Filter-Wise Quantization of Deep Neural Networks for IoT Devices

Hoseung Kim, Geunhye Jo, Hayun Lee, and Dongkun Shin
Sungkyunkwan University, Suwon, Korea
{ghtmd123, shurui, lhy920806, dongkun}@skku.edu

***Abstract*—Network quantization is an effective compression technique of deep neural networks (DNNs) for on-device machine learning at consumer devices. Existing layer-wise quantization techniques allocate different bitwidths to different network layers. In this paper, we propose a filter-wise quantization technique based on the differentiable neural architecture search (DNAS). We use a two-level network structure and a novel candidate generation algorithm, which can substantially prune the large search space. The effectiveness of our technique was validated with MobileNetV2 on ImageNet.**

I. Introduction

As deep neural networks (DNNs) have shown exceptional performance in various fields, many deep learning-based applications are emerging for consumer devices [1]. To achieve high accuracy, DNNs require more than millions of parameters and billions of floating-point operations (FLOPs). Therefore, it is hard to run DNNs on resource constrained consumer devices.

Several network compression techniques have been proposed to solve this problem. Quantization uses low bitwidths of weights and activations to compute DNNs, and thus significantly reduce memory and computational cost. Recently-proposed layer-wise or filter-wise quantization techniques, where a different bitwidth is used for each network layer [2] or convolution filter [3], can compress the network further than using a single bitwidth over all layers and filters. One challenge is how to determine an appropriate bitwidth of each layer or filter. Previous studies [2], [3] used a reinforcement learning (RL)-based neural architecture search (NAS), where the training and evaluation for network architecture must be performed repeatedly resulting in an enormous amount of training time.

In this paper, we propose an efficient filter-wise quantization scheme based on the differentiable neural architecture search (DNAS) [4], which considers the search space as continuous and explores it using a gradient descent technique. To use the DNAS technique for filter-wise quantization, we propose a two-level network search and an efficient candidates generation algorithm.

II. Methodology

Among various DNAS algorithms, we adopted FBNet [4] for our filter-wise quantization. For each network layer, multiple architecture candidates with different bitwidths of weights and activations (e.g., 4-bit weights and 3-bit activations) are generated. Each candidate has a sampling probability and only one candidate per layer is sampled based on the sampling probability and the selected architecture is executed in the forward propagation. To consider both the network accuracy and the execution latency on the target device, we use the following loss function of the architecture a:

$$L(a) = \texttt{CE}(\texttt{a}) \cdot \alpha \log(\texttt{LAT}(\texttt{a}))^{\beta} \quad (1)$$

CE is the cross entropy loss, and LAT is the execution latency of the network on the target device. To get the latency value during the network architecture search, a latency lookup table is pre-built in advance, which provides the latencies at different bitwidths of weights and activations. The sampling probability is continuous, so the loss is differentiable to the sampling probability. Therefore, we can train the sampling probability also while training the network.

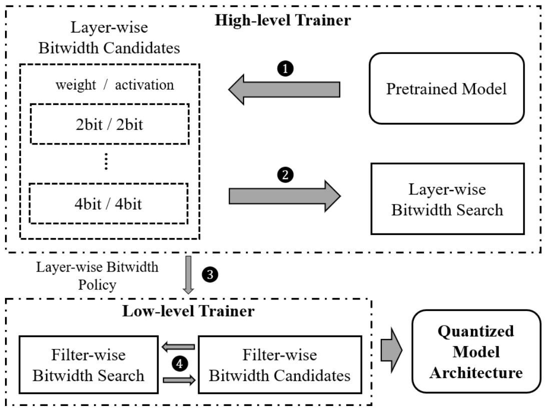
High-level Trainer
Layer-wise Bitwidth Candidates
weight / activation
2bit / 2bit
4bit / 4bit
❶
Pretrained Model
❷
Layer-wise Bitwidth Search
Layer-wise Bitwidth Policy
❸
Low-level Trainer
Filter-wise Bitwidth Search
❹
Filter-wise Bitwidth Candidates
Quantized Model Architecture

Fig. 1. The overall flow of the proposed technique.

Fig. 1 shows the entire flow of our network search scheme, which consists of a high-level trainer and a low-level trainer. With a pretrained model, the predetermined number of candidates are generated for each layer, each of which uses different bitwidths of weights and activations (❶). The high-level trainer trains the network with the candidates to get the layer-wise bitwidths (❷). After the network training, the high-level trainer gets a layer-wise bitwidth policy by sampling the candidates, and then pass it to the low-level trainer (❸). With the layer-wise bitwidth policy, the low-level trainer generates the filter-wise candidates, each of which allocates a different bitwidth

This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.IITP-2017-0-00914, Software Framework for Intelligent IoT Devices)

to each filters. At the low-level trainer, the activation bitwidth is set to be same to that of the transferred policy. The weight bitwidth of the transferred policy is used as a total budget of all the filters in a layer.

To generate filter-wise quantization candidates at the low-level trainer, the quantization sensitivity of the n-th filter in the l-th layer is defined as follows:

$$C_{l,n} = \frac{w_{l,n}^{max} - w_{l,n}^{min} + \epsilon}{(\kappa_l)^{b_{l,n}}} \tag{2}$$

$w_{l,n}^{max}$ and $w_{l,n}^{min}$ are the minimum and maximum absolute values of the target filter, respectively. ϵ is a noise value sampled from the normal distribution $\epsilon \sim \mathcal{N}(0, \sigma^2)$ for each candidate. $b_{l,n}$ is the bitwidth assigned to the target filter, and κ, called the quantization efficiency parameter, is related to the amount of change on quantization error according to the assigned bitwidth. To get the value of κ, we directly measured the quantization errors of each layer at several different bitwidths of the pretrained model and used an exponential regression with the measured values.

The low-level trainer assigns the filter-wise bitwidth of each candidate as follows: First, we define the bitwidth range of filters, b^{min} to b^{max}. Then, each filter in a candidate is assigned with b^{min} of bitwidth, and its quantization sensitivity C is calculated. Until the average bitwidth of all the filters in a candidate reaches the budget, the low-level trainer repeatedly adds 1-bit to the filter which has the largest C and its assigned bit is not larger than b^{max} while updating C of each filter. This step is performed for all candidates in each layer.

After the candidate generation, the low-level trainer trains the network by sampling candidates (❹). After training, a candidate is sampled for each layer and it becomes one of the candidates for the next training. If the same set of candidates at network layers are sampled during several times, the training has been converged and the candidates can be determined as the final quantization policy.

Our technique can find good filter-wise candidates due to the quantization sensitivity-based generation. In addition, it can effectively explore the search space with the noise value. To ensure the convergence, we reduce the value of σ, which is associated with ϵ, as the search step iterates.

III. Experiments

To compare several quantization techniques, we used MobileNetV2 on ImageNet. The original model accuracy is 71.81%. To measure the latency of the multi-bit quantized network, we run the quantized model at a PYNQ-Z2 FPGA board. We used our-implemented bit-wise operation accelerator [5] at the FPGA board.

Fig. 2 shows the trade-offs between latency and accuracy at several different quantization techniques. `Fixed` uses the same bitwidth at all layers and all filters. `Layer` is the layer-wise quantization technique. We used HAQ [2] bitwidth allocation policy. `Filter` is our filter-wise quantization technique. `Filter` shows better latency-accuracy trade-offs than

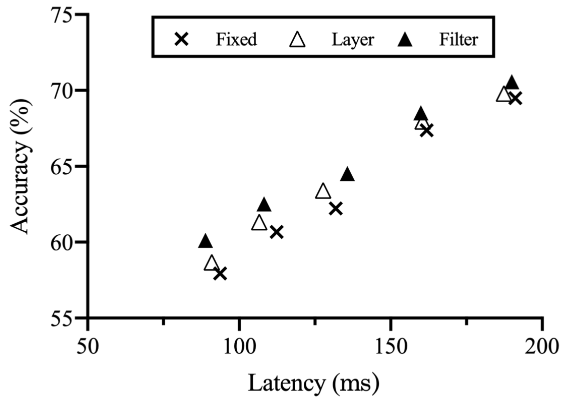

Fig. 2. Latency and accuracy comparison of different quantization methods.

TABLE I
Accuracy comparison of fixed and layer-wise value of κ

Type of κ	Activation	Weight	Accuracy (%)
fixed	3.77	4.35	59.54
layer-wise			**60.13**
fixed	3.93	5.86	64.17
layer-wise			**64.53**
fixed	6	5.78	70.32
layer-wise			**70.58**

`Layer` and `Fixed`. The difference is more significant at lower latency models.

Table I shows the effect of layer-wise κ. Whereas different κ values were used at different layers in `layer-wise` technique, `fixed` used the same value of κ at all layers. The layer-wise κ shows a higher accuracy.

IV. Conclusion

We proposed a filter-wise quantization technique based on DNAS. The large search space of the filter-wise quantization can make the network training difficult and can require a long training time. To effectively explore the search space, we proposed a two-level searching scheme and an efficient filter-wise candidate generation method.

References

[1] P. Sundaravadivel, K. Kesavan, L. Kesavan, S. P. Mohanty, and E. Kougianos, “Smart-log: A deep-learning based automated nutrition monitoring system in the IoT,” *IEEE Trans. Consum. Electron*, vol. 64, no. 3, pp. 390–398, Aug. 2018.

[2] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, “HAQ: hardware-aware automated quantization,” *CoRR*, vol. abs/1811.08886, 2018. [Online]. Available: http://arxiv.org/abs/1811.08886

[3] Q. Lou, F. Guo, L. Liu, M. Kim, L. Jiang, ”Autoq: Automated kernel-wise neural network quantization.” *arXiv preprint arXiv:1902.05690*, 2019.

[4] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, “FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019.

[5] Y. Umuroglu, L. Rasnayake, and M. Sjalander, “Bismo: A scalable bit-serial matrix multiplication overlay for reconfigurable computing,” in *Field Programmable Logic and Applications (FPL), 2018 28th International Conference on*, ser. FPL ’18, 2018.