Neural Computing and Applications
https://doi.org/10.1007/s00521-021-06752-7

S.I.: 2020 INDIA INTL. CONGRESS ON COMPUTATIONAL INTELLIGENCE

Greedy search algorithm for partial quantization of convolutional neural networks inspired by submodular optimization

Satoki Tsuji[1,3] · Fuyuka Yamada[1,3] · Hiroshi Kawaguchi[2] · Atsuki Inoue[2] · Yasufumi Sakai[3]

Received: 2 April 2021 / Accepted: 10 November 2021

Abstract
Recent results of studies have indicated that neural network quantization effects on inference accuracy vary among layers. Therefore, partial quantization and mixed precision quantization have been studied for neural network accelerators with multi-precision designs. However, these quantization methods typically require network training, which entails a high computational cost because of the exponentially increasing search space with respect to the number of layers N. However, an insufficient search leads to a significant degradation of inference accuracy. For partial quantization, this paper presents a greedy search algorithm that can derive practical combinations of quantization layers without re-training; notably, the proposed method exhibits particularly low computational complexity $O(N^2)$. The proposed greedy search algorithm achieved $4.2\times$ model size compression with only 0.03% accuracy degradation in ResNet50 and $2.5\times$ compression with +0.015% accuracy gain in Xception. The computational cost of the greedy search algorithm was only 2.6 hours for a single V100 GPU in the case of MobileNetV2 quantization for ImageNet classification. Furthermore, we accelerated the proposed algorithm to computational complexity $O(N)$ and achieved $4.15\times$ model size compression with only 0.072% accuracy degradation in ResNet50.

Keywords Machine learning · Deep neural networks · Quantization · Neural architecture search · Submodular optimization

1 Introduction

High deep neural network inference performances with safety and low latency are expected to be achieved using edge devices such as sensors and smartphones (edge computing). Security robustness and real-time operation can be improved considerably by reducing data traffic to and from computationally powerful cloud servers. However, training and inference entail high computational costs. The latest models are substantially deep. Therefore, one must employ sufficient computational resources, such as a parallel GPU, on the cloud. To accommodate these limitations, neural networks must be compressed such that they can run on edge devices with low computational power. Moreover, for the operation of practical edge AI systems, multiple deep neural networks must be installed on edge devices to expand their functionality. Accordingly, each model size should be sufficiently small.

Several model compression techniques have been proposed to reduce neural network computations: pruning [1] to reduce the number of parameters, knowledge distillation

✉ Satoki Tsuji
tsuji.satoki@fujitsu.com

Fuyuka Yamada
yamada.fuyuka@fujitsu.com

Hiroshi Kawaguchi
kawapy@godzilla.kobe-u.ac.jp

Atsuki Inoue
ainoue@godzilla.kobe-u.ac.jp

Yasufumi Sakai
sakaiyasufumi@fujitsu.com

1 Graduate School of System Informatics, Kobe University, 1-1 Rokkoudai, Nada Ward, Kobe City, Hyogo Prefecture 6578501, Japan

2 Graduate School of Science, Technology and Innovation, Kobe University, 1-1 Rokkoudai, Nada Ward, Kobe City, Hyogo Prefecture 6578501, Japan

3 Fujitsu Research, Fujitsu Ltd., 4-1-1 Kamikodanaka, Nakahara Ward, Kawasaki City, Kanagawa Prefecture 2118588, Japan

[2] to inherit the knowledge of the larger teacher model to the smaller student model, and quantization [3] to entail low bit precision. Among these techniques, we specifically examine quantization because it is independent of the network architecture. Quantization can reduce the model size, latency, memory bandwidth, and power consumption; however, it also degrades the inference accuracy. To improve this trade-off, partial and mixed precision quantization, which changes the bit width depending on a part of the model, such as layers and blocks, are applicable. Conventional studies of neural network quantization uniformly quantized an entire model with the same bit width [4, 5]. However, the effects of quantization on inference accuracy differ for each layer or block because of differences in their distributions of values. Consequently, one can quantize models efficiently by adjusting the bit widths for the respective layers. Furthermore, the development of deep-learning-specific hardware such as the tensor core [6] and tensor processing unit [7], which support multiple bit widths, has accelerated in recent years. Therefore, we propose a greedy algorithm to search for per-layer bit width allocations that can maintain high inference accuracy for partial quantization. Therefore, we intend to improve the trade-off related to the increase in quantization noise and accuracy degradation using a greedy search for more efficient combinations of layers for quantization.

2 Related work

In recent years, growing interest has arisen in hardware-aware architectures of neural networks. As an attempt to automate the design of these architectures, several studies [8, 9] that search efficient models specific to mobile devices has been proposed, which are based on neural architecture search (NAS) [10]. Actually, the NAS framework has been applied to mixed precision quantization [11] and a variety of bit-width optimization methods for each layer have been focused on. Furthermore, a single-path simplified supernet relaxed the hardness of training in NAS while supporting mixed precision quantization [12]. Examples include a model compression technique that determines the bit widths for convolution and fully connected layers by human heuristics [13], a measurement method [14] to estimate the effects of parameter quantization errors on individual layers for finding the optimal quantization bit width, reinforcement learning-based search using hardware feedback of actual edge devices [15], quantizer parameterization and bit-width estimation by step size and dynamic range [16], and automatic selection of the relative quantization bit width of each layer based on the approximated Hessian spectrum [17]. Mixed precision quantization is available for various bit widths (e.g., 1–8bits) at each layer, but it tends to result in a high computational cost because of its exponential search space with respect to the number of layers N.

Relaxing this search space and limiting the quantization bit width to a single one is called partial quantization. Under this approach, the process is simplified to search for combinations of layers with little degradation in inference accuracy caused by quantization. With regard to partial quantization, a sensitivity analysis has been proposed, which selects layers for quantization based on pre-evaluated effects of quantization for each layer on inference accuracy as layer sensitivity [18]. Sensitivity analysis is a reasonable approach with computational complexity of $O(N)$. However, this search method may neglect the combinations of quantization layers that result in higher inference accuracy because it does not consider relative changes in sensitivity as quantization progresses. Since the quantization error of a layer accumulates in subsequent layers, the accuracy computed by quantizing only one layer is most likely to diverge from the actual layer sensitivity for multiple layers quantized.

In addition to mixed precision, post-training quantization, which avoids the training cost of recovering accuracy after quantization, has also received increasing attention. Examples of this topic include a data-free quantization that does not require fine-tuning or hyperparameter selection [19], a layer-wise optimization of step size minimizing cross-entropy loss [20], a kernel-wise quantization scheme that minimizes the mean squared error [21]. Based on this background, we propose a post-training quantization scheme using a greedy search algorithm for partial quantization. This algorithm can determine the combinations of quantization layers that result in higher inference accuracy of quantized models according to the change in sensitivity because of quantization progress with computational complexity of $O(N^2)$. An overview of the proposed method is shown in Fig. 1. We show that a greedy search can maintain the original accuracy even if almost the entire model is quantized. The specific contributions of this work are the following.

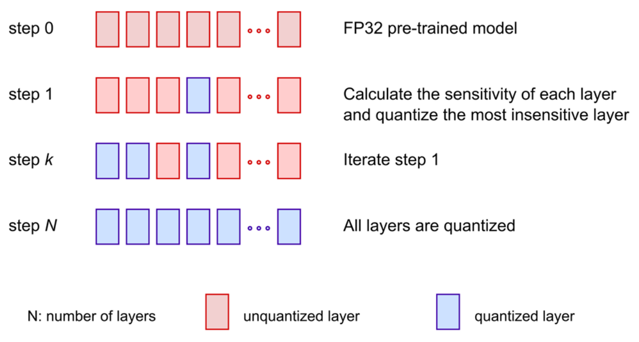

Fig. 1 Overview of Greedy search algorithm for partial quantization

Table 1 Results of quantization model and search time on ImageNet classification by latest quantization strategies

Model (Param size)	Method	Baseline	Acc	ΔAcc	Param size (MB)	Search time (V100 GPU-hours)
ResNet18 (44.59MB)			**71.21**	**+0.18**	3.98	
	DNAS [11]	71.03	69.58	-1.45	**2.11**	40
			70.66	**+0.38**	10.5	
	DQ Param [16]	70.28	70.08	-0.2	5.4	60
			69.7	**0**	11.2	
	DFQ [19]	69.7	66.3	-3.4	8.36	
	LAPQ [20]	69.7	68.8	-0.9	11.2	
	MMSE [21]	69.64	67.42	-2.22	5.62	
	AdaRound [35]	69.68	68.55	-1.13	5.57	PTQ[1]
			69.75	**-0.01**	11.4	
	Greedy search (Ours)	69.76	69.47	-0.29	9.44	**2.2**
ResNet34 (83.15MB)	SPOS [12]	75	74.6	-0.4	10.2[2]	288 (GTX 1080Ti)
			73.3	**+0.02**	25.1	
			73.03	**-0.25**	18.3	
	Greedy search (Ours)	73.28	72.96	-0.32	16.7	**9.9**
MobileNetV2 (13.37MB)			71.47	-0.4	1.79	
	HAQ [15]	71.87	70.9	-0.97	**1.38**	96[3]
			70.59	**+0.41**	3.14	
	DQ Param [16]	70.18	69.74	-0.44	1.55	123.3
	DFQ [19]	71.7	71.2	-0.5	3.34	
	AdaRound [35]	71.72	69.25	-2.47	1.67	PTQ[1]
			72.8	**-0.03**	3.95	
	Greedy search (Ours)	72.83	**72.73**	**-0.1**	3.44	**2.6**

In the table, we abbreviate accuracy as ”Acc” and parameter as ”Param”. We show the outstanding performance figures in bold achieved by each method

[1] PTQ denotes post-training quantization and we cannot confirm the detailed search time from their paper.

[2] The parameter size of ResNet34 for SPOS is estimated value according to the BitOPs in [12].

[3] The search time of MobileNetV2 for HAQ refers to the design cost in [36]

1. For the mixed precision quantization problem, which used to cost dozens of hours or several days (for a single GPU) to optimize due to its exponentially large search space, we propose a quantization scheme that completes the search in a few hours with a greedy algorithm by relaxing it to a simple combinatorial optimization problem. Existing methods such as NAS involving network training are highly sensitive to fine-tuning and initialization, making it difficult to quantitatively estimate their large computational complexity. However, the proposed method is free from most of these opaque costs and completes the search in polynomial time $O(N^2)$ (N denotes the number of layers).
2. We evaluate the proposed method in ImageNet classification task and compare it with state-of-the-art quantization methods. The performance differences in terms of accuracy and model size of the quantized networks between the proposed method and the search methods involving large training cost are minimal. Furthermore, we reduce the accuracy degradation by approximately 0.3% to 1% compared to other post-training quantization methods. See Table 1 for more details.
3. The proposed greedy algorithm achieves more than 4× compression of the parameter size with less than 1% accuracy degradation on various convolutional neural networks, including ResNet18, ResNet34, ResNet50, and DenseNet161.

3 Approach

In this section, we detail the greedy search algorithm which has been proposed to solve the problem of partial quantization without retraining. First, we formulate a partial quantization problem into a simple combinatorial optimization problem. Next, we describe submodular optimization which is an effective approach to the combinatorial optimization problem. Subsequently, we propose a greedy search algorithm inspired by submodular optimization. Finally, we describe the quantization scale and rounding as the quantization details used in our experiments.

3.1 Submodular optimization

The partial quantization of neural networks can be regarded as an optimization problem to determine a combination of quantization layers that maximizes inference accuracy. This problem can be formulated as in Equation (1), by defining the objective function as the inference accuracy.

$$\begin{aligned} &\arg\max_{\mathbf{S}} \quad \mathrm{Acc}(\mathbf{S}) \\ &\text{subject to} \quad |\mathbf{S}| = k \quad (k = 0, 1, 2, \cdots, N) \end{aligned} \tag{1}$$

where **S** denotes the set of quantization layers and Acc(**S**) is a set function that denotes the inference accuracy when the layers **S** are quantized. k denotes the number of quantization layers, which varies from zero to N. The problem formulation in Equation (1) maintains accuracy by searching for the combinations of layers to be quantized; however, Equation (1) is not a direct formulation to simultaneously improve the model compression ratio and accuracy through quantization. For a more efficient quantization, the objective function in Equation (1) can be redefined by introducing a factor #PARAM(**S**), which denotes the number of quantized parameters, as Equation (2).

$$\begin{aligned} &\arg\max_{\mathbf{S}} \quad \mathrm{Acc}(\mathbf{S}) \times (\log \#\mathrm{PARAM}(\mathbf{S}))^{\beta} \\ &\text{subject to} \quad |\mathbf{S}| = k \quad (k = 0, 1, 2, \cdots, N) \end{aligned} \tag{2}$$

where β is a coefficient to tune the relative importance of the quantized parameters as the model compression ratio. By adjusting β, one can search for quantization layers according to the priorities of compression and accuracy. When $\beta = 0$, Equation (2) becomes a problem of maximizing accuracy without considering the number of quantized parameters as a factor related to the model size. The formulation of Equation (2) can improve the accuracy degradation and the model compression ratio through quantization. However, as even this relaxed search space 2^N exponentially increases with the number of layers N, brute force approach is extremely difficult use.

Submodular optimization uses discrete structures captured as the convexity of set functions. This approach is often able to derive practical solutions in polynomial time, and thus can be an effective approach to solve such a combinatorial optimization problem for which the search space exponentially increase. In particular, in the maximization problem of a monotonic set function with submodularity, a greedy algorithm can theoretically derive an approximate solution with a minimum ratio of $1 - \frac{1}{e}$ (≈ 0.632) between the approximate value and the optimal value [22]. Moreover, a greedy algorithm can find empirically or statistically practical and approximate solutions [23, 24]. For humans, information entails submodularity (reducing marginal utility). Assuming that the accuracy of neural networks has similar properties, we apply the greedy algorithm to the combinatorial optimization problem in Equation (2). Specifically, we assume the following pseudo-submodularity (3):

mean ??

$$\mathrm{Acc}(\mathbf{A} \cup \{j\}) - \mathrm{Acc}(\mathbf{A}) \geq \mathrm{Acc}(\mathbf{B} \cup \{j\}) - \mathrm{Acc}(\mathbf{B}) \quad (\forall \mathbf{A} \subseteq \mathbf{B} \subseteq \mathbf{V}, \quad \forall j \in \mathbf{V} \setminus \mathbf{B}) \tag{3}$$

Here, **V** denotes a universal set of all layers of a neural network to be quantized. We search for the combinations of quantization layers through a greedy algorithm that imitates submodular optimization.

Algorithm 1 Greedy search algorithm for partial quantization.

```
Input: Pre-trained FP32 model and the set of all its layers $\mathbf{V}(|\mathbf{V}| = N)$
1: $\mathbf{S} \leftarrow \emptyset$ (S is the set of layers to be quantized)
2: for $k = 1, 2, ..., N$ do
3:     $max\_obj = 0$
4:     for $j \in \mathbf{V} \setminus \mathbf{S}$ do
5:         if $max\_obj < \mathrm{Acc}(\mathbf{S} \cup \{j\}) \times (\log \#\mathrm{PARAM}(\mathbf{S} \cup \{j\}))^{\beta}$ then
6:             $max\_obj = \mathrm{Acc}(\mathbf{S} \cup \{j\}) \times (\log \#\mathrm{PARAM}(\mathbf{S} \cup \{j\}))^{\beta}$
7:             $quantized\_layer = j$
8:         end if
9:     end for
10:    Push $quantized\_layer$ into S
11: end for
Output: Ordered set S
```

3.2 Greedy search algorithm

We propose a greedy search algorithm inspired by submodular optimization to derive practical and approximate solutions for the combinatorial optimization problem in Equation (2). Our proposed algorithm presented in Algorithm 1 is simple; it greedily selects a quantization layer at a time until the quantization of all layers is completed. The zeroth step is to prepare a pre-trained FP32 model as shown in Fig. 1. The first step is to calculate (involve computing of inference) the objective function in Equation (2) as the layer sensitivity for all the unquantized layers, select one layer with the largest value and quantize it. This first step is iterated until all layers are quantized. In other words, the layers are selected in order of quantization efficiency as defined by the objective function in Equation (2). By plotting the inference accuracy after quantizing the selected layer at each step, we eventually obtain an improved trade-off between model size and accuracy as quantization progresses, as shown in Figs. 3 and 4. Users can choose the desirable quantization architectures of bit allocation among the plotted models for each computational resource.

For a search space 2^N, the computational complexity of the proposed algorithm is only $O(N^2)$, and the search is completed in a time of $\frac{1}{2}N(N+1)$ inferences. The search time depends only on the number of layers N and the inference time of the quantized model. Therefore, the time complexity does not vary even if the pre-quantization weights or the order of quantized layers are different in each search trial. For example, the greedy search algorithm determines the quantization architecture of MobileNetV2 [25] in approximately 2.6 V100 GPU-hours using 50k images to validate the ImageNet-1k dataset [26].

3.3 Quantization function

In this section, we describe the quantization details applied in our experiments to search for bit allocation for each layer. We adopt linear quantization with zero bias (also called offset or zero-point). The quantization scale is the range of values to be quantized. Larger scales include more outliers but result in coarser quantization. Several ways to determine the quantization scale have been proposed, such as using the maximum absolute value [27] and analyzing the value that minimizes the mean squared error [28]. The weight and activation distributions are different for each layer. Therefore, we use different scale values for each layer and defined the scale factor c as

$$c = \arg\max{}_x \mathrm{MSE}(\mathbf{W_k} \parallel \mathrm{quantize}(\mathbf{W_k}, x)) \tag{4}$$

where $\mathrm{MSE}(\cdot \parallel \cdot)$ is the mean squared error that corresponds to the distance between two tensors, $\mathbf{W}_k$ denotes a tensor to be quantized (e.g., weight and activation), and $\mathrm{quantize}(\cdot, x)$ quantizes an input tensor with scale x as follows:

$$\mathrm{quantize}(\mathbf{W_k}, x) = \mathrm{round}\left(\frac{\mathrm{clamp}(\mathbf{W_k}, x)}{\frac{x}{(2^{bit}-1)}}\right) \times \frac{x}{(2^{bit}-1)} \tag{5}$$

where $\mathrm{clamp}(\cdot, x)$ is to truncate the values into $[\min(\mathbf{W_k}), \min(\mathbf{W_k}) + x]$, bit is the quantization bit width, and $\mathrm{round}(\cdot)$ is the rounding function to the nearest even. For all of our quantization experiments, this rounding in Equation (5) is applied. We apply these scales and rounding to both the weight and activation quantization to avoid outliers. Furthermore, we employ multiples of a power of 2 for scale c to enable easy quantization in hardware such as deep-learning accelerators. We apply the per-layer quantization which uses a scale to fit the distribution of values for each layer based on Equations (4) and (5) in our experiments of partial quantization.

4 Experiments

In this section, we apply partial quantization using our proposed algorithm to multiple models and examine its performance. First, we show the experimental results of 8-bit and 6-bit quantization. Next, we discuss the effectiveness of the proposed method using different models. Further, we show the results of 4-bit quantization and discuss future work. Finally, we propose a faster algorithm.

We evaluated the greedy search algorithm for partial quantization in ImageNet classification using various convolutional neural networks. Owing to the differences among the depths of the networks, we used the number of layers or blocks for search space N. We quantized both weights and activations for all models. All experiments were conducted on PyTorch [29]. We quantized multiple convolutional neural networks according to the order of quantization layers (blocks) searched by Algorithm 1 and plotted the trade-off between inference accuracy and model size calculated based on the number of quantized parameters.

Figure 2 presents a flowchart of the algorithms. The results were compared with those obtained from sensitivity analysis [18]. For sensitivity analysis 2(a), the inference accuracy for all layers was calculated in advance, even when only one layer was quantized. The calculated accuracies were then sorted as sensitivities of layers, and the layers were quantized in ascending order of sensitivity. In other words, sensitivity analysis results are equivalent to the order of the quantization layer in step 1 in our proposed algorithm (Fig. 1). In contrast, our method 2(b) updated the sensitivity-based ordering each time we quantized one layer.

4.1 Partial quantization in ImageNet classification

We used multiple convolutional neural networks (ResNet34/50 [30], EfficientNet-B0/B1/B3 [31], Xception [32], MobileNetV2, and DenseNet121/161 [33]) and plotted the changes in the inference accuracy of ImageNet classification when these networks were quantized in steps. First, the results of ResNet34, 50 and Xception 8-bit quantization at the layer level are shown in Fig. 3a. In Xception, the inference accuracy was reduced by 1.5%, compared to that of the original FP32 model, under complete 8-bit quantization (equivalent to 4.0× model size compression); however, we achieved +0.015% accuracy increase with 2.5× model size compression with the greedy

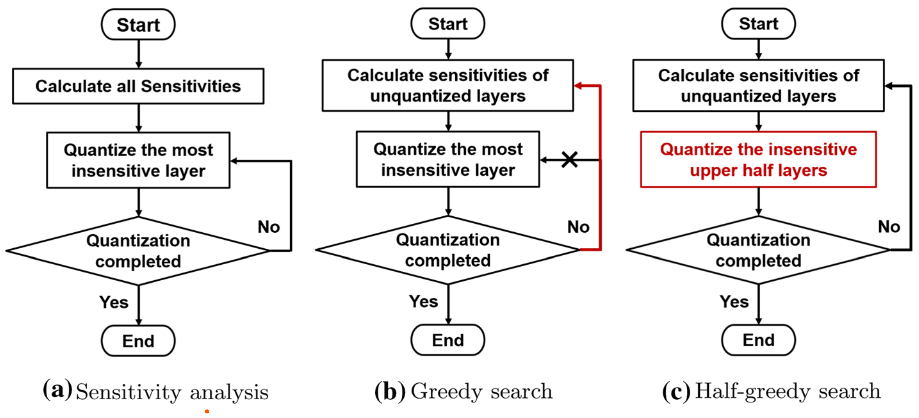
Start
Calculate all Sensitivities
Quantize the most insensitive layer
Quantization completed
No
Yes
End
(a) Sensitivity analysis
Calculate sensitivities of unquantized layers
Quantize the most insensitive layer
(b) Greedy search
Quantize the insensitive upper half layers
(c) Half-greedy search

Fig. 2 Flowcharts of the algorithms for partial quantization and **a** sensitivity analysis, **b** greedy search algorithm, and **c** half-greedy search

search algorithm. Regarding ResNet50, comparing sensitivity analysis and greedy search for 8-bit quantization is inappropriate because the accuracy was reduced by only 0.2% even with complete quantization. Therefore, we also experimented with 6-bit quantization to widen the range of accuracy degradation. The corresponding result is shown in Fig. 4. Figure 4a shows that ResNet50 could be compressed by 4.2× with only a 0.03% accuracy degradation by quantizing the greedily searched combinations of layers. In addition, the 6-bit quantization result of Xception is shown in Fig. 4b and Xception could be compressed by 3.6× with 3.1% accuracy degradation. We also quantized DenseNet to 6-bit for evaluation of our greedy search algorithm. (Here, we have quantized the transition and fully-connected layer in advance because they have numerous parameters.) The results in Fig. 4c and d show that the proposed algorithm achieves up to 0.3% accuracy improvement over the sensitivity analysis for comparable model sizes in both DenseNet121 and 161. Furthermore, our method improved the trade-off between model size and accuracy by tuning β to optimize the accuracy and the number of quantized parameters.

Second, the results of EfficientNet-B0, B1, B3 (using the weight parameters trained by Noisy Student [34]), and MobileNetV2 8-bit-quantization at the block level are shown in Fig. 3b. For these models, both methods yielded similar results because the inference accuracy of specific layers was significantly reduced owing to quantization. These layers tended to have a small number of parameters and large outliers; thus, both algorithms quantized those layers at the end in common. Therefore, in models with highly sensitive layers, the sensitivity analysis achieved a trade-off almost equal to that achieved by the greedy search algorithm. Whereas, the greedy search algorithm found more efficient quantized models than the sensitivity analysis in many cases although the curves in Figs. 3 and 4 indicate that the partial quantization problem of neural networks does not have submodularity in a precise sense.

Lastly, we compared the proposed method with various state-of-the-art quantization methods in terms of quantized models and their search times. Table 1 shows that our

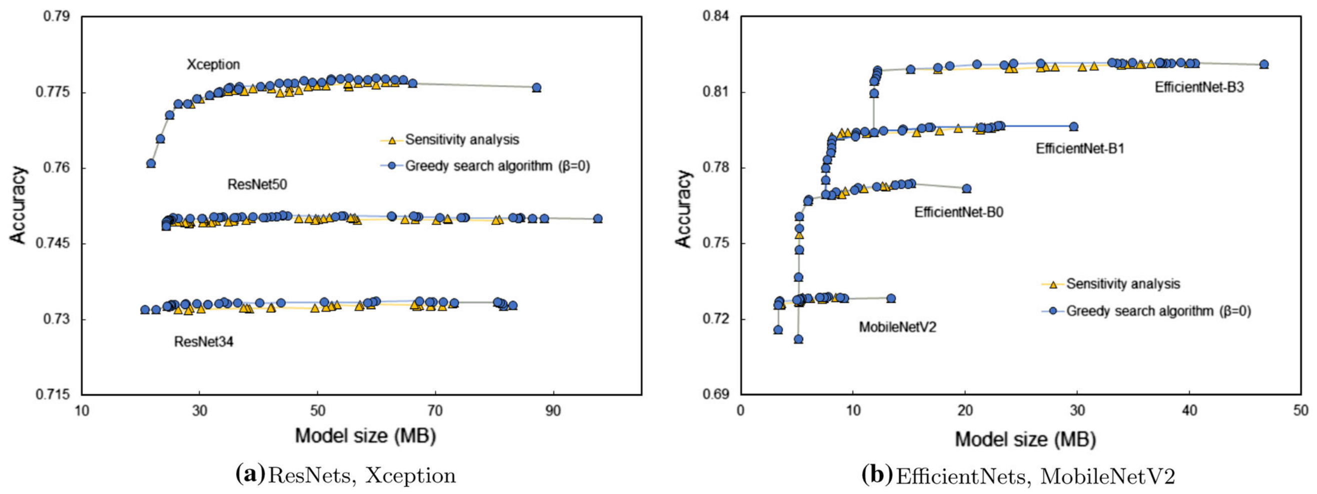

Fig. 3 8bit-quantization (W8A8) with sensitivity analysis and greedy search algorithm of **a** ResNet34/50, Xception and **b** EfficientNet-B0/B1/B3, MobileNetV2

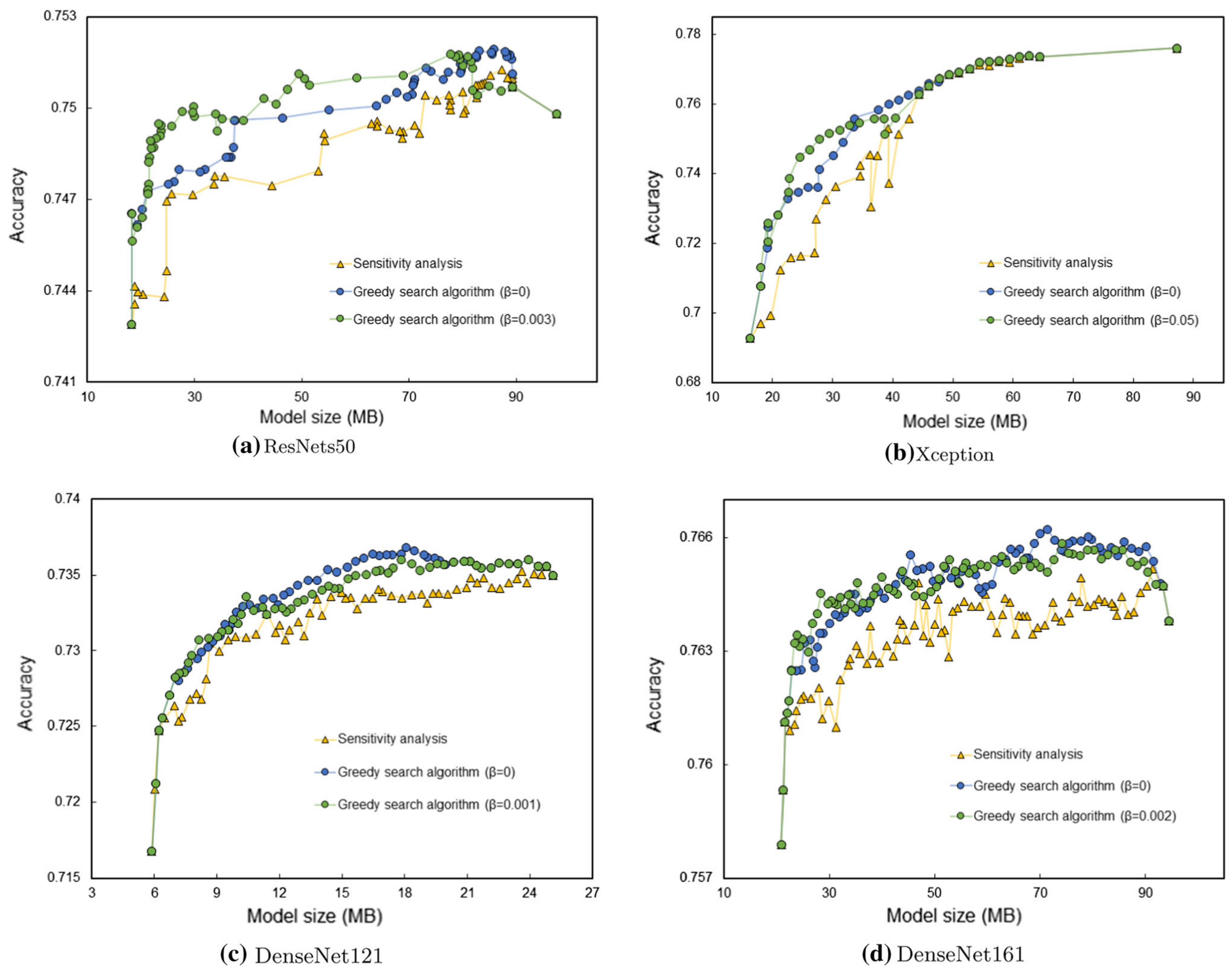

Fig. 4 6-bit-quantization (W6A6) with sensitivity analysis and greedy search algorithm of **a** ResNet50, **b** Xception, **c** DenseNet121, and **d** DenseNet161

Table 2 Order of quantization layers in 6-bit quantization of ResNet50 searched by the greedy search algorithm ($\beta = 0$)

Sensitivity ranking		step 1 $\rightarrow$	step 2 $\rightarrow$	step 3 $\rightarrow$
	1	L4 (0.75064)	quantized	quantized
$\uparrow$	2	L7 (0.75062)	L20 (0.7515)	quantized
insensitive	3	L34 (0.7504)	L21 (0.75118)	L21 (0.75176)
	4	L37 (0.7503)	L9 (0.751)	L32 (0.7516)
sensitive	5	L31 (0.75018)	L6 (0.75084)	L7 (0.75154)
$\downarrow$	6	L20 (0.75008)	L13 (0.75084)	L35 (0.75148)
	7	L21 (0.74992)	L11 (0.75074)	L17 (0.75142)
	:	:	:	:

greedy search requires an order of magnitude less search time than NAS or quantization methods that require network training. In addition, the greedy search achieved smaller degradation of accuracy than other post-training quantization methods.

4.2 Greedy search algorithm vs sensitivity analysis

To discuss the effectiveness of the proposed greedy search algorithm, we examined the order of quantization layers given by Algorithm 1, and the results are presented in Table 2. Sensitivity analysis continues to refer to the quantization order obtained in step 1 until the end. However, the table shows that the quantization order changes at each step. This indicates that the appropriate layers for quantization change exist each time a layer is quantized.

To quantitatively evaluate this change, we calculated the correlation coefficient between the sensitivities at $\beta = 0$ (just the inference accuracy after quantizing each layer) in steps 1 and 3. Table 3 lists the results calculated using the

Table 3 Correlation coefficient γ between the sensitivities in steps 1 and 3

	ResNet50 6bit	Xception 6bit	EfficientNet-B0 8bit	MobileNetV2 8bit
γ	**0.68**	**0.89**	0.98	0.99

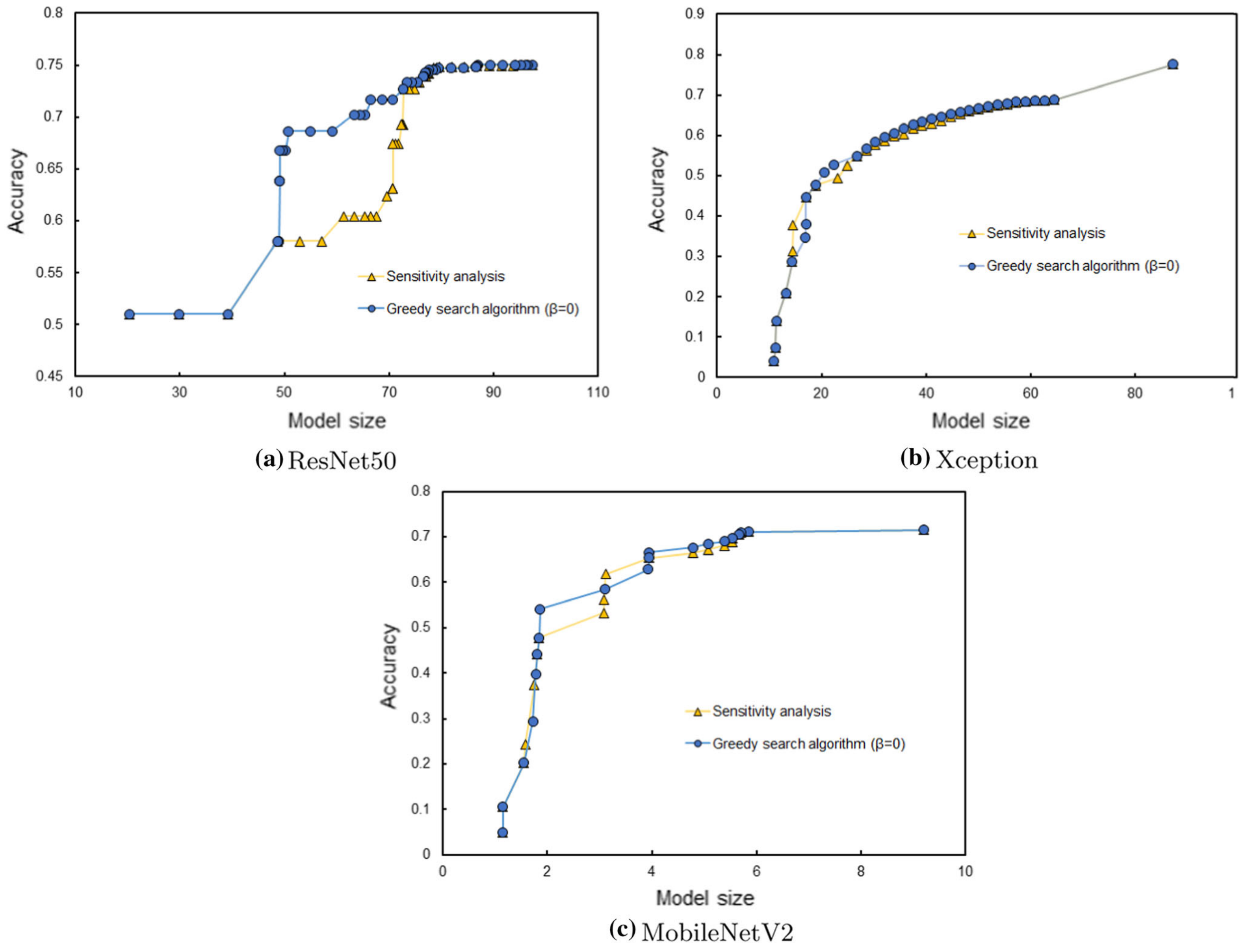

Fig. 5 4-bit-quantization (W4A8) with sensitivity analysis and greedy search algorithm of **a** ResNet50, **b** Xception, and **c** MobileNetV2

multiple networks. We show the values of correlation coefficient in bold that are relatively small in the four models. Considering Figs. 3 and 4, the larger the correlation coefficient γ, the closer the search results of the greedy search algorithm and the sensitivity analysis. Therefore, our proposed algorithm is particularly effective for convolutional neural networks with similar layer sensitivities because the quantization order is almost unchanged when the sensitivities of specific layers are high.

4.3 4-bit quantization

Greater accuracy degradation resulting from quantization leads to greater differences in the search algorithm. We examined the accuracy transition at a lower bit width, 4-bit quantization. Here, we quantized only weights to 4 bits and activations to 8 bits. Figure 5 shows the results of 4-bit-quantization for ResNet50, Xception, and MobileNetV2 in the same partial quantization experiments, as shown in Figs. 3 and 4. For each model in Fig. 5, the greedy search algorithm found more efficient combinations of quantization layers in terms of model size and inference accuracy (plot points near the upper left in the figures). However, the

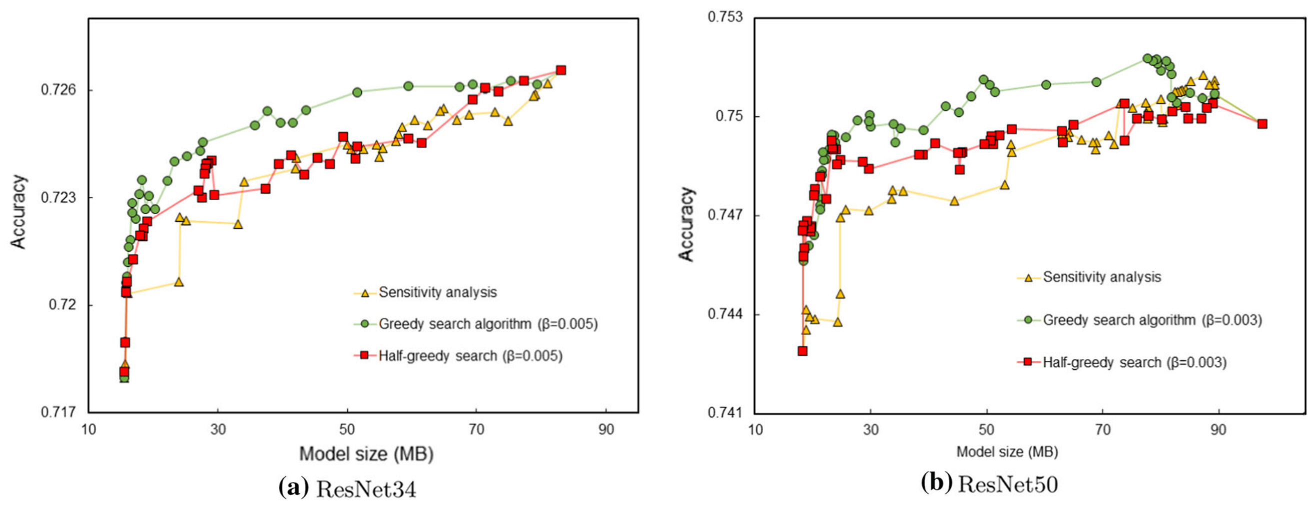

Fig. 6 6bit-quantization (W6A6) with half-greedy search of 6(a) ResNet34 and 6(b) ResNet50

Table 4 Computational cost and achieved accuracy degradation of each search algorithm

Algorithm	Number of inference	Computational complexity	Search time (1GPU-hours)		Accuracy degradation (%)	
			ResNet34	ResNet50	ResNet34 (28.5MB)	ResNet50 (23.5MB)
Sensitivity analysis	N	$O(N)$	0.8	2.6	-0.42	-0.53
Half-greedy search	$2N$	$O(N)$	**1.3**	**4.5**	**-0.25**	**-0.072**
Greedy search	$\frac{1}{2}N(N+1)$	$O(N^2)$	9.9	38.5	**-0.20**	**-0.052**

inference accuracies significantly decreased when all the layers of the models were quantized. This indicates that, in addition to the search algorithm for partial quantization, the quantization function and granularity need to be improved. Thus, one must combine other quantization techniques with post-training 4-bit quantization in future work, such as the adoption of stochastic rounding [37], a optimization of rounding up or down [35], a per-channel optimization [38] of quantization scale, and a scheme that uses Kullback–Leibler divergence rather than mean squared error to define the distance of the distributions.

4.4 Faster proposed algorithm

The proposed greedy search algorithm further improves the trade-off between model size and inference accuracy because of quantization; however, its computational complexity $O(N^2)$ is not negligible, although it is in polynomial time, considering the recent tendency of neural networks to become deeper as their accuracy improves. For example, applying the greedy search algorithm to a model with the number of layers $N = 100$ would require 5,050 inferences.

Therefore, we introduce a faster algorithm, half-greedy search shown in Fig. 2c. Because the quantization layers with significant accuracy degradation are biased toward the last stage of partial quantization, this algorithm quantizes half of the unquantized layers at every step. Its search is completed in $2N$ inferences with computational complexity $O(N)$ because the number of inferences required is halved every step. The result of searching for partial quantization of ResNet34 and ResNet50 by applying the half-greedy search is shown in Fig. 6. In both models, half-greedy search found combinations that were close to the efficient quantization layer combinations found by the greedy search algorithm. In addition, half-greedy search reduced the search cost by 34 hours than the greedy search algorithm in partial quantization experiments of ResNet50. A comparison of the computational complexity of each algorithm and the performance of the searched quantization models is shown in Table 4. We show the figures in bold of the search time for the half-greedy search and the accuracy degradation of the greedy and half-greedy search. The half-greedy algorithm can be positioned between sensitivity analysis and the greedy search algorithm in terms of the search result and cost.

5 Conclusion

We formulated the partial quantization of neural networks as a simple combinatorial optimization problem and showed that the proposed greedy search algorithm derives practical solutions for this problem. The proposed algorithm is inspired by a submodular function and has a low computational complexity $O(N^2)$. The greedy search algorithm was evaluated using ImageNet classification with various convolutional neural networks, and it improved the trade-off between inference accuracy and model size compared to that achieved by sensitivity analysis. Specifically, 4.2× model size compression with only 0.03% accuracy degradation in ResNet50 and 2.5× compression with +0.015% accuracy gain in Xception were achieved. In addition, we quantitatively evaluated the conditions under which our method performed effectively compared with the sensitivity analysis method. The curves corresponding to the greedy search algorithm in Figs. 3 and 4 show that the partial quantization problem of neural networks does not have submodularity in a precise sense. However, as the number of quantized layers increases, the accuracy degradation attributable to the quantization of one layer tends to increase. One can search for efficient architectures using a greedy algorithm. Moreover, we discussed 4-bit post-training quantization with the proposed algorithm in future work and introduced a faster algorithm.

Declarations

Conflict of interest All the authors declare that they have no conflict of interest.

References

1. Molchanov P, Tyree S, Karras T, Aila T, Kautz J (2016) Pruning convolutional neural networks for resource efficient inference arXiv preprint arXiv:1611.06440
2. Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network arXiv preprint arXiv:1503.02531
3. Wu J, Leng C, Wang Y, Hu Q, Cheng J (2016) Quantized convolutional neural networks for mobile devices In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4820–4828
4. Zhou S, Wu Y, Ni Z, Zhou X, Wen H, Zou Y (2016) Dorefa-net: training low bitwidth convolutional neural networks with low bitwidth gradients arXiv preprint arXiv:1606.06160
5. Jacob B, Kligys S, Chen B, Zhu M, Tang M, Howard A, Adam H, Kalenichenko D (2018) Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 2704–2713
6. Markidis S, Chien SWD, Laure E, Peng IB, Vetter JS (2018) Nvidia tensor core programmability, performance precision. In: 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp 522–531 https://doi.org/10.1109/IPDPSW.2018.00091
7. Jouppi NP, Young C, Patil N, Patterson D, Agrawal G, Bajwa R, Bates S, Bhatia S, Boden N, Borchers A, et al. (2017) In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th Annual International Symposium on Computer Architecture, pp 1–12
8. Wu B, Dai X, Zhang P, Wang Y, Sun F, Wu Y, Tian Y, Vajda P, Jia Y, Keutzer K (2019) Fbnet: hardware-aware efficient convnet design via differentiable neural architecture search In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 10734–10742
9. Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, Le QV (2019) Mnasnet: platform-aware neural architecture search for mobile In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 2820–2828
10. Zoph B, Le QV (2016) Neural architecture search with reinforcement learning arXiv preprint arXiv:1611.01578
11. Wu B, Wang Y, Zhang P, Tian Y, Vajda P, Keutzer K (2018) Mixed precision quantization of convnets via differentiable neural architecture search arXiv preprint arXiv:1812.00090
12. Guo Z, Zhang X, Mu H, Heng W, Liu Z, Wei Y, Sun J (2020) Single path one-shot neural architecture search with uniform sampling In: European Conference on Computer Vision, pp 544–560. Springer
13. Han S, Mao H, Dally WJ (2015) Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding arXiv preprint arXiv:1510.00149
14. Zhou Y, Moosavi-Dezfooli S-M, Cheung N-M, Frossard P (2018) Adaptive quantization for deep neural network In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 32
15. Wang K, Liu Z, Lin Y, Lin J, Han S (2019) Haq: Hardware-aware automated quantization with mixed precision In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 8612–8620
16. Uhlich S, Mauch L, Cardinaux F, Yoshiyama K, Garcia JA, Tiedemann S, Kemp T, Nakamura A (2019) Mixed precision dnns: all you need is a good parametrization arXiv preprint arXiv:1905.11452
17. Dong Z, Yao Z, Gholami A, Mahoney MW, Keutzer K (2019) Hawq: Hessian aware quantization of neural networks with mixed-precision In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 293–302
18. Wu H, Judd P, Zhang X, Isaev M, Micikevicius P (2020) Integer quantization for deep learning inference: principles and empirical evaluation arXiv preprint arXiv:2004.09602
19. Nagel M, Baalen Mv, Blankevoort T, Welling M (2019) Data-free quantization through weight equalization and bias correction In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 1325–1334
20. Nahshan Y, Chmiel B, Baskin C, Zheltonozhskii E, Banner R, Bronstein AM, Mendelson A (2019) Loss aware post-training quantization arXiv preprint arXiv:1911.07190
21. Choukroun Y, Kravchik E, Yang F, Kisilev P (2019) Low-bit quantization of neural networks for efficient inference In: ICCV Workshops, pp 3009–3018

22. Nemhauser G. L., Wolsey L. A. (1981) Maximizing submodular set functions: formulations and analysis of algorithms In: North-Holland Mathematics Studies, vol. 59, pp. 279–301. Elsevier
23. Lin H, Bilmes J (2010) Multi-document summarization via budgeted maximization of submodular functions In: Human language technologies: the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp 912–920
24. Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Series B (Methodol) 58(1):267–288
25. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C (2018) Mobilenetv2: inverted residuals and linear bottlenecks In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4510–4520
26. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp 248–255 Ieee
27. Vanhoucke V, Senior A, Mao MZ (2011) Improving the speed of neural networks on cpus. In: Proceedings of the Deep Learning and Unsupervised Feature Learning NIPS Workshop, vol. 1
28. Banner R, Nahshan Y, Hoffer E, Soudry D (2018) Post-training 4-bit quantization of convolution networks for rapid-deployment arXiv preprint arXiv:1810.05723
29. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, et al. (2019) Pytorch: an imperative style, high-performance deep learning library arXiv preprint arXiv:1912.01703
30. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 770–778
31. Tan M, Le Q (2019) Efficientnet: rethinking model scaling for convolutional neural networks In: International Conference on Machine Learning, pp 6105–6114 PMLR
32. Chollet F (2017) Xception: deep learning with depthwise separable convolutions In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1251–1258
33. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4700–4708
34. Xie Q, Luong M-T, Hovy E, Le QV (2020) Self-training with noisy student improves imagenet classification In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 10687–10698
35. Nagel M, Amjad RA, Van Baalen M, Louizos C, Blankevoort T (2020) Up or down? adaptive rounding for post-training quantization In: International Conference on Machine Learning, pp 7197–7206 PMLR
36. Wang T, Wang K, Cai H, Lin J, Liu Z, Wang H, Lin Y, Han S (2020) Apq: joint search for network architecture, pruning and quantization policy In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 2078–2087
37. Gupta S, Agrawal A, Gopalakrishnan K, Narayanan P (2015) Deep learning with limited numerical precision In: International Conference on Machine Learning, pp 1737–1746 PMLR
38. Krishnamoorthi R (2018) Quantizing deep convolutional networks for efficient inference: a whitepaper arXiv preprint arXiv:1806.08342