Journal of Industrial Information
Technology and Application Vol.5 No.2 *ISSN(Online): 2586-0852*

Real-time Face and Landmark Localization for Mobile Applications

Myoung-Kyu Sohn*, Sang-Heon Lee, Hyunduk Kim
Division of Automotive Technology, DGIST, Daegu, Republic of Korea

Abstract. Deep learning has been applied in many areas to solve pattern recognition problems. These methods have made many advances and have shown considerable potential. In particular, it exhibits promising performance in the computer vision field such as object detection and recognition with the CNN (Convolutional Neural Network). To achieve higher accuracy, the network has been deeper and complex. Thus, the system has to process the network with the help of the GPU for inference within a reasonable amount of time. In real-world applications, many devices have some limitations such as the inability to use GPUs. In this paper, we build a deep network for face and landmark localization and demonstrate how to convert this network that works well on PC with GPU to work on a mobile platform without GPU. And, in this conversion, we propose an optimization method to enable real-time operation on mobile and show the experiment results.

Keywords; deep learning, real-time face detection; mobile application, convolution network

1. Introduction

Recently, many deep learning models that operate in real-time have emerged. In many cases, complex models have been proposed for desktops using GPUs [1,2,3]. Also, simplified networks for operation on a mobile device that does not use a GPU are being developed [4,5]. However, in reality, making it work well in real-time on a mobile platform comes with various difficulties. In this paper, we will investigate how to convert a deep learning model running in real-time using GPUs on a desktop to an optimized model on mobile. Figure 1 shows an overall procedure that converts a model running on a desktop to a model for mobile. In general, a deep learning system is built by learning the parameters of the system using a GPU. After that, the generated model is converted to match the characteristics of the mobile platform to be deployed. The optimization for real-time operation on the mobile can be done at the stage of training a

* Corresponding author: smk@dgist.ac.kr

Received: Feb. 9, 2021; Revised: Mar. 5, 2021; Accepted: Jun. 30, 2021

model, converting the model and testing it on the mobile, and repeating this to generate a model optimized for the final mobile platform. In this paper, we optimize the deep learning model[6,7] that detects faces and five major landmarks on the faces for running on a mobile platform and show experimental results by the optimization.

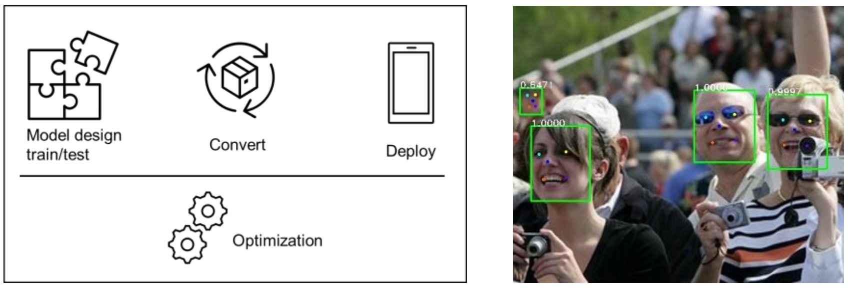

Figure 1. The overall process of building a deep learning model in a mobile platform (left), a sample of the output of face and landmark localization system (right).

2. Proposed Method

To optimize our model on a mobile platform, three approaches are applied. The first is reducing the number of parameters of the model[8,9]. The second is reducing the number of calculations during running the model. And the last is reducing the variable size of the model.

Reducing the number of parameters means changing the model itself. We try to reduce the size of the model while keeping the overall model architecture. Here we apply the width multiplier. This is to reduce the number of output layers in the convolution operation. In this case, because the model itself has changed, it needs to be trained again. The width multiplier α is a value that determines how much the network will be reduced. If it is equal to 1, it becomes the baseline model, and if it is smaller than 1, it becomes the scale-down model. Figure 2 illustrates the model reduction method by the above method. The second is to reduce the number of calculations during actual inference with the model that has already been learned, which is possible by reducing the size of the input. With this approach, the model that has already been trained can be used as it is, without changing the model, i.e., without retraining. The last method is to optimize the model when converting it to a model for the mobile platform. In general, the parameters of the trained model are stored as 32-bit variables in PC but can be reduced to 16-bit or 8-bit to optimize for mobile. Such conversion can be applied differently depending on the operation used by the model and the deep learning tool. In applying the above optimization, it is important to note that there is a trade-off between inference time and detection accuracy.

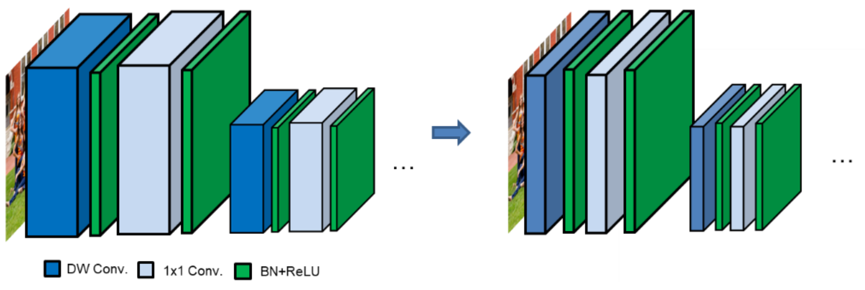

Figure 2. Scale-down model by width multiplier

3. Experiments

The experiment was conducted in the following environment. For desktop(PC), the CPU of Intel Xeon 2.6GHz and the GPU of Nvidia Titan V with 5120 CUDA Core was used. For mobile, Samsung Galaxy S10 with Exynos 9820 was used. We also run the model on a single CPU core for inference. Tensorflow[10] version 2.3 was used as a tool for deep learning.

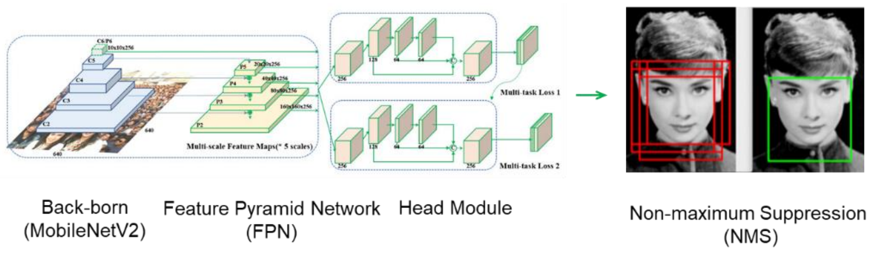
Multi-scale Feature Maps(* 5 scales)
Multi-task Loss 1
Multi-task Loss 2
Back-born (MobileNetV2)
Feature Pyramid Network (FPN)
Head Module
Non-maximum Suppression (NMS)

Figure 3. Blocks of face and landmark localization system.

The baseline model[6] was analyzed before applying the optimization. Figure 3 depicts the processing modules of the system used for face and landmark detection. Table 1 shows the step-by-step processing time for each block running on the desktop. The total inference time shows that using a GPU is about 10 times faster than using a GPU. The difference is the greatest in backbone networks that mainly perform convolution operations.

Table 1. Inference time(ms) in each module of the system running on PC. (Back-born: MobileNetV2, FPN: Feature Pyramid Network, NMS: Non-Maximum Suppression)

	Back-born	FPN	Header	NMS	Total
with GPU	3.50	1.48	1.37	9.64	15.99
CPU only	105.40	1.72	12.32	43.47	162.90

Optimization was performed by conversion optimization and model optimization. In conversion optimization, the model was converted using the dynamic optimization method provided by Tensorflow. The size of the model stored by this conversion is reduced by 4 times compared to before optimization. Table 2 shows the inference time when executed on mobile without optimization and when executed on mobile after applying optimization.

In model optimization, two methods were performed: a method of reducing the model by using a width multiplier and a method of reducing the size of an image input to the system. All experiments were conducted on CPU, and dynamic optimization conversion was applied in all mobile cases. Table 3 shows the experimental results for this. In the case of using the scale-down model, it was more effective on the mobile platform than on the PC in terms of inference time. Finally, the inference time on mobile was reduced from 350ms to 38ms after all optimizations.

There are trade-offs between detection accuracy and inference time when applying optimization for mobile deployment. Table 4 shows the detection accuracy according to each optimization method. The smaller the network and the smaller the input size, the lower the detection accuracy. Therefore, it is noted that optimizing the model within the accuracy allowed in target applications is required. Fig. 4 shows the sample of the resulting image in our target application and shows that even the most optimized model works well in real-time.

Table 2. Inference time(ms) according to the optimization of model conversion.

	No Optimization	Dynamic Optimization
MobileNetV2	471.00	350.00

Table 3. Inference time(ms) according to the optimization of the model (width multiplier: 1.0 and 0.35).

Model / Platform	PC		Mobile	
Input size	640	320	640	320
MobileNetV2_1.0	162.90	53.97	350	96
MobileNetV2_0.35	92.95	31.97	138	38

Table 4. Mean AP(%) in face detection on Widerface dataset (IoU = 0.5).

Model / Dataset	Easy		Medium		Hard	
Input size	640	320	640	320	640	320
MobileNetV2_1.0	0.91	0.87	0.86	0.83	0.55	0.54
MobileNetV2_0.35	0.89	0.84	0.83	0.80	0.51	0.50

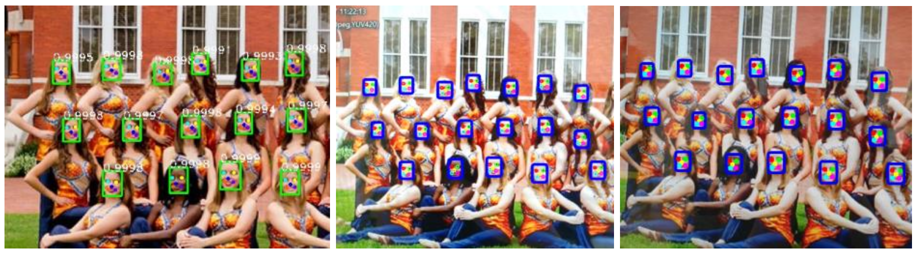

Figure 4. Samples of the face and landmark localization running on PC or mobile (first: PC, second: mobile with the non-optimized model, third: mobile with the optimized model).

4. Conclusion

In this paper, we propose a method to optimize a deep learning model as a model for a mobile platform. In optimization, we investigate how to reduce the parameter of the model or reduce the number of operations. Through experiments, we examine the inference time and detection accuracy according to each optimization method. It was confirmed that the optimized model works in real-time on mobile.

Acknowledgment

This work was supported by the DGIST R&D Program of the Ministry of Science and ICT (21-IT-02)

References

[1] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[2] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[3] Romualdo D. Atibula II, Robert R. Roxas, “Automated Taxonomic Classification of Phytoplankton Using Deep Learning”, JIETA, Vol.2, no.4, pp.120-125, 2019.

[4] Ren, Shaoqing, et al. "Faster R-CNN: towards real-time object detection with region proposal networks." IEEE transactions on pattern analysis and machine intelligence 39.6 (2016): 1137-1149.

[5] Jiang, Xiaotang, et al. "MNN: A universal and efficient inference engine." arXiv preprint arXiv:2002.12418 (2020).

[6] Deng, Jiankang, et al. "Retinaface: Single-stage dense face localisation in the wild." arXiv preprint arXiv:1905.00641 (2019).

[7] Juangtak Ryu, Jong Kwan Lee, "Deep Learning-based Face Identification System", JIITA, vol.3 no.4, pp.315-317, 2019.

[8] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[9] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[10] https://www.tensorflow.org