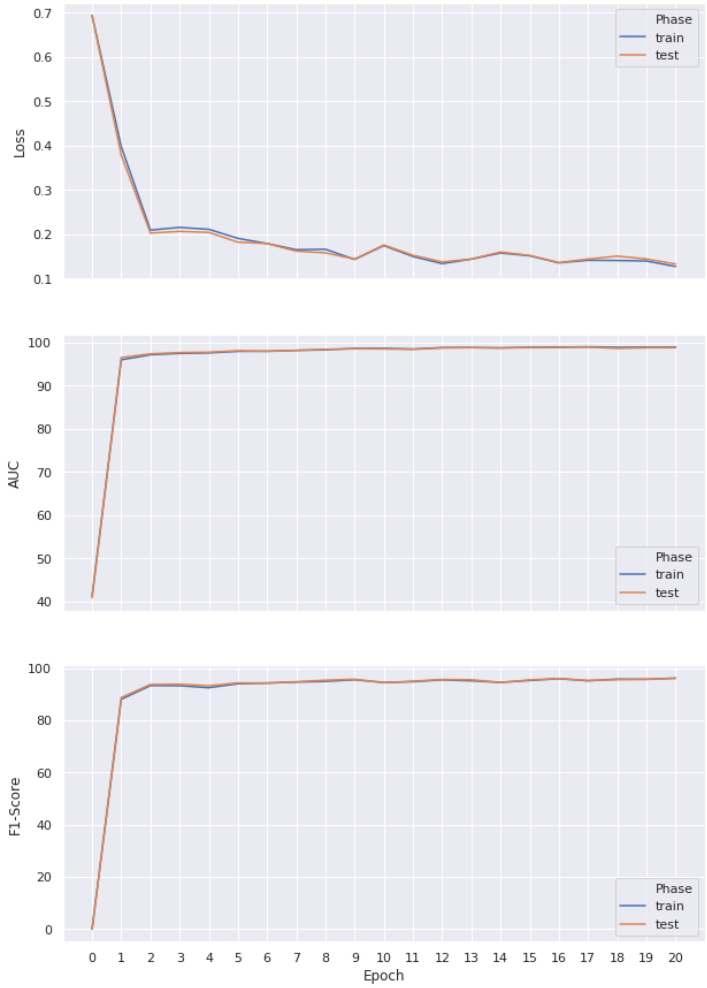

Figure 4. 7 Layers CNN Convergence

Conclusions

We found out that architecture changes had slight improvements over the data, and most of the improvements resulted from data augmentation.

MobileNet

Preprocessing

To obtain better generalization, we've used several transformations:

- Random Cropping
- Random Horizontal Flipping
- Normalizing each color channel.

Model Architecture

We've used MobileNet V3 (https://arxiv.org/pdf/1905.02244.pdf).

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

Figure 5. Specification for MobileNetV3

Loss Function

We've used Cross Entropy loss function.

Optimizer

We've used Adam-Weighted Optimizer, along with Learning Rate Scheduler. Our initial learning rate was `0.001`, and scheduler gamma is `0.1`.

Regularization

We found out that Dropout didn't affect results at all (Possibly due to Batch normalization).
Our optimizer utilizes Weight decay, which limits the learned change in weights for each batch, we've used the default value of `0.01`.

Convergence

We were able to reach an accuracy of `97.95` on the test set.

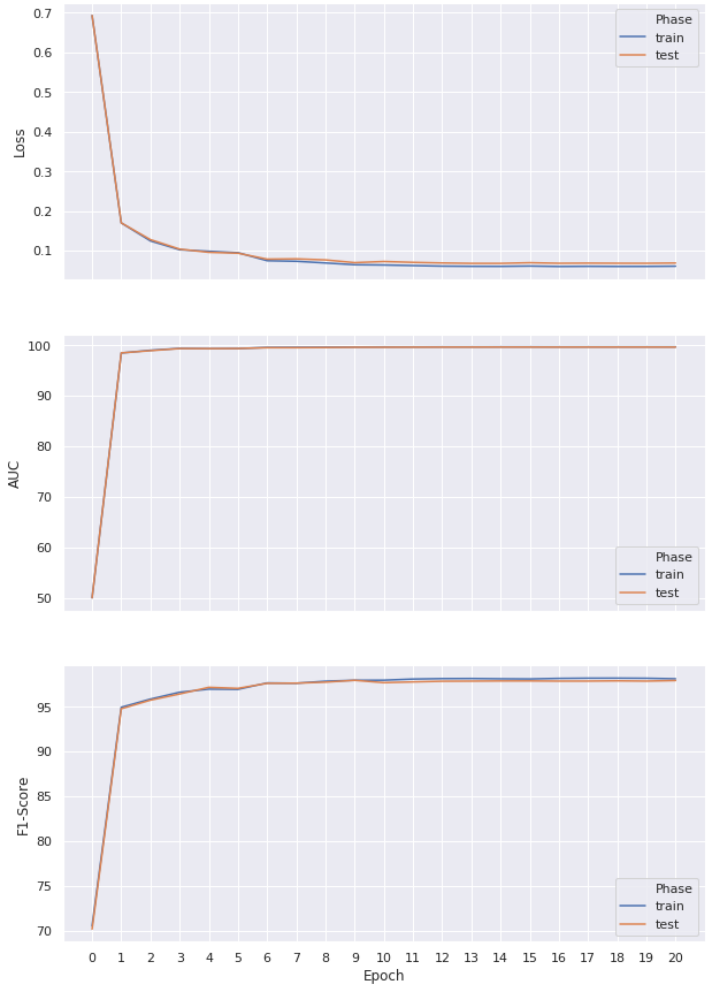

Figure 6. MobileNet Convergence

Conclusions

We've tested multiple published models, however, MobileNet intrigued us the most, since it is intended to by used by low budget processing units (In smartphones), which we believe fit the task (: Side note - PyTorch has an implementation of MobileNetV2, but we've found out that V3 performed just slightly better.