LowGradQ: Adaptive Gradient Quantization for Low-Bit CNN Training via Kernel Density Estimation-Guided Thresholding and Hardware-Efficient Stochastic Rounding Unit

Sangbeom Jeong, Seungil Lee, and Hyun Kim
Department of Electrical and Information Engineering and Research Center for Electrical and Information Technology
Seoul National University of Science and Technology, Seoul 01811, Korea
sangbeom@seoultech.ac.kr, seungil66@seoultech.ac.kr, hyunkim@seoultech.ac.kr

***Abstract*—This paper proposes a hardware-efficient INT8 training framework with dual-scale adaptive gradient quantization (DAGQ) to cope with the growing need for efficient on-device CNN training. DAGQ captures both small- and large-magnitude gradients, ensuring robust low-bit training with minimal quantization error. Additionally, to reduce the computational and memory demands of stochastic rounding in low-bit training, we introduce a reusable LFSR-based stochastic rounding unit (RLSRU), which efficiently generates and reuses random numbers, minimizing hardware complexity. The proposed framework achieves stable INT8 training across various networks with minimal accuracy loss while being implementable on RTL-based hardware accelerators, making it well-suited for resource-constrained environments.**

***Index Terms*—CNN, low-bit training, gradient quantization**

I. Introduction

Convolutional neural networks (CNNs) have become essential in computer vision tasks [1], [2], driving widespread adoption across industries. In particular, with the growing demand for personalized models and privacy protection, on-device CNN training on resource-constrained edge devices has gained attention. To enable efficient training on edge devices, gradient quantization must be employed [3], [4]. However, gradients pose considerable challenges for quantization owing to the prevalence of extreme outliers and the complex structure of the loss space. Existing gradient quantization methods involve channel-wise quantization, which requires numerous scale factors or repeated use of the cosine distance, ultimately resulting in high computational complexity. To address these challenges, we propose a hardware (HW)-efficient INT8 training framework for CNNs. Our main contributions are as follows:

- We leverage kernel density estimation (KDE) to conduct a more precise analysis of gradient distribution, clearly defining the characteristics of CNN gradients.
- We propose a dual-scale adaptive gradient quantization (DAGQ) method with INT8 precision, which effectively handles extreme gradient distributions by distinguishing small- and large-magnitude gradients through optimal thresholding using a subset of gradients.
- To enable HW-friendly stochastic rounding (SR) for gradient quantization, we propose a reusable linear feedback shift register (LFSR)-based SR unit (RLSRU), which enhances random number generation efficiency through HW-efficient operations and random number reuse.

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2020-0-01304, Development of Self-learnable Mobile Recursive Neural Network Processor Technology) and National R&D Program through the National Research Foundation of Korea(NRF) funded by Ministry of Science and ICT(2020M3H2A1078119). Corresponding author: Hyun Kim.

II. Proposed Methods

A. Overall Framework of the Proposed Gradient Quantization

Fig.1 outlines the proposed gradient quantization framework. During backward propagation (BP), the optimal threshold search (OTS) identifies the optimal threshold for each layer. The OTS calculates the standard deviation (std) of the layer's gradients and uses the magnitude factor (mf) from prior KDE analysis to define a threshold search range ($\text{th}_1 \sim \text{th}_N$ in Fig. 1), distinguishing small and large gradients. Mean squared error (MSE) is computed solely for positive large-magnitude gradients exceeding each candidate threshold, and the threshold with the minimum MSE is selected as the optimal threshold. Quantization is then applied to all gradients using this threshold, with random numbers for SR generated by the RLSRU.

B. Dual-Scale Adaptive Gradient Quantization

Gradient histograms typically reveal that most CNN gradients cluster near zero but span a wide range. Maximum absolute value-based clipping, commonly used in weight and activation quantization, distorts the scale factor, causing small gradients to quantize to zero and significant information loss. However, clipping large gradients as outliers also harms training, requiring tailored quantization strategies for small and large gradients. To overcome histogram limitations, we use KDE for precise gradient distribution analysis, as it provides reliable, continuous density estimates independent of bin size.

Building on this, we propose DAGQ, an INT8 training method that divides gradients into small- and large-magnitude regions using a threshold, applying different scale factors for quantization. This approach captures gradient characteristics

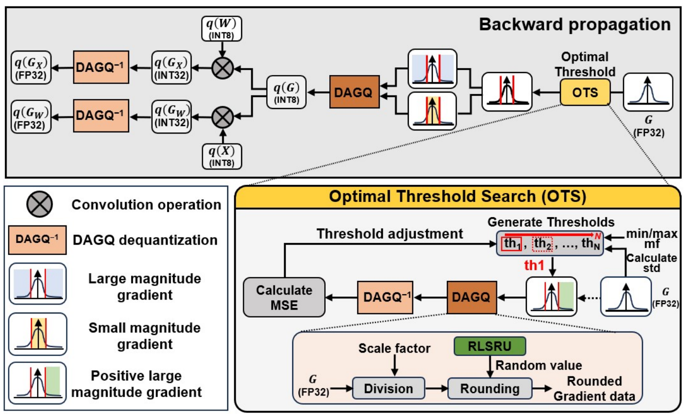

Fig. 1: Dataflow of gradient distribution-aware INT8 quantization in the backward propagation.

effectively while maintaining uniform quantization, simplifying HW design. To optimize the threshold, which is critical for quantization performance and training stability, we design an OTS algorithm using MSE as the metric. Since large gradients have a greater impact on training and the gradient distribution is nearly symmetrical, MSE is calculated only for positive gradients above the threshold. The search range is defined by the gradient's std and the mf range from KDE analysis. Potential thresholds are uniformly generated within this range based on a search precision hyperparameter. Gradients above each candidate threshold are quantized, and MSE is computed. Finally, the threshold minimizing MSE is selected for quantization across all gradients. This OTS algorithm reduces HW overhead by processing a subset of gradients, achieving low quantization error and faster network convergence.

C. LFSR-Based Efficient Stochastic Rounding Unit

Fig. 2 illustrates the full process through which the proposed RLSRU generates the random numbers needed for SR. The large seed generator utilizes a 4-bit LFSR to produce a large seed (ls), which is passed to the small seed generator. The small seed generator generates a small seed (ss) in the number of batch units based on the received ls. These ss values serve as the initial values for the 4-bit LFSR within the LFSR unit set. Depending on the user-defined batch and channel units, the LFSR unit set generated unit random numbers ($urns$). To significantly reduce the number of random numbers generated by the LFSR unit set, the generated $urns$ are then rearranged at the batch and channel levels to produce the final random numbers required for network training.

III. Experiment Results

Table I shows comparative experiments with SOTA INT8 gradient quantization methods [3]–[5] using ResNet [1] and MobileNetV2 [2]. Ours refers to the DAGQ method using the random number generation library provided by the PyTorch framework, and Ours* refers to the DAGQ method using RLSRU. Ours exhibited excellent performance, achieving a 0.51% improvement over FP32 in ResNet50. Notably, despite challenges in MobileNetV2's sparsely connected depthwise convolution structure, the proposed method achieved a 0.24% performance improvement, demonstrating its versatility.

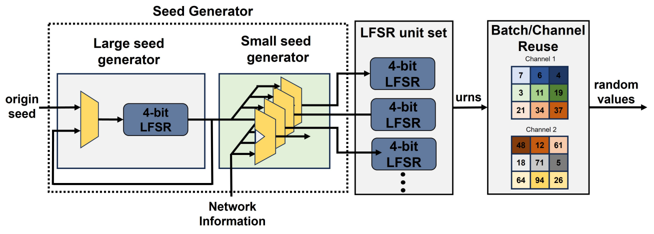

Fig. 2: Overview of the proposed RLSRU framework.

TABLE I: Top-1 accuracies on the ImageNet dataset

Model	Method	FP32 (%)	INT8 (%)	ACC drop(%)
ResNet18	DAI8 [3]	70.22	70.21	-0.01
	SRU-Q [5]	69.52	69.04	-0.48
	ESRU [4]	71.10	70.91	-0.19
	Ours	69.76	69.94	**+0.18**
	Ours*	69.76	69.50	-0.26
ResNet34	DAI8 [3]	73.46	73.40	-0.06
	Ours	73.30	73.79	**+0.49**
	Ours*	73.30	73.11	-0.19
ResNet50	DAI8 [3]	76.50	76.59	+0.09
	ESRU [4]	77.59	77.56	-0.03
	Ours	76.14	76.65	**+0.51**
	Ours*	76.14	75.85	-0.29
MobileNetV2	DAI8 [3]	72.44	71.92	-0.52
	Ours	72.84	73.08	**+0.24**
	Ours*	72.84	72.35	-0.49

Table II compares the proposed RLSRU with other LFSR-based SRU designs regarding HW implementation. When applied to ResNet18, Ours* with the RLSRU method achieved a 0.22% higher performance than SRU-Q [5] (See Table I), while utilizing only 12% and 14% of SRU-Q's LUT and FF resources, respectively. Although ESRU exhibited a slightly higher accuracy than Ours* (See Table I), RLSRU enhances randomness by reusing random numbers at the batch and channel levels with its seed generator, allowing it to achieve comparable performance to ESRU while using only 3.3% and 2.4% of ESRU's LUT and FF resources, respectively.

TABLE II: Resource Comparison with different LFSR designs

Method	LFSR Bit-width	LUTs	FFs
SRU-Q [5]	12bit	4,423	5,596
ESRU [4]	3bit	16,384	32,768
Ours	4bit	**542**	**785**

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
[2] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.
[3] K. Zhao *et al.*, "Distribution adaptive int8 quantization for training cnns," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 3483–3491, 2021.
[4] S.-E. Chang *et al.*, "Esru: Extremely low-bit and hardware-efficient stochastic rounding unit design for low-bit dnn training," in *2023 Design, Automation & Test in Europe Conference & Exhibition*, pp. 1–6, 2023.
[5] S. Jeong, D. Choi, and H. Kim, "Sru-q: Hardware-friendly stochastic rounding unit-based gradient quantization for cnn training," in *IEEE 6th International Conference on AI Circuits and Systems*, pp. 457–461, 2024.