IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 January 2024, accepted 1 March 2024, date of publication 7 March 2024, date of current version 14 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3375120

RESEARCH ARTICLE

Fast On-Device Learning Framework for Single-Image Super-Resolution

SEOK HEE LEE[1], KARAM PARK[1], SUNWOO CHO[1], HYUN-SEUNG LEE[2], KYUHA CHOI[2], AND NAM IK CHO[1], (Senior Member, IEEE)

[1]Department of Electrical and Computer Engineering, INMC, Seoul National University, Gwanak-gu, Seoul 08826, South Korea
[2]Samsung Electronics, Yeongtong-gu, Suwon-si, Gyeonggi-do 16677, South Korea

Corresponding author: Nam Ik Cho (nicho@snu.ac.kr)

This work was supported in part by Samsung Electronics Company Ltd., and in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by Korean Government (MSIT) (Artificial Intelligence Innovation Hub) under Grant 2021-0-02068.

ABSTRACT When implementing a super-resolution (SR) model on an edge device, it is common to train the model on a cloud using pre-determined training images. This is due to the lack of large-scale training data and computation power available on the edge device. However, such frameworks may encounter a domain gap issue because input images to these devices often have different characteristics than those used in training. Therefore, it is essential to continually update the model parameters through on-device learning, which takes into account the limited computation power of edge devices and makes use of on-site input images. In this paper, we present a fast and efficient on-device learning framework for an SR model that aims to overcome the challenges posed by restricted computation and domain gap issues. Specifically, we propose an architecture for training the SR model in a quantized domain, which helps to reduce the quantization errors that accumulate during training. Additionally, we propose cost-constrained gradient pruning and meta-learning-based fast training schemes to enhance restoration performance within a smaller number of iterations. Experimental results show that our approach can maintain the restoration performance for unseen inputs on a lightweight model achieved by our quantization scheme.

INDEX TERMS Gradient pruning, meta-learning, neural network acceleration, neural network compression, neural network quantization, on-device learning, pruning, super-resolution.

I. INTRODUCTION

The advancement of deep learning has greatly enhanced the performance of single-image super-resolution [1], [2], [3], [4], [5], [6], [7], which has also motivated consumer electronics manufacturers to implement SR models on edge devices like TVs, smartphones, tablets, etc. To install an SR model on an edge device, it is common to pre-train the model on a cloud using pre-determined training images due to the lack of large-scale training data and computation power available on the edge device.

Typically, when preparing a large number of training image pairs in the cloud, low-resolution images are generated by filtering high-resolution images with bicubic or Gaussian kernels, followed by downsampling. However, the SR model trained in this way may face a domain gap issue [8], [9] when deployed on edge devices. This is because input images on these devices often have different characteristics compared to those used in training. Specifically, real-world images are degraded by filtering with unknown kernels different from the synthetic Gaussian or bicubic. Such a domain gap problem can make it challenging to achieve the expected restoration performance. For example, Fig. 1 shows a conventional scheme, which produces blurry SR output due to the domain gap. In comparison, our method depicted in Fig. 2 yields a better output for the same input.

To overcome the domain gap, it is crucial to regularly update the model parameters by using input images on the device that has limited computational power. This process is known as on-device learning [10], [11], [12], [13], [14], [15].

The associate editor coordinating the review of this manuscript and approving it for publication was Senthil Kumar.

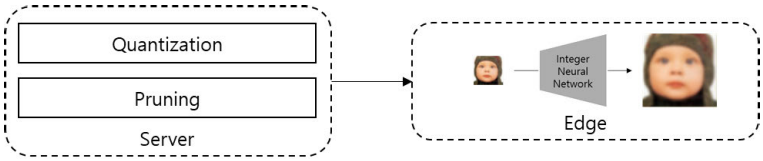


FIGURE 1. Conventional framework of super-resolution deep network.

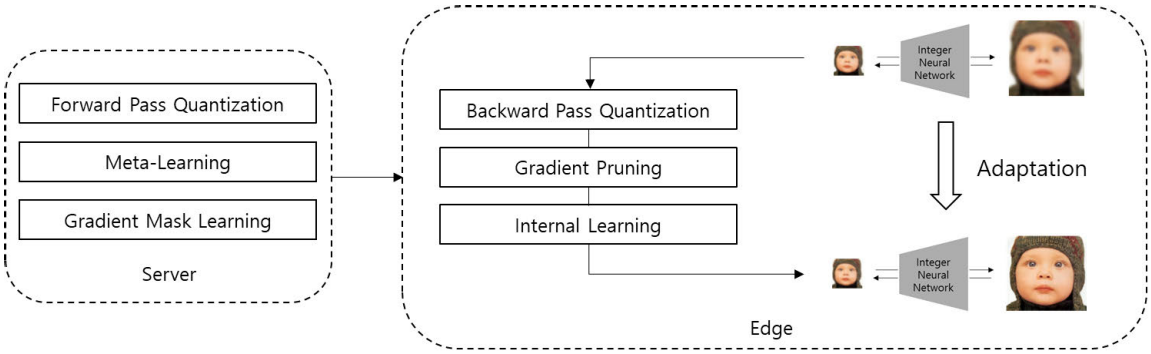


FIGURE 2. Proposed on-device super-resolution deep network.

While on-device learning can help the model adapt to the data and prevent a decrease in restoration performance, there are three main challenges that make it difficult to implement this technology in real-world scenarios.

First, edge devices are different from cloud servers in that they have limited resources. Most edge devices are constrained to use only integer operations and have much less memory and/or computing power. Although there have been efficient integer neural network models [16], [17], [18], [19], [20], [21], [22], [23], they tend to show inferior performance compared to floating-point (FP) arithmetic models due to the accumulation of quantization errors. Therefore, it is essential to reduce quantization errors occurring during training on such devices, unlike conventional methods. Second, training neural networks requires a backpropagation process, which requires three times more computation than inference and a lot of memory to store intermediate feature maps for gradient computation. Due to such high computational demands of the training process, it is challenging to train the model on an edge device. Hence, it is required to reduce the resource for the backpropagation. Third, the backpropagation is usually repeated multiple times during the training process, and thus there is a strong demand to minimize the number of iterations.

To address these issues, we propose an effective and efficient framework for on-device SR model training, as shown in Fig. 2. Precisely, we introduce a training method for the SR model in the integer domain. It is not easy to train conventional neural networks in the integer domain on resource-constrained edge devices, and most existing studies have focused on quantization for the inference of neural networks. While the existing quantization methods for the SR are suitable for inference [24], [25], [26], [27], [28], they are not suitable for training the model in an integer-only arithmetic environment. Our proposed method treats all parameters of the network and all values used for training as integers. This makes it easy to apply the model to general hardware systems that use integer-only arithmetic. Additionally, the proposed method minimizes the accumulated quantization errors during the training process compared with FP models.

Next, we propose a fast learning method that uses gradient pruning [29], [30], [31], [32], [33], which is well-suited for resource-constrained environments. By means of pruning [34], [35], [36], [37], [38], [39], we can decrease the amount of computation and memory needed for training, and concentrate resources on the weights that are important for the training process. Additionally, our gradient pruning algorithm is designed particularly for resource reduction in backpropagation.

Finally, we propose a meta-learning-based training method that utilizes input images to the device as training data [40], [41], [42], [43], [44], which helps in reducing the number of backpropagations while quickly adjusting weights to reduce the domain gap. By using the meta-learning method, we can prepare various types of degraded images to obtain the optimal starting point and quickly adapt to any quality of degraded images.

II. RELATED WORK

A. SUPER-RESOLUTION

Numerous CNN-based networks have been proposed to solve Single Image Super-Resolution (SISR) problems [1], [2], [3], [45], [46]. Initially, the SR models were trained using synthetic high-resolution (HR) and low-resolution (LR)

image pairs, where the LR images are produced by filtering HR images with bicubic or Gaussian kernels followed by downsampling [1], [2], [3], [46], [47], [48]. Recently, many methods have been developed to perform practical and realistic SR, considering that the real-world LR images are generated through unknown kernels [49], [50]. Meanwhile, the zero-shot learning method has been proposed to train an SR model by exploiting the input image itself as a training image [9]. However, it requires numerous iterations to train the network and also has limited performance due to the lack of large-scale external datasets.

B. META-LEARNING

Meta-learning is a technique that enables a machine learning system to learn from multiple training episodes and use this knowledge to learn more efficiently in future training phases [51], [52]. Model-agnostic meta-learning (MAML) is a method that utilizes learning-to-learn by creating simulations of multi-task scenarios [40], [42]. This helps to acquire task-agnostic knowledge that can be applied to new but similar tasks in test time. Reptile [41] is another approach that simplifies the optimization problem of MAML. Instead of solving the second-order derivatives, it searches for the average parameters of the models learned from multiple episodes. The concept of meta-learning has been applied in the field of SR to address the issue of domain gaps in real-world situations. Specifically, the MZSR [53] has demonstrated its superiority in handling scenarios where LR images are degraded by multiple kernels. Additionally, it provides a more efficient training phase during test time as compared to traditional methods.

C. QUANTIZATION

Quantizing deep learning models is essential for their use in resource-constrained environments that require integer-arithmetic computations [54], [55]. However, training neural networks in the integer domain is challenging due to the greater sensitivity in gradients and errors during back-propagation caused by the accumulation of quantization errors. Several studies have focused on different methods for training binary or ternary parameters [56], [57], [58], [59], while others have tried to maintain performance in an integer domain using integer-arithmetic operations [60], [61]. To address this challenge, researchers have proposed Quantization-aware training (QAT), which simulates the floating-point network to act as an 8-bit integer network during the training phase, while still maintaining the floating-point in backpropagation [16]. Another method called WAGE has been developed to train a ternary network using 8-bit integers, while the first and last layers remain as floating points [62]. Finally, a new architecture called NITI has been proposed, which is suitable for training with integer-arithmetic operations [18].

Several studies have investigated the quantization of SR networks to reduce the model size while preserving its performance [26], [63], [64], [65], [66]. In particular, the binary quantization method has been used for SR networks [63], [64], and some researchers have explored the possibility of implementing it in a partially or fully quantized domain [26], [65], [66]. However, there has been no method to train SR networks using integer-arithmetic environments with these techniques.

D. PRUNING, SPARSE LEARNING, AND ON-DEVICE LEARNING

Various pruning techniques [35], [36], [37], [38], [39], [67], [68] have been developed to reduce the number of parameters in a neural network, under the assumption that there exist redundancies in the network. The main goal is to eliminate the redundancies while achieving the best possible performance [69]. Researchers have also explored pruning methods in backpropagation to alleviate the training burden [32], [33]. For more examples of pruning techniques, Repr [70] temporarily pruned part of the parameters in rotation to boost the training performance. MeProp [71] has been suggested to prune the top-k activation gradients for each layer of the MLP in terms of their magnitudes. In addition, a study was conducted to prune the activation gradients close to zero [30]. With the advancement of hardware support in network sparsity, SDGP [29] structurally pruned the gradient by exploiting the 2:4 sparsity supported by hardware. It has been found in [72] that the distribution of gradients is close to log-normal, and they used this knowledge to set the threshold for gradient pruning.

There are other methods of reducing the training burden besides gradient pruning. Specifically, TinyTL reduces memory usage during training by only learning bias while freezing weight so that no intermediate activation is saved [12]. Experiments have even been conducted to implement the network in a limited-resource environment. In [11], they proposed skipping less important layers or sub-tensors. Also, in backpropagation, they update only bias for skipped layers but prunes gradient in the other. Besides, sparse-MAML [31] combines sparsity and meta-learning to learn which parameters to learn by learning adjustable masks.

III. OVERVIEW

The proposed method consists of two stages: offline and online training. During the offline phase, pre-training is carried out to facilitate more efficient learning in the online phase. In the online phase, limited resources and the characteristics of actual input images to the device are taken into account, and on-device training is performed in the integer domain.

The offline phase also consists of two main stages. The first is the meta-learning step, which enables rapid adaptive learning of real images. The second stage is the gradient mask learning stage, which is used for fast learning based on gradient pruning during online learning. In the online phase, training data are generated from input images to the device, which fine-tunes the model learned during the meta-learning

process. Our learning method, based on integer arithmetic and gradient pruning, is applied during the learning process, taking into account the limited conditions of the edge device.

Before proceeding with the details of our training methodology, we describe the relationship of an LR and its corresponding HR image as

$$\mathrm{I}_{LR}^{\mathrm{k}} = (\mathrm{I}_{HR} * \mathrm{k}) \downarrow_s \tag{1}$$

where I_{HR} represents a high-resolution image, I_{LR} represents a low-resolution image, k represents a degradation kernel, $*$ represents a convolution, and $\downarrow_s$ represents downsampling by scale s.

IV. OFFLINE TRAINING

A. META-LEARNING

Meta-learning is a type of training that involves simulating the training process itself. Unlike regular training or pre-training, the goal of meta-learning is to identify the best starting point for parameters. This helps the model to adapt to different data characteristics more easily.

We consider a super-resolving model parameterized by θ, denoted as $\mathcal{F}_\theta(\cdot)$. Before the actual meta-learning step, the model parameter θ is pre-trained with the large-scale DIV2K [73]. Using bicubic degradation, it is trained with the paired dataset $(\mathrm{I}_{HR}, \mathrm{I}_{LR}^{bic})$, denoted as $\mathcal{D}_{bic}$. The model is trained by the loss expressed as

$$\mathcal{L}^{\mathcal{D}_{bic}}(\theta) = \mathbb{E}_{\mathcal{D}_{bic}\sim(\mathrm{I}_{HR},\mathrm{I}_{LR}^{bic})}[\|\mathrm{I}_{HR} - \mathcal{F}_\theta(\mathrm{I}_{LR}^{bic})\|_1], \tag{2}$$

where $\mathcal{L}$ is the pixel-wise L1 loss. The loss is used as a default training loss throughout our algorithm.

Inspired by MAML [40], ZSSR [9], and MZSR [53], we follow their process to achieve meta-learned parameters. Specifically, a task $\mathcal{T}_i$ is sampled from a task distribution $p(\mathcal{T})$. The task dataset $\mathcal{D}_{meta}$ is synthesized and consists of pairs $(\mathrm{I}_{HR}, \mathrm{I}_{LR}^{\mathrm{k}})$ with diverse kernel environments. In our method, a kernel distribution $p(\mathrm{k})$ is considered, where each kernel in the distribution is an isotropic Gaussian kernel for degradation kernel with random size and covariance. $\mathcal{D}_{meta}$ is divided into two groups: $\mathcal{D}_{tr}$ for task-level training and $\mathcal{D}_{te}$ for task-level test.

In meta-learning, the model parameters θ are adapted to a new task $\mathcal{T}_i$ and become the adapted parameters θ_i after gradient descent updates. For one gradient update, the adapted parameters are described as

$$\theta_i = \theta - \alpha\nabla_\theta\mathcal{L}_{\mathcal{T}_i}^{tr}(\theta), \tag{3}$$

where α is the learning rate for task-level training. The model parameters θ are optimized for minimizing the average losses across the task-level tests with respect to θ_i. For this, the meta-objective is described as

$$\begin{aligned}&\arg\min_\theta \sum_{\mathcal{T}_i\sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{te}(\theta_i)\\ &= \arg\min_\theta \sum_{\mathcal{T}_i\sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{te}(\theta - \alpha\nabla_\theta\mathcal{L}_{\mathcal{T}_i}^{tr}(\theta)).\end{aligned} \tag{4}$$

In our paper, the stochastic gradient descent is used for the meta-optimization across tasks. The updates of the model parameters θ are described as

$$\theta \leftarrow \theta - \beta\nabla_\theta \sum_{\mathcal{T}_i\sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{te}(\theta_i), \tag{5}$$

where β is the learning rate for the meta-learning.

B. GRADIENT PRUNING

Unlike weight pruning, the goal of gradient pruning is to reduce the resources used in backpropagation while keeping the size of the parameters. The parameters of the l-th layer out of L layers are denoted as $\theta^{(l)} \in \mathbb{R}^{f_h\times f_w\times c_{in}\times c_{out}}$, where f_h and f_w are the height and the width of parameter, and c_{in} and c_{out} are the number of channel-in and channel-out respectively.

During the internal learning, the updates of $\theta^{(l)}$ with the gradient pruning are expressed as

$$\theta^{(l)} \leftarrow \theta^{(l)} - \gamma m^{(l)} g^{(l)}, \tag{6}$$

where γ is the learning rate for the internal learning, $m^{(l)} \in \{0, 1\}^{c_{out}}$ is the gradient mask, and $g^{(l)} \in \mathbb{R}^{f_h\times f_w\times c_{in}\times c_{out}}$ is the gradient of the layer l. As a result, the number of calculations can be reduced because the size of the gradient to be obtained and the size of the updated parameters are reduced.

In our method, the gradient masks are obtained during the backpropagation with the external dataset $\mathcal{D}_{gp}$, and $\mathcal{D}_{meta}$ is used as $\mathcal{D}_{gp}$. Also unlike the standard training, the model parameters θ from the meta-learning are frozen, while the gradient masks m are trained.

There may be multiple possible ways to find the masks. The proposed method sets a certain constraint on the computation or memory while maximizing the importance score $\mathcal{I}$. The s-th importance score in the layer l, denoted as $\mathcal{I}_{(s)}^{(l)}$, is defined as

$$\mathcal{I}_{(s)}^{(l)} = \sum_{j=1}^{s} \hat{\mathcal{I}}_{(j)}^{(l)}, \tag{7}$$

where $\hat{\mathcal{I}}^{(l)} \in \mathbb{R}^{c_{out}}$ is the L2-normalized gradients along the filter axis, sorted in the descending order. The objective of the proposed gradient pruning method is defined as

$$\begin{aligned}&\max_{c^{(1)},\ldots,c^{(L-1)}} \sum_{l=1}^{L-1}\sum_{s=1}^{c^{(l)}} \mathcal{I}_{(s)}^{(l)}\\ &s.t.\ \sum_{l=1}^{L-1} \mathcal{R}(\overline{c^{(l-1)}}, c^{(l)}) \le \tau\\ &c^{(l)} \in \mathcal{C}^{(l)},\end{aligned} \tag{8}$$

where $c^{(l)} = \|m^{(l)}\|_1$ is the remaining number of filters in the gradient mask, $\mathcal{R}$ denotes the cost function, τ denotes the cost costraint, $\overline{c^{(l)}}$ denotes the remaining number of filters in the mask at the current training step, and $\mathcal{C}^{(l)} = \left\{8, \ldots, 8\left\lfloor c_{out}^{(l)}/8 \right\rfloor\right\}$ is the possible number of remaining mask filters. The pruning in the last layer L is not considered

since it is essential to reconstruct the output. Then, the optimization problem in Eq. (8) belongs to a classic 0-1 knapsack problem, and it is solved by the meet-in-the-middle algorithm [34].

Consequently, the mask is trained such that the sum of the total gradient is maximized while satisfying the cost function $\mathcal{R}$. The cost function $\mathcal{R}$ is defined as the amount of calculation used for the backpropagation and expressed as

$$\mathcal{R}(\overline{c^{(l-1)}}, c^{(l)}) = c^{(l)}\overline{c^{(l-1)}} + c^{(l)}c_{out}^{(l-1)}, \tag{9}$$

where $\overline{c^{(0)}} = \overline{c^{(1)}}$. Eq. (9) considers the cost for calculating both activation gradient and weight gradient in the simplified form. Specifically, the first term is the calculation cost of the activation gradient propagating from layer l to the previous layer $l-1$, and the second term is the calculation cost of the weight gradient in layer l.

Our cost-constraint optimization formula is modified to be optimized for the gradient pruning, unlike the conventional pruning method [35], [74], which was applied only to the weight or simple optimization formula. Again, weight pruning reduces the overall size of the network in inference, while gradient pruning reduces the amount of computation and memory resources that occur during training.

V. ONLINE TRAINING

A. RAPID INTERNAL LEARNING BASED ON GRADIENT PRUNING

The zero-shot super-resolution is performed in the internal learning step. The given input image I_{LR} is downsampled with its corresponding degradation kernel to generate I_{son}. For degradation kernel, kernel estimation algorithms such as [75] can be used. Few gradient updates with gradient pruning are performed using a single pair of $(\mathrm{I}_{LR}, \mathrm{I}_{son})$. The output SR image is predicted by feeding the input image back to the trained network.

The gradients are masked by the gradient masks m and updated to the parameters θ as described in Eq. (6). Since the pruned filters of the gradient are fixed, the pruned gradient filters do not need to be calculated. Therefore, the input values used for calculating the gradient of the corresponding filter are no longer required, so they are removed. This reduces the size of the resulting gradient tensor and also reduces the size of the input tensor of the operation. In consequence, the meta-learning parameters allow large performance improvement within a few updates, and the gradient pruning reduces the computation during training.

B. INTEGER-ARITHMETIC LEARNING

In the integer-arithmetic neural network, all values previously expressed as FP are now expressed as 8-bit integer types (INT8). Inspired by [18], we express one real number tensor $X \in \mathbb{R}^n$ in two integer tensors: the value $v \in \mathbb{Z}^n$, and the scale $s \in \mathbb{Z}$, where n is the size of a flattened tensor into one

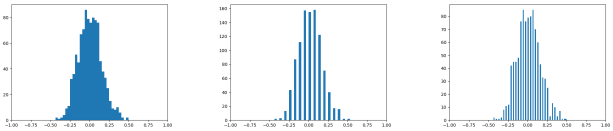


FIGURE 3. Comparison of distribution between static and dynamic quantizations.

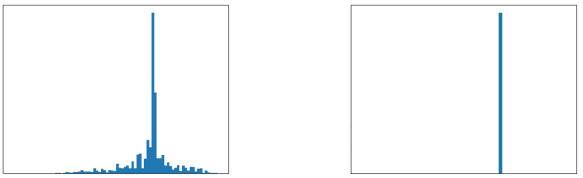


FIGURE 4. Gradient distribution collapses after scaling. Before updating the weight, the scale of the gradient has to be matched with the scale of weight. The distribution of the gradient is collapsed if the difference between the two scales is large.

dimension. The quantization function Q is defined as

$$Q(X, s) = \left\lfloor 2^7 X / 2^s \right\rceil = v, \tag{10}$$

and the dequantization function $DQ(v, s)$ is the reverse of the quanziation function.

Generally, there are two ways to choose the scale in Eq. (10): dynamic and static quantization. Dynamic quantization dynamically calculates the scale at run time. The scale s for the dynamic quantization is expressed as

$$s = \lceil log_2(max(|X|)) \rceil . \tag{11}$$

In contrast, static quantization fixes the scale s for each tensor in advance. The scales are measured by feeding the training dataset into the network in the offline stage and fixed in the online stage. In detail, each scale for a given sample is measured by Eq. (11), and the final scale is the maximum scale over the dataset. Because it uses a fixed scale, further computation for scale conversion does not take place.

As a result, the dynamic quantization allows the finer discretization based on the input distribution. The quantization errors in the dynamic quantization are reduced compared to the errors in the static quantization, as shown in Fig. 3. We suggest using dynamic quantization, which shows better restoration performance in our experiments.

The biggest difference from the existing integer neural network is that both inference and training are performed in the integer domain. Our integer model updates the integer gradients to integer weights through backpropagation. The scaling process is to match the scale of the gradient to the scale of the weight, and it needs to be done before the update. The value of scaled gradient v_{scaled} is expressed as

$$v_{scaled} = Q(DQ(v_g, s_g), s_\theta) \tag{12}$$

where v_g denotes the value of the gradient, s_g denotes the scale of the gradient, and s_θ is the scale of the weight. After

TABLE 1. The average PSNR results on Set5 [76] degraded with Gaussian blur kernels followed by direct subsampling with ×4. The models are evaluated after 10 iterations of internal learning.

Methods	Init	Prune Ratio				
		0%	27%	50%	67%	85%
Bicubic				27.41		
Pretrain ($\mathcal{D}_{bic}$)	29.26	29.76	29.76	29.74	29.66	29.47
Pretrain ($\mathcal{D}_{meta}$)	29.26	29.76	29.76	29.74	29.67	29.53
Pretrain ($\mathcal{D}_{meta}$) + 8-bit	29.13	29.64	29.64	29.62	29.56	29.39
Pretrain ($\mathcal{D}_{meta}$) + MAML	16.25	30.11	30.12	30.11	30.09	30.05
Pretrain ($\mathcal{D}_{meta}$) + MAML + 8-bit	15.89	29.89	29.87	29.85	29.84	29.79

TABLE 2. Acceleration rate of cost and memory for our proposed method.

Categories	Acceleration Rate				
	0% (FP)	27%	50%	67%	85%
Cost	× 1	× 5.14	× 6.20	× 7.14	× 7.81
Memory	× 1	× 4.43	× 5.25	× 6.08	× 7.12

the scaling process, the gradient update in integer arithmetic is modified as

$$v_\theta \leftarrow Q(v_\theta - \gamma m v_{scaled}, s_\theta) \tag{13}$$

where v_θ denotes the value of the weight, and m is the gradient mask.

As shown in Eq. (12), the scaling is a trade-off between the range of the scale s and the precision of value v. One has to narrow the range for finer precision or vice versa. However, when the difference between s_g and s_θ is large, the gradient distribution is severely damaged, as shown in Fig. 4. This phenomenon causes a problem in which the weights are not updated or converged as intended.

To solve this problem, we propose storing the weight as 32-bit integers, denoted as $\hat{\theta}$. The 32-bit weight $\hat{\theta}$ is expressed with two integer tensors: the 32-bit value $\hat{v}_\theta$ and the 8-bit scale s_θ. The 32-bit integer value of the weight $\hat{v}_\theta$ is defined as

$$Q_{32}(\theta, s_\theta) = \left\lfloor 2^{31}\theta / 2^{s_\theta} \right\rceil = \hat{v}_\theta, \tag{14}$$

where Q_{32} is the quantization function for 32-bit integer. Since $\hat{v}_\theta$ is stored at 32-bit, it allows for higher precision compared to 8-bit weight. As a result, when scaling the gradient, the distribution can be preserved without causing any damage to it. More specifically, the stored 32-bit weight is copied and converted back to 8-bit for the convolutional operations during forward propagation.

In a general integer operation, the result of the back-propagation is accumulated as 32-bit in the accumulator and quantized back to 8-bit to obtain 8-bit gradients. The proposed method preserves the 32-bit gradients $\hat{v}_g$ without quantizing back to 8-bit in the accumulator. As a result, the scaling equation in Eq. (12) and the gradient update to the 8-bit integer weight in Eq. (13) are modified as

$$\begin{aligned} \hat{v}_{scaled} &= Q_{32}(DQ_{32}(\hat{v}_g, s_g), s_\theta), \\ \hat{v}_\theta &\leftarrow Q_{32}(\hat{v}_\theta - \gamma m \hat{v}_{scaled}, s_\theta). \end{aligned} \tag{15}$$

It is worth mentioning that there is a method called quantization-aware training (QAT) [16] that is used for training a quantized network. However, this method trains networks in a simulated environment on a server, and the trained inference system is implemented on the edge devices later. This means that it is a method developed for creating an integer network that performs better in the quantized domain, but it does not take into account the domain gap problem. On the other hand, our proposed method solves the domain gap problem by utilizing on-device integer arithmetic training in actual edge environments.

Algorithm 1 Offline Training

Input: High resolution dataset $\mathcal{D}_{HR}$, degradation kernel $p(k)$, and cost constraint τ
Input: α, β : learning rates
Output: Model parameter θ and gradient mask m

1: Pretrain θ with $\mathcal{D}_{bic}$ synthesized by bicubic downsampling of $\mathcal{D}_{HR}$
2: Generate task distribution $p(\mathcal{T})$ using $\mathcal{D}_{HR}$ and $p(k)$
3: **while** *not done* **do**
4: Sample task batch $\mathcal{T}_i \sim p(\mathcal{T})$
5: **for** *all* $\mathcal{T}_i$ **do**
6: Update adapted parameters θ_i with gradient
7: descent ($\mathcal{D}_{tr}$): $\theta_i = \theta - \alpha \nabla_\theta \mathcal{L}^{tr}_{\mathcal{T}_i}(\theta)$
8: **end for**
9: Update model parameter θ with $\mathcal{D}_{te}$:
10: $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}^{te}_{\mathcal{T}_i}(\theta_i)$
11: **end while**
12: Synthesize $\mathcal{D}_{gp}$
13: **while** *not done* **do**
14: Compute importance score $\mathcal{I}$ by Eq. (7)
15: Compute possible cost using $\mathcal{R}$ in Eq. (9)
16: Find gradient mask m by solving Eq.(8) using
17: meet-in-the-middle algorithm [34]
18: **end while**

VI. ALGORITHMS

The full offline training algorithm is outlined in Algorithm 1. Lines 2-11 present the meta-learning process where the model parameter is trained by the meta-objective. Lines 12-18 are the gradient mask learning obtained by optimizing the cost-constraint gradient pruning

Algorithm 2 demonstrates the process of online training. Internal learning, gradient pruning, and integer-arithmetic learning are all combined together to perform online training.

Algorithm 2 Online Training

Input: LR test image I_{LR}, meta-learned model parameter θ, gradient mask m, the number of gradient updates n and learning rate γ

Output: Super-resolved image I_{SR}

1: Compute 32-bit model parameter $\hat{\theta}$ by quantizing θ by Eq. (14)
2: Generate LR son I_{son} by downsampling with corresponding degradation kernel
3: **for** *n steps* **do**
4: Quantize activations using Eq. (10)
5: Evaluate loss $\mathcal{L}(\hat{\theta}) = \|I_{LR} - \mathcal{F}_{\hat{\theta}}(I_{son})\|_1$
6: Update model parameter $\hat{\theta}$ using Eq. (15)
7: **end for**
8: **return** $I_{SR} = \mathcal{F}_{\hat{\theta}}(I_{LR})$

As a result, the final super-resolved image I_{SR} is obtained after a few gradient updates n.

VII. EXPERIMENTS

A. TRAINING DETAILS

We adopt an 8-layer CNN architecture with residual learning, which is the same network used in ZSSR [9] and MZSR [53]. However, all biases in the architecture are removed, which makes the number of parameters 225K. DIV2K [73] is used as the training dataset, and we set the extracted image patches to 64. We employ ADAM optimizer [77] to meta-train the network and employ stochastic gradient descent (SGD) optimizer for the rest. Also, the bicubic method [45], [50] is used for subsampling. Our method is evaluated by measuring the Y channel of YCbCr colorspace with famous super-resolution benchmarks: Set5 [76], BSD100 [78], and Urban100 [79]. For the experimental setting for online training, the ground-truth degradation kernel is used to estimate the degradation kernel of the input image. However, existing off-the-shelf kernel prediction algorithms such as DCLS [75] can be used in actual usage. All of our experiments are conducted in Pytorch environment [80].

B. RESULTS

To show the efficacy of the proposed method, the average PSNR is measured on Set5 [76] degraded with Gaussian blur kernels and downsampled by $\times 4$. The result is shown in Table 1. As a result, the pre-trained model trained with the degraded dataset $\mathcal{D}_{meta}$ shows slightly better performance than the model trained with the bicubic dataset $\mathcal{D}_{bic}$. Obviously, the degraded dataset is close to the test dataset compared to the bicubic dataset, so it is essential to choose the training dataset close to the test environment if it is possible.

Next, the initial PSNR of the pre-trained model is 29.26dB, which is higher than bicubic interpolation (27.41dB) and our proposed method's initial point (16.25dB). After the internal learning, the PSNR of our proposed method increases to 30.11 dB. The results show that meta-learning successfully allows the models to quickly adapt to the degradation of the input image.

TABLE 3. Average PSNR results of static and dynamic quantization applied to the different networks on Set5 with $\times 4$.

PSNR (dB)	Bicubic	FP	8-bit	8-bit(Ours)
Static	28.42	30.83	30.29	30.66
Dynamic			30.63	30.75

Our quantization method is applied to the pre-trained model and our proposed model. The pre-trained model yields 29.64dB, which is decreased by 0.12dB compared to the FP model, and our proposed model shows 29.89dB, which is decreased by 0.22dB. As a result, both quantized models are successfully trained in the integer domain. Therefore, our method reduces the quantization errors and prevents the models from being collapsed during integer training.

Computational costs and memory usage in internal learning are calculated to evaluate the effectiveness of our gradient pruning method applied to the models. The calculation of the memory used for backpropagation is theoretically acquired by

$$Memory = \sum_l (h^{(l)} w^{(l)} \tilde{c}_{out}^{(l)} bit_{\mathcal{F}} + \tilde{c}_{out}^{(l)} \tilde{c}_{int}^{(l)} bit_{\theta} + h^{(l)} w^{(l)} c_{in}^{(l)} bit_{\mathcal{F}})/8, \quad (16)$$

where h and w are the height and width of the intermediate feature maps, respectively, $\tilde{c}_{in}$ and $\tilde{c}_{out}$ are the number of in and out-channels after pruning, respectively, and $bit_{\mathcal{F}}$ and bit_{θ} are the bit of the network without the weight and the weight, respectively. For the experiment, the height and the width are set to 32, and the original network is calculated with all bits set to 32. The cost of backpropagation is calculated by

$$Cost = \sum_l bit_{\mathcal{F}} \{h^{(l)} w^{(l)} c_{in}^{(l)} (2\tilde{c}_{out}^{(l)} - 1) + \tilde{c}_{in}^{(l)} \tilde{c}_{out}^{(l)} (2h^{(l)} w^{(l)} - 1)\}, \quad (17)$$

which is similar to the calculation of multiply-accumulate (MAC).

The effectiveness of our gradient pruning method is evaluated as shown in Table 1 and Table 2. At 27% and 50% pruning ratio, the PSNRs are decreased by 0.04dB at most compared to the unpruned models. In addition, the cost is accelerated by $\times 6.20$ for the pruning ratio of 50%, and the memory is reduced by $\times 5.25$ compared to the FP model. In an extreme setting, our proposed model in our 8-bit environment at a pruning ratio of 85% shows 29.79dB, and it is 0.1dB lower than the unpruned model. Consequently, our gradient pruning shows that it is able to accelerate the cost and the memory usage during internal learning while preventing the domain gap problem even in an integer-arithmetic environment.

The visualizations in Fig. 5 match with the former result. The resulting images of the pre-trained model show that

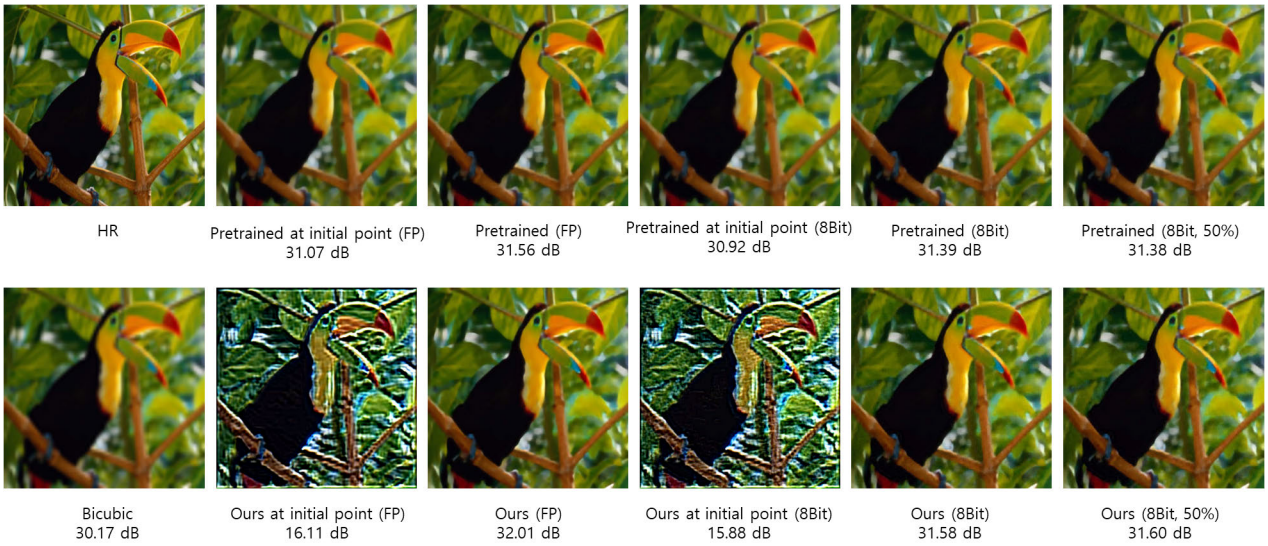


FIGURE 5. Visualization of the initial point and after 10 iterations of each model. Top row images are trained with $\mathcal{D}_{bic}$, and bottom row images are trained with our offline and online training process.

TABLE 4. Average PSNR results on integer-arithmetic super-resolution model trained on $\mathcal{D}_{bic}$ with ×4 and without meta-learning and online learning.

Dataset	Bicubic	FP	8-bit	8-bit(Ours)
Set5	28.42	30.83	30.63	30.75
BSD100	25.95	27.04	26.95	27.00
Urban100	23.14	24.82	24.72	24.79

TABLE 5. Comparison of gradient memory during backpropagation between FP model, standard 8-bit model, and our proposed integer model. The size of height and width is fixed to 32.

Methods	Gradient Memory (KB)	
Original	19,371	(× 1.00)
8-bit	4,840	(× 4.00)
8-bit(Ours)	5,517	(× 3.51)

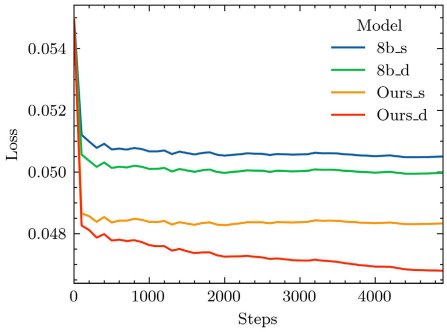


FIGURE 6. Comparison of training loss between the standard 8-bit and our 8-bit method with static or dynamic quantization.

the restoration performance improves after internal learning. However, our proposed method shows better restoration performance in the end after internal learning, even if the initial images have artifacts. In addition, the images in our 8-bit environment improve after internal learning in both the pre-trained and ours, which means that our integer-arithmetic learning successfully reduced the quantization errors.

VIII. DISCUSSIONS

A. INTEGER LEARNING

Depending on quantization methods, the distribution is modified differently, and dynamic quantization can effectively prevent degradation caused by quantization errors compared to static quantization. Fig. 6 compares training losses between static and dynamic quantization on a standard 8-bit network and our 8-bit network from scratch using Set5 as a train set. The loss indicates that dynamic versions converge better than static versions, which also means that dynamic quantization has smaller quantization errors compared to static quantization. Table 3 shows the restoration performances after training, and both results indicate that PSNR with dynamic quantization is higher than that with static quantization, as expected. In particular, the dynamic version of the standard integer network has 0.34dB higher PSNR than that of the static version. When it is applied to our integer network, the PSNR is 30.75dB, which is very close to that of FP version 30.83dB. Thus, dynamic quantization is suitable for handling quantization errors during training by keeping the highest precision at all times, and it is used as a default setting.

Again, our proposed 8-bit network is compared to the standard 8-bit network trained from scratch with Set5 in the aspect of the training loss. When the 32-bit weight is applied to the 8-bit network, the training loss converges more effectively, as shown in Fig. 6. Our 8-bit network with dynamic quantization converges the best, whereas others have already saturate in the earlier step.

Our proposed network trained from scratch is evaluated on various SR datasets in a bicubic downsampling scenario,

TABLE 6. Comparison of restoration performance between our gradient pruning model and layer-wise model. K % of the front layers are pruned for the layer-wise model so that it performs similarly to fine-tuning. PSNR is measured after 10 iterations of internal learning.

Methods	Bic	Init	PSNR / Prune Ratio				
Layer-wise	27.41	29.26	29.76 / 0%	29.75 / 17%	29.49 / 50%	29.31 / 66%	29.27 / 83%
Proposed			29.76 / 0%	29.76 / 27%	29.74 / 50%	29.66 / 67%	29.47 / 85%

TABLE 7. Comparison of acceleration rate between our gradient pruning model and layer-wise model in terms of cost and memory.

Methods	Categories	Acceleration Rate				
Layer-wise	Prune ratio	0%	17%	50%	66%	83%
	Cost	× 1	× 1.20	× 1.49	× 1.71	× 1.99
	Memory	× 1	× 1.17	× 1.42	× 1.58	× 1.79
Proposed	Prune ratio	0%	27%	50%	67%	85%
	Cost	× 1	× 1.26	× 1.52	× 1.79	× 1.95
	Memory	× 1	× 1.16	× 1.37	× 1.56	× 1.79

as shown in Table 4. In all cases, our network shows better performance than the standard 8-bit network. Furthermore, its PNSR is close to that of the FP version. For Set5, the FP model shows 30.83dB while the 32-bit weight network yields 30.75dB, which is very close to the original version. Therefore, applying 32-bit weight and dynamic quantization can be an effective method from the viewpoint of reducing degradation due to quantization errors for the SR tasks.

Memory usage for the network with 32-bit weight and with the standard 8-bit is calculated, as shown in Table 5. The 8-bit network reduces memory usage by 4, while the 32-bit weight network reduces by 3.51 compared to the original. Using the 32-bit weight allows the recovery of image restoration performance at the cost of small additional memory usage.

B. GRADIENT PRUNING

Our method is compared to layer-wise pruning to test the feasibility. For layer-wise pruning, $K\%$ of the front layers are pruned such that the last layers are trained for fine-tuning. The results in Table 6 show that our method has better restoration performance in all pruning ratios. In particular, the PSNR of the layer-wise model at 50% pruning ratio is 29.49dB, whereas our method is 0.25dB higher. In this observation, our gradient pruning successfully distinguishes between the important and the redundant gradient filter from the optimization.

Computational cost and memory usage are calculated to test the reduction in resource usage during training. The result is shown in Table 7. The cost and the memory of the layer-wise model and our proposed method show the close acceleration ratio in a similar pruning ratio. At 50% pruning ratio, the cost and the memory of the layer-wise are $\times 1.49$ and $\times 1.42$, respectively, whereas those of the proposed method are $\times 1.52$ and $\times 1.37$. The cost of the proposed method at 50% is more accelerated compared to the layer-wise method, and their performances in the memory show the opposite aspects. The aspect is due to our cost-constraint gradient pruning algorithm, which allocates the sparsity of the mask between layers for better restoration performance at specific cost constraints.

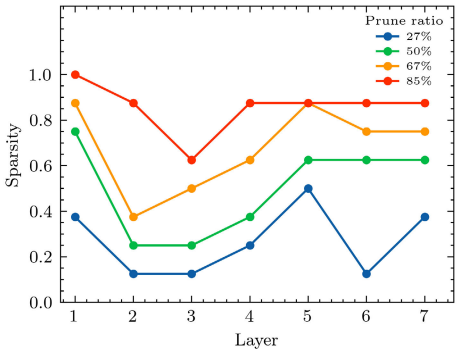


FIGURE 7. Sparsity investigation of the different pruned models according to the depth of the layer.

The sparsity of each layer in our gradient pruning method is investigated, as shown in Fig 7. Obviously, the models of higher pruning ratios have more sparse masks overall. Particularly, the first layers of the models have a sparse mask compared to the masks in the other layers. Our cost-constraint gradient pruning method penalizes the computation happening at the earlier layers because the pruned masks in the earlier layers generally reduce more resources than the later layers. The second and third layers show less sparsity compared to the other layers, which could be a further research topic.

C. META-LEARNED INITIAL POINT

We visualized the result at the initial point and after 10 gradient updates between the network trained with D_{bic} and the network meta-learned with D_{meta} in Fig. 5. The result of the network trained with D_{bic} at the initial point shows decent quality compared to that of the meta-learned network, and the performance slightly improves after the gradient updates. Obviously, it is targeted for super-resolving images with bicubic degradation at the inference, and it does not consider adapting and super-resolving images with unknown degradation.

On the other hand, the image at the initial point of the meta-learned network is distorted to a degree. To explain the case, the meta-learned initial point is where the network can easily adapt to various degradations in a few gradient updates, and it does not aim to super-resolve the given image without adaptation. Accordingly, it quickly adapts to the given image with unknown degradation after gradient updates. Also, the report in [53] showed similar results for the initial point, which can be seen as the characteristics of MZSR [53].

IX. CONCLUSION

In this paper, we have proposed an efficient, lightweight, on-device learning framework for super-resolution neural networks. We achieved this by combining meta-learning, gradient pruning, and integer-arithmetic learning, to overcome the domain gap problem. Our proposed method uses 32-bit integer weight instead of 8-bit weight, which helps to overcome the distribution collapses due to the scaling process during updating weights. Also, we use dynamic quantization to reduce quantization errors during training. We use cost-constraint gradient pruning to accelerate the training process and reduce resource usage, while optimizing the restoration performance under a certain cost. Lastly, we combine meta-learning to adapt to the degradation of the given input images. In future work, we plan to implement the proposed system into hardware that is ready to be implemented into an edge device.

REFERENCES

[1] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1646–1654.

[2] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 105–114.

[3] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1132–1140.

[4] L. Wang, X. Dong, Y. Wang, L. Liu, W. An, and Y. Guo, "Learnable lookup table for neural network quantization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2022, pp. 12423–12433.

[5] Z. Lu, J. Li, H. Liu, C. Huang, L. Zhang, and T. Zeng, "Transformer for single image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 456–465.

[6] B. Niu, "Single image super-resolution via a holistic attention network," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Aug. 2020, pp. 191–207.

[7] X. Wang, L. Xie, C. Dong, and Y. Shan, "Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 1905–1914.

[8] H. Nam, H. Lee, J. Park, W. Yoon, and D. Yoo, "Reducing domain gap by reducing style bias," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8686–8695.

[9] A. Shocher, N. Cohen, and M. Irani, "Zero-shot super-resolution using deep internal learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3118–3126.

[10] S. Dhar, J. Guo, J. Liu, S. Tripathi, U. Kurup, and M. Shah, "A survey of on-device machine learning: An algorithms and learning theory perspective," *ACM Trans. Internet Things*, vol. 2, no. 3, pp. 1–49, 2021.

[11] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, and S. Han, "On-device training under 256kb memory," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 22941–22954.

[12] H. Cai, C. Gan, L. Zhu, and S. Han, "TinyTL: Reduce memory, not parameters for efficient on-device learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1–22.

[13] H.-Y. Chiang, N. Frumkin, F. Liang, and D. Marculescu, "MobileTL: On-device transfer learning with inverted residual blocks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, 2023, pp. 7166–7174.

[14] L. Yang, A. S. Rakin, and D. Fan, "RepNet: Efficient on-device learning via feature reprogramming," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12267–12276.

[15] Y. Yang, G. Li, and R. Marculescu, "Efficient on-device training via gradient filtering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 3811–3820.

[16] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.

[17] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–8.

[18] M. Wang, S. Rasoulinezhad, P. H. W. Leong, and H. K.-H. So, "NITI: Training integer neural networks using integer-only arithmetic," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 11, pp. 3249–3261, Nov. 2022.

[19] Z. Li and Q. Gu, "I-ViT: Integer-only quantization for efficient vision transformer inference," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 17065–17075.

[20] H. Lin, J. Lou, L. Xiong, and C. Shahabi, "Integer-arithmetic-only certified robustness for quantized neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 7808–7817.

[21] Y. Yang, L. Deng, S. Wu, T. Yan, Y. Xie, and G. Li, "Training high-performance and large-scale deep neural networks with full 8-bit integers," *Neural Netw.*, vol. 125, pp. 70–82, May 2020.

[22] S. Kim, A. Gholami, Z. Yao, M. W. Mahoney, and K. Keutzer, "I-BERT: Integer-only BERT quantization," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2021, pp. 5506–5518.

[23] A. Ghaffari, M. S. Tahaei, M. Tayaranian, M. Asgharian, and V. P. Nia, "Is integer arithmetic enough for deep learning training?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 27402–27413.

[24] Z. Tu, J. Hu, H. Chen, and Y. Wang, "Toward accurate post-training quantization for image super resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 5856–5865.

[25] G. Berger, M. Dhingra, A. Mercier, Y. Savani, S. Panchal, and F. Porikli, "QuickSRNet: Plain single-image super-resolution architecture for faster inference on mobile platforms," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2023, pp. 2187–2196.

[26] H. Wang, P. Chen, B. Zhuang, and C. Shen, "Fully quantized image super-resolution networks," in *Proc. 29th ACM Int. Conf. Multimedia*, Oct. 2021, pp. 639–647.

[27] M. Ayazoglu, "Extremely lightweight quantization robust real-time single-image super resolution for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 2472–2479.

[28] H. Li, "PAMS: Quantized super-resolution via parameterized max scale," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 564–580.

[29] B. McDanel, H. Dinh, and J. Magallanes, "Accelerating DNN training with structured data gradient pruning," in *Proc. 26th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2022, pp. 2293–2299.

[30] X. Ye, "Accelerating CNN training by pruning activation gradients," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Aug. 2020, pp. 322–338.

[31] J. Von Oswald, D. Zhao, S. Kobayashi, S. Schug, M. Caccia, N. Zucchet, and J. Sacramento, "Learning where to learn: Gradient sparsity in meta and continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 5250–5263.

[32] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 440–445.

[33] Z. Zhang, P. Yang, X. Ren, Q. Su, and X. Sun, "Memorized sparse backpropagation," *Neurocomputing*, vol. 415, pp. 397–407, Nov. 2020.

[34] R. Humble, M. Shen, J. A. Latorre, E. Darve, and J. Alvarez, "Soft masking for cost-constrained channel pruning," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2022, pp. 641–657.

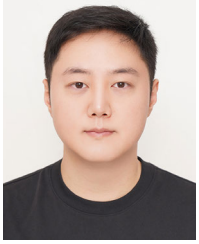
[35] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1398–1406.

[36] M. Shen, P. Molchanov, H. Yin, and J. M. Alvarez, "When to prune? A policy towards early structural pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12237–12246.

[37] Y. Li, K. Adamczewski, W. Li, S. Gu, R. Timofte, and L. Van Gool, "Revisiting random channel pruning for neural network compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 191–201.

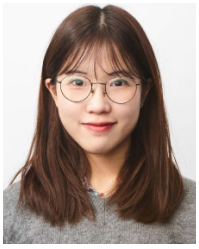
[38] S. Yu, Z. Yao, A. Gholami, Z. Dong, S. Kim, M. W. Mahoney, and K. Keutzer, "Hessian-aware pruning and optimal neural implant," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2022, pp. 3665–3676.

[39] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen, "Rigging the lottery: Making all tickets winners," in *Proc. Int. Conf. Mach. Learn.*, Vienna, Austria, Nov. 2020, pp. 2943–2952.
[40] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.
[41] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," 2018, *arXiv:1803.02999*.
[42] A. Antoniou, H. Edwards, and A. Storkey, "How to train your MAML," 2018, *arXiv:1810.09502*.
[43] M. Abbas, Q. Xiao, L. Chen, P.-Y. Chen, and T. Chen, "Sharp-MAML: Sharpness-aware model-agnostic meta learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 10–32.
[44] Z. Chi, L. Gu, H. Liu, Y. Wang, Y. Yu, and J. Tang, "MetaFSCIL: A meta-learning approach for few-shot class incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 14146–14155.
[45] K. Zhang, W. Zuo, and L. Zhang, "Learning a single convolutional super-resolution network for multiple degradations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3262–3271.
[46] Z. Li, J. Yang, Z. Liu, X. Yang, G. Jeon, and W. Wu, "Feedback network for image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3862–3871.
[47] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, accurate, and lightweight super-resolution with cascading residual network," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 252–268.
[48] J. W. Soh, G. Y. Park, J. Jo, and N. I. Cho, "Natural and realistic single image super-resolution with explicit natural manifold discrimination," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8114–8123.
[49] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin, "Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 814–81409.
[50] J. Gu, H. Lu, W. Zuo, and C. Dong, "Blind super-resolution with iterative kernel correction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1604–1613.
[51] S. Thrun and L. Pratt, "Learning to learn: Introduction and overview," in *Learning to Learn*. Cham, Switzerland: Springer, 1998, pp. 3–17.
[52] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5149–5169, Sep. 2022.
[53] J. W. Soh, S. Cho, and N. I. Cho, "Meta-transfer learning for zero-shot super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3513–3522.
[54] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 126–136, Jan. 2018.
[55] J. Choi, Z. Wang, S. Venkataramani, P. I-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: Parameterized clipping activation for quantized neural networks," 2018, *arXiv:1805.06085*.
[56] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–9.
[57] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 525–542.
[58] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016, *arXiv:1602.02830*.
[59] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.
[60] R. Banner, I. Hubara, E. Hoffer, and D. Soudry, "Scalable methods for 8-bit training of neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–7.
[61] D. Das, N. Mellempudi, D. Mudigere, D. Kalamkar, S. Avancha, K. Banerjee, S. Sridharan, K. Vaidyanathan, B. Kaul, and E. Georganas, "Mixed precision training of convolutional neural networks using integer operations," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–11.
[62] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14.
[63] J. Xin, N. Wang, X. Jiang, J. Li, H. Huang, and X. Gao, "Binarized neural network for single image super resolution," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 91–107.
[64] Y. Ma, H. Xiong, Z. Hu, and L. Ma, "Efficient super resolution using binarized neural network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 694–703.
[65] C. Hong, H. Kim, S. Baik, J. Oh, and K. M. Lee, "DAQ: Channel-wise distribution-aware quantization for deep image super-resolution networks," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2022, pp. 913–922.
[66] C. Hong, S. Baik, H. Kim, S. Nah, and K. M. Lee, "CADyQ: Content-aware dynamic quantization for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2022, pp. 367–383.
[67] Y. Guo, H. Yuan, J. Tan, Z. Wang, S. Yang, and J. Liu, "GDP: Stabilized neural network pruning via gates with differentiable polarization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5219–5230.
[68] H. Mostafa and X. Wang, "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 4646–4655.
[69] Z. Wang, C. Li, and X. Wang, "Convolutional neural network pruning with structural redundancy reduction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14908–14917.
[70] A. Prakash, J. Storer, D. Florencio, and C. Zhang, "RePr: Improved training of convolutional filters," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10658–10667.
[71] X. Sun, X. Ren, S. Ma, and H. Wang, "meProp: Sparsified back propagation for accelerated deep learning with reduced overfitting," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3299–3308.
[72] B. Chmiel, L. Ben-Uri, M. Shkolnik, E. Hoffer, R. Banner, and D. Soudry, "Neural gradients are near-lognormal: Improved quantized and sparse training," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–9.
[73] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1122–1131.
[74] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–15.
[75] Z. Luo, H. Huang, L. Yu, Y. Li, H. Fan, and S. Liu, "Deep constrained least squares for blind image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 17621–17631.
[76] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L.-A. Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proc. Brit. Mach. Vis. Conf.*, 2012, p. 135.
[77] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds. San Diego, CA, USA, May 2015, pp. 1–21.
[78] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th IEEE Int. Conf. Comput. Vis. (ICCV)*, 2001, pp. 416–423.
[79] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5197–5206.
[80] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1–16.



SEOK HEE LEE received the B.S. and M.S. degrees in electrical engineering from Southern Methodist University, Dallas, TX, USA, in 2015 and 2016, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Seoul National University, Seoul, South Korea. His research interests include image processing, neural network compression, computer vision, and deep learning.

KARAM PARK received the B.S. degree from Seoul National University, Seoul, South Korea, in 2018, where he is currently pursuing the Ph.D. degree. His research interests include image processing, image restoration, and computer vision.



SUNWOO CHO received the B.S. degree from Seoul National University, Seoul, South Korea, in 2019, where she is currently pursuing the Ph.D. degree. Her research interests include image processing, super-resolution, meta-learning, and computer vision.

HYUN-SEUNG LEE received the B.S. and M.S. degrees in electronic engineering from Sogang University, in 2005 and 2007, respectively. He is currently pursuing the Ph.D. degree with Seoul National University. He is working with the Visual Display Division, Hardware Platform Laboratory, Samsung Electronics Company Ltd. His research interests include the restoration of images and videos.

KYUHA CHOI received the B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2006, and the Ph.D. degree in electrical engineering and computer science from Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2013. He is currently with the Visual Display Division, Samsung Electronics Company, Suwon-si, South Korea. His research interests include super-resolution, de-noising, and IR enhancement.

NAM IK CHO (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in control and instrumentation engineering from Seoul National University, Seoul, South Korea, in 1986, 1988, and 1992, respectively. From 1991 to 1993, he was a Research Associate with the Engineering Research Center for Advanced Control and Instrumentation, Seoul National University. From 1994 to 1998, he was an Assistant Professor of electrical engineering with the University of Seoul. In 1999, he joined the Department of Electrical and Computer Engineering, Seoul National University, where he is currently a Professor. His research interests include image processing, adaptive filtering, digital filter design, and computer vision.