IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

IEEE CONSUMER TECHNOLOGY SOCIETY SECTION

Received 22 November 2022, accepted 21 December 2022, date of publication 26 December 2022, date of current version 18 January 2023.

Digital Object Identifier 10.1109/ACCESS.2022.3232258

RESEARCH ARTICLE

Skip-Concatenated Image Super-Resolution Network for Mobile Devices

GANZORIG GANKHUYAG[1], JINGANG HUH[1], MYEONGKYUN KIM[1], KIHWAN YOON[1], HYEONCHEOL MOON[1], SEUNGHO LEE[1], JINWOO JEONG[1], SUNGJEI KIM[1], AND YOONSIK CHOE[2], (Life Senior Member, IEEE)

[1]Korea Electronics Technology Institutes (KETI), Seongnam-si 13509, South Korea
[2]Department of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea

Corresponding authors: Sungjei Kim (sungjei.kim@keti.re.kr) and Yoonsik Choe (yschoe@yonsei.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant Funded by Korea Government [Ministry of Science and ICT (MSIT)] (Development of Intelligent Media Aspect Ratio Conversion Technology That Maintains Properties) under Grant 2021-0-00802; and in part by the Ministry of Culture, Sports, and Tourism and Korea Creative Content Agency, under Project R2020040238.

ABSTRACT Single-image super-resolution technology has been widely studied in various applications to improve the quality and resolution of degraded images acquired from noise-sensitive low-resolution sensors. As most studies on single-image super-resolution focused on the development of deep learning networks operating on high-performance GPUs, this study proposed an efficient and lightweight super-resolution network that enables real-time performance on mobile devices. To replace the relatively slow element-wise addition layer on mobile devices, we introduced a skip connection layer by directly concatenating a low-resolution input image with an intermediate feature map. In addition, we introduced weighted clipping to reduce the quantization errors commonly encountered during float-to-int8 model conversion. Moreover, a reparameterization method was selectively applied without increasing the cost in terms of inference time and number of parameters. Based on the contributions, the proposed network has been recognized as the best solution in Mobile AI & AIM 2022 Real-Time Single-Image Super-Resolution Challenge with PSNR of 30.03 dB and NPU runtime of 19.20 ms.

INDEX TERMS Concatenation, mobile super resolution, quantization-aware training, single image super resolution, TFLite.

I. INTRODUCTION

The single-image super-resolution (SISR) technology transforms a low-resolution (LR) image into a high-resolution (HR) image that provides higher pixel density and more textual information than the LR image. In general, super-resolution is utilized in computer vision applications such as remote sensing [1], [2], underwater applications [3], [4], medical image processing [5], [6], and multimedia applications [7]. However, when transforming LR to HR images, the super-resolution is known as an ill-posed problem and multiple types of HR images exist. To predict a suitable corresponding HR image, traditional methods such as interpolation-based [8] and representation-based [9], [10] methods have been proposed.

In recent years, deep learning methods have advanced rapidly, and deep learning-based SISR methods have been studied to achieve state-of-the-art performance, for example, SRCNN [11], a method based on convolutional neural networks (CNN), has resulted in significant improvements. Several novel ideas and methods have been introduced, such as various types of deep learning network architectures [12], [13], [14], [15], loss functions [16], [17], training strategies and techniques [18], [19], and attention network [20], [21].

However, the majority of the superior SISR methods have focused on the reconstruction quality of the HR images from LR images utilizing expensive high-performance GPUs. To enhance the quality of reconstruction, they proposed various techniques and networks that evolved into the SISR

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

model with an extensive number of parameters and high computational complexity [22]. However, the complex network structure, various types of deep learning techniques, and numerous parameters create challenges in deploying traditional SISR methods in mobile device environments. Owing to the limited computing resources and hardware of mobile devices compared with desktop or cloud resources, the SISR model requires a lightweight network and hardware-friendly deep learning techniques such as quantization with INT8.

To perform the SISR task on mobile devices with deployment requirements, we propose a skip-concatenated image super-resolution network (SCSRN) that can transform LR images to HR images with substantial accuracy and real-time inference speed. Concisely, the major proposed contributions are stated as follows:

1) We propose a highly efficient super-resolution network (SCSRN) that can deliver higher accuracy at faster speed compared to previous mobile SR models. Notably, we excluded the element-wise addition operation that is a labor-intensive task on mobile devices, and instead, introduced a lighter skip-concatenated layer that can avoid memory replications to equalize the input dimensions.
2) This study proposes a quantization error robust training method. With skip connection, the distribution of the kernel weights tends to become asymmetric during training. However, the asymmetric distribution causes serious degradation of image quality in quantization as a mobile device supports only symmetric quantization for kernels. To this end, the valid range of the weights were constrained during training.
3) We selectively applied a reparameterized convolution (RepConv) layer to improve the image quality, while maintaining the model size and inference speed. Interestingly, based on the experiments, the application of the RepConv layer to all layers in SCSRN could compromise the reconstructed image quality.
4) We decompose a clipped rectified linear unit (ReLU), which was originally introduced to prevent the incorrect output overflow and underflow in the inference, into min(x) and ReLU operations and merge the ReLU into the last convolution layer. Before merging, the ReLU takes the latency up to 2.5 ms in mobile devices, but we successfully removed latency of ReLU in the inference time.

The remainder of the paper is organized as follows. In Section 2, we discuss the related works of super-resolution. The proposed method is described in Section 3. The effectiveness of our SCSRN model is validated in Section 4. Finally, the conclusions of this study are summarized in Section 5.

II. RELATED WORK

A. SINGLE IMAGE SUPER-RESOLUTION METHODS

The SISR methods can be classified into traditional SISR and data-driven deep learning methods. Traditional SISR methods have been proposed as interpolation-based [8] and representation-based methods [9]. In principle, the interpolation-based method considers the relationship of neighboring pixels, whereas the representation-based method reconstructs the HR image by deriving a mapping function between the LR and HR cropped patches. However, both methods exhibit limitations in reconstructing detailed features and patterns. The recently proposed CNN-based method delivered excellent performance by solving several problems that cannot be resolved using traditional methods.

A CNN-based SISR model is trained to target HR from a given LR. With the advent of CNN-based SISR networks such as SRCNN [11], this pipeline produced remarkable performance in the SISR task. First, the CNN-based SR model stacked more deep layers to improve the performance, but this assessment caused a gradient-vanishing problem and exhibited a limitation in terms of the image quality. Thereafter, very deep super-resolution [23] and deeply-recursive convolutional [24] networks employed deeply stacked residual blocks to resolve this issue. In addition, enhanced deep super-resolution network (EDSR) [18] demonstrated batch-normalization (BN), which exhibited remarkable performance in classifying tasks and normalizing the features of the SISR model with degraded performance. To improve the stability of training without BN layers, EDSR uses the residual-scaling method. Consequently, EDSR achieved a state-of-the-art result by enhancing the feature representation of the model.

Recently, residual channel attention network [20] and storage area network [25] significantly improved the performance by adopting channel attention mechanism. However, this attention mechanism requires considerable memory during inference owing to its spatial and nonlocal operation. Moreover, it offers limited application in low-power devices such as mobile or IoT devices.

B. LIGHTWEIGHT SUPER-RESOLUTION METHODS FOR MOBILE

Although CNN-based SISR models significantly improve the restored image quality, these advancements have increased the extent of computation and memory required in the inference stage. Furthermore, the demand for such applications in low-power devices (e.g., mobile and IoT) has increased in various computer-vision tasks. To satisfy this demand, research is currently being conducted to reduce the computational complexity and design efficient network structures for the SISR network.

Research on lightweight super-resolution network for mobile devices can be classified into network optimization for enhancing the performance in the same network structure and the design of hardware-friendly architecture for reducing the inference time. The representative methods of network optimization include pruning, quantization, and knowledge distillation.

Pruning is categorized into filter pruning (a.k.a structured pruning) [26] and weight-level pruning [27] (a.k.a

unstructured pruning). Unstructured pruning does not save inference time, whereas structured pruning reduces the complexity of inference. Filter pruning for SR network [26], [28] constitutes a promising approach for achieving a reasonable trade-off between performance and complexity by eliminating the filters that do not influence the network performance.

The representative quantization methods include mixed precision [29] and quantization-aware training (QAT) [30]. Mixed precision training can improve the performance by searching the optimized bits per layer. In contrast, QAT is a fake quantization in the network training process, which simulates an 8-bit integer training process with clamping and approximation by employing fake quantization during training. Thus, the complete model training is still performed with the original precision, but the training and inference processes are simulated with 8-bit quantization. This approach minimizes performance degradation during deployment by reducing the quantization error in the quantization options of the model format.

Knowledge distillation is a method of transferring knowledge from the teacher model to a lightweight structure of the student networks [31]. In principle, knowledge distillation can be segmented into feature distillation [32], [33] and image domain distillation [34]. For instance, feature distillation trains a feature map in a student network to resemble that in the teacher network. In contrast, the representation distillation trains the output of a student network (HR image in case of SISR) to resemble that of the teacher network.

Finally, the approaches to design a hardware-friendly network structure seek appropriate structures based on the profiling results derived during the inference process [35], [36], [37]. Thus, it is a process of searching a structure that can be compromised in terms of performance and speed. Fig. 1 shows the mobile-friendly network structures. The ABPN [35] consists of seven 3×3 convolution layers and utilizes the element-wise add operation between channel-wise duplicated LR images and output feature maps. The NCNET [36] introduces a nearest convolution layer, which is operated as the nearest interpolation of the LR image, instead of the channel-wise duplication. XLSR [37] uses channel split blocks (GBlock) and concatenation operations.

In this paper, we propose an efficient approach to design a hardware-friendly network (e.g., reparameterized block and removing the element-wise operation) and develop an appropriate training strategy for it, such as robust training of quantization errors.

III. THE PROPOSED METHOD

The proposed method is detailed in this section. First, we illustrate the proposed network architecture, SCSRN, and explain its novelty with respect to the anchor-based plain net(ABPN) [35] that served as our inspiration. Subsequently, we describe the reparameterized block (RepConv block) that is an over-parameterized strategy employed to improve network performance. Third, we introduce the weight-constrained QAT method to minimize the quantization error. Lastly, the speed-up method and training strategy are described for the SCSRN model.

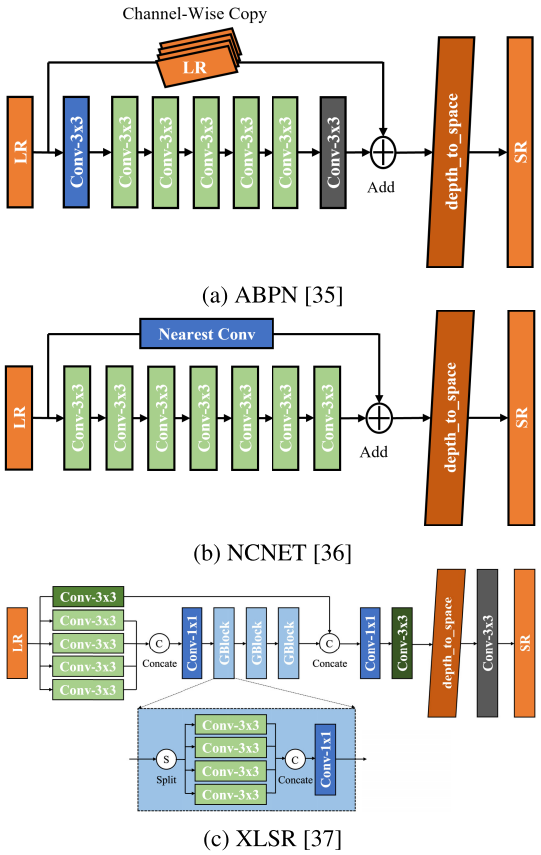


FIGURE 1. Mobile device-friendly network structures.

A. NETWORK ARCHITECTURE OF SCSRN

The overall structure of the SCSRN is illustrated in Fig 2, which is an ABPN-inspired structure comprising four components. The first component is a feature extraction layer that extracts the features from an LR image. The second component is a backbone comprising four RepConv blocks to learn deeper features. The third component includes two transition layers for the residual learning effect after directly concatenating the feature maps and LR. The final component (depth_to_space) involves the pixel re-arrangement for restoring the HR image.

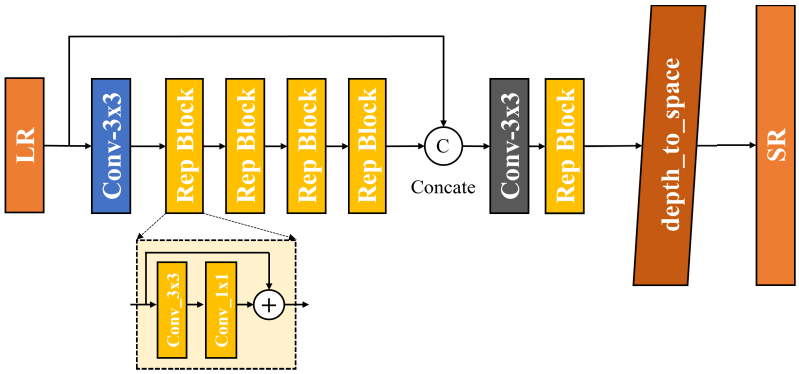
For deeper comprehension, let I_{LR} and I_{HR} denote the input and output of the network. We obtain the features F_0 as follows:

$$F_0 = H_{FE}(I_{LR}), \tag{1}$$

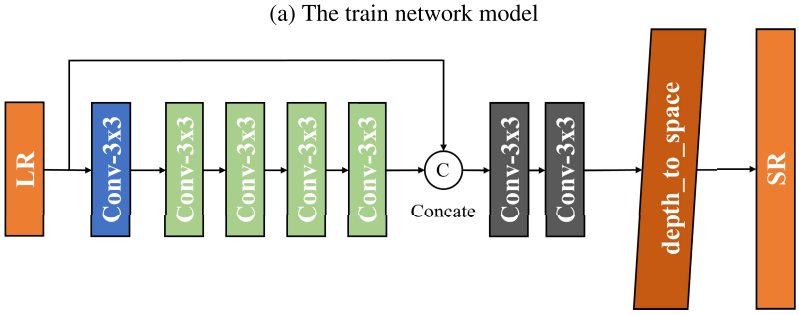
where $H_{FE}(\cdot)$ denotes the function that extracts features from an image. Subsequently, we obtained the i-th features F_i by

$$F_i = H_{BB_i}(F_{i-1}), \quad i = 1, \ldots 4, \tag{2}$$

where H_{BB_i} represents the function for the i-th deep feature, which contains high frequency and texture information.



(a) The train network model



(b) The inference network model

FIGURE 2. Network structure of skip-concatenated image super-resolution network.

Thereafter, we concatenated the feature and I_{LR} with the channel axis, expecting a residual effect to pass through the two transition layers for obtaining F_{HR} as follows:

$$F_{HR} = H_{TR}(H_{TR}(concat(F_4, I_{LR}))), \tag{3}$$

where H_{TR} denotes the transition layer. Using the pixel rearrangement function H_{RA}, we derived I_{HR} from F_{HR} and clipped all the pixels between 0 and 255. Thereafter, the ReLU function was applied in the tail layer of the network to constrain the lower bound (0) of the pixel value, and the *min* function was utilized to maintain the upper bound value (255).

$$I_{HR} = H_{RA}(min(F_{HR}, 255)) \tag{4}$$

Unlike the ABPN [35] and NCNET [36], the input LR image and feature maps were concatenated directly in the middle of the network, and the last two transition and depth_to_space layers provide a smooth transformation from the concatenated features to the SR image. The concatenation operation helps to reduce the quantization error because the input LR image containing 8-bit pixel values is not corrupted by the INT8 quantization. Moreover, the ABPN duplicates the LR image on multiple instances to obtain two equal input dimensions for the element-wise addition operation. In contrast, the skip-concatenated operation saves inference time by omitting the multiple memory-copy operations in the ABPN.

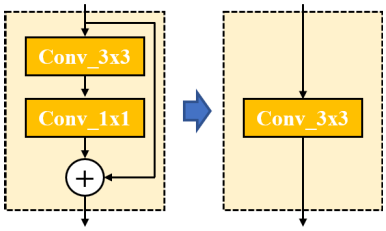


FIGURE 3. RepConv Block. (Left) train mode model and (Right) inference mode model.

B. RepConv BLOCK

In the inference stage, we applied the reparameterization method to improve the reconstructed image quality without any architectural variations. According to [38] and as depicted in Fig. 3, the reparameterization can be reconstructed if it maintains the linearity property, even if the convolution layer overlaps in various manner. To express this mathematically, we applied it as follows:

In Fig. 3-(Left), let the weight and bias of Conv_3 $\times$ 3 be $W1$ and $b1$, those of Conv_1 $\times$ 1 be $W2$ and $b2$, and the input and output are x and y. Therefore, it can be expressed as Eq. (5).

$$y = (W_2^T W_1^T + I)x + (W_2^T b_1 + b_2), \tag{5}$$

where I represents identity matrix caused by the addition operator. Similarly, Fig. 3-(Right) presents Eq.

$$y = W_3^T x + b_3 \tag{6}$$

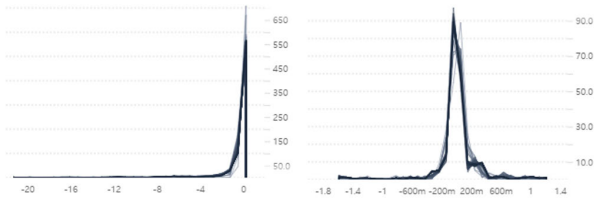


FIGURE 4. Distribution of weight: (Left) asymmetric and (Right) symmetric.

Therefore, we can see Fig. 3-(Left) becomes Fig. 3-(Right) through Eq. (7) and (8). Considering Eq. (7) and (8), the left-hand side learns high-level information in the training step and can be simplified in the inference step according to the above-mentioned operation.

$$(W_2^T W_1^T + I) = W_3^T \tag{7}$$

$$(W_2^T b_1 + b_2) = b_3 \tag{8}$$

We experimented by applying the RepConv to each convolution layer, which confirmed the most advantageous method of application as that displayed in Fig. 2. In addition, the "Xavier normalization" is an appropriate weight initialization method, and the weight initializing setting can be further improved. To reduce the quantization error in the QAT stage, we converted the RepConv train form into a simplified inference form (convolution 3×3 layer) after the fine-tuning stage. The results obtained with the application of RepConv are described in the experimental results section.

C. WEIGHT-CONSTRAINED QUANTIZATION AWARE TRAINING

To execute deep learning models on mobile devices with low memory or computational power, TensorFlow [39] supports two types of model optimization methods, namely, post-training quantization (PTQ) and QAT. As both techniques perform quantization, performance degradation is inevitable. In particular, PTQ performs quantization after completing training, and the operation method is converted from float32 to float16 or int8. In contrast, QAT performs quantization during the training step. Generally, the application of PTQ after QAT yields less performance loss compared to the direct application of the PTQ to the model. Therefore, we applied the *training* (*scratch*) $\rightarrow$ *QAT* $\rightarrow$ *PTQ* quantization method in sequence.

Despite the application of the mentioned procedure, the quantization error was larger than expected, because the weights of the first convolution layer exhibited an asymmetric distribution, as portrayed in Fig. 4-(Left). In the 8-bit quantization specification of Tensorflow Lite [40], the symmetric quantization for weights was allowed only because the distribution of the weights was assumed to be symmetric. Thus, the accumulation of quantization errors in the first layer degraded the overall performance. To mitigate the performance degradation, the mean value of the distribution was constrained to zero using the weight clipping technique in the range $[-2, 2]$, which altered the distribution after training, as depicted in Fig. 4-(Right). Notably, the range of weight clipping was set as $[-3, 3]$ for RepConv Block and $[-2, 2]$ for others.

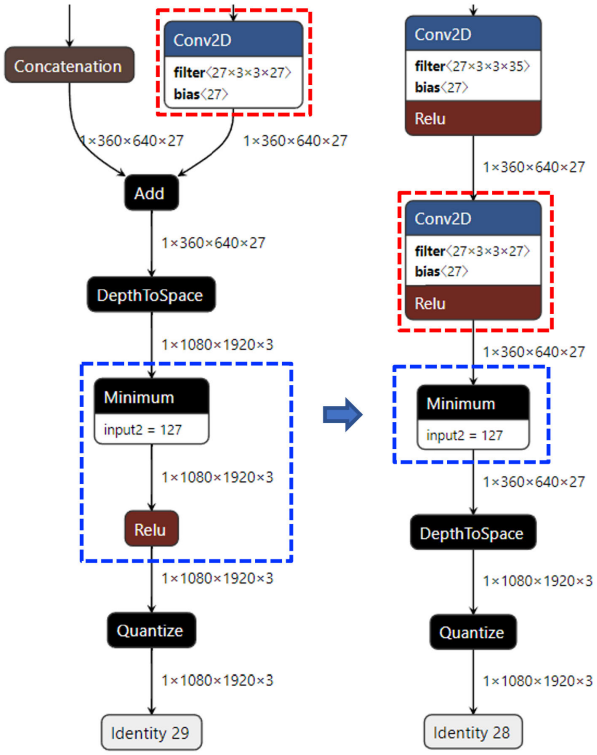


FIGURE 5. Network structure after quantization by Netron tool. (Left) ABPN and (Right) SCSRN.

D. SPEED UP METHOD

To further improve the speed, the network was analyzed using an external tool called Netron [41]. A visualization of ABPN [35] and SCSRN (Right) is presented in Fig. 5. As depicted in Fig. 5, ABPN [35] typically omits the ReLU function in the terminating convolution layer of the super-resolution network and uses a Clipped ReLU function to avoid the normalization of an incorrect output [42]. The clipping operation is performed after pixel rearrangement (i.e., DepthToSpace operation of Tensorflow) in Fig. 5-(Left).

Although the latency of the combined convolution with ReLU was less than 1 ms, that of the single ReLU operation was up to 3∼4ms based on the layer-wise profiling. This is potentially caused by the hardware (HW) architecture in which Convolution-BatchNormalization-ReLU modules are designed and operated in a single HW unit, and ReLU is not singularly implemented in mobile devices.

$$\begin{aligned} ClippedReLU(F_{HR}) &= ReLU(min(F_{HR}, 255)) \\ &= min(ReLU(F_{HR}), 255) \end{aligned} \tag{9}$$

Based on Eq. (9), Clipped ReLU function can be expressed in various forms. Herein, the ReLU function was merged with the terminating convolution layer and the minimum

TABLE 1. Ablation study. (Red indicates best values within DIV2K val dataset.)

				RepConv								
ConvBase	Add	Concate	WC=[-2,2]	WC=[-2,2]	WC=[-3,3]	DCT loss	MinClip	Scratch	Fine Tuning	QAT	TFLite (int8)	Inference (ms)
√								30.23	30.24	30.03	30.03	32.8
√	√							30.24	30.25	30.06	30.06	50.5
√		√						30.32	30.33	30.24	30.19	
√		√	√					30.32	30.33	30.23	30.22	
√		√	√	√				30.31	30.32	30.22	30.22	43.3
√		√	√		√			30.32	30.33	30.23	30.23	
√		√	√		√	√		30.32	30.33	30.24	30.25	
√		√	√		√	√	√	30.32	30.33	30.24	30.25	40.8

operation was excluded to maintain the upper bound (255). The DepthToSpace forms the pixel rearrangement operation and does not harm the merging of convolution and ReLU functions. Consequently, the inference time can be saved with no visual degradation.

E. TRAINING STRATEGY

Only the training dataset of DIV2K was used in the training process. We trained our model in three steps including the scratch training step, fine-tuning step with different loss function, and QAT step.

1) SCRATCH TRAIN STEP

In the first step, our model was trained from scratch. The LR patches were cropped from LR images with 128 × 128 size and 16 mini-batch sizes. The Adam optimizer was used with a 0.001 learning rate during scratch training. The cosine warm-up scheduler was used with a 0.1 percentage warm-up ratio. The total number of epochs was set to 800. We use $l1$ loss is expressed in Eq. (10).

$$L^{l1}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left|f(I_{lr}^{i}) - I_{hr}^{i}\right|, \tag{10}$$

where θ represents the trainable parameters of the proposed network, and n denotes the number of training patched images. I_{lr}^{i} and I_{hr}^{i} indicate the LR patch images and corresponding HR patch images. $f(\cdot)$ denotes the function of the proposed work.

2) FINE-TUNING STEP

In the second step, the model was initialized with the weights trained in the first step. To improve the accuracy, we used $l2$ loss as expressed in Eq. (11). Fine tuning with $l2$ loss improves the peak signal-to-noise ratio (PSNR) value by 0.01 $\sim$ 0.02 dB. In this step, the initial learning rate was set as 0.00002, and the Adam optimizer was used along with a step scheduler (i.e., learning rate halved at every 40 epochs). The total epoch was set to 200 epochs. Moreover, we applied a channel shuffle augmentation.

$$L^{l2}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(f(I_{lr}^{i}) - I_{hr}^{i}\right)^{2} \tag{11}$$

3) QUANTIZATION-AWARE TRAINING STEP

In the third stage, the same training setting was used as step two with the exception that the QAT model was initialized with weights trained from the second step and the total training epoch was set to 300. In addition, the learning rate was set as 0.00001. Furthermore, a discrete cosine transform (DCT) domain $l1$ loss function was applied between the ground truth HR and predicted HR images, expressed in Eq. (12).

$$L^{l1}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left|DCT(f(I_{lr}^{i})) - DCT(I_{hr}^{i})\right|, \tag{12}$$

where $DCT(\cdot)$ represents the DCT domain transformation operation.

IV. SIMULATION RESULTS

The simulation setup and the results obtained with the proposed model are described herein. In particular, we elaborate the performance improvement step applying the concatenation method, weight clipping, and RepConv. Moreover, we demonstrate the reduction in inference time by excluding the Clipped ReLU. Thereafter, the proposed model was compared with previous studies, wherein FSRCNN [12], XLSR [37], SESR [38], ABPN [35], and NCNET [36] were tested on five standard datasets. Eventually, we compared the proposed model with the previous studies on the devices for scale 3 on Samsung Galaxy Z Fold4 with Snapdragon 8+ Gen 1 and Galaxy Note20 with Snapdragon 865+.

To ensure a justified comparison, all experiments were conducted in the same experimental environment. The training process was executed using RTX A6000 GPUs. As discussed earlier, we used TensorFlow 2.5.0 version for the all three training steps and the TFlite generation step. In particular, the DIV2K train dataset was used for training (i.e., 800 images of DIV2K).

TABLE 2. Quantitative results on benchmark datasets. (Red indicates best PSNR/SSIM values within each dataset. All PSNR/SSIM values are measured with RGB channel.)

Scale	Algorithm	Set5		Set14		B100		Urban100		DIV2K	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
		FP32/INT8	FP32/INT8	FP32/INT8	FP32/INT8	FP32/INT8	FP32/INT8	FP32/INT8	FP32/INT8	FP32/INT8	FP32/INT8
X2	Bicubic	32.07	0.9208	28.06	0.8541	26.08	0.79	24.94	0.8285	31.27	0.9057
	FSRCNN [12]	34.06/28.95	0.9401/0.7770	29.93/27.05	0.8888/0.7520	29.68/26.81	0.8856/0.7453	27.20/25.20	0.8788/0.7470	32.78/28.07	0.9271/0.7477
	XLSR [37]	35.15/29.67	0.9474/0.7912	30.66/27.61	0.8971/0.7596	30.33/27.24	0.8950/0.7455	28.80/26.28	0.9062/0.7717	33.76/28.92	0.9364/0.7588
	SESR [38]	34.52/34.51	0.9431/0.9426	30.29/30.28	0.8989/0.8915	29.98/29.98	0.8898/0.8893	27.85/27.90	0.8929/0.8923	33.14/33.14	0.9352/0.9299
	ABPN [35]	34.90/34.79	0.9457/0.9441	30.56/30.56	0.8960/0.8946	30.36/30.30	0.8952/0.8936	28.90/28.84	0.9074/0.9058	33.71/33.59	0.9360/0.9342
	NCNET [36]	34.93/34.26	0.9458/0.9332	30.60/30.34	0.8962/0.8857	30.40/30.10	0.8959/0.8855	28.95/28.66	0.9082/0.8969	33.74/33.14	0.9363/0.9226
	SCSRN(our)	35.23/35.10	0.9478/0.9442	30.77/30.74	0.8986/0.8953	30.42/30.33	0.8964/0.8931	29.07/28.91	0.9095/0.9044	33.92/33.68	0.9377/0.9329
X3	Bicubic	27.02	0.82	24.1	0.7169	25.58	0.732	22.06	0.7029	28.4	0.8321
	FSRCNN [12]	30.24/27.05	0.8852/0.7592	26.89/25.06	0.8000/0.7008	26.78/24.91	0.7825/0.6775	24.13/22.91	0.7779/0.6791	29.39/26.18	0.8577/0.7181
	XLSR [37]	31.35/30.29	0.9038/0.8596	27.53/27.01	0.8150/0.7737	27.30/26.65	0.7957/0.7533	25.22/24.75	0.8149/0.7731	30.13/28.84	0.8721/0.8161
	SESR [38]	31.24/31.20	0.9017/0.8991	27.44/27.51	0.8127/0.8106	27.31/27.28	0.7960/0.7937	25.27/25.25	0.8165/0.8141	30.09/30.01	0.8712/0.8685
	ABPN [35]	31.29/31.21	0.9020/0.8997	27.48/27.55	0.8133/0.8113	27.33/27.29	0.7962/0.7941	25.33/25.28	0.8179/0.8152	30.13/30.04	0.8720/0.8695
	NCNET [36]	31.32/31.22	0.9023/0.8977	27.50/27.53	0.8141/0.8097	27.36/27.29	0.7970/0.7927	25.39/25.36	0.8196/0.8165	30.27/30.18	0.8728/0.8682
	SCSRN(our)	31.60/31.61	0.9068/0.9049	27.65/27.75	0.8180/0.8158	27.41/27.38	0.7992/0.7965	25.51/25.44	0.8228/0.8185	30.32/30.25	0.8754/0.8722
X4	Bicubic	26.89	0.7936	23.96	0.6732	24.31	0.6524	21.24	0.6351	26.82	0.7702
	FSRCNN [12]	28.11/25.65	0.8265/0.6791	25.21/23.75	0.7243/0.6064	25.33/23.75	0.7004/0.5777	22.52/21.58	0.6930/0.5789	27.60/25.03	0.7961/0.6304
	XLSR [37]	29.10/27.95	0.8577/0.7903	25.85/25.44	0.7468/0.7002	25.79/25.17	0.7183/0.6633	23.40/22.99	0.7373/0.6886	28.26/27.17	0.8151/0.7467
	SESR [38]	29.36/29.33	0.8633/0.8594	26.00/26.10	0.7512/0.7484	25.89/25.85	0.7217/0.7186	23.60/23.58	0.7453/0.7419	28.41/28.33	0.8188/0.8146
	ABPN [35]	29.20/29.21	0.8583/0.8566	25.87/25.26	0.7472/0.7456	25.86/25.85	0.7201/0.7180	23.57/23.55	0.7438/0.7417	28.32/28.27	0.8164/0.8142
	NCNET [36]	29.38/29.31	0.8634/0.8586	26.01/26.08	0.7516/0.7479	25.90/25.87	0.7220/0.7182	23.63/23.59	0.7463/0.7422	28.42/28.33	0.8190/0.8145
	SCSRN(our)	29.43/29.49	0.8638/0.8632	26.05/26.17	0.7532/0.7513	25.94/25.92	0.7237/0.7211	23.73/23.68	0.7493/0.7449	28.48/28.43	0.8207/0.8175

A. ABLATION STUDY

We analyze the contributions of each module in terms of five keywords: concatenation, weight clipping, DCT domain loss, MinClip and RepConv. In Table 1, we compared the performance according to the network structure of SCSRN.

1) CONCATENATION

The baseline (ConvBase) is a stack of seven successive convolution layers with 32 channels. To preserve the input dimensions, an "Add" operation was included with the input LR image replication before the transition layers. The replication operation was used to render the input LR image channels to $out\_channel * scale^2$, which improved the PSNR by 0.004 dB during the scratch training step. However, the TFlite performance decreased by 0.182 dB, and owing to the replicating operations, the inference time increased by 17.7 ms. Instead of the "Add" operation, we applied the concatenation (Concat) layer before the transition layers to preserve the input information. Consequently, the overall image accuracy improved during the fine-tuning stage by 0.089 dB and 0.163 dB at Tflite (int8).

2) WEIGHT CLIPPING

Thus, we applied weight clipping (WC) to resolve the asymmetric weight distribution and improve these results using the TFLite (int8) by 0.033 dB. This result depicts that the conversion loss of FP32 into INT8 was reduced by WC. However, upon applying the RepConv block with WC $= [-2,2]$ for specific layers, the same input and output dimensions were obtained, the scratch performance was deteriorated marginally, and the performance with TFLite (int8) did not improve. When we modified the RepConv block with WC $= [-3,3]$, the final performance with TFLite (int8) increased slightly by 0.002 dB.

3) DCT DOMAIN LOSS

Furthermore, we introduced the DCT domain $l1$ loss in the QAT step, which improved the PSNR value by 0.02 at TFlite (int8). The results of DCT domain $l1$ loss and without DCT domain $l1$ loss at QAT step provided 0.01dB better result at DIV2K validation dataset. To select a more better one, we tested these models on five benchmark datasets. The results are listed in Table 3, wherein the average PSNR of the training method including the DCT domain $l1$ loss was 0.006 dB higher than that of other methods. Thus, the DCT domain $l1$ loss was selected for our final SCSRN model.

4) MinClip

Overall, the experimental investigation revealed the significant influence of the Clipped ReLU on the inference time. Thus, we resolved this issue by replacing the Clipped ReLU with the minimum function (MinClip), and the inference time was reduced by 2.5 ms. The inference time in Table 1 measured at Galaxy Note20 mobile device with AI-Benchmark tool [43]

5) RepConv

We discovered that the application of the RepConv block on the entire network degrades the performance by 0.23 dB at TFlite(int8). Thus, the RepConv block was applied only on the backbone and transition layers, which exhibited the same input and output dimensions. These layers bear an identity connection that aid in information propagation, prevents the occurrence of vanishing gradients in deep networks [38], [44], and improves the the PSNR performance. The results are summarized in Table 4.

B. COMPARISONS WITH THE STATE-OF-THE-ARTS

We compare the proposed algorithm with conventional algorithms: FSRCNN [12], XLSR [37], SESR [38], ABPN [35], and NCNET [36]. The proposed model and previous works

TABLE 3. Comparison of SCSRN without DCT domain $l1$ loss and with DCT domain $l1$ loss on various datasets. (Red indicates int8 best PSNR values for each dataset).

Algorithm	Set5(int8)	Set14(int8)	B100(int8)	Urban100(int8)	DIV2K(int8)	**Average**
w/o_DCT_loss	31.56	27.73	27.38	25.48	30.23	28.476
w/_DCT_loss	31.61	27.74	27.38	25.45	30.25	28.486

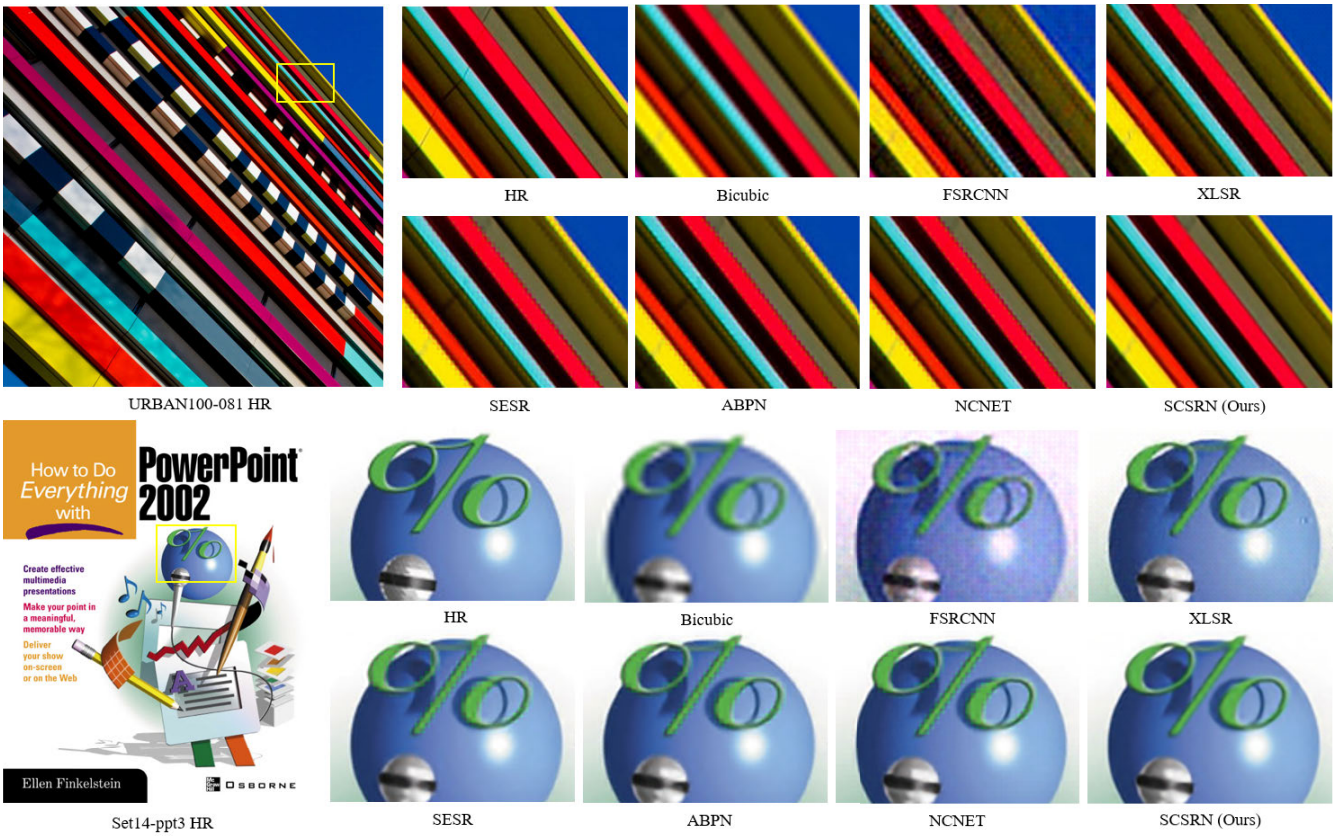


FIGURE 6. Visual results between the proposed method and baselines (x3 SR).

TABLE 4. Comparison of SCSRN (w/o MinClip, DCT loss) with RepConv_All.

	Scratch	Fine Tuning	QAT	TFLite (int8)
RepConv	30.32	30.33	30.23	30.23
RepConv_All	30.04	30.23	30.17	30.00

are tested on five benchmark datasets: Set5, Set14, B100, Urban100 and DIV2K validation set. For fair comparison, We measure PSNR and SSIM of each algorithm on RGB domain. ABPN and NCNET officially provide the source codes that operates in the RGB domain, so we obtained the results using this source code. In case of FSRCNN [12] and SESR [38] to support the RGB channels instead of using only Y. Therefore, the number of channels in the first and last convolution layers were adjusted from one to three. There is no officially released code for XLSR [37], so we reimplemented it. Three different scales are tested X2, X3 and X4.

Table 2 compares PSNR/SSIM scores. In this table, FP32 refers to the result before quantization and INT8 refers to the result after quantization. FSRCNN and XLSR were quantized by PTQ, and the rest of the algorithms were quantized by QAT. The proposed algorithm provides the superiror performance for all the benchmark datasets before/after quantization. FSRCNN and XLSR with PTQ show high PSNR degradation by INT8 quantization. On the other hand, in algorithms using QAT, PSNR degradation due to INT8 quantization is quite small from 0.1 to 0.2 dB.

We show visual result of our final INT8 quantized model previous works in Fig. 6. SCSRN successfully reconstruct edges in the HR images and reveal better shaped SR outputs compared to the previous works. We note that FSRCNN [12] was designed without considering quantization, thus the image quality is significantly degraded by INT8 quantization.

TABLE 5. Runtime (ms) simulation result on the mobile devices (scale x3).

	Runtime NNAPI (ms)		#Params
	Gal. Note20	Gal. Z Fold4	
FSRCNN [12]	65.20	21.70	25.35K
XLSR [37]	101.10	6.33	21.95K
SESR [38]	58.70	15.90	23.64K
ABPN [35]	50.80	17.10	42.54K
NCNET [36]	41.01	4.92	53.00K
SCSRN(ours)	40.80	4.62	53.01K

TABLE 6. Official competition results of mobile AI & AIM 2022 real-time super-resolution challenge [45].

	PSNR	SSIM	NPU Runtime (ms)	Score
SCSRN(our)	30.03	0.8738	19.20	22.22
maciejos_s	29.88	0.8705	15.90	21.84
CCjiahao	29.82	0.8697	15.10	21.08
zion_	29.76	0.8675	15.00	19.59
NBCS	29.80	0.8675	16.10	19.27

C. THE INFERENCE TIME AT THE MOBILE DEVICE

The model inference time were reported using the Galaxy Note20 with Snapdragon 865+ and Galaxy Z Fold4 with Snapdragon 8+ Gen 1 on the commercial mobile devices. To measure the inference time, we used AI Benchmark [43] to obtain the NNAPI execution periods, as expressed in Table 5. After 100 iterations, the inference times of all iterations were averaged. The proposed model involved a larger number of parameters and operations compared to other methods. However, the results of the inference time demonstrate that the proposed SCSRN model is faster than all such existing models. This proves the superior efficiency of the developed skip-concatenation network on mobile super-resolution image tasks.

D. THE MAI2022 REAL-TIME SUPER-RESOLUTION CHALLENGE

This research was conducted for participating in the MAI2022 Real-Time Super Resolution Challenge. The final results of MAI2022 are displayed in Table 6. The PSNR result of MAI2022 was obtained on the DIV2K test dataset, and the inference time was evaluated on Synaptics Dolphin smart TV platform with a dedicated NPU (VS680). The score of each final submission was evaluated according to Eq. (13) (C denotes a constant normalization factor):

$$\text{Final Score} = \frac{2^{2\cdot \text{PSNR}}}{C \cdot \text{runtime}} \tag{13}$$

V. CONCLUSION

This study proposed an efficient and lightweight super-resolution network by directly concatenating an input LR image and an intermediate feature map at the middle of the network. To reduce the quantization error, we introduced weight clipping. Moreover, the reparameterization method was selectively applied and provided the improved super-resolution image quality without any performance degradation in terms of inference time. Based on the contributions, the proposed network achieved 30.03 dB in PSNR and 19.20 ms in NPU runtime at the Mobile AI & AIM 2022 Real-Time Single-Image Super-Resolution Challenge.

ACKNOWLEDGMENT

Their work is primarily motivated by the previous challenge works in Mobile AI Workshop 2021, especially two pioneering research [35], [37]. Moreover, the authors deeply thank the challenge organizers of the Mobile AI and AIM 2022 Workshop [46] for the opportunity of participating in this challenge.

REFERENCES

[1] C. Tuna, G. Unal, and E. Sertel, "Single-frame super resolution of remote-sensing images by convolutional neural networks," *Int. J. Remote Sens.*, vol. 39, no. 8, pp. 2463–2479, 2018.
[2] P. Wang and E. Sertel, "Channel–spatial attention-based pan-sharpening of very high-resolution satellite images," *Knowl.-Based Syst.*, vol. 229, Oct. 2021, Art. no. 107324.
[3] H. Lu, Y. Li, S. Nakashima, H. Kim, and S. Serikawa, "Underwater image super-resolution by descattering and fusion," *IEEE Access*, vol. 5, pp. 670–679, 2017.
[4] Y. Chen, K. Niu, Z. Zeng, and Y. Pan, "A wavelet based deep learning method for underwater image super resolution reconstruction," *IEEE Access*, vol. 8, pp. 117759–117769, 2020.
[5] J. S. Isaac and R. Kulkarni, "Super resolution techniques for medical image processing," in *Proc. Int. Conf. Technol. Sustain. Develop. (ICTSD)*, Feb. 2015, pp. 1–6.
[6] D. Qiu, L. Zheng, J. Zhu, and D. Huang, "Multiple improved residual networks for medical image super-resolution," *Future Gener. Comput. Syst.*, vol. 116, pp. 200–208, Mar. 2021.
[7] K. Hayat, "Multimedia super-resolution via deep learning: A survey," *Digit. Signal Process.*, vol. 81, pp. 198–217, Oct. 2018.
[8] L. Zhang and X. Wu, "An edge-guided image interpolation algorithm via directional filtering and data fusion," *IEEE Trans. Image Process.*, vol. 15, no. 8, pp. 2226–2238, Aug. 2006.
[9] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, Nov. 2010.
[10] Z. Shao, L. Wang, Z. Wang, and J. Deng, "Remote sensing image super-resolution using sparse representation and coupled sparse autoencoder," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 8, pp. 2663–2674, Aug. 2019.
[11] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2015.
[12] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 391–407.
[13] 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)*, 2018, pp. 252–268.
[14] C. Ledig, L. Theis, F. Huszar, 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. 4681–4690.
[15] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep Laplacian pyramid networks for fast and accurate super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 624–632.
[16] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 694–711.
[17] A. Bulat and G. Tzimiropoulos, "Super-FAN: Integrated facial landmark localization and super-resolution of real-world low resolution faces in arbitrary poses with GANs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 109–117.

[18] 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. 136–144.
[19] Y. Wang, F. Perazzi, B. McWilliams, A. Sorkine-Hornung, O. Sorkine-Hornung, and C. Schroers, "A fully progressive approach to single-image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 864–873.
[20] Y. Zhang, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 286–301.
[21] X. Luo, Y. Xie, Y. Zhang, Y. Qu, C. Li, and Y. Fu, "LatticeNet: Towards lightweight image super-resolution with lattice block," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 272–289.
[22] Z. Wang, J. Chen, and S. Hoi, "Deep learning for image super-resolution: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3365–3387, Mar. 2020.
[23] 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.
[24] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1637–1645.
[25] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, "Second-order attention network for single image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11065–11074.
[26] Y. Zhang, H. Wang, C. Qin, and Y. Fu, "Aligned structured sparsity learning for efficient image super-resolution," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 2695–2706.
[27] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.
[28] Z. Hou and S.-Y. Kung, "Efficient image super resolution via channel discriminative deep neural network pruning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 3647–3651.
[29] J. Liu, Q. Wang, D. Zhang, and L. Shen, "Super-resolution model quantized in multi-precision," *Electronics*, vol. 10, no. 17, p. 2176, Sep. 2021.
[30] S. A. Tailor, J. Fernandez-Marques, and N. D. Lane, "Degree-quant: Quantization-aware training for graph neural networks," 2020, *arXiv:2008.05000*.
[31] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, pp. 1789–1819, Mar. 2021.
[32] Z. He, T. Dai, J. Lu, Y. Jiang, and S.-T. Xia, "FAKD: Feature-affinity based knowledge distillation for efficient image super-resolution," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2020, pp. 518–522.
[33] S. Park and N. Kwak, "Local-selective feature distillation for single image super-resolution," 2021, *arXiv:2111.10988*.
[34] Y. Wang, S. Lin, Y. Qu, H. Wu, Z. Zhang, Y. Xie, and A. Yao, "Towards compact single image super-resolution via contrastive self-distillation," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 1122–1128, doi: 10.24963/IJCAI.2021/155.
[35] Z. Du, J. Liu, J. Tang, and G. Wu, "Anchor-based plain net for mobile image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 2494–2502.
[36] Z. Luo, Y. Li, L. Yu, Q. Wu, Z. Wen, H. Fan, and S. Liu, "Fast nearest convolution for real-time efficient image super-resolution," 2022, *arXiv:2208.11609*.
[37] 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.
[38] K. Bhardwaj, M. Milosavljevic, L. O'Neil, D. Gope, R. Matas, A. Chalfin, N. Suda, L. Meng, and D. Loh, "Collapsible linear blocks for super-efficient super resolution," *Proc. Mach. Learn. Syst.*, vol. 4, pp. 529–547, Jan. 2022.
[39] M. Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/
[40] Google. *TensorFlow Lite 8bit Quantization Spec*. Accessed: Jan. 2022. [Online]. Available: https://www.tensorflow.org/lite/performance/quantization_spec
[41] L. Roeder. (Dec. 2017). *Netron, Visualizer for Neural Network, Deep Learning, and Machine Learning Models*. [Online]. Available: https://zenodo.org/record/6551590#.Y7ak-3ZBzIU
[42] A. Ignatov, R. Timofte, M. Denna, and A. Younes, "Real-time quantized image super-resolution on mobile NPUs, mobile AI 2021 challenge: Report," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 2525–2534.
[43] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. Van Gool, "AI benchmark: Running deep neural networks on Android smartphones," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, 2018, pp. 1–15.
[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[45] A. Ignatov et al., "Efficient and accurate quantized image super-resolution on mobile NPUs, mobile AI & AIM 2022 challenge: Report," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, 2022, pp. 1–38.
[46] (2022). *Mobile AI & AIM 2022 Organizers*. [Online]. Available: https://data.vision.ee.ethz.ch/cvl/aim22/

GANZORIG GANKHUYAG received the B.S. degree in information and communication engineering from Huree University, Mongolia, in 2006, the M.S. degree in electronic engineering from Konkuk University, Seoul, South Korea, in 2009, and the Ph.D. degree in electronic engineering from Yonsei University, Seoul, in 2020. Since 2018, he has been associated with the Mobility Platform Research Center, Korea Electronics Technology Institute, Seongnam, South Korea, where he is currently a Research Engineer. His research interests include signal processing, artificial intelligence processing, and video codec.

JINGANG HUH received the B.S. degree in electronic engineering from Gangneung-Wonju National University, Gangwon, South Korea, in 2015, and the M.S. degree in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2017. Since 2016, he has been associated as a Research Engineer with the Intelligent Image Processing Research Center, Korea Electronics Technology Institute, Seongnam, South Korea. His research interests include multimedia signal processing, artificial intelligence processing, and distributed system development.

MYEONGKYUN KIM received the B.S. degree in computer engineering from Baekseok University, Cheonan, South Korea, in 2014, and the M.S. degree from the Department of Computer Software, Hanyang University, Seoul, South Korea, in 2016. He is currently serving as a Research Engineer with the Intelligent Image Processing Research Center, Korea Electronics Technology Institute, Seongnam, South Korea. His research interests include multimedia signal processing, artificial intelligence processing, and VR/AR technologies.

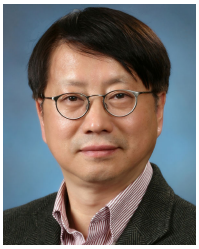
KIHWAN YOON received the B.S. degree from the Department of Electronic and Electrical Engineering, Dankook University, Yongin, South Korea, in 2019. He is currently pursuing the Integrated M.S. and Ph.D. degree in electrical and computer engineering from the University of Seoul, Seoul, South Korea. In addition, he is currently serving as a Research Engineer with the Intelligent Image Processing Research Center, Korea Electronics Technology Institute, Seongnam, South Korea. His research interests include multimedia signal processing and artificial intelligence processing.

HYEONCHEOL MOON received the B.S. and M.S. degrees in electronics and information engineering from Korea Aerospace University, Goyang, South Korea, in 2018 and 2020, respectively. Since 2021, he has been serving as a Research Engineer with the Intelligent Image Processing Research Center, Korea Electronics Technology Institute, Seongnam, South Korea. His research interests include multimedia signal processing and artificial intelligence processing.

SEUNGHO LEE received the B.S. and M.S. degrees from the Department of Computer and Software, Hanyang University, Seoul, South Korea, in 2014 and 2017, respectively. From 2017 to 2019, he worked as a System Engineer at Pixtree Inc., Seoul. Since 2019, he has been serving as a Research Engineer with the Intelligent Image Processing Research Center, Korea Electronics Technology Institute, Seongnam, South Korea. His research interests include multimedia signal processing, compression, and artificial intelligence processing.

JINWOO JEONG received the B.S., M.S., and Ph.D. degrees in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2004, 2006, and 2011, respectively. From 2011 to 2015, he served as a Senior Video Signal Processing Engineer at Samsung Electronics Company Ltd., Suwon, South Korea. Since 2016, he has been working as a Principal Research Engineer with the Intelligent Image Processing Research Center, Korea Electronics Technology Institute, Seongnam, South Korea. His research interests include multimedia signal processing, artificial intelligence processing, and VR/AR technologies.

SUNGJEI KIM received the B.S., M.S., and Ph.D. degrees in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2004, 2006, and 2011, respectively. From 2011 to 2015, he was a Senior Video Signal Processing Engineer at Samsung Electronics Company Ltd., Suwon, South Korea. Since 2015, he has been serving as a Principal Research Engineer with the Intelligent Image Processing Research Center, Korea Electronics Technology Institute, Seongnam, South Korea. His research interests include multimedia signal processing, artificial intelligence processing, and VR/AR technologies.



YOONSIK CHOE (Life Senior Member, IEEE) received the B.S. degree in electrical engineering from Yonsei University, Seoul, South Korea, in 1979, the M.S.E.E. degree in systems engineering from Case Western Reserve University, Cleveland, OH, USA, in 1984, the M.S. degree in electrical engineering from Pennsylvania State University, State College, PA, USA, in 1987, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1990. From 1990 to 1993, he worked as a Principal Research Staff at the Industrial Electronics Research Center, Hyundai Electronics Company Ltd. Since 1993, he has been associated with the Department of Electrical and Electronic Engineering, Yonsei University, where he is currently serving as a Distinguished University Professor of industry. His research interests include video coding, video communication, statistical signal processing, and deep learning-based digital image processing.