Column-wise Quantization of Weights and Partial Sums for Accurate and Efficient Compute-In-Memory Accelerators

Jiyoon Kim[1], Kang Eun Jeon[2], Yulhwa Kim[3*], and Jong Hwan Ko[2*]

[1]Department of Artificial Intelligence, [2]Department of Electrical and Computer Engineering
[3]Department of Semiconductor Systems Engineering, Sungkyunkwan University
{jiyun19, kejeon, yulhwakim, jhko}@skku.edu

Abstract—**Compute-in-memory (CIM) is an efficient method for implementing deep neural networks (DNNs) but suffers from substantial overhead from analog-to-digital converters (ADCs), especially as ADC precision increases. Low-precision ADCs can reduce this overhead but introduce partial-sum quantization errors degrading accuracy. Additionally, low-bit weight constraints, imposed by cell limitations and the need for multiple cells for higher-bit weights, present further challenges. While fine-grained partial-sum quantization has been studied to lower ADC resolution effectively, weight granularity, which limits overall partial-sum quantized accuracy, remains underexplored. This work addresses these challenges by aligning weight and partial-sum quantization granularities at the column-wise level. Our method improves accuracy while maintaining dequantization overhead, simplifies training by removing two-stage processes, and ensures robustness to memory cell variations via independent column-wise scale factors. We also propose an open-source CIM-oriented convolution framework to handle fine-grained weights and partial-sums efficiently, incorporating a novel tiling method and group convolution. Experimental results on ResNet-20 (CIFAR-10, CIFAR-100) and ResNet-18 (ImageNet) show accuracy improvements of 0.99%, 2.69%, and 1.01%, respectively, compared to the best-performing related works. Additionally, variation analysis reveals the robustness of our method against memory cell variations. These findings highlight the effectiveness of our quantization scheme in enhancing accuracy and robustness while maintaining hardware efficiency in CIM-based DNN implementations. Our code is available at https://github.com/jiyoonkm/ColumnQuant.**

Index Terms—**Compute-in-memory, convolutional neural networks, quantization**

I. INTRODUCTION

In recent years, compute-in-memory (CIM) has become an efficient paradigm for implementing deep neural networks (DNNs) [1], [2], reducing data transfer between memory and computational units. Nevertheless, CIM-based architectures face significant analog-to-digital converter (ADC) overhead, which increases with ADC precision, affecting both area and energy consumption [1], [2]. Low-resolution ADCs alleviate this issue but require partial-sum quantization, introduces errors that degrade network accuracy. Furthermore, accuracy is further degraded by low-bit weight constraints due to cell representation limits and the need for multiple cells to support higher-bit weights. Consequently, the efficiency and accuracy also heavily rely on weight quantization [12] in CIM-based DNNs.
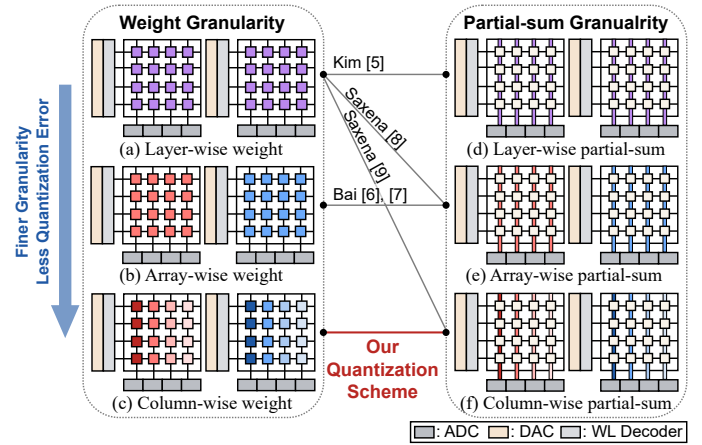

Fig. 1. Overview of the proposed quantization method and previous works.

To effectively lower the ADC resolution, the partial-sum quantization has been explored in previous works [5]–[9]. Finer granularity in partial-sum quantization, such as array-wise [6]–[8] and column-wise [9], has played a key role in enhancing the accuracy of DNN models with low-bit partial-sums. Partial-sum quantization has also progressed from a post-training quantization (PTQ) approach [5]–[7] to a quantization-aware training (QAT) [8], [9], enabling models to recover accuracy loss during training. As a result, previous works have restored the accuracy of partial-sum quantized models to levels comparable to those without partial-sum quantization.

Meanwhile, the weight quantization granularity is crucial since low-bit weight models set an upper bound for the accuracy achievable with partial-sum quantization. Although some attempts have refined weight granularity [6], [7], the improvements have been modest and insufficient to fully optimize performance, leaving the potential for further enhancement.

To overcome these obstacles, we introduce an innovative strategy that aligns weight and partial-sum quantization granularity, specifically at the column-wise level as illustrated in Fig. 1. Our method offers several key advantages: first, it provides finer control over quantization, allowing column-wise weight quantization to capture weights accurately, which in turn enables more precise column-wise partial-sum quantization. This improved accuracy is achieved without increasing dequantization overhead. In addition, it enhances training efficiency by eliminating the need for two-stage training, typically required
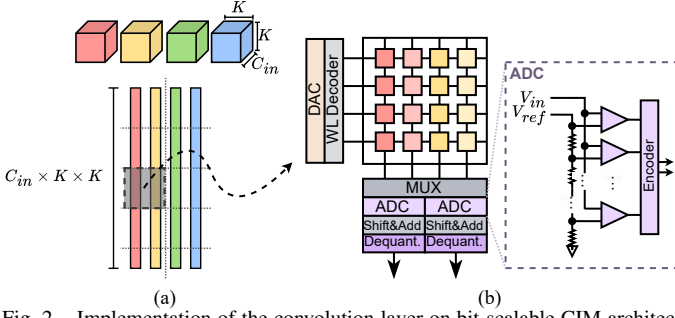
*Corresponding authors.

Fig. 2. Implementation of the convolution layer on bit-scalable CIM architecture. (a) Im2col mapping and tiling process. C_{in} represents the number of input channels, and K denotes the kernel size. (b) Bit-scalable CIM architecture.

TABLE I
RELATED WORKS ON PARTIAL-SUM QUANTIZATION

Related Works	Weight			Partial-sum		
	Granularity	*Train from scratch*	*Learnable scale factor*	*Granularity*	*Train from scratch*	*Learnable scale factor*
Kim [5]	Pretrained			Layer	✗ (PTQ)	✓
Bai [6]	Array	✗ (PTQ)	✗	Array	✗ (PTQ)	✓
Bai [7]	Array	✗ (PTQ)	✗	Array	✗ (PTQ)	✓
Saxena [8]	Layer	✓	✗	Array	✗ (2-stage QAT)	✓
Saxena [9]	Layer	✓	✓	Column	✗ (2-stage QAT)	✓
Ours	Column	✓	✓	**Column**	✓	✓

when the granularities differ. Finally, it is robust to memory cell variations due to independent column-wise scale factors.

However, implementing column-wise quantization poses challenges due to the time costs and increased complexity associated with fine-grained weights and partial-sums. Mitigating these difficulties, we propose the first open-source framework designed for CIM-oriented convolution with a novel array tiling method that preserves stretched kernels within each array, removing the bottlenecks of the im2col approach. Group convolution further eliminates sequential convolution delays and simplifies access to array-wise partial-sums.

Our experiments on ResNet-20 with CIFAR-10 and CIFAR-100, and ResNet-18 with ImageNet show notable accuracy improvements of 0.99%, 2.69%, and 1.01%, respectively compared to the best-performing related works. Furthermore, variation analysis confirms the robustness of our method to memory cell variations, emphasizing its effectiveness in improving both accuracy and hardware efficiency in CIM-based DNNs.

II. BACKGROUND & RELATED WORK

A. DNN Inference with CIM

Fig. 2 briefly illustrates an overview of CIM mapping, tiling, and architecture, where the bit-scalable multiplication-accumulation(MAC) operations are performed directly within the memory array [1], [2]. In Fig. 2(a), convolutional weights are mapped into CIM arrays and then tiled to match the size of the array. In this example, image-to-column (im2col) mapping is used, where each convolutional kernel is stretched into a column vector. In Fig. 2(b), weights are stored across multiple cells depending on the weight bit precision and the number of bits per cell. Inputs, provided through the word lines(WLs), are processed via a digital-to-analog converter (DAC) and fed into the memory cells. Within the array, weights and input bits are multiplied, generating partial-sums as electrical currents. After the MAC operation, partial-sums are routed through a multiplexer, and digitized by ADCs introducing severe overhead in terms of area and energy consumption. The reference voltage for each ADC, V_{ref}, is set by the scale factor corresponding to its input partial-sums, ensuring that the digitization process accurately captures their varying magnitudes. Once digitized, the partial-sums undergo a shift-and-add operation. Finally, the partial-sums are dequantized, with the stored scale factors multiplied to restore the original values as closely as possible.

B. Related Work

Several studies have proposed solutions for the ADC challenge in CIM, but major differences remain in the quantization scheme, training strategy, and learnable scale factor utilization. Specifically, the quantization scheme refers to the quantization granularity where elements are grouped and assigned a common scale factor during quantization. Finer granularities apply unique scale factors to smaller groups, while coarser granularities use a single scale factor for larger groups. In CIM-based architectures, quantization granularities for both weights and partial-sums range from conventional layer-wise to column-wise, as depicted in Fig. 1, affecting DNN performance and efficiency. In the figure, each area with the same color represents elements sharing a common quantization scale factor: weights in (a) to (c) and partial-sums in (d) to (f).

In [5], the quantization scheme illustrated in Fig. 1(a) and (d) is used, relying on post-training quantization (PTQ). While PTQ simplifies the quantization process, it often results in critical quantization errors since the model is not trained for lower precision. Furthermore, although a learnable scale factor is applied to partial-sums, the absence of a learnable scale factor for weights limits the model's ability to fully optimize its performance in a quantized setting.

The quantization approaches adopted by [6] and [7] correspond to the methods in Fig. 1(b) for weights and (e) for partial-sums, but the adaptability is still constrained by PTQ. The lack of learnable scale factors for weights constrains the model's ability to adjust to quantization, making it less robust in handling quantization errors.

The authors in [8] and [9] adopted layer-wise granularity for weights as seen in Fig. 1(a), but partial-sums are quantized differently—array-wise in [8] and column-wise in [9]. Both methods rely on a two-stage QAT approach due to the mismatch between weight and partial-sum granularities, where weights undergo QAT from scratch, while partial-sums are only quantized during the second stage of training. This leads to inefficiencies, as weights are overfitted to full-precision partial-sums during the first stage, delaying the model's adaptation to partial-sum quantization errors and hindering optimization. Moreover, applying a learnable scale factor only to the partial-sums, as in [8], restricts the model's flexibility to adapt across both weights and partial-sums simultaneously, reducing its overall effectiveness.

In summary, the related works did not implement fine-grained quantization for both weights and partial-sums, nor did
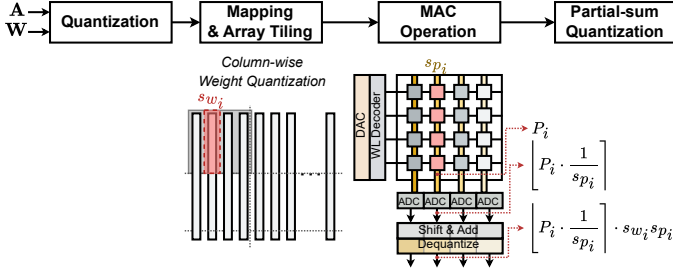
Fig. 3. Matrix multiplication of a DNN layer with the proposed column-wise weight and partial-sum quantization. (A: activations, W: weights, P_i: partial-sums, s_{w_i}: scaling factor for weights, s_{p_i}: scaling factors for partial-sums)

they apply learnable scale factors to both. Additionally, these approaches left room for improving training efficiency. Our approach addresses these limitations by aligning the quantization granularities of weights and partial-sums to column-wise level. Table I compares recent works and our method across three key factors: the finest quantization granularity, the ability to train from scratch, and the use of learnable scale factors.

III. PROPOSED METHOD

In this section, we introduce our column-wise quantization approach for both weights and partial-sums, evaluating its significance in relation to dequantization overhead and training efficiency. Additionally, we present our CIM-oriented convolution framework that seamlessly reflects the hardware architecture, addressing the inefficiencies associated with implementing column-wise quantization.

A. Column-wise Weight and Partial-sum Quantization

As illustrated in Fig. 3, in typical CIM-based architectures, the convolution operation begins with the independent quantization of weights, $\mathbf{W}$, and activations, $\mathbf{A}$. The quantized weights are mapped and tiled into arrays, which are then used in the MAC operation on each array, followed by the quantization of the resulting partial-sums.

Expanding on this workflow, our method improves it by employing column-wise quantization for both weights and partial-sums, as shown in Fig. 3. For instance, a column of weights in an array is quantized and multiplied by the corresponding quantized activation, generating the partial-sum as follows,

$$P_i = \left\lfloor \frac{W_i}{s_{w_i}} \right\rceil A_{q_i}, \tag{1}$$

where $\lfloor z \rceil$ rounds z to the nearest integer. After the MAC operation, the resulting partial-sums are quantized:

$$\left\lfloor \left\lfloor \frac{W_i}{s_{w_i}} \right\rceil A_{q_i} \cdot \frac{1}{s_{p_i}} \right\rceil = \left\lfloor P_i \cdot \frac{1}{s_{p_i}} \right\rceil. \tag{2}$$

As demonstrated in the equations, the weight and partial-sum quantization processes are clearly distinct due to the separate rounding functions, enabling independent optimization for better flexibility and precision in the overall quantization process. As the quantizer, we employ the LSQ [10] method to train scale factors for both weights and partial-sums separately, optimizing the model for fine-grained quantization effectively. In addition, we extend LSQ to support scale factors at varying granularities, including column-wise quantization.
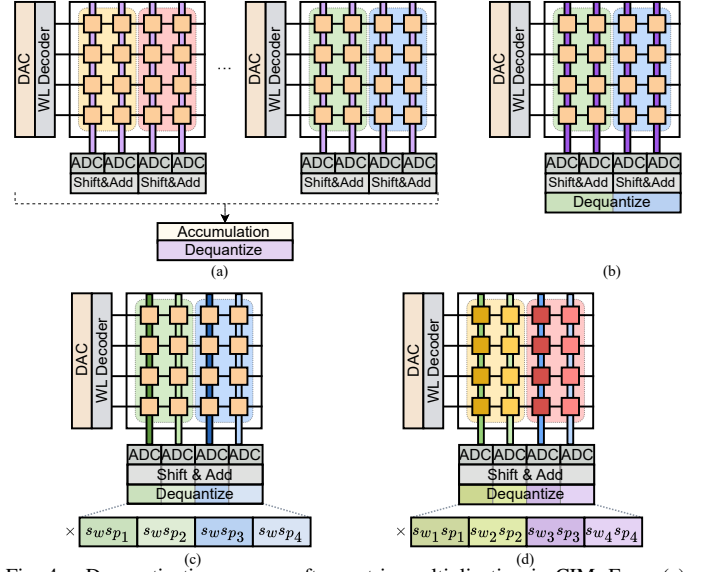

Fig. 4. Dequantization process after matrix multiplication in CIM. From (a) to (c), weights are quantized layer-wise, with partial-sums quantized (a) layer-wise, (b) array-wise, or (c) column-wise. (d) shows our proposed column-wise weight and partial-sum quantization. This figure assumes that the 4-bit weights are implemented with two 2-bit cells. Memory cells with the same background color belong to the same output channel.

Finally, each partial-sum is dequantized:

$$\left\lfloor \left\lfloor \frac{W_i}{s_{w_i}} \right\rceil A_{q_i} \cdot \frac{1}{s_{p_i}} \right\rceil \cdot s_{w_i} s_{p_i} = \left\lfloor P_i \cdot \frac{1}{s_{p_i}} \right\rceil \cdot s_{w_i} s_{p_i}. \tag{3}$$

If, on the other hand, weights are quantized at the layer-wise level, the weight quantization is simplified with a single scale factor, s_w, for all weights in the layer. However, when partial-sums remain quantized column-wise, the rest of the process for each $N \times N$ array follows the same steps as in the former case:

$$\left\lfloor \left\lfloor \frac{W_1}{s_w} \right\rceil A_{q_1} \cdot \frac{1}{s_{p_1}} \right\rceil \cdot s_w s_{p_1},$$
$$\dots, \left\lfloor \left\lfloor \frac{W_N}{s_w} \right\rceil A_{q_N} \cdot \frac{1}{s_{p_N}} \right\rceil \cdot s_w s_{p_N}. \tag{4}$$

To summarize, we propose column-wise quantization for both weights and partial-sums, which provides superior accuracy compared to other quantization schemes while maintaining the same dequantization overhead. Also, the column-wise granularity alignment enhances training efficiency by enabling precise one-stage QAT, as further detailed in the following sections. Moreover, it exhibits robustness against device-level variations, as the independent scale factors are assigned to each column effectively.

B. Dequantization in Column-wise Quantization

Although distinct scale factors are required for weights and partial-sums as previously discussed, the hardware architecture allows for their dequantization to be performed simultaneously, thereby enhancing efficiency. Fig. 4 provides a visual representation of this process, with weights quantized layer-wise and partial-sums quantized (a) layer-wise, (b) array-wise, or (c) column-wise.

In the conventional layer-wise quantization shown in Fig. 4(a), outputs from each array are first accumulated, followed by a single dequantization step for the entire layer,
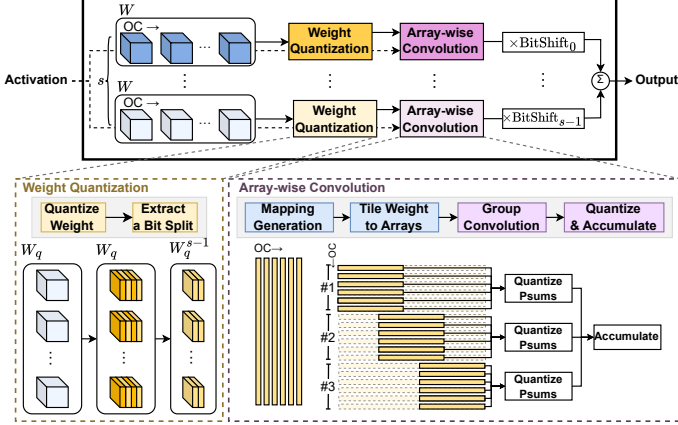
Fig. 5. Convolution framework overview for bit-scalable CIM. The figure outlines the process, starting with weight duplication and quantization into bit-splits. Each bit-split undergoes array-wise convolution with weight mapping, tiling, and partial-sum quantization. Finally, outputs from each split are bit-shifted and accumulated to form the final convolution output.

requiring the overhead of one scale factor multiplication. On the other hand, if partial-sums are quantized at array-wise level, those from the same output channel within an array are first shift-and-added. Subsequently, each channel undergoes separate dequantization as illustrated in Fig. 4(b), increasing the overhead to $n_{\mathrm{array}} \times n_{\mathrm{oc}}$ multiplications for a layer, where n_{array} and n_{oc} are the number of arrays and output channels in an array. Further granularity refinement is seen in Fig. 4(c), where each column of partial-sums has its unique scale factor, s_{p_1} to s_{p_4}. While the layer-wise weight scale factor, s_w, is shared across all columns, the system must store and apply the appropriate multiplied scale factor to each column, resulting in per-column overhead of $n_{\mathrm{split}} \times n_{\mathrm{array}} \times n_{\mathrm{oc}}$ multiplications, where n_{split} is the number of bit-splits.

Our quantization scheme extends the concept of column-wise quantization by applying unique scale factors to each column for both weights, s_{w_1} to s_{w_4}, and partial-sums, s_{p_1} to s_{p_4}. The multiplied scale factors are applied to each partial-sum that flows out from its corresponding column, as shown in Fig. 4(d). This approach appears to intensify the dequantization complexity due to the fine weight granularity. However, the key discovery is that, without introducing additional dequantization overhead, aligning column-wise quantization for weights and partial-sums enables finer and more precise control over the quantization process, compared to the layer-wise weight with column-wise partial-sum configuration.

C. A Convolution Framework for Column-wise Quantization

We propose a custom convolution layer framework designed to efficiently implement convolution layers with column-wise weight and partial-sum quantization. Conventionally, handling individual columns introduces challenges of time costs and complexities in data management, often resulting in inefficiencies. We resolve these obstacles by utilizing a unique array tiling method combined with group convolution. As illustrated in Fig. 5, we propose solutions to overcome these challenges and enhance processing efficiency, which emulates the convolution process in bit-scalable CIM architectures.

In CIM arrays, quantized weights break down into smaller segments, bit-split weights, to fit the number of capable bits

per memory cell. When replicating bit-scalable operations, it is necessary to enable access to each bit-split of the weight. To facilitate this, we duplicate the original weight, W, according to the number of bit-splits, s, allowing independent processing of each bit-split during the quantization and convolution stages.

Subsequently, these duplicated weights are quantized to W_q at layer-wise, array-wise, or column-wise granularity. Each quantized weight then undergoes array-wise MAC operation, facilitated by weight mapping and tiling. In the conventional im2col method, time-consuming linear operations are used to compute MAC results, creating a noticeable bottleneck. To improve processing efficiency, we propose a novel tiling method that transforms the linear operation into a convolution. By strategically adjusting the tiling stride, we ensure stretched kernels remain intact in each array, as described in Fig. 5, and reshape it into a 4-dimensional convolutional weight.

Moreover, sequential array-wise convolution introduces critical overhead, as it requires indexing arrays one by one. To mitigate this challenge, we utilize group convolution, matching the number of groups to the number of arrays, as illustrated in Fig. 5. This removes sequential indexing delays and simplifies access to array-wise partial-sums, resulting in faster convolution and subsequent partial-sum quantization.

Following the array-wise convolution, the generated partial-sums are quantized and accumulated. Importantly, weight and partial-sum granularities are independently selected from layer-wise, array-wise, or column-wise quantization, enabling tailored optimization based on the specific requirements of the model and hardware configuration. Finally, the convolution outputs from each bit-split weight are shifted and accumulated to produce one convolutional layer output.

D. Efficient One-stage QAT via Granularity Alignment

In [9], a two-stage QAT approach was employed to manage the granularity mismatch between weights and partial-sums, with partial-sums only quantized in the second stage of training to reduce training costs. In contrast, our method achieves efficient one-stage QAT from scratch by aligning the granularity of both weights and partial-sums to the column-wise level. This approach simplifies training by eliminating the need for separate stages to handle the partial-sum quantization, and ensures consistent optimization of both without compromising granularities.

IV. Experimental Results

A. Settings

We evaluated the impact of quantization granularities on network accuracy and dequantization overhead, applying one-stage QAT of ResNet-20 [3] on CIFAR-10 and CIFAR-100, and ResNet-18 [3] on ImageNet [4]. Each experiment employed distinct quantization and array size settings, as detailed in Table II. For ResNet-20, we compared accuracy across layer-wise, array-wise, and column-wise quantization for weights and partial-sums, while, for ResNet-18, we replicated granularities from related works for direct comparison. Additionally, we examined the effect of granularity alignment on the dequantization process and training efficiency, and conducted variation

TABLE II

EXPERIMENTAL SETTINGS FOR EVALUATING
THE IMPACT OF QUANTIZATION GRANULARITIES ON ACCURACY

Dataset		CIFAR-10	CIFAR-100	ImageNet
Model		ResNet-20	ResNet-20	ResNet-18
Bit Precision	Activation	3b	4b	3b
	Weight	3b (1b/cell)	4b (2b/cell)	3b (3b/cell)
	Partial-sum	Binary	3b	2b
Array Size		128x128		256x256
Training Scheme		Train from scratch for 200 epochs		Train from scratch for 90 epochs
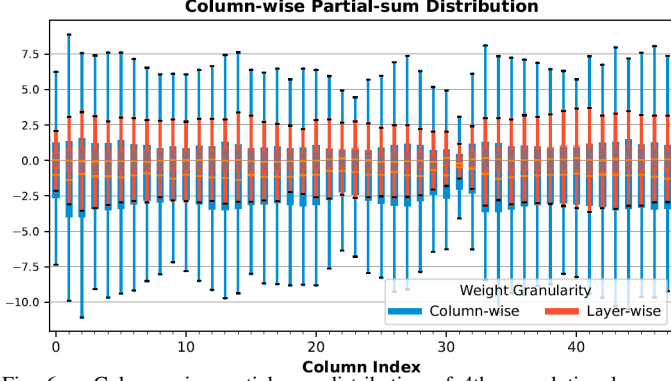
Fig. 6. Column-wise partial-sum distribution of 4th convolution layer of ResNet-20 for CIFAR-10, comparing dynamic ranges between layer-wise and column-wise weight quantization.

analysis based on the granularity combinations used in related works, applying ResNet-20 settings on CIFAR-10.

B. Impact of Quantization Granularity

The granularity of weight quantization is essential for enhancing partial-sum representation capability as well as that of weights. Fig. 6 compares the integer-valued column-wise partial-sum distributions for a specific layer from ResNet-20.

The figure clearly demonstrates that the column-wise weight quantization yields a larger dynamic range for the partial-sums, improving their representation ability. In contrast, layer-wise weight quantization uses a single scale factor for the entire layer, resulting in a more uniform distribution that limits the adaptability to differences across columns. By assigning distinct scale factors to each column, our column-wise weight quantization captures the weights more accurately, enabling more precise partial-sum computations. Thus, this approach is particularly effective for fine-grained partial-sum quantization, surpassing coarser schemes.

This advantage is further confirmed in our following experiments across ResNet20 on CIFAR-10 and CIFAR-100, and ResNet18 on ImageNet. Fig. 7(a) presents a comparison of CIFAR-10 results with different granularity combinations. The result presents that our proposed method attains the highest accuracy among all quantized models and related works. Our method achieves a top-1 accuracy of 90.21%, which is the closest to the full-precision(FP) model accuracy of 90.70%, surpassing all related works. Specifically, compared to [9], which employed layer-wise weight and column-wise partial-sum quantization, our approach improves accuracy by 0.99%.

This trend is observed in Fig. 7(b) for CIFAR-100 and Table III for ImageNet.
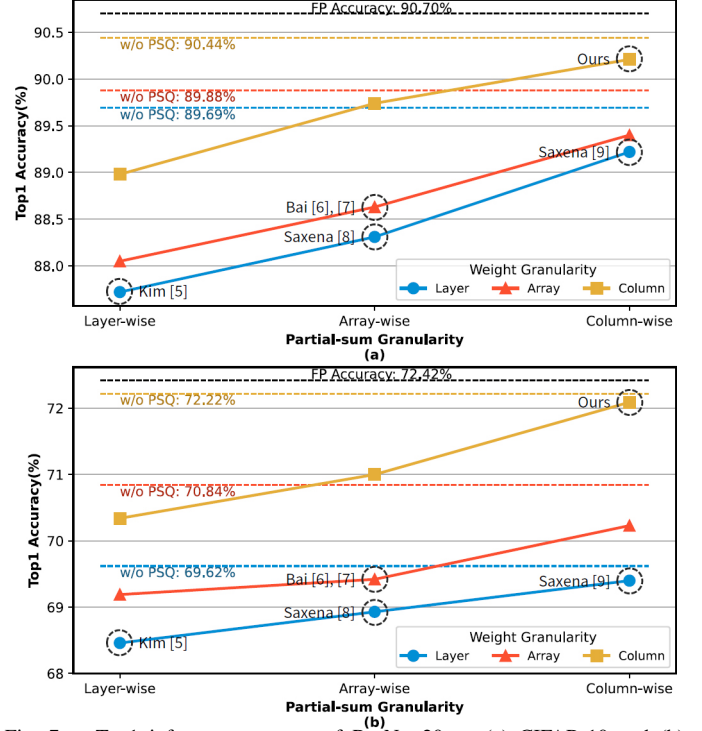

Fig. 7. Top1 inference accuracy of ResNet-20 on (a) CIFAR-10 and (b) CIFAR-100 with various granularity of weight and partial-sum quantization. Colored dashed lines represent the accuracy achieved without partial-sum quantization (PSQ) for each weight granularity. The accuracy results corresponding to the proposed method and previous works are highlighted.

TABLE III
INFERENCE ACCURACY OF RESNET-18 ON THE IMAGENET DATASET

Quantization Scheme	Weight Granularity			Partial-sum Granularity			Accuracy
	Layer	Array	Column	Layer	Array	Column	
Full-precision Model							69.13%
Kim [5]	✓			✓			64.86%
Bai [6], [7]		✓			✓		66.53%
Saxena [8]	✓				✓		65.94%
Saxena [9]	✓					✓	67.50%
Ours			✓			✓	**68.51%**

In each case, the model with column-wise quantization for both weights and partial-sums consistently outperforms those with coarser granularities. For example, in CIFAR-100, column-wise weight and partial-sum quantization achieves a top-1 accuracy of 72.09%, compared to 69.40% achieved in [9] and 72.22% in the baseline model without partial-sum quantization. For ImageNet, our approach achieves 68.51%, closest to the full-precision model accuracy at 69.13%.

C. Dequantization Analysis

In Fig. 8, all possible quantization schemes are categorized based on dequantize operation overhead per layer along the x-axis. The results confirm that using finer weight quantization granularity achieves higher accuracy under the same dequantization overhead. This shows the advantage of column-wise weight granularity in improving model performance without increasing hardware complexity.

D. Impact of One-stage Quantization-aware Training

Fig. 9 compares four QAT schemes, each using different combinations of weight and partial-sum granularities. The plus marks show that using the quantization scheme from [9], the
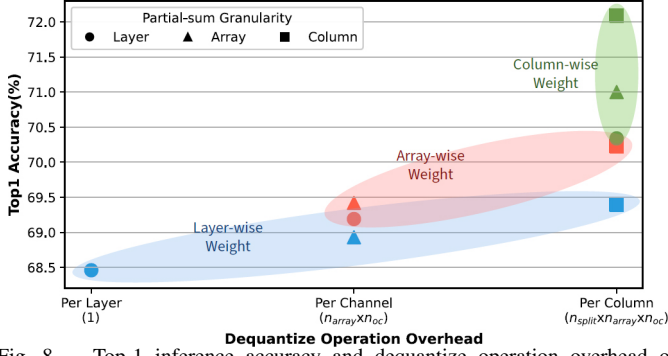
Fig. 8. Top-1 inference accuracy and dequantize operation overhead of ResNet-20 on CIFAR-100 with various granularities of weight and partial-sum quantization. Weights and activations are quantized to 4 bits, and 2-bit-per-cell arrays are used for the evaluation.
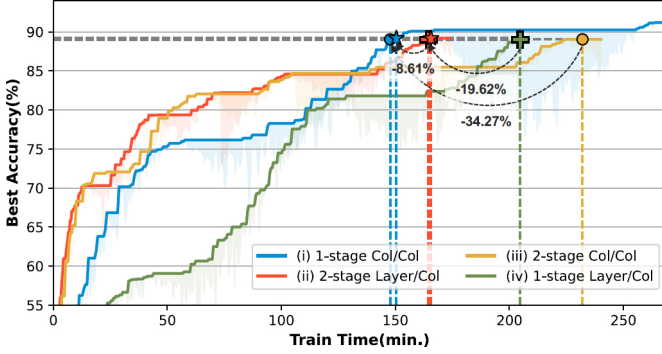

Fig. 9. Comparison of QAT schemes in terms of accuracy and train time. 'A/B' represents the granularity of 'weight/partial-sum' quantization. Plus marks show when case (ii) and (iv) achieved case (iv)'s best accuracy, circle marks indicate when case (i) and (iii) attained case (iii)'s best accuracy, and star marks denote when case (ii)'s best accuracy was reached in case (i) and (ii).

two-stage QAT approach achieves comparable accuracy to one-stage QAT with 19.62% less training cost. In contrast, the circle marks show that one-stage QAT yields a higher accuracy in our quantization scheme, and 34.27% less training cost compared to its two-stage counterpart.

The results verify the impact of aligning weight and partial-sum quantization granularities. While the weight granularity is coarser than that of the partial-sums in cases (ii) and (iv), both share column-wise granularity in cases (i) and (iii). The training delay between weights and partial-sums in case (iii) causes the weights to become overly tuned to full-precision partial-sums during the first stage, hindering performance in the second stage. Moreover, the star marks indicate that case (i) achieves the highest accuracy of the case (ii) with 8.61% less training cost, further validating our approach that the granularity alignment at the column-wise level ensures a more straightforward and efficient training process.

E. Evaluation of Variation Robustness

Non-idealities in nonvolatile memory, such as device variations, cause accuracy degradation in CIM accelerators. In this section, we conducted a variation analysis on models using our proposed quantization scheme and those of related works. As described in [11], memory device variations are modeled by a log-normal distribution with a mean of zero. To assess
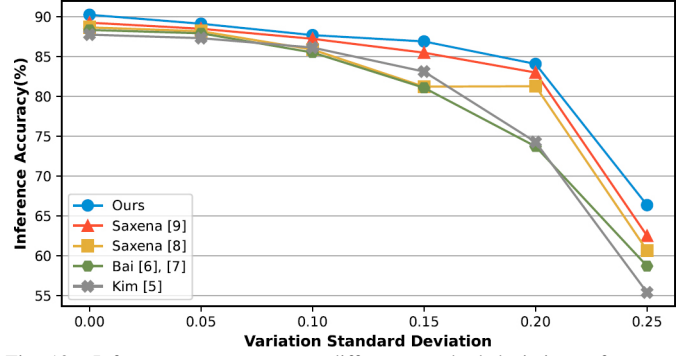

Fig. 10. Inference accuracy across different standard deviations of memory cell variation, comparing our quantization scheme with those of related works.

robustness, we introduced log-normal noise to the weights as follows,

$$w_{var} = w \cdot e^{\theta}, \qquad (5)$$

where θ is the noise, which follows a normal distribution with a zero mean, w is the ideal weight, and w_{var} is the weight after variation.

We evaluated inference accuracy across various standard deviations, as illustrated in Fig. 10. The result shows that models trained with our column-wise quantization method consistently achieve higher inference accuracy across all levels of variation, outperforming other quantization schemes. This gap highlights the robustness of our column-wise quantization approach in preserving model performance under hardware-induced variations, delivering both higher accuracy and greater resilience to memory cell variations.

V. CONCLUSION

We propose an innovative quantization strategy that aligns weight granularity with partial-sums at the column-wise level. Our method improves accuracy without increasing dequantization overhead and enhances training efficiency by removing the need for two-stage training. Although managing fine-grained weights and partial-sums presents challenges, we address them through an open-source CIM-oriented convolution framework, incorporating a novel array tiling method and group convolution. Our experiments on ResNet-20 (CIFAR-10 and CIFAR-100) and ResNet-18 (ImageNet) show substantial accuracy improvements—0.99%, 2.69%, and 1.01%, respectively—over leading methods. Additionally, the method's robustness to memory cell variations confirms its effectiveness in enhancing both accuracy and hardware efficiency in CIM-based accelerators.

REFERENCES

[1] A. Shafiee, *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in ISCA, 2016.

[2] Z. Zhu *et al.*, "A configurable multi-precision CNN computing framework based on single bit RRAM," in DAC, 2019.

[3] 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.

[4] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in Proceedings of the Advances in Neural Information Processing Systems , 2012.

[5] Y. Kim, H. Kim, and J.-J. Kim, "Extreme partial-sum quantization for analog computing-in-memory neural network accelerators," ACM Journal on Emerging Technologies in Computing Systems, vol. 18, no. 4, pp. 1–19, Oct. 2022.

[6] J. Bai, W. Xue, Y. Fan, S. Sun, and K. Wang, "Partial sum quantization for computing-in-memory-based neural network accelerator," IEEE Transactions on Circuits and Systems II-express Briefs, vol. 70, no. 8, pp. 3049–3053, Aug. 2023.

[7] J. Bai, S. Sun, W. Zhao, and W. Kang, "CIMQ: A hardware-efficient quantization framework for computing-in-memory based neural network accelerators," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 43, no. 1, pp. 189–202, Jan. 2024.

[8] U. Saxena, I. Chakraborty, and K. Roy, "Towards ADC-less compute-in-memory accelerators for energy efficient deep learning," 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), Mar. 2022.

[9] U. Saxena and K. Roy, "Partial-sum quantization for near ADC-less compute-in-memory accelerators," 2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), Aug. 2023.

[10] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," International Conference on Learning Representations, Apr. 2020.

[11] G. Charan, A. Mohanty, X. Du, G. Krishnan, R. V. Joshi, and Y. Cao, "Accurate inference with inaccurate RRAM devices: a joint algorithm-design solution," IEEE Journal on Exploratory Solid-State Computational Devices and Circuits, vol. 6, no. 1, pp. 27–35, June 2020.

[12] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," 2021, arXiv:2103.13630.