TOWARDS ACCURATE AND EFFICIENT SUB-8-BIT INTEGER TRAINING

Anonymous authors
Paper under double-blind review

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 $1.85\times/15.3\%$ 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.

1 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 Jacob et al. (2018); Li et al. (2019); Uhlich et al. (2019); Liu et al. (2022); Lin et al. (2020); He et al. (2019); Fang et al. (2023), it should be noted that the computational complexity of training is approximately three times that of inference Banner et al. (2018). 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 Zhou et al. (2016). 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 Sun et al. (2020); Xi et al. (2023); Chen et al. (2020). The second is the sharp loss landscape of low-precision networks Cacciola et al. (2023). Sharp local minimal points disrupt optimization significantly Foret et al. (2020); Li et al. (2018).

To address the challenge of wide gradient range and significant quantization errors, common approaches include (1) employing a non-uniform quantization format Sun et al. (2020); Cambier et al. (2020); Zhong et al. (2022), (2) utilizing smaller quantization granularity, or (3) suppressing outliers before quantization Xi et al. (2023); Chen et al. (2020). 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. 2). 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 Xi et al. (2023); Chen et al. (2020); Sun et al. (2020); Fu et al. (2021); Wu et al. (2022); Das et al. (2018); Banner et al. (2018); Chmiel et al. (2021); Ghaffari et al. (2022). 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 Dong et al. (2019); Foret et al. (2020). 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 Ioffe & Szegedy (2015); Ba et al. (2016); Ulyanov et al. (2016); Wu & He (2018). 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 $1.85\times/15.3\%$ 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.

2 Related Work and Challenges

2.1 New Quantization Data Formats

Developing new data formats with wide representing range reduces quantization loss significantly. Sun et al. Sun et al. (2020) 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. Fu et al. (2021) explores a dynamic data format with varying bit-widths to balance accuracy and efficiency by periodically adjusting the bit-width. Cambier et al. Cambier et al. (2020) 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. Zhong et al. (2022) 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.

2.2 Fine-grained Quantization

Fine-grained quantization can reduce quantization error significantly. However, naive fine-grained quantization may lead to incompatibility with GEMM Xiao et al. (2023); Zhong et al. (2022). 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

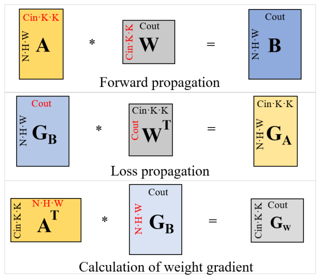


Figure 1: Matrix multiplications in training. Inner dimensions are labeled in red.

Table 1: Difference between our method and prior studies.

	New data format Sun et al. (2020), Fu et al. (2021), Cambier et al. (2020)	Fine-grained Das et al. (2018), Wu et al. (2022)	Outlier suppression before quantization Chen et al. (2020), Xi et al. (2023)	Ours
General data format	✘	✔	✔	✔
GEMM supporting	-	✘	✔	✔
No pre-processing on quantization	✔	✔	✘	✔
Fully-quantized normalization layers	✘	✘	✘	✔

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.

2.3 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. Chen et al. (2020) implements the reflection by multiplying gradients with a constructed Hadamard matrix. Similarly, Xi et al. Xi et al. (2023) 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 1.

3 ShiftQuant

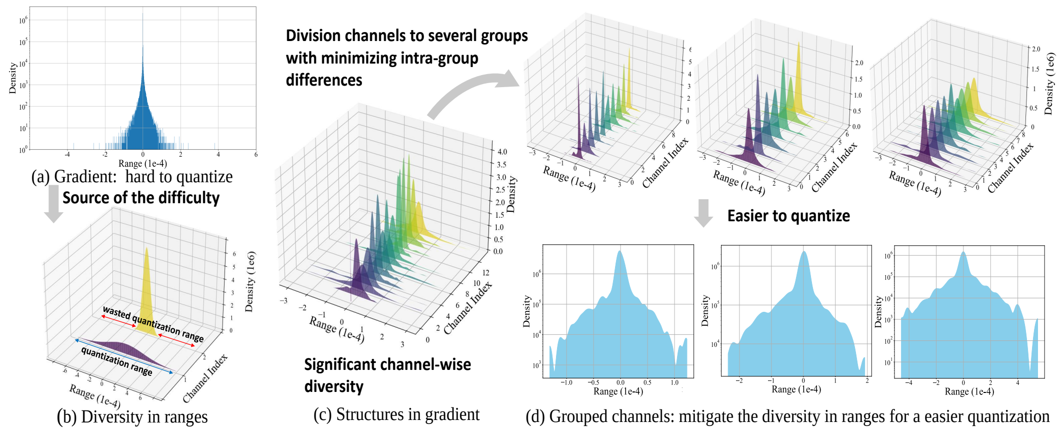


Figure 2: The difficulty of quantizing gradients comes from the diversity between channels. ShiftQuant aims to minimize the diversity through strategic grouping channels. (a) Gradients exhibit extremely bell-curve distribution, which is very difficult to quantize. (b) Diversity in magnitude of channels 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.

3.1 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.

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:

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

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.

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 ($\tau_p - r_i = 2(\tau_q - r_j)$). 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:

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

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 3(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.

3.2 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.

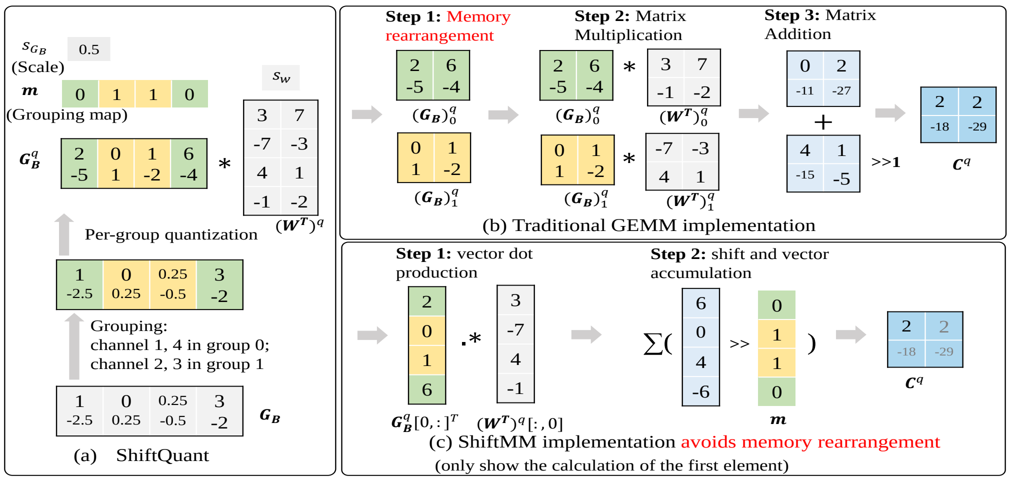

s_{G_B} (Scale) 0.5
m (Grouping map) 0 1 1 0
s_W
G_B^q
$(W^T)^q$
Per-group quantization
Grouping: channel 1, 4 in group 0; channel 2, 3 in group 1
G_B
(a) ShiftQuant
Step 1: Memory rearrangement
Step 2: Matrix Multiplication
Step 3: Matrix Addition
$(G_B)_0^q$
$(G_B)_1^q$
$(W^T)_0^q$
$(W^T)_1^q$
>>1
C^q
(b) Traditional GEMM implementation
Step 1: vector dot production
Step 2: shift and vector accumulation
$G_B^q[0,:]^T$
$(W^T)^q[:,0]$
m
C^q
(c) ShiftMM implementation avoids memory rearrangement
(only show the calculation of the first element)

Figure 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 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.

Implemented by GEMM

As illustrated in Figure 3(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.

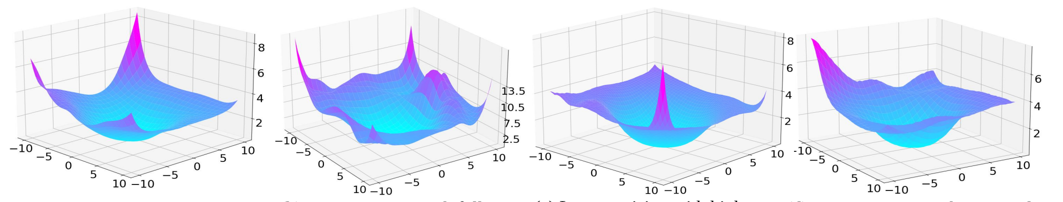
Implemented without memory rearrangement

ShiftQuant essentially assigns different shift amounts to each index of the inner dimension. As depicted in Figure 3(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.

4 Fully-quantized L1 Normalization Layers

4.1 Observation and Proposal of Normalization Layers

As shown in Figure 4, the loss landscape of low-precision networks is sharper and more challenging to optimize compared to that of full-precision networks. As existing literature Santurkar et al. (2018) 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



(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

Figure 4: Low-precision networks feature sharpen loss landscape, which disrupts convergence. We visualize the loss landscape of ResNet20 on CIFAR10 by the method in Li et al. (2018).

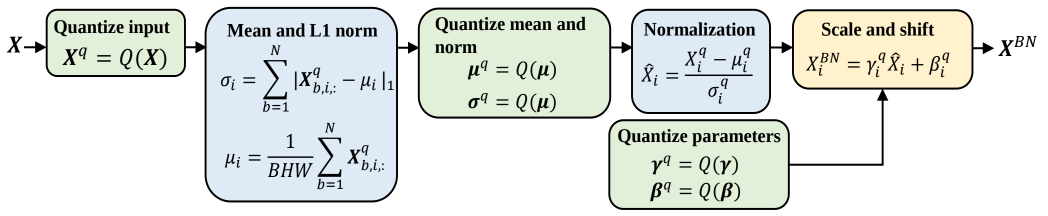


Figure 5: Procedures of the proposed fully-quantized normalization layer (take BatchNorm for example). The input $\boldsymbol{X} \in \mathcal{R}^{N \times C \times HW}$, where N, C, HW denote the batchsize, number of channels, number of spatial elements, respectively.

the low-precision networks with full-precision ones. As shown in Figure 4(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.

Two approaches can mitigate this problem, enhancing the smoothening ability of normalization layers and enhancing their tolerance to quantization. We apply L1 normalization in our fully-quantized normalization layers. Firstly, L1 normalization has stronger regularization effort than L2 normalization Huber (1996); Candès et al. (2006), leading to stronger smoothening ability (see Sec. 4.2). Secondly, the L1 norm of activation are usually larger than its L2 norm. Larger L1 norm can tolerate more quantization errors than L2 norm in the processing of normalization (see Appendix. C.2).

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 4(d), our fully-quantized L1 normalization layer achieves similar smoothening effort as the full-precision L2 normalization layer.

4.2 Theoretical Analysis

The Lipschitzness constant reflects the smoothness of loss landscape Boyd & Vandenberghe (2004); Bottou et al. (2018). The Lipschitzness of L1 normalization layers L^1 and L2 normalization layers L^2 satisfies the following relationship:

$$\frac{L^1}{L^2} \leq \frac{\|\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_1} \leq 1. \tag{5}$$

$\boldsymbol{x}$ is a vectorized input activation of normalization layers. A smaller Lipschitzness constant implies that the gradients of the loss function do not change drastically with small changes in input, which means the loss landscape is smoother Boyd & Vandenberghe (2004); Bottou et al. (2018). Detailed proof can be found in Appendix. C.1.

Table 2: Results on ResNets.

Dataset	Model	Baselines		CPT Fu et al. (2021)	StateQuant Chen et al. (2020)	Ultra-low Sun et al. (2020)	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**
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

Table 3: Results on Transformers.

Dataset	Model	Traning type	Metric	Baselines		Xi et al. (2023)	Ultra-low Sun et al. (2020)	Ghaffari et al. (2022)	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-based	Pretrain	BLEU	27.5	21.1	**27.17**	25.4	-	27.09

* Xi et al. (2023) applies 8-bit integer format on gradients and pruns half of gradients.

5 Experiments

5.1 Performance Analysis

We evaluate our integer training framework on ResNets He et al. (2016) and Transformers Vaswani et al. (2017); Dosovitskiy et al. (2020). We also report the performance of recent state-of-art methods, including new data format methods CPT Fu et al. (2021), Ultra-Low Sun et al. (2020), reflection methods StateQuant Chen et al. (2020), Xi et al. (2023), and full integer training architecture Ghaffari et al. (2022). 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.

Image classification We select various network architectures and perform experiments on CIFAR10 Krizhevsky et al. (2009) and ImageNet Deng et al. (2009) 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 He et al. (2016). As shown in Table 2, 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] Dosovitskiy et al. (2020), 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 2, 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.

Machine translation We train a transformer-based architecture[2] on the WMT14 English-German (en-de) dataset Bojar et al. (2014). 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 3. Our method achieves a negligible loss of accuracy.

Transductive and inductive prediction

[1]https://github.com/jeonsworld/ViT-pytorch
[2]https://github.com/facebookresearch/fairseq

Table 4: Results on Temporal Graph Network (TGN Kumar et al. (2019)) .

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

Table 5: Results on Recurrent Neural Network (RNN Medsker et al. (2001)) .

Dataset	Metric	Full-Precision	StateQuant	Ours
ElectricityLoadDiagrams20112014	ND	0.069	0.078	0.074
	RMSE	0.526	0.516	0.475

Table 6: 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

Table 7: Ablation on L1 normalization.

	FP	L2BN	L1BN	QL2BN	QL1BN
ResNet20	91.25	90.78	90.95	13.34	90.40
ResNet38	92.04	91.43	91.67	11.92	91.46

We train a Temporal Graph Network (TGN Kumar et al. (2019)) on the Wikipedia [3] dataset. All the linear layers in TGN utilize 8-bit weights, activations, and gradients. Our method even surpasses the performance on full-precision training.

Time series prediction

We train a Recurrent Neural Network (RNN Medsker et al. (2001)) 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 Xi et al. (2023), it demonstrates superior hardware efficiency, which we will discuss in Sec. 5.3. 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.

5.2 Parameter Analysis and Ablation study

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 conducte experiments on ResNet20 and ViT-B using different group settings on the CIFAR10 dataset. As shown in Table 6, the performance 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.

Ablation study on L1 normalization

As shown in Table 7, fully-quantized L2 normalization layers are not suitable for low-precision training. As a contrast, our fully-quantized L1 normalization layers achieve competitive performance.

5.3 Hardware Overhead

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. Meanwhile, we implement one comparative method Xi et al. (2023). We assess the throughput across various sizes of linear layers. As depicted in Figure 6, 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

[3]http://snap.stanford.edu/jodie/wikipedia.csv

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

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

comes from memory rearrangement. Moreover, 6-bit ShiftMM outperforms 4-bit Xi et al. (2023). Two ingredients lead to this, the extra computation before quantization and hardware-unfriendly pruning in Xi et al. (2023).

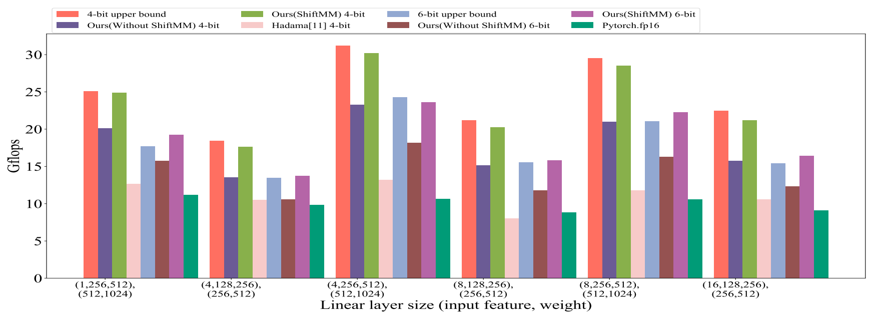


Figure 6: Comparison of simplest per-tensor quantization (the upper bound of efficiency), ShiftQuant (GEMM implementation and ShiftMM implementation), Xi et al. (2023), Pytorch.fp16 on ARMv8.

Throughput analysis on GPU

As shown in Figure 7, we analyze the performance of our 6-bit ShiftMM, 4-bit Xi et al. (2023), 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, Xi et al. (2023) does not take full advantage of the low bitwidth.

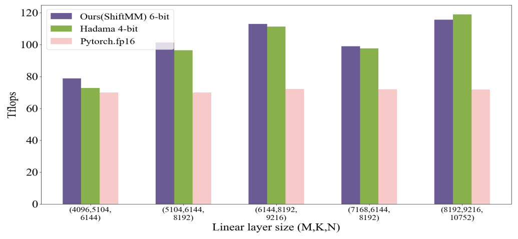


Figure 7: Throughput of ShiftMM, Hadama Xi et al. (2023), and Pytorch.fp16 on Nvidia RTX 3090.

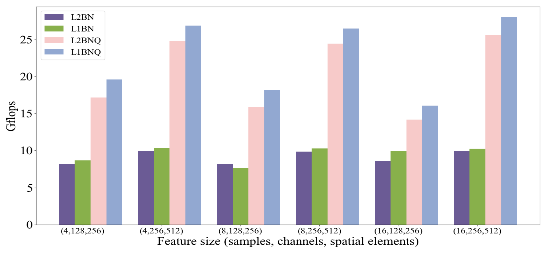


Figure 8: Comparision of different normalization layers on ARMv8.

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 8, 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.

Time proportion for each part of ShiftQuant

We evaluate the time proportion for each part of ShiftQuant on ARMV8. As depicted in Figure 9, 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.

End-to-end acceleration performance

As shown in Table 9, 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

Table 8: 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

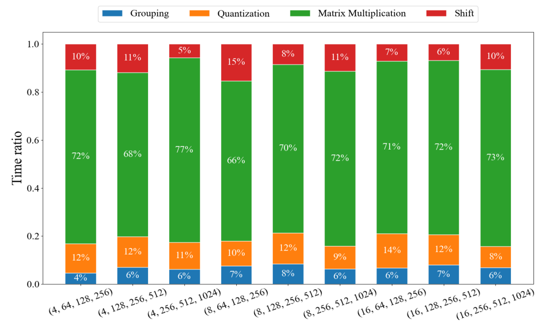


Figure 9: Time proportion for each part of ShiftQuant.

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.

Table 9: End-to-end acceleration on different platforms.

Platform	ARMV8				RTX3090							
Batchsize	1	4	8	16	32	40	48	64	80	96	128	192
ViT-B-16	2%	14%	26%	35%	9%	12%	15%	21%	18%	26%	34%	62%
ViT-L-16	3%	18%	31%	43%	12%	19%	23%	26%	28%	33%	39%	68%

6 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.

References

Barry C Arnold, Narayanaswamy Balakrishnan, and Haikady Navada Nagaraja. *A first course in order statistics*. SIAM, 2008.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth Workshop on Statistical Machine Translation*, pp. 12–58, 2014.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Matteo Cacciola, Antonio Frangioni, Masoud Asgharian, Alireza Ghaffari, and Vahid Partovi Nia. On the convergence of stochastic gradient descent in low-precision number formats. *arXiv preprint arXiv:2301.01651*, 2023.

Léopold Cambier, Anahita Bhiwandiwalla, Ting Gong, Mehran Nekuii, Oguz H Elibol, and Hanlin Tang. Shifted and squeezed 8-bit floating point format for low-precision training of deep neural networks. *arXiv preprint arXiv:2001.05674*, 2020.

Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

Jianfei Chen, Yu Gai, Zhewei Yao, Michael W Mahoney, and Joseph E Gonzalez. A statistical framework for low-bitwidth training of deep neural networks. *Advances in Neural Information Processing Systems*, 33:883–894, 2020.

Brian Chmiel, Ron Banner, Elad Hoffer, Hilla Ben Yaacov, and Daniel Soudry. Logarithmic unbiased quantization: Simple 4-bit training in deep learning. *arXiv preprint arXiv:2112.10769*, 2021.

Dipankar Das, Naveen Mellempudi, Dheevatsa Mudigere, Dhiraj Kalamkar, Sasikanth Avancha, Kunal Banerjee, Srinivas Sridharan, Karthik Vaidyanathan, Bharat Kaul, Evangelos Georganas, et al. Mixed precision training of convolutional neural networks using integer operations. *arXiv preprint arXiv:1802.00930*, 2018.

Herbert A David and Haikady N Nagaraja. *Order statistics*. John Wiley & Sons, 2004.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on Computer Vision and Pattern Recognition*, pp. 248–255. Ieee, 2009.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 293–302, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16091–16101, 2023.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

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

Alireza Ghaffari, Marzieh S Tahaei, Mohammadreza Tayaranian, Masoud Asgharian, and Vahid Partovi Nia. Is integer arithmetic enough for deep learning training? *Advances in Neural Information Processing Systems*, 35:27402–27413, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 4340–4349, 2019.

Peter J Huber. *Robust statistical procedures*. SIAM, 1996.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning*, pp. 448–456. pmlr, 2015.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf*, 2009.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1269–1278, 2019.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in Neural Information Processing Systems*, 31, 2018.

Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *arXiv preprint arXiv:1909.13144*, 2019.

Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 1529–1538, 2020.

Zechun Liu, Kwang-Ting Cheng, Dong Huang, Eric P Xing, and Zhiqiang Shen. Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 4942–4952, 2022.

Larry R Medsker, Lakhmi Jain, et al. Recurrent neural networks. *Design and Applications*, 5(64-67): 2, 2001.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in Neural Information Processing Systems*, 31, 2018.

Xiao Sun, Naigang Wang, Chia-Yu Chen, Jiamin Ni, Ankur Agrawal, Xiaodong Cui, Swagath Venkataramani, Kaoutar El Maghraoui, Vijayalakshmi Viji Srinivasan, and Kailash Gopalakrishnan. Ultra-low precision 4-bit training of deep neural networks. *Advances in Neural Information Processing Systems*, 33:1796–1807, 2020.

Stefan Uhlich, Lukas Mauch, Kazuki Yoshiyama, Fabien Cardinaux, Javier Alonso Garcia, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Differentiable quantization of deep neural networks. *arXiv preprint arXiv:1905.11452*, 2(8), 2019.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. URL `https://api.semanticscholar.org/CorpusID:13756489`.

Qiaojun Wu, Yuan Li, Song Chen, and Yi Kang. Drgs: Low-precision full quantization of deep neural network with dynamic rounding and gradient scaling for object detection. In *Proceedings of International Conference on Data Mining and Big Data*, pp. 137–151. Springer, 2022.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision*, pp. 3–19, 2018.

Haocheng Xi, Changhao Li, Jianfei Chen, and Jun Zhu. Training transformers with 4-bit integers. *Advances in Neural Information Processing Systems*, 36:49146–49168, 2023.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *Proceedings of International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023.

Kai Zhong, Xuefei Ning, Guohao Dai, Zhenhua Zhu, Tianchen Zhao, Shulin Zeng, Yu Wang, and Huazhong Yang. Exploring the potential of low-bit training of convolutional neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(12):5421–5434, 2022.

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

APPENDIX

A PROOF OF THE EQUIVALENCE BETWEEN PROBLEM (1) AND THE DIRECT OPTIMIZATION OF QUANTIZATION VARIANCE

Following Eq. 23, given a variable x, the quantization variance under stochastic rounding is:

$$\begin{aligned} Var[Q(x)] &= (l(x)-x)^2 \cdot p_Q(x) + (u(x)-x)^2 \cdot (1-p_Q(x)) \\ &= (x-l(x)) \cdot (u(x)-x), \end{aligned} \tag{6}$$

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:

$$\min\ E[Var(Q(x))]. \tag{7}$$

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

$$\begin{aligned} E[Var(Q(x))] &= \sum_{m=1}^{Q_N} \{ \int_{l_m}^{u_m} (x-l_m) \cdot (u_m - x) \} \\ &= \sum_{m=1}^{Q_N} \{ \int_{l_m}^{u_m} (l_m + u_m) \cot x \cdot p(x) dx - \int_{l_m}^{u_m} x^2 \cdot p(x) dx - \int_{l_m}^{u_m} l_m u_m \cdot p(x) dx \}. \end{aligned} \tag{8}$$

$p(x)$ 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:

$$p(x) = \sum_{k=1}^{\inf} \alpha_k \frac{1}{2\lambda_k} e^{-\frac{|x-\mu_k|}{\lambda_k}}. \tag{9}$$

Insert Eq. 9 to Eq. 8:

$$\begin{aligned} E[Var(Q(x))] = \sum_{k=1}^{\inf} \alpha_k \sum_{m=1}^{Q_N} \{ &\int_{l_m}^{u_m} (l_m + u_m) \cot x \cdot \frac{1}{2\lambda_k} e^{-\frac{|x-\mu_k|}{\lambda_k}} dx \\ &- \int_{l_m}^{u_m} x^2 \cdot \frac{1}{2\lambda_k} e^{-\frac{|x-\mu_k|}{\lambda_k}} dx \\ &- \int_{l_m}^{u_m} l_m u_m \cdot \frac{1}{2\lambda_k} e^{-\frac{|x-\mu_k|}{\lambda_k}} dx \}. \end{aligned} \tag{10}$$

We first analyze on one Laplace component:

$$\begin{aligned} t_k(x) = &\sum_{m=1}^{Q_N}\{\int_{l_m}^{u_m}(l_m+u_m)\cot x\cdot\frac{1}{2\lambda_k}e^{-\frac{|x-\mu_k|}{\lambda_k}}dx \\ &-\int_{l_m}^{u_m}x^2\cdot\frac{1}{2\lambda_k}e^{-\frac{|x-\mu_k|}{\lambda_k}}dx \\ &-\int_{l_m}^{u_m}l_m u_m\cdot\frac{1}{2\lambda_k}e^{-\frac{|x-\mu_k|}{\lambda_k}}dx\}, let z=\frac{x-\mu_k}{\lambda_k} \\ &=\sum_{m=1}^{Q_N}\{\overbrace{\frac{l_m+u_m}{2}\int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}}(\lambda_k z+\mu_k)e^{-|z|}dz}^{f^m(x)} \\ &\overbrace{-\frac{1}{2}\int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}}(\lambda_k z+\mu_k)^2e^{-|z|}dz}^{g^m(x)} \\ &\overbrace{-\frac{l_m u_m}{2}\int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}}e^{-|z|}dz}^{q^m(x)}\}. \end{aligned} \tag{11}$$

Analysis on $f^m(x)$

$$\int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}}(\lambda_k z+\mu_k)e^{-|z|}dz=\begin{cases}\lambda_k[(-\frac{u_m-\mu_k}{\lambda_k}+1)e^{-\frac{u_m-\mu_k}{\lambda_k}}-(-l+1)e^{-\frac{l_m-\mu_k}{\lambda_k}}] \\ +\mu_k[e^{-\frac{l_m-\mu_k}{\lambda_k}}-e^{-\frac{u_m-\mu_k}{\lambda_k}}] & ,\frac{u_m-\mu_k}{\lambda_k}>\frac{l_m-\mu_k}{\lambda_k}>0 \\ 2(\mu_k-\lambda_k)-(\lambda_k\frac{l_m-\mu_k}{\lambda_k}+\mu_k-\lambda_k)e^{\frac{l_m-\mu_k}{\lambda_k}} \\ -(\lambda_k\frac{u_m-\mu_k}{\lambda_k}+\mu_k-\lambda_k)e^{-\frac{u_m-\mu_k}{\lambda_k}} & ,\frac{u_m-\mu_k}{\lambda_k}>0>\frac{l_m-\mu_k}{\lambda_k} \\ \lambda_k[(\frac{u_m-\mu_k}{\lambda_k}-1)e^{\frac{u_m-\mu_k}{\lambda_k}}-(l-1)e^{\frac{l_m-\mu_k}{\lambda_k}}] \\ +\mu_k[e^{\frac{u_m-\mu_k}{\lambda_k}}-e^{\frac{l_m-\mu_k}{\lambda_k}}] & ,\frac{l_m-\mu_k}{\lambda_k}<\frac{u_m-\mu_k}{\lambda_k}<0.\end{cases} \tag{12}$$

With assumption of symmetry on quantization range, we have:

$$\begin{aligned}\sum_{m=1}^{Q_N}\int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}}(\lambda_k z+\mu_k)e^{-|z|}dz =&\lambda_k[(1-\frac{u_{Q_N}-\mu_k}{\lambda_k})e^{\frac{u_{Q_N}-\mu_k}{\lambda_k}}-(l_1-1)e^{\frac{l_1-\mu_k}{\lambda_k}}] \\ &+\mu_k[e^{\frac{l_1-\mu_k}{\lambda_k}}-e^{-\frac{u_{Q_N}-\mu_k}{\lambda_k}}]+2(\mu_k-\lambda_k).\end{aligned} \tag{13}$$

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:

$$\begin{aligned}
\sum_{m=1}^{Q_N} f^m(x) &= (\tau - \frac{\tau}{Q_N})\beta(u_{Q_N}, l_0) - \frac{2\tau}{Q_N}\sum_{m=1}^{Q_N} f(u_m, l_1) \\
&= \frac{Q_N\tau - \tau}{Q_N} \cdot 2\lambda \cdot e^{-\frac{u_{Q_N}-\mu_k}{\lambda_k}} - \frac{2\tau}{Q_N}\sum_{m=1}^{Q_N} \lambda[e^{-|u_m|} \\
&+ e^{-\frac{l_1-\mu_k}{\lambda_k}} - |u_m|e^{-|u_m|} - \frac{l_1-\mu_k}{\lambda_k} e^{-\frac{l_1-\mu_k}{\lambda_k}} - 2] \\
&= 2\lambda\tau\frac{Q_N-1}{Q_N} e^{-\frac{u_{Q_N}-\mu_k}{\lambda_k}} - 2\lambda\tau\frac{1}{Q_N}\cdot(Q_N-1)\cdot[e^{\frac{l_1-\mu_k}{\lambda_k}} - \\
&\frac{l_1-\mu_k}{\lambda_k} e^{-\frac{l_1-\mu_k}{\lambda_k}} - 2] - \frac{2\lambda\tau}{Q_N}\sum_{m=1}^{Q_N}[e^{-|u_m|} - |u_m|e^{-|u_m|}] \\
&= 2\lambda\tau\frac{Q_N-1}{Q_N}(\frac{u_{Q_N}-\mu_k}{\lambda_k} e^{-\frac{u_{Q_N}-\mu_k}{\lambda_k}}) - \frac{2\lambda\tau}{Q_N}\sum_{m=1}^{Q_N}[e^{-|u_m|} - |u_m|e^{-|u_m|}], \\
&\approx 2\lambda\tau\frac{Q_N-1}{Q_N}(\frac{u_{Q_N}-\mu_k}{\lambda_k} e^{-\frac{u_{Q_N}-\mu_k}{\lambda_k}})
\end{aligned} \tag{14}$$

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:

$$\int_l^u (\lambda_k z + \mu_k)^2 e^{-|z|} dz = \begin{cases}
\lambda^2[-z^2e^{-z} - 2ze^{-z} - 2e^{-z}]|_l^u + \\
2\lambda\mu[-ze^{-z} - e^{-z}]|_l^u + \mu^2[-e^{-z}]|_l^u, & u > l > 0 \\
\lambda^2[(z^2 - 2z + 2)e^z|_l^0 + (-z^2 + 2z - 2)e^{-z}|_0^u] + \\
2\lambda\mu[(z-1)e^z|_l^0 + (-z+1)e^{-z}|_0^u] + \mu^2[e^z|_l^0 + (-e^{-z})|_0^u], & u > 0 > l \\
\lambda^2[z^2e^z - 2ze^z + 2e^z]|_l^u + \\
2\lambda\mu[ze^z - e^z]|_l^u + \mu^2[e^z]|_l^u, & l < u < 0.
\end{cases} \tag{15}$$

Accumulating the integers around different quantization levels:

$$\begin{aligned}
\sum_{m=1}^{Q_N} \int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}} (\lambda_k z + \mu_k)^2 e^{-|z|} dz &= \lambda^2[-e^{-z}(z^2 + 2z + 2)]|_0^u + \mu^2[-e^{-z}]|_0^u + \lambda^2[e^z(z^2 - 2z + 2)]|_l^0 + \mu^2[e^z]|_l^0 \\
&= \lambda^2[-e^{-\frac{u_m-\mu_k}{\lambda_k}}(\frac{u_m-\mu_k}{\lambda_k}^2 + 2\frac{u_m-\mu_k}{\lambda_k} + 2)] \\
&+ \mu^2[-e^{-\frac{u_m-\mu_k}{\lambda_k}} + 1] + \lambda^2[2 - e^{\frac{l_m-\mu_k}{\lambda_k}}(\frac{l_m-\mu_k}{\lambda_k}^2 - 2\frac{l_m-\mu_k}{\lambda_k} + 2)] \\
&+ \mu^2[1 - e^{\frac{l_m-\mu_k}{\lambda_k}}] \\
&= 2\mu^2(-e^{-\frac{u_m-\mu_k}{\lambda_k}} + 1) \\
&+ 2\lambda^2[-e^{-\frac{u_m-\mu_k}{\lambda_k}}(\frac{u_m-\mu_k}{\lambda_k} + 1)^2 - e^{-\frac{u_m-\mu_k}{\lambda_k}} + 2].
\end{aligned} \tag{16}$$

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:

$$\int_l^u e^{-|z|}dz = \begin{cases} e^{-l} - e^{-u}, & u > l > 0 \\ 2 - e^{l} - e^{-u}, & u > 0 > l \\ e^{u} - e^{l}, & l < u < 0. \end{cases} \tag{17}$$

Thus we have:

$$\sum_{m=1}^{Q_N} \int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}} e^{-|z|} = 2 - 2e^{-u} \tag{18}$$

Following the same derivation as Eq. (14), we can find:

$$\sum_{m=1}^{Q_N} \frac{l_m + u_m}{2} \int_{\frac{l_m-\mu_k}{\lambda_k}}^{\frac{u_m-\mu_k}{\lambda_k}} e^{-|z|} = \frac{\tau \cdot (\tau - \frac{2\tau}{Q_N})}{2}(2 - 2e^{-u}). \tag{19}$$

Combine the result of Eq. (14), Eq. (16) and Eq. (19), we can find:

$$\begin{aligned} t_k(x) = {} & 2\lambda\tau\frac{Q_N - 1}{Q_N}(ue_{-u} - 2) - \{\mu^2(-e^{-u} + 1) + \\ & \lambda^2[-e^{-u}(u+1)^2 - e^{-u} + 2]\} - \frac{\tau \cdot (\tau - \frac{2\tau}{Q_N})}{2}(2 - 2e^{-u}) \\ = {} & 2\lambda\tau\frac{Q_N - 1}{Q_N}(ue_{-u} - 2) + \mu^2(e^{-u} - 1) + \\ & \lambda^2[e^{-u}(u+1)^2 + e^{-u} - 2] + \frac{\tau \cdot (\tau - \frac{2\tau}{Q_N})}{2}(2e^{-u} - 2), \end{aligned} \tag{20}$$

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 Arnold et al. (2008); David & Nagaraja (2004):

$$E[max\, \boldsymbol{x}] \approx \sum_{k=1}^{\inf}(\mu_k + \lambda_k \ln(2NHW)) \tag{21}$$

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:

$$\begin{gathered} \min_{\tau_0,\tau_1,\cdots,\tau_{N_G-1},\tau_{N_G}} \sum_{g=1}^{N_G} \sum_{i \in P^g} E[Var(\boldsymbol{x}_i)], \\ s.t.\, r_{max} = \tau_0 \geq \tau_1 \geq \cdots \geq \tau_{N_G-1} \geq \tau_{N_G} = 0, \end{gathered} \tag{22}$$

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 $\frac{u_{Q_N}}{\tau_{\boldsymbol{x}_i}}$, 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 (22) 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

Unbiased quantizer

ShiftQuant utilizes stochastic rounding:

$$SR(x) = \begin{cases} u(x) & w.p. \quad \frac{x-l(x)}{u(x)-l(x)} \\ l(x) & w.p. \quad 1 - \frac{x-l(x)}{u(l)-l(x)} \end{cases}, \tag{23}$$

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:

$$E[D_q(X)] = E[SR((X - ZI) \cdot S) \cdot S^{-1} + ZI] = X. \tag{24}$$

$X \in \mathbb{R}^{N \times D}$ 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.

Low variance

The up bound of ShiftQuant's variance U_{dq} satisfies:

$$U_{dq} \leq \alpha \cdot \frac{N}{4} \sum_{j=1}^{D} s_j^{-2} + 2^{-N_G} \frac{ND}{4} s^{-2}, \tag{25}$$

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. (25) demonstrates that ShiftQuant achieves a closed performance to fine-grained quantizers.

Proof of Unbiased Quantization

$$\begin{aligned} E[D_q(X)] &= E[SR((X - ZI) \cdot S) \cdot S^{-1} + ZI] \\ &= E[SR(X - ZI)] \cdot S \cdot S^{-1} + ZI \\ &= (X - ZI) \cdot S \cdot S^{-1} + ZI \\ &= X. \end{aligned} \tag{26}$$

Clearly, ShiftQuant is a unbiased quantizer.

Proof of Low Variance

From Eq. (6), we can find the up bound of quantization variance:

$$Var[SR(x)] = (x - l(x))(u(x) - x) \leq \frac{[u(x) - l(x)]^2}{4}, \tag{27}$$

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

$$Var[Q_{pt}(X)] = \sum_{i=1}^{N} \sum_{j=1}^{D} \frac{r^2}{4B^2} = \frac{ND}{4} s^{-2}, \tag{28}$$

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:

$$Var[Q_{pt}(X)] = \sum_{i=1}^{N} \sum_{j=1}^{D} \frac{r_j^2}{4B^2} = \frac{N}{4} \sum_{j=1}^{D} s_j^{-2}, \tag{29}$$

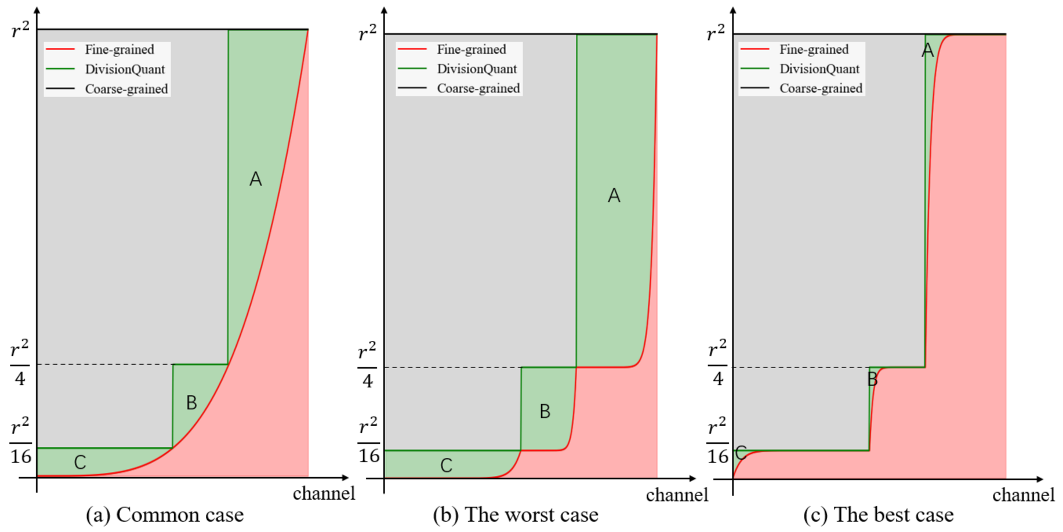


Figure 10: Quantization variance of coarse-grained quantization, fine-grained quantization, and ShiftQuant. We set number of groups as 3 in visualization. We sort the channels in ascending order by the range of channels.

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 10.

As shown in Figure 10(b), in the worst case, the gap of channels' ranges in a group is maximum. In this situation, we can find that:

$$\begin{aligned} U_{dq} &= U_{fq} + U_A + U_B + U_C \\ &\leq U_{fq} + U_{fq} * 3 + U_C \\ &= U_{fq} * 4 + U_C, \end{aligned} \tag{30}$$

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 10. For U_C, we have:

$$U_C < 2^{-2*N_G+2} \cdot U_{cq}, \tag{31}$$

where U_{cq} is the variance of coarse-grained quantization. With combining Eq. (30) and Eq. (31), we can find:

$$U_{dq} \leq 4U_{fq} + 2^{-2*N_G+2} \cdot U_{cq}. \tag{32}$$

As shown in Figure 10(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:

$$\begin{aligned} U_{dq} &= U_{fq} + U_A + U_B + U_C \\ &\approx U_{fq} \\ &\leq U_{fq} + 2^{-2*N_G+2} \cdot U_{cq}. \end{aligned} \tag{33}$$

With combining Eq. (32) and Eq. (33), we can find:

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

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

C Proof of L1 Normalization

C.1 Analysis on Lipschitzness Constant

A normalization layer contains two step: normalization, scale and shift:

$$\hat{\boldsymbol{x}} = \frac{\boldsymbol{x} - \mu}{\sigma}, \quad Normalization, \tag{35}$$

$$\boldsymbol{y} = \gamma \cdot \hat{\boldsymbol{x}} + \beta, \quad Scale\, and\, shift, \tag{36}$$

where $\boldsymbol{x}$ is the input vector. For example, $\boldsymbol{x} \in \mathcal{R}^{NHW}$ in batchnorm, N, H, W are batchsize, height, and width of the input activation respectively. When we get $\boldsymbol{y}$, it is feed to the next layer $f(\cdot)$. Let $\boldsymbol{z}$ denote the output of the following layer ($\boldsymbol{z} = f(\boldsymbol{y})$). The gradient of $\boldsymbol{y}$ is:

$$\begin{aligned} \frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{x}} &= \left(\frac{\gamma}{N\sigma}\right) \left(N \frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{z}} - \mathbf{1} \left\langle \mathbf{1}, \frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{z}} \right\rangle - \hat{\boldsymbol{x}} \left\langle \frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{z}}, \hat{\boldsymbol{x}} \right\rangle \right) \\ &= \frac{\gamma}{\sigma} \left(\left(\frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{z}} - \mathbf{1}\mu_g \right) - \frac{\hat{\boldsymbol{x}}}{\|\hat{\boldsymbol{x}}\|} \left\langle \left(\frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{z}} - \mathbf{1}\mu_g \right), \frac{\hat{\boldsymbol{x}}}{\|\hat{\boldsymbol{x}}\|} \right\rangle \right), \end{aligned} \tag{37}$$

where $\widehat{\mathcal{L}}$ is the loss function. $\mu_g = \left\langle \mathbf{1}, \frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{z}} \right\rangle$. The derivation of Eq. (37) can be found in Santurkar et al. (2018). Then we have:

$$\begin{aligned} \left\| \frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{x}} \right\|^2 &= \frac{\gamma^2}{\sigma^2} \left\| \left(\frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{z}} - \mathbf{1}\mu_g \right) - \frac{\hat{\boldsymbol{x}}}{\|\hat{\boldsymbol{x}}\|} \left\langle \left(\frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{z}} - \mathbf{1}\mu_g \right), \frac{\hat{\boldsymbol{x}}}{\|\hat{\boldsymbol{x}}\|} \right\rangle \right\|^2 \\ &\leq \frac{\gamma^2}{\sigma^2} \left(\left\| \left(\frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{z}} - \mathbf{1}\mu_g \right) \right\|^2 - \left\langle \left(\frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{z}} - \mathbf{1}\mu_g \right), \frac{\hat{\boldsymbol{x}}}{\|\hat{\boldsymbol{x}}\|} \right\rangle^2 \right), \\ &\leq \frac{\gamma^2}{\sigma^2} \left(\left\| \left(\frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{z}} - \mathbf{1}\mu_g \right) \right\|^2 - \left\| \frac{\partial \widehat{\mathcal{L}}}{\partial \boldsymbol{z}} - \mathbf{1}\mu_g \right\|^2 \cdot \left\| \frac{\hat{\boldsymbol{x}}}{\|\hat{\boldsymbol{x}}\|} \right\|^2 \right). \end{aligned} \tag{38}$$

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

$$\begin{aligned} &\sigma_1 = |\boldsymbol{x} - \mu|_1,\, \hat{\boldsymbol{x}}_1 = \frac{\boldsymbol{x} - \mu}{\sigma_1}, L1\, normalization \\ &\sigma_2 = |\boldsymbol{x} - \mu|_2,\, \hat{\boldsymbol{x}}_2 = \frac{\boldsymbol{x} - \mu}{\sigma_2}, L2\, normalization. \end{aligned} \tag{39}$$

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

$$\frac{L^1}{L^2} \leq \frac{\sigma_2^2}{\sigma_1^2} = \frac{|\boldsymbol{x} - \mu|_2^2}{|\boldsymbol{x} - \mu|_1^2}. \tag{40}$$

Moreover, as shown in Figure 11, 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.

C.2 Quantization-Toleration of L1 Normalization

In fully-quantized normalization layers, before normalization (Eq. (35)), we first quantize the statistics μ and σ:

$$\mu^q = Q(\mu), \sigma^q = Q(\sigma),\, Quantization. \tag{41}$$

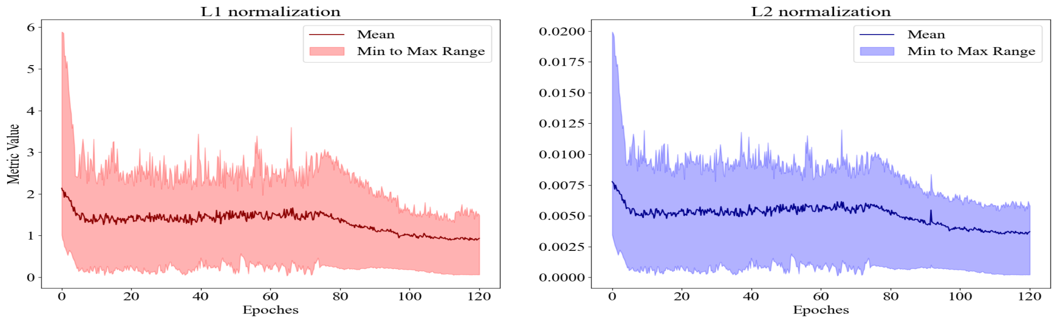


Figure 11: The L1 norm and L2 norm of the activations during all training stage. We measure the activation before the first BN layer in ResNet20.

$$\hat{\boldsymbol{x}} = \frac{\boldsymbol{x} - \mu^q}{\sigma^q}, \ Normalization. \tag{42}$$

In Eq. (42), σ^q is the denominator. Same noise imposed on smaller denominator will lead to larger fluctuation in Eq. (42). As shown in Figure 11, σ in L2 normalization is significantly smaller than in L2 normalization, which validates the weak quantization-tolerance of L2 normalization layers. Moreover, we visualize the quantization error of $\frac{1}{\sigma}$ under different normalization layers. As shown in Figure 12, L1 normalization achieves significantly smaller quantization gap than L2 normalization.

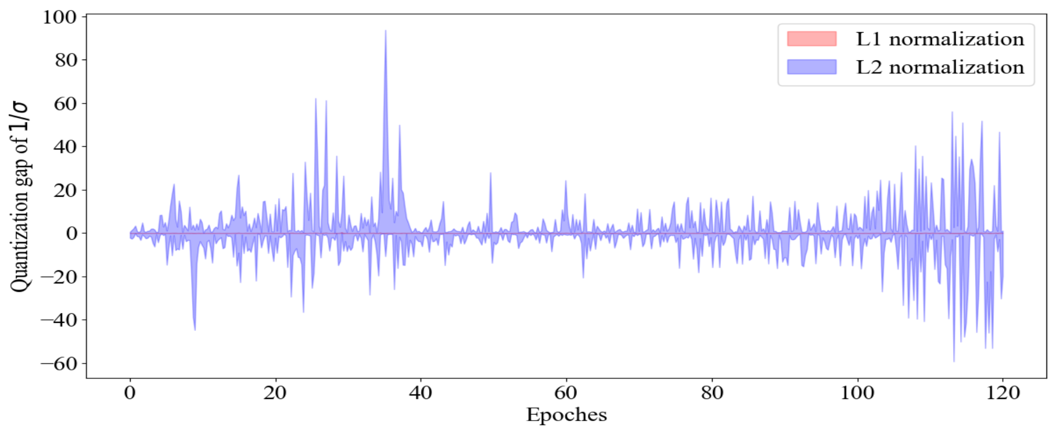


Figure 12: The quantization gap of $\frac{1}{\sigma}$ under L1 normalization and L2 normalization. During the whole training stage, the quantization error of L1 normalization is significantly smaller than L2 normalization.

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:

$$(\boldsymbol{G_B})^g_q = round(\frac{(\boldsymbol{G_B})^g}{\frac{\tau_0}{B} \cdot 2^{-g}}) = round(\frac{(\boldsymbol{G_B})^g}{s^{-1}_{\boldsymbol{G_B}} \cdot 2^{-g}}), \tag{43}$$

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

$$\begin{aligned}(\boldsymbol{G_B})^g \cdot (\boldsymbol{W}^T)^g &= [s^{-1}_{\boldsymbol{G_B}} \cdot 2^{-g} \cdot (\boldsymbol{G_B})^g_q] \cdot [s^{-1}_{\boldsymbol{W}} \cdot (\boldsymbol{W}^T)^g_q] \\ &= s^{-1}_{\boldsymbol{G_B}} \cdot s^{-1}_{\boldsymbol{W}} \cdot [(\boldsymbol{G_B})^g_q \cdot (\boldsymbol{W}^T)^g_q >> g],\end{aligned} \tag{44}$$

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:

$$\begin{aligned} \boldsymbol{G_A} &= \sum_{g=1}^{N_G} (\boldsymbol{G}_B)^g \cdot (\boldsymbol{W}^T)^g \\ &= s_{\boldsymbol{G}_B}^{-1} \cdot s_{\boldsymbol{W}}^{-1} \sum_{g=1}^{N_G} [(\boldsymbol{G}_B)^g{}_q \cdot (\boldsymbol{W}^T)^g{}_q >> g], \end{aligned} \tag{45}$$

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:

$$\begin{aligned} & s_{\boldsymbol{G}_B}^{-1} \cdot s_{\boldsymbol{W}}^{-1} \cdot 2^{-N_G} \sum_{g=1}^{N_G} [(\boldsymbol{G}_B)^g{}_q \cdot (\boldsymbol{W}^T)^g{}_q << (N_g - g)] \\ =& \sum_{g=1}^{N_G} s_{\boldsymbol{G}_B}^{-1} \cdot s_{\boldsymbol{W}}^{-1} \cdot 2^{-N_G} [(\boldsymbol{G}_B)^g{}_q \cdot (\boldsymbol{W}^T)^g{}_q << (N_g - g)] \\ =& \sum_{g=1}^{N_G} s_{\boldsymbol{G}_B}^{-1} \cdot (\boldsymbol{G}_B)^g{}_q \cdot (2^{-N_G} << (N_g - g)) \cdot s_{\boldsymbol{W}}^{-1} \cdot (\boldsymbol{W}^T)^g{}_q \\ =& \sum_{g=1}^{N_G} s_{\boldsymbol{G}_B}^{-1} \cdot (\boldsymbol{G}_B)^g{}_q \cdot 2^{-g} \cdot s_{\boldsymbol{W}}^{-1} \cdot (\boldsymbol{W}^T)^g{}_q \\ =& \sum_{g=1}^{N_G} (\boldsymbol{G}_B)^g \cdot (\boldsymbol{W}^T)^g \\ =& \boldsymbol{G_A}. \end{aligned} \tag{46}$$

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.

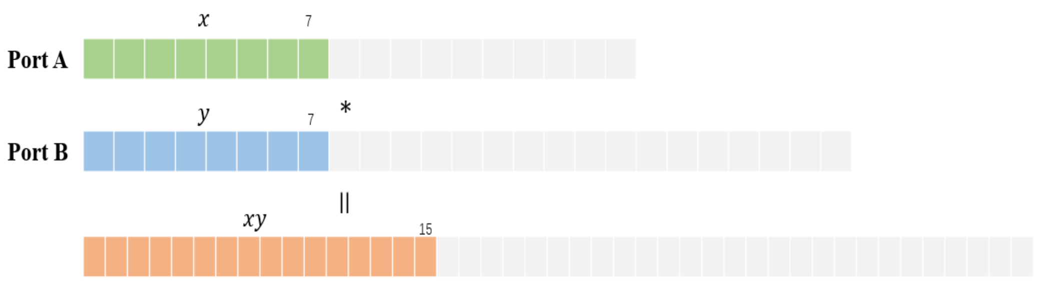


Figure 13: Direct utilization of DSP on 8-bit multiplication. Unused bits is colored in gray. A lot of bitwidth is wasted.

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 8.

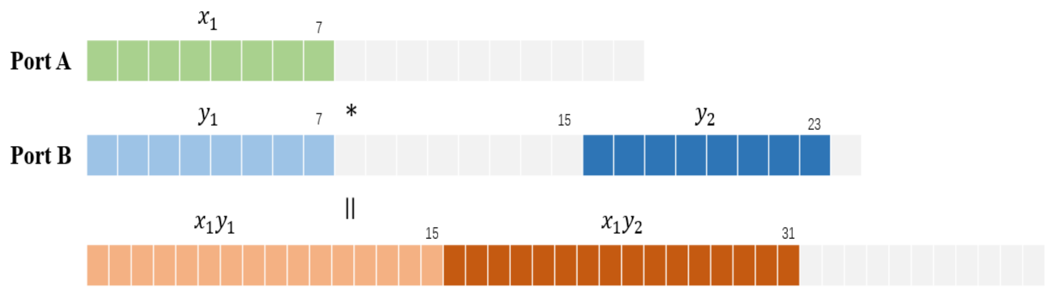


Figure 14: DSP reusing on 8-bit multiplication. One calculation can realize two multiplications.

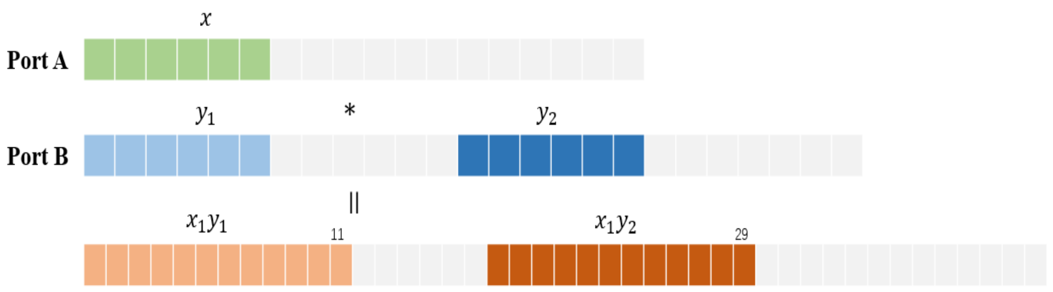


Figure 15: DSP reusing on 6-bit multiplication. One calculation can realize two multiplications.

As shown in Figure 16, the DSP48E1 block is capable of simultaneously executing four 4-bit multiplications. This aligns with the data presented in Table 8, where the DSP resource consumption for the 4-bit implementation is half that of the 6-bit and 8-bit implementations.

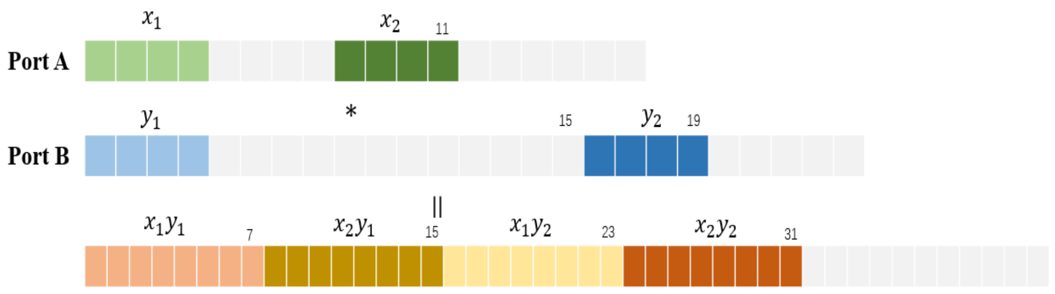


Figure 16: DSP reusing on 4-bit multiplication. One calculation can realize four multiplications.