Supplementary material
RepQ: Generalizing Quantization-Aware Training for Re-Parametrized Architectures

BMVC 2023 Submission # 311

In supplementary material, we provide two sections. The first one is devoted to the description of how we select hyperparameters for various algorithms and the motivation behind this selection. The second one provides a derivation of our approximate form of Batch Normalization (BN).

S1 Experimental setup

We have reproduced the results of the Floating-Point (FP) models using official repositories[1], except for AC-ResNet-18, which we implemented from scratch in TensorFlow. To ensure reproducibility, we provide the exact hyperparameters used for all experiments in Table S1. We maintain the same number of epochs, batch size, loss function, optimizer, weight decay, and schedule for quantized training as in the FP training. We only modify a Learning Rate (LR). We reduced the LR by a factor of ten for the Quantization-Aware Training (QAT) stage in comparison with FP training LR for the classification models. The RepQ method and the baselines are trained using the same LR, except for the Merged 8-bit quantization baseline. For it, we additionally reduced LR since it benefited the quality. For instance, the 8-bit RepVGG-A0 Merged baseline achieves a 68.65 accuracy with a LR of 0.01 and a 69.21 accuracy with a LR of 0.001.

Additional quantization parameters, referred to as steps, are introduced in [1]. We use independent steps for each channel of a weight tensor [2]. We adjust the ResNet-18 steps' LR to ensure training stability. For example, with steps' LR of 0.01, the 4-bit OREPA-Resnet-18 model converges to an accuracy of 65.49, and with steps' LR of 0.001, it achieves an accuracy of 71.49.

All aforementioned LR adjustments are displayed in Table S1.

We deliberately chose not to modify hyperparameters other than the LR to facilitate a fair comparison between the proposed methods and the baselines. Selecting optimal hyperparameters for each experiment can be challenging and resource-intensive. However, quantized models can sometimes benefit from certain hyperparameter fine-tuning. For instance, reducing the weight decay by a factor of two for the 4-bit OREPA-Resnet-18 model improves its quality from 71.49 to 72.15.

For 4-bit quantization, it is a common practice to keep the weights or inputs of the first and last layers in 8-bit, as these layers are more sensitive to quantization. In the case of 4-bit

[1] Links to official repositories that we used: ECBSR, OREPA, RepVGG.

ECBSR and OREPA-Resnet-18, we keep the first and last layers fully 8-bit. For AC-Resnet-18, only the first layer is 8-bit, while for RepVGG, only the input of the first layers is 8-bit. Although there is a slight inconsistency in the configurations of different models, it does not affect the comparison of the methods and the resulting conclusions.

For BNEst experiments, we replace BN layers with BN estimation layers on both FP and QAT stages. For ResNet-18, we leave two last layers with BN due to the edge effect described in Section S2 and replace BN with BN estimation in all other layers.

Regarding model selection, we report the highest quality achieved on the ImageNet validation set during training for the classification models. For ECBSR, we report the results of the last checkpoint since this model is tested on multiple test sets.

Model	RepVGG-A0 & RepVGG-B0	AC-Resnet-18	OREPA-Resnet-18	ECBSR
Epochs	120	100	120	1000
Batch size	256	256	256	32
Loss	Cross entropy + label smoothing 0.1	Cross entropy	Cross entropy	MAE
Optimizer	SGD momentum 0.9	SGD momentum 0.9	SGD momentum 0.9	Adam
Schedule	Cosine + 5 epochs linear warmup	Cosine	Cosine + 5 epochs linear warmup	Constant
Weight decay	0.0001	0.0001	0.0001	0.0
FP models' LR	0.1	0.1	0.1	0.0005
Quantized models' LR	0.01	0.01	0.01	0.0005
Merged 8-bit models' LR	0.001	0.001	0.001	0.0005
8-bit steps' LR	0.01	0.0001	0.0001	0.0005
4-bit steps' LR	0.01	0.001	0.001	0.0005

Table S1: Hyperparameters' setup.

S2 Batch Normalization Estimation

Here we derive the Batch Normalization Estimation formulas provided in Section 4.3.

Let's introduce notation. Consider an arbitrary convolution operation, denoted as $X * W$, where X is the input tensor with shape $[B,H,D,IN]$, and W is the weight tensor with shape $[K_h,K_w,IN,OUT]$. In this notation, B represents the batch size, H is the height of the feature map, D is the width of the feature map, K_h is the height of the weight tensor, K_w is the width of the weight tensor, IN is the number of input channels, and OUT is the number of output channels.

We define a flattening operator, denoted as F, which reshapes the tensor X from shape $[B,H,D,IN]$ to shape $[B\cdot H\cdot D,IN]$.

Using this notation, we can express the convolution operation as a sum of several matrix multiplications,

$$(X*W)^F = \sum_{\substack{0\leqslant i<K_h \\ 0\leqslant j<K_w}} X[,i:H-K_h+1+i,j:D-K_w+1+j,]^F W_{i,j}. \tag{S1}$$

The notation $[i:j;\ldots]$ represents a slicing operator commonly used in Python to extract specific elements from an array or tensor. $W_{i,j} = W[i,j]$ is a matrix of shape $[IN, OUT]$. This decomposition is convenient for further deductions.

We define $\mathbb{E}$ as an operator that computes the sample mean over the batch, height and width dimensions of the tensor. $\mathbb{E}$ is defined for the flattened input similarly, so that $\mathbb{E}\Big[X^F\Big] = \mathbb{E}\Big[X\Big]$. The resulting tensor will have a shape $[OUT,]$. Similarly, the operator $\mathbb{V}$ calculates the sample variance over the same batch, height, and width dimensions. These statistics, denoted as $\mathbb{E}$ and $\mathbb{V}$, are essential components used in regular Batch Normalization techniques.

With these notations and operators defined, we can now proceed to derive the estimations for Batch Normalization statistics for arbitrary convolutions, starting with the mean,

$$\begin{aligned}\mathbb{E}\Big[X * W\Big] &= \mathbb{E}\Big[(X*W)^F\Big] = \\ \mathbb{E}\Big[\sum_{\substack{0\leqslant i<K_h\\0\leqslant j<K_w}} X[,i:H-K_h+1+i,j:D-K_w+1+j,]^F W_{i,j}\Big] &= \\ = \sum_{\substack{0\leqslant i<K_h\\0\leqslant j<K_w}} \mathbb{E}\Big[X[,i:H-K_h+1+i,j:D-K_w+1+j,]\Big] W_{i,j}&.\end{aligned} \tag{S2}$$

We can make the assumption that $\mathbb{E}\Big[X[,i:H-K_h+1+i,j:D-K_w+1+j,]\Big] \approx \mathbb{E}\Big[X\Big]$, which implies that we can neglect the pixels near the edges of the feature map. This approximation error remains small when the weight tensor's shape is much smaller than the height and width of the feature maps, $K_h \ll H$, and $K_w \ll D$. In many networks, this condition holds true. For example, in state-of-the-art models, the ImageNet input image is regularly reshaped into 224x224, while the weights are typically 3x3 for most layers. However, it is worth noting that the height and width of the feature maps may be significantly reduced in the last layers, which can lead to less accurate estimates. That is why for ResNet-18 BNEst experiments, we left two last layers to use regular BN.

By employing this approximation, we can proceed to deduce the final mean estimate,

$$\mathbb{E}\Big[X * W\Big] \approx \tilde{\mathbb{E}}\Big[X * W\Big] = \sum_{\substack{0\leqslant i<K_h\\0\leqslant j<K_w}} \mathbb{E}\Big[X\Big] W_{i,j} = \mathbb{E}\Big[X\Big] \sum_{\substack{0\leqslant i<K_h\\0\leqslant j<K_w}} W_{i,j}. \tag{S3}$$

Similarly, we provide equations for estimating the variance,

$$\begin{aligned}\mathbb{V}\Big[X * W\Big] &= \mathbb{V}\Big[(X*W)^F\Big] = \\ \mathbb{V}\Big[\sum_{\substack{0\leqslant i<K_h\\0\leqslant j<K_w}} X[,i:H-K_h+1+i,j:D-K_w+1+j,]^F W_{i,j}\Big]&.\end{aligned} \tag{S4}$$

We further assume that the variance of the sum is approximately equal to the sum of the variances,

$$\mathbb{V}\Big[X * W\Big] \approx \sum_{\substack{0\leqslant i<K_h\\0\leqslant j<K_w}} \mathbb{V}\Big[X[,i:H-K_h+1+i,j:D-K_w+1+j,]^F W_{i,j}\Big]. \tag{S5}$$

Using equation (4) from the article,

$$\mathbb{V}\Big[X * W\Big] \approx \sum_{\substack{0 \leqslant i < K_h \\ 0 \leqslant j < K_w}} \mathbb{V}\Big[X[, i : H - K_h + 1 + i, j : D - K_w + 1 + j,]\Big] W_{i,j}^2. \tag{S6}$$

Similarly to $\mathbb{E}$, we neglect the edge effect, assuming,

$$\mathbb{V}\Big[X[, i : H - K_h + 1 + i, j : D - K_w + 1 + j,]\Big] \approx \mathbb{V}\Big[X\Big]. \tag{S7}$$

We get the final variance estimate,

$$\mathbb{V}\Big[X * W\Big] \approx \tilde{\mathbb{V}}\Big[X * W\Big] = \mathbb{V}\Big[X\Big] \sum_{\substack{0 \leqslant i < K_h \\ 0 \leqslant j < K_w}} W_{i,j}^2. \tag{S8}$$

References

[1] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned Step Size Quantization. In *Proceedings of the International Conference on Learning Representations*, 2020.

[2] Yuhang Li, Mingzhu Shen, Jian Ma, Yan Ren, Mingxin Zhao, Qi Zhang, Ruihao Gong, Fengwei Yu, and Junjie Yan. MQBench: Towards reproducible and deployable model quantization benchmark. In *Proceedings of the Conference on Neural Information Processing Systems*, 2021.