Towards Automatic Model Compression via A Unified Two-Stage Framework

Weihan Chen[a,b], Peisong Wang[a], Jian Cheng[a,*]

[a]*National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China*
[b]*School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, 100049, China*

Abstract

Deep Neural Networks have become ubiquitous in various domains. Meanwhile, the problems of massive storage and computation costs have hindered the deployment of these models to real-world applications. This paper proposes a novel and unified two-stage framework for automatic model compression. To determine the compression ratio of each layer, we improve the optimization from two aspects. First, to predict the performance of each compression policy, we propose Dynamic BN, which improves the correlation significantly with little computation overhead. Second, to search for the compression ratio allocation, we propose an efficient and hyperparameter-free solving algorithm based on the proposed Hessian matrix approximation and Knapsack problem reformulation. Moreover, comprehensive experiments and analyses are conducted on the CIFAR-100&ImageNet datasets and various network architectures to demonstrate its performance advantages over

*Corresponding author at NLPR, Institute of Automation, Chinese Academy of Sciences, No. 95, Zhongguancun East Road, Beijing 100190, China

Email addresses: chenweihan2018@ia.ac.cn (Weihan Chen), peisong.wang@nlpr.ia.ac.cn (Peisong Wang), jcheng@nlpr.ia.ac.cn (Jian Cheng)

existing model compression methods under the quantization-only, pruning-only, and pruning-quantization settings.

Keywords: Deep Neural Networks, Model Compression, Quantization, Pruning.

1. Introduction

Nowadays, Deep Neural Networks (DNNs) have been leading record-breaking results in a wide range of visual applications, including image classification [1], object detection [2], etc. However, such performance boosts are mostly built on deep and wide network architectures with many parameters. Consequently, energy consumption is dominated by DRAM access as large networks do not fit in on-chip storage and require a lot of memory bandwidth to fetch the weights [3]. Under this circumstance, various model compression techniques have been proposed, including pruning [4; 5], quantization [6], neural architecture search [7], etc., to reduce the storage and memory bandwidth required to run an inference with such large networks so they can be deployed on mobile and embedded systems.

Briefly speaking, the ultimate goal of model compression is to maximize the network performance while satisfying the given model size constraint. In general, we can formulate it as the following optimization problem:

$$(c^*, W^*) = \underset{c \in \mathbb{C}, W}{\arg\min} \quad \mathcal{L}(\mathcal{P}_c(W)), \tag{1}$$

where c is a method-specific parameter that determines the compression ratio of each layer (e.g., sparsity ratio or quantization bit-width of each layer); $\mathbb{C}$ denotes all the compression ratio allocations that satisfy the given model size constraint; $\mathcal{P}_c(\cdot)$ means the function that projects the original weight to its compressed version; $\mathcal{L}(\cdot)$ represents the training loss. Compared with the standard DNNs training, model

compression introduces the extra constrained and discrete parameter c, which is the root of its intrinsic difficulty.

Current mainstream methods [8; 9; 10] adopt a two-stage pipeline that decouples the original problem (1) as the following two sub-ones:

First, obtain the optimal compression ratio allocation c with the given pre-trained model weight W,

$$c^* = \arg\max_{c \in \mathbb{C}} \quad Score(c, W). \tag{2}$$

Second, optimize the model weight W with the given compression ratio of each layer c,

$$W^* = \arg\min_{W} \quad \mathcal{L}(\mathcal{P}_c(W)). \tag{3}$$

It is noted that the problem (3) is the standard DNNs training essentially. Thus, the problem (2), which means determining the compression ratio of each layer, gradually becomes the focus of model compression. Conventionally, this sub-problem is solved based on hand-crafted heuristics and domain expertise. As the design space is so huge that human heuristic is usually sub-optimal and the manual setting is time-consuming, recent works propose kinds of methods to determine the compression ratio allocation automatically.

Considering the formulation of problem (2), there are two key questions we need to answer to obtain the optimal compression policy automatically. First, what is the specific form of the function $Score(\cdot, \cdot)$ so that we can predict the performance of each compression ratio allocation efficiently and effectively? Second, given the performance prediction function, how can we automatically search for the optimal compression policy c^*?

For the first question, to obtain the real performance of one compression ratio allocation, we need to fine-tune the network with the allocation until convergence.

Although effective, this strategy is very time-consuming and may take several days for a single compression policy on the large-scale dataset. Instead, an efficient and widely-used delegate of the real predictor is the performance of the network that all layers are compressed directly without any fine-tuning [10; 8]. As our experimental results show, the relevance between the real performance and the above delegate is very low for various compression techniques and model size constraints. To fix this issue, we notice a huge gap exists between the distribution of intermediate layers' feature maps for the pre-trained and compressed weight. We suspect that the out-of-date mean and variance statistics of the Batch Normalization (BN) layer [11] lead to an underestimated and inaccurate performance prediction. Consequently, instead of using the fixed population statistics, we propose Dynamic BN, which improves the correlation significantly with little computation overhead.

For the second question, most existing works adopt kinds of black-box optimization algorithms which are originally used for hyperparameter search, such as Reinforcement learning (RL) [9; 8], Bayesian optimization [10], etc. However, these optimization algorithms usually introduce lots of new hyperparameters. Consequently, tuning these algorithms could be tricky and time-consuming. To fix this issue, we first carry out a second-order Taylor expansion on the performance prediction function $Score(\cdot, \cdot)$ and then make a two-step approximation of the Hessian matrix for its efficient computation. Finally, we can reformulate the problem (2) as a variant of the Knapsack problem and solve it via greedy search. Our proposed solving algorithm is computationally efficient. For all the experiments we conduct, it takes at most several minutes to finish the search process with a single RTX 2080Ti. What's more, our solving algorithm is hyperparameter-free, which avoids lots of time for manual tuning.

To summarize, we propose a novel and unified two-stage framework to obtain the compression policy automatically, and the main contributions are three-fold:

- To predict the performance of each compression policy efficiently and effectively, we propose Dynamic BN, which improves the correlation significantly with little computation overhead.
- To search for the compression ratio allocation, we propose an efficient and hyperparameter-free solving algorithm based on the proposed Hessian matrix approximation and Knapsack problem reformulation.
- We conduct extensive experiments and analyses to demonstrate the advantages of our method over existing model compression works.

2. Related Works

Pruning: Pruning entails systematically removing parameters from an existing network and can be divided into unstructured and structured ones according to the granularity of sparsity. For unstructured pruning, [3] proposes to zero out the weights whose magnitudes are smaller than a threshold and then fine-tune the pruned network to restore accuracy. To avoid the risk of irretrievable network damage, [12] proposes a dynamic network surgery framework that can recover the incorrectly pruned connections. To facilitate hardware acceleration, various structured pruning methods are introduced. [13] proposes a filter-level pruning method by leveraging the scaling factor of the Batch Normalization layer. By adding a structured sparsity regularizer, [14] proposes to reduce trivial filters, channels, or even layers.

Quantization: Instead of removing parameters, quantization reduces the model size by approximating real-valued weights with lower bit-width fixed-point representations. Since the gradient of the quantization function is zero almost everywhere, most quantization works adopt the straight-through estimator (STE) [15] for its gradient approximation and then fine-tune the quantized weights for better performance. Extremely, [16] constrains the weights to be binary (e.g. -1 or $+1$) or ternary (e.g. -1, 0 or $+1$) values. Consequently, except for model compression, it is possible to obtain acceleration at inference time by replacing the multiply-accumulate operations with cheaper accumulations. For more accurate binary neural networks, [17] proposes the Information Retention Network (IR-Net) to retain the information that consists in the forward activations and backward gradients. Furthermore, [18] proposes the Distribution-sensitive Information Retention Network (DIR-Net) to retain the information of gradients by jointly considering the updating capability and accurate gradient. [19] extends the above ideas to the pre-trained language models and proposes BiBERT towards the accurate fully binarized BERT.

Automatic Model Compression: Traditional model compression methods, either pruning or quantization, usually set the compression ratio of each layer through hand-crafted heuristics. As the design space is so huge that human heuristic is usually sub-optimal and the manual setting is time-consuming, recent works propose kinds of methods to determine these parameters automatically. In summary, we can divide these methods into one-stage and two-stage ones. For one-stage methods, they aim to optimize the problem (1) directly. For example, [20] learns the continuous unconstrained model weight W and discrete constrained compression ratio of each layer c simultaneously via the Alternating Direction Method of

Multipliers (ADMM). For two-stage methods, they solve the two sub-problems (2) and (3) sequentially. Specifically, [8] and [9] adopt reinforcement learning (RL) for DNNs pruning and quantization, respectively. [10] proposes to conduct pruning and quantization simultaneously and uses Bayesian optimization to search for the optimal compression ratio allocation as the hyper-parameter.

Second-order based Compression: The history of second-order information in model compression can be traced to the 1990s. For example, OBD [21] proposes network pruning based on second-order derivatives. Then, [22] extends the above ideas to deep neural networks. What's more, the HAWQ method proposed in [23] allows mixed-precision quantization based on the Hessian information. On the other hand, as the computation and storage complexity of the Hessian matrix is quadratic to the number of parameters, different approximations are made to simplify the calculation and make the storage more flexible. Among them, HAWQ [23] only utilizes the top Hessian eigenvalue to measure each layer's quantization sensitivity. Furthermore, HAWQ-V2 [24] consider the full Hessian spectrum, namely, the trace of the Hessian matrix, to determine the bit-width of each layer for mixed-precision quantization. Compared with existing ones, our Hessian approximation demonstrates a better trade-off between effectiveness and efficiency for quantization and pruning.

3. Methodology

In this section, we first introduce the notations and background briefly. Then, we detail the proposed Dynamic BN and solving algorithm step by step by answering the two questions raised for the sub-problem (2).

3.1. Background

Neural Networks: We denote a L-layer Deep Neural Network $f : \Omega \times \mathbb{X} \to \mathbb{Y}$ and a training dataset of N samples $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}) \in \mathbb{X} \times \mathbb{Y}$ with $n = 1, \ldots, N$. The model maps each sample $\mathbf{x}^{(n)}$ to a prediction $\hat{\mathbf{y}}^{(n)}$ with the given parameters $\theta \in \Omega$. Then the prediction is compared with the ground truth $\mathbf{y}^{(n)}$ and evaluated with a task-specific loss function $\ell : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}^+$, which leads to the following training loss to minimize $\mathcal{L} : \Omega \to \mathbb{R}$,

$$\begin{aligned} \mathcal{L}(\theta) &= \frac{1}{N} \sum_{n=1}^{N} \ell(f(\theta, \mathbf{x}^{(n)}), \mathbf{y}^{(n)}) \\ &= \frac{1}{N} \sum_{n=1}^{N} \ell^{(n)}(\theta). \end{aligned} \tag{4}$$

For the l-th convolution or full-connected layer, we denote the weight tensor as $W^{(l)}$ and its flattened version as $w^{(l)}$.

Quantization&Pruning: In general, we can formulate the quantization and pruning as the projection function $\mathcal{P} : \mathbb{R}^d \times \mathbb{C} \to \Pi_c$, which takes the original weight vector and compression ratio c as input and outputs the compressed weight vector. *For quantization and pruning, the compression ratio c denotes the quantization bit-width b and sparsity ratio r, respectively.* As our model compression framework is independent of the specific compression method (e.g., pruning, quantization, etc.), for brevity, we use the notation c to represent any single compression method or the combination of them in the following. Besides, in this paper, *we only consider uniform symmetric quantization and unstructured pruning for model weight compression.* Then, for quantization, Π_b equals to $s \times \{-2^{b-1}, \ldots, 0, \ldots, 2^{b-1} - 1\}^d$ with signed vectors, where s is the step size between two consecutive grid points. For pruning, Π_r represents the set of all

d-dimensional vectors with a sparsity ratio greater than or equal to r. In this paper, we adopt Mean Squared Error (MSE) as the the quantization and pruning criterion. Then, with the given compression ratio c, the above projection function is equivalent to the following minimization problem

$$\mathcal{P}(w, c) = \arg\min_{\hat{w} \in \Pi_c} \|w - \hat{w}\|_2. \tag{5}$$

For pruning, with the given sparsity ratio r, $\mathcal{P}(w, r)$ means that we first sort the elements in the vector in ascending order according to the magnitude and then set the first $r \times 100\%$ elements to be zero. For quantization, with the given bit-width b and step size s, we have

$$\mathcal{P}(w, b; s) = s \times clip(\lfloor w/s \rceil, -2^{b-1}, 2^{b-1} - 1). \tag{6}$$

What's more, we need to first solve the following optimization problem

$$s^* = \arg\min_{s} \|w - \mathcal{P}(w, b; s)\|_2 \tag{7}$$

to obtain the optimal step size s^* with the given bit-width b. Different methods, such as alternating optimization and one-dimensional grid search, can be adopted to solve it. Hence, we denote $\mathcal{P}(w, b) = \mathcal{P}(w, b; s^*)$ in the following.

3.2. Performance Prediction Function

For the first question, we need to give the specific form of the function $Score(\cdot, \cdot)$ so that we can predict the performance of each compression policy. As the real performance obtained by fine-tuning the compressed network until convergence is too time-consuming, an efficient and widely used delegate is the performance of the network that all layers are compressed directly without any fine-tuning. Therefore, as a preliminary attempt, we first explore the effectiveness

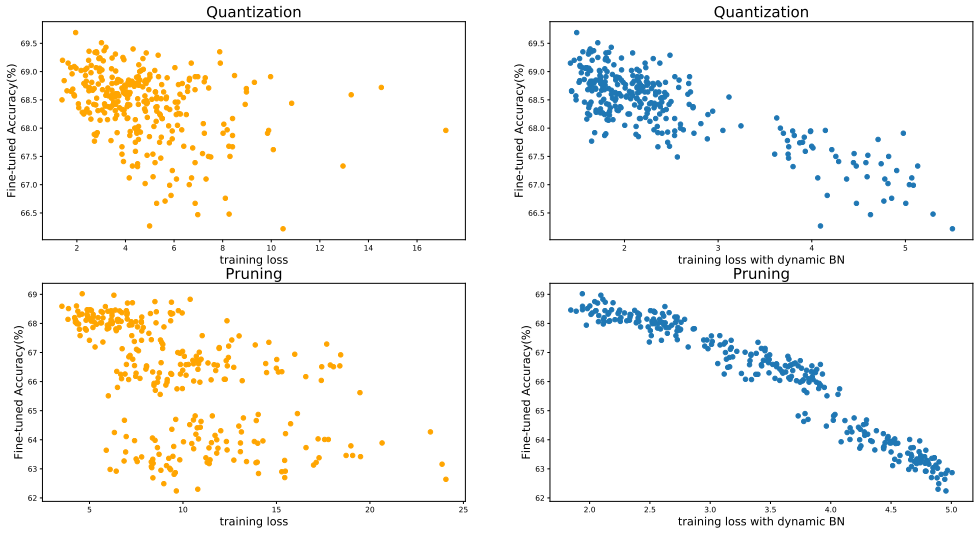


Figure 1: We randomly sample different compression ratio allocations of ResNet-20 on the CIFAR100 and compare the effectiveness of different performance predictors (training loss (**left**) v.s. training loss with Dynamic BN (**right**)) for quantization and pruning.

of this performance prediction function. *To keep the differentiability of the function* $Score(\cdot, \cdot)$ *with regard to the model weights, we run the inference with the compressed weights and utilize the training loss (4) as the performance predictor.* As we can see from the left of Figure 1, the relevance between the real and predicted performance is very low for either quantization or pruning.

To improve the performance predictor, we examine the impact of weight compression on network inference. The basic computation unit of convolution and full-connected layer is the inner product between weight vector $w \in \mathbb{R}^d$ and input vector $x \in \mathbb{R}^d$. We consider the perturbation due to weight compression as $\Delta w \in \mathbb{R}^d$ and denote the compressed weight vector as $\hat{w} = w + \Delta w$. Then, we denote the original and compressed output as $y = wx \in \mathbb{R}$ and $\hat{y} = \hat{w}x \in \mathbb{R}$, respectively. We assume that all elements $\{w_i\}_{i=1}^d$ $\left(\{x_i\}_{i=1}^d \text{ and } \{\Delta w_i\}_{i=1}^d\right)$ are

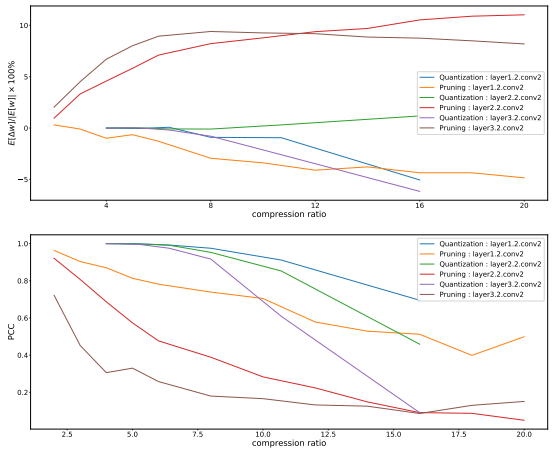


Figure 2: To justify the deviation of both expectation and variance, we first randomly take several layers of ResNet-20 on the CIFAR100, and then calculate the $\frac{E[\Delta w]}{|E[w]|}$ (**top**) and Pearson Correlation Coefficient (PCC) between $\hat{w}x$ and wx (**bottom**) for both quantization and pruning under different compression ratios.

i.i.d with probability density function of p_w (p_x and $p_{\Delta w}$). Besides, p_x and p_w ($p_{\Delta w}$) are mutually independent. Based on the above assumptions, we first have the expectation of compressed output

$$\begin{aligned} E[\hat{y}] &= E[\hat{w}x] = E[(w+\Delta w)x] = E[wx+\Delta wx] \\ &= E[y] + E[\Delta wx] = E[y] + E\Big[\sum_{i=1}^{d} \Delta w_i x_i\Big] \\ &= E[y] + \sum_{i=1}^{d} E[\Delta w_i x_i] = E[y] + \sum_{i=1}^{d} E[\Delta w_i]E[x_i] \\ &= E[y] + d \cdot \mu_{\Delta w} \cdot \mu_x, \end{aligned} \tag{8}$$

where $\mu_{\Delta w}$ and μ_x represent the expectation of $p_{\Delta w}$ and p_x, respectively. Conventionally, the input vector x is the output of the previous non-linear activation layer. For current commonly used activation functions, we have $E[x] \neq 0$ (e.g.

$E[x] > 0$ for ReLU layer). Hence, we can conclude that $E[\hat{y}] = E[y]$ if and only if $E[\Delta w] = 0$. To examine the condition, we randomly take several layers of ResNet-20 on the CIFAR100 and calculate the $\frac{E[\Delta w]}{|E[w]|}$ for quantization and pruning under different compression ratios. The experimental results in the top of Figure 2 show that $E[\Delta w]$ deviates significantly from zero in most cases. Except for the expectation, we also have the variance of compressed output

$$\begin{aligned} D[\hat{y}] &= D[\hat{w}x] = D[(w + \Delta w)x] = D[wx + \Delta wx] \\ &= D[wx] + D[\Delta wx] + 2 \cdot Conv(wx, \Delta wx), \end{aligned} \tag{9}$$

where $Conv(\cdot, \cdot)$ represents the covariance of two given random variables. Then, as

$$\begin{aligned} D[\Delta wx] &= Conv(\Delta wx, \Delta wx) = Conv((\hat{w} - w)x, \Delta wx) \\ &= Conv(\hat{w}x, \Delta wx) - Conv(wx, \Delta wx), \end{aligned} \tag{10}$$

we finally have

$$D[\hat{y}] = D[y] + Conv(\hat{w}x, \Delta wx) + Conv(wx, \Delta wx). \tag{11}$$

Hence, we conclude that $D[\hat{y}] = D[y]$ if and only if $\hat{w}x = -wx$. Similarly, to examine the condition, we randomly take several layers of ResNet-20 on the CIFAR100 and calculate the Pearson Correlation Coefficient (PCC) between $\hat{w}x$ and wx for quantization and pruning under different compression ratios. The experimental results at the bottom of Figure 2 show that there exists a strong positive correlation between $\hat{w}x$ and wx in most cases.

In summary, the above theoretical analysis shows a deviation of both expectation and variance for the compressed output $\hat{y}$. Besides, as shown in the left and middle of Figure 3, we also empirically notice the huge gap between the distribution of intermediate layers' feature maps for the pre-trained and compressed weight.

The previous works (e.g., [25]) also empirically notice the deviation of expectation only and propose bias correction to compensate for the accuracy degradation. Furthermore, in this paper, we propose that such deviation not only degrades the model performance but also leads to inaccurate performance prediction. Consequently, we propose Dynamic BN and detail it in the following.

Dynamic BN: Mathematically, the core of Batch Normalization layer is the following normalization operation:

$$y = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}, \tag{12}$$

where μ and σ denote the mean and variance statistics of layer input, respectively. To avoid interdependence among samples in the same batch, the BN layer adopts the fixed population statistics during inference. However, as the theoretical analysis and experimental results in Figure 3 show, there exists a huge deviation of these two statistics for the intermediate features. This observation enlightens us to improve the performance predictor by correcting the statistical deviation for the BN layer. A naive implementation method is to update the statistics offline by recalculating them on the training data, which makes the performance predictor non-differentiable to the model weights. Instead, to fix the issue while ensuring differentiability, we propose Dynamic BN and turn to mini-batch statistics

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i, \quad \sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2, \tag{13}$$

during our performance prediction with the training loss (4). As shown in the right of Figure 3, Dynamic BN can fill the gap of feature maps' distribution at the cost of little additional computation for mini-batch statistics calculation. Consequently, as shown in the right of Figure 1, the relevance between the real and predicted performance is improved significantly.

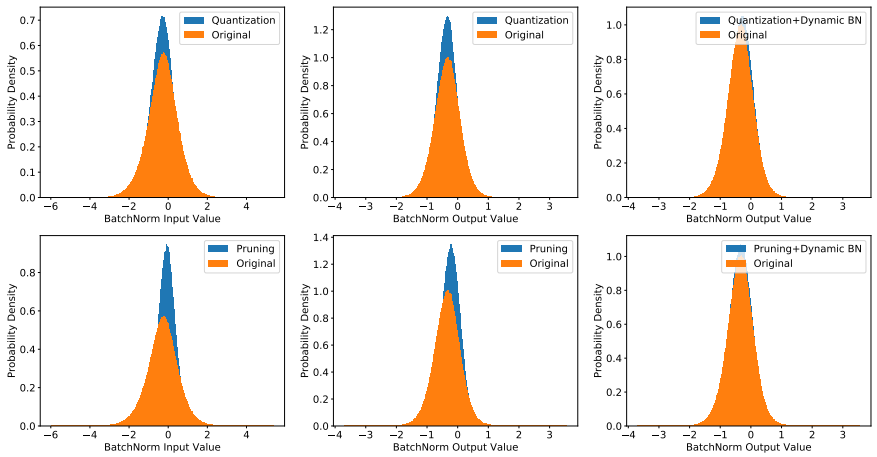


Figure 3: We randomly take a Batch Normalization layer from ResNet-20 on the CIFAR100 and compare the distribution of its input and output for the pre-trained weight (Original), compressed weight (Quantization/Pruning), and compressed weight with Dynamic BN (Quantization/Pruning + Dynamic BN).

3.3. Compression Ratio Determination

For the second question, with the given performance predictor, we need to solve the corresponding discrete constrained optimization problem (2) for the optimal compression ratio allocation. For brevity, here we only consider the original training loss without Dynamic BN as the prediction function in the following. The conclusions still hold when Dynamic BN is involved. First, we instantiate our formulation of the problem (2) as below:

$$\begin{aligned} \min_{\{c^{(l)}\}_{l=1}^{L}} \ & \frac{1}{N}\sum_{n=1}^{N} \ell(f(w+\Delta w, \mathbf{x^{(n)}}), \mathbf{y^{(n)}}) \\ \text{s.t.} \quad & \Delta w^{(l)} = \mathcal{P}(w^{(l)}, c^{(l)}) - w^{(l)} \\ & \sum_{l=1}^{L} \mathcal{M}(c^{(l)}, w^{(l)}) \leq target\_size, \ \ c^{(l)} \in \mathbb{C} \end{aligned} \tag{14}$$

where $\mathbb{C}$ denotes the candidate compression ratios of each layer. The function $\mathcal{M}(\cdot)$ calculates the size of compressed weight, which means

$$\mathcal{M}(b^{(l)}, w^{(l)}) = |w^{(l)}| \cdot \frac{b^{(l)}}{32} \tag{15}$$

for quantization as we assume the original weights are stored in single-precision, and

$$\mathcal{M}(r^{(l)}, w^{(l)}) = |w^{(l)}| \cdot (1 - r^{(l)}) \tag{16}$$

for unstructured pruning. $|\cdot|$ denotes the number of vector elements. Other notations follow the previous definitions.

However, the above problem is computationally expensive as we need to evaluate the network on the whole training dataset for all the feasible compression ratio allocations. Instead, to make the optimization computationally tractable, we first replace the original training loss with its second-order Taylor expansion

$$\begin{aligned} \mathcal{L}(w + \Delta w) &= \frac{1}{N} \sum_{n=1}^{N} \ell^{(n)}(w + \Delta w) \\ &\approx \mathcal{L}(w) + g_w^T \Delta w + \frac{1}{2} \Delta w^T H_w \Delta w, \end{aligned} \tag{17}$$

where $g_w = \nabla \mathcal{L}(w)$ denotes the gradient vector and $H_w = \nabla^2 \mathcal{L}(w)$ denotes the Hessian matrix. For the given pre-trained model weight, it's reasonable to assume that it has converged to a local minimum with nearly zero gradient vector. As a result, we can ignore both the zeroth-order constant and first-order term, and only keep the second-order loss perturbation

$$\Delta \mathcal{L} = \frac{1}{2} \Delta w^T H_w \Delta w \tag{18}$$

caused by model compression. However, on the one hand, the complexity of the Hessian matrix is quadratic to the number of weights. On the other hand, the

interaction between different weights only allows the black-box search algorithms, which have many hyper-parameters, for problem-solving. To fix these two issues, we propose a two-step Hessian approximation (i.e., GGN and Block-diagonal Approximation) and detail it in the following.

GGN Approximation: The original Gauss-Newton algorithm is an approximation to Newton's method for nonlinear least square problems, $\mathcal{L}(\theta) = \frac{1}{2}\sum_n (f(\theta, \mathbf{x}^{(n)}) - \mathbf{y}^{(n)})^2$. [26] generalized this idea as the following:

Definition 1. *For objectives of the form $\mathcal{L}(\theta) = \sum_n a_n(b_n(\theta))$, with $b_n : R^D \to R^M$ and $a_n : R^M \to R$, the Hessian can be written as*

$$\begin{aligned}\nabla^2\mathcal{L}(\theta) = &\sum_n (\nabla_\theta b_n(\theta))^T \nabla^2_{b_n} a_n(b_n(\theta))(\nabla_\theta b_n(\theta)) \\ &+ \sum_{n,m} [\nabla_{b_n} a_n(b_n(\theta))]_m \nabla^2_\theta b_n^{(m)}(\theta),\end{aligned} \tag{19}$$

where $[\cdot]_m$ selects the m-th component of a vector, and $b_n^{(m)}$ denotes the m-th component function of b_n. The generalized Gauss-Newton matrix (GGN) is defined as the part of Hessian that ignores the second-order information of b_n,

$$G(\theta) := \sum_n (\nabla_\theta b_n(\theta))^T \nabla^2_{b_n} a_n(b_n(\theta))(\nabla_\theta b_n(\theta)). \tag{20}$$

In general, the task-specific loss function (e.g., cross-entropy loss and square-error loss) ℓ is convex. Therefore, to ensure the approximated Hessian to be positive semidefinite and loss perturbation to be non-negetive, we decouple the training loss (4) as $b_n = f^{(n)}(w)$ and $a_n = \ell^{(n)}(b_n)$. According to (20), we obtain the following Hessian approximation

$$\tilde{H}_w = \frac{1}{N}\sum_{n=1}^{N} (\nabla f^{(n)}(w))^T \Sigma^{(n)} (\nabla f^{(n)}(w)), \tag{21}$$

where $\nabla f^{(n)}(w)$ denotes the Jacobian matrix of $f^{(n)}$ on w, $\Sigma^{(n)}$ denotes the Hessian matrix of $\ell^{(n)}$ on $f^{(n)}$. On the other hand, for any given loss function ℓ, we can write its $\Sigma^{(n)}$ explicitly. Take the cross-entropy loss as an example. It's easy to derive that

$$\Sigma_{ij}^{(n)} = \begin{cases} \frac{1}{(f_t^{(n)})^2} & i = j = t, \\ 0 & \text{otherwise}, \end{cases} \tag{22}$$

where t denotes the ground-truth label. Consequently, we reduce the complexity of the Hessian matrix from quadratic growth to linear growth concerning the number of weights. Then, we substitute the above approximation into Eq. (18) and approximate the loss perturbation as

$$\begin{aligned} \Delta\tilde{\mathcal{L}} &= \frac{1}{2}\Delta w^T \tilde{H}_w \Delta w \\ &= \frac{1}{2N}\sum_{n=1}^{N}[\nabla f^{(n)}\Delta w]^T \Sigma^{(n)}[\nabla f^{(n)}\Delta w]. \end{aligned} \tag{23}$$

Block-diagonal Approximation: To introduce more priors about the structure of the optimization problem (14), we assume that the Hessian matrix is block-diagonal with only non-zero terms for weights within the same layer. Therefore, we can further approximate the loss perturbation as

$$\Delta\tilde{\mathcal{L}}_{block} = \frac{1}{2}\sum_{l=1}^{L}(\Delta w^{(l)})^T \tilde{H}_{w^{(l)}} \Delta w^{(l)}. \tag{24}$$

Finally, based on the above two-step approximation, we can reformulate the problem (14) as

$$\begin{aligned} \min_{\{c^{(l)}\}_{l=1}^{L}} \ & \frac{1}{2}\sum_{l=1}^{L}(\Delta w^{(l)})^T \tilde{H}_{w^{(l)}} \Delta w^{(l)} \\ \text{s.t.} \quad & \Delta w^{(l)} = \mathcal{P}(w^{(l)}, c^{(l)}) - w^{(l)} \\ & \sum_{l=1}^{L}\mathcal{M}(c^{(l)}, w^{(l)}) \leq target\_size, \ \ c^{(l)} \in \mathbb{C} \end{aligned} \tag{25}$$

Obviously, the above problem (25) can be transformed into an integer linear programming problem. More specifically, it is equivalent to a special variant of the Knapsack problem called the Multiple-Choice Knapsack Problem (MCKP) [27]. Consequently, we propose a greedy search algorithm to solve it efficiently. More specifically, we choose a layer with the highest priority based on the proposed greedy criterion and then decrease its compression ratio. We repeat the above step until the target model size constraint is broken. The detailed steps are as follows:

1) Sort the candidate compression ratios of each layer in descending order and denote it as $\{c_i\}_{i=0}^{K}$.

2) Initialize each layer with the largest candidate compression ratio and denote the current solution as $c_{i^{(l)}}$.

3) Calculate the incremental profit density

$$\frac{\Delta\tilde{\mathcal{L}}^{(l)}_{block,c_{i^{(l)}}} - \quad \Delta\tilde{\mathcal{L}}^{(l)}_{block,c_{i^{(l)}+1}}}{\mathcal{M}(c_{i^{(l)}+1}, w^{(l)}) - \mathcal{M}(c_{i^{(l)}}, w^{(l)})} \tag{26}$$

for each layer and update the compression ratio of the layer l^* with the largest one to $c_{i^{(l^*)}+1}$.

4) Repeat step 3) until the target model size constraint is broken.

3.4. Summary of Algorithm Procedure

In the previous sections, we have introduced the two key techniques (i.e., Dynamic BN and efficient solving algorithm) in detail. To further understand the process of determining the optimal compression ratio allocation, we combine these two components and summarize the overall procedure in the Algorithm 1. Please refer to it for more implementation details.

Algorithm 1 Determination for the Optimal Compression Ratio Allocation

Input: training dataset $\{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$; pre-trained network with weights $\{w^{(l)}\}_{l=1}^{L}$; candidate compression ratios of each layer $\mathbb{C}$; target model size $\mathcal{T}$

Output: compression ratio of each layer $\{c_{i^{(l)}}\}_{l=1}^{L}$

/* Step 1: *calculate loss perturbation of each layer* */

calculate $\{\{\Delta w_c^{(l)} = \mathcal{P}(w^{(l)}, c) - w^{(l)}\}_{c\in\mathbb{C}}\}_{l=1}^{L}$ for quantization and/or pruning

initialize $\{\{\Delta\tilde{\mathcal{L}}_{block,c}^{(l)}\}_{c\in\mathbb{C}}\}_{l=1}^{L}$ with zero

for $n = 1$ **to** N **do**

  set *model.train()* for Dynamic BN

  compute output and gradient for $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$

  update $\{\{\Delta\tilde{\mathcal{L}}_{block,c}^{(l)}\}_{c\in\mathbb{C}}\}_{l=1}^{L}$ according to Eq. (24)

end for

/* Step 2: *greedy search for optimal compression ratio allocation* */

sort candidate compression ratios $\mathbb{C}$ in descending order and denote it as $\{c_i\}_{i=0}^{K}$

initialize the current compression ratio of each layer $c_{i^{(l)}}$ as $i^{(l)} = 0$

while cuurent model size below the target $\mathcal{T}$ **do**

  for $l = 1$ **to** L **do**

    obtain the next compression ratio $c_{i^{(l)}+1}$ and its corresponding loss perturbation $\Delta\mathcal{L}_{block,c_{i^{(l)}+1}}^{(l)}$

    calculate the incremental profit density of current layer as $\frac{\Delta\tilde{\mathcal{L}}_{block,c_{i^{(l)}}}^{(l)} - \Delta\tilde{\mathcal{L}}_{block,c_{i^{(l)}+1}}^{(l)}}{\mathcal{M}(c_{i^{(l)}+1}, w^{(l)}) - \mathcal{M}(c_{i^{(l)}}, w^{(l)})}$

  end for

  sort the incremental profit density among layers, denote the layer with the largest one as $l^{\star}$, then update the compression policy by $i^{(l^{\star})} \leftarrow i^{(l^{\star})} + 1$

end while

What's more, we analyze the computational complexity of the above model compression framework. As shown in the Algorithm 1, the computation cost of our method is mainly composed of two parts, namely the calculation of loss perturbation and the greedy search to solve MCKP. For the calculation of loss perturbation, we denote the number of images as N and the computational complexity of network forward/backward propagation for a single image as T. Then its total computational complexity is $\mathcal{O}(NT)$. Empirically, the loss perturbation converges rapidly, and a few hundred images are enough for its approximation. For the greedy search to solve MCKP, we denote the number of candidate compression ratios as $|\mathbb{C}|$ and the number of layers as L. First, the computational complexity of picking the largest element from an unordered sequence is $\mathcal{O}(L)$. As the maximum number of iterations of the loop is $|\mathbb{C}|L$, its total computational complexity is $\mathcal{O}(|\mathbb{C}|L^2)$. In summary, the total computational complexity for determining the optimal compression ratio allocation is $\mathcal{O}(NT + |\mathbb{C}|L^2)$.

4. Experiments

In this section, we first introduce the experimental setup thoroughly from different aspects. Then, we conduct a detailed comparative analysis of the proposed Dynamic BN and Hessian approximation to justify their effectiveness. Furthermore, we show the fine-tuned accuracy results of our compressed models and compare them with other state-of-the-art methods. Finally, we visualize the compression ratio of each layer to understand what our method learns.

4.1. Experimental Setup

Datasets: We conduct analysis on the CIFAR-100 dataset [28], which consists of 50k training images and 10k testing images in 100 classes. We compare with other

state-of-the-art compression methods on the ImageNet dataset [29], which consists of 1.2M training images and 50k validation images in 1000 classes. We report the Top-1/5 accuracy on the validation split for the performance measure. We perform standard data augmentation [1] for these two datasets. More specifically, for CIFAR-100, 4 pixels are padded on each side, and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. For ImageNet, the image is first resized so that the shorter side is 256, then a 224 × 224 crop is randomly sampled from the image or its horizontal flip, with the per-channel mean subtracted.

Table 1: Correlation analysis for different performance predictors under different model size constraints. The 'NZ-x%' and 'Ave. y-bits' mean that the percentage of non-zero elements and averaged bit-width of the model weights are x% and y-bits, respectively.

Method	Constraint	Original Loss			+Dynamic BN		
		PCC	SCC	Tau	PCC	SCC	Tau
Quant	Ave. 3-bits	0.23	0.23	0.15	0.86	0.66	0.49
	Ave. 4-bits	0.37	0.40	0.27	0.78	0.68	0.51
	Ave. 5-bits	0.40	0.35	0.25	0.79	0.66	0.50
Pruning	NZ-50%	0.12	0.11	0.07	0.62	0.63	0.45
	NZ-25%	0.10	0.16	0.11	0.60	0.59	0.42
	NZ-20%	0.06	0.13	0.08	0.88	0.89	0.71

Network Architectures: We evaluate our method on a wide range of network architectures. We use the ResNet-20 and ResNet-18/50 as in [1] for CIFAR-100 and ImageNet, respectively. What's more, we evaluate our method on the more challenging lightweight networks, including MobileNet-V1 [30] and MobileNet-V2 [31]. Except for MobileNet-V1, all the other models pre-trained on ImageNet are taken from the Pytorch model zoo.

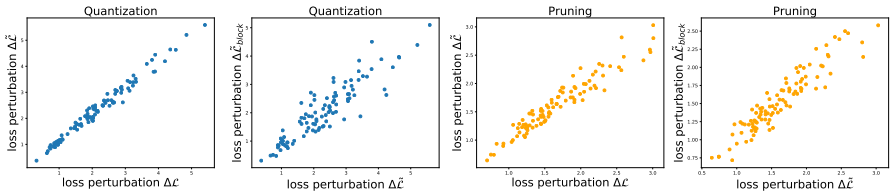
Model Compression Pipeline: As stated before, we adopt a two-stage framework

and decouple the model compression problem as the following two sub-ones. *First*, determine the optimal compression ratio allocation and quantize/prune the pre-trained model weight with the solved compression policy. More specifically, for quantization-only, the candidate compression ratios mean the candidate bit-widths ranging from 1-bit to 8-bit. For pruning-only, the candidate compression ratios mean the candidate sparsity ratios ranging from 10% to 95%. When quantization and pruning are conducted simultaneously, each candidate compression ratio corresponds to a tuple which consists of both quantization bit-width and sparsity ratio. As we note that the calculation result of (24) converges rapidly with a few hundred images, we only randomly sample 1024 images from the training set to figure out the approximate loss perturbation in the later experiments. *Second*, fine-tune the model weights with the solved compression ratio of each layer. The fine-tuning details are introduced in the following.

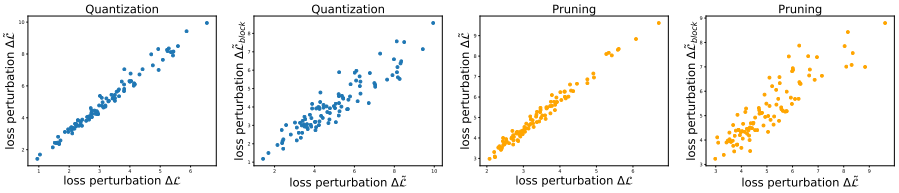
Fine-tuning Details: We perform fine-tuning after quantization and/or pruning to restore accuracy. We use the SGD optimizer with a momentum of 0.9 and a mini-batch size of 128/256 for the experiments on CIFAR-100/ImageNet. For quantization, we adopt the Straight-Through Estimator (STE) [16] to estimate the gradient. We use the cosine annealing strategy for all the experiments to decay the learning rate. For method analysis on CIFAR-100, fine-tuning is performed for 120 epochs with the initial learning rate/weight decay of 0.01/5e-4. For compression results on ImageNet, fine-tuning is performed for 60 epochs with an initial learning rate of 0.05. The weight decay is 1e-4 for ResNet-18/50 and 5e-5 for MobileNet-V1/V2, respectively.

Calculation of Compression Ratio: Our calculation of compression ratio is consistent with the existing works [10; 20]. For quantization, we assume the

original weights are stored in single precision of 32-bit, so b-bit quantization means compression ratio of $\frac{32}{b}\times$. For pruning, the sparsity ratio of r means the percentage of non-zero weights is $1-r$, and the corresponding compression ratio is $\frac{1}{1-r}\times$. When quantization and pruning are conducted simultaneously, the compression configuration of (b, r) means the compression ratio of $\frac{32}{b}\cdot\frac{1}{1-r}\times$.



(a) ResNet-18 on ImageNet



(b) ResNet-20 on CIFAR-100

Figure 4: Quantitative Analysis for the proposed two-step Hessian approximation. We take ResNet-18 on the ImageNet and ResNet-20 on the CIFAR-100 as examples and randomly sample 100 different compression ratio allocations for quantization and pruning, respectively. Then we compute and compare the corresponding true ($\Delta\mathcal{L}$) and two approximated ($\Delta\tilde{\mathcal{L}}$ and $\Delta\tilde{\mathcal{L}}_{block}$) loss perturbations for each compression policy.

4.2. Method Analysis

Dynamic BN: We argue that Dynamic BN could improve the performance prediction significantly in Section 3.2 and show the qualitative results in Figure 1. Here we further conduct quantitative analysis. In summary, we focus on two perfor-

mance predictors, namely the original training loss (4) and the improved one with Dynamic BN, and compare their correlation with the real performance. First, we randomly sample 100 different compression ratio allocations for quantization and pruning on ResNet-20 under different model size constraints. Then, we fine-tune each compressed network until convergence and get the corresponding sample pair which consists of the real performance and the value of the specific predictor. Finally, to judge the relevance of different predictors qualitatively, we adopt three widely used indicators that measure the correlation between sequences, namely Pearson Correlation Coefficient (PCC), Spearman Correlation Coefficient (SCC), and Kendall's Tau (Tau). As the predictors we compare are negatively correlated with the real performance, we only report their absolute values here. The results summarized in Table 1 show that Dynamic BN could significantly improve the relevance under different correlation indicators and model size constraints for quantization and pruning.

Hessian Approximation: To solve the optimization problem efficiently, we propose a two-step approximation that includes GGN and Block-diagonal approximation for the Hessian matrix in Section 3.3. Here we further conduct an empirical analysis to justify the effectiveness of the proposed strategies. Similarly, we randomly sample 100 different compression ratio allocations and compute the corresponding true loss perturbation, namely $\Delta\mathcal{L}$, and the two approximated loss perturbations, namely $\Delta\tilde{\mathcal{L}}$ and $\Delta\tilde{\mathcal{L}}_{block}$, for each compression policy. We repeat the above procedure for both ResNet-18 on the ImageNet and ResNet-20 on the CIFAR-100 with quantization and pruning. As the summarized results in Figure 4 show, these points are roughly distributed on a straight line passing through the origin. Therefore, we empirically justify that our approximated Hessian can still

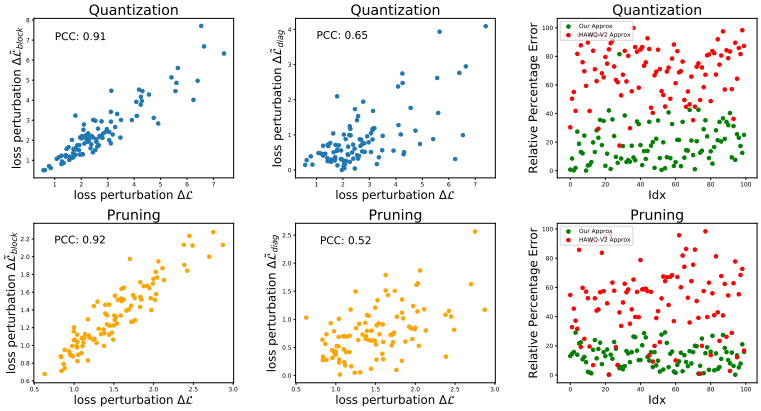


Figure 5: Comparison between the proposed two-step Hessian approximation and HAWQ-V2. We take ResNet-18 on ImageNet as the example and randomly sample 100 different compression ratio allocations for quantization and pruning. Then we compare the correlation (**left** and **middle**) and relative percentage error (**right**) between the true ($\Delta\mathcal{L}$) and two approximated ($\Delta\tilde{\mathcal{L}}_{block}$ and $\Delta\tilde{\mathcal{L}}_{diag}$) loss perturbations for each compression policy.

capture most of the information in the original one with much-reduced complexity.

Furthermore, we compare our two-step approximation with HAWQ-V2 [24], which only utilizes the trace information of the true Hessian matrix. We denote the loss perturbation via HAWQ-V2 approximation as $\Delta\tilde{\mathcal{L}}_{diag}$. The experimental results are summarized in Figure 5. More specifically, we compare with HAWQ-V2 approximation from two different aspects. First, we calculate the Pearson Correlation Coefficient (PCC) between the true loss perturbation ($\Delta\mathcal{L}$) and the two approximated ones ($\Delta\tilde{\mathcal{L}}_{block}$ and $\Delta\tilde{\mathcal{L}}_{diag}$), respectively. As the left and middle of Figure 5 show, our approximation demonstrates a higher correlation. Second, we examine the relative percentage error between the true loss perturbation and the two approximated ones (i.e., $|\frac{\Delta\mathcal{L}-\Delta\tilde{\mathcal{L}}_{block}}{\Delta\mathcal{L}}| \times 100\%$ and $|\frac{\Delta\mathcal{L}-\Delta\tilde{\mathcal{L}}_{diag}}{\Delta\mathcal{L}}| \times 100\%$). As the right of Figure 5 shows, our approximation demonstrates the lower error. All

the above experimental results justify that our two-step Hessian approximation is superior to HAWQ-V2 in estimating the loss perturbation.

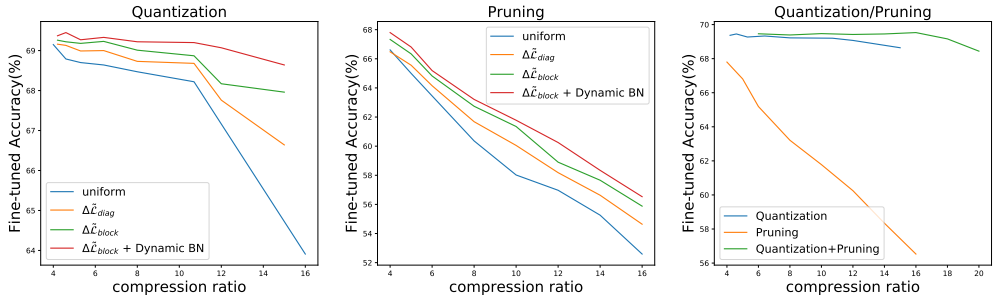


Figure 6: Fine-tuned Top-1 accuracy of ResNet-20 on the CIFAR-100 under different compression ratios. We first ablate the effectiveness of different components of the proposed framework for quantization (**left**) and pruning (**middle**), respectively. Besides, we justify that our method can utilize the complementary nature of quantization and pruning (**right**) and further improve the performance of the compressed model.

4.3. Fine-tuned Results

After analysis of the proposed framework, we further fine-tune the models with the solved compression ratio allocations and compare the accuracy results with other model compression approaches.

4.3.1. Results on the CIFAR-100

We first consider compressing ResNet-20 on the CIFAR100 dataset and summarize the results in Figure 6. As the left and middle of Figure 6 show, all the methods that utilize the second-order information demonstrate better performance than uniform compression ratio allocation for quantization and pruning under different model size constraints. Besides, the experimental results also justify that our two-step Hessian approximation ($\Delta\tilde{\mathcal{L}}_{block}$) is superior to HAWQ-V2 ($\Delta\tilde{\mathcal{L}}_{diag}$) in

Table 2: Summary of compression results on the ImageNet dataset. The 'NZ Ratio' and 'Ave. Bits' refers to the percentage of non-zero elements and averaged bit-width of the model weights, respectively. The 'Comp. Ratio' represents the overall compression ratio.

Network	Method	Automated	Pruning	Quantization	NZ Ratio	Ave. Bits	Comp. Ratio	Top-1	Top-5
ResNet-18	**Full-Precision**	-	-	-	-	-	**1.00×**	**69.76**	**89.08**
	Low-Bit-ADMM [32]	✗	✗	✓	-	3.0	10.67×	68.00	88.30
	LQ-Net [33]	✗	✗	✓	-	3.0	10.67×	69.30	88.80
	Ours	✓	✗	✓	-	**2.6**	**12.40×**	**70.19**	**89.40**
	Ours	✓	✗	✓	-	**2.3**	**13.91×**	**69.63**	**89.06**
	State of Pruning [34]	✗	✓	✗	50.0	-	2.00×	69.71	-
	State of Pruning [34]	✗	✓	✗	12.5	-	8.00×	63.86	-
	Ours	✓	✓	✗	**25.0**	-	**4.00×**	**69.76**	**89.23**
	Ours	✓	✓	✗	**12.5**	-	**8.00×**	**67.29**	**87.86**
ResNet-50	**Full-Precision**	-	-	-	-	-	**1.00×**	**76.13**	**92.86**
	Low-Bit-ADMM [32]	✗	✗	✓	-	3.0	10.67×	74.00	91.60
	LQ-Net [33]	✗	✗	✓	-	4.0	8.00×	76.40	93.10
	HAQ [9]	✓	✗	✓	-	3.0	10.57×	75.30	92.45
	HAQ [9]	✓	✗	✓	-	2.1	15.47×	70.63	89.93
	Ours	✓	✗	✓	-	**3.0**	**10.75×**	**76.92**	**93.31**
	Ours	✓	✗	✓	-	**2.1**	**15.45×**	**76.11**	**92.86**
	L-OBS [22]	✗	✓	✗	45.5	-	2.20×	-	≈85
	GSM [35]	✓	✓	✗	20.0	-	5.00×	74.30	91.98
	LR Rewinding [36]	✗	✓	✗	21.0	-	4.77×	75.30	-
	Ours	✓	✓	✗	**25.0**	-	**4.00×**	**76.24**	**93.01**
	Ours	✓	✓	✗	**20.0**	-	**5.00×**	**76.11**	**92.98**
	Ours	✓	✓	✗	**16.7**	-	**6.00×**	**75.81**	**92.86**
	CLIP-Q [10]	✓	✓	✓	30.6	3.3	31.81×	73.70	-
	Ours	✓	✓	✓	**27.8**	**3.2**	**36.00×**	**75.92**	**92.79**
MobileNet-V1	**Full-Precision**	-	-	-	-	-	**1.00×**	**70.20**	**89.26**
	DC [3]	✗	✗	✓	-	3.2	10.09×	65.93	86.85
	HAQ [9]	✓	✗	✓	-	3.1	10.22×	67.66	88.21
	DC [3]	✗	✗	✓	-	2.2	14.81×	37.62	64.31
	HAQ [9]	✓	✗	✓	-	2.2	14.81×	57.14	81.87
	Ours	✓	✗	✓	-	**3.2**	**10.00×**	**69.32**	**88.80**
	Ours	✓	✗	✓	-	**2.2**	**14.69×**	**65.55**	**86.64**
	Prune or Not Prune [37]	✗	✓	✗	50.5	-	1.98×	69.50	89.50
	Prune or Not Prune [37]	✗	✓	✗	25.9	-	3.86×	67.70	88.50
	Ours	✓	✓	✗	**50.0**	-	**2.00×**	**70.82**	**89.91**
	Ours	✓	✓	✗	**25.1**	-	**3.98×**	**68.25**	**88.30**
	CLIP-Q [10]	✓	✓	✓	52.6	4.6	13.19×	70.30	-
	ANNC [20]	✓	✓	✓	42.0	2.8	26.7×	66.49	87.29
	Ours	✓	✓	✓	**51.8**	**4.4**	**13.97×**	**70.47**	**89.49**
	Ours	✓	✓	✓	**34.7**	**3.3**	**27.82×**	**68.64**	**88.44**
MobileNet-V2	**Full-Precision**	-	-	-	-	-	**1.00×**	**71.88**	**90.29**
	DC [3]	✗	✗	✓	-	4.3	7.47×	71.24	89.93
	HAQ [9]	✓	✗	✓	-	4.3	7.47×	71.47	90.23
	Ours	✓	✗	✓	-	**4.3**	**7.50×**	**71.83**	**90.36**
	DC [3]	✗	✗	✓	-	3.3	9.69×	68.00	87.96
	HAQ [9]	✓	✗	✓	-	3.3	9.69×	70.90	89.76
	Ours	✓	✗	✓	-	**3.3**	**9.76×**	**71.05**	**90.12**
	DC [3]	✗	✗	✓	-	2.3	13.93×	58.07	81.24
	HAQ [9]	✓	✗	✓	-	2.3	14.07×	66.75	87.32
	Ours	✓	✗	✓	-	**2.3**	**14.00×**	**69.21**	**89.07**

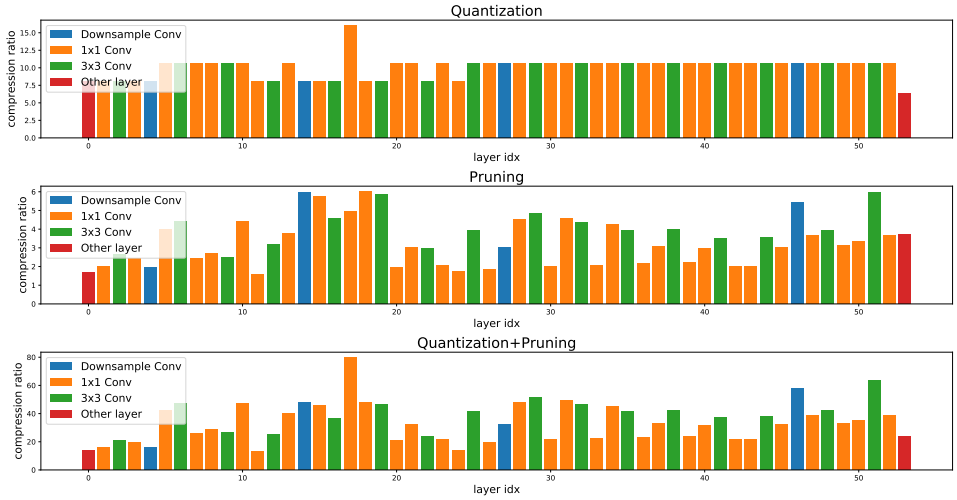
terms of fine-tuned accuracy. Then, we also ablate the effectiveness of the proposed Dynamic BN and Hessian approximation, and justify that these two techniques can be combined to improve the performance further. Finally, as the right of Figure 6 shows, our method can utilize the complementary nature of quantization and pruning and outperform any single one under the same compression ratio by conducting quantization and pruning simultaneously.

4.3.2. Results on the ImageNet

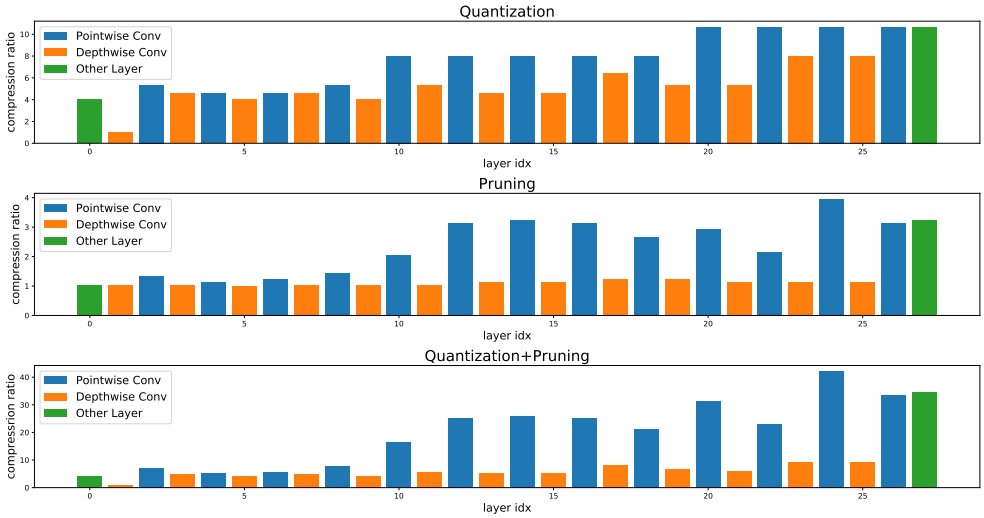
What's more, we compare our method with kinds of state-of-the-art model compression works on the ImageNet and summarize the results in Table 2. To verify the effectiveness of our method comprehensively, we cover the network architectures with a wide range of depth (e.g., ResNet-18 vs. ResNet-50) and different number of parameters (e.g., ResNet-50 vs. MobileNet-V1/V2). Besides, we also cover kinds of compression techniques to justify the generality of our method. According to the degree of automation for determining the compression ratio allocation, we compare with `hand-crafted methods` [32; 33; 34; 3; 22; 36; 37], and `automatic methods` [9; 35; 10; 20]. According to the adopted compression technique, we compare with `quantization-only` [32; 33; 3; 9], `pruning-only` [34; 22; 36; 37; 35] and `pruning-quantization` [10; 20] methods. For these various works considered here, we all obtain significant accuracy improvement under the similar model size constraint.

5. Compression Ratio Allocation Visualization

In this section, we visualize the optimized compression ratio allocations to understand our method further. We take MobileNet-V1 and ResNet-50 as examples and conduct quantization and pruning simultaneously. The results are summarized



(a) Compression ratio allocation for ResNet-50



(b) Compression ratio allocation for MobileNet-V1

Figure 7: Visualization of the optimized compression ratio allocations. We take MobileNet-V1 and ResNet-50 as examples and conduct quantization and pruning simultaneously. For each network architecture, except for the allocated total compression ratio of each layer (**bottom**), we also visualize the specific compression ratio caused by quantization (**top**) and pruning (**middle**), respectively.

in Figure 7. More specifically, except for each layer's allocated total compression ratio, we also visualize the specific contribution of quantization and pruning, respectively. We notice that our method prefers quantization rather than pruning for these two network architectures. This strategy is consistent with the experimental results that, compared with pruning, quantization will cause less performance degradation under the same compression ratio. For MobileNet-V1, we notice that the allocated compression ratios of the pointwise convolution layer are significantly higher than the ones of the depthwise convolution layer. Moreover, for the same type of layer, the allocated compression ratio becomes higher as the network layer becomes deeper. Unfortunately, these empirical rules do not apply to ResNet-50, which indicates that we could not accurately determine the compression ratio of each layer based on some simple human heuristics.

6. Conclusion and Future Work

Deep neural networks have shown great promise in various domains. How to efficiently and effectively compress these deep models is still an open research topic. This paper proposes a novel and unified two-stage framework for automatic model compression. Briefly speaking, we improve the optimization of compression ratio allocation from two aspects. First, we propose Dynamic BN that improves the performance prediction significantly with little additional computation cost. Second, we propose an efficient and hyperparameter-free solving algorithm based on the proposed Hessian matrix approximation and Knapsack problem reformulation. Extensive experiments and analyses are conducted to justify the advantages of our method over existing model compression works. However, our proposed method is partially based on Batch Normalization (BN) layer. In future work, we aim to

extend the framework to BN-free models with a similar idea. Moreover, we will explore combining more compression techniques (e.g., low-rank decomposition) for more downstream tasks (e.g., object detection). Finally, to enhance the proposed automatic model compression framework, we hope to explore more possible trade-offs between effectiveness and efficiency for both performance predictor and solving algorithm.

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[2] Q. Chen, P. Wang, A. Cheng, W. Wang, Y. Zhang, J. Cheng, Robust one-stage object detection with location-aware classifiers, Pattern Recognition 105 (2020) 107334.

[3] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, arXiv preprint arXiv:1510.00149 (2015).

[4] K. Yao, F. Cao, Y. Leung, J. Liang, Deep neural network compression through interpretability-based filter pruning, Pattern Recognition 119 (2021) 108056.

[5] Y. Yang, B. Chen, H. Liu, Bayesian compression for dynamically expandable networks, Pattern Recognition 122 (2022) 108260.

[6] T. Chu, Q. Luo, J. Yang, X. Huang, Mixed-precision quantized neural net-

works with progressively decreasing bitwidth, Pattern Recognition 111 (2021) 107647.

[7] M. Zhang, Y. Zhou, J. Zhao, S. Xia, J. Wang, Z. Huang, Semi-supervised blockwisely architecture search for efficient lightweight generative adversarial network, Pattern Recognition 112 (2021) 107794.

[8] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, S. Han, Amc: Automl for model compression and acceleration on mobile devices, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 784–800.

[9] K. Wang, Z. Liu, Y. Lin, J. Lin, S. Han, Haq: Hardware-aware automated quantization with mixed precision, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 8612–8620.

[10] F. Tung, G. Mori, Clip-q: Deep network compression learning by in-parallel pruning-quantization, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7873–7882.

[11] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International conference on machine learning, PMLR, 2015, pp. 448–456.

[12] Y. Guo, A. Yao, Y. Chen, Dynamic network surgery for efficient dnns, arXiv preprint arXiv:1608.04493 (2016).

[13] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2736–2744.

[14] W. Wen, C. Wu, Y. Wang, Y. Chen, H. Li, Learning structured sparsity in deep neural networks, Advances in neural information processing systems 29 (2016) 2074–2082.

[15] Y. Bengio, N. Léonard, A. C. Courville, Estimating or propagating gradients through stochastic neurons for conditional computation, CoRR abs/1308.3432 (2013). arXiv:1308.3432.

[16] M. Courbariaux, Y. Bengio, J.-P. David, Binaryconnect: Training deep neural networks with binary weights during propagations, in: Advances in neural information processing systems, 2015, pp. 3123–3131.

[17] H. Qin, R. Gong, X. Liu, M. Shen, Z. Wei, F. Yu, J. Song, Forward and backward information retention for accurate binary neural networks, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 2250–2259.

[18] H. Qin, X. Zhang, R. Gong, Y. Ding, Y. Xu, X. Liu, Distribution-sensitive information retention for accurate binary neural network, International Journal of Computer Vision 131 (1) (2023) 26–47.

[19] H. Qin, Y. Ding, M. Zhang, Q. Yan, A. Liu, Q. Dang, Z. Liu, X. Liu, Bibert: Accurate fully binarized bert, arXiv preprint arXiv:2203.06390 (2022).

[20] H. Yang, S. Gui, Y. Zhu, J. Liu, Automatic neural network compression by sparsity-quantization joint learning: A constrained optimization-based approach, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 2178–2188.

[21] Y. LeCun, J. Denker, S. Solla, Optimal brain damage, Advances in neural information processing systems 2 (1989).

[22] X. Dong, S. Chen, S. J. Pan, Learning to prune deep neural networks via layer-wise optimal brain surgeon, arXiv preprint arXiv:1705.07565 (2017).

[23] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, K. Keutzer, Hawq: Hessian aware quantization of neural networks with mixed-precision, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 293–302.

[24] Z. Dong, Z. Yao, Y. Cai, D. Arfeen, A. Gholami, M. W. Mahoney, K. Keutzer, HAWQ-V2: hessian aware trace-weighted quantization of neural networks, CoRR abs/1911.03852 (2019). arXiv:1911.03852.

[25] A. Finkelstein, U. Almog, M. Grobman, Fighting quantization bias with bias, CoRR abs/1906.03193 (2019). arXiv:1906.03193.

[26] N. N. Schraudolph, Fast curvature matrix-vector products for second-order gradient descent, Neural Comput. 14 (7) (2002) 1723–1738. doi:10.1162/08997660260028683.

[27] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack problems, Springer, 2004. doi:10.1007/978-3-540-24777-7.

[28] A. Krizhevsky, Learning multiple layers of features from tiny images, 2009.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[30] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017).

[31] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.

[32] C. Leng, Z. Dou, H. Li, S. Zhu, R. Jin, Extremely low bit neural network: Squeeze the last bit out with admm, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[33] D. Zhang, J. Yang, D. Ye, G. Hua, Lq-nets: Learned quantization for highly accurate and compact deep neural networks, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 365–382.

[34] D. Blalock, J. J. G. Ortiz, J. Frankle, J. Guttag, What is the state of neural network pruning?, arXiv preprint arXiv:2003.03033 (2020).

[35] X. Ding, G. Ding, X. Zhou, Y. Guo, J. Han, J. Liu, Global sparse momentum sgd for pruning very deep neural networks, arXiv preprint arXiv:1909.12778 (2019).

[36] A. Renda, J. Frankle, M. Carbin, Comparing rewinding and fine-tuning in neural network pruning, arXiv preprint arXiv:2003.02389 (2020).

[37] M. Zhu, S. Gupta, To prune, or not to prune: exploring the efficacy of pruning for model compression, arXiv preprint arXiv:1710.01878 (2017).