SepLUT: Separable Image-adaptive Lookup Tables for Real-time Image Enhancement

Canqian Yang[1*] Meiguang Jin[2*] Yi Xu[1†] Rui Zhang[1]
Ying Chen[2] Huaida Liu[2]

[1] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
{charles.young, xuyi, zhang_rui}@sjtu.edu.cn
[2] Alibaba Group
{meiguang.jmg, yingchen, liuhuaida.lhd}@alibaba-inc.com

Abstract. Image-adaptive lookup tables (LUTs) have achieved great success in real-time image enhancement tasks due to their high efficiency for modeling color transforms. However, they embed the complete transform, including the color component-independent and the component-correlated parts, into only a single type of LUTs, either 1D or 3D, in a coupled manner. This scheme raises a dilemma of improving model expressiveness or efficiency due to two factors. On the one hand, the 1D LUTs provide high computational efficiency but lack the critical capability of color components interaction. On the other, the 3D LUTs present enhanced component-correlated transform capability but suffer from heavy memory footprint, high training difficulty, and limited cell utilization. Inspired by the conventional divide-and-conquer practice in the image signal processor, we present SepLUT (separable image-adaptive lookup table) to tackle the above limitations. Specifically, we separate a single color transform into a cascade of component-independent and component-correlated sub-transforms instantiated as 1D and 3D LUTs, respectively. In this way, the capabilities of two sub-transforms can facilitate each other, where the 3D LUT complements the ability to mix up color components, and the 1D LUT redistributes the input colors to increase the cell utilization of the 3D LUT and thus enable the use of a more lightweight 3D LUT. Experiments demonstrate that the proposed method presents enhanced performance on photo retouching benchmark datasets than the current state-of-the-art and achieves real-time processing on both GPUs and CPUs.

1 Introduction

The lookup table (LUT) is a promising data structure to efficiently conduct specific transforms by replacing expensive runtime computation with cheap array caching and indexing. It precomputes the outputs of a function over a sampled domain of inputs and evaluates the same function via efficient lookup and interpolation operations. LUTs are widely used to optimize color transforms in the

* Equal Contribution † Corresponding Author
Work partially done during an internship of C. Yang at Alibaba Group.

image signal processor (ISP), a crucial component in the camera imaging pipeline and many display devices. To transform sensor signals into human-perceptible digital images, typical ISP follows the divide-and-conquer principle to utilize two types of LUTs, of 1D and 3D, for handling different transforms [15]. 1D LUTs are suitable for *component-independent transforms* that require no interaction between three color components, such as white-balancing, gamma correction, brightness adjustment, and contrast stretching. 3D LUTs further enable the mixture of different color components, thus supporting more sophisticated *component-correlated transforms*, like adjustments in hue and saturation.

The high effectiveness and efficiency of LUTs motivate recent advances in deep learning to propose learnable, image-adaptive LUTs for enhanced real-time image enhancement [25,30,18,17,21,11,14,26,1,35,32]. However, these methods encode a complete color transform to only a single type of LUTs, either 1D or 3D, but neglecting the limited capability of a single module to model component-independent and component-correlated transformations simultaneously. Such a paradigm limits the expressiveness of these methods. Specifically, methods based on 1D LUTs lack the critical model capability of interacting component information as they work on each color component independently. Though the methods based on 3D LUTs are able to handle both component-independent and component-correlated transforms, they model these two transforms in a coupled manner, which increases the capability requirement of the model. The reason owns to the lack of a prior component-independent transform that can rescale the input image range into a normalized and perceptually uniform color space for the 3D LUTs. Therefore, the 3D LUTs rely on increasing their sizes for adaption to the diversity of input color ranges. For example, [35,32] adopt 33-point 3D LUTs, while the ISP typically employs 17-point or even 9-point 3D LUTs [15]. The large LUT size introduces massive parameters, resulting in heavy memory burden and high training difficulty. Besides, adopting a relatively large LUT size will lead to insufficient cell utilization of the 3D LUTs since the colors appearing in a single input image usually occupy only a tiny sub-space of the entire color space, causing redundancy of the model capacity.

To simultaneously improve the model's expressiveness and efficiency, we propose a novel framework called separable image-adaptive lookup tables (SepLUT). It decouples a single color transform into component-independent and component-correlated sub-transforms instantiated as 1D and 3D LUTs, respectively. The idea is directly motivated by the common practice in ISP, where 1D and 3D LUTs play their roles in conjunction. As illustrated in Figure 1, we follow the paradigm of dynamic neural functions [6] to employ a CNN backbone network on a downsampled, fixed-resolution version of the input image for predicting the parameters of a 3× 1D LUT and a 3D LUT. The two generated LUTs are then applied to the original input image sequentially – the 3× 1D LUT rescales each color channel to adaptively adjust the brightness/contrast, followed by the 3D LUT to mix up three color components for manipulation on hue and saturation. The advantages are two-fold. On the one hand, the 3D LUTs can complement the 1D LUTs with color components interaction. On the other, the 1D LUTs

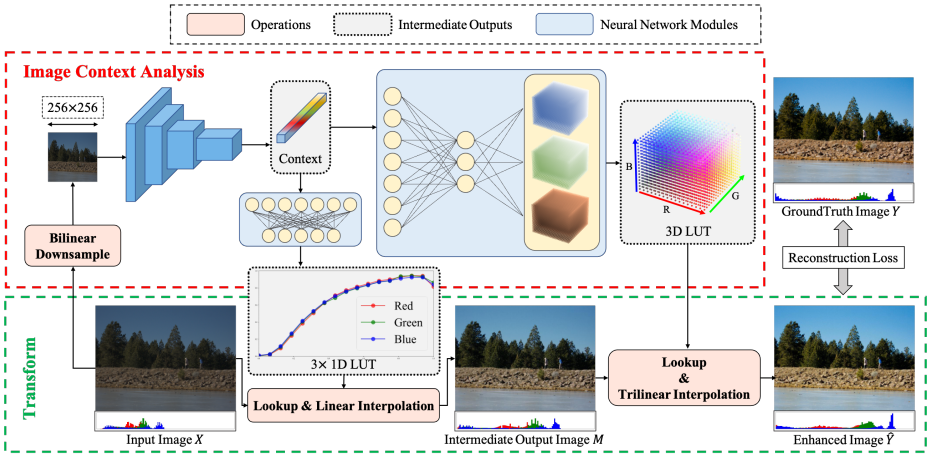


Fig. 1. Framework of the proposed method. Our method employs a lightweight CNN network to analyze the image context from a down-sampled, fixed-resolution version of the input image. The image context is used to guide the generation of a 3× 1D LUT and a 3D LUT in an image-adaptive fashion. The original input image is afterward enhanced by the cascade of the predicted 1D and 3D LUTs. Best viewed in color.

can redistribute the input colors into specific ranges of the following 3D LUTs, which increase the cell utilization of the 3D LUTs, thus reducing the redundant capacity and enabling the usage of smaller sizes. Furthermore, the consistency between the input and output spaces of a LUT allows trivial LUT quantization that provides significant lightweight property to the proposed method.

The contributions of this paper are three-fold: (1) We present a novel viewpoint of separating a single color transform into two sub-transforms, *i.e.*, the component-independent transform and the component-correlated transform. (2) We propose a general framework that adopts a cascade of 1D and 3D LUTs to instantiate the above two sub-transforms, making them facilitate each other and present overall lightweight property, high efficiency, and enhanced expressiveness. (3) We demonstrate the efficiency and effectiveness of the proposed method via extensive experiments on publicly available benchmark datasets.

2 Related Works

2.1 Lookup Tables

A lookup table (LUT) defines a table of values addressed by a set of indices. It usually serves as an effective and efficient representation of a univariate or multivariate function $y = f(x_1, \ldots, x_n), n = 1, 2, \ldots$ by enumerating all possible input combinations $\{(x_1, \ldots, x_n)\}$ and storing the corresponding output values y. The function can afterward be evaluated using only the memory access and interpolation, without performing the computation again. Therefore, LUTs are

commonly used in computer systems [20,29], especially some embedded devices [15], to accelerate computation. The simplest LUT is the 1D LUT indexed by a single variable ($n = 1$), which utilizes linear interpolation to generate values of specific indices that are not an element of the table. Another frequently-used LUTs are the 3D LUTs indexed by a triplet of three independent variables ($n = 3$), which require more complicated interpolation techniques such as the trilinear [28] and the tetrahedral [16] interpolations. According to [15], most of the modules in the practical ISP systems are implemented using either 1D or 3D LUTs due to their high efficacy and suitability for modeling color transforms.

Learnable LUTs The high efficiency and wide usage of the LUTs in ISP also attract efforts in deep learning-based image enhancement to learning more powerful LUTs via data-driven approaches. Previous works [25,30,18,17,21,11,14,26,1] mainly focus on learning 1D LUTs to mimic the color adjustment curves in popular image enhancement software such as Photoshop and Lightroom. They regress either a set of the control points of the curve, or the coefficients of some hand-crafted functions (*e.g.*, polynomial function). However, these methods usually suffer from the lack of correlation between color channels. Recently, [35,32] extended those 1D LUT-based methods into using 3D LUTs. They predict 3D LUTs with adaption to different image contents by learning several image-independent basis 3D LUTs and combining them using image-dependent weights. The 3D LUTs consider the relationship between color channels and thus provide higher expressiveness to model more complicated color transforms. However, the large model capacity of the 3D LUTs requires massive learnable parameters, making these methods suffer from heavy memory/storage footprints. In contrast to previous works based on a single kind of LUTs, we present a more general framework that considers the conjunction of both types of LUTs, thus providing an advanced solution to the above limitations.

2.2 Image Enhancement

Existing learning-based image enhancement methods can be roughly categorized into two paradigms, *i.e.*, the fully convolutional network-based methods, and the color transform-based methods. The first paradigm [5,24,8,36,34,4] is to train a fully convolutional network (FCN) that directly regresses the enhanced image from the input in a dense prediction fashion. However, these methods are still far from practical applications due to their heavy computational burdens and limited feasible input resolutions. In contrast, the second paradigm decouples the color transformations from the heavy CNN model for real-time and high-resolution processing. Specifically, these methods employ CNNs on a low-resolution, fixed-size version of the input image to predict image-adaptive parameters of some specific color transform functions. Typical color transform functions include affine transformation matrices [9,31,3,23], curve-based functions [25,30,18,17,21,11,14,26,1], multi-layer perceptrons (MLPs) [12] and 3D LUTs [35,32]. These learned transform functions can adapt to different input

Table 1. Architecture of the backbone network, where m is a hyper-parameter that serves as a channel multiplier controlling the width of each convolutional layer.

Id	Layer	Output Shape
0	Bilinear Resize	$3 \times 256 \times 256$
1	Conv3x3, LeakyReLU	$m \times 128 \times 128$
2	InstanceNorm	$m \times 128 \times 128$
3	Conv3x3, LeakyReLU	$2m \times 64 \times 64$
4	InstanceNorm	$2m \times 64 \times 64$
5	Conv3x3, LeakyReLU	$4m \times 32 \times 32$
6	InstanceNorm	$4m \times 32 \times 32$
7	Conv3x3, LeakyReLU	$8m \times 16 \times 16$
8	InstanceNorm	$8m \times 16 \times 16$
9	Conv3x3, LeakyReLU	$8m \times 8 \times 8$
10	Dropout (0.5)	$8m \times 8 \times 8$
11	AveragePooling	$8m \times 2 \times 2$
12	Reshape	$32m$

Table 2. Architecture of the 1D LUT generator, where S_o is the size (number of elements) of the 1D LUTs. "FC" denotes the fully-connected layer.

Id	Layer	Output Shape
0	FC	$3S_o$
1	Reshape	$3 \times S_o$
2	Sigmoid	$3 \times S_o$

Table 3. Architecture of the 3D LUT generator, where S_t is the size (number of elements along each dimension) of the 3D LUT.

Id	Layer	Output Shape
0	FC	K
1	FC	$3S_t^3$
2	Reshape	$3 \times S_t \times S_t \times S_t$

image contents and present high runtime efficiency. Some of them are also flexible and scalable on arbitrary input resolutions. Therefore, our work also follows this line of the color transform-based scheme and builds a novel framework with a cascade of 1D and 3D LUTs for real-time image enhancement.

3 Methods

3.1 Overall Framework

Figure 1 shows an overview of the proposed image enhancement framework, which follows the popular paradigm of dynamic neural functions. Specifically, a lightweight CNN network is employed as the backbone network on the input image to extract global context that will serve as the guide to generate image content-dependent color transform functions. The form of the color transform functions is designed as a cascade of a 3× 1D LUT and a 3D LUT, aiming to handle the component-independent and component-correlated transforms in a decoupled and sequential manner. Afterward, the generated functions enhance the quality of the input through efficient lookup and interpolation operations.

3.2 Global Image Context Analysis: Backbone Network

The backbone CNN network is central to achieving image-adaptiveness in the proposed framework. It is responsible for predicting the parameters of the subsequent 1D and 3D LUTs according to the image content through analyzing the input image $X \in [0,1]^{3\times H\times W}$. Since both the 1D and 3D LUTs in our method

are designed for global color transform, the backbone network only needs to capture a coarse understanding of the input image. Therefore, a low-resolution version (*e.g.*, 256×256) of the input image is sufficient and can substantially reduce the computational cost to a fixed level. The detailed architecture of the CNN backbone network is listed in Table 1, where m is a hyper-parameter controlling the channel width of each convolutional layer. The backbone network adopts 5 strided convolutional layers to downsample the input image into 1/32 resolution. At the end of the network are an average pooling layer and a reshape operation that further convert the feature maps into a compact vector representation $E \in \mathbb{R}^{32m}$. The vector representation captures some global attributes of the input image and will be fed into the subsequent modules as the guide to generate image content-dependent LUT parameters.

3.3 Component-independent Transform: 1D Lookup Tables

The component-independent transform aims to redistribute the input colors into a more perceptually uniform space that can increase the cell utilization of the following 3D LUT. In this paper, we propose to adopt the 3× 1D LUT for the above purpose, where three individual 1D LUTs are predicted for each color channel, respectively. The elements $\{T^c_{1D}\}_{c\in\{r,g,b\}}$ of the 1D LUTs are conditioned on the image context E to achieve image-adaptiveness, formulated as:

$$\{T^r_{1D}, T^g_{1D}, T^b_{1D}\} = g_{1D}(E), \tag{1}$$

where $T^c_{1D} \in [0,1]^{S_o}$ denotes a 1D LUT of S_o values for the channel $c \in \{r,g,b\}$. g_{1D} is the 1D LUT generator module that takes E as input and predicts all output values, whose architecture is detailed in Table 2. Note that the *sigmoid* layer is used to normalize the elements in the 1D LUTs into a valid range. Once the 1D LUTs are predicted, the component-independent transform can be performed on each pixel separably via simple linear interpolation:

$$M[c,h,w] = \text{linear\_interpolate}(T^c_{1D}, X[c,h,w]), \tag{2}$$

where $M \in [0,1]^{3\times H\times W}$ denotes the intermediate image transformed by the 1D LUTs. $h \in \mathbb{I}_0^{H-1}$ and $w \in \mathbb{I}_0^{W-1}$ are indices to traverse the image [3].

Intuitively, directly applying histogram equalization, a conventional technique in image processing to adjust image contrast, is another alternative as the component-independent transform to increase the distribution uniformity of each color component. However, histogram equalization maximizes the entropy of the image statically by mapping the input color ranges to an *exact uniform* distribution, which is not intelligent and not always necessary in the scenario of image enhancement. Instead, by adopting the data-driven approach to learn the 1D LUTs, our method is expected to be image-adaptive for the network and each input image. The quantitative comparisons can be found in Table 4.

[3] $\mathbb{I}_s^t$ denotes the integer set starting from s and ending at t, *i.e.*, $\mathbb{I}_s^t = \{s, \ldots, t\}$.

3.4 Component-correlated Transform: 3D Lookup Tables

After the previous component-independent transform, the component-correlated transform aims at mixing and interacting different color channels to achieve more sophisticated color transforms, such as alteration in hue and saturation. Considering the balance between expressiveness and efficiency, we choose the 3D LUTs to formulate such a transformation from a triplet to one another. A typical 3D LUT defines a 3D grid of S_t^3 elements, where S_t denotes the number of values along each color dimension. Similar to Section 3.3, all the S_t^3 elements in the 3D LUT should be automatically predicted by the neural network to consider the adaption to the diversity of various input images. Such an objective formulates a mapping from the image context E to a $3S_t^3$-dimension parameter space:

$$T_{3D} = g_{3D}(E), \tag{3}$$

where $T_{3D} \in [0,1]^{3\times S_t\times S_t\times S_t}$ denotes a 3D LUT of size S_t. g_{3D} is the 3D LUT generator module. To prevent the involvement of significant memory burden and training difficulty, we consider the *rank factorization* to decompose the complete mapping in Equation (3) into two sub-mappings h_1 and h_2:

$$g_{3D} = \ h_1 \circ h_2,\ h_1 : \mathbb{R}^{32m} \to \mathbb{R}^K,\ h_2 : \mathbb{R}^K \to \mathbb{R}^{3S_t^3}. \tag{4}$$

h_1 and h_2 can be instantiated with two respective fully-connected (FC) layers. Compared with a single FC layer, such a strategy reduces the number of parameters from $32m \times 3S_t^3$ to $K \times (32m + 3S_t^3)$, making the transform more feasible and easier to optimize. The detailed architecture can be found in Table 3.

Given the predicted 3D LUT T_{3D}, the final enhanced image $\hat{Y} \in [0,1]^{3\times H\times W}$ can be derived via a simple trilinear interpolation:

$$\hat{Y}[c,h,w] = \text{trilinear\_interpolate}(T_{3D}, M[c,h,w]). \tag{5}$$

3.5 Efficient Implementation via Quantization

Our method can also benefit from the model quantization technique to further reduce the memory and storage footprints. Specifically, the FC layers in the LUT generators (Tables 2 and 3) are equivalent to learning several image-independent basis LUTs encoded as the learnable parameters. The input to the FC layer serves as the image-dependent coefficients that linearly combine the basis LUTs into the final LUT. Note that the output of the LUT is simply a linear combination of the elements in the LUT and falls into the color space that can be naturally quantized. Therefore, thanks to the semantic consistency between the parameter space and the output space, the trained parameters of the LUT generators can be naturally quantized to lower-bit representation during the testing time without significant performance decline. It is worth noting that it is not trivial for other image enhancement methods to benefit from the model quantization techniques due to their inconsistency between parameter

and output spaces. More complicated model quantization approaches are required but would introduce cumbersome training protocols or substantially hurt the performance. Besides, since the LUTs and the input image can be quantized into fixed-point representation (*e.g.*, 8-bit integer), we can also replace the floating-point computation with the fixed-point counterpart in the lookup and interpolation procedure for further speedup, as shown in Table 5.

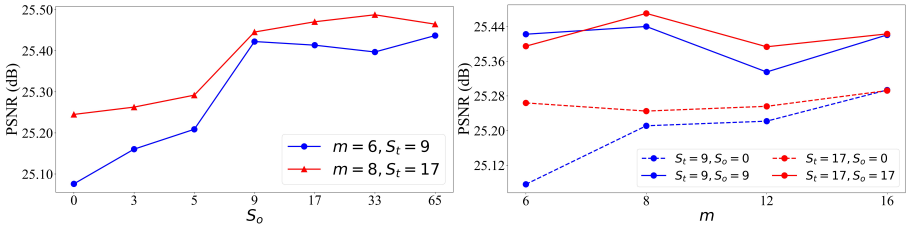
4 Experiments

4.1 Datasets

The publicly available MIT-Adobe FiveK [2] and PPR10K [22] datasets are adopted to evaluate the proposed method. The FiveK dataset contains 5,000 RAW images with five manually retouched ground truths (A/B/C/D/E). Version C is selected in our experiments. We use the commonly used settings [5,35] to split the dataset into 4,500 image pairs for training and the remaining 500 image pairs for testing. The 480P version of the dataset is used to speed up the training process, while the testing is conducted on both 480P and original 4K resolutions. The PPR10K dataset contains a larger scale of 11,161 RAW portrait photos with 3 versions of groundtruths (a/b/c). We follow the setting in [22] to utilize all three retouched versions as the groundtruth in different experiments and split the dataset into 8,875 pairs for training and 2,286 pairs for testing. Experiments are conducted on the 360P verison of the dataset. Following [35], experiments are organized on two typical application scenarios, *photo retouching*, and *tone mapping*. The former task retouches the input image in the same sRGB format, whereas the latter transforms the 16-bit CIE XYZ input images into 8-bit sRGB. We conduct both tasks on the FiveK dataset, but only the retouching task on PPR10K as done in [22]. As for the data augmentation strategies, we follow the settings in [35] and [22] to ensure a fair comparison.

4.2 Implementation Details

Tables 1 to 3 show the instantiations of modules in the proposed method. The parameters of the backbone network are randomly initialized as in [10]. To make a fair comparison, we also conduct experiments using the ResNet-18 [13] backbone network (initialized with ImageNet-pretrained [7] weights) on the PPR10K dataset, as done in [22]. As for the LUT generators, inspired by [35], we initialize the 3D LUT generator to predict an identity mapping at the early training stage to speed up the training convergence. The mean square error (MSE) loss is adopted to train the proposed method in an end-to-end manner. We do not introduce any other constraint or loss function to the predicted 1D and 3D LUTs, willing that they can be image-adaptive for the network and input image itself, not for any hand-crafted priors. We implement our method based on PyTorch 1.8.1 [27]. The standard Adam optimizer [19] is adopted to train the proposed method, with the mini-batch size set to 1 and 16 on FiveK and PPR10K, respectively. All models are trained for 400 epochs with a fixed learning rate of



(a) Different sizes (S_o) of the 1D LUT. (b) Different widths (m) of the backbone.

Fig. 2. Ablation studies on different sizes of the 1D LUT (a) and different widths of the backbone network (b). $S_o = 0$ indicates models with only 3D LUTs.

1×10^{-4} on an NVIDIA Tesla V100 GPU. K in Equation (4) is set to 3 and 5 for FiveK and PPR10K respectively, whereas S_o and S_t are set according to the purposes of the experiments. We provide them in the following sections.

4.3 Ablation Studies

In this section, we conduct several ablation studies on the retouching task with the FiveK dataset (480P) to verify the key components of the proposed method.

Size of Lookup Tables (1) **3D LUT**: We explore the effects of the size of the 3D LUT by varying S_t in the absence of the 1D LUT ($S_o = 0$) and $m = 8$. The experiments on the FiveK dataset (480P) for the photo retouching task show that, with only the 3D LUT, decreasing the LUT size S_t (from 33, 17 to 9) can significantly reduce the number of parameters (from 385K, 106K to 69K) without a substantial performance drop (from 25.27dB, 25.24dB to 25.21dB). Such a phenomenon suggests the capacity redundancy of the 3D LUT. Therefore, considering the balance between performance and model size, we select $S_t = 9$ and $S_t = 17$ as the default settings in this paper. (2) **1D LUT**: We also investigate the effects of different sizes of the 1D LUT when the size of the 3D LUT is fixed. As shown in Figure 2a, increasing the size of the 1D LUT improves the performance continuously and saturates after it exceeds that of the 3D LUT. A possible reason is that the precision of the 3D LUT serves as a bottleneck and will cancel the additional quantization granularity introduced by the 1D LUT.

Capacity of the Backbone Since the backbone network is responsible for providing a coarse analysis of the input image to guide the generation of the 1D and 3D LUTs, its capacity requirement should be correlated with the capacity or the size of the LUTs. For verification, we ablate the width of the backbone network by varying the hyper-parameter m under the same setting of the LUT sizes (S_o and S_t) and report the results in Figure 2b. The ablation results indicate that increasing the width of the backbone does not guarantee enhanced performance

Table 4. Ablation study on different instantiations of the component-independent transform. The results on the FiveK dataset (480P) for the photo retouching task are listed. "HE" is the abbreviation of Histogram Equalization. The ↑ and ↓ symbols indicate the larger or the smaller is better, respectively.

Strategies	$m = 6, S_t = S_o = 9$			$m = 8, S_t = S_o = 17$		
	PSNR ↑	SSIM ↑	#Params ↓	PSNR ↑	SSIM ↑	#Params ↓
HE	21.75	0.848	42.0K	21.76	0.847	106.7K
1D LUT	25.32	0.918	43.7K	25.41	0.917	111.1K
3× 1D LUT	25.42	0.921	47.2K	25.47	0.921	119.8K

Table 5. Effects of the quantization technique. The runtimes are measured on 480P input using an Intel(R) Xeon(R) Platinum 8163 CPU, whereas the memory footprints are represented by the number of the equivalent parameters.

Method	PSNR↑		CPU Runtime ↓		#Params↓		
	ori.	quant.	ori.	quant.	ori.	quant.	rel.
3D-LUT [35]	25.28	25.25	17.35	15.52	593.5K	332.5K	43.98%↓
$m = 6$, $S_t = S_o = 9$	25.42	25.35	25.34	15.65	47.2K	37.9K	19.59%↓
$m = 8$, $S_t = S_o = 17$	25.47	25.43	25.64	16.21	119.8K	76.3K	36.34%↓

but might increase the capacity redundancy and the training difficulty. Besides, a larger LUT size is inclined to require a stronger backbone network. Considering the trade-off between the performance and the memory footprint, we adopt $m = 6$ and $m = 8$ for $S_t = 9$ and $S_t = 17$, respectively.

Instantiation of the Component-independent Transform In this section, we compare several variants to investigate the proper instantiation of the transform, including the histogram equalization (HE) transform, a learnable 1D LUT, and a learnable 3× 1D LUT. Table 4 demonstrates that the 3× 1D LUT performs the best under two different model settings. The results show that the fixed and hand-crafted mechanism of uniforming the input color distribution cannot adapt to different image contents and distinct retouching styles. Hence it does not guarantee advanced performance on the image enhancement task. The learnable 1D LUTs avoid the above issues by end-to-end optimization, and the 3× 1D LUT provides more flexibility and intelligence than the single 1D LUT.

Quantization on Lookup Tables As described in Section 3.5, we quantify the parameters in both 1D and 3D LUT generators from 32-bit floats to 8-bit integers. The results in Table 5 show that such quantization can significantly reduce the storage/memory footprints with only a slight performance drop. It is worth noting that the above results are obtained by directly quantizing the trained model without any finetuning or quantization-aware training strategy, demonstrating the flexibility of the proposed method. Besides, the LUT quan-

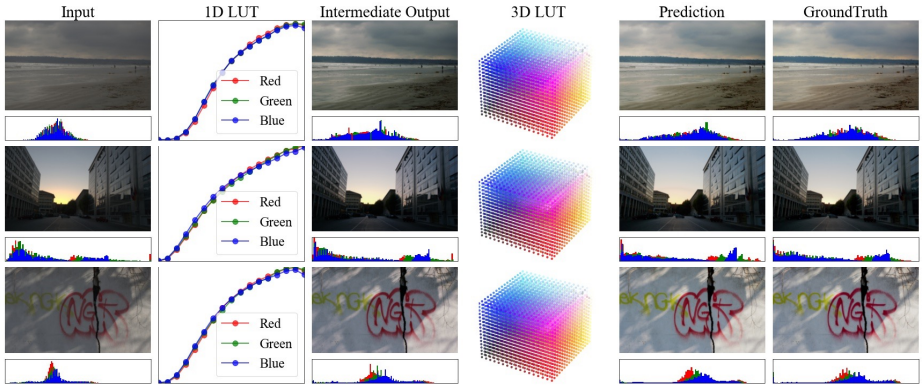


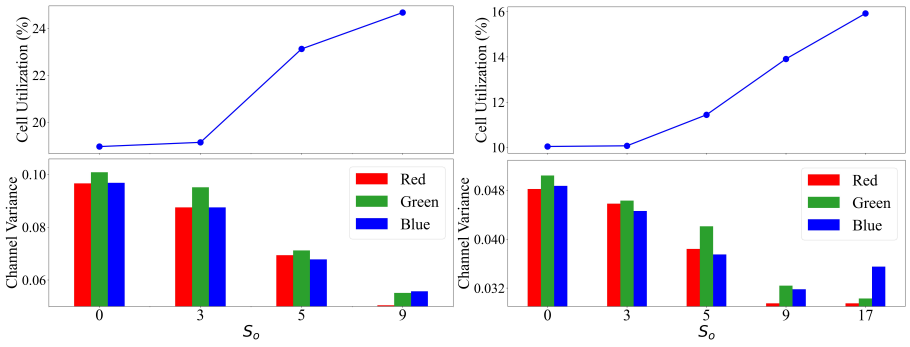
Fig. 3. Illustration of the pair of images, the corresponding histograms, the predicted LUTs, the intermediate transformed output and the final prediction. Images are selected from the FiveK [2] dataset (480P). Best viewed on screen.

tization enables the fixed-point arithmetic, which decreases the CPU inference time of our approach from about 25ms to 16ms. We also apply quantization and fixed-point arithmetic to another 3DLUT-based method [35] and find a similar phenomenon. The proposed framework benefits more from the fixed-point arithmetic in terms of runtime since both the 1D and 3D LUT transforms can be optimized, while [35] only includes the 3D LUT transform.

4.4 Analysis

To help draw an intuitive understanding of the behavior of the image-adaptive LUTs, we illustrate the intermediate outputs as shown in Figure 3. It can be observed that the 1D LUT is inclined to adaptively stretch the input brightness and image contrast, making them in a state more similar to those of the ground truth. Afterward, the 3D LUT is responsible for altering the hue and enhancing the saturation. Besides, to provide quantitative analysis, we also calculate and analyze some statistics of a series of trained models, including the color distribution and the cell utilization of the 3D LUTs, as detailed in follows.

Distribution of Each Color Component To quantitatively investigate the effects of the 1D LUTs on the color distribution, we compare the histogram uniformity of images M transformed by the 1D LUTs of different sizes. The histogram uniformity, to some extent, indicates the level of image contrast and can be approximated by the variance of the image histogram. The bottom row of Figure 4 shows the results averaged on the FiveK dataset. As the size of the 1D LUT (S_o) increases, the per-channel histogram variance of the image decreases, showing the progressive increase of the uniformity of the color distribution. The phenomenon is in line with our expectation that the 1D LUT would adjust the image contrast into a more uniform distribution in an image-adaptive manner.



(a) $m = 6, S_t = 9$. (b) $m = 8, S_t = 17$.

Fig. 4. Effects of the 1D LUT on the utilization of the 3D LUT (the top row) and the distribution of each color channel (the bottom row). Best viewed in color.

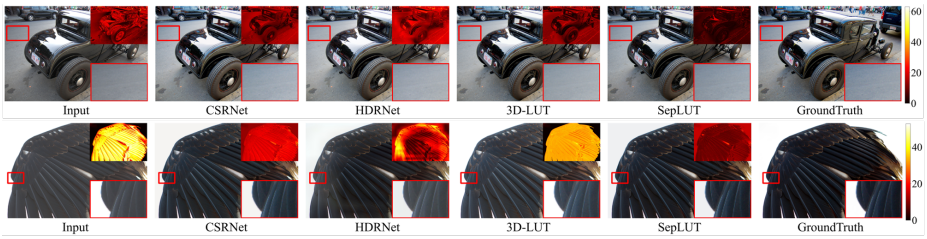


Fig. 5. Qualitative comparisons for **photo retouching** on the **FiveK** dataset (4K) [2]. The corresponding error maps are placed at the top-right of each image, where brighter colors indicate larger errors. Best viewed on screen.

Cell Utilization of 3D Lookup Tables A typical 3D LUT discretizes the entire 3D color space into a grid of cells. Unfortunately, when transforming a single input image, those cells are only partially utilized as the input rarely contains all possible colors. For example, the cell utilization of [35], which can be regarded as a special case of our framework that adopts a single 33-point 3D LUT, is only about 5.53%. To investigate the effects of the 1D LUT on the cell utilization of the 3D LUT, we count for each image the percentage of the cells that have valid pixels falling in. The results averaged on the FiveK dataset are reported in the top row of Figure 4. The 1D LUT is able to activate more cells to increase the model capability of the 3D LUT, and such ability becomes stronger as the 1D LUT becomes preciser (with larger S_o).

4.5 Comparisons with State-of-the-Arts

Quantitative and Qualitative Comparisons For comparisons with *state-of-the-art* real-time methods, we select two typical settings for our approach, namely $m = 6, S_o = S_t = 9$ and $m = 8, S_o = S_t = 17$, denoted as *Ours-S*

Table 6. Quantitative comparisons on the **FiveK** dataset [2] for **photo retouching**. "-" means the result is not available due to insufficient GPU memory. "*" indicates that the results are adopted from the original paper (some are absent ("/")) due to the unavailable source code. The best and second results are in red and blue, respectively.

Method	#Params	480p			Full Resolution (4K)		
		PSNR	SSIM	ΔE_{ab}	PSNR	SSIM	ΔE_{ab}
UPE [31]	927.1K	21.88	0.853	10.80	21.65	0.859	11.09
DPE [5]	3.4M	23.75	0.908	9.34	-	-	-
HDRNet [9]	483.1K	24.66	0.915	8.06	24.52	0.921	8.20
CSRNet [12]	36.4K	25.17	0.921	7.75	24.82	0.924	7.94
3D-LUT [35]	593.5K	25.29	0.920	7.55	25.25	0.930	7.59
SA-3DLUT [32]*	4.5M	25.50	/	/	/	/	/
Ours-S	47.2K	25.42	0.921	7.51	25.40	0.931	7.52
Ours-L	119.8K	25.47	0.921	7.54	25.43	0.932	7.56

Table 7. Quantitative comparisons on the **FiveK** dataset (480p) [2] for the **tone mapping**.

Method	480p		
	PSNR	SSIM	ΔE_{ab}
UPE [31]	21.56	0.837	12.29
DPE [5]	22.93	0.894	11.09
HDRNet [9]	24.52	0.915	8.14
CSRNet [12]	25.19	0.921	7.63
3D-LUT [35]	25.07	0.920	7.55
Ours-S	25.42	0.920	7.43
Ours-L	25.43	0.922	7.43

Table 8. Quantitative comparisons on the **PPR10K** dataset (360p) [22] for **photo retouching**, where a, b, and c denote the groundtruths retouched by three experts

GT	Metric	3D-LUT[35]	Ours-S	Ours-L
a	PSNR	25.64	26.19	26.28
	ΔE_{ab}	6.96	6.71	6.59
b	PSNR	24.70	25.17	25.23
	ΔE_{ab}	7.71	7.50	7.49
c	PSNR	25.18	25.51	25.59
	ΔE_{ab}	7.58	7.48	7.51

and *Ours-L*, respectively. Tables 6 to 8 report the quantitative comparisons in terms of PSNR, SSIM [33], and the L_2-distance in CIE LAB color space (ΔE_{ab}). The results of the selected methods are obtained via using their publicly available codes and default configurations. The proposed method outperforms others considerably with even fewer parameters. Note that in Table 6, SA-3DLUT [32] achieves slightly better performance than our method but at the cost of a significant model size increase (about 37 times) and speed decrease (about 3 times, see Table 9). We also provide some visual comparisons in Figure 5, where our method produces more visually pleasing results than others. For example, while the enhanced images of other methods suffer from incorrect brightness or hazy ill-effects, those of our approach present enhanced contrast and sufficient saturation. Please refer to the supplementary materials for more qualitative results.

Real-time Performance Comparisons To demonstrate the practicality of the proposed method, we evaluate the inference time on 100 images and report

Table 9. Running time (in millisecond) comparisons on different input resolutions. "-" means the result is not available due to insufficient GPU memory. The "*" symbol indicates that the results are adopted from the original paper (some are absent ("/")) due to the unavailable source code.

Resolution	GPU				CPU
	480P	720P	4K	8K	480P
UPE [31]	4.27	6.77	56.88	249.77	126.37
DPE [5]	7.21	-	-	-	553.36
HDRNet [9]	3.49	5.59	56.07	241.90	130.94
CSRNet [12]	3.09	8.80	77.10	308.78	349.61
SA-3DLUT [32]*	2.27	2.34	4.39	/	/
3D-LUT [35]	1.02	1.06	1.14	2.35	15.52
Ours-S	1.08	1.09	1.18	2.23	15.65
Ours-L	1.10	1.12	1.20	2.35	16.21

the average. Each image is tested 100 times on different resolutions including 480P (640×480), 720P (1280×720), 4K (3840×2160) and 8K (7680×4320). The time measure is conducted on a machine with an Intel(R) Xeon(R) Platinum 8163 CPU and an NVIDIA Tesla V100 GPU. As listed in Table 9, our method exceeds the requirement of real-time processing by a large margin on both GPUs and CPUs. The high efficiency of our approach mainly benefits from two factors. First, the downsampled, fixed-resolution input fed to the CNN network makes its computational cost fixed and insensitive to the input resolution. Second, the LUT transform is highly efficient as it is parallelizable on GPUs and can benefit from the fixed-point arithmetic on CPUs.

5 Conclusion

In this paper, we present a novel framework called SepLUT that simultaneously takes advantage of two different types of LUTs, both 1D and 3D, for real-time image enhancement. It separates a single color transform into component-independent and component-correlated sub-transforms. Extensive experiments demonstrate that such a scheme helps sufficiently exert the capabilities of both types of LUTs and presents several promising properties, including enhanced expressiveness, high efficiency, and light memory footprints. The feasibility of the proposed method reflects that the principle of divide-and-conquer can reduce the capability requirements and ease the optimization of each sub-module, which can significantly increase efficiency. Besides, a proper module decomposition can also benefit from the capability complement between the sub-modules and even achieve enhanced overall performance.

Acknowledgements. Yi Xu is supported in part by National Natural Science Foundation of China (62171282, 111 project BP0719010, STCSM 18DZ2270700) and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102).

References

1. Bianco, S., Cusano, C., Piccoli, F., Schettini, R.: Personalized image enhancement using neural spline color transforms. IEEE Transactions on Image Processing (TIP) **29**, 6223–6236 (2020) 2, 4
2. Bychkovsky, V., Paris, S., Chan, E., Durand, F.: Learning photographic global tonal adjustment with a database of input / output image pairs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 97–104 (2011) 8, 11, 12, 13
3. Chai, Y., Giryes, R., Wolf, L.: Supervised and unsupervised learning of parameterized color enhancement. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 992–1000 (2020) 4
4. Chen, C., Chen, Q., Xu, J., Koltun, V.: Learning to see in the dark. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3291–3300 (2018) 4
5. Chen, Y.S., Wang, Y.C., Kao, M.H., Chuang, Y.Y.: Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6306–6314 (2018) 4, 8, 13, 14
6. De Brabandere, B., Jia, X., Tuytelaars, T., Van Gool, L.: Dynamic filter networks. In: Proceedings of the 30th International Conference on Neural Information Processing Systems (NeurIPS). p. 667–675 (2016) 2
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 248–255 (2009) 8
8. Deng, Y., Loy, C.C., Tang, X.: Aesthetic-driven image enhancement by adversarial learning. In: Proceedings of the 26th ACM International Conference on Multimedia (ACMMM). pp. 870–878 (2018) 4
9. Gharbi, M., Chen, J., Barron, J.T., Hasinoff, S.W., Durand, F.: Deep bilateral learning for real-time image enhancement. ACM Transactions on Graphics (TOG) **36**(4), 1–12 (2017) 4, 13, 14
10. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. vol. 9, pp. 249–256. PMLR (2010) 8
11. Guo, C., Li, C., Guo, J., Loy, C.C., Hou, J., Kwong, S., Cong, R.: Zero-reference deep curve estimation for low-light image enhancement. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1780–1789 (2020) 2, 4
12. He, J., Liu, Y., Qiao, Y., Dong, C.: Conditional sequential modulation for efficient global image retouching. European Conference on Computer Vision (ECCV) pp. 679–695 (2020) 4, 13, 14
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016) 8
14. Hu, Y., He, H., Xu, C., Wang, B., Lin, S.: Exposure: A white-box photo post-processing framework. ACM Transactions on Graphics (TOG) **37**(2), 1–17 (2018) 2, 4
15. Karaimer, H.C., Brown, M.S.: A software platform for manipulating the camera imaging pipeline. In: European Conference on Computer Vision (ECCV). pp. 429–444 (2016) 2, 4

16. Kasson, J.M., Nin, S.I., Plouffe, W., Hafner, J.L.: Performing color space conversions with three-dimensional linear interpolation. Journal of Electronic Imaging **4**(3), 226–250 (1995) 4
17. Kim, H.U., Koh, Y.J., Kim, C.S.: Global and local enhancement networks for paired and unpaired image enhancement. In: European Conference on Computer Vision (ECCV). pp. 339–354. Springer (2020) 2, 4
18. Kim, H., Choi, S.M., Kim, C.S., Koh, Y.J.: Representative color transform for image enhancement. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 4459–4468 (2021) 2, 4
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) International Conference on Learning Representations (ICLR) (2015) 8
20. Kwok, W., Haghighi, K., Kang, E.: An efficient data structure for the advancing-front triangular mesh generation technique. Communications in numerical methods in engineering **11**(5), 465–473 (1995) 4
21. Li, C., Guo, C., Ai, Q., Zhou, S., Loy, C.C.: Flexible piecewise curves estimation for photo enhancement. arXiv preprint arXiv:2010.13412 (2020) 2, 4
22. Liang, J., Zeng, H., Cui, M., Xie, X., Zhang, L.: Ppr10k: A large-scale portrait photo retouching dataset with human-region mask and group-level consistency. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 653–661 (2021) 8, 13
23. Liu, E., Li, S., Liu, S.: Color enhancement using global parameters and local features learning. In: Proceedings of the Asian Conference on Computer Vision (ACCV). pp. 202–216 (2020) 4
24. Moran, S., Marza, P., McDonagh, S., Parisot, S., Slabaugh, G.: Deeplpf: Deep local parametric filters for image enhancement. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12823–12832 (2020) 4
25. Moran, S., McDonagh, S., Slabaugh, G.: Curl: Neural curve layers for global image enhancement. In: International Conference on Pattern Recognition (ICPR). pp. 9796–9803 (2021) 2, 4
26. Park, J., Lee, J.Y., Yoo, D., Kweon, I.S.: Distort-and-recover: Color enhancement using deep reinforcement learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5928–5936 (2018) 2, 4
27. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 32, pp. 8024–8035 (2019) 8
28. Selan, J.: Using lookup tables to accelerate color transformations. GPU Gems **2**, 381–392 (2005) 4
29. Sharif, M.H.: High-performance mathematical functions for single-core architectures. Journal of Circuits, Systems, and Computers **23**(04), 1450051 (2014) 4
30. Song, Y., Qian, H., Du, X.: Starenhancer: Learning real-time and style-aware image enhancement. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 4126–4135 (2021) 2, 4
31. Wang, R., Zhang, Q., Fu, C.W., Shen, X., Zheng, W.S., Jia, J.: Underexposed photo enhancement using deep illumination estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6849–6857 (2019) 4, 13, 14

32. Wang, T., Li, Y., Peng, J., Ma, Y., Wang, X., Song, F., Yan, Y.: Real-time image enhancer via learnable spatial-aware 3d lookup tables. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 2471–2480 (2021) 2, 4, 13, 14
33. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing (TIP) **13**(4), 600–612 (2004) 13
34. Wei, C., Wang, W., Yang, W., Liu, J.: Deep retinex decomposition for low-light enhancement. In: British Machine Vision Conference (BMVC). p. 155 (2018) 4
35. Zeng, H., Cai, J., Li, L., Cao, Z., Zhang, L.: Learning image-adaptive 3d lookup tables for high performance photo enhancement in real-time. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2020) 2, 4, 8, 10, 11, 12, 13, 14
36. Zhang, Y., Zhang, J., Guo, X.: Kindling the darkness: A practical low-light image enhancer. In: Proceedings of the 27th ACM International Conference on Multimedia (ACMMM). pp. 1632–1640 (2019) 4