ShiftQuant: Toward Accurate and Efficient Sub-8-bit Integer Training

Wenjin Guo, Donglai Liu, Weiying Xie, *Senior Member, IEEE*, Yunsong Li, Xuefei Ning, Zihan Meng, Shulin Zeng, *Member, IEEE*, Jie Lei, *Member, IEEE*, Zhenman Fang, *Senior Member, IEEE*, and Yu Wang, *Fellow, IEEE*

***Abstract*—Neural network training is a memory- and compute-intensive task. Quantization, which enables low-bitwidth formats in training, can significantly mitigate the workload. To reduce quantization error, recent methods have developed new data formats and additional pre-processing operations on quantizers. However, it remains quite challenging to achieve high accuracy and efficiency simultaneously. In this paper, we explore sub-8-bit integer training from its essence of gradient descent optimization. Our integer training framework includes two components: ShiftQuant to realize accurate gradient estimation, and L1 normalization to smoothen the loss landscape. ShiftQuant attains performance that approaches the theoretical upper bound of group quantization. Furthermore, it liberates group quantization from inefficient memory rearrangement. The L1 normalization facilitates the implementation of fully quantized normalization layers with impressive convergence accuracy. Our method frees sub-8-bit integer training from pre-processing and supports general devices. This framework achieves negligible accuracy loss across various neural networks and tasks (0.92% on 4-bit ResNets, 0.61% on 6-bit Transformers). The prototypical implementation of ShiftQuant achieves more than 1.85× /15.3% performance improvement on CPU/GPU compared to its FP16 counterparts, and 33.9% resource consumption reduction on FPGA than the FP16 counterparts. The proposed fully-quantized L1 normalization layers achieve more than 35.54% improvement in throughout on CPU compared to traditional L2 normalization layers. Moreover, theoretical analysis verifies the advancement of our method.**

***Index Terms*—Model compression, integer training, neural network acceleration.**

Received 18 December 2024; revised 22 April 2025; accepted 13 May 2025. Date of publication 16 May 2025; date of current version 30 October 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62322117, Grant 62371365, Grant U24B20136, and Grant U22B2014. This article was recommended by Associate Editor Y. Liu. *(Wenjin Guo and Donglai Liu contributed equally to this work.) (Corresponding author: Weiying Xie.)*

Wenjin Guo, Donglai Liu, Weiying Xie, Yunsong Li, and Zihan Meng are with the State Key Laboratory of the Integrated Services Networks, Xidian University, Xi'an 710071, China (e-mail: guowenjin@stu.xidian.edu.cn; wyxie@xidian.edu.cn; ysli@mail.xidian.edu.cn).

Xuefei Ning, Shulin Zeng, and Yu Wang are with the Department of Electronic and Computer Engineering and Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: yu-wang@tsinghua.edu.cn).

Jie Lei is with the School of Electrical and Data Engineering, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: jie.lei@uts.edu.au).

Zhenman Fang is with the School of Engineering Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: zhenman@sfu.ca).

This article has supplementary downloadable material available at https://doi.org/10.1109/TCSVT.2025.3570707, provided by the authors.

Digital Object Identifier 10.1109/TCSVT.2025.3570707

I. Introduction

RECENTLY, deep neural networks have made significant advancements in various tasks. However, this progress comes at the cost of high computational complexity. High energy cost and computational resource consumption hamper the deployment of these deep learning applications on both edge-device and high-end cloud servers. While existing studies primarily concentrate on minimizing resource consumption during inference [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], it should be noted that the computational complexity of training is approximately three times that of inference [13]. Consequently, there is an urgent demand for efficient training methods.

Reducing the bitwidth of data is a reasonable approach. The resource consumption of a neural network scales almost linearly with the bitwidth of data [14]. However, low-precision training faces two challenges. The first is the wide range of gradients. Very few outliers extremely expand the data range, resulting in high quantization error [15], [16], [17]. The second is the sharp loss landscape of low-precision networks [18]. Sharp local minimal points disrupt optimization significantly [19], [20].

To address the challenge of wide gradient range and significant quantization errors, common approaches include (1) employing a non-uniform quantization format [15], [21], [22], (2) utilizing smaller quantization granularity, or (3) suppressing outliers before quantization [16], [17]. However, all methods encounter practicality issues, as they cannot be easily adapted to existing General Matrix Multiply (GEMM) software and hardware implementations or extremely exacerbate the computation burden (detailed discussion in Sec. II). To this end, we analyze the structure of outliers in gradients and develop an efficient and effective quantizer, ShiftQuant. With strategic and structural grouping of channels, ShiftQuant achieves excellent outlier suppression at a negligible cost, and also supports GEMM.

Recent methods focus on developing new quantizers but neglect the impact of sharp loss landscape [13], [15], [16], [17], [23], [24], [25], [26], [27]. The sharp landscape of low-precision networks brings more local minimal points and leads to unstable convergence. Moreover, sharp curvature demands more bits to specify gradients [19], [28]. In this paper, we reveal that the source of the sharp landscape lies in

the limited smoothening and quantization-tolerated capacity of the normalization layers [29], [30], [31], [32]. Stronger smoothening comes from stronger regularization. In this paper, we introduce the stronger regularization, L1 normalization, into normalization layers, which achieves clearly smooth loss landscape under less computation.

Our main contributions include the following:

By analyzing the structure of outliers in gradients, we find that appropriately grouping channels can effectively reduce quantization errors. Hence, we develop a new quantizer, ShiftQuant. ShiftQuant applies a smart grouping strategy that effectively approximates the optimal solution for grouping with minimal computational overhead. ShiftQuant utilizes the common low-bitwidth integer format and supports GEMM. Moreover, we develop a specific implementation of the new quantizer, which avoids the inefficient memory rearrangement in per-group quantization for the first time.

We experimentally analyze the source of the sharp loss landscape in low-precision networks. Quantization weakens the smoothening effort of L2 normalization layers significantly. To this end, we develop L1 normalization layers, which achieves stronger smoothening with less computation.

We evaluate the proposed framework on various datasets. Our method achieves negligible performance loss on ResNets, Transformers, GNN, and RNN under sub-8-bit integer training, respectively. Furthermore, theoretical analysis substantiates the effectiveness of the new quantizer and normalization layers.

Our prototypical implementation of ShiftQuant achieves more than 1.85 × /15.3% performance improvement on CPU (ARMv8)/GPU (Nvidia RTX 3090) compared to Pytorch.fp16, and more than 33.9% resource consumption reduction on FPGA (ZC706) compared to FP16. Moreover, 6-bit ShiftQuant surpasses 4-bit integer training competitor in efficiency. The implementation of proposed fully-quantized L1 normalization layers achieves more than 35.54% improvement of throughout on CPU compared to traditional L2 normalization layers.

II. Related Work and Challenges

A. New Quantization Data Formats

Developing new data formats with wide representing range reduces quantization loss significantly. Sun et al. [15] proposes a new fixed-point format using radix-4, which, unlike the traditional radix-2 fixed-point format, offers a significantly larger representation range and higher resolution. Fu et al. [23] explores a dynamic data format with varying bit-widths to balance accuracy and efficiency by periodically adjusting the bit-width. Cambier et al. [21] proposes the S2FP8 format, where a vector is represented by an FP8 vector and two FP32 values, named squeeze and shift factors. Zhong et al. [22] develops the MLS tensor format, which balances the accuracy and computation efficiency in a tensor level. Although the excellent accuracy, new data formats demand customer design of new hardware units, hindering the deployment on contemporary devices inherently.

B. Fine-Grained Quantization

Fine-grained quantization can reduce quantization error significantly. However, naive fine-grained quantization may lead to incompatibility with GEMM [22], [33]. Since when fine-grained quantization is applied to the inner dimension, different indices have distinct floating-point scaling factors. Inevitably, when accumulating these indices, each element must first multiply its corresponding scaling factor, thereby introducing extra floating-point multiplication operations. As shown in Figure 1, three matrix multiplies are involved in training: forward propagation, loss propagation, and calculation of weight gradient. Almost all dimensions serve as the inner dimension. To apply GEMM, fine-grained quantization is not allowed.

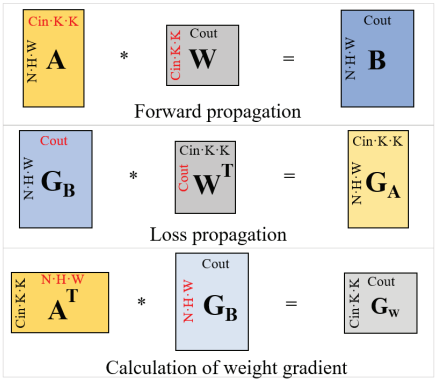


Fig. 1. Matrix multiplications in training. Inner dimensions are labeled in red.

C. Outlier Suppression Before Quantization

Before quantization, the data is reflected to an easy-quantization space through an invertible mapping. By conducting the inverse mapping after dequantization, the original data can be recovered with small loss. Chen et al. [17] implements the reflection by multiplying gradients with a constructed Hadamard matrix. Similarly, Xi et al. [16] applies the Hadamard reflection on quantizing activation during training transformers. Additional computation arises from constructing reflection. Moreover, the reflection operation is conducted in expensive floating-point format.

Our method utilizes the common integer format. Meanwhile, no pre-processing is applied before quantization. We highlight the difference between our method and prior methods at Table I.

III. ShiftQuant

A. Smart Channel Grouping

As shown in Figure 2(c), the distribution of gradient varies among channel dimension extremely. Channels with wide range expand the quantization range, leading to extremely fewer quantization levels to represent channels with small range (as shown in Figure 2(b)). High quantization variance comes from the diversity in channels' ranges. Merging the gap between channel's range can reduce quantization error significantly. Grouping channels is a reasonable approach. Appropriately grouping channels can minimize the diversity in each group. Then, per-group quantization can achieve low quantization variance and high efficiency at the same time.

TABLE I
DIFFERENCE BETWEEN OUR METHOD AND PRIOR STUDIES

	New data format [15], [23], [21]	Fine-grained [25], [24]	Outlier suppression before quantization [17], [16]	Ours
General data format	✘	✔	✔	✔
GEMM supporting	-	✘	✔	✔
No pre-processing on quantization	✔	✔	✘	✔
Fully-quantized normalization layers	✘	✘	✘	✔

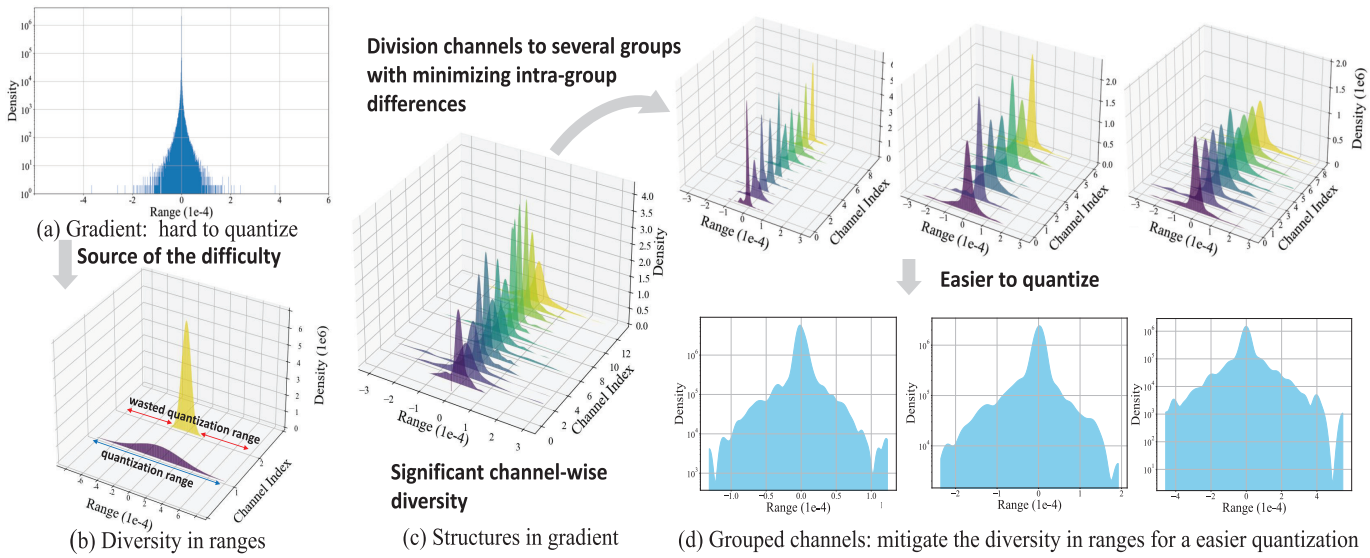


Fig. 2. The difficulty of quantizing gradients comes from the diversity between channels. ShiftQuant aims to minimize the diversity through strategic grouping channels. (a) Gradients exhibit extremely bell-curve distribution, which is very difficult to quantize. (b) Diversity in magnitude of channels leads to high quantization error. (c) High diversity of channels in gradients. (d) ShiftQuant divides channels to several groups. Each group characters small diversity, leading to easier quantization.

1) Optimizing Grouping Strategy: It is intuitive that smaller channels should be divided to one group, and larger channels should be divided to another group. The key of the grouping strategy is to find the threshold to distinguish small and large channels. To divide channels into N_G groups, we identify $N_G + 1$ thresholds $\tau_0, \tau_1, \cdots, \tau_{N_G-1}, \tau_{N_G}$.

The object of grouping is:

$$\min_{\tau_0,\tau_1,\cdots,\tau_{N_G-1},\tau_{N_G}} \quad \sum_{g=1}^{N_G} \sum_{i \in P^g} \frac{\tau_g}{r_i},$$
$$s.t.\, r_{max} = \tau_0 \geq \quad \tau_1 \geq \cdots \geq \tau_{N_G-1} \geq \tau_{N_G} = 0, \tag{1}$$

where P^g is a set that contains the indexes of channels allotted to group g. r_{max} is the minimum around ranges of channels. r_i is the range of i-th channel. r_i satisfies the following condition:

$$\tau_{g-1} < r_i \leq \tau_g, \; if \;\; i \in P^g. \tag{2}$$

$\sum_{i \in P^g} \frac{\tau_g}{r_i}$ reflects the diversity of channels' range in group g. Moreover, problem (1) is equivalent to directly optimize the quantization variance. The theoretical verification can be found in Appendix. A.

2) Efficient Power-of-Two Grouping: We can apply the optimal solution of problem (1) to group channels. Unfortunately, two serious obstacles will rise. The first is the complexity to solve. Problem (1) does not have a closed-form solution. It requires nonlinear dynamic programming, which demands iterations of complicated calculation to find the optimal solution. The second is the optimized thresholds are in floating-point formats. The scale of these groups does not follow integer-times relation. We have to accumulate the results of groups in floating-point format.

We analyze the characteristics of the objective function $\frac{\tau}{r}$, the ratio of group's upper threshold to the range of channel. We find that larger threshold can tolerate greater distance between channels' range and threshold. For instance, let $\tau_p = 2\tau_q$, $r_i = 2r_j$. The loss incured by grouping channel i to group p is equal to that of grouping channel j to group q ($\frac{\tau_p}{r_i} = \frac{\tau_q}{r_j}$). However, the distance between the threshold and channel's range of group p is twice of group q ($\tau_p - r_i = 2(\tau_q - r_j)$).

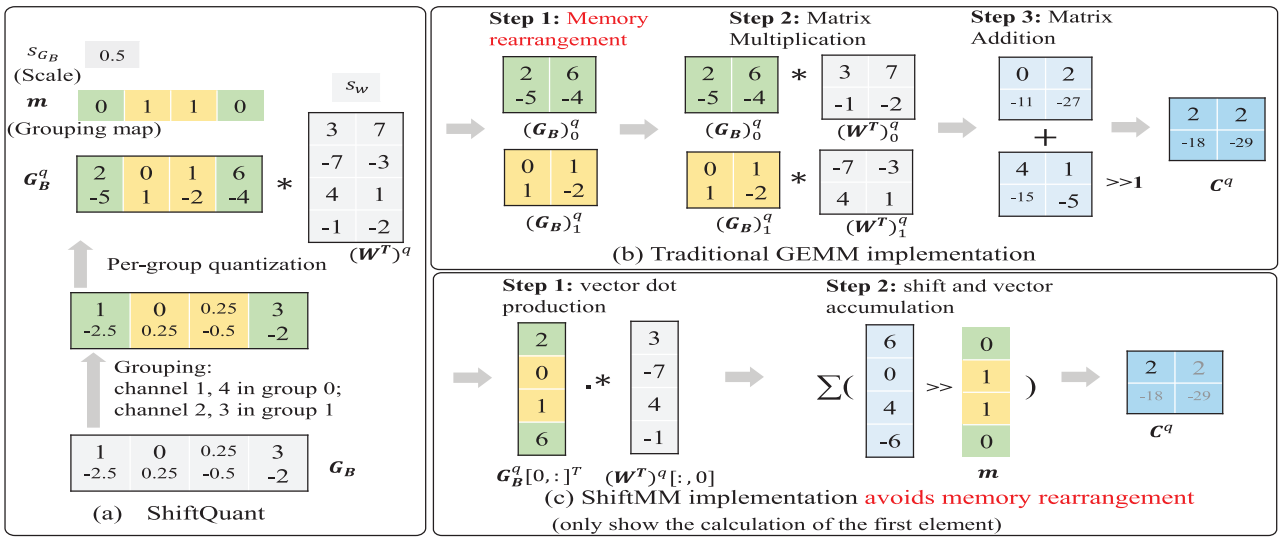


Fig. 3. The power-of-two grouping strategy in ShiftQuant paves a way for implementing per-grouping quantization without memory rearrangement. **Traditional implementation** approach (b) focuses on applying off-the-peg packages. It transfers original matrix multiplication to several small matrix multiplications based on the grouping map $\boldsymbol{m}$, which involves expensive memory rearrangement. **ShiftMM** (c) aims higher hardware efficiency. It replaces the memory rearrangement with only a low-cost shift operation. Meanwhile, it only demands slight changes on off-the-peg packages.

Therefore, to minimize the objective function, a group with larger threshold should hold larger range.

Consider the accuracy and hardware efficiency, we impose power-of-two relation on thresholds:

$$\tau_g = \tau_0 \cdot 2^g = r_{max} \cdot 2^g, g = 1, \cdots, G. \tag{3}$$

The above strategy partitions more ranges for groups with larger threshold. Moreover, it frees grouping from costly optimization and enables accumulation groups in integer format with only a shift operation. Take the calculation of $\boldsymbol{G_A}$ as example:

$$\begin{aligned} \boldsymbol{G}_A = \boldsymbol{G}_B \cdot \boldsymbol{W}^T &= \sum_{g=1}^{N_G} [s_{\boldsymbol{G}_B}^{-1} \cdot 2^{-g} \cdot (\boldsymbol{G}_B)_q^g] \cdot [s_{\boldsymbol{W}}^{-1} \cdot (\boldsymbol{W}^T)_q^g] \\ &= s_{\boldsymbol{G}_B}^{-1} \cdot s_{\boldsymbol{W}}^{-1} \sum_{g=1}^{N_G} [(\boldsymbol{G}_B)_q^g \cdot (\boldsymbol{W}^T)_q^g \gg g], \end{aligned} \tag{4}$$

where $(\boldsymbol{G}_B)_q^g$ and $(\boldsymbol{W}^T)_q^g$ denote the quantized g-th group of $\boldsymbol{G}_B$ and $\boldsymbol{W}^T$. Eq. (4) is also diagrammed in Figure 4(b). All of the calculation are performed on integer format. Detailed derivation is provided in Appendix. D. ShiftQuant achieves unbiased and low-variance quantization. Theoretical analysis is provided in Appendix. B.

B. Implementation Optimization

We devise two approaches to implement ShiftQuant. One is based on GEMM. Additionally, we develop a specific implementation method, named ShiftMM, which achieves higher hardware efficiency.

1) Implemented by GEMM: As illustrated in Figure 4(b), ShiftQuant divides the traditional matrix multiplication into n pairs of smaller matrix multiplications. Implementing ShiftQuant through GEMM involves three steps. Firstly, extracting the n pairs of smaller matrices, which requires memory rearrangement. Secondly, employing GEMM to perform the n pairs of matrix multiplications. Finally, shifting and summing the results of these n multiplications. This approach achieves a good balance between efficiency and versatility. However, the memory rearrangement increases time consumption, and the smaller matrix multiplications do not fully capitalize on the advantages of GEMM.

2) Implemented Without Memory Rearrangement: ShiftQuant essentially assigns different shift amounts to each index of the inner dimension. As depicted in Figure 4(c), we can perform the shifting directly before the accumulation phase of the vector multiplication. This technique offers two benefits. Firstly, it avoids memory rearrangement. Secondly, it maintains the same level of parallelism as the original matrix multiplication. We refer to this implementation method as ShiftMM. The implementation of ShiftMM just requires adding a shifting operation after the multiplication step in the standard GEMM code.

We present the implementation processes for ShiftMM and per-group quantization on GPU architectures respectively. In ShiftMM, the scale factors of individual groups obey a powers-of-two relationship, which eliminates the need to relocate group elements to contiguous addresses during matrix computations. Instead, the operation is realized through a single bit-shifting operation. In contrast, per-group quantization requires stochastic inter-group scales, necessitating a complex memory rearrangement step to reorder group elements prior to computation. This memory reorganization introduces significant computational overhead, ultimately leading to a substantial degradation in overall processing efficiency.

C. Theoretical Analysis on ShiftQuant

1) Theoretical Characters:

a) Unbiased Quantizer: ShiftQuant utilizes stochastic rounding:

$$SR(x) = \begin{cases} u(x) & w.p. \frac{x-l(x)}{u(x)-l(x)} \\ l(x) & w.p. 1 - \frac{x-l(x)}{u(l)-l(x)}, \end{cases} \tag{5}$$

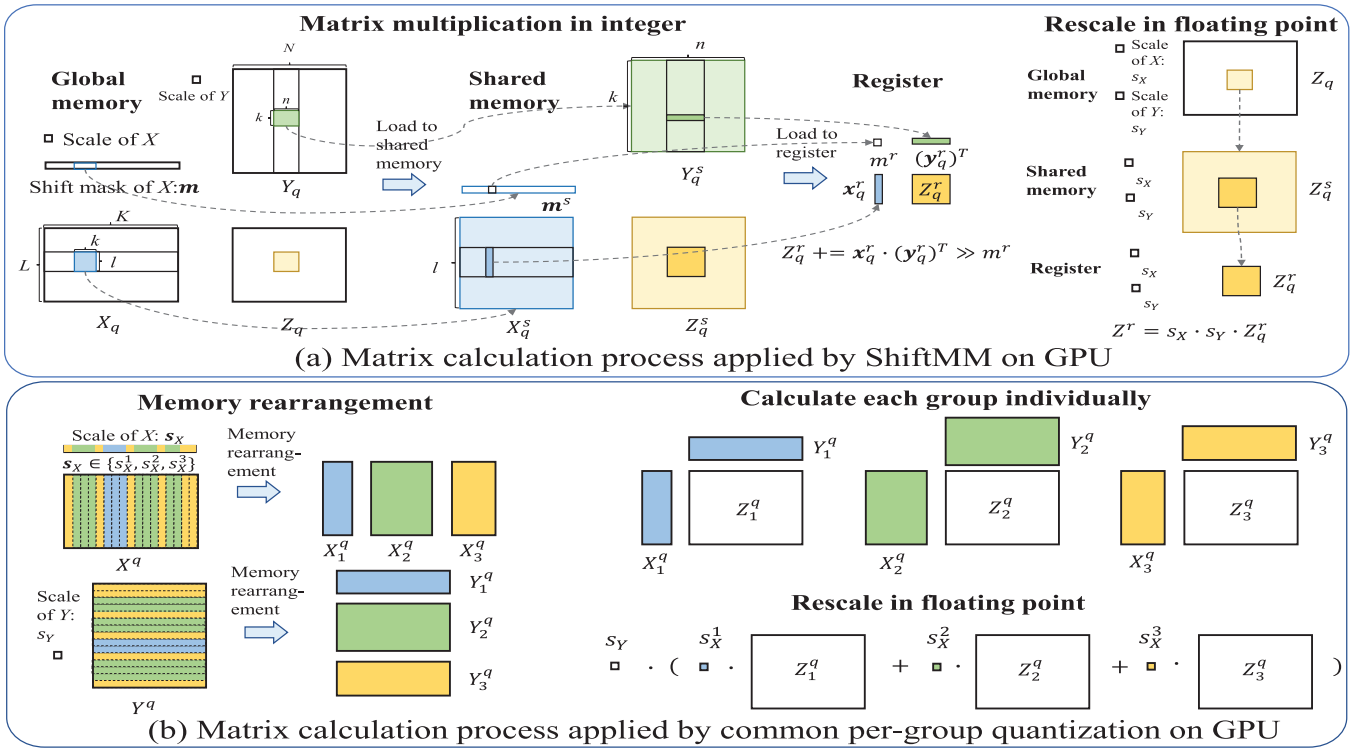


Fig. 4. The comparison between ShiftMM and per-group quantization in GPU-based matrix multiplication implementations. Benefiting from the power-of-two grouping strategy, ShiftMM achieves distinct processing for different groups solely through bit-shift operations during matrix multiplication. In contrast, traditional per-group quantization must first perform memory rearrangement to relocate elements of the same group to contiguous memory addresses before computation can proceed.

where $u(x) = s^{-1}\lceil s \cdot x \rceil$ is the upper quantization level of x. $l(x) = s^{-1}\lfloor s \cdot x \rfloor$ is the lower quantization level of x. s is the quantization scale. It is clear that ShiftQuant is unbiased:

$$E[D_q(X)] = E[SR((X - ZI) \cdot S) \cdot S^{-1} + ZI] = X. \quad (6)$$

$X \in \mathbb{R}^{N \times D}$ is the quantized matrix. The second dimension of X is the inner dimension. $S = diag(s_0, \cdots, s_{D-1})$. s_i is the scale applied in the i-th index of the inner dimension and $s_i \in \{s_l, 2s_l, \cdots, 2^{N_G-1}s_l\}$. s_l is the minimum scale. $Z = diag(z_0, z_1, \cdots, z_{D-1})$ represents the zero-point matrix. $I \in \mathbb{R}^{D \times D}$ is a identity matrix.

b) Low Variance: The up bound of ShiftQuant's variance U_{dq} satisfies:

$$U_{dq} \leq \alpha \cdot \frac{N}{4} \sum_{j=1}^{D} s_j^{-2} + 2^{-N_G} \frac{ND}{4} s^{-2}, \quad (7)$$

where α is depended on the distribution of channels' range, and $1 \leq \alpha \leq 4$. $\frac{N}{4}\sum_{j=1}^{D} s_j^{-2}$ is the up bound of fine-grained quantization's variance. $\frac{ND}{4}s^{-2}$ is the up bound of coarse-grained quantization's variance. Eq. (34) demonstrates that ShiftQuant achieves a closed performance to fine-grained quantizers.

IV. Fully-Quantized L1 Normalization Layers

A. Observation and Proposal of Normalization Layers

Compared with full-precision networks, full low-precision networks often encounter sharp loss landscapes during training, which makes the optimization process unstable and prone to getting trapped in local optima. In this section, through comparative experiments, we found that the fully quantized L2 normalization layer leads to a sharp loss landscape. We hypothesize that the weak regularization ability and quantization tolerance of the L2 normalization layer result in negative impacts on optimization. To address this issue, we propose an L1 normalization layer with stronger regularization ability and quantization tolerance, and verify this hypothesis through experiments and theoretical analysis.

As shown in Figure 6, the loss landscape of low-precision networks is sharper and more challenging to optimize compared to that of full-precision networks. As existing literature [34] points out that normalization layer help smoothen the loss landscape, we hypothesize that the quantization of the normalization layers might be a main cause of the sharp loss landscape of low-precision networks. To verify this hypothesis, we replace the quantized normalization layers in the low-precision networks with full-precision ones. As shown in Figure 6(c), the sharpness of the loss landscape disappeared, indicating that the quantization of the L2 normalization layers diminishes its smoothening capability, leading to the sharp loss landscape.

The quantized L2 normalization layer exhibits significantly reduced smoothing capability, necessitating the design of a normalization layer with stronger regularization capacity and higher quantization tolerance. Compared to L2 normalization, L1 normalization demonstrates stronger regularization

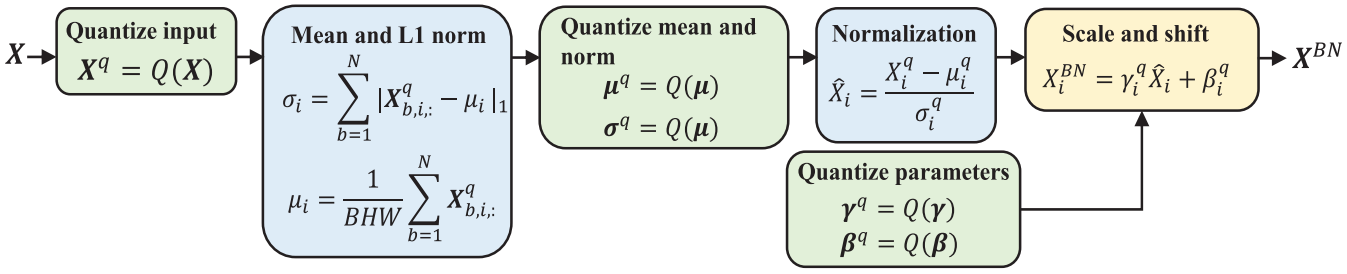
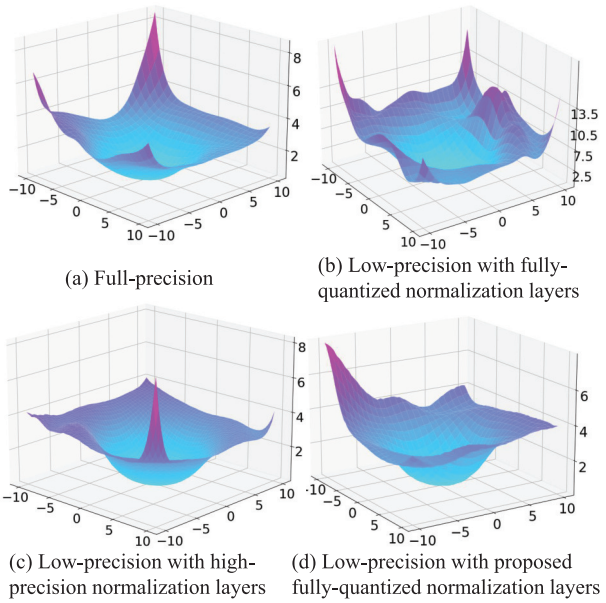


Fig. 5. Procedures of the proposed fully-quantized normalization layer (take BatchNorm for example). The input $\boldsymbol{X} \in \mathcal{R}^{N\times C\times HW}$, where N, C, HW denote the batchsize, number of channels, number of spatial elements, respectively.



(a) Full-precision (b) Low-precision with fully-quantized normalization layers (c) Low-precision with high-precision normalization layers (d) Low-precision with proposed fully-quantized normalization layers

Fig. 6. Low-precision networks feature sharpen loss landscape, which disrupts convergence. We visualize the loss landscape of ResNet20 on CIFAR10 by the method in [20].

capability and superior landscape smoothing ability [35], [36]. Furthermore, the L1 norm of the same activation is substantially larger than the L2 norm, leading to lower quantization loss in L1 normalization layers compared to their L2 counterparts.

The flow of the fully-quantized L1 normalization layer is shown in Figure 5. All of the input features, statistics, and parameters are quantized to low bitwidth formats. As shown in Figure 6(d), our fully-quantized L1 normalization layer achieves similar smoothening effort as the full-precision L2 normalization layer.

B. Theoretical Analysis

1) Analysis on Smoothing Capability: The smoothing effect of normalization layers on the loss landscape originates from their constraint on gradient magnitudes, which avoids the occurrence of abnormal (excessively large) gradients. We first conduct a quantitative analysis of the gradient-constraining capability of normalization layers, followed by a comparative study between L1-normalization and L2-normalization layers.

The normalization layer comprises two sequential operations: normalization, scale and shift:

$$\hat{\boldsymbol{x}} = \frac{\boldsymbol{x} - \mu}{\sigma}, \quad Normalization, \tag{8}$$

$$\boldsymbol{y} = \gamma \cdot \hat{\boldsymbol{x}} + \beta, \quad Scale\ and\ shift, \tag{9}$$

where $\boldsymbol{x}$ is the input activation. Here we vectorized the activation, and $\boldsymbol{x} \in \mathcal{R}^{NHW}$. μ is the mean of $\boldsymbol{x}$, and σ is the standard deviation. γ and β are the learned scaling and shift parameters. The normalization layers modify the gradient computation process in neural networks by regularizing input gradients:

$$\left\|\nabla_{\boldsymbol{x}_j}\widehat{\mathcal{L}}\right\|^2 \le \frac{\gamma^2}{\sigma_j^2}\left(\left\|\nabla_{\boldsymbol{x}_j}\mathcal{L}\right\|^2 - \frac{1}{m}\left\langle \mathbf{1}, \nabla_{\boldsymbol{x}_j}\mathcal{L}\right\rangle^2 - \frac{1}{m}\left\langle \nabla_{\boldsymbol{x}_j}\mathcal{L}, \hat{\boldsymbol{x}}_j\right\rangle^2\right), \tag{10}$$

where $\widehat{\mathcal{L}}$ is the loss function of the network with Batchnorm layers, and $\mathcal{L}$ is the loss function of same network without Batchnorm layers.

Only σ and $\hat{\boldsymbol{x}}$ are different in L1 normalization layers and L2 normalization layers. Then we have:

$$\sigma_1 = |\boldsymbol{x} - \mu|_1,\ \hat{\boldsymbol{x}}^1 = \frac{\boldsymbol{x} - \mu}{\sigma_1}, \quad L1\ normalization$$

$$\sigma_2 = |\boldsymbol{x} - \mu|_2,\ \hat{\boldsymbol{x}}^2 = \frac{\boldsymbol{x} - \mu}{\sigma_2}, L2\ normalization. \tag{11}$$

Thus the Lipschitzness constant of L1 normalization L^1 and L2 normalization L^2 satisfies:

$$\frac{L^1}{L^2} \le \frac{\sigma_2^2\left(\left\|\nabla_{\boldsymbol{x}_j}\mathcal{L}\right\|^2 - \frac{1}{m}\left\langle \mathbf{1}, \nabla_{\boldsymbol{x}_j}\mathcal{L}\right\rangle^2 - \frac{1}{m}\left\langle \nabla_{\boldsymbol{x}_j}\mathcal{L}, \hat{\boldsymbol{x}}_j^1\right\rangle^2\right)}{\sigma_1^2\left(\left\|\nabla_{\boldsymbol{x}_j}\mathcal{L}\right\|^2 - \frac{1}{m}\left\langle \mathbf{1}, \nabla_{\boldsymbol{x}_j}\mathcal{L}\right\rangle^2 - \frac{1}{m}\left\langle \nabla_{\boldsymbol{x}_j}\mathcal{L}, \hat{\boldsymbol{x}}_j^2\right\rangle^2\right)} < \frac{\sigma_2^2}{\sigma_1^2} = \frac{|\boldsymbol{x} - \mu|_2^2}{|\boldsymbol{x} - \mu|_1^2}. \tag{12}$$

Moreover, as shown in Figure 7, we record the L1 norm and L2 norm of activations around all training stage. The L1 norm is extremely larger than L2 norm during all training stage, leading to more smooth loss landscape.

Next, we directly analyze the smoothing effect of the normalization layer, i.e., examine its impact on the Hessian matrix of the gradient.

Let $\hat{\boldsymbol{g}}_j = \nabla_{\boldsymbol{x}_j}\mathcal{L}$ and $\boldsymbol{H}_{jj} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_j \partial \boldsymbol{x}_j}$ be the gradient and Hessian of the loss with respect to the layer outputs respectively. Then:

$$\left(\nabla_{\boldsymbol{x}_j}\widehat{\mathcal{L}}\right)^\top \frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{x}_j \partial \boldsymbol{x}_j}\left(\nabla_{\boldsymbol{x}_j}\widehat{\mathcal{L}}\right)$$

TABLE II
RESULTS ON RESNETS

Dataset	Model	Baselines		CPT [23]	StateQuant [17]	Ultra-low [15]	Ours
		FP	INT8	INT 3-6/3-6/6	INT 8/8/5	radix-4 4/4/4	INT 4/4/4
CIFAR10	ResNet20	91.25	87.97	90.13	-	-	**90.41**
	ResNet38	92.04	89.39	91.24	-	-	**91.46**
	ResNet56	93.03	88.82	91.97	-	-	**92.03**
	ResNet18	93.31	91.73	92.74	92.85	**93.76**	92.97
	ResNet34	94.44	92.15	93.28	93.31	-	**93.67**
CIFAR100	ResNet18	74.36	70.35	72.83	73.51	-	**73.96**
	ResNet50	77.39	72.04	74.01	75.94	-	**76.30**
ImageNet	ResNet18	69.40	61.63	68.01	69.48	68.99	**69.02**
	ResNet50	76.48	68.94	74.73	74.45	**75.51**	74.84

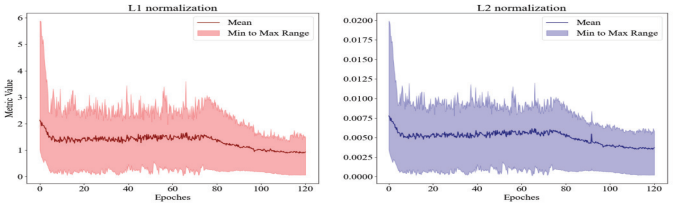


Fig. 7. The L1 norm and L2 norm of the activations during all training stage. We measure the activation before the first BN layer in ResNet20.

$$\leq \frac{\gamma^2}{\sigma^2}\left(\hat{\boldsymbol{g}}_j^\top \boldsymbol{H}_{jj}\hat{\boldsymbol{g}}_j - \frac{1}{m\gamma}\left\langle \hat{\boldsymbol{g}}_j, \hat{\boldsymbol{x}}_j\right\rangle \left\| \frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{x}_j}\right\|^2\right). \tag{13}$$

Similiar to the proof of Lipschitzness constant, we have:

$$\frac{\left(\nabla_{\boldsymbol{x}_j}\widehat{\mathcal{L}}_1\right)^\top \frac{\partial \widehat{\mathcal{L}}_1}{\partial \boldsymbol{x}_j \partial \boldsymbol{x}_j}\left(\nabla_{\boldsymbol{x}_j}\widehat{\mathcal{L}}_1\right)}{\left(\nabla_{\boldsymbol{x}_j}\widehat{\mathcal{L}}_2\right)^\top \frac{\partial \widehat{\mathcal{L}}_2}{\partial \boldsymbol{x}_j \partial \boldsymbol{x}_j}\left(\nabla_{\boldsymbol{x}_j}\widehat{\mathcal{L}}_2\right)} \leq \frac{\sigma_2^2}{\sigma_1^2}. \tag{14}$$

Therefore, both from the perspective of Lipschitzness constant and smoothing effects, the L1 normalization layer demonstrates significantly stronger regularization effects compared to the L2 normalization layer, which proves more advantageous for convergence under low-precision training conditions. The proof does not impose any assumptions on network structure or data distribution, thereby demonstrating strong universality. Detailed proof can be found in Appendix. C

V. EXPERIMENTS

A. Performance Analysis

We evaluate our integer training framework on ResNets [37], Transformers [38], [39], TGN (Temporal Graph Network [40]) and RNN (Recurrent Neural Network [41]). We also report the performance of recent state-of-art methods, including new data format methods CPT [23], Ultra-Low [15], reflection methods StateQuant [16], [17], and full integer training architecture [27]. In particular, CPT [23] achieves precise convergence by controlling the gradient bit-width with periodic variation between 3-8 bits. Ultra-Low [15] proposes a novel 4-bit fixed-point number type. StateQuant [17] develops a 5-bit quantizer that maps data into a quantization-friendly space before quantization. Similarly, [16] proposes a sub-8-bit quantization method. [27] introduces a new 8-bit data type. We apply ShiftQuant on activations and gradients, per-out-channel quantization on weights during forward propagation, and per-input-channel quantization on weights during loss propagation. We set the number of groups in ShiftQuant as 4 as default. We build two baselines. One is training in full-precision (fp32). The other is training in 8-bit integer (per-tensor quantization on weights, activations, and gradients). We adopt hyper-parameters, optimizers, and schedulers for all the evaluated models.

1) Image Classification: We select various network architectures and perform experiments on CIFAR10 [42], CIFAR100, and ImageNet [43] to validate the effectiveness of our method. In convolutional neural networks, we employ 4-bit weights, activations, and gradients, along with 8-bit fully-quantized L1 Batch Normalization (L1BNQ) on ResNets [37]. As shown in Table II, from the shallow ResNet18 to the deeper ResNet56, our method achieves accuracy comparable to that of the floating-point setup. For the fine-tuning tasks in Vision Transformers (ViT)[1] [39], we also use 4-bit weights, activations, and gradients. Since ViT pre-training utilized L2 Layer Normalization, we have to keep full-precision L2 layer normalization. It is a future work to quantize the L2 normalization layers of pre-trained models. As indicated in Table II, our approach achieves similar performance to the floating-point models in both the basic ViT-B and the expanded ViT-L models with negligible loss of accuracy. The experiments confirm the robustness of our method across different network architectures.

2) Machine Translation: We train a transformer-based architecture[2] on the WMT14 English-German (en-de) [44] and IWSLT14' En-De dataset. Within the entire network, the linear layers utilize 6-bit weights, activations, and gradients, along with an 8-bit fully-quantized L1 Batch Normalization (L1BN) layer. The computation of soft-max in attention mechanism is in floating-point. The results is reported in Table III. Our method achieves a negligible loss of accuracy.

3) Transductive and Inductive Prediction: We train a Temporal Graph Network (TGN [40]) on the Wikipedia[3] dataset.

[1]https://github.com/jeonsworld/ViT-pytorch
[2]https://github.com/facebookresearch/fairseq
[3]http://snap.stanford.edu/jodie/wikipedia.csv

TABLE III
RESULTS ON TRANSFORMERS

Dataset	Model	Traning type	Metric	Baselines		[16]	Ultra-low [15]	[27]	Ours
				FP	INT8	INT 4/4/4	Radix-4 4/4/4	INT 8/8/8	INT 6/6/6
CIFAR10	ViT-B	Fine-tune	Top1 Acc	98.85	94.03	98.36	-	-	**98.40**
	ViT-L	Fine-tune	Top1 Acc	98.90	93.69	98.47	-	98.80	**98.84**
WMT14	Transformer-	Pretrain	BLEU	27.54	21.13	**27.17**	25.4	-	27.09
IWSLT14'	based	Pretrain	BLEU	34.55	28.91	33.70	-	-	**33.81**

* [16] applies 8-bit integer format on gradients and pruns half of gradients.

TABLE IV
RESULTS ON TEMPORAL GRAPH NETWORK (TGN [40])

Dataset	Metric	Full-Precision	StateQuant	Ours
Wikipedia	AUC	0.936	0.939	0.939
	AP	0.945	0.945	0.947

TABLE V
RESULTS ON RECURRENT NEURAL NETWORK (RNN [41])

Dataset	Metric	Full-Precision	StateQuant	Ours
ElectricityLoad	ND	0.069	0.078	0.074
Diagrams20112014	RMSE	0.526	0.516	0.475

TABLE VI
PERFORMANCE UNDER DIFFERENT NUMBER OF GROUPS

number of groups	6	5	4	3	2	1
ResNet20	90.47	90.47	90.41	89.71	88.39	86.45
ViT-B	98.57	98.52	98.40	98.36	97.54	96.65

All the linear layers in TGN utilize 8-bit weights, activations, and gradients. Our method even surpasses the performance on full-precision training.

4) Time Series Prediction: We train a Recurrent Neural Network (RNN [41]) on the ElectricityLoadDiagrams2011-2014[4] dataset for time series prediction task. Our method achieves a negligible loss of accuracy.

Our method outperforms competitors across nearly all networks and tasks. In convolutional neural networks, our method achieves the smallest bitwidth. In transformers, although our method has a larger bitwidth compared to [16], it demonstrates superior hardware efficiency, which we will discuss in Sec. V-B. We are the first to evaluate integer training on both TGN and RNN architectures. Our method demonstrates strong accuracy retention, showcasing its applicability and robustness across diverse network architectures.

5) Analysis on Number of Groups: In ShiftQuant, an increase in the number of groups leads to finer quantization granularity, but an excessive number of groups can also increase the computational burden. We conducted experiments on ResNet20 and ViT-B using different group settings on the CIFAR10 dataset. As shown in Table VI, the performance

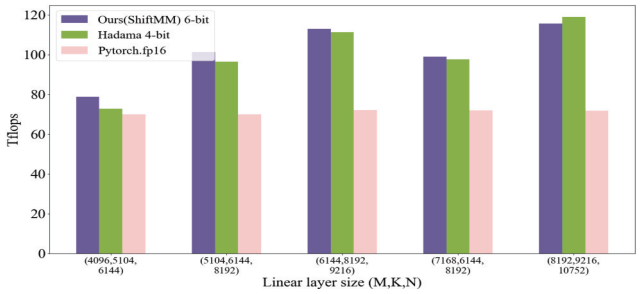


Fig. 8. Throughput of ShiftMM, Hadama [16], and Pytorch.fp16 on Nvidia RTX 3090.

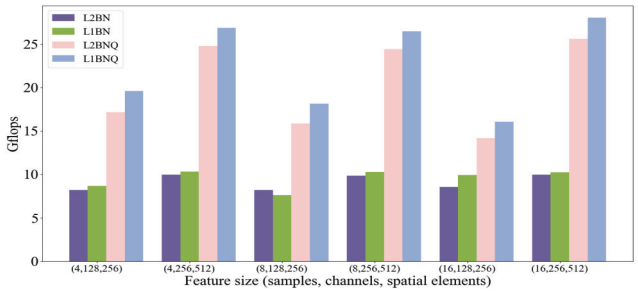


Fig. 9. Comparision of different normalization layers on ARMv8.

improvement reduces when number of groups increases. When the number of groups is 4, the grouping map can be represented using exactly 2 bits, while maintaining relatively high precision. Therefore, we set the number of groups as 4.

6) Ablation Study on L1 Normalization: As shown in Table VII, fully-quantized L2 normalization layers are not suitable for low-precision training. As a contrast, our fully-quantized L1 normalization layers achieve competitive performance.

B. Hardware Overhead

1) Throughput Analysis on ARMv8: To evaluate the efficiency of our method, we construct two baselines. The first is a fully-quantized linear layer, which utilizes the simplest per-tensor quantization and implements matrix multiplication by widely-used OpenBLAS.[5] This implementation reflects the upper bounds of efficiency under a given bitwidth. The second is made against a model using torch.fp16 precision.

[4]https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014

[5]https://www.openblas.net/

TABLE VII
ABLATION ON L1 NORMALIZATION

Dataset	Model	Metric	Baseline FP	QL1	QL2	L2
ImageNet	ResNet50	Top1 Acc	76.48	74.84	0.12	74.81
CIFAR10	ResNet20	Top1 Acc	91.25	90.40	13.34	90.78
WMT-14	Transformer-based	BLEU	27.5	27.09	1.78	27.05

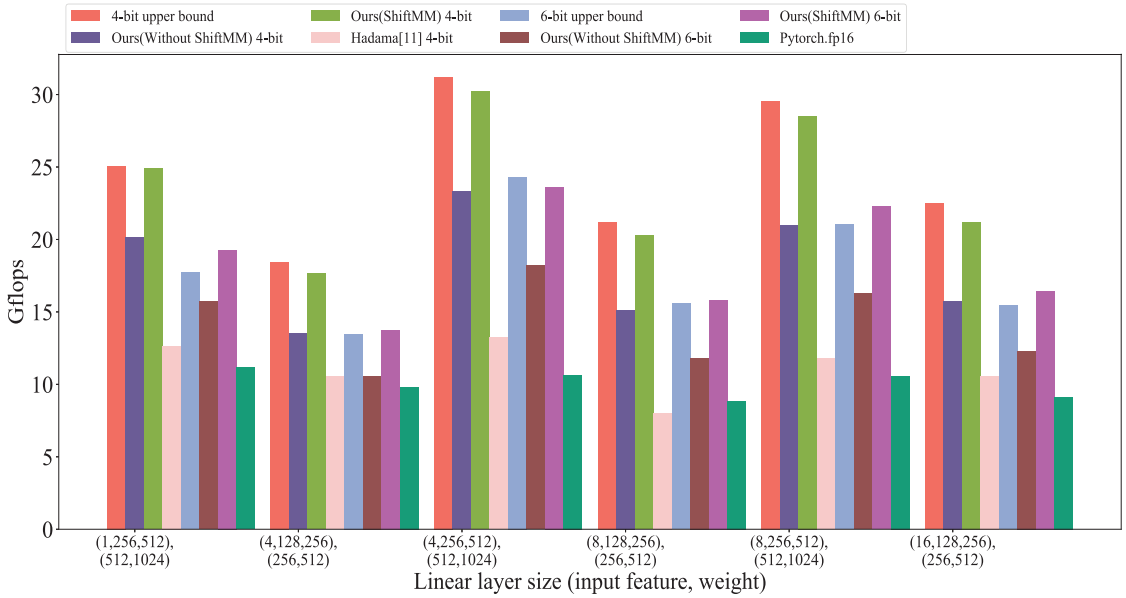


Fig. 10. Comparison of simplest per-tensor quantization (the upper bound of efficiency), ShiftQuant (GEMM implementation and ShiftMM implementation), [16], Pytorch.fp16 on ARMv8.

TABLE VIII
RESOURCE CONSUMPTION OF IMPLEMENTATION ON FPGA (ZC706)

	LUT priority						DSP priority					
Method	Baseline				Ours		Baseline				Ours	
Bitwidth	4	6	8	16(FP)	4	6	4	6	8	16(FP)	4	6
FF	414	576	744	1504	414	590	818	1042	1362	2978	994	1330
DSP	0	0	0	0	0	0	8	16	16	32	8	16
LUT	4671	5050	5406	9178	4799	5190	4595	4723	5619	5403	5203	5587
Latency (ms)	5.95	6.32	6.32	6.32	5.95	5.95	4.63	4.63	4.63	4.82	4.63	4.63

Meanwhile, we implement one comparative method [16]. We assess the throughput across various sizes of linear layers. As depicted in Figure 10, both the GEMM and ShiftMM implementations of our method significantly outperform the torch.fp16 model. ShiftMM closely matches the performance of the baseline 4-bit implementation. The performance gap between GEMM and ShiftMM implementation comes from memory rearrangement. Moreover, 6-bit ShiftMM outperforms 4-bit [16]. Two ingredients lead to this, the extra computation before quantization and hardware-unfriendly pruning in [16].

2) Throughput Analysis on GPU: As shown in Figure 8, we analyze the performance of our 6-bit ShiftMM, 4-bit [16], and Pytorch.fp16 on Nvidia RTX 3090. As the size grows, ShiftMM achieves more performance improvement than pytorch.fp16. Due to the heavy reflection operation and pruning, [16] does not take full advantage of the low bitwidth.

3) Resource Consumption on FPGA: To validate the efficiency of ShiftMM, we analyze the resource consumption of ShiftMM and basic GEMM on Xilinx ZC706 board. We perform a matrix multiplication with size (1024, 288, 32) on FPGA. For a comprehensive comparison, we select two resource bind strategy, LUT priority and DSP priority. We apply DSP reusing technique on implementation (details can be found in Appendix. E). As shown in Table VIII, in LUT priority setting, our INT6 implementation saves 60.7% FF and 43.5% LUT compared with FP16 implementation. In DSP reusing setting, we utilize DSP reusing technique. Our INT6 implementation saves 50% DSP, 55.3% FF with 3.4% overhead on LUT compared with FP16 implementation. Meanwhile, ShiftMM achieves closed resource utilization rate with the basic GEMM.

4) Time Proportion for Each Part of ShiftQuant: We evaluate the time proportion for each part of ShiftQuant on

TABLE IX
RESULTS ON DIFFERENT BIT-WIDTH

Dataset	Model	Metric	Baseline FP	4/4/4	5/5/5	6/6/6	7/7/7	8/8/8
ImageNet	ResNet50	Top1 Acc	76.48	74.84	75.08	75.27	75.42	75.57
WMT-14	Transformer-based	BLEU	27.5	25.11	26.53	27.09	27.23	27.31

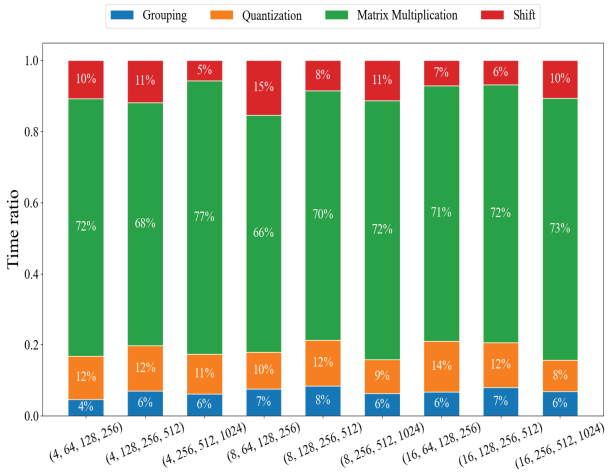


Fig. 11. Time proportion for each part of ShiftQuant.

TABLE X
END-TO-END ACCELERATION ON DIFFERENT PLATFORMS

Platform	ARMV8				RTX33090		
Batchsize	8	16	32	64	96	128	192
ViT-B-16	26%	35%	43%	56%	26%	34%	62%
ViT-L-16	31%	43%	55%	64%	33%	39%	68%

ARMV8. As depicted in Figure 11, the power-of-two grouping strategy and shift operation contribute minimally to the overall latency, with the majority of the time consumed by matrix computations. This observation aligns with the criteria for an efficient quantizer and further validates the superiority of ShiftQuant.

5) End-to-End Acceleration Performance: As shown in Table X, we evaluate the end-to-end training acceleration of the Vision Transformer (ViT) across various batch sizes on ARMV8 and RTX-3090. The subsequent tables illustrate the acceleration achieved by our method compared to FP16 counterparts. Our method markedly accelerates training. It is important to note that this was achieved through a basic implementation, without fully leveraging the potential of ShiftQuant. With further engineering optimizations, the acceleration performance could be significantly enhanced.

6) Performance Under Different Optimizer Settings: To validate the stability of the proposed integer training method under different optimizer parameter configurations, we conduct training experiments on ResNet-18 with various hyperparameter settings using the CIFAR-10 dataset.

As shown in Table. XI, the results demonstrate that the method maintains performance closely aligned with the floating-point (FP) baseline, confirming its robustness to variations in optimizer parameters.

TABLE XI
PERFORMANCE COMPARISON BETWEEN THE PROPOSED INTEGER TRAINING METHOD AND THE FP BASELINE UNDER DIFFERENT OPTIMIZER PARAMETER CONFIGURATIONS

lr	weight_decay	momentum	fp Top1 Acc	Ours
0.05	3e-4	0.7	91.24	90.84
		0.9	92.54	91.96
	5e-4	0.7	91.78	91.19
		0.9	92.76	92.14
	7e-4	0.7	91.87	91.32
		0.9	92.51	91.99
0.1	3e-4	0.7	92.31	91.98
		0.9	93.12	92.79
	5e-4	0.7	92.73	92.05
		0.9	93.31	92.97
	7e-4	0.7	92.58	92.14
		0.9	93.20	92.85
0.15	3e-4	0.7	92.08	91.73
		0.9	92.82	92.17
	5e-4	0.7	92.65	91.89
		0.9	93.03	92.76
	7e-4	0.7	92.18	91.65
		0.9	92.89	92.24

TABLE XII
COMPARISON OF COMPONENT-WISE LATENCY BETWEEN THE PROPOSED METHOD AND [16] ON GPU (TOTAL TIME AFTER 20 RUNS, METIRC:S)

Activation	Weight	[16]			Ours		
		Quant	Cal	Dequant	Quant	Cal	Dequant
128,128,256	256,512	0.093	0.091	0.034	0.47	0.037	0.039
128,256,512	512,1024	0.032	0.52	0.12	0.829	0.12	0.122
64,128,256	256,512	0.055	0.087	0.02	0.4	0.022	0.021
64,256,512	512,1024	0.168	0.272	0.065	0.58	0.07	0.065
32,128,256	256,512	0.034	0.086	0.014	0.37	0.016	0.011
32,256,512	512,1024	0.093	0.169	0.034	0.46	0.044	0.037
16,128,256	256,512	0.025	0.082	0.011	0.36	0.014	0.009
16,256,512	512,1024	0.056	0.167	0.021	0.4	0.026	0.021

7) Performance Under Different Bitwidth: We selected two challenging datasets, ImageNet and WMT14, and conducted integer quantization training experiments across multiple bit-widths for the ResNet and Transformer architectures respectively.

As shown in Table XI, with the increasing bit-width, the accuracy of our method progressively approaches floating-point training performance, which aligns with intuitive expectations.

VI. CONCLUSION

In this paper, we enhance sub-8-bit integer training from two aspects. We propose ShiftQuant to eliminate quantization noise in gradient estimation, and introduce fully-quantized

L1 normalization layers to smoothen the loss landscape for stable convergence. Comprehensive experiments validate the efficiency and accuracy of our method. Meanwhile, we have implemented ShiftQuant on multiple types of devices and proven its applicability. The first future direction is to further improve accuracy and efficiency, including developments in algorithms and implementation techniques. The second future direction is to apply ShiftQuant to other tasks, such as inference acceleration of large language models (LLMs). The excellent outlier suppression capacity of ShiftQuant may be useful for other challenging quantization tasks.

APPENDIX

A. Proof of the Equivalence Between Problem (1) and the Direct Optimization of Quantization Variance

Following Eq. 32, given a variable x, the quantization variance under stochastic rounding is:

$$\begin{aligned} Var[Q(x)] &= (l(x)-x)^2 \cdot p_Q(x) + (u(x)-x)^2 \cdot (1-p_Q(x)) \\ &= (x-l(x)) \cdot (u(x)-x), \end{aligned} \tag{15}$$

where $l(x)$ and $u(x)$ are lower quantization level and upper quantization level of x.

From the viewpoint of possibility, minimizing quantization variance is equal to minimize the expectation of quantization variance. The object function is:

$$\min \; E[Var(Q(x))]. \tag{16}$$

For Q_N levels quantization, above function can be represented as:

$$\begin{aligned} E[\mathrm{Var}(Q(x))] &= \sum_{m=1}^{Q_N} \left\{ \int_{l_m}^{u_m} (x-l_m) \cdot (u_m - x) \right\} \\ &= \sum_{m=1}^{Q_N} \left\{ \int_{l_m}^{u_m} (l_m+u_m) \cot x \cdot p(x) dx \right. \\ &\quad - \int_{l_m}^{u_m} x^2 \cdot p(x) dx \\ &\quad \left. - \int_{l_m}^{u_m} l_m u_m \cdot p(x) dx \right\}. \end{aligned} \tag{17}$$

$p(x)$ is the distribution of x. Any distribution can be representative as the mix of Laplace distributions with different parameters. Thus, $p(x)$ can be represent as:

$$p(x) = \sum_{k=1}^{\inf} \alpha_k \frac{1}{2\lambda_k} e^{-\frac{|x-\mu_k|}{\lambda_k}}. \tag{18}$$

Insert Eq. 18 to Eq. 17:

$$\begin{aligned} E[\mathrm{Var}(Q(x))] &= \sum_{k=1}^{\inf} \alpha_k \sum_{m=1}^{Q_N} \left\{ \int_{l_m}^{u_m} (l_m+u_m) \cot x \right. \\ &\quad \cdot \frac{1}{2\lambda_k} e^{-\frac{|x-\mu_k|}{\lambda_k}} dx \\ &\quad - \int_{l_m}^{u_m} x^2 \cdot \frac{1}{2\lambda_k} e^{-\frac{|x-\mu_k|}{\lambda_k}} dx \\ &\quad \left. - \int_{l_m}^{u_m} l_m u_m \cdot \frac{1}{2\lambda_k} e^{-\frac{|x-\mu_k|}{\lambda_k}} dx \right\}. \end{aligned} \tag{19}$$

We first analyze on one Laplace component:

$$\begin{aligned} t_k(x) &= \sum_{m=1}^{Q_N} \left\{ \int_{l_m}^{u_m} (l_m+u_m) \cot x \cdot \frac{1}{2\lambda_k} e^{-\frac{|x-\mu_k|}{\lambda_k}} dx \right. \\ &\quad - \int_{l_m}^{u_m} x^2 \cdot \frac{1}{2\lambda_k} e^{-\frac{|x-\mu_k|}{\lambda_k}} dx \\ &\quad \left. - \int_{l_m}^{u_m} l_m u_m \cdot \frac{1}{2\lambda_k} e^{-\frac{|x-\mu_k|}{\lambda_k}} dx \right\}, \text{ let } z = \frac{x-\mu_k}{\lambda_k} \\ &= \sum_{m=1}^{Q_N} \left\{ \overbrace{\frac{l_m+u_m}{2} \int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}} (\lambda_k z + \mu_k) e^{-|z|} dz}^{f^m(x)}, \right. \\ &\quad - \overbrace{\frac{1}{2} \int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}} (\lambda_k z + \mu_k)^2 e^{-|z|} dz}^{g^m(x)} \\ &\quad \left. - \overbrace{\frac{l_m u_m}{2} \int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}} e^{-|z|} dz}^{} \right\} \end{aligned} \tag{20}$$

Analysis on $f^m(x)$

$$\begin{aligned} &\int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}} (\lambda_k z + \mu_k) e^{-|z|} dz \\ &= \begin{cases} \lambda_k \left[\left(-\frac{u_m-\mu_k}{\lambda_k} + 1 \right) e^{-\frac{u_m-\mu_k}{\lambda_k}} \right. \\ \left. -(-l+1) e^{-\frac{l_m-\mu_k}{\lambda_k}} \right] \\ +\mu_k \left[e^{-\frac{l_m-\mu_k}{\lambda_k}} - e^{-\frac{u_m-\mu_k}{\lambda_k}} \right] \\ \frac{u_m-\mu_k}{\lambda_k} > \frac{l_m-\mu_k}{\lambda_k} > 0 \\ 2(\mu_k-\lambda_k) - \left(\lambda_k \frac{l_m-\mu_k}{\lambda_k} \right. \\ \left. +\mu_k - \lambda_k \right) e^{\frac{l_m-\mu_k}{\lambda_k}} \\ - \left(\lambda_k \frac{u_m-\mu_k}{\lambda_k} + \mu_k - \lambda_k \right) e^{-\frac{u_m-\mu_k}{\lambda_k}} \\ \frac{u_m-\mu_k}{\lambda_k} > 0 > \frac{l_m-\mu_k}{\lambda_k} \\ \lambda_k \left[\left(\frac{u_m-\mu_k}{\lambda_k} - 1 \right) e^{\frac{u_m-\mu_k}{\lambda_k}} \right. \\ \left. -(l-1) e^{\frac{l_m-\mu_k}{\lambda_k}} \right] \\ +\mu_k \left[e^{\frac{u_m-\mu_k}{\lambda_k}} - e^{\frac{l_m-\mu_k}{\lambda_k}} \right] \\ \frac{l_m-\mu_k}{\lambda_k} < \frac{u_m-\mu_k}{\lambda_k} < 0. \end{cases} \end{aligned} \tag{21}$$

With assumption of symmetry on quantization range, we have:

$$\begin{aligned} &\sum_{m=1}^{Q_N} \int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}} (\lambda_k z + \mu_k) e^{-|z|} dz \\ &= \lambda_k \left[\left(1 - \frac{u_{Q_N}-\mu_k}{\lambda_k} \right) e^{\frac{u_{Q_N}-\mu_k}{\lambda_k}} \right. \\ &\quad \left. -(l_1-1) e^{\frac{l_1-\mu_k}{\lambda_k}} \right] \\ &\quad + \mu_k \left[e^{\frac{l_1-\mu_k}{\lambda_k}} - e^{-\frac{u_{Q_N}-\mu_k}{\lambda_k}} \right] + 2(\mu_k-\lambda_k) \end{aligned} \tag{22}$$

For clarity, we represent above function as $\beta(u_{Q_N}, l_1), \beta(u_{Q_N}, l_1) = \sum_{m=1}^{Q_N} \int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}} (\lambda_k z + \mu_k) e^{-|z|} dz$. Then, we

have:

$$\begin{aligned}\sum_{m=1}^{Q_N} f^m(x) &= \left(\tau - \frac{\tau}{Q_N}\right)\beta\left(u_{Q_N}, l_0\right) - \frac{2\tau}{Q_N}\sum_{m=1}^{Q_N} f\left(u_m, l_1\right)\\ &= \frac{Q_N\tau - \tau}{Q_N}\cdot 2\lambda\cdot e^{-\frac{u_{Q_N}-\mu_k}{\lambda_k}} - \frac{2\tau}{Q_N}\sum_{m=1}^{Q_N}\lambda\Big[e^{-|u_m|}\\ &\quad + e^{-\frac{l_1-\mu_k}{\lambda_k}} - |u_m|\,e^{-|u_m|} - \frac{l_1-\mu_k}{\lambda_k}e^{-\frac{l_1-\mu_k}{\lambda_k}} - 2\Big]\\ &= 2\lambda\tau\frac{Q_N-1}{Q_N}e^{-\frac{u_{Q_N}-\mu_k}{\lambda_k}} - 2\lambda\tau\frac{1}{Q_N}\cdot(Q_N-1)\\ &\quad\cdot\left[e^{\frac{l_1-\mu_k}{\lambda_k}} - \frac{l_1-\mu_k}{\lambda_k}e^{-\frac{l_1-\mu_k}{\lambda_k}} - 2\right]\\ &\quad - \frac{2\lambda\tau}{Q_N}\sum_{m=1}^{Q_N}\left[e^{-|u_m|} - |u_m|\,e^{-|u_m|}\right]\\ &= 2\lambda\tau\frac{Q_N-1}{Q_N}\left(\frac{u_{Q_N}-\mu_k}{\lambda_k}e^{-\frac{u_{Q_N}-\mu_k}{\lambda_k}}\right)\\ &\quad - \frac{2\lambda\tau}{Q_N}\sum_{m=1}^{Q_N}\left[e^{-|u_m|} - |u_m|\,e^{-|u_m|}\right]\\ &\approx 2\lambda\tau\frac{Q_N-1}{Q_N}\left(\frac{u_{Q_N}-\mu_k}{\lambda_k}e^{-\frac{u_{Q_N}-\mu_k}{\lambda_k}}\right)\end{aligned} \tag{23}$$

where τ is the quantization range, and $u_{Q_N} = \frac{\tau}{2}, l_1 = -\frac{\tau}{2}$.

Analysis on $g^m(x)$

We first analyze $\int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}}(\lambda_k z+\mu_k)^2 e^{-|z|}dz$. Let $l = \frac{l_m-\mu_k}{\lambda_k}, u = \frac{u_m-\mu_k}{\lambda_k}$, we can find:

$$\begin{aligned}&\int_l^u (\lambda_k z+\mu_k)^2 e^{-|z|}dz\\ &= \begin{cases}\lambda^2[-z^2e^{-z} - 2ze^{-z} - 2e^{-z}]|_l^u\\ +2\lambda\mu[-ze^{-z} - e^{-z}]|_l^u + \mu^2[-e^{-z}]|_l^u,\\ u > l > 0\\ \lambda^2[(z^2-2z+2)e^z]|_l^0\\ +(-z^2+2z-2)e^{-z}|_0^u]+\\ 2\lambda\mu[(z-1)e^z|_l^0 + (-z+1)e^{-z}|_0^u]\\ +\mu^2[e^z|_l^0 + (-e^{-z})|_0^u],\\ u > 0 > l\\ \lambda^2[z^2e^z - 2ze^z + 2e^z]|_l^u +\\ 2\lambda\mu[ze^z - e^z]|_l^u + \mu^2[e^z]|_l^u,\\ l < u < 0.\end{cases}\end{aligned} \tag{24}$$

Accumulating the integers around different quantization levels:

$$\begin{aligned}&\sum_{m=1}^{Q_N}\int_{\frac{l_m-\mu_k}{\lambda}}^{\frac{u_m-\mu_k}{\lambda_k}}(\lambda_k z+\mu_k)^2\,e^{-|z|}dz\\ &= \lambda^2\left[-e^{-z}\left(z^2+2z+2\right)\right]\Big|_0^u + \mu^2\left[-e^{-z}\right]\Big|_0^u\\ &\quad + \lambda^2\left[e^z\left(z^2-2z+2\right)\right]\Big|_l^0 + \mu^2\left[e^z\right]\Big|_l^0\\ &= \lambda^2\left[-e^{-\frac{u_m-\mu_k}{\lambda_k}}\left(\frac{u_m-\mu_k^2}{\lambda_k} + 2\frac{u_m-\mu_k}{\lambda_k} + 2\right)\right]\\ &\quad + \mu^2\left[-e^{-\frac{u_m-\mu_k}{\lambda_k}} + 1\right] + \lambda^2\left[2 - e^{\frac{l_m-\mu_k}{\lambda_k}}\left(\frac{l_m-\mu_k^2}{\lambda_k}\right.\right.\\ &\quad \left.\left. -2\frac{l_m-\mu_k}{\lambda_k} + 2\right)\right] + \mu^2\left[1 - e^{\frac{l_m-\mu_k}{\lambda_k}}\right]\\ &= 2\mu^2\left(-e^{-\frac{u_m-\mu_k}{\lambda_k}} + 1\right)\\ &\quad + 2\lambda^2\left[-e^{-\frac{u_m-\mu_k}{\lambda_k}}\left(\frac{u_m-\mu_k}{\lambda_k} + 1\right)^2 - e^{-\frac{u_m-\mu_k}{\lambda_k}} + 2\right].\end{aligned} \tag{25}$$

Analysis on $q^m(x)$

We first analyze $\int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}} e^{-|z|}dz$. Let $l = \frac{l_m-\mu_k}{\lambda_k}, u = \frac{u_m-\mu_k}{\lambda_k}$, we can find:

$$\begin{aligned}&\int_l^u e^{-|z|}dz\\ &= \begin{cases}e^{-l} - e^{-u}, & u > l > 0\\ 2 - e^l - e^{-u}, & u > 0 > l\\ e^u - e^l, & l < u < 0.\end{cases}\end{aligned} \tag{26}$$

Thus we have:

$$\sum_{m=1}^{Q_N}\int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}} e^{-|z|} = 2 - 2e^{-u} \tag{27}$$

Following the same derivation as Eq. (23), we can find:

$$\sum_{m=1}^{Q_N}\frac{l_m+u_m}{2}\int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}} e^{-|z|} = \frac{\tau\cdot\left(\tau-\frac{2\tau}{Q_N}\right)}{2}(2-2e^{-u}). \tag{28}$$

Combine the result of Eq. (23), Eq. (25) and Eq. (28), we can find:

$$\begin{aligned}&t_k(x)\\ &= 2\lambda\tau\frac{Q_N-1}{Q_N}(ue_{-u} - 2) - \{\mu^2(-e^{-u}+1)\\ &\quad + \lambda^2[-e^{-u}(u+1)^2 - e^{-u} + 2]\} - \frac{\tau\cdot\left(\tau-\frac{2\tau}{Q_N}\right)}{2}(2-2e^{-u})\\ &= 2\lambda\tau\frac{Q_N-1}{Q_N}(ue_{-u} - 2) + \mu^2(e^{-u}-1)\\ &\quad + \lambda^2[e^{-u}(u+1)^2 + e^{-u} - 2] + \frac{\tau\cdot\left(\tau-\frac{2\tau}{Q_N}\right)}{2}(2e^{-u}-2),\end{aligned} \tag{29}$$

where $u = \frac{u_{Q_N}-\mu_k}{\lambda_k}$ and $u \gg 1$. Therefore, $t_k(x)$ is a monotonically increasing function of u at a single point. As the quantization range τ decreases, the value of $t_k(x)$ also decreases. It is evident that the same conclusion holds on $E[Var(Q(x))] = \sum_{k=1}^{\inf}\alpha_k t_k(x)$.

Now we consider quantizing a vector $\boldsymbol{x} \in \mathcal{NHW}$. The expectation of the maximum of x is [45], [46]:

$$E[max\,\boldsymbol{x}] \approx \sum_{k=1}^{\inf}(\mu_k + \lambda_k\ln(2NHW)) \tag{30}$$

The maximum of x is proportionate to λ_k. $E[Var(Q(x))]$ is a monotonically increasing function of $\frac{u_{Q_N}-\mu_k}{\lambda_k}$. Therefore, $E[Var(Q(x))]$ is also a monotonically increasing function of $\frac{u_{Q_N}}{\tau_{\boldsymbol{x}}}$ (In experiment, the distribution of gradients are extremely bell curve, which means μ_k is closed to 0). $\tau_{\boldsymbol{x}}$ is the magnitude of $\boldsymbol{x}$.

Now, we expand to grouping channels. The object of channel grouping is to minimize the quantization variance:

$$\min_{\tau_0,\tau_1,\cdots,\tau_{N_G-1},\tau_{N_G}} \quad \sum_{g=1}^{N_G}\sum_{i\in P^g} E[Var(\boldsymbol{x}_i)],$$
$$s.t.\ r_{max} = \tau_0 \geq \quad \tau_1 \geq \cdots \geq \tau_{N_G-1} \geq \tau_{N_G} = 0, \tag{31}$$

where $\boldsymbol{x}_i$ is the vectorized i-th channel. $E[Var(x_i)]$ is depend on the quantization range of the group and the range of x_i. As indicated before, $E[Var(x_i)]$ is a monotonically increasing function about $\frac{u_{Q_N}}{\tau_{x_i}}$, where $\tau_{\boldsymbol{x}_i}$ is the quantization range of $\boldsymbol{x}_i$. In group quantization, $u_{Q_N} = \tau_g$, where τ_g is the upper threshold of group g in Eq. (1). Hence that problem (31) and problem (1) has the same optimal point. Meanwhile, both of them are monotonically increasing about $\sum_{i\in P^g} \frac{\tau_g}{r_i}$.

B. Theoretical Analysis on ShiftQuant

1) Unbiased Quantizer: ShiftQuant utilizes stochastic rounding:

$$SR(x) = \begin{cases} u(x) & w.p. \frac{x-l(x)}{u(x)-l(x)} \\ l(x) & w.p. 1-\frac{x-l(x)}{u(l)-l(x)} \end{cases}, \tag{32}$$

where $u(x) = s^{-1}\lceil s\cdot x\rceil$ is the upper quantization level of x. $l(x) = s^{-1}\lfloor s\cdot x\rfloor$ is the lower quantization level of x. s is the quantization scale. It is clear that ShiftQuant is unbiased:

$$E[D_q(X)] = E[SR((X-ZI)\cdot S)\cdot S^{-1} + ZI] = X. \tag{33}$$

$X \in \mathbb{R}^{N\times D}$ is the quantized matrix. The second dimension of X is the inner dimension. $S = diag(s_0,\cdots,s_{D-1})$. s_i is the scale applied in the i-th index of the inner dimension and $s_i \in \{s_l, 2s_l,\cdots,2^{N_G-1}s_l\}$. s_l is the minimum scale. $Z = diag(z_0,z_1,\cdots,z_{D-1})$ represents the zero-point matrix. $I \in \mathbb{R}^{D\times D}$ is a identity matrix.

2) Low Variance: The up bound of ShiftQuant's variance U_{dq} satisfies:

$$U_{dq} \leq \alpha\cdot\frac{N}{4}\sum_{j=1}^{D} s_j^{-2} + 2^{-N_G}\frac{ND}{4}s^{-2}, \tag{34}$$

where α is depended on the distribution of channels' range, and $1 \leq \alpha \leq 4$. $\frac{N}{4}\sum_{j=1}^{D} s_j^{-2}$ is the up bound of fine-grained quantization's variance. $\frac{ND}{4}s^{-2}$ is the up bound of coarse-grained quantization's variance. Eq. (34) demonstrates that ShiftQuant achieves a closed performance to fine-grained quantizers.

3) Proof of Unbiased Quantization:

$$\begin{aligned} E[D_q(X)] &= E[SR((X-ZI)\cdot S)\cdot S^{-1} + ZI] \\ &= E[SR(X-ZI)]\cdot S\cdot S^{-1} + ZI \\ &= (X-ZI)\cdot S\cdot S^{-1} + ZI \\ &= X. \end{aligned} \tag{35}$$

Clearly, ShiftQuant is a unbiased quantizer.

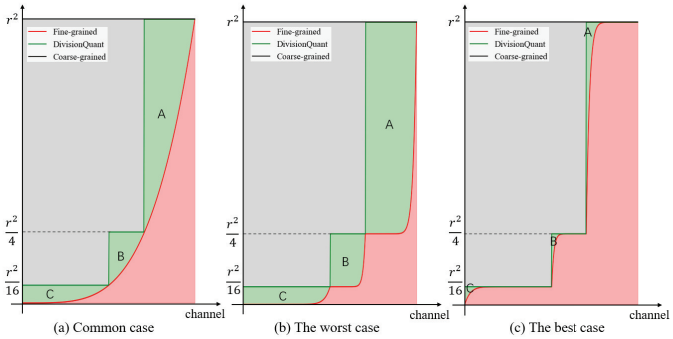


Fig. 12. Quantization variance of coarse-grained quantization, fine-grained quantization, and ShiftQuant. We set number of groups as 3 in visualization. We sort the channels in ascending order by the range of channels.

4) Proof of Low Variance: From Eq. (15), we can find the up bound of quantization variance:

$$Var[SR(x)] = (x-l(x))(u(x)-x) \leq \frac{[u(x)-l(x)]^2}{4}, \tag{36}$$

where $u(x), l(x)$ are the neighbouring quantization levels, and $u(x)-l(x) = \frac{r}{B} = s^{-1}$. r is the quantization range, and B is the number of quantization bins.

Now we expand Eq. (36) to $X \in \mathcal{R}^{N\times D}$. For coarse-grained quantization, we have:

$$Var[Q_{pt}(X)] = \sum_{i=1}^{N}\sum_{j=1}^{D}\frac{r^2}{4B^2} = \frac{ND}{4}s^{-2}, \tag{37}$$

where r is the range of X, and $r = max\,X - min\,X$. $s_{pt} = \frac{r}{B}$ is the quantization scale.

For per-channel quantization, we have:

$$Var[Q_{pt}(X)] = \sum_{i=1}^{N}\sum_{j=1}^{D}\frac{r_j^2}{4B^2} = \frac{N}{4}\sum_{j=1}^{D} s_j^{-2}, \tag{38}$$

where $r_j = max\,X_{:,j} - min\,X_{:,j}$, $s_j = \frac{r_j}{B}$.

We visualize the relation between fine-grained, coarse-grained quantization, and ShiftQuant in Figure 12.

As shown in Figure 12(b), in the worst case, the gap of channels' ranges in a group is maximum. In this situation, we can find that:

$$\begin{aligned} U_{dq} &= U_{fq} + U_A + U_B + U_C \\ &\leq U_{fq} + U_{fq} * 3 + U_C \\ &= U_{fq} * 4 + U_C, \end{aligned} \tag{39}$$

where U_{fq} is the variance of fine-grained quantization. U_A, U_B, and U_C denote the area of A, B, and C in Figure 12. For U_C, we have:

$$U_C < 2^{-2*N_G+2}\cdot U_{cq}, \tag{40}$$

where U_{cq} is the variance of coarse-grained quantization. With combining Eq. (39) and Eq. (40), we can find:

$$U_{dq} \leq 4U_{fq} + 2^{-2*N_G+2}\cdot U_{cq}. \tag{41}$$

As shown in Figure 12(c), in the best case, the channel's ranges in each group is equal. The area of A, B, and C are closed to 0. The variance of ShiftQuant satisfies:

$$U_{dq} = U_{fq} + U_A + U_B + U_C$$

$$\approx U_{fq}$$
$$\leq U_{fq} + 2^{-2*N_G+2} \cdot U_{cq}. \tag{42}$$

With combining Eq. (41) and Eq. (42), we can find:

$$U_{dq} \leq \alpha * U_{fq} + 2^{-2*N_G+2} \cdot U_{cq}$$
$$= \alpha \cdot \frac{N}{4} \sum_{j=1}^{D} s_j^{-2} + 2^{-2N_G+2} \frac{ND}{4} s^{-2}, \tag{43}$$

where $1 \leq \alpha \leq 4$.

C. Proof of L1 Normalization

Regarding the Hessian matrix of the gradient, we have:

$$\begin{aligned}
\frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{x}_j \partial \boldsymbol{x}_j} &= \frac{\partial}{\partial \boldsymbol{x}_j} \left(\left(\frac{\gamma}{m\sigma_j} \right) \left[m\hat{\boldsymbol{g}}_j - m\boldsymbol{\mu}_{(\hat{\boldsymbol{g}}_j)} - \hat{x}_j^{(b)} \langle \hat{\boldsymbol{g}}_j, \hat{\boldsymbol{x}}_j \rangle \right] \right) \\
&= \frac{\gamma}{m\sigma} \left(\frac{\partial}{\partial \boldsymbol{y}_q} \left[m\hat{\boldsymbol{g}}_j - m\boldsymbol{\mu}_{(\hat{\boldsymbol{g}}_j)} - \hat{\boldsymbol{x}}_j \langle \hat{\boldsymbol{g}}_j, \hat{\boldsymbol{x}}_j \rangle \right] \right) \cdot \frac{\partial \boldsymbol{y}_q}{\partial \boldsymbol{x}_j} \\
&\quad + \left(\frac{\partial}{\partial \boldsymbol{x}_j} \left(\frac{\gamma}{m\sigma_j} \right) \right) \cdot \left(m\hat{\boldsymbol{g}}_j - m\boldsymbol{\mu}_{(\hat{\boldsymbol{g}}_j)} \right. \\
&\quad \left. - \hat{\boldsymbol{x}}_j \langle \hat{\boldsymbol{g}}_j, \hat{\boldsymbol{x}}_j \rangle \right) \\
&= \left(\frac{\gamma}{\sigma_j} \right) \left(\boldsymbol{H}_{jj} - \frac{\partial \boldsymbol{\mu}_{(\hat{\boldsymbol{g}}_j)}}{\partial \boldsymbol{x}_j} - \frac{\partial}{\partial \boldsymbol{y}_j} \left(\frac{1}{m} \hat{\boldsymbol{x}}_j \langle \hat{\boldsymbol{g}}_j, \hat{\boldsymbol{x}}_j \rangle \right) \right) \\
&\quad \cdot \frac{\partial \boldsymbol{y}_j}{\partial \boldsymbol{x}_j} \\
&\quad + \left(\hat{\boldsymbol{g}}_j - \boldsymbol{\mu}_{(\hat{\boldsymbol{g}}_j)} - \frac{1}{m} \hat{\boldsymbol{x}}_j \langle \hat{\boldsymbol{g}}_j, \hat{\boldsymbol{x}}_j \rangle \right) \left(\frac{\partial}{\partial \boldsymbol{x}_j} \left(\frac{\gamma}{\sigma_j} \right) \right).
\end{aligned} \tag{44}$$

The detailed derivation process can be found in [34].

D. Detailed Implementation of ShiftQuant

The gradient $\boldsymbol{G_B} \in \mathbb{R}^{N_B \times C_B}$ is first partitioned into N_G distinct groups. Each group possesses a unique scaling factor. The g-th group $(\boldsymbol{G_B})^g \in \mathbb{R}^{N_B \times C_{N_g}}$ undergoes quantization according to the following formulation:

$$(\boldsymbol{G_B})_q^g = round \left(\frac{(\boldsymbol{G_B})^g}{\frac{\tau_0}{B} \cdot 2^{-g}} \right) = round \left(\frac{(\boldsymbol{G_B})^g}{s_{\boldsymbol{G_B}}^{-1} \cdot 2^{-g}} \right), \tag{45}$$

where τ_0 represents the magnitude of $\boldsymbol{G_B}$. Naturally, we have:

$$\begin{aligned}
(\boldsymbol{G_B})^g \cdot (\boldsymbol{W}^T)^g &= [s_{\boldsymbol{G_B}}^{-1} \cdot 2^{-g} \cdot (\boldsymbol{G_B})_q^g] \cdot [s_{\boldsymbol{W}}^{-1} \cdot (\boldsymbol{W}^T)_q^g] \\
&= s_{\boldsymbol{G_B}}^{-1} \cdot s_{\boldsymbol{W}}^{-1} \cdot [(\boldsymbol{G_B})_q^g \cdot (\boldsymbol{W}^T)_q^g \gg g],
\end{aligned} \tag{46}$$

where $\boldsymbol{W} \in \mathbb{R}^{C_A \times C_B}$, and $(\boldsymbol{W})_q^g \in \mathbb{R}^{C_A \times C_{N_g}}$. With aggregating the results from all N_G groups, we have:

$$\begin{aligned}
\boldsymbol{G_A} &= \sum_{g=1}^{N_G} (\boldsymbol{G_B})^g \cdot (\boldsymbol{W}^T)^g \\
&= s_{\boldsymbol{G_B}}^{-1} \cdot s_{\boldsymbol{W}}^{-1} \sum_{g=1}^{N_G} [(\boldsymbol{G_B})^g{}_q \cdot (\boldsymbol{W}^T)^g{}_q \gg g],
\end{aligned} \tag{47}$$

In the code implementation, we substitute the right shift operation with a left shift operation. This modification ensures that the outcome is perfectly congruent with the original dot product computation:

$$\begin{aligned}
& s_{\boldsymbol{G_B}}^{-1} \cdot s_{\boldsymbol{W}}^{-1} \cdot 2^{-N_G} \sum_{g=1}^{N_G} [(\boldsymbol{G_B})^g{}_q \cdot (\boldsymbol{W}^T)^g{}_q \ll (N_g - g)] \\
&= \sum_{g=1}^{N_G} s_{\boldsymbol{G_B}}^{-1} \cdot s_{\boldsymbol{W}}^{-1} \cdot 2^{-N_G} [(\boldsymbol{G_B})^g{}_q \cdot (\boldsymbol{W}^T)^g{}_q \ll (N_g - g)] \\
&= \sum_{g=1}^{N_G} s_{\boldsymbol{G_B}}^{-1} \cdot (\boldsymbol{G_B})^g{}_q \cdot (2^{-N_G} \ll (N_g - g)) \cdot s_{\boldsymbol{W}}^{-1} \cdot (\boldsymbol{W}^T)^g{}_q \\
&= \sum_{g=1}^{N_G} s_{\boldsymbol{G_B}}^{-1} \cdot (\boldsymbol{G_B})^g{}_q \cdot 2^{-g} \cdot s_{\boldsymbol{W}}^{-1} \cdot (\boldsymbol{W}^T)^g{}_q \\
&= \sum_{g=1}^{N_G} (\boldsymbol{G_B})^g \cdot (\boldsymbol{W}^T)^g \\
&= \boldsymbol{G_A}.
\end{aligned} \tag{48}$$

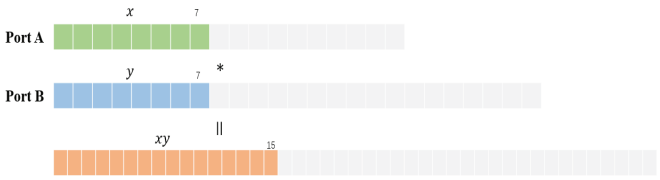


Fig. 13. Direct utilization of DSP on 8-bit multiplication. Unused bits is colored in gray. A lot of bitwidth is wasted.

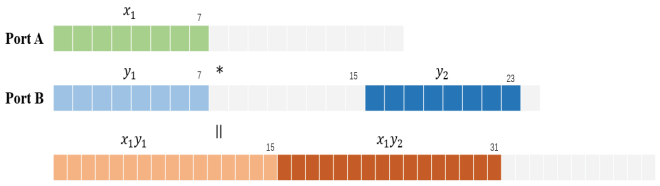


Fig. 14. DSP reusing on 8-bit multiplication. One calculation can realize two multiplications.

E. Implementation of DSP Reusing on FPGA

In the Xilinx ZC706 board, the interface for the multiplier within the FPGA's DSP module (Xilinx DSP48E1) is configured for 25-bit and 18-bit inputs. In low-bitwidth computations, the direct utilization of DSPs leads to the wastage of bit width (as shown in Figure 13). DSP reusing addresses this issue by packing multiple low-bitwidth data into a single data unit sized to the bit width. This approach enables the execution of multiple multiplications in a single computation cycle.

As shown in Figure 14 and 15, for 8-bit and 6-bit multiplications, the Xilinx DSP48E1 block can achieve a maximum of dual multiplexing, allowing for the execution of two multiplications in a single operation. This capability is also the reason why the DSP resource consumption for both 6-bit and 8-bit implementations is the same, as indicated in Table VIII.

As shown in Figure 16, the DSP48E1 block is capable of simultaneously executing four 4-bit multiplications. This

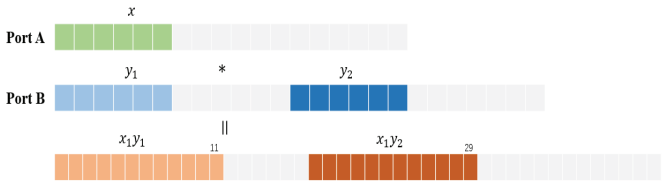


Fig. 15. DSP reusing on 6-bit multiplication. One calculation can realize two multiplications.

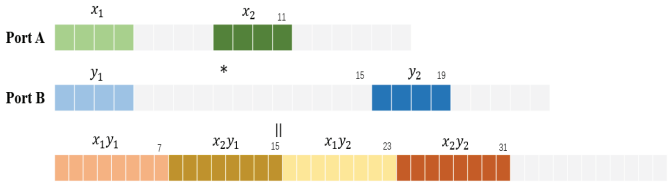


Fig. 16. DSP reusing on 4-bit multiplication. One calculation can realize four multiplications.

aligns with the data presented in Table VIII, where the DSP resource consumption for the 4-bit implementation is half that of the 6-bit and 8-bit implementations.

F. Time Consumption of Each Component on GPU

Here we present the component-wise latency comparison (quantization, calculation, dequantization) between the proposed method and [16] during matrix computation on the GPU platform (NVIDIA RTX-3090).

REFERENCES

[1] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.

[2] Y. Li, X. Dong, and W. Wang, "Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks," 2019, *arXiv:1909.13144*.

[3] S. Uhlich et al., "Differentiable quantization of deep neural networks," 2019, *arXiv:1905.11452*.

[4] Z. Liu, K.-T. Cheng, D. Huang, E. Xing, and Z. Shen, "Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 4942–4952.

[5] M. Lin et al., "HRank: Filter pruning using high-rank feature map," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1529–1538.

[6] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4340–4349.

[7] G. Fang, X. Ma, and M. Song, "Depgraph: Towards any structural pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2023, pp. 16091–16101.

[8] T. Chu, Z. Yang, and X. Huang, "Improving the post-training neural network quantization by prepositive feature quantization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 4, pp. 3056–3060, Apr. 2024.

[9] W. Xu et al., "Improving extreme low-bit quantization with soft threshold," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 4, pp. 1549–1563, Apr. 2023.

[10] J. Shi, M. Lu, and Z. Ma, "Rate-distortion optimized post-training quantization for learned image compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 5, pp. 3082–3095, May 2024.

[11] S. Y. Chang and H.-C. Wu, "Tensor quantization: High-dimensional data compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 8, pp. 5566–5580, Aug. 2022.

[12] Y. Xie, X. Hou, Y. Guo, X. Wang, and J. Zheng, "Joint-guided distillation binary neural network via dynamic channel-wise diversity enhancement for object detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 1, pp. 448–460, Jan. 2023.

[13] R. Banner, I. Hubara, E. Hoffer, and D. Soudry, "Scalable methods for 8-bit training of neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, May 2018, pp. 5145–5153.

[14] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.

[15] X. Sun et al., "Ultra-low precision 4-bit training of deep neural networks," in *Proc. NIPS*, vol. 33, 2020, pp. 1796–1807.

[16] H. Xi, C. Li, J. Chen, and J. Zhu, "Training transformers with 4-bit integers," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2023, pp. 49146–49168.

[17] J. Chen, Y. Gai, Z. Yao, M. W. Mahoney, and J. E. Gonzalez, "A statistical framework for low-bitwidth training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2020, pp. 883–894.

[18] M. Cacciola, A. Frangioni, M. Asgharian, A. Ghaffari, and V. Partovi Nia, "On the convergence of stochastic gradient descent in low-precision number formats," 2023, *arXiv:2301.01651*.

[19] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," 2020, *arXiv:2010.01412*.

[20] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2017, pp. 6391–6401.

[21] L. Cambier, A. Bhiwandiwalla, T. Gong, M. Nekuii, O. H. Elibol, and H. Tang, "Shifted and squeezed 8-bit floating point format for low-precision training of deep neural networks," 2020, *arXiv:2001.05674*.

[22] K. Zhong et al., "Exploring the potential of low-bit training of convolutional neural networks," *IEEE Trans. Comput.-Aided Desgin Integr. Circuits Syst.*, vol. 41, no. 12, pp. 5421–5434, Dec. 2022.

[23] Y. Fu et al., "CPT: Efficient deep neural network training via cyclic precision," 2021, *arXiv:2101.09868*.

[24] Q. Wu, Y. Li, S. Chen, and Y. Kang, "DRGS: Low-precision full quantization of deep neural network with dynamic rounding and gradient scaling for object detection," in *Proc. Int. Conf. Data Mining Big Data*, Jan. 2022, pp. 137–151.

[25] D. Das et al., "Mixed precision training of convolutional neural networks using integer operations," 2018, *arXiv:1802.00930*.

[26] B. Chmiel, R. Banner, E. Hoffer, H. B. Yaacov, and D. Soudry, "Logarithmic unbiased quantization: Simple 4-bit training in deep learning," 2021, *arXiv:2112.10769*.

[27] A. Ghaffari, M. S. Tahaei, M. Tayaranian, M. Asgharian, and V. P. Nia, "Is integer arithmetic enough for deep learning training?," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2022, pp. 27402–27413.

[28] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "HAWQ: Hessian aware quantization of neural networks with mixed-precision," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*. Seoul, South Korea, Oct./Nov. 2019, pp. 293–302.

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[30] J. Lei Ba, J. Ryan Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[31] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," 2016, *arXiv:1607.08022*.

[32] Y. Wu and K. He, "Group normalization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.

[33] G. Xiao et al., "SmoothQuant: Accurate and efficient post-training quantization for large language models," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 38087–38099.

[34] S. Santurkar, D. Tsipras, A. Ilyas, and A. Ma
dry, "How does batch normalization help optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, Dec. 2018, pp. 2488–2498.

[35] P. J. Huber, *Robust Statistical Procedures*. Philadelphia, PA, USA: SIAM, 1996.

[36] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

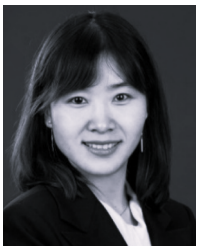
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[38] A. Vaswani et al., "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 5998–6008, Jun. 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID
[39] A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.
[40] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1269–1278.
[41] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design Appl.*, vol. 5, nos. 64–67, p. 2, 2001.
[42] A. Krizhevsky and G. Hinton. (2009). *Learning Multiple Layers of Features from Tiny Images*. [Online]. Available: https://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf
[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[44] O. Bojar et al., "Findings of the 2014 workshop on statistical machine translation," in *Proc. 9th Workshop Stat. Mach. Transl.*, May 2014, pp. 12–58.
[45] B. C. Arnold, N. Balakrishnan, and H. N. Nagaraja, *A First Course in Order Statistics*. Philadelphia, PA, USA: SIAM, 2008.
[46] H. A. David and H. N. Nagaraja, *Order Statistics*. Hoboken, NJ, USA: Wiley, 2004.

Wenjin Guo received the B.E. degree in telecommunications engineering from Xidian University, Xi'an, China, in 2020, where he is currently pursuing the Ph.D. degree with the State Key Laboratory of Integrated Service Network.

His research interests include machine learning and high-performance computing.

Donglai Liu received the B.E. degree in telecommunications engineering from Xidian University, Xi'an, China, in 2023, where he is currently pursuing the master's degree with the State Key Laboratory of Integrated Service Network.

His research interests include distribute learning, model compression, and deployment.



Weiying Xie (Senior Member, IEEE) received the Ph.D. degree in communication and information systems from Xidian University, Xi'an, in 2017.

She is currently a Professor with the State Key Laboratory of Integrated Services Networks, Xidian University. She has published over 50 articles in refereed journals and proceedings, including IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, and Conference on CVPR, AAAI. Her research interests include neural networks, machine learning, hyperspectral image processing, and high-performance computing.

Yunsong Li received the M.S. degree in telecommunication and information systems and the Ph.D. degree in signal and information processing from Xidian University, China, in 1999 and 2002, respectively.

He joined the school of Telecommunications Engineering, Xidian University, in 1999, where he is currently a Professor. He is currently the Director of the Image Coding and Processing Center, State Key Laboratory of Integrated Service Networks. His research interests include image and video processing and high-performance computing.

Xuefei Ning received the B.S. and Ph.D. degrees from Tsinghua University in 2016 and 2021, respectively. He is currently a Research Assistant Professor with the Department of Electronic Engineering, Tsinghua University. His research interests include efficient deep learning.

Zihan Meng received the B.S. degree from Xidian University, Xi'an, China, in 2023, where he is currently pursuing the Ph.D. degree with the State Key Laboratory of Integrated Service Network.

His research interests include model compression and deep learning.

Shulin Zeng (Member, IEEE) received the B.S. and Ph.D. degrees from Tsinghua University in 2014 and 2024, respectively. His research interests include software-hardware co-design for deep learning and virtualization in the cloud.

Jie Lei (Member, IEEE) received the M.S. degree in telecommunication and information systems and the Ph.D. degree in signal and information processing from Xidian University, Xi'an, China, in 2006 and 2010, respectively. He was a Visiting Scholar with the Department of Computer Science, University of California at Los Angeles, Los Angeles, CA, USA, from 2014 to 2015. Currently, he is a Research Fellow at the School of Electrical and Data Engineering, University of Technology Sydney. His research interests include wireless communication, remote sensing image processing, machine learning, and customized computing for big data applications.

Zhenman Fang (Senior Member, IEEE) received the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2014. He did his post-doctoral research at the University of California at Los Angeles (UCLA), Los Angeles, CA, USA, from 2014 to 2017. He was a Staff Software Engineer with Xilinx, San Jose, CA, USA, from 2017 to 2019. He is currently an Assistant Professor with the School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada. His research interests include customizable computing with specialized hardware acceleration, including emerging application characterization and acceleration, novel accelerator-rich and near-data computing architecture designs, and corresponding programming, runtime, and tool support. He is a member of the Association for Computing Machinery (ACM).

Yu Wang (Fellow, IEEE) received the B.S. and Ph.D. degrees (Hons.) from Tsinghua University, Beijing, in 2002 and 2007, respectively. He is currently a tenured Professor with the Department of Electronic Engineering, Tsinghua University. He has authored and co-authored more than 350 papers in refereed journals and conferences. His research interests include brain-inspired computing, parallel circuit analysis, application-specific acceleration and power/reliability aware circuit, and system design methodology. He served as a Program Committee Member for leading conferences in these areas, including top EDA conferences, such as DAC, DATE, ICCAD, and ASP-DAC, and top FPGA conferences, such as FPGA and FPT. He received the Best Paper Award from ASPDAC 2019, FPGA 2017, NVMSA 2017, and ISVLSI 2012, the Best Poster Award from HEART 2012, and the 11 best paper nominations (DAC22, ICT18, DATE18, DAC17, ASPDAC16, ASPDAC14, ASPDAC12, two in ASPDAC10, ISLPED09, and CODES09). He was a recipient of the Alexander von Humboldt Fellowship in 2019, the DAC Under-40 Innovators Award in 2018, and the IBM X10 Faculty Award in 2010. He served as the TPC Chair for ICFPT 2019 and 2011 and ISVLSI 2018, the Finance Chair for ISLPED 2012–2016, and the Track Chair for DATE 2017–2019 and GLSVLSI 2018. He serves as an Associate Editor for IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, *ACM Transactions on Design Automation of Electronic Systems*, and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.