ShiftLUT: Spatial Shift Enhanced Look-Up Tables for Efficient Image Restoration

Supplementary Material

In this supplementary file, we provide:

- Preliminary knowledge for the training, transferring, and testing process of LUT-based methods, as well as the SMS strategy in Sec. S1.
- An additional ablation study on comparing alternative shifting operations with LSS in Sec. S2.
- An additional ablation study on integrating LSS into other LUT-based methods in Sec. S3.
- More quantitative and qualitative comparisons among LUT methods on image restoration tasks in Sec. S4.
- Pseudo code of EAS in Sec. S5.

S1. Preliminary

Lookup Table (LUT) based super-resolution aims to replace expensive pixel-wise computations with efficient table lookup and interpolation. The core idea is to train a lightweight SR model with a constrained receptive field (RF). Once the model is fully trained, enumerates all possible input configurations within this RF and stores the corresponding high-resolution (HR) outputs in a precomputed LUT. During inference, the HR value is obtained exclusively by querying and interpolating LUT entries, yielding extremely fast runtime. As illstrated in Fig. 1, the overall process of LUT-based methods involves three primary stages: (1) training a neural network, (2) transferring its learned mapping into LUTs, and (3) performing inference entirely through LUT queries.

Training. Let $x_i \in \mathbb{R}^{k\times k}$ denote the low-resolution (LR) input patch corresponding to a receptive field of size $k \times k$ pixels. A compact CNN $f(\cdot)$ constrained to this RF is trained such that it predicts r^2 HR sub-pixels following the pixel-shuffle paradigm:

$$\hat{y}_i = f(x_i), \qquad \hat{y}_i \in \mathbb{R}^{r^2}, \tag{1}$$

where r is the upsampling factor. To enlarge the effective context while keeping RF small, the model is trained using 4 rotational augmentations:

$$\hat{y}_i = \frac{1}{4}\sum_{j=0}^{3} R_j^{-1}(f(R_j(x_i)))\,, \tag{2}$$

where R_j rotates the patch by $j \times 90°$. The trained network is only used once to build the LUT, and it is not required during inference.

Transferring. After training, enumerates every possible input configuration within the RF space and stores the network output into an n-dimensional LUT, where $n = k^2$ is the number of pixels in the receptive field.

Because a full LUT with 2^8 bins per dimension is prohibitively large for $n \geq 3$, previous studies adopt uniform sampling with interval s (e.g., $s = 16$ or 24) to address this issue. Each input pixel value $v \in [0, 255]$ is quantized by:

$$\tilde{v} = \mathrm{clip}\Big(\Big\lfloor \frac{v}{s} \Big\rceil, 0, N-1\Big)\,, \tag{3}$$

where $N = \frac{256}{s} + 1$ is the number of sampling bins. For each sampled tuple $(\tilde{v}_1, \ldots, \tilde{v}_n)$ compute:

$$\mathrm{LUT}[\tilde{v}_1, \ldots, \tilde{v}_n] = f(v_1, \ldots, v_n), \tag{4}$$

where $(v_1, \ldots, v_n)$ is the corresponding representative input. This produces an n-dimensional array storing r^2 HR sub-pixels per entry.

Testing. At test time, the HR reconstruction is performed solely via memory lookup and multidimensional interpolation. Given an LR patch $(v_1, \ldots, v_n)$, it identifies the nearest sampled bins by extracting the most significant bits:

$$\tilde{v}_i = \Big\lfloor \frac{v_i}{s} \Big\rfloor, \quad L_i = v_i \bmod s, \tag{5}$$

where L_i is the residual offset used for interpolation.

For n-dimensional LUTs, simplex-based interpolation (triangle in 2D, tetrahedron in 3D, 4-simplex in 4D) is adopted. Let $\{O_0, \ldots, O_n\}$ be the LUT vertices of the selected simplex and $\{w_0, \ldots, w_n\}$ the corresponding barycentric weights derived from $(L_1, \ldots, L_n)$. The final HR output is:

$$\hat{y} = \frac{1}{s}\sum_{i=0}^{n} w_i O_i. \tag{6}$$

This inference pipeline contains no convolution and only involves table lookup and a few integer operations, enabling extremely fast execution on resource-limited devices.

Separable Mapping Strategy (SMS). A major challenge in converting convolutions to LUTs is the exponential growth of storage with kernel size. For example, a 3×3 convolution requires a 9-dimensional LUT, which is impractical.

SMS [4] addresses this by decomposing an $k \times k$ convolution into k^2 independent 1×1 mappings, each represented

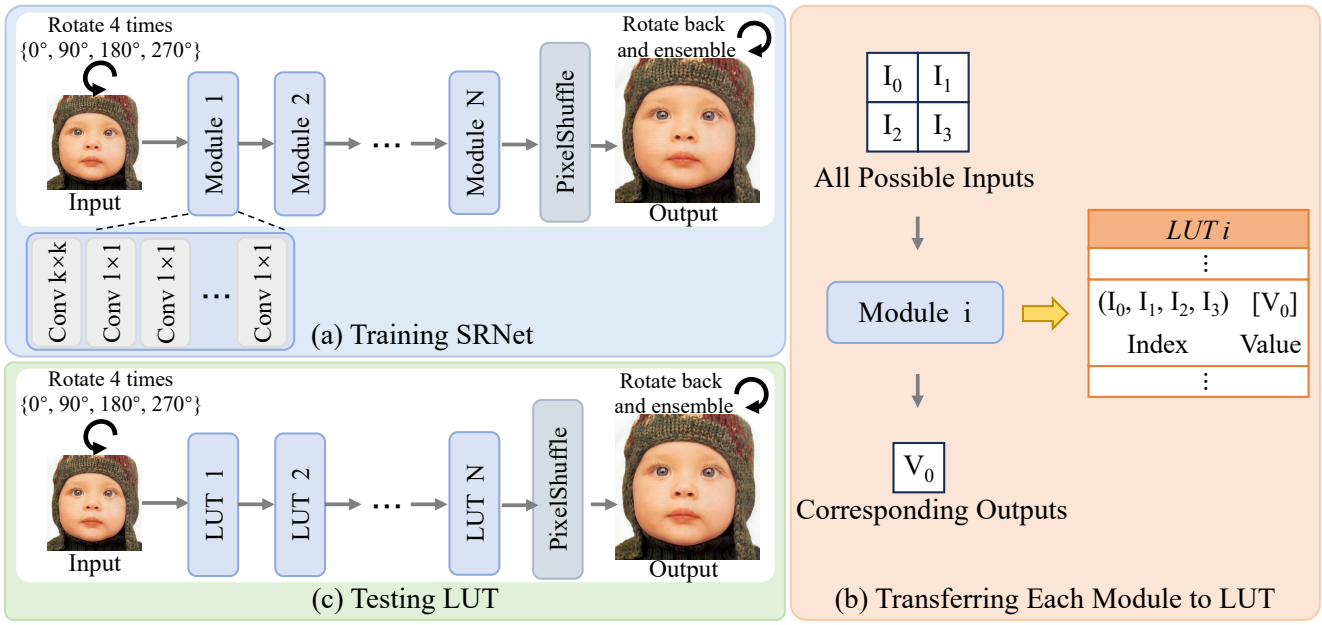

Figure 1. Illustration of training, transferring and testing process for LUT-based methods.

by a simple 1D LUT. This reduces the storage dependency from exponential (v^{k^2}) to linear ($s \times k^2$), where v denotes the number of sampled values. The final output is obtained by averaging the results of all sub-LUTs:

$$\hat{F}out = \frac{1}{k^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} LUT_{(i,j)}[x_{(i,j)}], \tag{7}$$

where $x_{(i,j)}$ is the input feature and $LUT_{(i,j)}$ the corresponding 1D LUT.

S2. Comparison of other shifting operations

In Section 3.2, we introduce LSS with a two-stage training strategy: in the first stage, the offset network predicts floating-point offsets, while in the second stage, the fixed integer-valued offsets are adopted. This quantization process inevitably degrades performance. As shown in Tab. 1, there is a 0.017dB performance drop between the two stages.

A natural question arises: *can LSS directly predict integer-valued offsets to avoid such degradation*? To investigate this, we explore two variants of the LSS module with discrete learnable offsets: the first approach employs Straight-Through Estimation (STE) to round the output offsets from the offset network while maintaining differentiability. The second approach adopts the Gumbel-Softmax method [2], which is commonly used in reinforcement learning to model discrete action states. Experimental results in Tab. 1 show that our two-stage training consistently attains the highest reconstruction accuracy across all test sets by a large margin, which also validates the effectiveness of the two-stage training strategy proposed in the main paper.

Table 1. Comparison of different shifting operations in ShiftLUT on standard SISR test sets for an upscaling factor of 4.

Method	Set5	Set14	BSDS100	Urban100
ShiftLUT (First Stage)	31.35	28.12	27.23	25.14
ShiftLUT (Second Stage)	31.33	28.11	27.21	25.12
ShiftLUT (STE)	30.91	27.83	27.04	24.77
ShiftLUT (Gumbel Softmax)	31.11	27.97	27.12	24.95

S3. Generalization of the LSS Module

To verify the generalization ability of the proposed LSS module, we integrate LSS into two representative LUT-based networks and report the resulting restoration accuracy for $\times 4$ SISR in Tab. 2. We choose SPLUT and TinyLUT as the target backbones because both architectures expose multi-channel intermediate features ($c > 1$), which is a necessary condition for LSS.

As shown in Tab. 2, when LSS is appended to SPLUT (feature channels $c = 4$), the effect is marginal: Set5 increases by $+0.02$ dB (30.01 $\rightarrow$ 30.03), Set14 by $+0.03$ dB, Manga109 by $+0.05$ dB. By contrast, inserting LSS into TinyLUT (feature channels $c = 16$) yields consistently larger gains across all datasets: Set5 $+0.06$ dB (31.18 $\rightarrow$ 31.24), Set14 $+0.03$ dB, BSDS100 $+0.04$ dB, Urban100 $+0.13$ dB and Manga109 $+0.15$ dB.

Table 2. Quantitative comparison of PSNR on standard benchmark datasets for x4 super-resolution tasks between the original version of SPLUT and TinyLUT, and the modified version integrating LSS.

	Storage Size	*Set5*	*Set14*	*BSDS100*	*Urban100*	*Manga109*
SPLUT	5632KB	30.01	27.20	26.68	24.13	27.00
SPLUT + LSS	5632KB	30.03	27.23	26.65	24.13	27.05
TinyLUT	171KB	31.18	28.01	27.13	24.92	28.83
TinyLUT + LSS	171KB	31.24	28.04	27.17	25.05	28.98

Table 3. PSNR-B results of different methods on Classic5 and LIVE1 (quality factor 20, 30, 40).

PSNR-B	**Method**	**Classic5**			**LIVE1**		
		20	30	40	20	30	40
LUT	SR-LUT	29.54	30.80	31.71	29.74	30.99	32.00
	MuLUT	30.30	31.57	32.32	30.52	31.79	32.66
	SPF-LUT+DFC	30.65	31.95	32.75	30.80	32.10	33.02
	ShiftLUT-F(ours)	31.15	32.33	33.15	31.07	32.31	33.29
DNN	ARCNN	30.59	31.98	32.79	30.79	32.38	33.14
	SwinIR	31.99	33.03	33.66	31.70	33.01	33.88

From these observations we draw two conclusions. First, LSS generalizes across different LUT-based architectures: integrating LSS into both SPLUT and TinyLUT produces non-negative or improved reconstruction performance, demonstrating applicability beyond the original backbone. Second, the magnitude of the benefit correlates with the intermediate feature channel dimensionality: networks with more channels (TinyLUT, $c = 16$) obtain larger and more consistent PSNR improvements than networks with fewer channels (SPLUT, $c = 4$). This behaviour is consistent with the design rationale of LSS, which leverages channel-wise spatial diversity into LUTs; hence, LSS is expected to be more effective when richer channel-wise representations are available.

S4. Additional Results of Image Restoration

Image Deblocking. We report the additional quantitative results for image deblocking under different quality factors (20, 30, and 40) on standard benchmarks in Tab. 3, which compares our method against several state-of-the-art LUT-based and DNN-based methods, including SR-LUT [3], MuLUT [5], SPF-LUT [6], as well as DNN-based approaches such as ARCNN [1], and SwinIR [7].

Image Denoising. We report the additional quantitative results for grayscale image denoising at a noise level of 25 and 50 on standard benchmarks in Tab. 4, which compares our method against several state-of-the-art LUT-based and DNN-based methods, including SR-LUT [3], MuLUT [5], SPF-LUT [6], as well as DNN-based approaches such as DnCNN [8], FFDNet [9] and SwinIR [7].

Table 4. PSNR results of different methods on Set12 and BSD68 (noise levels 25, 50).

PSNR	**Method**	**Set12**		**BSD68**	
		25	50	25	50
LUT	SR-LUT	27.19	22.62	26.85	22.39
	MuLUT	28.94	25.46	28.18	24.97
	SPF-LUT+DFC	29.43	26.07	28.50	25.50
	ShiftLUT-F(ours)	29.98	26.76	28.90	25.98
DNN	DnCNN	30.44	27.18	29.23	26.23
	FFDNet	30.54	27.32	29.19	26.29
	SwinIR	31.01	27.97	29.50	26.58

S5. Pseudo Code of EAS

Algorithm 1 provides the detailed pseudo-code for the proposed Error-bounded Adaptive Sampling (EAS) pipeline, which encompasses both the offline optimization and online inference stages.

Algorithm 1 EAS Pipeline: Optimization & Inference

Stage 1: Offline Optimization
Input: Pre-trained LUTs $\{\mathcal{L}_1, \ldots, \mathcal{L}_L\}$, tolerance ε
for each layer $l = 1 \ldots L$ **do**
    Search for max stride $s_l^* \in \mathcal{S}$ s.t. $\text{Error}(s_l^*) \leq \varepsilon$
    Store compressed $\mathcal{L}_l^{sparse}$
end for

Stage 2: Online Inference (Serial Cascade)
Input: Image x, Compressed $\{\mathcal{L}_1^{sparse}, \ldots\}$, Shared Buffer $\mathcal{B}$
for each layer $l = 1 \ldots L$ **do** ▷ Process layers sequentially
    // Step A: Refresh Buffer with current layer's LUT
    for $i \in 0 \ldots 2^{\text{MSB}} - 1$ **do**
        $\mathcal{B}[i] \leftarrow \text{Query}(\mathcal{L}_l^{sparse}, i, s_l^*)$ ▷ Pre-compute
    end for
    // Step B: Pixel-wise Inference (Zero-overhead lookup)
    for each pixel p in x **do**
        $x_p \leftarrow x_p + \mathcal{B}[x_p]$ ▷ Residual lookup using buffer
    end for
end for
return Processed feature map x

References

[1] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE international conference on computer vision*, pages 576–584, 2015. 3

[2] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 2

[3] Younghyun Jo and Seon Joo Kim. Practical single-image super-resolution using look-up table. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 691–700, 2021. 3

[4] Huanan Li, Juntao Guan, Lai Rui, Sijun Ma, and Lin Gu. Tinylut: Tiny look-up table for efficient image restoration at the edge. *Advances in Neural Information Processing Systems*, 37:85340–85359, 2024. 1

[5] Jiacheng Li, Chang Chen, Zhen Cheng, and Zhiwei Xiong. Mulut: Cooperating multiple look-up tables for efficient image super-resolution. In *European conference on computer vision*, pages 238–256. Springer, 2022. 3

[6] Yinglong Li, Jiacheng Li, and Zhiwei Xiong. Look-up table compression for efficient image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26016–26025, 2024. 3

[7] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1833–1844, 2021. 3

[8] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017. 3

[9] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018. 3