

OPQ: Compressing Deep Neural Networks with One-shot Pruning-Quantization

Peng Hu^{1,2}, Xi Peng², Hongyuan Zhu¹, Mohamed M. Sabry Aly³, Jie Lin^{1,*}

¹Institute for Infocomm Research, Agency for Science, Technology and Research, Singapore

²College of Computer Science, Sichuan University, Chengdu 610065, China

³Nanyang Technological University, Singapore

{penghu.ml, pengx.gm, hongyuanzhu.cn}@gmail.com; msabry@ntu.edu.sg; lin-j@i2r.a-star.edu.sg

Abstract

As Deep Neural Networks (DNNs) usually are overparameterized and have millions of weight parameters, it is challenging to deploy these large DNN models on resource-constrained hardware platforms, *e.g.*, smartphones. Numerous network compression methods such as pruning and quantization are proposed to reduce the model size significantly, of which the key is to find suitable compression allocation (*e.g.*, pruning sparsity and quantization codebook) of each layer. Existing solutions obtain the compression allocation in an iterative/manual fashion while finetuning the compressed model, thus suffering from the efficiency issue. Different from the prior art, we propose a novel One-shot Pruning-Quantization (OPQ) in this paper, which analytically solves the compression allocation with pre-trained weight parameters only. During finetuning, the compression module is fixed and only weight parameters are updated. To our knowledge, OPQ is the first work that reveals pre-trained model is sufficient for solving pruning and quantization simultaneously, without any complex iterative/manual optimization at the finetuning stage. Furthermore, we propose a unified channel-wise quantization method that enforces all channels of each layer to share a common codebook, which leads to low bit-rate allocation without introducing extra overhead brought by traditional channel-wise quantization. Comprehensive experiments on ImageNet with AlexNet/MobileNet-V1/ResNet-50 show that our method improves accuracy and training efficiency while obtains significantly higher compression rates compared to the state-of-the-art.

1 Introduction

In recent years, Deep Neural Networks (DNNs) have achieved great success in various applications, such as image classification (He et al. 2016; Li et al. 2020) and object detection (He et al. 2017). However, the massive computation and memory cost hinder their deployment on lots of popular resource-constraint devices, *e.g.*, mobile platform (Han, Mao, and Dally 2016; Li, Dong, and Wang 2019). The well-known over-parameterization of DNNs shows the theoretical basis to build lightweight versions of these large models (Zhang and Stadie 2020; Yang et al. 2020b; Li et al.

2020). Therefore, DNN model compression without accuracy loss has gradually become a hot research topic.

Network pruning is one of the popular compression techniques, by removing redundant weight parameters to slim the DNN models without losing any performance, *e.g.*, fine-grained weight pruning (Guo, Yao, and Chen 2016), filter pruning (Luo, Wu, and Lin 2017), etc. One key problem for weight pruning is to determine which neuron is deletable. Recently, the lottery ticket hypothesis (Frankle and Carbin 2019) proves that the removable neurons can be directly determined by the pre-trained model only, without updating the pruning masks during finetuning (*a.k.a.*, one-shot weight pruning). Obviously, it is promising to explore whether this one-shot mechanism generalizes to other compression techniques, *e.g.*, one-shot quantization. Like one-shot pruning, one-shot quantization is expected to support finetuning the quantized model without updating the quantizers derived from the pre-trained model.

Network quantization opts to represent weight parameters using fewer bits instead of 32-bit full precision (FP32), leading to reduced memory footprint and computation cost while maintaining accuracy as much as possible. Quantization could be roughly classified as layer-wise (He and Cheng 2018; Wang et al. 2019) and channel-wise (Krishnamoorthi 2018) methods. For layer-wise quantization, the weights of each layer are quantized by a common layer-specific quantizer. Conversely, for channel-wise quantization, each channel of a layer has its own channel-specific quantizer, *i.e.*, multiple different channel-specific quantizers are required for a layer as illustrated in Figure 1a. With comparable accuracy, the fine-grained channel-wise quantization usually achieves a higher compression rate than the coarse-grained layer-wise quantization, while at the expense of extra overhead introduced (*i.e.*, channel-wise codebooks, etc.) which in turn increases the difficulty for hardware implementation (Nagel et al. 2019; Cai et al. 2020).

To further maximize the compression rate, it is desirable to simultaneously conduct pruning and quantization on the DNN models (Wang et al. 2020; Yang et al. 2020a). However, the pruning-quantization strategy is more complex than either pruning or quantization alone. In other words, it is almost impossible for manually tuning the pruning ratios and quantization codebooks at fine-grained levels. To address this issue, recent methods adopt certain

*Corresponding author: Jie Lin.

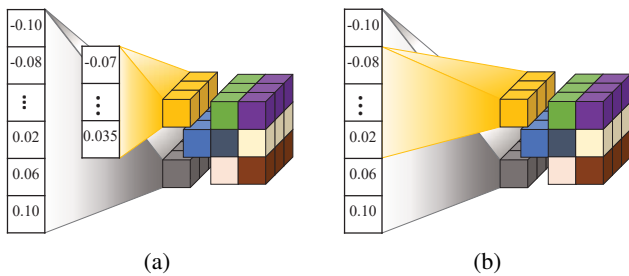


Figure 1: Difference between (a) existing channel-wise quantization and (b) the proposed unified quantization. Cubes with the same color denote a channel. In (a), each channel of a layer has its individual quantization codebook. On the contrary, in (b) all channels of a layer share a common codebook.

optimization techniques, *e.g.*, Bayesian optimization (Tung and Mori 2020), Alternating Direction Method of Multipliers (ADMM) (Yang et al. 2020a) to iteratively update the compression allocation (*i.e.*, pruning ratios and quantization codebooks) and weights of the compressed models. Although these pruning-quantization methods achieve promising results, optimizing the compression allocation parameters introduces considerable computational complexity at the finetuning stage.

To address the aforementioned problems, we proposed a novel One-shot Pruning-Quantization method (OPQ) to compress DNN models. Given a pre-trained model, a unified pruning error is formulated to calculate layer-wise pruning ratios analytically which subsequently derive layer-wise magnitude-based pruning masks. Then we compute a common channel-wise quantizer for each layer analytically by minimizing the unified quantization error of the pruned model. At the finetuning stage, the pruning mask and the channel-wise quantizer of each layer are fixed and only the weight parameters are updated in order to compensate for accuracy loss incurred by the model compression error. Thus, our method provides an analytical solution to obtain suitable pruning-quantization allocations in a one-shot manner, avoiding iterative, variable and complex optimization of the compression module during finetuning. Moreover, the proposed unified channel-wise quantization enforces all channels of each layer to share a common quantization codebook as illustrated in Figure 1. Unlike traditional channel-wise quantization methods, our unified quantization does not introduce any overheads (*i.e.*, channel-wise scales and offsets). Thus, our method embraces the benefits of both the fine-grained channel-wise quantization and the coarse-grained layer-wise quantization simultaneously.

The main contributions of this work are three-fold:

1. A One-shot Pruning-Quantization method for compression of DNNs. To our knowledge, this is the first work reveals that pre-trained model is sufficient for solving pruning and quantization simultaneously, without any iterative and complex optimization during finetuning.
2. A Unified Channel-wise Quantization method remarkably

reduces the number of channel-wise quantizers for each layer, and thus avoids the overhead brought by the traditional channel-wise quantization.

3. Extensive experiments prove the effectiveness of the one-shot pruning-quantization strategy. On AlexNet, MobileNet-V1, and ResNet-50, our method boosts the state-of-the-art in terms of accuracy and compression rate by a large margin.

2 Related Works

In this section, we briefly review the most related works from the following three aspects: pruning, quantization, and pruning-quantization methods.

2.1 Pruning Methods

Pruning aims at removing unimportant weights/neurons to slim the over-parameterized DNN models. Since fully-connected layers contain the dominant number of parameters in early stage of the classical DNN models, research works focus on removing the neurons in the fully-connected layers (Yang et al. 2015; Cheng et al. 2015). Follow-up works attempt to sparsify the weights in both convolutional and fully-connected layers (Srinivas and Babu 2015; Guo, Yao, and Chen 2016; Zhu and Gupta 2017), *e.g.*, with tensor low rank constraints, group sparsity (Zhou, Alvarez, and Porikli 2016), constrained Bayesian optimization (Chen et al. 2018), etc. Pruning methods can be roughly grouped into unstructured pruning (*e.g.*, weights) (Guo, Yao, and Chen 2016) or structured pruning (*e.g.*, filters) (Luo, Wu, and Lin 2017; He et al. 2020). The former shrinks network size by simply defining hard threshold for pruning criteria such as magnitude of weights. The latter removes the sub-tensors along a given dimension in the weight tensors with pruning criteria such as L1/L2 norm of filters. Structured pruning is compatible with existing hardware platforms, while unstructured pruning requires the support of hardware architecture with specific design. On the other side, unstructured pruning usually achieves much higher compression rate than structured pruning at comparable accuracy, which is an attractive feature for embedded systems that with extreme-low resource constraints.

2.2 Quantization Methods

Quantization enforces the DNN models to be represented by low-precision numbers instead of 32-bit full precision representation, leading to smaller memory footprint as well as lower computational cost. In recent years, numerous quantization methods are proposed to quantize the DNN models to as few bits as possible without accuracy drop, *e.g.*, uniform quantization with re-estimated statistics in batch normalization and estimation error minimization of each layer’s response (He and Cheng 2018), layer-wise quantization with reinforcement learning (Wang et al. 2019), channel-wise quantization (Banner, Nahshan, and Soudry 2019), etc. The extreme case of quantization is binary neural networks, which represent the DNN weights with binary values. For instance, Rastegari *et al.* adopted binary values

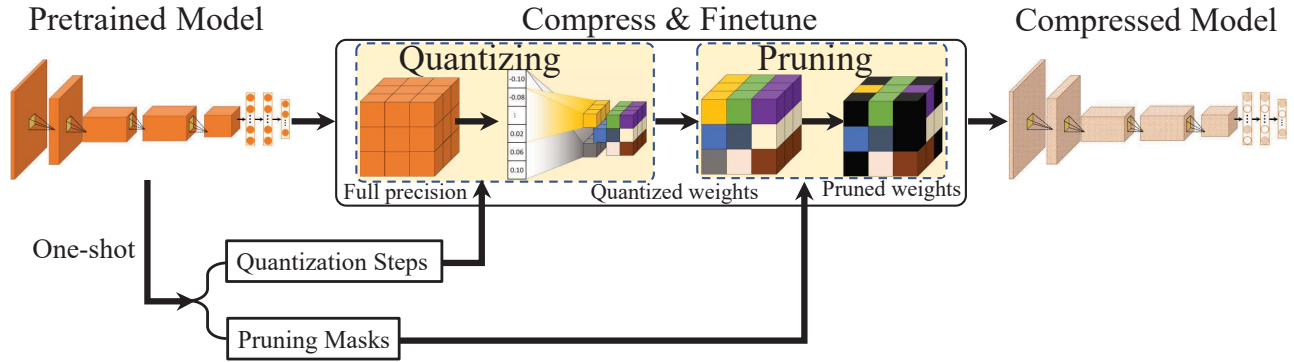


Figure 2: The pipeline of our method. Channels of each layer share the same quantizer (*i.e.*, the same codebook). Cubes in black color indicate pruned weights. Given a pre-trained model, the pruning masks $\{\mathbf{M}_i\}_{i=1}^L$ (see Section 3.2) and quantization steps $\{\Delta_i\}_i^L$ (see Section 3.3) are analytically derived in one-shot and fixed while finetuning the compressed model.

to approximate the filters of DNN, called Binary-Weight-Networks (Wu et al. 2016).

2.3 Pruning-quantization Methods

Obviously, both pruning and quantization can be simultaneously conducted to boost the compression rate. Han *et al.* proposed a three-stage compression pipeline (*i.e.*, pruning, trained quantization and Huffman coding) to reduce the storage requirement of DNN models (Han, Mao, and Dally 2016). In-parallel pruning-quantization methods are proposed to compress DNN models to smaller size without losing accuracy. Specifically, Tung *et al.* utilized Bayesian optimization to prune and quantize DNN models in parallel during fine-tuning (Tung and Mori 2020). In (Yang et al. 2020a), Yang *et al.* proposed an optimization framework based on Alternating Direction Method of Multipliers (ADMM) to jointly prune and quantize the DNNs automatically to meet target model size. For the aforementioned methods, the pruning masks and quantizers for all layers are manually set or iteratively optimized at finetuning stage. How to directly obtain the desirable compression module from the pre-trained models without iterative optimization at finetuning stage is less explored in earlier works.

3 The Proposed Method

3.1 Preliminaries

We first provide notations and definitions for neural network compression, *i.e.*, pruning and quantization. Let $\mathcal{W} = \{\mathbf{W}_i\}_{i=1}^L$ be the FP32 weight tensors of a L -layer pre-trained model, where \mathbf{W}_i is the weight parameter tensor of the i -th layer. To simplify the presentation, let W_{ij} ($j = 1, 2, \dots, N_i$) be the j -th element of \mathbf{W}_i and W_{ijk} ($k = 1, 2, \dots, N_{ij}$) be k -th element from the j -th channel of the i -th layer, where N_i is the number of weight parameters in \mathbf{W}_i and N_{ij} is the number of weight parameters from the j -th channel of i -th layer. The weights of each layer satisfy a symmetrical distribution around zero such as Laplace distribution (Ritter, Botev, and Barber 2018; Banner, Nahshan, and Soudry 2019). We define a probability density function $f_i(x)$ for the i -th layer. Different from the

prior art (Han, Mao, and Dally 2016; Yang et al. 2020a; Tung and Mori 2020), given a target pruning rate and quantization bit-rate, our method analytically solves the pruning-quantization allocations (*i.e.*, pruning masks $\{\mathbf{M}_i\}_{i=1}^L$ and quantization steps $\{\Delta_i\}_i^L$) with pre-trained weight parameters only in one-shot, meaning that the pruning masks and quantization steps are fixed while finetuning the weights of the compressed model. Given a pre-trained model, the pruning-quantization is conducted on its weight parameters \mathbf{W} by $\tilde{\mathbf{W}} = \mathbf{M} \circ (\Delta \lfloor \frac{\mathbf{W}}{\Delta} \rfloor)$, where \circ is the Hadamard product, $\lfloor \cdot \rfloor$ is the round operator, and $\Delta \lfloor \frac{\mathbf{W}}{\Delta} \rfloor$ represents the quantized weights with quantization step Δ . For finetuning the compressed model, the Straight-Through Estimator (STE) (Bengio, Léonard, and Courville 2013) is applied on $\Delta \lfloor \frac{\mathbf{W}}{\Delta} \rfloor$ to compute the gradients in the backward pass.

The pipeline of the proposed method is shown in Figure 2 and Algorithm 1. The details on solving $\{\mathbf{M}_i\}_{i=1}^L$ and $\{\Delta_i\}_i^L$ will be given in the following sections.

Algorithm 1 Optimization process of our method

Input: A pre-trained FP32 model with L layers, objective pruning rate p^* , objective quantization bitwidth B , batch size N_b , and maximum epoch number N_e .

Output: Finetuned compressed model.

- 1: Compute the pruning masks $\{\mathbf{M}_i\}_{i=1}^L$ for all layers (see Section 3.2).
 - 2: Calculate the quantization steps $\{\Delta_i\}_i^L$ for all layer (see Section 3.3).
 - 3: **for** $1, 2, \dots, N_e$ **do**
 - 4: **repeat**
 - 5: Randomly sample a minbatch from the training set.
 - 6: Compress the weights using $\{\Delta_i\}_i^L$ and $\{\mathbf{M}_i\}_{i=1}^L$ for the model.
 - 7: Forward propagate with the pruned and quantized weights, and compute the cross entropy loss.
 - 8: Update the model weights with descending their stochastic gradient.
 - 9: **until** all samples selected
 - 10: **end for**
-

3.2 Unified Layer-wise Weight Pruning

In this section, we present a general unified formulation to prune the weights \mathcal{W} of all layers given a pre-trained neural network model. The pruning problem aims to find which weights could be removed. To simplify the formulation, we reformulate the problem as finding the pruning ratios of all layers $\{p_i\}_{i=1}^L$, where p_i is the percentage of weights with small magnitude (*i.e.*, minor absolute values around zero) to be removed in the i -th layer. Specifically, it removes the weights in a symmetric range $[-\beta_i, \beta_i]$ around zero for the i -th layer like (Tung and Mori 2020), where β_i is a positive scalar value. Thus, the pruning rate of the whole model can be calculated by

$$p = \frac{1}{N} \sum_{i=1}^L \int_{-\beta_i}^{\beta_i} N_i f_i(x) dx = \frac{2}{N} \sum_{i=1}^L \int_0^{\beta_i} N_i f_i(x) dx. \quad (1)$$

Accordingly, the pruning error of the i -th layer can be formulated as follows:

$$\mathcal{L}_i^\beta = \sum_{j=1}^{N_i} (W_{ij})^2 \Big|_{|W_{ij}| \leq \beta_i} = 2 \int_0^{\beta_i} N_i x^2 f_i(x) dx. \quad (2)$$

Obviously, we aim to minimize the errors caused by weight pruning, with the objective that the pruned model is as consistent as possible with the original model. To achieve the goal, we have the following pruning objective function:

$$\begin{aligned} \beta_1^*, \dots, \beta_L^* &= \arg \min_{\beta_1, \beta_2, \dots, \beta_L} \frac{1}{N} \sum_{i=1}^L \mathcal{L}_i^\beta \\ &= \arg \min_{\beta_1, \beta_2, \dots, \beta_L} \frac{2}{N} \sum_{i=1}^L \int_0^{\beta_i} N_i x^2 f_i(x) dx. \end{aligned} \quad (3)$$

For a given objective pruning rate p^* , we could solve the minimizing problem of Equation (3) through the Lagrange multiplier. With the Lagrangian with a multiplier λ , we can obtain:

$$\begin{aligned} \mathcal{L}(\beta_1, \dots, \beta_L, \lambda) &= \sum_{i=1}^L \frac{2}{N} \int_0^{\beta_i} N_i x^2 f_i(x) dx \\ &\quad - \lambda \left(\frac{2}{N} \sum_{i=1}^L \int_0^{\beta_i} N_i f_i(x) dx - p^* \right). \end{aligned} \quad (4)$$

In order to solve for β_i , we set the partial derivative of the Lagrangian function $\mathcal{L}(\beta_1, \dots, \beta_L, \lambda)$ with respect to β_i as zero. Then, we can obtain $\beta_1^* = \beta_2^* = \dots = \beta_L^* = \sqrt{\lambda}$. Like the solution process of $\{\beta_i\}_{i=1}^L$, we can obtain the λ through setting the partial derivative of the Lagrangian function $\mathcal{L}(\beta_1, \dots, \beta_L, \lambda)$ with respect to λ as zero:

$$\begin{aligned} \frac{\partial \mathcal{L}(\beta_1, \dots, \beta_L, \lambda)}{\partial \lambda} &= \frac{2}{N} \sum_{i=1}^L \int_0^{\sqrt{\lambda}} N_i f_i(x) dx - p^* \\ &= 0. \end{aligned} \quad (5)$$

Since $f_i(x)$ could be a Laplace probability density function $\frac{1}{2\tau_i} e^{-\frac{|x|}{\tau_i}}$ (Banner, Nahshan, and Soudry 2019), the probability distribution function could be obtained by $F(x) =$

$\int_0^x f_i(y) dy = \int_0^x \frac{1}{2\tau_i} e^{-\frac{|y|}{\tau_i}} dy = -\frac{1}{2} e^{-\frac{y}{\tau_i}} \Big|_0^x$. Then, Equation (5) can be reformulated as follows:

$$\begin{aligned} \frac{\partial \mathcal{L}(\beta_1, \dots, \beta_L, \lambda)}{\partial \lambda} &\approx \frac{1}{N} \sum_{i=1}^L N_i \left(1 - e^{-\frac{\sqrt{\lambda}}{\tau_i}} \right) - p^* \\ &= 0. \end{aligned} \quad (6)$$

The well-known Levenberg-Marquardt algorithm could be used to fit $F(x)$ for the weights at each layer in order to derive the layer-wise scalar parameter τ_i . Moreover, we use Newton-Raphson method to solve Equation (6) in order to obtain λ , then $\{\beta_i\}_{i=1}^L$ could be derived with the obtained λ . $\{\beta_i\}_{i=1}^L$ can be used to calculate the pruning ratio of each layer by $p_i = \int_{-\beta_i}^{\beta_i} N_i f_i(x) dx$, and meanwhile derive the binary pruning mask $\mathbf{M}_i \in \{0, 1\}^{|\mathbf{W}_i|}$ via magnitude-based thresholding. The pruning mask \mathbf{M}_i is used to remove unimportant weights from the i -th layer by $\mathbf{M}_i \circ \mathbf{W}_i$, where \circ is the Hadamard product. The remaining unpruned weights are sparse tensors that could be stored by leveraging index differences and encoded with lower bits, one can refer to (Han, Mao, and Dally 2016) for more implementation details.

3.3 Unified Channel-wise Weight Quantization

Different from existing per-channel weight quantization (Krishnamoorthi 2018; Banner, Nahshan, and Soudry 2019), we enforce all channels of a layer to share a common scale factor and offset (*i.e.*, a common codebook), so that it does not introduce additional overheads (*i.e.*, channel-wise codebooks). To this end, we perform uniform quantization on the unpruned weights of all channels with a unified quantization step, *i.e.*, a unified codebook. That is to say, all channels of a layer share a common quantization step Δ between two adjacent quantized bins/centers. Let K_{ij} denote the number of bins assigned to the j -th channel of the i -th layer, which is the number of quantization centers required to store the unpruned weights of the channel. Denoting a positive real value α_{ij} as the maximum for the j -th channel (*i.e.*, $W_{ijk} \in [-\alpha_{ij}, \alpha_{ij}]$), the range $[-\alpha_{ij}, \alpha_{ij}]$ can be partitioned to K_{ij} equal quantization bins. Therefore, the number of quantization bins K_{ij} is established as follows:

$$K_{ij} = \left\lceil \frac{2\alpha_{ij}}{\Delta_i} \right\rceil, \quad (7)$$

where $\lceil \cdot \rceil$ is the round operator which rounds a value to the nearest integer, $\alpha_{ij} = \max \{|W_{ijk}| \mid k = 1, 2, \dots, \bar{N}_{ij}\}$, $|\cdot|$ is the absolute value operator, W_{ijk} is the k -th weight from the j -th channel of the i -th layer, \bar{N}_{ij} is the number of unpruned weights in the j -th channel of the i -th layer. Obviously, the average number of quantized bins required for the i -th layer is computed as follows:

$$K_i = \frac{1}{\bar{N}_i} \sum_{j=1}^{C_i} \bar{N}_{ij} K_{ij}, \quad (8)$$

where $\bar{N}_i = \sum_{j=1}^{\bar{N}_i} \bar{N}_{ij}$ is the number of all unpruned weights in the i -th layer. Thus, the average number of bins

for the whole model could be formulated as follows:

$$\frac{1}{\bar{N}} \sum_{i=1}^L K_i N_i \int_{\beta_i}^{+\infty} f_i(x) dx = \frac{1}{\bar{N}} \sum_{i=1}^L \sum_{j=1}^{C_i} \bar{N}_{ij} K_{ij} = 2^B, \quad (9)$$

where B is the objective number of bits required to store the unpruned weights for all layers, and $\bar{N} = \sum_{i=1}^L \bar{N}_i$ is the number of all unpruned weights in the model. With the quantization step Δ_i , each weight of the corresponding layer could be uniformly quantized as certain discrete levels. The channels from the same layer share a common quantization function denoted as:

$$Q_i(x) = \text{sgn}(x) \Delta_i \left\lfloor \frac{|x|}{\Delta_i} \right\rfloor, \quad (10)$$

where $\text{sgn}(x)$ is the sign function. Then, the mean-square-errors caused by quantization can be formulated as:

$$\mathcal{L}_q = \sum_{i=1}^L \frac{1}{\bar{N}_i} \sum_{j=1}^{N_i} M_{ij} (W_{ij} - Q_i(W_{ij}))^2. \quad (11)$$

Since $f_i(x)$ is a symmetric distribution function, we could simplify the quantization error \mathcal{L}_q in Equation (11) as:

$$\mathcal{L}_q = 2 \sum_{i=1}^L \frac{1}{\bar{N}_i} \int_{\beta_i}^{+\infty} N_i f_i(x) (x - Q_i(x))^2 dx \approx \sum_{i=1}^L \frac{\Delta_i^2}{12}, \quad (12)$$

where $\bar{N}_i = 2N_i \int_{\beta_i}^{+\infty} f_i(x) dx$ is the number of unpruned weights in the i -th layer.

To minimize \mathcal{L}_q with bit constraint, we introduce a multiplier λ (*i.e.*, a Lagrange multiplier) to enforce the bit-allocation requirement in Equation (9) as follows:

$$\begin{aligned} & \mathcal{L}(\Delta_1, \dots, \Delta_L, \lambda) \\ &= \sum_{i=1}^L \frac{1}{12} \Delta_i^2 + \lambda \left(\frac{1}{\bar{N}} \sum_{i=1}^L \sum_{j=1}^{C_i} \bar{N}_{ij} \frac{2\alpha_{ij}}{\Delta_i} - 2^B \right), \end{aligned} \quad (13)$$

where \bar{N}_{ij} is the number of unpruned connections in the j -th channel of the i -th layer. Similar to the Lagrange derivation for pruning in Section 3.2, we apply Lagrange derivation to derive Δ_i and λ as:

$$\Delta_i = \frac{1}{2^{B-1} \bar{N}} \sum_{i=1}^L \sum_{j=1}^{C_i} \bar{N}_{ij} \alpha_{ij}, \quad (14)$$

$$\lambda = \left(\frac{1}{2^{B-1} \bar{N}} \sum_{i=1}^L \sqrt[3]{\frac{\bar{N}}{12} \frac{\sum_{j=1}^{C_i} \bar{N}_{ij} \alpha_{ij}}{\sum_{j=1}^{C_i} \bar{N}_{ij} \alpha_{ij}}} \right)^3. \quad (15)$$

Combining Equations (14) and (15), we obtain the quantization quotas (*i.e.*, $\{\Delta_i\}_{i=1}^L$) for all layers, which can be used to quantize the given DNN model in the forward pass.

4 Experiments

In this section, we evaluate our method by compressing well-known convolutional neural networks, *i.e.*, AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGG-16 (Simonyan and Zisserman 2015), ResNet-50 (He et al.

2016), MobileNet-V1 (Howard et al. 2017). All experiments are performed on ImageNet (*i.e.*, ILSVRC-2012) (Deng et al. 2009), a large-scale image classification dataset consisted of 1.2M training images and 50K validation images. To demonstrate the advantage of our method, we compare our results with state-of-the-art pruning, quantization and in-parallel pruning-quantization methods. Finally, we conduct error analysis to verify the validity of our method.

4.1 Implementation Details

The proposed method is implemented by PyTorch. We set the batch size as 256 for all models at finetuning stage. The SGD optimizer is utilized to finetune the compressed models with the momentum (= 0.9), weight decay (= 10^{-4}), and learning rate (= 0.005). Since pruning has a greater impact on performance than quantization, we adopt two-stage strategy to finetune the compressed networks. First, we finetune the pruned networks without quantization till they recover the performance of the original uncompressed models. Second, pruning and quantization are simultaneously applied at the finetuning stage.

4.2 Comparisons with State-of-the-Art

We performed extensive experiments to evaluate the effectiveness of the proposed method by comparing with 12 state-of-the-art methods, including 6 pruning methods (*i.e.*, *i.e.*, Data-Free Pruning (Srinivas and Babu 2015), Adaptive Fastfood 32 (Yang et al. 2015), Less Is More (Zhou, Alvarez, and Porikli 2016), Dynamic Network Surgery (Guo, Yao, and Chen 2016), Circulant CNN (Cheng et al. 2015), and Constraint-Aware (Chen et al. 2018)), 3 quantization methods (*i.e.*, Q-CNN (Wu et al. 2016), Binary-Weight-Networks (Rastegari et al. 2016), and ReNorm (He and Cheng 2018)), 3 pruning-quantization methods (*i.e.*, Deep Compression (Han, Mao, and Dally 2016), CLIP-Q (Tung and Mori 2020), and ANNC (Yang et al. 2020a)).

AlexNet on ImageNet For AlexNet, we compare our method with pruning only (*i.e.*, Data-Free Pruning (Srinivas and Babu 2015), Adaptive Fastfood 32 (Yang et al. 2015), Less Is More (Zhou, Alvarez, and Porikli 2016), Dynamic Network Surgery (Guo, Yao, and Chen 2016), Circulant CNN (Cheng et al. 2015), and Constraint-Aware (Chen et al. 2018)), quantization only (*i.e.*, Q-CNN (Wu et al. 2016) and Binary-Weight-Networks (Rastegari et al. 2016)) and pruning-quantization (*i.e.*, Deep Compression (Han, Mao, and Dally 2016), CLIP-Q (Tung and Mori 2020), and ANNC (Yang et al. 2020a)) methods. Unlike our one-shot pruning-quantization framework, all the compared methods gradually learn their pruning masks and/or quantizers during fine-tuning process. Our method compresses the pre-trained uncompressed AlexNet which achieves 56.62% top-1 accuracy and 79.06% top-5 accuracy. Experimental results are shown in Table 1. From the table, we observe that our method compresses AlexNet to be $138.96\times$ smaller while preserving the accuracy of the uncompressed AlexNet on ImageNet. Our method could even achieve 0.46% top-1 and 1.20% top-5 accuracy improvements. Compared to ANNC with 118x compression rate, our method performs slightly

Method	Top-1 (%)	Top-5 (%)	Prune Rate (%)	Bit	Rate
Data-Free Pruning (Srinivas and Babu 2015)	55.60 (2.24↓)	-	36.24	32	1.57×
Adaptive Fastfood 32 (Yang et al. 2015)	58.10 (0.69↑)	-	44.12	32	1.79×
Less Is More (Zhou, Alvarez, and Porikli 2016)	53.86 (0.57↓)	-	76.76	32	4.30×
Dynamic Network Surgery (Guo, Yao, and Chen 2016)	56.91 (0.3↑)	80.01 (-)	94.3	32	17.54×
Circulant CNN (Cheng et al. 2015)	56.8 (0.4↓)	82.2 (0.7↓)	95.45	32	18.38×
Constraint-Aware (Chen et al. 2018)	54.84 (2.57↓)	-	95.13	32	20.53×
Q-CNN (Wu et al. 2016)	56.31 (0.99↓)	79.70 (0.60↓)	-	1.57	20.26×
Binary-Weight-Networks (Rastegari et al. 2016)	56.8 (0.2↑)	79.4 (0.8↓)	-	1	32×
Deep Compression (Han, Mao, and Dally 2016)	57.22 (0.00↑)	80.30 (0.03↑)	89	5.4	6.66×
CLIP-Q (Tung and Mori 2020)	57.9 (0.7↑)	-	91.96	3.34	119.09×
ANNC (Yang et al. 2020a)	57.52 (1.00 ↑)	80.22 (0.03↑)	92.6	3.7	118×
Ours	57.09 (0.46↑)	80.25 (1.20 ↑)	92.30	2.99	138.96 ×

Table 1: AlexNet on ImageNet.

Method	Top-1 (%)	Top-5 (%)	Prune Rate (%)	Bit	Rate
ThiNet-GAP (Luo, Wu, and Lin 2017)	67.34 (1.0↓)	87.92 (0.52↓)	94.00	32	16.63×
Q-CNN (Wu et al. 2016)	68.11 (3.04↓)	88.89 (1.06↓)	-	1.35	23.68×
Deep Compression (Han, Mao, and Dally 2016)	68.83 (0.33↑)	89.09 (0.41 ↑)	92.5	6.4	66.67×
CLIP-Q (Tung and Mori 2020)	69.2 (0.7 ↑)	-	94.20	3.06	180.47×
Ours	71.39 (0.24↓)	90.28 (0.09↓)	94.41	2.92	195.87 ×

Table 2: VGG-16 on ImageNet.

lower in Top-1 and slightly higher in Top-5, but with over 3x less finetuning time (160h vs 528h).

VGG-16 on ImageNet For VGG-16, we compare our method with pruning only (*i.e.*, ThiNet-GAP (Luo, Wu, and Lin 2017)), quantization only (*i.e.*, Q-CNN (Wu et al. 2016)) and pruning-quantization (*i.e.*, Deep Compression (Han, Mao, and Dally 2016), and CLIP-Q (Tung and Mori 2020)) methods. Our method compresses the pre-trained uncompressed VGG-16 of PyTorch which achieves 71.63% top-1 and 90.37% top-5 accuracy. Experimental results are shown in Table 2, one can see that our method can compress VGG-16 to be 195.87× smaller with only negligible accuracy loss comparing with the uncompressed model. Compared to other approaches, our method still achieves the best accuracy with the highest compression rate.

MobileNet-V1 on ImageNet For MobileNet-V1, we compare our method with pruning only (*i.e.*, To Prune or Not To Prune (Zhu and Gupta 2017)), quantization only (*i.e.*, Deep Compression (Han, Mao, and Dally 2016), ReNorm (He and Cheng 2018), and HAQ (Wang et al. 2019)) and pruning-quantization (*i.e.*, CLIP-Q (Tung and Mori 2020) and ANNC (Yang et al. 2020a)) methods. A pre-trained uncompressed MobileNet-V1 model, which achieves 70.28% top-1 and 89.43% top-5, is adopted for pruning and quantization. Table 3 shows the experimental results comparing with the aforementioned compression methods. From the table, we can see that our method is superior to all the other methods with better accuracy and higher compression rate. Specifically, our method achieves a 0.55% top-1 accuracy improvement (*i.e.*, 70.83%) with compressing rate 23.26×, while CLIP-Q (Tung and Mori 2020) with no improvement in top-1 accuracy at a much lower compression

rate 13.19×. Our method also outperforms the most recent in-parallel pruning-quantization method ANNC (Yang et al. 2020a) by a large margin in terms of accuracy improvement and compression rate.

ResNet-50 on ImageNet For ResNet-50, we compare our method with pruning only (*i.e.*, ThiNet (Luo, Wu, and Lin 2017)), quantization only (*i.e.*, Deep Compression (Han, Mao, and Dally 2016), HAQ (Wang et al. 2019), and ACIQ (Banner, Nahshan, and Soudry 2019)) and pruning-quantization (*i.e.*, CLIP-Q (Tung and Mori 2020)) methods. Our method compresses a pre-trained uncompressed ResNet-50 model which reports 76.01% top-1 and 92.93% top-5 on ImageNet. Table 4 shows the experimental results of our method comparing with the aforementioned state-of-the-art methods. From the table, we can see that our method could compress ResNet-50 to be 38.03× smaller with 0.40% top-1 and 0.11% top-5 improvements. Comparing to all the other methods, our method achieves the best accuracy (76.41% top-1 and 93.04% top-5) with the highest compression rate, while without gradually learning the pruning masks and quantizers at finetuning stage.

Summary In conclusion, extensive experimental results on ImageNet with various neural network architectures show that in-parallel pruning-quantization boosts the model compression rate by a large margin without incurring accuracy loss, compared to pruning only and quantization only approaches. More importantly, our One-shot Pruning-Quantization consistently outperforms the other in-parallel pruning-quantization approaches which gradually optimize the pruning masks and quantizers at finetuning stage, suggesting that pre-trained model is sufficient for determining the compression module prior to finetuning, and the com-

Method	Top-1 (%)	Top-5 (%)	Prune Rate (%)	Bit	Rate
To Prune or Not To Prune (Zhu and Gupta 2017)	69.5 (1.1↓)	89.5 (0.0↑)	50	32	2×
Deep Compression (Han, Mao, and Dally 2016)	65.93 (4.97↓)	86.85 (3.05↓)	-	3	10.67×
ReNorm (He and Cheng 2018)	65.93 (9.75↓)	83.48 (6.37↓)	-	4	8×
HAQ (Wang et al. 2019)	67.66 (3.24↓)	88.21 (1.69↓)	-	3	10.67×
CLIP-Q (Tung and Mori 2020)	70.3 (0.0↑)	-	47.36	4.61	13.19×
ANNC (Yang et al. 2020a)	69.71 (1.19↓)	89.14 (0.76↓)	-	3	10.67×
ANNC (Yang et al. 2020a)	66.49 (4.41↓)	87.29 (2.61↓)	58	2.8	26.7×
Ours	70.83 (0.55↑)	89.70 (0.27↑)	57.78	3.26	23.26×
Ours	70.24 (0.04↓)	89.30 (0.13↓)	67.66	3.08	32.15×

Table 3: MobileNet-V1 on ImageNet.

Method	Top-1 (%)	Top-5 (%)	Prune Rate (%)	Bit	Rate
ThiNet (Luo, Wu, and Lin 2017)	71.01 (1.87↓)	90.02 (1.12↓)	51.76	32	2.07×
Deep Compression (Han, Mao, and Dally 2016)	76.15 (0.00↑)	92.88 (0.02↑)	-	4	8×
HAQ (Wang et al. 2019)	76.14 (0.01↓)	92.89 (0.03↑)	-	4	8×
ACIQ (Banner, Nahshan, and Soudry 2019)	75.3 (0.8↓)	-	-	4	8×
CLIP-Q (Tung and Mori 2020)	73.7 (0.6 ↑)	-	69.38	3.28	31.81×
Ours	76.41 (0.40↑)	93.04 (0.11↑)	74.14	3.25	38.03×

Table 4: ResNet-50 on ImageNet.

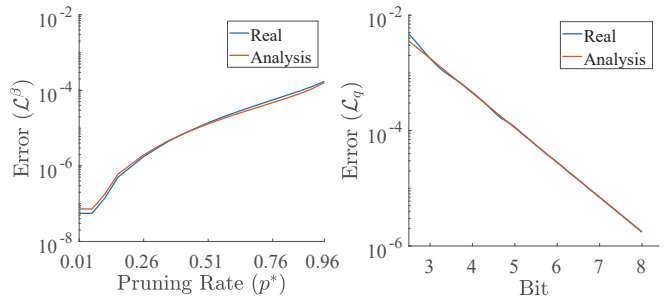
plex and iterative optimization of the compression module may not be necessary at finetuning stage.

4.3 Error Analysis

In this section, we investigate the relationships between the analytic errors and the real ones computed on ResNet-50. From Figure 3a, we can see that the Laplace probability density function approximates the real pruning error well, demonstrating the validity of Equation (6). From Figure 3b, we observe that the approximation error $\sum_{i=1}^L \frac{\Delta_i^2}{12}$ is in a good agreement with the real quantization error of Equation (11), which verifies the effectiveness of the proposed approximation. Therefore, the proposed method is able to capture the prior knowledge of the pre-trained models effectively, which in turn guides the design of our analytical solution for one-shot pruning-quantization.

5 Conclusion

In this paper, we propose a novel One-shot Pruning-Quantization method (OPQ) to compress DNN models. Our method has addressed two challenging problems in network compression. First, different from the prior art, OPQ is a one-shot compression method without manual tuning or iterative optimization of the compression strategy (*i.e.*, pruning mask and quantization codebook of each layer). The proposed method analytically computes the compression allocation of each layer with pre-trained weight parameters only. Second, we propose a unified channel-wise pruning to enforce all channels of each layer to share a common codebook, which avoids the overheads brought by the traditional channel-wise quantization. Experiments show that our method achieves superior results comparing to the state-of-the-art. For AlexNet, MobileNet-V1, ResNet-



(a) Pruning Error Analysis. (b) Quantization Error Analysis.

Figure 3: Error analysis on ResNet-50. (a) illustrates the difference between the analyzed and the real pruning errors. “Real” denotes the real pruning error, *i.e.*, $\frac{1}{N} \sum_{i=1}^L \sum_{j=1}^{N_i} (W_{ij})^2 |_{|W_{ij}| \leq \beta_i}$ in Equation (3). “Analysis” denotes the approximated pruning error using Laplace probability density functions, *i.e.*, $\frac{2}{N} \sum_{i=1}^L \int_0^{\beta_i} N_i x^2 f_i(x) dx$, where $f_i(x) = \frac{1}{2\tau_i} e^{-\frac{|x|}{\tau_i}}$. (b) shows the difference between the analyzed and the real quantization errors. “Real” presents the real quantization error, *i.e.*, Equation (11). “Analysis” denotes the approximated quantization error, *i.e.*, $\sum_{i=1}^L \frac{\Delta_i^2}{12}$ in Equation (12).

50, our method could improve accuracy even at very high compression rates. For future work, we will explore how to further compress DNN models, and implement our method with custom hardware architecture in order to validate the inference efficiency of the compressed models on practical hardware platforms.

Acknowledgments

This work is supported in part by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funds (Project No.A1892b0026 and Project No.A19E3b0099); the Fundamental Research Funds for the Central Universities under Grant YJ201949; and the NFSC under Grant 61971296, U19A2078, and 61836011.

Broader Impact Statement

Neural network compression is a hot topic which has drawn tremendous attention from both academia and industry. The main objective is to reduce memory footprint, computation cost and power consumption of the training and/or inference of deep neural networks, and thus facilitate their deployments on resource-constrained hardware platforms, *e.g.*, smartphones, for a wide range of applications including computer vision, speech and audio, natural language processing, recommender systems, etc.

In the paper, we mainly focus on exploring the one-shot model compression mechanism, in particular in-parallel pruning-quantization, for model compression at extremely high compression rate without incurring loss in accuracy. Our One-shot Pruning-Quantization method (OPQ) largely reduces the complexity for optimizing the compression module and also provides the community new insight into how can we perform efficient and effective model compression from alternative perspective. Although OPQ achieves promising performance in network compression, we should also care about the potential negative impacts including 1) the compression bias caused by OPQ because of unusual weight distribution, too lower objective compression rate, etc. Usually, this requires domain experts to manually evaluate them. 2) the robustness of the compressed models for decision making, especially in health care, autonomous vehicles, aviation, fintech, etc. Question to be answered could be is the compressed model vulnerable to adversarial attacks because of the introduction of pruning and quantization into the model? We encourage further study to understand and mitigate the biases and risks potentially brought by network compression.

References

Banner, R.; Nahshan, Y.; and Soudry, D. 2019. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Advances in Neural Information Processing Systems*, 7948–7956.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Cai, Y.; Yao, Z.; Dong, Z.; Gholami, A.; Mahoney, M. W.; and Keutzer, K. 2020. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13169–13178.

Chen, C.; Tung, F.; Vedula, N.; and Mori, G. 2018. Constraint-aware deep neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 400–415.

Cheng, Y.; Yu, F. X.; Feris, R. S.; Kumar, S.; Choudhary, A.; and Chang, S.-F. 2015. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision*, 2857–2865.

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

Frankle, J.; and Carbin, M. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Guo, Y.; Yao, A.; and Chen, Y. 2016. Dynamic network surgery for efficient dnns. In *Advances in neural information processing systems*, 1379–1387.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, X.; and Cheng, J. 2018. Learning compression from limited unlabeled data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 752–769.

He, Y.; Ding, Y.; Liu, P.; Zhu, L.; Zhang, H.; and Yang, Y. 2020. Learning Filter Pruning Criteria for Deep Convolutional Neural Networks Acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009–2018.

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

Krishnamoorthi, R. 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Li, Y.; Dong, M.; Wang, Y.; and Xu, C. 2020. Neural architecture search in a proxy validation loss landscape. In *International Conference on Machine Learning*, 5853–5862. PMLR.

Li, Y.; Dong, X.; and Wang, W. 2019. Additive Powers-of-Two Quantization: An Efficient Non-uniform Discretization for Neural Networks. In *International Conference on Learning Representations*.

- Luo, J.-H.; Wu, J.; and Lin, W. 2017. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, 5058–5066.
- Nagel, M.; Baalen, M. v.; Blankevoort, T.; and Welling, M. 2019. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE International Conference on Computer Vision*, 1325–1334.
- Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. XNOR-Net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, 525–542. Springer.
- Ritter, H.; Botev, A.; and Barber, D. 2018. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations*.
- Simonyan, K.; and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.
- Srinivas, S.; and Babu, R. V. 2015. Data-free parameter pruning for deep neural networks. In *British Machine Vision Conference (BMVC)*.
- Tung, F.; and Mori, G. 2020. Deep Neural Network Compression by In-Parallel Pruning-Quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42(3): 568–579.
- Wang, K.; Liu, Z.; Lin, Y.; Lin, J.; and Han, S. 2019. HAQ: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8612–8620.
- Wang, T.; Wang, K.; Cai, H.; Lin, J.; Liu, Z.; Wang, H.; Lin, Y.; and Han, S. 2020. APQ: Joint Search for Network Architecture, Pruning and Quantization Policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2078–2087.
- Wu, J.; Leng, C.; Wang, Y.; Hu, Q.; and Cheng, J. 2016. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4820–4828.
- Yang, H.; Gui, S.; Zhu, Y.; and Liu, J. 2020a. Automatic Neural Network Compression by Sparsity-Quantization Joint Learning: A Constrained Optimization-based Approach. In *IEEE Conference on Computer Vision and Pattern Recognition*.
- Yang, Z.; Moczulski, M.; Denil, M.; de Freitas, N.; Smola, A.; Song, L.; and Wang, Z. 2015. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, 1476–1483.
- Yang, Z.; Wang, Y.; Chen, X.; Shi, B.; Xu, C.; Xu, C.; Tian, Q.; and Xu, C. 2020b. Cars: Continuous evolution for efficient neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1829–1838.
- Zhang, M. S.; and Stadie, B. 2020. One-Shot Pruning of Recurrent Neural Networks by Jacobian Spectrum Evaluation. In *8th International Conference on Learning Representations*.
- Zhou, H.; Alvarez, J. M.; and Porikli, F. 2016. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, 662–677. Springer.
- Zhu, M.; and Gupta, S. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.