Towards Accurate Post-training Network Quantization via Bit-Split and Stitching

Peisong Wang [1] Qiang Chen [1] Xiangyu He [1] Jian Cheng [1]

Abstract

Network quantization is essential for deploying deep models to IoT devices due to its high efficiency. Most existing quantization approaches rely on the full training datasets and the time-consuming fine-tuning to retain accuracy. Post-training quantization does not have these problems, however, it has mainly been shown effective for 8-bit quantization due to the simple optimization strategy. In this paper, we propose a Bit-Split and Stitching framework (Bit-split) for lower-bit post-training quantization with minimal accuracy degradation. The proposed framework is validated on a variety of computer vision tasks, including image classification, object detection, instance segmentation, with various network architectures. Specifically, Bit-split can achieve near-original model performance even when quantizing FP32 models to INT3 without fine-tuning.

1. Introduction

Deep neural networks (DNNs) have been demonstrated to be effective in a wide range of computer vision tasks, including image recognition (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Chatfield et al., 2014; He et al., 2016), object detection (Girshick et al., 2014; Ren et al., 2015; Liu et al., 2016), segmentation (Long et al., 2015; Chen et al., 2019) and so on. At the same time, the high complexity including the huge parameter size and computing operations as well as the power consumption have hindered the deployment of deep networks to real-world applications. Thus network compression (Han et al., 2015; Wu et al., 2016; Wang & Cheng, 2016; Romero et al., 2014) has drawn great attention of researchers. Among these compression techniques, network quantization is widely used no matter on special hardware like TPU or general hardware like CPU and GPU. By turning the floating-point values within the networks to low-bit integers, the complex floating-point operations can be replaced by more efficient integer operations.

Despite its efficiency, training low-bit neural networks is nontrivial. In (Krishnamoorthi, 2018), the authors proposed two basic network quantization paradigms, i.e., quantization-aware training and post-training quantization. Most of current quantization approaches belong to the former, where low-bit networks are trained with quantization operations inserted. Quantization-aware training can achieve higher accuracy because the learned weights could be adjusted during training to fit the quantization operations. On the other hand, quantization-aware training has several drawbacks. It relies on the full training data and large computing resources like GPUs. Moreover, the tedious and time-consuming retraining procedure has posed high requirements in domain knowledge and experience for the users, which is hindering the application of quantization techniques.

By contrast, post-training quantization has many desirable properties. It does not need the training dataset, except for a very small amount of data for calibration, thus no privacy or data transmission problems will be caused. Moreover, post-training quantization is network architecture free, back-propagation free, and does not require domain knowledge or any optimization tricks. It is for this reason that post-training quantization is supported and preferable to accelerate the inference of many libraries and devices, such as the TensorRT (Migacz, 2017) on GPU, TF-Lite (Krishnamoorthi, 2018) on TPU and the SNPE on Qualcomm devices.

Despite the advantages of post-training quantization, it has the problem of significant accuracy degradation, especially when both activations and weights are quantized into very low-bit integers. Thus for current post-training quantization in TensorRT and TF-Lite as well as the SNPE, only 8-bit quantization is supported for the weights and activations. However, for hardware accelerators, every bit saving could result in significant resource reduction, including storage, computing units and more importantly the energy consumption (Horowitz, 2014). To this end, ZeroQ (Cai et al., 2020) utilizes knowledge distillation and mix-precision quantization, i.e., allowing different channels and filters to be quan-

[1]NLPR & AIRIA, Institute of Automation, Chinese Academy of Sciences. Correspondence to: Jian Cheng <jcheng@nlpr.ia.ac.cn>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

tized into different bit-widths using separate quantization scales. Besides the per-channel activation quantization and mix-precision quantization, ACIQ (Banner et al., 2019) also exploits run-time dynamic quantization, which means the quantization parameters (such as the clipping value or quantization scales) are determined at inference time. These techniques substantially improve the quantization performance, however, at the cost of remarkable complexity for inference architecture design. Thus naive post-training quantization with high accuracy still remains as an open challenge.

In this paper, we present Bit-Split and Stitching (Bit-split), a novel post-training quantization framework that is effective for very low-bit quantization. The motivation is to split integers into multiple bits, then optimize each bit, and finally stitch all bits back to integers (Fig. 1). We show that Bit-split can achieve near-original model performance even when quantizing FP32 models to INT3 without fine-tuning. Our main contributions are summarized as follows:

- We introduce a novel Bit-Split and Stitching (Bit-split) framework for post-training network quantization. Bit-split is effective and hyper-parameter free, which can be readily implemented and integrated into current network quantization libraries.
- Based on the Bit-split framework, we further propose Error Compensated Activation Quantization (ECAQ) method, which could lower the quantization error for activations.
- We evaluate our method on classification, object detection and instance segmentation using various neural networks, showing that Bit-split could achieve close to full-precision accuracy even for 3-bit quantization, setting new state-of-the-art post-training quantization results.

2. Background and Notation

A typical convolutional neural network consists of multiple convolutional and fully-connected layers, which can be formulated as a matrix multiplication followed by a nonlinear activation function Φ as follows:

$$\begin{aligned} Y &= W^T X \\ Z &= \Phi(Y) \end{aligned} \tag{1}$$

Here, W represents the learnable parameters for convolutional or fully-connected layers. Specifically for convolutional layers, $W^T \in \mathbb{R}^{N\times(C\cdot K_h\cdot K_w)}$, where N, C, K_h, K_w refer to output channels, input channels, kernel height and kernel width, respectively. X and Y stand for the input feature maps and output feature maps, i.e., $X \in \mathbb{R}^{(C\cdot K_h\cdot K_w)\times(F_h\cdot F_w)}$ and $Y \in \mathbb{R}^{N\times(F_h\cdot F_w)}$ where F_h and F_w are the height and width of input/output feature maps.

2.1. Network Quantization

Eq. (1) illustrates that most of the computation and storage reside in the matrix multiplication of convolutions. The purpose of network quantization is to map the floating-point values of W and X into a finite set with discrete elements, which can be encoded by low-bit numbers. Network quantization has many advantages, therefore has been widely studied, in which many design choices should be considered from different viewpoints.

Uniform quantization vs. non-uniform quantization: In non-uniform quantization, there are no restrictions for the discrete values within the finite quantization set. While in uniform quantization, the floating-point real numbers are linearly quantized into low-bit integers, with the same step value between two successive quantization integers. Therefore, uniform quantization can turn the floating-point operations into integer operations, which are more efficient than lookup tables used in non-uniform quantization.

Per-layer quantization vs. per-channel quantization: Quantization can be conducted under different granularities. In (Krishnamoorthi, 2018), the authors propose per-layer quantization and per-channel quantization. Per-layer quantization adopts a single quantizer (a quantization set) for an entire tensor W or X. While per-channel quantization utilizes a separate quantizer for each channel (for activation quantization) or for each kernel (for weight quantization). However, per-channel quantization for activations could complicate the convolution operations.

Unified quantization vs. mixed precision quantization: In unified quantization, we usually specify the bit-width for each layer or even for the whole network. There are many works also explore mixed precision quantization, where each channel or kernel can be quantized with different bit-widths. Mixed-precision quantization can lower accuracy loss caused by quantization, however, at the cost of more complicated quantization process as well as more complicated computing architecture at inference time.

2.2. Problems of Current Post-training Quantization

Low-bit network quantization turns the weights W as well as activations X of convolutions in Eq. (1) into integer numbers. The quantization operation could introduce much noise, resulting in dramatic accuracy loss compared with the original full-precision models. Especially for post-training quantization where no fine-tuning is allowed, the quantization error could not be compensated by training with respect to the training target.

Because no fine-tuning is allowed, most post-training approaches aim at finding a better criterion to minimize the '*distance*' between the pretrained model and the quantized one. For example, TF-Lite utilizes the maximum and minimum values of a pretrained model to determine the quantizer. While TensorRT tries to minimize the the KL divergence between the quantized distribution and the original distribution to determine the quantization parameters. These criteria are straightforward and work well for 8-bit quantization. However, the quantization error of these simple criteria could be catastrophic for lower-bit quantization.

Beside the suboptimal quantization criteria, the discrete optimization problem is another obstacle. In network quantization, the quantized parameters and activations to be optimized are fixed-point integers, which are hard to solve due to the discrete nature.

In this paper, instead of seeking an approximate criterion, we treat the network quantization as an optimization problem. More specifically, we learn a fixed-point mapping from input to the output for each convolution. Then for the optimization problem, we turn the low-bit discrete optimization into multiple bit optimization problems, which can be solved efficiently. We will introduce our proposed method in detail in Sec. 3.

2.3. Related Work

Network quantization has become a hot topic in deep learning community. In (Krishnamoorthi, 2018), the authors proposed two basic network quantization paradigms, i.e., quantization-aware training and post-training quantization. We will review current quantization works from these two paradigms.

Quantization-aware Training: In quantization-aware training, the low-bit networks are trained with the quantization operations inserted. Most of current low-bit quantization approaches belong to this quantization scheme. Quantization-aware training can result in higher accuracy because the learned weights could be adjusted during training to fit the quantization operations. Besides (Krishnamoorthi, 2018), many quantization-aware training techniques and tricks are investigated for higher accuracy, including scaling factor selection (Lin et al., 2016), multi-step quantization (Zhou et al., 2017; Wang et al., 2018), stochastic quantization (Gupta et al., 2015), data representation (Miyashita et al., 2016), multi-bit quantization (Lin et al., 2017b; Zhu et al., 2019; Li et al., 2017; Tang et al., 2017), fixed-point factorization (Wang & Cheng, 2017; Hu et al., 2018a), learnable quantization (Faraone et al., 2018; Wang et al., 2020), etc.

Post-training Quantization: Post-training quantization has recently drawn much attention. (Banner et al., 2019) proposes a run-time analytical clipping approach for integer quantization, which is further improved by per-channel bit allocation. The authors of (Nahshan et al., 2019) study post-training quantization from the viewpoint of loss landscape. Mixed-precision post-training quantization is also studied in (Cai et al., 2020). Data-free quantization for MobileNet is studied in (Nagel et al., 2019).

In this paper, we aim at the simplest post-training quantization scheme, i.e., we explore uniform quantization with unified bit-width for all layers. Moreover, we use per-channel weight quantization and per-layer activation quantization, which allows for efficient convolution implementation.

3. Method

In this section, we will introduce our proposed approach in detail. First in Sec. 3.1, we propose a Bit-Split and Stitching method for weight quantization, which learns the low-bit mapping between the input and output of convolutions. Then in Sec. 3.2, an efficient Error Compensated Activation Quantization method is introduced.

3.1. Bit-Split and Stitching

In this section, we present a novel Bit-Split and Stitching (Bit-split) method for low-bit weight quantization without retraining. To this end, we first formulate the network quantization problem as a low-bit mapping between the input and output of convolutions. More specifically, each convolution kernel w_k maps the input X to an output feature map y_k for $k = 0, \cdots N$, where N represents the number of kernels for the convolution. In the following description, we discard the index k for simplicity. Thus for M-bit weight quantization, our target is to learn low-bit kernel q to map the input X to an output feature map y as follows:

$$\underset{\alpha,q}{\text{minimize}} \parallel y - \alpha q^T X \parallel_F^2, \tag{2}$$

where q is the M-bit kernel to be learned. Note that for each convolutional kernel q, we introduce a floating-point scale factor α like in previous low-bit quantization methods (Lebedev et al., 2014; Hu et al., 2018b; Cai et al., 2017).

The optimization of Eq. (2) is non-trivial due to the M-bit constraint of q. Note that even though there are finite states for each element of q, however, the combined solution space is too large making exhaustive search impractical. In this section, we propose a novel Bit-Split and Stitching (Bit-split) method for the optimization problem of Eq. (2).

Bit-Split In Bit-split stage, we split the M-bit constraint of q into $(M-1)$ ternary optimization problems as follows:

$$\underset{\alpha,\{q_1,\cdots,q_{M-1}\}}{\text{minimize}} \parallel y - \alpha(2^0 q_1^T + \cdots + 2^{M-2} q_{M-1}^T)X \parallel_F^2,$$
$$s.t. \;\; q_m \in \{-1, 0, +1\}^{(C \cdot K_h \cdot K_w)} \text{ for } m = 1, \cdots, M-1 \tag{3}$$

where q_m denotes the m-th bit of q (from right to left), with the M-th bit (i.e., the left most bit) stands for the sign. Note

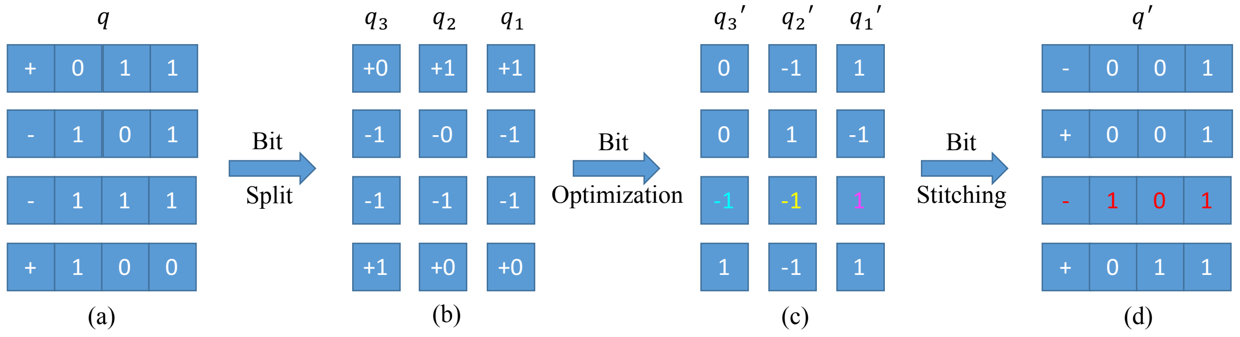

Figure 1. An illustration of Bit-Split and Stitching (Bit-split) framework for 4-bit weight quantization. In the first step of bit-split stage, each 4-bit value is split into 3 ternary values, which can be optimized separately in the second bit-optimization stage. The third stage stitching optimized bits back into integers, taking the third value for example, $2^0 \cdot \mathbf{1} + 2^1 \cdot (\mathbf{-1}) + 2^2 \cdot (\mathbf{-1}) = -5 =$ **-101**b.

that all $(M-1)$ bits share the same scale factor α. Moreover, each bit of the M-bit integer has its own implicit base, i.e., the m-th bit has an implicit base of 2^{m-1}.

The first step of Fig. 1 illustrates the bit-split operation, which take the 4-bit quantization for example. For 4-bit integers, there are one bit for sign and 3 bits that consist of 0 or 1. Thus, the 4-bit integer can be split into 3 ternary values (i.e., -1, 0 and 1, which takes the sign into consideration), which can be optimized separately. We will show how to optimize these split bits in the following Bit-Optimization section.

Bit-Optimization There are M elements that should be optimized in Eq. (3), i.e., the scale factor α and $M-1$ bits q_m for $m = 1, \cdots, M-1$. We utilize an iterative optimization procedure.

The optimization of α: From Eq. (2), we can see that α is a floating-point scalar, which can be easily solved given the quantized value q as follows:

$$\alpha = \frac{y^T X^T q}{q^T X X^T q} \tag{4}$$

The optimization of q_m: To solve Eq. (3) given the scale factor α and all other $M-2$ bits fixed, we can get the following optimization problem:

$$\begin{aligned} &\underset{q_m}{\text{minimize}} \parallel y_m - \alpha_m q_m^T X \parallel_F^2, \\ &s.t. \;\; q_m \in \{-1, 0, +1\}^{(C \cdot K_h \cdot K_w)} \end{aligned} \tag{5}$$

where y_m and α_m are independent of q_m:

$$\begin{cases} y_m = y - \alpha \sum_{i \neq m} 2^{m-1} q_i^T X, \\ \alpha_m = \alpha 2^{m-2} \end{cases} \tag{6}$$

By expanding Eq. (5) and denoting $A = \alpha_m^2 X X^T$ and $s = 2\alpha_m X y_m$, we could minimize the following equation:

$$\begin{aligned} J(q_m) &= y_m^T y_m - 2\alpha_m y_m^T X^T q_m + \alpha_m^2 q_m^T X X^T q_m \\ &= q_m^T A q_m + s^T q_m + const. \end{aligned} \tag{7}$$

The quadratic optimization problem of Eq. (7) is hard to solve. Here we utilize an iterative optimization procedure, i.e., to optimize each element of q_m with the rest elements fixed. In this way, the k-th element of q_m is as follows:

$$q_m^{(k)} = \begin{cases} -sign(r_{\bar{k}}) & \text{if } r_{\bar{k}} > A_{kk} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where $r_{\bar{k}} = s_k + 2 \sum_{i \neq k} A_{ki} q_m^{(i)}$.

Through iterative optimization from Eq. (4) to Eq. (8), we can get an approximate solution to Eq. (2).

Bit-Stitching Note that during bit-split stage, all bits corresponding to the same integer share the same sign, i.e., each integer after bit-split consists of 0/1's or 0/ -1's (Fig. 1(b)). However, after bit-optimization, this property does not hold any longer. In other words, an integer after bit-optimization may consist of 0/1/-1's (Fig. 1(c)), where no unified sign bit can be extracted.

To stitch all bits after bit-optimization back into integers, we find that we can simply add all bits together. More specifically, given the optimized bits $q_1^{'}, \cdots, q_{M-1}^{'}$, the new integer values can be obtained through the following equation:

$$q^{'} = 2^0 q_1^{'} + \cdots + 2^{M-2} q_{M-1}^{'} \tag{9}$$

Note that duiring bit-stitching, each of the $M-1$ bits also has its own implicit base, i.e., the m-th bit has an implicit base of 2^{m-1}. Fig. 1(c) to Fig. 1(d) illustrates the bit-stitching procedure.

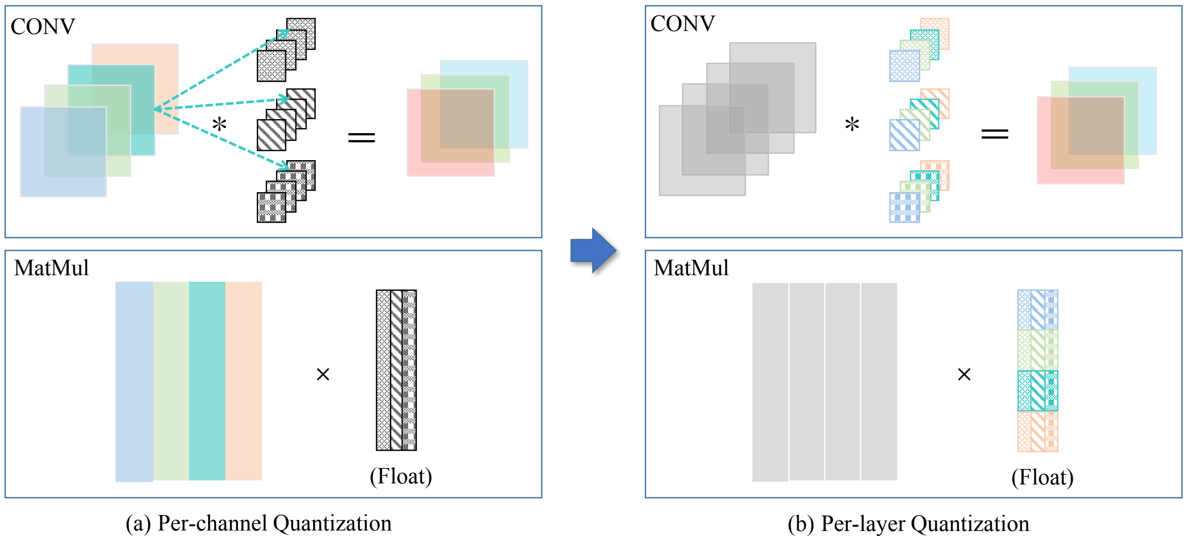

Figure 2. (a) Illustration of per-channel activation quantization. Each input channel is quantized using a separate scale factor denoted by a specific color. (b) Illustration of transforming per-channel quantization into per-layer quantization by rescaling. The scale factor of each input channel is moved into its corresponding 2D kernels of all filters. The bottom row shows how to conduct convolution (CONV) using matrix multiplication (MatMul) operations. Note that at this stage, the weights are not quantized yet. Best viewed in color.

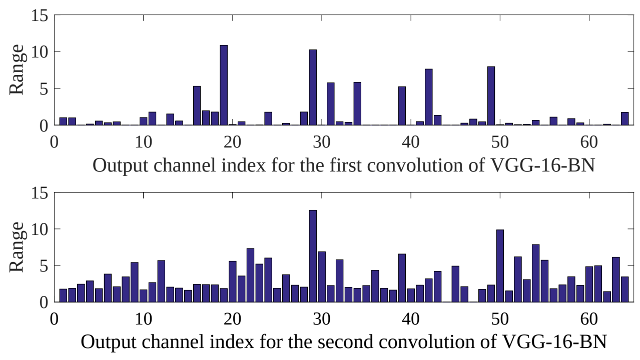

Figure 3. The ranges of activations for each channel (after ReLU) of the first and second convolutions of VGG-16-BN. It can be seen that the ranges for different channels differ severely.

3.2. Error Compensated Activation Quantization

Once the weights are quantized, the quantized weights stay fixed during inference time. However, it is not the case for activations which are dynamically produced at inference time. Thus we could not directly optimize the quantized activations. Therefore, we adopt Mean Squared Error (MSE) as the criterion to minimize the '*distance*' between the activations of pretrained model and those after quantization.

Different from previous approaches, which utilize one quantizer for a whole layer for activation quantization, our proposed Error Compensated Activation Quantization (ECAQ) method could benefit from per-channel quantization, but still has the same efficiency as per-layer quantization without any extra computation or the need of complicated convolution.

Per-channel Activation Quantization In this section, we first show that the activation ranges vary severely for different channels. For demonstration, we extract the output of the first and the second convolutional layers (after ReLU) of VGG-16-BN model for 2400 images which are randomly selected from the training set. We plot the range for each channel as shown in Fig. 3.

From Fig. 3, we can see that large differences are observed in output channel ranges for pretrained models. Due to the large differences between channel ranges, per-layer activation quantization may suffer from large quantization error. Thus in this paper, we explore channel-wise quantizer for activations. Specifically, each input channel is approximated by the quantized channel, which is scaled by a separate scale factor as follows:

$$X[c,:,:] \approx \beta[c] \cdot \hat{X}[c,:,:] \tag{10}$$

Here, β can be optimized by the following formulation:

$$\underset{\beta,\hat{X}}{\text{minimize}} \parallel X - \beta \cdot \hat{X} \parallel_F^2, \tag{11}$$

Fig. 2(a) shows the per-channel quantization scheme, where different channels with different quantization scales are denoted by different colors.

Per-channel quantization can significantly lower the quantization error, however, at the cost of more complicated convolution operations. We will show how to solve this problem in the next section.

From Per-channel to Per-layer Quantization As shown in the previous section, per-channel activation quantization could complicate convolution operations. This is because that convolutions (CONV) are implemented as matrix multiplications (MatMul), as shown in the bottom row of Fig. 2. When each channel has a separate scale factor, the large matrix multiplication is separated into multiple small matrix multiplications, which could dramatically reduce the computing efficiency.

In this section, we exploit the rescaling of activations and weights to facilitate simple convolution implementation when per-channel activation quantization is utilized. Specifically, we can move the scale factor of each input channel to its corresponding 2D kernels of all filters:

$$\begin{aligned} W * X &\approx W * \beta \cdot \hat{X} \\ &= W_\beta * \hat{X}, \end{aligned} \tag{12}$$

where W_β satisfies

$$W_\beta[:, c, :, :] = \beta[c] \cdot W[:, c, :, :] \tag{13}$$

The rescaling operation does not affect the convolution output. Note that at this stage, the weights are not quantized yet, i.e., the weights are floating-point values. After rescaling, we can further quantize the weights W_β using Bit-split method according to Sec. 3.1 as follows:

$$\underset{\alpha_\beta, q_\beta}{\text{minimize}} \parallel y - \alpha_\beta q_\beta^T \hat{X} \parallel_F^2, \tag{14}$$

where α_β and q_β represent the scale factor and quantized filter for the rescaled weights W_β. Fig. 2(b) illustrates the convolution after rescaling, which can also be efficiently implemented by matrix multiplication.

From Eq. (14), we can see that we are actually learning a mapping from the *per-layer quantized activations* $\hat{X}$ to the target output Y by taking the activation quantization and scale factors β into consideration. Thus we call the proposed method Error Compensated Activation Quantization. We summarize the overall optimization procedure of our proposed approach in Algorithm 1.

4. Experiments

In this section, we evaluate the efficiency of our proposed method. We first evaluate the Bit-Split and Stitching method for weight quantization. Then the performance of Error Compensated Activation Quantization is evaluated. We also compare our method with current post-training methods. All bit-width representations throughout this paper take the sign bit into consideration. Codes are available on GitHub at `https://github.com/wps712/BitSplit`.

Algorithm 1 Post-training quantization using Error Compensated Activation Quantization and Bit-Split and Stitching weight quantization.

```
Input: Pretrained model denoted by weights $\{W^l\}_{l=1}^L$, weight
    bit-width $M_w$ and activation bit-width $M_a$.
Output: Quantized weights $\{Q^l\}_{l=1}^L$, quantization scale $\{\alpha^l\}_{l=1}^L$
    for weights and $\{\beta^l\}_{l=2}^L$ for activations.
 1: for $l = 1; l \leq L$ do
 2:   Sampling a mini-batch images
 3:   Forward propagation to get $X^l$ and $Y^l$
 4:   if $l > 1$ then
 5:     Optimize activation quantization scale $\beta^l$ according to
        Eq. (11)
 6:     Move $\beta^l$ from activation to weights to obtain $W_\beta^l$ accord-
        ing to Eq. (13)
 7:   else
 8:     $W_\beta^l \leftarrow W^l$, $\hat{X}^l \leftarrow X^l$
 9:   end if
10:   $Q\_max = 2^{M_w - 1} - 1$
11:   $\alpha^l \leftarrow max(W_\beta^l)/Q\_max$
12:   $Q^l \leftarrow round(W_\beta^l/\alpha^l)$
13:   Calculate $Q_1^l, \cdots, Q_{M_w-1}^l$ using Bit-Split
14:   while not converge do
15:     Optimize $\alpha^l$ according to Eq. (4)
16:     for $m = 1; m < M_w$ do
17:       Optimize $Q_m^l$ according to Eq. (8)
18:     end for
19:     Calculate $Q^l$ using Bit-Stitching
20:   end while
21: end for
```

4.1. Weight Quantization using Bit-Split and Stitching

In this section, we evaluate the post-training weights quantization using our proposed Bit-Split and Stitching method. The top-1 and top-5 accuracy results of post-training quantization are reported using four popular convolutional models pre-trained on the ImageNet dataset. We use the PyTorch pretrained models for all experiments. The results are shown in Table 1. We quantize the weights into 3$\sim$8 bit while keeping activations un-quantized. We also report the TF-Lite results as a baseline for comparison.

From Table 1, it is clearly that the proposed Bit-Split and Stitching method has very small accuracy degradation for post-training weight quantization without fine-tuning. Our method consistently outperforms TF-Lite baseline method. The accuracy gap becomes larger as the bit-width goes down. Using the proposed Bit-Split and Stitching method, no obvious accuracy loss is observed for 4-bit quantization or higher. We also want to highlight that for 3-bit quantization, previous approaches like the TF-Lite fail to work, while in our Bit-Split and Stitching method, only 0.9%$\sim$1.7% top-5 accuracy drop is observed for various networks. To our knowledge, *this is the first work that achieves close-to-full-precision accuracy for 3-bit weight quantization without fine-tuning.*

Table 1. Comparison results of top-1 and top-5 accuracy (%) for post-training **weight quantization for various bit-widths**. Activations are un-quantized. For ResNet-18, the results with 8-bit per-layer activation quantization, denoted by Bit-split (A8), are also given, demonstrating that 8-bit activation quantization has no accuracy loss compared with floating-point activations. Bold values indicate the best results.

Model		8-bit		7-bit		6-bit		5-bit		4-bit		3-bit	
		Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
ResNet-18 (69.76, 89.08)	TF-Lite	69.63	88.96	69.67	89.02	69.06	88.72	66.81	87.39	55.53	79.21	0.85	2.68
	Bit-split	69.79	**89.15**	**69.84**	**89.15**	**69.83**	**89.12**	**69.70**	88.93	**69.11**	**88.69**	**66.76**	87.45
	Bit-split (A8)	**69.82**	**89.15**	69.82	89.05	69.80	**89.12**	69.64	**88.98**	69.10	**88.69**	66.75	**87.46**
ResNet-50 (76.15, 92.87)	TF-Lite	76.12	92.88	76.07	92.86	75.87	92.82	75.17	92.50	70.14	89.57	4.22	11.53
	Bit-split	**76.20**	**92.97**	**76.16**	**92.91**	**76.17**	**92.90**	**76.05**	**92.82**	**75.58**	**92.57**	**73.64**	**91.61**
ResNet-101 (77.47, 93.56)	TF-Lite	77.32	93.57	77.28	93.51	77.06	93.47	76.25	93.05	72.67	90.87	9.19	20.05
	Bit-split	**77.55**	**93.59**	**77.44**	**93.59**	**77.51**	**93.60**	**77.55**	**93.59**	**76.89**	**93.31**	**74.98**	**92.42**
VGG-16-BN (73.37, 91.50)	TF-Lite	73.36	91.51	73.34	91.48	73.12	91.36	72.37	90.86	66.36	87.26	1.16	4.49
	Bit-split	**73.43**	**91.61**	**73.37**	**91.52**	**73.22**	**91.53**	**73.37**	**91.50**	**72.97**	**91.35**	**72.11**	**90.77**

Table 2. Comparison results of top-1 accuracy (%) for post-training activation quantization of ResNet-18 (69.76, 89.08). Weights are quantized to 8-bit using Bit-split. The boldfaced and underlined values stand for the best and second best results for each setting.

Model	8-bit	7-bit	6-bit	5-bit	4-bit	3-bit
Per-layer	**69.82**	69.78	69.79	69.37	68.31	64.77
Per-channel	69.79	**69.86**	**69.79**	**69.63**	**68.83**	65.91
ECAQ	69.74	69.75	69.70	69.52	68.76	**66.21**

In Table 1, we also report the results when activations are quantized into 8-bit for ResNet-18, denoted by Bit-split (A8). We can see that 8-bit activation quantization has no loss compared with floating-point activations. The differences in results (less than 0.1%) may result in randomness.

4.2. Error Compensated Activation Quantization

In this section, we evaluate the proposed Error Compensated Activation Quantization (ECAQ) approach thoroughly. ResNet-18 network is adopted for demonstration. We quantize activations into 3∼8 bit and weights to 8-bit using Bit-split. The results are shown in Table 2.

Table 2 shows that per-channel activation quantization indeed generally outperforms per-layer quantization under various bit-width settings. However, as shown in Sec. 3.2, per-channel activation quantization can complicate convolution implementation. By contrast, both the proposed ECAQ method and direct per-layer quantization utilize a single quantizer for a whole layer, which is efficient at inference time. For higher bit quantization, ECAQ method achieves similar results (less than 0.1% differences) as direct per-layer quantization. However, for bit-width less than 5 bit, the proposed ECAQ consistently outperforms per-layer quantization. Especially, for 4-bit and 3-bit quantization, ECAQ outperforms direct per-layer quantization by 0.45% and 1.44%, respectively, with a single scale for a whole layer.

To further demonstrate the effectiveness of the proposed framework thoroughly, we give the results of Error Compensated Activation Quantization coupled with Bit-Split and Stitching weight quantization for various bit-widths on different networks. The results are shown in Table 3. Like previous settings, the top-1 and top-5 accuracy results for four popular convolutional models are reported. We quantize both weights and activations simultaneously into 3∼8 bit. We also use TF-Lite as a baseline for comparison.

Table 3 shows that when both activations and weights are quantized using the same bit-width, the proposed framework still dramatically outperforms baseline method (TF-Lite) by large margins. For ResNet-18, ResNet-50 and VGG-16-BN, the baseline method fails at 4-bit quantization. By contrast, the proposed method has less than 1.3% top-5 accuracy degradation for these three networks at 4-bit quantization for both weights and activations. Note that even when both weights and activations are quantized to 3-bit, only 5 points accuracy loss is observed, which is encouraging considering the very limited representation space and no fine-tuning is used. For the very deep ResNet-101, the baseline approach fails at 6-bit, while our approach still works at 3-bit quantization.

In summary, the results in Table 3 show that the proposed framework, i.e., Error Compensated Activation Quantization coupled with Bit-Split and Stitching weight quantization, is powerful for post-training uniform quantization. We will compare the proposed framework with previous works thoroughly in the following section.

4.3. Comparison with State-of-the-arts

In this section, we evaluate the proposed framework with existing post-training quantization approaches. The results are shown in Table 4, which are summarized under different bit-widths: A8W4 represents 8-bit activations and 4-bit weights, and A4W4 indicates both weights and activations

Table 3. Comparison results of Top-1 and Top-5 accuracy (%) for post-training quantization of **both weights and activations for various bit-widths**. Bold values indicate the best results.

Model		8-bit		7-bit		6-bit		5-bit		4-bit		3-bit	
		Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
ResNet-18	TF-Lite	69.57	89.02	69.46	88.87	67.95	88.02	61.47	83.43	18.84	36.33	0.13	0.61
(69.76, 89.08)	Bit-split	**69.74**	**89.09**	**69.68**	**89.07**	**69.58**	**88.96**	**69.28**	**88.77**	**67.56**	**87.76**	**61.30**	**83.47**
ResNet-50	TF-Lite	**76.05**	**92.93**	75.75	92.70	73.83	91.66	65.46	86.34	10.40	22.36	0.11	0.54
(76.15, 92.87)	Bit-split	75.96	92.83	**76.09**	**92.84**	**75.90**	**92.75**	**75.38**	**92.59**	**73.71**	**91.62**	**66.22**	**87.18**
ResNet-101	TF-Lite	76.78	93.31	74.07	91.79	31.78	55.96	0.82	2.65	0.25	0.98	0.09	0.54
(77.47, 93.56)	Bit-split	**77.23**	**93.55**	**77.20**	**93.47**	**76.93**	**93.42**	**76.07**	**92.95**	**74.68**	**92.18**	**63.96**	**85.65**
VGG-16-BN	TF-Lite	73.31	91.53	72.94	91.25	70.65	89.77	54.45	78.18	3.41	10.17	0.18	0.78
(73.37, 91.50)	Bit-split	**73.43**	**91.54**	**73.43**	**91.55**	**73.34**	**91.45**	**72.89**	**91.22**	**71.14**	**90.29**	**66.11**	**86.92**

Table 4. Comparison of different post-training quantization approaches on ImageNet classification benchmark. Top-1 accuracy (%) is reported. We also indicate whether per-layer or per-channel activation quantization, and unified-precision or mixed-precision quantization schemes are utilized. Bold values indicate the best results.

Model		Per-layer	Unified-precision	ResNet-18	ResNet-50	ResNet-101	VGG-16-BN
Full-precision		-	-	69.76	76.15	77.47	73.37
A8W4	TF-Lite (Krishnamoorthi, 2018)	√	√	55.5	70.1	72.6	66.4
	ACIQ (Banner et al., 2019)	×	√	67.4	74.8	76.3	71.7
	ACIQ-Mix (Banner et al., 2019)	×	×	68.3	75.3	**76.9**	72.4
	Bit-split	√	√	**69.1**	**75.6**	**76.9**	**73.0**
A4W4	TF-Lite (Krishnamoorthi, 2018)	√	√	18.8	10.4	0.3	3.4
	TensorRT (Migacz, 2017)	√	√	31.9	46.2	49.9	-
	LAPQ (Nahshan et al., 2019)	√	√	59.8	70.0	59.2	-
	ACIQ-Mix (Banner et al., 2019)	×	×	67.0	73.8	75.0	**71.8**
	Bit-split	√	√	67.6	73.7	74.7	71.1
	Bit-split-per-channel	×	√	**68.1**	**74.2**	**75.3**	**71.8**

are quantized into 4-bit. For comprehensive comparison, we also report whether per-layer or per-channel activation quantization, and unified-precision or mixed-precision quantization schemes are utilized.

Table 4 shows that the proposed bit-split frameworks dramatically outperforms previous results when per-layer activation quantization and unified precision quantization schemes are utilized. With 4-bit weight quantization, our bit-split framework even outperforms current state-of-the-art method ACIQ-Mix, which utilizes per-channel quantization, mixed-precision and dynamic quantization. When we utilize per-channel quantization for the proposed bit-split framework, the accuracy can further be improved, setting new state-of-the-arts for post-training quantization.

4.4. Object Detection and Instance Segmentation

To show the generalization ability of our proposed framework, we conduct experiments on object detection and instance segmentation. MS COCO dataset is used for evaluation. The experiments are conducted using mmdetection[1] toolbox. The pre-trained models are trained on 80k training images and 35k of validation images (trainval35k), and is evaluated on the remaining 5k validation images (minival). Input images are resized to 800 pixels in the shorter edge.

We evaluate bit-split framework using single-stage object detection of RetinaNet (Lin et al., 2017a), as well as the object detection and instance segmentation using two-stage Mask R-CNN (He et al., 2017). Both networks using ResNet-50 as backbone. We quantize all layers into 4bit except the first layer and the final output layers which are quantized to 8bit. Activations are quantized into 8bit. The results are shown in Table 5. We can see that there are about 0.8%$\sim$1.2% mAP degradation with 8bit activations and 4bit weights without fine-tuning, which demonstrates the generalization ability of the proposed framework.

Table 5. Object detection (bounding box AP) and instance segmentation (mask AP) results on COCO minival set.

Model		$AP_{0.5:0.95}$	$AP_{0.5}$	$AP_{0.75}$
RetinaNet	Full-precision	35.6	55.5	38.3
(Box)	A8W4	34.4	54.2	36.5
Mask R-CNN	Full-precision	37.3	59.0	40.2
(Box)	A8W4	36.2	57.5	39.3
Mask R-CNN	Full-precision	34.2	55.9	36.2
(Mask)	A8W4	33.4	54.4	35.4

[1]https://github.com/open-mmlab/mmdetection

5. Conclusion

In this work, we propose a Bit-Split and Stitching framework for lower-bit post-training quantization with minimal accuracy degradation. The proposed framework is effective and hyper-parameter free, which can be easily implemented and integrated into current network quantization libraries. Based on the Bit-split framework, we also propose an Error Compensated Activation Quantization method, which could lower the quantization error for activations. The proposed framework is validated on a variety of computer vision tasks, including image classification, object detection, instance segmentation with various network architectures, showing close to full-precision results even for 3-bit quantization without fine-tuning.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No.61906193), the Strategic Priority Research Program of Chinese Academy of Science (No.XDB32050200), the Advance Research Program (No.31511130301), National Key R&D Plan of China (No.2018AAA0103304) and Jiangsu Frontier Technology Basic Research Project (No.BK20192004).

References

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

Cai, Y., Yao, Z., Dong, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. Zeroq: A novel zero shot quantization framework. *arXiv preprint arXiv:2001.00281*, 2020.

Cai, Z., He, X., Sun, J., and Vasconcelos, N. Deep learning with low precision by half-wave gaussian quantization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

Chen, Q., Cheng, A., He, X., Wang, P., and Cheng, J. Spatialflow: Bridging all tasks for panoptic segmentation. *arXiv preprint arXiv:1910.08787*, 2019.

Faraone, J., Fraser, N., Blott, M., and Leong, P. H. Syq: Learning symmetric quantization for efficient deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. Deep learning with limited numerical precision. *CoRR, abs/1502.02551*, 392, 2015.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pp. 1135–1143, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

Horowitz, M. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, Feb 2014. doi: 10.1109/ISSCC.2014.6757323.

Hu, Q., Li, G., Wang, P., Zhang, Y., and Cheng, J. Training binary weight networks via semi-binary decomposition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 637–653, 2018a.

Hu, Q., Wang, P., and Cheng, J. From hashing to cnns: Training binary weight networks via hashing. In *AAAI*, February 2018b.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR*, abs/1806.08342, 2018.

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

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., and Lempitsky, V. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

Li, Z., Ni, B., Zhang, W., Yang, X., and Gao, W. Performance guaranteed network acceleration via high-order residual quantization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 2603–2611, 2017.

Lin, D. D., Talathi, S. S., and Annapureddy, V. S. Fixed point quantization of deep convolutional networks. In *Proceedings of the 33nd International Conference on*

Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, pp. 2849–2858, 2016.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017a.

Lin, X., Zhao, C., and Pan, W. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 344–352, 2017b.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., and Berg, A. C. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, 2016.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

Migacz, S. 8-bit inference with tensorrt. In *GPU technology conference*, volume 2, pp. 5, 2017.

Miyashita, D., Lee, E. H., and Murmann, B. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

Nagel, M., Baalen, M. v., Blankevoort, T., and Welling, M. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1325–1334, 2019.

Nahshan, Y., Chmiel, B., Baskin, C., Zheltonozhskii, E., Banner, R., Bronstein, A. M., and Mendelson, A. Loss aware post-training quantization. *arXiv preprint arXiv:1911.07190*, 2019.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Tang, W., Hua, G., and Wang, L. How to train a compact binary neural network with high accuracy? In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pp. 2625–2631, 2017.

Wang, P. and Cheng, J. Accelerating convolutional neural networks for mobile applications. In *Proceedings of the 2016 ACM on Multimedia Conference*, pp. 541–545. ACM, 2016.

Wang, P. and Cheng, J. Fixed-point factorized networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Wang, P., Hu, Q., Zhang, Y., Zhang, C., Liu, Y., and Cheng, J. Two-step quantization for low-bit neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4376–4384, 2018.

Wang, P., He, X., Li, G., Zhao, T., and Cheng, J. Sparsity-inducing binarized neural networks. In *AAAI*, pp. 12192–12199, 2020.

Wu, J., Leng, C., Wang, Y., Hu, Q., and Cheng, J. Quantized convolutional neural networks for mobile devices. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Zhou, A., Yao, A., Guo, Y., Xu, L., and Chen, Y. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

Zhu, S., Dong, X., and Su, H. Binary ensemble neural network: More bits per network or more networks per bit? In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.