PalQuant: Accelerating High-precision Networks on Low-precision Accelerators

Qinghao Hu$^{\star 1}$, Gang Li$^{\star 2}$, Qiman Wu$^{\star 3}$, and Jian Cheng^{1}

^{1} Institute of Automation, Chinese Academy of Sciences
^{2} Shanghai Jiao Tong University
^{3} Baidu Inc.
`huqinghao2014@ia.ac.cn, gliaca@sjtu.edu.cn, wuqiman@baidu.com, jcheng@nlpr.ia.ac.cn`

Abstract. Recently low-precision deep learning accelerators (DLAs) have become popular due to their advantages in chip area and energy consumption, yet the low-precision quantized models on these DLAs bring in severe accuracy degradation. One way to achieve both high accuracy and efficient inference is to deploy high-precision neural networks on low-precision DLAs, which is rarely studied. In this paper, we propose the PArallel Low-precision Quantization (PalQuant) method that approximates high-precision computations via learning parallel low-precision representations from scratch. In addition, we present a novel cyclic shuffle module to boost the cross-group information communication between parallel low-precision groups. Extensive experiments demonstrate that PalQuant has superior performance to state-of-the-art quantization methods in both accuracy and inference speed, e.g., for ResNet-18 network quantization, PalQuant can obtain 0.52% higher accuracy and 1.78× speedup simultaneously over their 4-bit counter-part on a state-of-the-art 2-bit accelerator. Code is available at `https://github.com/huqinghao/PalQuant`

Keywords: Quantization, Network Acceleration, CNNs

1 Introduction

Recently various model compression techniques have been proposed to deploy deep neural networks on resource-constrained edge devices. Among these, fixed-point quantization [16,30,31] that converts full-precision floating-point operation to low-bit integer counterpart has become the *de facto* method due to its hardware efficiency.

At present, most of the commercial CNN accelerators are designed for high-precision (such as INT16/INT8) arithmetic. One important reason for this is because the accuracy of a quantized network is hard to retain as the quantization bit-width narrows. Yet low-precision accelerators lead to orders of magnitude decrease in chip-area and energy consumption compared to the high-precision

* Equal Contribution.

hardware [10]. This motivates plenty of researchers to study how to improve the accuracy of quantized networks on low-precision accelerators [31,15,28,22,13]. While these methods focus on designing low-precision quantization algorithms, the accuracy of quantized networks may be limited by the low computation precision of accelerators.

Different from the above methods, we try to answer the question: *Is it possible to deploy high-precision networks on the existing well-designed low-precision accelerator to capture both model accuracy and inference efficiency?* To achieve this, a naive solution is decomposing a high-precision network at the bit level and conducting inference on low-precision hardware in a nibble iteration manner [1]. For example, to run an 8-bit network on a 2-bit accelerator, we can split each 8-bit operand into four 2-bit operands so that the original 8-bit multiplication can be carried out in $4 \times 4 = 16$ steps using 2-bit multipliers with proper shifting. Although the 2-bit operation is much cheaper than the 8-bit counterpart in terms of chip area and power consumption, there is no gain in inference latency since the total amount of bit operations is unchanged.

In this paper, we investigate the opportunity of fast and accurate inference of high-precision CNN models on low-precision hardware from the algorithm side. We propose PArallel Low-precision Quantization (PalQuant), a hardware-friendly, simple yet effective quantization method for efficient CNN acceleration. Different from the naive bit-level decomposition solution, PalQuant reduces the computational complexity by dividing the expanded low-precision channels into parallel groups.

To encourage information flowing across different groups, we propose a novel cyclic shuffle module that fuses features from two consecutive groups cyclically in a hardware-friendly way. One important property of cyclic shuffle is that it may serve as a complement to channel shuffle. This is mainly due to that cyclic shuffle fuses channel features at group level while the channel shuffle fuses a fraction of channel features from each group. Extensive experiments from both algorithm and hardware sides demonstrate that PalQuant can consistently achieve the highest accuracy and inference speedup than state-of-the-art methods. The contribution of this paper can be summarized as follows:

- We propose the PalQuant algorithm that enables efficient high-precision computation in low-precision accelerators via learning parallel low-precision representations from scratch.
- We propose a novel cyclic shuffle module to help the information flow across parallel low-precision convolution groups.
- Extensive experiments on ImageNet benchmark demonstrate that PalQuant outperforms state-of-the-art quantization methods in terms of both accuracy and computational cost. We also examine the speed-up of PalQuant on two CNN accelerators [33,14], which shows that PalQuant achieves 1.7× speedups than state-of-the-arts on ResNet-18.

2 Related Work

2.1 Quantization Methods

Deep Neural Network Quantization methods have become popular in recent years as they can reduce energy consumption and inference latency of deep neural networks. One line of quantization methods aims to reduce the quantization bit-width, while maintaining the network accuracy. Early quantization methods mainly focus on high bit-width fixed-point quantization [16,30], e.g. 8bit or 16bit quantization scheme, which brings in little accuracy degradation. Later, various binary [11,31,18] and ternary quantization methods [26,25,41] are proposed to reduce the multiplication operations in the network inference. Another line of research on quantization methods is to learn good quantization parameters such as quantization step-values, clipping values, and so on. Early quantization methods use fixed quantization parameters [19,16], dynamic parameters based on statistics of the data distribution [5,27], or seek the parameters that minimize the quantization error [31,38]. Recently, researchers propose to use trainable quantization parameters, e.g. clipping values [23,9] and step-size [13,22]. These parameters can be learned by gradient back-propagation which minimizes the task loss. Another related work is WRPN [29] which increases the number of filter maps and reduces the quantization bit-width of feature maps. Our proposed method differs with WRPN [29] in three aspects. First, Our PalQuant and WRPN [29] target at different problems. While WRPN [29] mainly tries to reduce the large memory footprint of high-precision feature maps via wide reduced-precision representations, our PalQuant enables efficient high-precision computations on low-precision accelerators. Second, PalQuant reduces computational complexity and memory access via parallel low-precision computation scheme, and it achieves higher or comparable network accuracy with less computational cost than WRPN [29]. Third, we propose a novel cyclic shuffle module to fuse features across different groups, that further strengthens the model representation.

2.2 Hardware Accelerators for CNN

The extremely high computational complexity of CNN poses a significant challenge to real-world deployment, especially for resource-constraint embedded devices. As a result, energy-efficient FPGA/ASIC-based CNN accelerators have gained increasing popularity recently in both academia and industry. To maintain network accuracy, the computing architecture of early CNN accelerators mainly exploits floating-point and high-precision fixed-point data types (such as 16-bit/8-bit). For example, Zhang *et al.* [37] designs the first FPGA-based accelerator for floating-point CNN inference. DianNao [7], Eyeriss [8], and TPU [21] are ASIC-based accelerators designed for 16-bit quantized CNN.

With the rapid development of quantization methods in the deep learning community, many low-precision CNN accelerators have been proposed to further improve hardware efficiency. With the help of power-of-two quantization [40], Li *et al.* [24] and Tann *et al.* [34] propose multiplier-free architectures for area and power-efficient inference by replacing conventional integer multiplications with shift-based operations. YodaNN [2] is an ASIC-based accelerator tailored for Binary Weight Networks [11]. Bit Fusion [33], Bitblade [32], and BPVeC [14] are precision-scalable CNN accelerators exploiting 2-bit multipliers for arbitrary-precision computation. Umuroglu *et al.* [35] and Zhao *et al.* [39] propose dedicated accelerators for Binarized Neural Networks [20], which can achieve the highest frame-per-second under extremely low area and energy consumption. Lascorz *et al.* [12] accelerates CNN inference through hardware/software co-design, while PalQuant is more general and can be deployed on a variety of accelerators, including bit-parallel and bit-serial accelerators.

3 Preliminaries

3.1 Notation

The input activation and weight of a fully-connected layer in a deep neural network are denoted by $X \in \mathcal{R}^{N\times S}$ and $W \in \mathcal{R}^{T\times S}$, respectively. The layer's output Y can be obtained by:

$$Y = XW^T \tag{1}$$

The quantized input activation and weight are denoted by $\hat{X} \in \mathcal{R}^{N\times S}$ and $\hat{W} \in \mathcal{R}^{T\times S}$, respectively. For a convolutional layer, the output Y can also be obtained by Eq. 1, since the weights and activations can be transformed into matrices.

3.2 Uniform Quantization

Recently uniform quantizer with trainable quantization parameters [15,23] becomes popular. Given a floating-point number x, it first maps x to a range $[0, 1]$ by the following equation:

$$x_n = clip\left(\frac{x-l}{u-l}, 0, 1\right) \tag{2}$$

where $clip(\cdot,\cdot,\cdot)$ denotes the clip function, l and u denotes lower bound and up bound, respectively. Then the integer value x_q can be obtained by the quantization function:$x_q = \lfloor (2^b-1)x_n \rceil$, where $\lfloor\cdot\rceil$ is rounding-to-nearest operation, and b denotes the quantization bit-width. And the de-quantization function is given by the following formula:

$$\hat{x} = \begin{cases} \frac{x_q}{2^b-1} & x \in weight \\ 2(\frac{x_q}{2^b-1} - 0.5) & x \in activation \end{cases} \tag{3}$$

where the formula maps the de-quantized weights to a range $[-1, 1]$, and restricts the de-quantized activations to be non-negative. An additional parameter α [23] is required to multiply the output activations of each layer, which adjusts the output scale of the whole feature map.

3.3 Bit-level Decomposing

Consider that there is an accelerator that supports only B-bit computation, a naive solution for running a high-precision (M-bit) model on this accelerator is to decompose M-bit representations to multiple B-bit at bit-level, then shift and sum up those B-bit operations results:

$$Y = \hat{X}\hat{W}^T = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1} \hat{X}_i \hat{W}_j^T * 2^{(i+j)*B} \tag{4}$$

where $\hat{X}_i$ and $\hat{W}_j$ are B-bit input and weights that are generated by splitting M-bit input and weights in bit-level, and $G = ceil(M/B)$. Note that the computational cost is unchanged comparing to the original M-bit computation.

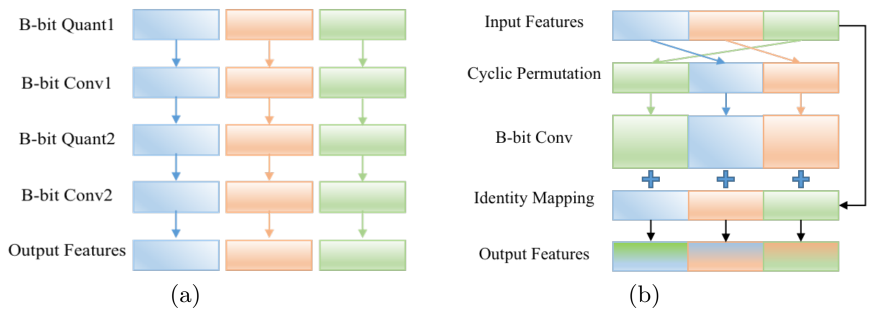

Fig. 1. (a) The proposed parallel low-precision computation scheme. Here the hyper-parameter G is fixed to 3. B-bit Quant and B-bit Conv denote a B-bit quantization layer and B-bit convolutional layer, respectively. (b) The proposed cyclic shuffle module. Here B-bit Conv denotes a B-bit quantized 1×1 group convolutional layer with G=3

4 Proposed Method

4.1 Parallel Low-Precision Quantization

Since the bit-level decomposing method (Eq.4) has the same amount of bit operations as the original high-precision computation, there is no gain in inference latency. At first, we try to reduce the inference latency by an approximation to Eq.4:

$$min_{\bar{W}_i} = \| \sum_{i=0}^{G-1}\sum_{j=0}^{G-1} \hat{X}_i \hat{W}_j^T * 2^{(i+j)*B} - \sum_{i=0}^{G-1} (\hat{X}_i {\bar{W}_i}^T) * 2^{i*B} \|_F^2 \tag{5}$$

where $\hat{W}_j$ and $\hat{X}_i$ are B-bit representations chunked from M-bit $\hat{W}$ and $\hat{X}$, respectively. We found that the B-bit computations in Eq.5 are naturally in parallel G groups, which provides the potential to be hardware-friendly. In addition, Eq.5 approximates the standard results with only G parallel B-bit computations while Eq.4 requires G^2 B-bit computations. This indicates that the proposed approximation only costs $\frac{1}{G}$ computation complexity of the bit-level decomposition method.

Yet there are two problems in Eq.5. One is that we empirically found that all G solutions of $\bar{W}_i$ degenerate to one similar solution, which results in an inferior network accuracy. We assume that this phenomenon has relations with two aspects: 1. B-bit representations are chunked from M-bit ones, thus they are coupling together with a strong connection[4]; 2. minimizing the feature map reconstruction loss easily causes the over-fitting problem. The other problem in Eq.5 is that chunking M-bit $\hat{X}$ to multiple B-bit representations tend to be inferior to learning $G\times$ B-bit representations directly by back-propagation, as mentioned in [29].

To tackle the above problems, we propose the parallel low-precision quantization scheme by learning multiple low-precision representations in parallel. It inherits the computation efficiency of parallel low-precision computation from Eq.5, and cures the solution degeneration problem via training B-bit representations through back-propagation. Specifically, we expand the activation channels of a convolutional layer by $G\times$, and split these activations into G groups along the channel dimension. Then we quantize the activations and weights to B-bit in each group, and there are G groups of B-bit weights in total. Fig.1(a) depicts the proposed parallel low-precision quantization scheme, which can be implemented naturally by quantized group convolution with expanded channels.

4.2 Cyclic Shuffle

Although the proposed parallel low-precision computation scheme enjoys wonderful hardware efficiency, it hinders information flowing across different groups of feature channels. To cure this problem, we propose a novel cyclic shuffle module to enhance information communication across different groups. First, we adopt cyclic permutation [4] to permute the channels in group-level. Given input X with shape $[N \times C \times H \times W]$, it can be reshaped to $\left[N \times G \times \hat{C} \times H \times W\right]$ where $\hat{C} = C/G$. We propose to re-order the channels in group-level via the cyclic permutation. Specifically, let $\mathbf{S} = \{0, 1, 2, 3 \cdots G-1\}$ denotes group indexing set, a cyclic permutation function $\pi(\cdot)$ for $\mathbf{S}$ is

$$\pi(i) \;=\; (i+1)\ mod\ G \tag{6}$$

A more clear notation of $\pi(\cdot)$ by Cauchy's two-line notation [36] is:

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & \cdots & G-2 & G-1 \\ 1 & 2 & 3 & 4 & 5 & \cdots & G-1 & 0 \end{pmatrix} \tag{7}$$

[4] For example, given the 4-bit number x, the lowest 2-bit number changes from 3 to 0 when x changes from 3 to 4 ([0 0 1 1] $\rightarrow$ [0 1 0 0])

where the first line denotes the channel group indexing set $\mathbf{S}$ and the second line denotes the corresponding order after permutation. After cyclic permutation, the output Y can be represented by:

$$Y[:,\pi(i),:,:,:] = X[:,i,:,:,:] \tag{8}$$

By using Eq.8, we can re-order feature channels by permuting the group-level feature cyclically. Under this scheme, the information from each group is received by its following one, which helps the information exchange between feature channels of different groups. Although we have exchanged the information between different groups by the cyclic permutation, each group still holds only one group of feature channels. In order to fuse the feature channels from different groups, we build a novel module, named cyclic shuffle, that composes of the cyclic permutation and a 1×1 group convolution with short-cut connection. Given the input X, the output Z of the cyclic shuffle module can be represented by the following formulas:

$$Y = \mathrm{CyclicPermute}(X) \tag{9}$$

$$Z = X + \mathrm{Conv}_{1\times1}(Y) \tag{10}$$

where CyclicPermute$(\cdot)$ denotes the cyclic permutation function (refer to Eq.8). In the cyclic shuffle module, the cyclic permutation re-orders the feature channels at group level, then the 1x1 group convolution learns a mapping function for channel fusing, finally the short-cut connection together with the element-wise addition operation aggregate feature channels from different groups. Fig. 1(b) depicts the proposed cyclic shuffle module.

Our proposed cyclic shuffle module enjoys benefits from three aspects: 1. According to Proposition. 1, after passing through G-1 cyclic shuffle modules, each group receives the information flows from all groups of feature channels, that helps cure the side-effect of group convolution. 2. The proposed cyclic shuffle module fuses feature channels at group level, which makes it possible for the intermediate results to stay on chip. As a result, it saves data access to the out-chip memory significantly. 3. Our cyclic shuffle module fuses feature channels in group-level while channel shuffle aggregates a part of feature channels from each group. Therefore, cyclic shuffle can be combined with channel shuffle to achieve higher network accuracy.

Proposition 1 *Suppose the number of groups is G, then G-1 cyclic shuffle modules are enough to assure that each group contains the information flows from all parallel groups.*

Proof. The proof is simple. The cyclic permutation $\pi(\cdot)$ in Eq.6 is a G-cycle [4], which means the group indexes will be the same as the original order after G times cyclic permutation. As each group fuses a the information flow from its previous group after passing through one cyclic shuffle module, then G-1 cyclic shuffle modules are enough to assure that each group contains all information from G group channels.

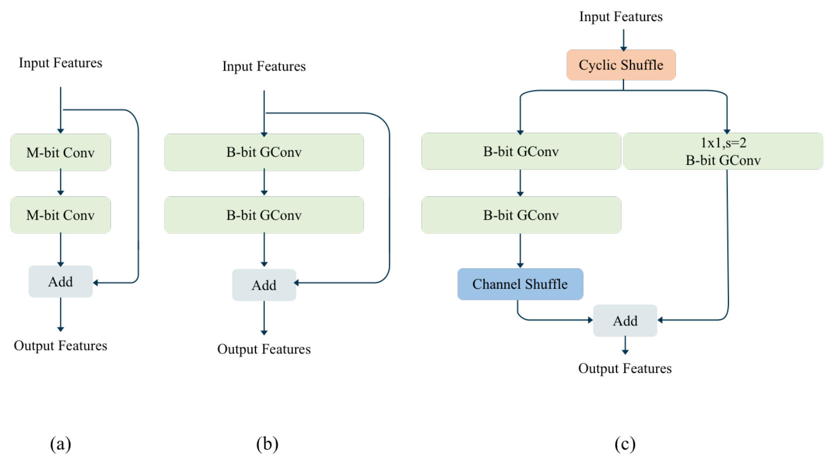

Fig. 2. (a) A typical building block of M-bit quantized ResNet network; (b) A typical building block of PalQuant with G=2;(c) A typical building block of PalQuant with shuffle modules (stride=2, G=2)

4.3 Overall Framework

In this subsection, we give the overall framework of our proposed method. To approximate M-bit computations on B-bit accelerator, we propose the parallel low-precision quantization scheme which expands the channel dimensions of each layer by G times ($G = ceil(M/B)$) and splits them into G parallel groups. Taking ResNet network quantization for an example, PalQuant makes a simple modification to the standard M-bit basic building block (Fig. 2 (a)): enlarging the input channels by $G\times$ and using B-bit quantized group convolution with the number of groups equals to G. The basic building block of PalQuant is depicted in Fig. 2 (b).

To cure the side-effect caused by the group convolution, we propose the cyclic shuffle module to help information communication between different groups. Considering that the cyclic shuffle module contains a 1×1 group convolution layer, we only place the cyclic shuffle module at the beginning of each stage , which brings little extra computational cost. Since cyclic shuffle and channel shuffle fuse information flows in different ways, we use channel shuffle module as a complement. As the channel shuffle in the middle of two consecutive group convolutions will impair the parallel computation flows, we move the channel shuffle module to the end of the convolution block. A typical building block of PalQuant with shuffle modules is depicted in Fig.2 (c).

5 Experiments

In this section, we first give details of experiment settings, then we compare our proposed algorithm PalQuant with state-of-the-art quantization algorithms. To demonstrate the hardware efficacy of PalQuant, we further conduct extensive experiments on CNN accelerators. Finally, we show the ablation study results and the generalization of PalQuant under different hyper-parameter B and G.

Network Architectures and Datasets. Since ResNet [17] is widely used in various computer vision tasks, here we adopt ResNet-18 and ResNet-34 as benchmark networks. All the experiments in this work are conducted on ImageNet dataset which has over 1 million training images and 50,000 validation images.

Training Details. In this work, we use uniform quantization for the input activation and weight of each convolutional or fully-connected layer. In the training stage, a straight-through estimator(STE) [3] is used to approximate the gradient through the rounding function. Based on this gradient estimator, we learn the lower bound l, upper bound u, and the scaling factor α through gradient back-propagation. Following experiment settings in previous work [23,13], the first and the last layer are not quantized unless otherwise specified. We implement our method under the Pytorch framework and train all the networks on a GPU server with eight Nvidia Titan 3090 GPUs. We adopt an SGD optimizer with the momentum set to 0.9. The weight decay is set to 1e-4, and the learning rate is initialized to 0.01 and adjusts with a cosine learning rate decay strategy. Following previous work [13], we train all the models from scratch for 90 epochs with a batch size of 256. In the training stage, random cropping, resizing, and horizontal flipping are adopted while only resizing and center cropping are used in the validation stage.

5.1 Comparison with State-of-the-arts Algorithms

We first compare our proposed PalQuant with state-of-the-art quantization algorithms. All the accuracy results of other methods, except for LSQ [13] and WRPN [29], are taken from corresponding papers. As LSQ [13] reports the results of pre-activation ResNet, we re-implement LSQ [13] under Pytorch framework and get the quantization results on standard ResNet. And we re-implement WRPN [29] via the same quantization function as ours, and get better results than the reported ones in [29].

Table 1 shows the top-1 accuracy of quantized ResNet-18 and ResNet-34 with different quantization precision. From the table, we can observe that PalQuant outperforms state-of-the-art methods in both accuracy and computational complexity. In detail, PalQuant (G=3) achieves +1.66% (72.66% v.s. 70.7%) higher top-1 accuracy than 4-bit LSQ [13] with 23.4% (22.3G v.s. 29.1G) less computational cost on ResNet-18. With comparable computational budget, PalQuant (G=4) outperforms 4-bit EWGS [23] by 1.06% on ResNet-34. Note that both LSQ[13] and EWGS[23] are finetuned from the pre-trained model, while our method are trained from scratch. In addition, PalQuant(G=3) obtains slightly higher top-1 accuracy than WRPN [29] with 23.4% less bitOps on ResNet-18. These results suggest that the proposed PalQuant has strong feature representation power, which leads to the highest accuracy in the experiments under less or equal computational budgets.

Table 1. Comparison Results of Quantized ResNet-18 and ResNet-34 on ImageNet. † Results of WRPN [29] on ResNet34 are taken from the paper

Method	Strategy	Precision	ResNet-18		ResNet-34	
			BitOps	Top1 Acc.	BitOps	Top1 Acc.
DSQ [15]	Fine-tuned	4b A,4b W	29.10G	69.60	58.74G	72.80
FAQ [28]	Fine-tuned	4b A,4b W	29.10G	69.80	58.74G	73.30
QIL [22]	Fine-tuned	4b A,4b W	29.10G	70.10	58.74G	73.70
EWGS [23]	Fine-tuned	4b A,4b W	29.10G	70.60	58.74G	73.90
LSQ [13]	Fine-tuned	4b A,4b W	29.10G	70.70	58.74G	73.50
PalQuant(**Ours**)	Scratch	**B=2,G=2**	**14.87G**	**71.12**	**29.68G**	**73.90**
PalQuant(**Ours**)	Scratch	**B=2,G=3**	**22.30G**	**72.36**	**44.53G**	**74.52**
PalQuant(**Ours**)	Scratch	**B=2,G=4**	29.73G	**72.74**	59.37G	**74.96**
EWGS [23]	Fine-tuned	6b A,6b W	65.48G	71.01	132.16G	74.14
LSQ [13]	Fine-tuned	6b A,6b W	65.48G	71.59	132.16G	74.43
WRPN [29]	Scratch	2×,2b A,2b W	29.10G	72.31	58.74G	73.32†
PalQuant(**Ours**)	Scratch	**B=2,G=3**	**22.30G**	**72.36**	**44.53G**	**74.52**
FAQ [28]	Fine-tuned	8b A,8b W	116.4G	70.00	234.94G	73.70
LSQ [13]	Fine-tuned	8b A,8b W	116.4G	71.10	234.94G	73.98
EWGS [23]	Fine-tuned	8b A,8b W	116.4G	71.14	234.94G	73.97
WRPN [29]	Scratch	2×,2b A,2b W	29.10G	72.31	58.74G	73.32†
PalQuant(**Ours**)	Scratch	**B=2,G=4**	**29.73G**	**72.74**	**59.37G**	**74.96**

5.2 Efficacy on Hardware Acceleration

We select the state-of-the-art CNN accelerator Bit Fusion [33] and BPVeC [14] to demonstrate the efficacy of the proposed PalQuant. Bit Fusion [33] is a systolic array-based accelerator designed for low-precision inference, it employs a bit-level decomposable fusion unit for 2/4/8-bit multiplication. To further improve hardware efficiency, BPVeC [14] exploits the Narrow Bit-width Vector Engines (NBVE), an energy and area efficient alternative to conventional MAC that consists of multiple 2-bit multipliers and an adder tree for SIMD-based inner-product computation. The basic building blocks of these two accelerators are shown in Fig. 3. In this experiment, we compare our proposed method against LSQ [13] in terms of inference latency and energy consumption. To mimic resource-constraint computing platforms, we configure both accelerators to 2-bit inference mode with a small on-chip buffer. In this scenario, the on-chip buffer cannot accommodate all weights and activations of a layer, leading to non-negligible data traffic in the memory hierarchy. We develop a cycle-accurate simulator to collect statistics of computation and on-chip buffer access. The power of computational logic and SRAM-based on-chip buffer under 45nm tech-

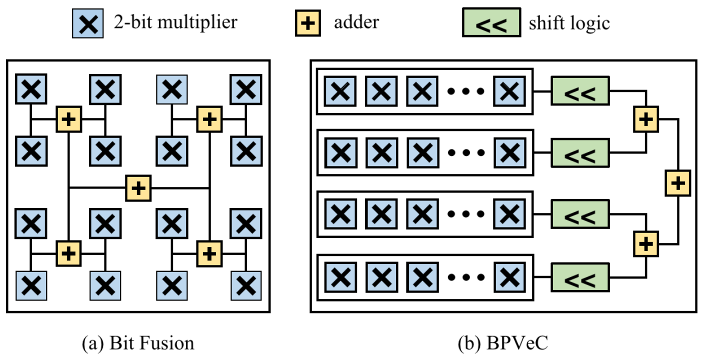

Fig. 3. The basic building blocks for 2-bit inference in Bit Fusion and BPVeC

nology are directly drawn from [33] and [14]. For off-chip data traffic, we assume 15pJ/bit per DDR4 access as in [14].

Table. 2 shows the results of hardware acceleration. It is clear to see that our proposed method consistently outperforms the baseline in performance, energy efficiency, and accuracy for the same precision of weight and activation. For example, PalQuant (G=2) obtains 0.52% higher accuracy, 1.78× speedup, and 1.91× energy efficiency simultaneously over 4-bit LSQ [13] on Bit Fusion accelerator [33]. The benefit mainly stems from the channel grouping in our method. On the one hand, channel grouping leads to reduced bitOps, which is the source of performance gain. On the other hand, channel grouping eliminates the data dependency between different groups, resulting in reduced on-chip buffer size and off-chip bandwidth requirement, which contributes to the low energy consumption. Besides, the channel shuffle unit at the end, rather than the middle, of a block encourages the intermediate results of group convolution staying on chip, which is another key for saving energy.

In the paper, we evaluate PalQuant on Bit Fusion and BPVeC, yet our algorithm can also witness similar benefits on other bit-parallel accelerators [32,6] which implement a high-precision multiplication as parallel low-precision multiplications followed by reduction. The fundamental reason is that compared to conventional low-precision quantization, our method can always reduce the number of computations and buffer accesses.

5.3 Ablation Study

In this subsection, we ablate the designs of the PalQuant algorithm based on the ResNet-18 network with hyper-parameter B=2 and G=2, and all the experiments follow the same training settings as above.

Ablation: Influence of Different Shuffle Modules Here we explore the benefits of the proposed cyclic shuffle module. Table. 3 provides the comparison of different shuffle modules. The baseline model in this table doesn't contain either

Table 2. Comparison of inference latency and energy consumption on two hardware accelerators. We use hyper-parameter B=2

DLA	Method	Precision	ResNet-18			ResNet-34		
			SpeedUp	Energy Efficiency	Top1 Acc.	SpeedUp	Energy Efficiency	Top1 Acc.
Bit Fusion [33]	LSQ [13]	4b A,4b W	1x	1x	70.70	1x	1x	73.50
	Ours	**G=2**	**1.78x**	**1.91x**	**71.12**	**1.78x**	**1.91x**	**73.90**
	LSQ [13]	6b A,6b W	1x	1x	71.59	1x	1x	74.43
	Ours	**G=3**	**2.52x**	**2.78x**	**72.36**	**2.50x**	**2.78x**	**74.52**
	LSQ [13]	8b A,8b W	1x	1x	71.14	1x	1x	73.97
	Ours	**G=4**	**3.21x**	**3.60x**	**72.71**	**3.13x**	**3.60x**	**74.96**
BPVeC [14]	EWGS [23]	4b A,4b W	1x	1x	70.60	1x	1x	73.90
	Ours	**G=2**	**1.77x**	**1.92x**	**71.12**	**1.87x**	**1.74x**	**73.90**
	EWGS [23]	6b A,6b W	1x	1x	71.01	1x	1x	74.14
	Ours	**G=3**	**2.56x**	**2.84x**	**72.36**	**2.46x**	**2.81x**	**74.52**
	EWGS [23]	8b A,8b W	1x	1x	71.14	1x	1x	73.97
	Ours	**G=4**	**3.12x**	**3.70x**	**72.71**	**3.07x**	**3.67x**	**74.96**

cyclic shuffle or channel shuffle. As we can see, using cyclic shuffle or channel shuffle separately achieves similar accuracy improvements. This indicates that both modules help the information communication between different groups. Besides, using cyclic shuffle and channel shuffle jointly demonstrate the highest accuracy gain (up to 3.19%) in terms of top-1 accuracy. This phenomenon verifies our assumption that cyclic shuffle and channel shuffle fuse information flow in different granularities and they may serve as a complement for each other.

We also explore the way to use channel shuffle. Table 3 shows that using per-stage or per-block channel shuffle has similar performance without cyclic shuffle. Together with cyclic shuffle, per-stage channel shuffle has higher accuracy improvement(+3.19% Top-1 accuracy on ImageNet) than per-block channel shuffle. In this paper, we use per-stage channel shuffle by default unless otherwise specified.

Ablation: Importance of Cyclic Permutation. As the cyclic shuffle module brings in one extra 1×1 group convolution, one may concern that most of the accuracy gains come from the additional computational cost. Experimental results in Table. 4 are against this opinion. PalQuant loses 0.76 % top-1 accuracy after dropping the cyclic permutation and keeping other parts unchanged while dropping the whole cyclic shuffle module causes 0.9 % top-1 accuracy decline. This indicates that the gain brought by the cyclic shuffle module is mainly due to cyclic permutation.

Table 3. Influence of different shuffle modules

Cyclic Shuffle	Channel Shuffle	Top1 Acc.	Top5 Acc.
		67.94	87.90
	per-block	70.31	89.58
	per-stage	70.24	89.51
✓		70.22	89.55
✓	per-block	70.62	89.69
✓	**per-stage**	**71.13**	**89.99**
Δ		**3.19↑**	**2.09↑**

Table 4. Importance of cyclic permutation

Module	Top1 Acc.	Top5 Acc.
PalQuant	**71.13**	**89.99**
w.o cyclic permutation	70.37 (−0.76)	89.57 (−0.42)
w.o cyclic shuffle	70.23 (−0.90)	89.51 (−0.48)

5.4 Generalization

Results With Different Numbers of Parallel Groups In this section, we explore the generalization of PalQuant under different numbers of parallel groups G. In Table. 5, our proposed method outperforms LSQ [13] consistently on a various number of parallel groups, which demonstrates the generalization ability of PalQuant. In addition, PalQuant shows superior performance (nearly 3% higher top-1 accuracy) than LSQ [13] under the same computational budget.

Table 5. Results with different numbers of parallel groups

Method	Precision	BitOps	Top1 Acc.	Top5 Acc.
LSQ [13]	4b A,4b W	29.10G	70.99	89.86
LSQ [13]	4b A,2b W	14.55G	69.18	88.93
PalQuant(**Ours**)	B=2,G=2	14.87G	**71.13**	**89.99**
LSQ [13]	6b A,6b W	65.48G	71.58	90.24
LSQ [13]	6b A,2b W	21.83G	69.47	89.06
PalQuant(**Ours**)	B=2,G=3	22.30G	**72.36**	**90.63**
LSQ [13]	8b A,8b W	116.40G	71.10	90.10
LSQ [13]	8b A,2b W	29.10G	70.71	89.70
PalQuant(**Ours**)	B=2,G=4	29.73G	**72.74**	**90.90**

Results With Different Quantization Precision Next, we explore the performance of PalQuant under different quantization precision B. Table. 6 shows that PalQuant outperforms LSQ [13] consistently across different low-precision bit-width, i.e. B=2, 3, and 4. This result means that our proposed scheme can be applied to CNNs accelerators with different quantization precision. We also quantize the weights with only half of activation bit-width in LSQ [13], which has nearly the same computation complexity as PalQuant. The result shows that our method outperforms LSQ [13] by a large margin under the comparable computational budget. The result of LSQ[13] with 6/3-bit quantization is provided in Appendix.

Table 6. Results with different quantization precision

Method	Precision	BitOps	Top1 Acc.	Top5 Acc.
LSQ [13]	4b A,4b W	29.10G	70.99	89.86
LSQ [13]	4b A,2b W	14.55G	69.18	88.93
PalQuant(**Ours**)	B=2,G=2	14.87G	**71.13**	**89.99**
LSQ [13]	6b A,6b W	65.48G	71.58	90.24
LSQ [13]	6b A,3b W	32.74G	70.896	89.712
PalQuant(**Ours**)	B=3,G=2	33.45G	**72.72**	**90.90**
LSQ [13]	8b A,8b W	116.40G	71.10	90.10
LSQ [13]	8b A,4b W	58.20G	71.46	90.15
PalQuant(**Ours**)	B=4,G=2	59.46G	**73.48**	**91.34**

6 Conclusion

In this paper, we propose the PalQuant, a parallel low-precision quantization method that achieves efficient and accurate high-precision network inference on low-precision accelerators. To help the information flow across parallel groups, we propose the cyclic shuffle module to aggregate parallel information at group level. Extensive experiments demonstrate that PalQuant outperforms state-of-the-art quantization methods in terms of both accuracy and computational complexity.

Acknowledgements This work was supported in part by National Key Research and Development Program of China (Grant No.2021ZD0201504), and National Natural Science Foundation of China (Grant No.62106267).

References

1. Abdel-Aziz, H., Shafiee, A., Shin, J.H., Pedram, A., Hassoun, J.H.: Rethinking floating point overheads for mixed precision DNN accelerators. CoRR **abs/2101.11748** (2021), `https://arxiv.org/abs/2101.11748`
2. Andri, R., Cavigelli, L., Rossi, D., Benini, L.: Yodann: An architecture for ultralow power binary-weight cnn acceleration. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **37**(1), 48–60 (2017)
3. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
4. Bogart, K.P.: Introductory combinatorics. Saunders College Publishing (1989)
5. Cai, Z., He, X., Sun, J., Vasconcelos, N.: Deep learning with low precision by half-wave gaussian quantization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5918–5926 (2017)
6. Camus, V., Mei, L., Enz, C., Verhelst, M.: Review and benchmarking of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing. Ieee Journal On Emerging And Selected Topics In Circuits And Systems **9**(4), 697–711 (2019)
7. Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y., Temam, O.: Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. ACM SIGARCH Computer Architecture News **42**(1), 269–284 (2014)
8. Chen, Y.H., Krishna, T., Emer, J.S., Sze, V.: Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE journal of solid-state circuits **52**(1), 127–138 (2016)
9. Choi, J., Wang, Z., Venkataramani, S., Chuang, P.I.J., Srinivasan, V., Gopalakrishnan, K.: Pact: Parameterized clipping activation for quantized neural networks. arXiv preprint arXiv:1805.06085 (2018)
10. Conti, F., Schiavone, P.D., Benini, L.: XNOR neural engine: a hardware accelerator IP for 21.6 fj/op binary neural network inference. CoRR **abs/1807.03010** (2018), `http://arxiv.org/abs/1807.03010`
11. Courbariaux, M., Bengio, Y., David, J.P.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: Advances in Neural Information Processing Systems. pp. 3123–3131 (2015)
12. Delmas Lascorz, A., Judd, P., Stuart, D.M., Poulos, Z., Mahmoud, M., Sharify, S., Nikolic, M., Siu, K., Moshovos, A.: Bit-tactical: A software/hardware approach to exploiting value and bit sparsity in neural networks. In: ASPLOS. pp. 749–763 (2019)
13. Esser, S.K., McKinstry, J.L., Bablani, D., Appuswamy, R., Modha, D.S.: Learned step size quantization. arXiv preprint arXiv:1902.08153 (2019)
14. Ghodrati, S., Sharma, H., Young, C., Kim, N., Esmaeilzadeh, H.: Bit-parallel vector composability for neural acceleration. 2020 57th ACM/IEEE Design Automation Conference (DAC) pp. 1–6 (2020)
15. Gong, R., Liu, X., Jiang, S., Li, T., Hu, P., Lin, J., Yu, F., Yan, J.: Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4852–4861 (2019)
16. Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P.: Deep learning with limited numerical precision. CoRR, abs/1502.02551 **392** (2015)

17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
18. Hou, L., Yao, Q., Kwok, J.T.: Loss-aware binarization of deep networks. arXiv preprint arXiv:1611.01600 (2016)
19. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. Advances in neural information processing systems **29** (2016)
20. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. pp. 4107–4115 (2016), https://proceedings.neurips.cc/paper/2016/hash/d8330f857a17c53d217014ee776bfd50-Abstract.html
21. Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al.: In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th annual international symposium on computer architecture. pp. 1–12 (2017)
22. Jung, S., Son, C., Lee, S., Son, J., Han, J.J., Kwak, Y., Hwang, S.J., Choi, C.: Learning to quantize deep networks by optimizing quantization intervals with task loss. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4350–4359 (2019)
23. Lee, J., Kim, D., Ham, B.: Network quantization with element-wise gradient scaling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021)
24. Li, F., Mo, Z., Wang, P., Liu, Z., Zhang, J., Li, G., Hu, Q., He, X., Leng, C., Zhang, Y., et al.: A system-level solution for low-power object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. pp. 0–0 (2019)
25. Li, F., Zhang, B., Liu, B.: Ternary weight networks. arXiv preprint arXiv:1605.04711 (2016)
26. Lin, Z., Courbariaux, M., Memisevic, R., Bengio, Y.: Neural networks with few multiplications. arXiv preprint arXiv:1510.03009 (2015)
27. McKinstry, J.L., Esser, S.K., Appuswamy, R., Bablani, D., Arthur, J.V., Yildiz, I.B., Modha, D.S.: Discovering low-precision networks close to full-precision networks for efficient embedded inference. arXiv preprint arXiv:1809.04191 (2018)
28. McKinstry, J.L., Esser, S.K., Appuswamy, R., Bablani, D., Arthur, J.V., Yildiz, I.B., Modha, D.S.: Discovering low-precision networks close to full-precision networks for efficient inference. In: 2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS). pp. 6–9. IEEE (2019)
29. Mishra, A., Nurvitadhi, E., Cook, J.J., Marr, D.: Wrpn: Wide reduced-precision networks. arXiv preprint arXiv:1709.01134 (2017)
30. Qiu, J., Wang, J., Yao, S., Guo, K., Li, B., Zhou, E., Yu, J., Tang, T., Xu, N., Song, S., et al.: Going deeper with embedded fpga platform for convolutional neural network. In: Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. pp. 26–35. ACM (2016)
31. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. In: European Conference on Computer Vision. pp. 525–542. Springer (2016)
32. Ryu, S., Kim, H., Yi, W., Kim, J.J.: Bitblade: Area and energy-efficient precision-scalable neural network accelerator with bitwise summation. In: Proceedings of the 56th Annual Design Automation Conference 2019. pp. 1–6 (2019)

33. Sharma, H., Park, J., Suda, N., Lai, L., Chau, B., Chandra, V., Esmaeilzadeh, H.: Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. In: 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). pp. 764–775 (2018). https://doi.org/10.1109/ISCA.2018.00069
34. Tann, H., Hashemi, S., Bahar, R.I., Reda, S.: Hardware-software codesign of accurate, multiplier-free deep neural networks. In: 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC). pp. 1–6. IEEE (2017)
35. Umuroglu, Y., Fraser, N.J., Gambardella, G., Blott, M., Leong, P., Jahre, M., Vissers, K.: Finn: A framework for fast, scalable binarized neural network inference. In: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. pp. 65–74 (2017)
36. Wussing, H.: The genesis of the abstract group concept: a contribution to the history of the origin of abstract group theory. Courier Corporation (2007)
37. Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., Cong, J.: Optimizing fpga-based accelerator design for deep convolutional neural networks. In: Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays. pp. 161–170 (2015)
38. Zhang, D., Yang, J., Ye, D., Hua, G.: Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In: Proceedings of the European conference on computer vision (ECCV). pp. 365–382 (2018)
39. Zhao, R., Song, W., Zhang, W., Xing, T., Lin, J.H., Srivastava, M., Gupta, R., Zhang, Z.: Accelerating binarized convolutional neural networks with software-programmable fpgas. In: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. pp. 15–24 (2017)
40. Zhou, A., Yao, A., Guo, Y., Xu, L., Chen, Y.: Incremental network quantization: Towards lossless cnns with low-precision weights. arXiv preprint arXiv:1702.03044 (2017)
41. Zhu, C., Han, S., Mao, H., Dally, W.J.: Trained ternary quantization. arXiv preprint arXiv:1612.01064 (2016)