

# AdaBM: On-the-Fly Adaptive Bit Mapping for Image Super-Resolution

Cheun Hong<sup>1</sup>

Kyoung Mu Lee<sup>1,2</sup>

<sup>1</sup> Dept. of ECE & ASRI, <sup>2</sup> IPAI, Seoul National University, Seoul, Korea

{cheun914, kyoungmu}@snu.ac.kr

## Abstract

Although image super-resolution (SR) problem has experienced unprecedented restoration accuracy with deep neural networks, it has yet limited versatile applications due to the substantial computational costs. Since different input images for SR face different restoration difficulties, adapting computational costs based on the input image, referred to as adaptive inference, has emerged as a promising solution to compress SR networks. Specifically, adapting the quantization bit-widths has successfully reduced the inference and memory cost without sacrificing the accuracy. However, despite the benefits of the resultant adaptive network, existing works rely on time-intensive quantization-aware training with full access to the original training pairs to learn the appropriate bit allocation policies, which limits its ubiquitous usage. To this end, we introduce the first on-the-fly adaptive quantization framework that accelerates the processing time from hours to seconds. We formulate the bit allocation problem with only two bit mapping modules: one to map the input image to the image-wise bit adaptation factor and one to obtain the layer-wise adaptation factors. These bit mappings are calibrated and fine-tuned using only a small number of calibration images. We achieve competitive performance with the previous adaptive quantization methods, while the processing time is accelerated by  $\times 2000$ . Codes are available at <https://github.com/Cheun/AdaBM>.

## 1. Introduction

Image super-resolution (SR) is a classic computer vision problem that aims to restore the high-resolution image (HR) from the corresponding low-resolution input image (LR). Since the emergence of deep neural networks, SR has been able to produce high-resolution, high-fidelity outputs. However, achieving such a level of quality was conditional upon utilizing computationally heavy SR models, necessitating demanding computing power and storage costs. Moreover, as the industry’s demand is stepping towards super-resolving larger inputs (e.g., 4K TVs), the computational

Method	Adaptive	Data	GT	BS	Iterations	Process Time
EDSR [29]	-	800	✓	16	150,000	30 hrs
PAMS [25]	✗	800	✓	16	15,000	2 hrs
DDTB [51]	✗	800	✓	16	30,000	4 hrs
PTQ4SR [39]	✗	100	✗	2	500	124 sec
CADyQ [14]	✓	800	✓	16	200,000	40 hrs
CABM [38]	✓	800	✓	16	350,000	70 hrs
AdaBM (Ours)	✓	100	✗	2	500	71 sec

Table 1. **Existing methods for quantization on SR.** Adaptive denotes whether bit-widths are adaptively allocated, GT denotes the requirement for ground-truth HR images, and BS denotes the batch size. Metrics are reported for quantizing EDSR ( $\times 4$ ).

burden has increased quadratically. Therefore, recent interest has shifted to reducing the computational costs of SR networks with little or no sacrifice on restoration accuracy. Among the surge towards lightweight SR models, quantization is one promising avenue to reduce the memory/inference cost of neural networks by replacing the 32-bit floating-point (FP) weights and activations with lower precision values.

Quantizing SR networks with minimal accuracy loss has been a challenging problem, as SR network activations exhibit predominantly variant distributions during test time, often leading to severe quantization errors. Few works address variant activations using a more fine-grained quantization function [15] or updating the quantization ranges to better fit dynamic variant activations [25, 51]. However, these methods neglect that the accuracy loss from quantization differs for different images and layers of the network. Some images are easier to reconstruct than others; in other words, they can still be accurately reconstructed with fewer computations. Thus, assigning lower bit-widths to such images leads to a better computational cost and accuracy trade-off. In this spirit, recent methods [14, 38] take into account the different quantization sensitivity of the images and adapt the bit-widths based on the input content.

To adaptively allocate bit-widths to images, existing methods [14, 38] employ several quantization function candidates of different bit-widths for each activation. Then, CADyQ [14] trains MLPs that predict the probability for each quantization function, in which the function with the

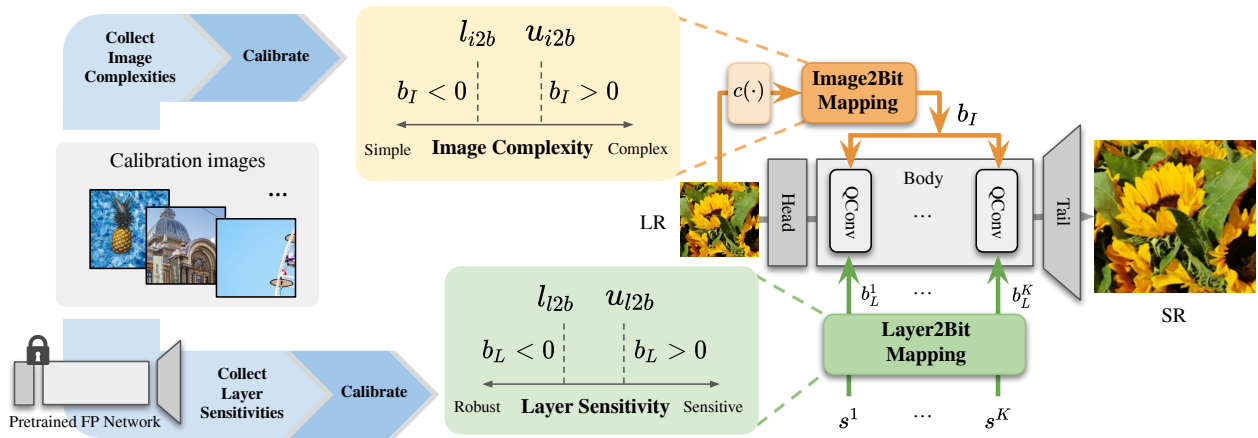


Figure 1. **Illustration of our adaptive bit-mapping.** During inference, an input image is mapped to the image-wise bit adaptation factor based on its complexity (Image2Bit Mapping), then together with the layer-wise bit adaptation factors pre-determined based on layer-wise sensitivities (Layer2Bit Mapping), the two factors adapt the quantization bit-widths. Higher bit-widths are assigned to sensitive layers and complex images. The thresholds of mapping modules are calibrated and fine-tuned using a small set of calibration images.

highest probability is selected at the test time. Additionally, CABM [38] builds a lookup table based on the trained MLPs to find a better bit-width for the input image. Although previous methods have achieved quantization on SR models with minimal accuracy loss, these approaches involve extensive quantization-aware training (QAT) with the full training dataset of LR and HR pairs. For example, as reported in Table 1, it takes 40/70 hours to obtain the final quantized model for CADyQ and CABM, respectively. In short, searching the proper bit-widths for each layer and image can provide a better accuracy-complexity trade-off, but the search cost for bit assignment is substantial.

To this end, we propose the first adaptive bit-mapping framework for image super-resolution that adapts the bit-widths for different images **on-the-fly**. Based on our observation that image-wise variance of quantization error and layer-wise variance are independent, we find that image and layer-wise adaptation can be separately processed. This dramatically reduces the search cost as only two policies are required for bit allocation: one to determine the image-wise adaptation factor for the test images and another to determine the layer-wise adaptation factors for all layers. Also, since we fix the layer-wise adaptation to be invariant of the input image, layer-wise adaptation factors can be pre-determined before test time. This allows our bit-mapping to be learned within the second level using a small subset of calibration LR images without corresponding HR.

For image-wise bit adaptation, we design an image-to-bit mapping module that maps an image to an image-wise bit adaptation factor based on the complexity of the image. The adaptation factor for an image is obtained using complexity thresholds; images that are more complex than the

upper complexity threshold are mapped to a positive adaptation factor that adjusts the bit-width to a higher bit. Then, calibration images are used to calibrate and fine-tune the complexity thresholds. Similarly, layer-wise bit adaptation factors are determined by the layer-to-bit mapping module based on each layer’s sensitivity to quantization. The sensitivity of the layer is calculated by processing the calibration images with the pre-trained FP network. The sensitive layers (larger than the upper sensitivity threshold) are mapped to a positive adaptation factor. Layer-wise adaptation factors are calibrated and directly fine-tuned using calibration images. Furthermore, to find a better quantization range for each layer-wise variant bit-width, we clip the range by optimizing the L2 distance between the quantized tensor using the assigned layer-wise bit and the FP counterpart.

In summary, we accelerate the expensive training process for adaptive quantization by calibrating the quantization parameters (ranges and bit-mappings) and then fine-tuning the quantization parameters for a few iterations with the calibration images. We achieve on-par performance with QAT-based methods with  $\times 2000$  less processing time, pushing adaptive quantization to a new frontier.

## 2. Related works

**Image super-resolution models.** Deep convolutional neural networks advanced the performance of image super-resolution (SR) problem [2, 24, 29, 34, 48]. However, these advances rely on enormous computing power and storage costs, which limits the applicability of these networks, such as deployment to mobile devices. Such limitations initiated the prevailing research for lightweight SR networks. New lightweight models are designed manu-

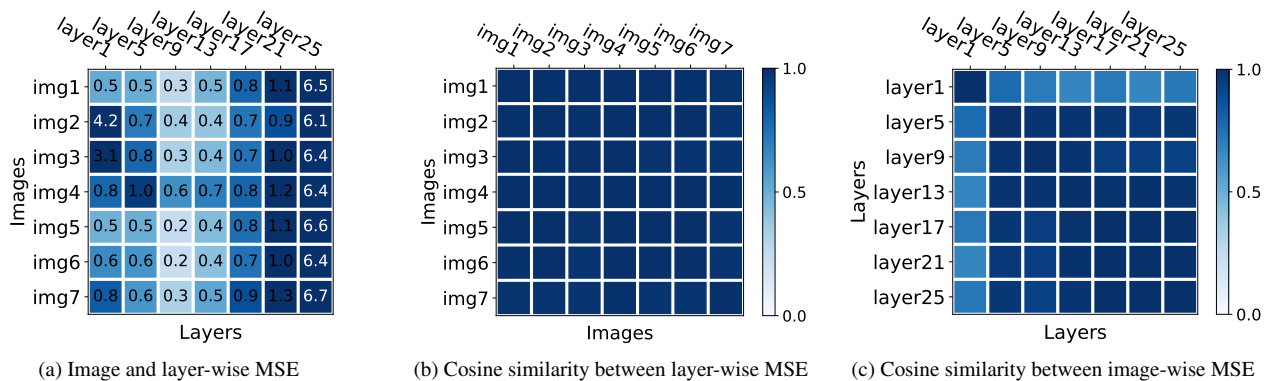


Figure 2. **Analysis on layer-wise and image-wise quantization sensitivity.** (a) Mean squared error (MSE) between statically quantized activation and corresponding FP is different per layer and per input image. (b) The *relative* orders of layer-wise MSEs tend to be invariant of the input image, as layer-wise MSEs of different images have high cosine similarity values. (c) Similarly, the *relative* orders of image-wise MSEs are consistent throughout network layers. This indicates that layer-wise bit adaptation and image-wise bit adaptation can be done separately, which effectively reduces the search cost for bit allocation.

ally [8, 17, 18, 32, 36, 45] or searched [7, 21, 27, 28]. Additionally, various compression techniques are investigated to compress existing pre-trained SR networks, such as quantization [3, 15, 25, 35, 40, 51], pruning [20, 33, 47, 50], and knowledge distillation [10, 49].

**Quantizing image super-resolution models.** Due to the benefits of cut-down memory and inference cost, quantization, which maps 32-bit floating point values of weights and activations to lower-bit values, became a promising solution for heavy SR models. Quantizing SR models with minimal accuracy degradation is challenging, since most SR networks [29, 48] have vastly distinct activation distributions. Consequently, researchers utilize learnable quantization range for different layers [13, 25, 35], adopt channel-wise quantization function [15], or design a dynamic module to adapt quantization range during test-time [51]. Although these works apply a fixed quantization level throughout the SR network, a work manually assigns different bit-width for each stage of the SR network [30]. Furthermore, recent works [14, 38] design an adaptive quantization framework with a bit-width prediction module that dynamically adapts the quantization level according to the content of the input image.

**Adaptive inference.** To make deep learning practical, allowing a pre-trained model to be responsive and adaptable to various deployment scenarios and resource constraints is crucial. Several works introduced adaptive SR networks to achieve efficient inference for a given input image [14, 23, 31, 38, 42–44, 46]. These works either adaptively reduce the network depth [43, 46] or the number of channels [23], or assign adaptive bit-widths to replace floating-point operations [14, 38]. However, although adaptive inference enables improved performance-complexity trade-off by assigning more computations to inputs more in

need, such trade-off comes at the cost of heavy re-training with the full training dataset of LR, HR pairs.

### 3. Proposed method

#### 3.1. Preliminaries

As SR networks are known to exhibit highly asymmetric activation distributions [14, 15, 38, 51], we follow the common practice of employing an asymmetric uniform quantizer for activations. The asymmetric quantizer is parameterized by the lower and upper bounds  $l, u$  and the bit-width  $b$ , in which the quantized integer is produced as follows:

$$\mathbf{X}_q = q(\mathbf{X}; b, l, u) = \lfloor \frac{\text{clip}(\mathbf{X}, l, u)}{S} - l \rfloor \cdot S + l, \quad (1)$$

where  $\text{clip}(\cdot, l, u) = \min(\max(\cdot, l), u)$ ,  $\lfloor \cdot \rfloor$  rounds the tensor to the nearest integer, and scaling factor  $S = \frac{u-l}{2^b-1}$  converts the range of the floating-point values to the range of  $b$ -bit. For weights, we employ a symmetric quantizer.

#### 3.2. Motivation

To obtain an accurate quantizer, finding a proper clipping range for activations  $[l_a, u_a]$  is important [25, 35, 51], but the bit-width  $b$  that determines the quantization level is also a crucial factor. Adopting higher bit-widths for quantization-sensitive layers and lower bit-widths for quantization-robust layers has been proven effective [5, 9, 37, 41]. Especially on SR networks, as quantization sensitivity largely varies for different layers and input images, adapting the bit-width of all quantizers based on the complexity of the input image and the sensitivity of the layer-wise input activations can provide better accuracy [14, 38]. However, finding a proper policy for such bit-width adaptation is time-consuming; for  $K$  convolutional layers and  $M$

bit-width candidates, the network has to learn  $K \cdot M$  policies to predict the bit-width. In addition, inference with  $K$  adaptive modules incurs additional inference costs.

In contrast, we observe that the layer-wise and image-wise adaptation can be processed separately, serving as a key to avoiding the extensive bit-allocation policy search. First, we observe that the *relative* order of layer-wise sensitivities tends to be consistent regardless of the input image. As shown in Figure 2, the layer-wise quantization errors of different images have high cosine similarity values, indicating that relative layer-wise sensitivities are independent of the image. Moreover, we find that the *relative* image-wise variant sensitivities of a specific layer are preserved throughout network layers. This means that layer-wise adaptation can be learned relatively among the layers and image-wise adaptation can be learned relatively within images. This simplifies the bit allocation problem to two policies: one policy for image-wise adaptation and one policy for layer-wise adaptation, which facilitates quick learning of bit-width allocation. Also, given that the layer-wise adaptation policy is invariant to the input image, it can be pre-determined before test time, thus only one adaptive module is processed during inference. In short, the static bit-width for the quantizing tensor of the  $j$ -th image of the mini-batch and the  $k$ -th convolutional layer,  $b^{j,k}$  is adapted separately by the image-wise adaptation factor  $b_I^j$  and layer-wise adaptation factor  $b_L^k$  as follows:

$$b^{j,k} = b_{base} + b_I^j + b_L^k, \quad (2)$$

The following sections describe how each factor is obtained.

### 3.3. Complexity-based image-to-bit mapping

Our goal is to determine a single universal bit adaptation factor for image-wise adaptation of the whole quantized network. This factor adaptively decreases or increases the layer-wise bit-widths based on the input image. To determine the bit-width factor for each image, image complexity can be an effective guiding metric. Image complexity can hint at quantization sensitivity; it is known that complex images (*i.e.*, larger edge density, higher spatial frequency, larger color diversity) tend to require more computational costs of reconstruction. Therefore, we design a mapping module that directly maps an image to a bit-width factor based on the complexity of the image. When the complexity of an image is larger than the complexity upper bound, it is mapped to a positive bit-width factor that globally increases the bit-widths of the network at the test time. Similarly, during inference, an image of complexity smaller than the complexity lower bound is mapped to a negative factor that globally decreases layer-wise bit-widths. The image-to-bit mapping function that maps the complexity of the  $j$ -th image of the mini-batch  $I_{LR}^j$  to bit adaptation factor  $b_I^j$  is

formulated as follows:

$$b_I^j = I2B(c(I_{LR}^j)) = \begin{cases} -1, & c(I_{LR}^j) < l_{i2b}, \\ +1, & c(I_{LR}^j) > u_{i2b}, \\ 0, & otherwise, \end{cases} \quad (3)$$

where  $c(\cdot)$  measures the complexity of the input image by calculating the average gradient density [14] of the image. We note that we simply set the bit factor to be  $\{-1, 0, 1\}$ , but it can be modified based on the hardware capacity. To determine the complexity thresholds of the image-to-bit mapping module  $l_{i2b}, u_{i2b}$ , we collect the complexity measure of the small set of calibration images. Then we use the  $p_I$ -th and  $(100 - p_I)$ -th percentiles of the complexity measures obtained to initialize  $l_{i2b}$  and  $u_{i2b}$ , respectively. Although the statistics serve as decent thresholds for allocating image-adaptive bit-width factors, we are able to obtain better sweet spots by fine-tuning the thresholds with the calibration images.

### 3.4. Sensitivity-based layer-to-bit mapping

It is commonly recognized that different layers of the network have different sensitivity to quantization [5, 9, 37, 41]. Certain layers are more robust to quantization than others, thus allocating low bit-widths to these layers can lead to a better trade-off between computational complexity and accuracy. We estimate the quantization sensitivity of each layer by calibrating the pre-trained model with the calibration images. By feeding the calibration images to the pre-trained floating-point network, we collect the standard deviation of the activations, which can be used as a metric to estimate the layer-wise quantization sensitivity [14]. Using the calibrated sensitivity of each layer, we build a layer-to-bit mapping module with sensitivity thresholds that decreases the bit-width when the layer-wise sensitivity is lower than the lower bound, and vice versa.

$$b_L^k = L2B(s^k) = \begin{cases} -1, & s^k < l_{l2b}, \\ +1, & s^k > u_{l2b}, \\ 0, & otherwise. \end{cases} \quad (4)$$

where  $s^k$  is the sensitivity of  $k$ -th convolutional layer obtained by collecting average standard deviation of activations. We use the  $p_L$ -th and  $(100 - p_L)$ -th percentiles of the obtained sensitivity measures to initialize the sensitivity thresholds  $l_{l2b}$  and  $u_{l2b}$ . Since sensitivity measures can be precalculated before test time, the layer-wise bit factor can as well be determined before test time. During inference, image-wise factor is adaptively obtained and then added to the pre-determined layer-wise factors. To enable a further better performance-complexity trade-off, we set the layer-wise bit factors as learnable parameters, which are updated using calibration images for a small number of iterations.

---

**Algorithm 1** Overall process of AdaBM

---

**Input:** Pre-trained 32-bit SR network  $\mathcal{P}$  of  $K$  layers, calibration dataset  $\mathcal{D}_{cal} = \{I_{LR}^i\}_{i=1}^N$ .

**Output:** Adaptively quantized network  $\mathcal{Q}$ .

**Initialization Phase:**

**for**  $i = 1, \dots, N$  **do**

    Measure image complexity  $c(I_{LR}^i)$

    Measure layer sensitivities  $\{s^k\}_{k=1}^K$  from  $\mathcal{P}(I_{LR}^i)$

    Given  $\{c(I_{LR}^i)\}_{i=1}^N$ , initialize I2B thresholds  $l_{i2b}, u_{i2b}$

    Given  $\{s^k\}_{k=1}^K$ , obtain L2B thresholds  $l_{l2b}, u_{l2b}$

    Initialize layer-wise bit factors  $\{b_L^k\}_{k=1}^K$  using Eq. (4)

    Initialize clipping range thresholds  $\{l_a^k, u_a^k\}_{k=1}^K$  based on layer-wise bits using Eq. (5)

**Finetuning Phase:**

**for** epoch = 1,  $\dots$ , #epochs **do**

    Update  $\{l_{i2b}, u_{i2b}\}$  and  $\{b_L^k\}_{k=1}^K$  using Eq. (12)

    Update  $\{l_a^k, u_a^k, u_w^k\}_{k=1}^K$  using Eq. (13)

---

### 3.5. Bit-aware clipping

For quantization clipping range, we first initialize the lower and upper bounds for each quantizer with the minimum and maximum statistics [19] collected by running calibration images through the FP network, using the exponential moving average (EMA). However, since bit-widths of quantizers are updated based on the input image and layer, the clipping range should also be updated. When the bit-width is decreased, the number of quantization levels are reduced, thus the optimal clipping range changes. Therefore, we adjust the clipping ranges with respect to the bit-width assigned to each layer. Inspired by OMSE [6], we adjust  $l_a, u_a$  to minimize the L2 distance between the FP tensor and the quantized tensor of the given layer-wise bit-width  $b$ .

$$\epsilon_* = \underset{\epsilon}{\operatorname{argmin}} \|\mathbf{X} - Q(\mathbf{X}; b, \epsilon \cdot l_a, \epsilon \cdot u_a)\|_2, \quad (5)$$

where  $\epsilon_*$  is searched ranging from 1.0 to 0.0 for 100 steps and the final adjusted clipping range is  $[\epsilon_* \cdot l_a, \epsilon_* \cdot u_a]$ . The process finds a better initial clipping range considering the layer-wise different bit-widths.

### 3.6. Finetuning

Calibrating the pre-trained model with the calibration images derives a decent initial point for the quantized network. Nevertheless, we fine-tune the mapping with the calibration images to obtain a better bit mapping for input image and layers. As layer-wise bit factors are pre-determined and fixed during test time, we directly fine-tune the layer-wise bit factors  $\{b_L^k\}_{k=1}^K$ . In contrast, image-wise bit factors are adapted at test time, thus we fine-tune the mapping module parameterized by threshold values  $l_{i2b}, u_{i2b}$ . However, the thresholding function of the image-to-bit mapping is not

differentiable. Accordingly, we approximate the thresholding function with  $\tanh$  [11] during back-propagation as:

$$\frac{\partial b_I^j}{\partial u_{i2b}} \approx \frac{\partial \tilde{b}_I^j}{\partial u_{i2b}} \quad \text{and} \quad \frac{\partial b_I^j}{\partial l_{i2b}} \approx \frac{\partial \tilde{b}_I^j}{\partial l_{i2b}}, \quad (6)$$

$$\text{where } \tilde{b}_I^j = \tanh(c(I_{LR}^j) - \frac{u_{i2b} + l_{i2b}}{2}). \quad (7)$$

Also, we update the clipping ranges for quantizing activations ( $l_a, u_a$ ) and weights ( $-u_w, u_w$ ). Since the quantizer includes the rounding function that is not differentiable, we employ STE [4] to approximate the rounding function as identity function during back-propagation as:

$$\frac{\partial \mathbf{X}_q}{\partial l_a} \approx \begin{cases} 1, & \mathbf{X} < l_a \\ 0, & \mathbf{X} \geq l_a \end{cases}, \quad \frac{\partial \mathbf{X}_q}{\partial u_a} \approx \begin{cases} 1, & \mathbf{X} > u_a \\ 0, & \mathbf{X} \leq u_a \end{cases}, \quad (8)$$

$$\frac{\partial \mathbf{W}_q}{\partial u_w} \approx \begin{cases} 1, & |\mathbf{W}| > u_w \\ 0, & |\mathbf{W}| \leq u_w \end{cases}. \quad (9)$$

The network weights of the SR network are frozen and only the quantization parameters are optimized using reconstruction losses from supervision of the pre-trained FP network  $\mathcal{P}$  following [39]. Reconstruction losses are given as:

$$\begin{aligned} \mathcal{L}_{pix} &= \frac{1}{N} \sum_i \|\mathcal{P}(I_{LR}^i) - \mathcal{Q}(I_{LR}^i)\|_1, \\ \mathcal{L}_{skt} &= \frac{1}{N \cdot K} \sum_i \sum_k \left\| \frac{F_{\mathcal{P}}^{i,k}}{\|F_{\mathcal{P}}^{i,k}\|_2} - \frac{F_{\mathcal{Q}}^{i,k}}{\|F_{\mathcal{Q}}^{i,k}\|_2} \right\|_2, \end{aligned} \quad (10)$$

where  $F_{\mathcal{P}}^{i,k}$  and  $F_{\mathcal{Q}}^{i,k}$  are the outputs of  $k$ -th layer of  $\mathcal{P}$  and  $\mathcal{Q}$  and  $N$  denotes the batch size. Such supervision only requires LR inputs and no ground-truth HR images, allowing us to fine-tune quantization parameters only with LR calibration images. Additionally, since we do not want overly high bit-widths to be assigned to increase the reconstruction accuracy, we regularize the bit-width to balance computational cost and accuracy.

$$\mathcal{L}_{bit} = \max\left(\frac{1}{N \cdot K} \sum_i \sum_k b^{i,k} - b_{tar}, 0\right), \quad (11)$$

where we set target bit-width  $b_{tar}$  as the static bit-width  $b_{base}$ . We use bit regularization together with the reconstruction loss to update the bit mapping parameters:

$$\mathcal{L}_{pix} + \lambda_{skt} \mathcal{L}_{skt} + \lambda_{bit} \mathcal{L}_{bit}. \quad (12)$$

For clipping ranges, we only apply the reconstruction loss:

$$\mathcal{L}_{pix} + \lambda_{skt} \mathcal{L}_{skt}, \quad (13)$$

Moreover, instead of optimizing all quantization parameters at once, we iteratively update the bit mapping parameters,

Model	QAT	GT	Process Time	W / A	Urban100		Test2K		Test4K	
					FAB	PSNR / SSIM	FAB	PSNR / SSIM	FAB	PSNR / SSIM
EDSR	-	-	-	32 / 32	32.0	26.04 / 0.784	32.0	27.71 / 0.782	32.0	28.80 / 0.814
EDSR-CADyQ	✓	✓	40 hrs	8 / 6MP	6.6	25.98 / 0.784	6.1	27.69 / 0.782	6.0	28.79 / 0.814
EDSR-CABM	✓	✓	70 hrs	8 / 6MP	5.8	25.90 / 0.782	5.6	27.65 / 0.781	5.5	28.73 / 0.812
EDSR-AdaBM (Ours)	✗	✗	<b>71 sec</b>	8 / 6MP	<b>5.7</b>	<b>25.96 / 0.782</b>	<b>5.3</b>	<b>27.65 / 0.779</b>	<b>5.2</b>	<b>28.71 / 0.809</b>
EDSR-CADyQ	✓	✓	40 hrs	4 / 4MP	4.9	25.12 / 0.753	4.9	27.43 / 0.771	4.9	28.49 / 0.803
EDSR-CABM	✓	✓	70 hrs	4 / 4MP	4.4	24.98 / 0.746	4.4	27.33 / 0.767	4.4	28.36 / 0.798
EDSR-AdaBM (Ours)	✗	✗	<b>71 sec</b>	4 / 4MP	<b>4.3</b>	<b>25.49 / 0.759</b>	<b>3.9</b>	<b>27.40 / 0.758</b>	<b>3.8</b>	<b>28.39 / 0.784</b>
SRResNet	-	-	-	32 / 32	32.0	25.86 / 0.779	32.0	27.64 / 0.781	32.0	28.72 / 0.813
SRResNet-CADyQ	✓	✓	53 hrs	8 / 6MP	6.4	25.89 / 0.779	6.2	27.65 / 0.780	6.2	28.73 / 0.812
SRResNet-CABM	✓	✓	93 hrs	8 / 6MP	5.8	25.79 / 0.776	5.6	27.60 / 0.779	5.6	28.67 / 0.811
SRResNet-AdaBM (Ours)	✗	✗	<b>92 sec</b>	8 / 6MP	<b>5.6</b>	<b>25.72 / 0.773</b>	<b>5.2</b>	<b>27.55 / 0.777</b>	<b>5.1</b>	<b>28.62 / 0.809</b>
SRResNet-CADyQ	✓	✓	53 hrs	4 / 4MP	4.1	25.39 / 0.761	4.1	27.38 / 0.771	4.1	28.46 / 0.804
SRResNet-CABM	✓	✓	93 hrs	4 / 4MP	3.8	25.42 / 0.764	3.8	27.46 / 0.774	3.6	28.52 / 0.806
SRResNet-AdaBM (Ours)	✗	✗	<b>92 sec</b>	4 / 4MP	<b>4.2</b>	<b>25.32 / 0.757</b>	<b>3.9</b>	<b>27.31 / 0.766</b>	<b>3.9</b>	<b>28.25 / 0.782</b>

Table 2. Comparisons with adaptive quantization methods on SR on EDSR and SRResNet of scale 4. Previous methods require extensive quantization-aware training (QAT) process with ground-truth (GT) HR images, which result in long process time.

clipping ranges for weights, then clipping ranges for activations. While the bit mapping parameters are updated, the clipping ranges of weights and activations are frozen. Afterwards, we freeze the bit mappings, then update the clipping ranges, such that clipping ranges are optimized for the current bit assignment policy. With such an update scheme, we achieve saturation within a small number of iterations. The overall process is summarized in Algorithm 1.

## 4. Experiments

### 4.1. Implementation details

First, we build the calibration dataset by randomly sampling 100 LR images from the DIV2K [1] training dataset. The calibration images are used to calibrate and fine-tune our bit mappings and the quantization ranges. The calibration is done for one epoch with a batch size of 16. The quantization range for activations is initialized using MinMax [19] and weights OMSE [6]. The hyperparameters for calibrating the bit mapping modules,  $p_I$  and  $p_L$  are set to 10 and 30. Then, after freezing the network weights, we fine-tune only the clipping ranges for the weights and activations and the parameters for mapping modules for 10 epochs with a batch size of 2 using the Adam optimizer [22]. The initial learning rate for the activation clipping ranges, the weight clipping ranges, the layer-wise bit factors, and the image-to-bit mapping module parameters are set as 0.01, 0.01, 0.01, and 0.1, respectively. Each learning rate is decayed by 0.9 every epoch. We balance the different loss terms with a loss weight of  $\lambda_{skt} = 10$ ,  $\lambda_{bit} = 50$ . For a fair comparison with existing adaptive quantization methods, we follow the common practice [14, 25, 38, 51] of quantizing weights and activations of the body part of the SR model. Also, we follow existing adaptive methods [14, 38] to apply adaptive quantization to activations and static quantization to weights.

### 4.2. Comparison with adaptive quantization

To demonstrate the effectiveness of our on-the-fly scheme for adaptive quantization, we compare it with existing adaptive quantization methods, CADyQ [14] and CABM [38], which we reproduce using the official codebase. The methods are evaluated on representative SR networks: EDSR [29] and SRResNet [24] of scale 4. The methods are tested on large input datasets, Urban100 [16], Test2K, and Test4k [23], in which Test2K and Test4K are produced by downsampling DIV8K [12] dataset. Following existing methods [14, 38], a large input image is split into small patches of size  $96 \times 96$ . For evaluation metrics, we measure reconstruction accuracy using the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM). To compare the computational complexity of the quantized network, we report the feature average bit-width (FAB) that is averaged throughout the images of the test dataset.

As shown in Table 2, although our method does not employ a quantization-aware training (QAT) process using the full training pairs with GT, we achieve a performance comparable to the existing methods. For example, our method achieves similar reconstruction accuracy with lower FAB on most of the settings. The results demonstrate that our method can accelerate the processing time in hours to the processing level in seconds for the first time without sacrificing the performance.

### 4.3. Comparison with static quantization

We propose the first adaptive quantization method that does not require an extensive quantization-aware training (QAT) process. To further validate the effectiveness of our adaptive approach, we compare it with existing static quantization methods that do not involve the QAT process. We compare with commonly used PTQ methods MinMax [19] and Per-

Model	FT	W / A	Set5		Set14		BSD100		Urban100	
			FAB	PSNR / SSIM	FAB	PSNR / SSIM	FAB	PSNR / SSIM	FAB	PSNR / SSIM
EDSR	-	32 / 32	32.0	32.10 / 0.894	32.0	28.58 / 0.781	32.0	27.56 / 0.736	32.0	26.04 / 0.785
EDSR-MinMax	✗	6 / 6	6.0	31.56 / 0.866	6.0	28.26 / 0.760	6.0	27.29 / 0.714	6.0	25.76 / 0.760
EDSR-Percentile	✗	6 / 6	6.0	24.30 / 0.793	6.0	24.31 / 0.728	6.0	24.68 / 0.700	6.0	21.93 / 0.696
EDSR-MinMax+FT	✓	6 / 6	6.0	31.61 / 0.870	6.0	28.31 / 0.762	6.0	27.34 / 0.718	6.0	25.81 / 0.763
EDSR-Percentile+FT	✓	6 / 6	6.0	27.23 / 0.832	6.0	25.89 / 0.747	6.0	25.82 / 0.716	6.0	23.35 / 0.723
EDSR-PTQ4SR	✓	6 / 6	6.0	31.80 / 0.884	6.0	28.34 / 0.768	6.0	27.37 / 0.722	6.0	25.79 / 0.769
EDSR-AdaBM (Ours)	✓	6 / 6MP	<b>5.7</b>	<b>31.92 / 0.887</b>	<b>5.6</b>	<b>28.47 / 0.777</b>	<b>5.4</b>	<b>27.47 / 0.731</b>	<b>5.7</b>	<b>25.89 / 0.778</b>
EDSR-MinMax	✗	4 / 4	4.0	26.83 / 0.624	4.0	25.04 / 0.546	4.0	24.57 / 0.503	4.0	23.12 / 0.536
EDSR-Percentile	✗	4 / 4	4.0	24.03 / 0.776	4.0	23.95 / 0.712	4.0	24.42 / 0.687	4.0	21.62 / 0.677
EDSR-MinMax+FT	✓	4 / 4	4.0	28.97 / 0.821	4.0	26.47 / 0.721	4.0	26.24 / 0.687	4.0	23.46 / 0.674
EDSR-Percentile+FT	✓	4 / 4	4.0	27.01 / 0.819	4.0	25.71 / 0.736	4.0	25.69 / 0.707	4.0	23.18 / 0.707
EDSR-PTQ4SR	✓	4 / 4	4.0	30.51 / 0.836	4.0	27.62 / 0.735	4.0	26.88 / 0.693	4.0	24.92 / 0.721
EDSR-AdaBM (Ours)	✓	4 / 4MP	<b>3.8</b>	<b>31.02 / 0.860</b>	<b>3.7</b>	<b>27.87 / 0.751</b>	<b>3.5</b>	<b>26.91 / 0.700</b>	<b>3.7</b>	<b>25.11 / 0.736</b>
RDN	-	32 / 32	32.0	32.24 / 0.895	32.0	28.67 / 0.784	32.0	27.63 / 0.739	32.0	26.29 / 0.793
RDN-MinMax	✗	6 / 6	6.0	30.59 / 0.863	6.0	27.54 / 0.752	6.0	26.65 / 0.703	6.0	24.79 / 0.733
RDN-Percentile	✗	6 / 6	6.0	18.87 / 0.778	6.0	18.33 / 0.667	6.0	19.88 / 0.651	6.0	16.81 / 0.632
RDN-MinMax+FT	✓	6 / 6	6.0	31.16 / 0.873	6.0	27.92 / 0.762	6.0	27.03 / 0.716	6.0	25.23 / 0.749
RDN-Percentile+FT	✓	6 / 6	6.0	21.32 / 0.812	6.0	20.74 / 0.702	6.0	21.87 / 0.677	6.0	18.67 / 0.670
RDN-PTQ4SR	✓	6 / 6	6.0	30.73 / 0.877	6.0	27.60 / 0.765	6.0	26.85 / 0.720	6.0	25.08 / 0.756
RDN-AdaBM (Ours)	✓	6 / 6MP	<b>5.7</b>	<b>31.56 / 0.881</b>	<b>5.6</b>	<b>28.14 / 0.769</b>	<b>5.5</b>	<b>27.20 / 0.722</b>	<b>5.7</b>	<b>25.31 / 0.755</b>
RDN-MinMax	✗	4 / 4	4.0	25.91 / 0.632	4.0	24.22 / 0.549	4.0	24.29 / 0.530	4.0	22.24 / 0.523
RDN-Percentile	✗	4 / 4	4.0	18.83 / 0.771	4.0	18.28 / 0.662	4.0	19.83 / 0.646	4.0	16.77 / 0.625
RDN-MinMax+FT	✓	4 / 4	4.0	28.50 / 0.810	4.0	26.15 / 0.703	4.0	26.00 / 0.673	4.0	23.35 / 0.673
RDN-Percentile+FT	✓	4 / 4	4.0	21.24 / 0.798	4.0	20.68 / 0.690	4.0	21.85 / 0.668	4.0	18.64 / 0.652
RDN-PTQ4SR	✓	4 / 4	4.0	28.32 / 0.813	4.0	26.11 / 0.709	4.0	25.82 / 0.671	4.0	23.31 / 0.668
RDN-AdaBM (Ours)	✓	4 / 4MP	<b>3.8</b>	<b>28.71 / 0.808</b>	<b>3.7</b>	<b>26.30 / 0.707</b>	<b>3.6</b>	<b>26.10 / 0.672</b>	<b>3.8</b>	<b>23.38 / 0.663</b>

Table 3. **Comparisons with static quantization without QAT on SR.** We evaluate on EDSR of scale 4 that consists of 16 residual blocks (64 channels) and RDN of scale 4.

centile [26], which calibrates the clipping parameters by using percentiles of weight/activation statistics (1% for lower bound and 99% for upper bound). However, these methods suffer severe accuracy degradation when applied directly to SR networks. Thus, we calibrate the quantized network with these methods and then fine-tune the quantization parameters using calibration images, denoted as MinMax+FT and Percentile+FT. Furthermore, we compare with a more recent approach on SR that finetunes the quantization parameters: PTQ4SR [39], which we reproduce for comparison. For a fair comparison with PTQ4SR, we follow their setting to apply static 8-bit quantization on head and tail parts, such that weights and activations of all convolutional layers are quantized. We note that unlike previous methods that employ asymmetric quantization functions for both weights and activations, we apply symmetric quantization to the weights and asymmetric quantization to the activations. We evaluate these methods on representative SR networks, EDSR [29] that consists of 16 residual blocks with 64 channel dimensions, and a more extensive SR network, RDN [48] of scale 4. As reported in Table 3, our method, AdaBM, outperforms existing methods for both 6-bit and 4-bit settings with lower FAB. This indicates that our method

achieves a better trade-off between reconstruction accuracy and computational costs. Further experiments to demonstrate the applicability of our method are provided in the supplementary document.

#### 4.4. Qualitative results

We also evaluate our framework qualitatively and compare it with the existing quantization methods without QAT. As shown in Figure 3, our method produces visually better-reconstructed images with lower average bit-width. Compared to other methods, the outputs of our method retain more details. Additionally, the adaptive allocation of our framework is visualized in Figure 4. Different bit-widths are assigned to different images and layers. Image-wise allocation results show that higher bits are assigned to complex images with more structural information and lower bits for images of fewer details (e.g., sky).

#### 4.5. Ablation study

We conduct an ablation study on each attribute of our framework to investigate the effect of the image-wise bit-mapping, layer-wise bit-mapping, bit-aware clipping, and fine-tuning with calibration images. As presented in Ta-

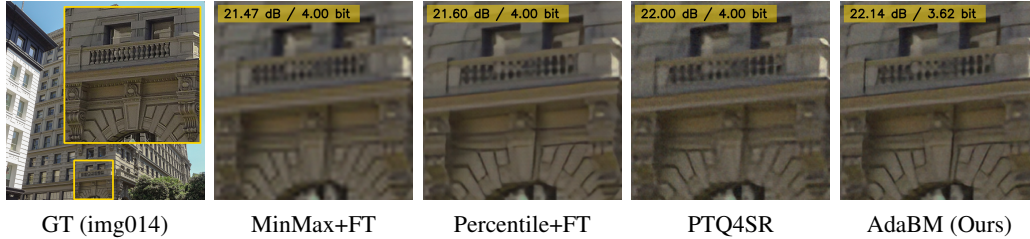


Figure 3. **Qualitative results** on Urban100 with 4-bit EDSR-based models. More results are provided in the supplementary document.



Figure 4. **Visualization of adaptive bit-mapping of AdaBM**. Results are from EDSR-AdaBM (w8 a6MP).

I2B	L2B	BaC	FT	FAB	PSNR/SSIM	FAB	PSNR/SSIM
-	-	-	-	4.0	28.70 / 0.734	4.0	24.27 / 0.628
✓	✓	✓	-	4.2	30.00 / 0.825	4.2	24.78 / 0.705
-	-	-	✓	4.0	29.02 / 0.823	4.0	23.51 / 0.677
✓	-	-	✓	4.2	29.18 / 0.804	4.1	23.80 / 0.662
-	✓	-	✓	4.0	30.47 / 0.836	4.0	24.89 / 0.714
-	-	✓	✓	4.0	30.68 / 0.837	4.0	25.12 / 0.724
✓	✓	✓	✓	<b>3.8</b>	<b>31.02 / 0.860</b>	<b>3.7</b>	<b>25.11 / 0.736</b>

Table 4. **Ablation study on each attribute of AdaBM** evaluated on Set5/Urban100 with 4-bit EDSR ( $\times 4$ ). I2B and L2B respectively denote the image-wise and layer-wise bit-mapping module. BaC refers to the bit-aware clipping and FT refers to finetuning.

le 4, using both image and layer-wise bit mapping is essential to achieve higher reconstruction accuracy with lower computational costs (FAB). Furthermore, bit-aware clipping gives an additional gain in reconstruction accuracy. Also, finetuning with the calibration images allows for a better accuracy-complexity Pareto frontier.

#### 4.6. Complexity analysis

We analyze the computational complexities and processing time of our framework compared to existing methods in Table 5. In terms of processing time, ours is over  $\times 2000$  faster than existing adaptive quantization methods. Also, the processing time is even shorter than that of several static quantization methods without QAT, showing our adaptive bit-mapping scheme has little or no processing time overhead over static quantization. Moreover, we measure the computational complexity: the model size required to store the model weights and the average bitOPs for Urban100 images. The results show that AdaBM consumes less bitOPs and model storage size compared to existing methods.

Method	Adaptive	FQ	Process Time	Model Size ( $r_{comp}$ )	BitOPs
EDSR	-	-	-	1517.6K (0.0%)	527.0T
CADyQ	✓	✗	40 hrs	489.2K (67.8%)	82.6T
CABM	✓	✗	70 hrs	485.5K (68.0%)	82.4T
AdaBM (Ours)	✓	✗	71 sec	485.5K (68.0%)	81.6T
MinMax+FT	✗	✓	75 sec	305.5K (79.9%)	9.2T
Percentile+FT	✗	✓	101 sec	305.5K (79.9%)	9.2T
PTQ4SR	✗	✓	124 sec	305.5K (79.9%)	9.2T
AdaBM (Ours)	✓	✓	76 sec	305.5K (79.9%)	9.0T

Table 5. **Complexity analysis for different quantization methods on SR**. Metrics are reported for quantizing EDSR ( $\times 4$ ). FQ denotes whether the network is fully quantized. The process time is measured on a single 2080Ti GPU.

## 5. Conclusion

In this paper, we present the first adaptive bit-mapping pipeline for image super-resolution that learns the bit-mapping policies on the fly. Our bit allocation is formulated by two policies of adapting the bit-width to a higher bit for sensitive layers and complex input images. Layers and images are directly mapped to bit-widths, in which the mapping modules are calibrated and fine-tuned using calibration images (*i.e.*, a small subset of LR images without corresponding HR images). The results demonstrate that our method achieves on-par accuracy with previous methods while the processing time is significantly reduced.

**Acknowledgment** This work was supported in part by the IITP grants [No. 2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University), No.2021-0-02068, and No.2023-0-00156], the NRF grant [No.2021M3A9E4080782] funded by the Korean government (MSIT).



## References

- [1] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *CVPR Workshops*, 2017. 6
- [2] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *ECCV*, 2018. 2
- [3] Mustafa Ayazoglu. Extremely lightweight quantization robust real-time single-image super resolution for mobile devices. In *CVPR Workshops*, 2021. 3
- [4] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 5
- [5] Weihan Chen, Peisong Wang, and Jian Cheng. Towards mixed-precision quantization of neural networks via constrained optimization. In *ICCV*, 2021. 3, 4
- [6] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *ICCV Workshops*, 2019. 5, 6
- [7] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, and Qingyuan Li. Fast, accurate and lightweight super-resolution with neural architecture search. In *ICPR*, 2021. 3
- [8] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014. 3
- [9] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *ICCV*, 2019. 3, 4
- [10] Qinquan Gao, Yan Zhao, Gen Li, and Tong Tong. Image super-resolution using knowledge distillation. In *ACCV*, 2018. 3
- [11] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *ICCV*, 2019. 5
- [12] Shuhang Gu, Andreas Lugmayr, Martin Danelljan, Manuel Fritsche, Julien Lamour, and Radu Timofte. Div8k: Diverse 8k resolution image dataset. In *ICCV Workshops*, 2019. 6
- [13] Cheeun Hong and Kyoung Mu Lee. Overcoming distribution mismatch in quantizing image super-resolution networks. *arXiv preprint arXiv:2307.13337*, 2023. 3
- [14] Cheeun Hong, Sungyong Baik, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Cadyq: Content-aware dynamic quantization for image super-resolution. In *ECCV*, 2022. 1, 3, 4, 6
- [15] Cheeun Hong, Heewon Kim, Sungyong Baik, Junghun Oh, and Kyoung Mu Lee. Daq: Channel-wise distribution-aware quantization for deep image super-resolution networks. In *WACV*, 2022. 1, 3
- [16] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015. 6
- [17] Zheng Hui, Xiumei Wang, and Xinbo Gao. Fast and accurate single image super-resolution via information distillation network. In *CVPR*, 2018. 3
- [18] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *ACMMM*, 2019. 3
- [19] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, 2018. 5, 6
- [20] Xinrui Jiang, Nannan Wang, Jingwei Xin, Xiaobo Xia, Xi Yang, and Xinbo Gao. Learning lightweight super-resolution networks with weight pruning. *Neural Networks*, 144:21–32, 2021. 3
- [21] Heewon Kim, Seokil Hong, Bohyung Han, Heesoo Myeong, and Kyoung Mu Lee. Fine-grained neural architecture search for image super-resolution. *Journal of Visual Communication and Image Representation*, 89:103654, 2022. 3
- [22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6
- [23] Xiangtao Kong, Hengyuan Zhao, Yu Qiao, and Chao Dong. Classsr: A general framework to accelerate super-resolution networks by data characteristic. In *CVPR*, 2021. 3, 6
- [24] Christian Ledig, Lucas Theis, Ferenc Huszar, 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. 2, 6
- [25] 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. 1, 3, 6
- [26] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *CVPR*, 2019. 7
- [27] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. Dhp: Differentiable meta pruning via hypernetworks. In *ECCV*, 2020. 3
- [28] Yawei Li, Wen Li, Martin Danelljan, Kai Zhang, Shuhang Gu, Luc Van Gool, and Radu Timofte. The heterogeneity hypothesis: Finding layer-wise differentiated network architectures. In *CVPR*, 2021. 3
- [29] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, 2017. 1, 2, 3, 6, 7
- [30] Jingyu Liu, Qiong Wang, Dunbo Zhang, and Li Shen. Super-resolution model quantized in multi-precision. *Electronics*, 10(17):2176, 2021. 3
- [31] Ming Liu, Zhilu Zhang, Liya Hou, Wangmeng Zuo, and Lei Zhang. Deep adaptive inference networks for single image super-resolution. In *ECCV*, 2020. 3
- [32] Yinglan Ma, Hongyu Xiong, Zhe Hu, and Lizhuang Ma. Efficient super resolution using binarized neural network. In *CVPR Workshops*, 2019. 3
- [33] Junghun Oh, Heewon Kim, Seungjun Nah, Cheeun Hong, Jonghyun Choi, and Kyoung Mu Lee. Attentive fine-grained structured sparsity for image restoration. In *CVPR*, 2022. 3

- [34] JoonKyu Park, Sanghyun Son, and Kyoung Mu Lee. Content-aware local gan for photo-realistic super-resolution. In *ICCV*, 2023. 2
- [35] Haotong Qin, Yulun Zhang, Yifu Ding, Yifan liu, Xianglong Liu, Martin Danelljan, and Fisher Yu. Quantsr: Accurate low-bit quantization for efficient image super-resolution. In *NeurIPS*, 2023. 3
- [36] Dehua Song, Yunhe Wang, Hanqing Chen, Chang Xu, Chunqing Xu, and DaCheng Tao. Addersr: Towards energy efficient image super-resolution. In *CVPR*, 2021. 3
- [37] Chen Tang, Kai Ouyang, Zhi Wang, Yifei Zhu, Wen Ji, Yaowei Wang, and Wenwu Zhu. Mixed-precision neural network quantization via learned layer-wise importance. In *ECCV*, 2022. 3, 4
- [38] Senmao Tian, Ming Lu, Jiaming Liu, Yandong Guo, Yurong Chen, and Shunli Zhang. Cabm: Content-aware bit mapping for single image super-resolution network with large input. In *CVPR*, 2023. 1, 2, 3, 6
- [39] Zhijun Tu, Jie Hu, Hanqing Chen, and Yunhe Wang. Toward accurate post-training quantization for image super resolution. In *CVPR*, 2023. 1, 5, 7
- [40] Hu Wang, Peng Chen, Bohan Zhuang, and Chunhua Shen. Fully quantized image super-resolution networks. In *ACMMM*, 2021. 3
- [41] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *CVPR*, 2019. 3, 4
- [42] Longguang Wang, Xiaoyu Dong, Yingqian Wang, Xinyi Ying, Zaiping Lin, Wei An, and Yulan Guo. Exploring sparsity in image super-resolution for efficient inference. In *CVPR*, 2021. 3
- [43] Shizun Wang, Jiaming Liu, Kaixin Chen, Xiaoqi Li, Ming Lu, and Yandong Guo. Adaptive patch exiting for scalable single image super-resolution. In *ECCV*, 2022. 3
- [44] Wenbin Xie, Dehua Song, Chang Xu, Chunqing Xu, Hui Zhang, and Yunhe Wang. Learning frequency-aware dynamic network for efficient super-resolution. In *ICCV*, 2021. 3
- [45] Jingwei Xin, Nannan Wang, Xinrui Jiang, Jie Li, Heng Huang, and Xinbo Gao. Binarized neural network for single image super resolution. In *ECCV*, 2020. 3
- [46] Ke Yu, Xintao Wang, Chao Dong, Xiaou Tang, and Chen Change Loy. Path-restore: Learning network path selection for image restoration. *IEEE TPAMI*, 2021. 3
- [47] Zheng Zhan, Yifan Gong, Pu Zhao, Geng Yuan, Wei Niu, Yushu Wu, Tianyun Zhang, Malith Jayaweera, David Kaeli, Bin Ren, et al. Achieving on-mobile real-time super-resolution with neural architecture and pruning search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4821–4831, 2021. 3
- [48] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, 2018. 2, 3, 7
- [49] Yiman Zhang, Hanqing Chen, Xinghao Chen, Yiping Deng, Chunqing Xu, and Yunhe Wang. Data-free knowledge distillation for image super-resolution. In *CVPR*, 2021. 3
- [50] Yulun Zhang, Huan Wang, Can Qin, and Yun Fu. Learning efficient image super-resolution networks via structure-regularized pruning. In *ICLR*, 2021. 3
- [51] Yunshan Zhong, Mingbao Lin, Xunchao Li, Ke Li, Yunhang Shen, Fei Chao, Yongjian Wu, and Rongrong Ji. Dynamic dual trainable bounds for ultra-low precision super-resolution networks. In *ECCV*, 2022. 1, 3, 6