Topic – Face Mask Detection with optimization

January 8, 2023

Denys Sikorskyi
ENS Paris-Saclay
dennis.sikorksyi@ens-paris-saclay.fr

Abstract

Creating an application to detect whether is person wearing a mask, doing it wrong or not doing it at all became an important matter through last years. Previously, it was important only for laboratory workers who can neglect this procedure, because their knowledge of safety or for surgeons who obviously did it much less often. However, during 2019 the world has experienced COVID virus and general virus safety became an urgent matter. Wearing a mask became obligatory in many countries to prevent the spread of the virus. Nevertheless, many people neglected the rules, so the urge for automated detection algorithm has arisen. Also, it was important to make the model small and efficient to implement it in all systems throughout the world.

1. Modeling

1.1. Classification with CNN

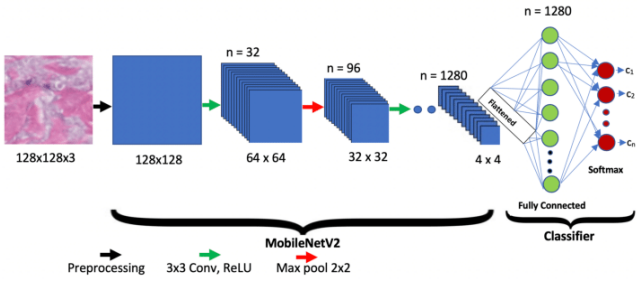

Figure 1: Architecture of the CNN model (MobileNetV2).

A convolutional deep learning network (CNN) is a type of deep learning model that is particularly suited for processing data with grid-like topology, such as images. It uses convolutional layers, which apply a set of filters to the input data to create feature maps that encode the presence of specific patterns or features at different positions in the data. These feature maps are then processed by one or more fully connected layers to make a prediction about the input data. CNNS can have multiple convolutional layers, each of which learns to detect more complex features by combining the features detected by the previous layer. In an image classification task, the fully connected layers might use the feature maps produced by the convolutional layers as input to predict the class of the input image.
Some recent developments in CNNs include: residual connections, which allow the networks to learn residual functions to bypass one or more layers; inception blocks, which use a combination of filters to learn different scales of features from the input data; depthwise separable convolutions, which apply a single filter to each input channel and can reduce the number of parameters in the network; transposed convolutions, which are used for upsampling and are the inverse of standard convolutions.

1.2. Datasets: Face Mask Detection

The dataset is taken from the Kaggle website – Face Mask Detection[1]. It consists of three folders – mask worn incorrectly, with mask and without mask. Each folder has 2994 images, so we won't have a problem with class distribution. The pictures have different quality and size, so they will be resized them to one format.

1.3. Optimization

1.3.1 Weight pruning

Weight pruning[2] is a technique for reducing the number of parameters in a neural network by removing weights that are close to zero. This can be done before or after training and can be done iteratively by gradually increasing a threshold for the magnitude of the weights and removing those below the threshold. Weight pruning can make a neural network more efficient and improve its generalization performance. Still, it is essential to be careful not to remove too many weights, as this can degrade the network's performance.

1.3.2 Quantization

Quantization[3] is the process of reducing the precision of a neural network model's weights and activations. This can be done using various methods, including uniform

Model	Accuracy	Fine-tuned accuracy	Model size (Mb)
MobileNetV2	99.39%	99.55%	8.55
InceptionV3	99.61%	99.50%	83.24
ResNet50	99.16%	99.05%	89.77
NASNetMobile	98.89%	99.39%	16.37

Table 1: Comparison of quantitative results between the models.

quantization, where all values are quantized to the same precision, and non-uniform quantization, where different values are quantized to different precision. Quantization can reduce the size of the model and improve its efficiency, but can also slightly reduce the model's accuracy. Fine-tuning techniques can be used to mitigate this effect.

1.3.3 Weight clustering

Weight clustering is a method for reducing the number of parameters in a neural network by grouping together similar weights and replacing them with a single representative weight. This can be done using a clustering loss function, and can improve the efficiency of the network, but can also slightly degrade its performance. Weight clustering is often used in combination with other model compression techniques, such as quantization and pruning.

1.3.4 Collaborative optimization

Collaborative optimization is a technique for training deep learning models that involves applying multiple optimization techniques, such as weight pruning, weight clustering, and quantization, in sequence to achieve the best balance of model size, accuracy, and inference speed. These techniques can be combined in various ways, including sparsity preserving clustering, sparsity preserving quantization aware training, cluster preserving quantization aware training, and sparsity and cluster preserving quantization aware training. Collaborative optimization can be used to compress a machine learning model and take advantage of hardware acceleration at inference time, and can be explored through various deployment paths to find the model with the desired characteristics.

2. Training and experimental results

2.1. Models

For the modeling 4 models were utilized: MobileNetV2, InceptionV3, Resnet50 and NASNetMobile. The InceptionV3 and ResNet50 were chosen as medium sized models that definitely will perform well on the datatset and the main task was to decrease the size not really thinking about the accuracy. MobileNetV2 and NASNetMobile were chosen as models that have initially small size and the main task was optimizing them carefully concerning the accuracy of the models.

2.2. Initial training and fine-tuning

For the beginning the models were trained to their initial capabilities and understand how far can optimization can be implemented with them. Therefore, the models were trained for 30 epochs and then fine-tuned for the same amount of epochs. All of them have shown great capability of solving the task with validation accuracy around 99% on the validation dataset. The results can be seen in the Table 1, the best accuracy is for InceptionV3 without fine-tuning – 99.61% and the smallest tflite model is MobileNetV2 which is around 8.5 Mbytes. The training plots can be seen in Appendix A.

2.3. Weight pruning with quantization

Next, the models were trained using weight pruning with post-training quantization. The sparsity of 20% was chosen to save the accuracy. The best model in terms of size to accuracy balance is MobileNetV2 with size of gziped tflite model of 2059408 bytes and the accuracy of 98.94%.

2.4. Weight clustering with quantization

The next step was training with weight clustering. The number of cluster was 16 to preserve the accuracy. After the training the quantization was applied. Once again, the best model was MobileNetV2 with size of 1110491 bytes and 96.6% accuracy.

2.5. Collaborative Optimization

Finally, the collaborative optimization was performed. Namely, sparsity and clustering preserving quantization

	WP with Q		WC with Q		Collaborative Optimization			
Model	Size(Mb)	Acc	Size(Mb)	Acc	QAT Size (Mb)	Acc	PCQAT Size (Mb)	Acc
MobileNetV2	2.06	98.94%	1.11	96.66%	2.02	99.39%	1.24	96.88%
InceptionV3	18.25	99.27%	5.46	98.81%	18.03	99.33%	9.46	97.83%
ResNet50	19.51	98.61%	6.84	86.85%	17.94	99.16%	9.21	97.38%
NASNetMobile	3.91	94.87%	1.54	80.38%	3.84	99.11%	1.79	96.54%

Table 2: Size and accuracy results for models and different optimization methods.

training. The operations performed in the next order: fine-tuning with pruning, clustering preserving sparsity training and finally the two models can be obtained. Next, the Quantization aware training (QAT) is performed and Sparsity preserving quantization aware training (PCQAT). Last time, the best model was MobileNetV2 with PCQAT size of 1236789 bytes and the accuracy of 96.88% and QAT of 2017440 bytes and 99.39% accuracy. The complete results can be seen in the Table 2.

3. Conclusion

During the work, it became obvious that the Face Mask Detection Task is quite solvable for all of the models. All of them yielded perfect results with accuracy on validation set higher than 99%. The top performer in pre optimization training was InceptionV3 with fantastic 99.61% of accuracy. The training for all models were fast, all of them achieved high accuracy in short time.

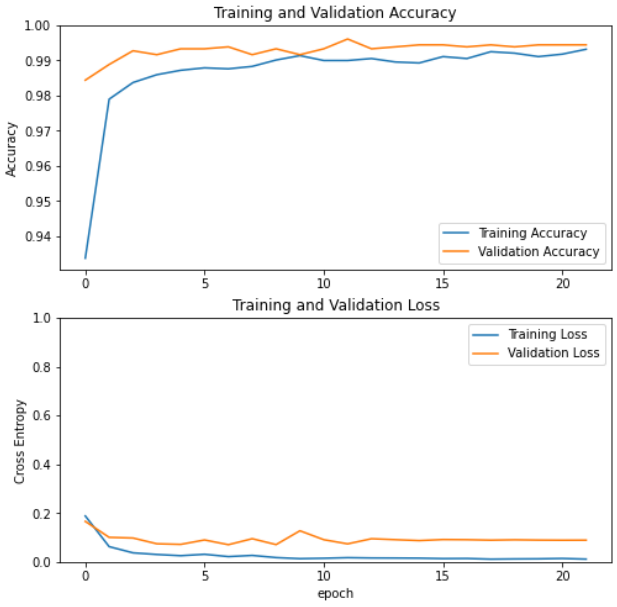

Figure 2: InceptionV3 Accuracy and Loss during training.

Next, during the optimization part we have the clear winner, the MobileNetV2 which got the smallest model (Weight Clustering with Quantization) – 1.11 Mb, but considerably worse accuracy – 96.66%. And the best model in terms accuracy was again MobileNetV2 – 99.39% with size of 2.02 Mb – the fifth smallest among all models. The InceptionV3 model has shown the best accuracy results during all optimization methods, but performing slightly worse during QAT than MobileNetV2. Other large model – ResNet50 has yielded similar results as InceptionV3, but failed Weight Clustering. The NASNetMobile wasn't able to hold high accuracy, because of small number of parameters, despite being bigger than MobileNetV2. In terms of methods, the Weight Clustering is the one who compresses models very effectively (by 87% approximately), but doesn't preserve the accuracy as well as other methods and can fail in some cases. The PCQAT is better in preserving accuracy, but worse in compressing the model (by 84% approximately). Both Weight Pruning and QAT have shown good results in terms of accuracy and worst in decreasing a size of the model, but anyway showing great drop in size by around 75%.

4. Possible improvements and prospects

During the training, it became obvious that all of models have more than enough capabilities of achieving high accuracy. One of the possible way to get smaller models than were presented is to just implement smaller models. Other way is to create simple CNN models by hand with some advanced blocks implemented by hands ensuring small size.

Also, it possible to use other optimization methods. During the training Weight Clustering suppressed the model better than other, so it can be a good idea to combine it with other methods and use sparsity preserving clustering training to combine the methods. The way of utilizing quantization can be changed and applied not only after the training, but during it, so some methods like clustering preserving quantization or sparsity preserving quantization methods can be tested.

References

[1] Kaggle Face Mask Detection website

[2] George Retsinas, Athena Elafrou, Georgios Goumas, Petros Maragos. WEIGHT PRUNING VIA ADAPTIVE SPARSITY LOSS, pages 1–2

[3] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, Tijmen Blankevoort: A White Paper on Neural Network Quantization, pages 2-4

[4] Mimi Zhang Weighted Clustering Ensemble: A Review, pages 1–3

[5] Collaborative Optimization

Appendix

A. Additional quantitative results

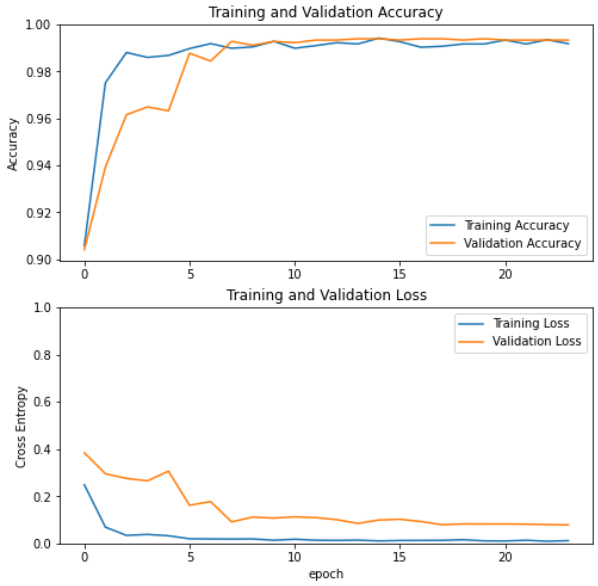

Figure 3: MobileNetV2 Accuracy and Loss during training.

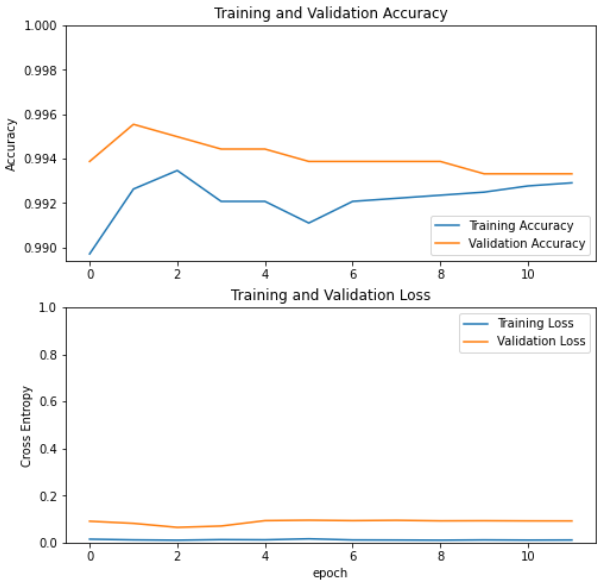

Figure 4: MobileNetV2 Accuracy and Loss during fine-tuning.

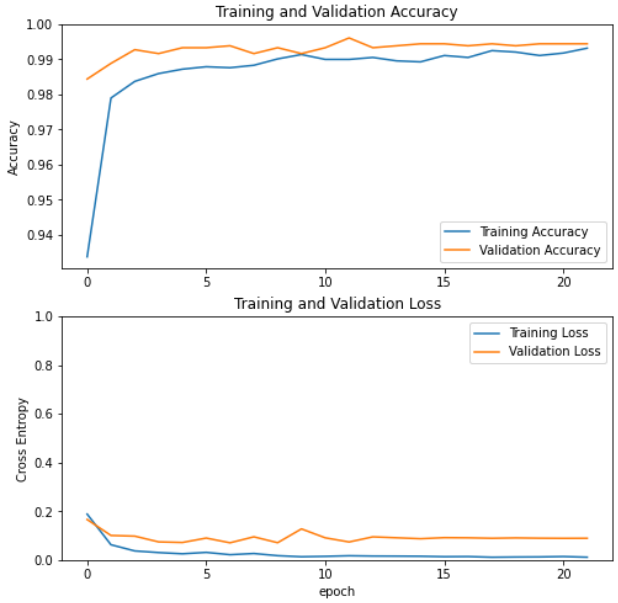

Figure 5: InceptionV3 Accuracy and Loss during training.

Figure 6: InceptionV3 Accuracy and Loss during fine-tuning.

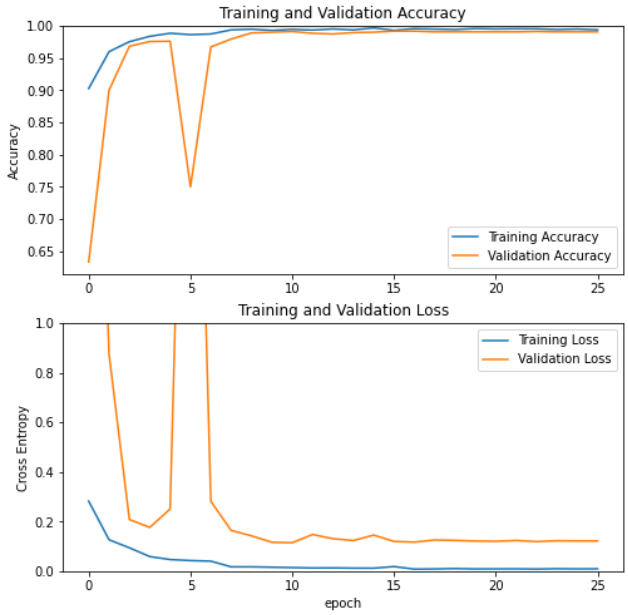

Figure 7: ResNet50 Accuracy and Loss during training.

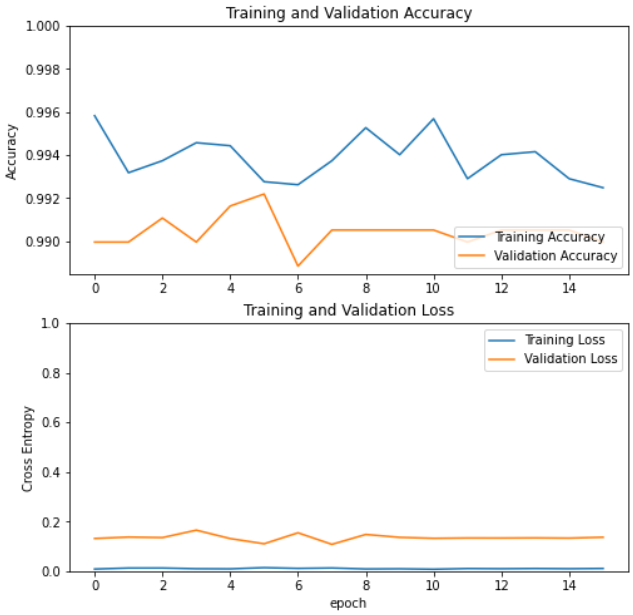

Figure 8: ResNet50 Accuracy and Loss during fine-tuning.

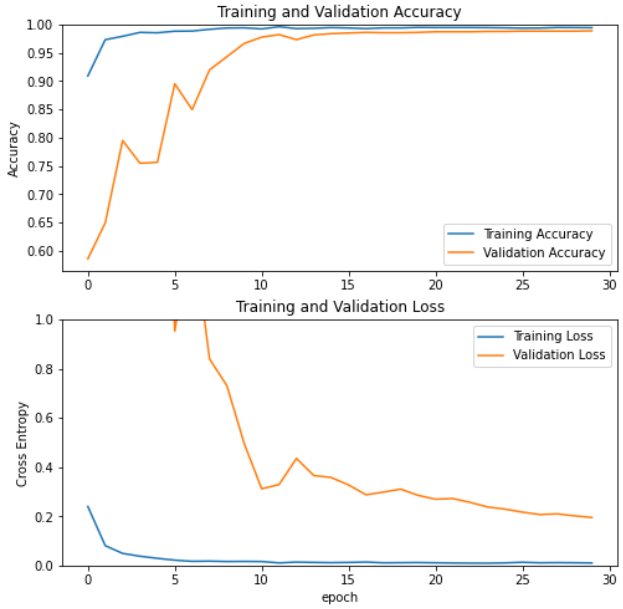

Figure 9: NASNetMobile Accuracy and Loss during training.

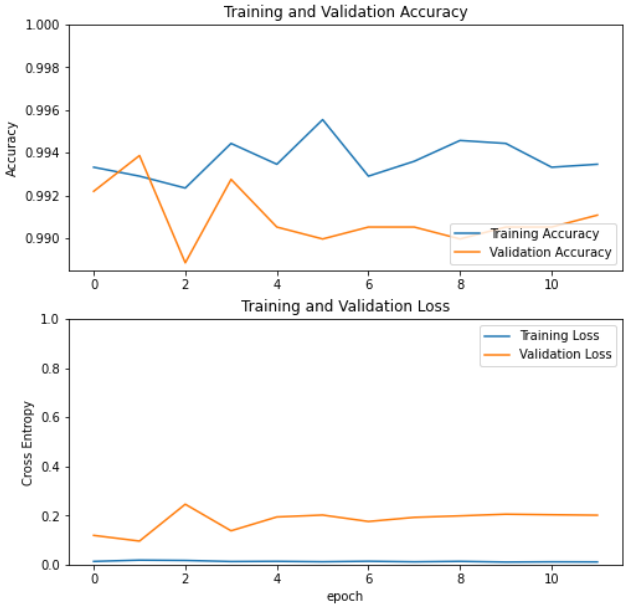

Figure 10: NASNetMobile Accuracy and Loss during fine-tuning.