An Automatic Neural Network Architecture-and-Quantization Joint Optimization Framework for Efficient Model Inference

Lian Liu, Ying Wang, *Member, IEEE*, Xiandong Zhao, Weiwei Chen, Huawei Li, *Senior Member, IEEE*, Xiaowei Li, *Senior Member, IEEE*, and Yinhe Han, *Senior Member, IEEE*

***Abstract*—Efficient deep learning models, especially optimized for edge devices, benefit from low inference latency to efficient energy consumption. Two classical techniques for efficient model inference are lightweight neural architecture search (NAS), which automatically designs compact network models, and quantization, which reduces the bit-precision of neural network models. As a consequence, joint design for both neural architecture and quantization precision settings is becoming increasingly popular. There are three main aspects that affect the performance of the joint optimization between neural architecture and quantization: 1) quantization precision selection (QPS); 2) quantization-aware training (QAT); and 3) NAS. However, existing works focus on at most twofold of these aspects, and result in secondary performance. To this end, we proposed a novel automatic optimization framework, DAQU, that allows jointly searching for Pareto-optimal neural architecture and quantization precision combination among more than 10^{47} quantized subnet models. To overcome the instability of the conventional automatic optimization framework, DAQU incorporates a warm-up strategy to reduce the accuracy gap among different neural architectures, and a precision-transfer training approach to maintain flexibility among different quantization precision settings. Our experiments show that the quantized lightweight neural networks generated by DAQU consistently outperform state-of-the-art NAS and quantization joint optimization methods.**

***Index Terms*—Automatic joint optimization, efficient model inference, network quantization, neural architecture search (NAS).**

I. Introduction

Deep neural networks (DNNs) are proven to have achieved state-of-the-art (SOTA) accuracy in many applications. However, due to the burden of computation cost and model size, it is hard to efficiently deploy large-scale DNN models onto resource-constrained platforms, especially on edge devices. Many prior techniques have been studied from different aspects to reduce the overhead of model inference, including lightweight neural architecture search (NAS) [1], [2], [3], [4], [5] and low-precision quantization [6], [7], [8], [9], [10], [11], [12], [13], [14]. For real-world deployment scenarios, both NAS and quantization methods are employed to promote the execution efficiency of network inference. For example, quantization can be conducted on top of NAS to further lower the bit width of the lightweight model acquired at the NAS stage. Compared with disparate optimization, the joint optimization of neural architecture and quantization proves to be a more efficient solution [15], [16], [19].

The aspects of orchestrating architecture and quantization optimization are threefold, including quantization precision selection (QPS), quantization-aware training (QAT), and NAS. However, all of the existing works that seek to find the efficient combinations of architecture and mixed-precision setting, cover at most twofold of the aspects above. Generally, we can classify existing works into three categories: 1) NAS-then-Quantization; 2) Quantized Weight-Sharing Supernet; and 3) quantization-aware NAS, as shown in Fig. 1(a)–(c). NAS-then-Quantization sequentially performs NAS followed by QPS optimization, and finally trains the quantized model. As shown in Fig. 2(a), since the search processes (NAS and QPS) are independent across different optimization stages, the acquired model does not exhibit charming performance. This is reflected by the fact that inconsistent accuracy ranking order between full-precision and quantized models searched by NAS-then-Quantization, as Table I reports. The Quantized Weight-sharing Supernet directly performs QAT on the entire supernet optimization space, obtaining a quantized supernet optimization space. Specifically, OQAT [15] trained a series of fixed-precision weight-sharing supernets with different bit widths, while BatchQuant (BQ) [16] proposed a robust quantizer formulation to further support stable training of mixed-precision supernet. However, Fig. 2(b) shows that the acquired supernet optimization space is not equivalent to the actual optimal space (space at the bottom of Fig. 2(a). As indicated in the second and third row of Table I, the rankings of different subnets still vary before and after retraining, which indicates that an accurate subnet model derived from supernet cannot guarantee its high performance after retraining. Quantization-aware NAS methods coupling QPS and NAS perform the optimization process directly in the

Manuscript received 3 April 2023; revised 27 September 2023; accepted 13 November 2023. Date of publication 5 December 2023; date of current version 23 April 2024. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 6209002 and Grant 62222411. This article was recommended by Associate Editor P. A. Beerel. *(Corresponding authors: Ying Wang; Huawei Li.)*

Lian Liu, Ying Wang, Huawei Li, Xiaowei Li, and Yinhe Han are with the State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China, and also with the Department of Computer Science, University of Chinese Academy of Sciences, Beijing 100190, China (e-mail: wangying2009@ict.ac.cn; lihuawei@ict.ac.cn).

Xiandong Zhao and Weiwei Chen are with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China.

Digital Object Identifier 10.1109/TCAD.2023.3339438

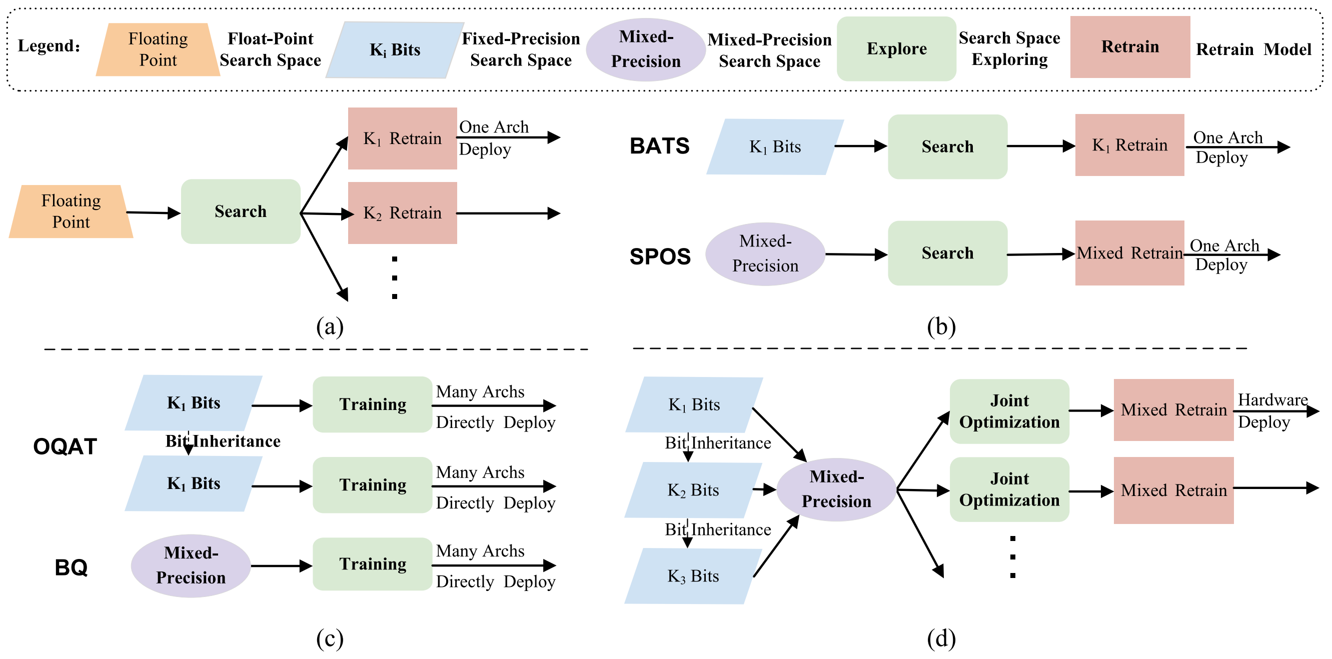


Fig. 1. Representative workflow of existing optimization frameworks (including our framework) on combining quantization and NAS. (a) NAS-then-Quantization. (b) Quantized Weight-Sharing Supernet. (c) Quantization-aware NAS. (d) DAQU (Ours).

TABLE I
ACCURACY INCONSISTENCY BETWEEN ORIGINAL NETWORKS AND RETRAINED NETWORKS

Categories	Models	Original Bits (W/A)	Original Top-1 Acc (%)	Retraining Bits (W/A)	Retraining Top-1 Acc (%)	Improvement
NAS-then-Quantization	LSQ + OFA-A	float points	76.2	4	73.2	**-3.0**
	LSQ + OFA-B	float points	77.0	4	71.8	-5.2
Quantized Weight-sharing Supernet	OQAT-MBV2-A	4	71.8	4	73.2	**+1.4**
	OQAT-MBV2-B	4	72.4	4	72.8	+0.4
Quantized Weight-sharing Supernet	BQ-A	2,3,4	72.3	2,3,4	73.6	**+1.3**
	BQ-B	2,3,4	73.0	2,3,4	73.5	+0.5

space of quantized models. However, as indicated in Fig. 2(c) the ignorant of QAT makes those methods, such as BATS [17], SPOS [18], and APQ [19], suffer from inferior performance. In conclusion, existing joint optimization methods, due to the incomplete consideration of threefold aspects, fail to guarantee the best combination of architecture and quantization setting.

To successfully orchestrate the threefold aspects, including QPS, QAT, and NAS, we propose DAQU, an automatic neural architecture and quantization joint optimization framework. Specifically, we adopt a differentiable optimization engine to naturally combine training (QAT) and searching (QPS and NAS). However, a naive differentiable optimization framework suffers from instability among billions of architectures and quantization precision settings [20]. On the one hand, searching for quantized neural architecture from scratch can lead to premature convergence to an inefficient model, as indicated in Section IV. On the other hand, training neural networks with different precision combinations causes unstable training processes and inaccurate results. To tackle these challenges, we propose a warm-up strategy to reduce the accuracy gap among different neural architectures and avoid converging to a poor-performance local optimum, and make the training curve stable by applying a precision-transfer policy among various mixed-precision combinations. Furthermore, we propose an MLP-based hardware performance predictor to efficiently evaluate hardware performance and guide the search procedure. In all, our framework conducts efficient searching that finds the best combination of architecture and quantization precision setting and produces Pareto-optimal quantized networks.

In general, the contributions of this work are as follows.

1) To the best of our knowledge, DAQU for the first time enables QPS, QAT, and neural architecture searching in a single optimization process to acquire high-performance ultralow-precision networks.
2) To avoid premature convergence to an inefficient model, a warm-up strategy is devised. Before joint optimization, the fixed-precision supernet is pretrained for a few epochs, improving the model accuracy and reducing the variance among subnets.
3) A precision-transfer policy is proposed to tackle the training instability among different mixed-precision combinations. Considering that different precision choices require different weight parameters, we use the backward information for a specific combination to update weight parameters for all mixed-precision choices.
4) The proposed automatic optimization framework, DAQU, has been implemented and testified in several

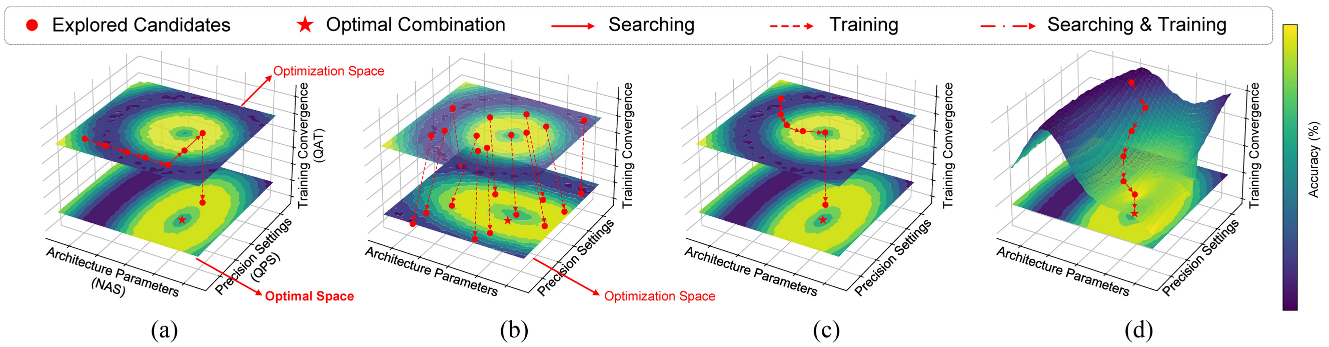


Fig. 2. Optimization process of existing frameworks. (a) Typical NAS-then-Quantization framework sequentially searches neural architecture parameters and precision settings in the original optimization space, and then conducts QAT to acquire the final optimized NN model. (b) Quantized Weight-Sharing Supernet framework directly conducts QAT on the original optimization space to get the final optimization space, which is different from the optimal space expected. (c) Quantization-aware NAS jointly searches the architecture and precision parameters to get an optimal combination in the optimization space. (d) DAQU framework jointly optimizes the architecture parameters and precision settings, gradually steering the original optimization space closer to the optimal space with the help of differentiable training.

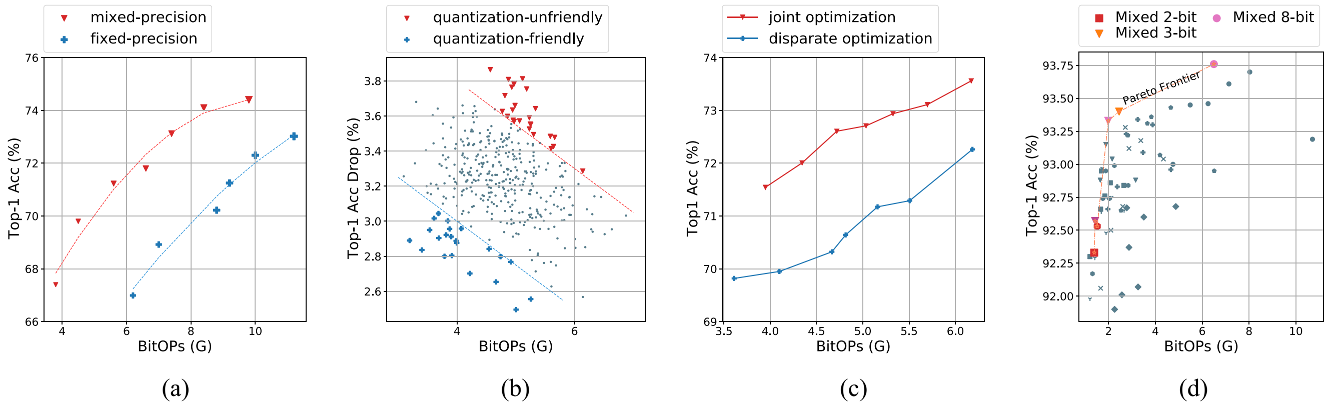


Fig. 3. Motivation on mixed-precision quantization and neural architecture joint optimization. We have conducted a series of experiments to verify our motivation. (a) Networks with a combination of mixed-precision settings outperform neural networks with fixed-precision. (b) Some networks are more friendly to be quantized, with less accuracy drop. (c) Joint optimization method on neural architecture and quantization policy searching consistently outperforms the method with disparate optimization. (d) Pareto frontier is composed of different NAS and quantization policy configurations.

practical use cases, such as embedded device deployment and algorithm/hardware co-design. Compared with existing works, DAQU achieves SOTA performance through the tradeoff between the hardware-aspect computation overhead and model accuracy.

II. Background and Motivation

A. Quantization-Aware Training

Quantization [6], [7], [10], [13], [23], which transfers floating-point values to fixed-precision ones, is a widely adopted policy in practice. In this article, we use LSQ [11], a SOTA uniform QAT approach, to achieve low-precision models. Given a floating-point value v and a predefined bit-width k, the quantizer can be represented as

$$\begin{aligned}\bar{v} &= \lfloor \text{clip}(v/s, Q_{\min}, Q_{\max}) \rceil \\ \hat{v} &= \bar{v} \times s \end{aligned} \tag{1}$$

where $\lfloor \cdot \rceil$ hints the rounding function, and clip($\cdot$) maps value in the integer range $[Q_{\min}, Q_{\max}]$. Here, the integer range for weights and activation is corresponding $[-2^{k-1}, 2^{k-1}-1]$ and $[0, 2^k - 1]$. The quantizer step size s is a learnable parameter that will be updated in each training iteration as

$$\frac{\partial \hat{v}}{\partial s} = \begin{cases} -v/s + \lfloor v/s \rceil & \text{if } \; Q_{\min} < v/s < Q_{\max} \\ Q_{\min} & \text{if } \; v/s \le Q_{\min} \\ Q_{\max} & \text{if } \; v/s \ge Q_{\max}. \end{cases} \tag{2}$$

Rather than using fixed-precision for all layers, layer-wise mixed-precision quantization [24], [25], [26] applies different bit widths on different layers. As shown in Fig. 3(a), mixed-precision networks lie on a better Pareto curve than fixed-precision ones. The reason is that different layers have different redundancy, which indicates using a mixed-precision model achieves more efficient inference.

B. Differential Neural Architecture Search

NAS [4], [18], [27], [28], [29] focuses on automating network architecture design. A well-designed neural architecture helps to improve accuracy and reduce inference overhead. Previous works [30], [31] also indicate that architecture design is an important part of quantization performance. As shown in Fig. 3(b), we sampled different architectures from OFA [1] and OQAT [15] to denote the accuracy drop of different networks. One can notice that networks with lower accuracy drop and fewer computation operations are more friendly for quantization. The observation in Table I manifests that optimizing a quantized network is not identical to quantizing an optimized full-precision network.

Among existing NAS methods, differentiable NAS [27], [32] (DNAS) is an attractive and efficient searching method. DNAS continuously relaxes the discrete search space and allows it to be optimized through the gradient. DNAS, as we use in this article, includes two advantages. On the one hand, a gradient-based optimization process helps to orchestrate the joint searching (QPS and NAS) and QAT process. On the other hand, the differentiable searching method is easily scalable to larger spaces by relaxing different discrete aspects simultaneously.

One classical method for relaxing discrete spaces is Gumbel-Softmax (GS). GS [49] is a reparameterization solution to satisfy the expectation of discrete posterior distribution. The formulation of GS can be represented as

$$\beta_t = \frac{\exp^{(\beta_t+\epsilon_t)}/\tau}{\sum_{i=1}^{N}\exp^{(\beta_i+\epsilon_i)}/\tau}$$
$$\epsilon \sim U(0,1) \quad (3)$$

where β denote the original parameter distribution, and ϵ is a sampled number that satisfy uniform distribution between 0 and 1. Also, we can use temperature coefficient τ to control the smoothness of distribution.

Unlike traditional Softmax, GS randomizes sampled results without affecting the expectation of original distribution. In this article, we adopt GS to design our differentiable optimization engine.

C. Joint Optimization of NAS and Quantization

As discussed in Sections II-A and II-B, mixed-precision quantization is better than fixed-precision, and searching for a quantization-friendly network is not identical to quantizing a well-designed full-precision network. Fig. 3(c) proves that joint optimization for architecture and quantization precision setting outperforms disparate NAS-then-Quantization. Furthermore, Fig. 3(d) demonstrates that the Pareto frontier is composed of different NAS-and-Quantization configurations, including different architectures and precision settings. Naturally, we hope to combine NAS with mixed-precision QAT, instead of using sequential optimization, to search for an optimal quantized model.

Existing quantization-aware NAS works either reduce search space or ignore the QAT impact on quantization performance (discussed in Section IV). BATS [17] searches for optimized binary neural architecture but fails to achieve promising performance due to the poor representation of the binary network. SPOS [18] only allows channel and precision search with uniform sampling. APQ [19] trains an accuracy predictor with sampled architectures and quantization precision settings, which actually leads to ambiguous proxy due to inadequate QAT. Overall, since QAT itself deforms the optimization space, searching for model architectures and precision settings on the initial optimization space results in a discrepancy between the searched and optimal targets, as shown in Fig. 2(c). This prevents obtaining the optimized quantized model. Recent works focus on training a quantized weight-sharing supernet that combines QAT with searching spontaneously. Specifically, OQAT [15] trains 4/3/2-bit fixed-precision supernet, respectively, with bit-inheritance strategy. BQ [16] proposes a robust mixed-precision quantizer, BQ, to train a mixed-precision supernet with 2/3/4-bit. However, due to the uncoupledness of weights distribution between the supernet and subnets (networks sampled from the supernet) [44], [45], directly applying QAT on the supernet (the overall optimization space) yields a trained optimization space that is not equivalent to the actual optimal space, as depicted in Fig. 2(b). This results in the inability to identify the optimal combination of architecture and quantization precision. Specifically, lack of QAT leads to inferior quantization performance [33], [34], while lack of QPS and NAS results in inconsistency before and after optimization. As a result, we emphasize that each of QPS, QAT, and NAS plays an important role in joint optimization. DAQU presents a differentiable optimization engine that covers all three aspects. Specifically, the optimization engine searches the model architecture and precision settings collaboratively by relaxing the overall search space. Concurrently, the differentiable QAT process progressively aligns the original optimization space with the optimal space, as Fig. 2(d) shows.

III. DAQU: Framework Overview

In this section, we give an overview of the proposed DAQU framework. We first formulate the joint optimization problem in Section III-A and further introduce how to evaluate the hardware performance during the rapid iterative optimization process in Section III-B. Finally, we present the overall workflow of our DAQU framework in Section III-C.

A. Problem Formulation

The whole optimization process includes searching combinations of neural architectures and precision settings simultaneously. Besides, ignoring the weight update in QAT causes inferior performance searching, as indicated in Table I. In consequence, the training process should also be considered in the formulation. Also, the hardware constraint (e.g., the number of operations, latency, and energy) is desired in our formulation.

Following the requirements mentioned above, we can formulate the joint optimization scheme as follows:

$$\min_{\alpha,\gamma} \quad \zeta_{\text{val}}\big(F(\omega^*, \mathcal{N}(\alpha), \mathcal{Q}(\gamma))\big) \quad (4)$$
$$\text{s.t.} \quad h_{\text{cost}}(\mathcal{N}(\alpha), \mathcal{Q}(\gamma)) \le H_{\text{constraint}} \quad (5)$$
$$\text{s.t.} \quad \omega^* = \arg\min_{\omega} \quad \zeta_{\text{train}}(F(\omega, \mathcal{N}(\alpha), \mathcal{Q}(\gamma))) \quad (6)$$

where α and γ are continuous variables parameterizing the probability of choosing different candidates for neural architecture design (e.g., kernel-size/expand-ratio/depth) and network precision setting (e.g., 2-/3-/4 bit), respectively; ω represents the weight of networks; ζ_{val}, ζ_{train}, and h_{cost} are the validation loss, training loss, and the hardware-level cost function for model deployment, respectively; F and

$H_{\text{constraint}}$ represent the inference procedure of the feedforward network and the target hardware-aspect cost (e.g., energy and latency); and $\mathcal{N}(\alpha)$ and $\mathcal{Q}(\gamma)$ denote the specific neural architecture and mixed-precision combination characterized by α and γ. Specifically, for neural network layer j, the corresponding variables $\mathcal{N}(\alpha^j) = sample([\alpha_1^j, \ldots, \alpha_m^j])$ and $\mathcal{Q}(\gamma^j) = sample([\gamma_1^j, \ldots, \gamma_n^j])$ represent sampling a corresponding design from m neural architecture parameters and n precision setting choices. The specific values α_i^j and γ_k^j indicate the probability of selecting the given neural architecture and precision setting so the sum of variables in each layer must be 1 ($\sum_{i=1}^{m} \alpha_i^j = 1, \sum_{k=1}^{n} \gamma_k^j = 1$). Furthermore, we employ the GS strategy in (3) to implement the sampling of these parameters, and during backpropagation, the update rule of GS is equivalent to softmax, which updates selection probability for each architecture and precision choice based on the training loss function. Therefore, we can realize the differentiable optimization of architecture parameters and precision settings through softmax-based backpropagation strategies.

Need to note that, our objective is to search for the architecture and precision settings under the optimal weight distribution after training [ω^* in (6)]. This requires us to further account for model training preferences and explore a more complex optimization space during the optimization process, as we illustrated in Fig. 2(d). However, existing optimization approaches [15], [16], [18] either overlook considerations of weight training or focus on maintaining the training stability of supernet while ignoring the search for the optimal architecture and precision setting.

B. Hardware-Aware Optimization

Hardware-Aspect Performance Prediction: Due to the diversity of hardware configuration (e.g., level and size of the memory and the number of the computing unit), the optimal quantized model for one specific hardware may not work for another. Therefore, a hardware-aware network search scheme should have an accurate measurement or a proxy of the hardware-level performance, e.g., run-time latency and energy overhead, for the sampled nodes in the optimization process. While the indirect performance indicator like model size and operation number is thought inaccurate, using a lookup table (LUT) [1], [4], [19] to directly calculate and keep the metrics of latency (energy as well) becomes a more popular way. However, prior LUT-based measurement method only tracks the overhead of each neural layer and ignores the cost of data movement and the fusion opportunity across layers. Motivated by [40], we employ an MLP-based predictor to evaluate the hardware-aspect overhead for running the network samples. As shown in Fig. 4, we propose a four-layer MLP model to evaluate the hardware performance of a given quantized network. It receives the specific architecture and precision parameters and outputs two cost metrics (latency and energy). We train the predictor by sampling hardware inference performance corresponding to the specific combination of architecture and precision setting. According to the results of cross-validation, the inference error between our proposed predictor and the actual hardware is less than 2.7%.

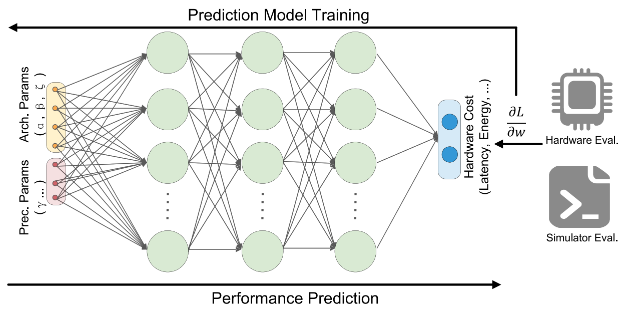


Fig. 4. Architecture of our proposed hardware performance predictor.

Hardware-Aware Loss Function: The optimal quantized network for the given hardware device has to achieve not only high accuracy but also efficient inference performance such as low latency and energy. We present a predictor-based model to estimate hardware cost, formulated as $h_{\text{cost}}(\mathcal{N}(\alpha), \mathcal{Q}(\gamma))$. Intuitively, we expect the loss of hardware cost should be large enough to point out the performance violation when the inference latency of the searched model greatly exceeds the given constraint. Conversely, if the hardware performance of the obtained model is close to the given hardware constraint, the loss of hardware cost $\mathcal{L}_{hw}$ should be small enough not to affect the exploration of high-accuracy models. Hence, we heuristically design a loss function for hardware-aspect evaluation as follows:

$$\mathcal{L}_{hw} = \max\left(0, \tan\frac{\pi \cdot (h_{\text{cost}} - H_{\text{constraint}})}{2 \cdot (h_{\text{cost}} + H_{\text{constraint}})}\right). \tag{7}$$

The given tangent function perfectly satisfies our requirements. Specifically, the loss is relatively low when exceeding a little more than the given hardware constraint, while the loss grows rapidly with the increase in hardware consumption.

In conclusion, we introduce MLP-based hardware cost h_{cost} into the hardware-aware loss function $\mathcal{L}_{hw}$ to solve the nondifferentiable hardware constraint described in (5). Finally, we can summarize the search target as

$$\begin{aligned} \alpha^*, \gamma^* = \underset{\alpha,\gamma}{\arg\min}\ & \zeta_{\text{val}}\big(F\big(\omega^*, \mathcal{N}(\alpha), \mathcal{Q}(\gamma)\big)\big) \\ & + \lambda \mathcal{L}_{hw}(\mathcal{N}(\alpha), \mathcal{Q}(\gamma)). \end{aligned} \tag{8}$$

Here, the hyperparameter λ is used to balance two loss functions ζ_{val} and $\mathcal{L}_{\text{hw}}$. We set $\lambda = 1$ in this article to achieve high-performance results in our evaluations (Section VII).

C. DAQU: The Overall Framework

Objective: The key objective of our DAQU framework is to find the best combination of neural architecture and quantization precision setting ($\mathcal{N}(\alpha), \mathcal{Q}(\gamma)$). As a result, DAQU presents an automatic differentiable optimization engine to incorporate QPS, QAT, and NAS. Instead of considering part of these three aspects, DAQU includes all in the joint optimization procedure.

WorkFlow: Fig. 1(d) shows the workflow of the proposed DAQU framework as follows.

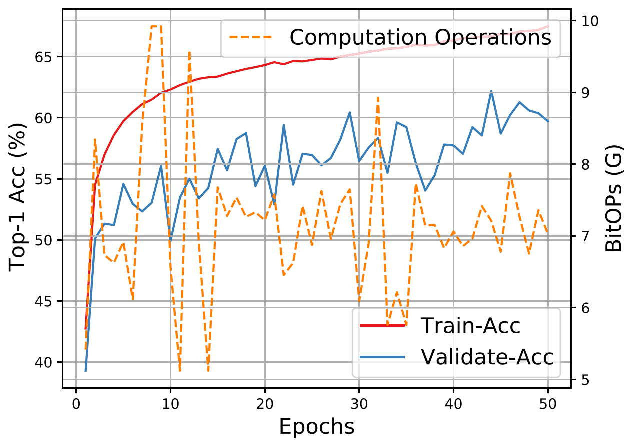


Fig. 5. Joint optimization curve for DAQU. In this process, both weight training and searching for neural architecture and mixed-precision combination are conducted simultaneously. The hardware constraint is set as 7 BitOPs.

1) Mixed-Precision Integration: In this step, we transform the original full-precision supernet into different fixed-precision supernets and integrate them into a mixed-precision supernet. However, naively transforming the full-precision network into low-precision ones without retraining causes unbearable accuracy loss and leads to premature convergence to an inefficient model. In consequence, we propose a warm-up strategy to retrain the fixed-precision supernet with a few epochs and adopt the bit inheritance method proposed in [15] to inherit weights from higher-precision supernet to acquire a mixed-precision supernet which is beneficial for joint optimization. For example, given precision choices $PC = [K_1, K_2, K_3], K_1 > K_2 < K_3$, we first fix the precision of the supernet to K_1 and perform warm-up training to obtain the corresponding model weight. Then, when conducting warm-up training on the supernet with precision K_2, we first initialize its model weights as supernet with K_1 bits, as shown in Fig. 1(d). The same strategy is also adopted for the supernet with a fixed precision of K_3. Finally, we store model parameters with different precisions in the same supernet, constituting a mixed-precision supernet model for subsequent optimization. Challenges and details of the warm-up strategy will be explained in Section IV.

2) Joint Optimization: The joint optimization procedure conducts QPS, NAS, and QAT simultaneously. As Fig. 5 shows, during the joint optimization, the quantized weight is updated by applying (6), while the combination of neural architecture and mixed-precision choice is searched by utilizing (8). To give concrete guidance of hardware constraint, we use the hardware-aware optimization method discussed in Section III-B in the program. As Fig. 5 indicates, when the computation operations of the searched model greatly exceed the given constraint, it shrinks rapidly. However, different precision networks require different parameters s and k [as indicated in (1)] to control their quantization performance. Conventional training method for a specific network is not able to tackle this challenge. We propose a precision-transfer method to keep the training curve stable when various different precision combinations are searched together. The details of the precision-transfer method will be clarified in Section V.

3) Retraining: Finally, we retrain the quantized network that is obtained during the joint optimization under the given optimization constraints. Compared with the conventional QAT approach, the retraining process consumes a shorter time by inheriting weights from the joint optimization procedure.

IV. DAQU: Warm-Up for Unbiased Searching

Existing QAT works [11], [13], [14] have announced that directly training a quantized model from scratch results in inferior performance when compared with retraining with inheriting full-precision weights. Consequently, we use the pretrained OFA supernet as the initial supernet model for the whole joint optimization process in this article. Based on the basic setup, we first analyze the defects of existing works on combining mixed-precision quantization and NAS in Section IV-A and show the effectiveness of our proposed warm-up strategy in Section IV-B.

A. Dilemma of Pretraining Supernet or Not

Existing joint optimization works of NAS and quantization either start from an untrained supernet [Quantization-aware NAS in Fig. 1(c)] or pretrain a supernet that shares weight among various subnets [Quantized Weight-Sharing Supernet in Fig. 1(b)]. However, both designs lose optimality during the optimization process.

Searching from an untrained supernet causes biased guidance, due to the loss of knowledge about the convergence of corresponding neural architecture. As Fig. 6(a) shows, the initial accuracy of all quantized subnets is nearly 0.1% and the Pearson rank between accuracy and computation operations is close to 0, which indicates that the optimization engine cannot distinguish the difference among various neural architectures. To make it worse, the accuracy of network models can be significantly improved in the initial few epochs (proved in Fig. 7), allowing the accuracy of the trained subnets to quickly outperform the untrained ones, which results in inevitable bias. For instance, two quantized subnets with the same precision setting, denoted as $\mathcal{N}(\alpha_0)$ and $\mathcal{N}(\alpha_1)$, in which $\mathcal{N}(\alpha_1)$ is a more quantization-friendly model that expected to acquire in our optimization engine

$$\begin{cases} \zeta_{\text{val}}\left(F\left(\omega_0^0, \mathcal{N}(\alpha_0), \mathcal{Q}(\gamma)\right)\right) \approx \zeta_{\text{val}}\left(F\left(\omega_1^0, \mathcal{N}(\alpha_1), \mathcal{Q}(\gamma)\right)\right) \\ \zeta_{\text{val}}\left(F\left(\omega_0^*, \mathcal{N}(\alpha_0), \mathcal{Q}(\gamma)\right)\right) > \zeta_{\text{val}}\left(F\left(\omega_1^*, \mathcal{N}(\alpha_1), \mathcal{Q}(\gamma)\right)\right) \end{cases} \tag{9}$$

where ω^0 and ω^* represent initialized (0 epoch) and well-retrained weight, respectively. However, without retraining before searching, the optimization engine will randomly pick one of these two subnets. Suppose that $\mathcal{N}(\alpha_0)$ is selected and trained in the first epoch, then the result can be described as

$$\zeta_{\text{val}}\left(F\left(\omega_0^1, \mathcal{N}(\alpha_0), \mathcal{Q}(\gamma)\right)\right) < \zeta_{\text{val}}\left(F\left(\omega_1^1, \mathcal{N}(\alpha_1), \mathcal{Q}(\gamma)\right)\right) \tag{10}$$

so that the expected subnet $\mathcal{N}(\alpha_1)$ cannot be acquired by this optimization engine.

On the other hand, training a weight-sharing supernet cannot guarantee acquiring an optimal subnet that is expected in our

Models	Without warm up	50 epochs for warm up	100 epochs for warm up	OQAT (1200 epochs)
Pearson Rank	0.033	0.350	0.560	0.788

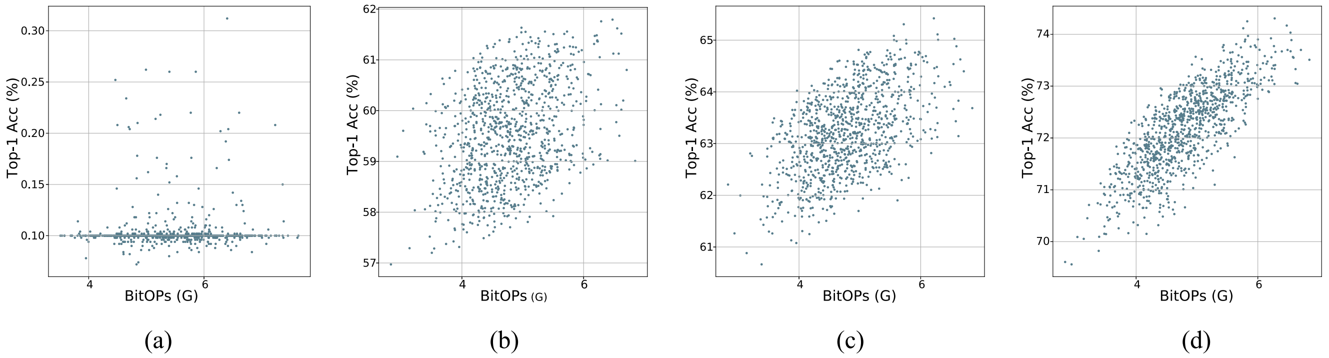


Fig. 6. Accuracy distribution of sampled 1000 quantized subnets with different warm-up settings. (a) Without warm-up. (b) Taking 50 epochs for warmup. (c) Taking 100 epochs for warmup. (d) OQAT [15] that takes 1200 epochs.

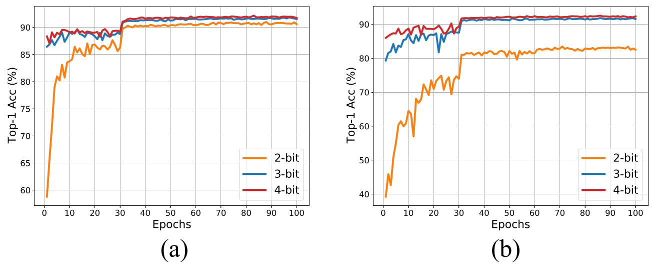


Fig. 7. QAT curve for different precision choices on typical neural networks. (a) VGG. (b) ResNet.

optimization engine, as illustrated in Table I. Specifically, the rankings of different subnets are still varying before and after retraining. Furthermore, as we can notice in Fig. 6(d), the Pearson rank in OQAT is close to 1, indicating that the relation among various subnets is quite stable. Therefore, searching from a well-trained supernet can lead to significant training overheads and suboptimal results.

B. Warm-Up Strategy Design

To tackle the dilemma above, we need to acquire a supernet that reveals the potential performance of all subnets and avoid biased architecture searching. Considering the training process below

$$\Delta\omega^i = \frac{\partial \zeta_{\text{train}}\big(F\big(\omega^i, \mathcal{N}(\alpha), \mathcal{Q}(\gamma)\big)\big)}{\partial \omega^i}$$
$$\omega^{i+1} = \omega^i + lr \cdot \Delta\omega^i \tag{11}$$

where ω^i denotes the weight of the ith training epoch, and lr represents the learning rate during training. From our observation in Fig. 7, the training curve is extremely steep at the beginning and gradually becomes smoother, converging to an optimal value. In consequence, after training a few epochs m, the subnet can partially reflect the potential performance of the well-trained subnet. We summarize it as an assumption.

Assumption 1: Once the supernet is trained for a few epochs m ($m \geq 20$), all the subnets derived from the supernet have obtained sufficient training to reflect their potential performance. That is to say, after training m epochs, the difference between different subnets is only a small perturbation δ compared to those well-trained subnets, where δ satisfies a normal distribution. It can be described as

$$\begin{aligned}&\zeta_{\text{val}}\big(F(\omega_0^m, \mathcal{N}(\alpha_0))\big) - \zeta_{\text{val}}\big(F(\omega_1^m, \mathcal{N}(\alpha_1))\big) + \delta\\ &\quad = \zeta_{\text{val}}\big(F(\omega_0^*, \mathcal{N}(\alpha_0))\big) - \zeta_{\text{val}}\big(F(\omega_1^*, \mathcal{N}(\alpha_1))\big)\\ &\quad\text{where}\ \ \delta \sim \mathcal{N}\Big(\mu, \sigma^2\Big).\end{aligned} \tag{12}$$

μ and σ^2 are the mean and variance of the normal distribution. In the overall optimization framework, they are determined by the given pretraining epochs m.

The statistical distribution and trend shown in Fig. 6 meet the assumption we proposed. Increasing the number of pretraining epochs for the supernet can effectively reduce the variance of the perturbation δ, making the distribution of the supernet closer to a well-trained one in Fig. 6(d). The variance σ^2 of δ has a significant impact on the performance of our optimization framework. Specifically, a variance σ^2 that is too small artificially prunes the optimization space and causes the optimization framework to prematurely reach a local optimum; while a variance that is too large can prevent effective convergence of the overall optimization and cause oscillations back and forth in the optimization space. In this article, we use the concept *warm-up* to achieve the tradeoff between these two extremes. Our warm-up strategy pretrains supernet by adopting the step size method adopted in OQAT, but only takes fewer epochs. Evaluation in Section VII-C shows that taking 50–100 epochs for warm-up can achieve the best performance. Therefore, we take 50 epochs for warm-up in our experimental results in Table II.

V. DAQU: Precision-Transfer for Stable Training

A. Expanding Step Size for Various Precision Settings

As discussed in [20], co-searching network architectures and precision by activating all the paths of the supernet can easily explode the memory consumption. A natural approach is to adopt hard GS proposed in [32] and [36] that actives only one path of all subnets. However, directly using hard GS results in unstable searching. Specifically, co-searching for network architecture and precision setting by hard GS demonstrates an ineffective search direction, and sequential training of a fixed network with multiple precision choices can not achieve competitive accuracy. The main reason for the aforementioned problem is unstable QAT during searching. As presented in Fig. 7, training with low precision requires more epochs to achieve an acceptable performance. However, activating one path using hard GS only improves the performance of specific precision setting, but obstacles to optimizing subnets with other precision choices.

Assuming the same step size parameter s is used for network layers with different precision settings, the detailed update strategy during the joint optimization process can be expressed as

$$\frac{\partial A^{l+1}}{\partial s^l} = \frac{\partial A^{l+1}}{\partial \hat{A}^l}\frac{\partial \hat{A}^l}{\partial s^l} \tag{13}$$

$$\frac{\partial A^{l+1}}{\partial A^l} = g^l \hat{\omega}^l_{\gamma_i} \tag{14}$$

where A^l and A^{l+1} denote the feature maps of the lth and $l+1$th layer, respectively. s^l represents the step size of lth layer and the gradients are estimated by the straight-through estimator (STE) [9]. Besides, $\hat{w}^l_{\gamma_i}$ represents the quantized weight under given precision setting γ_i and g^l is the gradient scale factor adopt in LSQ [11] for layer l. However, network models with different precision settings require different step size parameters s to achieve the best performance [50].

Naturally, one possible solution is to expand s into a learnable vector $\boldsymbol{s}$, where each s_{γ_i} is responsible for a candidate precision setting from precision set $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$. Consequently, the update process for each step size $s^l_{\gamma_i}$ can be depicted in

$$\frac{\partial A^{l+1}}{\partial s^l_{\gamma_i}} = \begin{cases} g^l \hat{w}^l_{\gamma_i} \dfrac{\partial \hat{A}^l_{\gamma_i}}{\partial s^l_{\gamma_i}}, & \text{if updating step size for activation} \\ g^l \hat{A}^l_{\gamma_i} \dfrac{\partial \hat{w}^l_{\gamma_i}}{\partial s^l_{\gamma_i}}, & \text{if updating step size for weights.} \end{cases} \tag{15}$$

B. Scaling Data Distribution During Backward

As indicated in [24], [25], and [26], the weight distributions for different precision settings are quite different. However, the update process in (15) cannot always acquire the accurate weight and activation data of precision setting γ_i, as we use the hard GS method that only activates one path of all subnets. Supposing that γ_i is activated, we cannot use the weight $w^l_{\gamma_i}$ to update the step size $s^l_{\gamma_j}$ directly. In consequence, we introduce another parameter ρ to scale the quantized weight data from a given precision setting γ_i to desired one γ_j. Based on the scaling parameter ρ, the update process of step size $s^l_{\gamma_j}$ is

$$\frac{\partial A^{l+1}}{\partial s^l_{\gamma_j}} = \begin{cases} g^l \rho_{i,j} \hat{w}^l_{\gamma_i} \dfrac{\partial \hat{A}^l_{\gamma_i}}{\partial s^l_{\gamma_i}} & \text{updating step size for activation} \\ g^l \rho_{i,j} \hat{A}^l_{\gamma_i} \dfrac{\partial \hat{w}^l_{\gamma_i}}{\partial s^l_{\gamma_i}} & \text{updating step size for weights.} \end{cases} \tag{16}$$

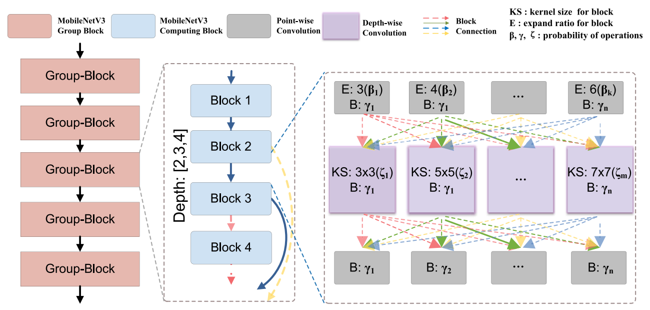


Fig. 8. Supernet architecture adopted in DAQU. The supernet follows architecture settings in OFA but provides layer-wise mixed-precision settings.

Generally, we propose the precision-transfer strategy, including expanding step size parameters for every precision setting and providing a scaling parameter for different data distributions. By employing the proposed precision-transfer training optimization strategy, we are able to train all subnets while ensuring training stability, with only one subnet being activated at a time.

VI. Experimental Setup

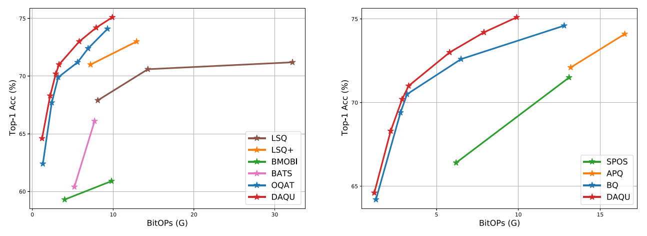
Supernet Architecture: Fig. 8 gives a glimpse of our proposed supernet architecture. Inherited from OFA, it constructs the NN architecture using the MobileNetV3 [37] blocks as the backbone. The whole supernet consists of 5 group-blocks and the dimension of the search space includes the depth for each group-block, kernel size, expand ratio, and precision setting for each block. Specifically, each block includes three convolution layers: two point-wise convolution (PWConv) layers and one depth-wise convolution (DWConv) layer. We can set different expand ratios for the first PWConv layer and the second PWConv layer follows the same expand ratio. Besides, the choices of kernel size for each DWConv layer range from 3×3 to 7×7. As we adopt a layer-wise mixed-precision supernet, each convolution layer (both PWConv and DWConv) can be set with different precision choices. Following the conventional quantization procedure [9], [13], we keep the first layer at full precision for both activations and weights. Herein, the optimizable parameters for kernel size, expand ratio, and precision setting for each block are denoted by ζ, β, and γ, respectively.

Search Space: According to our supernet architecture in Fig. 8, assume that each block has three selectable expand ratios for the PWConv layers and three kernel size options for the DWConv layer. In addition, for each convolution layer, we provide three available precision selections. Thus, each block has $(3 \times 3 \times 3^3) = 243$ components, each group-block has $243^2 \times (1 + 243 \times (1 + 243)) \approx 3.52 \times 10^9$ candidates. Thus, without considering the variable input resolution, our search space covers nearly $(3.52 \times 10^9)^5 \approx 5.40 \times 10^{47}$ subnets.

Software Environment: We implement the proposed framework using PyTorch. In all the training and searching stages, we employ a standard SGD optimizer with Nesterov momentum 0.9 and weight decay $5e^{-4}$ and use a stepLR scheduler for learning rate decay. We take 50 epochs to warm up each fixed-point supernet and adopt the bit-inheritance method proposed

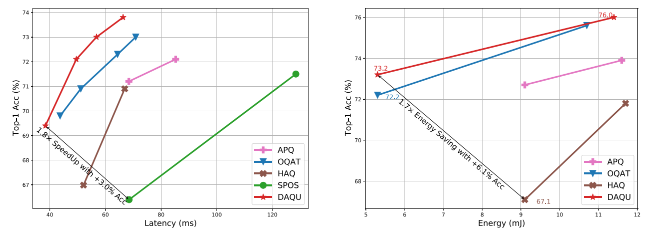
TABLE II
COMPARISON WITH SOTA EFFICIENT MODELS WITH MIXED LOW-PRECISION AND HIGH-PRECISION DAQU MODELS (DENOTED AS DAQU-LP AND DAQU-HP) UNDER HARDWARE CONSTRAINTS

Categories	Methods	Models	Precision (W/A)	BitOPs (G)	Top-1 Acc. (%)
NAS-then-Quantization	LSQ	OFA-Small	4	7.2	71.2
	LSQ	OFA-Middle	4	9.3	73.1
	LSQ	OFA-Large	4	16.6	74.3
Quantization-aware NAS	SPOS	ResNet-18	1,2,3,4	6.2	66.4
	SPOS	ResNet-34	1,2,3,4	13.1	71.5
	APQ	APQ-A	4,6,8	13.2	72.4
	APQ	APQ-B	4,6,8	16.5	74.1
Quantized Weight-sharing Supernet	OQAT	OQAT-4bit-M	4	6.9	72.4
	OQAT	OQAT-4bit-L	4	9.3	74.1
	BQ	BQ-A	2,3,4	6.3	72.5
	BQ	BQ-B	2,3,4	8.8	74.3
Single-stage Joint Searching with QPS, QAT and NAS	DAQU	DAQU-lp-S	2,3,4	**5.8**	73.0
	DAQU	DAQU-lp-L	2,3,4	7.6	**74.1**
	DAQU	DAQU-hp-S	2,4,8	**6.2**	73.2
	DAQU	DAQU-hp-L	2,4,8	9.6	**75.2**



(a) (b)

Fig. 9. Comparison with SOTA NAS and quantization joint optimization models on the ImageNet dataset. (a) Comparison with existing fix-precision models (LSQ, LSQ+, OQAT, BATS, and BMobi). (b) Comparison with existing mixed-precision models (SPOS, APQ, and BQ).



(a) (b)

Fig. 10. Hardware-aspect evaluation for SOTA models. (a) Latency evaluation on BISMO. (b) Energy evaluation on BitFusion.

in [15] during our mixed-precision integration procedure. Furthermore, we spend another 50 epochs to search a desired subnet and retrain it with 30 epochs. Need to note that, we only need to integrate the mixed-precision supernet once and it can be used for all different searching situations.

Hardware Platforms: The hardware platforms we adopted to measure performance are BitFusion [21], a SOTA ASIC accelerator for mixed-precision models, and BISMO [22], a scalable bit-serial accelerator for FPGA implementation. Since the basic computational block of the BitFusion architecture is a 2-bit MAC unit, which is extended spatially, we use $[2, 4, 8]$ as the optional model precision configurations to fully exploit the computational potential of the BitFusion architecture. In contrast, BISMO is a bit-serial-based FPGA implementation that can flexibly support various precisions. Therefore, for a fair comparison, we adopt the same mixed-precision configurations as BQ [16], which are $[2, 3, 4]$.

VII. EVALUATION

A. DAQU Versus SOTA Models

Table II presents a comparison with other NAS and Quantization joint optimization strategies. As indicated by the results, DAQU outperforms SOTA models under both fixed and mixed-precision configurations.

Comparison With Fixed-Precision Models: Fig. 9(a) shows that our searched models outperform existing best-performance fixed-precision methods, including OQAT [15], LSQ [11], LSQ+ [23], BATS [17], and BMobi [38]. We can easily notice that DAQU achieves a better Pareto curve than SOTA fix-precision models. Our benefit over OQAT is mainly based on introducing mixed-precision settings into the whole optimization space, as demonstrated in Section II-A.

Comparison With SOTA Mixed-Precision Models: We further compare with existing mixed-precision Quantization-aware NAS and Quantized Weight-sharing Supernet works, including APQ [19], SPOS [18], and BQ [16]. Noticing that under the mixed-precision setting, DAQU still produces an improved Pareto frontier. Our results demonstrate that a single-stage joint optimization method with QPS, QAT, and NAS can boost the performance of the quantized network, and achieve better accuracy and BitOPS tradeoff than the SOTA Quantized Weight-sharing Supernet method. The improvement is mainly due to the direct searching for quantized subnet rather than training the quantized supernet, as indicated in Fig. 2.

B. Hardware-Network Co-Exploration

To further validate the effectiveness of our hardware-aware optimization approach, we compare DAQU with SOTA methods on specific hardware platforms, as shown in Fig. 10.

Latency-Constrained Optimization on BISMO: As shown in Fig. 10(b), our DAQU outperforms SOTA mixed and fixed-precision methods. Specifically, our model can attain a better Pareto curve than OQAT when considering given latency constraints. Compared with SPOS, our framework can achieve 1.8× speedup with 3.0% accuracy improvement. We demonstrate our hardware-aware optimization method in Section III-B is able to appropriately exploit hardware features.

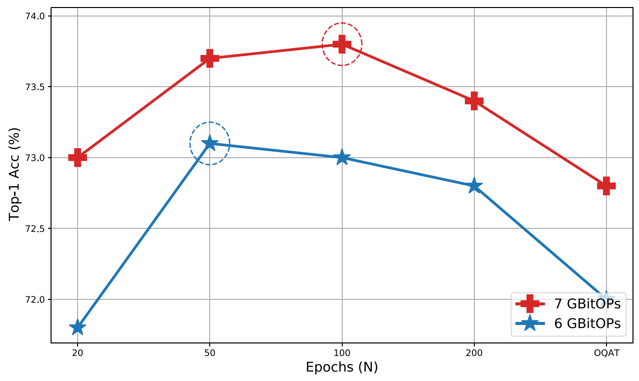


Fig. 11. Performance results with different warm-up epochs. Noticing that both 50 and 100 epochs have similar performance.

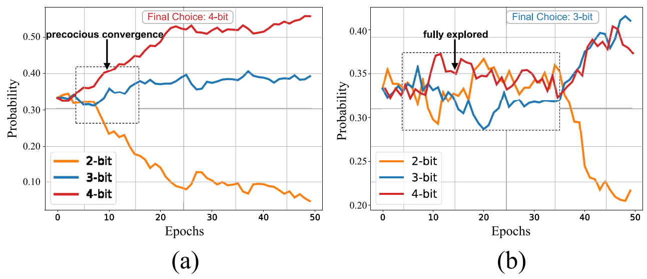


Fig. 12. Probability variation curves of precision search during the joint optimization stage. We compare the precision search curves under two strategies—with and without precision-transfer. (a) Precision searching without precision-transfer. (b) Precision searching with precision-transfer.

Energy-Constrained Optimization on BitFusion: We further compare DAQU with SOTA models under an energy-constrained optimization setting. Fig. 10(b) shows the results evaluated on BitFusion. Our models attain better tradeoffs between accuracy and energy than both SOTA mixed-precision and fixed-precision models under specific energy constraints. Specifically, our model improves 6.1% accuracy (from 67.1% to 73.2%) while saving 1.7× energy compared with the baseline model (MobileNetv2+HAQ). The result steadily shows the advantages of the DAQU framework for efficient model inference.

C. Impact of Warm-Up Epochs

To further explore the impact of different epochs, we conduct our DAQU with different warm-up epochs in Fig. 11. As one can see, the warm-up setting with small epochs (20 epochs) suffers from unstable and biased searching, leading to mediocre accuracy. Large epochs (200 epochs and OQAT) warm-up also performs inferior accuracy due to pruned search space by long-time QAT. An appropriate epoch, as circled in Fig. 11, is either 50 or 100 epochs. To reduce the whole warm-up epochs, we choose 50 epochs in this article.

D. Effectiveness of Precision-Transfer

To enable sufficient training of network weights under various mixed-precision combinations during joint optimization, we propose the precision-transfer strategy. Specifically, although only one subnet is activated during the forward pass, backpropagation from this subnet allows us to concurrently update the weight parameters of subnets with different precisions. This ensures all layer weights receive fair treatment regarding their corresponding precision settings.

Fig. 12 compares the precision searching curves under two strategies - with and without precision-transfer. As we can notice, without the precision-transfer strategy, the searching curves of precision settings rapidly converge and eventually select the highest precision (4-bit) as the final search result. This aligns with our observation in Fig. 7 that higher precisions are easier to train. Meanwhile, without precision transfer, the weights under other precisions cannot be well trained, resulting in the overall search process converging too quickly. In contrast, when we adopt the precision-transfer strategy during joint optimization, various mixed-precision combinations are fully explored, as shown in Fig. 12(b), and the precision 3-bit is eventually selected as an effective tradeoff between model accuracy and inference performance.

TABLE III
ABLATION STUDY ON OUR PROPOSED APPROACHES

Methods	BitOPs(G)	Top1-Acc. (%)
Naive Optimization	6.2	70.2
DAQU w/o precision-transfer)	6.4	71.3
DAQU w/o warm-up	6.1	71.6
DAQU	6.2	**72.8**

E. Overall Ablation Study

In Table III, we validate the advantages of approaches proposed in Sections IV and V. The Naive Optimization method is a basic differentiable optimization engine as in [20]. The models in the second and third rows are searched by our DAQU optimization engine without precision-transfer and warm-up strategy, respectively. With similar BitOPs, the quantized models obtained from our provided optimization engine surpass Naive Optimization models by more than 1%, despite lacking full-scale optimization. The last row shows that integrating both precision-transfer and warm-up into DAQU can finally achieve a 2.6% accuracy improvement. The huge accuracy improvement verifies both warm-up and precision-transfer strategies are beneficial for stable optimization among various neural architectures and quantization settings.

Fig. 13 further validates the effectiveness achieved by DAQU. DAQU can acquire the Pareto frontier under the accuracy and latency tradeoff for the BitFusion accelerator. Compared with the sequential optimization (optimize architecture parameters and precision settings sequentially) baseline, DAQU can achieve at least 1.64× speedup while maintaining the same model accuracy. Under the high-accuracy (over 73%) requirement, DAQU can attain an even higher speedup of 1.77×. Furthermore, even discarding some optimization strategies (DAQU w/o precision-transfer or warm-up), the results explored by our optimization engine still represent a better Pareto frontier compared to sequential optimization, as indicated in Fig. 13. However, organically combining the precision-transfer and warm-up strategies can better maintain the stability of the joint optimization process and prevent

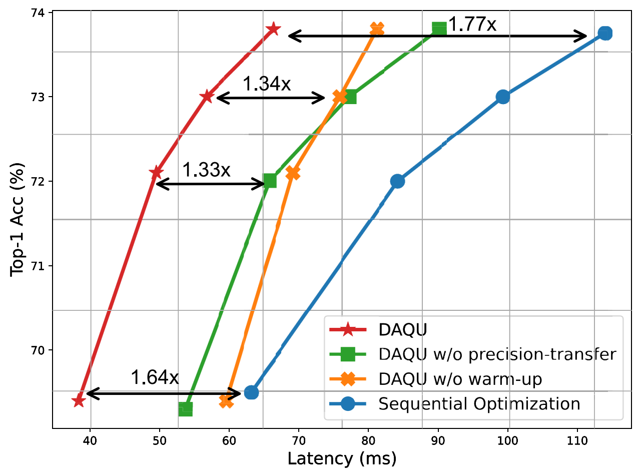


Fig. 13. Accuracy and latency tradeoff for DAQU, DAQU w/o precision-transfer, DAQU w/o warm-up, and the sequential optimization baseline.

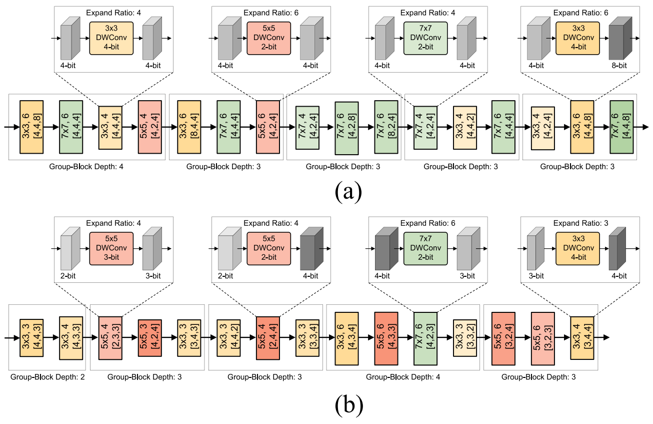


Fig. 14. Searched mixed-precision network model for different accelerators. (a) Searched mixed-precision (with precision setting [2, 4, 8]) network model for BitFusion accelerator. (b) Searched mixed-precision (with precision setting [2, 3, 4]) network model for BISMO accelerator.

precocious convergence. This enables the overall DAQU framework to demonstrate superior performance compared to using only some optimization strategies, as we have observed. Specifically, DAQU can achieve 1.34× and 1.33× speedup over frameworks without warm-up and precision-transfer optimization strategies, respectively, at the same model accuracy.

F. Network and Quantization Setting Searched by DAQU

Fig. 14 shows the two different mixed-precision networks searched by DAQU. We optimize the accuracy of mixed-precision models under latency constraints for two different hardware platforms (BitFusion and BISMO), exploring optimal network architectures under given latency constraints. As one can notice, optimizations for different hardware result in diverse network architectures. As a consequence, we will focus our discussion on explaining the reasons behind these divergent optimization results in the following.

1) *Kernel Size:* Compared to the mixed-precision network model searched by BISMO (BM-Net), the BitFusion-optimized model (BF-Net) favors larger kernel sizes. This is because the ASIC-implemented BitFusion has a larger on-chip compute array, which can better utilize on-chip resources to support computations with larger kernel sizes. Additionally, we notice that both searched networks favor larger kernel sizes in the middle blocks of the whole network. This implies that having a larger receptive field in the middle layers helps improve model accuracy.
2) *Expand Ratio:* Due to abundant computational resources in BitFusion, BF-Net chooses more aggressive expand ratios compared to BM-Net. However, despite the limited computational resources, BM-Net also generally opts for larger expand ratios in the last two group-blocks. This implies that the last two group-blocks contain rich information, which requires expanding channels to sufficiently improve model accuracy.
3) *Precision Setting:* Due to hardware constraints, BF-Net and BM-Net adopt different mixed-precision combinations (details in Section VI), leading to divergent optimization directions in precision selection. For BF-Net, since 4-bit quantization usually maintains model accuracy well while 2-bit has difficulty achieving adequate representational ability, most convolution layers opt for 4 bit with very few selecting 2-bit and 8-bit. In contrast, BM-Net fully utilizes different precision settings to constitute a mixed-precision network, efficiently selecting varied precisions based on the weight distribution of each layer during optimization. As shown in Fig. 14(b), 3-bit and 4-bit quantization adequately meets representational needs in most cases, while 2-bit is only effective in very few convolution layers.
4) *Block Depth:* The depth of most group blocks in both searched networks remains 3, implying that in most cases, 3-layer blocks sufficiently capture salient information from the current input.

VIII. Related Work

A. Network Quantization

Quantization [6], [7], [8], [9] is an orthodox approach for model compression, which compresses model size by reducing bit representation for weights and features. There are two main different research on quantization with and without (re)training. Recent works [41], [42], [43] show that 8-bit or even less bit can achieve similar accuracy compared with full precision networks using post-training. However, a bunch of works [10], [11], [12], [13], [14] with retraining have demonstrated comparable performance can be accomplished within 4-bit. So in the rest of this section, we pay to focus on QAT methods that provide lower bits for deployment.

With the consistent progress of quantization research, LSQ [11] provides SOTA performance on linear quantization, which is friendly to hardware. Instead of considering quantization methods, some training tricks are also considered in this process. DBLP [46] explores that we can use knowledge distillation to get higher accuracy. In [47], parallel methods are proposed to simultaneously accelerate the training process and improve quantized accuracy.

B. Neural Architecture Search

The study on NAS thrives in recent years [1], [4], [5] and they proved that NAS is an effective method that can both improve network performance and reduce the reliance on expertise or manual design. Early works use RL agents or EA algorithms to determine the neural architecture design. To further efficiently search for a neural architecture for scenario deployment on real-world devices, many different methods are proposed and persistently studied.

Speed-Up the Search Stage: To reduce training time, some works [44], [45] proposed weight-sharing methods to jointly train rather than iteratively train from scratch. More recently, works, such as OFA [1] and BigNas [2], supported diverse architectural settings without retraining, which can reduce computation costs. Instead of weight sharing, another line of work proves that there is no need to wait for this model to converge. Early termination is enough for analyzing the effectiveness of a specific model and resulting in less resource cost. Also, Atom-NAS [48] tries to combine the training and searching process together to accelerate NAS time.

Continuous Search Strategy: Traditional mainstream search methods regard NAS as a black-box optimization problem and make it inefficient to search. To tackle this problem, DARTS [27] proposed a novel method that relaxes the originally discrete search space continuously, which makes it possible to use gradients to efficiently optimize the architecture search space. Later, GDAS [32] proposes to use a differential architecture sampler in each training iteration to sample a subnet instead of optimizing all parameters simultaneously. By reducing memory occupation during the searching phase, GDAS further improves search efficiency.

IX. Conclusion

In this article, we analyze the weakness of existing joint optimization methods for architecture and quantization, including ignoring the importance of QAT and inconsistency between the original network and the searched network. To solve these problems, we present DAQU, a differentiable architecture and quantization joint optimization framework, to directly acquire an optimal quantized network by considering QPS, QAT, and NAS together. Furthermore, we propose warm-up and precision-transfer strategies to ensure stable searching among architectures and quantization precision settings. Compared with SOTA solutions, our DAQU framework consistently achieves considerable improvements in model accuracy and hardware-aspect computational cost saving. Conclusively, DAQU promises a generic way to unify the process of neural network architecture search and quantization for any perspective hardware platforms and datasets.

References

[1] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," 2019, *arXiv:1908.09791*.

[2] J. Yu et al., "BigNAS: Scaling up neural architecture search with big single-stage models," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 702–717.

[3] M. Tan et al., "MnasNET: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2820–2828.

[4] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," 2018, *arXiv:1812.00332*.

[5] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*.

[6] F. Li, B. Liu, X. Wang, B. Zhang, and J. Yan, "Ternary weight networks," in *Proc. Conf. Acoustics, Speech Signal Process. (ICASSP)*, 2016, pp. 1–5.

[7] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 187, pp. 1–30, 2018.

[8] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 2849–2858.

[9] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.

[10] X. Zhao, Y. Wang, X. Cai, C. Liu, and L. Zhang, "Linear symmetric quantization of neural networks for low-precision integer hardware," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–16.

[11] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–12.

[12] J. L. McKinstry et al., "Discovering low-precision networks close to full-precision networks for efficient embedded inference," 2018, *arXiv:1809.04191*.

[13] J. Choi et al., "Pact: Parameterized clipping activation for quantized neural networks," 2018, *arXiv:1805.06085*.

[14] S. Jung et al., "Joint training of low-precision neural network with quantization interval parameters," 2018, *arXiv:1808.05779*.

[15] M. Shen et al., "Once quantization-aware training: High performance extremely low-bit architecture search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 5340–5349.

[16] H. Bai, M. Cao, P. Huang, and J. Shan, "Batchquant: Quantized-for-all architecture search with robust quantizer," 2021, *arXiv:2105.08952*.

[17] A. Bulat, B. Martinez, and G. Tzimiropoulos, "Bats: Binary architecture search," in *Proc. Comput. Vis. (ECCV)*, 2020, pp. 309–325.

[18] Z. Guo et al., "Single path one-shot neural architecture search with uniform sampling," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 544–560.

[19] T. Wang et al., "APQ: Joint search for network architecture, pruning and quantization policy," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2078–2087.

[20] Y. Fu, Y. Zhang, Y. Zhang, D. Cox, and Y. Lin, "Auto-NBA: Efficient and effective search over the joint space of networks, bitwidths, and accelerators," 2021, *arXiv:2106.06575*.

[21] H. Sharma et al., "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2018, pp. 764–775.

[22] Y. Umuroglu, L. Rasnayake, and M. Själander, "BISMO: A scalable bit-serial matrix multiplication overlay for reconfigurable computing," in *Proc. 28th Int. Conf. Field Program. Logic Appl. (FPL)*, 2018, pp. 307–3077.

[23] Y. Bhalgat, J. Lee, M. Nagel, T. Blankevoort, and N. Kwak, "LSQ+: Improving low-bit quantization through learnable offsets and better initialization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 696–697.

[24] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8612–8620.

[25] Z. Dong, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer, "HAWQ: Hessian aware quantization of neural networks with mixed-precision," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 293–302.

[26] M. Rusci, A. Capotondi, and L. Benini, "Memory-driven mixed low precision quantization for enabling deep network inference on microcontrollers," in *Proc. Mach. Learn. Syst.*, 2020, pp. 326–335.

[27] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–13.

[28] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 4780–4789.

[29] B. Wu et al., "FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10734–10742.

[30] S. Yun and A. Wong, "Do all mobilenets quantize poorly? Gaining insights into the effect of quantization on depthwise separable convolutional networks through the eyes of multi-scale distributional dynamics," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 2447–2456.
[31] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "WRPN: Wide reduced-precision networks," 2017, *arXiv:1709.01134*.
[32] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1761–1770.
[33] C. D. Sa et al., "High-accuracy low-precision training," 2018, *arXiv:1803.03383*.
[34] G. Yang, T. Zhang, P. Kirichenko, J. Bai, A. G. Wilson, and C. D. Sa, "SWALP: Stochastic weight averaging in low-precision training," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7015–7024.
[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[36] S. Hu et al., "DSNAS: Direct neural architecture search without parameter retraining," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 12084–12092.
[37] A. Howard et al., "Searching for mobilenetv3," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1314–1324.
[38] H. Phan, Z. Liu, D. Huynh, M. Savvides, K.-T. Cheng, and Z. Shen, "Binarizing mobilenet via evolution-based searching," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13420–13429.
[39] R. Wang, M. Cheng, X. Chen, X. Tang, and C.-J. Hsieh, "Rethinking architecture selection in differentiable NAS," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–18.
[40] K. Choi, D. Hong, H. Yoon, J. Yu, Y. Kim, and J. Lee, "DANCE: Differentiable accelerator/network co-exploration," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, 2021, pp. 337–342.
[41] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," 2018, *arXiv:1806.08342*.
[42] M. Nagel, M. V. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1325–1334.
[43] D. Wu, Q. Tang, Y. Zhao, M. Zhang, Y. Fu, and D. Zhang, "EasyQuant: Post-training quantization via scale optimization," 2020, *arXiv:2006.16669*.
[44] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.
[45] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 550–559.
[46] A. Mishra and D. Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," 2017, *arXiv:1711.05852*.
[47] A. T. Elthakeb, P. Pilligundla, F. Mireshghallah, A. Cloninger, and H. Esmaeilzadeh, "Divide and conquer: Leveraging intermediate feature representations for quantized training of neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2880–2891.
[48] J. Mei et al., "Atomnas: Fine-grained end-to-end neural architecture search," 2019, *arXiv:1912.09640*.
[49] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–13.
[50] X. Huang et al., "SDQ: Stochastic differentiable quantization with mixed precision," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 9295–9309.

Lian Liu received the B.S. degree in computer science and technology from Nankai University, Tianjin, China, in 2021. He is currently pursuing the M.S. degree in computer science with the State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.

His research interests include computer architecture, hardware–software codesign, and deep learning model compression.

Ying Wang (Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from the Harbin Institute of Technology, Harbin, China, in 2007 and 2009, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 2014.

He is currently a Professor with ICT, CAS, and the Department of Computer Science, University of Chinese Academy of Sciences, Beijing. His research interests include computer architecture and VLSI design, specifically energy-efficient architecture, and machine learning accelerators.

Xiandong Zhao received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, Hubei, China, in 2017, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2023.

He currently serves as an Algorithm Engineer with AMD, Santa Clara, CA, USA. His current research interests include the field of computer architecture, and deep learning model compression and acceleration.

Weiwei Chen received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2016, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2022.

His research interests include hardware acceleration for deep learning, compilation frameworks for machine learning, and AutoML.

Huawei Li (Senior Member, IEEE) received the B.S. degree in computer science from Xiangtan University, Xiangtan, China, in 1996, and the M.S. and Ph.D. degrees from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 1999 and 2001, respectively.

She has been a Professor with ICT, CAS since 2008. She was a Visiting Professor with the University of California at Santa Barbara, Santa Barbara, CA, USA, from 2009 to 2010. Her current research interests include testing of very large-scale integration/SoC circuits, approximate computing architecture, and machine learning accelerators. She has published more than 200 technical papers and holds 34 Chinese patents in the above areas.

Prof. Li currently serves as the Secretary General of the China Computer Federation Technical Committee on Integrated Circuit Design and an Associate Editor of IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS and IEEE DESIGN & TEST. She served as the Steering Committee Chair for the IEEE Asian Test Symposium (ATS) from 2020 to 2022. She was the Technical Program Chair for ATS in 2007 and 2018, and the General Chair for ATS in 2014 and 2023. She has also served on the technical program committees for several IEEE/ACM conferences.

Xiaowei Li (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees in computer science from the Hefei University of Technology, Hefei, China, in 1985 and 1988, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 1991.

He was an Associate Professor with the Department of Computer Science and Technology, Peking University, Beijing, from 1991 to 2000. In 2000, he joined ICT, CAS, as a Professor. He is also a Professor with the Department of Computer Science, University of Chinese Academy of Sciences, Beijing. He has coauthored over 280 papers in journals and international conferences and holds 60 patents and 30 software copyrights. His current research interests include VLSI testing, design verification, and dependable computing.

Yinhe Han (Senior Member, IEEE) received the M.S. and Ph.D. degrees in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 2003 and 2006, respectively.

He is currently a Professor with ICT, CAS. His research interests include VLSI/SOC interconnection, testing, and fault tolerance.

Prof. Han is a recipient of the Best Paper Award at Asian Test Symposium 2003. He was the Program Co-Chair of Workshop of RTL and High Level Testing in 2009, and serves on the technical program committees of several IEEE and ACM conferences, including ATS and GVLSI. He is a member of CCF and ACM.