Search What You Want: Barrier Panelty NAS for Mixed Precision Quantization Supplementary Material

Haibao Yu[1⋆], Qi Han[1⋆], Jianbo Li[1,2], Jianping Shi[1], Guangliang Cheng[1 ⋆⋆], and Bin Fan[3 ⋆⋆]

[1] SenseTime Research, Beijing, China
[2] Peking University, Beijing, China
[3] University of Science and Technology Beijing, Beijing, China
{yuhaibao, hanqi, shijianping, chengguangliang}@sensetime.com, jianbo.li@pku.edu.cn, bin.fan@ieee.org

Abstract. Section 1 gives an example explaining how BOPs is calculated. Section 2 gives the detail proof for the Property 1. In Section 3, it gives the derivative forms of the Prob-1 regularizer and barrier penalty. In Section 4, we provide the searched mixed precision configurations for ResNet20 on Cifar-10 and ResNet50 Faster R-CNN on COCO. Finally, Section 5 discusses the effectiveness of our distribution reshaping method on mixed precision training.

1 Example of the BOPs calculation

Let us consider a convolutional layer with b-bit weights and a-bit activations. Assuming its size is $Ci \times Co \times K \times K$ (where Ci=input channel, K=kernel size, Co=output channel) and the output size is $1 \times Co \times H \times W$, then for a single element of the output, it consists of $Ci \times K^2$ multiplication operations and $Ci \times K^2$ addition operations. Each multiplication operation involves $b \times a$ bit operations, and each addition operation involves $b + a + log_2(Ci \times K^2)$ bit operations. To have a fair comparison with SOTA work like [3,4], we aslo keep the multiplication and addition sharing the same number of bit operations as $b \times a$. Then this convolutional layer yields a total number of bit operations $BOPs \approx FLOPs \times b \times a$. Moreover, to facilitate our comparison with fixed-precision, we refer $BOPs$ to the average number of bit operations as $(BOPs/FLOPs)^{1/2}$. With a preset bit budget B_{max}, the real BOPs budget is $B_{max}^2 \times FLOPs$.

⋆ Indicates equal contributions
⋆⋆ Indicates equal corresponding authors

2 Proof of Property 1

Property 1. Prob-1 function $f(\mathbf{x})$ achieves the minimal value if and only if there exists unique x_j to reach 1, where $\mathbf{x} = (x_1, \cdots, x_m)$ is m-dimension vector and

$$f(\mathbf{x}) = \prod_j (1 - x_j), \quad s.t. \ \sum x_j = 1, 0 \leq x_j \leq 1. \tag{1}$$

Proof. Firstly, we prove that if there exists unique x_j to reach 1 then $f(\mathbf{x})$ achieves the minimal value. For any x_j, it satisfies that $0 \leq x_j \leq 1$, such that $0 \leq (1 - x_j) \leq 1$. Thus $f(\mathbf{x})$ is always larger than or equal to 0. When there exists x_j to reach 1, $f(\mathbf{x})$ reaches 0, that means that $f(\mathbf{x})$ achieves the minimal value 0.
Secondly, we prove that if $f(\mathbf{x})$ achieves the minimal value then there exists unique x_j to reach 1. Clearly when $f(\mathbf{x})$ achieves the minimal value 0, there exists $1 - x_j$ to reach 0. Since $0 \leq x_j \leq 1$ and $\sum x_j = 1$, all other $x_{i,i \neq j}$ is equal to 0. Thus there exists unique x_j to reach 1.

We also introduce a simple example to illustrate the Property 1 as follows

$$\begin{aligned} &f(\mathbf{x}) = (1 - x_1)(1 - x_2), \\ &s.t. \quad x_1 + x_2 = 1, 0 \leq x_1, x_2 \leq 1, \end{aligned} \tag{2}$$

where $\mathbf{x} = (x_1, x_2)$. By replacing x_2 as $1 - x_1$ we can rewrite the problem as

$$f(\mathbf{x}) = (1 - x_1)x_1, \quad s.t. \quad 0 \leq x_1 \leq 1.$$

Obviously, $f(\mathbf{x})$ achieves minimum if and only if x_1 is assigned to be 0 or 1.

3 Derivatives of Prob-1 Regularizer and Barrier Penalty

3.1 Derivative of Barrier Penalty

Consider the barrier penalty

$$\mathcal{L}_c^*(\theta) = -\mu log(log(\mathcal{B}_{max} + 1 - E(\mathcal{SN}))), \tag{3}$$

where E denotes the expected complexity cost $\mathcal{F}(\mathcal{SN}; \theta)$ of the supernet.
We first take the derivative of the barrier penalty about $E(\mathcal{SN})$ as $\frac{\partial \mathcal{L}_c^*}{\partial E}$.

$$\begin{aligned} \frac{\partial \mathcal{L}_c^*}{\partial E} &= \frac{\partial(-\mu log(log(\mathcal{B}_{max} + 1 - E(\mathcal{SN}))))}{\partial E} = -\mu \frac{\partial log(log(\mathcal{B}_{max} + 1 - E(\mathcal{SN})))}{\partial E} \\ &= \frac{-\mu}{log(\mathcal{B}_{max} + 1 - E(\mathcal{SN}))} \frac{\partial log(\mathcal{B}_{max} + 1 - E(\mathcal{SN}))}{\partial E} \\ &= \frac{-\mu}{log(\mathcal{B}_{max} + 1 - E(\mathcal{SN}))} \frac{1}{\mathcal{B}_{max} + 1 - E(\mathcal{SN})} \frac{\partial(\mathcal{B}_{max} + 1 - E(\mathcal{SN}))}{\partial E} \\ &= \frac{\mu}{log(\mathcal{B}_{max} + 1 - E(\mathcal{SN}))(\mathcal{B}_{max} + 1 - E(\mathcal{SN}))} \end{aligned} \tag{4}$$

Then, according to the chain rule, we have

$$\frac{\partial \mathcal{L}_c^*}{\partial \theta} = \frac{\partial \mathcal{L}_c^*}{\partial E}\frac{\partial E}{\partial \theta} = \frac{\mu}{log(\mathcal{B}_{max}+1-E(\mathcal{SN}))(\mathcal{B}_{max}+1-E(\mathcal{SN}))}\frac{\partial E}{\partial \theta}. \tag{5}$$

Here $\frac{\partial E}{\partial \theta}$ has been discussed in [1, 3].

3.2 Derivative of Prob-1 Regularizer

For the ith block of the supernet, $p_{i,j}$ represents the importance of the jth operation. Since $\{p_{i,j}\}$ is obtained by Softmax, we have $\sum_j p_{i,j} = 1$ and $0 \leq p_{i,j} \leq 1$.

For the Prob-1 regularizer with the following form,

$$\mathcal{L}_{prob-1} = \sum_l \prod_m (1 - p_{l,m}). \tag{6}$$

The partial derivative $\frac{\partial \mathcal{L}_{prob-1}}{\partial p_{i,j}}$ is

$$\begin{aligned}\frac{\partial \mathcal{L}_{prob-1}}{\partial p_{i,j}} &= \frac{\partial(\sum_l \prod_m (1-p_{l,m})}{\partial p_{i,j}} = \frac{\partial(\sum_{l\neq i} \prod_m (1-p_{l,m}) + \prod_m (1-p_{i,m}))}{\partial p_{i,j}} \\ &= \frac{\partial(\prod_m (1-p_{i,m}))}{\partial p_{i,j}} = \frac{\partial(\prod_{m\neq j}(1-p_{i,m}) * (1-p_{i,j}))}{\partial p_{i,j}} \\ &= \prod_{m\neq j}(1-p_{i,m}) * (-1)\end{aligned} \tag{7}$$

4 Mixed Precision Configuration

In this section, we provide the exact mixed precision configurations for different blocks for ResNet20 on Cifar-10 as well as ResNet50 Faster R-CNN on COCO.

Table 1. Mixed precision configuration for ResNet20 on Cifar-10. We abbreviate block type as "B-Type". "Params" represents the number of weight. "w-bits" and "a-bits" represent the bitwidths for weights and activations, respectively. We report three mixed precision configurations under different BOPs constraints $\mathcal{B}_{max}$ = 4-bit, 3.5-bit, 3-bit. We quantize the first convolutional layer into 8-bit.

Block	B-Type	Params	FLOPs	$\mathcal{B}_{max}$=3		$\mathcal{B}_{max}$=3.5		$\mathcal{B}_{max}$=4	
				w-bit	a-bit	w-bit	a-bit	w-bit	a-bit
Block 0	Conv	4.32×10^2	4.42×10^6	8	8	8	8	8	8
Block 1	BasicBlock	4.61×10^3	4.72×10^6	3	3	3	3	6	4
Block 3	BasicBlock	4.61×10^3	4.72×10^6	3	3	2	4	4	4
Block 3	BasicBlock	4.61×10^3	4.72×10^6	3	3	4	4	4	4
Block 4	BasicBlock	1.38×10^4	3.54×10^6	3	3	4	4	4	3
Block 5	BasicBlock	1.84×10^4	4.72×10^6	2	4	3	3	3	3
Block 6	BasicBlock	1.84×10^4	3.54×10^6	2	4	2	4	2	4
Block 7	BasicBlock	5.53×10^4	4.72×10^6	3	3	3	3	3	3
Block 8	BasicBlock	7.37×10^4	4.72×10^6	3	3	3	3	3	3
Block 9	BasicBlock	7.37×10^4	4.72×10^6	3	3	3	3	3	3

Table 2. Mixed precision configuration for ResNet50 Faster R-CNN on COCO. We abbreviate block type as "B-Type", number of blocks as "N-Block". All layers in the same block share the same quantization bitwidth.

Part	B-Type	N-Block	w_bit	a_bit
backbone.layer0	Conv	1	4	8
backbone.layer1	Bottleneck	3	[4,3,3]	[4,5,5]
backbone.layer2	Bottleneck	4	[3,4,4,4]	[5,4,3,4]
backbone.layer3	Bottleneck	6	[4,4,4,3,4,4]	[6,3,4,5,4,3]
backbone.layer4	Bottleneck	3	[3,4,4]	[5,4,6]
neck	Conv	8	[4,4,4,4,4,4,4,4]	[4,4,4,4,4,4,4,4]
roi_head	Conv	3	[4,4,4]	[4,4,4]
bbox_head	Linear	4	[4,4,4,4]	[4,4,4,4]

5 Robust Training

In this section, we discuss the distribution reshaping strategy, which facilitates mixed precision training more robust and achieves higher accuracy.

We randomly select bitwidth from $\{(1,1),(2,2),(4,4),(8,8)\}$ for each block in ResNet20, and construct a mixed precision ResNet20. We train two mixed precision ResNet20 models with the same mixed precision configuration and training strategies, except whether applying uniformization or not. Both of them are trained for 160 epoches. Fig. 1 shows the training loss and validation accuracy of the two mixed precision ResNet20 models. In Fig. 1, training with uniformization achieves 89% Top-1 accuracy, while training without uniformization only

achieves no more than 60% Top-1 accuracy, even drops a lot at last. The Fig. 1 clearly demonstrate the robustness in training with uniformization.

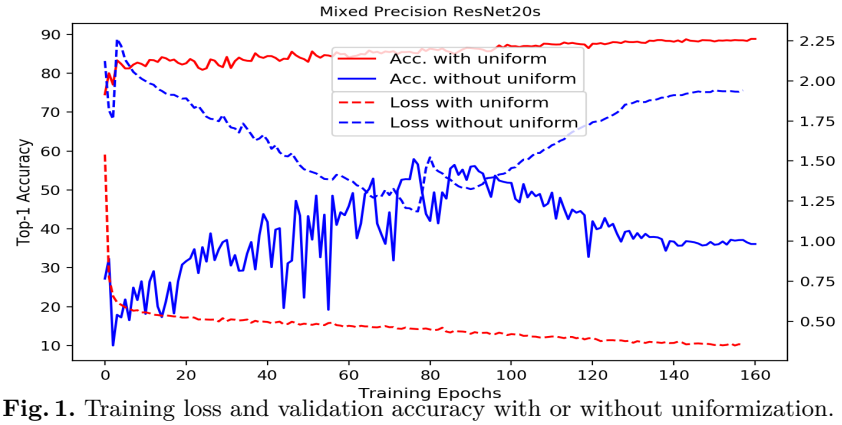

Fig. 1. Training loss and validation accuracy with or without uniformization.

Then we compare our distribution reshaping strategy with PACT [2] in Fig. 2. Our validation accuracy behaves more robust, and achieves higher Top-1 Accuracy than PACT [2].

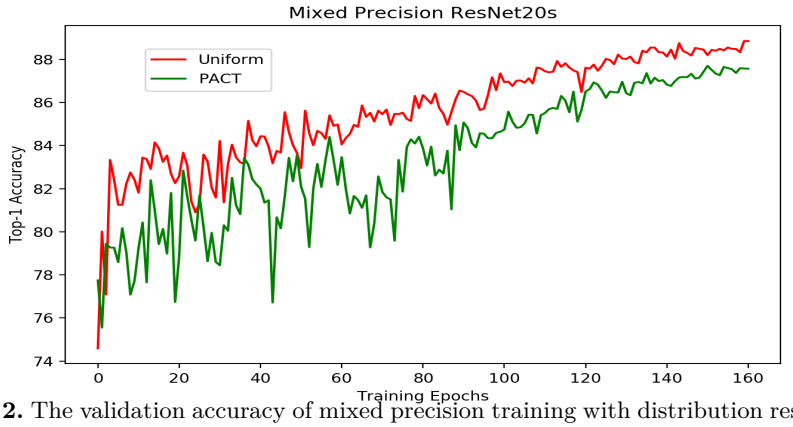

Fig. 2. The validation accuracy of mixed precision training with distribution reshaping strategy and PACT [2].

References

1. Cai, H., Zhu, L., Han, S.: Proxylessnas: Direct neural architecture search on target task and hardware. ICLR (2019)
2. Choi, J., Wang, Z., Venkataramani, S., Chuang, P.I.J., Srinivasan, V., Gopalakrishnan, K.: Pact: Parameterized clipping activation for quantized neural networks. arXiv preprint arXiv:1805.06085 (2018)
3. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. ICLR (2019)