Supplementary Material for "DBQ: A Differentiable Branch Quantizer for Lightweight Deep Neural Networks "

Table of Contents

1 Experimental Setup

In this section, we describe the experimental setup used for generating all our results.

1.1 CIFAR-10

Data Augmentation The CIFAR-10 dataset consits of 32×32 RGB images. For generating the training samples, we adopt the standard data augmentation used in [3] where each image is: 1) zero-padded with 4 pixels on each side; 2) horizontally flipped with probability 0.5; and 3) randomly cropped using a 32×32 window. During testing, we use the 32×32 images as is from the testing set. We also normalize the images, for both training and testing, using a per-channel mean and standard deviation calculated across the training set.

Training Hyperparameters For training the full precision (FP) ResNet-20 baseline on CIFAR-10, we use SGD with momentum $\beta = 0.9$, batch size of 100, and weight decay of $\lambda = 10^{-4}$. The FP model is trained for a total of $E_{\mathrm{T}} = 200$ epochs, with an initial learning rate $\eta_0 = 0.1$ and a cosine update rule [4]:

$$\eta_e = \frac{\eta_0}{2}\left(1 + \cos\left(\frac{e}{E_{\mathrm{T}}}\pi\right)\right) \tag{1}$$

During the fine-tuning process, i.e. training the model with weights initialized from the FP baseline, we train using the same setup as before, but for a fewer number of epochs $E_{\mathrm{T}} = 50$ and a smaller initial learning rate $\eta_0 = 0.01$. The DBQ models trained use a linear temperature increment schedule:

$$T_e = T_{\mathrm{init}} + e \cdot T_{\mathrm{inc}} \tag{2}$$

with an initial temperature $T_{\mathrm{init}} = 5$ and increments $T_{\mathrm{inc}} = 2.5$.

1.2 ImageNet

Data Augmentation For our ImageNet experiments, we follow the standard data augmentation used in [2], where during training, images are: 1) resized; 2) horizontally flipped; and 3) randomly cropped to 224×224. During testing, all images are resized to 256×256 and then cropped to 224×224. We also normalize the input images on a per-channel basis.

Training Hyperparameters For training the full precision MobileNetV1 baseline on ImageNet, we use a similar setup as our CIFAR-10 experiments, with a slightly different learning rate schedule. Similar to [1], the first E_{W} epochs are used for learning rate "warm-up":

$$\eta_e = \frac{(e+1)\eta_0}{E_W} \tag{3}$$

after which the remaining epochs utilize a cosine learning rate as described in (1). The hyperparameters used for both FP and quantization fine-tuning are specified in Table 1.

The full precision MobileNetV2 and ShuffleNetV2 baselines on ImageNet are pre-trained models obtained from PyTorch [5]. Their 2T quantized counterparts, MobileNetV2-2T and ShuffleNetV2-2T, are fine-tuned using the training hyperparameters described in Table 2.

	Batch Size	β	λ	η_0	E_{W}	E_{T}	T_{init}	T_{inc}
FP	512	0.9	4×10^{-5}	0.1	5	150	NA	NA
Quant.	512	0.9	4×10^{-5}	0.001	0	50	50	20

Table 1. Training hyperparameters used for MobileNetV1 experiments on the ImageNet dataset.

	Batch Size	β	λ	η_0	E_{W}	E_{T}	T_{init}	T_{inc}
MobileNetV2-2T	256	0.9	4×10^{-5}	5×10^{-4}	0	50	25	10
ShuffleNetV2-2T	512	0.9	4×10^{-5}	0.001	0	30	25	10

Table 2. Training hyperparameters used for quantized MobileNetV2 and ShuffleNetV2 experiments on the ImageNet dataset.

1.3 Visual Wake Words

Data Augmentation For data augmentation during training, we follow the exact setup as our ImageNet experiments with input normalization and random horizontal flips and crops. During testing, images are normalized, resized to 256×256, and then cropped to 224×224.

Training Hyperparameters The training setup used is identical to our ImageNet experiments as well, and Table 3 specifies the values of the hyperparameters used for both full precision and quantization training.

2 Gradient Derivations

In this section, we provide derivations for the gradient expressions of the loss function $\mathcal{L}$ with respect to the full precision weights $\mathbf{w} \in \mathbb{R}^D$ and the quantizer

	Batch Size	β	λ	η_0	E_{W}	E_{T}	T_{init}	T_{inc}
FP	512	0.9	4×10^{-5}	0.1	5	200	NA	NA
Quant.	512	0.9	4×10^{-5}	0.01	0	50	20	5

Table 3. Training hyperparameters used for experiments on the Visual Wake Words dataset.

parameters $\mathcal{P}_Q = \{\alpha_1, ..., \alpha_B, \gamma_1, \gamma_2, t_1, ..., t_{N-1}\}$. Recall that during training, the quantizer expression is:

$$\mathbf{z} = Q_T(\mathbf{w}) = \gamma_2 \left[\sum_{i=1}^{N-1} \left[\hat{f}_T(\gamma_1 \mathbf{w} - t_i) \sum_{j=1}^{B} b_{i,j} \alpha_j \right] - \sum_{j=1}^{B} \alpha_j \right] \tag{4}$$

where $\hat{f}_T$ is the smooth approximation using the Sigmoid function:

$$\hat{f}_T(u) = \frac{1}{1 + \exp(-Tu)} \tag{5}$$

whose derivative can be easily written as:

$$\frac{\partial \hat{f}_T(u)}{\partial u} = T \hat{f}_T(u) \Big[1 - \hat{f}_T(u) \Big] \tag{6}$$

2.1 Notation

The derivations of these gradients involves computing derivatives with vectors. Thus, in this section we establish the appropriate notation. The derivative of a scalar y with respect to a D-dimensional vector $\mathbf{x}$ is:

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \cdots & \frac{\partial y}{\partial x_D} \end{bmatrix} \tag{7}$$

whereas the derivative of a vector $\mathbf{y}$ with respect to a scalar x is:

$$\frac{\partial \mathbf{y}}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x} \\ \frac{\partial y_2}{\partial x} \\ \vdots \\ \frac{\partial y_D}{\partial x} \end{bmatrix} \tag{8}$$

The derivative of a scalar y with respect to another scalar x, assuming $y = g(\mathbf{z})$ and $\mathbf{z} = f(x)$, can therefore be computed using the chain rule:

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial x} = \begin{bmatrix} \frac{\partial y}{\partial z_1} & \frac{\partial y}{\partial z_2} & \cdots & \frac{\partial y}{\partial z_D} \end{bmatrix} \begin{bmatrix} \frac{\partial z_1}{\partial x} \\ \frac{\partial z_2}{\partial x} \\ \vdots \\ \frac{\partial z_D}{\partial x} \end{bmatrix} = \sum_{k=1}^{D} \frac{\partial y}{\partial z_k} \cdot \frac{\partial z_k}{\partial x} \tag{9}$$

2.2 Derivations

Post-quantization Scale We notice that:

$$\frac{\partial z_k}{\partial \gamma_2} = \frac{z_k}{\gamma_2} \tag{10}$$

which can be plugged in to get the gradient using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \gamma_2} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \gamma_2} = \sum_{k=1}^{D} \frac{\partial \mathcal{L}}{\partial z_k} \cdot \frac{\partial z_k}{\partial \gamma_2} = \frac{1}{\gamma_2} \sum_{k=1}^{D} \frac{\partial \mathcal{L}}{\partial z_k} z_k \tag{11}$$

Ternary Branch Scales We first compute $\forall j \in [B]$:

$$\frac{\partial z_k}{\partial \alpha_j} = \gamma_2 \left[\sum_{i=1}^{N-1} \left[\hat{f}_T(\gamma_1 w_k - t_i) b_{i,j} \right] - 1 \right] = \gamma_2 \left[\sum_{i=1}^{N-1} \left[g_{k,i} b_{i,j} \right] - 1 \right] \tag{12}$$

where $g_{k,i} = \hat{f}_T(\gamma_1 w_k - t_i)$ for brevity. Therefore, using the chain rule we obtain:

$$\frac{\partial \mathcal{L}}{\partial \alpha_j} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \alpha_j} = \sum_{k=1}^{D} \frac{\partial \mathcal{L}}{\partial z_k} \cdot \frac{\partial z_k}{\partial \alpha_j} = \gamma_2 \sum_{k=1}^{D} \frac{\partial \mathcal{L}}{\partial z_k} \left[\sum_{i=1}^{N-1} \left[b_{i,j} g_{k,i} \right] - 1 \right] \tag{13}$$

Quantizer Thresholds We first utilize (6) in order to compute $\forall i \in [N-1]$:

$$\begin{aligned} \frac{\partial z_k}{\partial t_i} &= \gamma_2 \Big[\frac{\partial \hat{f}_T(\gamma_1 w_k - t_i)}{\partial t_i} \sum_{j=1}^{B} b_{i,j} \alpha_j \Big] \\ &= -\gamma_2 T \Big[g_{k,i}(1 - g_{k,i}) \sum_{j=1}^{B} b_{i,j} \alpha_j \Big] = -\gamma_2 T \Big[h_{k,i} \sum_{j=1}^{B} b_{i,j} \alpha_j \Big] \end{aligned} \tag{14}$$

where $h_{k,i} = g_{k,i}(1 - g_{k,i})$ for brevity. Therefore using the chain rule we obtain:

$$\frac{\partial \mathcal{L}}{\partial t_i} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial t_i} = \sum_{k=1}^{D} \frac{\partial \mathcal{L}}{\partial z_k} \cdot \frac{\partial z_k}{\partial t_i} = -\gamma_2 T \sum_{k=1}^{D} \frac{\partial \mathcal{L}}{\partial z_k} \Big[h_{k,i} \sum_{j=1}^{B} b_{i,j} \alpha_j \Big] \tag{15}$$

Pre-quantization Scale Similarly, we utilize (6) in order to compute:

$$\frac{\partial z_k}{\partial \gamma_1} = \gamma_2 \left[\sum_{i=1}^{N-1} \Big[\frac{\partial \hat{f}_T(\gamma_1 w_k - t_i)}{\partial \gamma_1} \sum_{j=1}^{B} b_{i,j} \alpha_j \Big] \right] = \gamma_2 T w_k \left[\sum_{i=1}^{N-1} \Big[h_{k,i} \sum_{j=1}^{B} b_{i,j} \alpha_j \Big] \right] \tag{16}$$

and therefore applying the chain rule yields:

$$\frac{\partial \mathcal{L}}{\partial \gamma_1} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \gamma_1} = \sum_{k=1}^{D} \frac{\partial \mathcal{L}}{\partial z_k} \cdot \frac{\partial z_k}{\partial \gamma_1} = \gamma_2 T \sum_{k=1}^{D} \frac{\partial \mathcal{L}}{\partial z_k} w_k \left[\sum_{i=1}^{N-1} \Big[h_{k,i} \sum_{j=1}^{B} b_{i,j} \alpha_j \Big] \right] \tag{17}$$

Full Precision Weights Finally, in order to compute the gradient of $\mathcal{L}$ with respect to the full precision weights $\mathbf{w} = [w_1, ..., w_D]^{\mathrm{T}}$, we first compute $\forall k \in [D]$:

$$\begin{aligned}\frac{\partial z_m}{\partial w_k} &= \gamma_2 \left[\sum_{i=1}^{N-1} \left[\frac{\partial \hat{f}_T(\gamma_1 w_m - t_i)}{\partial w_k} \sum_{j=1}^{B} b_{i,j} \alpha_j \right] \right] \\ &= \begin{cases} \gamma_1 \gamma_2 T \left[\sum_{i=1}^{N-1} \left[h_{k,i} \sum_{j=1}^{B} b_{i,j} \alpha_j \right] \right], & \text{if } m = k \\ 0, & \text{otherwise} \end{cases}\end{aligned} \tag{18}$$

and using the chain rule, we obtain:

$$\frac{\partial \mathcal{L}}{\partial w_k} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial w_k} = \sum_{m=1}^{D} \frac{\partial \mathcal{L}}{\partial z_m} \cdot \frac{\partial z_m}{\partial w_k} = \gamma_1 \gamma_2 T \frac{\partial \mathcal{L}}{\partial z_k} \sum_{i=1}^{N-1} \left[h_{k,i} \sum_{j=1}^{B} b_{i,j} \alpha_j \right] \tag{19}$$

3 MobileNetV2 on ImageNet Comparisons

We compare DBQ and [6] on MobileNetV2 in Table 4. [6] has two versions trained models M1 and M2, where M1 is trained with a memory constraint and M2 is not. We find that DBQ-2T is smaller than M2 [6] at iso-accuracy on ImageNet and more accurate than M1 [6] but at a larger storage cost. We are unable to compare the computational complexities since [6] lacks sufficient information, hence we adopt the metrics reported in [6], which are weight storage (analogous to $\mathcal{C}_M$) and activation storage (analogous to $\mathcal{C}_R - \mathcal{C}_M$).

Model	**Top-1 Acc.** [%]	**Weight Storage** [MB]	**Activation Storage** [MB]
M1 [6] (w/ constr.)	69.74	1.55	0.57
M2 [6] (w/o constr.)	70.59	3.14	1.58
DBQ-2T	**70.54**	**2.43**	**1.15**

Table 4. The Top-1 accuracy on ImageNet and Storage costs for MobileNetV2 using our method (DBQ-2T) compared to [6].

4 DBQ Branch Sparsity

One of the advantages of implementing ternary-based dot products is leveraging weight sparsity, which is reflected in our sparsity-aware computational cost $\mathcal{C}_S$. In this work, we show that for MobileNetV1 on ImageNet with two ternary branch quantization (DBQ-2T-4), the computational cost can be reduced from 2.18×10^{10} FAs to 1.42×10^{10} ($\sim 35\%$ reduction) by simply skipping the operations involving zero weights. Table 5 reports the average branch level sparsity for

			Average Branch Sparsity [%]		
PW Layer	C_{in}	C_{out}	FX8	DBQ-1T	DBQ-2T
0	64	32	35.55	58.69	64.82
1	64	128	10.74	41.42	51.75
2	128	128	6.86	34.09	46.45
3	128	256	6.73	31.83	44.96
4	256	256	4.53	29.10	43.05
5	256	512	7.31	30.62	44.36
6	512	512	6.41	28.50	43.40
7	512	512	6.00	26.48	42.94
8	512	512	4.00	24.03	41.70
9	512	512	5.57	24.89	42.56
10	512	512	5.50	23.65	42.30
11	512	1024	7.00	23.17	42.41
12	1024	1024	10.69	28.25	45.77
Network Average			7.59	26.50	**43.78**

Table 5. Branch level sparsity for all the pointwise (PW) layers of MobileNetV1 on ImageNet. C_{in} and C_{out} denote the number of input and output channels respectively.

every point wise layer. For the DBQ-2T model, which quantizes PW layers to two ternary branches, we find that on average 43.78% of all PW weights are zero, which explains the massive 35% reduction in $\mathcal{C}_S$. In contrast, the DBQ-1T model, which quantizes all PW layers to one ternary branch, achieves a 26.5% average branch sparsity. While DBQ-2T has twice the number of branches compared to DBQ-1T, the per-branch sparsity is actually much higher for the DBQ-2T. In other words, while the number of pointwise parameters increases by 2× when going from 1T to 2T, due to the high branch sparsity, the number of non-zero parameters increases by 1.53× only. On the other hand, using 8b fixed-point for the PW layers yields very little weight sparsity (7.59%).

References

1. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch SGD: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017) 3
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 3
3. Huang, G., Liu, S., Van der Maaten, L., Weinberger, K.Q.: Condensenet: An efficient densenet using learned group convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2752–2761 (2018) 3
4. Loshchilov, I., Hutter, F.: SGDR: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016) 3

5. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NIPS Autodiff Workshop (2017) 4
6. Uhlich, S., Mauch, L., Cardinaux, F., Yoshiyama, K., Garcia, J.A., Tiedemann, S., Kemp, T., Nakamura, A.: Mixed precision DNNs: All you need is a good parametrization. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=HyxOslrFvH 7