

Progressive Element-wise Gradient Estimation for Neural Network Quantization

Kaiqi Zhao
Oakland University
kaiqizhao@oakland.edu

Abstract

Neural network quantization aims to reduce the bit-widths of weights and activations, making it a critical technique for deploying deep neural networks on resource-constrained hardware. Most Quantization-Aware Training (QAT) methods rely on the Straight-Through Estimator (STE) to address the non-differentiability of discretization functions by replacing their derivatives with that of the identity function. While effective, STE overlooks discretization errors between continuous and quantized values, which can lead to accuracy degradation — especially at extremely low bit-widths. In this paper, we propose Progressive Element-wise Gradient Estimation (PEGE), a simple yet effective alternative to STE, which can be seamlessly integrated with any forward propagation methods and improves the quantized model accuracy. PEGE progressively replaces full-precision weights and activations with their quantized counterparts via a novel logarithmic curriculum-driven mixed-precision replacement strategy. Then it formulates QAT as a co-optimization problem that simultaneously minimizes the task loss for prediction and the discretization error for quantization, providing a unified and generalizable framework. Extensive experiments on CIFAR-10 and ImageNet across various architectures (e.g., ResNet, VGG) demonstrate that PEGE consistently outperforms existing backpropagation methods and enables low-precision models to match or even outperform the accuracy of their full-precision counterparts.

1. Introduction

Deep neural networks (DNNs) have substantial computational and memory requirements. As the use of deep learning grows rapidly on a wide variety of Internet of Things and devices, the mismatch between resource-hungry DNNs and resource-constrained devices also becomes increasingly severe [13, 14]. Quantization is one of the important model compression approaches to address this challenge by converting the full-precision model weights or activations to lower precision. In particular, Quantization-Aware Training (QAT) [7] has achieved promising results in creating low-bit models, which starts with a pre-trained model and performs quantization during retraining. However, existing QAT methods often suffer from noticeable accuracy degra-

dation, especially when targeting extremely low-bit precision. Moreover, no algorithm achieves consistent performance on every model architecture (e.g., VGG, ResNet, MobileNet) [11].

Most QAT approaches rely on the Straight-Through Estimator (STE) [1] to address the fundamental challenge of non-differentiability in the discretization step. Specifically, the round function used in quantization has zero or undefined derivatives, which leads to vanishing or exploding gradients during backpropagation. STE circumvents this by approximating the gradient as an identity function, enabling stable training. Despite its simplicity and widespread adoption, STE overlooks discretization errors between latent and discrete values, which become significant at low bit-widths, leading to substantial accuracy loss. To mitigate this issue, Element-Wise Gradient Scaling (EWGS) [8] was proposed to minimize the discretization error during backpropagation by scaling gradients according to the difference between continuous and discrete values. Although EWGS demonstrates improved results, its effectiveness remains tightly coupled with its specific forward propagation method (see Section 4.1 for detailed discussions), limiting its flexibility and generalizability. Therefore, we argue that there is a critical need for a simple, effective, and generalizable gradient estimation method that enhances training stability, mitigates accuracy degradation from reduced precision, and seamlessly integrates with any forward method to consistently improve overall quantization performance.

In this paper, we propose Progressive Element-wise Gradient Estimation (PEGE), a novel backward propagation method for QAT. PEGE first unifies the forward dynamics of diverse quantizers and progressively replaces full-precision weights with their low-precision counterparts during training. The replacement process is controlled by a novel logarithmic Curriculum Learning driven strategy which updates replacing rates with a dynamic, smoothing mechanism. PEGE then minimizes the discretization error to ensure that estimated gradients closely approximate those of the full-precision model. PEGE further formulates QAT as a co-optimization problem that simultaneously minimizes the task loss and the discretization error between full-precision weights/activations and their quantized counterparts, offering a unified and theoretically grounded perspective on quantization training.

Our comprehensive evaluation shows that PEGE sig-

nificantly outperforms the state-of-the-art backpropagation methods on widely used model architectures (VGG and ResNet) across CIFAR-10 and ImageNet datasets. Notably, PEGE enables low-precision models to match or even surpass the accuracy of their full-precision counterparts. For example, on ResNet-20 with CIFAR-10, PEGE converges faster and achieves higher accuracy than EWGS and STE by 0.97% and 0.45%, respectively. On ImageNet, PEGE improves the full-precision model’s accuracy by 0.21%, whereas STE reduces accuracy by 0.5% to 1.8%, and EWGS results in a 0.27% accuracy drop.

2. Background and Related Works

There are two main approaches to quantization. Post-Training Quantization (PTQ) [10] quantizes a pre-trained model without retraining and often leads to worse accuracy degradation than QAT [7], which performs quantization during retraining. This paper focuses on QAT. Many QAT works emphasize designing the forward and backward propagation of the quantizer, the function that converts continuous weights or activations to discrete values. Early works such as BNN [3] and XNOR-Net [12] employ channel-wise scaling in the forward pass, while DoReFa-Net [15] introduces a universal scalar for all filters. Recent works utilize trainable parameters for the quantizer, improving control over facets such as clipping ranges (e.g., PACT [2], LSQ [4], APoT [9], and DSQ [5]), i.e., the bounds where the input values are constrained, and quantization intervals (e.g., QIL [6] and EWGS [8]), i.e., the step size between adjacent quantization levels. However, existing QAT methods lead to different levels of accuracy loss and are motivated by various heuristics, lacking a commonly agreed theory. Furthermore, MQbench [11] reveals that the difference between QAT algorithms is not as substantial as reported in their original papers; and no algorithm achieves the best performance on all architectures. Therefore, this paper aims to close this gap, providing a simple, effective, and generalizable gradient estimation method that seamlessly integrates with any forward method and consistently improves overall quantization performance across both low-bit and high-bit precisions.

3. Methodology

3.1. Forward Propagation

Let us define $Quant(\cdot)$ as a uniform quantizer that converts a full-precision input x to a quantized output $x_q = Quant(x)$. x can be the activations or weights of the network. First, the quantizer $Quant(\cdot)$ applies a clipping function $Clip(\cdot)$, which normalizes and restricts the full-precision input x to a limited range, producing a full-precision latent presentation x_c , as follows:

$$x_c = Clip(x, \{p_i\}_{i=1}^{i=K_c}, v, m), \quad (1)$$

where v and m are the lower and upper bounds of the range, respectively, $\{p_i\}_{i=1}^{i=K_c}$ denotes the set of trainable parameters needed for quantization, and K_c denotes the number

of parameters. Note that different quantizers have different schemes for $Clip(\cdot)$. For example, in PACT, the lower bound v is set to 0 and the upper bound m is a trainable parameter optimized during training. The quantizer requires only one parameter, that is, $\{p_1|p_1 = m, K_c = 1\}$, and the clipping function is described as: $x_c = Clip(x, \{p_1|p_1 = m\}, 0, m) = 0.5(|x| - |x - m| + m)$. In EWGS, v and m are set to 0 and 1, respectively, and every quantized layer uses separate parameters (i.e. p_1 and p_2) for quantization intervals: $x_c = Clip(x, \{p_1, p_2\}, 0, 1) = clip(\frac{x-p_1}{p_1-p_2}, 0, 1)$.

Then, the quantizer $Quant(\cdot)$ converts the clipped value x_c to a discrete quantization point x_q using the function $R(\cdot)$ that contains a round function:

$$x_q = R(x_c, b, \{q_i\}_{i=1}^{i=K_r}), \quad (2)$$

where b is the bit width and $\{q_i\}_{i=1}^{i=K_r}$ denotes the set of trainable parameters. Note that $\{q_i\}_{i=1}^{i=K_r}$ is not necessary for some quantizers. For example, in EWGS, if activations are the input, $x_q = R(x_c, b) = \frac{round((2^b-1) \cdot x_c)}{2^b-1}$; and if weights are the input, $x_q = R(x_c, b) = 2(\frac{round((2^b-1) \cdot x_c)}{2^b-1} - 0.5)$. In some quantizers, like PACT and LSQ, the trainable parameters in the function $R(\cdot)$ are the same as those in the clipping function $Clip(\cdot)$, that is, $\{q_i|q_i = p_i\}_{i=1}^{i=K_r}$ and $K_r = K_c$.

In summary, the quantizer $Quant(\cdot)$ is described as: $x_q = Quant(x, \alpha, b, v, m)$, where α denotes a shorthand for the set of all the parameters in the functions $R(\cdot)$ and $Clip(\cdot)$: $\alpha = \{\{p_i\}_{i=1}^{i=K_c}, \{q_i\}_{i=1}^{i=K_r}\}$.

3.2. Backward Propagation

Directly training a quantized network using backpropagation is impossible since the quantizer $Q(\cdot)$ is non-differentiable. This issue arises due to the round function in Eq. 2, which produces near-zero derivatives almost everywhere. To solve this problem, most QAT works use Straight-Through Estimator (STE) [1] to approximate the gradients: $\frac{\partial L}{\partial x_c} = \frac{\partial L}{\partial x_q}$. Instead of the commonly used STE for backpropagation, PEGE uses two core strategies, including a logarithmic curriculum-driven progressive precision replacement, and an adaptive gradient correction based on discretization error.

Progressive Precision Replacement. PEGE stochastically mixes different full-precision and quantized weights/activations in each backpropagation step. Consider backpropagation step \mathcal{T} , we generate an independent Bernoulli random variables, denoted as $r_{\mathcal{T}}$, which is either equal to 1 or 0:

$$r_{\mathcal{T}} \sim Bernoulli(p) \quad (3)$$

where $0 < p \leq 1$, and $r_{\mathcal{T}} \in \{0, 1\}$. The replacing rate p controls the probability of $r_{\mathcal{T}}$ being 1. When $r_{\mathcal{T}}$ is equal to 1, the updated parameters come from the quantized parameters, i.e., all the full-precision parameters are replaced. This mechanism enables quantized weights/activations to progressively receive guidance from their full-precision counterparts through gradient-level interaction.

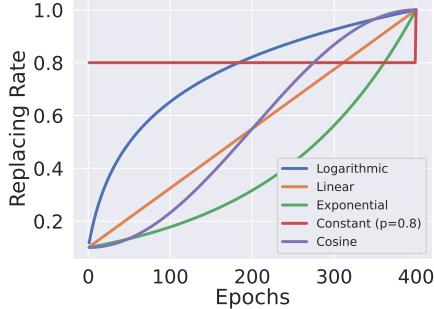


Figure 1. The replacing curves of five replacement schedulers.

We propose a novel logarithmic curriculum replacement strategy to update the replacement rate. Let \mathcal{T} denote the \mathcal{T} -th iteration/training step, so that the replacing rate $p_{\mathcal{T}}$ at iteration \mathcal{T} can be calculated as:

$$p_{\mathcal{T}} = \min(\log_B^{k\mathcal{T}+b}, 1.0) \quad (4)$$

where B is the base value of the logarithmic function, \log_B^b is the basic replacing rate, and k is the coefficient. Figure 1 illustrates the replacing curves of Constant, Linear, Logarithmic, and Exponential Replacement Schedulers.

Adaptive Gradient Estimation. After progressive replacement, PEGE introduces a novel formula to integrate the discretization error $(x_c - x_q)$, representing the deviation between full-precision and its quantized weights/activations:

$$\frac{\partial L}{\partial x_c} = \frac{\partial L}{\partial x_q} + \mu \cdot (x_c - x_q), \quad (5)$$

where $\mu \geq 0$ is a scaling factor that modulates the influence of the discretization error. To align this modulation with training dynamics, PEGE uses an Exponential Curriculum Learning driven strategy to update μ : $\mu_{\mathcal{T}} = \mu_{\max} \cdot (1 - e^{-k\mathcal{T}})$, with μ_{\max} as the maximum value, k controlling the rate of growth, and \mathcal{T} as the iteration step.

3.3. Optimization Objective

We frame QAT as an optimization problem minimizing both the discretization error as well as the discrepancy between model predictions and true labels. The goal is precise quantization without sacrificing predictive accuracy. Thus, the optimization objection is defined as follows:

$$\begin{aligned} & \min_{W_f, \alpha_W, \alpha_A} L(W_f) \\ \text{s.t. } & W_q = \text{Quant}_W(W_f, \alpha_W, b_W, v_W, m_W) \\ & A_q = \text{Quant}_A(A_f, \alpha_A, b_A, v_A, m_A), \end{aligned} \quad (6)$$

where $L(\cdot)$ is the cross entropy loss with labels, $\text{Quant}_W(\cdot)$ and $\text{Quant}_A(\cdot)$ are the quantizers for weights and activations, and W_f/A_f and W_q/A_q are the model's full-precision and quantized weights/activations.

To the best of our knowledge, we are the first to generalize diverse quantizers, encompassing forward and backward propagations into a unified formulation of an optimization problem. This generalization and formulation enable our

backward method to be flexible in integrating various forward methods and effectively improve the overall quantization performance (see Section 4.1 for the results).

4. Evaluation

Models and Datasets. We evaluate our proposed method on a variety of image classification models and datasets, including 1) ResNet-20 and VGG-16 on CIFAR-10 which contains 50K (32×32) RGB training images, belonging to 10 classes, and 2) ResNet-18 on ImageNet containing about 1.4M training images and 50K validation images, with 1000 categories.

Implementation Details. We implement all the techniques on PyTorch version 1.10.0 and Python version 3.9.7, and conducted experiments on eight Nvidia V100 GPUs. The learning rate decays to 0.0 using a cosine annealing schedule. Nesterov SGD optimizer is used with a momentum of 0.9 on ImageNet, and Adam optimizer is used for CIFAR-10, respectively. The batch size is set to 64 for CIFAR-10 and to 256 for ImageNet.

4.1. Results

Results on CIFAR-10. Table 1 presents the top-1 test accuracy of ResNet-20 and VGG-16 models on CIFAR-10, where activations and weights are quantized to 2 bits using the same forward method, i.e., EWGS, with different backward methods, including STE [1], EWGS [8], and the proposed PEGE.

Our method significantly outperforms 1) EWGS by 0.97% on ResNet-20 and by 0.11% on VGG-16, and 2) STE by 0.45% on ResNet-20 and by 0.21% on VGG-16, respectively. In addition, our method enables a 2-bit model to achieve a comparable accuracy (91.62% vs 91.95% on ResNet-20 and 93.73% vs 93.84% on VGG-16) with its full-precision model with a compression ratio of 16 \times (quantized from 32 bits to 2 bits). *To our best knowledge, the 2-bit ResNet-20 and 2-bit VGG-16 quantized by our method achieve the state-of-the-art top-1 test accuracy.*

Figure 2a illustrates the top-1 test accuracy evolution for quantized ResNet-20 in each epoch during training on CIFAR-10. Compared to EWGS and STE, PEGE accelerates the convergence speed of 2-bit ResNet-20.

Results on ImageNet. As shown in Table 2, our method is the only one that enables a small, 4-bit ResNet-18 to surpass the top-1 test accuracy of its full-precision counterpart on ImageNet. Specifically, our method improves the full-precision model's accuracy by 0.21%, whereas STE reduces accuracy by 0.5% to 1.8%, and EWGS results in a 0.27% accuracy drop. Moreover, PEGE significantly outperforms EWGS and STE in top-1 test accuracy for 4-bit quantization by 0.48% and 0.45%, respectively, using PACT as the forward method. Furthermore, although PACT initializes the low-precision model with a much better full-precision model in its original paper (71.0% vs. 69.8%), its final accuracy (69.2%) remains lower than ours (70.01%).

Table 1. Top-1 test accuracy (%) on CIFAR-10. “W*A \times ” denotes that the weights and activations are quantized into *bit and \times bit, respectively.

Bit-Width	Model	Forward Method	Backward Method	Accuracy	Accuracy Difference
W2A2	ResNet-20	Full Precision (FP)	Full Precision (FP)	91.95	-
			EWGS	90.65	-1.30
		STE	91.17	-0.78	
		Ours	91.62	-0.33	
	VGG-16	Full Precision (FP)	93.84	-	
			EWGS	93.59	-0.25
		STE	93.52	-0.32	
		Ours	93.73	-0.11	

Table 2. Top-1 test accuracy (%) on ImageNet. “STE (Orig. with FP: 71)” denotes the accuracy cited from their original paper [2] where the full precision model’s accuracy is 71%.

Bit-Width	Model	Forward Method	Backward Method	Accuracy	Accuracy Difference
W4A4	ResNet-18	Full Precision (FP)	Full Precision (FP)	69.80	-
			STE (Orig. with FP: 71)	69.20	-1.80
		PACT	STE	69.56	-0.24
			EWGS	69.53	-0.27
			Ours	70.01	+0.21

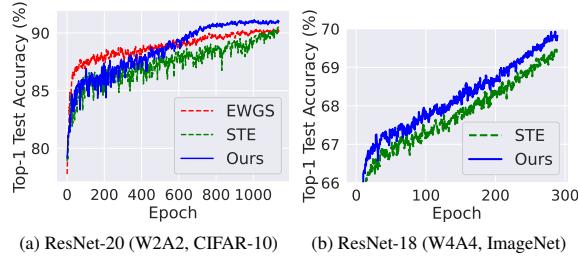


Figure 2. Top-1 test accuracy evolution of a) ResNet-20 (W2A2) on CIFAR-10 and 2) ResNet-18 (W4A4) on ImageNet during training, using different backward methods with the same forward methods (EWGS for CIFAR-10 and PACT for ImageNet).

Figure 2b demonstrates that PEGE enables 4-bit ResNet-20 to converge significantly faster than STE on ImageNet. Notably, PACT originally employs STE as its backward method. *These results confirm that simply replacing the commonly used STE with our method can substantially improve both the convergence speed and final accuracy of the whole quantization.*

4.2. Ablation Study

Effect of Training Time. As shown in Table 3, both our method and STE benefit from a longer training time, whereas EWGS does not show such improvement. Specifically, training 3 \times longer increases the top-1 test accuracy of our method and STE by 0.57% and 0.82%, respectively, on 2-bit ResNet-20 with CIFAR-10. In contrast, extending the training time for EWGS leads to a 0.19% accuracy drop.

Effect of Precision Replacement Scheduler. Table 4 shows the top-1 test accuracy of 2-bit ResNet-20 trained using constant, linear, exponential, cosine, and logarithmic replacement schedulers as well as a basic baseline without

Table 3. Top-1 test accuracy (%) of different training epochs on CIFAR-10. The forward method is EWGS.

Model	Training Epochs	STE	EWGS	Ours
ResNet-20 (FP: 91.95)	400	90.35	90.84	91.05
	1200	91.17 (0.82 \uparrow)	90.65 (0.19 \downarrow)	91.62 (0.57 \uparrow)

Table 4. Top-1 test accuracy (%) of different precision replacement schedulers on CIFAR-10.

Model	None	Linear	Constant (p=0.8)	Exponential	Cosine	Logarithmic
ResNet-20 (FP: 91.95)	90.35	90.54	89.82	90.32	89.54	90.9

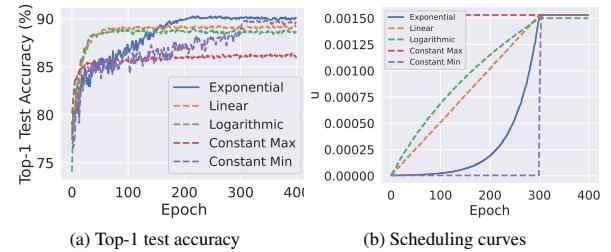


Figure 3. Illustration of a) top-1 test accuracy evolution of ResNet-20 (W2A2) on CIFAR-10, using five scheduling strategies including constant, linear, logarithmic, and exponential strategies and b) the corresponding scheduling curves for each strategy.

any scheduler, on CIFAR-10. Figure 1 illustrates the replacing curves of each scheduler. The proposed log-curriculum based replacement scheduler achieves the highest accuracy, outperforming the none-scheduler baseline by 0.55% and the other four schedulers by 0.36% to 1.36%.

Effect of μ Scheduler. Figure 3a shows the top-1 test accuracy evolution of ResNet-20 quantized to 2-bit weights and activations on CIFAR-10, trained with different μ scheduling strategies, including constant, linear, logarithmic, and exponential strategies. Figure 3b shows the corresponding value changes for each strategy. Exponential μ Scheduler achieves the best accuracy, aligning with the behavior commonly observed in Alternating Direction Method of Multipliers (ADMM)-based optimization.

5. Conclusions

This paper proposes Progressive Element-wise Gradient Estimation (PEGE), a novel and generalizable backward propagation method for Quantization-Aware Training (QAT). PEGE introduces a logarithmic curriculum-based mixed-precision replacement strategy and an adaptive gradient estimation mechanism, allowing for more effective learning of quantized models. PEGE is simple to implement, compatible with diverse forward quantizers, and introduces negligible computational overhead, making it highly practical and scalable. We comprehensively evaluate PEGE across multiple datasets and architectures, showing that it consistently outperforms state-of-the-art backpropagation methods in QAT, and even enables low-precision models to match or exceed the performance of their full-precision counterparts.

References

- [1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. [1](#), [2](#), [3](#)
- [2] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018. [2](#), [4](#)
- [3] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016. [2](#)
- [4] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019. [2](#)
- [5] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazen 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*, pages 4852–4861, 2019. [2](#)
- [6] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4350–4359, 2019. [2](#)
- [7] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018. [1](#), [2](#)
- [8] Junghyup Lee, Dohyung Kim, and Bumsub Ham. Network quantization with element-wise gradient scaling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6448–6457, 2021. [1](#), [2](#), [3](#)
- [9] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *arXiv preprint arXiv:1909.13144*, 2019. [2](#)
- [10] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021. [2](#)
- [11] 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. *arXiv preprint arXiv:2111.03759*, 2021. [1](#), [2](#)
- [12] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016. [2](#)
- [13] Zhaofeng Zhang, Sheng Yue, and Junshan Zhang. Towards resource-efficient edge ai: From federated learning to semi-supervised model personalization. *IEEE Transactions on Mobile Computing*, 2023. [1](#)
- [14] Kaiqi Zhao, Animesh Jain, and Ming Zhao. Automatic attention pruning: Improving and automating model pruning using attentions. In *International Conference on Artificial Intelligence and Statistics*, pages 10470–10486. PMLR, 2023. [1](#)
- [15] 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. [2](#)