Single-Path Bit Sharing for Automatic Loss-Aware Model Compression

Jing Liu, Bohan Zhuang, Peng Chen, Chunhua Shen, Jianfei Cai, Fellow, IEEE, and Mingkui Tan, Member, IEEE

Abstract—Network pruning and quantization are proven to be effective ways for deep model compression. To obtain a highly compact model, most methods first perform network pruning and then conduct quantization based on the pruned model. However, this strategy may ignore that the pruning and quantization would affect each other and thus performing them separately may lead to sub-optimal performance. To address this problem, performing pruning and quantization jointly is essential. Nevertheless, how to make a trade-off between pruning and quantization is non-trivial. Moreover, existing compression methods often rely on some pre-defined compression configurations (i.e., pruning rates or bitwidths). Some attempts have been made to search for optimal configurations, which however may take unbearable optimization cost. To address these issues, we devise a simple yet effective method named Single-Path Bit Sharing (SBS) for automatic loss-aware model compression. To this end, we consider the network pruning as a special case of quantization and provide a unified view for model pruning and quantization. We then introduce a single-path model to encode all candidate compression configurations, where a high bitwidth value will be decomposed into the sum of a lowest bitwidth value and a series of re-assignment offsets. Relying on the single-path model, we introduce learnable binary gates to encode the choice of configurations and learn the binary gates and model parameters jointly. More importantly, the configuration search problem can be transformed into a subset selection problem, which helps to significantly reduce the optimization difficulty and computation cost. In this way, the compression configurations of each layer and the trade-off between pruning and quantization can be automatically determined. Extensive experiments on CIFAR-100 and ImageNet show that SBS significantly reduces computation cost while achieving promising performance. For example, our SBS compressed MobileNetV2 achieves 22.6× Bit-Operation (BOP) reduction with only 0.1% drop in the Top-1 accuracy.

Index Terms—Bit sharing, loss-aware model compression, network pruning, network quantization.

I. INTRODUCTION

Deep neural networks (DNNs) [42] have achieved great success in many challenging computer vision tasks, including image classification [19], [30], [41], object detection [47], [68], [76], image generation [6], [21], [23], and video analysis [73], [81], [93]. However, a deep model usually has a large number of parameters and consumes enormous computational resources, which presents great obstacles for many applications, especially on resource-constraint devices, such as smartphones. To reduce the number of parameters and computational overhead, many methods [32], [95], [99] have been proposed to perform model compression by removing the redundancy while maintaining the performance.

In recent years, we have witnessed the remarkable progress of model compression methods. Specifically, network pruning [32], [33] removes the uninformative modules and network quantization [36], [95] maps the full-precision values to low-precision ones. To obtain a highly compact model, most methods [27], [57], [91] first perform pruning and then conduct quantization based on the pruned model. However, performing pruning and quantization separately may lead to suboptimal performance as it ignores that the two procedures often affect each other. Therefore, it is essential and urgent to perform pruning and quantization jointly in practical applications, which, however, is nontrivial and may incur some new challenges.

First, finding the optimal trade-off between pruning and quantization is non-trivial as they would affect each other. For example, if a model is under-pruned, we can apply aggressive network quantization to the pruned model to achieve a high compression ratio. In contrast, if a model is over-pruned, the resulting model is more sensitive to the quantization noise. In this case, the quantization bitwidth of the pruned model has to be high to preserve the performance.

Second, to achieve better performance, one may assign different configurations (i.e., pruning rates and bitwidths) according...
We evaluate our SBS on CIFAR-100 and ImageNet. The convolutional operation and probability are the convolutional operation and probability. We explore the exponential search space using gradient-based optimization. The super-bitwidth is the highest bitwidth (i.e., 32-bit) in the search space, and the mixed-precision quantization problem can be formulated as a subset selection problem, which significantly reduces the number of parameters, computation cost, and optimization difficulty. Therefore, the mixed-precision quantization problem can be formulated as a subset selection problem, which significantly reduces the number of parameters, computation cost, and optimization difficulty. Relying on the single-path model, we further introduce learnable binary gates to encode the choice of bitwidth and learn the binary gates and network parameters jointly. In this way, the configurations of each layer can be automatically determined and the trade-off between pruning and quantization can be optimized.

Our main contributions are summarized as follows:

- We devise a novel single-path scheme that encapsulates multiple configurations in a unified single-path framework, which requires fewer model parameters compared with multi-path scheme.
- We transform the model compression into a subset selection problem and explicitly share the quantized values among various bitwidths. As a result, SBS enables the candidate configurations to learn jointly rather than separately and thus significantly reduces the computation cost and alleviates the optimization difficulty.
- We formulate the quantized representation as a gated combination of the lowest bitwidth representation and a series of re-assignment offsets. By training the binary gates and network parameters, the configuration of each layer and the trade-off between pruning and quantization can be automatically determined.
- We evaluate our SBS on CIFAR-100 and ImageNet over various network architectures. Extensive experiments show that the proposed method achieves the state-of-the-art performance. For example, on ImageNet, our SBS compressed MobileNetV2 achieves 22.6× Bit-Operation (BOP) reduction with only 0.1% performance decline in terms of the Top-1 accuracy.
II. RELATED WORK

Network Quantization: Network quantization represents the weights, activations and even gradients with low precision to yield compact DNNs. With low-precision integers [95] or power-of-two representations [44], the heavy matrix multiplications can be replaced by efficient bitwise operations, leading to much faster test-time inference and lower resource consumption. According to the quantization bitwidth, existing quantization methods can be roughly categorized into two categories, namely, fixed-point quantization [17], [29], [38], [39], [52], [72], [94], [95] and binary quantization [36], [50], [53], [63], [64], [66], [98]. To improve the quantization performance, extensive methods have been proposed to learn accurate quantizers [2], [4], [11], [17], [38], [94]. Specifically, given a convolutional layer, let \( x \) and \( w \) be the output activations of the previous layer and the weight parameters of given layer, respectively. First, following [7], [11], [38], one can normalize \( x \) and \( w \) into scale \([0, 1]\) by \( T_x(\cdot) \) and \( T_w(\cdot) \), respectively:

\[
\begin{align*}
  z_x &= T_x(x) = \text{clip} \left( \frac{x}{v_x}, 0, 1 \right), \\
  z_w &= T_w(w) = \frac{1}{2} \left( \text{clip} \left( \frac{w}{v_w}, -1, 1 \right) + 1 \right),
\end{align*}
\]

where \( v_x \) and \( v_w \) are trainable quantization intervals indicating the range of weights and activations to be quantized. Here, the function \( \text{clip}(v, v_{low}, v_{up}) = \min(\max(v, v_{low}), v_{up}) \) clips any number \( v \) into the range \([v_{low}, v_{up}]\). Then, one can apply the following function to quantize the normalized activations and parameters, namely \( z_x \in [0, 1] \) and \( z_w \in [0, 1] \), to discretized ones:

\[
D(z, s) = s \cdot \text{round} \left( \frac{z}{s} \right),
\]

where \( s \) denotes the normalized step size, \( \text{round}(x) = \left\lfloor x - 0.5 \right\rfloor \) returns the nearest integer of a given value \( x \) and \( \lceil \cdot \rceil \) is the ceiling function. Typically, for \( k \)-bit quantization, the normalized step size \( s \) can be computed by

\[
s = \frac{1}{2^k - 1}.
\]

Last, the quantized activations and weights can be obtained by \( Q_x(x) = T_x^{-1}(D(z_x, s)) = v_x \cdot D(z_x, s) \) and \( Q_w(w) = T_w^{-1}(D(z_w, s)) = v_w \cdot (2 \cdot D(z_w, s) - 1) \), where \( T_x^{-1}(\cdot) \) and \( T_w^{-1}(\cdot) \) denote the inverse functions of \( T_x(\cdot) \) and \( T_w(\cdot) \), respectively. In general, the function \( D(\cdot, \cdot) \) is non-differentiable. Following [36], [95], one can use the straight through estimation (STE) [1] to approximate the gradient of \( D(\cdot, \cdot) \) by the identity mapping, namely, \( \partial D(z, s) / \partial z \approx 1 \).

To reduce the optimization difficulty introduced by non-differentiable discretization, several methods have been proposed to approximate the gradients of \( D(\cdot, \cdot) \) [13], [56], [96]. Moreover, most previous works assign the same bitwidth for all layers [7], [17], [37], [38], [44], [65], [87], [95], [97], [98]. Though attractive for simplicity, setting a uniform precision places no guarantee on optimizing network performance, since different layers have different redundancy and arithmetic intensity. Therefore, several studies proposed mixed-precision quantization [5], [9], [15], [16], [78], [80], [83], [86], [90] that assigns different bitwidths according to the redundancy of each layer. In this paper, based on the proposed single-path bit sharing model, we devise an approach that efficiently searches for appropriate bitwidths for different layers through gradient-based optimization. Apart from quantization, our SBS also conducts pruning and automatically learns the trade-off between them, which often results in compact models with better performance.

Neural Architecture Search (NAS) and Pruning: NAS aims to automatically design efficient architectures. According to the search algorithm, existing methods either based on reinforcement learning [25], [62], [100], evolutionary search [58], [67], [89] or gradient-based methods [3], [48], [85]. In particular, gradient-based NAS has gained increased popularity, where the search space is relaxed to be continuous, allowing efficient architecture search using gradient descent. Depending on whether each operation can be added via a separate path or not, the search space can be categorized into multi-path design [3], [48] and single-path formulation [74], [75].

While prevailing methods optimize the network topology, we focus on searching optimal pruning and quantization configurations for a given network. Moreover, network pruning can be treated as fine-grained NAS [14], [22], [49] which removes redundant modules to reduce the model size and accelerate the run-time inference speed, giving rise to methods based on weight pruning [24], [27], [28], [51], filter pruning [32], [33], [59], [99], or layer pruning [8], [35], [46], [84], etc. Apart from filter pruning, we also perform network quantization to obtain more compact networks.

AutoML for Model Compression: Recently, much effort has been devoted to automatically determining the pruning rate [14], [49], [77] or the bitwidth [5], [55], [86] of each layer, either based on reinforcement learning [31], [55], [80], evolutionary search [49], [82], gradient optimization [5], [18], [86], etc. To increase the compression ratio, several methods have been proposed to jointly optimize pruning and quantization strategies. In particular, some work only support weight quantization [34], [77], [91] or use fine-grained pruning [34], [88]. However, the resultant models cannot be implemented efficiently on edge devices. To handle this, several methods [82], [86], [92] have been proposed to consider filter pruning, weight quantization, and activation quantization jointly. Compared with these methods, we carefully design the compression search space by sharing the quantized values between different candidate configurations, which significantly reduces the number of parameters, search cost, and optimization difficulty. Compared with those methods that share the similarities of using quantized residual errors [10], [20], [45], [79], our proposed method recursively uses quantized residual errors to decompose a quantized representation into a set of candidate bitwidths and parameterize the bitwidth selection via a series of binary gates. Compared with Bayesian Bits [79], our SBS differs in several aspects: 1) We theoretically verify the theorem of quantization decomposition and it can be applied to non-power-of-two bitwidths (See Section V-E), which is a
general case of the one in Bayesian Bits. 2) The optimization problems are different. Specifically, we formulate model compression as a single-path subset selection problem while Bayesian Bits casts the optimization of the binary gates into a variational inference problem that requires more relaxations and hyperparameters. 3) Our compressed models with less or comparable BOPs outperform those of Bayesian Bits by a large margin on ImageNet (See Table IV).

III. PROPOSED METHOD

Given a pre-trained model, we focus on automatic model compression for pruning and quantization jointly, which poses two new challenges. First, finding the optimal trade-off between pruning and quantization is non-trivial since they may affect each other. Second, to determine the optimal configurations (e.g., pruning rates and bitwidths) for each layer, one may consider different configurations as different paths and reformulate the configuration search problem as a path selection problem [86], as shown in Fig. 1(a). However, when the search space becomes large, it suffers from a huge number of parameters and high computation cost. Moreover, different candidate configurations are trained separately and thus the optimization of the compressed model may become more challenging.

In this paper, we first provide a unified view for model compression so that we can perform pruning and quantization jointly. Specifically, we consider network pruning as a special case of quantization and formulate the joint pruning and quantization problem as a mixed-precision quantization problem. We then propose a Single-path Bit Sharing (SBS) scheme to encode all candidate configurations into a single path model, as shown in Fig. 1(b). It is worth mentioning that our SBS has much fewer model parameters than the most related work [86] and thus requires less computation cost. Moreover, we are able to cast the mixed-precision problem into a subset selection problem and alleviate the optimization difficulties. In the following subsections, we will illustrate each component of our method.

A. Single-Path Bit Sharing Decomposition

To illustrate the single-path bit sharing decomposition, we begin with an example of 2-bit quantization for $z \in \{z_x, z_w\}$. Specifically, we use the following equation to quantize $z$ to 2-bit using Eqs. (3) and (4):

$$z_2 = D(z, s_2), \quad s_2 = \frac{1}{2^2 - 1},$$

where $z_2$ and $s_2$ are the quantized value and the step size of 2-bit quantization, respectively. Due to the large step size, the residual error $z - z_2 \in [-s_2/2, s_2/2]$ may be large and results in a significant performance decline. To reduce the residual error, an intuitive way is to use a smaller step size, which indicates that we quantize $z$ to a higher bitwidth. Since the step size $s_4 = 1/(2^4 - 1)$ in 4-bit quantization is a divisor of the step size $s_2$ in 2-bit quantization, the quantized values of 2-bit quantization are shared with those of 4-bit quantization. Based on 2-bit quantization, the 4-bit counterpart introduces additional unshared quantized values. In particular, if $z_2$ has zero residual error, then 4-bit quantization maps $z$ to the shared quantized values (i.e., $z_2$). In contrast, if $z_2$ has non-zero residual error, 4-bit quantization is likely to map $z$ to the unshared quantized values. In this case, 4-bit quantization can be regarded as performing quantized value re-assignment based on $z_2$. Such a re-assignment process can be formulated as

$$z_4 = z_2 + r_4,$$

where $z_4$ is the 4-bit quantized value and $r_4$ is the re-assignment offset based on $z_2$. To ensure that the results of re-assignment fall into the 4-bit quantized values, the re-assignment offset $r_4$ must be an integer multiple of the step size $s_4$. Formally, $r_4$ can be computed by performing 4-bit quantization on the residual error of $z_2$:

$$r_4 = D(z - z_2, s_4), \quad s_4 = \frac{1}{2^4 - 1}.$$  

According to Eq. (6), a 4-bit quantized value can be decomposed into the 2-bit representation and its re-assignment offset. Similarly, an 8-bit quantized value can also be decomposed into the 4-bit representation and its corresponding re-assignment offset. In this way, we can generalize the idea of decomposition to arbitrary effective bitwidths as follows.

**Theorem 1:** (Quantization Decomposition) Let $z \in \{0, 1\}$ be a normalized full-precision input, and $\{b_j\}_{j=1}^K$ be a sequence of candidate bitwidths. If $b_j$ is an integer multiple of $b_{j-1}$, i.e., $b_j = \gamma_j b_{j-1} (j > 1)$, where $\gamma_j \in \mathbb{Z}^+ \setminus \{1\}$ is a multiplier, then the quantized approximation $z_{b_K}$ can be decomposed as:

$$z_{b_K} = z_{b_1} + \sum_{j=2}^{K} r_{b_j},$$

where $r_{b_j} = D(z - z_{b_{j-1}}, s_{b_j}),$ $z_{b_j} = D(z, s_{b_j}),$ $s_{b_j} = \frac{1}{2^{b_j} - 1}$.  

From Theorem 1, the quantized representation $z_{b_K}$ can be decomposed into the sum of the lowest bitwidth representation $z_{b_1}$ and a series of recursive re-assignment offsets. In this way, we can gradually reduce the error brought from quantization by introducing the recursive re-assignment offsets. To measure the quantization error, we introduce the following corollary.

**Corollary 1.** (Normalized Quantization Error Bound) Given $z \in [0, 1]^d$ being a normalized full-precision vector, $z_{b_K}$ being its quantized vector with bitwidth $b_K$, where $d$ is the cardinality of $z$. Let $c_K = \frac{\|z - z_{b_K}\|}{\|z\|}$ be the normalized quantization error, then the following error bound w.r.t. $K$ holds:

$$|c_K - c_{K+1}| \leq \frac{C}{2^{b_K} - 1},$$

where $C = \frac{d}{\|z\|}$ is a constant.

To empirically demonstrate Corollary 1, we perform quantization on a random toy data $z \in [0, 1]^{100}$ and show the normalized quantization error change $|c_K - c_{K+1}|$ and error bound $\frac{C}{2^{b_K} - 1}$ in Fig. 2. From the results, the normalized quantization error change decreases quickly as the bitwidth increases and is
bounded by $C/(2^{b_K} - 1)$. We put the proof of Corollary 1 in the supplementary material, available online.

Note that in Theorem 1, both the smallest bitwidth $b_1$ and the multiplier $\gamma_j$ can be set to arbitrary appropriate integer values (e.g., 2, 3, etc.). To obtain a hardware-friendly compressed network,\(^1\) we set $b_1$ and $\gamma_j$ to 2, which ensures that all the decomposition bitwidths are power-of-two. Moreover, since the normalized quantization error change is small when the bitwidth is greater than 8 as indicated in Corollary 1 and Fig. 2, we only consider those bitwidths that are not greater than 8-bit (i.e., $b_j \in \{2, 4, 8\}$) in our paper. An empirical study on the non-power-of-two bitwidths can be found in Section V-E.

B. Single-Path Bit Sharing Model Compression

1) Binary Gate for Quantization: In a neural network, different layers have diverse redundancy and contribute differently to the accuracy and efficiency of the network. To determine the bitwidth for each layer, we introduce a layer-wise binary quantization gate $g_{b_j} \in \{0, 1\}$ (j > 1) on each of the re-assignment offsets in Eq. (8) to encode the choice of the quantization bitwidth $b_j$ as:

$$g_{b_j} = H(||z - z_{b_{j-1}}||_1 - \alpha_{b_j}),$$
$$z_{b_K} = z_{b_1} + g_{b_2}(r_{b_2} + \cdots + g_{b_{K-1}}(r_{b_{K-1}} + g_{b_K}r_{b_K})), \quad (10)$$

where $\alpha_{b_j}$ is a layer-wise threshold that controls the choice of bitwidth and $H(A)$ is the step function which returns 1 if $A \geq 0$ and returns 0 otherwise. Here, we use the quantization error to determine the choice of bitwidth. Specifically, if the quantization error is greater than $\alpha_{b_j}$, we activate the corresponding pruning gate to increase the bitwidth to reduce the residual error, and vice versa.

2) Binary Gate for Pruning: Note that in Eq. (10), we can also consider filter pruning as a special case of quantization, which makes it possible to perform network pruning and quantization jointly. To avoid the prohibitively large filter-wise search space, we propose to divide the filters into groups based on channel indices and consider the group-wise sparsity instead.

To be specific, we introduce a binary gate $g_{c,b_j}$ for each group to encode the choice of pruning as:

$$g_{c,b_j} = H(||w_{c}||_1 - \alpha_{b_j}),$$
$$z_{c,b_K} = g_{c,b_1}(z_{c,b_1} + g_{b_2}(r_{c,b_2} + \cdots + g_{b_{K-1}}(r_{c,b_{K-1}} + g_{b_K}r_{c,b_K}))), \quad (11)$$

where $z_{c,b_j}$ is an element of the $c$th group filters with $b_j$-bit quantization and $r_{c,b_j}$ is the corresponding re-assignment offset obtained by quantizing the residual error $z_{c} - z_{c,b_{j-1}}$. Here, $\alpha_{b_j}$ is a layer-wise threshold for filter pruning. Following PFEC [43], we use the $\ell_1$-norm criteria to evaluate the importance of different groups of filters. Specifically, if a group of filters is important, the corresponding pruning gate will be activated, and vice versa.

3) Normalization for Binary Gate: Note that the binary gates for quantization are layer-wise while those for pruning are group-wise, which may lead to different scales of thresholds. To mitigate the effect of different scaling, we perform normalization before applying the step function $H(\cdot)$. Specifically, given an evaluation metric $A(||z - z_{b_{j-1}}||_1$ for quantization and the $\ell_1$-norm of the $c$th group of filters $||w_{c}||_1$ for pruning), we obtain the normalized metric $\tilde{A}$ by $\tilde{A} = A/N_A$, where $N_A$ is the number of elements in $A$. We then feed $\tilde{A} - \alpha$ into the step function to obtain the output of the binary gate, where $\alpha$ is the corresponding pruning or quantization threshold.

C. Learning for Loss-Aware Compression

1) Gradient Approximation for the Step Function: Instead of manually determining the thresholds of pruning and quantization, we propose to learn them via a gradient descent method. Unfortunately, the step function in Eqs. (10) and (11) is non-differentiable. To address this, we propose to use straight-through estimator (STE) [1, 95] to approximate the gradient of $H(\cdot)$ by the gradient of the sigmoid function $S(\cdot)$, which can be formulated as

$$\frac{\partial g}{\partial \alpha} = \frac{\partial H(\tilde{A} - \alpha)}{\partial \alpha} \approx \frac{\partial S(\tilde{A} - \alpha)}{\partial \alpha}$$
$$= -S(\tilde{A} - \alpha)(1 - S(\tilde{A} - \alpha)), \quad (12)$$

where $g$ is the output of a binary gate.

2) Objective Function for SBS: Let $W$ be the model parameters and $\alpha$ be the compression configuration that are composed of pruning and quantization thresholds. To design a hardware-efficient model, the objective function should reflect both the accuracy and computation cost of a compressed model. Following [3], we incorporate the computation cost into the objective function and formulate the joint objective as:

$$\mathcal{L}(W, \alpha) = \mathcal{L}_{ce}(W, \alpha) + \lambda \log R(\alpha), \quad (13)$$

where $\mathcal{L}_{ce}(\cdot, \cdot)$ is the cross-entropy loss, $R(\cdot)$ is the computation cost of the network and $\lambda$ is a hyperparameter that adjusts the importance of the computation cost term $\log R(\alpha)$ in the loss function. In fact, a larger $\lambda$ indicates that we put more penalty on computation cost term and thus results in a compressed model with lower resource consumption. Following single-path
NAS [74], we use a similar formulation of computation cost to preserve the differentiability of the objective function.

For simplicity, we consider performing weight quantization only. Let $G_l^t$ be the number of filters groups for layer $l$ and $R_{c,b}^l$ be the computation cost of the $c$th group of filters with $b$-bit quantization for layer $l$. The computation cost $R(\cdot)$ is formulated as follows:

$$R(\alpha) = \sum_{l=1}^{L} \sum_{c=1}^{G_l^t} g_{b,c,b}^l (R_{c,b_1}^l + g_{b_2,b_2}^l (R_{c,b_2}^l - R_{c,b_1}^l) + \cdots + g_{b_K}^l (R_{c,b_K}^l - R_{c,b_{K-1}}^l)), \quad (14)$$

where $g_{b,c,b}^l$ and $g_{b,c,b_1}^l$ are the binary gates for $b$-bit quantization and the pruning decision for the $c$th group of filters, respectively. Similarly, the computation cost for activation quantization can be easily derived by replacing the binary gates of weights with those of activations.

Note that in Eq. (12), we approximate the gradient of the step function $H(\cdot)$ by the gradient of the sigmoid function $S(\cdot)$. Therefore, the objective function in Eq. (13) remains differentiable. By minimizing the objective using gradient descent, the configurations of each layer are automatically determined. Moreover, we are able to make a trade-off between pruning and quantization. However, the gradient approximation of the binary gate may inadvertently introduce noisy signals, which is more severe when we quantize both weights and activations.

To alleviate the noisy signals in the gradient approximation of the binary gate, we propose to train the binary gates of weights and activations in an alternating manner. Let $\alpha_w$ and $\alpha_x$ be the thresholds for weights and activations quantization, respectively. The training algorithm of the proposed SBS is shown in Algorithm 1. Starting from a pre-trained model $M^0$, we first initialize $\alpha_w$ and $\alpha_x$ to 0. Then, we train the compressed model $M$ for $T$ epochs. For each epoch, we train the model parameters $W$ with alternating updates of $\alpha_w$ and $\alpha_x$. An empirical study over the effect of alternating training scheme is put in Section V-C. Once the training is finished, we are able to obtain a compressed model by removing unactivated re-assignment offsets and filters. Following the common practice in model compression [14], [16], [49], [80], [86], we then fine-tune the resulting compressed model to compensate for the accuracy loss of model compression. During fine-tuning, we use the quantization method mentioned in Section II with the searched bitwidths rather than the quantization decomposition. Therefore, the inference cost of our SBS is the same as the traditional quantization methods.

Algorithm 1: Training Method for SBS.

**Input:** A pre-trained model $M^0$, the number of candidate bitwidths $K$, a sequence of candidate bitwidths $\{b_j\}_{j=1}^K$, the number of training epochs $T$, the number of training iteration $I$, and hyperparameters $\lambda$.

**Output:** A compressed model $M$.

1: Initialize $M$ using $M^0$.
2: Initialize the weights quantization thresholds $\alpha_w$ and activations quantization thresholds $\alpha_x$ to 0.
3: for epoch $t \in \{1, \ldots, T\}$ do
4: for iteration $i \in \{1, \ldots, I\}$ do
5: Calculate the binary gates for quantization and pruning using Eqs. (10) and (11).
6: if $i = 2n$, $n \in \mathbb{N}^+$ then
7: Update $W$ and $\alpha_w$ by minimizing Eq. (13).
8: else
9: Update $W$ and $\alpha_x$ by minimizing Eq. (13).
10: end if
11: end for
12: end for

weights. Formally, we have the following results on the quantization errors incurred by the two schemes.

**Proposition 1.** Consider a linear regression problem $\min_{w \in \mathbb{R}^d} \mathbb{E}_{(x,y) \sim D} [(y - wx)^2]$ with data pairs $\{(x,y)\}$, where $x \in \mathbb{R}^d$ is sampled from $\mathcal{N}(0, \sigma^2 I)$ and $y \in \mathbb{R}$ is its response. Consider using SBS and DNAS to quantize the linear regression weights. Let $w^t_L$ and $w^t_D$ be the quantized regression weights of SBS and DNAS at the iteration $t$ of the optimization, respectively. Then the following equivalence holds during the optimization process:

$$\lim_{t \to \infty} \mathbb{E}_{(x,y) \sim D} [(y - w^t_L x)^2] = \lim_{t \to \infty} \mathbb{E}_{(x,y) \sim D} [(y - w^t_D x)^2],$$

where $w^t_L = w^t_{L,b_1} + g_{b_2}^t (r_{b_2}^t + g_{b_3}^t (r_{b_3}^t + \cdots + g_{b_K}^t r_{b_K}^t))$,

$$r_{b_j}^t = D(w^t_{L,b_j} - w^t_{L,b_{j-1}} - b_{j-1}), j = 2, \ldots, K,$

$$w^t_D = \sum_{i=1}^{K} p_i^t w_{b_i}^t, \sum_{i=1}^{K} p_i^t = 1. \quad (15)$$

From Proposition 1, the multi-path scheme (DNAS) converges to our single-path scheme during the optimization. However, our single-path scheme contains fewer parameters and consumes less computational overhead. For example, the multi-path scheme maintains $K$ paths while our proposed single path scheme maintains only single-path. Thus, the parameters and computation cost reduction is $(K-1)/K$.

To further study the difference between our single-path scheme and multi-path scheme, we consider a linear regression scenario $\min_{w \in \mathbb{R}^d} \mathbb{E}_{(x,y) \sim D} [(y - wx)^2]$ and compare the quantization errors incurred by the two quantization schemes over the regression weights. Specifically, we construct a toy linear regression dataset $\{(x_i, y_i)\}_{i=1}^N$ where we randomly sample $N = 10,000$ data $x \in \mathbb{R}^d$ from Gaussian distribution $\mathcal{N}(0, I)$.
weights to initialize the compressed models following [16], [80]. For CIFAR-100, we use the same data augmentation as in [30], including randomly cropping and horizontal flipping. For ImageNet, images are resized to $256 \times 256$, and then a $224 \times 224$ patch is randomly cropped from an image or its horizontal flip for training. For testing, a $224 \times 224$ center cropped is chosen.

Following [44], we introduce weight normalization during training. We use SGD with nesterov [60] for optimization. The momentum term is set to 0.9. We first search configurations for 30 epochs on CIFAR-100 and 10 epochs on ImageNet. The learning rate is set to 0.001. We then fine-tune the searched compressed network to recover the performance drop. On CIFAR-100, we fine-tune the searched network for 200 epochs with a mini-batch size of 128. The learning rate is initialized to 0.1 and is divided by 10 at 80th and 120th epochs. For ResNet-18 on ImageNet, we fine-tune the searched network for 15 epochs with a mini-batch size of 256. For MobileNetV2 on ImageNet, we fine-tune for 150 epochs. For all models on ImageNet, the learning rate starts at 0.01 and decays with cosine annealing [54].

C. Compared Methods

To investigate the effectiveness of SBS, we consider the following methods for comparisons: SBS: our proposed method with joint pruning and quantization; SBS-Q: SBS with quantization only; SBS-P: SBS with pruning only; and several state-of-the-art model compression methods: DNAS [86]: it uses multi-path search scheme to search for optimal bitwidths; HAQ [80]: it uses reinforcement learning to automatically determine the quantization policies; HAWQ [16]: it uses second-order Hessian information to guide the bitwidth search of each layer; DQ [78]: it uses a gradient-based method to learn the bitwidths; DJPQ [92]: it combines the variational information bottleneck method to structured pruning and mixed-bit precision quantization and learns optimal configurations using a gradient-based method; Bayesian Bits [79]: the authors cast configuration search problem of pruning and quantization into a variational inference problem and use gradient-based method to learn configurations. We do not consider other methods since they have been compared in the methods mentioned above. Except DQ, all the compared methods use the same strategy that first conducts configurations search based on the same pre-trained model and then finetunes the resulting model with the searched configurations.

D. Comparisons on CIFAR-100

We apply SBS to compress ResNet-20 and ResNet-56 on CIFAR-100. We report the results under BOPs and memory footprints constraints in Tables I and II. Compared with the full-precision counterparts, 4-bit quantized models achieve comparable performance or even better performance. This can be attributed to the redundancy removal and regularization effect of network quantization. Similar phenomenon is also observed in LSQL [17]. Compared with fixed-precision models, mixed-precision methods are able to further reduce the BOPs while preserving performance. Critically, SBS-Q outperforms state-of-the-arts DQ, HAQ, and DNAS with less computation cost.

Fig. 3. Loss comparison between the single-path (Ours) and multi-path [86] schemes on a toy dataset with details discussed in Section III-D.

and obtain corresponding responses $y = \mathbf{w}^\top \mathbf{x} + \Delta$. Here, $\mathbf{w}^* \in \mathbb{R}^{10}$ is a fixed weight randomly sampled from $[0, 1]^{10}$ and $\Delta \in \mathbb{R}$ is a noise sampled from $\mathcal{N}(0, 1)$. We then apply SBS and DNAS to quantize the regression weights. As shown in Fig. 3, our method converges faster and smoother than the multi-path scheme.

IV. EXPERIMENTS

A. Datasets and Evaluation Metrics

We evaluate the proposed SBS on two image classification datasets, including CIFAR-100 [40] and ImageNet [12]. CIFAR-100 consists of 50 k training samples and 10 k testing images with 100 classes. ImageNet contains 1.28 million training samples and 50 k testing images for 1,000 classes.

We measure the performance of different methods using the Top-1 and Top-5 accuracy. Experiments on CIFAR-100 are repeated 5 times and we report the mean and standard deviation. For fair comparisons, we measure the computation cost by the Bit-Operation (BOP) count for all the compared methods following [26], [92]. The BOP compression ratio is defined as the ratio between the total BOPs of the uncompressed and compressed model. We can also measure the computation cost by the total weights and activations memory footprints following DQ [78]. Similarly, the memory footprints compression ratio is defined as the ratio between the total memory footprints of the uncompressed and compressed model. Moreover, following [48], [74], we use the search cost on a GPU device (NVIDIA TITAN Xp) to measure the time of finding an optimal compressed model.

B. Implementation Details

To demonstrate the effectiveness of the proposed method, we apply SBS to various architectures, such as ResNet [30] and MobileNetV2 [70]. All implementations are based on PyTorch [61]. Following HAQ [80], we quantize all the layers, in which the first and the last layers are quantized to 8-bit. Following ThiNet [59], we only perform filter pruning for the first layer in the residual block. For ResNet-20 and ResNet-56 on CIFAR-100 [40], we set $B$ to 4. For ResNet-18 and MobileNetV2 on ImageNet [69], $B$ is set to 16 and 8, respectively. We tune $\lambda$ to obtain compressed models under different resource constraints. We first train the full-precision models and then use the pre-trained weights to initialize the compressed models following [16], [80].
TABLE I
COMPARISONS OF DIFFERENT METHODS W.R.T. BIT-OPERATION (BOP) COUNT ON CIFAR-100. “BOP COMP. RATIO” DENOTES THE BOP COMPRESSION RATIO.

| Network | Method       | BOPs (M) | BOP comp. ratio | Search Cost (GPU hours) | Top-1 Acc. (%) | Top-5 Acc. (%) |
|---------|--------------|----------|-----------------|-------------------------|----------------|----------------|
| ResNet-20 | Full-precision | 41796.6  | 1.0             | –                       | 67.5           | 90.8           |
|         | 4-bit precision | 674.6    | 62.0            | –                       | 67.1 ± 0.3     | 90.4 ± 0.2     |
|         | DQ [78]       | 1180.0   | 35.4            | 2.2                     | 67.7 ± 0.4     | 90.4 ± 0.5     |
|         | HAQ [80]      | 653.4    | 64.0            | 5.8                     | 67.5 ± 0.1     | 90.4 ± 0.3     |
|         | DNA [66]      | 660.0    | 62.9            | 2.8                     | 67.8 ± 0.3     | 90.4 ± 0.2     |
|         | SBS-P (Ours)  | 6856.6   | 1.5             | 6.2                     | 67.3 ± 0.1     | 90.7 ± 0.2     |
|         | SBS-Q (Ours)  | 649.5    | 64.4            | 0.8                     | 68.1 ± 0.1     | 90.5 ± 0.0     |
|         | SBS (Ours)    | 630.6    | 66.3            | 1.0                     | 68.1 ± 0.3     | 90.6 ± 0.2     |

ResNet-56

| Network | Method       | Memory footprints (KB) | M.f. comp. ratio | Top-1 Acc. (%) | Top-5 Acc. (%) |
|---------|--------------|------------------------|-----------------|----------------|----------------|
|         | Full-precision | 5653.4    | 1.0             | 71.7           | 92.2           |
|         | 4-bit precision | 711.7     | 7.9             | 70.9 ± 0.3     | 91.2 ± 0.4     |
|         | DQ [78]       | 725.2     | 7.8             | 70.9 ± 0.4     | 91.7 ± 0.3     |
|         | HAQ [80]      | 700.0     | 8.1             | 71.3 ± 0.1     | 91.1 ± 0.1     |
|         | DNA [66]      | 708.9     | 8.0             | 71.5 ± 0.2     | 91.3 ± 0.1     |
|         | SBS-P (Ours)  | 6776.6    | 1.2             | 71.5 ± 0.1     | 91.6 ± 0.2     |
|         | SBS-Q (Ours)  | 674.5     | 8.4             | 71.5 ± 0.2     | 91.6 ± 0.2     |
|         | SBS (Ours)    | 657.3     | 8.6             | 71.6 ± 0.1     | 91.8 ± 0.4     |

TABLE II
COMPARISONS OF DIFFERENT METHODS W.R.T. MEMORY FOOTPRINTS ON CIFAR-100. “M.F. COMP. RATIO” DENOTES THE MEMORY FOOTPRINTS COMPRESSION RATIO.

![Fig. 4. Performance comparisons of different methods with different BOPs and memory footprints. We use different methods to compress ResNet-56 and report the results on CIFAR-100.](image)

Fig. 4. Performance comparisons of different methods with different BOPs and memory footprints. We use different methods to compress ResNet-56 and report the results on CIFAR-100.

and memory footprints. For example, SBS-Q ResNet-56 outperforms DNA by 0.3% on the Top-1 accuracy while achieving $65.3 \times$ BOPs reduction. By performing pruning and quantization jointly, SBS achieves the best performance while further reducing computation cost and memory footprints of the compressed models.

We also show the results of the compressed ResNet-56 with different BOPs and memory footprints in Fig. 4. From the results, SBS consistently outperforms all the other methods under variant BOPs and memory footprints settings. Moreover, our proposed SBS achieves significant improvement in terms of BOPs and memory footprints compared with fixed-precision quantization, especially at low BOPs and memory footprints settings. For example, compared with the fixed-precision counterpart, SBS compressed ResNet-56 consumes much less computational overhead (783.52 v.s. 1156.46 BOPs) and fewer memory footprints (395.25 v.s. 536.24 KB) but achieves comparable performance.

To evaluate the efficiency of the proposed method, we also compare the search cost of different methods. From Table I, the search cost of the proposed SBS is much smaller than the state-of-the-art methods. For example, for ResNet-20, the search cost of SBS is $2.8 \times$ lower than DNA while the search cost of SBS for ResNet-56 is nearly $5.1 \times$ lower than DNA. Compared with SBS-Q, SBS only introduces a small amount of overhead (e.g., 0.2 GPU hour for ResNet-56). These results show the superior efficiency of SBS-Q and SBS.

E. Comparisons on ImageNet

To evaluate the effectiveness of our method, we first apply SBS and SBS-Q to compress ResNet-18 and ResNet-50 and evaluate the performance on ImageNet. From Table III, SBS-Q compressed models with lower BOPs achieve better performance than the fixed-precision counterparts. For example, for
TABLE III
COMPARISONS BETWEEN OUR PROPOSED METHOD AND FIXED-PRECISION QUANTIZATION ON IMAGENET

| Network  | Method     | Top-1 Acc. (%) | Top-5 Acc. (%) | BOPs (G) |
|----------|------------|----------------|----------------|----------|
| ResNet-18 | Full-precision | 69.8 | 89.1 | 1857.6 |
|          | 8-bit precision | 70.0 | 89.3 | 116.1 |
| SBS-Q (Ours) |          | 70.2 | 89.4 | 114.3 |
| SBS (Ours)    |          | 70.3 | 89.4 | 109.4 |
| ResNet-50     | Full-precision | 76.8 | 93.3 | 4187.3 |
|              | 8-bit precision | 76.3 | 93.0 | 261.7 |
| SBS-Q (Ours)  |          | 76.5 | 93.1 | 253.1 |
| SBS (Ours)    |          | 75.7 | 92.7 | 215.7 |
| SBS-Q (Ours)  |          | 75.8 | 92.8 | 71.0  |
| SBS (Ours)    |          | 75.9 | 92.8 | 70.4  |

Fig. 5. Performance comparisons of various methods for ResNet-18 and ResNet-50 under different BOPs on ImageNet.

ResNet-18, SBS-Q with 1.8 G fewer BOPs outperforms 8-bit quantization by 0.2% on the Top-1 accuracy. These results justify the effectiveness and necessity of bitwidth search. By combining pruning and quantization, SBS yields compressed models with higher accuracy and lower BOPs. For example, for ResNet-50, SBS outperforms SBS-Q by 0.1% on the Top-1 accuracy while reducing 9.2 G BOPs. We also show the results of compressed ResNet-18 and ResNet-50 with different BOPs in Fig. 5. Compared with SBS-Q, SBS achieves better accuracy-BOPs trade-off, which shows the benefit of performing pruning and quantization jointly.

To compare SBS with other state-of-the-art methods, we apply different methods to compress ResNet-18 and MobileNetV2. From Table IV, SBS-Q with less computation cost outperforms the state-of-the-art baselines. Specifically, SBS-Q compressed MobileNetV2 surpasses the one compressed by HAQ with more BOPs reduction. By combining pruning and quantization, SBS further improves the performance while reducing the computation cost of the compressed models. For example, SBS compressed MobileNetV2 reduces the BOPs by 22.6× while only resulting in 0.1% performance degradation in terms of the Top-1 accuracy.

We also illustrate the detailed configurations (i.e., bitwidth and pruning rate) of each layer from the compressed ResNet-18 and ResNet-50. From Figs. 6 and 7, SBS assigns higher bitwidth to $1 \times 1$ convolutional layers (including downsampling layers). One possible reason is that the number of parameters and computation cost of $1 \times 1$ convolutional layers are much smaller than other layers. Compressing these layers may lead to a significant performance decline. Besides, SBS allocates fewer bitwidth to $3 \times 3$ convolutional layers to reduce BOPs. Note that the bit-width allocation might vary among weights, activations, and various models due to distinct tensor dimensions or network
building blocks (non-bottleneck for ResNet-18 and bottleneck for ResNet-50), which affects layer complexity and compression trade-offs. For filter pruning, SBS inclines to prune more filters in the shallower layers of ResNet-18 and middle layers of ResNet-50, which significantly reduces the number of parameters and computational overhead. More detailed configurations of the other models are put in the supplementary material, available online.

F. Hardware Resource-Constrained Compression

To investigate the effect of SBS on hardware devices, we further apply SBS to compress MobileNetV2 under various resource constraints on the BitFusion architecture [71], which is a state-of-the-art spatial ASIC accelerator for neural networks. We measure the computation cost by the latency and energy on a simulator of the BitFusion with a batch size of 16. We report the results on ImageNet in Table VI. Compared with fixed-precision quantization, SBS achieves better performance with lower latency and energy. For example, SBS compressed MobileNetV2 with 3.3 ms lower latency outperforms 8-bit MobileNetV2 by 0.5% on the Top-1 accuracy. These results show the promising hardware efficiency of our SBS.

V. FURTHER STUDIES

In this section, we conduct further studies for our SBS. 1) We investigate the effect of the bit sharing scheme in Section V-A. 2) We explore the effect of the one-stage compression scheme in Section V-B. 3) We study the effect of the alternating training scheme in Section V-C. 4) We explore the effect of different group sizes in Section V-D. 5) We investigate the effect of SBS with non-power-of-two bitwidths in Section V-E. 6) We compare the training from scratch scheme with the fine-tuning strategy in Section V-F.
TABLE VII

| Network   | Method   | Top-1 Acc. (%) | Top-5 Acc. (%) | BOPs (M) |
|-----------|----------|----------------|----------------|----------|
| SBS - P   | SBS - Q  | 70.7±0.2       | 91.3±0.2       | 1092.4   |
| ResNet-56 | SBS-Q → SBS-P | 70.9±0.3       | 91.4±0.3       | 1067.1   |
| SBS       |          | 70.9±0.3       | 91.4±0.2       | 1064.1   |

TABLE VIII

| Network   | Method               | Top-1 Acc. (%) | Top-5 Acc. (%) | BOPs (M) |
|-----------|----------------------|----------------|----------------|----------|
| ResNet-56 | Joint training       | 71.3±0.2       | 91.6±0.3       | 1942.4   |
|           | Alternating training | 71.3±0.1       | 91.5±0.4       | 1918.8   |

TABLE IX

| Network   | B | Top-1 Acc. (%) | Top-5 Acc. (%) | BOPs (M) |
|-----------|---|----------------|----------------|----------|
| ResNet-20 | 1 | 67.7±0.2       | 90.5±0.1       | 648.3    |
|           | 2 | 68.0±0.1       | 90.4±0.1       | 631.2    |
|           | 4 | 68.1±0.3       | 90.6±0.2       | 630.6    |
|           | 8 | 67.9±0.1       | 90.4±0.1       | 631.1    |
| ResNet-56 | 1 | 71.3±0.1       | 91.6±0.2       | 1945.3   |
|           | 2 | 71.3±0.1       | 91.6±0.1       | 1949.5   |
|           | 4 | 71.6±0.1       | 91.8±0.4       | 1915.6   |
|           | 6 | 71.5±0.2       | 91.4±0.2       | 1956.5   

A. Effect of the Bit Sharing Scheme

To investigate the effect of the bit sharing scheme, we apply SBS to quantize ResNet-20 and ResNet-56 with and without the bit sharing scheme and report the results on CIFAR-100. Here, SBS without the bit sharing denotes that we compress the models with the multi-path scheme [86]. We report the Top-1/Top-5 accuracy and BOPs in Table V. We also present the search cost and GPU memory footprints measured on a GPU device (NVIDIA TITAN Xp). From the results, SBS with the bit sharing scheme obtains more compact models and consistently outperforms the ones without the bit sharing scheme while significantly reducing the search cost and GPU memory footprints. For example, ResNet-56 with the bit sharing scheme outperforms the counterpart by 0.2% in the Top-1 accuracy and achieves 3.6× reduction on GPU memory and 6.7× acceleration during training.

B. Effect of the One-Stage Compression

To investigate the effect of the one-stage compression scheme, we perform model compression with both the one-stage and the two-stage compression schemes on ResNet-56. Specifically, the one-stage compression scheme means that we perform filter pruning and quantization jointly. The two-stage compression scheme is that we perform filter pruning and network quantization in a separate stage. For convenience, we denote the two-stage compression scheme as A → B, where A → B denotes that we first perform A and then conduct B. We consider both SBS-Q → SBS-P and SBS-P → SBS-Q for comparisons. For the two-stage compression method, we have conducted extensive experiments to find a trade-off between pruning and quantization that leads to high accuracy in the final compressed model following [92]. From Table VII, the resulting model obtained by the one-stage scheme with less computation cost outperforms the two-stage counterparts. For example, SBS with 1064.1 M BOPs outperforms SBS-P → SBS-Q by 0.2% on the Top-1 accuracy. These results show the superiority of performing pruning and quantization jointly.

C. Effect of the Alternating Training Scheme

To investigate the effect of the alternating training scheme introduced in Algorithm 1, we apply SBS to compress ResNet-56 with a joint training scheme and an alternating training scheme on CIFAR-100. Here, the joint training scheme denotes that we train the binary gates of weights and activations jointly. The alternating training scheme indicates that we train the binary gates of weights and activations in an alternating way, as mentioned in Section III-C2. From the results of Table VIII, the model trained with the alternating scheme achieves better performance than those of the joint scheme while consuming lower computational overhead, which demonstrates the effectiveness of the proposed alternating training scheme.

D. Effect of Different Group Sizes

To investigate the effect of different group sizes B, we apply SBS to compress ResNet-20 and ResNet-56 with different B ∈ {1, 2, 4, 8} and show the results in Table IX. From the table, the performance of the compressed models first improves and then degrades with the increase of B. For example, the compressed ResNet-56 with B = 4 outperforms those of B = 1 by 0.3% in terms of the Top-1 accuracy. In fact, a large B leads to a small search space. With limited computing resources, we are able to find good configurations with high probability and thus improve performance. In contrast, a too large B results in extremely small search space, which may ignore many good compression configurations and thus limits the performance. Since our SBS achieves the best performance with B = 4, we use it by default for the experiments on CIFAR-100.

E. Effect of SBS With Non-Power-of-Two Bitwidths

To investigate the effect of our SBS on non-power-of-two bitwidths, we apply SBS to compress ResNet-20/56 on CIFAR-100 and ResNet-18/50 on ImageNet with the non-power-of-two bitwidths. As shown in Theorem 1, if we set b_1 to 3 and γ_j to 2, we have the quantization decomposition for an example of the non-power-of-two bitwidths (i.e., b_j ∈ {3, 6, 12, 24, · · ·}). Similar to the power-of-two bitwidths, we only consider those bitwidths that are not greater than 12-bit since the normalized quantization error change is small when the bitwidth is greater than 12 according to Corollary 1. We denote SBS with the candidate bitwidths of {3, 6, 12} as SBS* for convenience. From Tables X and XI, SBS* obtains fewer BOPs while achieving better performance than the fixed-precision counterparts. For
example, for ResNet-50, SBS∗ outperforms the 6-bit counterpart by 0.2% on the Top-1 accuracy. These results show the effectiveness of SBS with the non-power-of-two bitwidths. Moreover, the non-power-of-two bitwidths are not hardware-friendly due to the bit wasting in values packing (See Appendix D, available online), which seriously degrades the hardware utilization. Therefore, we only consider the power-of-two bitwidths in our experiments.

F. Training From Scratch v.s. fine-Tuning Scheme

To investigate the effect of SBS with training from scratch scheme, we apply SBS to compress ResNet-20 and ResNet-56 on CIFAR-100. The experimental settings are the same as Section IV-B except that we do not use any pre-trained model. From Table XII, the models obtained by our SBS surpass the full-precision counterparts. Moreover, the performance improvement brought from SBS is smaller than those that are from fine-tuning. For example, for ResNet-56, the improvement of SBS over the fixed-precision quantization is 0.3% while the fine-tuning counterpart is 0.7% in Table I. As the model are not well-trained, the searched configurations may not be accurate, which hinders the effect of our method.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed an automatic loss-aware model compression method called Single-path Bit Sharing (SBS) for pruning and quantization jointly. The proposed SBS introduces a novel single-path bit sharing model to encode all bitwidths in the search space, where a quantized representation will be decomposed into the sum of the lowest bitwidth representation and a series of re-assignment offsets. Based on this, we have further introduced learnable binary gates to encode the choice of different compression configurations. By jointly training the binary gates and network parameters, we are able to make a trade-off between pruning and quantization and automatically learn the configuration of each layer. Experiments on CIFAR-100 and ImageNet have shown that SBS is able to achieve significant computation cost and memory footprints reduction while preserving performance. In the future, we plan to combine our proposed SBS with other methods, e.g., neural architecture search, to find a more compact model with better performance.

REFERENCES

[1] Y. Bengio, N. Léonard, and A. Courville, “Estimating or propagating gradients through stochastic neurons for conditional computation,” 2013, arXiv:1308.3412.
[2] 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. 2978–2985.
[3] H. Cai, L. Zhu, and S. Han, “ProxylessNAS: Direct neural architecture search on target task and hardware,” in Proc. Int. Conf. Learn. Representations., 2019, pp. 1–13.
[4] Z. Cai, X. He, J. Sun, and N. Vasconcelos, “Deep learning with low precision by half-wave Gaussian quantization,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 5406–5414.
[5] Z. Cai and N. Vasconcelos, “Rethinking differentiable search for mixed-precision neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 2349–2358.
[6] J. Cao, Y. Guo, Q. Wu, C. Shen, J. Huang, and M. Tan, “Improving generative adversarial networks with local coordinate coding,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 1, pp. 211–227, Jan. 2022.
[7] P. Chen, J. Liu, B. Zhuang, M. Tan, and C. Shen, “AQD: Towards accurate quantized object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 104–113.
[8] S. Chen and Q. Zhao, “Shallowing deep networks: Layer-wise pruning based on feature representations,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 12, pp. 3048–3056, Dec. 2019.
[9] W. Chen, P. Wang, and J. Cheng, “Towards mixed-precision quantization of neural networks via constrained optimization,” in Proc. IEEE Int. Conf. Comp. Vis., 2021, pp. 5350–5359.
[10] Y. Chen, T. Guan, and C. Wang, “Approximate nearest neighbor search by residual vector quantization,” Sensors, vol. 10, pp. 11259–11273, 2010.
[11] J. Choi et al., “PACT: Parameterized clipping activation for quantized neural networks,” 2018, arXiv:1805.06085.
[12] 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.
[13] R. Ding, T.-W. Chin, Z. Liu, and D. Marculescu, “Regularizing activation distribution for training binarized deep networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 11408–11417.
[14] X. Dong and Y. Yang, “Network pruning via transformable architecture search,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2019, pp. 759–770.
[15] Z. Dong, Z. Yao, D. Arfeen, A. Gholami, M. W. Mahoney, and K. Keutzer, “HAWQ-V2: Hessian aware trace-weighted quantization of neural networks,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2020, pp. 18518–18529.
[16] Zong, Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, “HAWQ: Hessian aware quantization of neural networks with mixed-precision,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 293–302.
[17] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, “Learned step size quantization,” in Proc. Int. Conf. Learn. Representations, 2020, pp. 1–12.
[18] S. Gao, F. Huang, W. Cai, and H. Huang, “Network pruning via performance maximization,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 9270–9280.
[19] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, “Res2Net: A new multi-scale backbone architecture,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 2, pp. 652–662, Feb. 2021.
[20] Y. Gong, L. Liu, M. Yang, and L. Bourdev, “Compressing deep convolutional networks using vector quantization,” 2014, arXiv:1412.6113.
[21] I. Goodfellow et al., “Generative adversarial nets,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2014, pp. 2672–2680.
[22] S. Guo, Y. Wang, Q. Li, and J. Yan, “DMCP: Differentiable Markov channel pruning for neural networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 1536–1544.
[23] Y. Guo, Q. Chen, J. Chen, Q. Wu, Q. Shi, and M. Tan, “Auto-embedding generative adversarial networks for high resolution image synthesis,” IEEE Trans. Multimedia, 2019, pp. 2726–2737.
[24] Y. Guo, A. Yao, and Y. Chen, “Dynamic network surgery for efficient DNNs,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2016, pp. 1379–1387.
[25] Y. Guo et al., “NAT: Neural architecture transformer for accurate and compact architectures,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2019, pp. 755–767.
[26] Z. Guo et al., “Single path one-shot neural architecture search with uniform sampling,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 544–560.
[27] S. Han, H. Miao, and W. J. Dally, “Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding,” in Proc. Int. Conf. Learn. Representations, 2016, pp. 1–14.
[28] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 1135–1143.
[29] T. Han, D. Li, J. Liu, L. Tian, and Y. Shan, “Improving low-precision network quantization via bin regularization,” in Proc. IEEE Int. Conf. Comput. Vis., 2021, pp. 5261–5270.
[30] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.
[31] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, “AMC: AutoML for model compression and acceleration on mobile devices,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 815–832.
[32] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, “Filter pruning via geometric median for efficient convolutional networks acceleration,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 4340–4349.
[33] Y. He, X. Zhang, and J. Sun, “Channel pruning for accelerating very deep neural networks,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 1398–1406.
[34] P. Hu, X. Peng, H. Zhu, M. S. S. Aly, and J. Lin, “OPQ: Compressing deep neural networks with one-shot pruning-quantization,” in Proc. AAAI Conf. Artif. Intell., 2021, pp. 7780–7788.
[35] Z. Huang and N. Wang, “Data-driven sparse structure selection for deep neural networks,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 317–334.
[36] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 4107–4115.
[37] Q. Jin, L. Yang, and Z. Liao, “Towards efficient training for neural network quantization,” 2019, arXiv:1912.10207.
[38] S. Jung et al., “Learning to quantize deep networks by optimizing quantization intervals with task loss,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 4250–4259.
[39] S. Kim, A. Gholami, Z. Yao, M. W. Mahoney, and K. Keutzer, “I-BERT: Integer-only bert quantization,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2021, pp. 5506–5518.
[40] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Tech. Rep. 0, University of Toronto, Toronto, Ontario, 2009.
[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2012, pp. 1106–1114.
[42] Y. LeCun et al., “Backpropagation applied to handwritten zip code recognition,” Neural Comput., vol. 1, pp. 541–551, 1989.
Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Jianfei Cai (Fellow, IEEE) received the PhD degree from the University of Missouri-Columbia. He is currently a professor and serves as the Head of the Data Science and AI Department at Faculty of IT, Monash University, Australia. Before that, he had served as Head of Visual and Interactive Computing Division and Head of Computer Communications Division in Nanyang Technological University (NTU). His major research interests include computer vision, multimedia and visual computing. He has successfully trained 30+ PhD students with three getting NTU SCSE Outstanding PhD thesis award. He is a co-recipient of paper awards in ACCV, ICCM, IEEE ICIP and MMSP. He serves or has served as an associate editor for International Journal of Computer Vision, IEEE Transactions on Image Processing, IEEE Transactions on Multimedia, and IEEE Transactions on Circuits and Systems for Video Technology as well as serving as Area Chair for CVPR, ICCV, ECCV, IJCAI, ACM Multimedia, ICME and ICIP. He was the Chair of IEEE CAS VSPC-TC during 2016–2018. He has also served as the leading TPC Chair for IEEE ICME 2012 will be the leading general chair for ACM Multimedia 2024.

Mingkui Tan (Member, IEEE) received the bachelor degree in environmental science and engineering, in 2006 and master degree in control science and engineering, in 2009, both from Hunan University, in Changsha, China, and the PhD degree in computer science from Nanyang Technological University, Singapore, in 2014. He is currently a professor with the School of Software Engineering at South China University of Technology. From 2014–2016, he worked as a Senior Research Associate on computer vision in the School of Computer Science, University of Adelaide, Australia. His research interests include machine learning, sparse analysis, deep learning and large-scale optimization.