DAQ: Channel-Wise Distribution-Aware Quantization for Deep Image Super-Resolution Networks

Cheeun Hong* Heewon Kim* Sungyong Baik Junghun Oh Kyoung Mu Lee
Department of ECE, ASRI, Seoul National University
{cheeun914, ghimhw, dsybaik, dh6dh, kyoungmu}@snu.ac.kr

Abstract

Since the resurgence of deep neural networks (DNNs), image super-resolution (SR) has recently seen a huge progress in improving the quality of low resolution images, however at the great cost of computations and resources. Recently, there has been several efforts to make DNNs more efficient via quantization. However, SR demands pixel-level accuracy in the system, it is more difficult to perform quantization without significantly sacrificing SR performance. To this end, we introduce a new ultra-low precision yet effective quantization approach specifically designed for SR. In particular, we observe that in recent SR networks, each channel has different distribution characteristics. Thus we propose a channel-wise distribution-aware quantization scheme. Experimental results demonstrate that our proposed quantization, dubbed Distribution-Aware Quantization (DAQ), manages to greatly reduce the computational and resource costs without the significant sacrifice in SR performance, compared to other quantization methods.

1. Introduction

Image super-resolution (SR), one of the fundamental computer vision tasks, targets for rejuvenating a given low-resolution (LR) input image to its high-resolution (HR) counterpart. The task boasts a wide range of application, including but not limited to medical [17, 39], satellite [3, 45] image processing, military surveillance [46], and automotive industry, giving rise to a great amount of attention from the computer vision community. Together with the interest from the community and the rapid development of deep neural networks (DNNs), SR has recently witnessed an unprecedented breakthrough in performance.

Recent SR works have enjoyed outstanding performance from increasing the network size [13, 28, 32] and employing more complex network structures, such as residual block and spatial attention [48, 47, 32]. However, a network with the larger size and complexity inherently demands heavier computing resources. The deployment of such resource-hungry systems becomes challenging especially in practical applications that often have limited resources available.

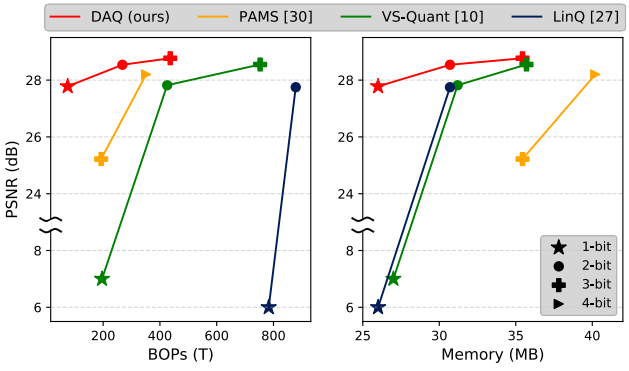
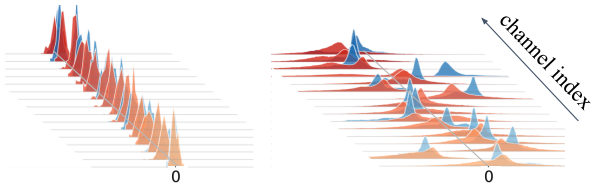


Figure 1: Quantizing EDSR(×4) [32] on Set14. The proposed quantization method (**DAQ**) achieves higher PSNR with less resource consumption (BOPs and memory).

Quantization has gained the popularity as one of the promising methods to reduce the amount of required resources for DNNs. Through discretizing the values of network weights [22], features [7], or gradients [50], quantization has significantly reduced the computation loads while minimizing the accuracy loss, particularly on high-level vision tasks, such as classification. SR has yet to benefit from the recent advances in quantization mainly because SR requires pixel-level accuracy [43, 35] and most SR methods have developed ad-hoc network structures [31], such as the absence of batch normalization (BN) [32]. Few recent works have attempted to achieve quantization for SR via either adopting a learnable parameter for each binary convolution weight (BinarySR [35]); employing a bit accumulation mechanism to approximate the full-precision convolution (BAM [43]); and learning the layer-wise max scale value of weights and features (PAMS [31]). However, these methods have neglected the unique distribution characteris-

*equal contribution



(a) ResNet-18 (b) EDSR

Figure 2: Channel-wise feature map distributions of two distinct images (red and blue) in pre-trained (a) ResNet [20] on image classification task and (b) EDSR [32] on image SR task. In SR networks, channels present diverse non-zero distributions that also vary upon the input image.

tics of SR networks, which have domain-specific designs.

On the other hand, this work starts with an observation that the recent state-of-the-art SR networks have distinct distributions for each channel, compared to networks that are designed for semantic-level tasks, as illustrated in Figure 2. Further, we observe that the channel distributions vary upon input images.

Upon the observations, we propose a **D**istribution-**A**ware **Q**uantization (DAQ) that performs an effective channel-wise quantization. In particular, DAQ computes the distribution of each channel to determine the per-channel quantization transformation (or scaling) parameters that are adaptive to each input. The adaptive scaling approach facilitates effective clipping of outliers that dominate the quantization error. Despite such compelling observation, per-channel quantization for feature map has yet to be deeply explored, due to the large computational overhead that could nullify the computation benefits of quantization [38]. We adjust the process of quantized convolution to significantly reduce the computational overhead, enabling DAQ to effectively *and* efficiently quantize SR networks.

Overall, our contributions can be summarized as follows:

- To the best of our knowledge, we present the first feasible channel-wise quantization method for image super-resolution networks in ultra-low precision with marginal accuracy loss.
- Our scheme is directly applicable to existing SR networks without any architectural modification or specialized training scheme.
- Experimental results show that the proposed DAQ outperforms state-of-the-art SR quantization methods, yielding higher PSNR with less resource consumption.
- DAQ effectively reduces quantization error, achieving accurate quantization on pre-trained SR networks even without retraining.

2. Related Work

2.1. Image super-resolution

Image super-resolution (SR) is a conventional computer vision problem that increases the resolution of an image while restoring its structural details. Convolutional neural network (CNN) based approaches [32, 48, 13, 28] have achieved a great performance improvement in this field. However, CNNs require heavy computational costs and memory footprints that greatly limit their applicability on resource-constrained devices. To alleviate this problem, recent works [14, 40, 23, 9, 18, 27] have focused on improving the model efficiency in terms of FLOPs, parameters, and run-time by designing novel lightweight network architectures. However, these networks still consume a large amount of resources since the convolution layers of CNNs operate in 32-bit floating-point (FP32).

2.2. Network quantization

Network quantization is a promising research area that aims to map 32-bit floating-point (FP32) values of feature maps and weights to lower bits (or precision) values. However, quantized networks usually suffer from severe accuracy degradation. Many works tackle this problem by optimizing the uniform [10, 50, 7] or non-uniform [19, 44, 26, 16] intervals between the quantized values. This work focuses on quantization with the uniform interval that can be accelerated in the hardware with simple arithmetic pipelines. Recently, research on specializing the precision of each part of the network [6, 34] and learning the quantization method without network retraining [29, 49, 2, 8, 37, 4] have been actively conducted. However, these approaches focus on semantic-level tasks, such as image classification, which may be an easier task for network quantization than SR, which requires pixel-level accuracy [35, 43].

To this end, few recent works have focused on network quantization specifically for SR. BinarySR [35] presented the first attempt to quantize SR networks, however for weights only. BAM [43] and BTM [25] either modified the network architecture with multiple skip connections or utilized specialized convolution layers to quantize both weights and feature maps. Recently, Hwang *et al.* [24] proposed an SR network compression method that jointly performs channel pruning and different-bit quantization for each layer. The conventional approaches have neglected the first and last layers, but FQSR [41] quantized all the weights and feature maps of SR networks. PAMS [31] may be considered to be similar to our proposed method in that it learns quantization ranges with uniform intervals, however for each layer in contrast to our method that performs channel-wise quantization, which aligns well with the observation from Figure 2. Furthermore, the works mentioned above have proposed specialized training schemes such as

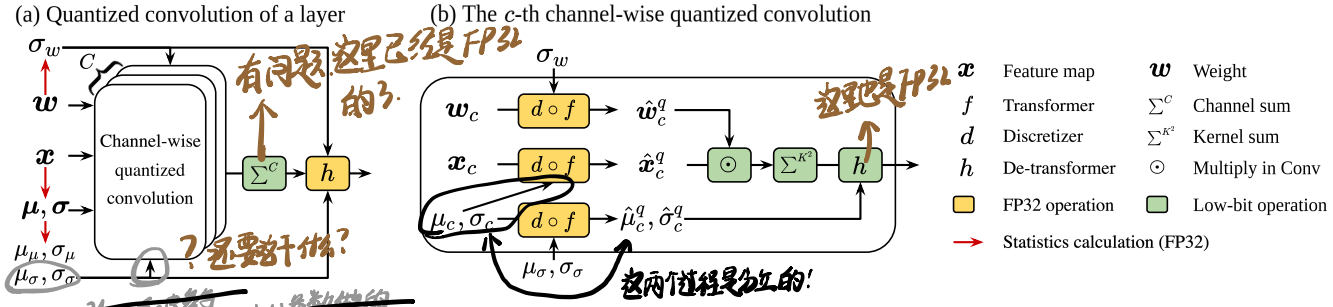


Figure 3: Overview of our distribution-aware quantization (DAQ) process.

knowledge distillation, self-supervised learning, and gradient update rules. Although these custom network architectures and training schemes may reduce performance degradation to some extent, their ad-hoc techniques or specialized networks generally make it difficult to apply quantization methods to various networks.

On the other hand, we investigate a quantization range tailored to each channel in SR networks, while existing SR-based methods quantized all channels in a feature map with the same range. Very few recent works have managed to achieve per-channel quantization, however for semantic-level tasks [42]. To the best of our knowledge, our method is the first to achieve per-channel scaled quantization that manages to greatly reduce performance degradation.

3. Method

3.1. Motivation

Network quantization for image super-resolution (SR) has suffered from substantial performance degradation that occurs due to quantization of feature map values [35]. Most previous works [43, 25] tackle this problem by manually designing architectures and heuristic training schemes while their quantization functions are *fixed* for a feature map. Our work started by investigating the characteristics of the feature maps in SR network.

The distribution of each channel in SR network (See Figure 2(b)) has diverse values for its mean and variance. Moreover, channel distributions are observed to be different for each input image. This observation has led to our assumption that quantizing each channel in different functions may effectively reduce the quantization error in SR networks while it may not be as beneficial in semantic-level tasks. To this end, we formulate a quantized convolution method as described in following sections.

3.2. Distribution-Aware Quantization (DAQ)

3.2.1 Overview

Quantization function is a sequence of processes that discretizes a FP32 tensor to lower precision. General pipeline of quantization on neural networks are processed as following. Input feature and weight are respectively scaled (or transformed) to integer range, and then discretized. Then, the quantized convolution is performed with discretized input feature of integer values and discretized weight of integer values. Then the convolution output is descaled back via de-transformation, which is a costly procedure for channel-wise quantization. Our proposed method follows a general pipeline of quantization function, except that a standard de-transformation is replaced with our newly proposed efficient de-transformation scheme. Overall, our DAQ convolution operates in a sequence of the following processes, as summarized in Figure 3(a):

- Channel-wise quantized convolution (Figure 3(b)).
  1. Per-channel transformation and discretization of input feature maps (Section 3.2.2)
  2. Transformation and discretization of convolution filter weights (Section 3.2.3).
  3. Per-channel convolution over quantized weights and quantized input feature maps (dot product operation in a sliding window manner: multiplication of the two and kernel-wise sum).
  4. Proposed computationally efficient per-channel de-transformation (Section 3.2.4).
- Element-wise sum of outputs of the channel-wise convolution ($\sum^{C}$) in low-bit precision (Section 3.2.4).
- De-transformation of the convolution output to full precision FP32.

3.2.2 Quantization of feature map

As we discussed above, we define a quantization function specialized in each channel. Formally, given a feature map $\boldsymbol{x} \in \mathbb{R}^{B\times C\times H\times W}$, we quantize b-th element of the mini-batch and c-th channel $\boldsymbol{x}_{b,c} \in \mathbb{R}^{H\times W}$, where B, C, H, and W denote the mini-batch size, number of channels, height and width of the feature map, respectively. In the subsequent sections, we drop the subscript b for notational simplicity.

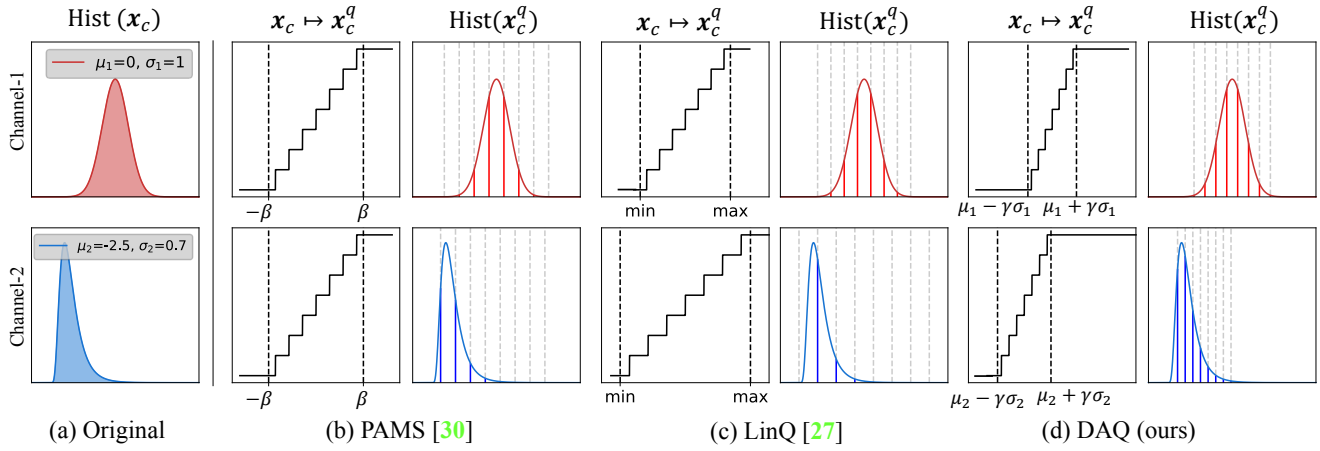


Figure 4: Illustration of feature map quantization ranges with examples of two distribution from different channels (red and blue). (b) PAMS [31] exploits learnable layer-wise fixed scale parameter β. (c) channel-wise linear quantization (LinQ) [36] utilizes the whole input tensor range as the quantization range, keeping the outliers that dominate quantization error. (d) Our DAQ adaptively determines the channel-wise quantization range with μ and σ that effectively clip outliers. $\gamma = 2^{n-1} \cdot s(n)$ is the network hyperparameter.

Adaptive transformer Existing quantization works for SR [31] often adopt linear scaling before discretizing the FP32 values to integer values. The linear scaling might be a good transformer for tensor with a zero-mean and bell-shaped distribution, like the channel-wise feature of ResNet visualized in Figure 2(a). However, the distribution of each channel, as shown in Figure 2(b), has a non-zero mean and even varies across different input images, suggesting that an adaptive transformer might be a better alternative for each channel distribution. We define the transformer $f(\cdot)$ using a standardization function to normalize values of $\boldsymbol{x}_c$ before discretization. $f(\cdot)$ is formally given by,

$$\hat{\boldsymbol{x}}_c \equiv f(\boldsymbol{x}_c) = \frac{\boldsymbol{x}_c - \mu_c}{\sigma_c \cdot s(n)}, \tag{1}$$

where μ_c and σ_c, respectively, denote the average and the standard deviation of the c-th channel $\boldsymbol{x}_c$ and $s(n)$ determines an uniform interval between quantized values by the bit-width n. Table 1 presents the actual values of $s(n)$ which represent the optimal quantization for Gaussian distribution. We interchangeably refer to statistics μ_c and σ_c as quantization transformation parameters as they are used to perform transformation during quantization process.

Adaptive discretizer In general, a discretizer clips the values of $\hat{\boldsymbol{x}}_c$ to a range given by the quantization bit-width n, followed by the mapping of the values to integers. Our adaptive discretizer is formulated as follows:

$$\hat{\boldsymbol{x}}_c^q \equiv d(\hat{\boldsymbol{x}}_c) = \lfloor g(\hat{\boldsymbol{x}}_c) \rceil, \tag{2}$$

Table 1: Step sizes $s(n)$ with respect to bit-width n. The sizes are the optimal values for uniform quantization of Gaussian distribution [33].

Bit-width n	1	2	3	4	8
Step size $s(n)$	1.596	0.996	0.586	0.335	0.031

where $g(\cdot)$ is the clamp function of a range $(-2^{n-1} + \alpha, 2^{n-1} + \alpha]$ and $\lfloor \cdot \rceil$ rounds up a value to the nearest higher integer. The shifting parameter α is a function of $s(n)$, μ_c, and, σ_c formally given by,

$$\alpha = \begin{cases} \max(2^{n-1} - \frac{\mu_c}{\sigma_c \cdot s(n)} - 1, 0) & \text{after ReLU} \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

allowing the minimum quantized value (of original scale) to be zero after ReLU function. $d(\cdot)$ is distribution-dependent due to its dependency on channel statistics, μ_c and σ_c.

Figure 4 illustrates the effectiveness of our distribution-aware quantization (DAQ). Our method reduces quantization error by a large margin for diverse channel distributions. We use $\boldsymbol{x}_c^q \equiv f^{-1}(\hat{\boldsymbol{x}}_c^q)$ for visualization, where $\hat{\boldsymbol{x}}_c^q$ denotes low-bit quantized integer and f is the transform function of each method.

3.2.3 Quantization of weight

Weight distributions over filters are empirically observed to have less degree of variance than feature maps in SR networks. Hence, a single quantization function is applied to the weight of each convolution $\boldsymbol{w} \in \mathbb{R}^{C \times C_{out} \times K \times K}$, where C and C_{out} denote the number of input and output chan-

nels, respectively, and K is the kernel size of the convolution filter. The distribution of weight values are observed to be bell-shaped with zero-mean, and is independent of the input image. Thus, for quantizing the weight $\boldsymbol{w}$, we assume the mean of $\boldsymbol{w}$ to be 0, which simplifies a transformer for weight quantization $f(\boldsymbol{w})$ as $f(\boldsymbol{w}) = \frac{\boldsymbol{w}}{\sigma_w \cdot s(n)}$, where σ_w denotes the standard deviation of $\boldsymbol{w}$. The shifting parameter α of the weight discretizer is set to 0 since the weight values are not zeroed out by activation.

3.2.4 Adaptive de-transformer

Standard per-channel de-transformer Once per-channel quantized convolution response is obtained, normally one would de-scale (or de-transform) each per-channel convolution output back to its full-precision before the summation of channel-wise convolution outputs along the channel dimension. However, de-transformation of each channel-wise convolution output to full precision (FP32) using quantization transform parameters (in this case, channel-wise statistics μ_c and σ_c) will result in a large computational overhead of the overall quantized convolution process, due to the FP32 summation of de-transformed convolution responses along the channel dimension.

Efficient per-channel de-transformer Recently, Dai *et al.* [11] reduces the overhead by decomposing scaling factors into channel-specific quantized values and a global FP32 value for the low-bit addition in image classification networks. Similarly, our solution achieves efficient implementation but with the distribution awareness. We propose to perform quantization on statistics $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ to obtain quantized statistics, allowing us to perform the summation of the channel-wise convolution responses in low-bit precision. Formally, recall μ_c and σ_c, the mean and the standard deviation of the c-th channel in Equation (1), as the c-th elements of $\boldsymbol{\mu} \in \mathbb{R}^C$ and $\boldsymbol{\sigma} \in \mathbb{R}^C$. We **q**uantize the **q**uantization transform parameters, thus named QQ. The quantized transform parameters using Equation (1) and (2) approximate their FP32 versions:

$$\mu_c \approx \mu_c^q \equiv \sigma_\mu \cdot s(m) \cdot \hat{\mu}_c^q + \mu_\mu, \tag{4}$$

$$\sigma_c \approx \sigma_c^q \equiv \sigma_\sigma \cdot s(m) \cdot \hat{\sigma}_c^q + \mu_\sigma, \tag{5}$$

where m is the bit-width of QQ, and σ_* and $\mu_*, * \in \{\mu, \sigma\}$, denote the mean and the standard deviation of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Note that σ_* and $\mu_*, * \in \{\mu, \sigma\}$, is FP32 values which are independent to channels while $\hat{\sigma}_c^q$ and $\hat{\mu}_c^q$ are low-bit integers specific to the channels. Using the quantizing quantization parameters, our quantized channels become,

$$\begin{aligned} \boldsymbol{x}_c^q &= \sigma_c \cdot s(n) \cdot \hat{\boldsymbol{x}}_c^q + \mu_c \\ &\approx \sigma_c^q \cdot s(n) \cdot \hat{\boldsymbol{x}}_c^q + \mu_c^q. \end{aligned} \tag{6}$$

As a result, we can place the de-transformation process after the summation of the channel-wise convolution and eliminate the potential possibility of a computation overhead that could have occurred from channel-wise quantized convolution (please refer to Section 3.3.1 and the supplementary document).

3.3. Hardware Implementation

3.3.1 Computation of per-channel detransformer

As discussed in Section 3.2.4, the computational overhead from the standard per-channel de-transformer is largely reduced by our efficient adaptive de-transformation process. The approximation by Equation (6) replaces a large proportion of high-precision operations with low-precision operations by modifying arithmetic pipelines. For instance, the sum of quantized channels ($\sum_c \boldsymbol{x}_c^q$) should be operated in FP32 ($\sum_c (\sigma_c \cdot s(n) \cdot \hat{\boldsymbol{x}}_c^q + \mu_c)$) due to the channel-specific mean and standard deviation of FP32 values. However, the sum of the approximated quantized channels ($\sum_c (\sigma_c^q \cdot s(n) \cdot \hat{\boldsymbol{x}}_c^q + \mu_c^q)$) can expand the low-bit channel sum ($\sigma_\sigma \cdot s(n) \cdot s(m) \sum_c \hat{\sigma}_c^q \cdot \hat{\boldsymbol{x}}_c^q + \mu_\sigma \cdot s(m) \sum_c \hat{\boldsymbol{x}}_c^q + \sigma_\mu \cdot s(m) \sum_c \hat{\mu}_c^q + \mu_\mu$).

3.3.2 FP32 Arithmetic Computation

The proposed method adaptively quantizes the value of interest with respect to its distribution by using standardization function as the tranformer. The computation complexity of Equation (1) can be specified into the complexity for calculating the mean and the standard deviation, and for the transformer function itself, of which are both $O(n)$. Specifically, the overall complexity of Equation (1) is $C(5HW + 3)$ for the feature map and $3K^2 C_{in} C_{out}$ for the weight. Weight standardization is independent of input images and thus occurs only once before the inference, which allows us to measure its computational complexity only once. On the other hand, the complexity of feature standardization is calculated per test image.

3.4. Training

Recently, training quantized networks (or *quantization-aware training*) is a well-known approach for accuracy recover. The back-propagation in quantized networks calculates the gradient of loss ℓ with respect to $\boldsymbol{w}^q$. Quantization function is generally not differentiable, and thus not possible to directly apply back-propagation. Consequently, we adopt the straight-through estimator [50] to approximate the gradients calculation. We approximate the partial gradient $\frac{\partial \boldsymbol{w}^q}{\partial \boldsymbol{w}}$ with an identity mapping ($\frac{\partial \boldsymbol{w}^q}{\partial \boldsymbol{w}} \approx 1$). Accordingly, $\frac{\partial \ell}{\partial \boldsymbol{w}^q}$ can be approximated by

$$\frac{\partial \ell}{\partial \boldsymbol{w}^q} = \frac{\partial \ell}{\partial \boldsymbol{w}} \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{w}^q} \approx \frac{\partial \ell}{\partial \boldsymbol{w}}. \tag{7}$$

4. Experiments

4.1. Experimental setup

Models and datasets To verify the effectiveness and the generalizability of the proposed quantization method, we conduct experiments on widely used SR networks, including EDSR [32], RDN [48] and SRResNet [30]. We use pre-trained models publicly available at codes[1]. We retrain the quantized models on DIV2K dataset [1], which consists of 800 2K-resolution images. We finetune the DAQ-applied EDSR model using the training settings from [32], where a batch size is set to be 4, a learning rate is 10^{-4}, and the number of updates is 3×10^5. DAQ-applied RDN is fine-tuned using 3×10^4 updates with batch size 16, and initial learning rate 10^{-4}. SRResNet-DAQ is finetuned with an initial learning rate of 10^{-5} for 6×10^5 iterations while we follow the other training settings from [30]. We evaluate the quantized models on four standard benchmark datasets: Set5, Set14, B100, and Urban100.

Implementation details Our proposed quantization method is directly applied to existing SR models without modifying the network architecture. Along with the majority of quantization works, we quantize weights and feature maps in all layers except the first convolution and the last reconstruction modules. We set m=4 for quantizing quantization parameters.

BOPs The number of floating-point operations (FLOPs) is a conventional measure for the computational complexity of full-precision models. However, a low-precision model often consists of operations with inputs of different low bit-widths. To take the low-precision operations into account, we weight the number of bit operations by multiplying $n \cdot m$, where n and m denote the bit-widths of two different inputs, respectively for each operation. The weighted number of bit operations is referred to as BOPs. The reported BOPs are measured to generate a FHD image (1920×1080).

Energy A quantized network boasts energy efficiency as lower-precision operations consume less energy. We adopt the approximation of energy consumption introduced in [12, 21] and compute it for generating a FHD image.

Memory Another benefit of network quantization is the reduction in memory requirement for model storage. The n-bit weights require $\frac{n}{32}$ amount of memory of FP32 weights.

4.2. Comparisons with SotA methods

SR quantization methods To demonstrate the effectiveness of our channel-wise quantization paradigm, we compare our method with PAMS [31] and BinarySR [35], which adopt per-layer quantization ranges. Without architecture changes of existing networks, Table 2 presents outstanding performances of our method (DAQ) on most measures and all benchmarks. Specifically, RDN-DAQ achieves 1 dB higher PSNR than RDN-PAMS on Urban100 with less resource consumption of BOPs, Energy, and Memory. SRResNet-DAQ requires only 50 % BOPs of SRResNet-BinarySR of which weights are 1-bit while outperforming 1.51 dB on Set5.

On the other hand, BTM [25] and BAM [43] employ specialized architectures or training schemes for low-precision SR networks. Our proposed quantization method is orthogonal to these techniques, providing additional performance gains, as shown in the supplementary document.

Comparison with per-channel quantization methods To show the importance of distribution awareness in per-channel quantization function as proposed in this work, we compare DAQ with the SR models quantized by other per-channel quantization methods LinQ [29] and VS-Quant [11], which are however tailed for classification and do not consider per-channel distribution in contrast to DAQ. Table 2 shows that EDSR-DAQ outperforms EDSR-LinQ and EDSR-VS-Quant over than 1 dB on Urban100 while consuming less resources, validating the benefits of formulating the channel-wise quantization function with distribution statistics.

Qualitative comparisons Figure 5 presents the output images from SotA models compared in Table 2. Our method generates a visually clean output image, while RDN-PAMS [31] and SRResNet-BinarySR [35] suffer from a checkerboard artifact. Interestingly, EDSR-LinQ generates a sharper image than EDSR-PAMS with lower PSNR scores. The result of SRResNet-DAQ has a more realistic structure (at the bottom of the image).

Comparison with methods without retraining To further demonstrate the effectiveness of DAQ in dynamically reducing the quantization error, we evaluate DAQ and other quantization methods without fine-tuning, as shown in Table 3. DAQ is shown to greatly reduce the quantization error to a large extent compared to other methods. EDSR-LinQ calculates min/max values for each channel-wise tensor during inference, which is the same process in the previous section. DFQ [37] controls weight ranges and biases of the pre-trained models during inference, proposed for image classification. EDSR-DFP is a re-implemented version of DFQ on EDSR, based on the publicly available codes. EDSR-DAQ outperforms the compared methods over 1.4 dB with less computation costs. The results corroborate our

[1] https://github.com/thstkdgus35/EDSR-PyTorch

Table 2: Comparisons of existing quantization methods on EDSR [32], RDN [48] and SRResNet [30] of scale 4.

Method	Precision		BOPs	Energy	Memory	Parameters	PSNR (dB)			
	w	a					Set5	Set14	B100	Urban100
EDSR	32	32	10019.3 T	22504.3 mJ	172.3 MB	43.1M	32.46	28.77	27.69	26.54
EDSR-PAMS [31]	4	4	351.3 T	366.7 mJ	40.2 MB	43.1M	31.59	28.20	27.32	25.32
EDSR-LinQ [29]	2	2	878.4 T	2616.6 mJ	30.7 MB	43.1M	31.08	27.75	27.05	24.45
EDSR-VS-Quant [11]	2	2	425.6 T	555.7 mJ	31.0 MB	43.1M	31.10	27.82	27.11	24.94
EDSR-DAQ (ours)	2	2	267.8 T	328.6 mJ	30.7 MB	43.1M	32.05	28.54	27.50	25.97
RDN	32	32	5636.0 T	12659.0 mJ	89.2 MB	22.3M	32.24	28.67	27.63	26.29
RDN-PAMS [31]	4	4	204.8 T	222.3 mJ	12.3 MB	22.3M	30.44	27.54	26.87	24.52
RDN-LinQ [29]	2	2	474.5 T	1044.3 mJ	6.9 MB	22.3M	30.90	27.73	27.05	24.65
RDN-VS-Quant [11]	2	2	239.1 T	311.6 mJ	6.9 MB	22.3M	31.16	27.89	27.12	24.83
RDN-DAQ (ours)	2	2	168.8 T	220.1 mJ	6.9 MB	22.3M	31.61	28.21	27.31	25.52
SRResNet	32	32	324.6 T	728.2 mJ	6.1 MB	1.6M	31.94	28.43	27.46	25.71
SRResNet-BinarySR [35]	1	32	38.9 T	58.3 mJ	1.5 MB	1.6M	30.16	27.19	26.66	24.24
SRResNet-LinQ [29]	2	2	37.8 T	58.8 mJ	1.7 MB	1.6M	31.44	28.03	27.21	25.05
SRResNet-VS-Quant [11]	2	2	22.5 T	41.8 mJ	1.7 MB	1.6M	31.24	27.95	27.15	24.89
SRResNet-DAQ (ours)	2	2	19.1 T	38.5 mJ	1.7 MB	1.6M	31.67	28.26	27.32	25.39

Figure 5: Qualitative evaluation of existing quantization methods on EDSR [32], RDN [48] and SRResNet [30] of scale 4, as in Figure 5. Evaluation is done on 'barbara' of Set14 and 'img060' and 'img097' of Urban100. Quantitative measures are also denoted (PSNR / BOPs).

Table 3: Comparison of quantization methods *without retraining* on pre-trained EDSR [32] of scale 4.

Method	Precision		BOPs	Energy	Memory	PSNR (dB)			
	w	a				Set5	Set14	B100	Urban100
EDSR	32	32	10019.3 T	22504.3 mJ	172.3 MB	32.46	28.77	27.69	26.54
EDSR-LinQ [29]	2	2	878.4 T	2616.6 mJ	30.7 MB	30.26	26.78	26.44	23.14
EDSR-DFQ [37]	4	4	351.3 T	366.7 mJ	40.2 MB	31.20	27.77	26.29	24.18
EDSR-DAQ	2	2	267.8 T	328.6 mJ	30.7 MB	31.42	28.12	27.23	25.56

observations from channel distributions (Figure 2 and assumptions that the per-channel and distribution-aware quantization is effective in SR networks.

Table 4: Ablation study of the proposed method on feature map quantization. 2-bit quantization is evaluated on EDSR.

Granularity	Distribution aware scaling	QQ	BOPs	PSNR (Urban100)
Layer	✗	✗	83.5 T	6.99 dB
Layer	✓	✗	90.1 T	7.17 dB
Channel	✗	✗	878.4 T	7.26 dB
Channel	✓	✗	889.3 T	25.98 dB
Channel	✓	✓	**267.8 T**	**25.97 dB**

Table 5: Ablation study of the proposed method on weight quantization. Evaluation done on EDSR.

Distribution aware scaling	Granularity	Precision		
		4-bit	2-bit	1-bit
✓	Layer	26.48	26.07	25.07
	Output channel	26.50	26.15	24.94
	Input channel	26.47	26.10	25.09
	Kernel	26.45	26.12	25.31
✗	Layer	26.20	7.16	7.17
	Output channel	26.51	7.11	7.17
	Input channel	26.50	23.84	7.17
	Kernel	26.54	26.47	17.73

Table 6: Ablation study on various distribution assumptions.

EDSR (×4)	Gaussian	Uniform	Laplacian	Gamma
4-bit	26.51	24.52	26.21	26.25
1-bit	24.20	23.64	23.82	23.71

4.3. Ablation study

Feature map quantization To verify the efficacy of our quantization scheme in practical SR networks, we do ablation on each of our contribution. In Table 4, granularity represents the tensor to be quantized, where "Layer" quantizes the feature map with a single quantization range and "Channel" quantizes each input channel with a specialized range. The non-distribution-aware scaling uses min/max values to normalize distributions. As shown in Table 4, channel-wise and distribution-aware property are both important for accurate low-bit quantization, but suffer from heavy computational resources. Utilizing quantization function for the quantization transform parameters (QQ) significantly reduces BOPs, while maintaining a high PSNR.

Weight quantization Though our main contributions in the method are focused on feature map quantization, we also evaluate the effectiveness of the proposed weight quantization in Table 5. Distribution awareness in quantization plays an key role in maintaining the performance, especially in lower-precisions. Layer-wise quantization presents a better trade-off between computational efficiency and performance among different extents of granularity.

Distribution assumption Scaled step size $s(n)$ in quantization function is a pre-determined hyper-parameter. Many quantization works [5, 15] select different scaled step size for different bit-widths assuming a certain distribution of the tensor to be quantized. We follow the Gaussian assumption in the main experiments, while results on different distribution assumptions, including Gaussian, Uniform, Laplacian, and Gamma distribution are shown in Table 6.

5. Conclusion

We propose an efficient yet effective quantization method for deep SR networks. Based on the observation that each channel of SR networks has distinct distribution, we introduce a channel-wise **D**istribution-**A**ware **Q**uantization (DAQ) method. However, channel-wise quantization with standard de-transformation incurs a large computational overhead. Instead, we utilize an efficient de-transformation by quantizing quantization parameters, reducing the computational costs of channel-wise quantization significantly. Our proposed scheme reduces the quantization error to a large extent, achieving high performance even without quantization-tailored training. Evaluations on various SR networks demonstrate the outstanding performance of DAQ, compared to other quantization methods, with similar amount of resource consumptions.

Acknowledgment This work was supported in part by IITP grant funded by the Korean government (MSIT) [NO.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)]

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPRW*, 2017.

[2] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *NIPS*, 2019.

[3] K. Vani C. Heltin Genitha. Super resolution mapping of satellite images using hopfield neural networks. *Recent Advances in Space Technology Services and Climate Change (RSTSCC)*, 2010.

[4] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *CVPR*, 2020.

[5] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *CVPR*, 2017.

[6] Zhaowei Cai and Nuno Vasconcelos. Rethinking differentiable search for mixed-precision neural networks. In *CVPR*, 2020.

[7] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

[8] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *ICCVW*, 2019.

[9] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, Jixiang Li, and Qingyuan Li. Fast, accurate and lightweight super-resolution with neural architecture search. *arXiv preprint arXiv:1901.07261*, 2019.

[10] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Training deep neural networks with low precision multiplications. In *ICLRW*, 2015.

[11] Steve Dai, Rangha Venkatesan, Brian Zimmer Mark Ren, William Dally, and Brucek Khailany. Vs-quant: Per-vector scaled quantization for accurate low-precision neural network inference. In *MLSys*, 2021.

[12] William Dally. High-performance hardware for machine learning. In *NeurIPS Tutorial*, 2015.

[13] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 2014.

[14] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*. Springer, 2016.

[15] Zhen Dong, Zhewei Yao, Yaohui Cai, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. In *NeurIPS*, 2020.

[16] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

[17] Hayit Greenspan. Super-resolution in medical imaging. *The Computer Journal, Oxford University Press Oxford, UK*, 2009.

[18] Yong Guo, Yongsheng Luo, Zhenhao He, Jin Huang, and Jian Chen. Hierarchical neural architecture search for single image super-resolution. *arXiv preprint arXiv:2003.04619*, 2020.

[19] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[21] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014.

[22] Lu Hou and James T. Kwok. Loss-aware weight quantization of deep networks. In *ICLR*, 2018.

[23] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *ACMMM*, 2019.

[24] Jiwon Hwang, A. F. M. Shahab Uddin, and Sung-Ho Bae. A layer-wise extreme network compression for super resolution. *IEEE Access*, 2021.

[25] Xinrui Jiang, Nannan Wang, Jingwei Xin, Keyu Li, Xi Yang, and Xinbo Gao. Training binary neural network without batch normalization for image super-resolution. In *AAAI*, 2021.

[26] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *CVPR*, 2019.

[27] Heewon Kim, Seokil Hong, Bohyung Han, Heesoo Myeong, and Kyoung Mu Lee. Fine-grained neural architecture search. *arXiv preprint arXiv:1911.07478*, 2019.

[28] Jiwon Kim, Jungkwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016.

[29] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[30] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[31] Huixia Li, Chenqian Yan, Shaohui Lin, Xiawu Zheng, B. Zhang, F. Yang, and Rongrong Ji. Pams: Quantized super-resolution via parameterized max scale. In *ECCV*, 2020.

[32] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPRW*, 2017.

[33] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *ICML*, 2016.

[34] Qian Lou, Feng Guo, Lantao Liu, Minje Kim, and Lei Jiang. Autoq: Automated kernel-wise neural network quantization. *arXiv preprint arXiv:1902.05690*, 2019.

[35] Yinglan Ma, Hongyu Xiong, Zhe Hu, and Lizhuang Ma. Efficient super resolution using binarized neural network. In *CVPRW*, 2019.

[36] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

[37] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *ICCV*, 2019.

[38] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization, 2021.

[39] M. Dirk Robinson, Stephanie J. Chiu, Cynthia A. Toth, Joseph A. Izatt, Joseph Y. Lo, and Sina Farsiu. New applications of super-resolution in medical imaging. *Digital Imaging and Computer Vision, CRC Press.*, 2010.

[40] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016.

[41] Hu Wang, Peng Chen, Bohan Zhuang, and Chunhua Shen. Fully quantized image super-resolution networks. In *ACMMM*, 2021.

[42] Ziwei Wang, Jiwen Lu, Chenxin Tao, Jie Zhou, and Qi Tian. Learning channel-wise interactions for binary convolutional neural networks. In *CVPR*, 2019.

[43] Jingwei Xin, Nannan Wang, Xinrui Jiang, Jie Li, Heng Huang, and Xinbo Gao. Binarized neural network for single image super resolution. In *ECCV*, 2020.

[44] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *ECCV*, 2018.

[45] H. Zhang, Z. Yang, L. Zhang, and H. Shen. Super resolution reconstruction for multi-angle remote sensing images considering resolution differences. *Remote Sensing*, 2014.

[46] L. Zhang, H. Zhang, H. Shen, and P. Li. A super resolution reconstruction algorithm for surveillance images. *Signal Processing*, 2010.

[47] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, 2018.

[48] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, 2018.

[49] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. *arXiv preprint arXiv:1901.09504*, 2019.

[50] 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.

DAQ: Channel-Wise Distribution-Aware Quantization for Deep Image Super-Resolution Networks
– *Supplementary Document* –

Cheeun Hong* Heewon Kim* Sungyong Baik Junghun Oh Kyoung Mu Lee
Department of ECE, ASRI, Seoul National University
{cheeun914, ghimhw, dsybaik, dh6dh, kyoungmu}@snu.ac.kr

A. Implementation details on computation cost

In the main paper, computational resources (BOPs and estimated energy consumption) are measured with respect to quantization bit-width, considering the overflow of low-bit arithmetic operations. Overflow occurs when an arithmetic operation attempts to create a numeric value that is outside the range that can be possibly represented. Taking integer overflow into account is especially essential in low-bit networks, since the output ranges of low-bit multiplication and addition are strictly limited. Various techniques are used to avoid the integer overflow, such as using the overflow checker or value sanity testing. For ultra-low precision operations where the integer overflow is highly likely to occur, we design an appropriate large bit-width for each operation, under the assumption of an integer overflow. For instance, when the sum is accumulated over vector of size C with each n-bit element, the output buffer is $n+log_2(C-1)$-bit. Also, the output buffer for multiplication of two n-bit elements is $(2n-1)$-bit.

B. Derivation of convolution operations for DAQ

Section 3.3 in the main paper claims that quantization can step forward to hardware efficiency simply by postponing the de-transformation process as late as possible (See Equation (B), (D), and (G)). As more operations are done before de-transformation, in other words, in the state of real integer, the more efficient the quantization becomes.

Section 3.2 of the main paper presents a convolution operation with a n-bit *channel-wise* quantized feature map and a n-bit layer-wise quantized weight tensor. Given a feature map $\boldsymbol{x} \in \mathbb{R}^{C\times H\times W}$, c-th channel, j,k-th element of the feature map is denoted as $x_c[j,k]$ with the indexing operator $[\cdot,\cdot]$. Given a weight $\boldsymbol{w} \in \mathbb{R}^{C\times C_{out}\times K\times K}$, c-th input channel and i,u,v-th element of a part of weight tensor $\boldsymbol{w} \in \mathbb{R}^{C\times C_{out}\times K\times K}$ is denoted as $w_c[i,u,v]$ with indexing operator $[\cdot,\cdot,\cdot]$. Then, the output response $\boldsymbol{y} \in \mathbb{R}^{C_{out}\times H\times W}$ is the output of convolution between the given feature map and the weight in a sliding window manner. The i,j,k-th element of the output response is formulated as follows:

$$y[i,j,k] = \sum_{c=1}^{C}\sum_{u=1}^{K}\sum_{v=1}^{K} x_c[u+j, v+k]\cdot w_c[i,u,v]. \tag{A}$$

For simplicity, we drop the index subscript in the following equations to denote $x_c[u+j,v+k]$ as x_c and $w_c[i,u,v]$ as w_c. From our proposed quantization method, convolution with floating-point values can be approximated with low-precision values, as follows:

$$y[i,j,k] \approx \sum_{c=1}^{C}\sum_{u=1}^{K}\sum_{v=1}^{K} x_c^q \cdot w_c^q \tag{B}$$

$$= \sum_{c=1}^{C}\sum_{u=1}^{K}\sum_{v=1}^{K} (\sigma_c s(n)\cdot \hat{x}_c^q + \mu_c)\cdot(\sigma_w s(n)\cdot \hat{w}_c^q) \tag{C}$$

*equal contribution

$$= \sigma_w s(n)^2 \sum_{c=1}^{C} \sigma_c \cdot \sum_{u=1}^{K}\sum_{v=1}^{K} \hat{x}_c^q \cdot \hat{w}_c^q + \sigma_w s(n) \sum_{c=1}^{C} \mu_c \sum_{u=1}^{K}\sum_{v=1}^{K} \hat{w}_c^q. \tag{D}$$

Likewise overview Figure 3 in the main paper, the channel-wise de-transformation (see Equation (D)) derived from the procedure with element-wise de-transformation (see Equation (B)) can reduce costly operations with floating-point values. Although the computation costly operation of element-wise de-transformation is largely reduced in Equation (D), it still bears computational overhead, by operating the channel-wise summation in floating-point values (due to floating-point de-transformation parameters μ_c and σ_c). To alleviate this issue, main paper presents a scheme of quantizing quantization transformation parameters of $\boldsymbol{\mu} \in \mathbb{R}^C$ and $\boldsymbol{\sigma} \in \mathbb{R}^C$, to approximate μ_c and σ_c by using the distribution statistics of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. From Equation 4, 5, 6 of the main paper, Equation (D) can be approximated as follows:

$$\approx \sigma_w s(n)^2 \sum_{c=1}^{C} \sigma_c^q \cdot \sum_{u=1}^{K}\sum_{v=1}^{K} \hat{x}_c^q \cdot \hat{w}_c^q + \sigma_w s(n) \sum_{c=1}^{C} \mu_c^q \sum_{u=1}^{K}\sum_{v=1}^{K} \hat{w}_c^q \tag{E}$$

$$= \sigma_w s(n)^2 \sum_{c=1}^{C} (\sigma_\sigma s(m) \cdot \hat{\sigma}_c^q + \mu_\sigma) \cdot \sum_{u=1}^{K}\sum_{v=1}^{K} \hat{x}_c^q \cdot \hat{w}_c^q + \sigma_w s(n) \sum_{c=1}^{C} (\sigma_\mu s(m) \cdot \hat{\mu}_c^q + \mu_\mu) \sum_{u=1}^{K}\sum_{v=1}^{K} \hat{w}_c^q \tag{F}$$

$$\begin{aligned} &= \sigma_w s(n)^2 \sigma_\sigma s(m) \sum_{c=1}^{C} \hat{\sigma}_c^q \sum_{u=1}^{K}\sum_{v=1}^{K} \hat{x}_c^q \cdot \hat{w}_c^q + \sigma_w s(n)^2 \mu_\sigma \sum_{c=1}^{C}\sum_{u=1}^{K}\sum_{v=1}^{K} \hat{x}_c^q \cdot \hat{w}_c^q \\ &\quad + \sigma_w s(n) \sigma_\mu s(m) \sum_{c=1}^{C} \hat{\mu}_c^q \sum_{u=1}^{K}\sum_{v=1}^{K} \hat{w}_c^q + \sigma_w s(n) \mu_\mu \sum_{c=1}^{C}\sum_{u=1}^{K}\sum_{v=1}^{K} \hat{w}_c^q. \end{aligned} \tag{G}$$

The floating-point channel-wise summation (see Equation (D)) is replaced with lower-precision channel-wise summation (see Equation (G)). As shown in Table A, simply changing the operation order from Equation (A) to Equation (D) reduces the BOPs largely from 174T to 10T. Furthermore, **q**uantizing **q**uantization transformation parameters (QQ) in Equation (D) results in Equation (G), which further reduces the BOPs to 3T with 4-bit for QQ.

Table A: Computational cost comparison of a 2-bit (w2a2) channel-wise quantized convolution. C=C_{out}=256, K=3, (H, W)=$(480, 270)$

De-transformation		Eqn.	Number of (n-bit, m-bit) operations								BOPs
Type	QQ		(2, 2)	(3, 3)	(6, 4)	(6, 6)	(9, 9)	(14, 32)	(17, 32)	(32, 32)	
Element-wise	✗	Eqn. (B)	-	-	-	-	-	-	-	169937M	174015G
Channel-wise	✗	Eqn. (D)	76441M	67948M	-	8493M	-	8493M	-	8493M	10108G
Channel-wise	✓	Eqn. (G)	152882M	135895M	8493M	8493M	8460M	33M	33M	66M	3046G

C. Additional experiments

C.1. Comparison with SotA methods

Existing state-of-the-art quantized SR networks [23, 41] involve a specialized architecture or an ad-hoc training scheme for low precision SR networks, mostly concentrated on binary precision. BTM [23] exploits a new training scheme like knowledge distillation and specialized gradient update rule instead of the traditional straight-through estimator [46]. BAM [41] designs a new binarized SR network, namely BSRN, utilizing a bit accumulation module.

Our proposed quantization method is orthogonal to these techniques. EDSR-BTM and BSRN-BAM are re-implemented according to the respective paper, and we replace the typical quantization (binarization) function with our distribution-aware channel-wise quantization (DAQ) function. The re-implemented architectures and the DAQ-applied architectures are respectively trained with batch size 4 and other settings same as the baseline in each paper. Despite the channel-wise overhead, our proposed method DAQ gives clear auxiliary gain in performance, for about 0.3 dB in Set5, as shown in Table B.

Table B: Comparisons on existing low-precision SR networks, EDSR-BTM and BSRN-BAM of scale 4.

Method	Precision		BOPs	Energy	Parameters	PSNR (dB)			
	w	a				Set5	Set14	B100	Urban100
EDSR-BTM [23]	1	1	23.1 T	52.8 mJ	43.1M	31.30	28.05	27.22	25.08
EDSR-BTM [23] - DAQ	1	1	75.1 T	138.9 mJ	43.1M	31.60	28.19	27.34	25.30
BSRN-BAM [41]	1	1	2.9 T	8.5 mJ	1.2M	31.17	27.94	27.15	25.01
BSRN-BAM [41] - DAQ	1	1	7.2 T	23.7 mJ	1.2M	31.44	28.03	27.21	25.05

C.2. SR Networks with batch normalization layers

In the main paper, we made a comparison with quantization methods without retraining. Among the compared methods, DFQ [37] utilizes batch normalization (BN) parameters to further improve the quantization accuracy. However, the comparison backbone of Table 3, EDSR [31] removed the BN layers for improved performance, followed by several other SR networks. For further fair comparison with DFQ, we compare the quantization methods without retraining, on EDSR *with BN*. Table C shows that our method outperforms DFQ regardless of BN layers.

Table C: Comparison of quantization methods *without retraining* on pre-trained EDSR *with BN* of scale 4.

Method	Precision		BOPs (HD image)	Energy (HD image)	PSNR (Urban100)
	w	a			
EDSR w/ BN	32	32	10025.8 T	22516.0 mJ	26.04 dB
EDSR w/ BN - LinQ	4	4	357.8 T	378.4 mJ	22.79 dB
EDSR w/ BN - DFQ	4	4	364.3 T	390.1 mJ	23.07 dB
EDSR w/ BN - DAQ	2	2	213.4 T	333.5 mJ	**24.53 dB**