METAGRAD: ADAPTIVE GRADIENT QUANTIZATION WITH HYPERNETWORKS

Kaixin Xu[1,4]*, Alina Hui Xiu Lee*[3]*, Ziyuan Zhao*[1,2]*, Zhe Wang*[1,2,4]*, Min Wu*[1,2]*, Weisi Lin*[4]

[1]Institute for Infocomm Research, A*STAR, Singapore
[2] Artificial Intelligence, Analytics And Informatics (AI[3]), A*STAR, Singapore
[3] National University of Singapore
[4]School of Computer Science and Engineering, Nanyang Technological University, Singapore

ABSTRACT

A popular track of network compression approach is Quantization aware Training (QAT), which accelerates the forward pass during the neural network training and inference. However, not much prior efforts have been made to quantize and accelerate the backward pass during training, even though that contributes around half of the training time. This can be partly attributed to the fact that errors of low-precision gradients during backward cannot be amortized by the training objective as in the QAT setting. In this work, we propose to solve this problem by incorporating the gradients into the computation graph of the next training iteration via a hypernetwork. Various experiments on CIFAR-10 dataset with different CNN network architectures demonstrate that our hypernetwork-based approach can effectively reduce the negative effect of gradient quantization noise and successfully quantizes the gradients to INT4 with only 0.64 accuracy drop for VGG-16 on CIFAR-10.

Index Terms— Network Compression, Network Quantization, Quantization-Aware Training, Gradient Quantization, Convolution Neural Networks

1. INTRODUCTION

Quantization-aware Training (QAT) is a popular track of research that simulates the neural network quantization (weights and activations) during the course of training to curb the inference-time accuracy drop of low-bit models (*e.g.* INT8 quantization). On the other hand, theoretical and empirical analysis [1, 2] show that backpropagation accounts for more computations than forward during training. By quantizing the gradients in the backward pass, the training of QAT models can be further accelerated utilizing the low-precision compute kernels.

Despite this huge potential benefit, quantizing backward gradients is particularly difficult compared to quantizing forward variables, as backward variables are not included in the computation graph and thus cannot be optimized by the objective function, which could diverge the parameter update towards the wrong directions. In the past, a few gradient quantization works have been proposed [3, 4] that primarily achieved stable INT8 forward and backward quantization by designing gradient quantizers tailored to gradient characteristics. However, due to the abovementioned difficulties, lower than INT8 quantization is extremely under-explored, with the only early attempt DoReFa [5] ending up with poor results under INT6 gradients. This shows that designing a gradient quantizer with heuristics has already pushed to its limit.

To address the challenge of low-bit gradient quantization and curb its great performance degradation, in this paper, we propose a novel quantization approach that is adaptive to arbitrary quantization noise and best maintains the stable training direction. This meta-quantizer is realized by incorporating the hypernetwork [6], which is trained end-to-end along with the network to predict the quantized version of the gradient from a given full-precision gradient. In this regard, the low-precision gradients are now loss-aware since we incorporate them into the computation graph, resulting in a more robust QAT training. The proposed meta gradient quantizer generates higher test performance on various CNNs architectures.

2. RELATED WORKS

2.1. Quantization-aware Training (QAT)

DoReFa-Net [5] proposed to optimize the clipping value and the scaling factor of the uniform quantizers for weights and activations separately. It was validated on image classification tasks under multiple bit-widths, but only with the rather simple AlexNet architecture. Most QAT works quantize weights and activations simultaneously by optimizing the uniform quantization parameters [7, 8, 9], layer-wise or channel-wise mixed-precision quantization [10, 11], or leveraging non-uniform quantization such as Logarithmic quantizer [12]. Most recent QAT works [5, 8, 9] used "Straight-Through Estimator" (STE) [13] to estimate the gradient of the non-differentiable quantization function, while another work [14] softened the linear quantization operation in order to match the true gradient with STE.

2.2. Gradient Quantization in QAT

An earlier attempt [5] adopted a primitive quantizer design based on a uniform quantizer for gradients (without scaling and other optimization), and large performance drops were observed when training with low-bit gradients. SBM [2] adopted fixed-point 8-bit gradient quantization but only focused on improving the quantization schemes in the forward pass. WAGEUBN [15] quantized gradients to 8-bit integers but also showed a huge performance gap against its full-precision counterpart. In UINT8 [3], the authors considered the sharp and wide distribution of gradients and proposed to clip the gradients according to the deviation of the gradient distribution before quantization, achieving results on-par with full-precision training. To compensate the quantization loss on gradient, AFP [16] and CPT [17] used higher precision data to aid low-precision training. DAINT8 [4] adopted a bespoke 8-bit channel-wise gradient quantization to suppress the negative effect of quantization noise during training. Gradient quantization with less than INT8 representations remains largely unexplored. FP4 [18] managed to train modern CNN architectures using 4-bit gradients without significant accuracy loss, but the gradients were represented as floating-point numbers.

2.3. Hypernetworks

Another interesting line of work uses the concept of hypernetwork, that is first proposed in [6], to turn non-differential parts in neural networks into differentiable operations. MetaPruning [19] trains a hypernetwork to generate sparse layer weights for structured pruning of CNNs automatically. MetaQuant [20] replaces the STE [13] in the QAT training with a hypernetwork to reduce the estimation error of STE.

3. PRELIMINARIES

Quantization is a process that maps full-precision (FP32) values to a set of discrete values. Under normal gradient quantization schemes [5, 3, 4], both error signals $\nabla_{\mathbf{x}} L$ w.r.t. activation x calculated real-time during backpropagation and the gradient of weight $\nabla_{\mathbf{W}} L$ are in low-precision, which is similar to quantization scheme of forward pass. For visual simplicity, we omit the training loss L in the latter text and express gradients as $\nabla_{(\cdot)}$. We discuss integer quantization in this work. For typical gradient quantization (e.g., in [3]), symmetric uniform quantization is used. Given the gradients ∇L and a clipping value $c \in (0, \max(|\nabla L|)]$, the symmetric uniform quantizer with bit-width B can be formulated as:

$$(\nabla L)_q = Q(\nabla L) = \text{round}\left(\text{clip}(\nabla L, c) \cdot \frac{2^{B-1}-1}{c}\right), \tag{1}$$

where $\text{clip}(\nabla L, c) = \min(\max(\nabla L, -c), c)$. The quantized gradients are de-quantized as $\tilde{\nabla L} = (\nabla L)_q \cdot \frac{c}{2^{B-1}-1}$.

4. METHODOLOGIES

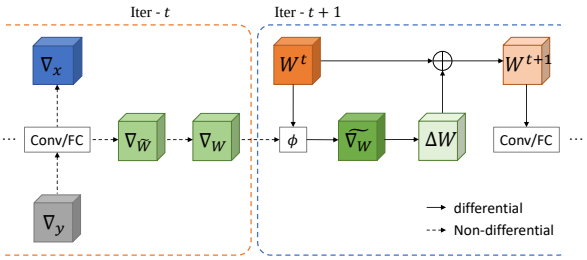

Fig. 1. Illustration of delayed updating strategy of meta-quantization of gradients.

4.1. Meta-quantizer for Gradients

We now explain in detail the design of meta-quantizer for gradients using hypernetwork. The main challenge we aim to solve here is how to incorporate the quantization operation of gradients in backpropagation into a forward computation graph so that the gradient quantizer is loss-aware. To achieve this, we first define a light-weight, differentiable neural network f_ϕ that is shared across all quantizable layers in the main model f. f_ϕ is parameterized by ϕ, and it inputs the propagated full-precision gradients or a combination of gradients and other variables (such as weight $\mathbf{W}$) and outputs the corresponding quantized gradient. The detailed structure of the hypernetwork f_ϕ will be discussed in latter sections. Inspired by MetaQuant [20], we adopt the delayed updating strategy for layer weights (see Fig. 1). Specifically, first, at the t-th iteration of training, when the backpropagation passing through a Conv/FC layer, $\nabla_{\mathbf{W}}$ is obtained by chain-rule,

$$\nabla_{\mathbf{W}} = \left[\frac{\partial \mathbf{y}}{\partial \tilde{\mathbf{W}}}\right]^\top \left[\frac{\partial \tilde{\mathbf{W}}}{\partial \mathbf{W}}\right]^\top \nabla_{\mathbf{y}}, \tag{2}$$

where $\tilde{\mathbf{W}}$ is the quantized weight in forward pass. Next, we use meta-quantizer to quantize $\nabla_{\mathbf{W}}$ before updating $\mathbf{W}$. This is realized by treating full-precision $\nabla_{\mathbf{W}}$ as input to the hypernetwork f_ϕ, which outputs the low-precision gradient $\widetilde{\nabla_{\mathbf{W}}}$, given by,

$$\widetilde{\nabla_{\mathbf{W}}} = f_\phi(\nabla_{\mathbf{W}}, \mathbf{W}). \tag{3}$$

To obtain the weight update amount $\Delta\mathbf{W}$ for iteration $t+1$, the quantized gradient is then further refined by optimization strategy π given by e.g. SGD with momentum or Adam method, and scaled by learning rate μ, given by,

$$\Delta\mathbf{W} = -\mu\pi\left(\widetilde{\nabla_{\mathbf{W}}}\right), \tag{4}$$

where the implementation of π varies according to different optimization options, which is typically differentiable for common optimizers such as SGD and Adam. Specifically, for SGD, $\pi\left(\widetilde{\nabla_{\mathbf{W}}}\right) = \widetilde{\nabla_{\mathbf{W}}}$. From Fig. 1, one can see that all the computations in between gradient $\nabla_{\mathbf{W}}$ and the weight update $\mathbf{W}^{t+1}$ are differentiable, and we successfully incorporate the

meta-quantizer into the computation graph of the main network. Hence, the complete weight update procedure during the forward pass can be written as:

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \mu\pi\left(f_\phi(\nabla_{\mathbf{W}}, \mathbf{W})\right). \tag{5}$$

4.2. Design choices of hypernetwork

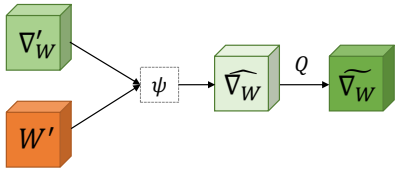

Fig. 2. General architecture of the proposed hypernetwork for meta gradient quantization.

As mentioned in Sec 4.1, in order to incorporate the gradients into the computation graph, the general design of hypernetwork f_ϕ should be a differentiable network that satisfies

$$f_\phi : \mathbb{R}^{k\times 1} \times \mathbb{R}^{k\times 1} \to \mathbf{S}^{k\times 1}, \tag{6}$$

where $\mathbf{S}$ is a finite set that satisfies

$$\mathbf{S} = \left\{ \frac{ck}{2^{B-1}-1} \mid -2^{B-1} \le k \le 2^{B-1}, k \in \mathbb{Z} \right\}. \tag{7}$$

This can be achieved by letting $f_\phi = f_\psi \circ Q$, where $f_\psi : \mathbb{R}^{N\times 1} \times \mathbb{R}^{N\times 1} \to \mathbb{R}^{N\times 1}$ and $Q : \mathbb{R}^{N\times 1} \to \mathbf{S}^{N\times 1}$ is the quantizer defined in Eq. 1. By assigning the gradient of Q using STE, f_ϕ becomes differentiable.

With this, based on [20], for FC layer, we can first transpose both the gradient $\nabla_{\mathbf{W}} \in \mathbb{R}^{n\times c}$ and the weight $\mathbf{W} \in \mathbb{R}^{(n)\times c}$ to $\mathbb{R}^{(nc)\times 1}$ tensors $\vec{\nabla_{\mathbf{W}}}$ and $\vec{\mathbf{W}}$ respectively, then input them to f_ϕ; and for Conv layer kernels, we can transpose $\mathbb{R}^{n\times c\times k\times k}$ to $\mathbb{R}^{(nck^2)\times 1}$. And for the design of f_ψ, we could simply adopts the two designs in [20]:

$$\texttt{MultiFC:} \quad f_\psi = \nabla'_{\mathbf{W}} \cdot FCs(\mathbf{W}'), \tag{8}$$

$$\texttt{LSTMFC:} \quad f_\psi = \nabla'_{\mathbf{W}} \cdot FCs(LSTM(\mathbf{W}')). \tag{9}$$

However, we found that such coordinate-wise neural network design [21] cannot make good use of the intra-layer relations of the weights and gradients. Therefore, we design another f_ψ as:

$$\texttt{DualLSTMFC:} \quad f_\psi = FCs(LSTM(\mathcal{J})), \tag{10}$$

where $\mathcal{J} = [\mathbf{W}'; \nabla'_{\mathbf{W}}] \in \mathbb{R}^{N\times 2}$ denoting a concatenation. This design can better exchange the information between weight and gradient tensors and potentially quantize the gradients in favor of the weight updating. In our empirical evaluations, we also observed a consistent improvement from `LSTMFC` design.

4.3. Training of meta-quantizer

According to Eq. 5, the gradient received by the hypernetwork f_ϕ at $t+1$-th iteration can be obtained by chain-rule:

$$\frac{\partial L}{\partial \phi^{t+1}} = \frac{\partial L}{\partial \mathbf{W}^{t+1}} \frac{\partial \mathbf{W}^{t+1}}{\partial \phi^{t+1}} = -\mu\pi' \cdot \frac{\partial L}{\partial \mathbf{W}^{t+1}} \frac{\partial f_\phi(\nabla_{\mathbf{W}}, \mathbf{W})^\top}{\partial \phi^{t+1}}. \tag{11}$$

We can further get the gradients w.r.t. ψ according to $f_\phi = f_\psi \circ Q$ by:

$$\begin{aligned} \frac{\partial L}{\partial \psi^{t+1}} &= \frac{\partial L}{\partial \phi^{t+1}} \frac{\partial \phi^{t+1}}{\partial \psi^{t+1}} \\ &\overset{\text{STE}}{\approx} = \frac{\partial L}{\partial \phi^{t+1}} = -\mu\pi' \frac{\partial L}{\partial \mathbf{W}^{t+1}} \frac{\partial f_\phi(\nabla_{\mathbf{W}}, \mathbf{W})^\top}{\partial \phi^{t+1}}, \end{aligned} \tag{12}$$

and therefore (omit superscript $t+1$):

$$\nabla_\psi = \left[\frac{\partial L}{\partial \psi}\right]^\top \approx -\mu\pi' \frac{\partial f_\phi(\nabla_{\mathbf{W}}, \mathbf{W})}{\partial \phi^{t+1}} \nabla_{\mathbf{W}}. \tag{13}$$

Since the gradients of the meta-gradient-quantizer are defined, we can train the hypernetwork together with the main model in an end-to-end manner.

5. EXPERIMENTS

5.1. Implementation Details

We first tested different hypernetwork choices, namely MultiFC, LSTMFC and DualLSTMFC on 4 bit quantization of 2 architectures, ResNet-20 and VGG-16, using CIFAR-10 dataset. We conducted the experiments for multiple CNNs architectures on CIFAR-10 dataset for INT8 and INT4 QAT training. We compare ourselves to UINT8 [3] and DAINT8 [4]. For fair comparison, we used dorefa [5] as the quantization method and `DualLSTMFC` as the hypernetwork choice. For all experiments, SGD with momentum was chosen for optimizer. For INT8 experiments, the number of hidden layers used for VGG-16 and MobileNet-V2 was 11 while for ResNet-20 and ResNet-32, 100 and 150 hidden layers were used respectively. Initial learning rate was 0.01 for ResNet-20 and 0.005 for ResNet-32, MobileNet-V2 and VGG-16, learning rate weight decay was 0.001 for ResNet-20 and 0.0001 for ResNet-32, MobileNet-V2 and VGG-16. For INT4, learning rate weight decay of 0.0001 were used for all and for VGG-16 and MobileNet-V2, the number of hidden layers used was 11 while for ResNet-20, ResNet-32 and ResNet-56, 150 hidden layers were used and for ResNet-110 100 hidden layers were used. Initial learning rate was 0.01 for ResNet-20 and 0.005 for ResNet-32, 0.001 for ResNet-56, 0.0005 for MobileNet-V2 and VGG-16 and 0.0001 for ResNet-110.

5.2. Experimental Results

INT4 Results. Tab. 1 shows the comparison of Top-1 accuracies of different CNNs on CIFAR-10 INT4 quantization. For UINT8 [3], we report the results from our re-implementation using the same weight and activation quantizer schemes which is DoReFa. Under 4-bit, it is obvious that the proposed quantization scheme achieves much less accuracy degradation than UINT8 for all models, as shown in Tab. 1. Particularly for VGG-16, we even observe a whopping 69.38 improvement from UINT8, which we suspect is due to the lack of

Model	Method	FP32 /%	Acc /%	Δ /%
ResNet-20	UINT8	91.36	73.45	-17.91
	Ours	91.36	**91.2**	**-0.16**
ResNet-32	UINT8	91.42	63.46	-27.96
	Ours	91.42	**90.56**	**-0.86**
ResNet-56	UINT8	91.95	65.27	-26.68
	Ours	91.95	**90.24**	**-1.71**
ResNet-110	UINT8	91.67	62.57	-29.10
	Ours	91.67	**89.16**	**-2.51**
MobileNet-V2	UINT8	93.57	88.12	-5.45
	Ours	93.57	**89.00**	**-4.57**
VGG-16	UINT8	92.25	22.23	-70.02
	Ours	92.25	**91.61**	**-0.64**

Table 1. Comparison with other methods under INT4 quantization. Δ denotes the performance drop from full-precision (FP32) results: $\Delta = \text{Acc} - \text{FP}$ (the higher the better).

residual structures in the VGG-16 model that exacerbates the negative effects of gradient quantization error. Our method consistently supports stabilized gradient quantization training, where the degradation from FP32 across all models is at most 4.57.

INT8 Results. Tab. 2 shows the comparison of Top-1 accuracies of different CNNs on CIFAR-10 INT8 quantization. We observe that under 8-bit, we perform the best compared to both state-of-the-art gradient quantization [3, 4] under almost all examined models, especially for VGG-16, where the baseline UINT8 simply fail to converge, while our quantization scheme successfully maintains the performance drop with FP to only -0.05. The only exception is ResNet-20, where we perform slightly worse than DAINT8 [4], but this is expected since DAINT8 uses the more fine-grained channel-wise quantization while ours and UINT8 [3] use the de-facto simple layer-wise quantization. Yet, we still have a lower performance drop from FP32 than DAINT8 on MobileNet-V2, revealing the great potential of our approach that obtains performance gain only by incorporating the gradients into the objective function.

5.3. Ablation Studies and Other Discussions

We conduct an ablation study for the choice of hypernetwork design on CIFAR-10 with two models, ResNet-20 and VGG-16 under INT4 quantization. Tab. 3 shows a clear pattern of `DualLSTMFC` > `LSTMFC` > `MultiFC` for both models, indicating that our newly proposed `DualLSTMFC` design can more effectively generate quantized gradients in favor of the weight updating, resulting in the lowest accuracy degradation. Yet, the overall performance discrepancy among them is not large, indicating that the choice of hypernetwork only marginally affects the effectiveness of the proposed method.

Model	Method	FP32 /%	Acc /%	Δ /%
ResNet-20	UINT8	91.36	91.10	-0.26
	DAINT8	92.35	**92.76**	**0.41**
	Ours	91.36	91.30	-0.06
ResNet-32	UINT8	91.42	90.70	-0.72
	Ours	91.42	**91.40**	**-0.02**
MobileNet-V2	UINT8	93.57	93.38	-0.19
	DAINT8	94.73	**94.37**	-0.36
	Ours	93.57	93.51	**-0.06**
VGG-16	UINT8	92.25	31.32	-60.93
	Ours	92.25	**92.20**	**-0.05**

Table 2. Comparison with other methods under INT8 quantization.

Model	Hypernetwork	FP32	Acc /%	Δ /%
ResNet-20	`MultiFC`	91.36	91.06	-0.30
	`LSTMFC`	91.36	91.15	-0.21
	`DualLSTMFC`	91.36	**91.20**	**-0.16**
VGG-16	`MultiFC`	92.25	91.50	-0.75
	`LSTMFC`	92.25	91.54	-0.71
	`DualLSTMFC`	92.25	**91.61**	**-0.64**

Table 3. Ablation study of hypernetwork design under INT4 quantization.

6. CONCLUSIONS

In this work, to improve the stability of gradient quantization during QAT, we presented a novel meta-quantizer for gradient quantization that is able to automatically optimize the quantizer throughout training and thus more robust against quantization noise produced in the backward pass. Extensive experiments on the CIFAR-10 dataset with various CNNs demonstrate that the meta-quantizer helps the QAT training stabilize under lower gradient bit-rates, which vastly outperforms the baselines, especially under extremely low bit-widths.

Acknowledgement

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its Funds (Project Number A1892b0026, and C211118009). Any opinions, findings and conclusions or recommendations ex- pressed in this material are those of the author(s) and do not reflect the views of the A*STAR.

7. REFERENCES

[1] Jcjohnson, "Cnn-benchmarks: Benchmarks for popular cnn models," 2016.

[2] Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry, "Scalable methods for 8-bit training of neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[3] Feng Zhu, Ruihao Gong, Fengwei Yu, Xianglong Liu, Yanfei Wang, Zhelong Li, Xiuqi Yang, and Junjie Yan, "Towards unified int8 training for convolutional neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1969–1979.

[4] Kang Zhao, Sida Huang, Pan Pan, Yinghan Li, Yingya Zhang, Zhenyu Gu, and Yinghui Xu, "Distribution adaptive int8 quantization for training cnns," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 3483–3491.

[5] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[6] David Ha, Andrew M. Dai, and Quoc V. Le, "Hypernetworks," in *International Conference on Learning Representations*, 2017.

[7] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 365–382.

[8] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha, "Learned step size quantization," in *ICLR*, 2020.

[9] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak, "Lsq+: Improving low-bit quantization through learnable offsets and better initialization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 696–697.

[10] Qing Jin, Linjie Yang, and Zhenyu Liao, "Adabits: Neural network quantization with adaptive bit-widths," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[11] Qian Lou, Feng Guo, Minje Kim, Lantao Liu, and Lei Jiang, "Autoq: Automated kernel-wise neural network quantizations," in *ICLR*, 2020.

[12] Daisuke Miyashita, Edward H Lee, and Boris Murmann, "Convolutional neural networks using logarithmic data representation," *arXiv preprint arXiv:1603.01025*, 2016.

[13] Yoshua Bengio, Nicholas Léonard, and Aaron Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv:1308.3432*, 2013.

[14] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4852–4861.

[15] Yukuan Yang, Lei Deng, Shuang Wu, Tianyi Yan, Yuan Xie, and Guoqi Li, "Training high-performance and large-scale deep neural networks with full 8-bit integers," *Neural Networks*, vol. 125, pp. 70–82, 2020.

[16] Xishan Zhang, Shaoli Liu, Rui Zhang, Chang Liu, Di Huang, Shiyi Zhou, Jiaming Guo, Qi Guo, Zidong Du, Tian Zhi, et al., "Fixed-point back-propagation training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2330–2338.

[17] Yonggan Fu, Han Guo, Meng Li, Xin Yang, Yining Ding, Vikas Chandra, and Yingyan Lin, "Cpt: Efficient deep neural network training via cyclic precision," *arXiv preprint arXiv:2101.09868*, 2021.

[18] X Sun, N Wang, C Chen, J Ni, A Agrawal, X Cui, S Venka, K El, V Srini, and K Gopa, "Ultra-low precision 4-bit training of deep neural networks," in *NeurIPS*, 2020.

[19] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun, "Metapruning: Meta learning for automatic neural network channel pruning," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3296–3305.

[20] Shangyu Chen, Wenya Wang, and Sinno Jialin Pan, "Metaquant: Learning to quantize by learning to penetrate non-differentiable quantization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[21] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas, "Learning to learn by gradient descent by gradient descent," *Advances in neural information processing systems*, vol. 29, 2016.