Reconstructed Convolution Module Based Look-Up Tables for Efficient Image Super-Resolution

Guandu Liu[1,3] Yukang Ding[2] Mading Li[2] Ming Sun[2] Xing Wen[2] Bin Wang[1,3*]
[1]School of Software, Tsinghua University
[2]Kuaishou Technology
[3]Beijing National Research Center for Information Science and Technology (BNRist), China
liugd21@mails.tsinghua.edu.cn, {dingyukang, limading, sunming03}@kuaishou.com
td.wenxing@gmail.com, wangbins@tsinghua.edu.cn

Abstract

Look-up table (LUT)-based methods have shown the great efficacy in single image super-resolution (SR) task. However, previous methods ignore the essential reason of restricted receptive field (RF) size in LUT, which is caused by the interaction of space and channel features in vanilla convolution. They can only increase the RF at the cost of linearly increasing LUT size. To enlarge RF with contained LUT sizes, we propose a novel Reconstructed Convolution (RC) module, which decouples channel-wise and spatial calculation. It can be formulated as n^2 1D LUTs to maintain $n \times n$ receptive field, which is obviously smaller than $n \times n$D LUT formulated before. The LUT generated by our RC module reaches less than 1/10000 storage compared with SR-LUT baseline. The proposed Reconstructed Convolution module based LUT method, termed as RCLUT, can enlarge the RF size by 9 times than the state-of-the-art LUT-based SR method and achieve superior performance on five popular benchmark dataset. Moreover, the efficient and robust RC module can be used as a plugin to improve other LUT-based SR methods. The code is available at https://github.com/liuguandu/RC-LUT.

1. Introduction

Single image super-resolution (SR) aims to recover the high-resolution (HR) image from a low-resolution (LR) image and to bring back clearer edges, textures, and details while increasing the resolution of the input. In the past decade, deep learning-based SR methods [43, 23, 18, 44, 35, 20, 13, 26, 27] have made remarkable improvements compared with traditional SR methods (*e.g.*, interpolation based [17], sparse coding based [34, 33, 40, 5, 12]). However, such kind of methods often possess huge amount of parameters, which has high computation cost and cannot be practically used on devices with limited computational resources. Exploring the pratical and real-time SR solutions have been a growing trend in the single image super resolution (SISR) community.

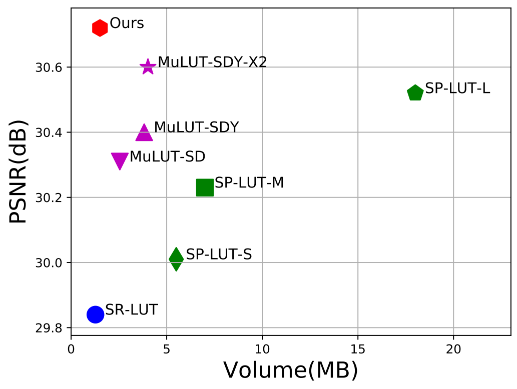


Figure 1. Comparison of PSNR and storage on Set5 benchmark dataset for $\times 4$ SR task. We compare our method with prior LUT-based SR methods. Our method achieves the superior performance with relatively less LUT storage.

Look-up table (LUT) based super-resolution (SR) methods like SR-LUT [16] cache trained SR network results for every potential input in a LUT, simplifying inference by replacing runtime computation with faster indexing. However, this strategy requires a limited receptive field (RF), as the input space size grows exponentially with an increase in input pixels. Specifically, SR-LUT has a 2×2 input size, yielding a 3×3 RF by rotation ensemble, and requires $b^{2\times 2} \times r^2$ bytes to store its LUT for upscaling by a factor of r ($b = 255$). For example, a 3×3 vanilla convolution generates 255^9 mappings, consuming 1.72 TB in LUT form.

[*]Corresponding Author

A larger RF allows a model to capture more intricate semantics and structures within an image, playing a critical role in the training process. However, for SR-LUT, the exponential growth of LUT size significantly limits RF improvements. Attempts to solve this, such as MuLUT [22] and SPLUT [24], propose cascading multiple parallel LUTs to expand the RF to 9×9 and 6×6, respectively. These methods attempt to divide the entire LUT into several sub-LUTs, with a linear increase in LUT size. However, they don't address the primary cause of restricted RF size in LUT-based methods. Furthermore, the RF size of these LUT-based methods falls short compared to DNN SR methods, resulting in performance inferior to simple DNN models, such as FSRCNN[9].

Vanilla convolution amalgamates features across both spatial and channel dimensions, necessitating that the LUT format traverses all potential combinations and permutations of input pixels. Essentially, spatially dependent convolution refers to other feature points within the spatial neighborhood to generate the input-output mapping of the current feature point. Given that the most significant factor constraining the performance of LUT-based methods is spatially dependent convolution, we question the status quo. Specifically, we contemplate decoupling the spatial and channel calculations of the convolution process and propose storing the LUT with spatially independent convolution. This could potentially address the inherent limitations of existing LUT-based methods, providing a new way forward for this class of algorithms.

In this paper, we propose a novel Reconstructed Convolution (RC) method designed to decouple the spatial and channel calculations, thus effectively increasing the RF of the LUT while significantly reducing storage requirements. This decoupled operation enables the network to circumvent the constraints posed by the necessity of traversing all spatial pixel combinations. As a result, our method can employ $n \times n$ 1D LUTs to approximate the effect of an $n \times n$ convolution layer, cutting the LUT size from the initial b^{n^2} to $b \times n^2$.Building on our RC approach, we introduce a practical Reconstructed Convolution module-based LUT method (RCLUT) tailored for the SR task, allowing for an expansion of the RF with minimal storage consumption. As illustrated in Figure 1, our RCLUT attains competitive performance with minimal LUT size, demonstrating an optimal balance between performance and LUT storage.

Additionally, our RC method can be efficiently integrated as a plugin module. Although the RC module sacrifices the interactive information between two-dimensional features, its minimal storage requirements and large receptive field successfully offset the shortcomings of previous LUT-based methods. In our implementation, we integrate the RC module on top of SRLUT, resulting in a cost of only about 1/10,000 of the original storage for a 13-fold improvement in the RF size.

To conclude, the contribution of our work includes:

- We propose a novel Reconstructed Convolution (RC) method with large RF, which decouples the spatial and channel calculation of convolution. Based on the RC method, our RCLUT model achieves significant performance with less storage.
- Our RC method can be designed as a plugin module, which brings improvement to LUT-based SR methods with slight increasing size.
- Extensive results show that our method obtains superior performance compared with SR methods based on LUT. It is a new state-of-the-arts LUT method in SR task.

2. Related Work

2.1. Traditional Super-Resolution

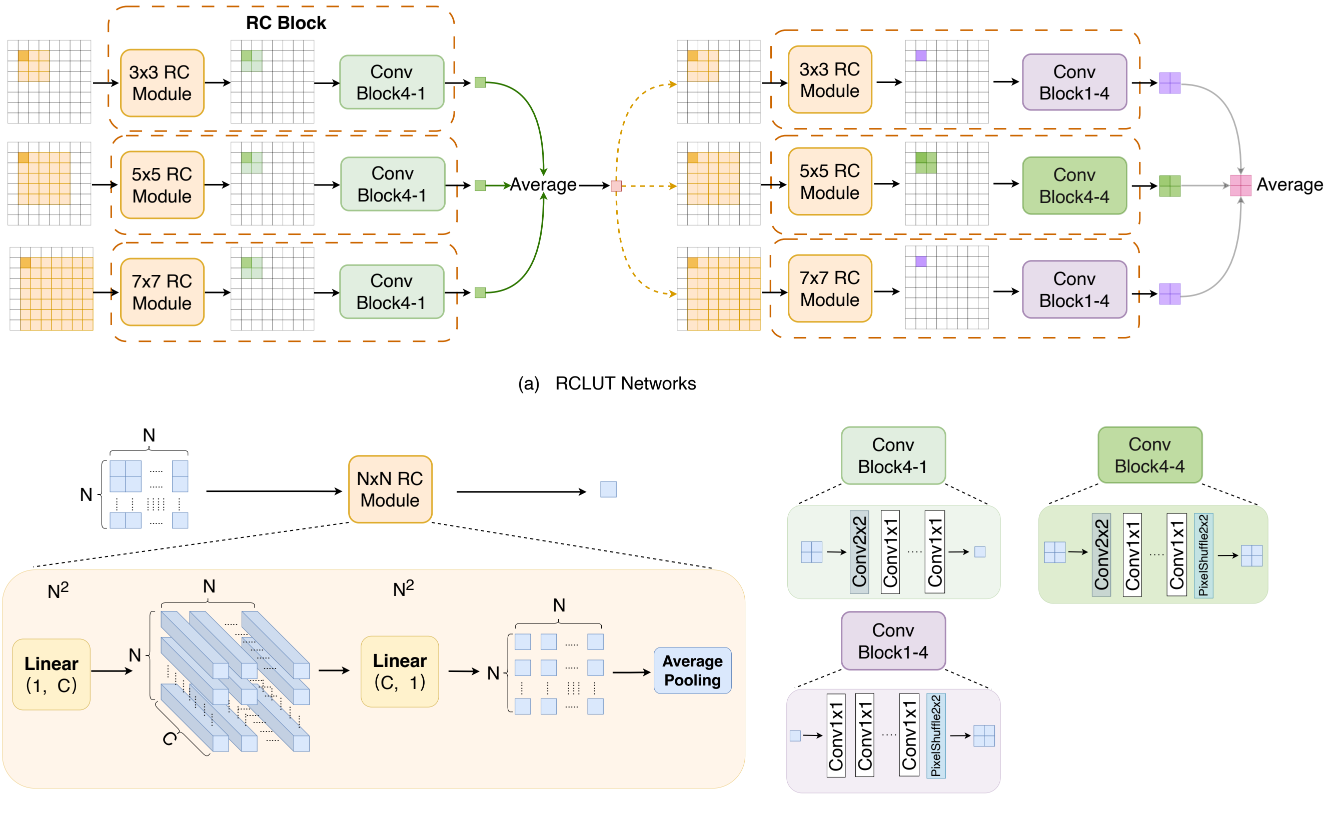
Interpolation-based methods such as Nearest, Bilinear, and Bicubic [17] are highly efficient but often yield blurry results as they overlook the content of images. Sparse coding-based methods [33, 34, 39, 36, 4, 12] offer promising results superior to interpolation-based methods by restoring low-resolution images to high-resolution ones via a learned sparse dictionary, albeit at a significantly increased runtime cost. The RAISR method [28], which learns a set of filters for different patch attributes, provides satisfying visual results. However, it requires the calculation of patch gradients and angles during the prediction phase, making it slower than interpolation methods.

2.2. Deep Learning Based Super-Resolution

Deep learning-based SR methods have seen tremendous progress, with architectures like SRCNN [8], VDSR, EDSR, and RCAN [18, 23, 43] aiming for performance enhancement. However, these methods require substantial computational resources. To mitigate this, ESPCN [29] and FSRCNN [9] utilize smaller networks. Other strategies include parameter reduction [2, 35, 6, 41, 38, 7] or quantization [21, 30, 42, 3], maintaining performance while reducing computational demand. Nonetheless, these models remain too heavy for devices with limited computational resources, like mobile devices.

2.3. LUT Based Super-Resolution

As mentioned in Section 1, Jo and Kim propose a pioneering work (SR-LUT), which utilizes LUTs in SR [16]. They train a simple deep SR network with limited receptive field (RF), and transfer inputs (as indices) and network outputs (positions as indices and pixel intensities as values) to a LUT. The LUT is used to generate final results during



(a) RCLUT Networks

(b) NxN RC Module

(c) Conv Blocks

Figure 2. Overview of RCLUT networks. (a) The overall framework of our RCLUT networks. It contains two cascaded stages, each stage module has three branches with different receptive field RC Blocks. (b) The details of RC method for $N \times N$ RF size. It is composed of Linear and Average Pooling operations. (c) The illustration of various convolution blocks in RC Block. The input and out values of Conv Block4-1 are 4 and 1, respectively. The Conv Block1-4 contains 1 input value and 4 output value for less storage. The Conv Block4-4, which references 4 values as input and predicts 4 output values, is only used once in the second stage module of RCLUT.

the test phase. Since there are no additional calculations required, SR-LUT can run as fast as interpolation based SR methods, *i.e.* tens of milliseconds, on mobile devices. Since the size of a LUT increases exponentially with the indexing capacity, the RF is limited to 3×3 in SR-LUT. However, the size of RF is proved crucial according to [11] and SR-LUT naturally obtains inferior performance. Li *et al.* [22] propose MuLUT, which increases the RF by introducing hand-crafted indexing patterns and cascading LUTs. By these two schemes, MuLUT achieves significant improvement over SR-LUT with the cost of a linear growth of the total LUT size. Similarly, Ma *et al.* [24] also adapt the idea of cascading LUTs to enlarge the RF. They propose a framework of series-parallel LUT (SPLUT) which introduces channel-level LUTs and parallelly processes the two components separated from the original 8-bits input. These two method devides the single LUT of SR-LUT to multi-LUTs to increase the RF of SR models.

However, their good performance is based on the linear growth of lut size at the expense of large amounts of storage resources. The trade-off between the RF and LUT size is still a challenging problem. Our method overcomes the burden of heavy LUT size when increasing the RF, thanks to the novel reconstructed convolution (RC) method, which approximates a vanilla convolution performance with exaggeratedly small LUT size.

3. Method

3.1. Overview

To tackle the challenge of LUT size growth when enlarging the receptive field (RF), we propose a reconstructed convolution (RC) method. RC can represent a $n \times n$ RF convolution as $n \times n$ 1D LUTs, approximating the performance of vanilla convolution. Using this efficient RC method, we develop a Reconstructed Convolution module-based LUT method, dubbed as RCLUT, which employs parallel and cascaded structures. Moreover, the RC module can act as a flexible Plugin module, capable of enhancing the performance of other LUT-based methods.

3.2. Reconstructed Convolution Module

First, we delve into the issue of LUT size escalation with the increase in RF. As stated in [22], the LUT size exponentially balloons with an increase in indexing inputs, as an RF of size $n \times n$ should be cached into a LUT of dimension $n \times n$. To detail this, consider the convolution of an 8-bit $n \times n$ input to an 8-bit $r \times r$ output. The full SR-LUT would require $(2^8)^{(n^2)} \times r^2$ bytes. Each pixel offers 2^8 potential values, thus the whole $n \times n$ input can manifest $(2^8)^{(n^2)}$ different possible combinations of pixel values. For each input, there are r^2 output pixels. This indicates that the input size is the primary factor responsible for the explosion in SR-LUT size.

In order to overcome the problem of Look-Up Table (LUT) size explosion with increasing receptive field (RF), we introduce a novel formulation of convolution known as Reconstructed Convolution (RC). Unlike the conventional convolution with $n \times n \times c\_out$ filters that calculates $n \times n$ spatial features and then adds up the c_out results in the channel dimensions, our RC method reverses this order. It initially breaks the interconnections between different pixels and uses a minimal network to perform channel-wise increment and reduction operations on each pixel individually. This is followed by an average pooling operation to generate the final feature map. In order to represent an $N \times N$ RF, the RC method first uses N^2 linear operations (with c_in=1, c_out=C) to enhance the dimensionality of the $N \times N$ patches to obtain a feature map of $N \times N \times C$. Subsequently, another set of N^2 linear operations (with c_in=C, c_out=1) is employed to reduce the feature map channel to 1, leading to an $N \times N \times 1$ feature map. Finally, an average pooling operation merges the spatial features to produce the output value. The training process of reconstructed convolution can be formulated as

$$\begin{aligned} z_{(m+i,m+j)} &= W'^{T}_{ij}(W^{T}_{ij}x_{(m+i,n+j)} + b_{ij}) + b'_{ij}, \\ y_{mn} &= \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} z_{(m+i,m+j)}. \end{aligned} \tag{1}$$

In this formula, m, n are the coordinates of the pixels to be computed in the current feature map, and N is the kernel size of reconstructed convolution. $W \in R^{1\times C}$, maps single-channel pixels into high-dimensional features, while $W' \in R^{C\times 1}$ maps high-dimensional features back to single-channel features. b, b' represent the bias terms. In this setup, the reconstructed convolution for each pixel can be transformed into a 1D look-up table (LUT), and an average pooling operation is used to cheaply expand the receptive field (RF), as shown in Figure 3(b). In comparison with the 4D LUT format of SR-LUT, shown in Figure 3(a), our 1D LUTs offer advantages in storage and inference speed. As to the inference stage, for anchor I_0 in Figure 3(b), the corresponding SR values $\mathbf{V'}$ from reconstructed convolution are obtained by

$$\mathbf{V'} = \frac{1}{N^2}\sum_{i=0}^{N^2-1}(LUT_i[I_i]). \tag{2}$$

While N represents the size of reconstructed convolution. And it needs N^2 LUTs for pixel retrieval. The total size of all 1D-LUT is $2^8 \times n^2 \times r^2$. As shown in Table2, by transforming $n \times n$ convolutions to n^2 1D LUTs (followed by global pooling), the explosion of LUT sizes with respect to RF can be successfully contained.

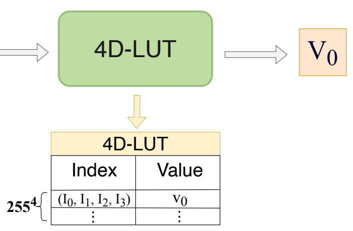
3.3. RCLUT Network Architecture

RC module processes spatial and channel-wise feature separately, which lacks interactive information between both of them. As it somewhat prevents model performance, we add Conv block [14] behind the RC module to complement interactive information from the space and channel features. Specifically, we build the Reconstructed Convolution Block, which contains one RC Modue and one Conv Block [14]. Its corresponding SR values $\mathbf{V}$ can be obtained from $\mathbf{V'}$,

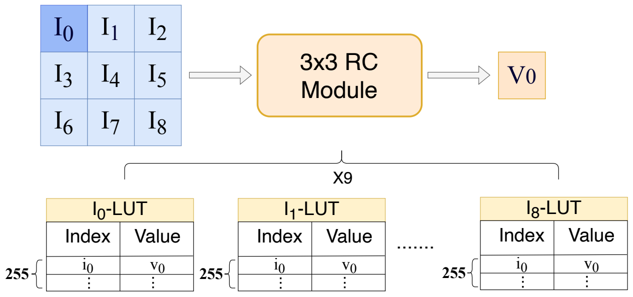
$$\mathbf{V}_{(0,0)} = LUT[\mathbf{V'}_{(0,0)}][\mathbf{V'}_{(0,1)}][\mathbf{V'}_{(1,0)}][\mathbf{V'}_{(1,1)}]. \tag{3}$$

The RC module allows for easy expansion of the receptive field (RF) without a significant increase in computational load. Inspired by this, we utilize RC blocks with different RF sizes to capture diverse image details. Specifically, we construct a Reconstructed Convolution based Look-Up Table (RCLUT) network, which employs multiple RC blocks. The RCLUT network (shown in Figure 2(a)) consists of two cascaded stages, each containing three parallel branches. To achieve different scale feature maps like [32], these RFs of three branches respectively are 3×3, 5×5, 7×7 through our RC Blocks.

In addition, within the first stage of our network, three Conv Block4-1 are utilized after the RC modules. In the second stage, an RC module with 5×5 RF is coupled with a Conv Block4-4, which provides upscaling capabilities. To minimize the overall LUT size, two Conv Block1-4s are placed after the other two branches. This is because the Conv Block1-4 in the LUT format only requires $(2^8) \times 4$B storage, while the Conv Block4-4 demands $(2^8)^4 \times 4$B storage, without LUT-sampling. Working in tandem with the two cascaded stages and the rotation strategy from [22, 24], the RCLUT network can achieve a 27×27 RF size while maintaining only a 1.515MB LUT size by sampling to 2^4. Compared to MuLUT, the RF size of RCLUT is $9\times$ larger, but the LUT size is $2.68\times$ smaller. Our RCLUT method offers a far more efficient approach to balancing the trade-off between RF and LUT size.



(a) 4D-LUT Inference



(b) RC Module Inference

Figure 3. The different LUT inference formulations of SR-LUT and RC module

Discuss the receptive field (RF) size between RCLUT and MuLUT. Assuming that the RF size of RC module in 1st stage/2nd stage is N_1/N_2 and Conv Block is M_1/M_2. For RCLUT, it couples the two modules to increase the RF size to $M_1 + N_1 - 1$. Thanks to the rotation operation [16, 22], the RF size of RCLUT 1st stage can further achieve to

$$RF_1 = 2(M_1 + N_1 - 1) - 1. \tag{4}$$

For 2nd stage, since the anchor point in convolution process becomes the central point, RF_1 will shrink by half. Thus the RF size of RCLUT is

$$\begin{aligned} RF_2 &= 2(\frac{(RF_1 - 1)}{2} + M_2 + N_2 - 1) - 1 \\ &= 2M_1 + 2M_2 + 2N_1 + 2N_2 - 7. \end{aligned} \tag{5}$$

While for MuLUT, its RF size increases to $2M_1 - 1$ through rotation. Furthermore, the cascading 2nd stage also helps to increase the RF size to

$$\begin{aligned} RF_2' &= 2(\frac{((2M_1 - 1) - 1)}{2} + M_2 - 1) - 1 \\ &= 2M_1 + 2M_2 - 3. \end{aligned} \tag{6}$$

RCLUT's RF size can be $2N_1 + 2N_2 - 4$ more than MuULT's in the case that M_1 and M_2 of the two methods correspond equally. We select the branch where 7x7 module to calculate the final RF size of the network, where $N_1 = 7, M_1 = 2, N_2 = 7, M_2 = 1$. Under this setting, RCLUT's RF size is 27×27. The RF size of MuLUT is 9×9, where $M_1 = 3, M_2 = 3$.

3.4. RC-Plugin Module

In order to further demonstrate the effectiveness of the Reconstructed Convolution method, we design our RC method as a Plugin module, featuring a 5×5 RF size as shown in Figure 2(b). This RC-Plugin Module can be flexibly incorporated into LUT based SR methods (e.g., SR-LUT and MuLUT). With a 5×5 RF size, the RC-Plugin Module can extend the RF size of the base LUT methods, adding only 0.415 KB to the LUT size by sampling to 2^4. To easily integrate our RC-Plugin module, we only position it ahead of other LUT methods. Given the different architectures of base models, we add three RC-Plugin modules in MuLUT and one in SR-LUT, as shown in Figure 4. This helps to extend the RF size of the MuLUT network from 9×9 to 17×17.

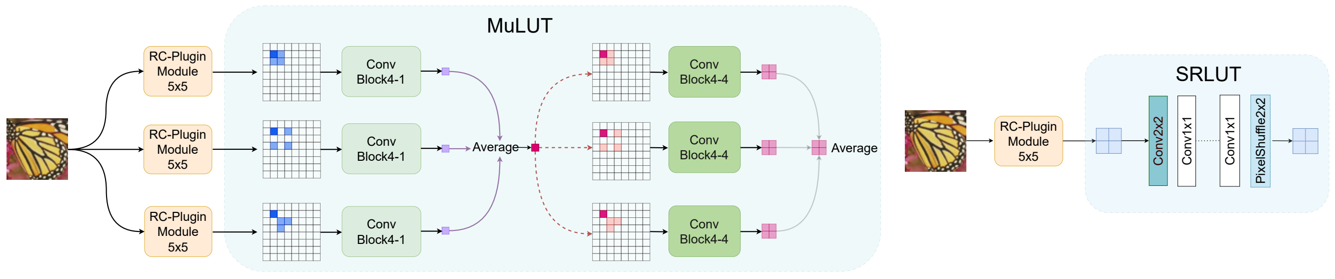
4. Experiment

In this section, we first introduce the datasets and training details of our proposed RCLUT networks. Following this, we compare RCLUT with several state-of-the-art LUT-based SR methods through quantitative and qualitative evaluations. Additionally, we quantify the advantages of our RC-Plugin module when incorporated into other LUT-based methods. Finally, we conduct ablation studies to demonstrate the effectiveness of our RCLUT model and the RC module.

4.1. Datasets and Experimental Setting.

Datasets and Metrics. We train the RCLUT Networks on the DIV2K dataset [1], which consists of 2K resolution images for the SR task. We utilize the widely used 800 training images from DIV2K to train our RCLUT model. Our focus is on the $\times 4$ upscaling factor in the SR task, where low-resolution (LR) images are simply downscaled via *Bicubic* interpolation. We evaluate the performance of our method using public benchmarks of Set5, Set14, BSD100 [25], Urban100 [15], and Manga109 [10]. For fair comparison, we use only the Peak Signal-to-Noise Ratio (PSNR) and structural similarity index (SSIM) [37] both at the Y-channel as evaluation metrics. Additionally, we consider the storage size of the LUTs to assess the efficiency of other methods.

Training Setting. Our training setup is as follows: We set the channel increasing number C in the RC Linear Operation to 64. Our RCLUT model is trained for $200,000$ iterations using the Adam optimizer [19], with a learning rate of $1e^{-4}$ and a batch size of 32. The mean-squared error (MSE) loss function is selected as the optimization ob-



(a) MuLUT with RC Plugin

(b) SR-LUT with RC Plugin

Figure 4. The overall framework of MuLUT and SR-LUT with RC-Plugin module. (a)As for three parallel branches architecture in MuLUT, three RC-Plugin Modules of 5×5 RF are put ahead of each branch. (b)SR-LUT is equipped with one RC-Plugin Module.

Table 1. The comparison with other methods.

	Method	RF Size	LUT size	Set5	Set14	BSDS100	Urban100	Manga109
Interpolation	Nearest	1×1	-	26.25/0.7372	24.65/0.6529	25.03/0.6293	22.17/0.6154	23.45/0.7414
	Bilinear	2×2	-	27.55/0.7884	25.42/0.6792	25.54/0.6460	22.69/0.6346	24.21/0.7666
	Bicubic	4×4	-	28.42/0.8101	26.00/0.7023	25.96/0.6672	23.14/0.6574	24.91/0.7871
LUT	SR-LUT [16]	3×3	1.274MB	29.82/0.8478	27.01/0.7355	26.53/0.6953	24.02/0.6990	26.80/0.8380
	SPLUT-L [24]		18MB	30.52/0.8631	27.54/0.7520	26.87/0.7091	24.46/0.7191	27.70/0.8581
	MuLUT [22]	9×9	4.062MB	30.60/0.8653	27.60/0.7541	26.86/0.7110	24.46/0.7194	27.90/0.8633
	RCLUT (Ours)	27×27	1.513MB	**30.72/0.8677**	**27.67/0.7577**	**26.95/0.7145**	**24.57/0.7253**	**28.05/0.8655**
Sparse coding	NE + LLE [5]	-	-	29.62/0.8404	26.82/0.7346	26.49/0.6970	23.84/0.6942	26.10/0.8195
	Zeyde [40]	-	-	26.69/0.8429	26.90/0.7354	26.53/0.6968	23.90/0.6962	26.24/0.8241
	ANR [33]	-	-	29.70/0.8422	26.86/0.7368	26.52/0.6992	23.89/0.6964	26.18/0.8214
	A+ [34]	-	-	30.27/0.8602	27.30/0.7498	26.73/0.7088	24.33/0.7189	26.91/0.8480
DNN	FSRCNN [9]	17×17	-	30.72/0.8660	27.61/0.7550	26.98/0.7150	24.62/0.7280	27.90/0.8610

jective. We also employ the rotation training strategy to enhance performance and the RF. Data augmentation techniques such as random flip and rotation are used to improve our model's capabilities. We train the RCLUT model using PyTorch [31] on Nvidia V100 GPUs.

Caching LUT Setting. When the model converges, we convert the RCLUT networks to a multi-LUTs format with an interval of 2^4 to reduce the size. Due to the cascade strategy, results from the first stage need to be quantized to integers, so we employ the same Re-indexing method as used in MuLUT. Additionally, to ensure that the performance of the LUTs is on par with that of the networks, we also adopt the LUT-aware Finetuning Strategy proposed by MuLUT.

4.2. Quantitative Evaluation

We compare our method with various SR methods, including 3 interpolation based methods(nearest neighbor, bilinear and bicubic interpolation), 4 sparse coding methods(NE+LLE [5], Zeyde *et al*. [40], ANR [33] and A+ [34]), 1 DNN-based method(FSRCNN [9]) and 3 LUT-based method(SR-LUT [16], MuLUT [22] and SPLUT [24]).

The quantitative results are showcased in Table 1. It can be seen that, thanks to the substantially larger RF size, our RCLUT model delivers the highest PSNR performance amongst LUT-based and sparse coding methods. Compared with the prior state-of-the-art method [22], we manage to improve the PSNR value by 0.12dB on the Set5 dataset, leveraging the benefits of the 27×27 RF size. Furthermore, RCLUT outperforms FSRCNN on the Set14 and Manga109 benchmark datasets and yields competitive performance on other datasets.

In order to compare efficiency, we also report the statistics of multiplications, additions and LUT storage sizes of LUT-based methods. As indicated in Table 1, our RCLUT model delivers superior performance but maintains a similar LUT size as SR-LUT, which is only 0.39 MB larger. Additionally, RCLUT's LUT size is $11.88\times$ smaller than that of SPLUT-L, but still achieves a 0.2dB higher PSNR value on the Set5 dataset. Even though the performance is boosted by an average of 0.11dB PSNR value, RCLUT's size is only $2.68\times$ smaller than MuLUT's.

The extensive performance and LUT size comparisons confirm that our RCLUT is a more efficient method than previous LUT-based methods, providing a better balance between accuracy and storage. The RCLUT incorporates considerably more pixel information to ensure restoration ability with less increase in size, achieved through the RC module.

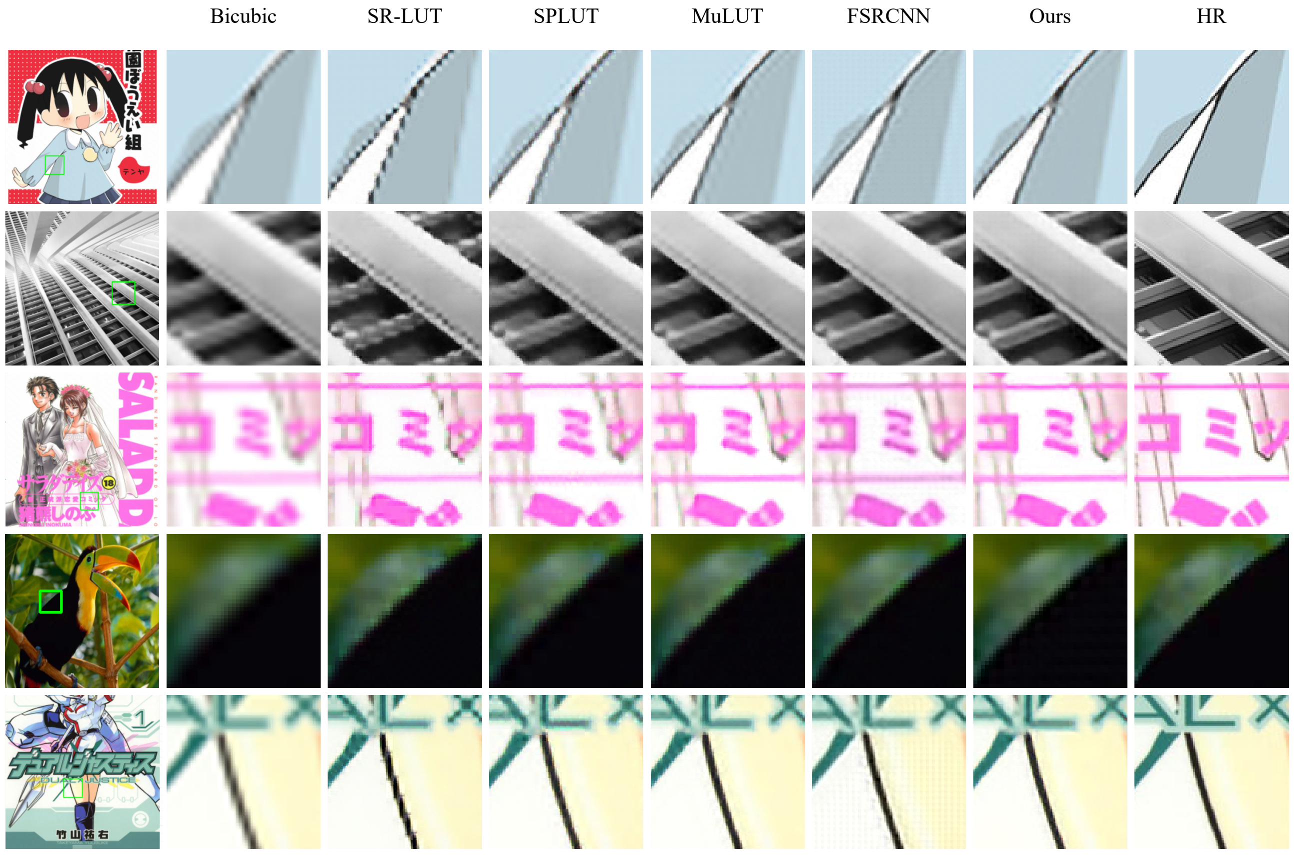


Figure 5. Visual comparison for $\times 4$ SR task on benchmark datasets. The results show our RCLUT can generate sharp edges without severe artifacts compared with other methods.

Table 2. SR-LUT and 1D-LUT size estimation when storing 8bit output for 8bit input with upscaling factor $r = 4$. SR-LUT becomes unpractical when RF is larger than 2×2.

RF	Full size SR-LUT	Sampled SR-LUT	Full size 1D-LUT
2×2	64GB	1.274MB	16KB
3×3	6.7×10^7PB	1.726TB	36KB
5×5	2.3×10^{46}PB	8.2×10^{16}PB	100KB
$n \times n$	$(2^8)^{(n^2)} \times r^2$B	$(2^4+1)^{(n^2)} \times r^2$B	$2^8 \times n^2 \times r^2$B

4.3. Qualitative Evaluation

In this section, we primarily compare our method with 3 LUT-based SR methods and the FSRCNN network. Figure 5 illustrates the visual quality of 5 cases in benchmark datasets, and additional visual results are provided in supplementary material. As observed in the first and the last example, Bicubic interpolation results in blurry outputs, SR-LUT introduces noticeable blocking artifacts due to the limited RF size, FSRCNN produces serious checkerboard artifacts, and MuLUT and SPLUT present relatively better quality than SR-LUT but still contain noise. In contrast, our method achieves more satisfactory results, illustrating sharper edges and fewer artifacts. The other three examples demonstrate that our RCLUT model recovers clearer edges and more natural textures, even when compared with the DNN method (FSRCNN). These progressive visual results verify the effectiveness of RCLUT in leveraging an exaggerated RF size.

4.4. The effectiveness of RC-Plugin Module

One additional experiment verifies that the RC method can be utilized as a plugin module, providing benefits to other LUT-based methods due to its efficiency and flexibility. We employ an RC module with a 5×5 RF size as the RC-Plugin module, termed as RC-5, and place one RC-Plugin module ahead of SR-LUT and three RC-Plugin

Table 3. The effectiveness RC-Plugin module.

Method	Set5	Set14	BSDS100	Urban100	Manga109	Volume
MuLUT [22]	30.60	27.60	26.86	24.46	27.90	4.062 MB
RC-5 + MuLUT	**30.77**	**27.71**	**26.96**	**24.58**	**28.11**	4.062 MB (+ 1.245 KB)

modules ahead of each branch in MuLUT, as illustrated in Figure 4. As demonstrated in Table 3 and the first and third rows in Table 2, when equipped with the RC-Plugin module, RC-5+SR-LUT and RC-5+MuLUT respectively gain 0.54dB and 0.17dB PSNR value over the base methods on the Set5 dataset. As indicated by the table size in Table 3 and Table 2, the LUT size increase is negligible when equipped with the RC-Plugin module. This validates that our RC-Plugin module is both effective and efficient.

4.5. Ablation Studies

In this section, we discuss the effectiveness of RCLUT networks and the implement of RC module.

The effectiveness of Linear operations in RC module. Our Reconstructed Convolution is founded on Linear and Average Pooling operations. This section will discuss the impact of Linear operations in the RC module. As depicted in Table 4, when equipped with only Average Pooling and followed by a Conv Block4-4, the method can achieve a 5×5 RF size, but it diminishes the efficacy of Conv Block4-4 (as seen in the SR-LUT baseline). Conversely, the 5×5 RC module with both Linear and Average Pooling operations, termed as RC-5, enhances the subsequent method by over 6.21dB in terms of PSNR value on the Set5 dataset. This experiment demonstrates that channel-wise features are critical to the SR model and our Linear operations provide an ability to extract deep features. Conversely, the combination of Conv Block1-4 and RC-5 exhibits performance decline compared to the model of Conv Block4-4 and RC-5. We surmise that our RC module is deficient in spatial features fusion and needs to be combined with traditional convolution blocks, like Conv Block4-4, to achieve a complementary effect.

The effectiveness of RF of RC module. We have examined the impact of receptive field (RF) size, ranging from 3×3 to 9×9, on the RCLUT model. These tests were conducted on a single-stage, single-branch network architecture for ease of implementation. As shown in Table 5, the SR-LUT augmented with a 3×3 RC module, dubbed RC-3, sees a PSNR increase of 0.48 dB on the Set5 dataset. Additionally, RF sizes of 5×5 and 7×7 respectively improve the PSNR value by 0.06dB and 0.09dB over the RC-3 module. However, an RF size of 9×9 only yields a 0.02 PSNR increase over the RC-7. These results suggest that as the RF size increases, our RC module continues to enhance performance up to an inflection point around 7×7. This explains our choice of 7×7 as the maximum RF size in the RC Blocks.

The effectiveness of cascade and parallel branches in RCLUT. We experimented with single-stage models incorporating varying RC modules and Conv Block4-4 configurations. RCLUT-3 refers to the single-stage model with a 3×3 RC module, while RCLUT-3_5 represents a single-stage model with two branches comprising 3×3 and 5×5 RC Blocks. As presented in Table 6, it is observed that a multi-branch architecture enhances the performance of the RCLUT model. In particular, RCLUT-3_5_7 achieves a 0.34 dB higher PSNR value than RCLUT-3 on the Set5 dataset. Interestingly, RCLUT-5_7_9 exhibits comparable performance to RCLUT-3_5_7, which suggests that a 7×7 RF size is sufficient for our RCLUT model, aligning with the findings in the previous section. Moreover, the two-stage cascaded model RCLUT-3_5_7-$X \times 2$ shows a 0.04dB PSNR performance improvement on the Set5 dataset compared to the single-stage model RCLUT-5_7_9. These experiments underscore the effectiveness of the cascading and parallel branches architecture adopted in RCLUT.

The inference speed of LUT-based methods. In our experiments, we generate a high-definition (HD) image of 1280×720 using a super-resolution (SR) factor of 4x. These tests were conducted following the same setup as described in the MuLUT [22] research. For these tests, we use Ω to denote the RC plug-in module which has a receptive field size of 5×5. For EDSR, we implement it using the CPU-version of the PyTorch library, with its runtime being measured on a MacBook Pro that has a 2.6 GHz Intel Core i7 processor. For all the other methods, their runtime was measured on an IQOO Neo5 smartphone.

We'd like to highlight that typically, processors found in personal computers, such as laptops, are generally more powerful than those found in mobile devices like smartphones. Therefore, when analyzing the results, it's important to consider the capabilities of the hardware being used for the testing.

5. Conclusion

In this paper, we propose a Reconstructed Convolution method, which decouples the calculation of spatial and channel-wise feature. The RC method can be formulated as multi-1D LUTs to maintain large RF with less LUT size. Moreover, We propose a RCLUT model based on RC Blocks. It achieves significant performance with slight increasing LUT size. On the other hand, our RC method can be used as a Plugin module to improve the ability of LUT-

Table 4. Ablation study of RC module.

Avg Pooling 5x5	RC-5	Conv Block4-4	Conv Block1-4	Set5	Set14	BSDS100	Urban100	Manga109	Volume
		✓		29.82	27.01	26.53	24.02	26.80	1.274 MB
✓		✓		24.19	23.09	23.71	20.78	21.88	1.274 MB
	✓		✓	29.49	26.80	26.44	23.77	26.09	0.83 KB
	✓	✓		**30.40**	**27.40**	**26.77**	**24.26**	**27.33**	1.274 MB

Table 5. Ablation study of receive field size on RC.

Method	Set5	Set14	BSDS100	Urban100	Manga109	Volume
SR-LUT [16]	29.82	27.01	26.53	24.02	26.80	1.274 MB
RC-3 + SR-LUT	30.34	27.37	26.75	24.24	27.28	1.274 MB(+0.149 KB)
RC-5 + SR-LUT	30.40	27.40	26.77	24.26	27.33	1.274 MB(+0.415 KB)
RC-7 + SR-LUT	30.43	27.44	26.80	24.31	27.44	1.274 MB(+0.813 KB)
RC-9 + SR-LUT	30.45	27.46	26.81	24.33	27.48	1.274 MB(+1.345 KB)

Table 6. Ablation study of multi-branch on RC.

Method	Set5	Set14	BSDS100	Urban100	Manga109	Volume
RCLUT-3	30.34	27.37	26.75	24.24	27.28	1.274 MB
RCLUT-3_5	30.57	27.57	26.89	24.46	27.68	2.548 MB
RCLUT-5_7	30.64	27.63	26.93	24.53	27.85	2.548 MB
RCLUT-3_5_7	30.68	27.64	26.94	24.55	27.90	3.822 MB
RCLUT-5_7_9	30.68	27.63	26.94	24.56	27.93	3.822 MB
RCLUT-3_5_7-X2	**30.72**	**27.67**	**26.95**	**24.57**	**28.05**	1.513 MB

Table 7. Runtime of different LUT-based methods.

	LUT					DNN
	SR-LUT	SR-LUT$^{\Omega}$	MuLUT	MuLUT$^{\Omega}$	RCLUT	EDSR
Runtime (ms)	152	157(+3%)	253	266(+5%)	232	8083

based SR methods. Extensive experiments show that our RCLUT is a new the-state-of-art LUT-method for SR task, it works much more efficiently than other prior methods.

6. Acknowledgments

This work was supported by the NSFC under Grant 62072271.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPR*, pages 126–135, 2017.

[2] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *ECCV*, pages 252–268, 2018.

[3] Mustafa Ayazoglu. Extremely lightweight quantization robust real-time single-image super resolution for mobile devices. In *CVPR*, pages 2472–2479, 2021.

[4] Chenglong Bao, Jian-Feng Cai, and Hui Ji. Fast sparsity-based orthogonal dictionary learning for image restoration. In *ICCV*, pages 3384–3391, 2013.

[5] Hong Chang, Dit-Yan Yeung, and Yimin Xiong. Super-resolution through neighbor embedding. In *CVPR*, pages I–I, 2004.

[6] Chang Chen, Zhiwei Xiong, Xinmei Tian, Zheng-Jun Zha, and Feng Wu. Camera lens super-resolution. In *CVPR*, pages 1652–1660, 2019.

[7] Zhen Cheng, Zhiwei Xiong, Chang Chen, Dong Liu, and Zheng-Jun Zha. Light field super-resolution with zero-shot learning. In *CVPR*, pages 10010–10019, 2021.

[8] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199, 2014.

[9] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, pages 391–407, 2016.

[10] Azuma Fujimoto, Toru Ogawa, Kazuyoshi Yamamoto, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. Manga109 dataset and creation of metadata. In *Proceedings of the 1st international workshop on comics analysis, processing and understanding*, pages 1–5, 2016.

[11] Jinjin Gu and Chao Dong. Interpreting super-resolution networks with local attribution maps. In *CVPR*, pages 9199–9208, 2021.

[12] Shuhang Gu, Wangmeng Zuo, Qi Xie, Deyu Meng, Xiangchu Feng, and Lei Zhang. Convolutional sparse coding for image super-resolution. In *ICCV*, pages 1823–1831, 2015.

[13] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *CVPR*, pages 1664–1673, 2018.

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017.

[15] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, pages 5197–5206, 2015.

[16] Younghyun Jo and Seon Joo Kim. Practical single-image super-resolution using look-up table. In *CVPR*, pages 691–700, 2021.

[17] Robert Keys. Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, pages 1153–1160, 1981.

[18] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, pages 1646–1654, 2016.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*, 2014.

[20] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, pages 624–632, 2017.

[21] Huixia Li, Chenqian Yan, Shaohui Lin, Xiawu Zheng, Baochang Zhang, Fan Yang, and Rongrong Ji. Pams: Quantized super-resolution via parameterized max scale. In *ECCV*, pages 564–580, 2020.

[22] Jiacheng Li, Chang Chen, Zhen Cheng, and Zhiwei Xiong. Mulut: Cooperating multiple look-up tables for efficient image super-resolution. In *ECCV*, pages 238–256, 2022.

[23] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR*, pages 136–144, 2017.

[24] Cheng Ma, Jingyi Zhang, Jie Zhou, and Jiwen Lu. Learning series-parallel lookup tables for efficient image super-resolution. In *ECCV*, pages 305–321, 2022.

[25] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, pages 416–423, 2001.

[26] Yiqun Mei, Yuchen Fan, Yuqian Zhou, Lichao Huang, Thomas S Huang, and Honghui Shi. Image super-resolution with cross-scale non-local attention and exhaustive self-exemplars mining. In *CVPR*, pages 5690–5699, 2020.

[27] Ben Niu, Weilei Wen, Wenqi Ren, Xiangde Zhang, Lianping Yang, Shuzhen Wang, Kaihao Zhang, Xiaochun Cao, and Haifeng Shen. Single image super-resolution via a holistic attention network. In *ECCV*, pages 191–207, 2020.

[28] Yaniv Romano, John Isidoro, and Peyman Milanfar. Raisr: rapid and accurate image super resolution. *IEEE Transactions on Computational Imaging*, pages 110–125, 2016.

[29] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, pages 1874–1883, 2016.

[30] Dehua Song, Yunhe Wang, Hanting Chen, Chang Xu, Chunjing Xu, and DaCheng Tao. Addersr: Towards energy efficient image super-resolution. In *CVPR*, pages 15648–15657, 2021.

[31] Eli Stevens, Luca Antiga, and Thomas Viehmann. *Deep learning with PyTorch*. Manning Publications, 2020.

[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015.

[33] Radu Timofte, Vincent De Smet, and Luc Van Gool. Anchored neighborhood regression for fast example-based super-resolution. In *ICCV*, pages 1920–1927, 2013.

[34] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, pages 111–126, 2015.

[35] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *ECCV*, pages 0–0, 2018.

[36] Yi-Qing Wang. A multilayer neural network for image demosaicking. In *ICIP*, pages 1852–1856, 2014.

[37] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. In *ICIP*, pages 600–612, 2004.

[38] Zeyu Xiao, Zhiwei Xiong, Xueyang Fu, Dong Liu, and Zheng-Jun Zha. Space-time video super-resolution using temporal profiles. In *ACM Multimedia*, pages 664–672, 2020.

[39] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. In *ICIP*, pages 2861–2873, 2010.

[40] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International Conference on Curves and Surfaces*, pages 711–730. Springer, 2012.

[41] Haochen Zhang, Dong Liu, and Zhiwei Xiong. Two-stream action recognition-oriented video super-resolution. In *ICCV*, pages 8799–8808, 2019.

[42] Xindong Zhang, Hui Zeng, and Lei Zhang. Edge-oriented convolution block for real-time super resolution on mobile devices. In *ACM Multimedia*, pages 4034–4043, 2021.

[43] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, pages 286–301, 2018.

[44] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, pages 2472–2481, 2018.