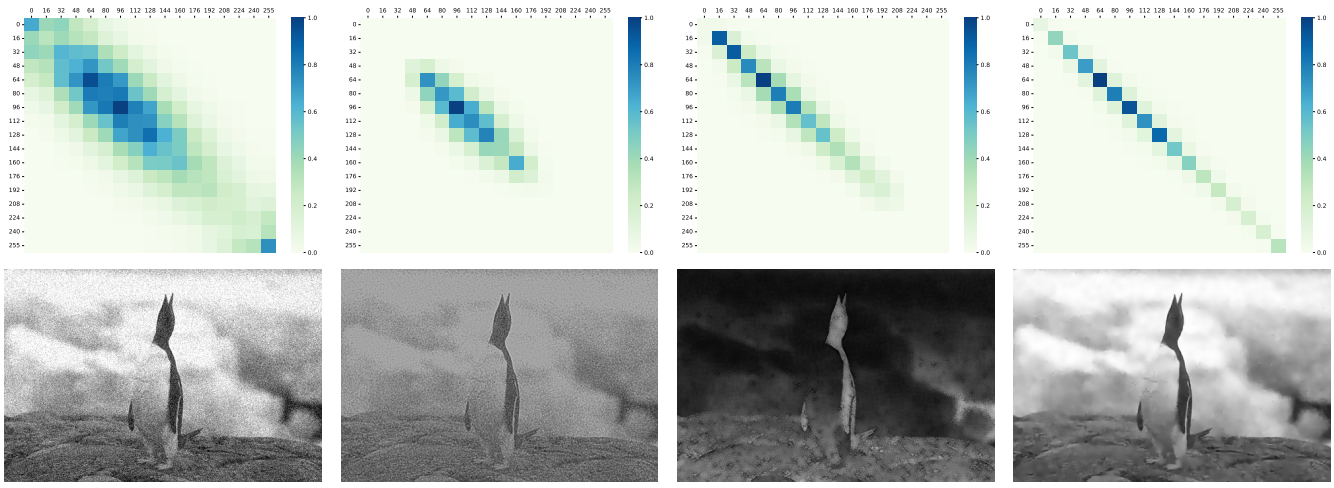
Look-Up Table Compression for Efficient Image Restoration (Supplementary Material)

Yinglong Li Jiacheng Li Zhiwei Xiong*
University of Science and Technology of China
{yllee, jclee}@mail.ustc.edu.cn zwxiong@ustc.edu.cn



Noisy Image *1st stage feature* *2nd stage feature* *Denoised Image*

Figure 1. The visualizations of the occurrence statistics of adjacent pixel pairs in noisy images, intermediate features, and denoised images, which are tested on the BSD68 dataset [16] at a noise level of 25 using SPF-LUT. *Noisy Image*: the input image of SPF-LUT. *1st stage feature* and *2nd stage feature*: the feature maps output by intermediate LUT groups in SPF-LUT. *Denoised Image*: the output of SPF-LUT.

In this supplement material, we include the following sections:

- additional analysis on the diagonal-dominance property (Sec. 1),
- more implementation details of SPF-LUT (Sec. 2),
- partial compression of LUT-based methods (Sec. 3),
- comparison with SPLUT (Sec. 4),
- comparison with an alternative solution (Sec. 5),
- efficiency evaluation of DFC framework and discussion on deployment of LUTs (Sec. 6),
- additional results of image restoration (Sec. 7).

1. Additional Analysis on the Diagonal-Dominance Property

We make an additional analysis by collecting retrieval statistics on feature maps output by intermediate LUT groups of SPF-LUT for grayscale image denoising. First, we obtain the feature maps outputted by intermediate LUT groups of SPF-LUT on the BSD68 dataset at a noise level of 25. Then, we count the occurrence frequency of pairs of two spatially adjacent pixels, which is the occurrence frequency of coordinate indexes of LUTs in the next LUT group. As shown in Fig. 1, the occurrence frequency of adjacent pixel pairs on input images (*Noisy Image*), features (*1st stage feature* and *2nd stage feature*), and output images (*Denoised Image*) all exhibit a diagonal distribution. These results show that not only low-quality data but also intermediate features learned by cascaded LUTs follow the diagonal-dominance property. Furthermore, the change in the diagonal distribution provides another interesting perspective for us to understand the denoising process. The wide width of the diagonal distribution on *Noisy Image* is caused by noise points. The denoising process narrows the width of the diagonal distribution on *1st stage feature* and *2nd stage feature*, and presents an extremely narrow diagonal distribution in the final *Denoised Image*. Thus, the grad-

*Corresponding author.

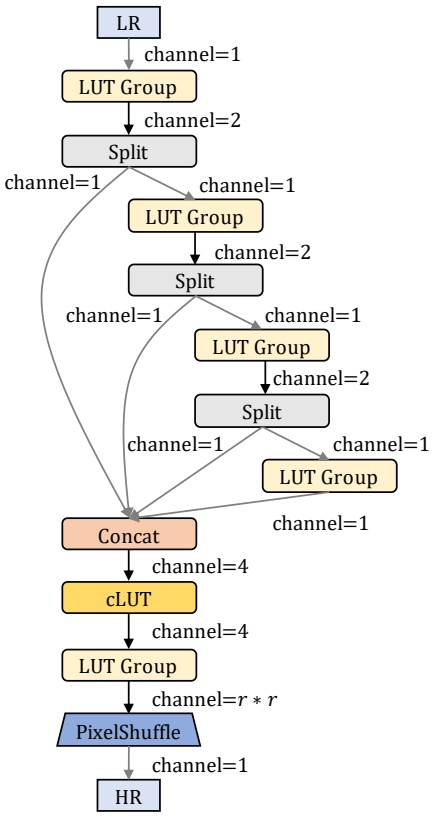


Figure 2. The architecture of SPF-LUT, where the *cLUT* denotes a channel-wise LUT, and r means the scaling factor.

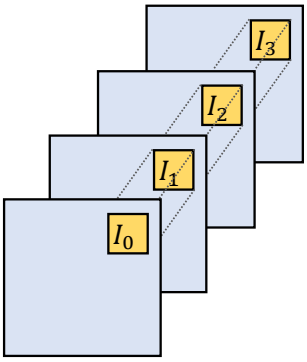


Figure 3. Channel-wise LUT with four dimensions of index. Similar to color-to-color LUTs [27], a channel-wise LUT performs channel-wise indexing by using pixels across different channels as indexes.

ually decreasing width of the diagonal distribution means that the noise points are gradually removed, and the clear image is gradually restored.

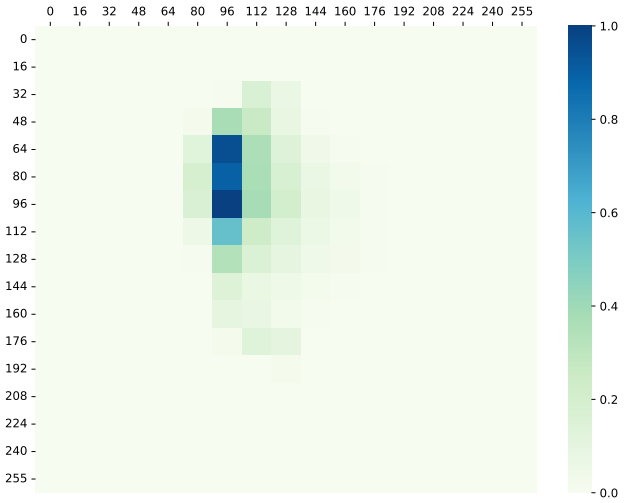


Figure 4. The visualization of the occurrence statistics of pixel pairs across the second channel and third channel of the intermediate four-channel features, which are obtained on the BSD68 dataset [16] for grayscale image denosing. The darker color means a higher occurrence frequency.

2. More Details about SPF-LUT

The detailed architecture of SPF-LUT for super-resolution is shown in Fig. 2. In the first three LUT groups, each LUT group outputs two-channel feature maps, which are split into two parts. One part is left for concatenation, and the other part is fed to the next LUT group. The fourth LUT group only outputs a feature map with one channel. Then, the concatenated four-channel feature map is fed into a channel-wise LUT, as illustrated in Fig. 3, and a four-channel output is obtained. We use one LUT group for each channel of the four-channel feature and average the feature maps outputted by these four LUT groups together to obtain a feature with $r * r$ channels, where r is the scaling factor. Finally, the r^2-channel feature is fed to a PixelShuffle layer [20] to obtain a high-resolution (HR) result. We adapt the proposed SPF-LUT to denoising, deblocking, and deblurring by removing the PixelShuffle layer of the architecture in Fig. 2.

In order to take the channel-wise LUT into account when calculating the compression ratio (CR) of SPF-LUT, we collect the occurrence frequency of pixel pairs across the second channel and the third channel of the input features of the channel-wise LUT. As shown in Fig. 4, the occurrence frequency of pixel pairs across channels violates the diagonal-dominance property. Thus, the channel-wise LUT is not compressed by DFC. CR of SPF-LUT is calculated as

$$CR = \frac{S_{DFC} + S_{channel}}{S}, \tag{1}$$

where S_{DFC} is the storage size of compressed spatial-wise

Table 1. Quantitative comparison of PSNR and SSIM on standard benchmark datasets for $\times 4$ super-resolution tasks between the original version of RCLUT [14] and the compressed version using DFC.

	Storage Size	*Set5*	*Set14*	*BSDS100*	*Urban100*	*Manga109*
RCLUT [14]	1.513MB	30.61/0.8652	27.55/0.7539	26.84/0.7111	24.40/0.7174	27.78/0.8603
RCLUT +DFC	0.200MB	30.54/0.8634	27.50/0.7512	26.80/0.7087	24.35/0.7153	27.66/0.8583

Table 2. Quantitative comparison of PSNR and SSIM on standard benchmark datasets for $\times 4$ super-resolution tasks between SPLUT [15] and SPF-LUT.

	Storage Size	*Set5*	*Set14*	*BSDS100*	*Urban100*	*Manga109*
SPLUT-L [15]	18.000MB	30.52/0.8631	27.54/0.7520	26.87/0.7091	24.46/0.7191	27.70/0.8581
SPLUT-S [15]	5.500MB	30.01/0.8520	27.20/0.7430	26.68/0.7020	24.13/0.7060	27.00/0.8430
SPF-LUT	17.284MB	31.11/0.8764	27.92/0.7640	27.10/0.7197	24.87/0.7378	28.68/0.8796
SPF-LUT +DFC	2.018MB	31.05/0.8755	27.88/0.7632	27.08/0.7190	24.81/0.7357	28.58/0.8779

Table 3. Comparison with the direct subsampling at a similar compression ratio, where the direct subsampling is denoted as +DS, and "ft." means the LUT-aware finetuning strategy.

Method	*Set5*		*Set14*	
	PSNR	SSIM	PSNR	SSIM
SPF-LUT	31.11	0.8764	27.92	0.7640
SPF-LUT +DFC w/o ft.	30.49	0.8646	27.55	0.7543
SPF-LUT +DFC w/ ft.	31.05	0.8755	27.88	0.7632
SPF-LUT +DS w/o ft.	30.06	0.8471	27.37	0.7455
SPF-LUT +DS w/ ft.	30.95	0.8733	27.80	0.7613

LUTs in SPF-LUT, S is the storage size of the uncompressed original SPF-LUT, and $S_{channel}$ is the storage size of the uncompressed channel-wise LUT. In our SPF-LUT, the storage size of the channel-wise LUT is only 0.319MB. Thus, a channel-wise LUT is only a small fraction of all LUTs in the SPF-LUT, and the storage size of a channel-wise LUT only accounts for 1.8% (0.319MB/17.284MB) of the total storage size of the SPF-LUT for $\times 4$ super-resolution.

3. Partial Compression of LUT-based Methods

We further evaluate the performance of using our DFC to compress the latest LUT-based model, RCLUT [14]. The RCLUT introduces a RC module to increase the receptive field size. The RC module comprises n^2 1-dimensional (1D) LUTs, with each 1D LUT dedicated to processing each individual pixel within an $n \times n$ image patch. Ultimately, the output values from all the 1D LUTs are averaged to produce the output of the RC module. The RCLUT employs a parallel and cascading structure, similar to MuLUT [11], to leverage the capabilities of multiple spatial-wise LUTs, positioning RC modules ahead of each spatial-wise LUT.

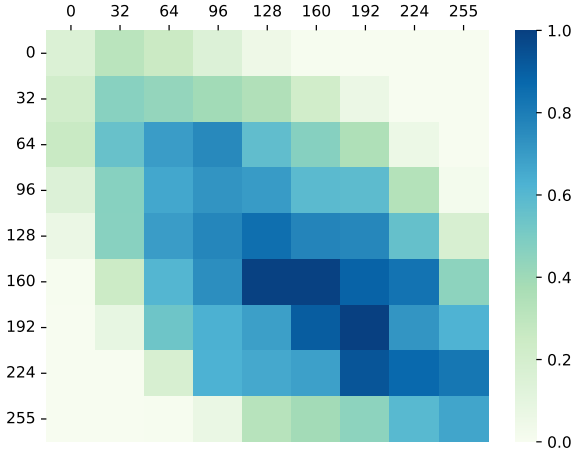


Figure 5. Visualization of the magnitude of value changes of SPF-LUT +DS after LUT-aware finetuning strategy. The change in values after fine-tuning is also distributed diagonally.

Since 1D LUTs in the RC module disrupt the interaction between adjacent pixels, resulting in the absence of the diagonal-dominance property of 1D LUTs, here we only apply our DFC to spatial-wise LUTs with patches of size 2×2 as inputs in RCLUT. We report the PSNR and SSIM results for $\times 4$ super-resolution on the five standard benchmarks in Table 1. As shown in Table 1, RCLUT +DFC can achieve comparable performance to the original uncompressed version, which validates that our DFC framework can be applied to more spatial-wise LUT-based models.

4. Comparison with SPLUT

SPLUT [15] is another baseline model different from SR-LUT [10] and MuLUT [11], which processes different im-

Table 4. The comparison of efficiency and performance between different models. We evaluate the energy cost and peak memory of generating 1280×720 high-quality images with $\times 4$ super resolution. We also report the storage size of the libraries that different models are dependent on. The PSNR is measured on standard benchmark datasets for $\times 4$ super-resolution.

	Method	int8 Add.	int8 Mul.	int32 Add.	int32 Mul.	float32 Add.	float32 Mul.	Energy Cost(pJ)	Peak Memory	Dependent Library Size	Set14 PSNR	BSDS100 PSNR
Classical	Bicubic					14.7M	14.7M	67.8M	2.3MB	numpy: 16.5MB	26.00	25.96
LUT	SR-LUT [10]	15.8M	0.1M	28.6M	22.3M			72.5M	47.0MB	numpy: 16.5MB	27.18	26.59
	SR-LUT [10] +DFC	17.2M	0.1M	28.6M	22.3M			72.5M	40.5MB	numpy: 16.5MB	27.14	26.57
	MuLUT [11]	53.3M	0.2M	93.0M	71.8M			233.6M	50.7MB	numpy: 16.5MB	27.60	26.86
	MuLUT [11] +DFC	61.6M	0.2M	93.0M	71.8M			233.9M	41.3MB	numpy: 16.5MB	27.56	26.83
	SPF-LUT	222.9M	1.0M	390.1M	301.5M			980.5M	65.4MB	numpy: 16.5MB	27.92	27.10
	SPF-LUT +DFC	256.1M	1.0M	390.1M	301.5M			981.5M	45.9MB	numpy: 16.5MB	27.88	27.08
DNN	RRDB [25]					1.0T	1.0T	4.7T	843.6MB	torch(CPU): 186.3MB	28.88	27.82
	EDSR [13]					2.9T	2.9T	13.3T	2.3GB	torch(CPU): 186.3MB	28.80	27.71

Table 5. Operational energy costs for different data types.

Operation	int8	int32	float16	float32
Add.(pJ)	0.03	0.1	0.4	0.9
Mult.(pJ)	0.2	3.1	1.1	3.7

age information separately with a series-parallel structure. Here, we report the comparison of SPF-LUT and SPF-LUT +DFC with the two versions of SPLUT in Table 2. As listed in Table 2, with our DFC, SPF-LUT shows significant advantages in the performance-storage trade-off. For example, SPF-LUT +DFC outperforms SPLUT-S (the lightweight version of SPLUT) by 1.58dB on Manga109, with 60% less storage.

5. Comparison with an Alternative Solution

An alternative way to compress spatial-wise LUTs is to directly subsample the LUTs at a large sampling interval. We use direct subsampling to compress SPF-LUT at a compression ratio of about 10%, which is denoted as SPF-LUT +DS, and compare it with SPF-LUT +DFC under two settings, with and without LUT-aware finetuning. The two settings are denoted as w/o ft. and w/ ft. in Table 3, respectively.

As observed, our DFC framework obtains superior performance over the +DS solution in both settings, showing that the information along the diagonal of spatial-wise LUTs plays a key role in maintaining performance. Furthermore, it is worth noting that the LUT-aware finetuning strategy reduces the performance gap between SPF-LUT +DFC and SPF-LUT +DS. Without finetuning, the difference between the two models is 0.45dB on the Set5 $\times 4$ dataset, while the difference is about 0.1dB with LUT-aware finetuning. Here, we visualize the value changes of spatial-wise LUTs in SPF-LUT +DS after LUT-aware finetuning. As shown in Fig. 5, the dark color means the significant value change after LUT-aware finetuning, which is mainly distributed along the diagonal, similar to the diagonal-dominance property we observe in this paper. Thus, the LUT-aware finetuning strategy obtains performance gains by mainly finetuning the values cached in the diagonal LUT cells, which is consistent with our observation.

6. Efficiency Evaluation of DFC Framework and Discussion on Deployment of LUTs

Following MuLUT [11], we conduct an assessment to estimate the theoretical energy costs [21] associated with compressed LUT models for super-resolution tasks, and we also include Bicubic, RRDB [25], and EDSR [13] as references. As listed in Table 4, the results highlight a crucial distinction. LUT-based models perform only integer-type operations, affording them a significant computational efficiency edge over DNN models. Notably, our proposed DFC framework sustains this advantage and reduces the storage size. The costs associated with operational energy, segregated by data types, are itemized in Table 5, derived from literatures [8, 22].

The peak memory is measured with memray[1] on CPU. As listed in Table 4, LUT-based models have lower peak memory than DNN models at runtime, implying less runtime resource requirements for the LUT-based models.

The storage size of the dependent library listed in Table 4 indicates the enormous additional storage overhead required by DNN models compared to LUT models.

Moreover, we also have a discussion on the deployment advantages of the LUT models. The deployment comparison between DNNs and LUTs is important and has been extensively discussed in terms of running time and energy cost in existing works [10, 11, 15]. Here, we provide another implementation perspective. As listed in Table 6, we report the comparison of efficiency and performance between SPF-LUT and a lightweight DNN model ARCNN [5, 26] in the image deblocking task. Both ARCNN and its fast version require more energy, running memory, and dependent library size. For mobile and other edge devices, LUT models

[1] https://github.com/bloomberg/memray

Table 6. The comparison of efficiency and performance between SPF-LUT and ARCNN [5, 26] for image deblocking under a quality factor of 10. We evaluate the energy cost and peak memory of generating 320×180 high-quality images.

	Method	Storage Size	Energy Cost (pJ)	Peak Memory	Dependent Library Size	LIVE1 PSNR-B
LUT	SPF-LUT	3017.849KB	160.6M	19.8MB	numpy: 16.5MB	28.62
	SPF-LUT +DFC	595.926KB	161.6M	16.7MB	numpy: 16.5MB	28.61
DNN	ARCNN [5]	415.812KB	28921.9M	116.7MB	torch(CPU): 186.3MB	28.77
	Fast-ARCNN [26]	232.761KB	3948.7M	84.1MB	torch(CPU): 186.3MB	28.65

Table 7. Quantitative comparison of PSNR/SSIM on standard benchmark datasets for $\times 2$ and $\times 3$ super-resolution. The gray background means LUT-based models are compressed using DFC. For LUT-based models, the best and second-best results are depicted with red and blue, respectively.

PSNR/SSIM	*Method*	*Sacle*	*Set5*	*Set14*	*BSDS100*	*Urban100*	*Manga109*
Classical	Bicubic	$\times 2$	33.63/0.9292	30.23/0.8681	29.53/0.8421	26.86/0.8394	30.78/0.9338
	NE + LLE [2]	$\times 2$	35.79/0.9491	31.82/0.8996	30.77/0.8787	28.48/0.8803	33.95/0.9590
	ANR [23]	$\times 2$	35.85/0.9500	31.86/0.9006	30.82/0.8800	28.49/0.8807	33.94/0.9597
	A+ [24]	$\times 2$	36.57/0.9545	32.34/0.9056	31.21/0.8860	29.23/0.8938	35.32/0.9670
LUT	SR-LUT [10]	$\times 2$	35.74/0.9489	31.87/0.8978	30.74/0.8767	28.65/0.8808	34.02/0.9598
	SR-LUT [10] +DFC	$\times 2$	35.64/0.9482	31.81/0.8970	30.69/0.8759	28.57/0.8794	33.98/0.9592
	MuLUT [11]	$\times 2$	36.65/0.9541	32.49/0.9065	31.23/0.8865	29.31/0.8910	35.78/0.9674
	MuLUT [11] +DFC	$\times 2$	36.53/0.9533	32.38/0.9054	31.16/0.8853	29.18/0.8889	35.56/0.9662
	SPF-LUT	$\times 2$	37.08/0.9562	32.79/0.9095	31.48/0.8902	29.76/0.8984	36.50/0.9706
	SPF-LUT +DFC	$\times 2$	37.06/0.9563	32.75/0.9092	31.45/0.8901	29.70/0.8978	36.42/0.9703
DNN	EDSR [13]	$\times 2$	38.11/0.9602	33.92/0.9195	32.32/0.9013	32.93/0.9351	39.10/0.9773
	RCAN [31]	$\times 2$	38.30/0.9617	34.14/0.9235	32.41/0.9025	33.17/0.9377	39.60/0.9791
Classical	Bicubic	$\times 3$	30.40/0.8678	27.55/0.7736	27.20/0.7379	24.45/0.7343	26.94/0.8554
	NE + LLE [2]	$\times 3$	31.87/0.8958	28.64/0.8085	27.92/0.7727	25.41/0.7755	28.70/0.8889
	ANR [23]	$\times 3$	31.95/0.8970	28.69/0.8102	27.96/0.7745	25.45/0.7768	28.78/0.8900
	A+ [24]	$\times 3$	32.63/0.9090	29.16/0.8190	28.28/0.7832	26.04/0.7974	29.90/0.9099
LUT	SR-LUT [10]	$\times 3$	32.10/0.9002	28.92/0.8102	27.99/0.7726	25.66/0.7808	29.54/0.9024
	SR-LUT [10] +DFC	$\times 3$	32.02/0.8988	28.86/0.8091	27.95/0.7717	25.60/0.7787	29.45/0.9009
	MuLUT [11]	$\times 3$	32.75/0.9089	29.34/0.8215	28.31/0.7841	26.10/0.7945	30.72/0.9161
	MuLUT [11] +DFC	$\times 3$	32.75/0.9088	29.31/0.8210	28.28/0.7835	26.06/0.7933	30.67/0.9153
	SPF-LUT	$\times 3$	33.37/0.9168	29.70/0.8290	28.59/0.7919	26.64/0.8122	31.65/0.9279
	SPF-LUT +DFC	$\times 3$	33.33/0.9162	29.66/0.8287	28.57/0.7917	26.59/0.8110	31.57/0.9272
DNN	EDSR [13]	$\times 3$	34.65/0.9280	30.52/0.8462	29.25/0.8093	28.80/0.8653	34.17/0.9476
	RCAN [31]	$\times 3$	34.78/0.9299	30.63/0.8477	29.33/0.8107	29.02/0.8695	34.58/0.9502

can be implemented with standard JAVA, while Pytorch Android alone takes ∼90MB[2], let alone the possible requirement of GPUs. From the perspective of energy costs, peak memory, and dependent library size, LUT models are advantageous over DNNs for deployment on resource-limited edge devices.

In summary, the DFC framework effectively preserves the computational efficiency inherent to original LUT models, enabling LUT models to be more practical by releasing their storage requirements.

[2] https://oss.sonatype.org/#nexus-search;quick~pytorch_android

7. Additional Results of Image Restoration

Image Super-Resolution. We report the quantitative results for $\times 2$ and $\times 3$ super-resolution on five standard benchmarks [1, 9, 16, 17, 28] in Table 7. Since the RRDB [25] has no reference versions for $\times 2$ and $\times 3$ super-resolution tasks, following SR-LUT [10] and MuLUT [11], we report the results of RCAN [31] instead of RRDB. The visual comparisons of $\times 2$ and $\times 3$ super-resolution tasks on standard benchmarks are provided in Fig. 6 and Fig. 7, re-

Table 8. The comparison of PSNR on standard benchmark datasets for grayscale image denoising at noise levels of 25 and 50. The gray background means LUT-based models are compressed using DFC. For LUT-based models, the best and second-best results are depicted with red and blue, respectively.

PSNR	*Method*	*Set12*		*BSD68*	
		25	50	25	50
LUT	SR-LUT [10]	27.19	22.62	26.85	22.39
	SR-LUT [10] +DFC	27.16	22.59	26.66	22.36
	MuLUT [11]	28.94	25.46	28.18	24.97
	MuLUT [11] +DFC	28.72	25.17	27.90	24.68
	SPF-LUT	29.49	26.14	28.55	25.55
	SPF-LUT +DFC	29.43	26.07	28.50	25.50
Classical	BM3D [4]	29.97	26.72	28.57	25.62
	WNNM [7]	30.26	27.05	28.83	25.87
	TNRD [3]	30.06	26.81	28.92	25.97
DNN	DnCNN [29]	30.44	27.18	29.23	26.23
	FFDNet [30]	30.43	27.32	29.19	26.29
	SwinIR [12]	31.01	27.91	29.50	26.58

spectively. We also provide more visual results for the $\times 4$ super-resolution task in Fig. 8.

Image Denoising. We report the quantitative results for grayscale image denoising at a noise level of 25 and 50 on standard benchmarks [16, 29] in Table 8. We provide more visual results for the noise level of 15 in Fig. 9. The visual comparisons for the noise levels of 25 and 50 on standard benchmarks are provided in Fig. 10 and Fig. 11.

Image Deblocking. Image deblocking is an image processing task that aims to reduce or eliminate blocking artifacts in images caused by compression algorithms such as JPEG. These artifacts typically appear in edge and texture regions of the image, leading to a degradation in visual quality. The goal of deblocking is to reduce or eliminate these artifacts through filtering or other processing of the image, resulting in an improvement in the visual quality of the image. We report the quantitative results for image deblocking under different quality factors (20, 30, and 40) on standard benchmarks [6, 19] in Table 9. The metric is PSNR-B. The visual comparisons for the quality factor of 10, 20, and 30 on standard benchmarks are provided in Fig. 12, Fig. 13, and Fig. 14, respectively.

Image Deblurring. Image deblurring refers to the process of restoring a blurred image. Blurring may be caused by motion blur due to camera or handheld device movement, as well as out-of-focus blur resulting from imperfect lenses or incorrect focus distances. The goal of image deblurring is to restore the clarity and details of the original image, making it more clear and recognizable. For image deblurring, we train the LUT-based models and evaluate them on the GoPro dataset [18]. The GoPro dataset is a large-scale dataset created to simulate real-world blur by frame averaging, which has been used to train and evaluate deep deblurring algorithms. The dataset is comprised of 3214 image pairs, which are divided into training and test sets, with 2103 pairs and 1111 pairs, respectively. The visual comparisons are provided in Fig. 15.

Table 9. The comparison of PSNR-B on standard benchmark datasets for image deblocking under different quality factors (QF). The gray background means LUT-based models are compressed using DFC. For LUT-based models, the best and second-best results are depicted with red and blue, respectively.

	Method	*QF*	*Classic5*	*LIVE1*
LUT	SR-LUT [10]	20	29.54	29.74
	SR-LUT [10] +DFC	20	29.48	29.68
	MuLUT [11]	20	30.30	30.52
	MuLUT [11] +DFC	20	30.22	30.38
	SPF-LUT	20	30.71	30.86
	SPF-LUT +DFC	20	30.65	30.80
Classical	JPEG	20	27.50	27.56
	SA-DCT [6]	20	29.75	29.82
DNN	ARCNN [5]	20	30.59	30.79
	SwinIR [12]	20	31.99	31.70
LUT	SR-LUT [10]	30	30.80	30.99
	SR-LUT [10] +DFC	30	30.77	30.94
	MuLUT [11]	30	31.57	31.79
	MuLUT [11] +DFC	30	31.36	31.58
	SPF-LUT	30	31.96	32.11
	SPF-LUT +DFC	30	31.95	32.10
Classical	JPEG	30	28.94	28.92
	SA-DCT [6]	30	30.82	30.91
DNN	ARCNN [5]	30	31.98	32.38
	SwinIR [12]	30	33.03	33.01
LUT	SR-LUT [10]	40	31.71	32.00
	SR-LUT [10] +DFC	40	31.62	31.91
	MuLUT [11]	40	32.32	32.66
	MuLUT [11] +DFC	40	32.16	32.51
	SPF-LUT	40	32.83	33.08
	SPF-LUT +DFC	40	32.75	33.02
Classical	JPEG	40	29.92	29.95
	SA-DCT [6]	40	31.58	31.77
DNN	ARCNN [5]	40	32.79	33.14
	SwinIR [12]	40	33.66	33.88

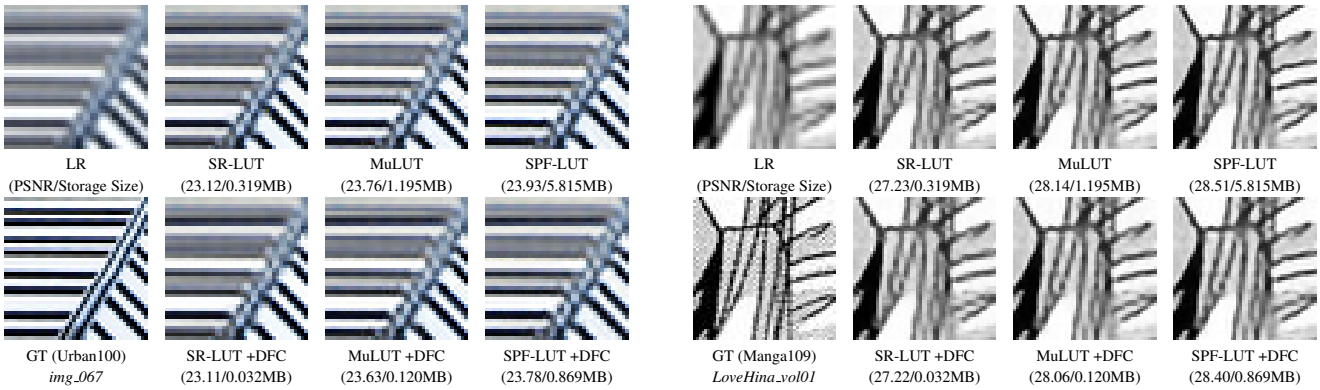


Figure 6. Qualitative comparison for $\times 2$ super-resolution on standard benchmark datasets [9, 17].

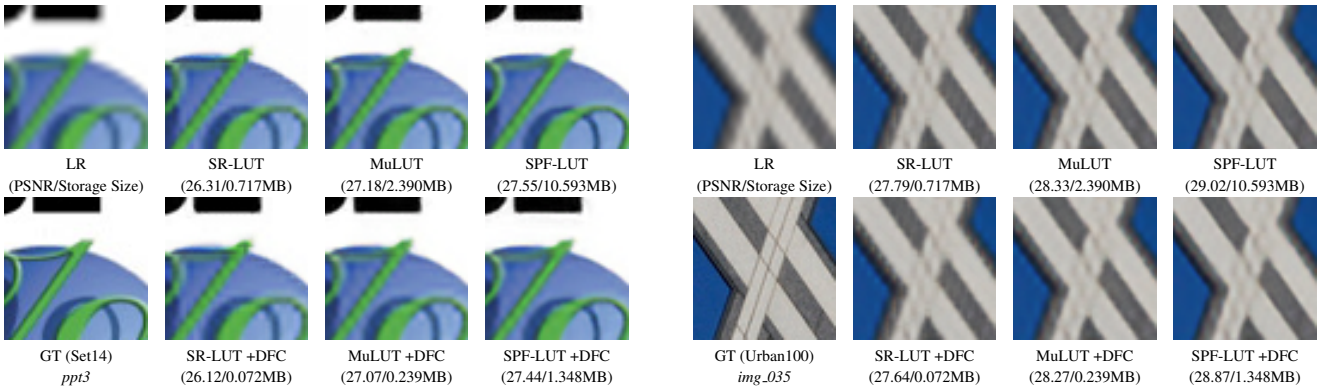


Figure 7. Qualitative comparison for $\times 3$ super-resolution on standard benchmark datasets [9, 28].

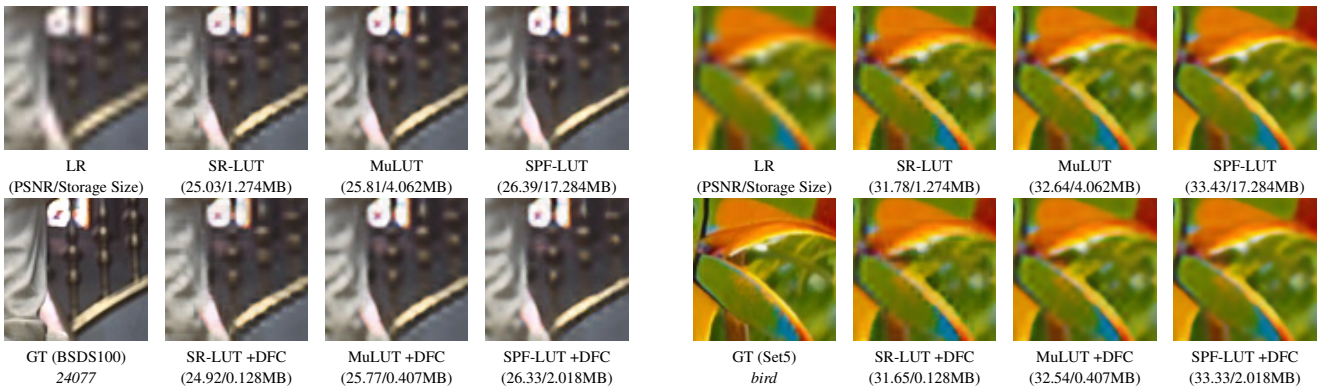


Figure 8. Qualitative comparison for $\times 4$ super-resolution on standard benchmark datasets [1, 16].

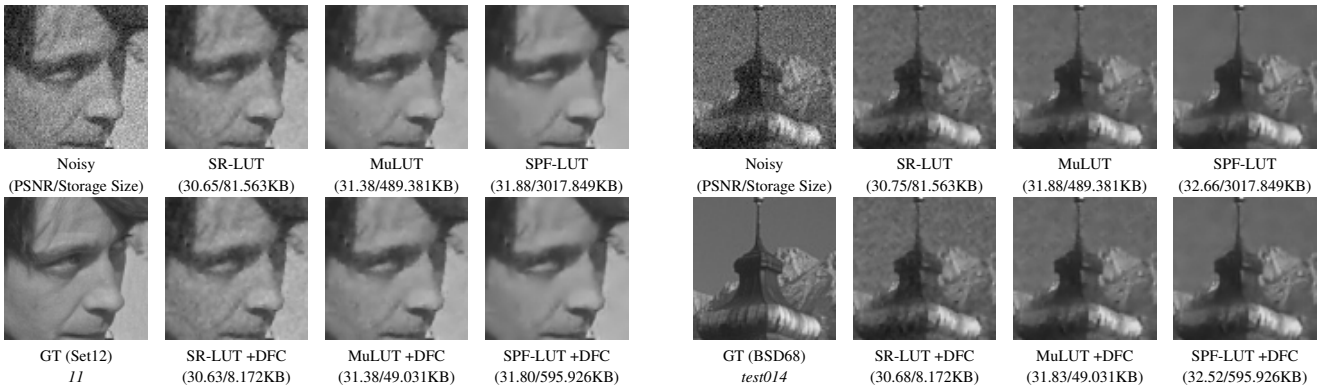


Figure 9. Qualitative comparison for grayscale image denoising at a noise level of 15 on standard benchmark datasets [16, 29].

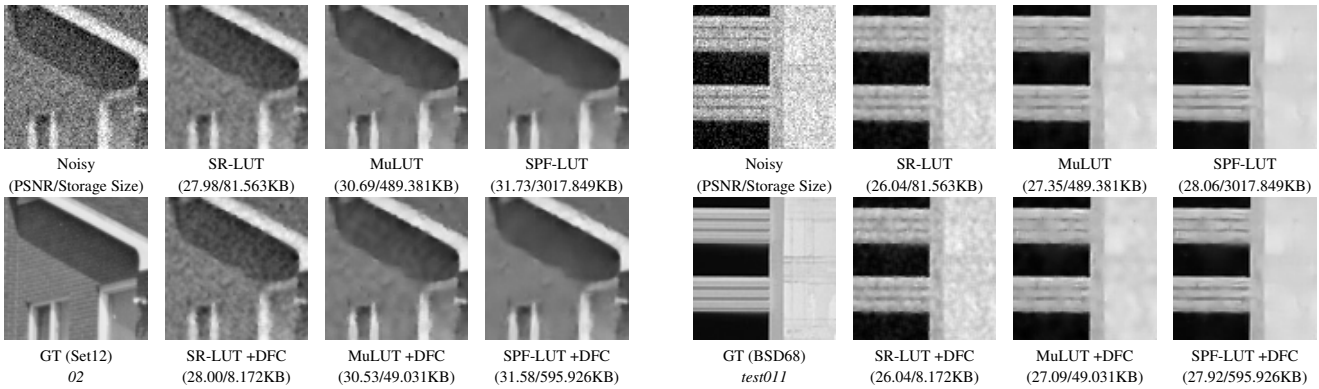


Figure 10. Qualitative comparison for grayscale image denoising at a noise level of 25 on standard benchmark datasets [16, 29].

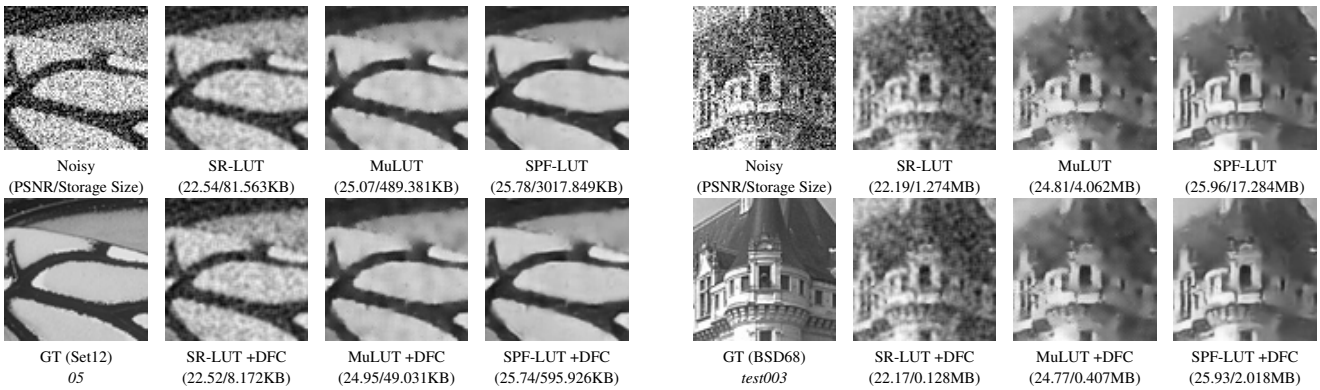


Figure 11. Qualitative comparison for grayscale image denoising at a noise level of 50 on standard benchmark datasets [16, 29].

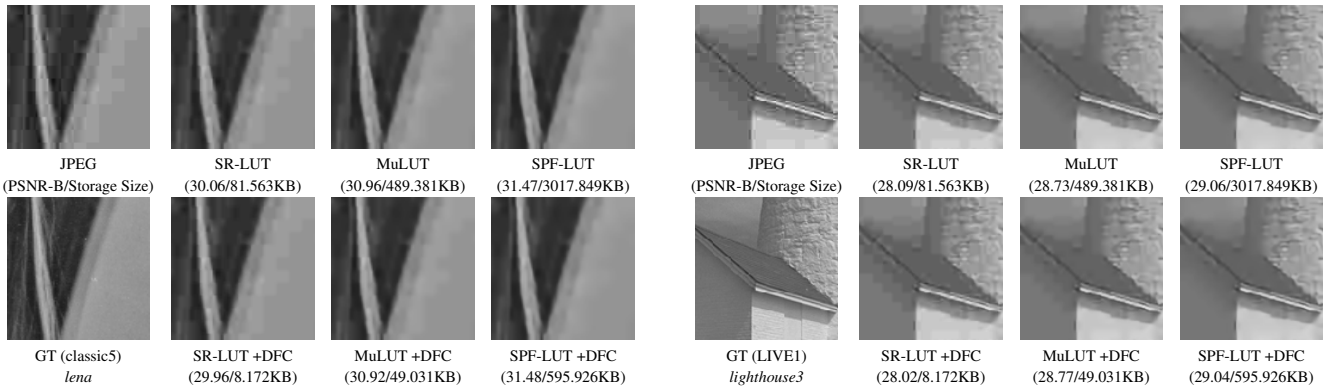


Figure 12. Qualitative comparison for image deblocking under the quality factor of 10 on standard benchmark datasets [6, 19].

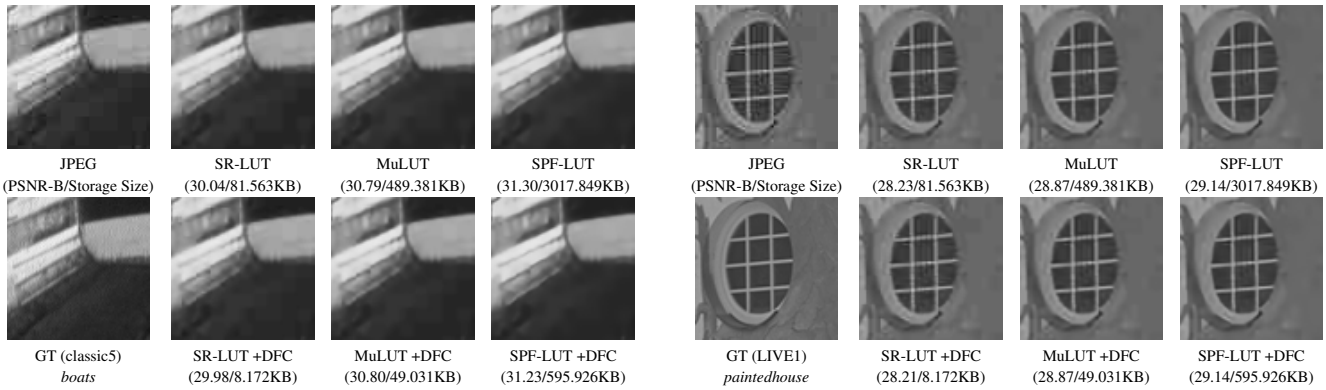


Figure 13. Qualitative comparison for image deblocking under the quality factor of 20 on standard benchmark datasets [6, 19].

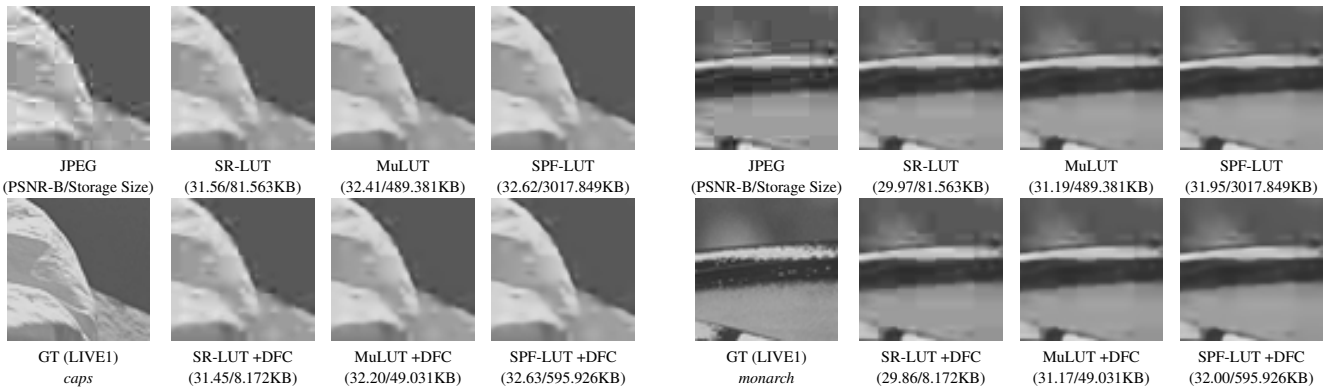


Figure 14. Qualitative comparison for image deblocking under the quality factor of 30 on standard benchmark datasets [6, 19].

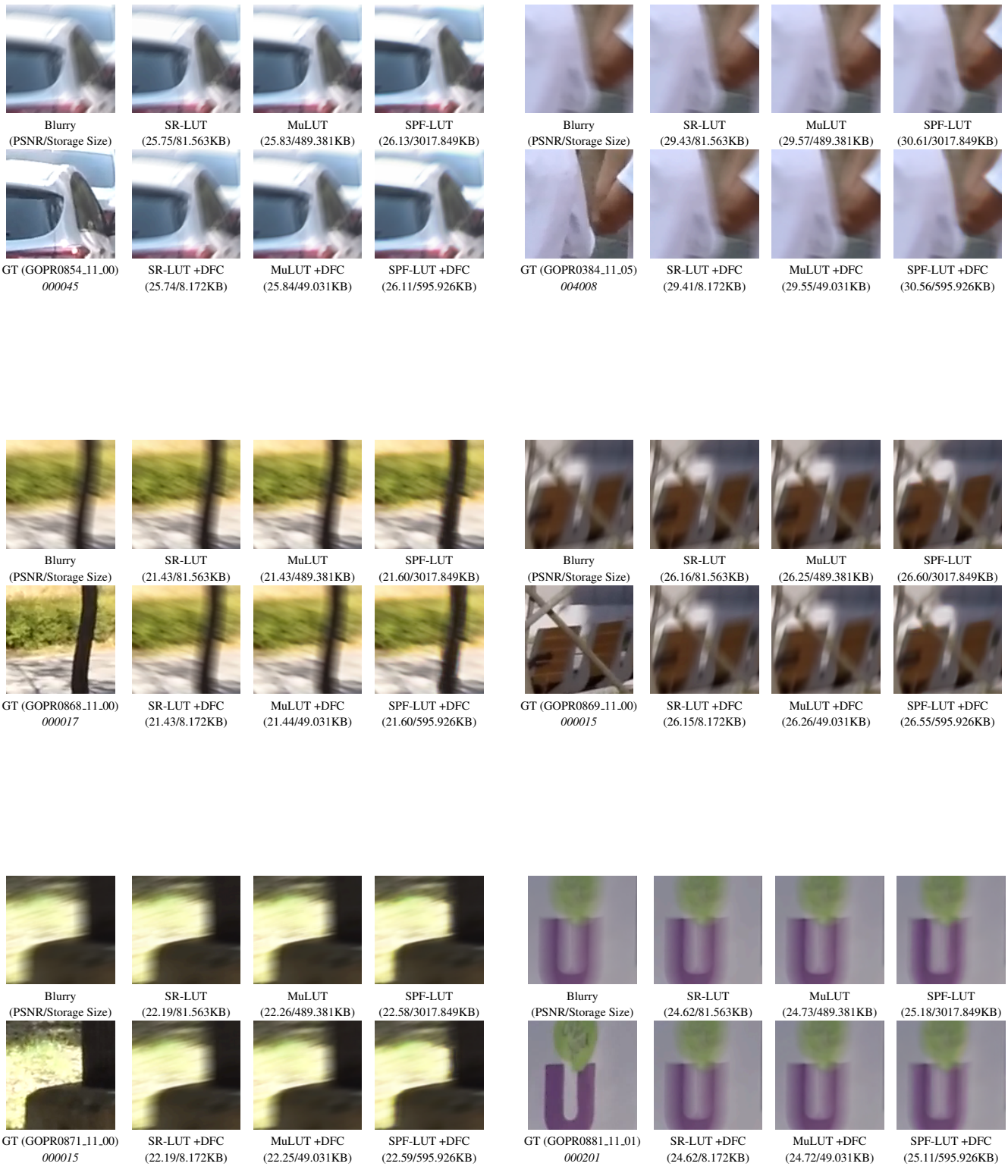


Figure 15. Qualitative comparison for image deblurring on standard benchmark datasets [18].

References

[1] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012. 5, 7

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

[3] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6): 1256–1272, 2016. 6

[4] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Trans. Image Process.*, 16(8):2080–2095, 2007. 6

[5] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In *ICCV*, 2015. 4, 5, 6

[6] Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images. *IEEE Trans. Image Process.*, 16(5):1395–1411, 2007. 6, 9

[7] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *CVPR*, 2014. 6

[8] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *ISSCC*, 2014. 4

[9] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015. 5, 7

[10] Younghyun Jo and Seon Joo Kim. Practical single-image super-resolution using look-up table. In *CVPR*, 2021. 3, 4, 5, 6

[11] Jiacheng Li, Chang Chen, Zhen Cheng, and Zhiwei Xiong. Mulut: Cooperating multiple look-up tables for efficient image super-resolution. In *ECCV*, 2022. 3, 4, 5, 6

[12] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *ICCV*, 2021. 6

[13] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, 2017. 4, 5

[14] Guandu Liu, Yukang Ding, Mading Li, Ming Sun, Xing Wen, and Bin Wang. Reconstructed convolution module based look-up tables for efficient image super-resolution. In *ICCV*, 2023. 3

[15] Cheng Ma, Jingyi Zhang, Jie Zhou, and Jiwen Lu. Learning series-parallel lookup tables for efficient image super-resolution. In *ECCV*, 2022. 3, 4

[16] 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*, 2001. 1, 2, 5, 6, 7, 8

[17] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multim. Tools Appl.*, 76:21811–21838, 2017. 5, 7

[18] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, 2017. 6, 10

[19] H Sheikh. Live image quality assessment database release 2. *http://live. ece. utexas. edu/research/quality*, 2005. 6, 9

[20] 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*, 2016. 2

[21] Dehua Song, Yunhe Wang, Hanting Chen, Chang Xu, Chunjing Xu, and DaCheng Tao. Addersr: Towards energy efficient image super-resolution. In *CVPR*, 2021. 4

[22] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017. 4

[23] Radu Timofte, Vincent De Smet, and Luc Van Gool. Anchored neighborhood regression for fast example-based super-resolution. In *ICCV*, 2013. 5

[24] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, 2015. 5

[25] 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 Workshops*, 2018. 4, 5

[26] Ke Yu, Chao Dong, Chen Change Loy, and Xiaoou Tang. Deep convolution networks for compression artifacts reduction. *arXiv preprint arXiv:1608.02778*, 2016. 4, 5

[27] Hui Zeng, Jianrui Cai, Lida Li, Zisheng Cao, and Lei Zhang. Learning image-adaptive 3d lookup tables for high performance photo enhancement in real-time. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(4):2058–2073, 2020. 2

[28] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International conference on curves and surfaces*, 2012. 5, 7

[29] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Trans. Image Process.*, 26(7):3142–3155, 2017. 6, 8

[30] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Trans. Image Process.*, 27(9):4608–4622, 2018. 6

[31] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, 2018. 5